TEXAS A&M UNIVERSITY
Higher Education Center
at McAllen

“Autocorrecting Automobile Dynamic Pendulum System”

MXET 375 – Applied DynamicSystems

FALL 2024

Prof. Aldo J Munoz-Vazquez, D.Sc.

Final Project

Team Members
Mark Perez
Jacob Reyes
Ricardo Rodriguez

McAllen, Texas, United States December 10, 2024

Modeling, Control and Simulation of an Autocorrecting Automobile Dynamic Pendulum System

Summary

The Autocorrecting Automobile Dynamic Pendulum System or vehicle pendulum, is a autocorrecting robot that stabilizes a fixed pendulum of mass by changing the acceleration of the vehicle in the direction the pendulum is falling. The main objective of this project is to design and develop a vehicle that will house a pendulum, as well as a controller that will stabilize the pendulum using motors to accelerate the vehicle.

Contents

INTRODUCTION

In this particular project, we were tasked with the development of an autocorrecting automobile dynamic pendulum system, in which we, jointly, worked on the modeling, control system, simulation, and physical modeling of the robotic vehicle system. In the current industry settings, control systems and automation are concepts that are being widely sought after, and by developing small-scale project such as the aforementioned project we have created, we are learning how to apply the knowledge learned via the concepts taught in the lectures and laboratory assignment into projects that mirror what can be used in today's industry. In this particular project, we used Simulink via Matlab in order to develop the control system and the Creo 3D modeling software in order to model and 3D print the pendulum used in this project.

In seeing how today's industry is gearing towards automations using control systems, we can view how impactful the creation of a project like this can be since it allows for us to mirror systems that can be found in today's industry while still gaining our undergraduate degrees. Most companies are moving towards having all their processes automated, for example, the Tesla automobile assembly line is fully automated through the use of robots. Since each car has to leave the assembly line with no mileage, the robots are automated to control the processes of the assembly line. Seeing that influential companies around the world are moving their companies from human operations to automated control systems, we can see that learning how to develop even small scale automation systems can be key to gaining knowledge of how the concepts we are learning in our courses can be applied to what is currently being used in the industry.

In the following sections found below, we will discuss each element of the project, which includes the software setup, the hardware components, testing, and the overall results of the project. The software setup can be broken down into its various components, which include the potentiometer setup, the MatLab function, and the PWM configuration. In the hardware section, we will discuss all the hardware elements that were used in the creation of this project as well as how they were integrated with each other in order to create the final product. In the testing and results section, we will discuss the various features that had to be tested in order to optimize the workings of the robot as well as how the final product worked. Lastly, we have also included a portion in this lab report discussing any constraints that had to be taken into account when designing this project as well as any improvements that could be added in the future to increase the efficiency and functionality of the robotic self-balancing pendulum system.

ROBOT/CONTROL DEVELOPMENT

In this section of the report, we will be analysing all our configurations made to the autocorrecting automobile dynamic pendulum system. These configurations will delve deeper in the software development, the hardware development, as well as the testing process. We will also include some improvements that will better facilitate this dynamic system for future iterations.

I. Software/Program Development

We will be examining the program used to allow our car to be fully automated to autocorrect the pendulum on top in such a way that it would allow said pendulum to stay vertically upwards at a perfect 90 degrees angle from the horizontal line of our car. We will be dividing this software development into three major components. To first get us started, we will firstly be creating a Simulink Project from our MatLab Software.

I.1 Potentiometer Setup

We firstly generated an analog input form our common Arduino library. This analog input will work as a potentiometer, which will then determine the position of the pendulum at any given moment. To do this we will need to configure the analog voltage given from the potentiometer to a value that is better represented in radians. To do this we will need to configure the following equation:

$$Radians = AI \times 0.225 \times \frac{\pi}{180}$$

Equation 1: Equation to get radians from analog voltage signals.

If the analog signal times the constant gain of 0.225 equals a negative value, then we must set a sum for that negative value of 139.5. This will create Equation 2 to get the radiants of a negative value.

$$Radians = ((AI \times 0.225) - 139.5) \times \frac{\pi}{180}$$

Equation 2: Equation to get radians from negative analog voltage signals.

We follow these two equations to configure the potentiometer as origin in the middle of an imaginary coordinate system. We first examine that the variable *AI* is actually the voltage analog input and the *Radiants* is the output of said calculation in the form of radiants. The origin of the pendulum would be perfectly vertical in a 90 degree angle due to the horizontal plane; this means that the middle voltage of the potentiometer will result in a value of 0 radians in this position.

This configuration is perfectly reflected in the following figure down below of our report. This figure examines how exactly we will configure these formulas in our Simulink Modeling. This will be accomplished with gains, constants, and sum functions as well as a much needed conversion to a double integer value.

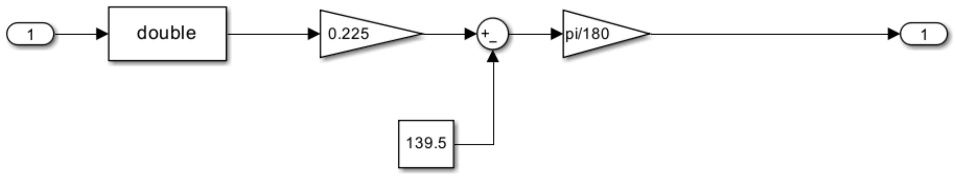
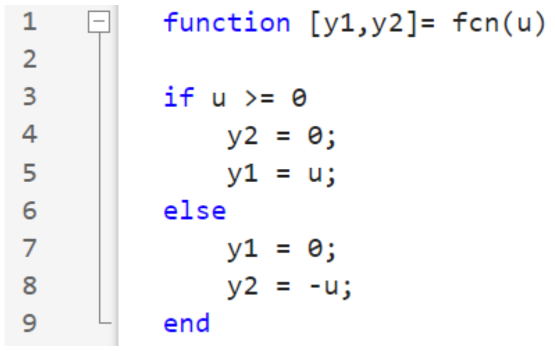


Figure 1: Simulink Model of Potentiometer Setup to Radians Location.

I.2 MatLab Function

The next component of our software development would be our actual MatLab function. This function will decide if the robot will drive forward or backwards depending on the position of the pendulum when the potentiometer leaves the 90 degree angle. To do this we shall create a function that will take the output of our potentiometer and connect it directly to an output_1 while firstly setting output_2 to zero if the potentiometer output is positive. However, if the output of the potentiometer is negative, it will cause us to set the output of the negative potentiometer value into output_2 after zeroing out output_1. This special configuration will work as a special function that will allow the car to go forward with a specific intensity of the output of the potentiometer as well as backwards with the same principle. Effectively creating a similarity to a bang bang controller.

This MatLab Function will help us generate the outputs for the PWM signal that will be fed to the H-Bridge and subsequently the engines. This function can be seen in Figure 2.


```
function [y1,y2]= fcn(u)

if u >= 0
    y2 = 0;
    y1 = u;
else
    y1 = 0;
    y2 = -u;
end
```

Figure 2: MatLab Function for Pendulum Car.

I.3 PWM Output Configuration

This final component deals with the final connection in our Simulink Model to effectively connect the MatLab Function to our PWM output pins of our Arduino Mega 2560. Firstly, we will generate two PWM outputs from our Arduino Common Libraries. These two outputs will be connected directly to the outputs of the MatLab Function Block. Meanwhile, we will connect the output of our potentiometer setup to the input of our MatLab Function with a gain of 200. This gain will work to increase our small radiant values to a value where the PWM will effectively move without problems.

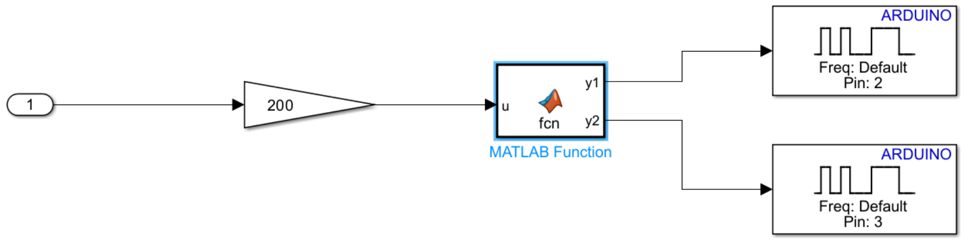


Figure 3: Simulink Models Connections.

As we can see from the figure above, we have completed the full configuration for this project with the heavy influence of Aldo Muñoz Vasquez. In the figure above, the value "1" represents the output of our potentiometer configuration seen in subsection I.1 after it has gone through an extensive PID controller to allow for a more smoother and rapid transition of radiants. This PID will hold uncommon characteristics due to the need for real time fast configurations. This means the PID controller would have a proportional value of 1000, an integral value of 1, an derivative value of 1, and a filter coefficient of 100. This contributed highly to our software success for when to choose the correct bang bang configuration.

With the software done, we will be able to next examine our hardware components and our connectivity to each hardware component in correlation to software.

II. Hardware Development

Once the software portion of the project had been completed, we shifted our focus onto the hardware development portion, which would work hand-in-hand with our program to create the autocorrecting car pendulum robot.

Before we started assembling the robot we needed to attach some wires to the potentiometer and to our DC motors. We soldered three wires to our potentiometer, these being power, ground and the wiper. All three of these wires will be connected to our arduino. We then soldered two wires to each DC motor, these wires will be connected to output 1 of our motor driver, they will supply the voltage and current needed to turn the motors. Next, we conducted several tests to verify the functionality of the program. We conducted these tests by wiring up the arduino, and our two motors to the motor driver. We supplied the motor diver with 9 volts from the outlet. It was necessary to use a motor diver and an outside power source since the arduino would not be able to supply enough power to turn both motors. Once we verified that the motors would spin in accordance with the input from the potentiometer, we were able to start assembly.

When we received the robot, it was completely disassembled (*Figure 4*). We began assembly by attaching the motors to the gear boxes and fixing the wheels to the output of the gearbox. We then attached the gearbox assembly to the base of the robot, but before we could do that we had to prep the surface of the robot. We prepped the surface by cleaning off any old adhesive with acetone and scraping the surface with high grit sandpaper. The gearbox assembly was then adhered to the base using industrial adhesive and left to dry. In the meantime we searched for options to use for our pendulum. We decided to use a CAD software to 3D print a pendulum that would attach to our potentiometer. One group member designed the pendulum using Creo, and sent the file to get printed by our lab technician. The rest of the group members continued working on assembling the robot.

The next step of assembly was securing the arduino and motor driver, as well as routing all the wires to their correct positions. We mounted the motor diver onto the base by drilling holes in the base, and threading Zip Ties through the included holes in the motor driver. We installed the motor driver on the underside of the robot to maximize the space on top of the robot. We then created a platform for the arduino out of cardboard and popsicle sticks and secured it to the base of the robot. Finally we created a mount for the potentiometer that would support the potentiometer itself and the pendulum. This was the most difficult step while assembling the robot, since we were limited on materials. We decided to cut several squares of cardboard and cut a slot in them for the potentiometer to sit in, we then glued the pieces together and created a cube of cardboard with an insert for the potentiometer. We then collected the pendulum from the 3D printer and friction fitted the potentiometer arm into a slot in the pendulum. We then glued the pendulum assembly onto the topside of the robot's base, we made sure that the pendulum assembly sat in the center of the robot. This was the end of the assembly process (*Figure 5*).

III. Testing & Results

Once the program and hardware configurations had been completed, we, then, moved to the testing phase of the project's development, and the main region of testing that was done was to test where the placement of the components would allow for the best motion of the robotic vehicle. Other areas of testing included altering the values within our Simulink Proportional Integral Derivative (PID) and altering the length of the pendulum being used in our project.

The first set of testing that was done was testing the placement of the main hardware components on our robotic vehicle board. There were many aspects that went into the decisions on the placement of the components, which included placing the microcontroller, potentiometer, and h-bridge in locations where they would fit on the board without causing any issues with any of the hardware components for the robot itself. Examples of this included altering the orientation of the microcontroller in order for the microcontroller to not have any issues such as contact with the wheels of the vehicle especially since the microcontroller will have a AC power supply cord connected straight to the board, and positioning the h-bridge on the underside of the vehicle where connections to the motors would be easier. One of the biggest aspects in terms of the placement of the hardware components on the robotic vehicle was weight distribution. Once the components had be secured to the chassis of the robotic vehicle, we noticed that since we had placed most of the weight on the rear end of the chassis, whenever the pendulum veered towards the rear of the vehicle, the motors did not provide enough torque to move the vehicle fast enough to respond to the pendulums falling motion.

Another aspect of testing was testing the Proportional Integral Derivative (PID) portion of the program that was developed in Simulink, and in testing the values that were set for the four factors mentioned above, we noticed that we could have the car move more precisely based on the potentiometers movement via the pendulum. The values stated above were the finalized values after testing, but one thing that can be stated about those chosen values are not exactly realistic values that can be observed naturally without the use of simulation software such as Simulink.

Lastly, we tested how the length of the pendulum affected the response of our robot, which we did by first testing the robotic vehicle with the full length of 12 inches, and once we saw that the response of the robotic vehicle was not up to our expectations, we jointly made the decision to decrease the length in order to lessen the mass of the rod itself as well as the effect is has when swaying back and forth. After testing the pendulum with the new shortened length, we noticed that there was a positive response with the movement of the pendulum car. One other addition that we could have made had we known that weight would be one of the biggest factors within the workings of our project would be to make the pendulum rod hollow using our Creo 3D modeling software in order to lessen the weight of the pendulum.

Overall, the resulting robotic vehicle was a subpar attempt at creating a pendulum car that would self balance the pendulum, but if we were to make the alterations mentioned in this report, we would be able to have a better response with the robot.

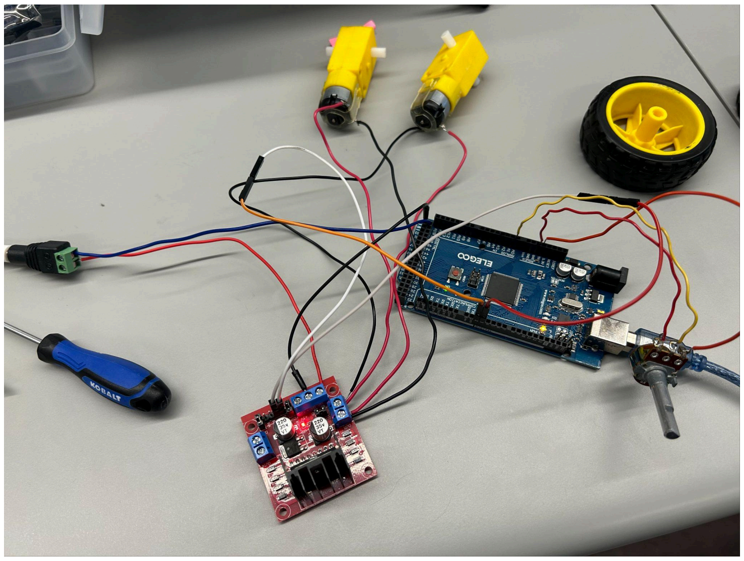


Figure 4: Unassembled Robot

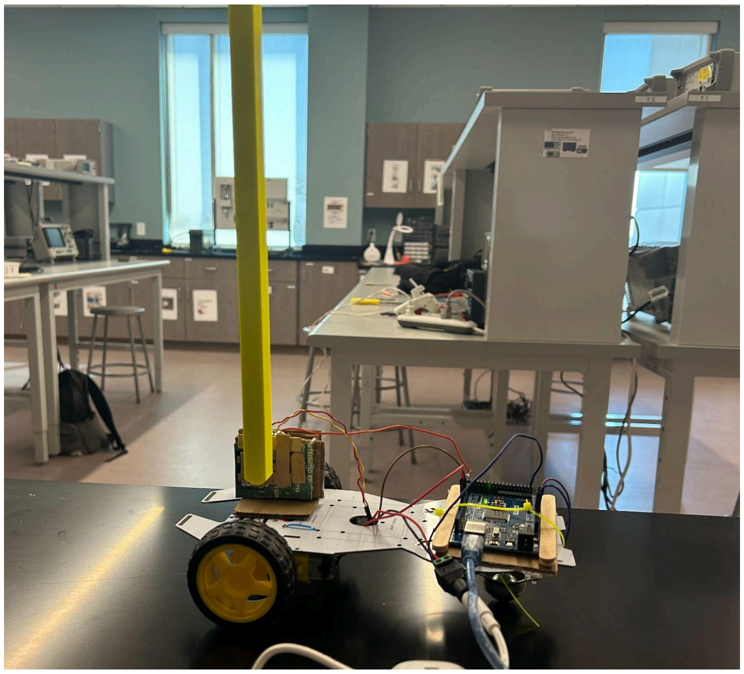


Figure 5: Completed Robot

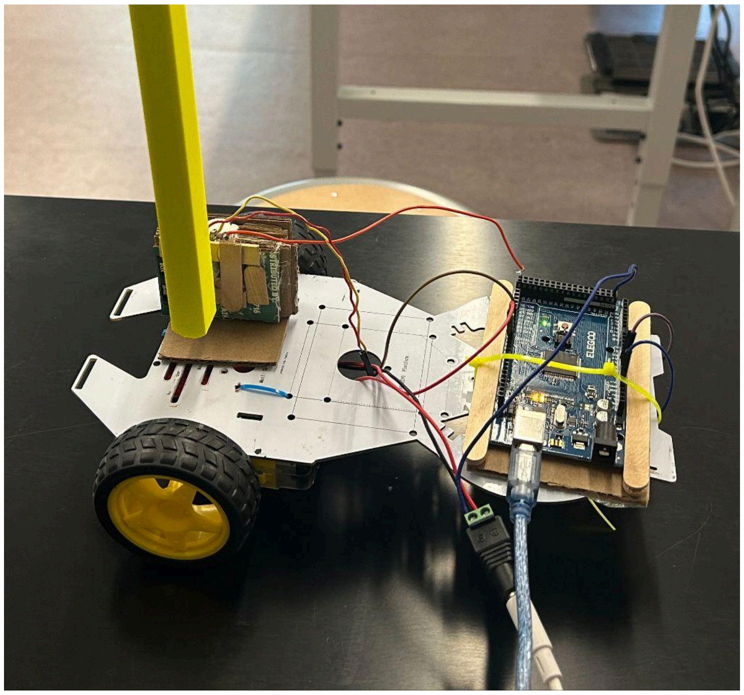


Figure 6: Close Up of Completed Robot

Constraints & Improvements

The constraints and improvements part of this lab will be further discussed in this portion of the lab. This section will allow us to summarize our shortcomings in the Testing and Results section of this lab as well as how we would overcome them in future installments of this project.

IV.1 Constraints

The constraints we encountered would be composed of several items. The first constraint would be the battery pack, or better yet the lack thereof of said battery pack. During our experiment, we had to keep connected to a suitable outlet to continuously power our H-Bridge and subsequently our motors. This led to limited movement for our autocorrecting automobile dynamic pendulum system which then hindered the overall movement of our car. The similar but different constraint we analysed next would be required connection to the computer the Arduino Mega 2560 Board had with our computer to actually simulate the program we developed. This led to massive movement constraint due to the small connection wire provided to us. Another constraint we faced was the power of our motors, the motors we used in the experiment did not have enough torque to move the vehicle quickly. The final constraint we encountered was the limited road length. This constraint just specifies that this constraint will work a lot better if we had an infinitely long stretch of length in which we could conduct safely and without external interference to our pendulum car.

IV.2 Improvements

The improvements we would make to fundamentally get rid of our constraint would be the addition of a battery pack that will allow us to fundamentally get rid of our current battery supply which is the outlet. This would allow for more freedom to the pendulum car when traveling back and forward, this however will cause there to be more weight in the pendulum car as a result. To counteract this problem we would also like to upgrade the DC motors on the vehicle, with higher power motors. Motors with more torque would be beneficial due to the fact that the vehicle would be able to accelerate more quickly, thus stabilizing the pendulum more efficiently.

The following improvement would be to utilize another software other than MatLab that would allow for us to directly download and run without a computer connection required. This software can write the language in C or C++ in our arduino software. This would ultimately cause more improved movement and less connection constraints.

The final improvement would be to conduct this experiment in a large, low friction, stretch of land. This could include but not be limited to a large open space, a hallway, etc. These few but simple improvements would help us deeply with the running of this particular experiment for future iterations.

Conclusions

In conclusion, we discovered that our final project pertaining to the automated autocorrecting pendulum car system, although met with various constraints, met the required functionality. We discovered that the pendulum worked perfectly in tandem with our Arduino Microcontroller to properly develop an input voltage signal that was then developed to function properly in a MatLab Function where our controller would then choose the drive configuration to stay upright. Our team developed a deep understanding of how to gather a position of a pendulum by using a potentiometer, as well as utilizing our MatLab software to properly program two Pulse Width Modulus. This final project allowed us to discover how we can apply our basic knowledge gained from this course to working dynamic projects such as these.

Contributions

We would like to thank the following individuals for their hard work and dedication to this project.

- Dr. Aldo Munoz Vazquez: Professor who inspired the project, as well as provided their knowledge and expertise in control systems.
- Omar Rodriguez: Engineering Lab Technician, provided access to the electronics lab and 3D printer as well as provided the group with materials to use for the project.
- Jacob Reyes: Lead Hardware and Manufacturing Developer, worked on assembling the vehicle, as well as design for arduino platform and pendulum assembly.
- Mark Perez: Lead Testing and Screening Developer, worked on testing the vehicle, as well as aided in the assembly of the vehicle.
- Ricardo Rodriguez: Lead Software Developer, worked on the code for the vehicle in SimuLink. Also responsible for creating the 3D model of the pendulum.