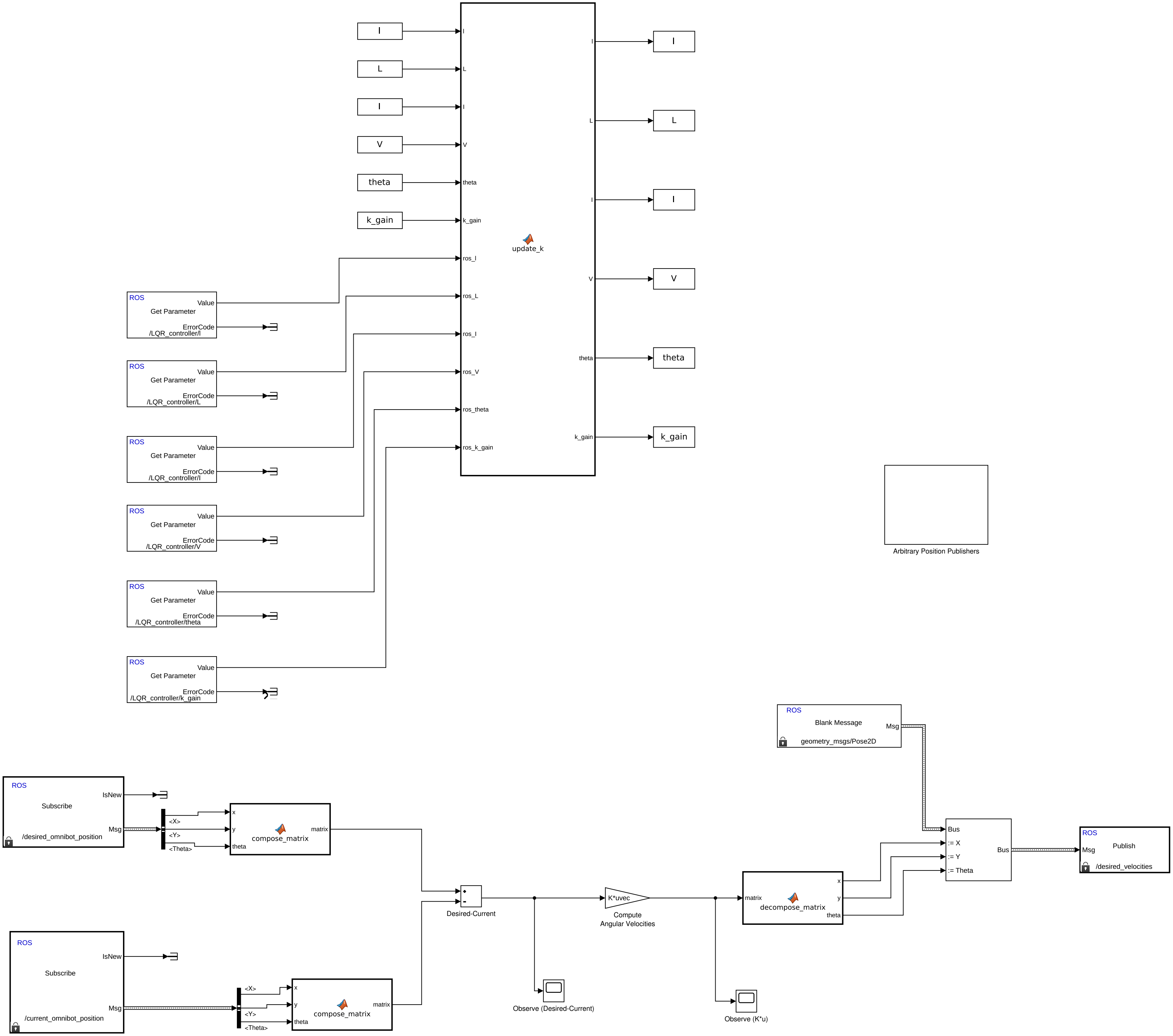
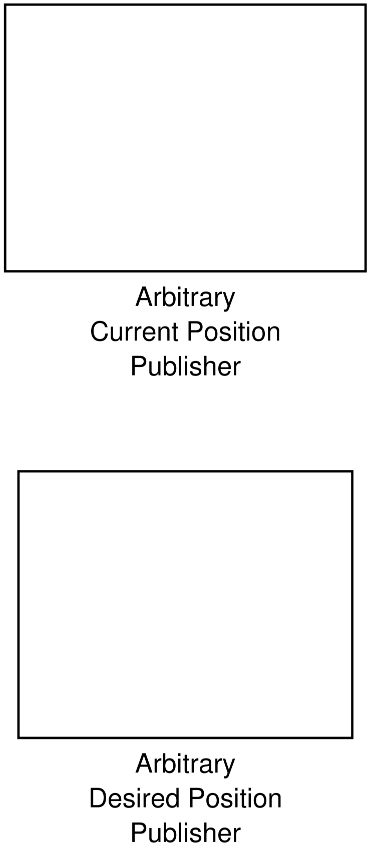
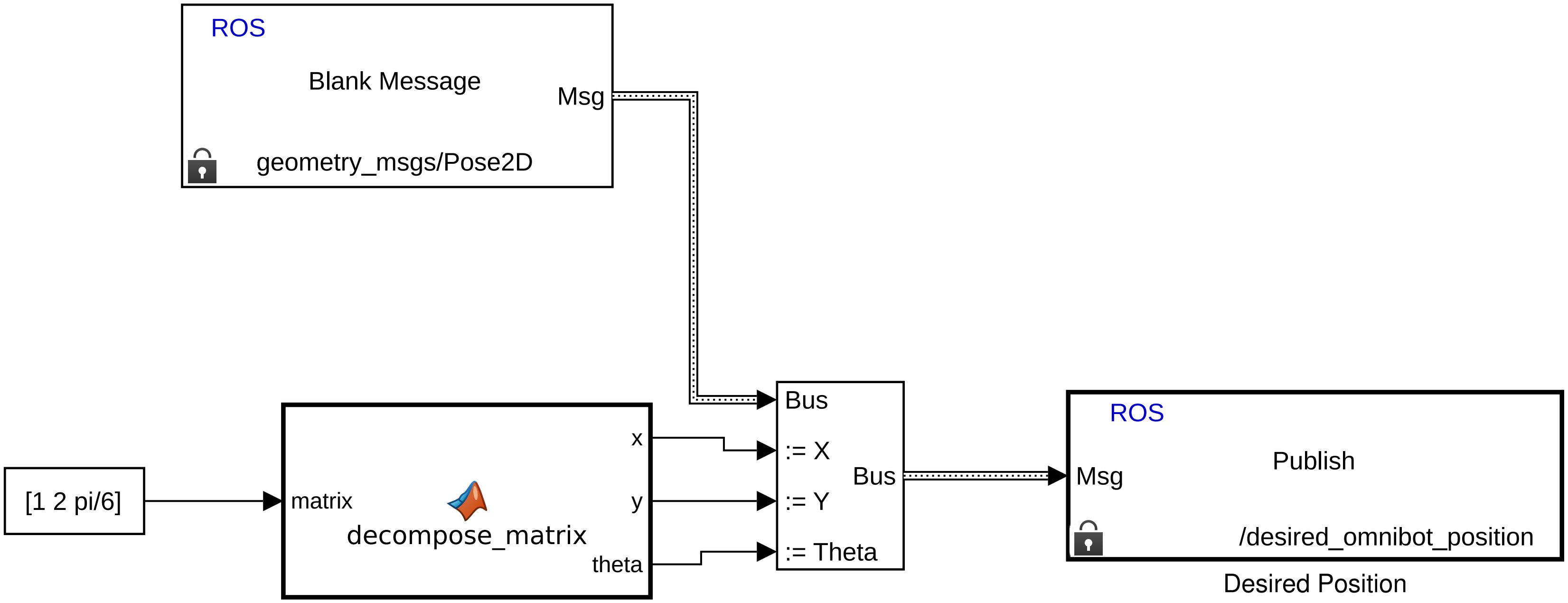

l
L
I
V
theta
k_gain
l
L
I
V
theta
k_gain
ros_l
ros_L
ros_I
ros_V
ros_theta
ros_k_gain
update_k
l
L
I
V
theta
k_gain
l
L
I
V
theta
k_gain
ROS
Value
Get Parameter
ErrorCode
/LQR_controller/l
ROS
Value
Get Parameter
ErrorCode
/LQR_controller/L
ROS
Value
Get Parameter
ErrorCode
/LQR_controller/I
ROS
Value
Get Parameter
ErrorCode
/LQR_controller/V
ROS
Value
Get Parameter
ErrorCode
/LQR_controller/theta
ROS
Value
Get Parameter
ErrorCode
/LQR_controller/k_gain
Arbitrary Position Publishers
ROS
Blank Message
Msg
geometry_msgs/Pose2D
ROS
Subscribe
IsNew
Msg
/desired_omnibot_position
<X>
<Y>
<Theta>
x
y
theta
compose_matrix
matrix
ROS
Subscribe
IsNew
Msg
/current_omnibot_position
<X>
<Y>
<Theta>
x
y
theta
compose_matrix
matrix
+
-
Desired-Current
K*uvec
Compute
Angular Velocities
Observe (Desired-Current)
Observe (K*u)
matrix
decompose_matrix
x
y
theta
Bus
:= X
:= Y
:= Theta
Bus
ROS
Publish
Msg
/desired_velocities

```
function matrix = compose_matrix(x,y,theta)
    matrix = [x y theta];
end
```

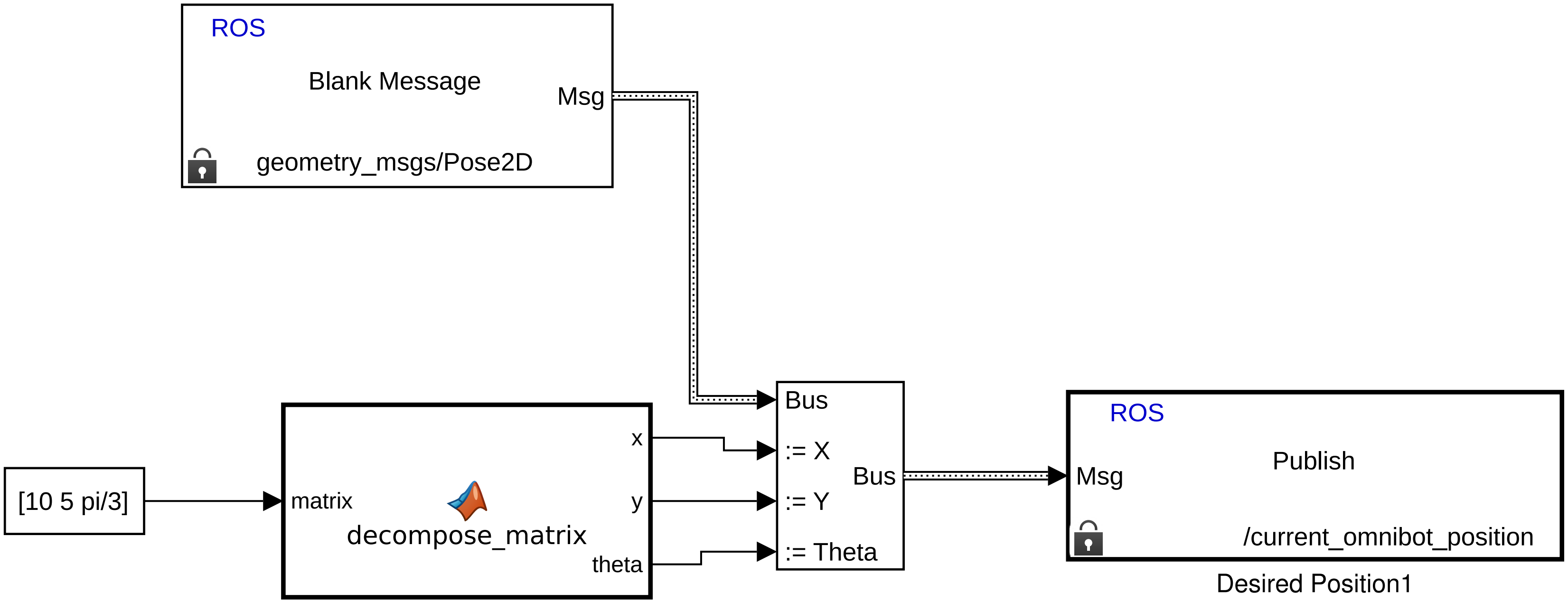
```
function [x, y, theta] = decompose_matrix(matrix)
    x = matrix(1);
    y = matrix(2);
    theta = matrix(3);
end
```

```
function matrix = compose_matrix(x,y,theta)
    matrix = [x y theta];
end
```


Arbitrary
Current Position
Publisher
Arbitrary
Desired Position
Publisher


ROS
Blank Message
Msg
geometry_msgs/Pose2D
[1 2 pi/6]
matrix
decompose_matrix
x
y
theta
Bus
:= X
:= Y
:= Theta
Bus
ROS
Publish
Msg
/desired_omnibot_position
Desired Position

```
function [x, y, theta] = decompose_matrix(matrix)
    x = matrix(1);
    y = matrix(2);
    theta = matrix(3);
end
```


ROS
Blank Message
Msg
geometry_msgs/Pose2D
[10 5 pi/3]
matrix
decompose_matrix
x
y
theta
Bus
:= X
:= Y
:= Theta
Bus
ROS
Publish
Msg
/current_omnibot_position
Desired Position1

```
function [x, y, theta] = decompose_matrix(matrix)
    x = matrix(1);
    y = matrix(2);
    theta = matrix(3);
end
```

```
function [l, L, I, V, theta, k_gain] = update_k(l, L, I, V, theta, k_gain, ros_l, ros_L, ros_I, ros_V, ros_theta, ros_k_gain)

    ros_k_gain = str2num(ros_k_gain);
    if ros_k_gain ~= k_gain
        k_gain = ros_k_gain;
        return % terminate, since updates to 'k_gain' override any other update
    end

    old_LQR_params = [
        l, L, I, V, theta
    ];

    ros_LQR_params = [
        ros_l, ros_L, ros_I, ros_V, ros_theta
    ];

    if ros_LQR_params~=old_LQR_params
        l = ros_l;
        L = ros_L;
        I = ros_I;
        V = ros_V;
        theta = ros_theta;
        k_gain = compute_K(l, L, I, V, theta);

        % update rosparam
        new_ros_k_gain = mat2str(k_gain);
        rosparam('set', '/LQR_controller/k_gain', new_ros_k_gain)
    end
end
```