INVERTED PENDULUM

Model based control design

to

balance the inverted pendulum by LQR.

[](https://www.youtube.com/embed/3ScTiR30tBo?feature=oembed)

Table of Contents

[INTRODUCTION: 2](#_Toc41405623)

[DESCRIPTION: 2](#_Toc41405624)

[Modeling 3](#_Toc41405625)

[Nonlinear Modeling 3](#_Toc41405626)

[Linearization 4](#_Toc41405627)

[Discretization 5](#_Toc41405628)

[Controller Design 6](#_Toc41405629)

[Construction 8](#_Toc41405630)

[Mechanical 9](#_Toc41405631)

[Electronics & Programming 9](#_Toc41405632)

[Simulation (Rohan) 11](#_Toc41405633)

[Hardware Implementation 13](#_Toc41405634)

[Appendix 13](#_Toc41405635)

[Electronics Specification & Connection 17](#_Toc41405636)

[Encoder node: 17](#_Toc41405637)

[Main Controller | Scope node (hardware is same for both) 18](#_Toc41405638)

[EPOS motion controller node 19](#_Toc41405639)

[Programming 19](#_Toc41405640)

[Encoder Node 19](#_Toc41405641)

[Main Controller Node 19](#_Toc41405642)

[Epos Motion Controller Node: 20](#_Toc41405643)

# INTRODUCTION:

The task is to balance a simple pendulum around its unstable equilibrium, using only horizontal forces on the cart. The design process for this project required a great deal of planning and testing before deciding on a final design since there were so many alternatives to choose from. But due to the use of available DC motor with encoder that works on CAN bus made the task more complicated but doable.

# DESCRIPTION:

The Inverted Pendulum setup, shown in Fig.1, consists of a linear guide, timing belt and pulley. This linear guide further modified by adding encoder with 3D printed encoder mount for the measurement of angular position. The Microcontroller we use is Arduino MEGA and EPOS P 24/5 Programmable Positioning Controller DC motor Driver is used to drive a motor. The motor works on CAN bus so the CAN modules are used to communicate the data through the CAN bus. The cart inverted pendulum system is a very common system for testing many control methods such as PID, LQR, pole placement, genetic algorithms, LQG, etc. A normal pendulum that is a pendulum facing downward is a stable system because there is a gravitational force that will force the normal pendulum back to its starting position. Otherwise, the inverted pendulum or pendulum facing upwards is an unstable system. This system also has many problems such as nonlinear, underactuated, and complex systems. Controlling the inverted pendulum system is about the position and balance point of the pendulum.

# Modeling

## Nonlinear Modeling

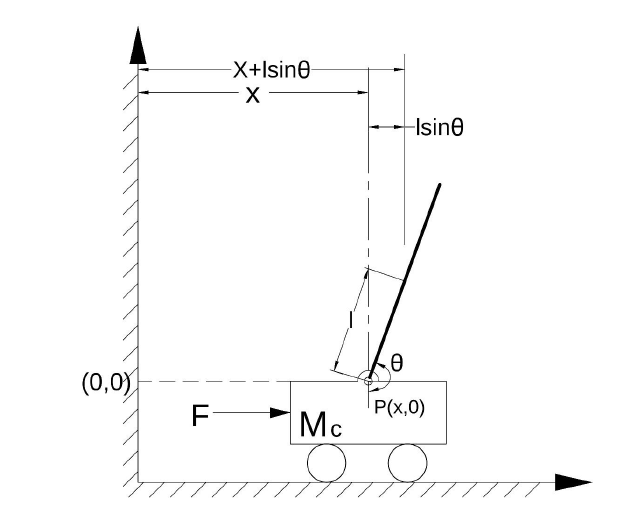


Figure 1 Complete mechanical assembly of inverted pendulum with belt and pulley

A cart inverted pendulum system is shown in Figure 1. The cart of cart is moved along X-axis by DC motor. Belt and pulley are used in mechanical transmission to provide the force from the actuator. The angular acceleration and the angular velocity of DC motor are controlled to get the acceleration and the velocity of the cart.

Variables used:

mass of cart= M (kg)

mass of pendulum =m (kg)

friction =c

gravity =g m/s^2

length of pendulum= l

torque constant= Kt

radius of pulley =r

just for the sake of simplification Kr=Kt/r

**Equation of the system**:

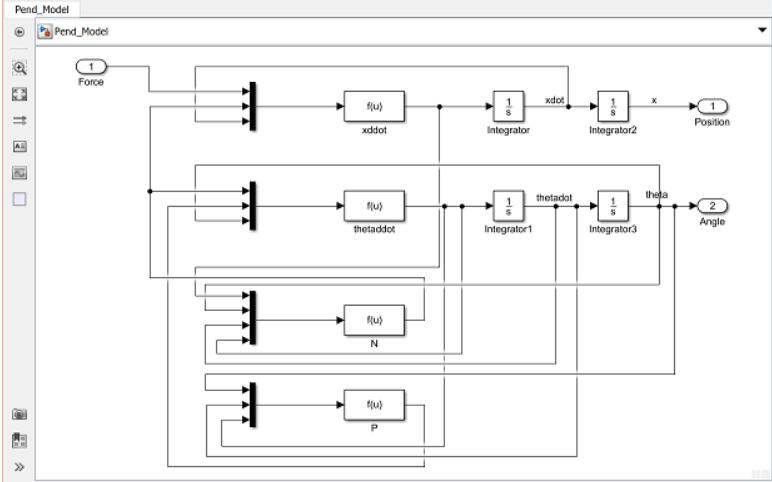
The equation of the whole system can be derived by summing the forces in the free-body diagram of the pendulum in the horizontal direction.

….(1)

To get the second equation of motion for this system, sum the forces perpendicular to the pendulum. Solving the system along this axis greatly simplifies the mathematics. You should get the following equation.

........(2)

The above non-linear equation can also be represented in Simulink as follows



## Linearization

Since this is a non-linear equation, we need to linearize it. This can be done by considering values of:

=

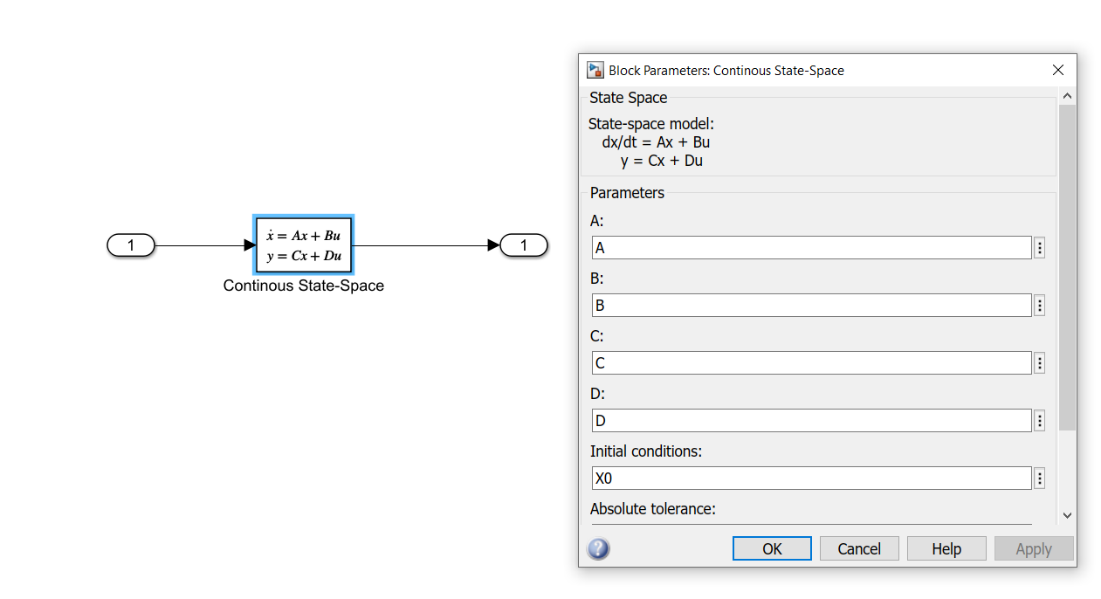
After substituting these values in equation 1 and 2 we get

(*I +m*)-*mgl …..(3)*

(*M + m*)…..(4)

After taking the Laplace transform of 3 and 4 and arranging the terms, we can represent the whole equation in the state space form as follows:

It can also be represented in Simulink as follows:



*(Representation of state-space in Simulink)*

## Discretization

Since we are implementing a digital controller it is essential to convert the above continuous state-space equations to a discrete form. This can be done by employing the MATLAB function c2d. In our case the sampling time is 1/100 sec/sample.

Ts = 1/100;

sys\_d = c2d(sys\_ss,Ts,'zoh')

We can simply retrieve the state space A,B,C and D by

Ad = sys\_d.a;

Bd = sys\_d.b;

Cd = sys\_d.c;

Dd = sys\_d.d;

## Controller Design

**Controller gain(K)**

**LQR method:**

LOR method is deployed to find the optimal controller gain. This is done by giving penalties to the state that needs more attention.

In our case the position of the cart was heavily penalized and then was the angle of pendulum.

The element (1,1) is the penalize the position of the cart and (3,3) penalize the angle of the pendulum.

The higher the value of R the slower would be the response rate of the system hence drawing less current.

Hence the gain matrix of the controller (K) can be found as follows

**Estimator Gain (L)**

**Pole Placement method:**

The observer poles needs to be 4-10 times faster than the controller poles. To do so we take the the slowest pole of the close loop system and then multiply somewhere in the range of 4-10 times its value.

In our case the value of the slowest pole was -2 so the poles we chose where [-15,-16,-17,-18]

The estimator gain matrix was determined as follows

P=[-15,-16,-17,-18]

L=place(A’,C’,P),

**Precompensation value:**

The controller we have designed so far meets our transient requirements, but now we must address the steady-state error. In contrast to the other design methods, where we feedback the output and compare it to the reference input to compute an error, with a full-state feedback controller we are feeding back all of the states.

This can be calculated as follows:

N = 1/dcgain(ss(A-B\*K,B,C,D))

**Gain to convert force into Ampere**

In the following system the input(u) given to the plant is in the form of force(N). Since we are actually implementing this, we need to convert the force into amperes as the motor drives the cart. This can be done as follows:

Torque=Force\* (Radius of Pulley)

Torque= Torque constant (Kt)\* Current

In our case Radius of pulley = 0.00636 m

Torque constant(Kt) =0.0243 Nm/A

Hence Current force relation will be:

**Force\*(0.26211) = Current.**

A gain of 0.26211 must be added in between the controller and plant block to convert the force into current.

# Construction

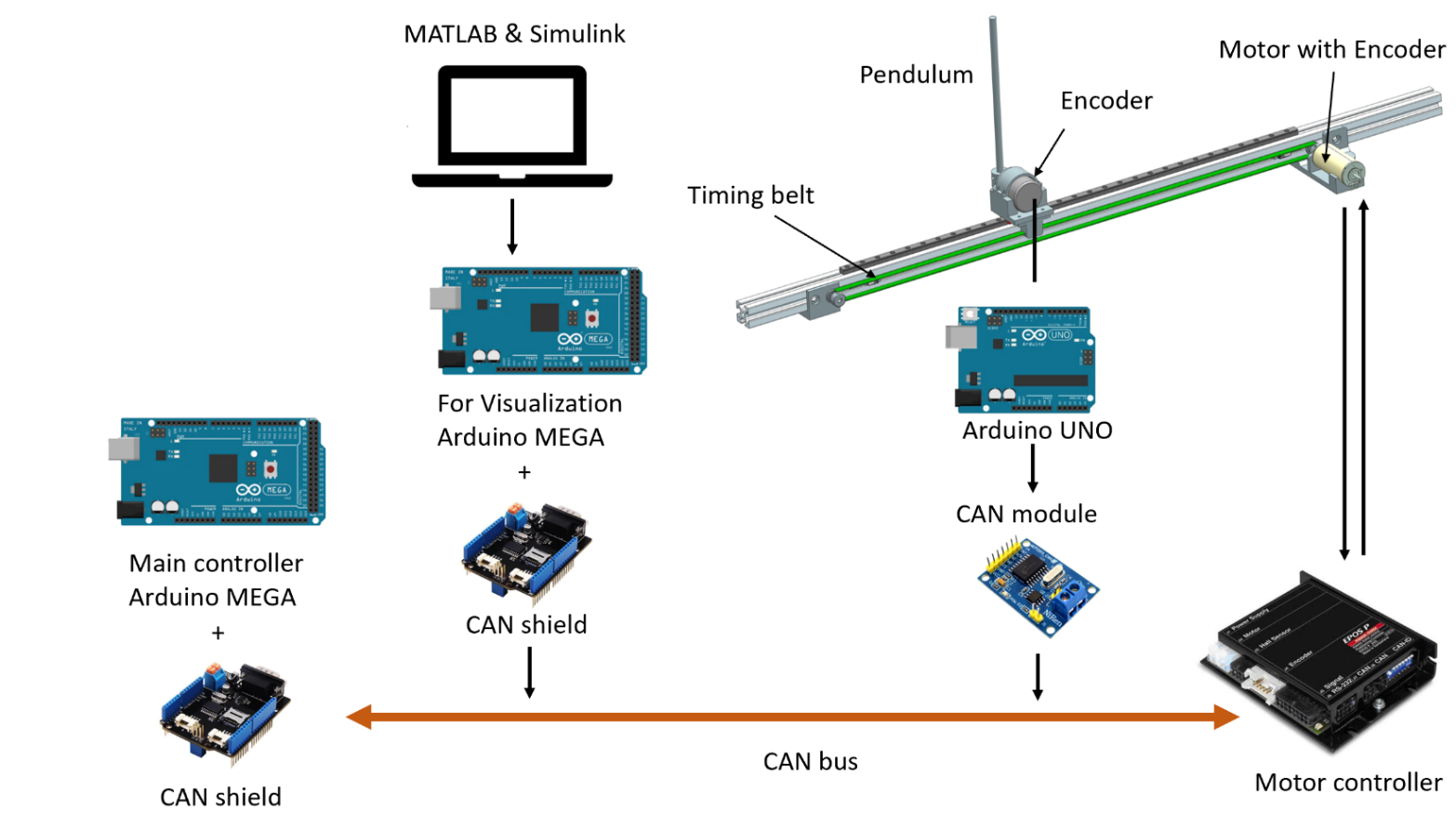


Figure 2 Hardware setup of inverted pendulum

* The Main controller is connected to the CAN bus through CAN shield and can read-write data from the CAN bus.
* The Arduino UNO is used to send the position of the pendulum to the CAN bus through CAN module.
* The position of cart is sent by the encoder on DC motor through DC motor controller.
* The other Arduino MEGA is used only for visualization of the data from the CAN bus to better understand what’s going on in the system. And helps to plot relation between the current, angle and position of cart.

The overall setup of inverted pendulum consists of following components:

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Part name** | **Quantity** |
| 1. | [Arduino MEGA](https://store.arduino.cc/usa/mega-2560-r3) | 2 |
| 2. | [Arduino UNO](https://store.arduino.cc/usa/arduino-uno-rev3) | 1 |
| 3. | [CAN Module](https://copperhilltech.com/mcp2515-can-bus-breakout-board-with-spi-interface/) | 1 |
| 4. | [Encoder LPD3806](https://www.aliexpress.com/item/32857035640.html) | 1 |
| 5. | [EPOS EC-max BLDC motor](https://www.maxongroup.us/maxon/view/product/motor/ecmotor/ecmax/ecmax30/272763) & [Encoder](https://www.maxongroup.us/maxon/view/product/sensor/encoder/Magnetische-Encoder/ENCODERMR/ENCODER-MR-TYPML-128-1000IMP-3KANAL/225778) | 1 |
| 6. | [EPOS2 P motor controller](https://www.maxongroup.us/maxon/view/product/control/Positionierung/378308) | 1 |
| 7. | Pendulum | 1 |
| 8. | [Pulleys](https://www.adafruit.com/product/1251) | 2 |
| 9. | [Timing belt](https://www.amazon.com/Mercurry-Meters-timing-Rostock-GT2-6mm/dp/B071K8HYB4/ref=sr_1_2?dchild=1&keywords=gt2+belt&qid=1590524162&sr=8-2) | 1.5 meters |
| 10. | [End switch](https://www.alliedelec.com/product/omron-electronic-components/ss-5glt/70175377/?utm_source=google&utm_medium=cpc&adpos=&scid=scplp70175377&sc_intid=70175377&gclid=CjwKCAjw_LL2BRAkEiwAv2Y3SSSRKXiNJ9Wq_DGJpqNDLwRnHE6-WCmMjkPcUQwcLqsw5nK1jYwAIhoCeyYQAvD_BwE&gclsrc=aw.ds) | 2 |
| 11. | [CAN shield](https://store.arduino.cc/usa/can-bus-shield-v2) compatible with Arduino | 2 |
| 12. | [Linear guide and slider](https://www.amazon.com/TEN-HIGH-linear-Miniature-Linear-MGN12C/dp/B06Y5GDP9Z/ref=sr_1_11?dchild=1&keywords=MGN9%2B600mm&qid=1590524422&s=industrial&sr=1-11&th=1) |  |

## Mechanical

The motor is mounted on 3D printed motor housing which is bolted on to the aluminum T-slot extrusion. The steel guide way with the slider is mounted on the aluminum T-slot as well.

A separate encoder housing is mounted on the slider to which the pendulum is mounted. The whole assembly on the slider is nothing but the cart of the inverted pendulum. The main aim is to keep the design light and robust and easy to assemble and disassemble.

The cart is displaced use a timing belt and pulley mechanism.

Future scope modification: Pendulum setup can be modified with additional link and an encoder to perform balancing of double pendulum and also triple pendulum in the future.

To increase the lenth of the guideway will require a longer timing belt with an idler pulley in the mechanism for better stability and response.

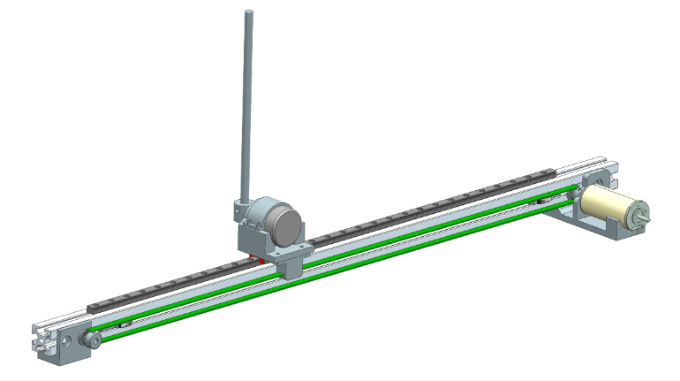


Figure 1 Mechanical setup of inverted pendulum

The whole setup can be modified for future extension in the projects like implementing double and triple pendulum balancing.

## Electronics & Programming

As explained in overall setup there are 3 main Electronic Sensor/Actuator.

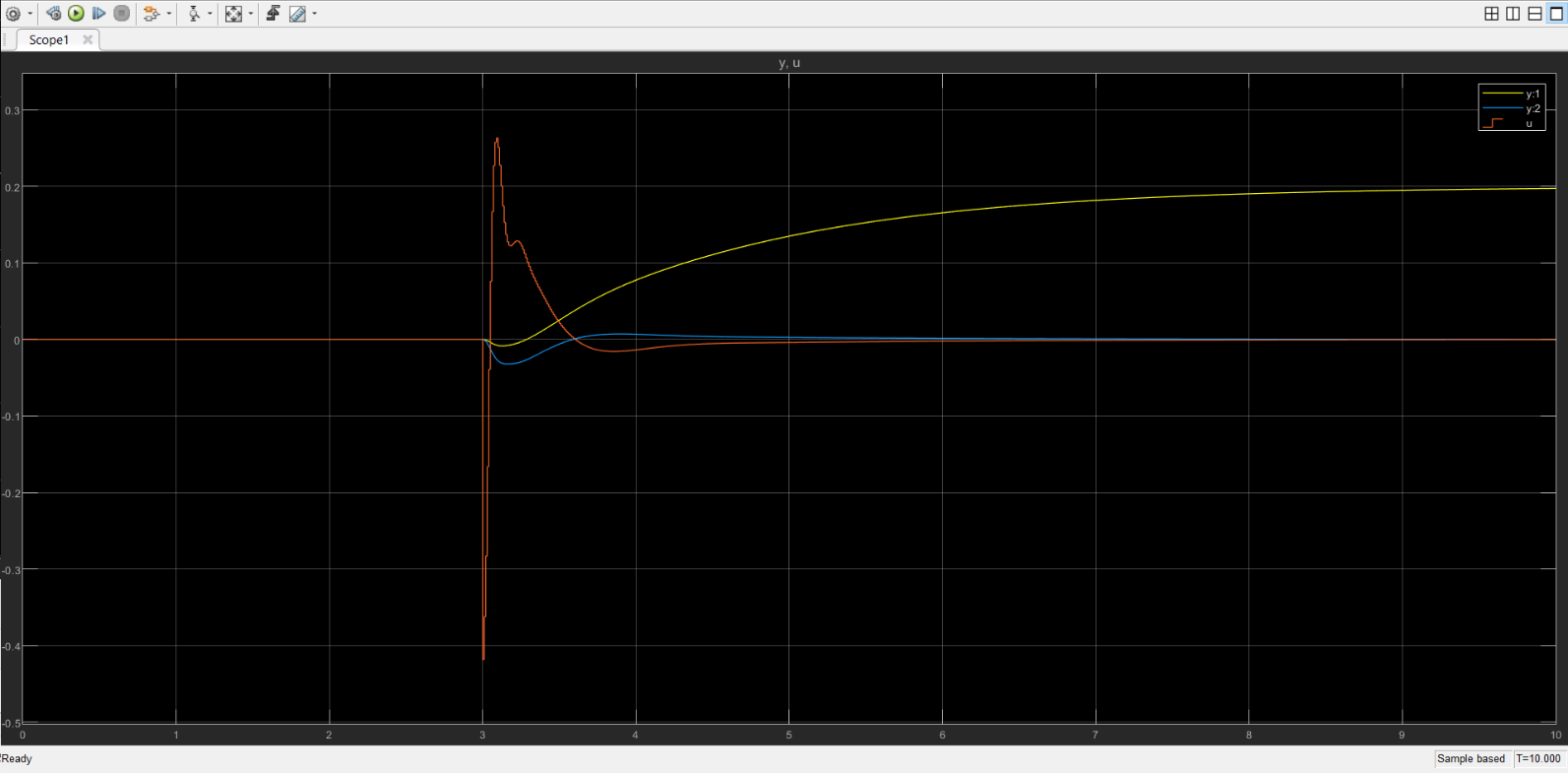
1. Encoder attached at the base of the Pendulum to measure the Angular position, encoder itself is mounted the carriage and part of the cart.
2. Encoder attached to the BLDC motor which measures relative position of the cart on the linear rail.
3. BLDC motor actuates the cart by provide commanded force via pully mechanism.

There are 4 programmable microcontroller/nodes for distributed computing. For communication between these nodes we are using CAN protocol. 3 of the nodes are Arduinos with CAN shield/module attached on it and 1 EPOS Motion controller node with CANopen protocol.

1. Encoder slave node which processes the angular encoder pulses and sends angular position as CAN msg on the CAN bus
2. Main Controller master node has control logic programmed onto it to receive pendulum and cart position, process it, and send current command for the motor on the CAN bus.
3. Epos motion controller slave node has 1 BLCD motor and encoder attached to it which deals with executing the current command received from the master node and sending the cart position reading to the master node via CAN bus
4. Scope slave node just reads all the CAN messages and send them to Simulink via USB connection and Simulink shows that data onto the scope for better visualization.

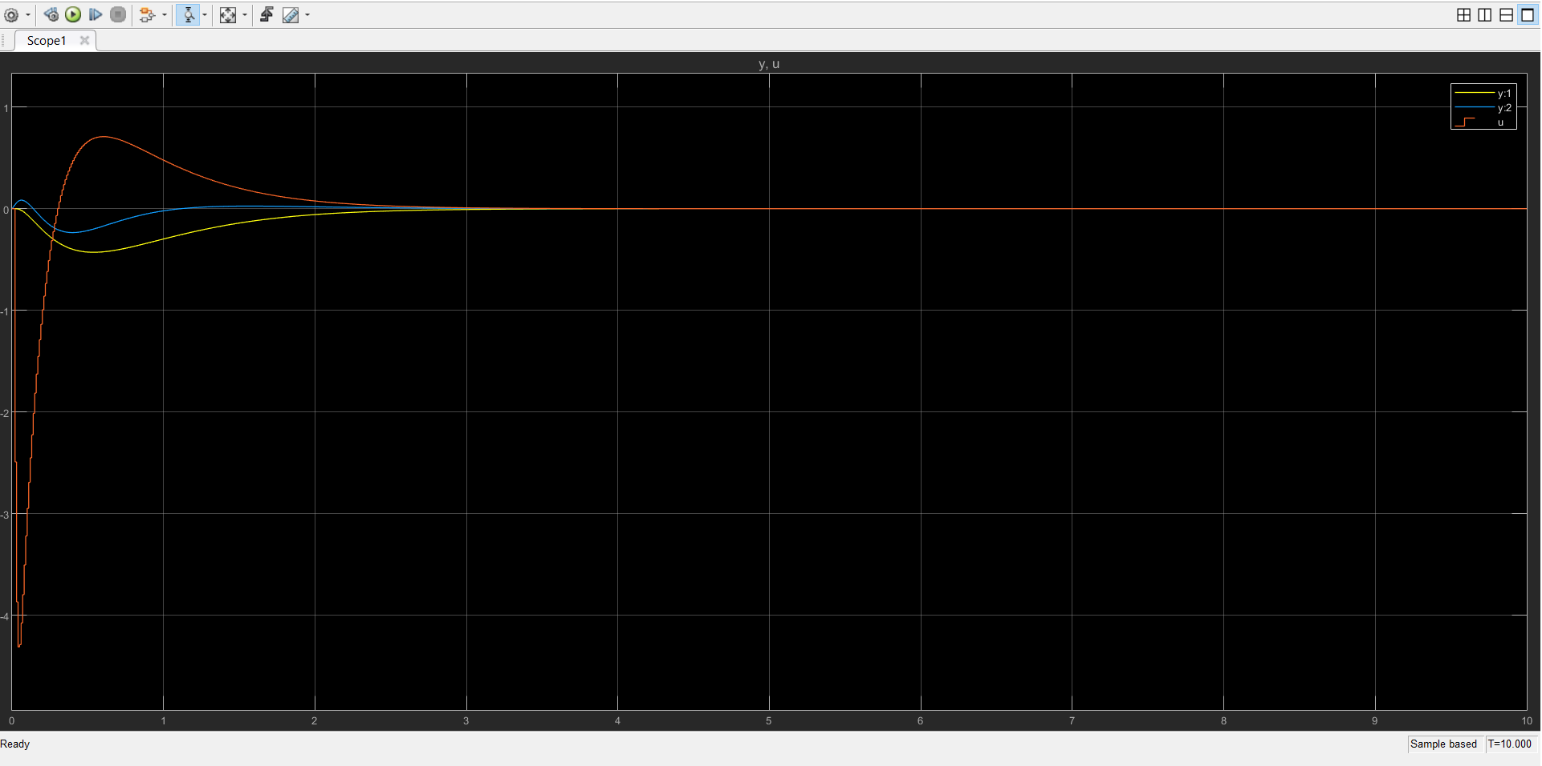
# Simulation (Rohan)

**When cart is displaced by 200mm the current required the angle and position is 0.27 Amp**



*(When the cart is displaced the pendulum angle changes due to its inertia and there’s a small dip in the angular plot. The system tries to correct the error to make it stable )*

**On certain disturbance to the pendulum, the system tries to stabilize itself. Once it gets stabilized the cart goes back to initial position.**



*(On giving a certain disturbance of 2 rad/sec to the pendulum the system tries to correct itself as fast as it can. The current drawn is around 4 Amps. The cart, after stabilizing the pendulum in upright position comes back to its original position)*

# Hardware Implementation

The controller used is Arduino MEGA with CAN shield as the data is communicated through CAN bus. The DC motor with Encoder is used which makes use of EPOS motor controller that works on CAN.

# Appendix

## Modeling Specification

M = 0.195; % Mass of cart (Kg)

m = 0.17; % Mass of pendulum(Kg)

b = 0.5; % Co-efficient of Friction(N\*sec/m)

l = 0.2; % length of pendulum(m)

g = 9.8; % gravity (m/sec^2)

I = (m\*l^2)/3; % Moment of inertia(Kg\*m^2)

X0 = [0 0 pi 0]'; %initial conditions

X\_d = [0 0 0 0]'; %desired conditions

Ts = 1/100;

p = I\*(M+m) + M\*m\*l^2; %denominator for the state space

**State-space**

**Gain values for LQR**

Q = C'\*C;

Q(1,1) = 5000;

Q(3,3) = 1000;

R = 0.1;

**Controller gain**

Kd= [-1.602296539197740e+02,-70.952507750545960,1.553862832586561e+02,22.516437624066600]

**Estimator gain**

Ld=[1.42,-0.03; 49.94,-2.06 ;0.02,1.44 ;-2.07,52.84]

**Precompensation Value**

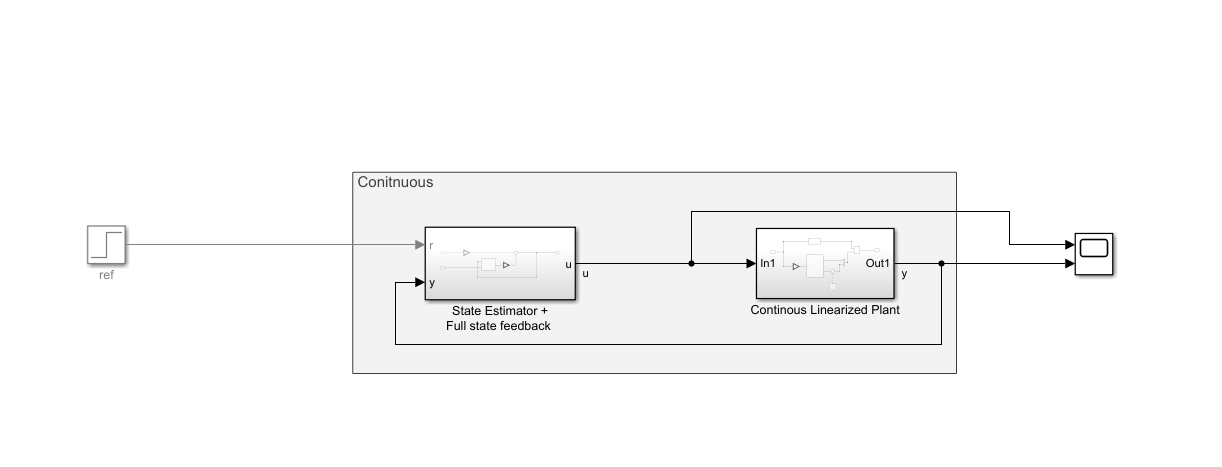
Ndc= -1.60e+02

**Force to current converted value**

N to Amp= 0.26211

**Continuous State Space Simulink Model**

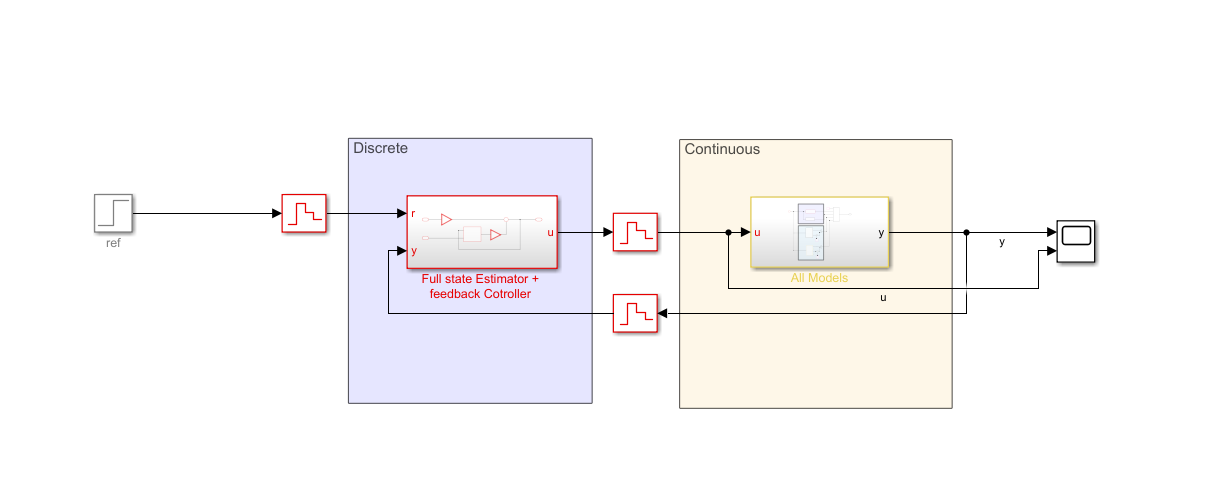
The states of the plant are fed to the estimator and then to the feedback controller. This block was made just to understand the dynamics of the system and it needs to be discretized to actually implement on hardware

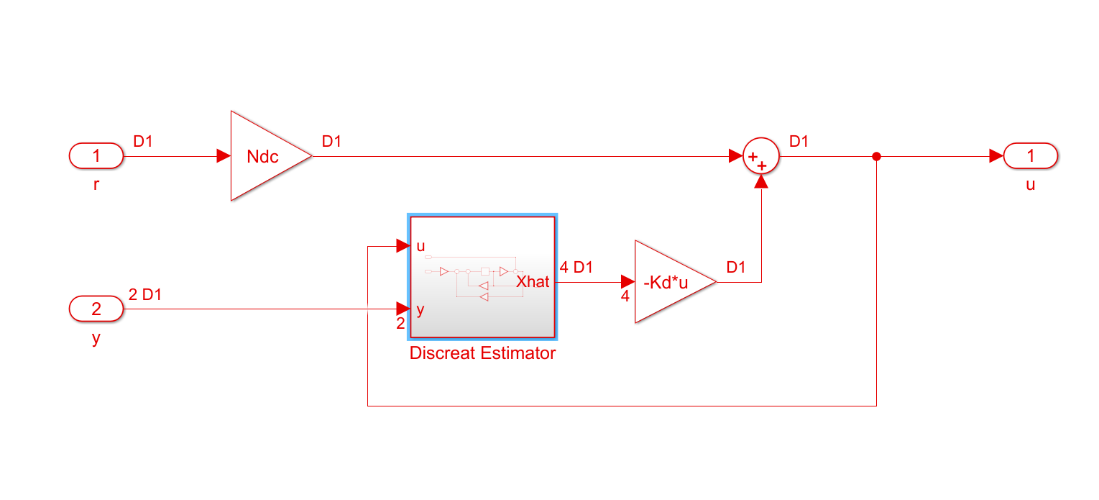


*(This is a continuous representation of the inverted pendulum)*

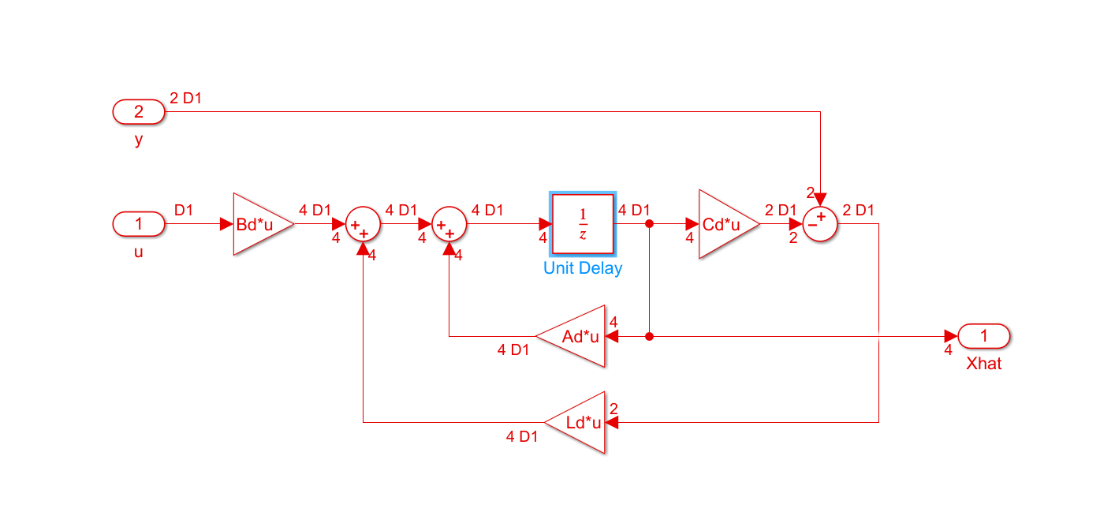
**Discretized State Space Model**

Since we are actually implementing the Simulink block in the controller, it is necessary to discretize the block. The sample time is 100Hz. To do so zero order hold blocks were used at appropriate places and discrete integrators were used.

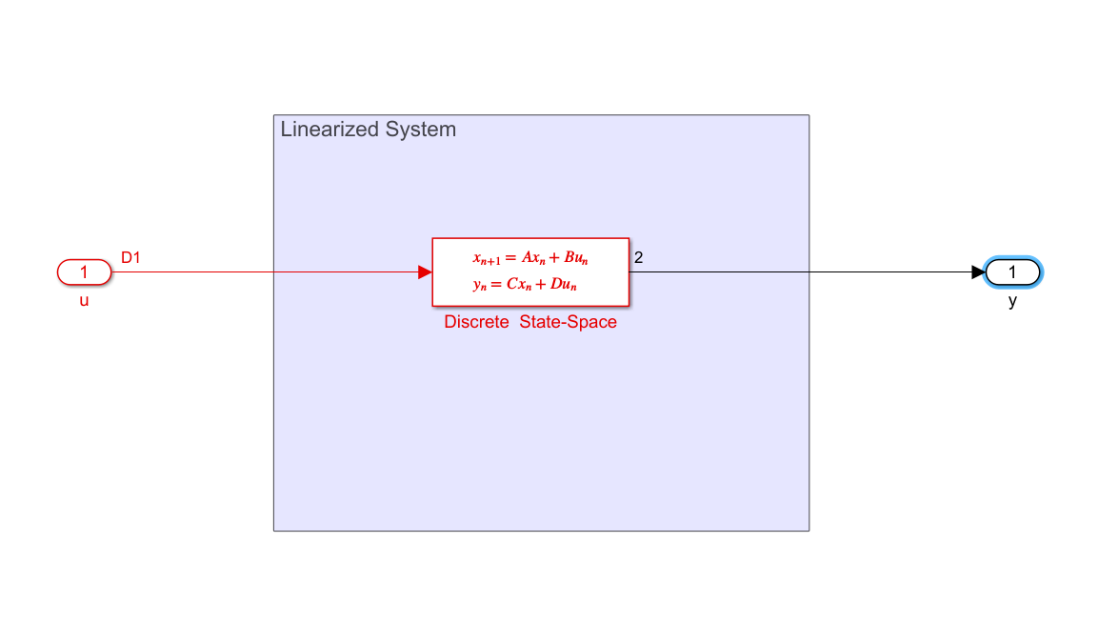




*The inner circuitry of the Full state estimator.*



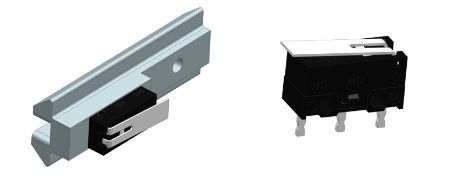
*The inner circuitry of Discrete Estimator*



*The inner circuitry of the Plant*

**Note: You can also simulate the whole system in Sims-cape which would give you an idea of how the system would behave virtually.**

Link:<http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SimulinkSimscape>

End sensor mount:

* The end sensor mounts are also 3D printed with PLA material.
* Adjustable to change the effective length of the system. Center to center distance i.e. ‘x’.

Figure 4 Pendulum cart with encoder

Important values form the mechanical setup:

* 1 revolution of DC motor = 40mm of distance traveled by cart.
* Length of the rail = 520mm (with which the cart can stabilize the pendulum)

## Electronics Specification & Connection

### Encoder node:

A picture containing clock

Description automatically generated

Figure 5 Encoder Node schematic

1. Arduino UNO
2. Encoder IPD3806 dual channel 600PPR
3. CAN module MCP2515
   1. as shown in the figure above channel A and B of the encoder is connected to pin 2 and 3 of the Arduino as those are the only two pins with interrupt enabled.
   2. MCP2515 uses SPI communication with Arduino, CAN module pins for the communications are SCK, MISO, MOSI and CS which are connected to respective Digital pins 13, 12, 11 and 10 where 10 is slave select pin
   3. MCP2515 port CAN H and CAN L are connected to CAN H and CAN L of the CAN bus.

### A circuit board Description automatically generatedMain Controller | Scope node (hardware is same for both)

Figure 6 Main controller master node | Scope slave node schematic

1. Arduino mega 2560
2. CAN Bus Shield ( <https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/> )

### EPOS motion controller node

A picture containing table

Description automatically generated

Figure 7 EPOS motion Controller

1. Maxon EC-max BLDC motor
2. Maxon MR encoder
3. Maxon EPOS2 P controller
4. Limit-switches (for safety)

Connection are made as shown in the figure above for more details about the wiring connection for EPOS controller refer the documentation [here](https://www.maxongroup.com/medias/sys_master/root/8834322563102/EPOS2-P-24-5-Cable-Starting-Set-En.pdf). also check out this getting started document for EPOS2 P controller [here](https://www.maxongroup.com/medias/sys_master/root/8834322432030/EPOS2-P-24-5-Getting-Started-En.pdf).

## Programming

Encoder Node: Its and Arduino Uno with CAN module as you already know, You can find the most up to date code for it [here](https://github.com/imkishan96/Inverted_Pendulum/blob/master/CAN/CAN_Send/CAN_Send.ino).

Main Controller Node:

* It is programmed using Simulink support package for Arduino hardware. You can find details about how to set up Simulink support package [here](https://www.mathworks.com/help/supportpkg/arduino/index.html).
* Simulink model deployed onto the main controller is shown in the figure below. Find model [here](https://github.com/imkishan96/Inverted_Pendulum/blob/master/fresh_model/Arduino_Fresh_model.slx).

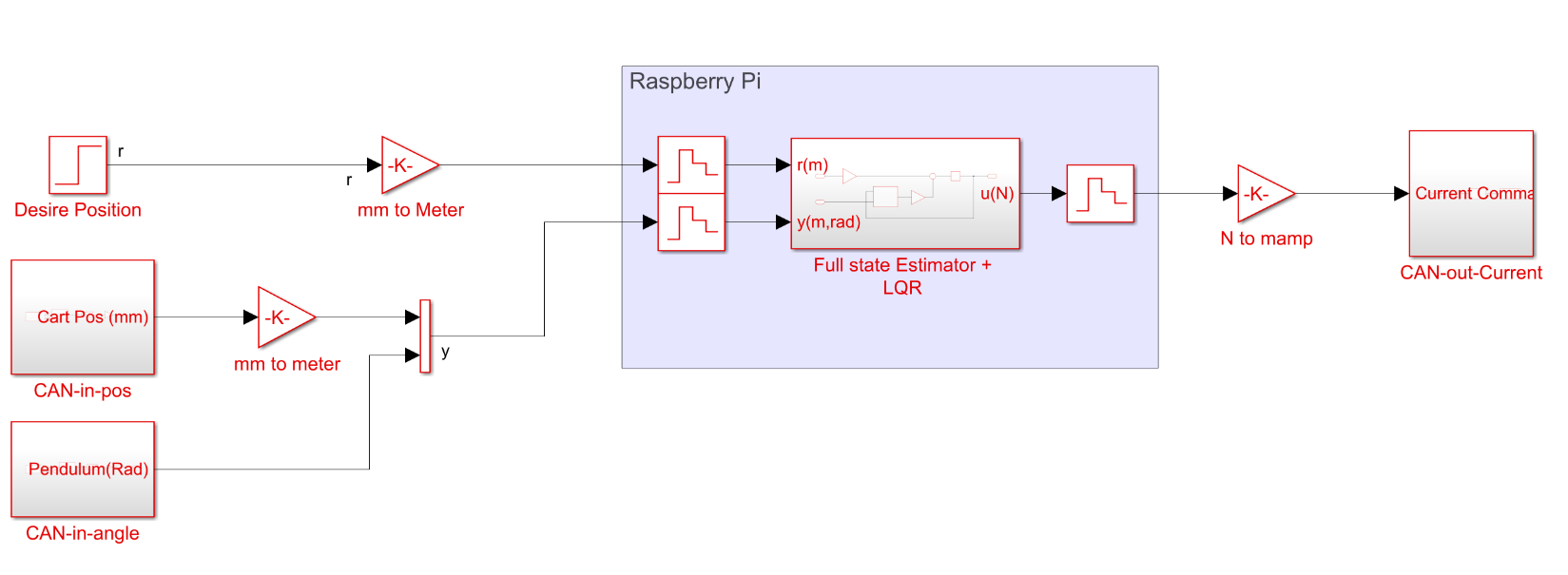


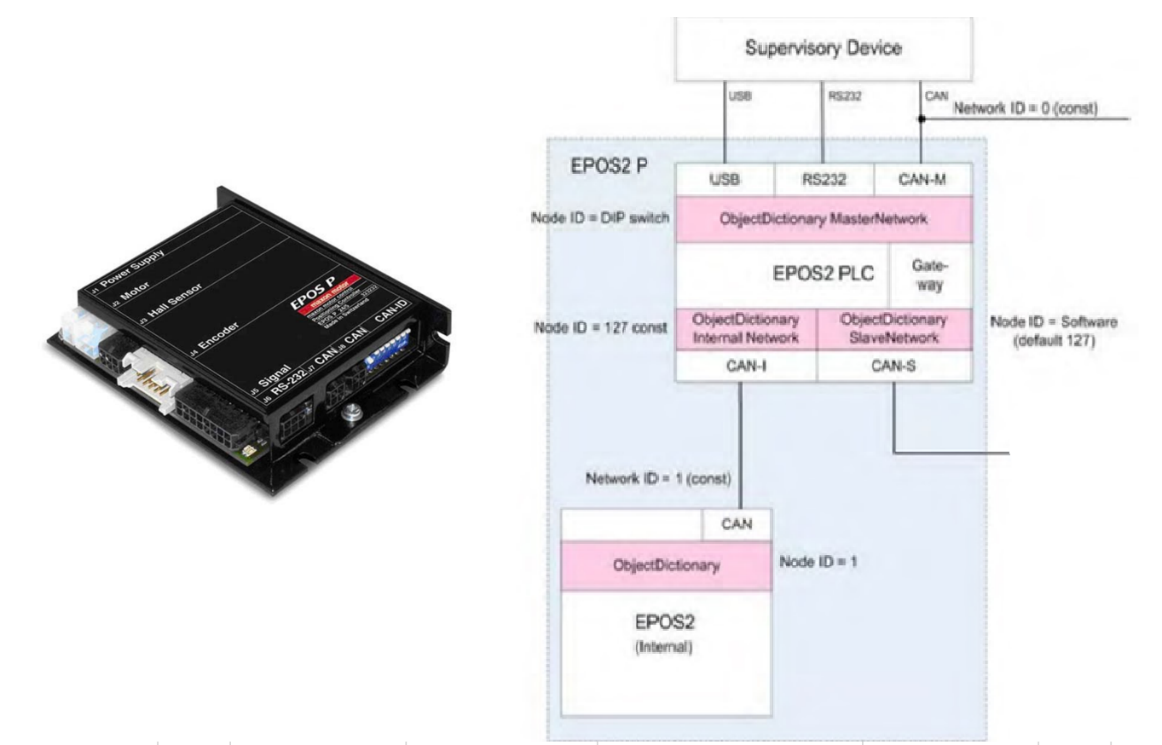
Figure 8 Main controller node Simulink model

* But before you deploy this model you need to run the MATLAB file which you can find [here](https://github.com/imkishan96/Inverted_Pendulum/blob/master/fresh_model/fresh_code.m). Also make sure to select proper model settings as described below.
  + Hardware Implementation -> Hardware Board: Arduino Mega 2560
  + Target hardware resources -> SPI Properties -> CAN SPI SS pin: 9 or 10 depending upon the CAN shield
  + CAN Properties -> Interrupt pin: 2 (Check CAN Bus speed and oscillator frequency)

### Epos Motion Controller Node:

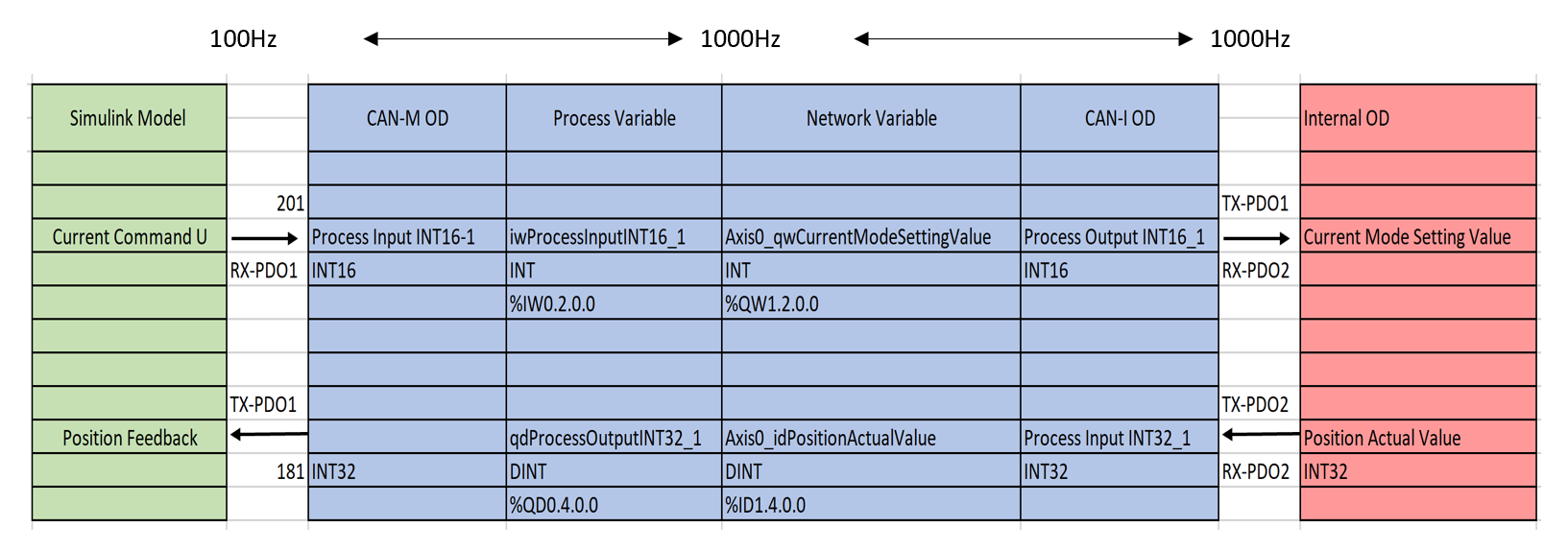
If you are new with EPOS studio check out [this](http://storkdrives.com/wp-content/uploads/2013/10/28b-Motion-Control-for-Newbies.pdf) document before tinkering.

Motor-controller EPOS PLC:



The flow of data through the motor controller is explained by the following CAN-OPEN protocol:

CAN-OPEN protocol:



* The Simulink model is made available to the Arduino and then transferred to CAN bus through the CAN Module.
* This data is made available in the motor PLC in CAN-M OD and is process made available in the CAN-I OD.
* The data from the plant is transferred to CAN at 100Hz
* Exchange rate between the process variable and network variable is 1000Hz which is fast. The CAN-I OD exchange data at 1000Hz and can go beyond it.
* The data is transfer to the CAN BUS at 500kbits
* The INTERNAL OD then fetches the data from CAN-I OD and the motor is then commanded with the control signals.

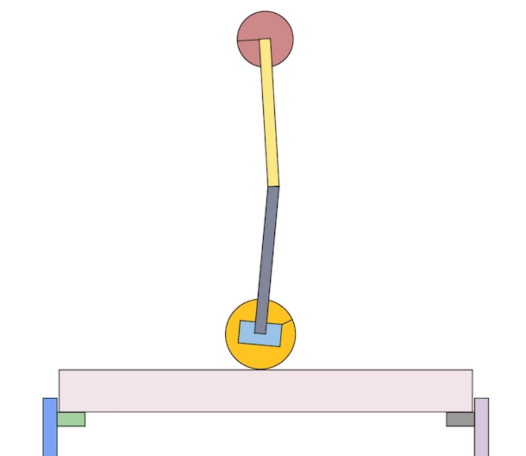
**Note:** To know more about the mathematical and state-space representation please refer the site given below.

<http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling>

**Conclusion:**

State feedback control with a full-state estimator for a cart invert-ed pendulum system has been successfully conducted. The gain state feedback 𝑲 has been obtained using Linear Quadratic Regulator. The gain of full-state observer 𝑳 also been obtained using pole placement control method. The simulation result shows that the cart inverted pendulum system can reach the desired position and angle with the optimal responses. The LQR method gives more close results as compared to the direct derivative method.

**Future Scope:**



* The complete setup can me tuned and can be used to perform the stabilization of inverted pendulum with upright position.
* The setup can be modified by having two links of pendulum for stabilization and can be tuned to perform the controller to stabilize the two links.
* Implementing the swing up command to perform the controller action with the pendulum starting from vertical downward position.