

CSCI 5143 Final Project Report: Cart Pole

Mason Hawver and Kuba Kedzior

12/12/24

1. Introduction

For our final project, we wanted to learn the basics of control theory by building and controlling a classical cart-pole system. A diagram of the system can be seen in Figure 1 and 6. This system consists of a cart on a rail that can move freely forwards and backwards. A pendulum is attached to the cart, and it can swing freely. A motor drives the cart forwards and backward through a pulley system where the motor's rotational movement is translated to the linear movement of the cart. The pendulum, however, isn't driven by a motor, but its motion is driven by the laws of physics because it can freely rotate. The relation between the cart's linear movement and the pendulum's rotational movement can be described using the state-space model in equation (1).

The equations may seem complex; however, they clearly illustrate how the cart-pole system can be modeled through the cart's position (x) and velocity ($\dot{x}$), along with the pendulum's angle (θ) and angular velocity ($\dot{\theta}$). The changes in these variables ($\dot{\mathbf{x}}$) are functions of the current state ($\mathbf{x}$) and the input force (u).

- x : The horizontal position of the cart on a track.
- $\dot{x}$: The velocity of the cart, indicating how quickly its position changes.
- θ : The angle of the pendulum, measured from the vertical direction.
- $\dot{\theta}$: The angular velocity of the pendulum, describing the rate at which its angle changes.
- u : The external force applied to the cart, used to control the system.
- M : The mass of the cart.
- m : The mass of the pendulum.
- ℓ : The length of the pendulum (distance to its center of mass).
- g : The acceleration due to gravity.
- L : The rotational Inertia of the pendulum.

Together, these equations provide a complete dynamic model of the cart-pole system, capturing how the system evolves over time based on its current state and the

applied input force. The pendulum's and cart's motions are coupled together; both affect one another. When the pendulum swings down from an upright position, it moves the cart, and when the cart moves left and right, it swings the pendulum.

$$\dot{\mathbf{x}} = f(\mathbf{x}, u), \quad \text{where } \mathbf{x} = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^T \quad (1)$$

$$f(\mathbf{x}, u) = \begin{bmatrix} \dot{x} \\ \frac{mg \sin \theta \cos \theta - 2(u + m\ell \dot{\theta}^2 \sin \theta)}{m \cos^2 \theta - 2(M+m)} \\ \dot{\theta} \\ \frac{(M+m)g \sin \theta - \cos \theta (u + m\ell \dot{\theta}^2 \sin \theta)}{2(M+m)\ell - m\ell \cos^2 \theta} \end{bmatrix}$$

The challenge for this system is to design a controller, an algorithm that determines the necessary inputs u to achieve desired behavior in a dynamic system, such as stability or tracking a reference. Here, the controller must swing the pendulum from a downward position $\theta = 0$ (Figure 1) to an upright position $\theta = \pi$ (Figure 2) and keep it stable, a process called "swinging up and stabilizing." The cart-pole system is inherently nonlinear, so for the balancing controller, one approach is to linearize the system around the upright position $\theta = \pi$. The linearized system will simplify the equations and can be obtained by approximating the dynamics using a first-order Taylor expansion¹:

$$A = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}=[0 \quad 0 \quad \pi \quad 0]^T}$$

$$B = \left. \frac{\partial f}{\partial u} \right|_{\mathbf{x}=[0 \quad 0 \quad \pi \quad 0]^T}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-gm}{2M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-g(M+m)}{2\ell(M+m)-\ell m} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2}{2M+3m} \\ 0 \\ \frac{1}{2\ell(M+m)-\ell m} \end{bmatrix} u$$

¹The specific details of all these equations are not necessary to continue reading the rest of the report.

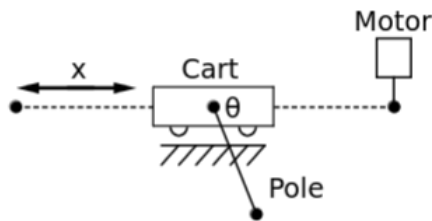


Figure 1: Cart Pole System in Downright Position

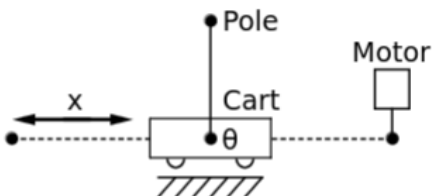


Figure 2: Cart Pole System Balancing Upright

We chose this project to learn the basics of control theory and apply it to balancing dynamic systems like the cart-pole. Hopefully, that knowledge is apparent throughout this report. Additionally, we were pleasantly surprised to learn a lot about design tools and techniques when building the system, which will help in future projects. The system we build will also serve as a foundation for exploring advanced topics in control theory, state estimation, and other topics.

1.1 Practical Insights

Our project doesn't solve any practical problems beyond the ones we set for ourselves. Its value lies in the knowledge gained through designing, building, and controlling the cart-pole system, which helped us understand the fundamentals of control theory and state-space modeling. From this experience, we developed skills in system modeling, controller design, and hands-on implementation, which are applicable to more complex engineering challenges that will solve practical problems.

1.2 Course Relevance

To design a controller for this system, we needed to know the current state of the system, so we could create a closed feedback loop that would hopefully stabilize the system. To know the current state of the system, sensors are needed to measure the system. We decided to use two rotational encoders to measure the movement of both the cart and the pendulum, and a couple of flight sensors to measure the position of the cart. These sensors would allow us to measure the cart's change in position, the pendulum's change in angle, and the cart's absolute position. This was our first error: to assume we didn't

need to measure the absolute angle of the pendulum.

To use all these sensors to measure the system, we would need to implement all of their drivers, software to communicate and receive measurements from the sensor, on a microcontroller. Furthermore, a driver for the motor would also have to be developed on the microcontroller, so we could both measure the system and drive the system. Finally, because we decided to implement the system's controller to swing up and balance the pendulum on a computer, the microcontroller would need to communicate with the controller on the computer to feed it measurements and perform its actions by driving the motor. This project is related to embedded systems because we implemented the communication practicals and sensor integration on a microcontroller, enabling real-time control of the cart-pole system.

1.3 Relevant Topics and Tools

The relevant topics and tools for this project include: Control Theory, State Estimation, State-space modeling, Linear quadratic control, PID control (proportional, integral, and derivative control), I2C and UART communication, CRC (Cyclic Redundancy Check) protocols, PWM (Pulse Width Modulation) for DC motor drivers, quadrature encoder drivers, CAD (Computer-Aided Design) with on-shape and solid-works, 3D printing, laser wood cutting, and soldering.

2. Design and Implementation

To build our cart-pole system, we began by researching existing designs in the literature and previous projects. Our approach was heavily influenced by key sources [1, 5, 6] that explored sensor integration, component selection, and system design for cart-pole systems. These references helped shape our decisions regarding the sensors, actuators, and physical setup, leading us to develop a system that balances precision, simplicity, and cost.

2.1 Hardware Components and Integration

Our cart-pole system is built around a 2-meter-long 2020 V-slot aluminum rail, which forms the foundation of the physical setup. The key components of our system include a DC motor, optical rotary encoders, a microcontroller, and various mechanical parts such as gears and tensioners.

2.1.1 Motor Selection

We chose a 550 rpm, 0.5A, 0.03 hp DC motor for controlling the motion of the cart. DC motors were selected because they offer a simple implementation with fine control over speed, making them ideal for our precision requirements. While brushless motors with torque

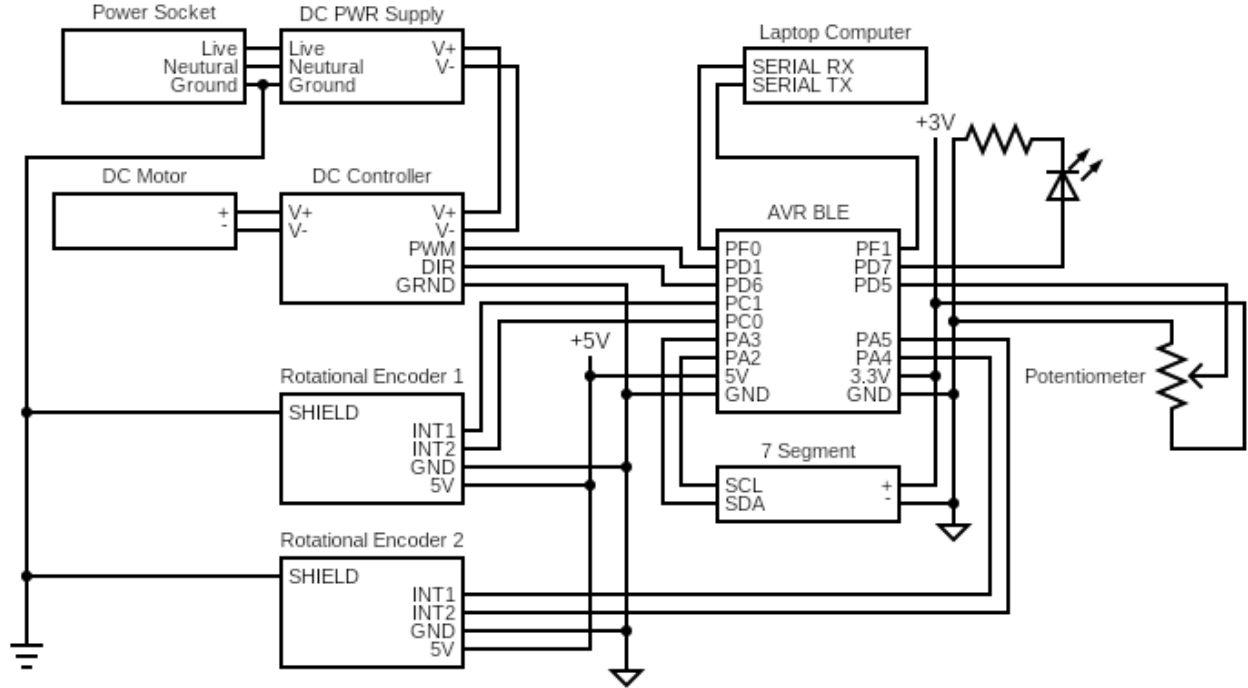


Figure 3: Circuit Diagram of the Cart Pole System

controllers would have provided higher efficiency and precision, they were more expensive and complex to integrate into the system. Given our goal of simplicity and cost-effectiveness, the DC motor was a suitable choice for controlling the cart's motion along the rail.

The DC motor is powered and controlled by two main components: a DC power supply and a DC motor driver. The DC power supply converts AC power from the wall into DC power, which is then used to power the motor. The motor driver takes this DC power and uses a PWM (pulse-width modulation) signal, along with a logical signal, to control the motor's speed and direction. The PWM signal regulates the voltage applied to the motor, allowing the microcontroller to precisely control the motor's speed. The logical signal determines the direction of the current, enabling the microcontroller to reverse the motor's rotation when necessary. Refer to the circuit diagram for more detail (Figure 3).

2.1.2 Encoder Selection and Integration

To measure the position and rotation of the pendulum and cart, we chose optical rotary encoders, which offer high accuracy with 600 clicks per rotation, providing precise detection of angular displacement. We preferred optical encoders over alternatives like Hall effect sensors due to their high resolution and ability to detect incremental changes in position with precision. However, one limitation of optical encoders is that they cannot measure

absolute position, unlike Hall effect sensors, which can track the absolute angle.

Initially, we underestimated this downside in optical encoders, assuming it would not have an impact. However, over longer periods, the optical encoder's estimation of the pendulum's angle would drift from the actual position. This became significant, leading to inaccurate position readings. A Hall effect sensor, while less accurate in terms of resolution, does not suffer from drift and can maintain an absolute measurement of position over time.

To address this issue, we considered using both sensors together. By combining the high accuracy of the optical encoder with the drift-free absolute position measurement of the Hall effect sensor, we could use sensor fusion techniques, such as a Kalman filter or complementary filter. This approach would allow us to leverage the strengths of both sensors — combining the precision of the encoder with the stability of the Hall effect sensor — while mitigating their individual limitations.

The encoders interface with the microcontroller using two logical signals, A and B. These signals represent the states of the optical readings from the encoder (Figure 4). The time between rising and falling edges of signals A and B are linearly related to the angular displacement, and they can be analyzed by a microcontroller to keep track of the angular displacement. By analyzing the relative timing of these signals, the microcontroller can also determine the direction of motion.

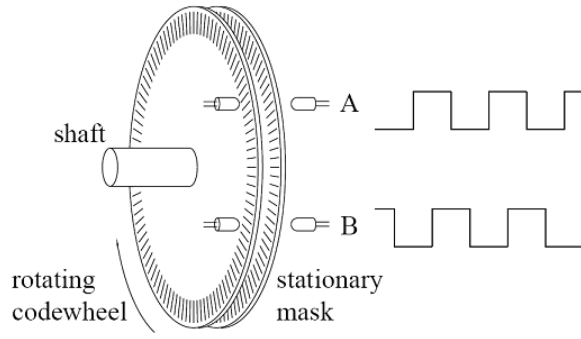


Figure 4: Internals of an Optical Encoder [3]

The microcontroller keeps track of the encoders movement by storing it's state at every rising and falling edge. The state of the encoder is the concatenated signal from A and B. For example, if A is high and B is low then the state is 10; if A is low and B is high then the state is 01; if A and B are high then the state is 11. Then, using the encoder transition table 1, the microcontroller determines changes in position based on the current and previous states of signals A and B. Each row in the table represents a possible previous state (combinations of A and B), while each column represents a possible new state after a transition. The values in the table correspond to the positional change associated with each transition, whether positive, negative, or zero.

Whenever a rising or falling edge is detected on signals A or B, the microcontroller records the new state, combines it with the previous state, and references the transition table to determine the change in position. This change is added to a position counter, which keeps track of the encoder's total movement.²

Previous ↓ New →	00	01	10	11
00	0	-1	1	0
01	1	0	0	-1
10	-1	0	0	1
11	0	1	-1	0

Table 1: Encoder Transition Table.

2.1.3 Time of flight sensor

Two time-of-flight sensors were included in the original design for cart location validation. Because the Time-of-flight sensors measured absolute distance instead of the change in distance, they were less prone to drifting and giving inaccurate readings on the cart's location. The VL53L0X from STM were chosen due to their cost and easy availability. The sensors were addressed over a TWI

²A value of zero from the transition table 1 represents a combination of signals that does not correspond to a valid movement, typically due to signal noise or an invalid state transition. In such cases, the position remains unchanged.

Bus that linked them to the AVR and mounted to the belt-side rotational encoder.

As driver development for the VL53L0X began, it became apparent that the data sheet was missing the communication protocol. The lack of information in the data sheet increased the complexity of driver development, as blogs and existing drivers had to be used for reference instead. As Development progressed, driver development was abandoned in favor of using an Arduino with an existing library to translate between the AVR and sensor. Ultimately, driver development was abandoned due to a lack of belt drift compared to pendulum rotation drift along with the high time cost to incorporate the sensor into the existing system.

2.1.4 Mechanical Design and Assembly

For the mechanical aspects of the system, we used On-shape for CAD modeling and design. Starting with an existing project [4], we edited files, created new components, and tested designs through 3D printing and laser cutting. Here is a link to our on-shape project for all the CAD files. This approach allowed us to quickly iterate on our designs and make adjustments as needed. We also designed parts such as larger gears for the motor to increase speed, pulley tensioners, stands to support the rail, and electronics mounts.

Once all components were designed and fabricated, we assembled the system by mounting the motor and one encoder above the rail, vertically facing downward. The motor and encoder were attached to a pulley system that ran parallel to the rail. The cart was designed to snugly fit around the rail, enabling smooth back-and-forth motion. The pulley was attached to the cart, allowing the motor to drive it, while the encoder measured the cart's movement. On the cart the other encoder was mounted to measure the motion of the pendulum. The pendulum was mounted on that encoder's shaft. The assembled system can be seen in Figure 5.

2.1.5 Why We Chose This Hardware Setup

We selected this specific hardware setup based on its balance of simplicity, cost, and functionality. The DC motor provided an effective solution for controlling the cart's speed and direction, while the optical rotary encoders gave us the accuracy required for precise positioning. Additionally, the use of on-shape and rapid prototyping through 3D printing allowed us to quickly iterate on our designs and make improvements, ensuring that the system met our performance and cost requirements. This setup strikes a balance between performance and ease of implementation, making it ideal for our cart-pole system.

2.2 Software

2.2.1 Networking

using the USART protocol and Serial. Packets were sent bi-directionally in the form of packets. A Dedicated packet structure was created for each direction of communication. Outgoing packets from the micro-controller had 2 32-bit integers while incoming only contained one byte. To validate data the outgoing packets contained both a delimiter and CRC to help ensure that data was accurate. CRC calculations were modeled after the RS-232 Telecommunications protocol. Furthermore, 3-byte Delimiters were sent after every packet to streamline data identification and alignment.

To handle communications on the AVR BLE. A previous USART driver was modified and expanded to support the system's needs. To keep the driver from blocking when waiting, all functions related to receiving data were developed so that they would exit when needing to waiting for a register state, and pick up where they left off if called again. On top of the UART driver, packet creator and parsing functions were developed to streamline the use of the communications protocol.

To handle communications on the computer. The SerialPy library was leveraged to handle communication with the board. Running in a simple loop, the library reads packets until the expected 3-byte delimiter is processed. It would then read all previous data into a buffer and verify it with the CRC algorithm. If the data bytes were valid the function would run the PID controls algorithm and produce an output voltage that would be sent out using the library.

Multiple difficulties were encountered during the communications protocol development related to integrating the AVR BLE with the existing controller code. Because the code base was built around interrupts, sending operations would block other interrupts while a transmission was outgoing. This resulted in less accurate encoder data as the AVR would not register interrupts and the position would start to drift. To combat the interrupt problems both sending and receiving protocols were changed to be non-blocking. This was achieved by giving both functions static variables and allowing them to exit during times when they would typically wait for an interrupt flag. Both interrupts were also moved to the main loop of the code base to reduce the time spent handling interrupts inside the system.

After implementing the non-blocking protocol, a second issue was discovered involving the AVR receiving function not registering all incoming bytes. Due to the low precision of the motor driver, the packet protocol was reduced to a single byte. This did not validate the data, but was still reliable enough for our purposes because errors in the output voltage could be tolerated due to the high transmission rate. This eliminated the issue of bytes not being read as the next byte would encode the same data as the previous byte.

2.2.2 Controller

Initially, we considered designing a controller based on a model of the system. We successfully derived a model for the system with the DC motor, but this model required measuring the current applied to the motor in order to make the system fully observable. Additionally, the state-space model described earlier (Equation 1) assumed that the action driving the system was a force applied to the cart. However, our motor couldn't accurately control the torque on the pulley, meaning it couldn't apply a precise force to the cart. This limitation made the state-space model impractical for our system.

Since we couldn't model the system accurately, we opted for a simpler control approach: a PID (Proportional-Integral-Derivative) controller.

Despite not having a model, we could still measure the system's state with the state vector:

$$\mathbf{x} = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^T$$

Using this state vector, we designed a PD (Proportional-Derivative) controller. The control input u is the motor's velocity, which is controlled by adjusting the voltage applied to the DC motor via a PWM signal. The voltage is linearly related to the speed of the cart ($\dot{x}$).

When the pendulum starts to fall from an upright position, the cart must move beneath the pendulum's center of gravity to catch it and prevent the pendulum from falling. The term $k_1\theta$ in the controller achieves this behavior, as when θ increases (pendulum tilts), the cart moves in the corresponding direction to catch the pendulum. The value of k_1 must be large enough to be able to catch the pendulum before it fully falls over, but there is also a limit of how large k_1 can be because the motor can only spin so fast.

The term $k_2\dot{\theta}$ serves as a damping term. If the pendulum is swinging fast in one direction, the cart moves in the opposite direction to slow it down, helping stabilize the system. This is formulated the same way friction is.

The remaining terms, $-k_3x$ and $-k_4\dot{x}$, help center the cart. For example, if x is positive it will attribute a negative term to our action u causing the cart to move backwards towards the center. The $k_4\dot{x}$ term is again a damping term, so the cart would not oscillate around the center. Importantly, the values of k_3 and k_4 are chosen to be significantly smaller than k_1 and k_2 to prioritize balancing the pendulum over centering the cart.

Thus, the control law is:

$$\begin{aligned} u = \mathbf{k}^T \mathbf{x} &= [-k_3 \quad -k_4 \quad k_1 \quad k_2] \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \\ &= k_1\theta + k_2\dot{\theta} - k_3x - k_4\dot{x} \end{aligned} \quad (2)$$

This control law allows us to update the motor's velocity u in response to the system's state, achieving both stabilization and positioning.

In our controller design, we used the following values for the controller gains: $k_1 = 14$, $k_2 = 0.1$, $k_3 = 0$, and $k_4 = 0$. These values were selected based on a balance between the available time for tuning and the need to achieve some level of system performance.

However, we were unable to fully utilize all the terms in our controller due to drift in both the position and angular readings. This drift caused inaccuracies in our state measurements, limiting the effectiveness of the centering and damping terms. Additionally, we had limited time to experiment with different values for the controller gains, preventing further fine-tuning of the system. As a result, the controller focused primarily on stabilizing the pendulum's angle ($k_1\theta$ and $k_2\dot{\theta}$) rather than centering the cart (k_3x and $k_4\dot{x}$).

3. Final Product and Reflection

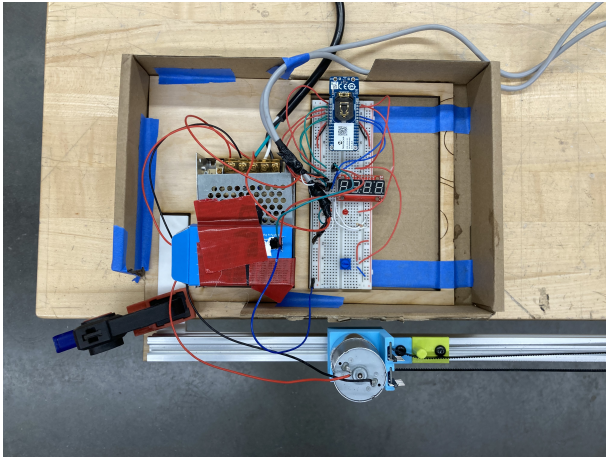


Figure 5: Electronics of The Physical System



Figure 6: Full Physical System in the Downright Position

We successfully built a system and partially designed a controller to balance the pendulum for a short period of time. The system is pictured in Figure 5 and 6. You can view a video of the balancing pendulum here: [video demo](#). The system was operated by first setting it up in a predetermined calibration position. The calibration position consists of moving the cart to a marked centerpoint on the rail and allowing the pendulum to hang freely without any sway. The AVR board was then powered through connection to the computer, allowing the codebase to startup at its predetermined values of a max angle and no cart deviation. Afterwards the controller python script was ran to start processing the sensor values and giving a desired motor voltage. The pendulum was then manually flipped up with assistance of the python script to assure it is at an angle deviation of 0. Finally the DC power supply was plugged into a power socket allowing the controller to actuate the DC motor. This resulted in the cart actively balancing the rod until enough drift accumulated in the pendulum encoder for it to no longer target an optimal position. Once the rod stops balancing the user would exit out of the python program, sending one final command to the motor to reset it to a zero voltage state.

3.1 Achievements

We set out to build a cart-pole system, estimate its state, and control it. We successfully built the system and achieved functional state estimation, despite some imperfections. Using a PD controller, we were able to balance the pendulum for short durations.

Along the way, we gained valuable skills in CAD, 3D printing, woodworking, and soldering. We also learned the basics of control theory and state-space modeling for a cart-pole system. Additionally, we applied our knowledge of networking and microcontrollers to develop drivers and protocols, enabling us to create the system's software and firmware.

Along the way, we gained valuable skills in CAD modeling, 3D printing, wood laser cutting, soldering, and more. To build the system, we had to understand DC power circuitry to select and integrate components like the DC motor driver and power supply, ensuring they would work together. We also developed drivers for components such as rotary encoders, time-of-flight sensors, and DC motors, significantly expanding our knowledge of embedded systems. Additionally, we learned advanced networking techniques to interface with sensors and establish full-duplex communication with the computer. This capability allows for future controller designs to be developed and tested on a computer while keeping the firmware unchanged, offering a flexible platform for exploring more advanced control and estimation techniques.

Furthermore, We take pride in the clean, professional aesthetic of our design, the neatly house electronics, and the flawless integration of our CAD-designed parts. Fi-

nally, we value that this system could be used a foundation to learn more complex algorithms for control theory and optimal estimation.

3.2 Failures

Initially, we aimed to design a swing-up and balancing controller using energy pumping and linear quadratic regulation (LQR). However, due to the high cost of system construction and the challenges of obtaining accurate sensor readings, we focused solely on the balancing controller, using a PID (Proportional-Integral-Derivative) approach. While this allowed us to succeed partially, as we fell short of our original goals due to several factors.

One significant issue was sensor drift, which we did not anticipate when designing this system. Sensor drift occurs when the sensors gradually lose calibration, leading to inaccurate readings. This drift impacted the accuracy of our system, especially when tracking the pendulum's position. Once the sensor started to drift away from the actual position it was measuring, there was no way to recover or detect when it occurred on the software side. Furthermore, after further examination of our rotary encoders, we found that the documentation explicitly stated that vibrations could cause inaccurate readings. Since we mounted an encoder to a moving base, the vibrations introduced ample opportunities for sensor drift to occur. Unfortunately, this problem cannot be resolved with our current setup. Additional sensors would be needed to correct for drift and provide more reliable measurements. For instance, adding a Hall effect sensor to complement the rotary encoders would allow for more precise tracking of the pendulum's position and prevent drift from accumulating. Without these additional sensors, the drift compromised the performance of the PID controller because, without accurate data, accurate motor commands cannot be generated — crippling the system.

3.3 Improvements and Extensions

After underestimating the impact of sensor drift, we would make addressing the sensor drift a priority. We would add more sensors, such as a Hall effect sensor, and implement sensor fusion techniques like Kalman filtering to combine data from multiple sources and minimize the impact of drift. Our optical encoder is accurate but drifts, and the Hall-Effect sensor is inaccurate but never drifts. So by fusing these two sensors we would get the best of both worlds: accuracy and minimal drift.

If we had more time, we would also use a brush-less motor with torque control for more accurate system control. Because if you analyze the system mathematically the system is driven by the acceleration [cite state-space model from intro] of the cart not the speed, so having a torque controller which controls the cart's acceleration would be pivotal.

We would also revisit the controller design. While the

PID controller was a reasonable choice given the time constraints. However, it doesn't take into consideration the dynamical model of the system (Equation 1). Therefore, a more sophisticated control approach that does consider the systems dynamics, such as LQR (Linear Quadratic Control), would provide better results.

Finally, we would extend the system to include the swing-up controller as we initially planned. This would only require drift-less sensor readings. The swing-up controller works simply by taking the gradient of the system's energy with respect to the action your system can take. By doing so, it can determine the most efficient way to drive the system from a hanging state to an upright position. For more information on the topic refer section 3.6 of [2].

3.4 Why you Should Care

Although we fell short of our initial expectations, we have built a system that will serve as a springboard for further learning. It will allow us to explore more advanced concepts in control theory, sensor fusion, and system dynamics. With a solid understanding of the system's hardware and software, we can now focus on improving our controller's performance and robustness, especially by addressing issues like sensor drift. This project has also opened doors to deeper dives into energy-based control methods, such as the swing-up controller, which will be an exciting next step. Additionally, our hands-on experience with CAD modeling, embedded systems, and control theory lays a strong foundation for tackling more complex engineering projects in the future. This project exemplifies the quote: "A jack of all trades is a master of none but better than a master of one" - Unknown.

References

- [1] Instructables. Inverted pendulum control theory and dynamics. <https://www.instructables.com/Inverted-Pendulum-Control-Theory-and-Dynamics/>, 2024. Accessed: 2024-12-12.
- [2] Russ Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines, 2022. Section 3.6: Swing Up Control, accessed December 12, 2024.
- [3] Northwestern University. Encoder diagram, n.d. Accessed: 2024-12-12.
- [4] Ben Wiener. Base onshape project, 2018. Accessed: 2024-12-12.
- [5] Ben Wiener. Zipy. <https://blog.benwiener.com/programming/2018/05/10/cart-pole.html>, 2018. Accessed: 2024-12-12.

- [6] YouTube. Inverted pendulum control. <https://www.youtube.com/watch?v=BAG7Gtw-cqE>, 2024. Accessed: 2024-12-12.