1. Items done this session:

Last time, we'd fixed the swing up part, that the robot can successfully swing up, let it fall down and wait until q2dot slow down below 0.5 then swing up again.

For this time, adding the improvements:
- Move "q1" during balance.
- Create 4th Hybrid state to reposition "q1" before swinging up.
- Integrate external sensor (balance beam rail) to move "q1".

We started to build the balance block for the robot to stay balance when it swing the stick to around the top region.

Firstly, for the negative feedback system H(s) = AX + Bu, we want to calculate the value k of the (-k) gain in order to move the poles to desired value.

For linearization of A and B, A = { df(i) / dX(j) } 4 by 4 matrix for i, j = 1~4 and B = { df(i) / du } 4 by 1 matrix for i = 1~4.

Therefore, the A =
[0, 1, 0, 0;
 0, A22, A23, A24;
 0, 0, 0, 1;
 0, A42, A43, A44]

and the B =

[0; B2; 0 ; B4]
where

A22 = -(theta2*theta5)/(-theta3^2 + theta1*theta2)

A23 = -(49*theta3*theta4)/(- 5*theta3^2 + 5*theta1*theta2)

A24 = (theta3*theta6)/(-theta3^2 + theta1*theta2)

A42 = (theta3*theta5)/(-theta3^2 + theta1*theta2)

A43 = (49*theta1*theta4)/(- 5*theta3^2 + 5*theta1*theta2)

A44 = -(theta1*theta6)/(-theta3^2 + theta1*theta2)

B2 = theta2/(theta1*theta2 - theta3^2)

B4 = -(theta3)/(theta1*theta2 - theta3^2)

poles = -[5.0, 5.1, 5.2, 5,3]

[theta1, ..., theta6] are system parameters that are estimated from data collecting lab.

With numerical A and B, now we can calculate using MATLAB function place, k = place(A, B, poles) and we got k = [-0.8285, -1.2024, -15.0566, -1.9463].

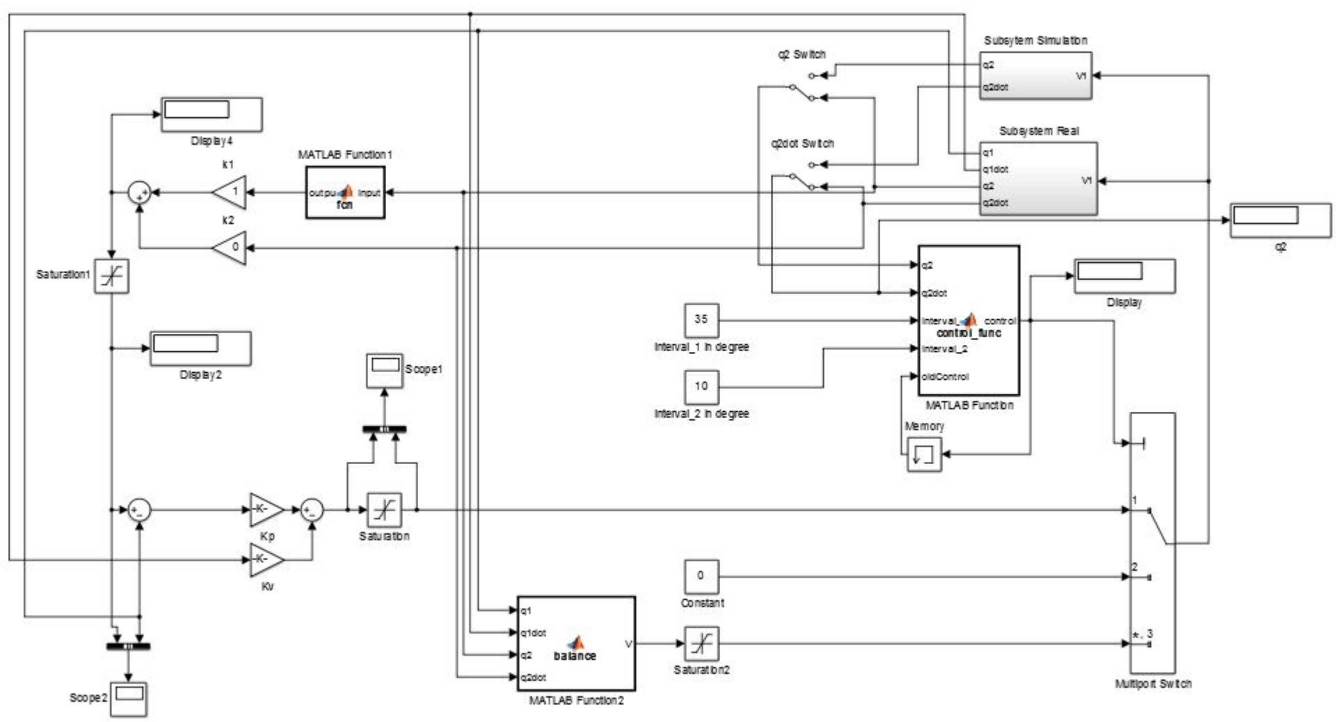
Finally, u (or v1) = -k*X.

We built the negative feedback as a balance function, taking input of X = [q1, q1dot, q2, q2dot] and output desired u (v1) to the "state3" of multi-port switch.

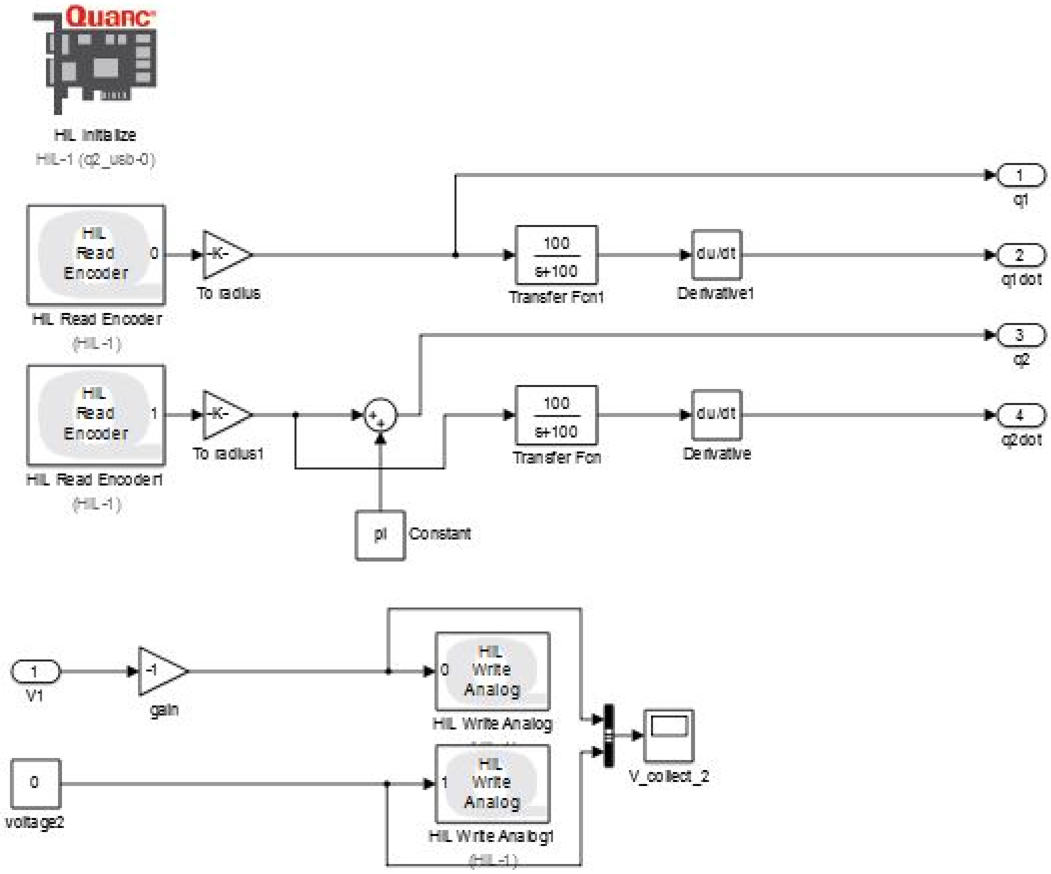
Balance Function:

```
function V = balance(q1, q1dot, q2, q2dot)
q2_new = atan2(sin(q2), cos(q2));
k =[-0.8285, -1.2024, -15.0566, -1.9463];
X = [q1;q1dot;q2_new;q2dot];
V = -k*X;
```

Final Design:



Subsystem Real:



Control Function:

```
function control  = control_func(q2, q2dot, interval_1, interval_2, oldControl)
% interval_1 = 25;
% interval_2 = 10;
q2 = q2 - pi;
q2 = atan2(sin(q2), cos(q2));

point_a = (180-interval_1)*pi/180;
point_b = (-180+interval_1)*pi/180;
point_c = (-interval_2)*pi/180;
point_d = (interval_2)*pi/180;


if(q2 > point_c && q2 < point_d)
    currentState = 1;
elseif((q2 > point_d && q2 < point_a)||(q2 > point_b && q2 < point_c))
    currentState = 2;
else
    currentState = 3;
end

if(oldControl == 1 && currentState == 3 )
    control = 3;
elseif oldControl == 2 && currentState == 1 && abs(q2dot) < 0.5
    control = 1;
elseif oldControl == 3 && currentState == 2
    control = 2;
else
    control = oldControl;
end
end
```

Transfer Function:

```
function output = fcn(input)
input = input - pi;
output = atan2(sin(input), cos(input));
```

When finishing the balance implementation, the robot can stay balance on the top itself. Being interrupted, it'll try to maintain balance through observing q2 and changing q1. When it is fall down, waiting for the swing velocity below 0.5

and starting another swing up.