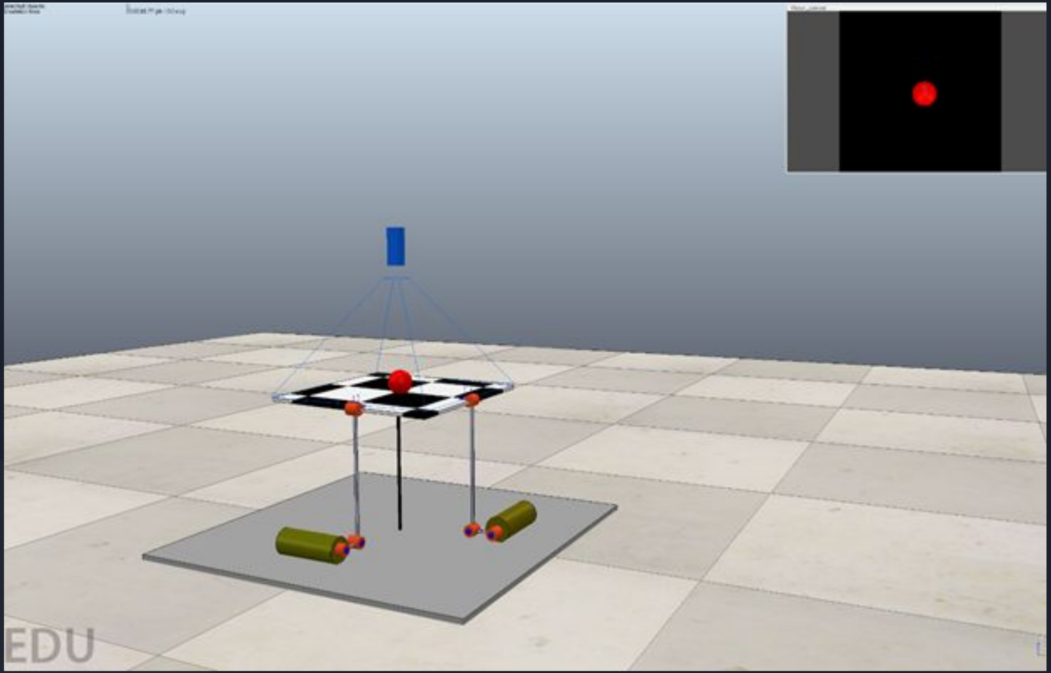
Ball and Plate



Team:
John Danley
Jonathan Gomez
Marco Machuca
Angel Mendoza
Chen Hung Yang

Background

The aim for this project is to model a ball and plate system with 2 degrees of freedom, and design a closed loop controller that can adjust the position of the ball using Matlab, Simulink and Coppelia.

Mathematical Modeling

- Modeled as 2 independent ball & Beam systems

$$m_b\,\ddot{x}(t) = m_b\,g\,\sin\alpha(t) - \frac{J_b\,\ddot{x}(t)}{r_b^2}. \qquad (1)$$

$$\ddot{x}(t) = \frac{2\,m_b\,g\,r_{arm}\,r_b^2}{L_{plate}\,(m_b\,r_b^2 + J_b)}\,\theta_l(t). \qquad (2)$$

$$P_s(s) = \frac{K}{s\,(\tau s + 1)}. \qquad (3)$$

$$P_{bb}(s) = \frac{X(s)}{\Theta_l(s)} = \frac{K_{bb}}{s^2} \qquad (4)$$

$$P(s) = \frac{X(s)}{V_m(s)} = \frac{K_{bb}\,K}{s^3\,(\tau\,s + 1)} \qquad (5)$$

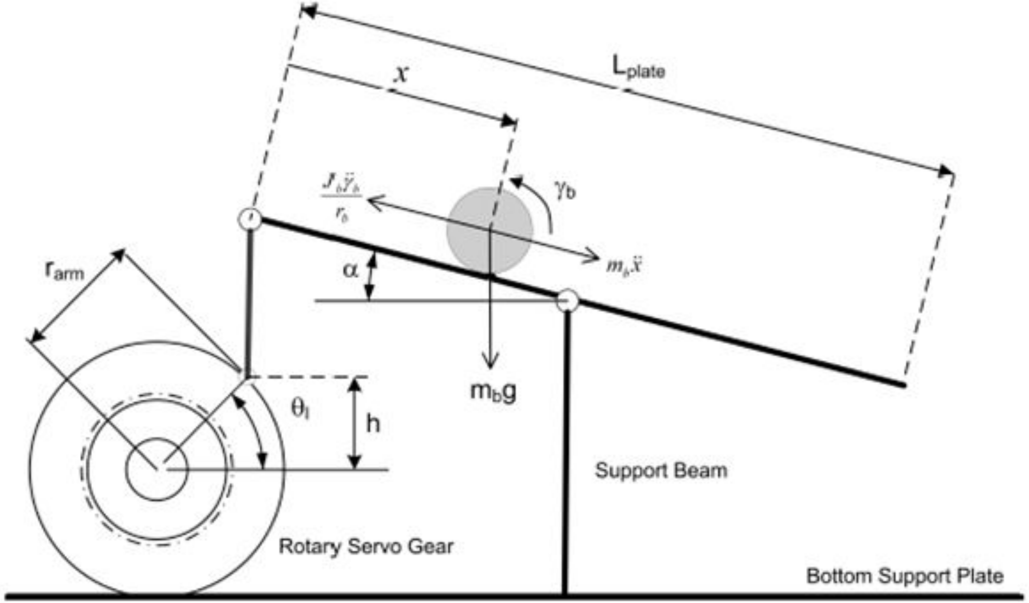
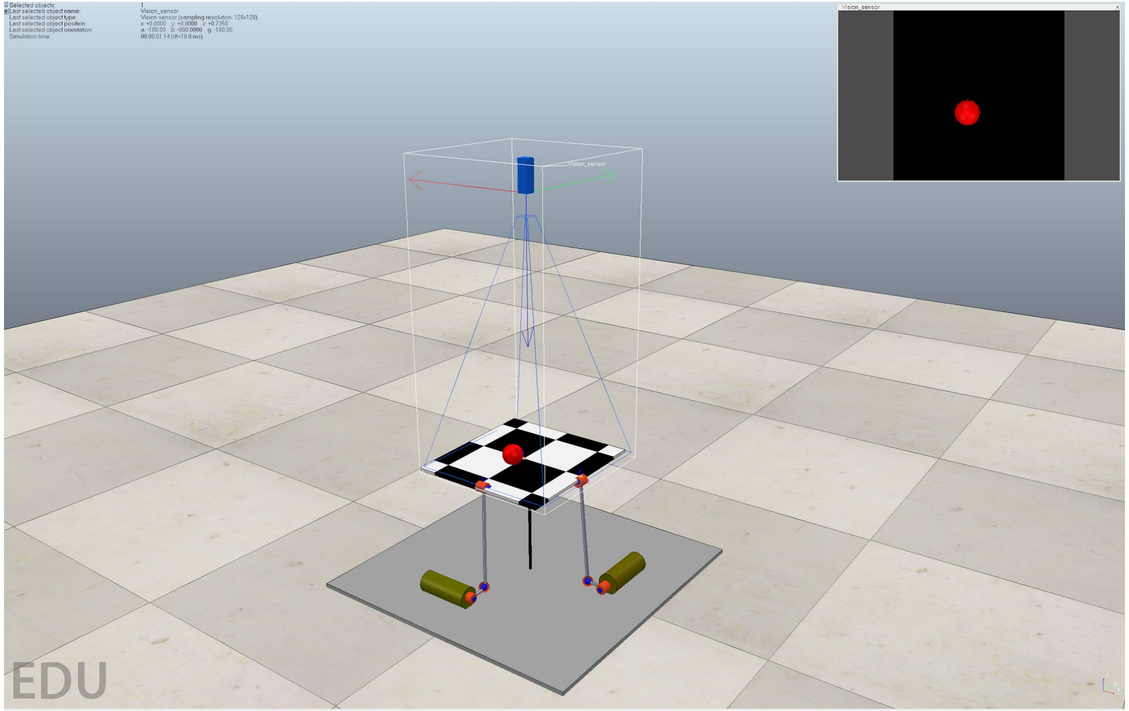
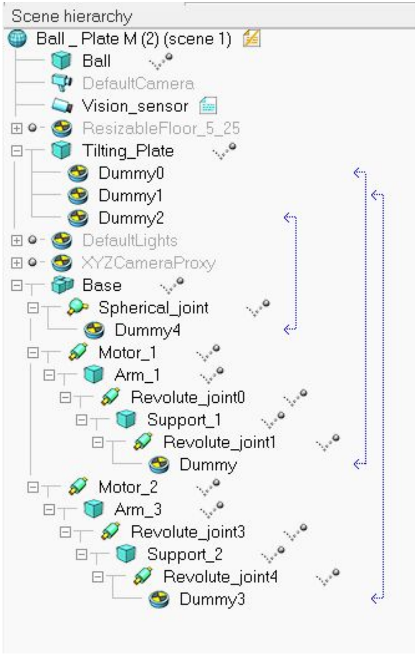


Figure 1: System Diagram. (Mesner & Tillbury,)

Simulation Model Designed in Coppelia

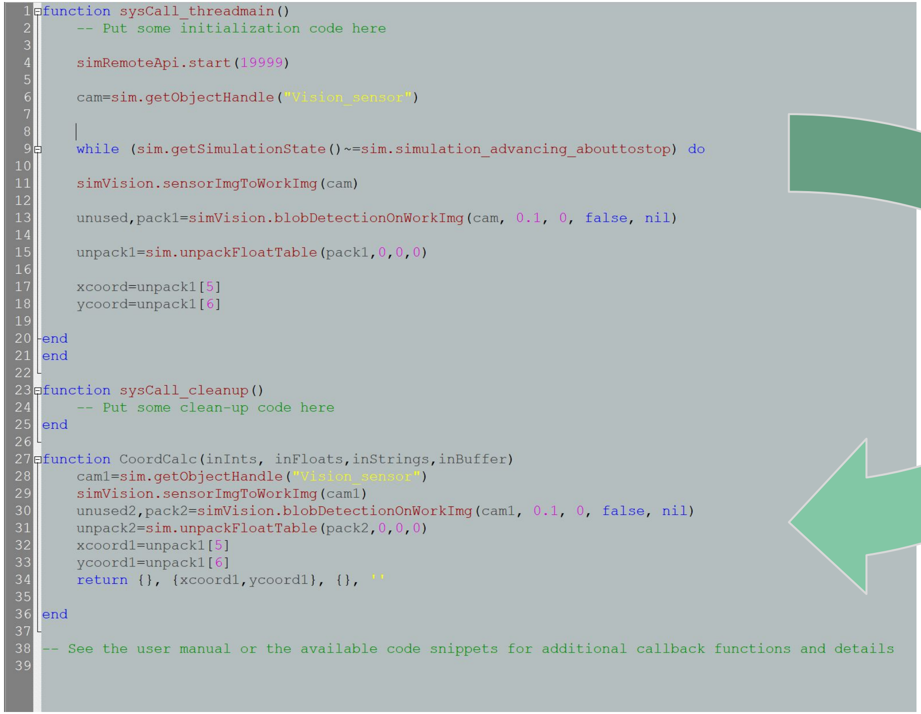


```
Selected objects:                          1
Last selected object name:                 Vision_sensor
Last selected object type:                 Vision sensor (sampling resolution: 128x128)
Last selected object position:             x: +0.0000   y: +0.0000   z: +0.7350
Last selected object orientation:          a: -180.00   b: -000.0000   g: -180.00
Simulation time:                           00:00:01.14 (dt=10.0 ms)
```



```
Scene hierarchy
Ball _ Plate M (2) (scene 1)
    Ball
    DefaultCamera
    Vision_sensor
⊞   ResizableFloor_5_25
⊟   Tilting_Plate
        Dummy0
        Dummy1
        Dummy2
⊞   DefaultLights
⊞   XYZCameraProxy
⊟   Base
    ⊟   Spherical_joint
            Dummy4
    ⊟   Motor_1
        ⊟   Arm_1
            ⊟   Revolute_joint0
                ⊟   Support_1
                    ⊟   Revolute_joint1
                            Dummy
    ⊟   Motor_2
        ⊟   Arm_3
            ⊟   Revolute_joint3
                ⊟   Support_2
                    ⊟   Revolute_joint4
                            Dummy3
```

Camera Sensor Code



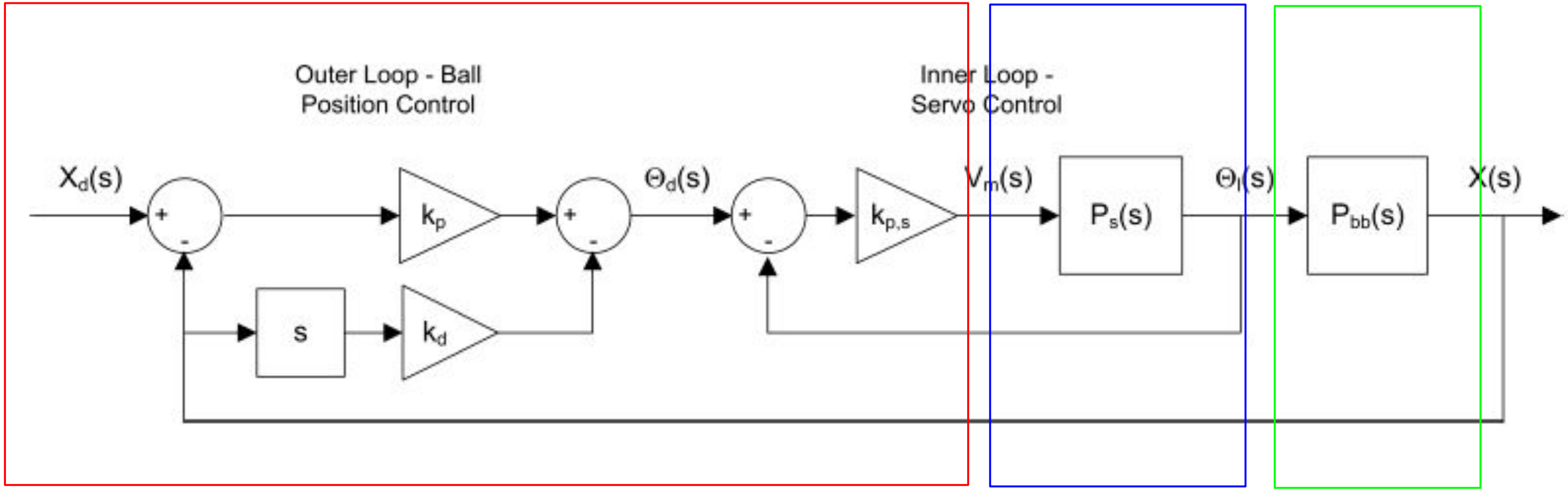
```
function sysCall_threadmain()
    -- Put some initialization code here

    simRemoteApi.start(19999)

    cam=sim.getObjectHandle("Vision_sensor")


    while (sim.getSimulationState()~=sim.simulation_advancing_abouttostop) do

    simVision.sensorImgToWorkImg(cam)

    unused,pack1=simVision.blobDetectionOnWorkImg(cam, 0.1, 0, false, nil)

    unpack1=sim.unpackFloatTable(pack1,0,0,0)

    xcoord=unpack1[5]
    ycoord=unpack1[6]

end
end

function sysCall_cleanup()
    -- Put some clean-up code here
end

function CoordCalc(inInts, inFloats,inStrings,inBuffer)
    cam1=sim.getObjectHandle("Vision_sensor")
    simVision.sensorImgToWorkImg(cam1)
    unused2,pack2=simVision.blobDetectionOnWorkImg(cam1, 0.1, 0, false, nil)
    unpack2=sim.unpackFloatTable(pack2,0,0,0)
    xcoord1=unpack1[5]
    ycoord1=unpack1[6]
    return {}, {xcoord1,ycoord1}, {}, ''

end

-- See the user manual or the available code snippets for additional callback functions and details
```

Keep track of ball position in physical environment

File Order

1. Simulink File Name
   a. V1.slx
2. Coppelia File Name
   a. Ball _ Plate M (2)
3. Matlab File Name
   a. setup_2dbb.m
4. Matlab File Name
   a. matlabAPI.m

SIMULINK topology

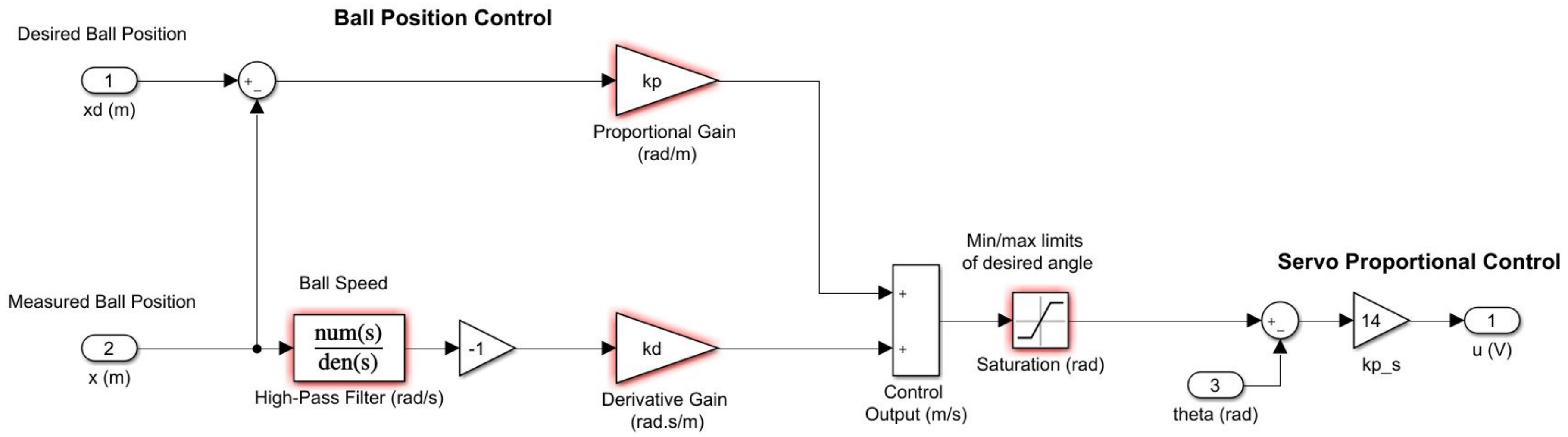


Controller

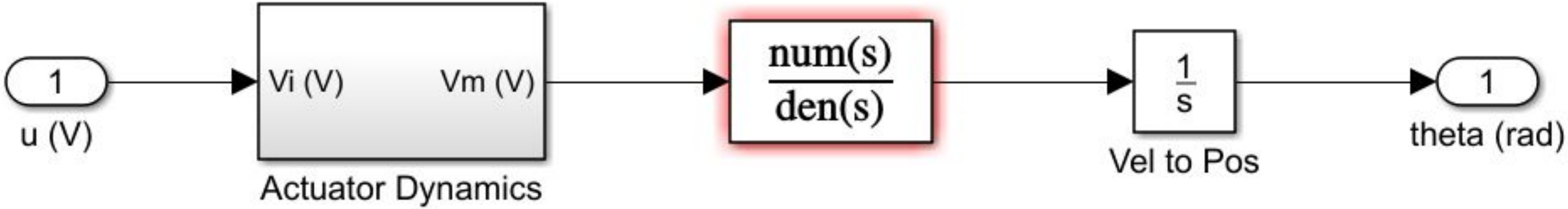
Motor Plant transfer function

Ball and Plate plant transfer function

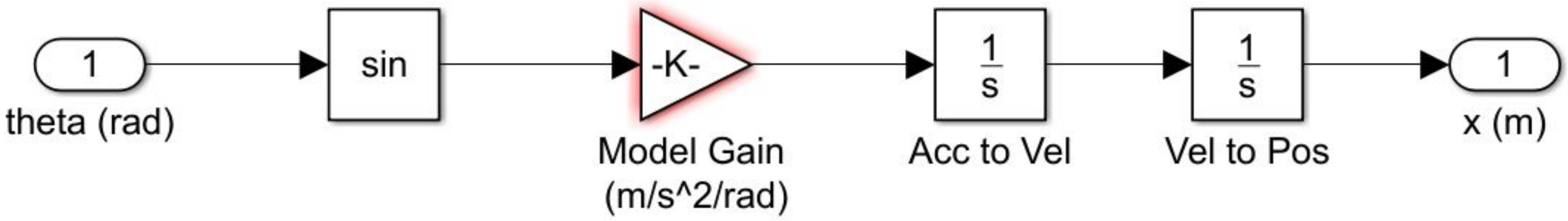
SIMULINK BLOCK DIAGRAM



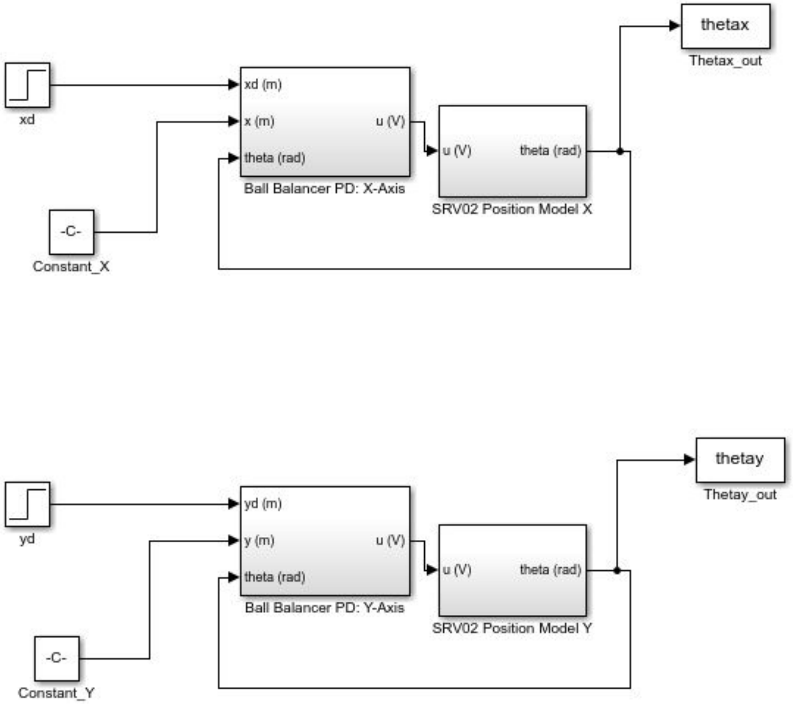
SIMULINK BLOCK DIAGRAM



SIMULINK BLOCK DIAGRAM

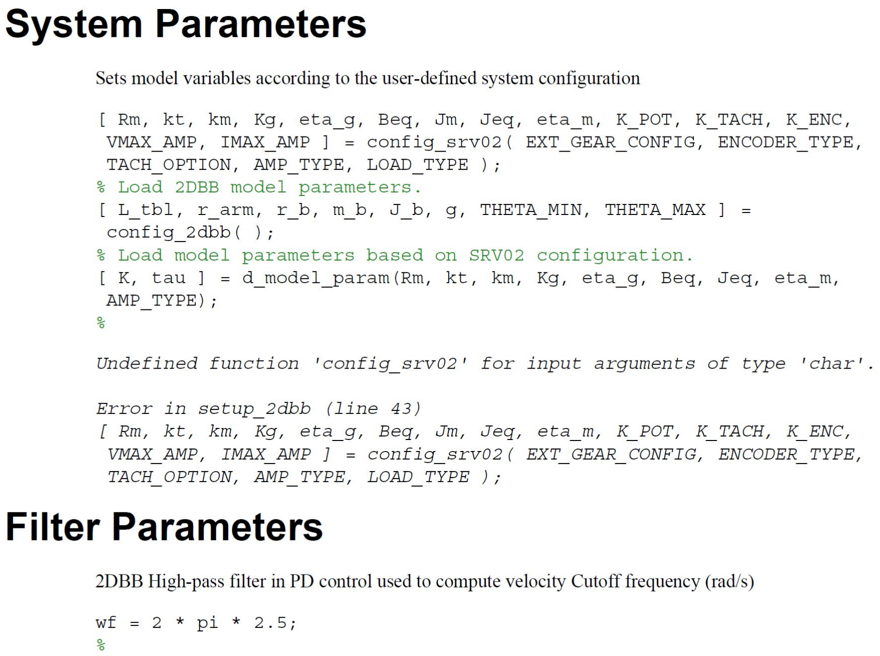


Final SIMULINK BLOCK DIAGRAM



Initialize the work space

1. Api code used to communicate with SIMULINK



System Parameters

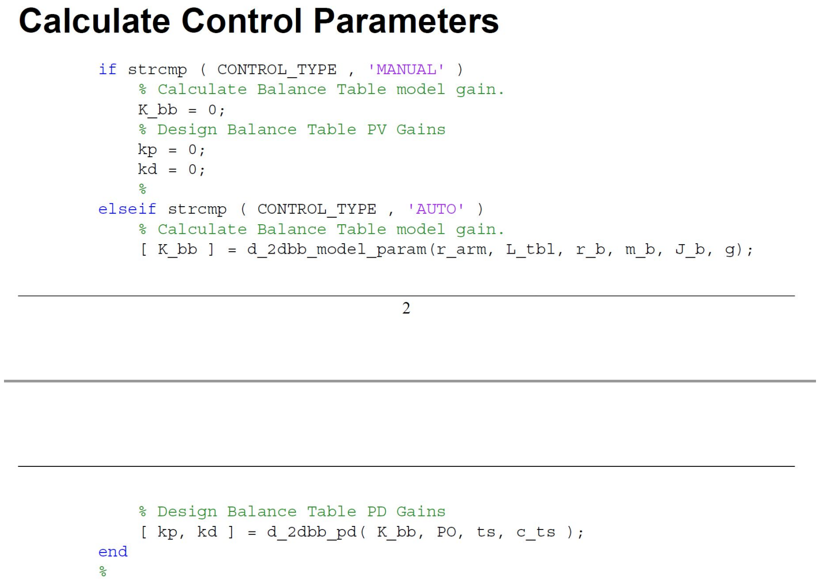
Sets model variables according to the user-defined system configuration

```
[ Rm, kt, km, Kg, eta_g, Beq, Jm, Jeq, eta_m, K_POT, K_TACH, K_ENC,
 VMAX_AMP, IMAX_AMP ] = config_srv02( EXT_GEAR_CONFIG, ENCODER_TYPE,
 TACH_OPTION, AMP_TYPE, LOAD_TYPE );
% Load 2DBB model parameters.
[ L_tbl, r_arm, r_b, m_b, J_b, g, THETA_MIN, THETA_MAX ] =
 config_2dbb( );
% Load model parameters based on SRV02 configuration.
[ K, tau ] = d_model_param(Rm, kt, km, Kg, eta_g, Beq, Jeq, eta_m,
 AMP_TYPE);
%

Undefined function 'config_srv02' for input arguments of type 'char'.

Error in setup_2dbb (line 43)
[ Rm, kt, km, Kg, eta_g, Beq, Jm, Jeq, eta_m, K_POT, K_TACH, K_ENC,
 VMAX_AMP, IMAX_AMP ] = config_srv02( EXT_GEAR_CONFIG, ENCODER_TYPE,
 TACH_OPTION, AMP_TYPE, LOAD_TYPE );
```

Filter Parameters

2DBB High-pass filter in PD control used to compute velocity Cutoff frequency (rad/s)

```
wf = 2 * pi * 2.5;
%
```



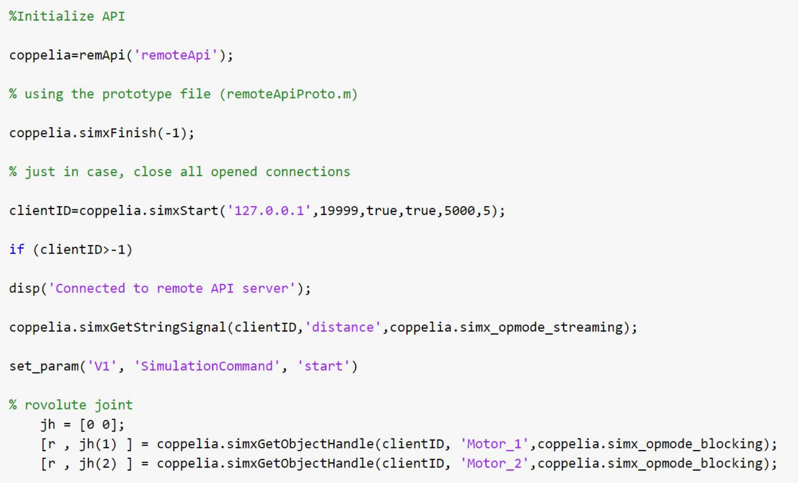
Calculate Control Parameters

```
if strcmp ( CONTROL_TYPE , 'MANUAL' )
    % Calculate Balance Table model gain.
    K_bb = 0;
    % Design Balance Table PV Gains
    kp = 0;
    kd = 0;
    %
elseif strcmp ( CONTROL_TYPE , 'AUTO' )
    % Calculate Balance Table model gain.
    [ K_bb ] = d_2dbb_model_param(r_arm, L_tbl, r_b, m_b, J_b, g);
```

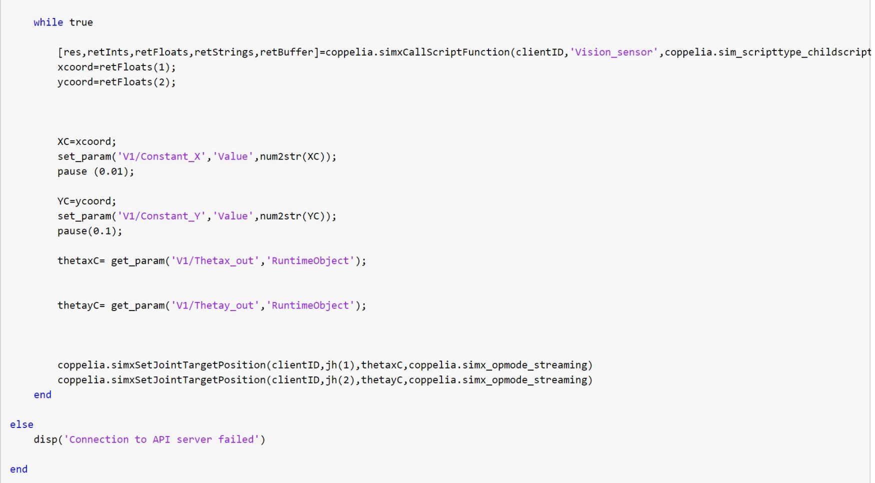
2

```
    % Design Balance Table PD Gains
    [ kp, kd ] = d_2dbb_pd( K_bb, PO, ts, c_ts );
end
%
```

MATLAB API



```
%Initialize API

coppelia=remApi('remoteApi');

% using the prototype file (remoteApiProto.m)

coppelia.simxFinish(-1);

% just in case, close all opened connections

clientID=coppelia.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)

disp('Connected to remote API server');

coppelia.simxGetStringSignal(clientID,'distance',coppelia.simx_opmode_streaming);

set_param('V1', 'SimulationCommand', 'start')

% rovolute joint
    jh = [0 0];
    [r , jh(1) ] = coppelia.simxGetObjectHandle(clientID, 'Motor_1',coppelia.simx_opmode_blocking);
    [r , jh(2) ] = coppelia.simxGetObjectHandle(clientID, 'Motor_2',coppelia.simx_opmode_blocking);
```



```
    while true

        [res,retInts,retFloats,retStrings,retBuffer]=coppelia.simxCallScriptFunction(clientID,'Vision_sensor',coppelia.sim_scripttype_childscript
        xcoord=retFloats(1);
        ycoord=retFloats(2);



        XC=xcoord;
        set_param('V1/Constant_X','Value',num2str(XC));
        pause (0.01);

        YC=ycoord;
        set_param('V1/Constant_Y','Value',num2str(YC));
        pause(0.1);

        thetaxC= get_param('V1/Thetax_out','RuntimeObject');


        thetayC= get_param('V1/Thetay_out','RuntimeObject');



        coppelia.simxSetJointTargetPosition(clientID,jh(1),thetaxC,coppelia.simx_opmode_streaming)
        coppelia.simxSetJointTargetPosition(clientID,jh(2),thetayC,coppelia.simx_opmode_streaming)
    end

else
    disp('Connection to API server failed')

end
```