Colorado State University

ENGR 695: Independent Study

Analysis of Integration of Modeling Tools Through Model Based Systems Engineering Practices to the Development of an Inverted Pendulum

Matt Pipan

Department of Systems Engineering

May 6, 2024
COLORADO STATE UNIVERSITY – SYSTEMS ENGINEERING
Engineering Building 202, 6029 Campus Delivery, Fort Collins, CO 80523-6029

As a candidate for the Colorado State University's Master of Engineering Program specializing in Systems Engineering, I am producing this document as the fulfillment of the requirement to graduate as set forth by the Systems Engineering Department and the SYSE 695 Independent Study course.

The project for this course is to be evaluated by:

- Dr. Daniel Herber (Advisor)
- Dr. Tom Bradley (Advisor)

Abstract

In this paper, we apply a Model-Based Systems Engineering (MBSE) approach towards the development of an inverted pendulum. This project ultimately results in the creation of a physical hardware demonstration that is created by applying the systems engineering V-model development cycle. Furthermore, this project integrates multiple 'digital threads' of relevant project information into a single source of truth MBSE model and explores different options for linking system information such as simulation data, 3D CAD model data, and datasheets.

An inverted pendulum is an inherently unstable system consisting of a freely-swinging rod attached to a motorized cart, posing a complex challenge for controls engineers through the design of a feedback loop and system model to automatically balance the pendulum in an upright position. Because of the system's complexity, this project was undertaken as an opportunity to independently work through the product lifecycle while applying systems engineering principles to deepen my understanding of the tangible benefits of systems engineering in an appropriately scoped challenge.

This MBSE approach utilizes the Systems Modeling Language (SysML) and the Magic Systems of Systems Architect (MSOSA) platform to construct the MBSE framework, MATLAB and Simulink to mathematically model the system and control framework, and SolidWorks to physically model the system. Each supporting software package is integrated into the MBSE model to create an authoritative hub-and-spoke model for system analytics that captures everything from system requirements to manufacturing and performance data of a complete system in a single model.

To validate the proposed MBSE framework, a physical prototype of the inverted pendulum created in the MBSE model was manufactured and tested. This prototype served to validate the system design and was capable of maintaining balance in the unstable system for over sixty seconds as set forth in the project requirements.

In summary, this project contributes to advancing the state of the art of digital engineering by exploring improved mechanisms for interoperability of disparate engineering data packages into a single information hub. Additionally, the application of systems engineering principles to a complex system that spans multiple technical domains in a shortened project lifecycle serves to demonstrate the enhanced effectiveness of projects guided by systems engineering.

Table of Contents

Table of Figures

1. Introduction

The inverted pendulum is a classic controls problem in feedback and controls theory. Fundamentally, it consists of a mass balanced above its pivot point mounted to a cart that is typically motorized. This system is *unstable*—a pendulum balanced vertically will simply fall over if it is disturbed. However, the pendulum can be balanced vertically by moving the cart to remain underneath the pendulum's center of mass.

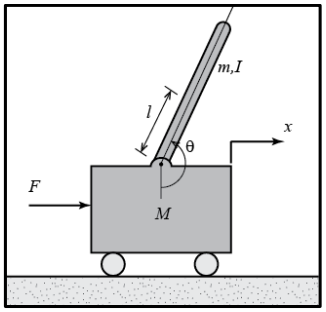


Figure 1: The Inverted Pendulum [1]

By developing a mathematical model of the system, we can predict the pendulum's behavior to a disturbance and thus, design a control loop to maintain the pendulum's upright stability. The inverted pendulum is the subject of this project's model-based systems engineering development.

An inverted pendulum is a complex system that benefits from the implementation of systems engineering principles in its design. It blends both structural and dynamic hardware with mathematical theory design and electrical system design to achieve a performance requirement. These interconnected systems-of-systems all have separate requirements and integration challenges that, if not for the systems engineering method, would pose a significant challenge to harness. For these reasons, the inverted pendulum is a complex system that benefits from systems engineering principles to manage complexity and interoperability of the integrated system.

Traditional systems engineering often uses a documents-based approach towards the development of a new system using the traditional systems engineering Vee model [2].

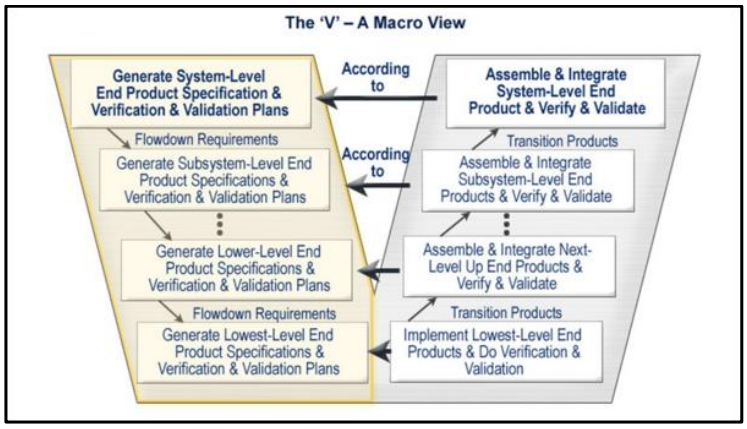


Figure 2: The Systems Engineering Vee Activity Diagram (SEBOK)

In practice, the traditional method systems engineering typically results in a document-heavy project management where systems engineering takes place in glimpses of time where the document was published. This can lead to issues where even one document having a typo can create inefficiency and confusion in project execution. The increasing complexity of modern systems mandates a more modern approach to systems engineering to address the limits of traditional approaches [3].

Model-based systems engineering is the "formalized application of modeling to support system requirements, design, analysis, verification, and validation activities" [4]. This transformation to digital systems engineering mirrors the trend that has taken place across other disciplines—pen and paper drawings have been replace by 3D CAD modeling, a calculator has been replaced by finite element analysis for computational fluid dynamics or structural engineering, and new manufacturing methods can be analyzed from a simulation before a drill bit ever bites metal.

This project will utilize:

- Magic Systems of Systems Architect (MSOSA)[*] for the MBSE model
- Matlab & Simulink for the inverted pendulum mathematical model and control loop
- SolidWorks for the 3D CAD modeling

[*] Magic Systems of Systems Architect (MSOSA) is one of many commercially available MBSE platforms. MSOSA is now owned by Dassault Systèmes but is functionally equivalent to other MBSE packages that utilize the SysML framework or that it has been called previously including No Magic, Catia, or Cameo.

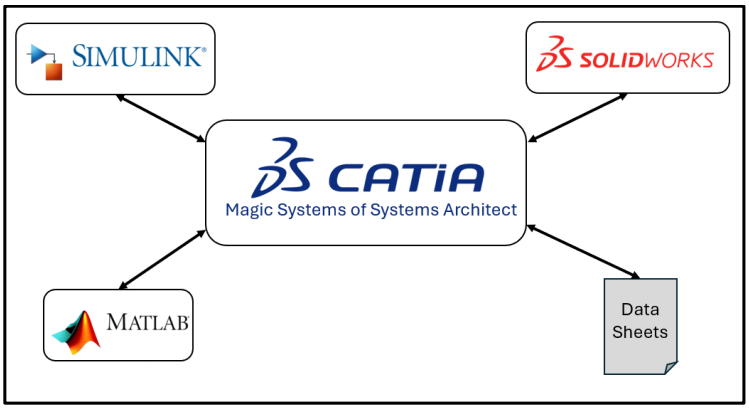


Figure 3: Inverted Pendulum Software Integration Platforms

This project takes a model-based approach to systems engineering to create an MBSE model detailing the development of an inverted pendulum including requirements creation, system modelling, requirements validation, and prototype creation. Instead of laboring through systems development with silo-ed project domains and disciplines, model-based engineering allows the MBSE model to act as a Single-Source-of-Truth (SSOT) for the development of the system. This motivation for analysis tool integration is explored further in Section 4.1.

2. Inverted Pendulum Architecture

2.1. System Requirements

The first phase of system design is requirements gathering. In this phase, we identified four broad categories of requirements. These requirements were compiled in a requirements table in the MBSE software. Then, each category was visualized in a requirements diagram.

In order to begin gathering requirements, it is critical to understand the customer's desires and use cases. In the case of this project, I am the system engineer and customer. I envision this project as a demo piece that not only can be used for educational purposes for highlighting the benefits of good systems engineering, but also as an interactive technical demo of controls theory in practice.

The first category of requirements are safety requirements and are illustrated in Figure 4 below. This category is concerned with maintaining the safety of the operator and the hardware. These safety requirements require that the system shall have safety shutdown mechanisms and reduce risk to the operator where possible.

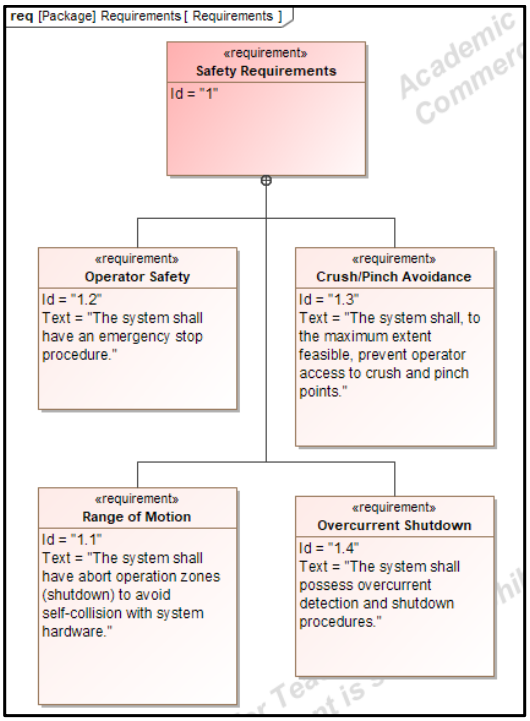


Figure 4: Safety Requirements Diagram

These safety requirements were derived from previous experience with automated control machinery and implements industry best practices.

The next category of requirements that we gathered for the inverted pendulum were operational requirements that describe the mandated performance capabilities of the system. These requirements were primarily derived from stakeholder requirements and desired system performance. Operational requirements are illustrated in Figure 5 below.

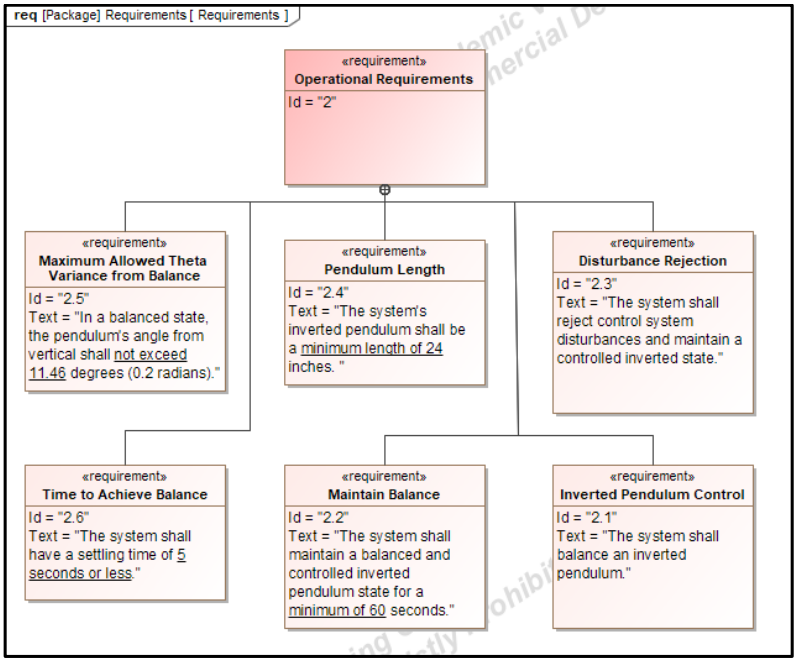


Figure 5 Operational Requirements Diagram

These requirements were initially created based on estimated desired performance. However, after a mathematical model of the simulation was developed, the operational requirements were further refined to be more representative of a realistic system.

Next, a set of non-functional requirements (NFRs) was developed and shown in Figure 6 below. These requirements often describe the "-ilities" that a system needs as well as requirements that do not directly influence the system's performance such as cost. Like many of the other requirements discussed herein, these requirements were created based on stakeholder input based on the desired use case as well as other project constraints. For example, the cost requirement was based off a nominal budget without extensive budget analysis.

Some of the "-ility" NFRs include portability and usability. These quality attributes of the system describe the intended traits of the system but can often be hard to truly quantify. In the usability requirement (requirement ID 3.4), we define the traits by describing a limit on the noise generated by the system as well as by adding a test case relationship that describes exactly how the requirement is to be met, noted by the "Verify" relationship.

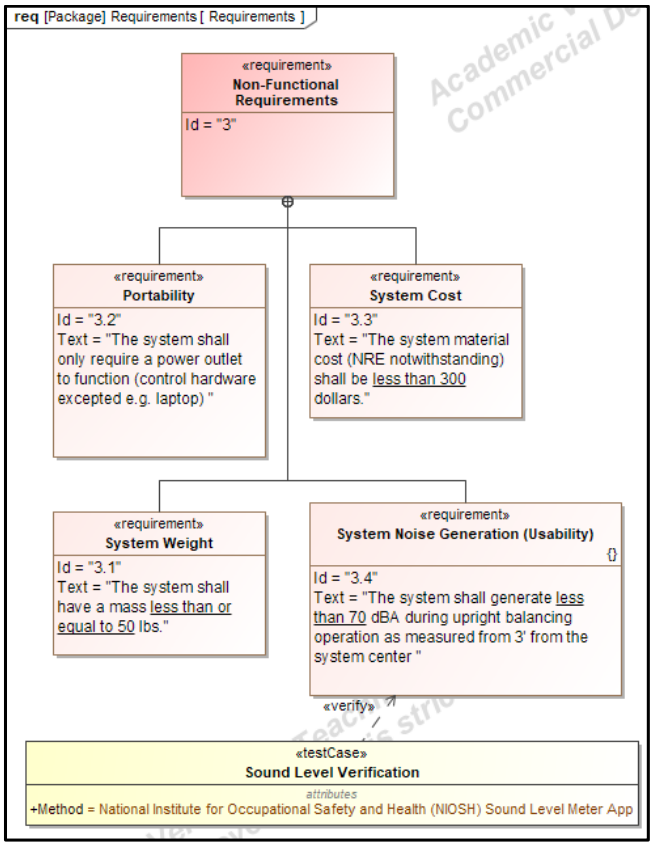


Figure 6: Non-Functional Requirements Diagram

The methodology illustrated in the chart above (of linking test cases to requirements blocks) could be extended to nearly every requirement in the system. In a different system with more concrete operational requirements, we could use this mechanism to link specifications such as MIL-STD-188 or DO-160 and respective applicable sections. For example, if this inverted pendulum design was created as part of a research and development effort for a Department of Defense (DoD) program, it would likely have to go through some MIL-STD-810 environmental stress screening testing. The application of test case blocks to individual requirements allows us to link standards and methods directly to the requirements.

Finally, the last section of requirements derived were concerning system power. These requirements were created as a result of the existing DC power availability that was available to me. This constraint that drove these requirements is added to Figure 7 below.

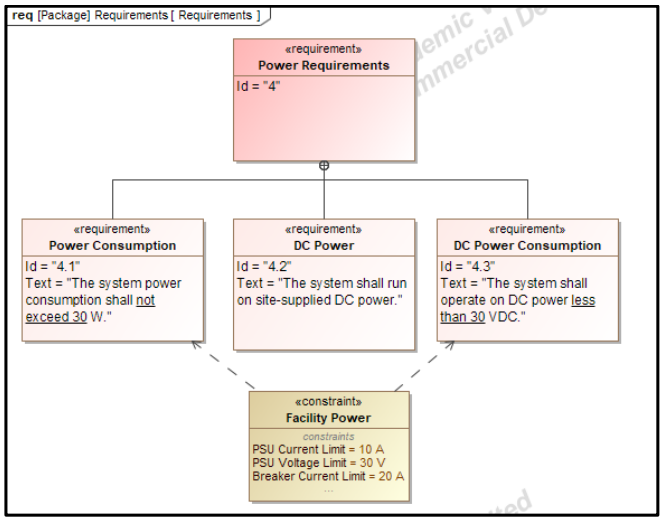


Figure 7: Power Requirements Diagram

Similarly to the discussion regarding test case blocks, constraints could be added where necessary in another system application using the methodology described above. Notably, not every requirement needs a constraint block—only those with outside factors driving a limit on performance.

Finally, we can allocate requirements to specific domains of the system. One example of this is provided in Figure 8 below. This is a clear-cut example of the benefits of model-based systems engineering—we can trace requirements through the lifecycle from conception through validation as the model matures.

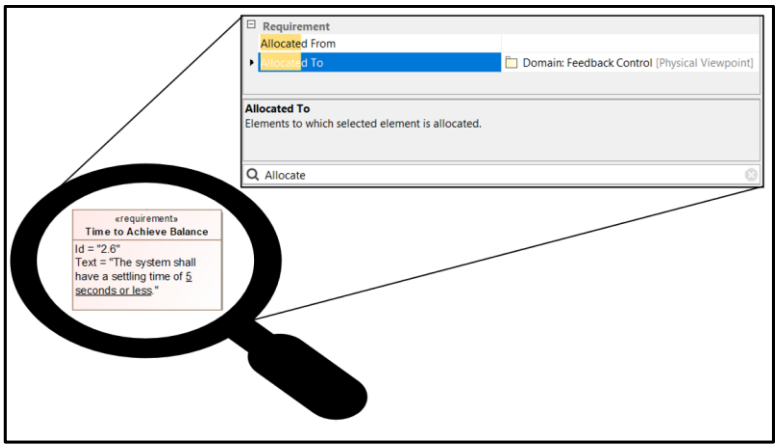


Figure 8: Requirements Allocation to Domains

2.2. Views

After defining the system requirements, we can start to add more definition to the system. We begin by identifying the major domains of the systems. Each domain is a separate group of related components within the system that work towards similar objectives.

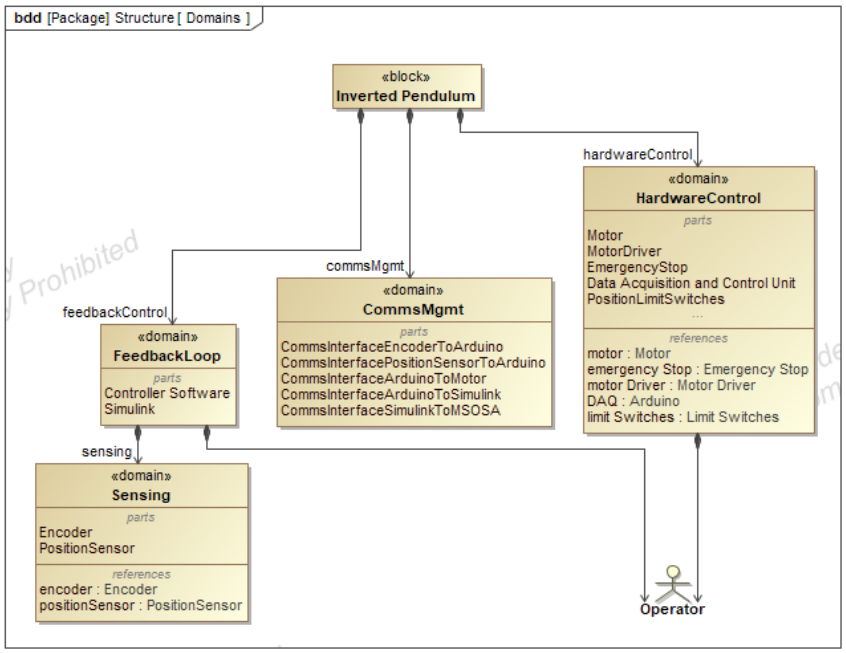


Figure 9: Inverted Pendulum Domains

This project has three primary domains: Communication management, hardware control, and the feedback loop.

The communication management domain is primarily concerned with managing integration of the various components of the inverted pendulum. It consists of parts that address the communications interfaces. For example, this domain explicitly addresses how, for example, the encoder will be able to talk to the Arduino which will then pass on the information to Simulink for processing.

The hardware control domain is all about how the system will implement the feedback. It consists of parts such as the motor for driving the cart and an emergency stop for stopping any commands in case of emergency.

The feedback loop domain consists of software blocks that actually will process and command the physical part of the system to remain balanced. Furthermore, it has an additional subdomain with parts such as the encoder or position sensor. This sensing subdomain has components that act as the input of system characteristics for the feedback loop that provides for full-state estimation.

We then establish a use-case diagram for the inverted pendulum system. Figure 10 below illustrates several use cases that the operator will be able to engage in.

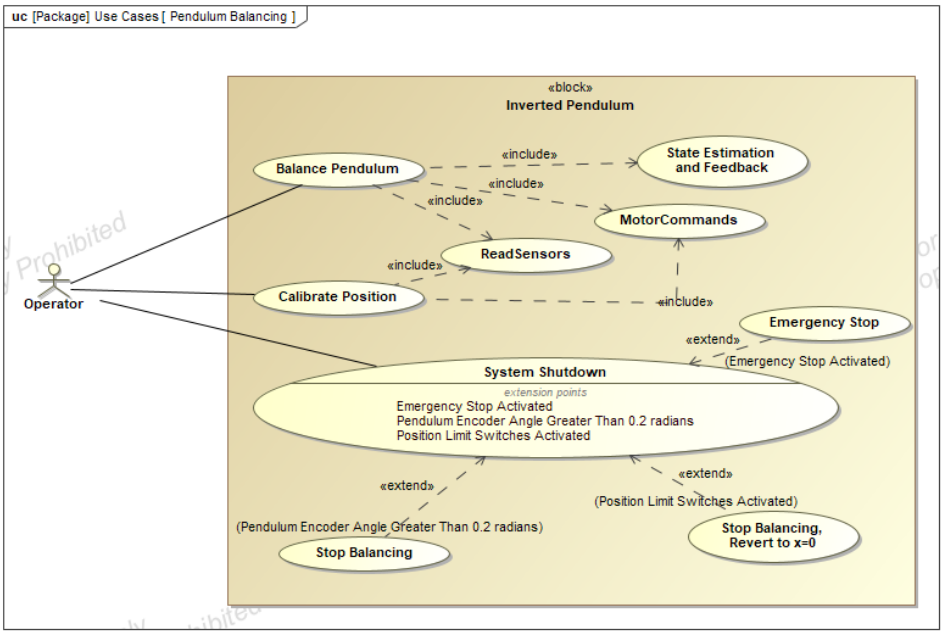


Figure 10: Pendulum Balancing Use Case Diagram

Among possible use cases for the inverted pendulum, we have three primary modes: Calibration, balancing, and system shutdown. Some of the use cases discussed above include extension points. These describe more detailed use cases in the event of the extension point being activated. For example, the 'System Shutdown' use case has an extension point for stopping balancing of the pendulum when the encoder angle of the pendulum reads a value greater than 0.2 radians (11 degrees). This diagram helps provide an easily digestible guide to operational modes of the inverted pendulum. Without MBSE implementation on this project, it is my experience that explicit documentation like this that encourages deeper conceptual exploration could go unfinished.

Next, we create an activity diagram of the system operation. The activity diagram in Figure 11 below. This diagram further refines the use cases in Figure 10 and more concretely defines which element of the system is responsible for what action by requiring that actions are assigned to vertical swim lanes owned by a part of the system. One primary advantage of this activity diagram is that it helps spur data discovery (i.e. what capabilities does each component need to be able to import, utilize, repackage, and export messages between system components?).

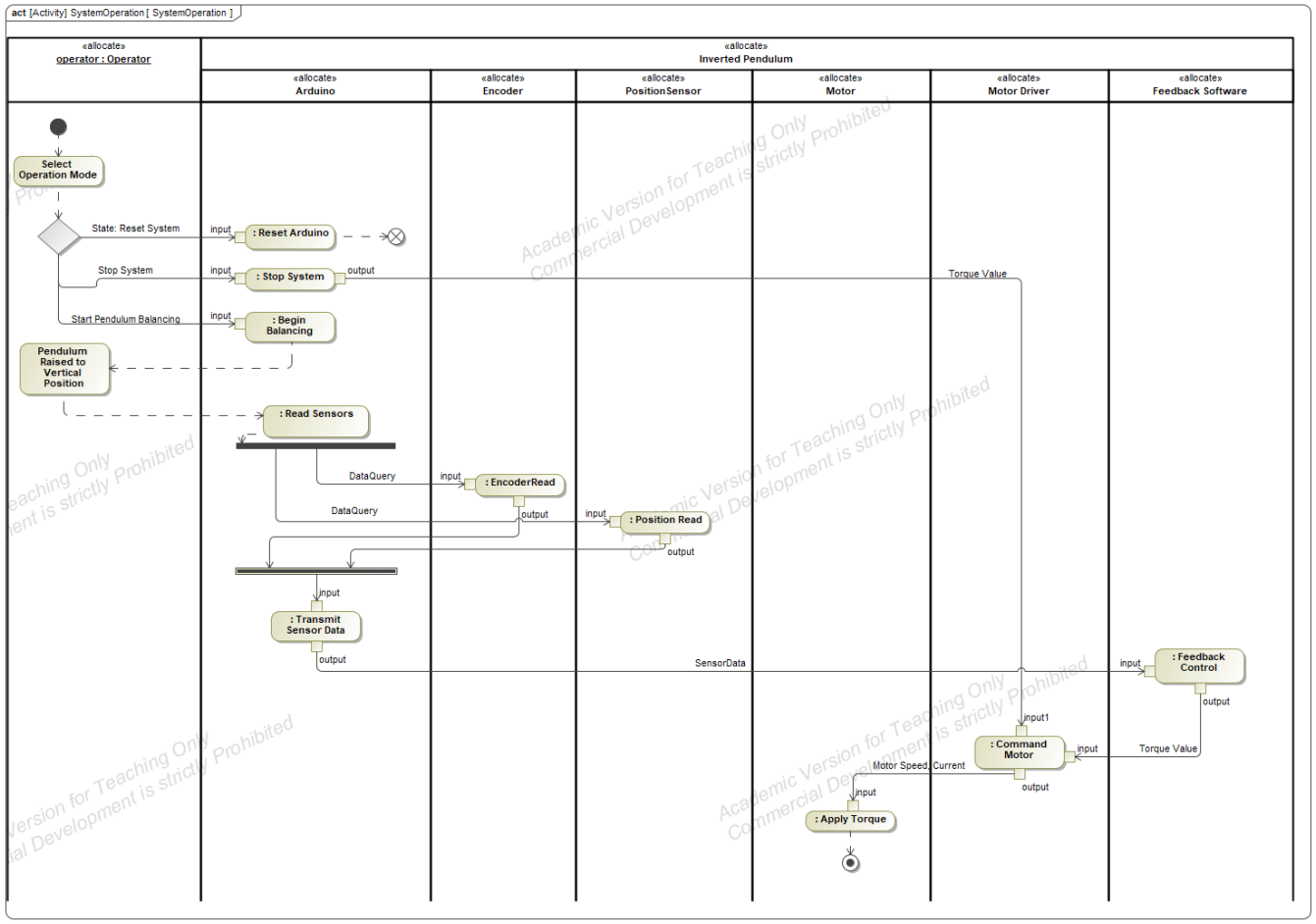


Figure 11: System Operation Activity Diagram

The activity diagram applies a more detailed perspective of the system operation. We further refine this behavior in Figure 12 below with the implementation of a sequence diagram.

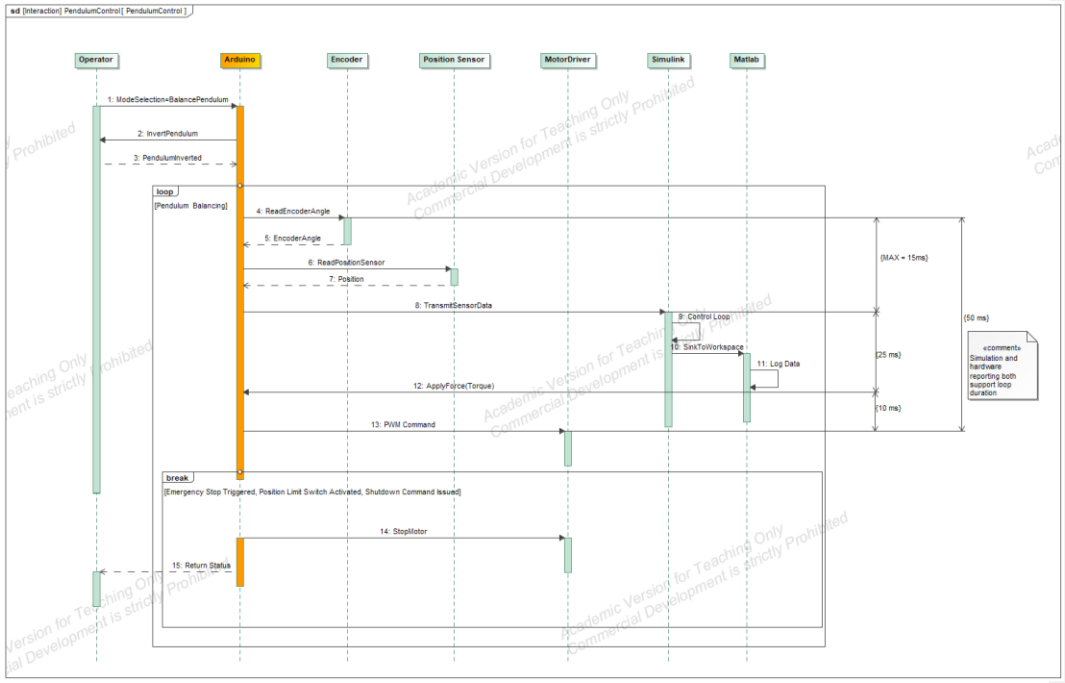


Figure 12: Pendulum Balancing Sequence Diagram with Timing Analysis

In the sequence diagram above, we created two conditions: *loop* and *break*. This diagram helps illustrate and refine the system's behavior. After initialization and startup steps, it enters the loop of balancing where the inverted pendulum will continuously maintain balance by reading sensor data and commanding the motor to apply torque as required to maintain balance. Within this loop, we have a break condition with trigger conditions such as the activation of the emergency stop where the system would cease the looping balancing operation.

Furthermore, we apply some timing analysis based on preliminary analysis of hardware capability and simulation requirements.

Once we have reached this point in the system development, we can begin making point hardware design decisions. We trace our requirements of parts of the system to individual components and select implementations of them by creating instances.

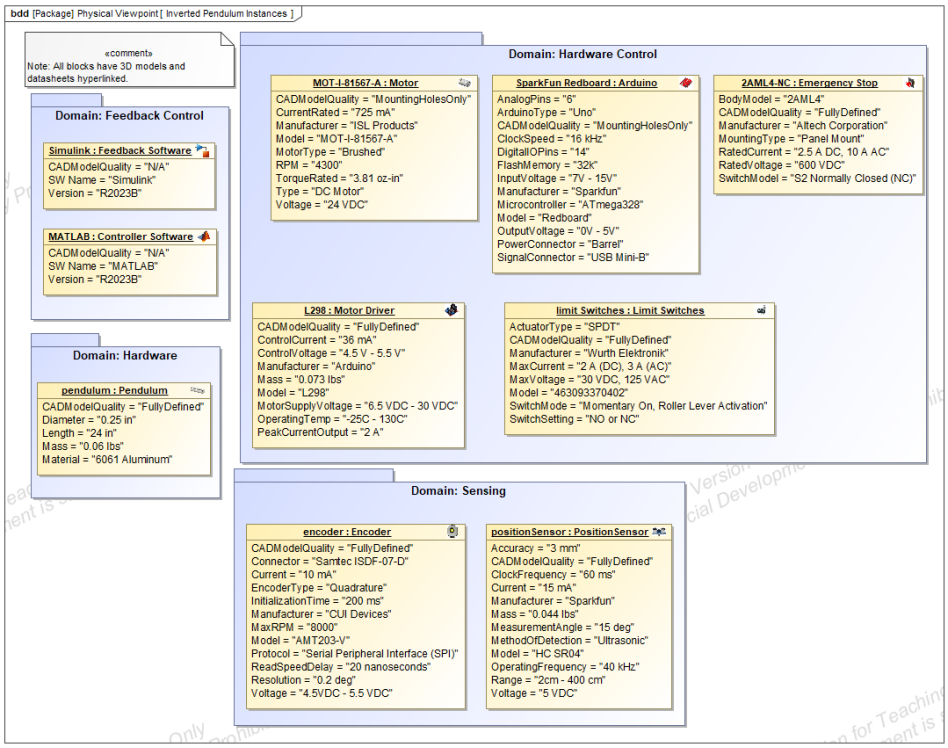


Figure 13: Inverted Pendulum Hardware Instances

We group each hardware instantiation in a package of the domain that they belong to. By adding hardware properties, we can at-a-glance see relevant information about each component. This is an advantage of MBSE because a traditional document-based system will still have this information, but likely without an easily accessible resource for it (e.g. individual hardware datasheets referenced in a project data management (PDM) repository that may or may not have relationships to other relevant system components).

Furthermore, we can link other hardware data in the MBSE model. In this example, we have direct hyperlinks to full hardware data sheets as well as 3D models of the blocks.

Part of developing the physical viewpoint of the system requires integration planning. We can begin to manage interfaces of the system by creating interface blocks for each part of the system that must interface with one another. Note that the use of interface blocks here is used atypically to list out actual interface connections for interfaces in the inverted pendulum system. This diagram helped with integration planning by allowing us to explicitly list each connection point on the interfaces as well as include other relevant connection information.

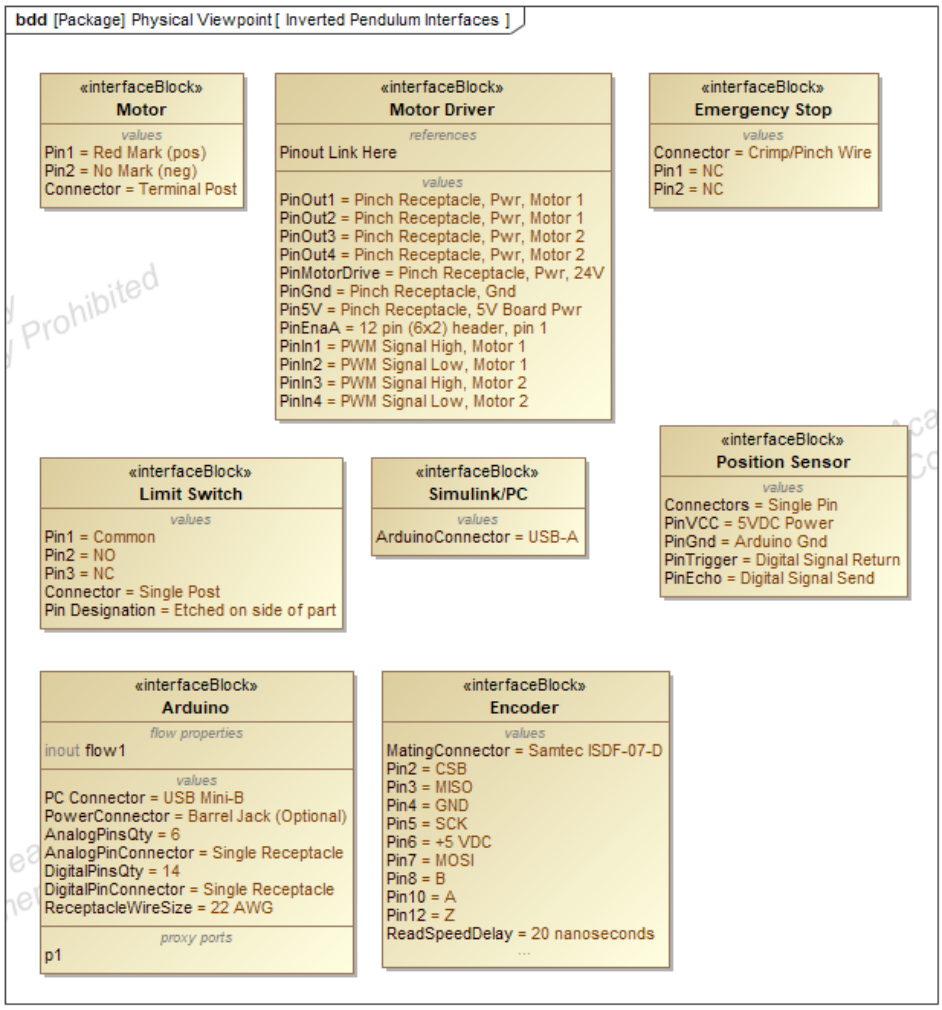


Figure 14: Inverted Pendulum Control Interfaces

By mapping out all connection points, we burn down integration risk by identifying where there may be challenges or adapters required. For example, we use this diagram to identify the total number of input/output pins required for the data acquisition device (DAQ) that helped us source an appropriate hardware instantiation for this block.

Then, we can map out how each interface comes together and detail the interconnections down to the pin that each element of the system will communicate through in Figure 15. This analysis of system integration takes place before any hardware is purchased and allows us, as systems engineers, to create a conceptual idea of the model that both reduces risks and evaluates capability before the system is built.

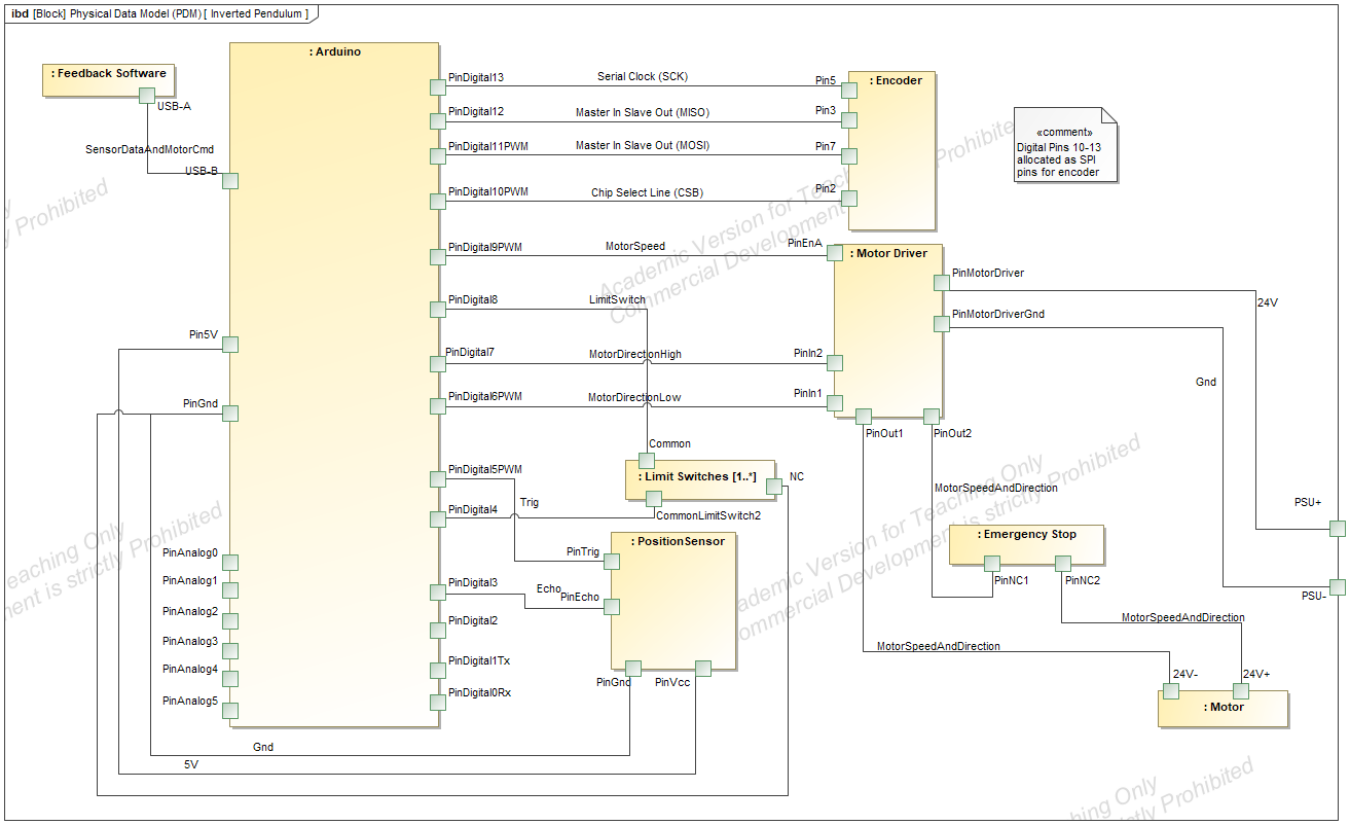


Figure 15: Inverted Pendulum Internal Block Diagram

3. Prototype Creation

3.1. Background

Prototyping is an important early step in system development because it allows system engineers to burn-down risks in system implementation before official system development. A good prototype can communicate a great deal of information about the system. Prototyping offers a glimpse into the system performance and allows us to ascertain whether or not the proposed solution is acceptable to the stakeholders. In other words, a prototype can efficiently answer both questions of "Did we build the system right?" and "Did we build the right system?"

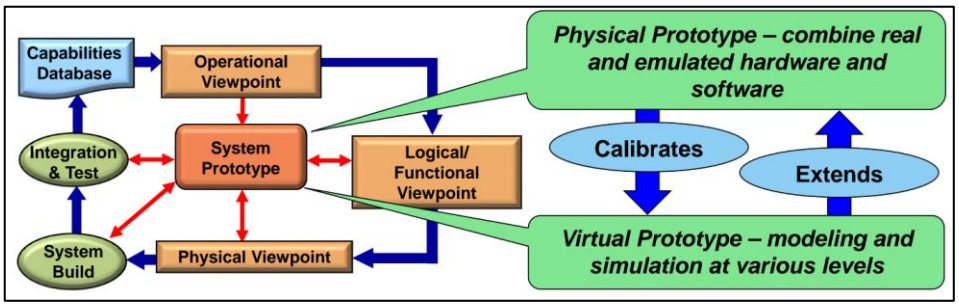


Figure 16: Role of Prototyping in the System Development Lifecycle [6]

A system prototype helps bridge the gap between idyllic models of a system and the idiosyncrasies that can usually only be discovered when real software runs on real hardware. Figure 16 illustrates this relationship.

In this section, we discuss the inverted pendulum virtual and physical prototypes and their impact on discovering emergent behavior or system information that was not identified in the design phase.

3.2. Physical Prototyping

A critical benefit of building a physical prototype is supporting early interaction with the design to make sure it meets stakeholder needs and to identify system integration roadblocks. For this project, the construction of a physical prototype helped circumvent several potential system integration roadblocks such as the original design of the cart having too high of a static coefficient of friction. Without a prototype, this issue would not have been identified early and could be costly to revise later if this system were to begin being made at production rates.

The inverted pendulum system was designed with design for manufacturability (DFM) principles in mind to meet Requirement ID 3.3: System Cost. For the same reasons, commercial off-the-shelf components were heavily utilized where available in conjunction with additive manufacturing in order to create subassemblies that were cheap to manufacture but rigid enough to withstand the dynamic loads induced by the control loop.

This system uses a belt drive system where the motor is located on the cart that moves to balance the pendulum. We chose a single belt drive system because it minimized cost for a slightly higher performance to system performance. Other options include using a rack and

pinion implementation, but this option was down selected against primarily because of cost and a circular belt drive system had more risk involved with integration and the structural properties of 3D printed material.

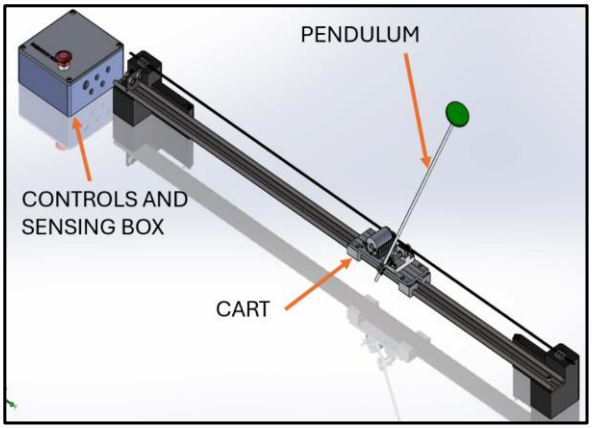


Figure 17: Prototype CAD Model Overview

We have three aspects of interest in the physical design. First, the electronics box. This encases the Arduino controller, the motor driver, and provides a sturdy interface for the emergency stop.

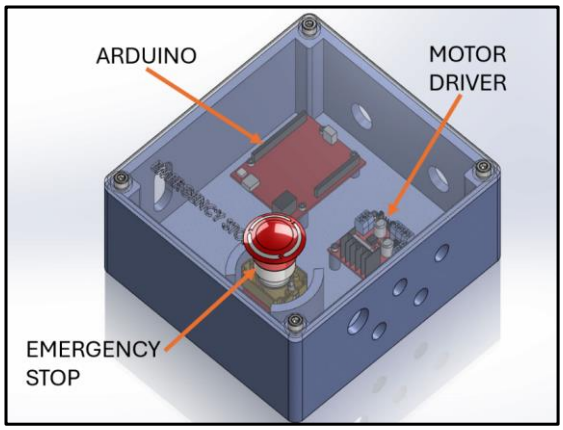


Figure 18: Prototype Electronics Box CAD Overview

The electronics box also has passthroughs for cables for power and signals into and out of the controls box. The box and lid are both 3D printed and utilize heat-set brass inserts that are melted into the 3D printed filament to allow for fasteners to secure the lid to the base.

The rail end caps are rigid attachment points for the linear rail used in this design, a 1.5-meter long IGUS profile WS-10-40 rail. Additionally, it provides a mounting interface for both the motor belt clamp plates and the limit switches. These end caps also utilize additive

manufacturing and utilize brass heat-set inserts for fastener attachment to the 3D printed component. Additionally, one end cap (pictured) has a mounting bracket for the ultrasonic position sensor.

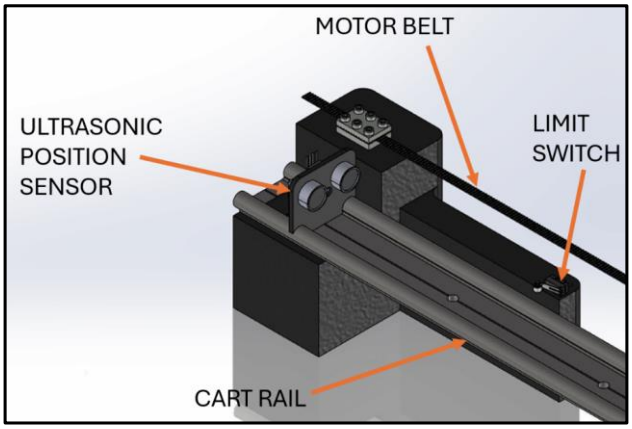


Figure 19: Prototype End Caps CAD Model

Finally, the motorized cart subassembly is where the motor, pendulum, and encoder reside, pictured in Figure 20 below. Among the components that utilize additive manufacturing are a motor mounting block and a 90° shaft-pendulum converter that enables the encoder to read the angle of the pendulum from vertical.

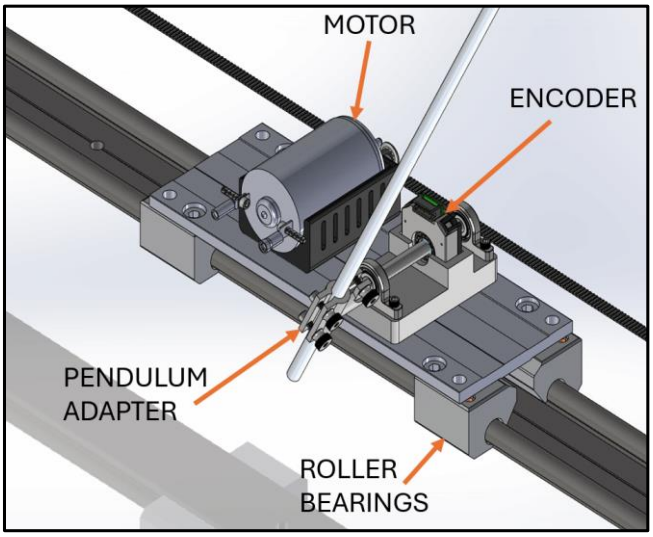


Figure 20: Prototype Motorized Cart CAD Model

Following the design of the system, the physical prototype was assembled, and integration of the various subsystems and control loop could begin.

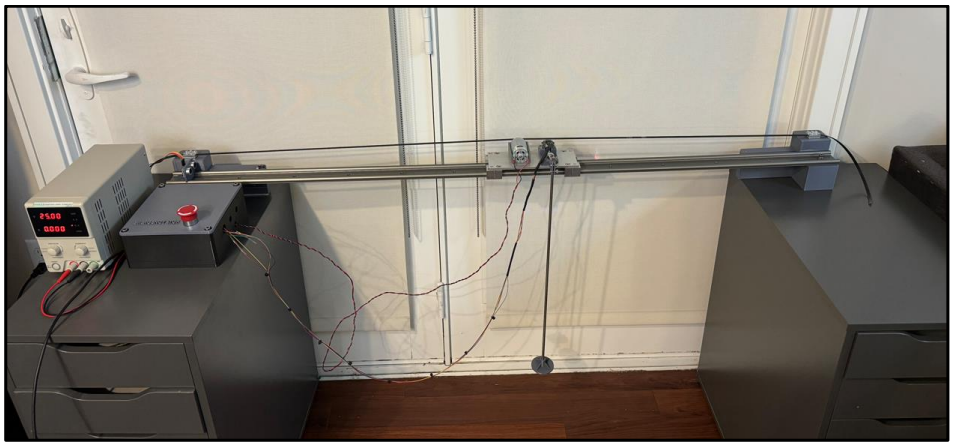


Figure 21: Physical Prototype of the Inverted Pendulum

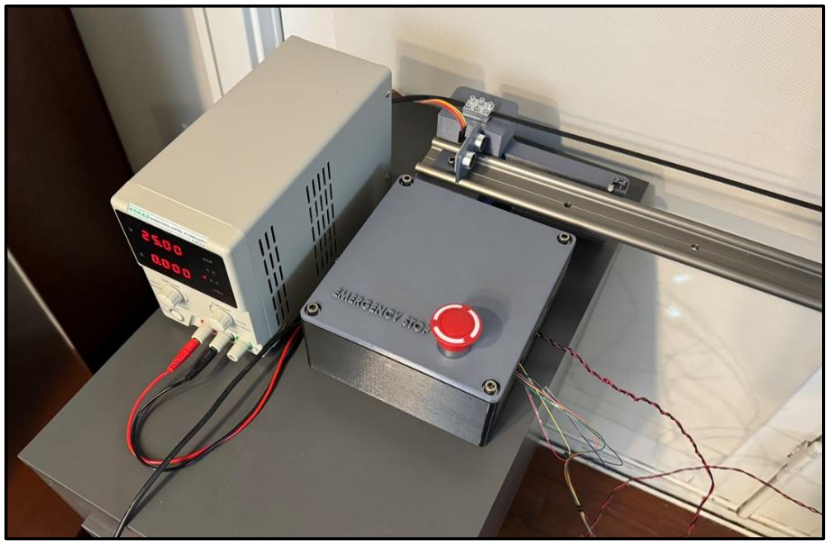


Figure 22: Physical Prototype of the Controls Box

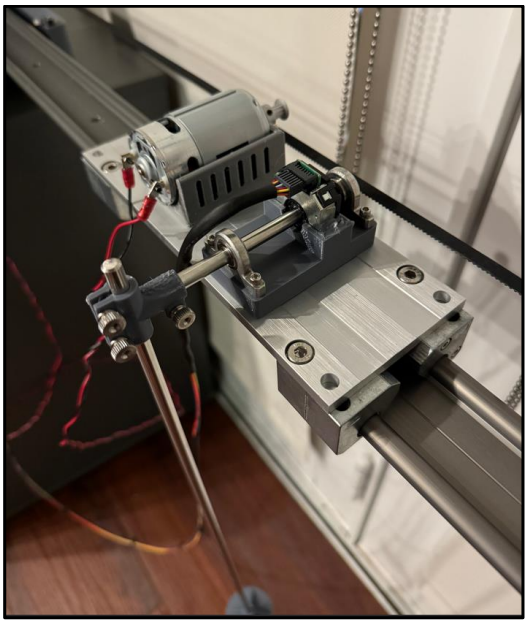


Figure 23: Physical Prototype of the Cart

Another benefit of constructing the physical prototype is that it helps uncover aspects of the design that may not have been accounted for. In addition to helping construct a refined bill of materials (BOM) for the system, it also helps us illuminate potential system costs. Requirement ID 3.3: System Cost establishes a nominal budget of $300 for the system materials. Using an MBSE platform, we can construct a rigorous bill-of-materials estimate including prototype cost.

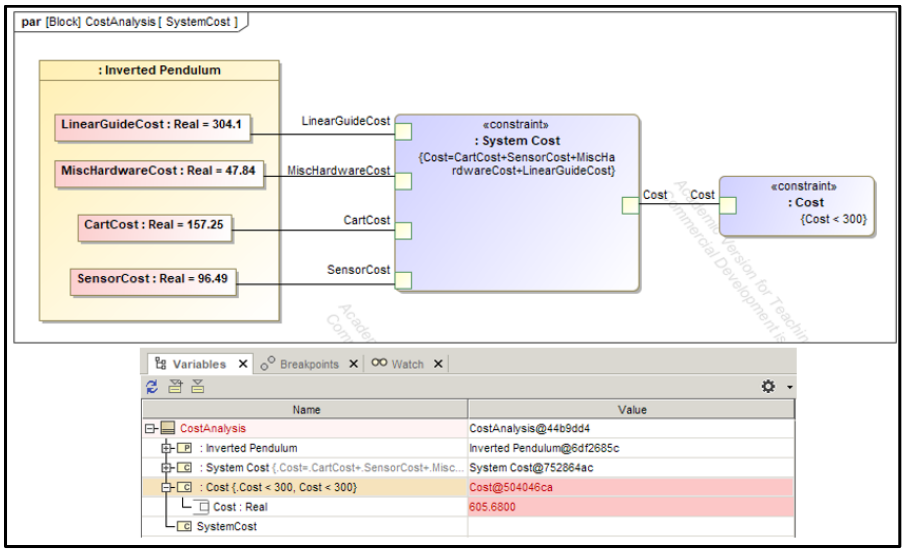


Figure 24: Cost Executable Analysis in MSOSA

The BOM of the prototype could be linked to a product data management system in order to keep the BOM current as the system develops. Furthermore, that BOM could be cross-referenced through the purchasing and acquisition team to have up-to-the-minute cost estimates for the system as the system changes.

For this prototype, the initial nominal budget was overrun. However, valuable lessons were extracted and would be rolled into a revised budget estimate as the system continues to develop. The main cost factors that were unaccounted for were the precision rail system which totaled $304, accounting for approximately 101% of the entire initial budget estimate for these components alone. These unexpected costs were driven largely by the inverted pendulum's cart using IGUS' plastic iGlide J bushings rated for low friction. For the motor picked out for the prototype, this sticking friction ("stiction") was too great to overcome at low motor speeds. This required replacement of the stock bushings with compatible roller bearings that greatly reduced the sliding friction of the cart when driven by the motor. This is one example of a learned constraint that arose from physical prototyping of the inverted pendulum.

3.3. Virtual Prototyping and Executable Architecture

3.3.1. Mathematical Inverted Pendulum Model

The inverted pendulum mathematical model describes the state of the system and is used to predict the system's response based on a given input. To derive the governing equations of motion for the inverted pendulum, we begin by summing the forces on a free body diagram of the system in Figure 25 below.

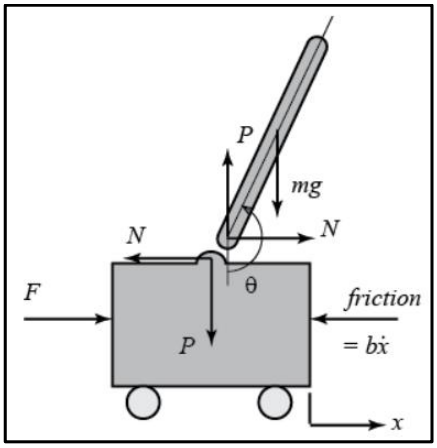


Figure 25: Inverted Pendulum Free Body Diagram [1]

Using Newton's second law, F = ma, we can sum the forces in both the horizontal and vertical directions to arrive at the system equations of motion

$$F = (M + m)\ddot{x} + b\dot{x} = ml\ddot{\theta}$$

$$\ddot{x} = \frac{(I + ml^2)\ddot{\theta} - mgl\theta}{ml}$$

Where:
$F = Force\ Induced\ on\ Cart\ by\ Motor$
$\ddot{x} = Acceleration\ of\ cart$
$\dot{x} = velocity\ of\ cart$
$M = mass\ of\ cart$
$m = mass\ of\ pendulum$
$b = coefficient\ of\ friction$
$l = length\ to\ pendulum\ center\ of\ mass$
$I = moment\ of\ inertia\ of\ pendulum$
$\ddot{\theta} = Pendulum\ angular\ acceleration$
$\theta = angle\ of\ pendulum\ (0° = vertical)$
$g = gravitational\ acceleration$

Equation 1: Pendulum Equations of Motion

We are able to take these equations of motion and put them into state space form. State space is better suited to controlling this system because it is more adept at handling multiple input, multiple output (MIMO) systems. Conversely, a PID controller would be an example of a control methodology that is single input, single output (SISO). A SISO controller would allow us to control one state variable of the system such as the pendulum angle. The MIMO implementation discussed herein allows us to control multiple state variables (pendulum angle and cart position in this case). This means that we are able to both balance the pendulum and ensure that the inverted pendulum system can operate within safe operable bounds on the rail it is constrained to.

The standard form of a state space control feedback equation can be written as follows:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Equation 2: Standard Form of State Space Feedback Control

We can then put the system equations of motion from Equation 1 into the standard form given in Equation 2 to get the state space form of the mathematical model with our A, B, C, and D matrices.

$$\begin{vmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{vmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Ml^2} & \frac{m^2gl^2}{I(M+m)+Ml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Ml^2} & \frac{mgl(M+m)}{I(M+m)+Ml^2} & 0 \end{bmatrix} \begin{vmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{vmatrix} + \begin{vmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Ml^2} \\ 0 \\ \frac{ml}{I(M+m)+Ml^2} \end{vmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{vmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \end{vmatrix} u$$

Equation 3: State Space Inverted Pendulum Model

3.3.2. PID Controller Design

For this project, closed-loop control is required to maintain a stable system state for the inverted pendulum. Open-loop control is often better suited for nominal operational environments or ideal models. With the imperfections inherent in the prototype manufacturing and reduced product lifecycle, we opt for closed-loop control. As discussed in 3.1, we determined previously that this system is both controllable *and* observable.

This project uses proportional-integral-derivative (PID) control. At a high level, PID control works by balancing a process output (in this case, the pendulum angle) around a setpoint (vertical) by setting the gain of the proportional, integral, and derivative terms. The proportional controller gain addresses the current error between the system output and the setpoint. The integral term integrates the past errors to address accumulated errors. The derivative term is forward-looking, and predicts future error based on system rate of change.

Because PID control is a single-input, single-output controller, we implement a layered PID controller approach. The primary PID controller optimizes the angle of the encoder to remain vertical. Then, a secondary PID loop will address error between the setpoint of the linear motion of the cart and the cart's current position. This approach is inferior to the LQR controller discussed in the next section.

Tuning the angular PID gains was performed manually following the principles of the Ziegler-Nichols PID tuning method. Because PID tuning is generally considered part science and part art form, deviations from the Ziegler-Nichols method are expected. Additionally, the secondary PID controller gains were significantly lighter than those implemented in the main PID loop. This was done in order to ensure that the angular control was the primary driver for

system performance and that the controllers would not prioritize keeping the cart in the middle of the track over keeping the pendulum vertical.

Initially, gains were configured inside of the Matlab workspace and were sent through a Simulink file over serial communication to the Arduino. The Arduino would return serial data containing the angle of the pendulum and the position of the cart. However, this approach was altered to be run entirely on the Arduino hardware for a lightweight approach.

Commented [P1]: Clarify Simulink implementation vs HIL vs lightweightt final arduino standalone implementation

Refer to the GitHub repository Arduino file '*Layered_PID_InvertedPendulum.ino*' for final tuning parameters that satisfied the system requirements for maintaining stability.

Communication with the Arduino via Matlab and Simulink was done over serial communication. The configuration for the Simulink blocks used to initialize, receive, and send serial communication is shown below.

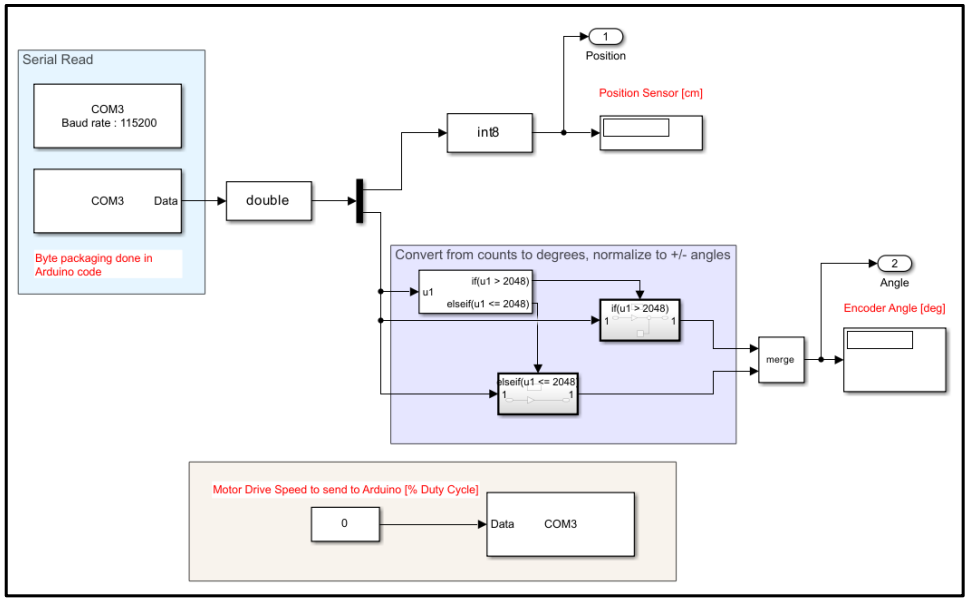


Figure 26: Serial Communication in Simulink

3.3.3. LQR Controller Design

While PID control was the only fully implemented control function, we explored the benefits of designing a linear-quadratic regulator (LQR) controller. Due to project schedule constraints, LQR was designed but not fully implemented on the physical prototype. Moreover, using PID control allowed us to meet the relevant system requirements of being able to balance the inverted pendulum for a minimum of 60 seconds.

An LQR feedback loop with properly weighted matrices are multiple-input, multiple-output (MIMO) controllers. In other words, this was chosen over a single-input, single-output (SISO) controller because the desired behavior of the system is such that we want to be able

to control both the angle of the pendulum as well as the position of the cart. With an implementation of a SISO controller—such as by using PID control—we would primarily only be able to control the pendulum angle and system performance would thereby be reduced if a disturbance resulted in the motorized cart reaching the limit on operational distance. Furthermore, LQR implements a cost function associated with both control effort and system response. This means that by altering our gain matrices, we can change, for example, how quickly our system reaches a steady state based or constrain how much control effort is allowed if our motor performance is a limiting performance factor.

Finally, we also implement a state observer into the control loop. An observer is effectively another mathematical model of the inverted pendulum. We tie the input of the observer to the output of the controller. In this configuration, the observer will estimate the full state of the system prior to measuring the state of the system later. The output of the observer is fed to the controller in order to control the system since it allows us to do full-state estimation of the system when we only have access to two state parameters: position and pendulum angle.

3.3.4. Simulink Model

The Simulink model of the LQR controller design discussed above is pictured in Figure 27 below. This model emulates a step response of the ideal digitally modeled system and the control necessary to keep the pendulum balanced.

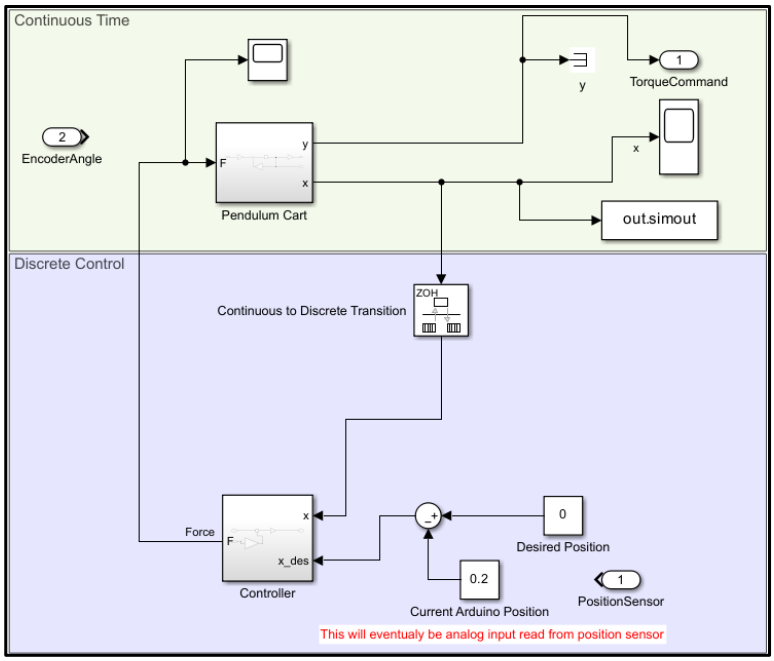


Figure 27: Simulation Model of Inverted Pendulum and LQR Control Loop

Figure 27 above is a simulation without any real hardware inputs. In this simulation, we command the cart to move 0.2 meters while keeping the pendulum balanced.

To implement the control loop on the physical prototype, we begin developing the Simulink interface below. This Simulink file will read serial data from the Arduino and perform the control loop calculations for the motor inputs to balance the pendulum and return the carriage to the middle of the track.

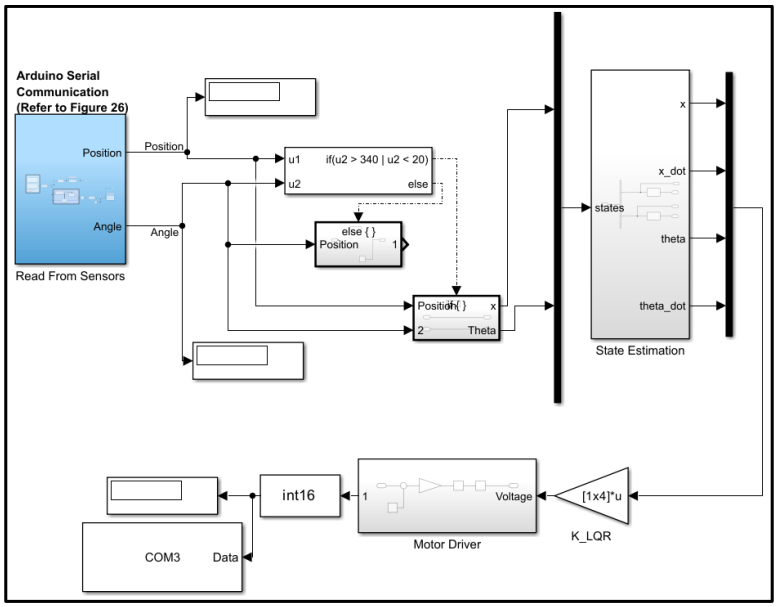


Figure 28: Work-In-Progress LQR Controller Implementation

For this implementation, the cost matrix used for tuning system performance is below.

$$R = 0.4 \qquad Q = 20 * \text{eye}(4) = \begin{matrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{matrix}$$

Equation 4: LQR Cost Matrix Parameters

The simulation results of this control loop show results as expected. We observe that the system, though initially unstable, will converge to a stable balanced point for a step response where the cart's desired position is 0.2 meters while the pendulum remains balanced in Figure 29 below.

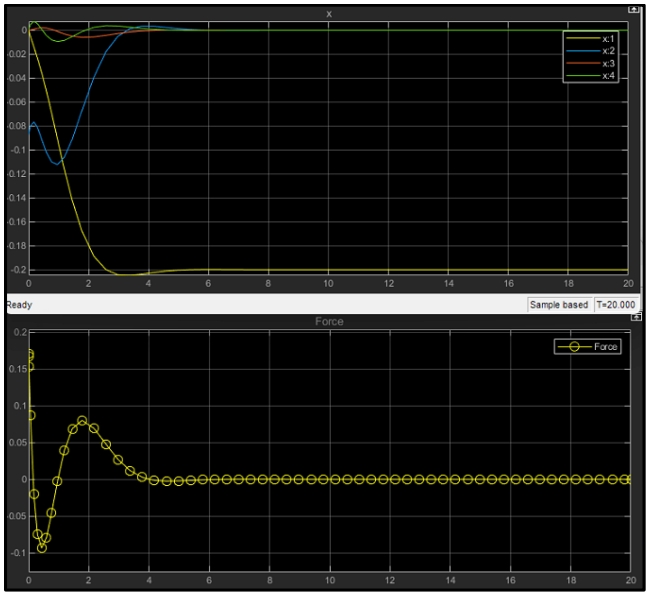


Figure 29: System Convergence After Control Input

4. MBSE Integration with External Tools

4.1. Motivation

Implementing model-based systems engineering is a continuously evolving field. As a result, MBSE—while still an unquestionable improvement upon document-centric systems engineering—does come with its own set of challenges. A study conducted by Carnegie Mellon University's Software Engineering Institute cited that some challenges that faced MBSE implementation was "limited process maturity, with no integrated end-to-end process", point solutions that did not work effectively together, and limited tool maturity [6].

In traditional document-based systems engineering, many of these problems still exist depending on the project management implementation. For example, a traditional system engineering approach often results in technical silos of information that flow through systems engineer, as illustrated in Figure 30.

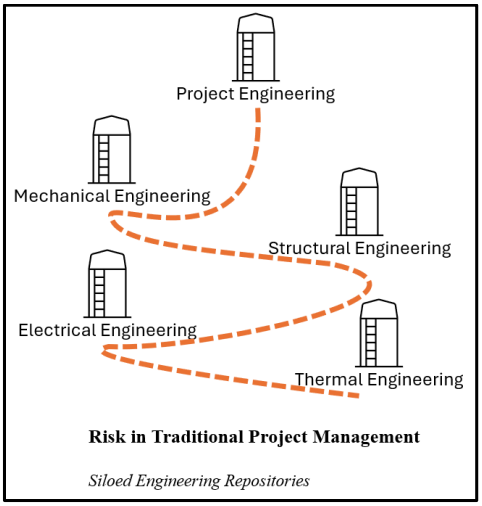


Figure 30: Example of Document Based Project Development Through Siloing

In this configuration, an electrical engineer could identify a circuit board component that needs to be replaced on the bill of materials. The system engineer would be informed and then would need to talk to the thermal engineer to determine how that affects heat transfer capability. Then, we would need to contact a mechanical engineer to implement a new cooling solution or a materials engineer to help specify a new material that could handle the newly derived requirements.

Conversely, the inherent benefit of model-based systems engineering is that we can connect resources from each engineering discipline into a hub-and-spoke model that allows us more efficiently determine the roll-on effects of changing a singular part of a system with regards to other elements of the larger system.

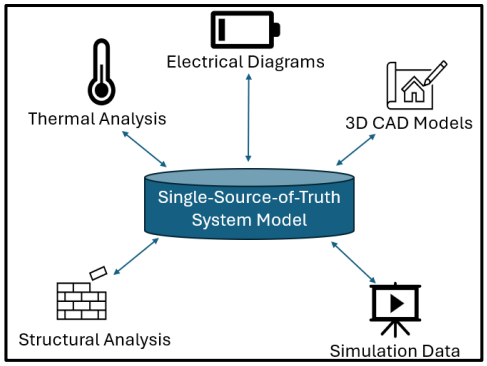


Figure 31: Single Source of Truth Modeling Through MBSE

In the same example discussed above, the new power requirement could be plugged into a new instance and simulations could be run from the MBSE model to efficiently determine system impacts.

Next, interoperability of various modeling packages with MBSE modeling is another challenge that is being developed. A project may use several different analysis or modeling software packages in project development. The ability to natively link them to the MBSE model is not well-documented or widely implemented yet. However, the upside to being able to accomplish this effectively is large because it improves the effectiveness of the art and science of systems engineering because it removes the siloing effect of needing an engineer to hold a system engineer's hand anytime we want to experiment tweaking a design.

4.2. Application

In this project, we put a concerted effort towards exploring options for integration of external software packages into the MBSE model effectively. The integration goals in this project were threefold:

- Link hardware data to MBSE block instances.
- Link 3D CAD models to MBSE model for system visualization.
- Link MATLAB Simulation data to model to requirement verification.

As illustrated in Figure 8, typical practice in MBSE modeling is to include relevant performance properties in blocks. In this project, we performed this implementation where block properties were added. To further this effort, we also linked datasheets for each block (where applicable). For hardware such as the motor, this is a significant improvement over simply putting less detailed information because we can link to datasheets with torque and power curves on a graph that help account for variable performance based on requirements.

Next, we also identified best practices for linking 3D CAD data to the model. A SysML block definition diagram (BDD) consisting solely of blocks can be hard to visualize an implementation of for a real system. To help users, engineers, management, and other stakeholders see how a system may actually be implemented, a traditional systems engineering approach is to usually create a few renderings of the system (after the design phase) and use them to communicate. In this project, we created a robust offering of hardware CAD models that are linked to their instances in the MBSE model. Variations of this methodology were experimented with to evaluate their effectiveness. Originally, we linked to native SolidWorks part files but this implementation was bloated since it required SolidWorks installation, licensing, and an excess of time to allow SolidWorks startup and model rebuilding. Then, we attempted to use 3D PDF files. This rectified the former issue of speed. However, viewing 3D PDF files mandates installation of Adobe Acrobat. Finally, we reiterated this linkage with SolidWorks' eDrawings platform. These eDrawings can be exported in executable formats to not require any installed software which allows for lightweight computing usage. Additionally, these eDrawings can be marked to convey important model information such as "For Reference Only" for low fidelity CAD models. Finally, eDrawings maintain the capabilities of making measurements that were exclusive to the SolidWorks file previously. The usage of eDrawings for this purpose is recommended because it is free, rapid, and efficient. The only drawback is that exporting CAD models into eDrawings

requires another step for a drafting engineer. However, if a company has a product data management (PDM) system where files are managed, we can simply link to the PDM system which means this approach requires a one-time initialization and then it will remain linked as the system develops.

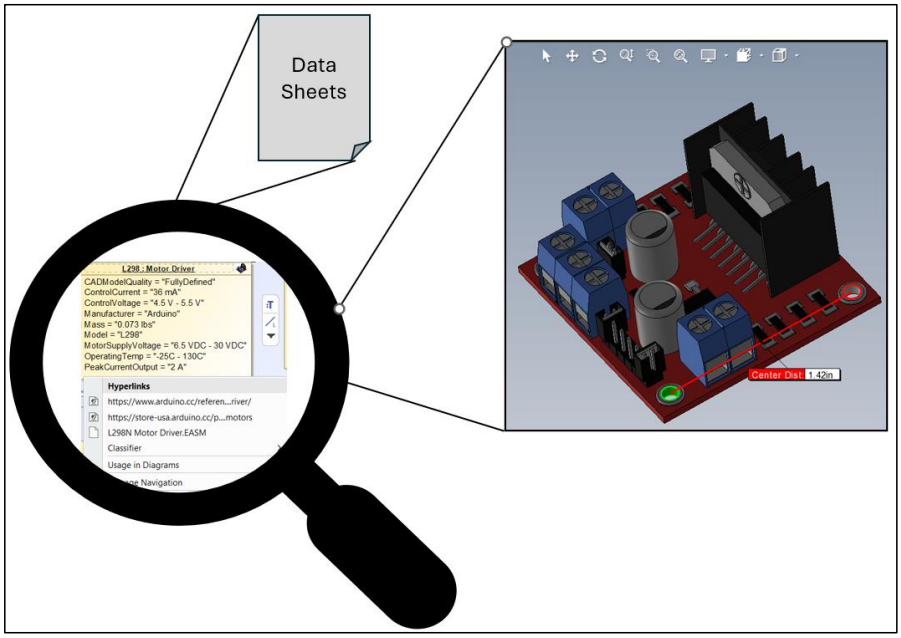


Figure 32: Linkage of Product Data and CAD to MBSE Block

Finally, this project implemented MATLAB and Simulink integration into the MBSE model. The technical demo of this capability is evinced in the validation of requirement 2.6: Time to Achieve Balance, requiring states that the inverted pendulum shall have a settling time of less than 5 seconds.

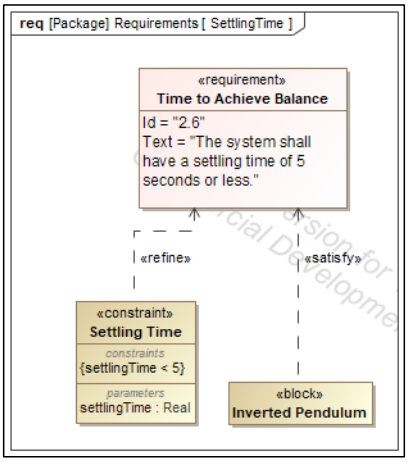
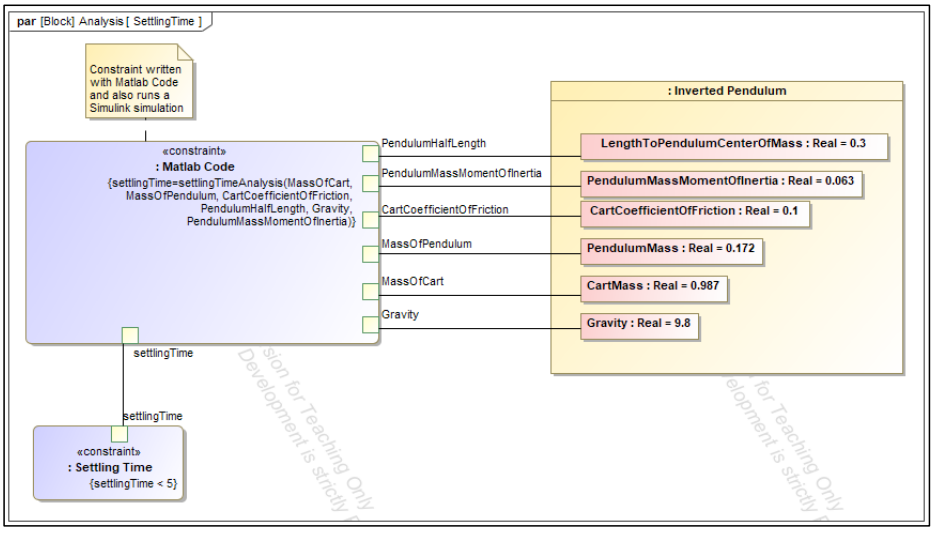


Figure 33: Requirement Validation for Settling Time

The connection from MSOSA to MATLAB is done through the package *Magic Model Analyst*[†]. This package creates an extendable model execution framework that has native MATLAB integration. Through the simulation features of Magic Systems of Systems Architect, we are able to execute MATLAB code in the model-based systems engineering software. So, by writing MATLAB code inside MSOSA, we can call the Simulink simulation of the inverted pendulum. The MATLAB code has provisions for determining the settling time of the system. This information is passed back through to the MBSE software and used as a property for comparison against requirement 2.6.



[†] The plugin Magic Model Analyst was previously branded as Cameo Systems Toolkit (CSK) and is included in this footnote to improve search results.

Figure 34: MBSE Requirement Validation Framework

The executable architecture that we create by interfacing with Matlab and the simulation tools offered in the MBSE software allows us to rapidly iterate on system conditions in parametric diagrams to validate that our system meets requirements. In Figure 35, we observe that the constraint for settling time is met and displayed in green after the Matlab simulation is run.

In a future application, the construction of parametric diagrams and the use of executable architecture could be iterated one to create different instances of the system for evaluation. For example, a new instance of the pendulum cart (e.g. a lightweight cart or a cart with a different rod length) could be created and its parameters tweaked to conduct trade studies for different system configurations.

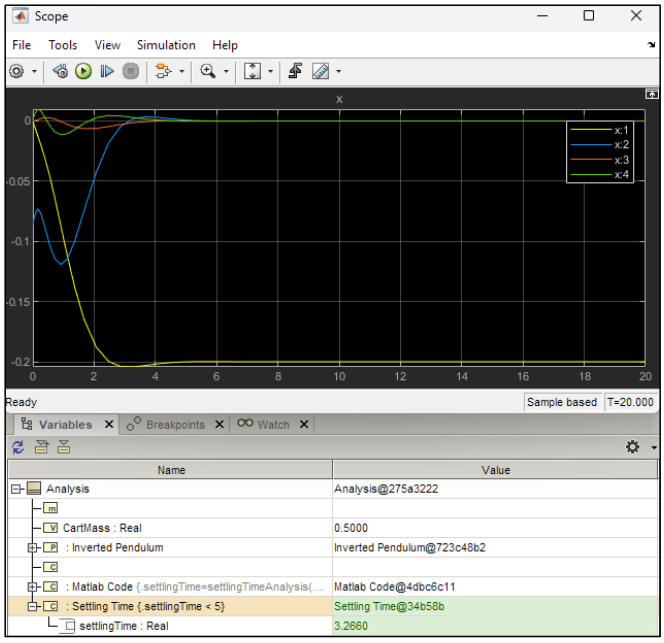


Figure 35: Requirement Validation Through MATLAB Simulation

The integration of outside information and analysis tools within the MBSE model enhances its effectiveness for managing system change and accessing at-a-glance information readily. Each integration goal was achieved and optimized where possible for minimizing wasted time, minimizing licensing needs, and optimizing information transfer.

5. System Verification and Validation

After the completion of the system build, we can revisit our requirements and see how the constructed system meets them.

We performed automatic requirement verification in the MBSE platform by linking relevant information to the individual requirements.

First, linking a test case to the requirement characterizes how the requirement will be addressed. For example, the test case for the system noise generation declares that the National Institute for Occupational Safety and Health (NIOSH) Sound Level Meter App will be used to monitor generated noise from a distance of 3 feet away during balancing operation.

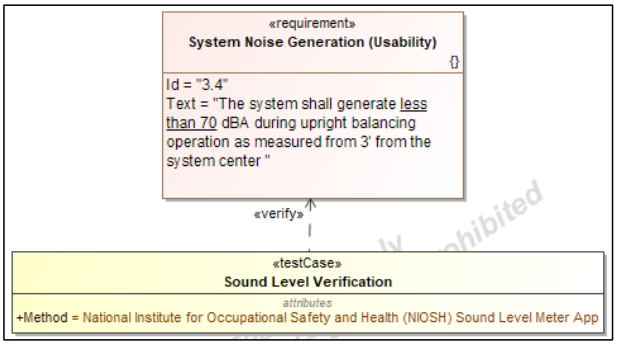


Figure 36: Requirement Test Case Linking

Next, we also link the property of the inverted pendulum block that will contain the relevant metric for satisfying the requirement. The allowed specification, or bounds, are automatically extracted from the requirement text. The margin column illustrates by how much the metric is passing or failing.

The inverted pendulum system met 16 of 17 identified requirements as illustrated in Figure 37 below.

The only metric not satisfied by the inverted pendulum system was the estimated system cost. The cost ended up far surpassing the initial cost estimates primarily as a result of the reasons detailed in Section 3.2 concerning the linear guides and bearings specified for the linearly moving cart.

The ultimate performance test for the system was done to ensure that the pendulum could balance in a stabilized inverted position for a minimum of 60 seconds. The inverted pendulum passed this test and balanced for over 86 seconds before the emergency stop was initiated and the test was over. A recorded video of this test can be found at the link in the README of the GitHub repository or at the link in Appendix B of this report.

#	Name	Text	Verified By	Property	Bounds	Value	Margin
1	1 Safety Requirements	The system shall safely operate.					
2	1.1 Range of Motion	The system shall have abort operation zones (shutdown) to avoid self-collision with system hardware. The value shall be equal to 1 if this requirement is met in the shutdown test case.	Shutdown Test	AbortOperationZones : Rea	=1	1	0
3	1.2 Operator Safety	The system shall have an emergency stop procedure. The value shall be equal to 1 if this requirement is met.	Shutdown Test	EmergencyStopFeature : R	=1	1	0
4	1.3 Crush/Pinch Avoidance	The system shall, to the maximum extent feasible, prevent operator access to crush and pinch points. The value shall be equal to 1 if this requirement is met.	Inspection	CrushPointsMitigated : Rea	=1	1	0
5	1.4 Overcurrent Shutdown	The system shall possess overcurrent detection and shutdown procedures. The value shall be equal to 1 if this feature is implemented.	Shutdown Test	OvercurrentMitigation : Rea	=1	1	0
6	2 Operational Requirements	The system shall meet the following operational requirements.					
7	2.1 Inverted Pendulum Control	The system shall balance an inverted pendulum. The value shall be equal to 1 if this requirement is met.	60 Second Balancing Test	invertedPendulumBalance :	=1	1	0
8	2.2 Maintain Balance	The system shall maintain a balanced and controlled inverted pendulum state for a minimum of 60 seconds.	60 Second Balancing Test	MaintainBalanceTime : Rea	>=60	86	26
9	2.3 Disturbance Rejection	The system shall reject control system disturbances and maintain a controlled inverted state. The value shall be equal to 1 if true.	Operational Test	disturbancerejection : Real	=1	1	0
10	2.4 Pendulum Length	The system's inverted pendulum shall be a minimum length of 24 inches.	Inspection	PendulumLength : Real	>=24	24	0
11	2.5 Maximum Allowed Theta Variance from Balance	In a balanced state, the pendulum's angle from vertical shall not exceed 11.46 degrees (0.2 radians).	60 Second Balancing Test	MaxVerticalAngleFromSetp	<=11.46	8.3	3.16
12	2.6 Time to Achieve Balance	The system shall have a settling time of 5 seconds or less.	Analysis	SettlingTime : Real	<=5	3.266	1.734
13	3 Non-Functional Requirements	The system shall meet the following non-functional requirements.					
14	3.1 System Weight	The system shall have a mess less than or equal to 50 lbs.	Inspection	Mass : Real	<=50	11	39
15	3.2 Portability	The system shall only require a power outlet to function (control hardware excepted e.g. laptop) The value shall be equal to 1 if true.	Inspection	Portability : Real	=1	1	0
16	3.3 System Cost	The system material cost (NRE notwithstanding) shall be less than 300 dollars.	Analysis	SystemCost : Real	<300	605.68	-305.68
17	3.4 System Noise Generation (Usability)	The system shall generate less than 70 dBA during upright balancing operation as measured from 3' from the system center	Sound Level Verification	SoundLevel : Real	<70	64.2	5.8
18	4 Power Requirements	The system shall meet the following power requirements.					
19	4.1 Power Consumption	The system power consumption shall not exceed 30 W.	Analysis	PowerConsumption : Real	<=30	3.5	26.5
20	4.2 DC Power	The system shall run on site-supplied DC power. (shall be equal to 1 if the system uses DC power).	Inspection	DCPowerUsage : Real	=1	1	0
21	4.3 DC Power Consumption	The system shall operate on DC power less than 30 VDC.	Inspection	DC Voltage : Real	<30	25	5

Figure 37: Inverted Pendulum Requirement Validation

6. Future Work and Lessons Learned

Desired future work for this system includes:

- Full implementation of the LQR controller – This controller is expected to give greater control over inverted pendulum behavior.
- Swing-up Control – The prototype currently requires the operator to manually rotate the pendulum to be approximately vertical to start balancing. An algorithm could be developed to swing the pendulum back and forth to get enough energy to swing up vertically on its own.
- Disturbance Rejection Analysis – The controller could be tuned in order to maximize disturbance rejection. In other words, we could tune the controller such that bumping the pendulum with varying amounts of force would allow for it to remain robustly balanced.

Through the course of the development of the physical prototype, there were several lessons learned and recommendations for the final system design that include:

- Consider switching to a screw linear actuator for the linear motion instead of the roller track and belt to reduce chatter and noise/force experienced between the motor and belt from sudden and extreme motor inputs.
- While not slow on a human scale, the ultrasonic sensor used to measure cart position is slow compared to other sensor measurements in this controller. Consider using a rotary encoder in conjunction with the motor on the screw linear actuator from the recommendation above to measure cart position at a much faster sampling rate instead of the ultrasonic position sensor.
- Consider using a locking connector for the rotary encoder. The implemented Samtec connector will, rarely, become disconnected after sudden movements.
- Consider implementing roller cable carriers (such as the IGUS e-chain line) to manage cables running to and from the carriage.
- The original roller bushings on the IGUS carriage were bushings (PN: WW-10-40-20). These had to be replaced with compatible roller bearings (IGUS PN: WJRM-21-10) because the original ones experienced too much sticking friction for the motor to be able to move the cart.

7. Conclusion

In this project, we applied systems engineering methodology to the design and development of an inverted pendulum that can be controlled to remain vertically balanced. We followed the systems engineering lifecycle down the left half of the Systems Engineering Vee (pictured in Figure 2) through the design of the system and closed the loop going up the right half of the Vee to validate and test the system.

The model-based systems engineering approach was tailored to the inverted pendulum system. One goal of this project was to get more familiar with model-based systems engineering packages and apply them to a real-life system over an abbreviated project lifecycle. To that end, this project was a success. Model-based systems engineering has benefits to an organization that

may seem intangible. However, the benefits of MBSE (and systems engineering as a whole) were realized firsthand. In this project, MBSE was used to great effect by encouraging us to define system boundaries, interfaces, requirements, and necessary functions before any system design was started. This ensured that a complete set of requirements from which to sculpt the physical prototype existed.

In this project, we put an emphasis on exploring interoperability of MBSE with analytics and design tools. We effectively got several design and analysis tools integrated into the MBSE platform. Through this integration, we created an effective hub-and-spoke model that linked the system from requirements to prototype. Specific functionality implemented included:

- Using SolidWorks, we linked 3D models to the MBSE blocks that realized the system components. This allowed us to find a block in the MBSE model and, in one-click, open a physical preview of the part. This is an incredibly useful integration for communicating to stakeholders.
- Using hyperlinks, we linked product datasheets to the MBSE blocks realizing the system components. In one click, we could open the manufacturer datasheet to quickly ascertain information that was not readily available in the instance specification block.
- Using Matlab and Simulink, we created executable architecture to verify system properties. With Matlab integration, we could write Matlab code inside of the MBSE platform (MSOSA) and run a Simulink simulation and receive simulation data to satisfy a requirement.

By bridging the gap between system design and model-based system engineering, we advance the state-of-the-art in model-based systems engineering application for control systems.

The creation of a physical prototype derived from the requirements and structure set forth in the MBSE model was significantly aided by the MBSE architecture. The systems engineering approach ensured that no feature was left forgotten, or requirement was left unaddressed. This project was a successful implementation of systems engineering principles that allowed for individual exploration and practice in the field of systems engineering in the developing field of digital engineering.

Finally, a physical prototype of the designed system was created that met nearly all of the identified requirements. This inverted pendulum control system was capable of balancing in an upright position that satisfies our internal requirements.

8. Appendix A: Miscellaneous Material

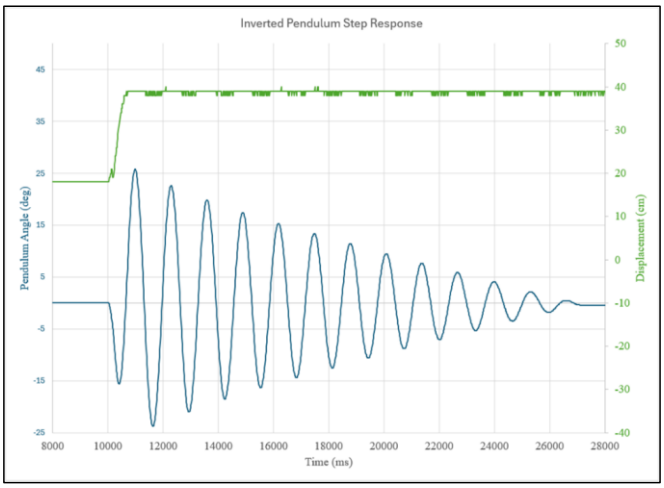


Figure 38: Pendulum Step Response (Physical Model)

9. Appendix B: 60 Second Balancing Test Link

https://youtu.be/Adzkz2kPocA

10. References

1. **University of Michigan and Carnegie Mellon University**. (1997). *Inverted pendulum: System modeling*. Control Tutorials for MATLAB and Simulink - Inverted Pendulum: System Modeling. https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

2. **Fairley, D., & Forsberg, K**. (2023, November 20). *System lifecycle process models: VEE*. Systems Engineering Body of Knowledge (SEBOK). https://sebokwiki.org/wiki/System_Lifecycle_Process_Models:_Vee

3. **National Defense Industrial Association (NDIA)**. (2011). (report). *Final Report of the Model-Based Engineering (MBE) Subcommittee.*

4. **Internation Council of Systems Engineers (INCOSE)**. (2007). Systems Engineering Vision 2020. *INCOSE, San Diego, CA.* Version 2.03, TP-2004-004-02. Published September 2007.

5. **Herber, D.** Session 7: Executable Architecture and Simulation. *SYSE 667: Advanced Model-Based Systems Engineering*.

6. **Nichols, B**. (2021, December 13). Challenges in Making the Transition to Digital Engineering. *Software Engineering Institute (SEI)*. https://insights.sei.cmu.edu/blog/some-challenges-in-making-the-transition-to-digital-engineering/

11. Acknowledgements

This project would have been near-insurmountable without the help of several resources for mastering the control theory and system implementation that are not directly referenced in the sections above. I would like to extend a special thanks to the following people for their insight and knowledge:

- Dr. Daniel Herber, Colorado State University, Department of Systems Engineering
- Dr. Steven Brunton, Author of *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*
- John Borky and Thomas Bradley, Authors of *Effective Model-Based Systems Engineering*
- Brett Beauregard, Author of the Arduino PID Library