**Politecnico di Torino**

**Master's Degree in Mechatronics Engineering**

**Master's Degree Thesis**

# Design and Implementation of an Inverted Pendulum Control System using FPGA and Reinforcement Learning

**Supervisors**

**Prof. Stefano Alberto Malan**

**Prof. Eduardo de la Torre**

**Candidate**

**Akbarkhon Abdusamadov**

**June 2023**

# Acknowledgements

I would like to express my sincere gratitude to my advisors, Eduardo de la Torre from Universidad Politécnica de Madrid, and Stefano Alberto Malan from Politecnico di Torino, for their invaluable guidance and support throughout my double master's program. Their expertise and encouragement have been instrumental in the completion of this research and the writing of this thesis.

I would also like to extend my heartfelt thanks to all the members of the Center for Industrial Electronics at the School of Industrial Engineering, Universidad Politécnica de Madrid, who contributed to my thesis. Their insightful feedback and valuable suggestions have significantly improved the quality of my work.

Furthermore, I am deeply grateful to my friends and family for their unwavering love and support during this process. Without them, this journey would not have been possible.

Finally, I want to express my sincere gratitude to all the participants in my study. Their willingness to share their experiences and insights has been invaluable to my research and has significantly contributed to the success of this thesis. Thank you for your time and contribution.

I am grateful to everyone who has supported me throughout this arduous process. Your guidance and assistance have been indispensable, and without your unwavering support, this thesis would not have been possible.

# Abstract

The inverted pendulum system is a widely recognized control problem that revolves around the task of stabilizing a pendulum connected to a movable base. The primary objective is to keep the pendulum in an upright position, despite its inherent instability, by manipulating the movement of the cart. This system serves as a crucial benchmark for evaluating and testing control algorithms, finding applications in diverse domains such as robotics and industrial automation. Effectively controlling the inverted pendulum has far-reaching implications for the advancement of autonomous vehicles, bipedal robots, and other dynamic systems that rely on maintaining balance and stability.

This master's thesis project aims to tackle the inverted pendulum problem from various perspectives, with three main objectives. Firstly, an environment of the inverted pendulum will be constructed in OpenAI Gym, and reinforcement learning techniques, specifically Proximal Policy Optimization (PPO), will be utilized to achieve balance. By training an agent to control the base, the objective is to demonstrate the capability of reinforcement learning algorithms in solving control problems.

Secondly, a MATLAB/Simulink mathematical model of the cartpole system, representing the inverted pendulum, will be developed using Lagrangian mechanics. This model will provide insights into the dynamics of the system and serve as a foundation for designing and implementing control strategies. Control strategies such as PID (Proportional-Integral-Derivative) and LQR (Linear Quadratic Regulator) will be designed and implemented in Simulink. The system will be controlled in simulation, and preliminary results will be obtained to determine parameters for the real model.

Lastly, the control strategies designed before will be tested on an existing real model. A special HW/SW system will be designed using the PYNQ-Z1 FPGA board. The pendulum position will be observed using a rotary encoder with an optical interrupter, while the cart position will be estimated from the control output applied to the stepper motor by the processor. All input sensor signals will be processed by the programmable logic (PL) before being passed to the processor system (PS). Control algorithms, such as PID and LQR strategies, will be implemented in software using high-level programming languages like C through the Xilinx Vitis application environment. The integration of hardware and software components aims to showcase real-time control of the physical inverted pendulum and highlight the advantages of using FPGA boards in control and sensing applications.

By achieving these objectives, this thesis contributes to the understanding and application of control strategies and reinforcement learning algorithms in the context of a problem such as the inverted pendulum system. The results and insights gained from the simulations, mathematical modeling, and physical implementation will enhance the knowledge of control systems and their practical implementation. Furthermore, the successful utilization of reinforcement learning in OpenAI Gym environments demonstrates the potential for solving complex control problems.

## UNESCO Nomenclature

1203.04 Artificial Intelligence

1203.09 Computer-assisted Design

1203.25 Sensors System Design

1203.26 Simulation

3304.12 Control Devices

3304.16 Logic Design

3304.17 Real-time Systems

# Table of Contents

# Chapter 1

## 1. Introduction

An inverted pendulum on a cart, also known as a cart-pole system, is a system consisting of a pendulum with its mass center located above its pivot point and pivoted on a rigid rod. The inverted pendulum is inherently unstable and highly nonlinear, requiring a control system to compensate for the pendulum angle deviation by applying a balancing horizontal force at its pivot point.

In this thesis, the inverted pendulum system is implemented using a metallic rod attached to a wheelless cart that moves along two metallic guides. The cart is driven by a belt and pulley system connected to a stepper motor. The system consists of two degrees of freedom: the position of the cart, represented as "x," and the angular position of the pendulum, represented as "theta." This setup is illustrated in Figure 1.1.

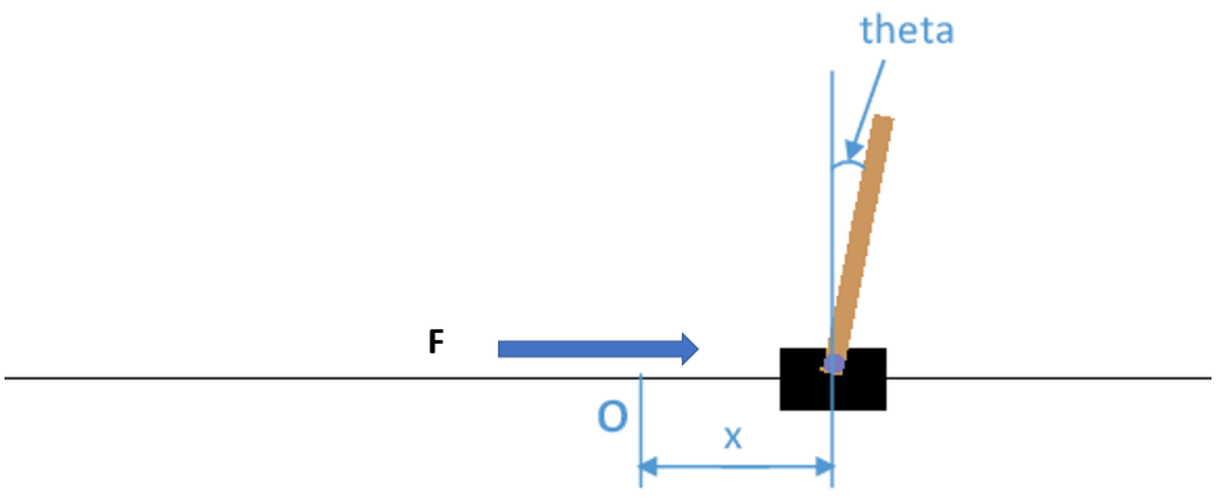


Figure 1.1: Inverted Pendulum

Despite the inverted pendulum system having two degrees of freedom, the control input for the system is a single horizontal force applied to the cart. The main objective is to design an embedded control system, including the necessary electronics, to ensure that the pendulum remains in its upright position while the cart maintains its reference position. The design of the control system requires careful consideration to effectively achieve this goal.

## 1.1 Thesis Background

This Master's Thesis is an extension of two previous Bachelor's Theses projects conducted at the Center for Industrial Electronics (CEI) within the School of Industrial Engineering (ETSII) at the Universidad Politécnica de Madrid (UPM). The projects focused on the physical design of a guided cart inverted pendulum model and the development of a control system for its electronic components. The CEI provided the necessary tools and the actual inverted pendulum model for the projects.

The current Master's Thesis builds upon the previous work and aims to further advance the field. It is part of a double master's degree program in Mechatronics Engineering from Politecnico di Torino and Industrial Electronics from Universidad Politécnica de Madrid. The objective of this research is to design hardware and software applications for control strategies using System on Chip (SoC) technology, which combines a processor and FPGA (Field-Programmable Gate Array) technology. Additionally, the thesis explores the implementation of both conventional control algorithms and reinforcement learning algorithms within the inverted pendulum system.

This Master's Thesis focuses on the development of new IPs, hardware modules, and software functions to enhance the control system of the inverted pendulum. The PYNQ-Z1 prototyping board, previously used in the Bachelor's Theses projects, is also utilized in this research. Furthermore, the reinforcement learning algorithms applied to the inverted pendulum control are tested using the OpenAI Gym environment [1].

By combining FPGA technology, control theory principles, and reinforcement learning, this research aims to improve the stability and performance of the inverted pendulum control system. It contributes to the field of control systems and showcases advancements in control strategy design and implementation for the inverted pendulum system.

## 1.2 Objectives and System Overview

This thesis aims to achieve three main objectives:

1. Designing a mathematical model to simulate the cart and pole system in Simulink/Matlab. Several controllers will be designed and tested to maintain the pendulum in an upright position. The control system will be based on PID and LQR control methods.
2. Development of a control system using the Zynq-7020 FPGA board, based on preliminary results obtained from the Simulink model. This involves designing and testing hardware-based IPs for sensor signal processing and generation, as well as developing a platform and application program for controlling all signals and the stepper motor. The aim is to achieve pendulum balancing using the processing system (PS) of the board.
3. Familiarization with the Gym OpenAI toolkit and the design of a cartpole prototype

environment. This involves creating a simulation model of the cart and pole system to test control strategies without the need for a physical model. Additionally, reinforcement learning techniques will be applied to a black box model of the inverted pendulum using Gym OpenAI in Python. The goal is to achieve cartpole balancing.

Figure 1.2 illustrates a simplified sketch of the cartpole system. The wheelless cart moves horizontally using a belt and pulley system, with the trajectory limited by the belt length. The pendulum is attached to the cart and can freely rotate 360 degrees in both clockwise and counterclockwise directions. All sensor signals and motor control signals will be processed by hardware modules and the microcontroller of the Zynq-7020 FPGA device. Additionally, a mathematical model of the inverted pendulum will be developed to describe the mechanical behavior of the system and facilitate simulation using appropriate tools.

The next step involves implementing the derived model in MATLAB/Simulink for simulation purposes and testing and validating the proposed control strategies. Simultaneously, a black box model of the cartpole environment will be designed and controlled using reinforcement learning techniques in OpenAI Gym. This approach allows the controller to learn to keep the pendulum upright without relying on the knowledge of underlying mathematical equations. All tasks within OpenAI Gym will be programmed in Python.

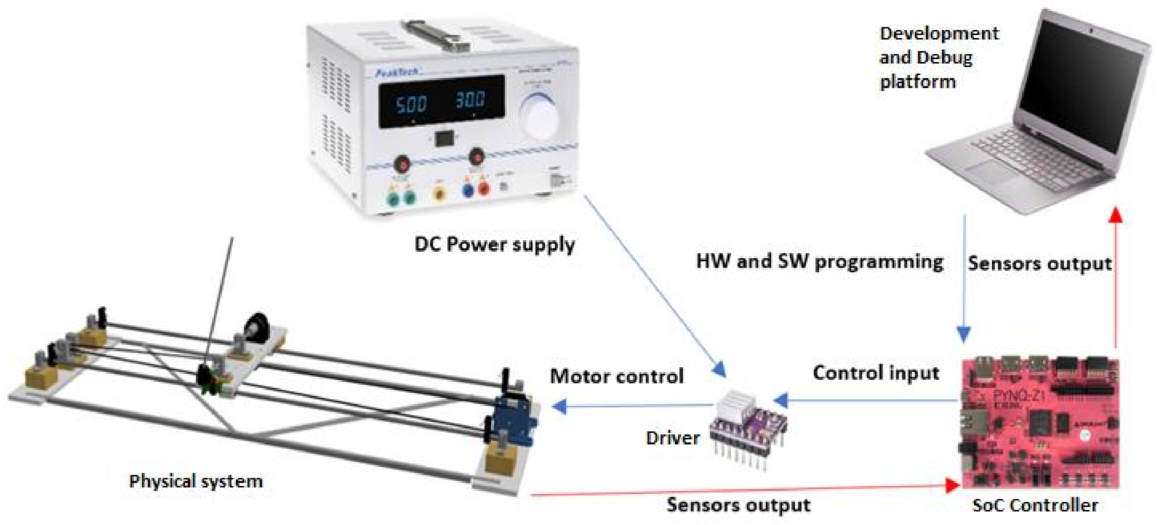


Figure 1.2: Simplified Sketch of the Inverted Pendulum Control System

Finally, after implementing the hardware and software designs using Xilinx Vivado and Xilinx Vitis, respectively, the inverted pendulum control system will be deployed and tested on the PYNQ prototyping board. Real-time experiments will be conducted to evaluate the performance and robustness of the implemented control strategies, including the PID controller and the LQR controller. The results obtained from these experiments will provide valuable insights into the effectiveness of the proposed control approaches and their applicability in practical scenarios.

## 1.3 Thesis structure

Chapter 1 introduced the inverted pendulum control system and discusses possible control strategies that can be applied to it. It also provides a brief overview of the reconfigurable logic device, specifically the FPGA, and its characteristics. The chapter covers the goals and objectives of the thesis.

Chapter 2 focuses on the system design and introduces the physical components of the system, including sensors, actuators, and the step motor. The chapter explains the overall working principle of the system and provides an overview of its physical model.

Chapter 3 presents a mathematical model approximation for the inverted pendulum system. It covers the theoretical aspects of the applied methods, including tests and parameter selection for the proposed mathematical model. Additionally, the chapter describes the design and visualization of the Simulink model within the platform.

Chapter 4 details the design of the inverted pendulum system model in MATLAB/Simulink and the development of control strategies using PID and LQR controllers. It includes the tuning of the controllers and an analysis of the effectiveness of the control strategy.

Chapter 5 explores the OpenAI Gym platform and its application in implementing a reinforcement learning method for the control of the cartpole system. The chapter covers the theory and application of PPO reinforcement learning and provides Python scripts related to environment design and the reinforcement learning of the controller.

Chapter 6 focuses on hardware design and provides a comprehensive explanation of the Hardware Block Design methodology used to configure an FPGA for signal detection, processing, and control input generation in a stepper motor driver carrier. It describes the design of Intellectual Property (IP) modules for essential components such as the optical interrupter, limit switch, rotary encoder, and motor driver. The chapter also includes an analysis of component utilization and energy consumption in the FPGA hardware design, providing valuable insights into resource allocation and optimization strategies.

Chapter 7 focuses on the software design of the control system. It explains the development of special functions using the C programming language and pre-designed IP components to implement a PID controller. These functions enable parallel computing and generate control outputs that are transformed into control inputs for the stepper motor. The chapter also discusses the implementation of PID and LQR control strategies.

Chapter 8 analyzes the results obtained from the PID and LQR controller system, evaluating the overall behavior and functionality of the system. The chapter also addresses the physical aspects and limitations of the system.

Chapter 9 concludes the thesis and outlines future work. The conclusion summarizes the design and implementation of the system, including its hardware and software components, and presents the results of the analysis. The future work section identifies potential areas for improvement, such as enhancing accuracy and incorporating new features into the system.

# Chapter 2

## 2. Physical System Components

The inverted pendulum control system model and its parts of both sensors and actuators architecture and functionality are described in this chapter. The system model used in this thesis was developed by a former student of the Technical University of Madrid, Óscar Mate Castro, for his final degree project [2].

The model consists of a pendulum hinged on a wheelless cart. The cart is driven by a belt and pulley system connected to the stepper motor. The cart moves along two supporting metallic shafts. A rotary encoder and an optical interrupter serve as sensors to estimate the pendulum angular position. Four limit switches are placed at each end of the supporting shafts to detect when the cart reaches the bounds of its trajectory. The system is mounted on the supporting platform consisting of two PVC bases connected by aluminum rods connected with each other in "L" shape to achieve tolerance to perturbances during cart movements and motor vibrations. The stepper motor is placed in a special plastic mold which is screwed to the base. The overall system 3D model can be seen in the figure below.

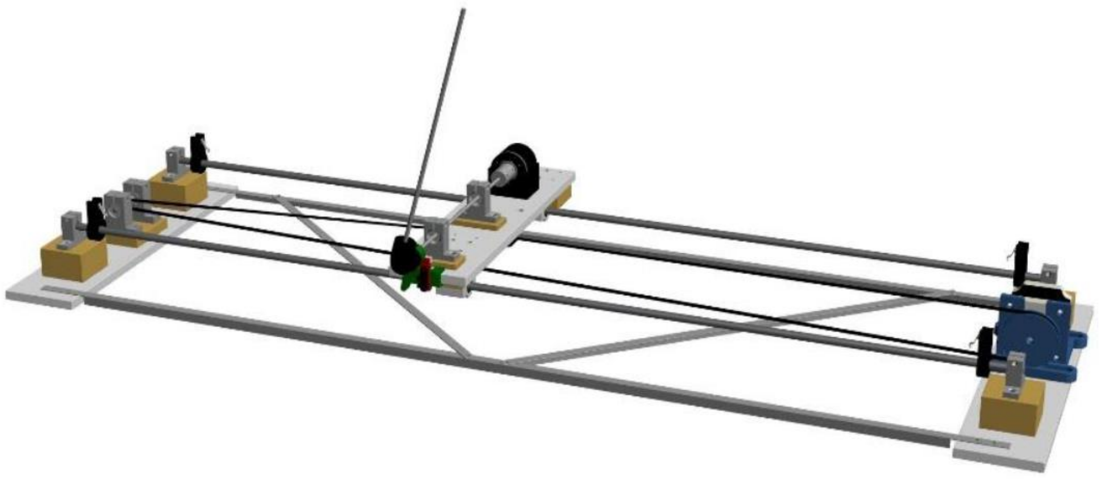


Figure 2.1: 3D model of the system [2]

The stepper motor is controlled by the motor driver carrier which is powered from a DC Power supply and is connected to the PYNQ prototyping board containing the FPGA which is programmed by the PC. HW design is written in VHDL in Vivado by Xilinx and the PYNQ processor is programmed in C language on Xilinix Vitis software. The overall control system signal and power intercommunication can be seen in the diagram below.

## 2.1 Optical Interrupter

The optical interrupter is used to establish the initial reference of the angular position of the pendulum when it is in the downward position. Thus, when the pendulum rotates, a relative encoder provides signals to measure the angular displacement of the pendulum from that initial position. This sensor would not be necessary if an absolute encoder had been used.

The optical interrupter used in the inverted pendulum system is based on the TCST2103 optical sensor [3]. This sensor includes an infrared emitter, a phototransistor, and a logic circuit. Its main function is to detect the presence of an object between its optical axes and provide protection against external light sources. When an object is detected, the red light emitted by the sensor is deactivated, indicating the presence of the object to the user. Figure 2.2 illustrates the component location and schematic of the optical interrupter within the system.

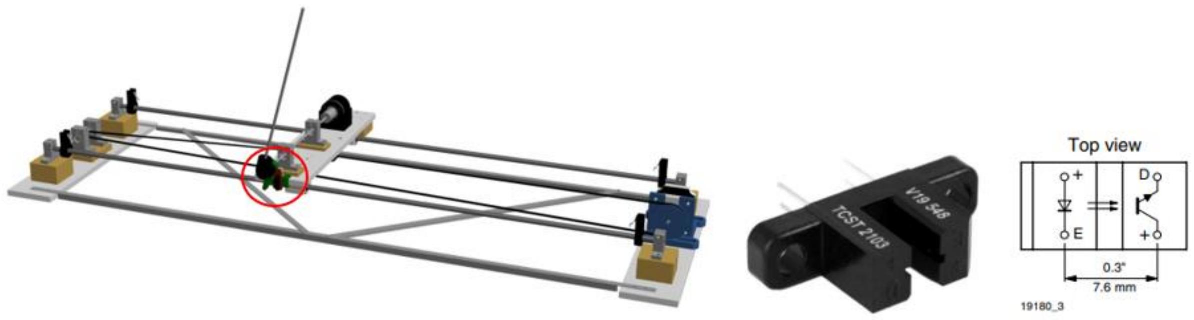


Figure 2.2: Optical interrupter

A pendulum is aggregated with an additional small rod to trigger the sensor when it passes or stays at downward position. As illustrated in Figure 2.3, it is located next to the coupling of the arm with the encoder shaft.

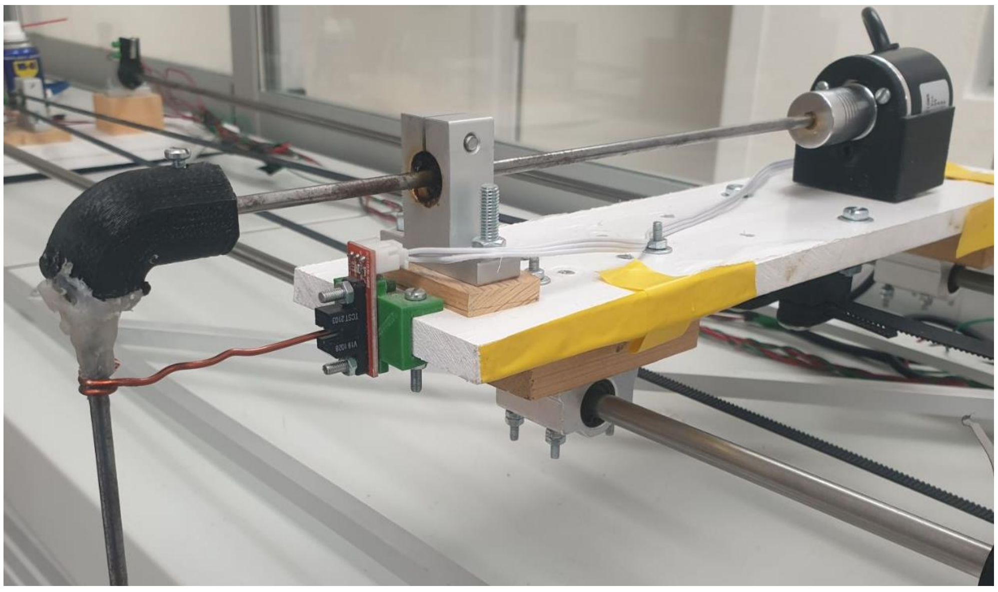


Figure 2.3: Pendulum and optical interrupter design

The output signal from the Optical Interrupter IP is a 3.3V DC signal, which is directly connected to a pin on the PYNQ-Z1 prototyping board. Although the rising and falling edges of the switch signal are generally reliable, it is crucial to consider the possibility of external noise interference during the signal reading process. To ensure precise and accurate signal reading while mitigating the impact of potential noise, a hardware-based debounce filter will be implemented. This debounce filter, with a debounce delay set to 1ms, effectively filters out any undesired noise and guarantees reliable detection of the signal. By applying the debounce filter, the system achieves robust and accurate signal processing, enhancing the overall performance of the optical interrupter module.

## 2.2 Rotary Encoder

To accurately determine the current angle of the inverted pendulum, a two-channel magnetic relative rotary encoder has been implemented. The primary purpose of this relative encoder is to monitor and track the angular displacement of the pendulum.

As depicted in Figure 2.4, the pendulum is connected to the relative encoder through a shaft. This shaft is secured to the cart using two ball bearings, ensuring stable movement. Additionally, the pendulum is firmly attached to the shaft via a joint that can be adjusted using a bolt. This arrangement allows for precise measurement of the pendulum's angle by the relative encoder.

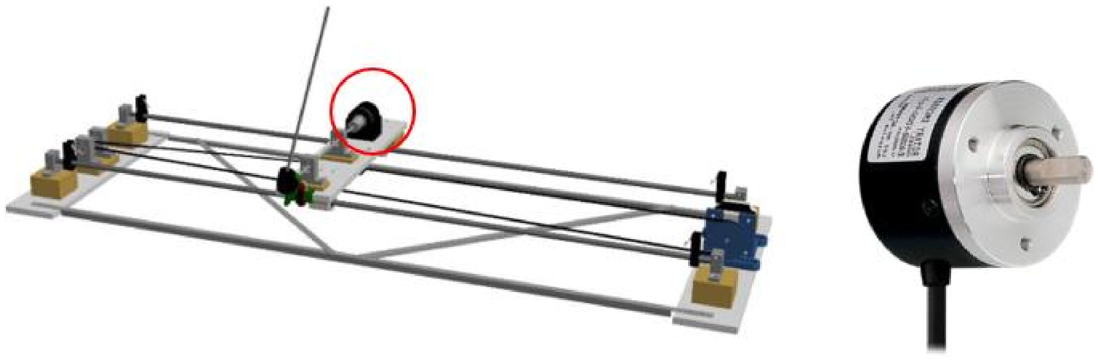


Figure 2.4: Rotary encoder

The relative encoder translates the angular position into a digital signal. The most common type of rotary encoder is a quadrature encoder. Two output signals in quadrature (90° out of phase) are used to encode the position of the pendulum as shown in Figure 2.5. During rotation, 2 rotating tracks A and B produce square waves with 90° difference in phase. Channel A leads Channel B when the pendulum rotates clockwise, and Channel B leads Channel A when the rotation is counterclockwise.

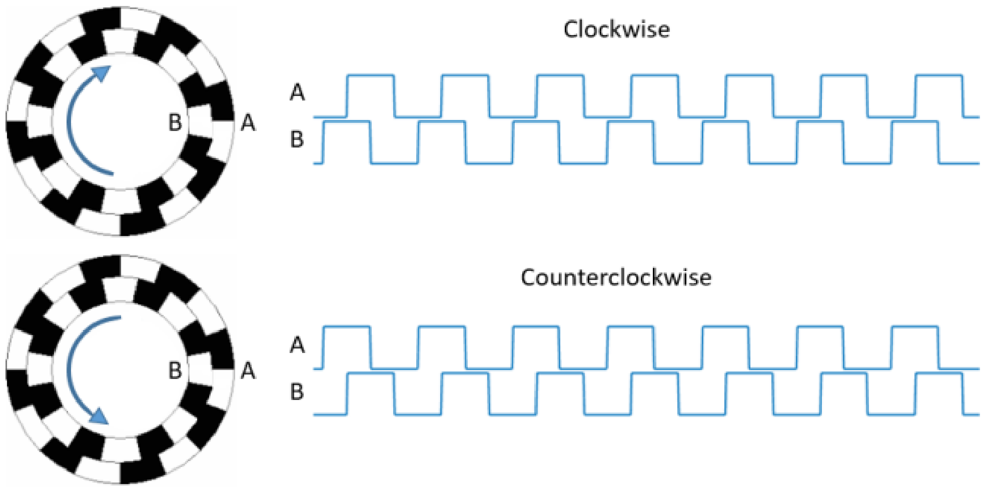
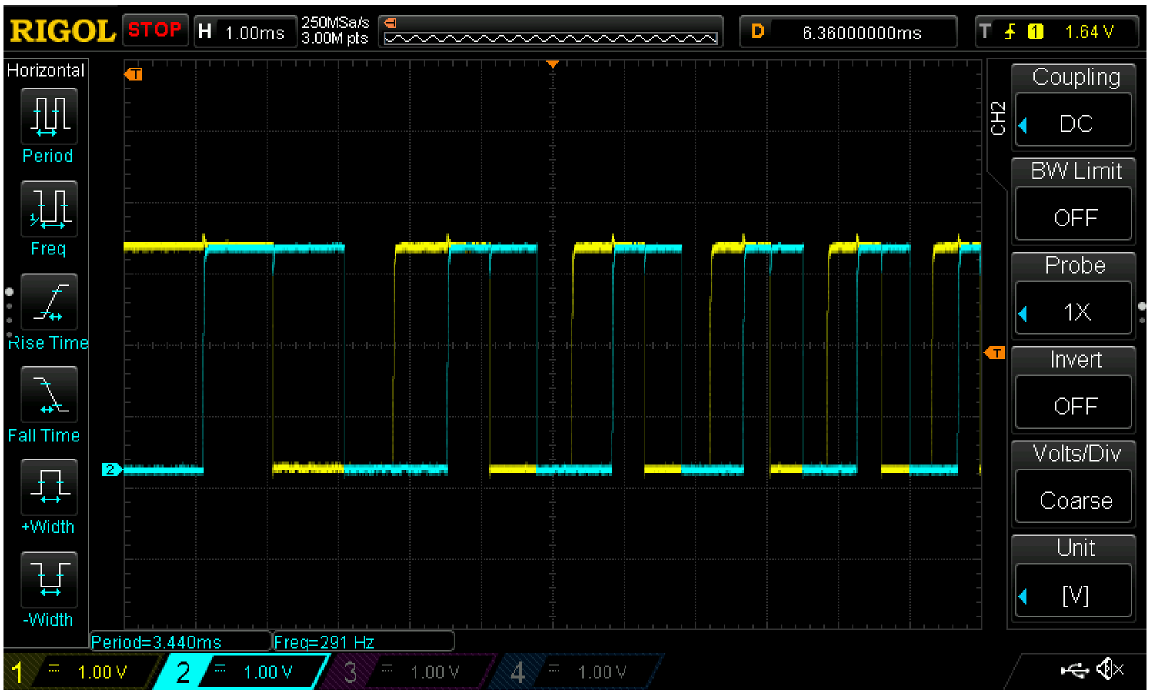


Figure 2.5: Rotary encoder working principle [4].

The encoder employed in the system generates 600 pulses per rotation in each channel. Although this resolution may not be considered high, it is sufficient to meet the requirements of the model. The output signals are measured with a nominal supply voltage of 3.3VDC, and a pull-up resistor of 18 kΩ is connected in series with the sensor outputs. These signals maintain the same voltage level as the supply voltage and exhibit a phase difference of 90º [2].

The direction of rotation can be determined by analyzing the two-channel signals. As depicted in Figure 2.6, the blue signal leads the yellow signal, indicating a turning sense of 1. Conversely, in Figure 2.7, the yellow signal leads the blue signal, indicating a turning sense of 2. It is worth noting that the pulse duration generated by the tracks becomes shorter as the rotation speed increases.



Period=3.440ms
Freq=291 Hz

Figure 2.6: Output signal of the encoder (Turning sense 1).

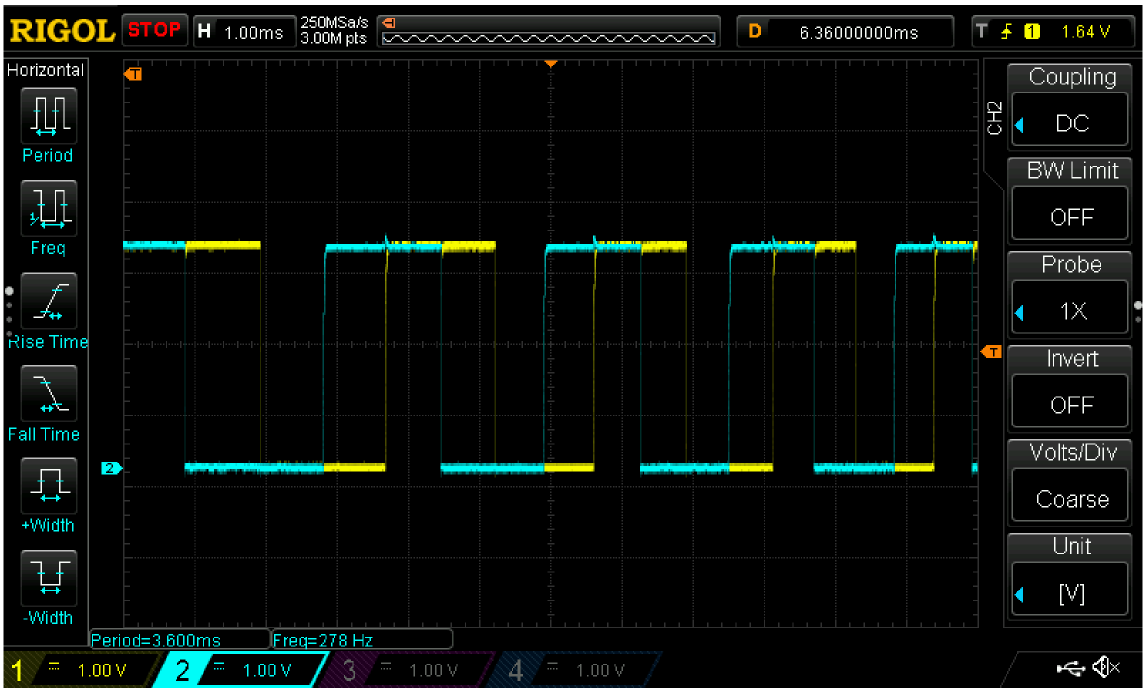


Figure 2.7: Output signal of the encoder (Turning sense 2)

While the signals generated by the relative encoder do not exhibit any bounces at medium and high rotating speeds, it is observed that both output signals experience a significant number of bounces at low speeds, as depicted in Figure 2.8. This poses a challenge for accurate position detection, particularly when the pendulum is positioned upwards and oscillates at a low frequency.

The presence of these bounces in the signals can lead to incorrect readings and inaccurate position estimation of the pendulum. Additional measures, such as implementing a debouncing mechanism is necessary to mitigate these bounces and improve the reliability of the position detection system, especially in scenarios where the pendulum oscillates at low frequencies.

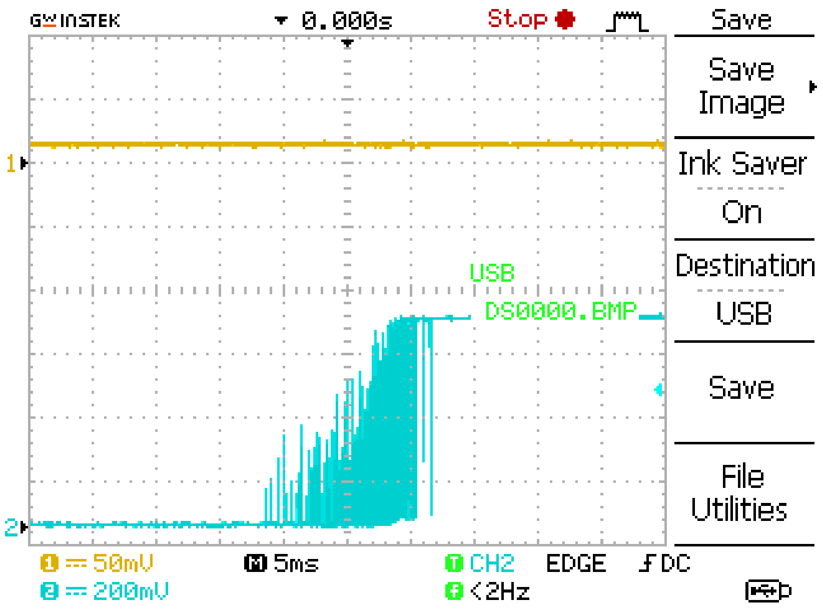


Figure 2.8: The output signal of rotary encoder containing extra bounces.

## 2.3 Limit Switches

Limit switches are crucial components that detect the constraints limit on the rails, providing valuable feedback to the control system. The system incorporates the reliable SS-5GL13 limit switch with a simulated roller lever actuator for accurate detection. Figure 2.9 illustrates the location of the limit switches and the SS-5GL13 component [5], giving a visual reference to their placement in the system.

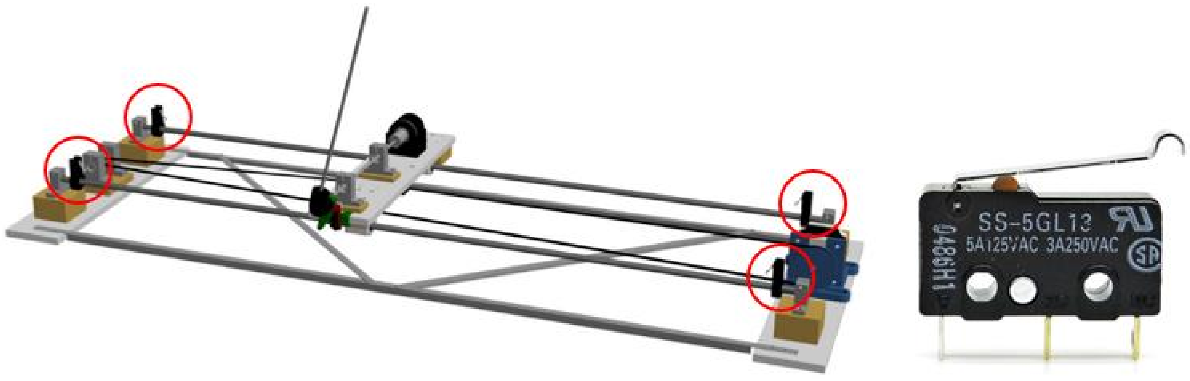


Figure 2.9: Limit switches

To ensure the safe and reliable reading of the limit switch output signal, the NC terminal of the switch is connected in series with an 18 kΩ pull-up resistor. This configuration allows the signal to be measured in the presence of a 3.3 VDC nominal supply voltage provided by the PYNQ board's voltage supply. By utilizing the pull-up resistor, the voltage levels of the signal are maintained, ensuring compatibility with the PYNQ board's input requirements. This setup guarantees the safety and proper functioning of the output signal, making it suitable for accurate reading and further processing by the PYNQ board.

The output signals from the terminals of the limit switches can exhibit extra bounces, particularly at the rising and falling edges. These bounces can introduce errors in the signal readings, impacting the overall system performance. Figure 2.10 illustrates the bounces generated by the limit switch specifically at the rising edge. To mitigate these bounces and ensure reliable signal detection, a specialized anti-bounce filter with a 1 millisecond delay is implemented in hardware, as described in Chapter 6. This filter effectively eliminates the undesired bounces and improves the accuracy and stability of the signal readings.

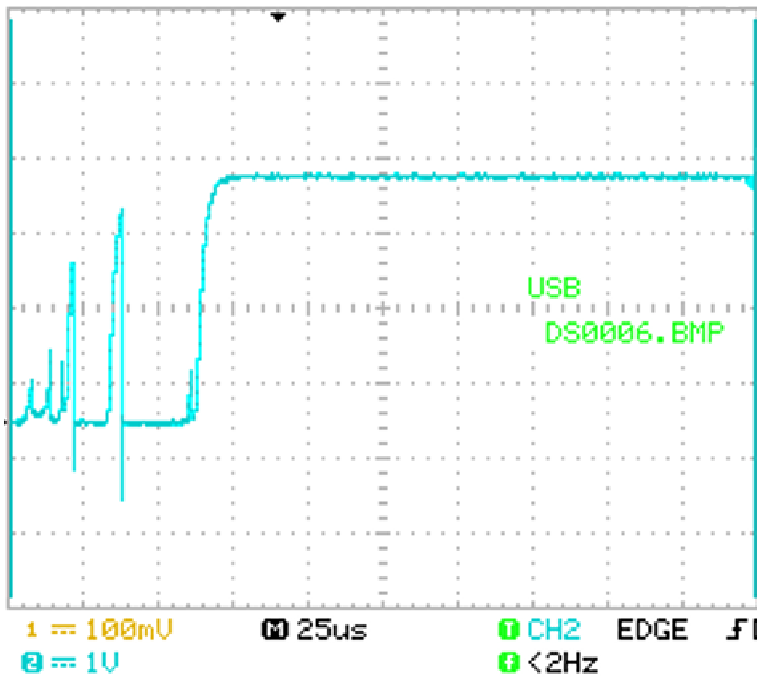


Figure 2.10: The output signal of limit switches containing extra bounces.

## 2.4 Stepper Motor

To ensure precise movement and accurate position tracking, a stepper motor was chosen for the system. The stepper motor is connected to the cart using a pulley belt system, as depicted in Figure 2.11. This configuration allows for controlled and precise motion of the cart, enabling precise positioning and smooth operation of the system.

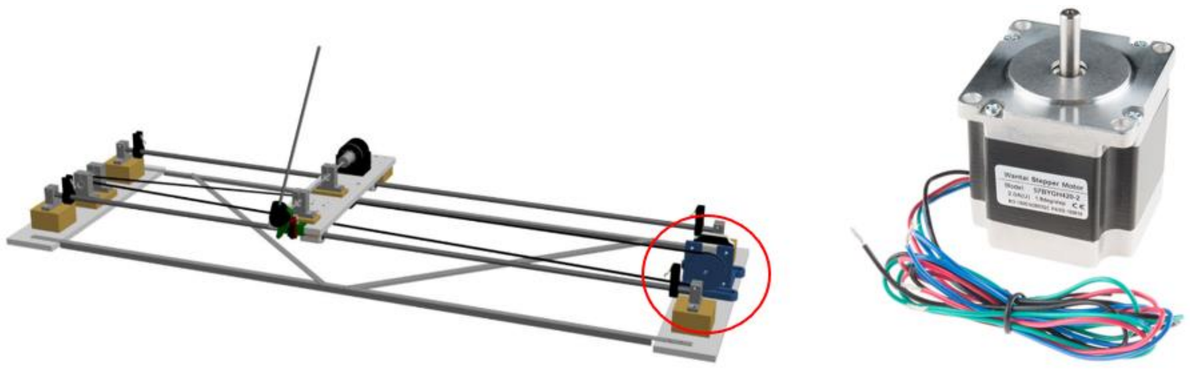


Figure 2.11: Stepper motor

To achieve continuous rotation, a stepper motor requires continuous alteration in the energization of its coils. This control is achieved using a stepper driver, which allows for precise control over the angle, angular position, and acceleration of the motor. It is important to note that the stepper driver controls the rotation frequency of the motor, which indirectly affects the torque provided by the motor.

During the design stage, it is crucial to consider the maximum torque produced by the motor and the mass of the cartpole. This consideration ensures that the system operates within safe limits and prevents any potential deterioration or instability. By properly accounting for these factors, the stepper motor can be effectively utilized to control the movement of the cartpole system, providing accurate positioning and control.

The selection of an appropriate motor is critical in meeting the system requirements, which include a maximum cart acceleration of 30 m/s² and a maximum rotation speed of 100 rpm. The maximum acceleration value was determined based on simulations and real experiments conducted during the project.

To estimate the maximum force and torque required from the motor, other factors such as friction acting on the cart are disregarded. By considering the masses of the cart and pendulum, as well as the pulley radius, precise calculations can be performed to determine the force $F_{required}$ and torque $T_{required}$ necessary to move the cartpole system accurately.

$$F_{required} = \left(M_{cart} + m_{pendulum}\right) * a = (0.455kg + 0.044kg) * 30\frac{m}{s^2} = 14.97\ N$$

$$T_{required} = F_{required} * r_{pulley} = 9.98N * 25.21mm = 377\ mNm$$

The Wantai 57BYGH420-2 stepper motor has been chosen for this application as it meets all the mentioned requirements. Detailed specifications of the motor can be found in Table 2.1.

| **Quantity** | **Value** | **Unit of measurement** |
|---|---|---|
| No. of Phases | 4 | **-** |
| Step Angle | 1.8 ±5% | **°/step** |
| Rated Voltage | 3.6 | **V** |
| Rated Current | 2.0 | **A/phase** |
| Phase Resistance | 1.5 ±10% | **Ω/ phase** |
| Phase Inductance | 2.5 ± 20% | **mH/phase** |
| Detent Torque | 0.039 | **mNm** |
| No. of Channels | 4 | **-** |
| Rotor inertia | 300 | **g*cm$^2$** |
| Holding torque | 882.6 | **mNm** |
| Mass | 0.7 | **Kg** |

Table 2.1: Wantai 57BYGH420-2 stepper motor specifications [6]

To ensure the motor operates correctly and remains undamaged, it is essential to regulate the power supply within the motor's rated voltage and current limits. Any excess voltage or current can cause permanent damage to the motor. Therefore, it is important to measure the output signals of the motor driver and configure it as necessary. The appropriate wire connection for the motor can be seen in Figure 2.12.

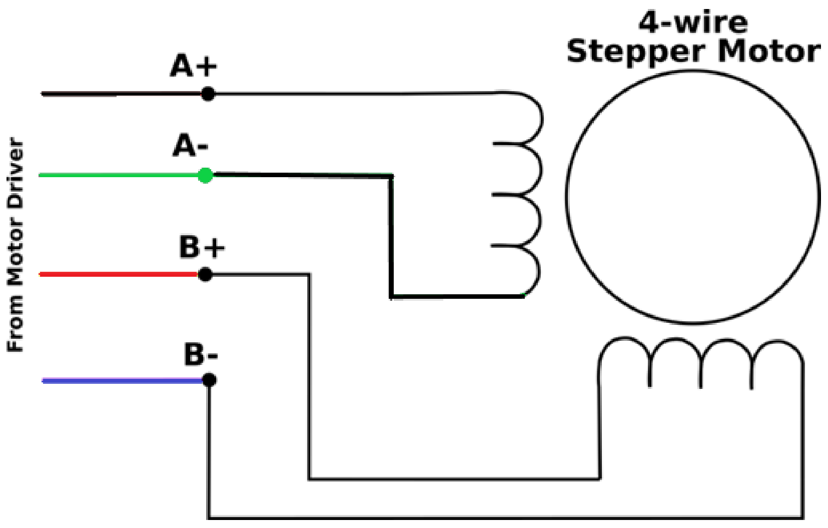


Figure 2.12: Stepper Motor connection

## 2.5 Stepper Motor Driver Carrier

Based on its desirable characteristics and functionality, the DRV8825 bipolar stepper motor driver board by Pololu was chosen for the system motor. This motor driver offers various features, including adjustable current limiting, over-temperature and over-current protection, and supports up to 6 microstepping modes, including 1/32 step resolution. To ensure safe operation, the driver board is equipped with a soldered potentiometer that allows for precise regulation of the current flowing to the motor, preventing potential damage. The operational voltage supply range for the motor driver carrier is between 8.2V and 45V, providing flexibility in terms of power requirements [7]. For a detailed understanding of the pin configuration and naming conventions, refer to Figure 2.13.

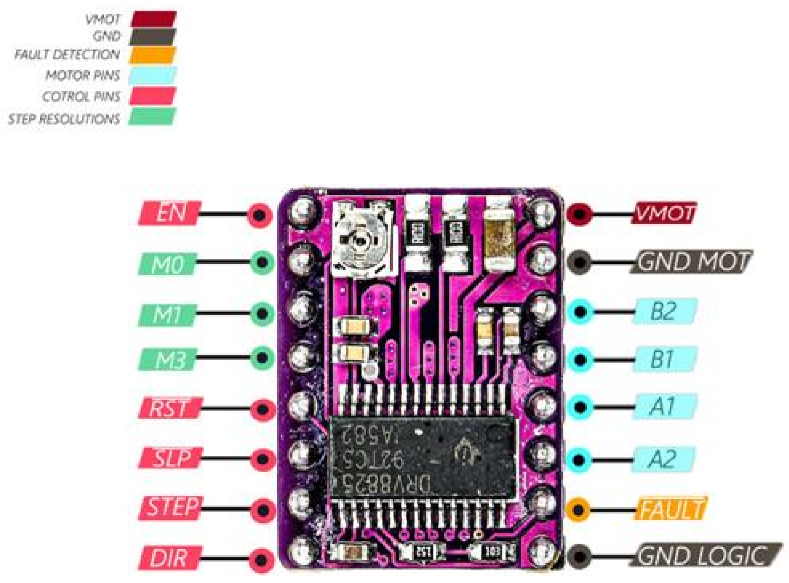


Figure 2.13: Stepper motor driver DRV8825 carrier by Pololu [7].

### 2.5.1 Control Input

The Pololu driver carrier simplifies the interface with its 16 pins, compared to the 32 pins of the DRV8825 by Texas Instruments. This design choice enhances user convenience when connecting and operating the motor driver. Detailed information about the functions and internal structure of each logic input pin is provided in the comprehensive table below. It is important to consider the presence of internal pulldown or pullup structures for certain pins during the hardware design phase on the FPGA board.

To initiate motor startup, a minimum wiring configuration requires connecting signals to the nRESET, nSLEEP, STEP, and DIR pins, which are described in detail in Table 2.2. It should be noted that input pins are considered "high" when their voltage falls within the range of 2.2V to 5.25V.

| Name | Descipríon |
|---|---|
| **nENBL** | Logic high to disable device outputs and indexer operation, logic low to enable. Internal pulldown. |
| **MODE0** | Micro stepping mode: full step, 1/2, 1/4, 1/8, 1/16, 1/32. Internal pulldown. |
| **MODE1** | |
| **MODE2** | |
| **nRESET** | High level to exit “reset” state and low level to “reset” “Pulldown” internal |
| **nSLEEP** | High level to exit from low consuming mode and low level to enter low consuming mode. Internal pulldown. |
| **STEP** | Rising edge to advance one micro step. Internal pulldown. |
| **DIR** | Level sets the direction of stepping. Internal pulldown. |

Table 2.2 Motor Driver control inputs description [8]

When establishing the connection between the microcontroller (PYNQ board) and the motor driver carrier, it is crucial to ensure that the output signals from the PYNQ board pins are configured to 3.3V. This voltage setting guarantees accurate signal transmission, prevents signal distortion, and minimizes the impact of noise interference.

### 2.5.2 Microstepping Modes

The DRV8825 stepper motor driver offers 6 microstepping modes to adjust the motor resolution according to the specific application requirements (refer to Table 2.3). In the full step mode, the driver consumes less power. However, the movement of the cart is characterized by abrupt steps and moderate vibrations, resulting in a lower resolution.

On the other hand, the 1/32 step mode provides smoother motor movement, higher resolution, and reduced vibrations during operation. However, this mode requires a higher power consumption compared to the full step mode.

In the case of the inverted pendulum model, maintaining a high resolution of the cart displacement is crucial since the system is inherently unstable in the upright position. Therefore, pendulum balancing is implemented using the 1/32 microstepping mode to achieve precise control and stability.

Regarding the step frequency, the DRV8825 driver has a minimum limit of 2 microseconds in the full step mode. However, in the 1/32 microstepping mode, the step frequency is thirty-two times smaller than in the full step mode, resulting in a value of 62.5 nanoseconds. This finer step resolution allows for more precise movements and control of the stepper motor.

| MODE2 | MODE1 | MODE0 | STEP MODE |
|---|---|---|---|
| 0 | 0 | 0 | Full step (2 -phase excitation) with 71 % current |
| 0 | 0 | 1 | Half step (1-2 phase excitation) |
| 0 | 1 | 0 | 1/4 step |
| 0 | 1 | 1 | 1/8 step |
| 1 | 0 | 0 | 1/16 step |
| 1 | 0 | 1 | 1/32 step |
| 1 | 1 | 0 | 1/32 step |
| 1 | 1 | 1 | 1/32 step |

Table 2.3 Motor Driver step modes

### 2.5.3 Power Dissipation and Current Limiting

The DRV8825 stepper motor driver board has a maximum voltage rating of 2.5 V, but due to the internal sense resistors, it is limited to 2.2 V. To supply more current to the motor and achieve higher torque, it is essential to keep the driver temperature low. The Pololu driver carrier can handle current demands below 1.5 A without additional cooling, but for currents above 1.5 A per coil, an additional cooling system should be considered. In the case of the stepper motor used in the inverted pendulum model, which has a current rating of 2 A, a heat sink was attached to the DRV8825 driver board. This allows for faster heat exchange and prevents permanent damage to the component [7]. Figure 2.14 illustrates the attachment of the heat sink.

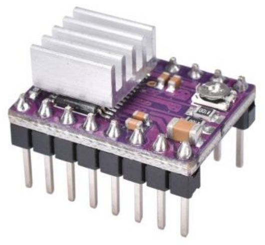


Figure 2.14: The heat sink attached to the motor driver.

One of the most important considerations when working with a stepper motor driver is to properly adjust the current limit. Supplying a current higher than the motor's rated value can result in motor damage. To ensure the current limit is set correctly, it is necessary to follow a minimal wiring configuration, as depicted in Figure 2.15. It is crucial not to connect the drive output until the current limit is appropriately adjusted.

In order to safeguard the driver from motor supply spikes, it is recommended to connect a 100-microfarad electrolytic capacitor to the motor supply pins of the driver carrier. Once the wiring is completed, step and direction signals are generated by the microcontroller, in this case, the PYNQ board. The driver is then connected to the DC power supply, while the ground of the microcontroller is connected to the voltage between the microcontroller ground and the surface of the potentiometer, as indicated in Figure 2.15.

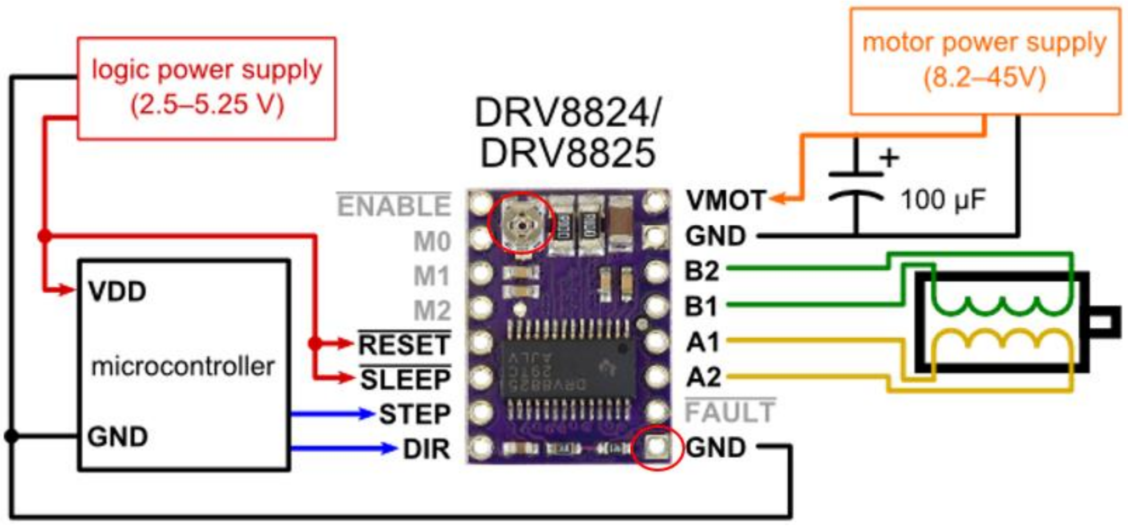


Figure 2.15: Minimal wiring for the current limiting adjustment and connection to the motor [7].

To adjust the current limit, it is necessary to measure the reference voltage using a multimeter. The potentiometer is then rotated clockwise to increase the voltage reference value or counterclockwise to decrease it. Each driver model has its own specific voltage reference calculation formula. For the DRV8825 driver, the manufacturer provides a formula that should be utilized to adjust the current limiting accurately.

$$Current\ limit[A] = V_{ref}[V] \cdot 2$$

Therefore, according to the driver voltage reference calculation formula, the reference voltage should be set to half of the rated current of the stepper motor.

$$V_{ref}[V] = \frac{Current\ limit[A]}{2}$$

In the case of the motor used in this work, which has a rated current of 2A, the reference voltage was adjusted to 0.94V, slightly below the reference value of 1V. This adjustment ensures that the current supplied to the motor remains within the specified limits, preventing potential damage and ensuring proper motor operation.

## 2.6 PYNQ – Z1 Prototyping Board

The control system for the inverted pendulum utilizes the Digilent PYNQ-Z1 prototyping board, which features the ZYNQ XC7Z020-1CLG400C FPGA device from Xilinx. This device belongs to the Zynq-7020 family and follows the “All Programmable Systems on Chips” (APSoC) architecture. It consists of a programmable logic (PL) and a processor system (PS) with a 650MHz double-core Cortex-A9 processor.

For a visual representation of the PYNQ-Z1 board and its overall layout, refer to Figure 2.16.

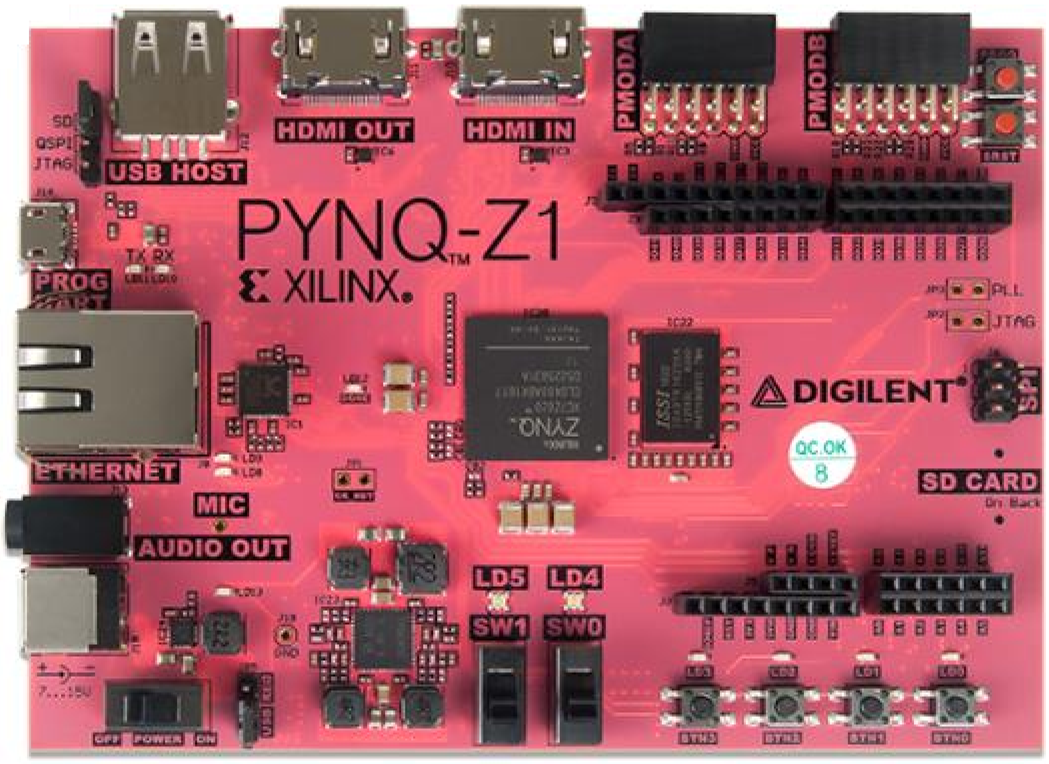


Figure 2.16: Digilent PYNQ- Z1 prototyping board

The PL and PS components of the Zynq device enable a powerful combination of configurable logic circuits and software programs. The communication between the PL and PS is established using the AXI protocol (“Advanced eXtensible Interface”) [9]. This communication allows for the design of systems that leverage the flexibility of the PL for hardware acceleration and the processing capabilities of the PS for executing software programs.

The PYNQ-Z1 board also includes a Texas Instruments TPS65400 Phasor Measurement Unit, which provides multiple supply voltage options such as 3.3V, 1.8V, 1.5V, and 1.0V from the main power input. The power input for the PYNQ board is derived from a USB port. The power

switch SW4 enables or disables all on-board power supplies, while the power indicator LED (LD13) indicates when all the supply rails reach their nominal voltage [10]. The optical interrupter, rotary encoder, and limit switches are supplied with 3.3V from the PYNQ board.
The control system design utilizes the Vivado IP integrator tool [11] to build a processor-based design, combining the capabilities of the PL and PS. The embedded software for the system is programmed and debugged using the Xilinx Vitis unified software platform. Custom IPs, developed in VHDL ("Very high-speed integrated circuits Hardware Description Language") using Xilinx Vivado, act as peripherals that extend the processing capabilities of the Zynq Processing System with custom hardware logic. This enables the acceleration of application-specific algorithms.

The processing system (PS) is programmed in the C language using the Xilinx Vitis toolkit. By leveraging both custom IP and embedded software, the control system architecture achieves a flexible and efficient design for the inverted pendulum application.

# Chapter 3

## 3. Mathematical Modeling and Simulink Simulation of the Cartpole System

In order to efficiently design a control strategy for the inverted cartpole system, it is crucial to have a mathematical model that accurately represents the system's dynamics, including the motion equations of the cart and pendulum positions. The mathematical model should be designed considering the physical model's limitations and constraints.

There are two main approaches to obtain the motion equations of the system: the Newtonian approach and the Lagrangian approach. The Newtonian approach involves applying Newton's laws of motion and considering the forces acting on the system. However, in combined systems like the cartpole, deriving the final equation requires accounting for the interaction between the two components [12, p.17].

On the other hand, the Lagrangian approach is more flexible for systems with multiple components, as it is based on the energy of each component. The motion equations can be derived using the Euler-Lagrange equations. Additionally, the Lagrangian approach simplifies the derivation process by working with scalar quantities, unlike the vector operations required in the Newtonian approach [12, p.17].

Once the mathematical model is obtained, it can be implemented in Simulink to simulate and test various control strategies. This allows for the evaluation and validation of the control strategies effectiveness before applying them to the real physical model. By conducting simulations in Simulink, potential malfunctions and risks associated with incorrect control can be identified and addressed, thus safeguarding the physical system.

Simulink offers several advantages in this context, including time-saving benefits. Simulations can be performed much faster compared to real-time testing, enabling rapid iterations and refinement of control strategies. Additionally, using Simulink for testing purposes can help save energy since the physical system is not actively operated during simulation.

## 3.1 Euler-Lagrange Equation

To obtain the motion equations of the cartpole system, the Lagrangian technique can be applied. For a system consisting of rigid bodies that can be considered as particles, the Lagrangian function is defined as the difference between the system's kinetic energy $T$ and potential energy $V$ [12 p.19-20]. That is,

$$L = T - V.$$

In generalized coordinates, the position is denoted as $q$, while the velocity is denoted as $\dot{q}$. For a multi-particle system, the Lagrangian function can be expressed as follows:

$$L = L(q_i, \dot{q}_i, t), \qquad i = 1, \dots, n.$$

The Euler-Lagrange equation is then derived from this Lagrangian function:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0$$

However, in a more general case where non-conservative forces, such as damping force, Coulomb friction, or other dissipative forces, are present, a more generalized equation can be obtained:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$

Here, $Q_i$ represents the generalized term for the non-conservative forces acting on the system.

By incorporating non-conservative forces within the Lagrangian framework, it becomes feasible to analyze systems subjected to dissipative forces or external perturbations. This approach facilitates a comprehensive investigation of various physical systems, enabling the consideration of both conservative and non-conservative forces in a unified manner. However, in the specific case of our system, we choose to simplify the problem by neglecting non-conservative forces such as the drag force acting on the pendulum and the force of kinetic friction acting on the cart.

## 3.2 Modelling the Cartpole System

The motion equations for the cartpole system are derived using the Euler-Lagrange equation, neglecting all non-conservative forces acting on the system, as mentioned before.

When modeling the cartpole system, the first consideration should be the control input of the system. In the current implementation of the cartpole system, a stepper motor is used, which is able to provide the cart with velocity control of high precision. Thus, it is convenient to consider the acceleration of the cart as the control input.

However, both the stepper motor and the stepper motor driver have limitations in generating torque at high speeds. As a result, the control input, which is the cart acceleration, will be bounded within a range where the torque provided by the motor would be sufficient to achieve the desired acceleration. This constraint applies independently of any forces from the moving pendulum acting on the cart, kinetic friction, or any other dissipative forces acting on the system [13].

Figure 3.1 illustrates the cartpole system, showcasing its control input, acceleration $u$, and the relevant process variables: the pendulum's angle and the cart's position $x$. In the figure, point $O$ designates the origin of the system coordinates, and it is important to note that the hinge point of the pendulum always lies on the x-axis. The cart is assigned a mass of $M$, while the pendulum has a mass of $m$.

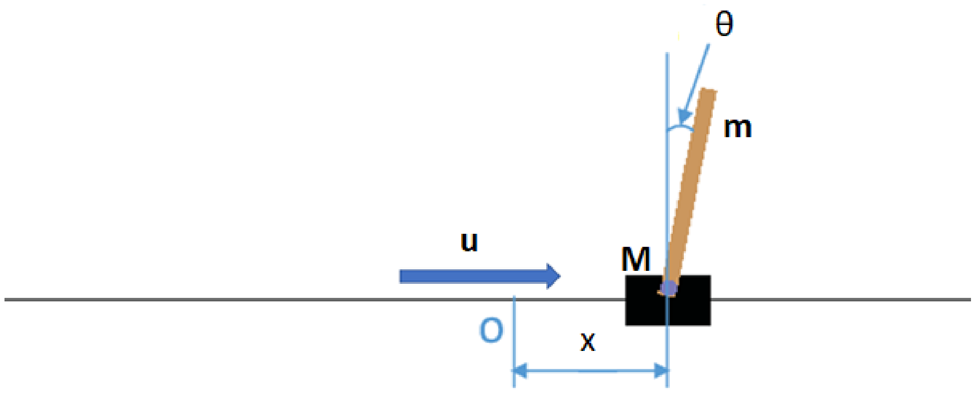


Figure 3.1: The Cartpole Control System

Therefore, the differential equation of the cart position can be expressed in terms of the control input as following:

$$\ddot{x} = u. \quad (3.1)$$

To derive the motion equation for the pendulum angle, the Euler-Lagrange equation is utilized.

For the cart, its kinetic energy is solely contributed by the translational term, given by:

$$T_{cart} = \frac{1}{2}M\dot{x}^2.$$

While the pendulum has both rotational and translational kinetic energy, the cart only contributes to translational kinetic energy. Denoting $l$ as the distance from the hinge point to the center of gravity of the pendulum, and $I$ as the moment of inertia of the pendulum around its center of gravity, considering the pendulum's position about the origin $O$ as $(x + lsin\theta,$ $lcos\theta)$, the kinetic energy equation is:

$$T_{pendulum} = \frac{1}{2}m{\dot{x}_p}^2 + \frac{1}{2}I\dot{\theta}^2$$

$$= \frac{1}{2}m\left((\dot{x} + l\dot{\theta}cos\theta)^2 + (-l\dot{\theta}sin\theta)^2\right) + \frac{1}{2}I\dot{\theta}^2$$

$$= \frac{1}{2}m\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 + ml\dot{x}\dot{\theta}cos\theta + \frac{1}{2}I\dot{\theta}^2.$$

Therefore, the kinetic energy of the system is the sum of the kinetic energies of the pendulum and the cart:

$$T = T_{cart} + T_{pendulum} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 + ml\dot{x}\dot{\theta}cos\theta + \frac{1}{2}I\dot{\theta}^2.$$

Similarly, the potential energy of the system is the sum of the potential energies of the cart (which is zero) and the pendulum (which is gravitational potential energy):

$$V = V_{cart} + V_{pendulum} = 0 + mglcos\theta = mglcos\theta.$$

The Lagrangian function for the cartpole system is indeed the difference between the kinetic energy $T$ and potential energy $V$ of the system:

$$L = T - V = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 + ml\dot{x}\dot{\theta}cos\theta + \frac{1}{2}I\dot{\theta}^2 - mglcos\theta.$$

Using the Euler-Lagrange equation, the motion equation for the pendulum position $\theta$ can be derived:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = (ml^2 + I)\ddot{\theta} + ml\ddot{x}cos\theta - mglsin\theta = 0.$$

Further simplification gives:

$$\ddot{\theta} = \frac{ml}{ml^2 + I}(gsin\theta - \ddot{x}cos\theta) = \frac{ml}{ml^2 + I}(gsin\theta - ucos\theta). \quad (3.2)$$

From this differential equation of the pendulum position, it can be observed that the system exhibits high nonlinearity. In order to apply conventional control strategies such as PID and LQR, which are designed for linear systems, it is necessary to linearize the system. Linearization simplifies the control design process by approximating the system as linear around an operating point, enabling the use of linear control techniques.

## 3.3 Simulink model of the Cartpole System

To construct a reliable simulation model of the cartpole control system, it is essential to measure or define all the relevant system parameters. Table 3.1 provides an overview of the important system parameters. Notably, the table reveals that the center of gravity of the pendulum does not align with the midpoint of the pendulum. This discrepancy was estimated empirically by swinging the pendulum at various points until it reached an equilibrium position.

Furthermore, the maximum speed of the cart can be calculated by considering the maximum rotational period per step $f_{\max}$ of the stepper motor and its maximum step angle $\alpha_{max}$, along with the pulley radius $r_{\text{pulley}}$. By utilizing these parameters, we can determine the maximum achievable speed of the cart.

The calculation for the maximum cart speed $v_{max}$ is as follows:

$$v_{max} = \frac{\pi \cdot \alpha_{max}}{180°} \cdot f_{\max} \cdot r_{\text{pulley}} = 0.01 * \pi * 1000 * 0.02521 = 0.792\text{m/s}.$$

The moment of inertia of the pendulum $I$ about its center of gravity needs to be approximated since its structure is complicated to estimate precisely. In the model, we consider the pendulum as an ideal thin rod with a small diameter and similar point masses at both ends. By assuming this simplified configuration, we can calculate the moment of inertia of the pendulum using the following formula:

$$I = \frac{1}{12} ML^2 = 0.0047\ kgm^2$$

| Symbol | Description | Value |
|---|---|---|
| $\boldsymbol{M}$ | Cart mass | 0.455 $kg$ |
| $\boldsymbol{m}$ | Pendulum mass | 0.044 $kg$ |
| $\boldsymbol{L}$ | Pendulum length | 0.353 $m$ |
| $\boldsymbol{l}$ | Distance from the hinge point to the center of gravity of the pendulum | 0.141 $m$ |
| $\boldsymbol{g}$ | Gravitational acceleration | 9.81 $m/s^2$ |
| $\boldsymbol{I}$ | Pendulum moment of inertia about the center of gravity | $0.0047\ kgm^2$ |
| $\boldsymbol{v_{max}}$ | Maximum cart speed | 0.792 $m/s$ |
| $\boldsymbol{x_{min}, x_{max}}$ | Cart position limits | - 0.48 $m$; 0.48 $m$ |
| $\boldsymbol{u_{max}}$ | Cart maximum acceleration absolute value | 20 $m/s^2$ |

Table 3.1: Cartpole control system parameters

The Cartpole simulation model is developed within the Simulink environment, considering the derived motion equations (3.1) and (3.2) without linearization. The equations are formulated in the continuous-time domain to simulate the dynamics of the real non-linear inverted pendulum control system accurately.

In the cartpole system simulation block, the positions of the pendulum and cart are outputs, representing the measurements available in the actual system. The control input for the system is the acceleration of the cart, as mentioned previously.

The implementation of the cartpole model in Simulink can be visualized in Figure 3.2. This Simulink model allows for the simulation and analysis of the cartpole system, providing insights into its behavior and enabling the development and testing of control strategies for stabilization and trajectory tracking.

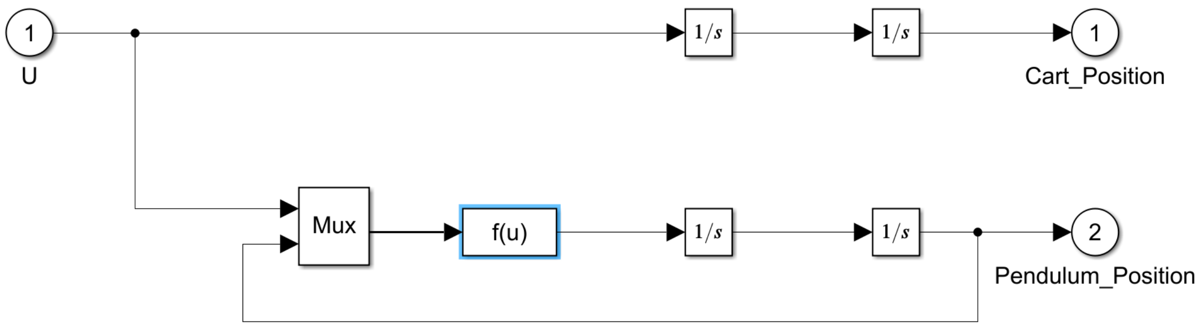
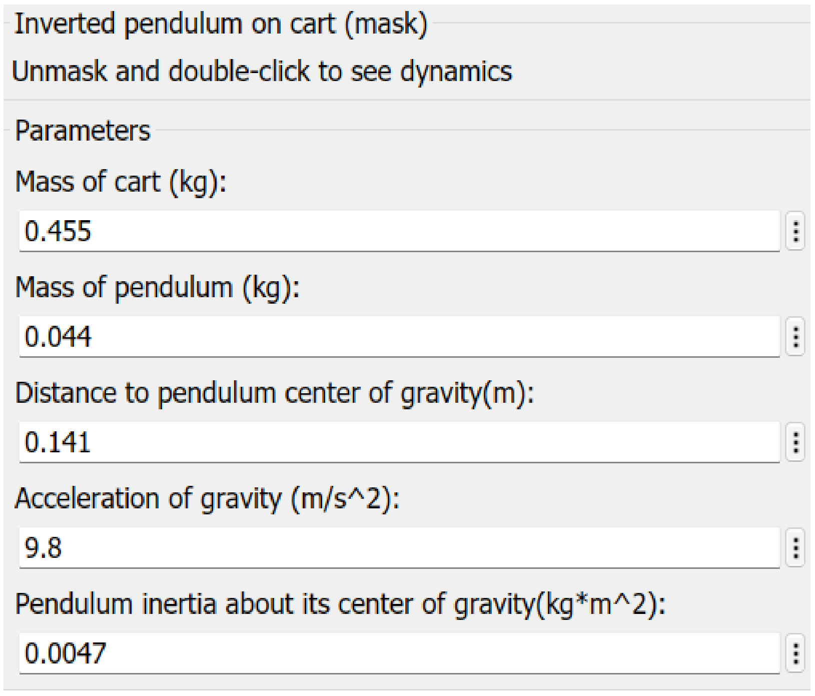


Figure 3.2: The Cartpole model in Simulink

The system parameters can be easily modified, if needed, using the Block mask feature in Simulink. In Figure 3.3, the Cartpole Block mask is depicted, showcasing how system parameters can be introduced or modified.



Inverted pendulum on cart (mask)

Unmask and double-click to see dynamics

Parameters

Mass of cart (kg): 0.455

Mass of pendulum (kg): 0.044

Distance to pendulum center of gravity(m): 0.141

Acceleration of gravity (m/s^2): 9.8

Pendulum inertia about its center of gravity(kg*m^2): 0.0047

Figure 3.3: Cartpole Block parameters

To visualize the behavior of the pendulum during the control strategy application, an animation block from the "Inverted Pendulum with Animation" example by MathWorks [14] is used. This animation block is created using MATLAB® Handle Graphics® and provides a visual representation of the pendulum's motion.

In Figure 3.4, the animation block is shown, which includes a playback function and the ability to manually select the art position reference using left and right arrows. The trajectory length of the cart is adjusted to match the dimensions of the physical model, set to 0.96 meters.

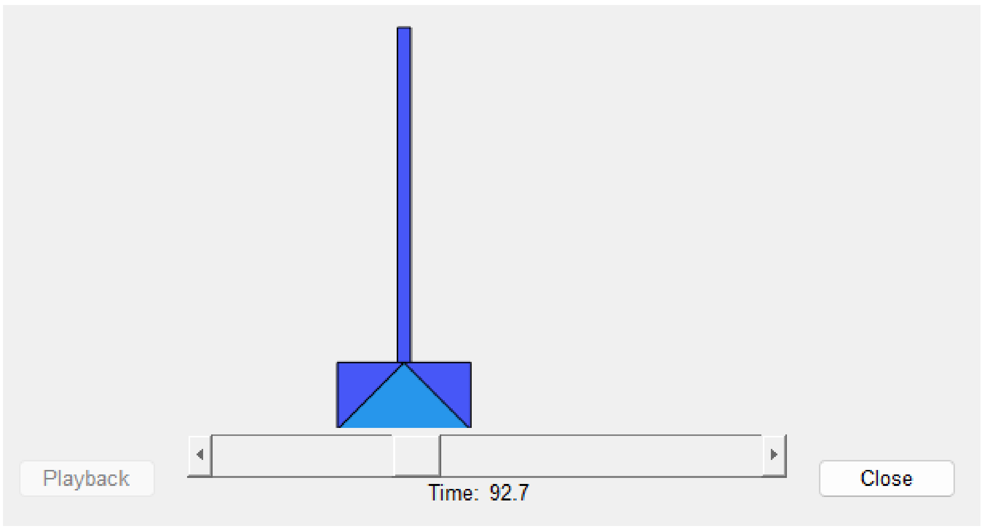


Figure 3.4: The system animation block

By incorporating the animation block into the system, the user can observe the real-time movement of the pendulum during the control strategy application. This visualization enhances understanding and analysis of the system's dynamics and performance.

# Chapter 4

## 4. Control Strategies for the Cartpole System

This chapter focuses on the design and implementation of two control strategies, the Linear Quadratic Regulator (LQR) controller and the Proportional-Integral-Derivative (PID) controller, for the cartpole system. The nonlinear model of the system is linearized to enable the application of the LQR strategy, resulting in a Linear Time-Invariant (LTI) model of the cartpole system.

The PID controller is a feedback controller that tracks the error between the desired setpoint and the measured process variable. It applies control action based on proportional, integral, and derivative terms. The PID controller parameters are tuned using Ziegler-Nichols' oscillation tuning method, which involves step-by-step selection of the three PID controller coefficients based on specific criteria. The tuning process aims to stabilize the system through experimentation.

The LQR controller utilizes feedback gain to minimize errors and achieve the desired system state. It is an optimal control method that does not require extensive tuning procedures like the PID controller. The feedback gain is obtained by solving the Algebraic Riccati Equation.

Both the PID and LQR controllers are implemented in a digital format to match the control system's processor, and they operate with a control period of 25 milliseconds, corresponding to a frequency of 40 KHz. The value for control period was selected empirically both by simulation and application on the real system. The control period ensures the cart velocity is updated appropriately to avoid aliasing effects while maintaining effective control.

To determine the feedback gain of the LQR controller, the cartpole system is linearized around the upright position, which is the point that requires control. Additionally, a digitized linear time-invariant model of the inverted pendulum is obtained.

This chapter presents simulation results for both the LQR and PID control strategies. The results provide insights into the performance, stability and response time of each controller. The simulations allow for an evaluation of the effectiveness of the control strategies for the cartpole system.

## 4.1 State-space representation and Linearization of LTI model

The state vector $\mathbf{x}$ for the inverted pendulum control system consists of four terms: the cart position, cart velocity, pendulum angular position, and pendulum angular velocity. It can be represented as:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Therefore, the nonlinear model equations of the inverted pendulum can be written in matrix form using the state vector as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ u \\ \dot{\theta} \\ \frac{ml}{ml^2 + I}(gsin\theta - ucos\theta) \end{bmatrix} = \begin{bmatrix} x_2 \\ u \\ x_4 \\ \frac{ml}{ml^2 + I}(gsinx_3 - ucosx_3) \end{bmatrix}$$

To obtain the Linear Time-Invariant (LTI) representation of the cartpole system, the model needs to be linearized at a specific point. In this case, we linearize the model around $\theta = 0$, as the upright position is the point at which it should be maintained. The linearized model equation is as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ u \\ x_4 \\ \frac{ml}{ml^2 + I}(\ gx_3 - u) \end{bmatrix}$$

To represent the system in the LTI space, where x(t) is the input variables matrix and y(t) is the output variables matrix, the following equations are obtained:

$$\dot{\mathbf{x}}(t) = \mathbf{Ax}(t) + \mathbf{Bu}(t)$$

$$\mathbf{y}(t) = \mathbf{Cx}(t) + \mathbf{Du}(t)$$

The output variables matrix y(t) includes the cart position and pendulum angle, and the LTI equations are:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{mgl}{ml^2 + I} & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\frac{ml}{ml^2 + I} \end{bmatrix} \mathbf{u}(t)$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \mathbf{u}(t)$$

These equations represent the relationship between the system's state variables (cart position, cart velocity, pendulum angle, and pendulum angular velocity) and the output variables (cart position and pendulum angle) in the LTI representation.

## 4.2 System Discretization and quantization error

Once the LTI cartpole model is obtained for the pendulum, it can be discretized assuming a specific sampling time. The linearized LTI model in MATLAB can be passed to the *"c2d"* function, which applies the discretization process. In this case, the zero-order hold method is used with a discretization period of 25 milliseconds. This process converts the continuous-time LTI model into a discrete-time model, which can be implemented in a digital control system.

Therefore, the control input, which is the acceleration, is updated and generated at each control period, which is set to 25 milliseconds. However, it's important to note that due to the discretization process, there may be errors introduced. This is because the acceleration is obtained by incrementing the velocity every 100 microseconds, resulting in 2500 iterations during one control period. It's worth mentioning that a constant acceleration would result in a linear increase in velocity, but the discretization process introduces a step-like behavior. These discretization errors should be considered when designing the control system to ensure accurate and effective control of the cartpole system.

Similarly, the pendulum angular position, which is measured by the encoder, will also be subject to quantization errors. The encoder implemented in the model has 600 counts per revolution, resulting in a resolution of 0.01 radians for the pendulum angle measurement. While there may be some quantization errors, they are expected to be small and not significantly affect the overall performance of the control system.

Figure 4.1 illustrates the quantization error in the angular position of the pendulum.

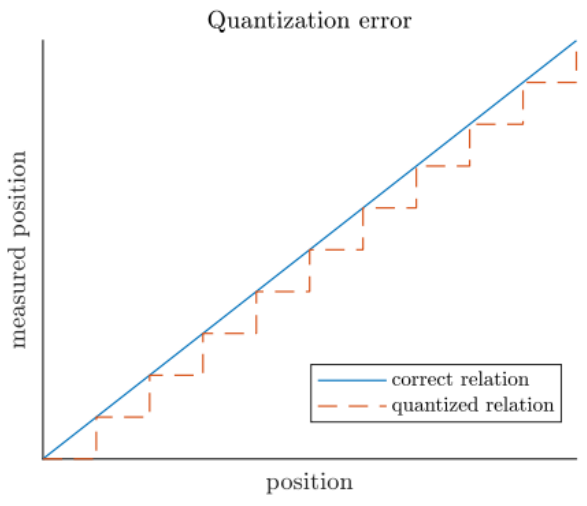


Figure 4.1: Angular position quantization error [13 p.52]

## 4.3 LQR Controller Design

The LQR controller is a type of state-feedback controller. It uses feedback from the system's states to compute the control input that minimizes the cost function. By measuring the current states of the system, the LQR controller calculates the appropriate control action u to be applied to drive the system towards the desired state or trajectory.

The LQR controller aims to minimize the error between the actual system states and the desired states. This error is typically represented as the difference between the current state vector and the desired state vector. By continuously measuring and comparing the states, the LQR controller adjusts the control input to minimize this error and regulate the system's behavior.

The control action u in the LQR controller is given by:

$$u = -\mathbf{K}(\mathbf{x} - \mathbf{x}_{\boldsymbol{desired}})$$

where $u$ is the control input, $\mathbf{K}$ is the feedback gain matrix, $\mathbf{x}$ is the current state vector, and $\mathbf{x}_{\boldsymbol{desired}}$ is the desired state vector. The feedback gain matrix $\mathbf{K}$ determines the contribution of each state variable to the control input. By multiplying the difference between the current state vector and the desired state vector by the feedback gain matrix, the control action is computed. The LQR control strategy is effective in handling both steady-state and transient errors. It provides a systematic approach to achieve stability and optimal performance by actively monitoring and adjusting the system's states.

The feedback gain K in the discrete-time LQR control is calculated by solving the Algebraic Riccati equation. Since the cartpole control model is discrete-time, the Infinite Horizon Discrete-Time LQR control was selected for this system [15].

The discretized LTI cartpole model can be represented in state-space form as follows:

$$\mathbf{x}(k+1) = A_d\mathbf{x}(\mathrm{k}) + B_d\boldsymbol{u}(k) \qquad (4.1)$$

The optimal solution $\boldsymbol{u}^*(\boldsymbol{k})$ for the infinite horizon linear quadratic optimal control problem is given by:

$$\boldsymbol{u}^*(\boldsymbol{k}) = -\boldsymbol{K}\mathbf{x}(k) = -\left(R + B_d^T P B_d\right)^{-1} B_d^T P A_d \boldsymbol{x}(k)$$

In the above equation, P is a positive symmetric matrix, and it is the solution of the Discrete Algebraic Riccati Equation:

$$P = A_d^T P A_d + Q - {A_d}^T P B_d\left(R + B_d^T P B_d\right)^{-1} B^T P A_d$$

Here, $Q$ and $R$ are positive semidefinite weighting matrices that determine the trade-off between control effort and state deviations. The solution of the Algebraic Riccati Equation yields the optimal feedback gain matrix $K$, which is used to compute the control action $\boldsymbol{u}^*(\boldsymbol{k})$ based on the current state $\mathbf{x}(k)$ [15].

In MATLAB environment, the feedback gain matrix K can be calculated using the *"dlqr"* function. The *"dlqr"* function computes the optimal gain matrix K for the discrete-time linear quadratic regulator (LQR) problem. It takes as inputs the system matrices $A_d$ and $B_d$, and the weighting matrices $Q$ and $R$. The function call to calculate $K$ is as follows:

$$K = dlqr(A_d, B_d, Q, R)$$

### 4.3.1 Augmented LQR Control

The LQR controller implemented in the Simulink simulation demonstrated satisfactory results in terms of correctly tracking the set points for both the pendulum position and the cart position. However, in the real-world implementation, there was an observed offset in the cart position relative to its set point, despite the pendulum reaching an upright position. This offset can be attributed to uncertainties present in the system matrices $A_d$ and $B_d$, which are part of the state space representation of the cartpole control system as described in equation (4.1).

Additionally, step disturbances can contribute to the offset in the cart position by affecting either the control action (acceleration) or the system states (cart and pendulum positions and velocities). These disturbances introduce additional factors that can impact the overall performance of the system and contribute to tracking errors.

To mitigate these set point tracking errors and enhance the performance of the controller, integral action can be incorporated into the LQR control. This is achieved by augmenting the existing system state with an additional discrete-time integral of the cart position tracking state. By incorporating integral action, the controller can effectively address steady-state errors and eliminate the offset in the cart position by continuously integrating and compensating for the tracking errors [3.1 p.69-70]. The augmented state vector, $\boldsymbol{x_{aug}}(k)$, is defined as follows:

$$\mathbf{x}_{\boldsymbol{aug}}(k) = \begin{bmatrix} \mathbf{x}(k) \\ q(k) \end{bmatrix} \quad (4.2)$$

Here, $\mathbf{x}(k)$ represents the existing four states, and $q(k)$ is the discrete-time integral of the cart position error. The integral term, $q(k+1)$, is updated recursively according to the following equation:

$$q(k+1) = q(k) + (x_{cart\_ref}(k) - x_{cart}(k)) \quad (4.3)$$

Considering equations (4.1), (4.2), and (4.3), the state space representation for the augmented system can be expressed as

$$\mathbf{x}_{\boldsymbol{aug}}(k+1) = A_{aug\_d}\mathbf{x}_{\boldsymbol{aug}}(k) + B_{aug\_d}\boldsymbol{u}(k) = \begin{bmatrix} A_d & 0_{4\times1} \\ -1 & 0_{1\times4} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ q(k) \end{bmatrix} + \begin{bmatrix} B_d \\ 0 \end{bmatrix} \boldsymbol{u}(k)$$

In this representation, $A_{aug\_d}$ is the augmented system matrix that combines the original system matrix $A_d$ with additional rows and columns to account for the integral term. $B_{aug\_d}$ is the augmented input matrix, and $\boldsymbol{u}(k)$ represents the control input. By augmenting the state vector and updating the system matrices, the LQR control system with integral action can effectively address steady-state errors, eliminate the cart position offset, and enhance the overall performance of the control system.

### 4.3.2 LQR Controller Tuning

Once the state space representation is defined, the feedback gain K of the control input $\boldsymbol{u}(k)$ is calculated in the same way as was described previously is calculated in MATLAB environment using *"dlqr"* function but with augmented states space matrixes $A_d$ and $B_d$ and the cost function matrices $Q$ and $R$.

The tuning procedure for the $Q$ and $R$ matrices of the LQR involves selecting appropriate weighting factors that determine the trade-off between control effort and state deviations. The $Q$ matrix represents the weights assigned to the state variables, determining their importance in the control objective.

In the case of the cartpole system, a higher weight of 100 is assigned to the pendulum angular position in order to prioritize its control. Conversely, the cart position is assigned a smaller weight of 4. The cart velocity and pendulum variables have even smaller weights compared to the positions. The integral error tracking state is assigned the smallest weight to minimize steady-state errors. This choice is made to balance the response speed with the stability of the pendulum, as larger weights destabilized its equilibrium.

Thus, the following $Q$ matrix was selected to control the cartpole:

$$Q = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The $R$ matrix represents the weight assigned to the control effort or input. Increasing the values in the $R$ matrix penalizes larger control efforts if excessive control action is undesirable. On the other hand, decreasing the values in the $R$ matrix allows for more aggressive control action if a faster response is desired. In this case, $R$ is selected to be 0.005.

With the chosen $Q$ and $R$ matrices, the feedback gain matrix $K$ for the system is obtained as:

$$K = [\ -10.6101 \quad -11.4893 \quad -64.5583 \quad -10.0341 \quad 3.1604\ ]$$

### 4.3.3 Simulink model of the cartpole control model with LQR controller

The Cartpole control model with augmented LQR controller is implemented in Simulink to test the response of the model to the applied control input. This allows for preliminary controller tuning before applying it to the real physical model. Figure 4.2 shows the Simulink model, which includes the non-linear cartpole model with animation described in detail in Chapter 3.

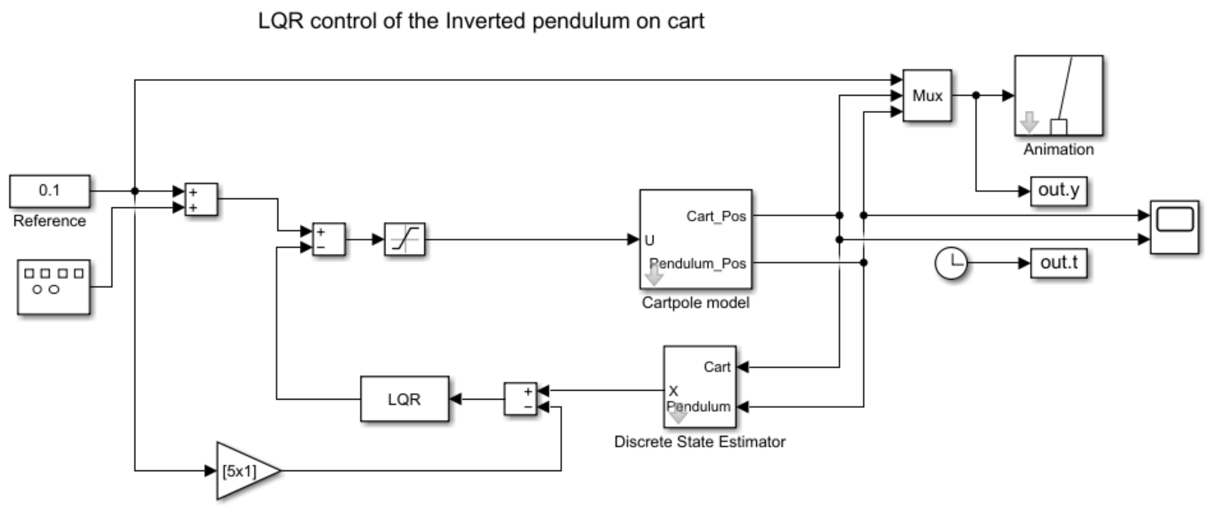


Figure 4.2: Simulink model of the LQR control system

The outputs of the cartpole model block, namely the pendulum and cart positions, are connected to a discrete state estimator. This estimator generates the state vector, which is then used to calculate the control input, specifically the acceleration. The acceleration is limited within a certain range by the saturation block, which ensures it does not exceed the lower and upper saturation values.

Additionally, the user has the option to define the reference cart position in the reference block, allowing for flexibility in setting the desired position for the control system. To simulate disturbances that can occur in real-world applications, a square noise with low frequency and small amplitude is added to the reference cart position.

Figure 4.3 provides an insight into the internals of the state observer. It takes two continuous input signals, cart, and pendulum position, and utilizes a preconfigured discrete state space block to estimate an augmented state vector of 5 elements. This augmented state vector includes the cart and pendulum velocities, as well as the integral of the cart position error.

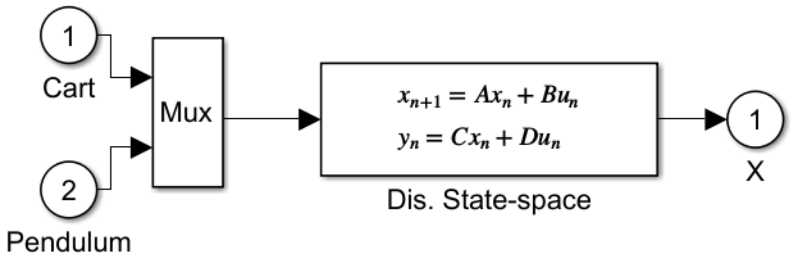


Figure 4.3: Discrete time cartpole state observer block

Simulation of the augmented LQR controller, as depicted in Figure 4.4, involves modeling and analyzing the four states of the inverted pendulum control system. The figure illustrates the cart position, cart velocity, pendulum angle, pendulum angular velocity, and the control input, which corresponds to the acceleration applied to the system.

In this simulation, the reference position for the cart is set to 0.2m, representing the desired position of the cart. The augmented LQR controller is designed to regulate the system and maintain the pendulum in this upright position, even in the presence of disturbances or external forces.

Through the simulation, the performance and effectiveness of the augmented LQR controller can be evaluated, ensuring stable and precise control of the inverted pendulum system.

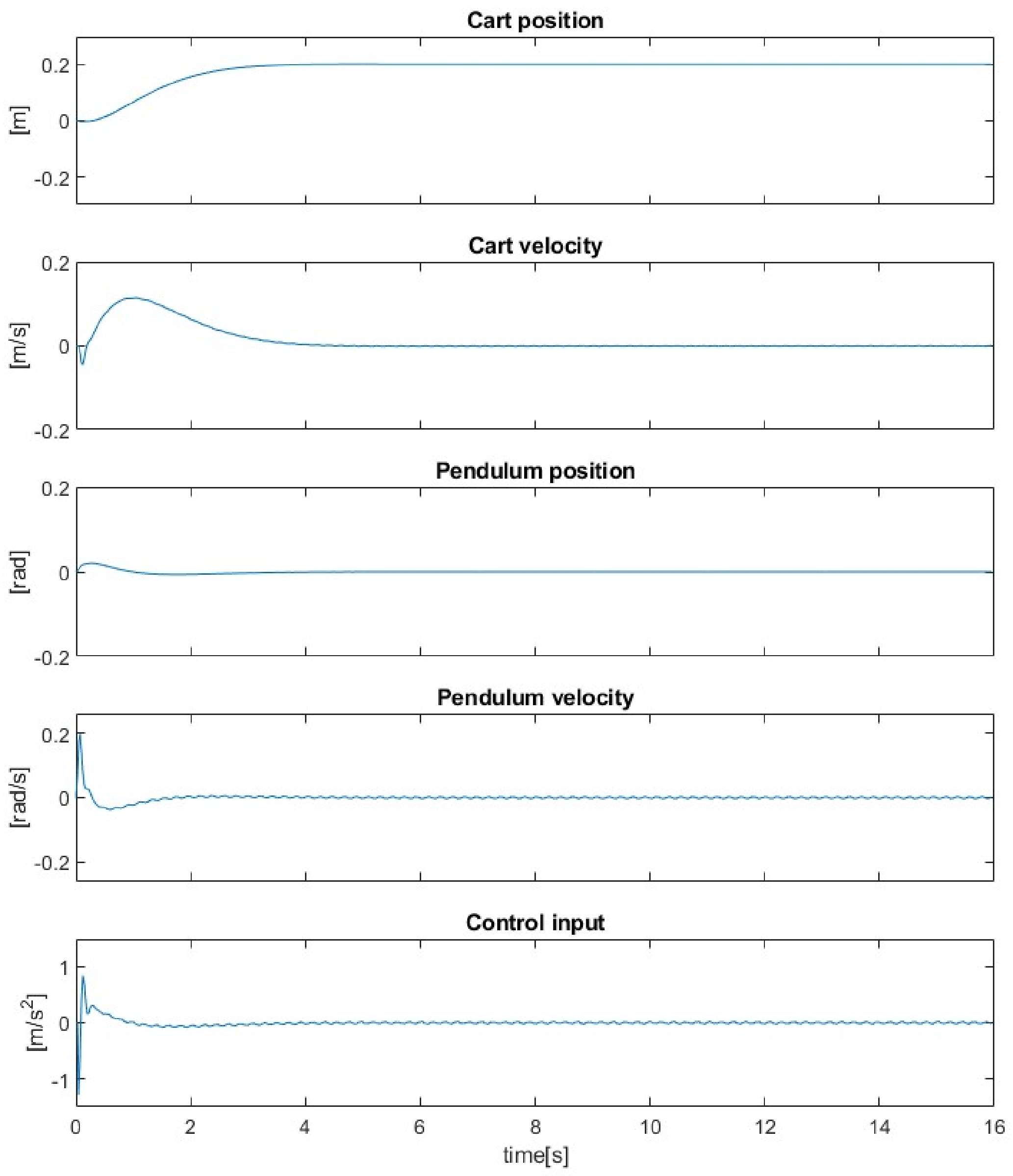


Figure 4.4: Pendulum Balancing with Augmented LQR and Cart Reference Position at 0.2m.

## 4.4 PID Controller Design

As an alternative to the augmented LQR controller, the PID control strategy is applied to balance the inverted pendulum at the upright position. The design of the PID control strategy is based on the previous thesis work of Francisco Javier Collado Valero, a former student from the Technical University of Madrid [22 p. 37-41]. However, several modifications were made to the PID parameters selection, and the control input was changed from cart velocity to cart acceleration. This change was made due to the better stability characteristics observed with acceleration control input compared to velocity, which exhibited oscillation and stability problems.

The PID control strategy involves applying two independent PID controllers, one for the cart position and another for the pendulum position. The control outputs generated by each PID block are summed before being applied to the cartpole model, as shown in Figure 4.5. Unlike the LQR controller, the PID controller does not require state vectors; instead, it tracks the error between the desired and actual values of the process variable. Therefore, a state estimator is not necessary. Instead, the output signals of the cartpole model are discretized using a zero-order hold block available in the Simulink library. The same discretization process is applied to the control output using the zero-order hold block. All three signals, including the model outputs, control output, and reference signals, are discretized with a discretization period equal to the control period of 25 milliseconds. This is done to simulate the implementation of the discrete PID controller in real-world applications, where it would be implemented in C code.

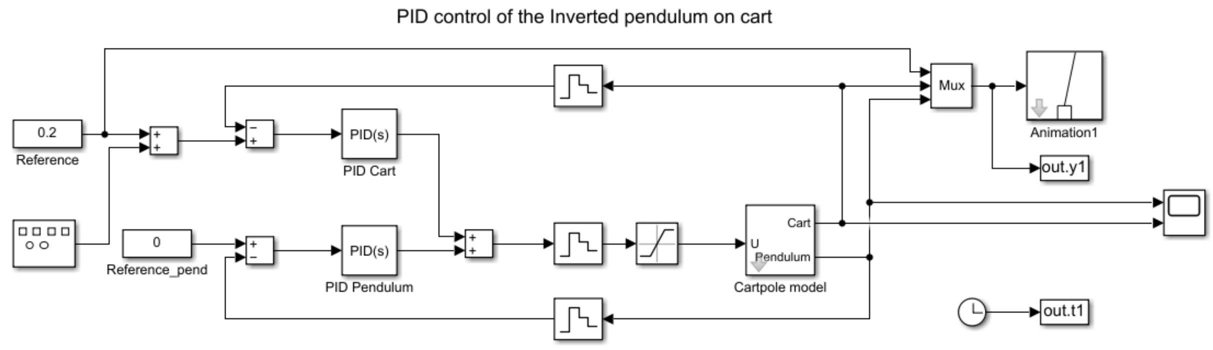


Figure 4.5: Simulink model of the PID control system

Similar to the Simulink model of the cartpole control model implemented with the LQR controller, control input saturation, the cart position reference, and the introduction of a square disturbance are included in the PID control model.

The PID controllers prioritize the pendulum angular position error while giving less importance to the cart position. The tuning of the PID parameters is performed using the Zeigler-Nichols technique. Initially, the PID controller for the pendulum position is tuned by setting the cart position PID parameters to zero. Once the pendulum position is maintained at the upright position, mutual tuning of both control loops is performed.

The tuning procedure for the PID controller using the Zeigler-Nichols technique involves the following steps:

1. In the initial step, a P (Proportional) controller is utilized. The value of $\boldsymbol{K_u}$, representing the gain at which the closed-loop system exhibits steady oscillations, is determined. Additionally, the oscillation period, $\boldsymbol{T_u}$, is identified. Obtaining $\boldsymbol{K_u}$ can be accomplished through closed-loop simulations or real system experiments.
2. Once $\boldsymbol{K_u}$ and $\boldsymbol{T_u}$ are known, the parameters of the PID controller can be selected based on Table 4.1.

| Controller type | $K_p$ | $K_I$ | $K_D$ |
|---|---|---|---|
| P | $0.5K_u$ | - | - |
| PI | $0.45K_u$ | $1.2K_p/T_u$ | - |
| PD | $0.8K_u$ | - | $K_pT_u/8$ |
| PID | $0.6K_u$ | $2K_p/T_u$ | $K_p\,T_u/8$ |

Table 4.1 PID tuning parameters

3. The selection of the suitable controller relies on the desired performance criteria. The effects of independently increasing the PID parameters can be observed in Table 4.2.

| Parameter | Rise time | Overshoot | Steady-state error | Stability |
|---|---|---|---|---|
| $K_p$ | decrease | increase | decrease | degrade |
| $K_I$ | decrease | increase | eliminate | degrade |
| $K_D$ | minor change | decrease | no effect | improve if “small” |

Table 4.2 PID parameters increase effect.

By selecting the appropriate PID controller parameters based on the system response characteristics and desired performance, the PID controllers can be tuned to effectively control the system.

Finally, the model is simulated, and the results of the PID controller are shown in Figure 4.6. The simulation demonstrates the behavior of both the cart position and pendulum position as the control input, which is the acceleration, is applied. It can be observed that both the cart and pendulum positions converge to their respective reference values.

During the simulation, the reference position for the cart is set to 0.2m, representing the desired position for the cart. The PID controller adjusts the control input based on the error between the current positions and the reference position, aiming to minimize this error and achieve accurate positioning of both the cart and the pendulum.

The simulation results validate the effectiveness of the PID controller in regulating the cart and pendulum positions, highlighting its capability to achieve the desired reference values.

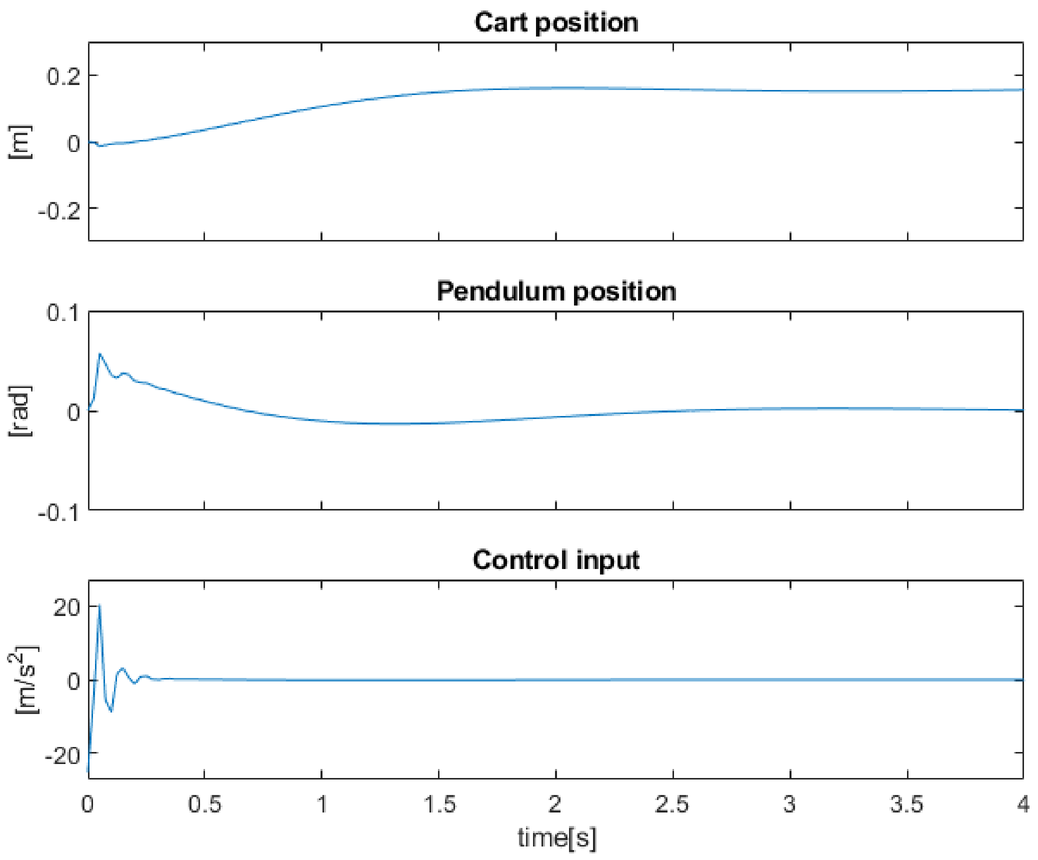


Figure 4.6: Pendulum Balancing with PID Controllers and Cart Reference Position at 0.2m.

# Chapter 5

## 5. Development of a reinforcement learning based control system in OpenAI Gym

Gym OpenAI [16] is a toolkit for reinforcement learning research that provides a simple interface to various environments simulating real-world problems, including robotics and game playing. While most environments in the OpenAI platform are designed for gaming, there is an exception called “MuJoCo” that offers a library of physics-based environments, such as the “Inverted Double Pendulum” shown in Figure 5.1. This environment demonstrates the capabilities of the MuJoCo library in providing realistic physics simulations for reinforcement learning and control tasks.

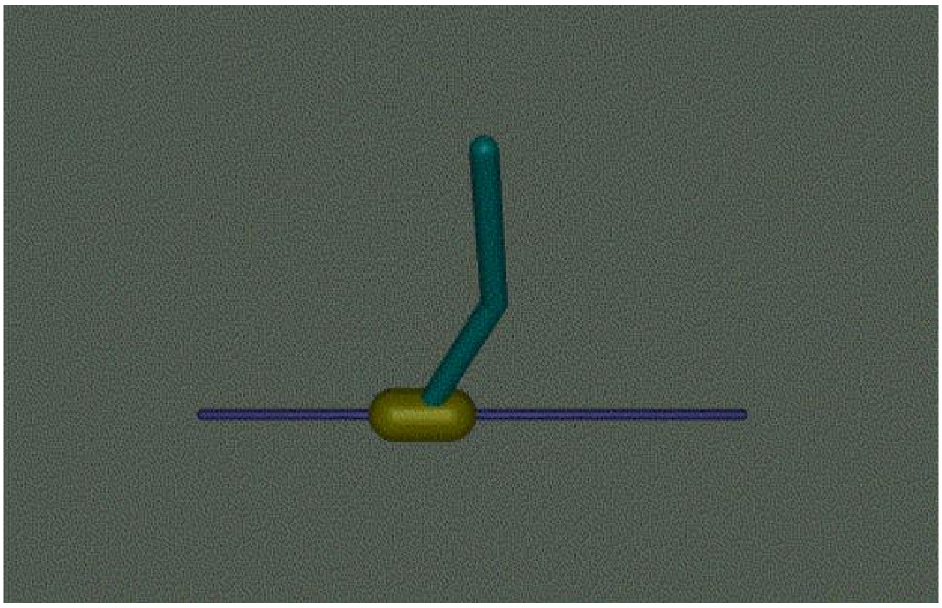


Figure 5.1: Inverted Double Pendulum of MuJoCo library.

Among the environments in OpenAI Gym, there is the "CartPole" system, where the objective is to balance a pole attached to a moving cart. The "CartPole" environment is selected to simulate the physical model of an inverted pendulum. This environment serves as a testbench for developing and evaluating reinforcement learning algorithms and control systems.

In this research, the existing “Cart Pole” model from Gym OpenAI is adapted to the real physical cart pole control model. This adaptation allows the application and evaluation of a specific type of reinforcement learning algorithm for controlling the physical cart pole system. By bridging the gap between simulation and reality, the effectiveness and transferability of reinforcement learning techniques in real-world control tasks are explored. The insights gained from this study can contribute to the development of robust and adaptive control systems across various domains.

Programming and code executions for this research are conducted in Jupyter Notebook [17] using the IPython (“Interactive Python”) command shell. This setup provides a convenient and interactive environment for implementing, testing, and analyzing reinforcement learning algorithms and control strategies for the cart pole system.

## 5.1 Reinforcement learning

The concept of reinforcement learning involves determining the optimal mapping from situations to actions in order to maximize a reward signal. These problems are considered closed-loop as the actions taken by the learning system affect its subsequent inputs. Unlike traditional machine learning approaches where actions are explicitly defined, reinforcement learning requires the system to explore and determine the actions that lead to the greatest reward. In complex and challenging scenarios, actions can not only impact immediate reward but also affect subsequent rewards through their impact on future situations. [18]

In reinforcement learning, an agent interacts with its environment by taking actions and receiving rewards. In the case of the “Cart Pole” system, the agent must control the movement of the cart to balance the pole, and it receives a reward for keeping the pole upright. By using machine learning techniques, the agent can learn from its experience and improve its control over time.

One of the main advantages of using Gym OpenAI and the “Cart Pole” (CP) environment is that it provides a standard platform for comparing different reinforcement learning algorithms and control systems. It also offers a simple and convenient way to design, implement, and evaluate these algorithms. By utilizing OpenAI Gym, researchers can focus on the development of new and innovative reinforcement learning techniques, rather than on the design of the environment and the underlying physical system, which in turns significantly decrease time dedication to recreating the model.

Thus the “Cart Pole” environment provides a valuable resource for reinforcement learning research, particularly in the areas of control systems and machine learning. By offering a standard platform and a simplified environment, researchers can concentrate on the development of new algorithms and approaches to solving problems in AI.

## 5.2 OpenAI Gym Cart Pole environment description

This environment is based on a version of the cart-pole problem originally developed by Barto, Sutton, and Anderson in their work titled "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problem". It involves a freely rotating pole attached to a cart via a hinge, allowing the pole to move along a track. The system assumes the neglect of static and Coulomb frictions. [19]

At the beginning of each episode, the pendulum starts in an upright position on the cart. The primary objective is to balance the pole by applying forces in horizontal directions to the cart. The agent must select actions from a simple action array, which only has two possible values. These values correspond to the direction of the fixed force to be applied on the cart, as indicated in the table below.

| **Num** | **Action** |
|---|---|
| 0 | Push a cart to the left |
| 1 | Push a cart to the right |

Table 5.1: Action space of the “Cart Pole” environment [20]

The observation space in the OpenAI CartPole model provides information about the system's variables. It is represented as a 4-dimensional array, which includes the current values of the cart's position, cart's velocity, pole's angle, and pole's angular velocity. These variables collectively define the state of the system at any given time. The range of permissible values for each quantity within the observation space can be found in Table 5.2.

| **Num** | **Observation** | **Minimum value** | **Maximum value** |
|---|---|---|---|
| 0 | Cart Position | -4.8 m | 4.8 m |
| 1 | Cart Velocity | -Infinity | Infinity |
| 2 | Pole Angle | ~ -0.418 rad (-24°) | ~ 0.418 rad (24°) |
| 3 | Pole Angular Velocity | -Infinity | Infinity |

Table 5.2: Observation space of the “Cart Pole” environment [20]

The ranges mentioned above represent the possible values for each element of the observation space. However, it's important to note that these ranges don't necessarily reflect the permissible values for the state space during an ongoing episode.

Specifically, for the cart position on the x-axis (index 0 of the observation array), it can take values between -4.8 m and 4.8 m. However, if the cart position exceeds the range of -2.4 m to 2.4 m, the episode will automatically terminate.

Similarly, the pole angle has an observation value range between **±24°.** Nevertheless, if the pole angle goes beyond the range of ±**12°,** the episode will be terminated. These limits ensure that the system remains within a stable and controllable range during the learning process. [20]

When a new episode starts in the "Cart Pole" environment, the initial state of all observations is assigned a uniformly random value within the range of (-0.05, 0.05). This initialization places the cart close to the center and the origin, while the pole remains in an upright position, and all velocities are negligible.

Throughout each step of the episode, the agent receives a reward of +1 for successfully balancing the pendulum. In the "v1" version of the model, the episode is limited to a maximum of 500 steps. Therefore, in the best-case scenario where the agent maintains balance for the entire episode, the total reward value can reach a maximum of 500.

These settings ensure that the initial state provides a neutral starting point for each episode, and the reward mechanism encourages the agent to strive for optimal performance by balancing the pendulum over multiple steps.

## 5.3 Custom environment design in OpenAI Gym

Designing an environment in OpenAI Gym is important for solving problems effectively. There are several important elements to consider when designing a custom environment, such as:

1. State representation: This defines the current state of the environment and should accurately capture relevant information for the problem.
2. Action space: The available actions the agent can take must be defined.
3. Reward function: Feedback for the agent's actions should be provided and encourage a successful outcome by rewarding or penalizing depending on the outcome of the applied action.
4. Observation space: The information available to the agent at each step should be defined.
5. Episode termination: The environment should define when an episode ends, for example, when the pole falls, or a time limit is reached.

By taking these elements into consideration, a well-designed environment that accurately represents the problem and provides useful information to the reinforcement learning algorithm can be created.

In this thesis proper cart pole environment is designed taking as a reference already existing “Cart Pole” model by Barto, Sutton, and Anderson. Several parameters such as cart and pole dimensions, permissible values for pole angle and cart position, force magnitude and various rendering parameters are modified to meet some requirements of the existing physical model. During the design of a custom environment, physical parameters such as the mass of the cart and pendulum, their lengths, the track length, the maximum frequency of the stepper motor, the sampling time of sensors and controllers, and the capacities of the FPGA board are all taken into consideration.

The cart position value has an observation value range between **±4.8** (in meters) and its permissible values are in the range between **±0.48,** which gives the distance equal to 0.96 meters that corresponds to the trajectory limitations of the real model. The pole angle observation value range remained unchanged while its permissible values for the state space is between **±7.5°.** The environment initiation and the permissible value range definition can be seen in Listing 5.2.

```python
import math
import gym
from gym import spaces, logger
from gym.utils import seeding
import numpy as np


class AkbarCartPoleEnv(gym.Env):

    metadata = {"render.modes": ["human", "rgb_array"], "video.frames_per_second": 50}

    def __init__(self):

         # Angle at which to fail the episode
        self.theta_threshold_radians = 7.5 * 2 * math.pi / 360
        self.x_threshold = 0.48
```

Listing 5.2: Cart position and pole angle permissible values definition in Python

The control system is configured with a sampling time of 25 milliseconds, and the maximum force applied by the belt and pulley system to the cart is set to $15\ N$, as determined in Chapter 2. These values, along with other relevant physical dimensions of the cart and pole, are declared in Listing 5.3.

```python
        #Physical parameters of the system
        self.gravity = 9.8
        self.masscart = 0.455
        self.masspole = 0.044
        self.total_mass = self.masspole + self.masscart
        self.length = 0.177  # actually half the pole's length
        self.polemass_length = self.masspole * self.length
        self.force_mag = 15.0
        self.tau = 0.025  # seconds between state updates
        self.kinematics_integrator = "euler"
```

Listing 5.3: The physical parameters definition

As shown in Listing 5.4, the Python code implements the equations for cart displacement and pole rotation, which are derived from the initial model. The control input is represented by a force with a constant magnitude, as defined earlier, and its sign (positive or negative) depends on the action value. The cart and pole accelerations (“thetaacc” and “xacc”) are calculated based on the force applied during each control period. After determining the accelerations, the system's four states, namely the cart position “x” and velocity “x_dot”, and the pole position “theta” and velocity “theta_dot”, are computed using the equations provided in Listing 5.4.

```
x, x_dot, theta, theta_dot = self.state
force = self.force_mag if action == 1 else -self.force_mag
costheta = math.cos(theta)
sintheta = math.sin(theta)

# For the interested reader:
# https://coneural.org/florian/papers/05_cart_pole.pdf
temp = (
    force + self.polemass_length * theta_dot ** 2 * sintheta
) / self.total_mass
thetaacc = (self.gravity * sintheta - costheta * temp) / (
    self.length * (4.0 / 3.0 - self.masspole * costheta ** 2 / self.total_mass)
)
xacc = temp - self.polemass_length * thetaacc * costheta / self.total_mass

if self.kinematics_integrator == "euler":
    x = x + self.tau * x_dot
    x_dot = x_dot + self.tau * xacc
    theta = theta + self.tau * theta_dot
    theta_dot = theta_dot + self.tau * thetaacc
```

Listing 5.4: System equations

In the OpenAI Gym Cart Pole environment, two rendering modes are available: "human" and “rgb_array”. For this thesis, the “human” rendering mode was chosen, and a 3x1 scale was applied to the linear dimensions of the cart and pole, as illustrated in Listing 5.5. The purpose of this adjustment was to enhance the visualization of the objects within the environment.

```
def render(self, mode="human"):
    screen_width = 900
    screen_height = 600

    world_width = self.x_threshold * 2
    scale = screen_width / world_width
    carty = 100  # TOP OF CART
    polewidth = 3*3.0
    polelen = 3*scale * (2 * self.length)
    cartwidth = 3*30.0
    cartheight = 3*15.0
```

Listing 5.5: Rendering and parameters for the model visualization.

Additionally, the custom environment includes visual representations of the cart and pole, along with the corresponding position and angle indicators. These visual elements provide a clear and intuitive display of the system's state during simulation. The custom environment of the Cartpole system is depicted in Figure 5.2.

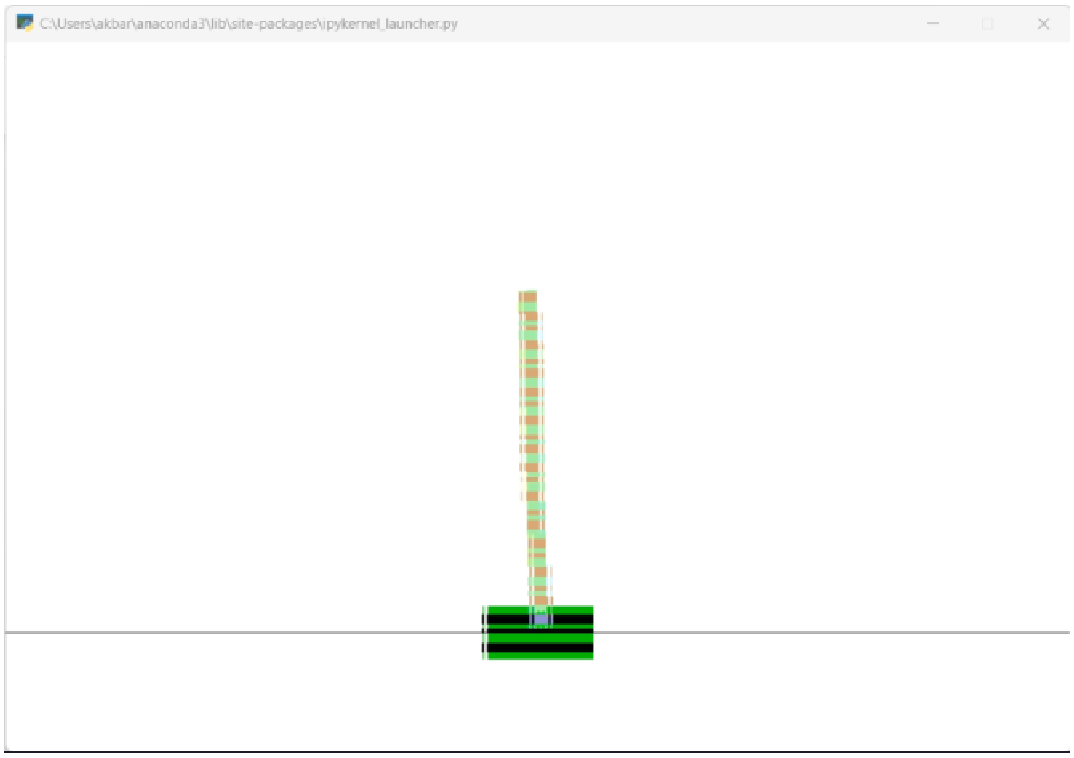

C:\Users\akbar\anaconda3\lib\site-packages\ipykernel_launcher.py

Figure 5.2 Custom environment visualization

To validate the custom environment, random actions are applied to the pendulum by moving the cart either to the left or right at each sampling time. These actions are assigned using the function "env.action_space.sample()", as depicted in Listing 5.6. The environment is then tested for 10 episodes, with each episode limited to 500 sampling time periods, which is equivalent to 12.5 seconds. During each episode, the cumulative rewards, also known as the score, are recorded. By comparing the scores across episodes, the performance and effectiveness of the custom environment can be evaluated.

```
env = AkbarCartPoleEnv()
```

```
for episode in range(1, 11):
    score = 0
    state = env.reset()
    done = False

    while not done:
        env.render("human")
        action = env.action_space.sample()
        n_state, reward, done, info = env.step(action)
        score += reward

    print('Episode:', episode, 'Score:', score)
env.close()
```

Listing 5.6 Custom environment validation

Listing 5.7 displays the cartpole score values, which are notably low compared to the maximum reward value of 500 per episode. This observation suggests that the considered cart pole system is unstable in the upright position. The low scores indicate that the pendulum frequently deviates from its balanced state, indicating the need for a robust control strategy to stabilize the system and improve its performance.

```
Episode: 1 Score: 7.0
Episode: 2 Score: 10.0
Episode: 3 Score: 11.0
Episode: 4 Score: 17.0
Episode: 5 Score: 18.0
Episode: 6 Score: 21.0
Episode: 7 Score: 6.0
Episode: 8 Score: 14.0
Episode: 9 Score: 5.0
Episode: 10 Score: 5.0
```

Listing 5.7: Scores of Episodes with Random Actions Applied

The cart pole environment testing process can be seen accessing the link provided below: https://upm365-my.sharepoint.com/:v:/g/personal/a_abdusamadov_alumnos_upm_es/EYfRTkfhRVtBhYDa6fXtL4oBxikjUu6GjSgSIueV4JdnmQ?e=dEYIZB

It is important to note that the episode ends when the angle value goes out of range. Therefore, the falling of the pendulum is not shown in the video above.

## 5.4 PPO type Reinforcement learning with a black box model of pendulum.

PPO, or Proximal Policy Optimization, is a type of reinforcement learning algorithm that was developed in 2017 by OpenAI. It combines ideas from both value-based and policy-based methods. It is designed to be both effective and computationally efficient, making it a popular choice for many real-world applications. PPO reaches a balance between ease of implementation, complexity of the sample to be learned and simple tuning. It computes an update at each step that in turn leads the minimization of the cost function value and guarantees relatively small values of the deviation from the previous policy [21]

In the context of an inverted pendulum on a cart, PPO can be used to train an agent to control the cart and keep the pendulum upright. The agent receives a reward signal based on the position of the pendulum and the velocity of the cart, and it must learn to take actions that maximize this reward signal.

Several libraries, such as “baselines3” and “pyglet”, need to be installed to use PPO reinforcement learning. To apply PPO to the Custom Cart Pole environment designed in this thesis, the environment should be defined and vectorized using the “DummyVecEnv” policy provided by the “stable_baselines3” library as shown in Listing 5.8. The state representation, action space, reward function, and observation space should be defined in a way that accurately reflects the problem to be solved. The next step is to define the model and indicate the policy. “MlpPolicy” is chosen for PPO model training, which takes the current state as the input variable and estimates an appropriate action.

```
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv
from stable_baselines3.common.evaluation import evaluate_policy
```

```
env   = AkbarCartPoleEnv()
env = DummyVecEnv([lambda: env])
model = PPO('MlpPolicy', env, verbose=1)
```

Listing 5.8 PPO model

PPO reinforcement learning will preliminarily test the Cart Pole environment with a random control strategy to provide a functionality reference. This involves collecting a batch of experience data by executing actions in the environment and observing the resulting rewards and state transitions. The agent then updates its policy using the experience data and the PPO algorithm, in order to improve its performance and optimize the reward signal. [19]

The process of training the agent and refining its policy is repeated iteratively until a satisfactory level of performance is achieved. This involves running reinforcement learning algorithms over multiple iterations. Once the training is complete, the resulting policy can be applied in real-time to control the inverted pendulum on a cart, showcasing the agent's proficiency in the task.

For the reinforcement learning of the custom Cart Pole model, a time period of 40,000 timesteps was selected. It was found that training the model with fewer timesteps yielded inferior performance results. Listing 5.9 provides details of the calculations involved in training the model, which were performed using the CPU device.

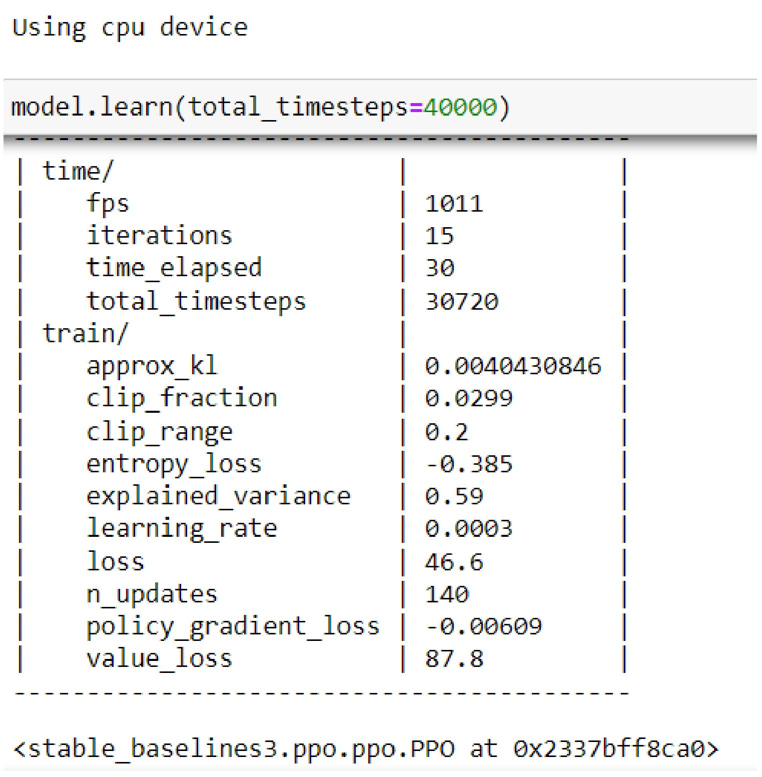
Once the model is trained, various data related to time consumption and the applied policy are presented. This includes the following:

1. Clip Fraction: A parameter used in Proximal Policy Optimization (PPO) algorithms to limit the update of policy parameters. It determines the portion of the ratio between the new and old policies that is utilized during the update step.
2. Clip Range: Another parameter in PPO algorithms that defines the range within which the ratio of new and old policies is clipped. It ensures that the update to the policy parameters remains within a specific range to prevent drastic changes.
3. Entropy Loss: A component of the PPO algorithm that encourages exploration and prevents the policy from becoming too deterministic. It is a measure of the uncertainty or randomness in the policy's output distribution.
4. Loss: The overall loss value computed during the training process. It represents the discrepancy between the predicted and expected values and is used to update the

policy parameters.

5. N_updates: The number of policy update steps performed during the training process. Each update step involves collecting new data, computing gradients, and adjusting the policy parameters.

These metrics and parameters provide insights into the training process and can be used to evaluate the effectiveness and efficiency of the trained model.

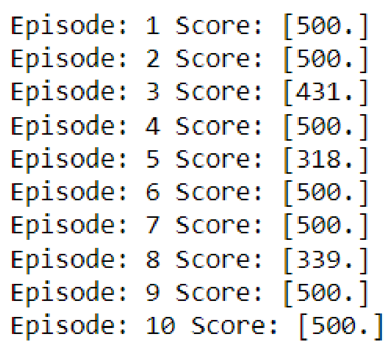


```
Using cpu device

model.learn(total_timesteps=40000)
------------------------------------------
| time/                   |              |
|    fps                  | 1011         |
|    iterations           | 15           |
|    time_elapsed         | 30           |
|    total_timesteps      | 30720        |
| train/                  |              |
|    approx_kl            | 0.0040430846 |
|    clip_fraction        | 0.0299       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.385       |
|    explained_variance   | 0.59         |
|    learning_rate        | 0.0003       |
|    loss                 | 46.6         |
|    n_updates            | 140          |
|    policy_gradient_loss | -0.00609     |
|    value_loss           | 87.8         |
------------------------------------------

<stable_baselines3.ppo.ppo.PPO at 0x2337bff8ca0>
```

Listing 5.9 PPO model training

Overall, PPO is an effective choice for solving the inverted pendulum problem using reinforcement learning, and it provides a flexible and efficient framework for training agents in complex environments. By combining ideas from both value-based and policy-based methods, it can help to ensure that the agent is able to learn effectively and make meaningful decisions based on the information it has available.

Finally, the trained model is tested to evaluate the agent’s performance.  For testing the custom environment random actions are applied to the trained model.   As each episode can last only 500 sampling time 10 episodes of the action are analyzed. High score value for each episode proves the effectiveness of PPO reinforcement training technique.



```
Episode: 1 Score: [500.]
Episode: 2 Score: [500.]
Episode: 3 Score: [431.]
Episode: 4 Score: [500.]
Episode: 5 Score: [318.]
Episode: 6 Score: [500.]
Episode: 7 Score: [500.]
Episode: 8 Score: [339.]
Episode: 9 Score: [500.]
Episode: 10 Score: [500.]
```

Listing 5.6 Trained model validation

The trained model testing episodes can be seen accessing the link provided below:
https://upm365-my.sharepoint.com/:v:/g/personal/a_abdusamadov_alumnos_upm_es/Edb1sH4b4XFKI3GXhKbchNYB9GNDQcPAhwaRNJktu8GnOg?e=RoNBGr

# Chapter 6

## 6. Hardware Block Design

This chapter focuses on the design and integration of Hardware Intellectual Properties (IP) within the block design framework. The objective is to receive and process signals using VHDL code, simulate them with a testbench in Vivado, and integrate them in the PL seamlessly using the Vitis toolkit. Additionally, the chapter explores the utilization of ILA (Integrated Logic Analyzer) for debugging purposes.

The design process involves employing the Vivado design suite by Xilinx and the Zynq processing system. Various IPs are developed specifically for fast signal processing and filtering tasks, including the optical interrupter, rotary encoder, limit switches, and motor driver control input generator. These IPs are responsible for signal detection, information processing, and generating output signals within the Zynq system. The chapter provides an explanation of the design details for these IPs, encompassing the functional blocks and their interdependencies.

Furthermore, the chapter discusses the simulation of the IPs using a testbench in Vivado to ensure their functionality. The integration of these IPs into the block design is also discussed, emphasizing the utilization of the Vitis toolkit. This integration process ensures smooth collaboration and interaction among the different IP modules. It is worth noting that certain conceptual ideas utilized in the design of the “Rotary Encoder” IP and “Limit Switches” IP were derived from the final degree thesis of a former student, Francisco Javier Collado Valero, from the Technical University of Madrid [22].

Moreover, the chapter highlights the significance of ILA debugging. By employing ILA, it becomes possible to monitor and analyze internal signals in real-time within the IPs and the overall system. This debugging capability aids in identifying and resolving potential issues, thereby enhancing the reliability and efficiency of the block design.

Thus, this chapter provides a comprehensive exploration of the design, simulation, integration, and debugging processes involved in creating custom IPs within the block design framework. Following these processes enables the development of a robust and efficient system capable of effectively receiving, processing, and generating signals for various applications.

## 6.1 Optical Interrupter IP

The IP of Optical interrupter serves for filtering the input signal generated by sensor applying debouncing filter. The output signal **“debounced_intr”** is 1 bit signal and takes the value of 1 when an object is detected by the sensor, and 0 when no object is detected. This signal is used to indicate the presence or absence of an object and is referred to as a “debounced” signal because it is used to eliminate any “bouncing” or false signals that may be generated by the sensor.

As shown in Figure 6.1, the optical interrupter sensor is connected to the input signal **“intr”**, which is read from the Digital IO pin of the PYNQ Prototyping board. The **“s00_axi_aclk”** signal is used to synchronize the operation of the IP with the Processing System clock frequency and to control the timing of various operations within the IP. The **“s00_axi_aresetn”** is an active-low reset signal used to reset the internal state of the IP.

Additionally, in the hardware design, there is a slave interface called **“S00_AXI”** that enables data transfer between the IP and the FPGA processing system. This interface allows communication between the IP and the processor. In this particular case, the output signal **“debounced_intr”** is connected to the processor through the **“S00_AXI”** slave register. This enables the processor to read the status of the debounced interrupt signal from the IP. The slave register acts as a communication bridge, allowing the processor to access and control various parameters and signals within the IP module.

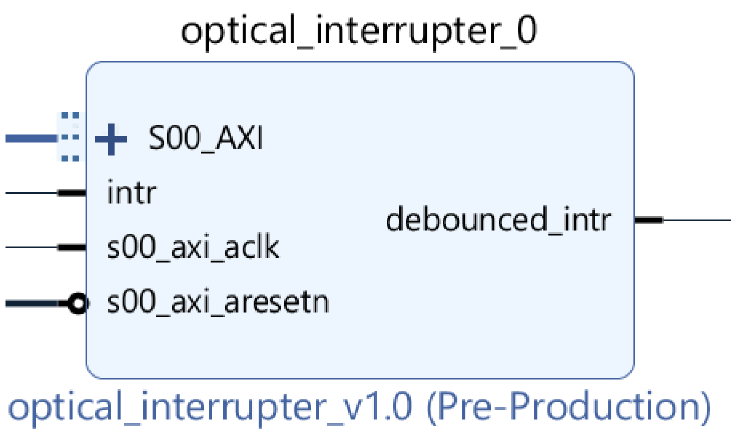


Figure 6.1: IP of the Optical Interrupter

### Simulation of Optical Interrupter IP

To ensure the accuracy of the Optical Interrupter IP's functionality, a simulation tool provided within the Vivado toolkit is utilized. A specialized testbench, written in VHDL, is created to simulate the IP module. The testbench module is specifically designed to test the Device Under Test (DUT), which, in this case, is the Optical Interrupter IP. Its primary objective is to verify the correct operation of the DUT by applying input stimuli and validating the expected outputs [23]. By utilizing the testbench in Vivado, designers can simulate different scenarios and evaluate the performance of the IP before deploying it. This process aids in the detection and resolution of potential issues or bugs prior to implementing the IP in real-world applications.

In the context of input signal **“intr”**, a debounce mechanism is applied to filter out bounces, considering a debounce time of 1ms. This ensures that the input signal is considered stable after the bouncing has stopped. Consequently, the output signal **“debounced_intr”** takes on a value of 1 when the duration of the input signal is longer than 1ms, as shown in Figure 6.2.

In addition, it is important to note that in the simulations of all IPs, the **“reset”** signal is generated by inverting the **“s00_axi_aresetn”** input signal. The **“reset”** signal is active high, which means it is in its active state when it is at a high voltage level.

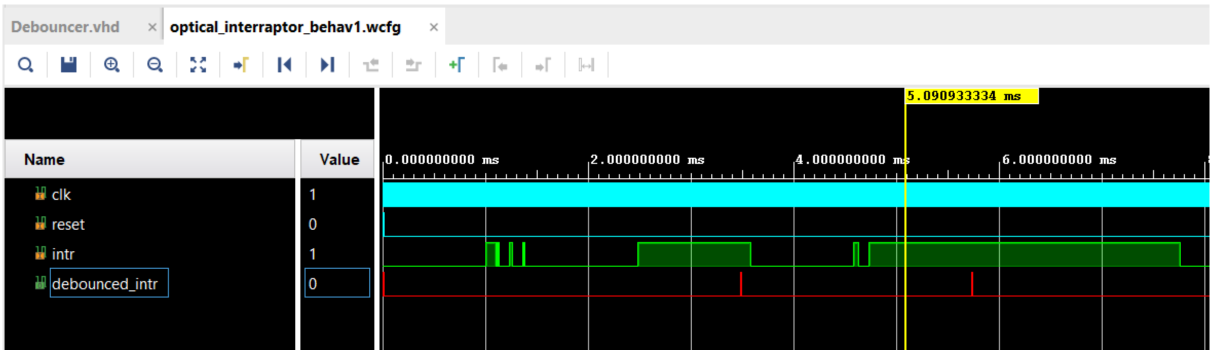


Figure 6.2: Simulation of the Optical Interrupter IP

## 6.2 Rotary Encoder IP

The relative angle encoder is the sensor used to measure the angular displacement of the pendulum. This encoder provides a digital output, which means that the output signal takes the form of a square wave with 600 pulses per revolution. The optical interrupter's output signal, referred to as **“intr_pos”**, is used as a reference signal to indicate the stable position of the pendulum when it is in the downward position.

The encoder IP module has two external inputs, namely **“ch_a”** and **“ch_b”**, which are connected to Digital IO pins on the FPGA board. These signals are generated by the relative encoder and have a phase difference of 90 degrees during the rotation of the pendulum.

As depicted in Figure 6.3, the encoder IP module contains two output signals: **“dir”** and **“pos_ctr”**. Both of them are connected to the processor via the **“S00_AXI”** slave register. The **“dir”** signal indicates the turning sense, with a value of '1' representing clockwise rotation and '0' representing counterclockwise rotation. On the other hand, the **“pos_ctr”** signal is a 32-bit value that serves as the position counter, keeping track of the angular position of the pendulum.

To provide a finer resolution for position tracking, the counter module is designed to count up to 2399. In this configuration, the rest position of the pendulum corresponds to a count of '0', while the upright position corresponds to a count of 1199. This division of the counter's range allows for more precise monitoring and control of the pendulum's position.

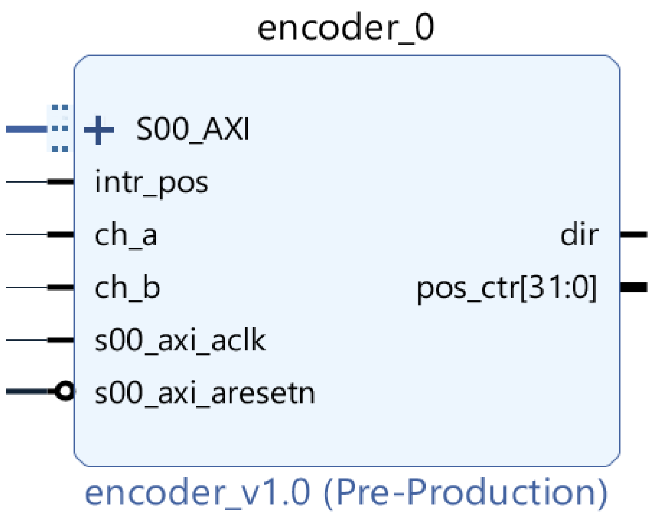


Figure 6.3: IP of angle encoder

## Simulation of Rotary Encoder IP

The input signals from the encoder exhibit debounces, which can make it challenging to accurately observe the signals. To address this issue, dedicated debounce mechanisms are implemented for both the **“ch_a”** and **“ch_b”** signals. These mechanisms store the previous states of the signals, **“a_prev”** and **“b_prev”**, respectively, with a 250-microsecond delay. Although the debounce filter introduces a delay into the observation, its impact is negligible within the control system's period, which is equal to 25 milliseconds. Therefore, it will not negatively affect the control system's performance.

By implementing the debounce filtering, transient signal states with durations shorter than 250 microseconds, which are considered as bounces, are effectively eliminated. This ensures that only stable and reliable signal states are registered and used for further processing and analysis.

Once the external input signals are filtered, the **“dir”** and **“pos_ctr”** signals can be generated. As illustrated in Figure 6.4, the **“pos_ctr”** value increments or decrements depending on whether the **“a_prev”** or **“b_prev”** signal is ahead at both the rising and falling edges of the signals. Additionally, when the interrupt signal **“intr_pos”** takes a value of 1, indicating that the pendulum has reached the downturn point, the position counter **“pos_ctr”** is reset to 0. It's important to note the priority of the **“intr_pos”** signal over the **“a_prev”** and **“b_prev”** signals.

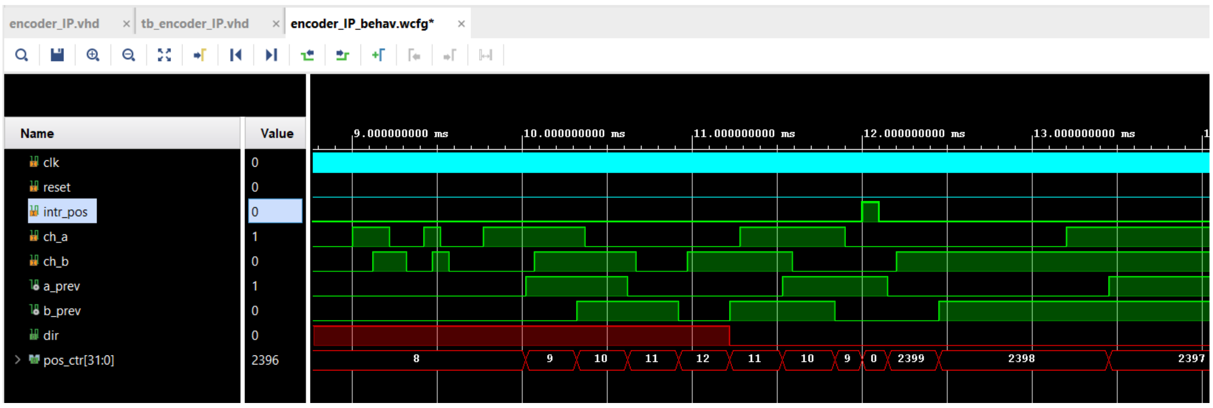


Figure 6.4: Simulation of the Rotary Encoder IP

## 6.3 Limit Switch IP

The limit switch functionality is a critical component for ensuring the accurate positioning of the cart within the system. The IP module receives four input signals, namely **“ls_left_1”**, **“ls_left_2”**, **“ls_right_1”** and **“ls_right_2”**, as depicted in Figure 6.5. These limit switches are strategically positioned to detect the end-of-travel positions of the cart, providing vital information to the control system.

The Limit Switch IP has only one external output called **“ls_out”**, which is connected to the Motor Driver IP. This output signal likely controls the motor driver to stop the cart when the limit switch is triggered.

Additionally, there are two output signals, **“out_right”** and **“out_left”**, which are sent to the Processing System via the **“S00_AXI”** interface. These signals indicate whether the cart is on the right or left side of the shaft, providing positional information to the system.

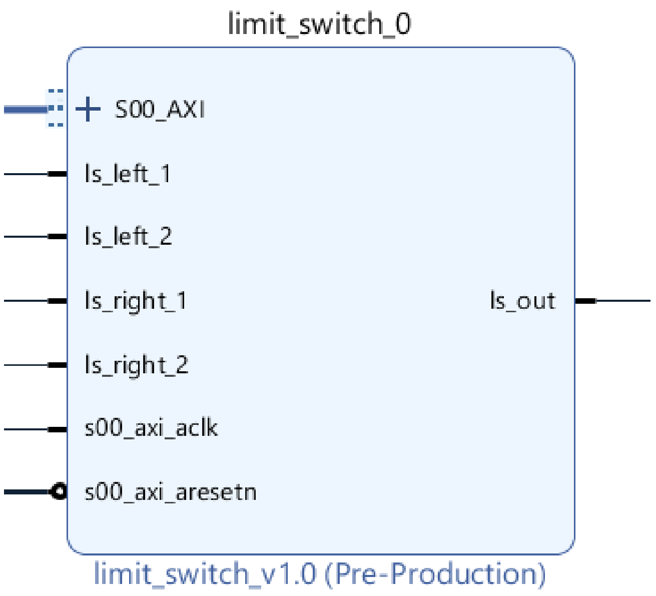


Figure 6.5: IP of limit switches

## Simulation of Limit Switch IP

To mitigate the issues caused by the extra bounces in the limit switch signals, a debounce filter is employed. A debounce filter similar to the one used for a rotary encoder is chosen, with a debounce delay of 1ms. This filter effectively removes the noise and fluctuations present at the rising and falling edges of the limit switch signals, ensuring reliable and stable detection of the end-of-travel positions.

The filtered values of the input signals are labeled as **"db_left_1"**, **"db_left_2", "db_right_1"** and **"db_right_2"**. As illustrated in Figure 6.6, these filtered signals exhibit the elimination of the extra bounces observed in the original signals. While the filtered signals introduce a 1ms delay compared to the original signals, this delay is inconsequential to the control system as its control period is significantly greater.

The signals **"out_left"** and **"out_right"** are derived from the filtered input limit switch signals by applying an OR function. These signals indicate the current status of the cart's position, specifically whether it is on the left or right side. On the other hand, the **"ls_out"** signal detects if any of the limit switches have been triggered, providing valuable information to the control system.

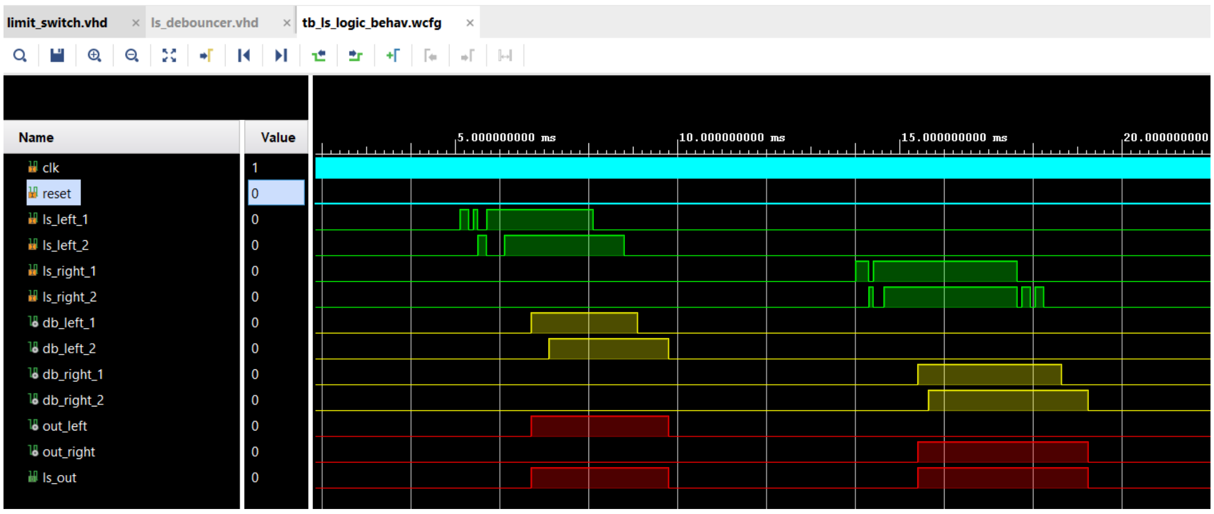


Figure 6.6: Simulation of the Limit Switch IP

## 6.4 Stepper Motor Driver IP

The Stepper Motor Driver IP is responsible for controlling the motor driver within the system. Similar to the previous IP blocks, it receives several input signals, including **"en_n"** from the Limit Switch IP, **"s00_axi_clk"**, and **"s00_axi_aresetn"** for synchronization and reset purposes. The **"en_n"** signal is an active low enable signal connected to the **"ls_out"** output port, which enables or disables the generation of the step signal when one of the limit switches is pressed.

Furthermore, the IP module receives two input signals, namely **“register_dr”** and **“step_period”**, from the Processing System through the **“S00_AXI”** interface. The **“register_dr”** signal is a 8-bit signal that contains all the driver output signals. This allows for the control and configuration of various aspects of the motor driver's operation.

The **“step_period”** input signal is utilized to modify the maximum count number of the frequency divider. By adjusting this value, the corresponding output signal **“step”** can be generated to control the stepper motor driver effectively. This enables precise control over the motor's movement and speed within the system.

Furthermore, the Motor Driver IP has 8 output ports that correspond to the number of control input ports of the DRV8825 motor driver carrier by Pololu. These output ports are used to control various aspects of the motor driver's operation and can be seen in Figure 6.7.

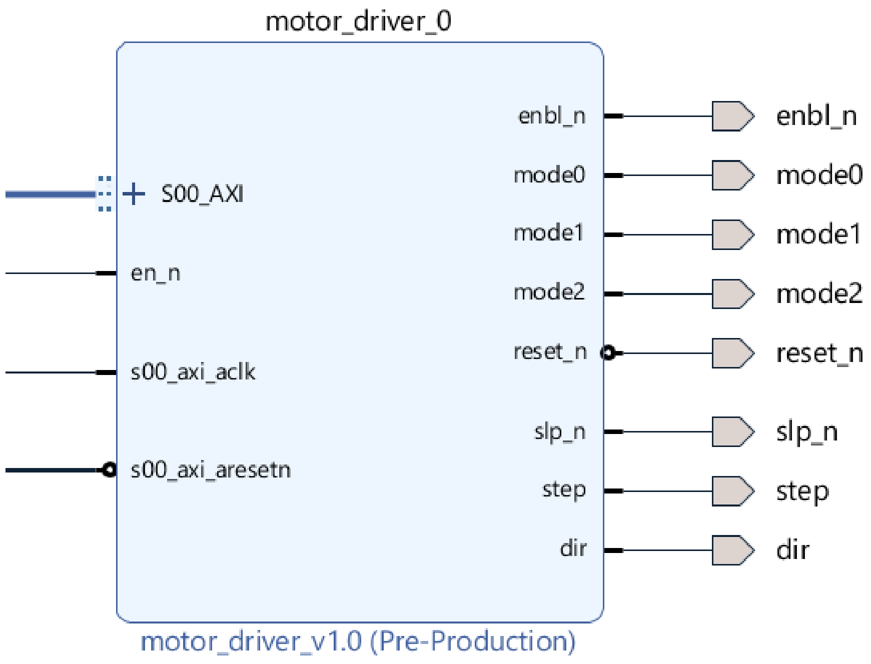


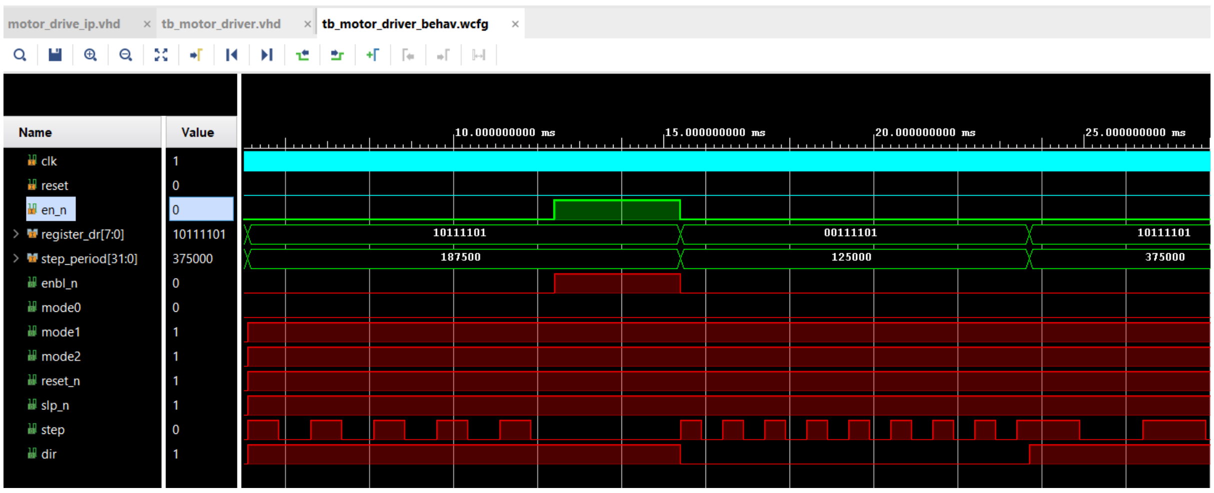
Figure 6.7: IP of the stepper motor driver

## Simulation of Stepper Motor Driver IP

Six out of the eight output signals, namely **“mode0”**, **“mode1”**, **“mode2”**, **“reset_n”**, **“slp_n”**, **“dir”** directly mirror the value received from the Processor unit through the **“register_dr”** signal. However, the motor driver enable signal, **“enbl_n”**, is a combination of the input signal **“en_n”** from the Limit Switch IP and another value from the **“register_dr”** signal of the Processor. This configuration enables two modes of control for the stepper motor drive.

In the security mode, **“enbl_n”** is set equal to the value of **“en_n”**. This ensures that when the cartpole reaches the end of the carrier or if any of the limit switches are disconnected or malfunctioning, the motor is disabled by the **“en_n”** signal from the Limit Switch IP. In autonomous mode, **“enbl_n”** is controlled directly by the processor unit. This allows the motor to be used even in the event of limit switch damage or connection problems.

Another output signal, **“step”**, is generated using a frequency divider based on the value of **“step_period”** received from the processor. The main reason to generate the **“step”** signal by hardware but directly from the processor is that its implementation requires timer counters which adds unwanted latency and delays to the inverted pendulum control system. Figure 6.8 illustrates that the **“step”** signal becomes zero when one of the limit switches is pressed.



| Name | Value |
|---|---|
| clk | 1 |
| reset | 0 |
| en_n | 0 |
| register_dr[7:0] | 10111101 |
| step_period[31:0] | 375000 |
| enbl_n | 0 |
| mode0 | 0 |
| mode1 | 1 |
| mode2 | 1 |
| reset_n | 1 |
| slp_n | 1 |
| step | 0 |
| dir | 1 |

Figure 6.8: Simulation of the Stepper Motor Driver IP

## 6.5 Control System Hardware Block Design

The hardware block design of the control system consists of several IPs, including the Optical Interrupter, Rotary Encoder, Limit Switch, Stepper Motor Driver, Zynq7 Processing System IP, AXI Interconnect, and Processor System Reset. This configuration is illustrated in Figure 6.9.

In this design, all IPs, except the Processing System, are considered programmable logic (PL) components. They are interconnected using the AXI GP (General Purpose) interface, and the communication between them is managed by the **AXI Interconnect IP**. Acting as a central hub, the AXI Interconnect IP facilitates communication between the different IPs in the system. The Zynq7 Processing System IP serves as the master interface, while the other IPs function as slave interfaces.
As the master interface, the Processing System IP has the ability to request and send data to the slave IPs. On the other hand, the slave IPs can only receive and send data when requested by the Processing System. This arrangement allows for efficient data transfer and control between the various components in the system.

The **AXI Interconnect IP** provides an advanced and extensible interface for communication between the IP blocks. It enables scalable and flexible communication, allowing the IP blocks to share data and control signals. This integration of multiple functionalities in a single system leads to improved efficiency and optimized designs. The AXI Interconnect IP supports multiple masters and slaves, enabling concurrent communication between IP blocks, which is essential for complex designs with diverse functionalities.

The **Zynq7 Processing System IP** is a pre-packaged IP block provided by Xilinx Vivado Design Suite. It allows for easy integration of a processing system into the FPGA design. The IP includes the dual-core ARM Cortex-A9 processor, memory controllers, peripheral controllers, and other system components necessary for creating a complete processing system.

The **Processor System Reset IP** provides a controlled and synchronized reset mechanism for the processing system. It ensures that all the components within the processing system, including the processor cores, memory controllers, and peripheral interfaces, are properly reset and brought to a known state during system startup or when a reset condition occurs. The PS Reset IP helps maintain the stability and reliability of the processing system.

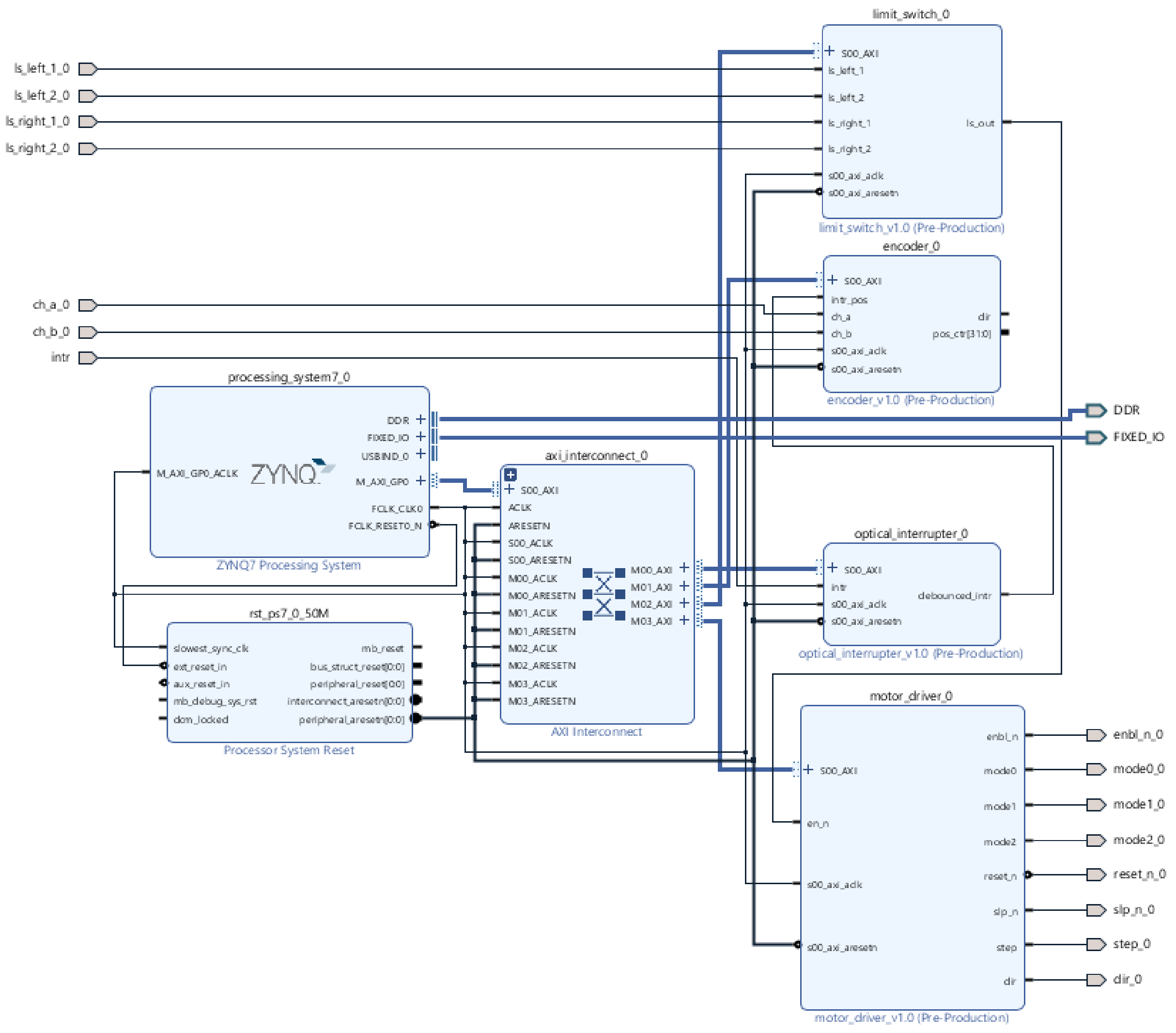


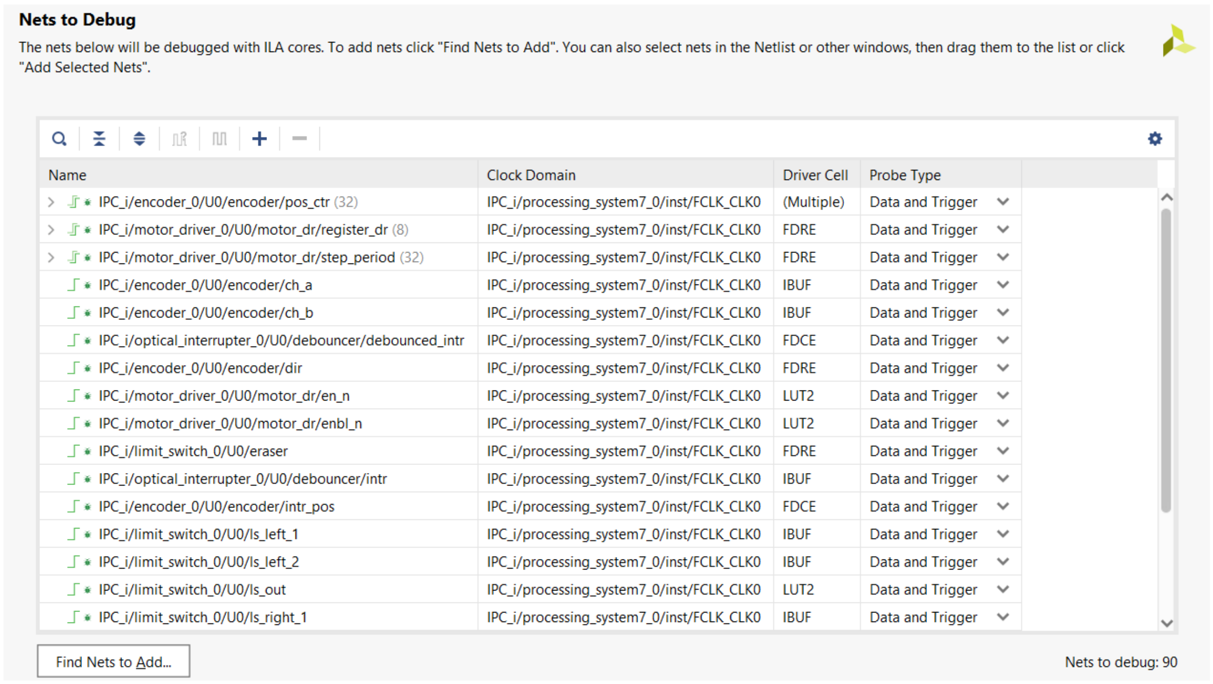
Figure 6.9: Hardware Block Diagram of the Control System

## 6.6 Hardware Debugging with ILA

In order to ensure the correct functionality and identify potential errors in the Programmable Logic (PL) design, the use of debugging techniques is essential. Xilinx Vivado offers a powerful tool called the Integrated Logic Analyzer (ILA) for this purpose. The ILA enables real-time observation and analysis of internal signals within IPs during simulation or hardware debugging, providing detailed insights into their functionality.

The Vivado ILA empowers designers to select specific signals of interest, set triggers, and capture waveforms for analysis. With features like waveform display, navigation, and cross-probing, designers can visually analyze signal behavior and correlate it with other design views. Additionally, the ILA offers signal analysis tools such as measurements, bus decoding, and protocol analysis, facilitating a deeper understanding of the signals.

During the design stage of the IPs, it is crucial to select and mark the desired signals to be observed using the “mark_debug” attribute in the VHDL logic of the control system IPs. This ensures that the relevant signals are available for effective debugging of the PL. Figure 6.10 provides an overview of some of the 90 selected nets designated for debugging purposes.



**Nets to Debug**

The nets below will be debugged with ILA cores. To add nets click "Find Nets to Add". You can also select nets in the Netlist or other windows, then drag them to the list or click "Add Selected Nets".

| Name | Clock Domain | Driver Cell | Probe Type |
|---|---|---|---|
| IPC_i/encoder_0/U0/encoder/pos_ctr (32) | IPC_i/processing_system7_0/inst/FCLK_CLK0 | (Multiple) | Data and Trigger |
| IPC_i/motor_driver_0/U0/motor_dr/register_dr (8) | IPC_i/processing_system7_0/inst/FCLK_CLK0 | FDRE | Data and Trigger |
| IPC_i/motor_driver_0/U0/motor_dr/step_period (32) | IPC_i/processing_system7_0/inst/FCLK_CLK0 | FDRE | Data and Trigger |
| IPC_i/encoder_0/U0/encoder/ch_a | IPC_i/processing_system7_0/inst/FCLK_CLK0 | IBUF | Data and Trigger |
| IPC_i/encoder_0/U0/encoder/ch_b | IPC_i/processing_system7_0/inst/FCLK_CLK0 | IBUF | Data and Trigger |
| IPC_i/optical_interrupter_0/U0/debouncer/debounced_intr | IPC_i/processing_system7_0/inst/FCLK_CLK0 | FDCE | Data and Trigger |
| IPC_i/encoder_0/U0/encoder/dir | IPC_i/processing_system7_0/inst/FCLK_CLK0 | FDRE | Data and Trigger |
| IPC_i/motor_driver_0/U0/motor_dr/en_n | IPC_i/processing_system7_0/inst/FCLK_CLK0 | LUT2 | Data and Trigger |
| IPC_i/motor_driver_0/U0/motor_dr/enbl_n | IPC_i/processing_system7_0/inst/FCLK_CLK0 | LUT2 | Data and Trigger |
| IPC_i/limit_switch_0/U0/eraser | IPC_i/processing_system7_0/inst/FCLK_CLK0 | FDRE | Data and Trigger |
| IPC_i/optical_interrupter_0/U0/debouncer/intr | IPC_i/processing_system7_0/inst/FCLK_CLK0 | IBUF | Data and Trigger |
| IPC_i/encoder_0/U0/encoder/intr_pos | IPC_i/processing_system7_0/inst/FCLK_CLK0 | FDCE | Data and Trigger |
| IPC_i/limit_switch_0/U0/ls_left_1 | IPC_i/processing_system7_0/inst/FCLK_CLK0 | IBUF | Data and Trigger |
| IPC_i/limit_switch_0/U0/ls_left_2 | IPC_i/processing_system7_0/inst/FCLK_CLK0 | IBUF | Data and Trigger |
| IPC_i/limit_switch_0/U0/ls_out | IPC_i/processing_system7_0/inst/FCLK_CLK0 | LUT2 | Data and Trigger |
| IPC_i/limit_switch_0/U0/ls_right_1 | IPC_i/processing_system7_0/inst/FCLK_CLK0 | IBUF | Data and Trigger |

Figure 6.10: HW nets to debug with ILA.

During the debugging process of the inverted control system, errors related to the Rotary Encoder IP were discovered. It was found that the position counter occasionally displayed incorrect values, which were attributed to immediate counter incrementation or decrementation when the pendulum was at rest or moving at a low frequency. Further investigation revealed that the rotary encoder device generated additional bounces at low frequencies.

To address this issue, a specialized debounce filter was implemented in the input signals of the Rotary Encoder IP. This filter effectively mitigated the extra bounces, resulting in more accurate and reliable position readings from the rotary encoder device.

## 6.7 Layout and source consumption

In the hardware design view layout of the control system, as depicted in Figure 6.11, the physical arrangement of the design components and the routing resources employed to establish connections between them can be observed. This view offers a visual representation of how the various elements of the system are positioned and interconnected in a physical sense.

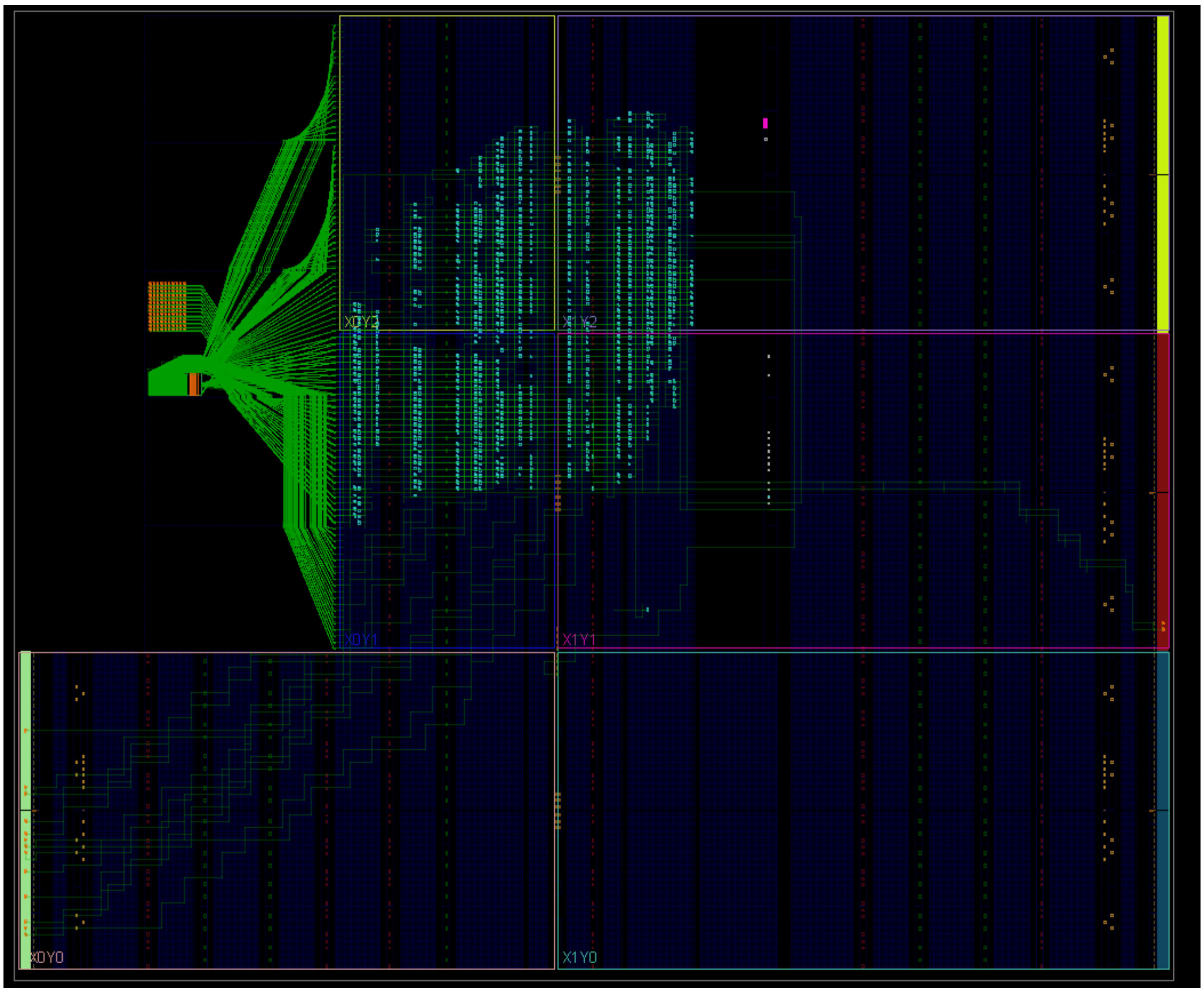
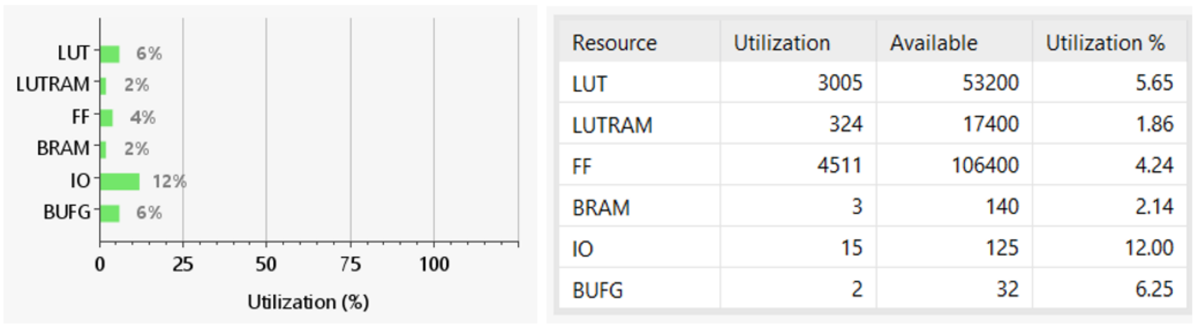


Figure 6.11: Implemented HW Design View Layout

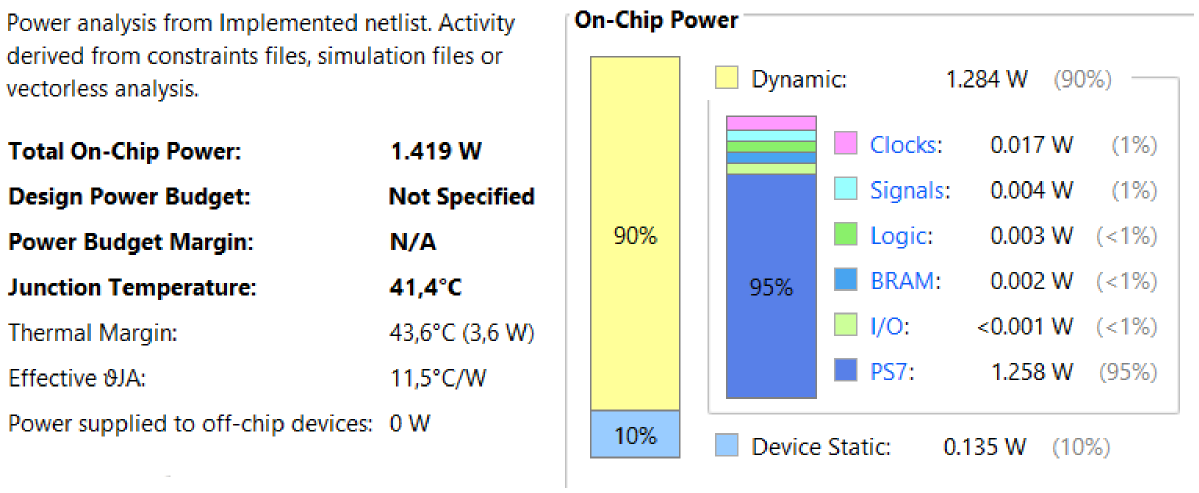
The implemented hardware (HW) design demonstrates relatively low resource utilization, as indicated in Figure 6.12. The utilization percentages of the major resources are as follows: IO utilization is 13%, BUFG (Clock Buffer) and LUT (Look-Up Table) utilization are both 6%, while the least utilized resources are LUTRAM (Look-Up Table RAM) and BRAM (Block RAM) with a utilization of 2%. These utilization percentages provide insights into the efficient allocation and utilization of hardware resources within the design.



| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 3005 | 53200 | 5.65 |
| LUTRAM | 324 | 17400 | 1.86 |
| FF | 4511 | 106400 | 4.24 |
| BRAM | 3 | 140 | 2.14 |
| IO | 15 | 125 | 12.00 |
| BUFG | 2 | 32 | 6.25 |

Figure 6.12: Implemented HW Design Resource Utilization

Upon analyzing the total power consumption of the control system, it is evident that the Processing System (PS7) accounts for the majority of the power consumption, amounting to 95% of the total. In contrast, the Programmable Logic portion consumes only 5% of the total power, with a significant portion attributed to clock-related components. This can be observed in Figure 6.13, which provides a visual representation of the power distribution within the system, highlighting the substantial contribution of clocks to the overall power consumption.



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

| | |
|---|---|
| **Total On-Chip Power:** | **1.419 W** |
| **Design Power Budget:** | **Not Specified** |
| **Power Budget Margin:** | **N/A** |
| **Junction Temperature:** | **41,4°C** |
| Thermal Margin: | 43,6°C (3,6 W) |
| Effective ϑJA: | 11,5°C/W |
| Power supplied to off-chip devices: | 0 W |

**On-Chip Power**

| | | | |
|---|---|---|---|
| 🟨 | Dynamic: | 1.284 W | (90%) |
| 🟪 | Clocks: | 0.017 W | (1%) |
| 🟦 | Signals: | 0.004 W | (1%) |
| 🟩 | Logic: | 0.003 W | (<1%) |
| 🟦 | BRAM: | 0.002 W | (<1%) |
| 🟩 | I/O: | <0.001 W | (<1%) |
| 🟦 | PS7: | 1.258 W | (95%) |
| 🟦 | Device Static: | 0.135 W | (10%) |

Figure 6.13: Implemented HW Design Power Consumption

# Chapter 7

## 7. System Software design

Once the hardware design is completed, it is exported to Xilinx Vitis to create the platform. This platform serves as the foundation for creating the application project, which involves designing system software. The signal received from the previously designed IPs is processed using the processing system within the Pynq-Z1 board. The software design is implemented in the C programming language.

The processor operates at a frequency of 125MHz, and the control period of the cartpole system is set to 40 Hz, as defined in Chapter 4. The acceleration block updates the cart velocity value at a frequency of 10kHz. This ensures that the updates of variables and other iterations occur frequently enough to prevent any detrimental latency or delays in the controller's performance.

In this chapter, the software design for both the PID and LQR control systems will be explored. These control strategies were implemented as separate applications using the same hardware platform. While they share common features such as the system state machine and motor control, they differ in terms of the state observer, error tracking, and controller type.

The system state machine consists of three states: cart centering, pendulum balancing, and stop. In the cart centering state, the cart starts from a random position and is then moved to the center of the shafts, representing the midpoint of its available trajectory. In the pendulum balancing state, the control strategy is applied, and the stepper motor is activated to maintain the pendulum in a balanced position. Finally, in the stop state, the motor is stopped, and this state is activated when the cart reaches the end switches.

Throughout this chapter, a detailed explanation of the functions utilized in the software design for both the PID and LQR control systems will be provided.

## 7.1 Cart centering function

At the beginning of the cartpole control experiment, the cart should be positioned at the center. The cart centering block is designed to move the cart to the midpoint of the shaft, aligning it with the reference of the cart's position. A flow chart illustrating the cart centering function is depicted in Figure 7.1.

At the start of the experiment, the system checks if the left switch has already been pressed. If not, the cart will be moved to the left with maximum velocity until the left end switch is pressed. If the left end switch is already triggered, the cart will be slightly moved to the right. The position of the left switch is taken as 0.48 m, which corresponds to half of the trajectory. Then, the cart is moved to the right with maximum speed until it reaches the center position with a value of zero.

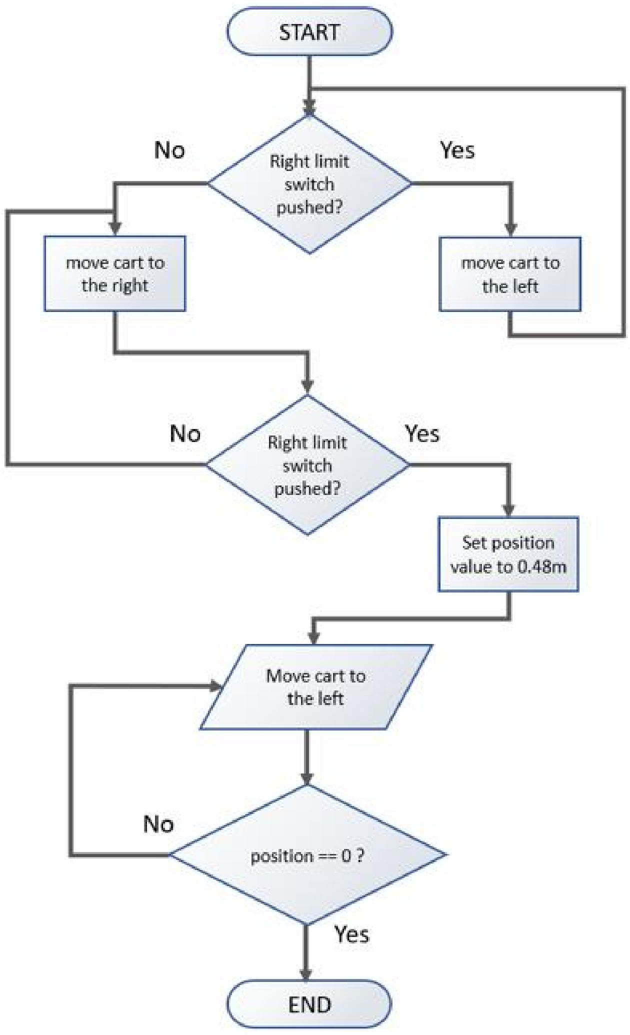


Figure 7.1: Cart centering flowchart

## 7.2 Acceleration Block

The control input of the cartpole control system is acceleration, which is received from the controller block at each control period of 25 milliseconds. Acceleration is used to adjust the motor speed during each velocity modification period, which is set to 100 microseconds (10kHz). This short duration ensures precise and incremental changes in velocity, minimizing discretization errors in the acceleration value and preventing unwanted oscillations in the cart and pendulum movement. By maintaining stability and minimizing errors, the overall performance of the system is improved.

To determine the motor step period, denoted as $steps,$ it is derived based on the cart speed, $\mathrm{v}$, using the following formula:

$$steps = \frac{0.01\pi * R_{pulley}[m] * f_{FPGA}[Hz]}{\mathrm{v}[\mathrm{m/s}] * mode}$$

Here, $f_{FPGA}$ represents the frequency of the FPGA board, which is included in the formula as it is sent to the Stepper Motor Driver IP. The value $0.01\pi$ corresponds to the angle made by one step of the motor, and $mode$ represents the microstepping mode number of the motor as defined in section 2.5.2.

After calculating the steps value, it is checked whether it falls within the range of the maximum and minimum period values. If the calculated value exceeds these limits, it is adjusted to the corresponding maximum or minimum value.

Finally, the position of the cart is incremented or decremented based on the velocity update period $T$ and the direction of movement. If the cart moves right, the direction is set to 1; if it moves left, the direction is set to -1. The formula for calculating the position increment is as follows:

$$\Delta\, position_{cart}[m] \;=\; \frac{direction * 0.01\pi * R_{pulley}[m] * f_{FPGA}[Hz]}{steps\; * mode} * T[s]$$

In the Acceleration Block, the values of all variables are updated during each velocity update period $T$. During other times, the block remains switched off using the usleep() function, which is a sleep function that pauses program execution for a specified amount of time. This allows the system to synchronize the update of variables and control actions with the desired timing and ensures that the block operates according to the defined control strategy.

## 7.3 LQR Controller

The LQR (Linear Quadratic Regulator) controller is designed to balance the pendulum at the upright position. However, it is only activated after the cart centering process is completed, and the pendulum is manually positioned in the upright position. Since the swing-up strategy was not implemented in this thesis, the pendulum is manually set to the upright position.

As the rotary encoder used in the system is relative, it is crucial to establish the correct reference point to accurately define the pendulum's upright position. To achieve this, the initial position of the pendulum is set to 0 when it is in the downward position. Once the angle of the pendulum reaches $\pi$ radians, it is reset to 0, representing the upright position. At this point, the LQR controller is activated to maintain the upright position and balance the pendulum.

The position of the pendulum in radians is determined using the "pos_ctr" signal received from the Rotary Encoder IP, as described in Chapter 6. Considering that the maximum count number corresponding to $2\pi$ is equal to 2399, the following formula can be used to calculate the pendulum position:

$$position_{pendulum}[rad] = \frac{pos\ ctr - 1199}{1200} * \pi$$

This formula normalizes the “pos_ctr” value by subtracting 1199 and dividing it by 1200 to obtain a range of values from $-\pi$ to $\pi$. Multiplying this normalized value by $\pi$ gives the pendulum position in radians. This calculation allows for mapping the rotary encoder counts to the corresponding angular position of the pendulum.

The control input calculations in augmented LQR control strategy requires 5 system states. In addition to the pendulum position, which was discussed earlier, the pendulum velocity needs to be estimated at each control period. The calculation for the pendulum velocity is as follows:

$$velocity_{pendulum}\left[\frac{rad}{s}\right] = \frac{position_{pendulum}[rad] - previous\ position_{pendulum}[rad]}{T_{control}}$$

The cart position and velocity, on the other hand, are calculated in the acceleration block, as mentioned previously. The integral of the cart position error is also calculated, taking into account the reference position of the cart $reference_{cart}$ and its current position $position_{cart}(k)$. The integral is updated using the following equation:

$$q(k) = q(k-1) + (reference_{cart} - position_{cart}(k)) * T_{control}$$

These calculations of system states is crucial for the augmented LQR control strategy to effectively balance the pendulum and maintain stability in the system.

## 7.4 PID controller

The PID (Proportional-Integral-Derivative) controller is chosen as an alternative to the LQR control system and is used to balance the pendulum at the upright position. Similar to the LQR controller, the pendulum needs to be manually positioned in the upright position to initialize the control procedure for the PID controller.

The PID controller implementation involves tracking the error between the reference position and the actual positions of the cart and pendulum. The control input is generated based on proportional, derivative, and integral states.

The error at time instant k is calculated as the difference between the reference position and the measured position:

$$error(k) = reference - position(k)$$

The proportional term $P$ is determined by multiplying the error by the proportional gain $K_P$:

$$P = K_P * error(k)$$

The integral term $I(k)$ takes into account the accumulated error over time. It is calculated by integrating the error with respect to time, multiplied by the integral gain $K_I$ and the control period $T_{control}$:

$$I(k) = I(k-1) + K_I * error(k) * T_{control}$$

Thus control input is the sum of proportional, integral and derivative terms.

The derivative term $D(k)$ considers the rate of change of the error. It is obtained by taking the difference between the current error $error(k)$ and the previous error $error(k-1)$, divided by the control period $T_{control}$, and multiplied by the derivative gain $K_D$:

$$D(k) = K_D * \frac{\left(error(k) - error(k-1)\right)}{T_{control}}$$

By summing the PID terms for both positions, the control input value is obtained, which represents the acceleration applied to the cart. This control input is continuously adjusted by the PID controllers based on the errors in both positions. This allows the PID controllers to effectively balance the pendulum and maintain the desired position.

# Chapter 8

## Obtained Results

Both the LQR and PID control strategies were tested on the real physical system, and their results differed. The augmented LQR controller demonstrated satisfactory performance by successfully maintaining the pendulum in its upright position. However, the PID controller failed to achieve the same stability, resulting in the pendulum eventually falling from its upright position and the cart exhibiting large oscillations.

One of the main reasons for the PID controller's failure could be attributed to the lack of a proper tuning procedure. Since there are differences between the simulated model and the real model, the PID parameters may not have been appropriately adjusted to account for these variations. Furthermore, the PID controller is not considered an optimal control strategy, and it may struggle to handle even small deviations from the linearization point, which is the upright position of the pendulum.

In the case of the augmented LQR controller, both the cart and pendulum positions closely align with their reference position values, as depicted in Figure 8.1. The figure shows all five states of the inverted pendulum control system, including the control input, which is the acceleration. The reference position for the cart is set to 0, representing the desired position of the cart.

It can be observed that the pendulum position has a slight negative offset from the reference value when it is stabilized at the upright position. This offset could be attributed to several factors. Firstly, the usage of a relative rotary encoder instead of an absolute encoder introduces precision errors in the pendulum position detection. Secondly, the optical interrupter used to detect the initial value of the pendulum when it is in the downward position may introduce a delay in counting the angular increment, affecting the accuracy of the position detection. Lastly, the offset can be caused by the fact that the center of gravity of the pendulum is not perfectly centered but slightly shifted to one side.

The cart position reaches its reference value in 10 seconds, while the pendulum angular position stabilizes within less than 1 second, as shown in Figure 8.1. Initially, the pendulum is balanced at the upright position with a significant difference from the cart reference position. The integral of the cart position error contributes to eliminating the cart position deviation, and it reaches a negative steady-state value. Increasing the cost function of the integral state could lead to quicker convergence of the cart position to its reference value, but it may also increase the overshoot in the pendulum angular position, leading to instability.

In addition, the cart exhibits a small, insignificant back-and-forth movement when the pendulum is balanced at the upright position.This movement is insignificant and does not affect the overall stability of the system.

Considering these factors, the augmented LQR controller proves to be a more effective choice for balancing the pendulum in this system. It provides stable control and mitigates the deviations caused by inherent system limitations and measurement errors.

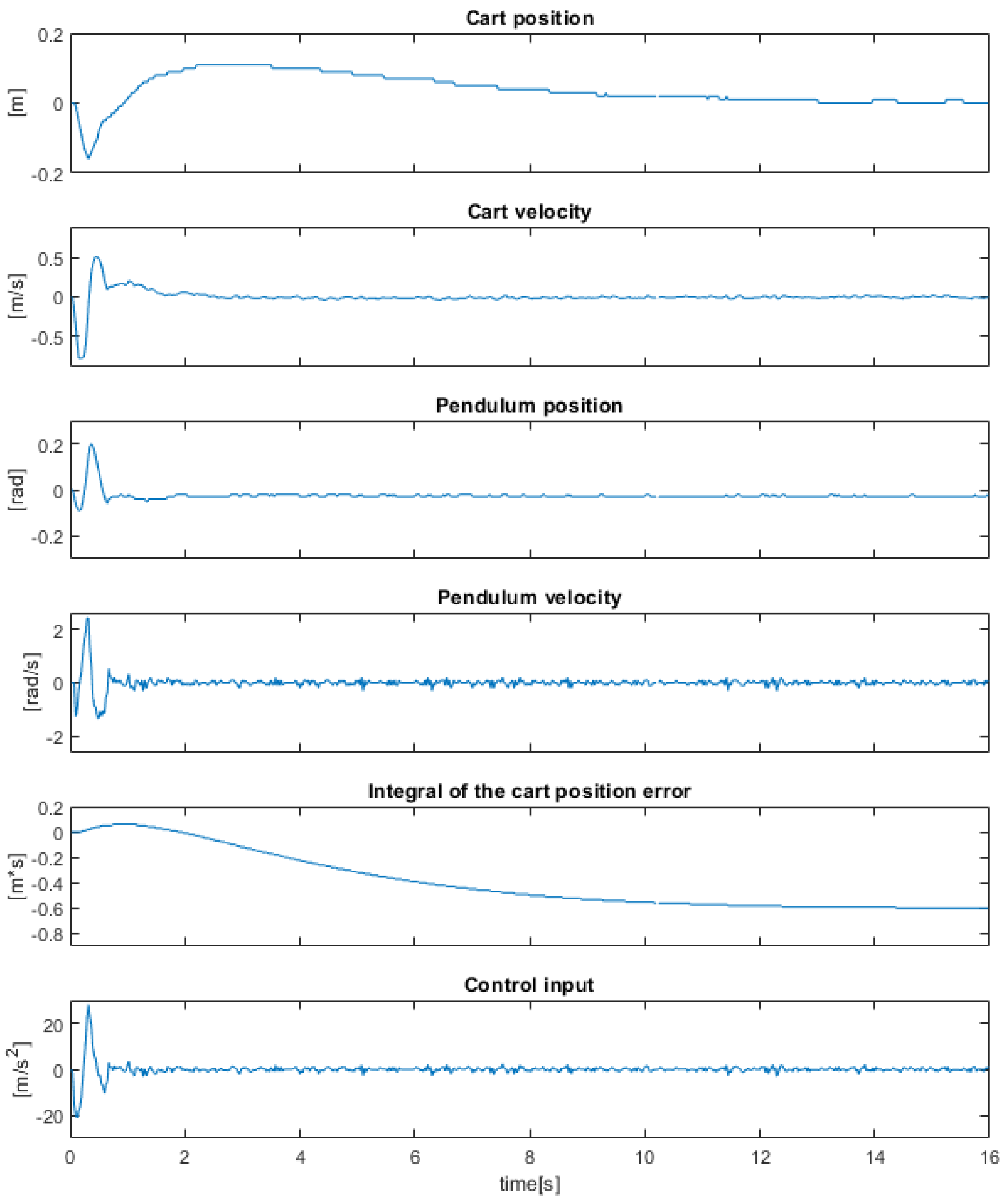


Figure 8.1: Pendulum Balancing with Augmented LQR Controller and Cart Reference Position at 0.

In addition, the LQR system has the capability to attenuate disturbance forces up to a certain limit. When a disturbance force is applied to the pendulum, the cart responds with oscillations of high frequency. However, these oscillations are eventually damped, and the cart settles back to its initial reference position. The system's ability to handle different disturbances and its overall performance can be observed in the video provided at the following link: https://upm365-my.sharepoint.com/:v:/g/personal/a_abdusamadov_alumnos_upm_es/EcCMuMleDXFLnQcBTkaCM5sBFpfxer8MPqbuN4CfVXYdQQ?e=FgvgJK

# Chapter 9

## 9.1 Conclusion

In conclusion, this master's thesis successfully tackled the challenge of balancing the inverted pendulum in the upright position through various approaches. The development of the OpenAI Gym environment provided a reliable platform for simulating the pendulum system, and the Proximal Policy Optimization reinforcement learning technique demonstrated effective control and maintenance of the upright position.

The design and implementation of control strategies, such as PID and LQR, within the Simulink environment showcased their potential in achieving stable pendulum balancing. While the PID controller faced challenges in adapting to the system's nonlinear dynamics and exhibited limitations in tuning, the augmented LQR controller leveraged optimal control theory and successfully addressed system limitations and measurement errors, resulting in robust and precise control performance. The LQR system also demonstrated the capability to attenuate disturbance forces up to a certain limit, ensuring stable performance at steady state.

The integration of a specialized hardware design utilizing the PYNQ-Z1 SoC board showcased the advantages of FPGA boards in real-time control and sensing applications. The ability to filter sensor signals and generate accurate control signals highlighted the practical benefits of using FPGA technology.

Through the exploration and analysis of different control strategies and reinforcement learning algorithms, this thesis provides valuable insights into the field of control systems and their practical implementation. The successful utilization of the OpenAI Gym environment exemplifies the potential of reinforcement learning techniques in solving complex control problems.

Overall, this research contributes to the advancement of control methodologies in the context of the inverted pendulum system, paving the way for further developments and improvements in the field of control systems and robotics.

## 9.2 Future Work

Future work for this thesis can cover various areas of development and improvement. Firstly, there is a need to further investigate the implementation of the PPO reinforcement learning model on the physical pendulum system. This would involve designing a specialized hardware system that seamlessly integrates with the model, allowing for real-time control and improved signal accuracy, thus enhancing pendulum control effectiveness.

Another aspect to consider for future work is the implementation of the "pendulum swings up" strategy. This strategy aims to regulate the pendulum's energy to achieve a swinging-up motion. By incorporating this strategy into the reinforcement learning framework, control performance can be enhanced, leading to a more efficient and dynamic pendulum balance.

Furthermore, extending the reinforcement learning solution to other types of systems, such as double and triple pendulum models available in the OpenAI Gym libraries, would provide valuable insights and broaden the scope of the research.

Additionally, exploring the potential of remote control for the pendulum system would be valuable. This could be achieved by incorporating wireless protocols and introducing new functionalities for wireless communications. Remote control would enable the manipulation and monitoring of the pendulum's behavior from a distance, offering convenience and the ability to control multiple pendulum systems simultaneously or integrate the system into larger networks or systems.

# Bibliography

[1] https://www.gymlibrary.dev/environments/classic_control/
[2] Oscar Mate Castro (2017) “Control de un péndulo invertido mediante FPGA”, Trabajo fin de grado. ETSII.
[3] (2023) “Transmissive Optical Sensor with Phototransistor Output”. VISHAY INTERTECHNOLOGY, INC.
[4] Michael Don and Mark Ilg (2020) “Quadrature decoder design using microcontroller on-chip configurable logic”. CCDC Army Research Laboratory, MD
[5] Snap Action Switch, Omron Electronics Components LLC, p. 95-105
[6] JiangSu WanTai Motor Co., Ltd, “Wantai motor”
[7] “DRV8825 Stepper Motor Driver Carrier, High Current” available at https://www.pololu.com/product/2133
[8] Texas Instruments (2014) “DRV8825 Stepper Motor Controller IC”, Texas Instruments Incorporated.
[9] IEEE. (2000). IEEE Standard VHDL Language Reference Manual
[10] (2017) “PYNQ-Z1 Board Reference Manual”. Digilent, 1300 Henley Court Pullman, WA
[11] (2002) “Designing IP Subsystems Using IP Integrator”. Inc. Xilinx
[12] Patrick Hamill (2014) “A Student’s Guide to Lagrangians and Hamiltonians”, Cambridge University Press, New York
[13] Halvard Hanekam (2021) “Implementation and Control of an Inverted Pendulum on a Cart”, Master’s thesis in Technology and Safety in the High North, Norway, p. 38
[14] "Inverted Pendulum with Animation" Simulink example, the MathWorks Inc.: https://www.mathworks.com/help/simulink/slref/inverted-pendulum-with-animation.html
[15] F.L. Lewis, D. Vrabie, and V.L. Syrmos, Optimal Control, 3rd edition, John Wiley 2013.
[16] https://www.gymlibrary.dev/
[17] https://docs.jupyter.org/en/latest/
[18] Richard S. Sutton and Andrew G. Barto (2014, 2015) “Reinforcement Learning: An Introduction”. A Bradford Book, Cambridge, MA, USA, p.2
[19] A. G. Barto, R. S. Sutton and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-13, no. 5, p. 834-846.
[20] https://www.gymlibrary.dev/environments/classic_control/cart_pole/
[21] https://openai.com/blog/openai-baselines-ppo/#ppo
[22] Francisco Javier Collado Valero (2018) “Comparación entre controlador SW y HW de un péndulo inverso sobre FPGA”, Trabajo fin de grado. ETSII.
[23] Sarah L. Harris, David Harris, 4 - Hardware Description Languages, Editor(s): Sarah L. Harris, David Harris, Digital Design and Computer Architecture, Morgan Kaufmann, 2022, p. 170-235