Pontificia Universidad Católica de Chile

"Design, implementation of an electronic circuit and control strategy for the readjustment of an inverted pendulum, in the Control Laboratory"

Technical Report

Jay March

Texas A&M University at College Station

630004322

7 - 17 - 24

Abstract

This report details the development and implementation of a PID controller for stabilizing an inverted pendulum on a cart. The project encompassed several stages which included circuit design, microcontroller setup, schematic and PCB design, PCB implementation, and the integration with MATLAB & Simulink for PID control. The initial steps of the project focused on testing the preexisting components from the system such as the potentiometers and motors and their behavior.

A circuit was built on a breadboard to adjust the voltage ranges, a combination of amplifiers, unity gain differential amps, and subtractor amps were used to accomplish this. These signals were then able to be transmitted to an Arduino UNO microcontroller which was programmed to read ADC values from the potentiometers along with interpolation and online calibration for enhanced accuracy.

The circuit was then transferred to an online schematic using EASYEDA, ensuring all connections were verified before designing the PCB. After printing and soldering the PCB, continuity checks, and sensor connections ensured signal accuracy. MATLAB & Simulink facilitated testing Arduino communication, interpolating variables, and simulating pendulum dynamics. The PID controllers were modeled in Simulink and transitioned to MATLAB scripts for real-world applications.

The final stage involved implementing and fine-tuning the PID controller on an Arduino MEGA 2560, optimizing for speed and efficiency, and testing various gain settings to achieve optimal pendulum stabilization. The results demonstrated effective stabilization of the inverted pendulum, highlighting the potential of the PID controller in control systems and robotics applications

1. Introduction

1.1 Motivation:

The control and stabilization of an inverted pendulum is a classic control problem in the field of robotics. It is often used to demonstrate the effectiveness of various control strategies. The system's

inherent instability and nonlinearity make it an excellent testbed for classical control methods, such as the Proportional-Integral-Derivative (PID) controller. Successfully stabilizing an inverted pendulum has practical implications in various fields, including robotics, aerospace, and industrial automation. This project is motivated by the challenge of designing a robust and efficient PID controller to achieve precise control of an inverted pendulum system mounted on a cart.

1.2 Problem Statement:

The system is inherently unstable, with the pendulum tending to fall away from its equilibrium position. This requires continuous and precise control inputs to maintain balance. The project involves multiple stages, including circuit design and testing, microcontroller setup, PCB design and implementation, and integration with MATLAB & Simulink for controller tuning. The primary challenge is to develop a control system that can quickly and accurately respond to disturbances, maintaining the pendulum in its upright position.

1.3 Proposal:

This project proposes a comprehensive approach to solving the inverted pendulum stabilization problem through the following stages:

1. **Circuit Design and Testing**: Developing a reliable circuit to measure and output the pendulum's angle and position movement.
2. **Microcontroller Setup for Arduino UNO**: Programming the microcontroller to read sensor data and execute appropriate interpolation and calibration for angle and position.
3. **Circuit Schematic and PCB Design**: Creating a detailed circuit schematic and designing a PCB to ensure robust and accurate signal processing.
4. **PCB Implementation**: Assembling and testing the PCB to verify proper functionality and signal integrity.
5. **MATLAB & Simulink Package Development**: Utilizing MATLAB & Simulink for modeling the system, tuning the PID controller, and simulating the control strategy.
6. **Arduino PID Control using Arduino Mega 2560**: Implementing and fine-tuning the PID controller on an Arduino platform, optimizing the control response for real-time operation.

2. Materials and Methods

2.1 Methods:

2.1.1 Differential Equations: The dynamics of the inverted pendulum are described by a set of nonlinear differential equations. These equations account for the forces and torques acting on the pendulum and the cart. The equations below have been derived from the pendulum system and provide MATLAB and the PID controllers with a foundation for analyzing the system's behavior.

Nonlinear Differential Equations

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos\theta - ml\dot{\theta}^2 sin\theta = F \sim \textbf{(Equation 1)}$$

$$(I + ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}cos\theta \sim \textbf{(Equation 2)}$$

Linearized Equations of Motion

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\emptyset} = u \sim \textbf{(Equation 3)}$$

$$(I + ml^2)\ddot{\emptyset} - mgl\emptyset = ml\ddot{x} \sim \textbf{(Equation 4)}$$

2.1.2 Transfer Functions: The transfer functions of the inverted pendulum system are obtained by linearizing the differential equations around the respective setpoints (*Angle = ~0° / Position = ~0.1)*. These transfer functions are used to design and analyze the PID controller, providing insight into the system's stability and response characteristics.

Transfer Function of the Pendulum's Angle

$$P_{pend}(s) = \frac{\emptyset(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \quad \left[\frac{rad}{N}\right] \sim \textbf{(Equation 5)}$$

Transfer Function of the Cart Position

$$P_{cart(s)} = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \quad \left[\frac{m}{N}\right] \sim \textbf{(Equation 6)}$$

Continuous-time Transfer Functions

$$\emptyset = \frac{0.000229s}{7.523\times10^{-5}s^3 + 1.501\times10^{-5}s^2 - 0.001566s - 0.0002246} \sim \textbf{(Equation 7)}$$

$$X = \frac{0.0001501s^2 - 0.002246}{7.523\times10^{-5}s^4 + 1.501\times10^{-5}s^3 - 0.001566s^2 - 0.0002246} \sim \textbf{(Equation 8)}$$

2.1.3 MATLAB / Simulink Simscape Code Implementation: A MATLAB .m script was initially used to test receiving information from the Arduino and interpolating a test PWM value which was sent back to the microcontroller. Once this communication was established, MATLAB Simulink was used to model the inverted pendulum system and simulate the control strategy. In more detail the simscape multibody extension was used to re-create a real-life simulation with accurate parameters, including weight, length and other variables. The simulation was used to test the behavior of the pendulum under different strategies and tune the PID controllers for the angle and position.

2.1.4 Arduino Code Implementation: An Arduino UNO microcontroller was used to process the ADC measurements generated by the ADS1015 sensor; these readings were enhanced by the ADS1015 sensor which returned 12-bit resolution values with a +/- 6.144 V tolerance. Lastly, to prioritize a faster response for the PWM signal to be sent to the motor, the PID control system was transferred over to the microcontroller. An Arduino MEGA 2560 microcontroller was utilized to handle the extra processing needed to quickly receive and send information to the motor. Furthermore, the Arduino script utilized a 3-point interpolation calibration to adjust the ADC values to reliable angle (-30° - 30°) and position (-30 - 30) values which could be fed into the PID controllers. An online calibration was implemented to account

for the fluctuating voltage levels, an ISR interrupt function was used to stop the program and allowed calibration for the angle of the pendulum.

Online Calibration Algorithm

1) Press calibration mode (ISR connected to D2).

2) Program enters calibration mode and waits ready to read ADC values (LED is used to indicate calibration mode has been entered).

3) Position the pendulum to the left position and hold the **leftPos** switch button until the LED blinks indicating that the required number of readings has been acquired.

4) Repeat Step 3 for **middlePos** and **rightPos**.

5) Once the required number of readings for each position has been collected the program will then compute the average of each position and set the new constants for the 3-point interpolation calibration.

END

2.2 Hardware Description:

2.2.1 System IOs: The system relies on 2 primary input devices: angle and position potentiometers, each with 3 inputs. **Angle Potentiometers:** 3 prongs (power, ground, and signal), this provides the signal for the angle (*exact voltage range)*. **Position Potentiometer:** 3 prongs (power, ground, and signal), this provides the signal for the position *(exact voltage range)*. Additionally, it utilizes three power sources (+12V, -12V, and 5V) to ensure the proper functioning of all components. **+12V** and **-12V** sources are used to power the operational amplifiers and the motor driver, furthermore, the **+5V** source is used to power the potentiometers, motor driver, and ADC sensor. Four switch button inputs are used for the calibration process. The function of these button inputs is to collect data for the respective positions during the calibration process (**leftPos, middlePos, rightPos**) and the **Calibration Mode** button enters the calibration mode, triggering an ISR (Interrupt Service Routine) interrupt in the code. These inputs are integral for the data acquisition and control processes necessary for maintaining the balance and movement of the inverted pendulum system.

2.2.2 Electrical: The electrical components of the system include a Model 138 Angle Potentiometer (1 kΩ - 50 kΩ) and a Spectra Precision Position Potentiometer (1 kΩ - 50 kΩ) which are used to measure the angle and position of the pendulum / cart. The Faulhaber DC Mini motor - 23/1 - 3,7;1 is used to move the cart to stabilize the pendulum. It is given an input torque from the motor driver. The Sparkfun TB6612FNG Motor Driver is an electrical component used for actuating the motor, it receives a PWM value from the Arduino then sent in an appropriate signal to the previously mentioned DC motor. The Adafruit ADS1015 12-bit resolution Sensor is an analog to digital converter; it allows for the potentiometer's readings to be converted from a voltage range to an ADC reading (*Range: 0 – 1900)*. The Arduino UNO and Arduino MEGA 2560 are the primary microcontrollers used for this system. They are used to handle the pendulum signals and process the information through an Arduino script which

returns a respective PWM signal to balance the pendulum. Lastly a custom PCB was designed to facilitate the inputs and generate reliable outputs to control the system. Various design rules were used to ensure proper signal processing throughout the circuit. The power lines which are connected via the 2 pin terminals on the PCB had a line width of 1.2mm, this provides ample space for the transfer of electrons. Furthermore, 0.8mm is used for data lines, while still offering a large enough width for adequate movement, the data lines were made smaller to accommodate the restricted space on the PCB. Furthermore, 90 degree turns in the connections were avoided in order to ensure smooth pathways without much turbulence.

2.2.3 Operational Amplifiers: A series of operational amplifiers were utilized to adjust the voltage range of the pendulum's potentiometers. The initial range of the angle's potentiometer varied between 2V at the left most position (-30 degrees) and 3V at the right most position (30 degrees). This range had to be shifted to 0-5V to allow for a broader range and more measurable values. This is done by utilizing various buffer amps, a subtractor circuit, and an amplifier. Buffer amps are used to receive the signal from the potentiometer to prevent one stage's impedance load to the posterior stage impedance, this avoids any signal loss. Furthermore, a voltage divider was used to receive a constant 2V source from the 5V source, this will be used for the subtractor circuit. A unity gain differential amplifier was chosen to perform the subtraction necessary to achieve the 0-1V range from 2-3V. Lastly, this signal was amplified with a gain of 5 resulting in the 0-5V range. No filter was necessary as a clean signal was provided by the last 2 buffer amps which included a 100 nF capacitor to prevent any signal disturbance to the microcontroller.

2.2.4 Mechanical: The mechanical components of the system consist of a cart (M = 486 grams) that moves along a track (T_L = 86.74 cm) and an inverted pendulum (m = 211 grams / P_L = 60.90 cm) attached to the cart. The cart is driven by a 12 V DC motor, which is controlled based on the feedback from the pendulum's angle and position.

3. Results

3.1 MATLAB / SIMULINK:

3.1.1 Simscape Simulation Results: The simscape simulation gave us a better understanding of the pendulum's expected behavior under different distortions. The simulation was created using the pendulum's real-life dimensions and weight.

3.1.2 Arduino and MATLAB communication: Arduino and MATLAB communication was initially established in order to allow MATLAB to compute the necessary PID control signal for the microcontroller. Variables were read by the ADS1015 sensor and sent to the Arduino UNO, then after the necessary interpolation was performed were written to the serial monitor which was then read by the MATLAB script. The script performed a simple PD control system which sent back an interpolated PWM signal to the microcontroller which would finally send it to the motor driver. However, after further investigation into the optimization of the program it was found that the functions used by MATLAB and Arduino to ensure the serial monitor communication were considerably slowing down the flow of the program. Therefore, a solution for utilizing a new microcontroller, the Arduino MEGA 2560, was implemented with its own PID controller. This ensured the fastest response for the inputs / outputs.

3.2 Arduino Code:

3.2.1 Analog-Digital Converter (ADC): After initial testing with the Arduino UNO's built in ADC sensor a more suitable and accurate sensor was needed in order to ensure better signal processing for the potentiometers. Therefore, the solution was to incorporate an Adafruit ADS1015 12-bit resolution ADC sensor. The ADS1015 boasted a stronger resolution and most importantly a larger input tolerance which would be able to properly handle the 0-5V voltage range of the angle's potentiometer. The range of readings recorded by the ADC sensor were approximately 16 - 1820, for the angle potentiometer, and 202 - 1653, for the position potentiometer. The gain of the sensor was initialized as ⅔ in the program (ads1015.setGain(GAIN_TWOTHIRDS)) this allowed for a tolerance of +/- 6.144V. Hence, the reason why the ADC measurements did not reach the maximum of 2048 when at the highest position.

3.2.2 Online Calibration: The online calibration was created in order to allow for a faster on-board calibration that did not require the process of measuring the ADC values and hard coding them into the program. The online calibration utilized an ISR in order to prioritize the calibration in the program. This program allowed for more accurate results when the pendulum seemed to be slightly off the center. A recalibration was able to be performed at any point during the testing.

3.2.3 Interpolation: The 3-point interpolation calibration was used various times throughout the program to convert the ADC measurements into angle and position values which were used for inputs to the PID controller. The interpolation formula utilized 3 measurement points which corresponded to their calibration values in the form (x1, y1). X1 being the measured variable and Y1 having the interpolated constant. The formula for the interpolation is shown below. *Insert interpolation formula*

3.2.4 PID Controllers: The angle PID controller was optimized for the response given by the system and the motors.

Angle PID Controller Gains: Kp = 80, Ki = 0, Kd = 40

Position PID Controller Gains: Kp = 10, Ki = 0, Kd = 1

3.3 Electronic Circuit:

3.3.1 Schematic: Below entails the schematic diagram of the PCB, this includes the op amp circuit, buttons, sensors, and terminal / headers.

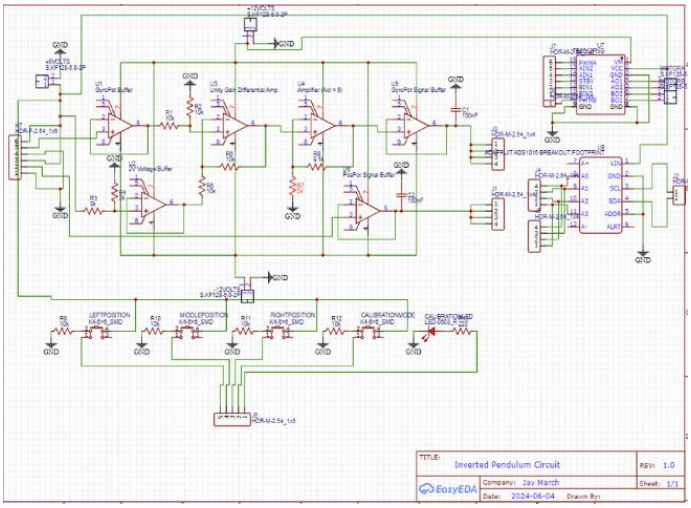

Figure 1: Circuit schematic of PCB for signal processing and motor control

3.3.2 PCB: Below is the schematic of the PCB with the appropriate connections following the design rules. The placement of the op amps, which are aligned in a series, was done so to be able to incorporate the +12 / -12V connections appropriately. This organization allows for a more structured PCB design allowing for more space and ensuring proper connections. The 2 prior iterations to this circuit followed a similar pattern however were not completed because of faulty connections, errors with ground connections / continuity, and poor component placement. The alignment of the op amps in the final iteration was a result of failure with prior alignments, the multiple iterations of the PCB allowed for mistakes to be fixed. The PCB ended up being organized, following the design rules of 1.2mm for power lines and 0.8mm for data lines. Furthermore, the reduction of vias was vital in keeping the PCB simple and efficient, as well as keeping the majority of connections on the ground layer of the PCB.

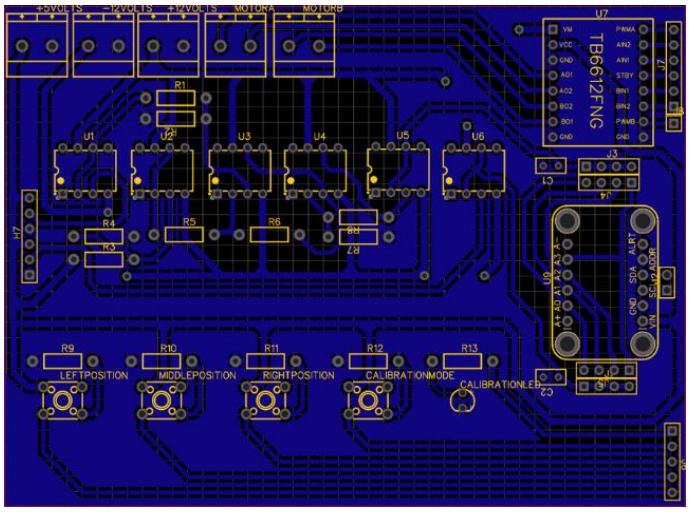

Figure 2: Custom PCB final design

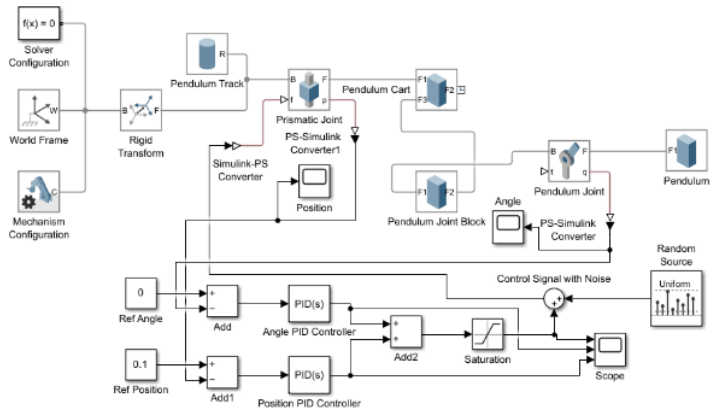

Figure 3: Schematic representation of MATLAB Simulink Simscape Multibody Simulation

3.4 Experimental Results:

3.4.1 PID Tuning: The finals results for the tuned PID values had **Kp = 80**, **Ki = 0**, **Kd = 40**. These values returned the most efficient result with adequate response time and minimal overshot. The Ki was kept as 0 due to the lack of difference when given a value greater than 0. And if statement was used in the code to minimize the cumulative error over time (Ki) as it would increase exponentially as time passes. Therefore, the controller was better utilized whenever Ki was set to either 0 or a very small value.

3.4.2 Microcontroller Change: The decision to switch from the Arduino UNO to the Arduino MEGA 2560 was made in order to increase the performance of the Arduino script as well as being able to incorporate the PID control system in the Arduino script rather than the MATLAB script. After noticing that the performance of the program was drastically reduced due to the function initiating Arduino & MATLAB communication, the decision was made to facilitate the control system within the Arduino Code. In order to accomplish this a faster microcontroller needs to be used that contains better processing power. Therefore, the switch was made to the MEGA 2560. This cleans up the overall program as less applications need to be used, and the system can run without needing to be connected to a laptop. These changes allowed for a faster program which leads to increased efficiency and speed.

3.4.3 Limitations of the System: There are a few limitations in the system that affected the progress of the control system. One of the limitations lies in the pendulum itself, due to its length, its center of mass is too high up. This makes it where the pendulum is impossible to bring to the set point of 0 when it is at position 30 / -30. Therefore, whenever testing the system it must start from position 0. Furthermore, the gears of the motor were severely worn out, along with small cracks along the track. This made it to where when the motor was quickly switching directions or speeds the motor would come off of the track and the pendulum would fall off. This prompts the testing to end until the pendulum is put back into its position. Lastly, due to the mechanical limitations of the pendulum system; old potentiometers, weak motor, and loose track, makes it more difficult to have an aggressive and fast response from the control system as the cart cannot handle this.

3.4.4 Issues: A few of the issues encountered while designing the circuit board for the system was at the very beginning when acquiring the signals from the potentiometers the voltage and implementing them

into the op amp circuit it seemed to lose voltage making the voltage range inaccurate. This was solved by the use of buffer amps to copy the voltage and allow for more accurate signals. This prevents one stage's impedance load to the posterior stage impedance; this avoids any signal loss. Another issue was while designing the PCB, multiple iterations of the board were required to ensure optimal signal processing. The connections had to be reworked until a more optimal connection route was achieved. Furthermore, design rules for the connections were set in place in order to account for proper signals to be transmitted through the PCB. Furthermore, the final copy of the PCB had issues with the copper mesh layers, which acted as the universal ground. Due to the large number of connections certain grounds were cut off from each other and not connected. This was fixed by the use of soldering jumpers to connect these grounds.

3.4.5 Non-Solved Issues: The system is capable of stabilizing the angle of the pendulum (Angle PID Controller); however, the cart seems to move in either direction with approximately constant velocity. This makes the control strategy more difficult as it requires an additional Position PID controller to consider the bounds of the track. The simple tap of the pendulum whenever it reaches the outer bounds returns the cart back to normal position however it simply oscillates between the edges of the track. The correct implementation of the Position PID controller is the last task needed for a fully working system.