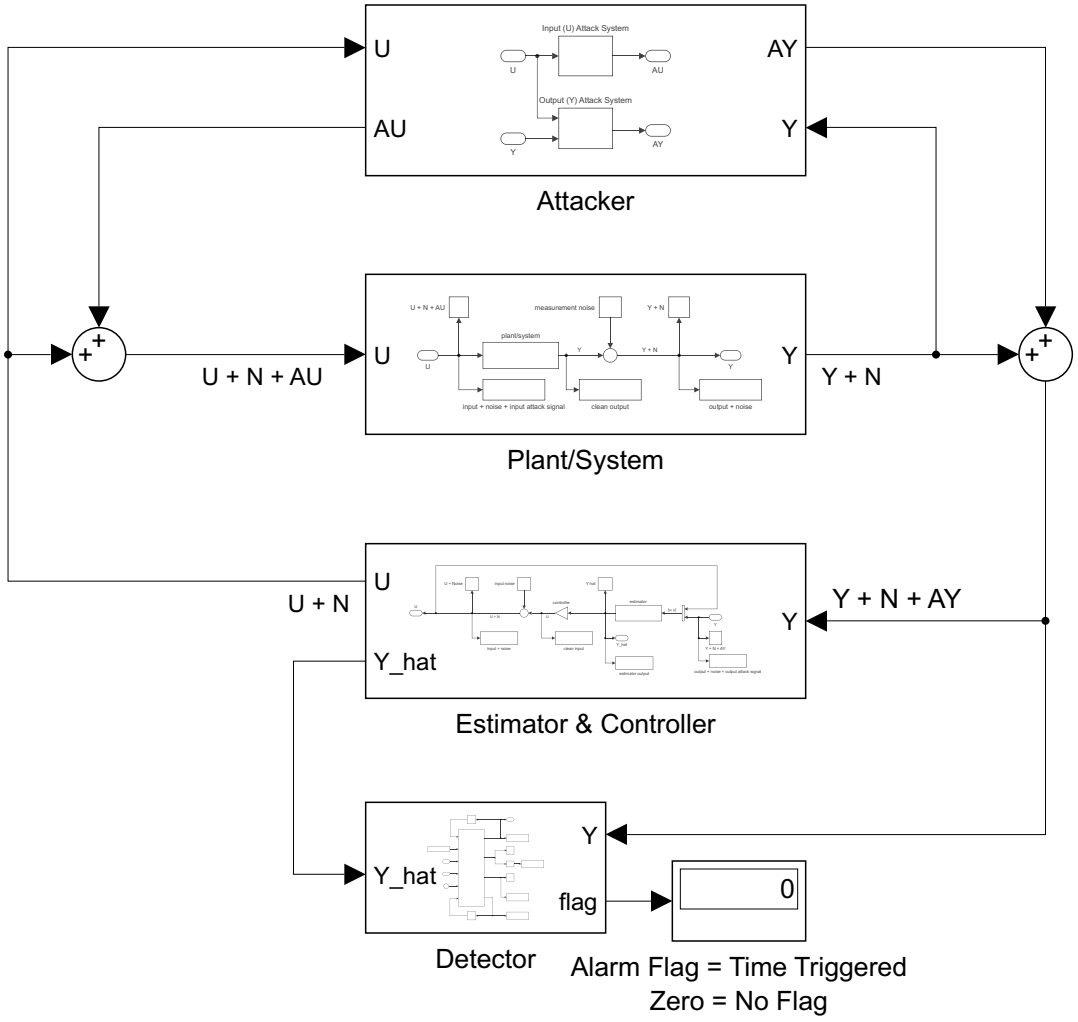
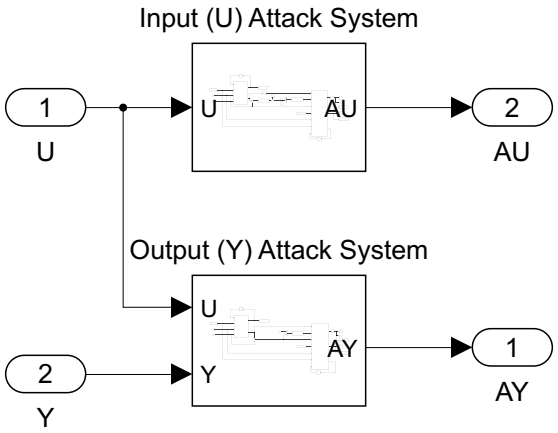
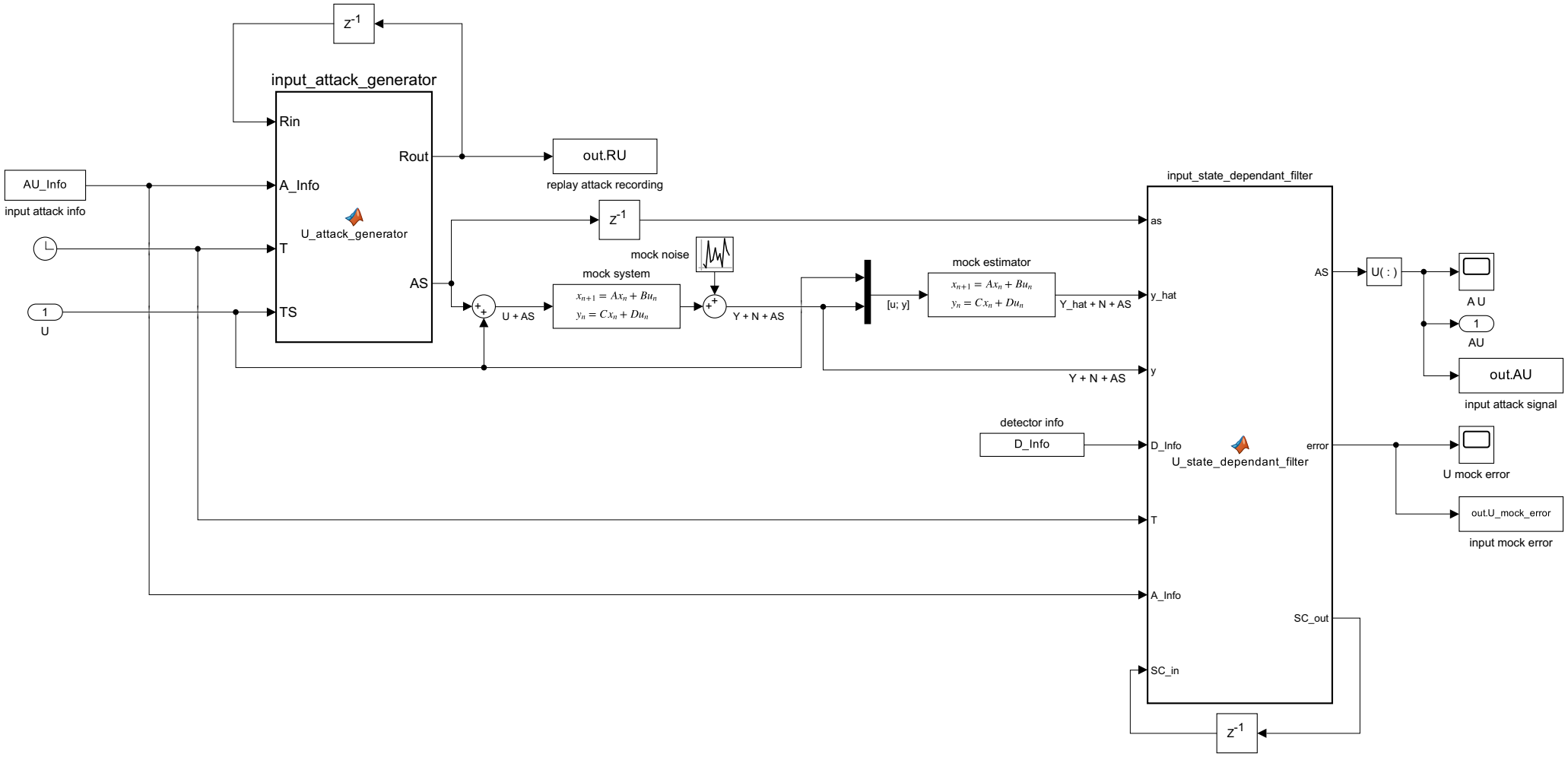

U
AU
AY
Y
Input (U) Attack System
Output (Y) Attack System
Attacker
U + N + AU
Y + N
plant/system
measurement noise
input + noise + input attack signal
clean output
output + noise
Plant/System
U + N
Y_hat
Y + N + AY
Estimator & Controller
flag
0
Detector
Alarm Flag = Time Triggered
Zero = No Flag

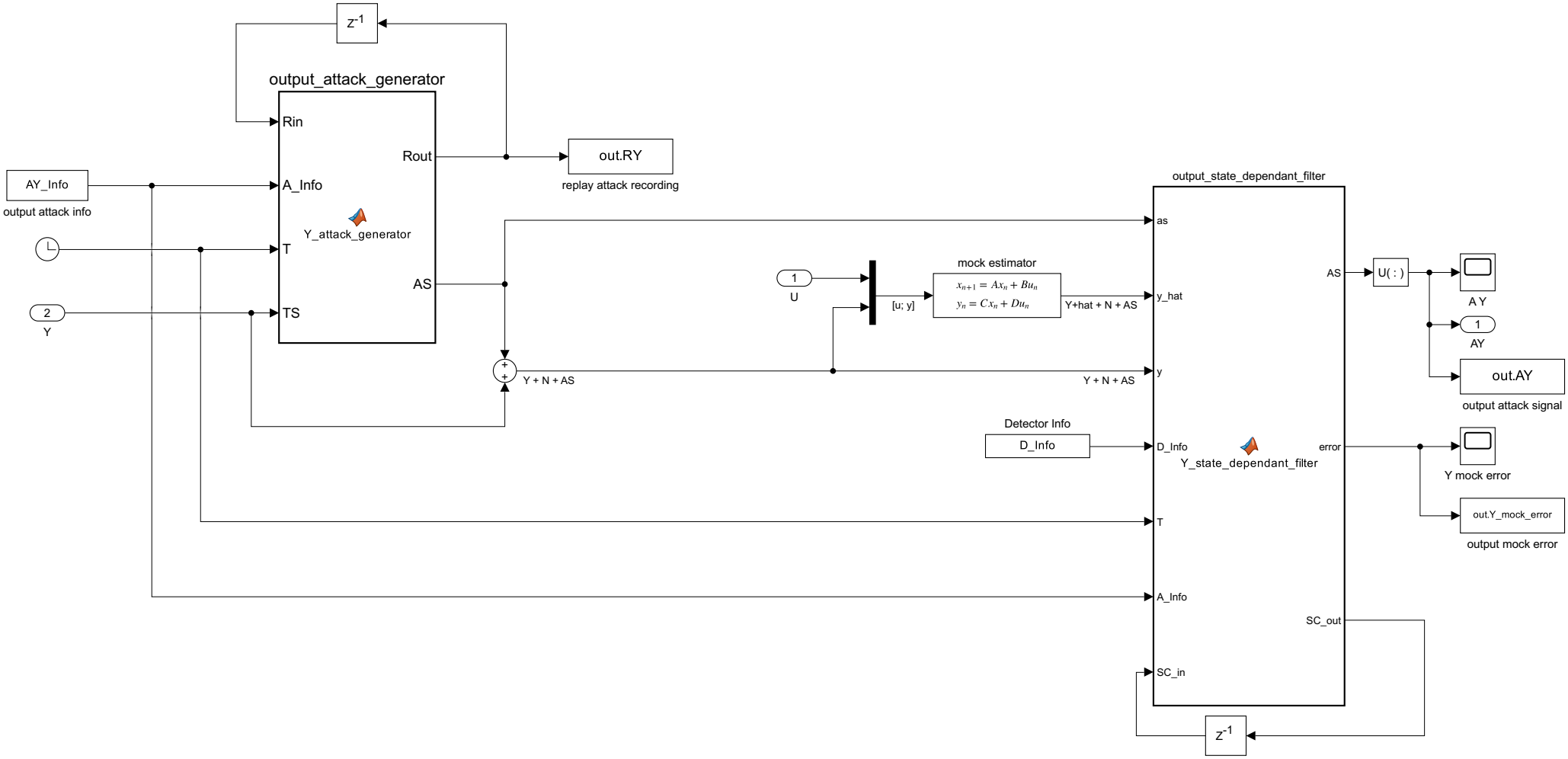

Input (U) Attack System
1
U
U
AU
2
AU
Output (Y) Attack System
U
2
Y
Y
AY
1
AY


z-1
input_attack_generator
Rin
Rout
A_Info
U_attack_generator
T
AS
TS
out.RU
replay attack recording
AU_Info
input attack info
1
U
z-1
mock noise
mock system
$x_{n+1} = Ax_n + Bu_n$
$y_n = Cx_n + Du_n$
U + AS
Y + N + AS
[u; y]
mock estimator
$x_{n+1} = Ax_n + Bu_n$
$y_n = Cx_n + Du_n$
Y_hat + N + AS
Y + N + AS
input_state_dependant_filter
as
y_hat
y
T
A_Info
SC_out
SC_in
detector info
D_Info
D_Info
U_state_dependant_filter
error
z-1
AS
U(:)
A U
1
AU
out.AU
input attack signal
U mock error
out.U_mock_error
input mock error

```
% Matthew Reaney, QUB
% April 2023
% https://github.com/mattr862/three-tank-water-system

function [Rout, AS] = U_attack_generator(Rin, A_Info, T, TS)
%outputs = Recording Outout, Attack Signal
%inputs = Recording input, attack info, time, target signal
AS = zeros(4,1); % intialize attack signal

%% Replay attack buffer
if T == 0
    Rin = zeros(100, 4);
    Rout = Rin;
    Rout(1,:) = transpose(TS);
else
    Rout = Rin;
    Rout(1,:) = transpose(TS);
    for i = 1:99
         Rout(i+1,:) = Rin(i,:);
         i = i + 1;
    end
end

%% Attack Modeling
if T >= A_Info(5) && T <= A_Info(6) % attack time period
    i = 1;
    for i = 1:4
        switch A_Info(i)
            case 1 % FDI
                AS(i) = A_Info(8)*rand(1);
            case 2 % Bias
                AS(i) = A_Info(9);
            case 3 % Dos
                AS(i) = -TS(i);
            case 4 % Sign_alt
                AS(i) = -2*TS(i);
            case 5 % Rerouting
                TSR = [TS(3); TS(4); TS(1); TS(2)]; % swaps values
                AS(i) = -TS(i) + TSR(i); % Combine with blank signal
            case 6 % Replay
                ti = uint16((A_Info(6)-A_Info(5))*100); % convert time to index value
                AS(i) = -TS(i) + Rin(ti,i);
            otherwise % None
                AS(i) = 0;
        end
        i = i + 1;
    end
end

%% Console output
if T == 0 && (A_Info(1) ~= 0 || A_Info(2) ~= 0 || A_Info(3) ~= 0 || A_Info(4) ~= 0)
    if A_Info(7) == 1
        fprintf('Attempting Stealthy Attacks on Input from %f to %.2f\n', A_Info(5), A_Info(6));
    else
        fprintf('Attacking on Input from %f to %.2f\n', A_Info(5), A_Info(6));
    end
    i = 1;
    for i = 1:4
        switch A_Info(i)
```

```
            case 1 % FDI
                fprintf('Input %i: FDI Attack\n', int8(i));
            case 2 % Bias
                fprintf('Input %i: Bias Attack\n', int8(i));
            case 3 % Dos
                fprintf('Input %i: Dos Attack\n', int8(i));
            case 4 % Sign_alt
                fprintf('Input %i: Sign_alt Attack\n', int8(i));
            case 5 % Rerouting
                fprintf('Input %i: Rerouting Attack\n', int8(i));
            case 6 % Replay
                fprintf('Input %i: Replay Attack\n', int8(i));
                if A_Info(5)-(A_Info(6)-A_Info(5)) < 0
                    fprintf('Warning: Not enough time to record for replay attack duration\n');
                end
            otherwise
        end
        i = i + 1;
    end
end
```

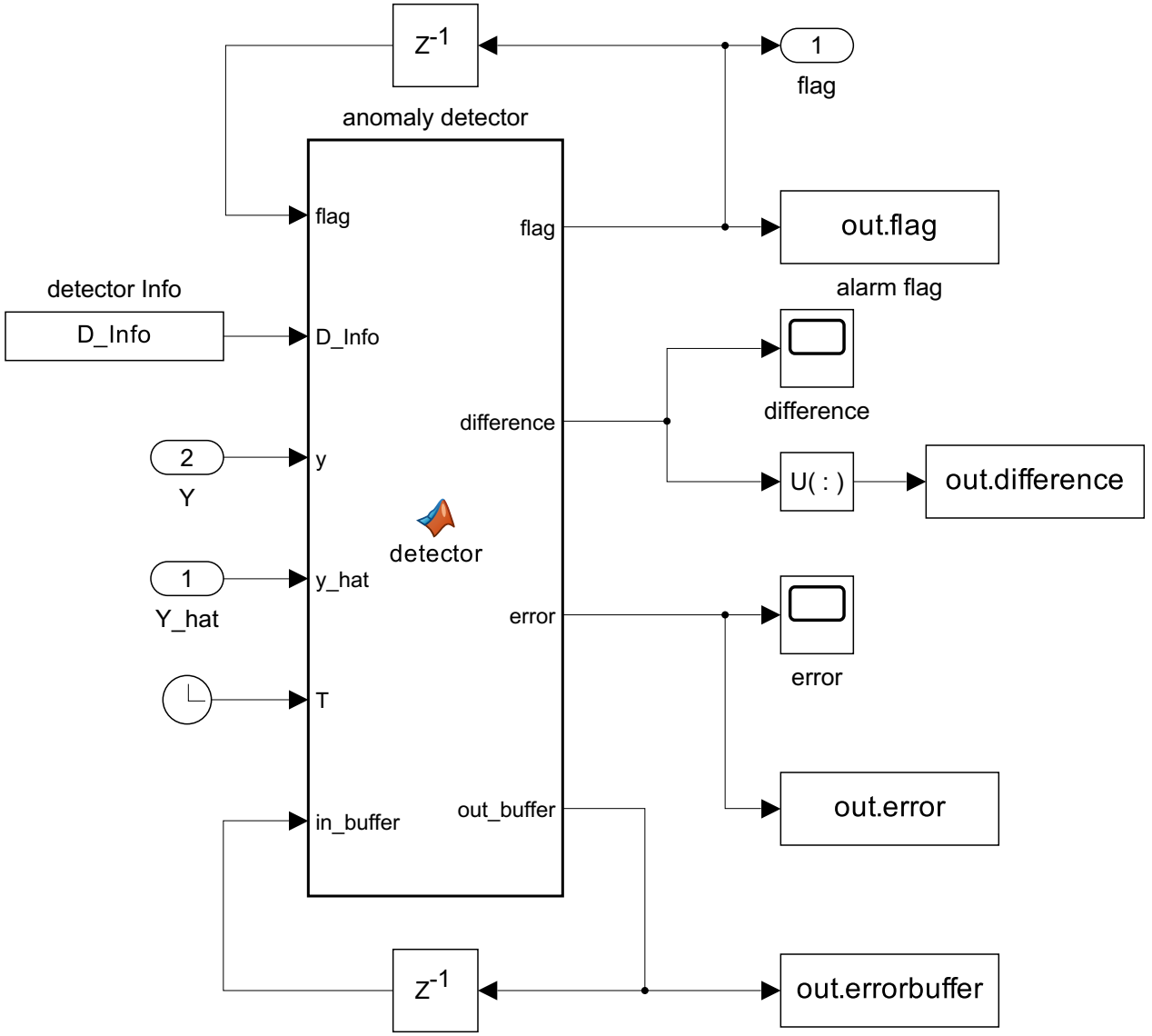
```
% Matthew Reaney, QUB
% April 2023
% https://github.com/mattr862/three-tank-water-system

function [AS, error, SC_out] = U_state_dependant_filter(as, y_hat, y, D_Info, T, A_Info,  SC_in)
%outputs = Attack Signal, error, Succcessful stealthy attack Count Out
%inputs = Attack Signal, Y_hat, Y, detector info, time, attack info,  Succcessful stealthy attack Count In
AS = zeros(4,1); % intialize attack signal

if A_Info(7) ~= 0
    %% Stealthy Attack Enabled
    % Succcessful stealthy attack Count (SC)
    if T == 0
        SC_in = 0;
    end
    SC_out = SC_in;

    % Stealty success console output
    if T == A_Info(6)
        fprintf('\nInput: Successfully preformed %i stealthy attacks out of a possible %.2i\n', int16(SC_out), int16((A_Info(6) - A
    end

    % Mock Detector Responce
    difference = (y - y_hat);
    error = norm(difference, Inf);

    %If signal is stealthy use it
    if error < D_Info(1) && T >= A_Info(5) && T <= A_Info(6)
        AS = as;
        SC_out = SC_in + 1;
    end

else
    %% Stealthy Attack disabled
    AS = as;
    error = 0;
    SC_out = 0;
end
```


z-1
output_attack_generator
Rin
Rout
out.RY
replay attack recording
AY_Info
output attack info
A_Info
Y_attack_generator
T
AS
2
Y
TS
Y + N + AS
1
U
[u; y]
mock estimator
$x_{n+1} = Ax_n + Bu_n$
$y_n = Cx_n + Du_n$
Y+hat + N + AS
output_state_dependant_filter
as
y_hat
y
Y + N + AS
Detector Info
D_Info
D_Info
Y_state_dependant_filter
error
T
A_Info
SC_out
SC_in
z-1
AS
U(:)
A Y
1
AY
out.AY
output attack signal
Y mock error
out.Y_mock_error
output mock error

```
% Matthew Reaney, QUB
% April 2023
% https://github.com/mattr862/three-tank-water-system

function [Rout, AS] = Y_attack_generator(Rin, A_Info, T, TS)
%outputs = Recording Outout, Attack Signal
%inputs = Recording input, attack info, time, target signal
AS = zeros(3,1); % intialize attack signal

%% Replay attack buffer
if T == 0
    Rin = zeros(100, 3);
    Rout = Rin;
    Rout(1,:) = transpose(TS);
else
    Rout = Rin;
    Rout(1,:) = transpose(TS);
    for i = 1:99
         Rout(i+1,:) = Rin(i,:);
         i = i + 1;
    end
end

%% Attack Modeling
if T >= A_Info(5) && T <= A_Info(6) % attack time period
    i = 1;
    for i = 1:3
        switch A_Info(i)
            case 1 % FDI
                AS(i) = A_Info(8)*rand(1);
            case 2 % Bias
                AS(i) = A_Info(9);
            case 3 % Dos
                AS(i) = -TS(i);
            case 4 % Sign_alt
                AS(i) = -2*TS(i);
            case 5 % Rerouting
                TSR = [TS(2); TS(1); TS(3)]; % swaps values
                AS(i) = -TS(i) + TSR(i); % Combine with blank signal
            case 6 % Replay
                ti = uint16((A_Info(6)-A_Info(5))*100); % convert time to index value
                AS(i) = -TS(i) + Rin(ti,i);
            otherwise % None
                AS(i) = 0;
        end
        i = i + 1;
    end
end

%% Console output
if T == 0 && (A_Info(1) ~= 0 || A_Info(2) ~= 0 || A_Info(3) ~= 0)
    if A_Info(7) == 1
        fprintf('Attempting Stealthy Attacks on Output from %f to %.2f\n', A_Info(5), A_Info(6));
    else
        fprintf('Attacking Output from %f to %.2f\n', A_Info(5), A_Info(6));
    end
    i = 1;
    for i = 1:3
        switch A_Info(i)
```

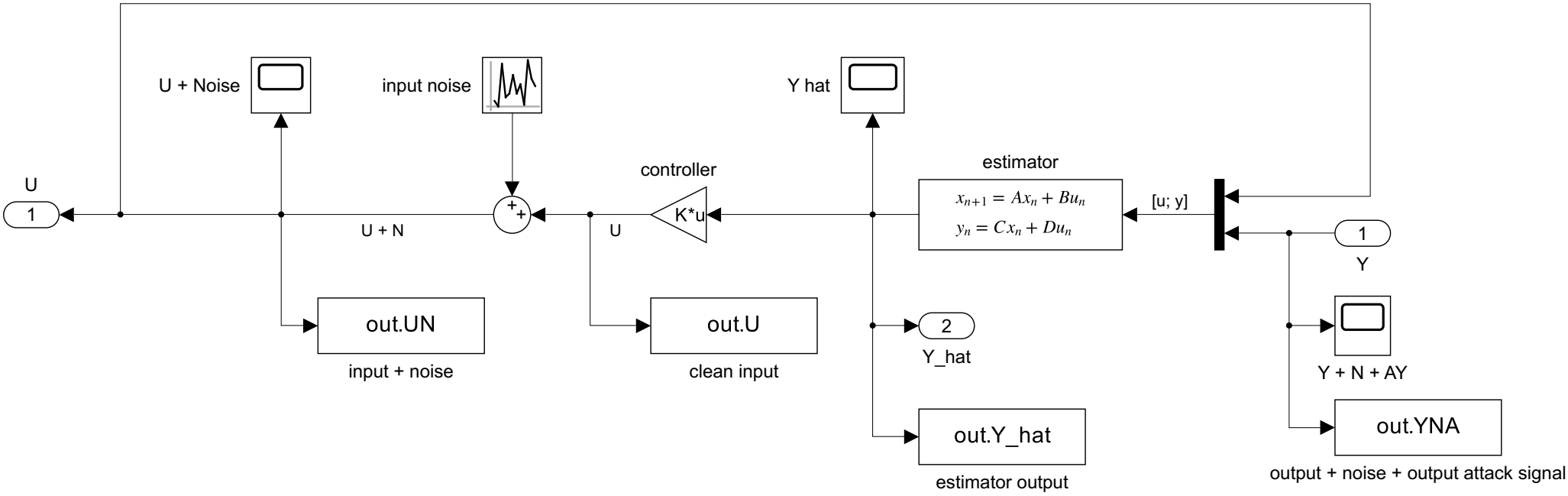
```
            case 1 % FDI
                fprintf('Output %i: FDI Attack\n', int8(i));
            case 2 % Bias
                fprintf('Output %i: Bias Attack\n', int8(i));
            case 3 % Dos
                fprintf('Output %i: Dos Attack\n', int8(i));
            case 4 % Sign_alt
                fprintf('Output %i: Sign_alt Attack\n', int8(i));
            case 5 % Rerouting
                fprintf('Output %i: Rerouting Attack\n', int8(i));
            case 6 % Replay
                fprintf('Output %i: Replay Attack\n', int8(i));
                if A_Info(5)-(A_Info(6)-A_Info(5)) < 0
                    fprintf('Warning: Not enough time to record for replay attack duration\n');
                end
            otherwise
        end
        i = i + 1;
    end
end
```

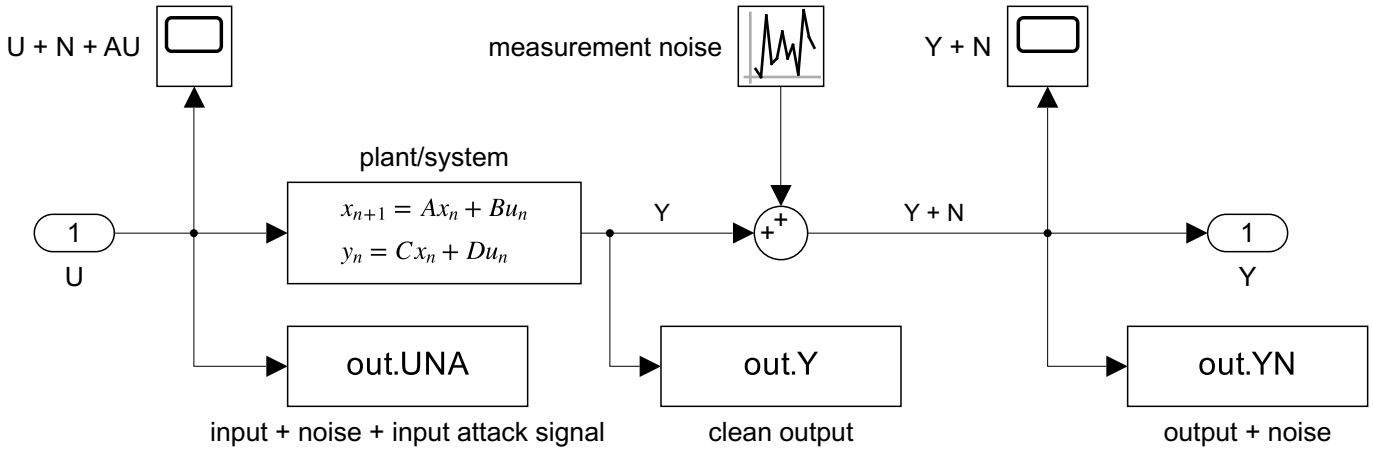
```
% Matthew Reaney, QUB
% April 2023
% https://github.com/mattr862/three-tank-water-system

function [AS, error, SC_out] = Y_state_dependant_filter(as, y_hat, y, D_Info, T, A_Info,  SC_in)
%outputs = Attack Signal, error, Succcessful stealthy attack Count Out
%inputs = Attack Signal, Y_hat, Y, detector info, time, attack info,  Succcessful stealthy attack Count In
AS = zeros(3,1); % intialize attack signal

if A_Info(7) ~= 0
    %% Stealthy Attack Enabled
    % Succcessful stealthy attack Count (SC)
    if T == 0
        SC_in = 0;
    end
    SC_out = SC_in;

    % Stealty success console output
    if T == A_Info(6)
        fprintf('\nInput: Successfully preformed %i stealthy attacks out of a possible %.2i\n', int16(SC_out), int16((A_Info(6) - A
    end

    % Mock Detector Responce
    difference = (y - y_hat);
    error = norm(difference, Inf);

    %If signal is stealthy use it
    if error < D_Info(1) && T >= A_Info(5) && T <= A_Info(6)
        AS = as;
        SC_out = SC_in + 1;
    end

else
    %% Stealthy Attack disabled
    AS = as;
    error = 0;
    SC_out = 0;
end
```


Z-1
1
flag
anomaly detector
flag
flag
out.flag
detector Info
alarm flag
D_Info
D_Info
difference
difference
2
y
Y
U(:)
out.difference
detector
1
y_hat
Y_hat
error
error
T
out.error
in_buffer
out_buffer
Z-1
out.errorbuffer

```
% Matthew Reaney, QUB
% April 2023
% https://github.com/mattr862/three-tank-water-system

function [flag, difference, error, out_buffer] = detector(flag, D_Info, y, y_hat, T, in_buffer)
%% prevent flags during intialising system
if T < 0.25
    flag = 0;
    difference = y - y_hat;
    error = norm(difference, Inf);
    in_buffer = zeros(100,1);
    out_buffer = in_buffer;
    out_buffer(1) = error;
else
    %% Calculate error/flag status for current iteration
    difference = y - y_hat; %difference between x and x^
    error = norm(difference, Inf); %absolute of infinity norm of error

    %% Store error in buffer
    out_buffer = in_buffer;
    out_buffer(1) = error;
    for i = 1:99
         out_buffer(i+1) = in_buffer(i);
         i = i + 1;
    end

    %% Evaluating error in buffer
    if flag == 0
        if D_Info(2) > 1 && D_Info(3) > 1 && D_Info(4) > 1
            % calculate number of consecutive errors exceeding the threshold
            consecutive = 0; i = 0;
            for i = 1:D_Info(3)
                if  out_buffer(i) >= D_Info(1)
                    consecutive = consecutive + 1;
                end
                i = i + 1;
            end
            if consecutive >= D_Info(3) && flag == 0 % consecutive errors check
                flag = T;
                fprintf(['\nAnomaly detector flag raised at %f\n' ...
                    'Due to sufficent consecutive errors exceeding the threshold\n'], T);
            end

            % calculate number of total errors exceeding the threshold
            total = 0; i = 0;
            for i = 1:D_Info(2)
                if  out_buffer(i) >= D_Info(1)
                    total = total + 1;
                end
                i = i + 1;
            end
            if total >= D_Info(4) && flag == 0 % total errors check
                flag = T;
                fprintf(['\nAnomaly detector flag raised at %f\n' ...
                    'Due to sufficent total errors exceeding the threshold\n'], T);
            end
        else
            if error >= D_Info(1)
                flag = T;
```

```
                fprintf('\nAnomaly detector flag raised at %f\n', T);
                fprintf('Error Value = %f\n', error);
            end
        end
    end
end
```


U + Noise
input noise
Y hat
U
1
controller
K*u
estimator
$x_{n+1} = Ax_n + Bu_n$
$y_n = Cx_n + Du_n$
[u; y]
U + N
U
1
Y
out.UN
input + noise
out.U
clean input
2
Y_hat
Y + N + AY
out.Y_hat
estimator output
out.YNA
output + noise + output attack signal


U + N + AU
measurement noise
Y + N
plant/system
$x_{n+1} = Ax_n + Bu_n$
$y_n = Cx_n + Du_n$
1
U
Y
Y + N
1
Y
out.UNA
out.Y
out.YN
input + noise + input attack signal
clean output
output + noise