Inverted Pendulum

Control Theory MT5104

Katrine Hansen
Per Henrik Hardeberg

SUPERVISOR
Dr. Janardhan Vistapalli

University of Agder - 2023
Faculty of Engineering and Science

Contents

List of Figures **ii**

1 Introduction and Objectives **1**

2 Hardware **1**

3 Start Up for Demo: Work Instruction **3**

3.1 Step 1 3

3.2 Step 3 3

3.3 Step 4 3

3.4 Step 5 4

3.5 Step 6 4

3.6 Step 7 5

3.7 Step 8 5

3.8 Step 9 5

4 System Modeling **6**

4.1 Kinematics for non-linear model 6

4.2 Linearisation of the Nonlinear model 7

5 Parameter Estimation **9**

6 LQR Control **10**

6.1 Angle Controller Test 10

6.2 P — controller example 1 11

6.3 P — controller example 2 11

6.4 Calculate LQR gains 12

7 Swing up Control **12**

8 Simulink Program **14**

8.1 Energy Swing Up Controller 15

8.2 LQR Controller 16

8.3 Real Time Data . . . 17
8.4 Control System - Case Controller . . . 18
8.4.1 Choose Controller And Limit Switch Safety . . . 18
8.4.2 Crash Protection . . . 19

9 Outcome and Conclusion **20**
9.1 Results . . . 20

10 Challenges and Further Work **21**
10.1 Hardware . . . 21
10.2 Software . . . 22
10.3 Arduino IDE vs Simulink . . . 22
10.4 Double pendulum . . . 22

11 Conclusion **23**

Bibliography **24**

A LQR Gain Calculation Matlab **25**

B Low Pass Filter Design **26**

List of Figures

1 Pendulum Setup . . . 1
2 Boards . . . 1
3 Other Components . . . 2
4 Power Supply . . . 2
5 Simulink File . . . 3
6 POWER SUPPLY OFF . . . 4
7 Pendulum in center . . . 4
8 Monitor and Tune . . . 4
9 Program Running . . . 5
10 POWER SUPPLY ON . . . 5

11 Free Body Diagram 6
12 Physical Plant 6
13 Friction Estimation 10
14 P controller example 2 11
15 P controller example 1 11
16 P-controller challenge 12
17 Initial position and desired position of the pendulum for the energy swing up control 12
18 The main parts of the Simulink model 14
19 Energy swing up controller 15
20 Energy swing up controller equations 15
21 Force controller equation 16
22 Force to voltage equation 16
23 LQR controller 16
24 Real time data 17
25 Filtered signal 18
26 Control system - case controller 18
27 Controller chooser and limit switch safety 19
28 Crash protection 19
29 Angle, Cart Position and Voltage Swing Up and Balance 20
30 Phase Plot 21
31 Pulley on Motor 21
32 Friction Challenge 22

Report revision	**Date**	**Change**	**Signature**
A	02.12.2023	First publishment	PHH & KAH
B	09.12.2023	Added equations and explanations	PHH & KAH

Table 1: Revision Table

1 Introduction and Objectives

The task given in this project is to get the inverted pendulum showed in figure 1 to swing up from an initial position and balance in an upright position. The setup is located in the Mechatronics Lab at Mahindra University. The board, Arduino Mega, is controlling a DC motor and reading the sensor values. The angle of the pendulum and the position of the cart is measured using two rotary encoders.

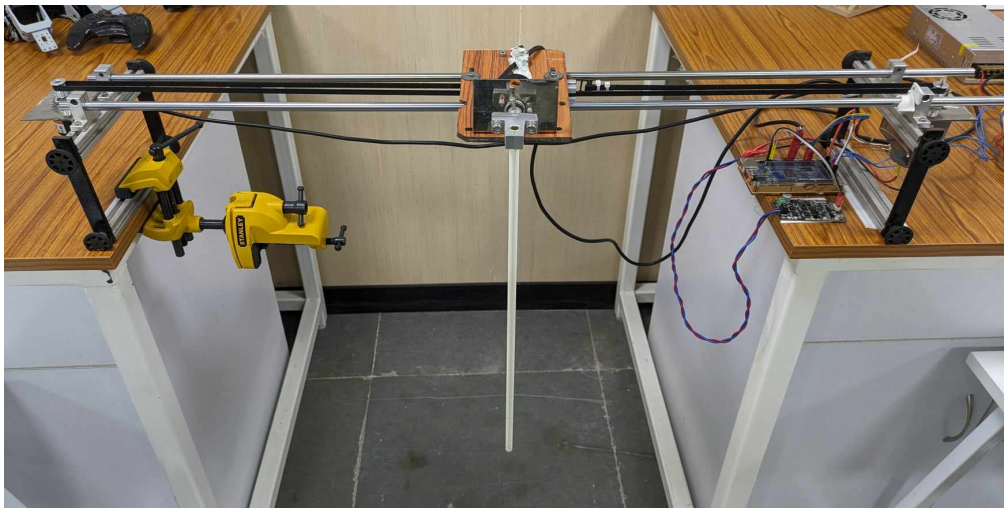
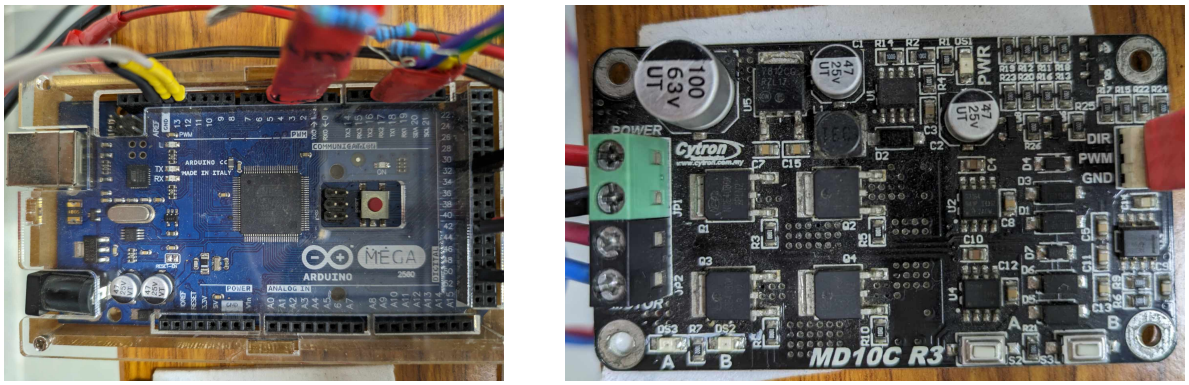


Figure 1: Pendulum Setup

This project is highly inspired by the inverted pendulum project of Jitendra Singh. A lot of inspiration and equations are found in his reports and models from his GitHub [2].

The objective of the project is to give the students and supervisors knowledge of complex control systems, as well as finishing a semi - complete setup in the Mechatronics Lab in such a way that it can be used for demonstration and further work for educational purposes in the future. Further application of the project could be to implement another pendulum or even two to get a more complicated system.

2 Hardware



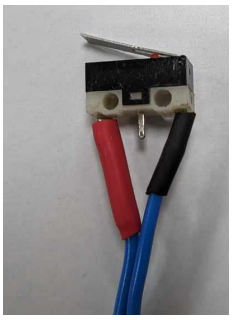
(a) Arduino Mega

(b) Motor Controller

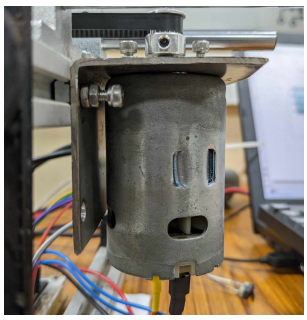
Figure 2: Boards

The microcontroller used is the Arduino Mega, showed in figure 2a. The motor controller is an MD10C R3 controller, showed in figure 2b.

(a) Rotary Encoder



(b) Enable



(c) DC motor

Figure 3: Other Components

There are two encoders. Both are 3806-opti encoders. However, one of them is 1000 PPR and the other is 600 PPR. Both of them look the same, as showed in figure 3a, however, they are labeled with 1000 or 600 accordingly. The limit switch presented in figure 3b is used as a safety switch for the system. The use of this is explained more in detail in the demo chapter 3. The DC motor is presented in figure 3c and the power supply is presented in figure 4.

Figure 4: Power Supply

Arduino Pin	**Connected to**
12	Motor Driver pin: DIR
13	Motor Driver pin: PWM
GND	Motor Driver pin: GND
16	Angle Encoder: VCC
17	Angle Encoder: GND
18	Angle Encoder: Phase A (Green) Pull-up Resistor Soldered to cable
19	Angle Encoder: Phase B (White) Pull-up Resistor Soldered to cable
5	Position Encoder: VCC
4	Position Encoder: GND
3	Position Encoder: Phase A (Green) Pull-up Resistor Soldered to cable
2	Position Encoder: Phase B (White) Pull-up Resistor Soldered to cable
42	Enable Switch: Input
43	Enable Switch: Output

Table 2: Connection Table Arduino Pins

The connections from the Arduino to the different components are presented in table 2. The supply voltages for the encoders are connected to different digital pins. Then, each of them are enabled in the software. This is to make a cleaner connection without a breadboard, as well as be able to enable them individually if necessary.

The motor controller is also connected to the 12 V power supply, and to the DC motor.

3 Start Up for Demo: Work Instruction

Read all the steps in this section carefully before starting the simulink model.

3.1 Step 1

Download folder from github:

https://github.com/PerHenrikHardeberg/PendulumProjectMU_public

3.2 Step 3

Open simulink model, filename PendulumProjectMU.slx:

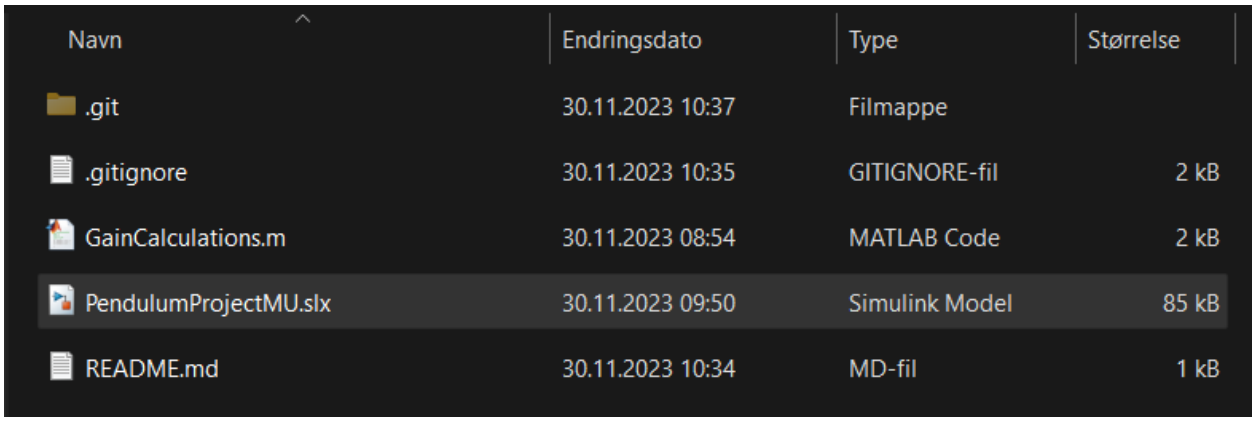


Figure 5: Simulink File

3.3 Step 4

Make sure to have the Arduino Support Package for simulink installed on your computer.

Connect the USB cable from the Arduino to your computer. If problems finding the Arduino, check the following:

- Modeling (top menu) » Model Settings (ctrl + E) » Hardware Implementation » Hoast-Board Connections » Host COM Port
- Set the correct step time: Modeling (top menu) » Model Settings (ctrl + E) » Solver (side menu). Set the solver to *Auto*, type to *fixed-step*, and step time to 0.01.

3.4 Step 5

Make sure POWER IS OFF ALL THE TIME DURING UPLOAD.

Figure 6: POWER SUPPLY OFF

3.5 Step 6

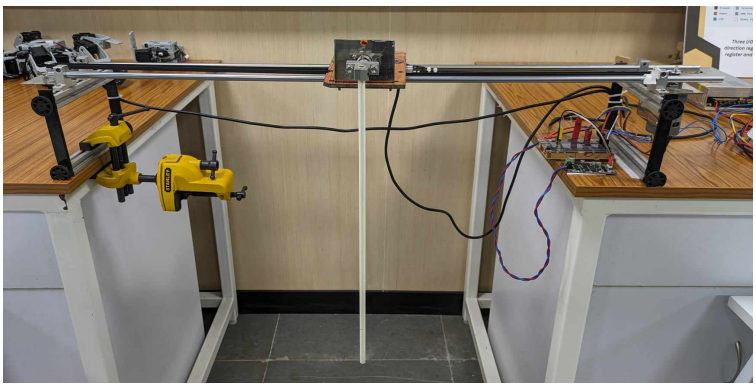


Figure 7: Pendulum in center

Make sure the pendulum is in the center of the work area, and the pendulum is hanging down as presented in figure 7. This is because the position x and θ is set to zero when the program is uploaded.

Be sure the power supply is turned off, like presented in the last step. Then the monitor and tune button is pressed to upload the program. The program will run on your computer and gets communicated to the microcontroller. Therefore, the computer must remain connected.

If there is a problem during upload, read the error messages. Also, make sure to check step 4 (Section 3.3).

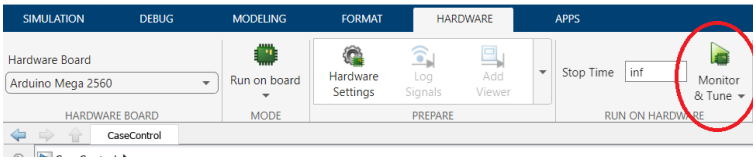


Figure 8: Monitor and Tune

3.6 Step 7

Make sure POWER IS OFF ALL THE TIME DURING UPLOAD.

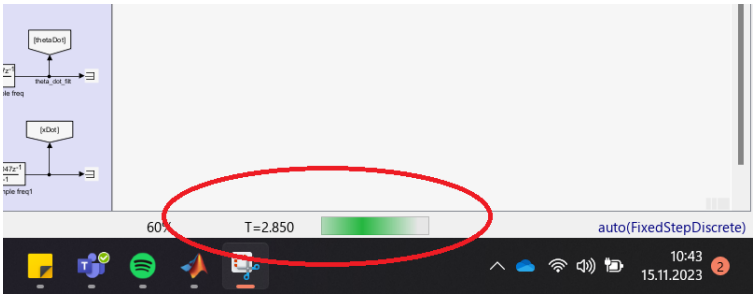


Figure 9: Program Running

After the program starts to run, the program running time will start to count seconds like presented in figure 9. Now, the power supply can be turned on, see figure 10.

Figure 10: POWER SUPPLY ON

3.7 Step 8

Press and hold the Enable button all the time to get the motor running. When the button is released, the motor PWM is set to 0 and the motor will stop immediately. This is a safety function.

3.8 Step 9

ALWAYS TURN SUPPLY OFF WHEN CHANGES ARE DONE IN THE MODEL BEFORE NEXT UPLOAD. The reason for this is that when the new script is uploaded, all Arduino pins go to high value, and then back to low. If the power supply is connected, the motor will run with maximum voltage and crash.

4 System Modeling

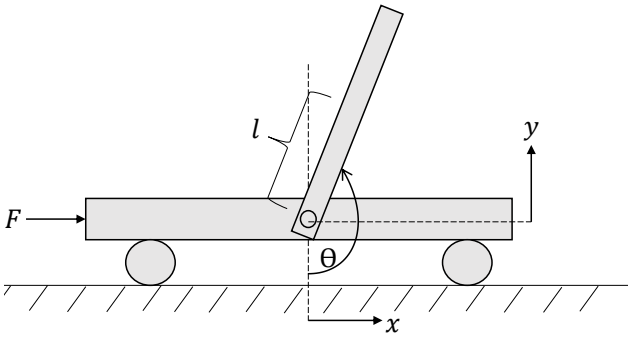


Figure 11: Free Body Diagram

The system can be modeled like a cart. The parameters from the model are used to calculate the gains for the different controllers. Because there was little to no information about the motor parameters, some parameters are roughly estimated. The most important observations from the system in figure 11 is that the angle $\theta = 0$ is where the pendulum is when the program is started. Also, the position $x = 0$ is where the cart is located when the program is started. This should be in the center of the work area of the cart.

Figure 12: Physical Plant

The input and output for the physical system is presented in figure 12. This is because the only input to the physical system is the motor voltage. This has a direct relation to the force applied to the cart. The only outputs from the system are the values from the two encoders. One attached to the belt, measuring the position of the cart x, and the other attached to the pendulum, measuring the angle θ. By differentiating x, $\dot{x}$ is found, and by differentiating θ, $\dot{\theta}$ is found. These values are used in the controllers.

4.1 Kinematics for non-linear model

Center position of the pendulum:

$$\begin{bmatrix} x_p \\ x_p \end{bmatrix} = \begin{bmatrix} x + l\,sin\,\theta \\ -l\,cos\,\theta \end{bmatrix} \tag{1}$$

Lagrangian, where E_K is kinetic energy and E_p is potential energy:

$$\mathcal{L} = E_K - E_p \tag{2}$$

Potential energy = 0, when pendulum has $\theta = \pi$

$$E_p = -m\,g\,l\,cos\,\theta \tag{3}$$

Total kinetic energy of the system E_K is including the translational and the rotational energy of the system and is given by equation 4, where M_c is the mass of the cart, m is the mass of the pendulum, and I is the moment of inertia of the pendulum:

$$E_K = \frac{1}{2}M_c\,\dot{x}^2 + \frac{1}{2}m\,\dot{x}^2 + \frac{1}{2}m\,l^2\,\dot{\theta}^2 + m\;\dot{x}\,\dot{\theta}\,cos\,\theta + \frac{1}{2}I\,\dot{\theta}^2 \tag{4}$$

Inserting equation 4 and 3 in equation 2 gives:

$$\mathcal{L} = \frac{1}{2}M_c\,\dot{x}^2 + \frac{1}{2}m\,\dot{x}^2 + \frac{1}{2}m\,l^2\,\dot{\theta}^2 + m\;\dot{x}\,\dot{\theta}\,cos\,\theta + \frac{1}{2}I\,\dot{\theta}^2 + m\,g\,l\,cos\,\theta \tag{5}$$

System with two degrees of freedom, represented with two Lagrangian equations in equation 6 and 7. F is external force, c is the friction coefficient between the cart and "ground", and b is the friction between the pendulum and the cart.

$$\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{\partial\mathcal{L}}{\partial x} = F - c\dot{x} \tag{6}$$

$$\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{\theta}} - \frac{\partial\mathcal{L}}{\partial\theta} = -b\dot{\theta} \tag{7}$$

Partial differentiate the Lagrangian:

$$\frac{\partial\mathcal{L}}{\partial\dot{x}} = M_c\,\dot{x} + m\,\dot{x} + m\,l\;\dot{\theta}\,cos\,\theta \tag{8}$$

$$\frac{\partial\mathcal{L}}{\partial x} = 0 \tag{9}$$

$$\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} = \ddot{x}(M_c + m) - m\,l\,\dot{\theta^2}\,sin\,\theta + m\,l\,\ddot{\theta}\,cos\,\theta \tag{10}$$

Equation 8 are inserted in to equation 6. The partial derivatives with respect to θ are also found and inserted in equation 7. The resultant two equation of motion is then:

$$(M_c + m)\ddot{x} + m\,l\,\ddot{\theta}\,cos\,\theta - m\,l\,\dot{\theta}^2\,sin\,\theta = F - c\dot{x} \tag{11}$$

$$(I + m\,l^2)\ddot{\theta} + m\,l\,\ddot{x}\,cos\,\theta + m\,g\,l\,sin\,\theta = -b\dot{\theta} \tag{12}$$

4.2 Linearisation of the Nonlinear model

Using equation 12 and sort for $\ddot{x}$:

$$\ddot{x} = \frac{-b\,\dot{\theta} - m\,g\,l\,sin\,\theta - (I + m\,l^2)\,\ddot{\theta}}{m\,l\,cos\,\theta} \tag{13}$$

Insert equation 13 in equation 11:

$$\ddot{\theta} = \frac{-(F\,m\,l\,cos\,\theta - c\,m\,l\dot{x}\,cos\,\theta + m^2\,l^2\,\dot{\theta}^2\,\sin\,\theta\,cos\,\theta + (M + m)(b\,\theta + m\,\dot{g}\,l\,sin\,\theta))}{m^2\,l^2\,\sin^2\,\theta + M\,m\,l^2 + (M + m)\,I} \tag{14}$$

Use equation 11 and solve for $\ddot{\theta}$:

$$\ddot{\theta} = \frac{F - c\,\dot{x} - (M+m)\,\ddot{x} + m\,l\,\dot{\theta}^2\,sin\,\theta}{m\,l\,cos\,\theta} \tag{15}$$

Insert equation 15 in equation 12:

$$\ddot{x} = \frac{b\,m\,l\,\dot{\theta}\,cos\,\theta + m^2\,l^2\,g\,sin\,\theta\,cos\,\theta + (I + m\,l^2)(F - c\,\dot{x} + m\,l\dot{\theta}^2\,sin\,\theta)}{m^2\,l^2\,sin^2\,\theta + M\,m\,l^2 + (M+m)I} \tag{16}$$

Then, the system should be represented in state space. The relationship is established:

$$q = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} \tag{17}$$

Differentiate q and insert $\dot{x}_3 = \ddot{x}$ and $\dot{x}_4 = \ddot{\theta}$. Thus, equation 16 and 14 inserted in equation 18.

$$\dot{q} = \begin{bmatrix} x_3 \\ x_4 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ \frac{b\,m\,l\,x_4\,cos\,x_2 + m^2\,l^2\,g\,sin\,x_2\,cos\,x_2 + (I+m\,l^2)(F - c\,x_3 + m\,l x_4^2\,sin\,x_2)}{m^2\,l^2\,sin^2\,x_2 + M\,m\,l^2 + (M+m)I} \\ \frac{-(F\,m\,l\,cos\,x_2 - c\,m\,l x_3\,cos\,x_2 + m^2\,l^2\,x_4^2\,\sin\,x_2\,cos\,x_2 + (M+m)(b\,x_2 + m\,\dot{g}\,l\,sin\,x_2))}{m^2\,l^2\,\sin^2\,x_2 + M\,m\,l^2 + (M+m)\,I} \end{bmatrix} \tag{18}$$

The reference point for the system, meaning the desired state for the balancing pendulum, is in the middle of the working area with upright standing pendulum as presented in equation 19. $U = F$.

$$(X_0, U_0) = ([0\ \pi\ 0\ 0],\ 0) \tag{19}$$

The system can be linearized around the reference point, and will then be represented as:

$$\begin{aligned} \dot{X} &= A\,X + B\,U \\ Y &= C\,X \\ A &= \frac{\partial f}{\partial X}_{(X_0,U_0)} \\ B &= \frac{\partial f}{\partial U}_{(X_0,U_0)} \end{aligned}$$

By inserting the point and partial derive it as presented in equation 20. To simplify the writing in the matrix, we define $\alpha = I(M+m) + M\,m\,l^2$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix}_{(X_0,U_0)} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m^2\,l^2\,g}{\alpha} & \frac{-(I+m\,l^2)c}{\alpha} & \frac{-b\,m\,l}{\alpha} \\ 0 & \frac{m\,g\,l(M+m)}{\alpha} & \frac{-m\,l\,c}{\alpha} & \frac{-b(M+m)}{\alpha} \end{bmatrix} \tag{20}$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial F} \\ \frac{\partial f_2}{\partial F} \\ \frac{\partial f_3}{\partial F} \\ \frac{\partial f_4}{\partial F} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{I+m\,l^2}{\alpha} \\ \frac{m\,l}{\alpha} \end{bmatrix} \tag{21}$$

The relationship between the force F and the applied voltage to the motor V_m is presented in equation 22. k_t, k_b and R_m are motor constants. r is the radius of the motor pulley.

$$F = \frac{K_t\,V_m\,r - k_t\,k_b\,\dot{x}}{R_m\,r^2} = \frac{K_t\,V_m\,r - k_t\,k_b\,x_3}{R_m\,r^2} \tag{22}$$

Inserting F = U, then BU becomes:

$$BU = \begin{bmatrix} 0 \\ 0 \\ \frac{I+m\,l^2}{\alpha} \\ \frac{m\,l}{\alpha} \end{bmatrix} F = \begin{bmatrix} 0 \\ 0 \\ \frac{(I+m\,l^2)k_t}{\alpha\,R_m\,r} \\ \frac{m\,l\,k_t}{\alpha\,R_m\,r} \end{bmatrix} V_m \tag{23}$$

The linear state space model representation is then:

$$\dot{X} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m^2\,l^2\,g}{\alpha} & \frac{-(I+m\,l^2)c}{\alpha} & \frac{-b\,m\,l}{\alpha} \\ 0 & \frac{m\,g\,l(M+m)}{\alpha} & \frac{-m\,l\,c}{\alpha} & \frac{-b(M+m)}{\alpha} \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{(I+m\,l^2)k_t}{\alpha\,R_m\,r} \\ \frac{m\,l\,k_t}{\alpha\,R_m\,r} \end{bmatrix} V_m \tag{24}$$

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} \tag{25}$$

The numerical values are found with Matlab script presented in appendix A.

5 Parameter Estimation

The reason this chapter is called parameter estimations, is because the system contains little to no information and the estimations are rough. The motor have no information. Still, the system was dismantled, part were measureded and weighted.

Some test in the lab was done, and others are assumed based on similar systems, as the motor used by Jitendra Singh in his experiment [2]. The first parameters for the LQR controller made the system very unstable and oscillating. Therefore, the gains were reduced drastically. A similar motor was found and dismantled, to weigh the rotor, measure the rotor diameter, and calculate the inertia of the rotor.

It is recommended to have more data on the system, such as a motor with datasheet included the needed parameters, a lighter cart, a heavier rod, and more linear friction. Still, the parameters estimated are presented in the Matlab script used to calculate the LQR gains, presented in appendix A.

Friction between the cart and the ground is measured by adding a ramp function as the input voltage. The cart will not move until the static friction is overcome. In that exact moment, the voltage is noted, and the static friction can be calculated from the force voltage relationship presented in equation 22. Figure 13 illustrates the experiment done to find the voltage needed to start moving the cart. This voltage can then be added in parallel to the controller.

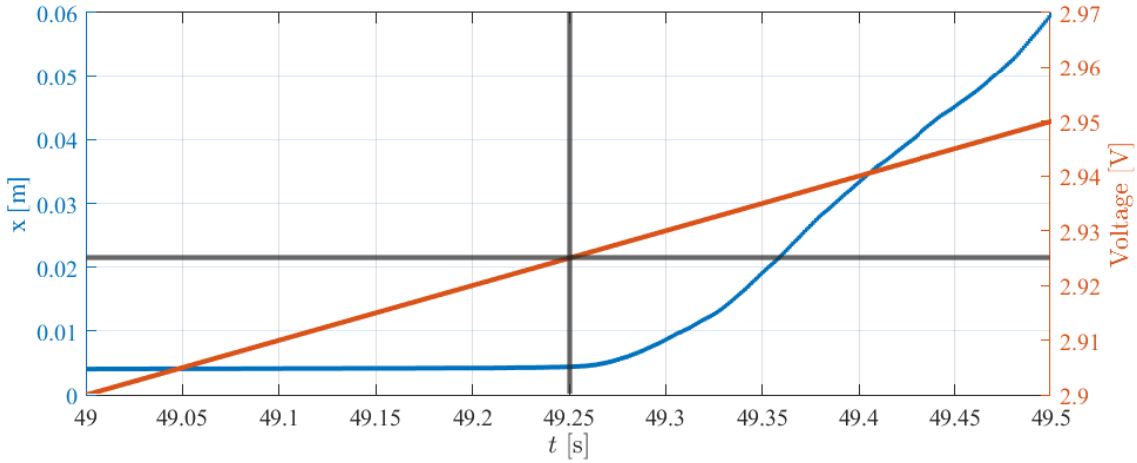


Figure 13: Friction Estimation

The friction of the cart was tested, but is still not linear, depending on where in the system the cart is located. Also, the friction in the joint between the pendulum and encoder could be modeled with experiment, but the friction was also reduced in this case, to get gains resembling the gains that worked best for the pendulum. Considering the dynamic changes of the static friction of the cart, the constant gain added to the voltage had to be reduced compared to the experiment.

6 LQR Control

The Linear Quadratic Controller (LQR) controller is used for this project because of the complexity of the system. First a proportional controller was tested, only controlling the force based on the angle error. That was not successful, two examples are presented on why it is not sufficient to control the pendulum without respect to the rest of the system.

6.1 Angle Controller Test

The examples shown are only with P-controller. However, the results would be the same with P, PI or PID controller, as long as we only consider the angle, and not the position and velocity of the cart. Therefore, the LQR controller is a good approach, taking the complete system into account.

6.2 P — controller example 1

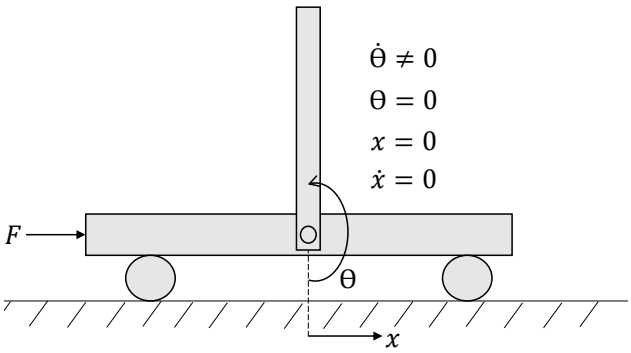


Figure 14: P controller example 2

Figure 14 shows the example of a situation where the pendulum is at the desired angle, and the P controller will remove the applied force. However, the angular velocity is not zero, and the pendulum will continue in the direction of velocity.

6.3 P — controller example 2

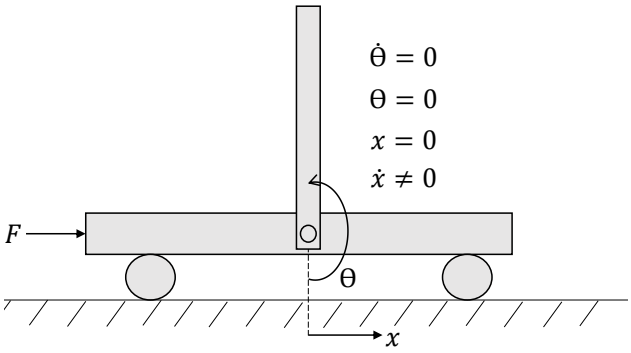


Figure 15: P controller example 1

Figure 15 shows a situation where the pendulum is in the desired position. However, the cart is not standing still. When the p-controller removes the force, due to no angle error, the cart, and thus also the pendulum, carry some kinetic energy. Therefore, when the force will be set to 0, the pendulum will continue in the direction the cart was moving towards, and thus it will never stabilize. This experiment is illustrated in 16.

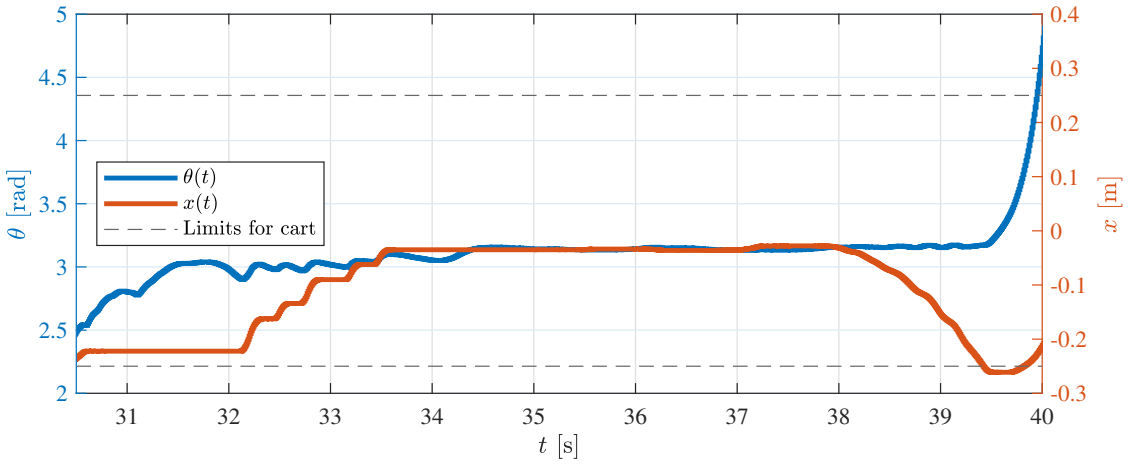


Figure 16: P-controller challenge

6.4 Calculate LQR gains

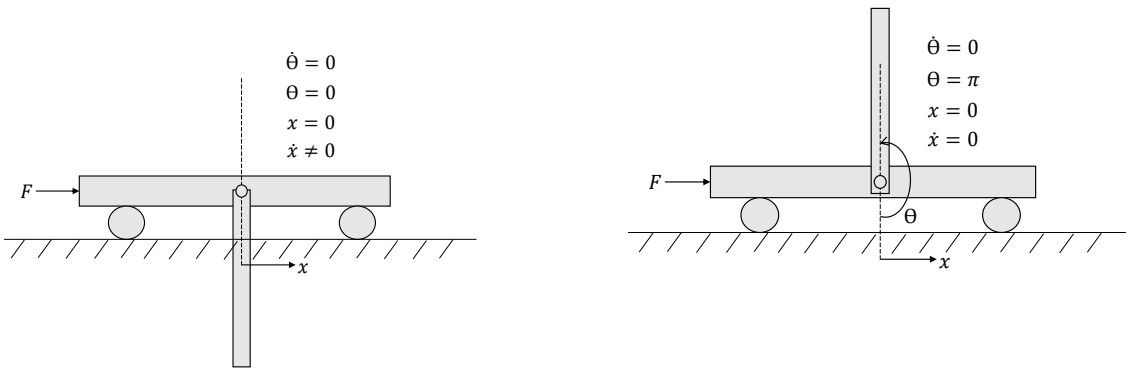
The calculation of LQR gains are done like in the documentation from Matlab for a similar example [1]. Later in the report, where the model is explained, observe that the gains in the LQR controller have opposite signs, like shown in equation 26.

$$KK = [K_x, K_\theta, K_{\dot{x}}, K_{\dot{\theta}}] = [-20,\ 120,\ -20,\ 5] \tag{26}$$

This is because when the angular velocity or linear velocity is present, the cart needs to go further to stop the momentum before balancing. The script used to calculate gains is presented in appendix A. Q and R are estimated like in the example from Matlab [1].

7 Swing up Control

For the project, an energy based swing up controller is used. The idea is the cart should start from rest with the initial conditions shown in figure 17a. Mathematically, an initial speed is necessary however, because of the resolution of the angle encoder, the velocity of the cart is rarely absolutely zero. The desired position is shown in figure 17b with an angle of 180 degrees.



(a) Initial position (b) Desired position

Figure 17: Initial position and desired position of the pendulum for the energy swing up control

The potential energy of the pendulum is used for the energy controller. The concept of the controller

is to add energy to the system if the actual potential energy is higher than the desired energy and remove energy if the actual potential energy is lesser than desired energy. Collocated partial feedback linearization and error dynamics for the energy swing up is used to control the pendulum. Additionally, as is explained in the next chapter, a simple term is added to keep the cart towards the middle during swing up.

From equation 12, the equation can be given with respect to $\ddot{\theta}$.

$$\ddot{\theta} = \frac{-b\dot{\theta} - ml\ddot{x}^d cos\theta - mglsin\theta}{I + ml^2} \tag{27}$$

Additionally, by inserting equation 27 into equation 11, the following equation 28 is retrieved.

$$F = (M+m)\ddot{x}^d + c\dot{x} - ml\dot{\theta}^2 sin\theta - ml\left(\frac{b\dot{\theta} + ml\ddot{x}^d cos\theta + mglsin\theta}{I + ml^2}\right) cos\theta \tag{28}$$

The control signal u is equal to the desired acceleration.

$$\ddot{x}^d = u$$

Inserting the previous equation into equation 27.

$$\ddot{\theta} = \frac{-b\dot{\theta} - mlucos\theta - mglsin\theta}{I + ml^2} \tag{29}$$

The following equation 30 gives the total energy of the pendulum.

$$E = \frac{1}{2}(I + ml^2)\dot{\theta}^2 + mgl(1 - cos\theta) \tag{30}$$

Also, we have the desired energy which is at the position shown in figure 17b. The desired energy is given in equation 31.

$$E_r = mgl(1 - cos\pi) = 2mgl \tag{31}$$

Now it is possible to write the error dynamics between the energy of the pendulum compared to the desired energy.

$$\tilde{E} = E - E_r \tag{32}$$

Taking the derivative of the error dynamics we get equation 33.

$$\dot{\tilde{E}} = \dot{E} = (I + ml^2)\dot{\theta}\ddot{\theta} + mglsin\theta\dot{\theta} \tag{33}$$

Now by inserting equation 29 into equation 33 we get the equation below.

$$\dot{\tilde{E}} = -b\dot{\theta}^2 - mlu\dot{\theta}cos\theta \tag{34}$$

Now, to design a controller that will stabilize the system over time, the following controller is designed, see equation 35. This contributes to making the terms with θ and $\dot{theta}$ to always being positive. In equation 36, the equation is shown after inserting the controller. Also, we can neglect the b term as it is very small. This therefor satisfies Lyapunov's stability criteria as $\tilde{E} \to 0$.

$$u = k\dot{\theta}cos\theta\tilde{E},\ k > 0 \tag{35}$$

$$\dot{\tilde{E}} = -b\dot{\theta}^2 - mlk\dot{\theta}^2cos^2\theta\tilde{E} \tag{36}$$

Finally, an equation for the controller can be found. See the equation below for the final controller equation.

$$u = sat_{u_{max}}(k(E - E_r)Sign(\dot{\theta}cos\theta)) \tag{37}$$

8 Simulink Program

The Simulink model consists of four main parts; the energy controller, the LQR controller, the control system and case controller that ensures the switching between controllers and, the real time data block that gets the pendulum angle and cart position from the encoders. Additionally, there is a separate block for encoder power supply. The whole system is shown in figure 18.

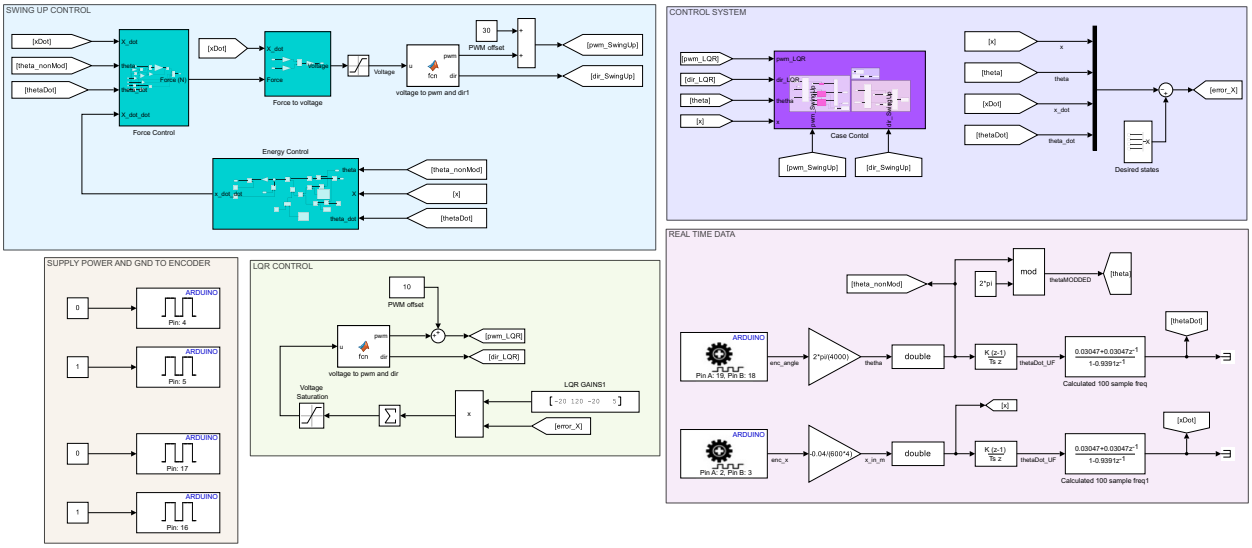


Figure 18: The main parts of the Simulink model

8.1 Energy Swing Up Controller

The swing up energy controller part consists of three separate blocks. The energy controller compares the desired potential energy with the actual potential energy and provides a desired acceleration. This acceleration, or desired control signal, is used to calculate a control force. This force is then transformed to voltage and given as input to the system. The blocks are shown in figure 19. The PWM offset is added to overcome the friction in of the cart. However, due to inconsistent friction, the offset may need changes in the future.

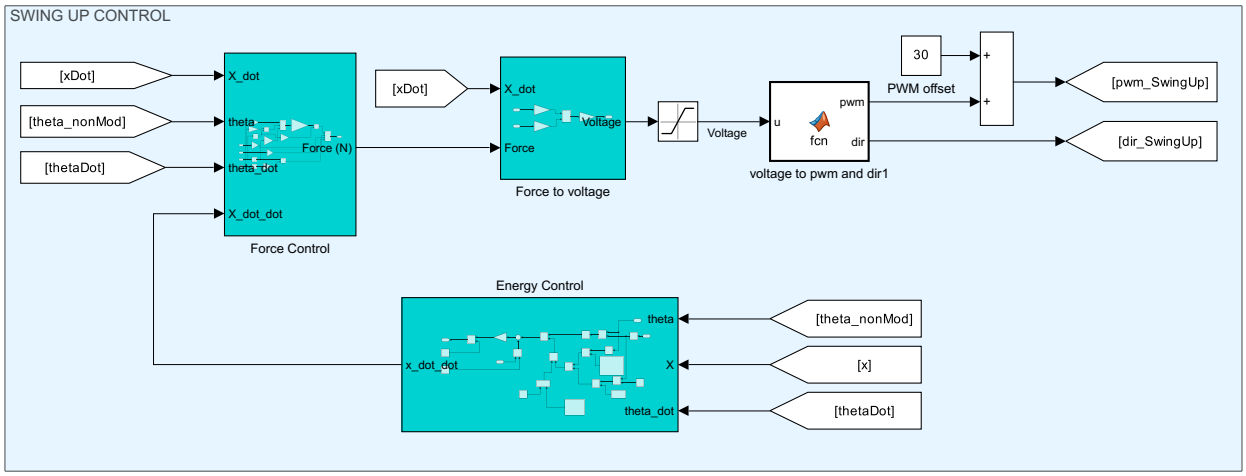


Figure 19: Energy swing up controller

Inside the *Energy Control* block, equation 37 is shown in the form of blocks. The equation is shown in figure 20. Where equation 30 is used to calculate the total energy of the pendulum. Equation 31 is applied as a constant. Also, a position gain is added to the position of the cart and added to the controller. This is to keep the cart towards the middle.

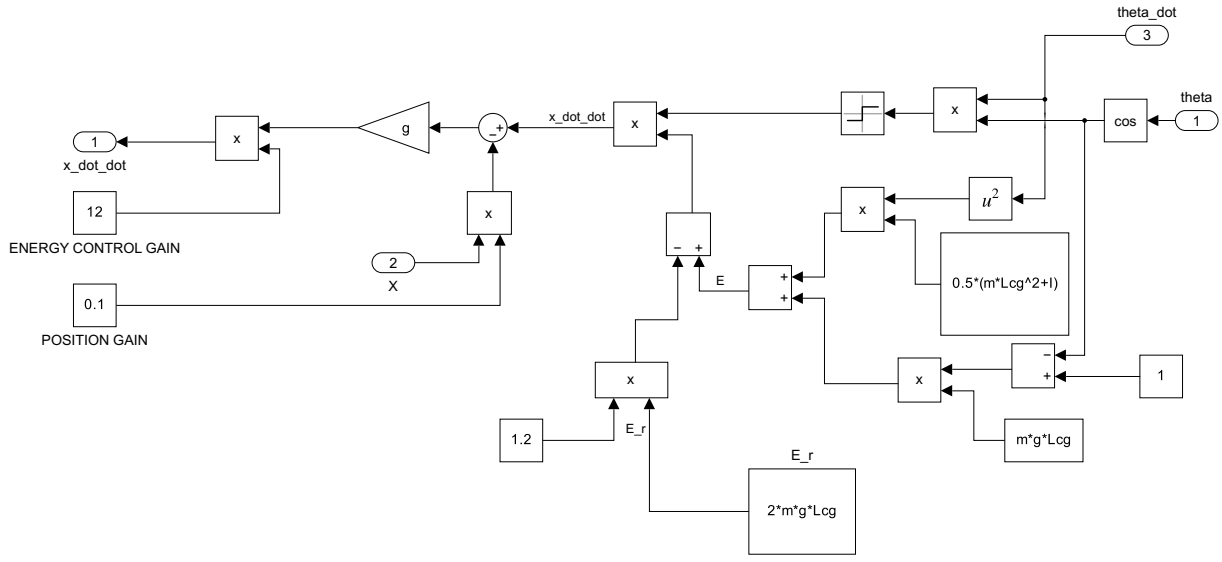


Figure 20: Energy swing up controller equations

Inside the *Force Control* block, equation 28 is described in blocks. See figure 21 for the block representation of the equation.

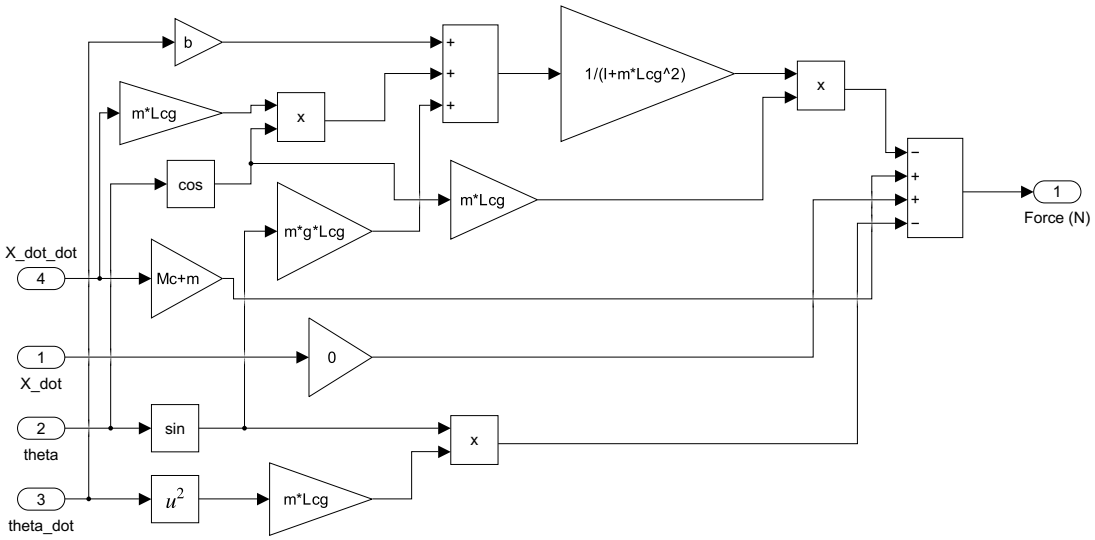


Figure 21: Force controller equation

The final block, *Force to voltage*, contains equation 22 describing the relationship between voltage and force, see figure 22

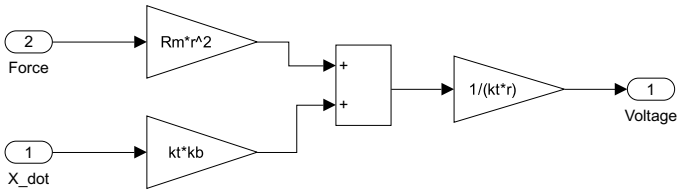


Figure 22: Force to voltage equation

8.2 LQR Controller

The LQR controller multiplies the gains calculated from the Matlab script with the error of the pendulum angle, angular velocity, cart position, and cart velocity accordingly. See figure 23. The summation of this is directly translated to voltage and provided to the system as PWM. The PWM offset is to overcome the static friction and need to change, as explained for the swing up control in section 8.1

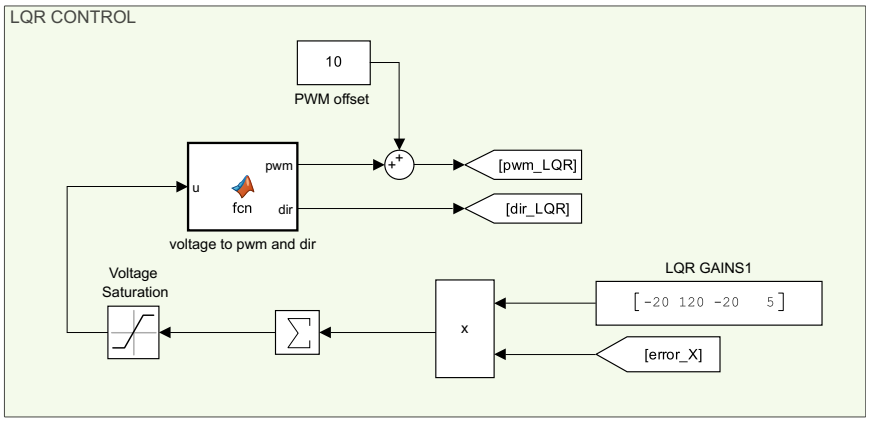


Figure 23: LQR controller

8.3 Real Time Data

This part of the Simulink program gets feedback data from the system or, the two encoders. The pulses from the encoders are transformed to angle in radians and distance in meters respectfully, see figure 24. To get the angular velocity and the cart velocity, differentiation of the signals is necessary. A low-pass filter is added to the derivatives to remove unnecessary peaks or, noise.

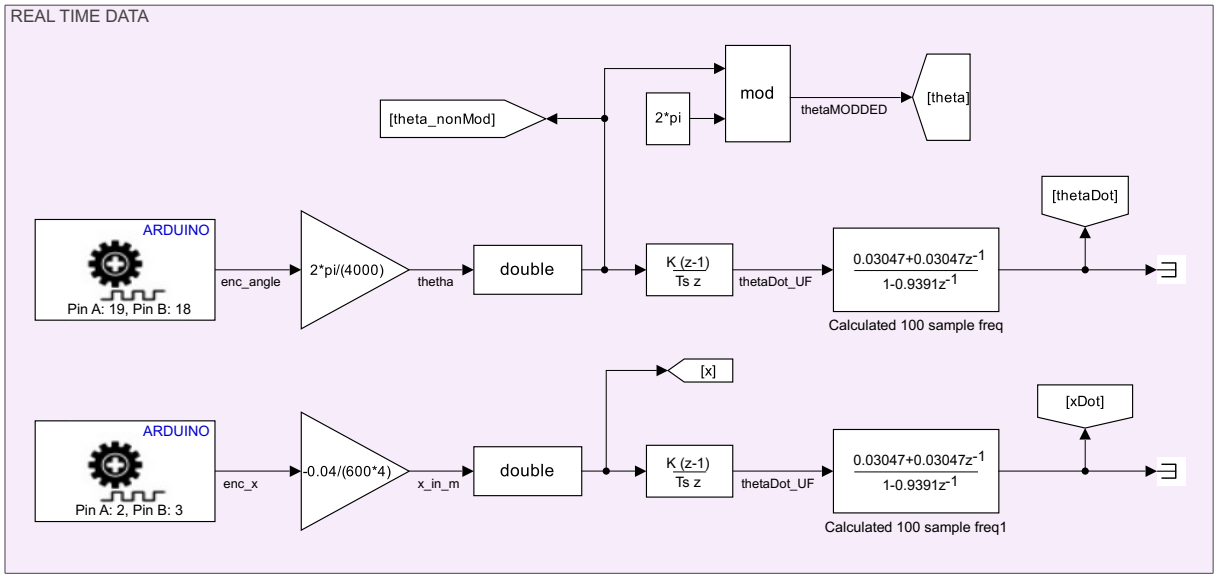


Figure 24: Real time data

When performing a differentiation of a discrete signal, a filter is usually necessary. For this project, we experienced noise because of the sampling time. Hence, the filter was designed for that specification. First, data with noise was logged from the actual system. The data is loaded into a Matlab script. The following code is run in Matlab to get the filter coefficients. Where *butter()* is an in-build function in Matlab that designs a filter with the given specifications. The last part of the code filters the data with the parameters retrieved from the function.

```
1 fs=100 % sample Freuency
2 fn=fs/2; % Nyquist Frequency
3 fc=1 % cut-off frequency
4 [b,a]=butter(1,fc/fn); % filter coefficients
5
6 thetaFiltered = filter(b,a,thetaData);
7 thetaDotFiltered = filter(b,a,thetaDotData);
```

The angular velocity measured before and after filtering is presented in figure 25. It is clear that the filter is reducing the spikes, and will reduce the oscillations for the controller. The test script for the low pass filter is presented in appendix B.

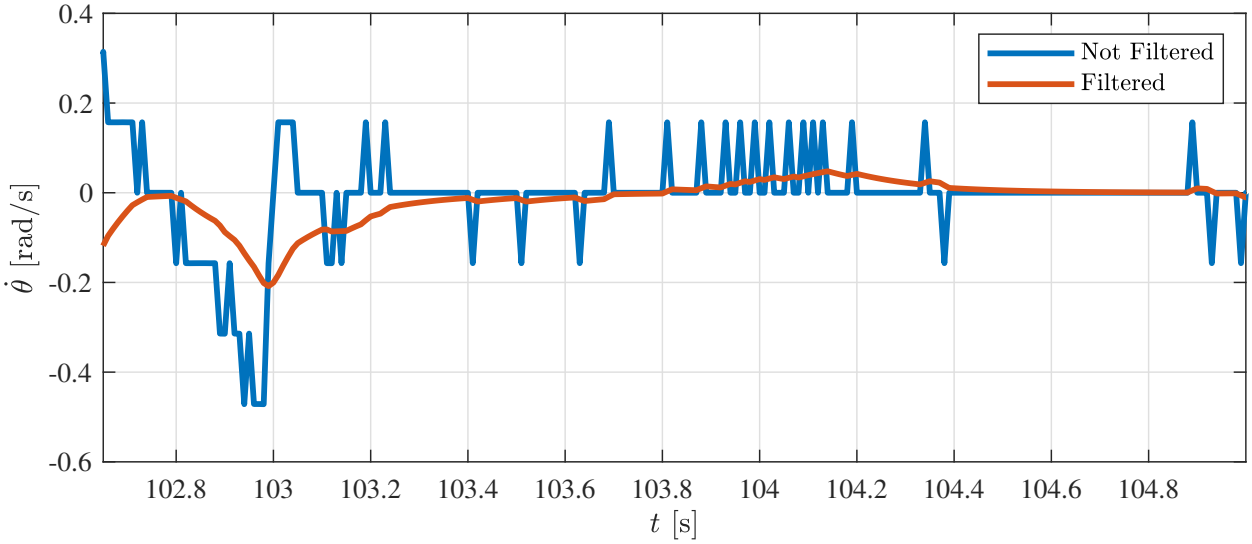


Figure 25: Filtered signal

8.4 Control System - Case Controller

Finally, the control system part, or the case controller part is responsible for choosing the correct control system, calculating the errors and also, some safety features. The system is shown in figure 26.

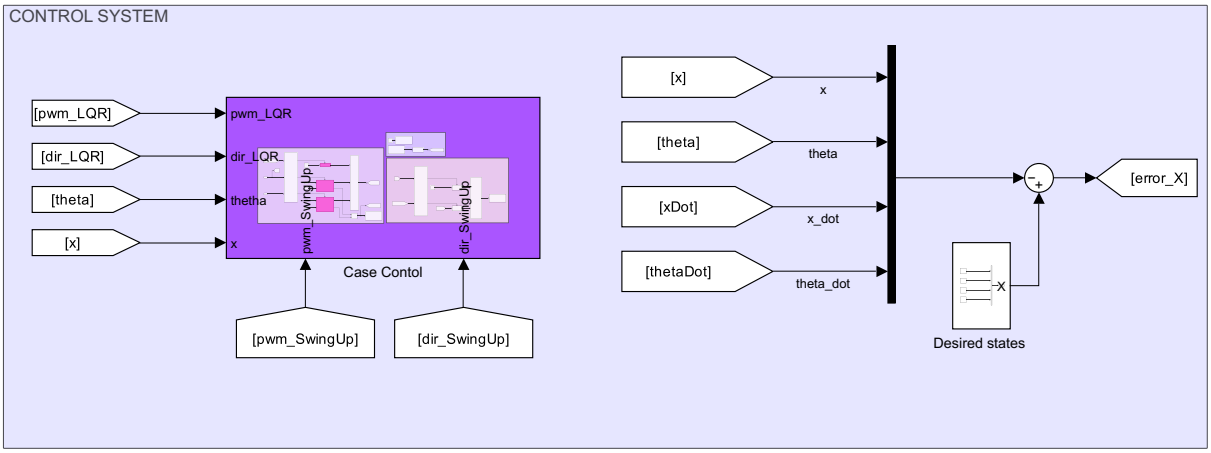


Figure 26: Control system - case controller

8.4.1 Choose Controller And Limit Switch Safety

Inside the case control block in the control system part there are different parts with different purposes. The first one decides which controller to use depending on the angle of the pendulum, see figure 27. Also, there is a safety function added that sets the pwm to zero if a limit switch is not pressed continuously.

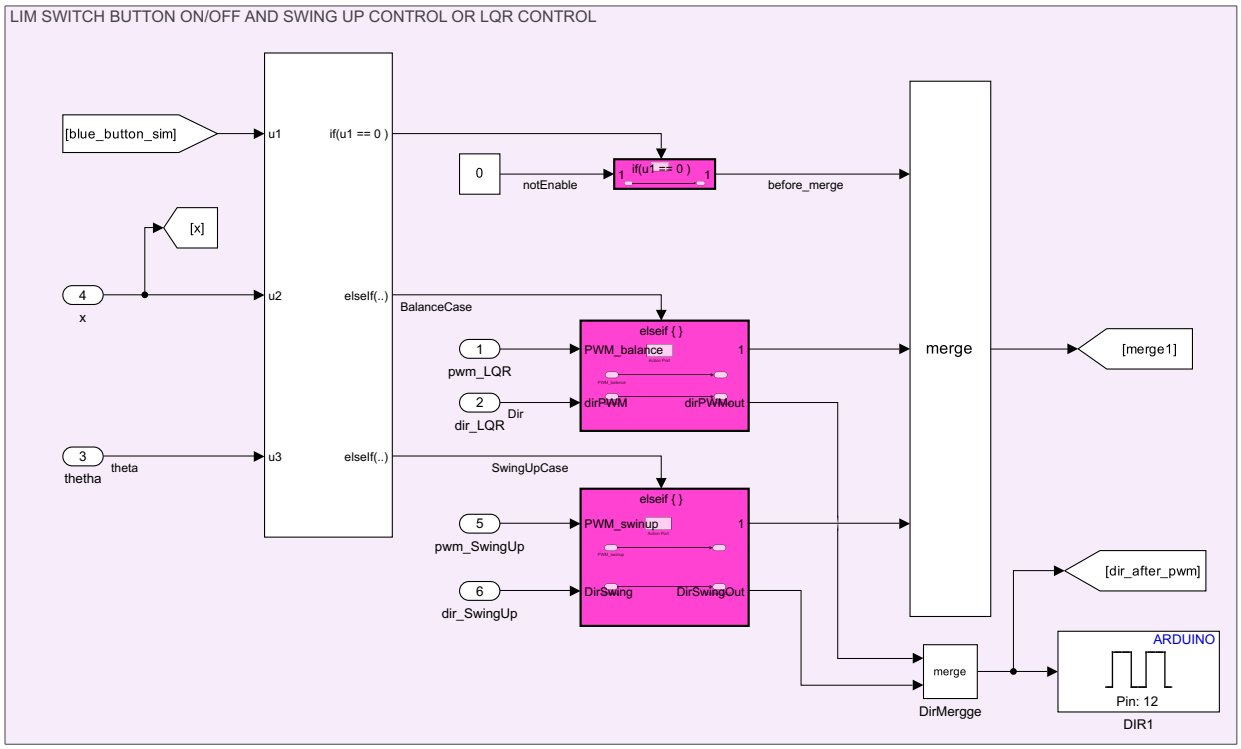


Figure 27: Controller chooser and limit switch safety

8.4.2 Crash Protection

The other part is responsible for the crash protection. In short, it will stop the motor if it is within some threshold and the velocity is going towards the end stop. However, if the cart is within the maximum allowable threshold position-wise, but the velocity is in the opposite direction, the cart is allowed to move. since, it will move away from the end-stop. The crash protection part is shown in figure 28.

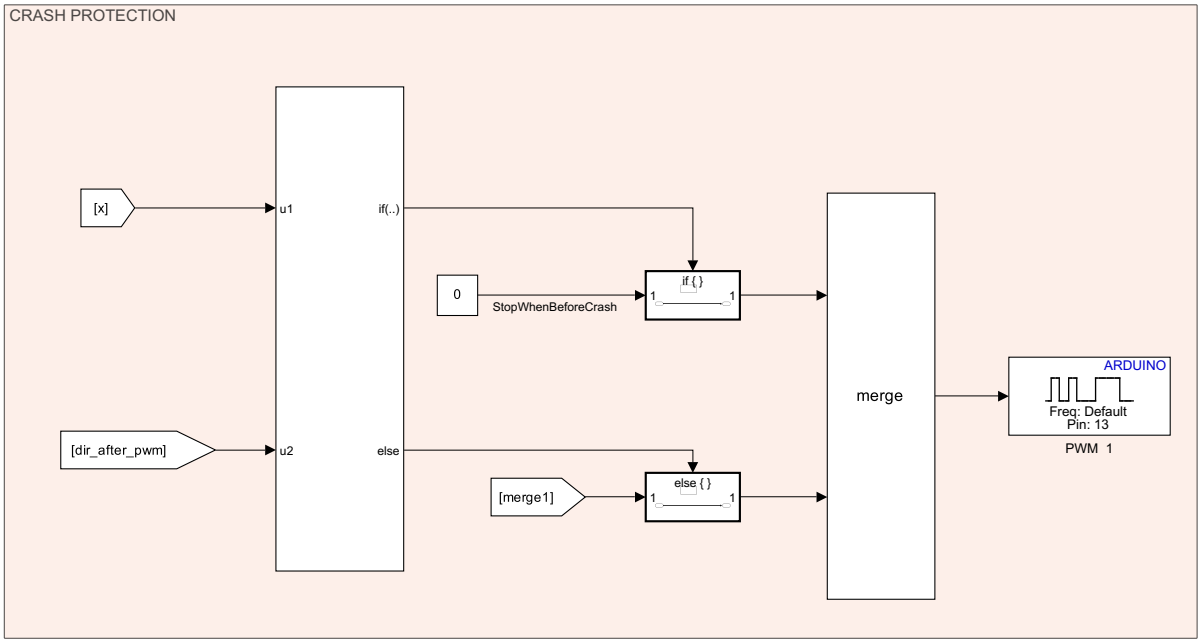


Figure 28: Crash protection

9 Outcome and Conclusion

9.1 Results

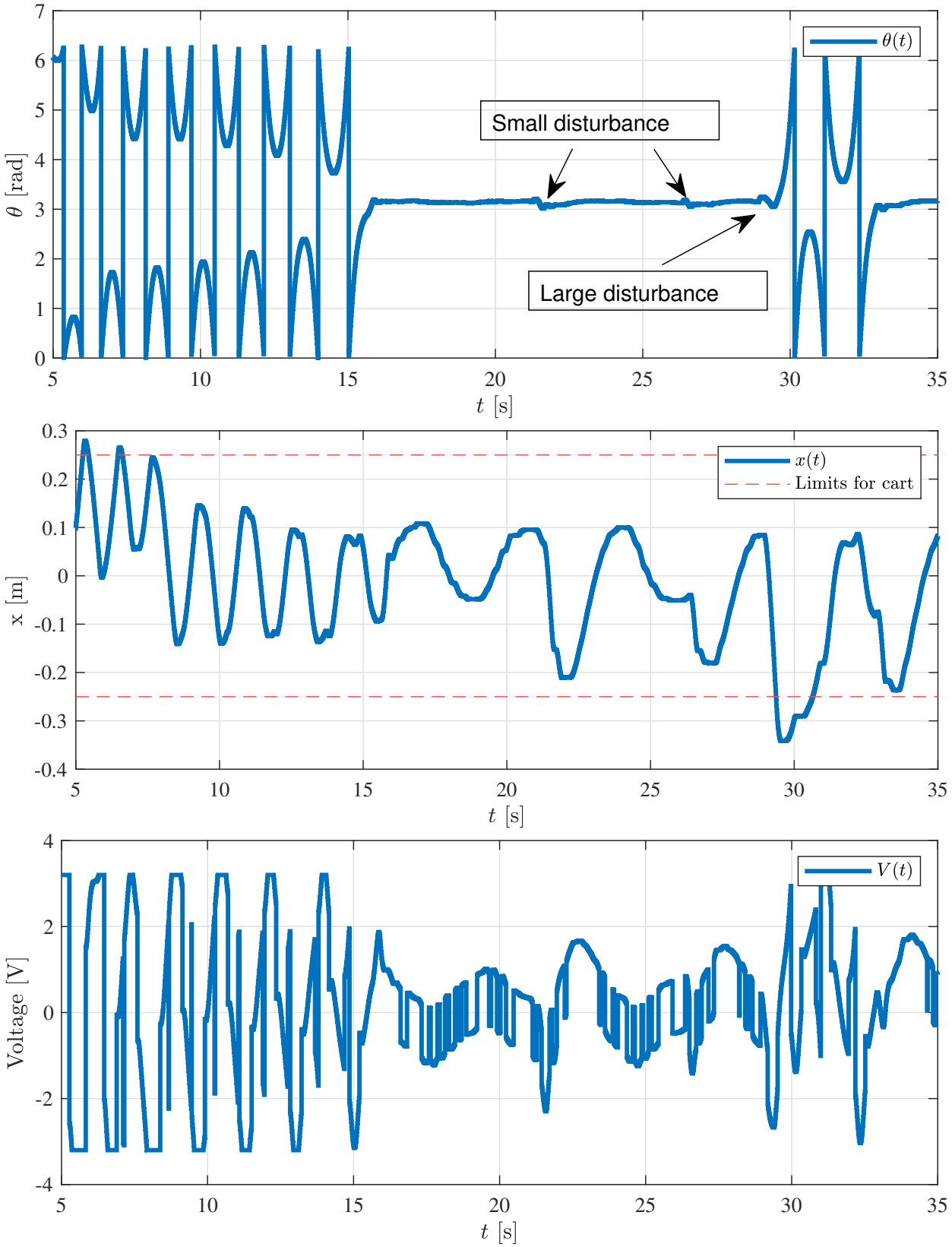


Figure 29: Angle, Cart Position and Voltage Swing Up and Balance

Figure 29 shows the cart position x, angle θ, and voltage V. Observe the time it takes for the swing up, as well as how the cart reacts to the disturbance. The voltage is saturated at around 3 V.

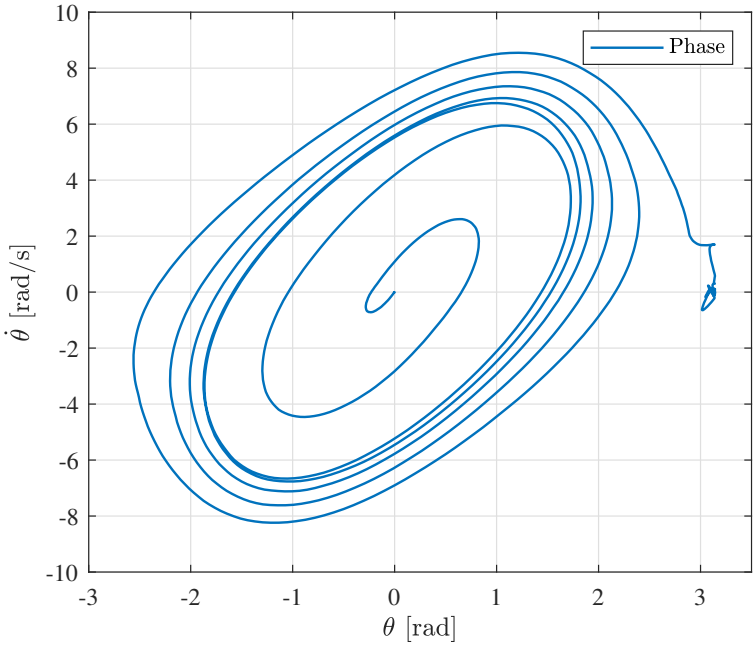


Figure 30: Phase Plot

Figure 30 is the phase plot illustrating the energy in the pendulum, by plotting the angle θ and the angular velocity $\dot{\theta}$. These are the two components of energy in the energy equation of the pendulum. Observe that from the third swing of the pendulum, the energy is not rising for some swings, before it rises again. This is because the cart is returning to oscillate around x = 0. This can also be seen in figure 29.

10 Challenges and Further Work

10.1 Hardware

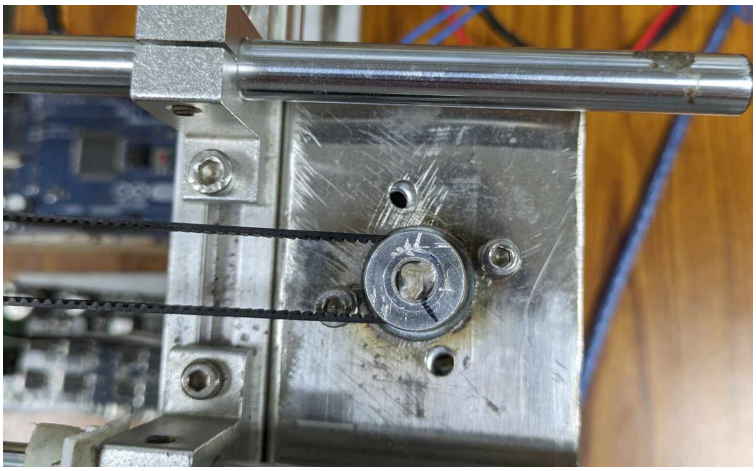


Figure 31: Pulley on Motor

The pulley shown in figure 31 is too wide big for the axle of the motor. This results in a nonlinear torque for the motor, depending on if the motor needs to tight the belt, as well as move the cart.

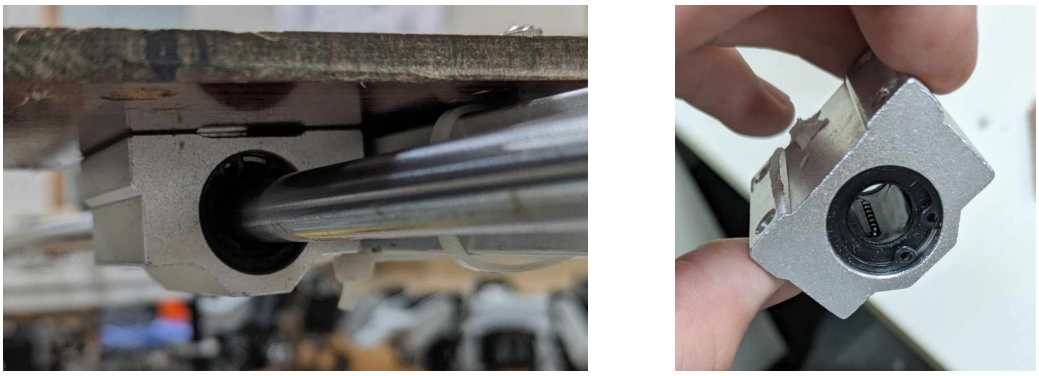


Figure 32: Friction Challenge

Figure 32 shows one of two bearings carrying the cart. These bearing, together with beams, have rust and dust inside the bearing. The axle was lubricated with some intervals during the project. However, that resulted in nonlinear friction for the cart.

10.2 Software

Many parameters, such as the friction mentioned above, motor parameters used to calculate gains for the different controllers, are not accurate. However, there could be other methods than LQR and Energy Based control that could give better results such as, a more robust balance and a decreased swing-up time.

10.3 Arduino IDE vs Simulink

The implementation of the Simulink program directly in Arduino IDE could be considered. This could make a stand-alone system with only the need of a power supply to run, instead of connecting a computer and waiting for the Simulink program to upload.

Some experiments using Arduino IDE were done. The reason Simulink was chosen for this project, was because the encoder value lost data for high angular velocity in the Arduino IDE. This could change, using another controller or encoders with lower resolution.

10.4 Double pendulum

This project is the first step of the goal of the mechatronics' lab at MU, to make double or triple pendulum systems. The students recommend not using the existing system for double pendulum due to its challenge regarding the hardware. However, using this as a reference to design a complete new system with different controllers, slider mechanism, encoders, and pendulum could make a successful double pendulum project.

11 Conclusion

The hardware setup was partly built, but the students finished the setup and, created a Simulink model for controlling the system. Two control systems were designed separately, a Linear Quadratic Controller (LQR) for the balancing part, and an energy based controller for the swing up of the pendulum. Additionally, other features were added and programmed, such as a safety limit switch and crash protection.

The goal of the project was to finish the existing setup, and create a working model. This was done successfully. However, the student experienced the challenges of working with hardware. Nonlinear friction and motor without data, among other things, made the project more complex than expected. The setup is working and can be used for educational purposes, manipulating the program and the hardware to get a different outcome. However, for a potential double pendulum project in the future, the setup should be rebuilt with different controller as well as sliders, cart, bearings, and pendulum.

Bibliography

[1] Matlab. *lqr*. URL: https://se.mathworks.com/help/control/ref/lti.lqr.html. (accessed: 8.12.2023).

[2] Jitendra Singh. *jitendra825*. URL: https://github.com/jitendra825. (accessed: 15.11.2023).

A LQR Gain Calculation Matlab

```
clear; close all; clc;
% This gain calculation script is based on the script of
%JITENDRA SINGH. Found on his GITHUB:
% https://github.com/jitendra825/Inverted-Pendulum-Simulink

% Equations for our system is presented in the report.

% Parameters from Our hardware:
Mc = 0.540 + 0.048;% [kg]mass of cart
l = 0.49; % [m] Pendelum length
Lcg = l/2; % [m] lenght to center of mass
l = Lcg;
m = 0.022;% [kg] Mass of pendelum rod
g = 9.81;

%Motor parameters:
Rm = 1.3;%[ohm] motor armature resistence
kb = 0.02;%[vs/rad]back emf constant
kt = 0.02;%[vs/rad]motor touque constant
r =  0.04/(2*pi); %[m]radius pulley motor (and encoder)
m_rotor = 0.100; %[kg] aprooximately
r_rotor =0.020/2; % m
Jm = 0.5*m_rotor*r_rotor^2;
M = Mc+Jm/r^2;

%Very rough estimate:
c = 20; %[N/(m/s)] viscous friction cart. Tias
b = 0.0001;% [Nm/rad/s]Viscous damping pendelum
I = 1/3*m*2*Lcg^2; % Uniform rod moment of inertia


%Calculations from Jitendra for LQR params estimate:
AA = I*(M+m) + M*m*(l^2);
aa = (((m*l)^2)*g)/AA;
bb = ((I +m*(l^2))/AA)*(c + (kb*kt)/(Rm*(r^2)));
cc  = (b*m*l)/AA;
dd  = (m*g*l*(M+m))/AA;
ee  = ((m*l)/AA)*(c + (kb*kt)/(Rm*(r^2)));
ff  = ((M+m)*b)/AA;
mm = ((I +m*(l^2))*kt)/(AA*Rm*r);
nn = (m*l*kt)/(AA*Rm*r);
A  =  [0 0 1 0; 0 0 0 1; 0 aa -bb -cc; 0 dd -ee -ff];
B = [0;0; mm; nn];
% Q and R are inspired by matlab example lqr:
Q = diag([1 1 0 0]); % Based on Matlab example, because only x and ...
   theta are output
R  = 1; % Based on matlab example, one input
KK = lqr(A,B,Q,R)

% after tuning with swing up active
% K = [-20   120  -20  5]
```

B Low Pass Filter Design

```
%%filter with 0.01 sample time in simulink

clc; close all; clear;


load("theta0_01.mat")
load("theteDotUF_0_01.mat")

t = theta.Time;
thetaData = theta.Data;
thetaDotData = theteDotUF.Data;


fs=100 %sample Freuency
fn=fs/2; % Nyquist Frequency
%fc=0.1; % Desired Cutoff frequency (example)
fc=1
[b,a]=butter(1,fc/fn);
% [h,w] = freqz(b,a,100);
% figure(1)
% subplot(2,1,1)
% plot(abs(h))
% subplot(2,1,2)
% plot(w)
%[b,a] = butter(6,20/(fs/2));
thetaFiltered = filter(b,a,thetaData);
thetaDotFiltered = filter(b,a,thetaDotData);

figure% TEST BODE OF FILTER
s = tf('s')
Gfilt = (b(1)*s +b(2))/(a(1)*s+a(2)) %This filter should be tried in ...
    simulink
bode(Gfilt)



figure %THETA
plot(t(:,1),thetaData(:))
hold on
plot(t,thetaFiltered(:))
legend('Theta UF','Theta F')

figure %THETADOT
plot(t(:,1),thetaDotData(:))
hold on
plot(t,thetaDotFiltered(:))
legend('ThetaDot UF','ThetaDot F')
xlim([102.65 105])
```