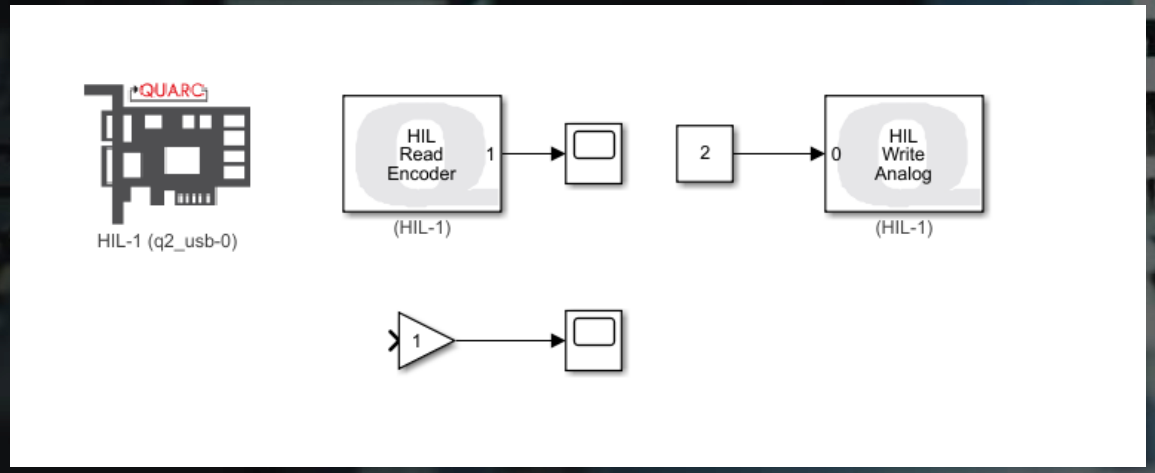
Inverted Pendulum Balancing System

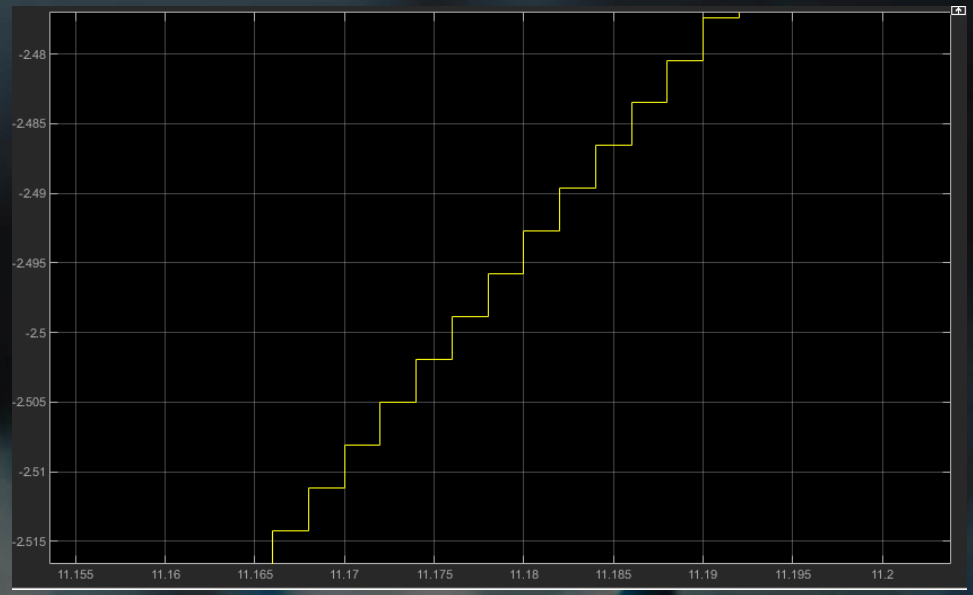
By: Seif Elkhashab & Mark Charnot

Getting To Know The Hardware

- Before Starting Any project, you must familiarize yourself with the hardware.
- Hardware in our system:
  - 1 Motor
  - 2 Encoders
  - 1 Amplifier
  - HIL-1 Microcontroller



Getting To Know The Hardware

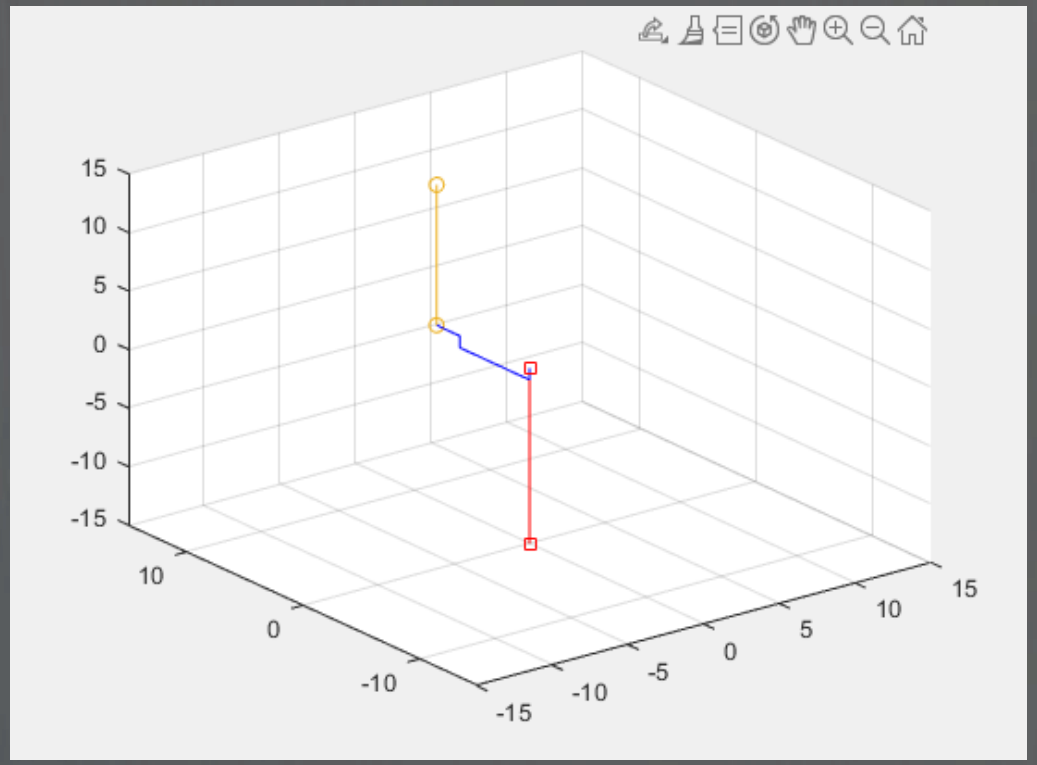


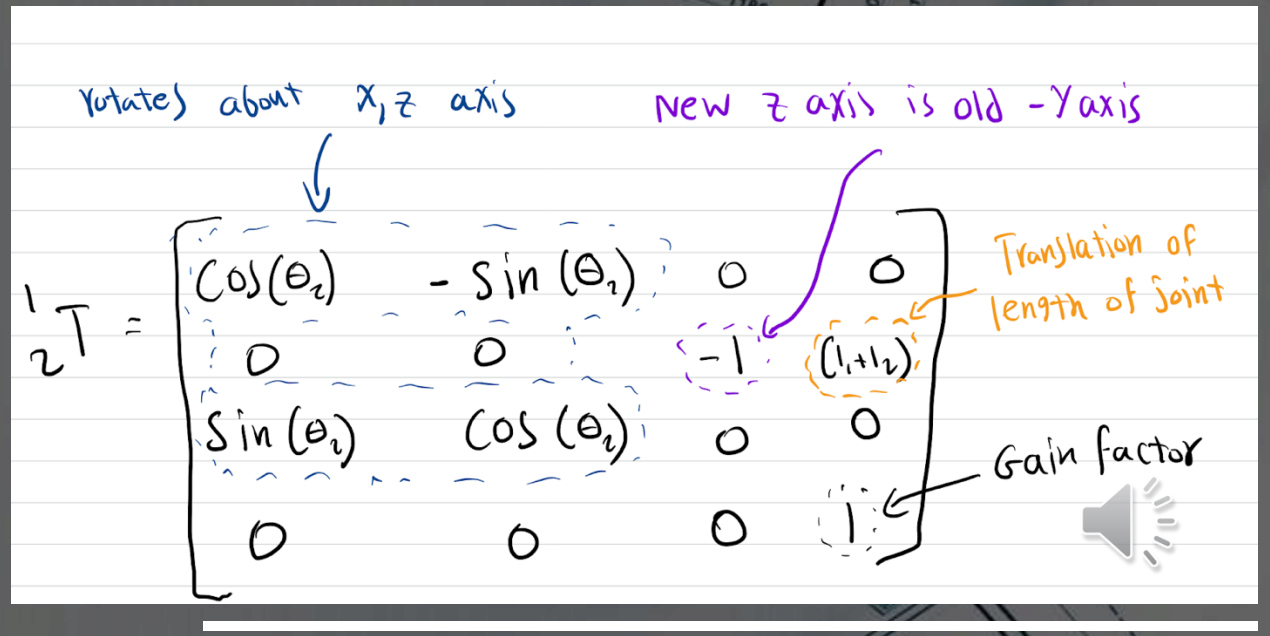
- The Encoders are **<u>Incremental</u>** and samples 4096 counts per revolution through a light sensor.

Building a Model

It's Important to Build a Model to be able to predict the behavior of the system.

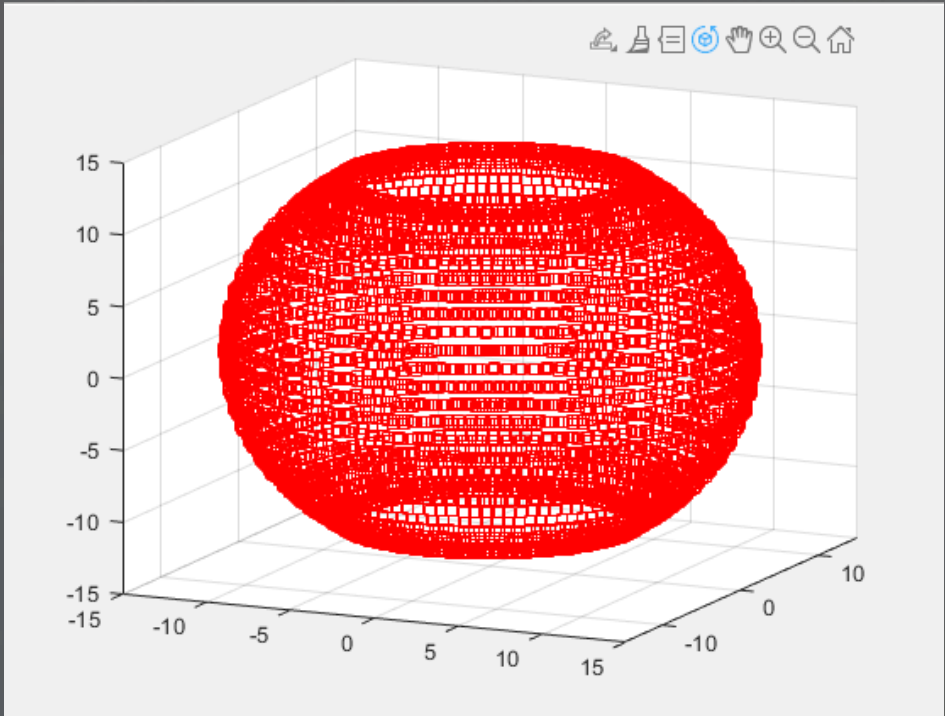
1. The Transformation Matrices Were Found
2. Pendulum was graphed





Building a Model

- The Work Envelope was also found which matched our predictions.
- The model had the ability to go to any point given that it in within the work envelope.



Adding Dynamics to the Model

The **Lagrangian** was used to formulate the governing mathematical representation of the system. **L = K – V**

If we rearrange this equation below, solving for acceleration, we can build a physics model to represent our system.

$$[\theta_1 + \theta_2 \sin^2(q_2)]\ddot{q}_1 + \theta_3 \cos(q_2)\ddot{q}_2 + 2\theta_2 \sin(q_2)\cos(q_2)\dot{q}_1\dot{q}_2 - \theta_3 \sin(q_2)\dot{q}_2^2 + \theta_5\dot{q}_1 = v$$

$$\theta_3 \cos(q_2)\ddot{q}_1 + \theta_2\ddot{q}_2 - \theta_2 \sin(q_2)\cos(q_2)\dot{q}_1^2 - \theta_4 g \sin(q_2) + \theta_6\dot{q}_2 = 0$$

System Parameter Identification

Some of the System Parameters depend on the physical characteristics of system, Link length, weight, etc.

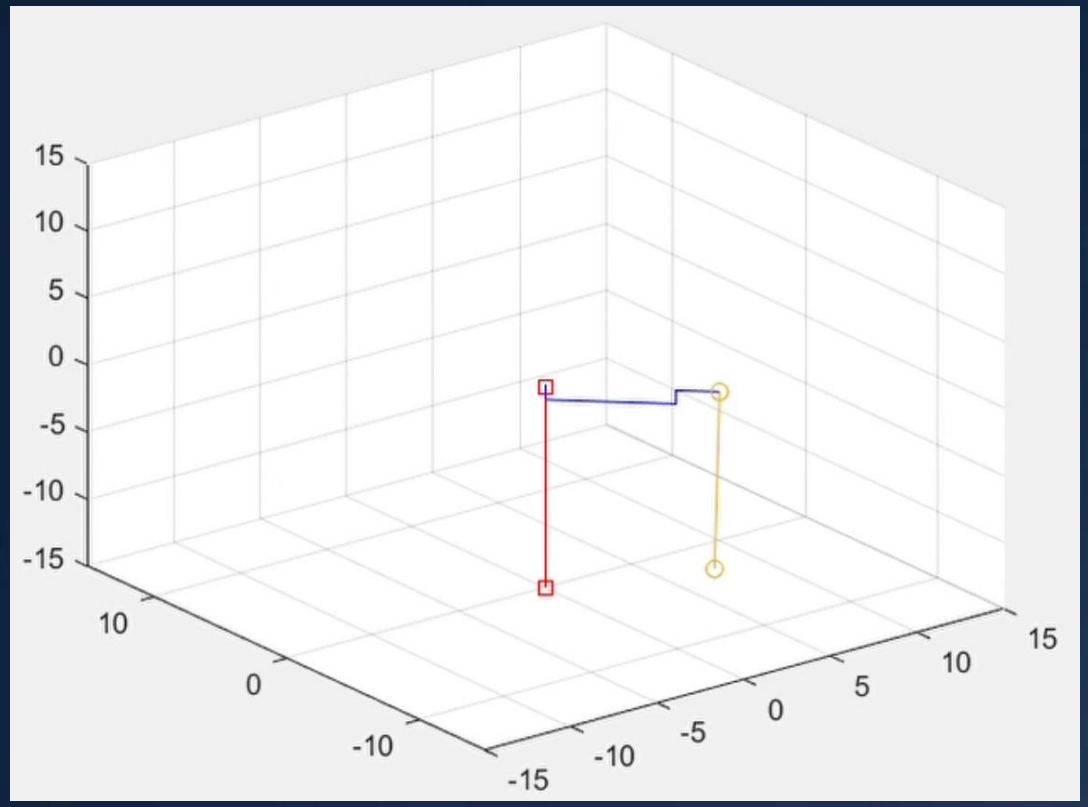
Instead, by collecting enough data and comparing between the actual and system states, the least squares method was used to find the ideal system parameters.

$$
\begin{aligned}
\theta_1' &= J_1 + m_2(l_1 + l_2')^2 \\
\theta_2' &= \tfrac{1}{3} m_2 (l_2)^2 \\
\theta_3' &= \tfrac{1}{2} m_2 (l_1 + l_2') l_2 \\
\theta_4' &= m_2 l_{c_2} \\
\theta_5 &= \beta_1 \frac{R_a}{k_r k_t} + k_r k_v, \\
\theta_6 &= \beta_2 \frac{R_a}{k_r k_t}.
\end{aligned}
$$

$$
A = \begin{pmatrix}
\overline{H}(t_1, t_0) & \overline{F}(t_1, t_0) \\
\overline{H}(t_2, t_0) & \overline{F}(t_2, t_0) \\
\cdots & \cdots \\
\overline{H}(t_N, t_0) & \overline{F}(t_N, t_0)
\end{pmatrix}
$$

$$
d \triangleq \begin{pmatrix}
\int_{t_0}^{t_1} v(s)\dot{q}_1(s)ds \\
\int_{t_0}^{t_2} v(s)\dot{q}_1(s)ds \\
\cdots \\
\int_{t_0}^{t_N} v(s)\dot{q}_1(s)ds
\end{pmatrix}
$$

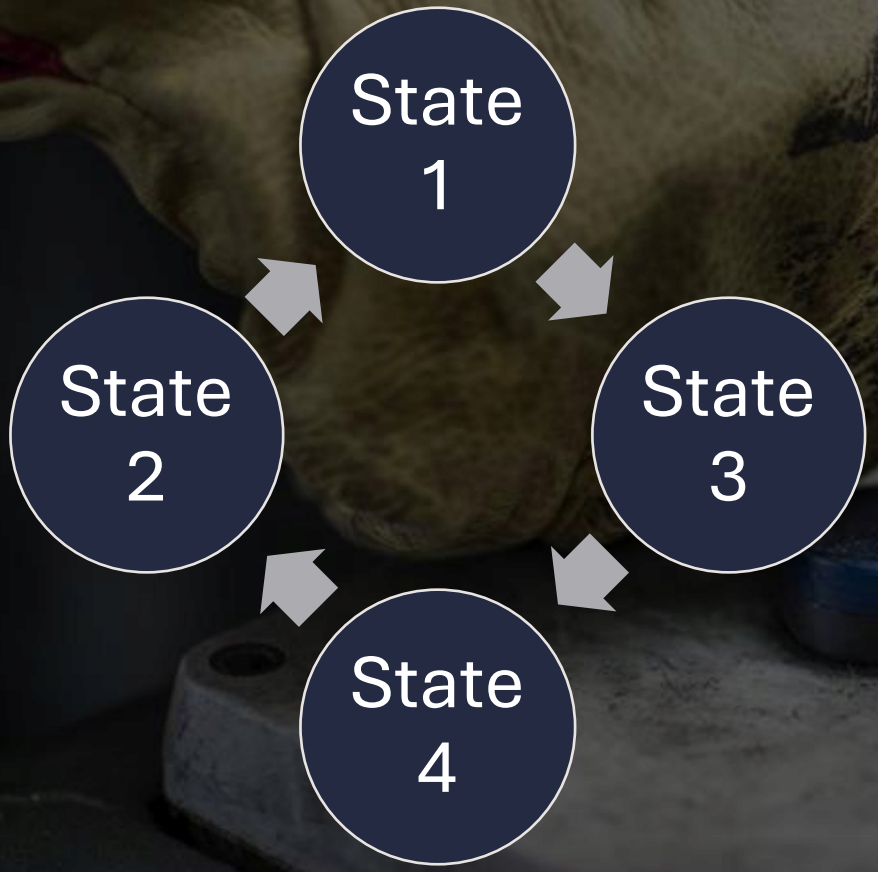
Building The Physics Engine



- Now that the ideal system parameters were found, the simulation now matches the actual pendulum very well.

Now, we can use the prediction from our simulation to give commands to our actual system in order to reach the desired position.

Creating the State Switch

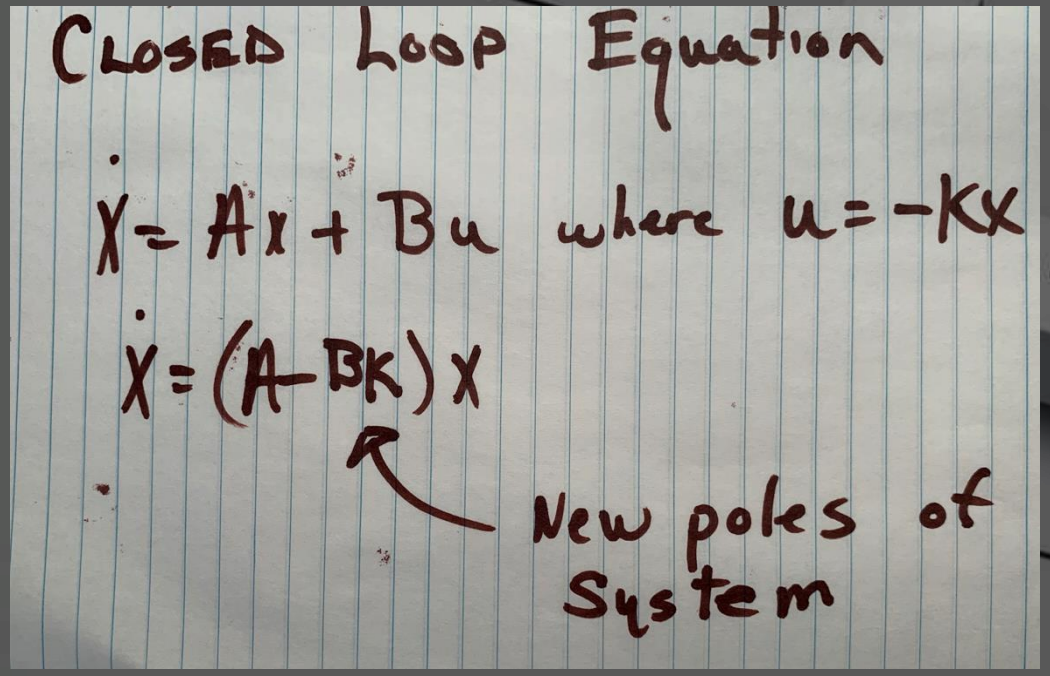


- The State Switch had 4 States:
  1. Swing Up State:
  2. Wait State:
  3. Balance State:
  4. Go Back To Initial 0:

```
% go from state 1 to state 3
if (currentstate==1) && (p2 < (200*pi/180) + 2*n*pi && p2 > (160*pi/180) + 2*n*pi)
    newstate=3;
% go from 3 to 4
elseif(currentstate==3) && (p2 > (5*pi/180) + 2*n*pi && p2 < (160*pi/180) + 2*n*pi)
    newstate=4;
elseif(currentstate==3) && (p2 < (-5*pi/180) + 2*n*pi && p2 > (-160*pi/180) + 2*n*pi)
    newstate=4;
elseif (currentstate==1) && (p2 > (-200*pi/180) + 2*n*pi && p2 < (-160*pi/180) + 2*n*pi)
    newstate=3;
% go from state 4 to state 2
elseif (currentstate==4) && (p1 < (10*pi/180) + 2*n*pi) && (p1 > -(10*pi/180) + 2*n*pi) && abs(v1) < 0.05
    newstate=2;
% go from state 2 to state 1
elseif (currentstate==2) && (p2 < (5*pi/180) + 2*pi*n  && p2 > (-5*pi/180) + 2*pi*n && abs(v2) < 0.01)
    newstate=1;

else
newstate=currentstate;
end
```

Building The Controller Feedback Loop



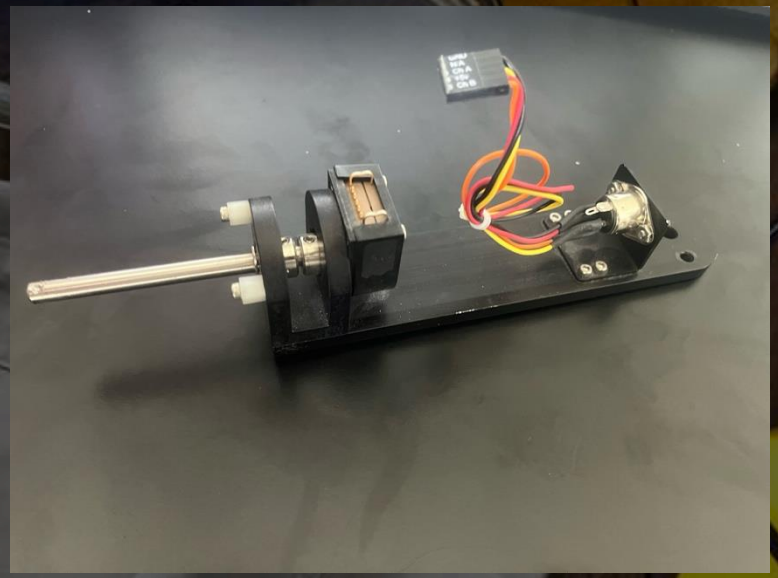
To create a balance algorithm, we simply linearize our dynamics about the point (0,0,0,0)

We do this, as this is the point that we defined as the fully upright position, with zero velocity.

To linearize, we take the Jacobian of the dynamics with respect to our states.

Problems Faced & Solutions Found

Hardware



Positive Feedback

The End