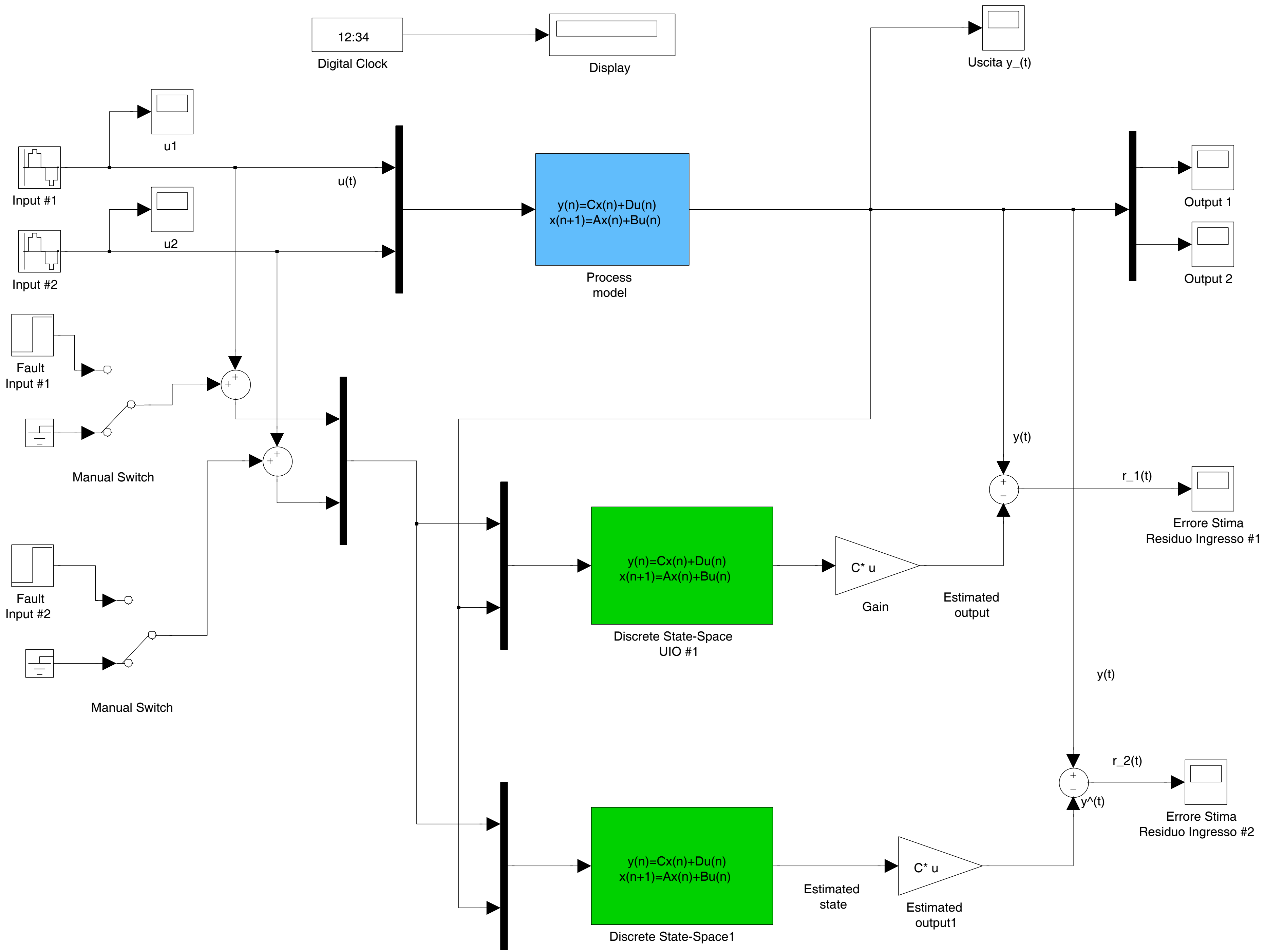
```
%%%
%%% Process model
%%%

clear all, close all, clc


Ts   = 0.1;


A = [0.9944 -0.1203 -0.4302;
     0.0017  0.9902 -0.0747;
     0       0.8187  0];

B = [1 0;
     0 1;
     0 0];

C = [1 0 0
     0 1 0];

D = zeros(2,2);

x0 = [0.01 0.02 0.03];



%%%
%%% Noise processes for the input and output sensors
%%%

q1 = (0.1)^2;  % Noise variance on input #1
q2 = (0.05)^2; % Noise variance on input #2


r1 = (7)^2;   % Noise variance on output #1
r2 = (0.6)^2; % Noise variance on output #2




return
```

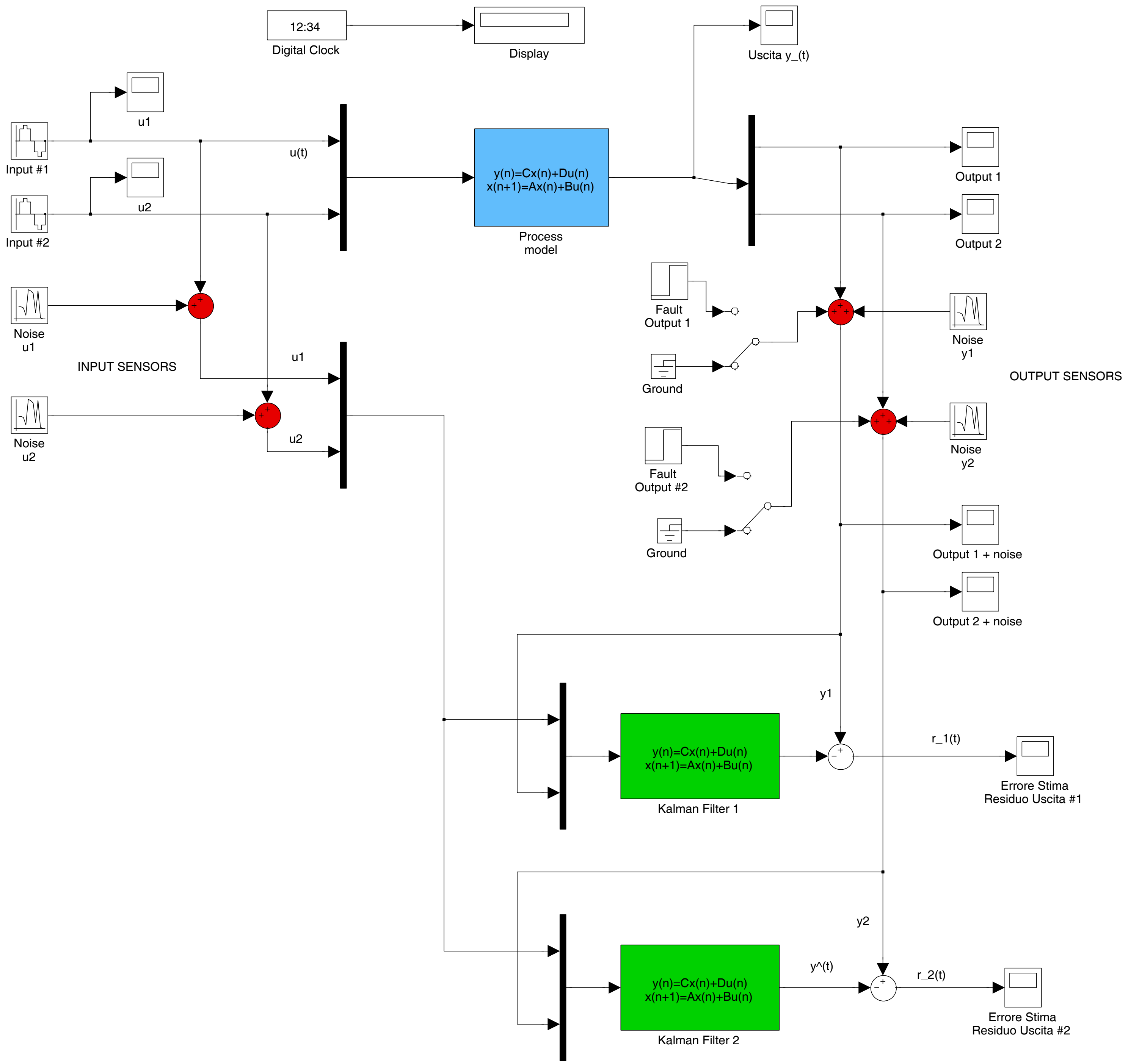
```
%%%
%%% UIO and KF designs
%%%

% It needs the A, B, C and D matrices

%%%
%%% Preliminary tests
%%%

% Controllability

rank(ctrb(A,B(:,1))) % == 3
rank(ctrb(A,B(:,2))) % == 3

% Observability

rank(obsv(A,C(1,:))) % == 3
rank(obsv(A,C(2,:))) % == 3

% UIO

rank(B(:,1)), rank(C*B(:,1)) % Must be equal
rank(B(:,2)), rank(C*B(:,2)) % Must be equal

H1 = B(:,1)*pinv(C*B(:,1));
rank(obsv((A-H1*C*A),C)) % == 3

H2 = B(:,2)*pinv(C*B(:,2));
rank(obsv((A-H2*C*A),C)) % == 3

%%%
%%% KF design for the 2 outputs
%%%

q1 = (0.1)^2;  % Noise variance on input #1
q2 = (0.05)^2; % Noise variance on input #2
Q = diag([q1 q2]);

r1 = (7)^2;   % Noise variance on output #1
r2 = (0.6)^2; % Noise variance on output #2

%%%
%%% KF for output #1
```

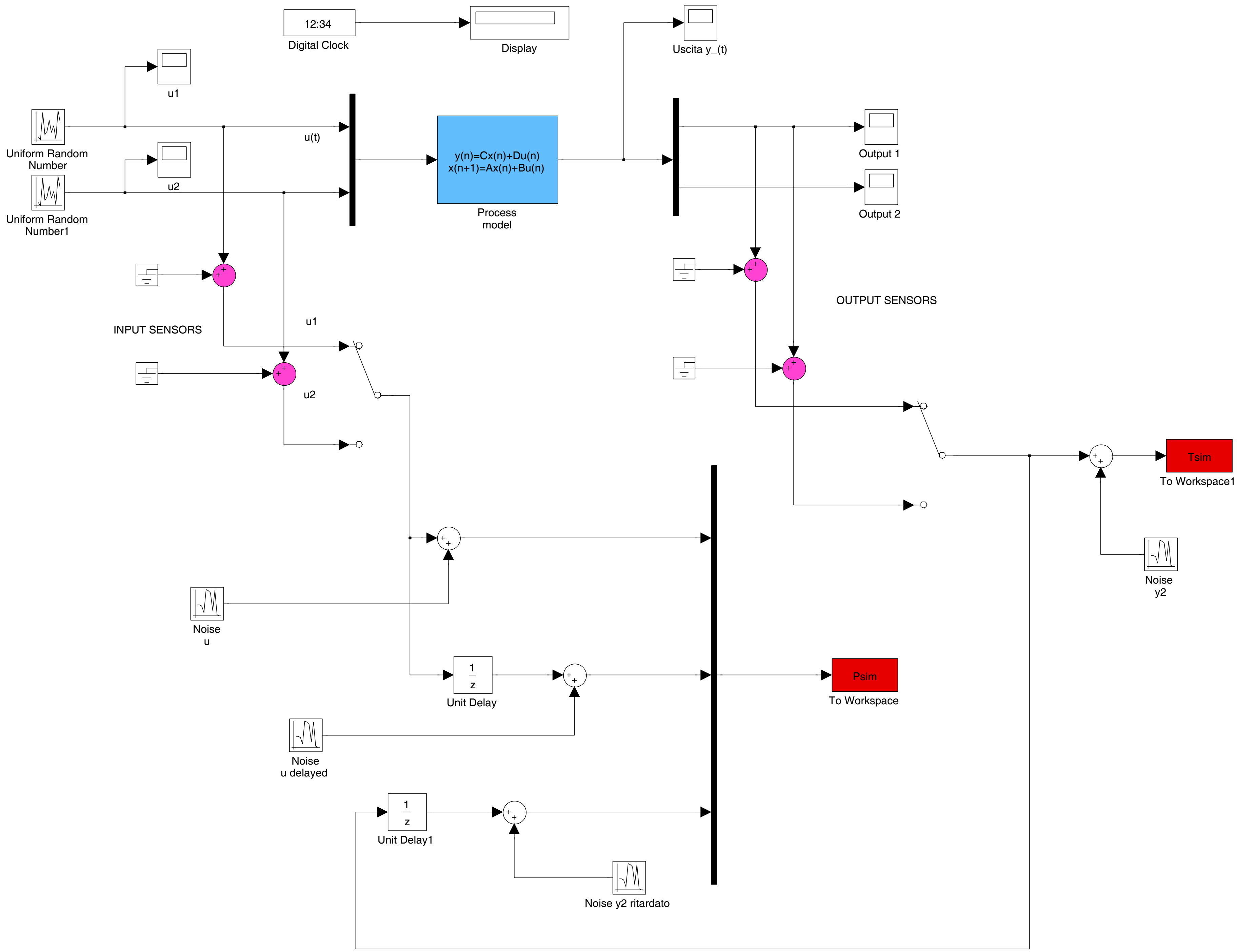
```
%%%

v = [0.75 0.80 0.85]; % Eigenvaues for the 2 UIOs

[Pkf1,Ekf1,Kkft1] = dare(A',C(1,:)',B*Q*B',r1); % KF #1 gain
Kkf1 = Kkft1';

%%% Kalman filter matrices: apart from the Kalman gain,
%%% it is an output observer!

Akf1 = A - Kkf1 * C(1,:);
Bkf1 = [B Kkf1];
Ckf1 = C(1,:);
Dkf1 = zeros(1,3); % 1 output and 3 inputs

%%%
%%% KF for output #2
%%%

[Pkf2,Ekf2,Kkft2] = dare(A',C(2,:)',B*Q*B',r2); % KF #1 gain
Kkf2 = Kkft2';

Akf2 = A - Kkf2 * C(2,:);
Bkf2 = [B Kkf2];
Ckf2 = C(2,:);
Dkf2 = zeros(1,3); % 1 output and 3 inputs


%%%
%%% UIO for isolation of the input #2
%%% just look at the matrix T1*B!
%%%

E1 = B(:,1);
H1 = E1*pinv(C*E1);
T1 = eye(size(H1*C)) - H1*C;
K11  = place( (A-H1*C*A)' , C', v )';
F1 = A-H1*C*A - K11*C;
K21  = F1*H1;
K1 = K11 + K21;

Auio1 = F1;
Buio1 = [T1*B K1];
Cuio1 = eye(3);
```

```
Duio1 = [zeros(3,2) H1];

%%%
%%% UIO for isolation of the input #1
%%% just look at the matrix T2*B!
%%%


E2 = B(:,2);
H2 = E2*pinv(C*E2);
T2 = eye(size(H2*C)) - H2*C;
K12  = place( (A-H2*C*A)' , C', v )';
F2 = A-H2*C*A - K12*C;
K22  = F2*H2;
K2 = K12 + K22;

Auio2 = F2;
Buio2 = [T2*B K2];
Cuio2 = eye(3);
Duio2 = [zeros(3,2) H2];

return
```



12:34
Digital Clock
Display
Uscita y_(t)
u1
Input #1
u(t)
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
u2
Input #2
Output 2
Fault Input #1
Manual Switch
y(t)
r_1(t)
Errore Stima Residuo Ingresso #1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
C* u
Gain
Estimated output
Discrete State-Space UIO #1
Fault Input #2
Manual Switch
y(t)
r_2(t)
y^(t)
Errore Stima Residuo Ingresso #2
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Estimated state
C* u
Estimated output1
Discrete State-Space1

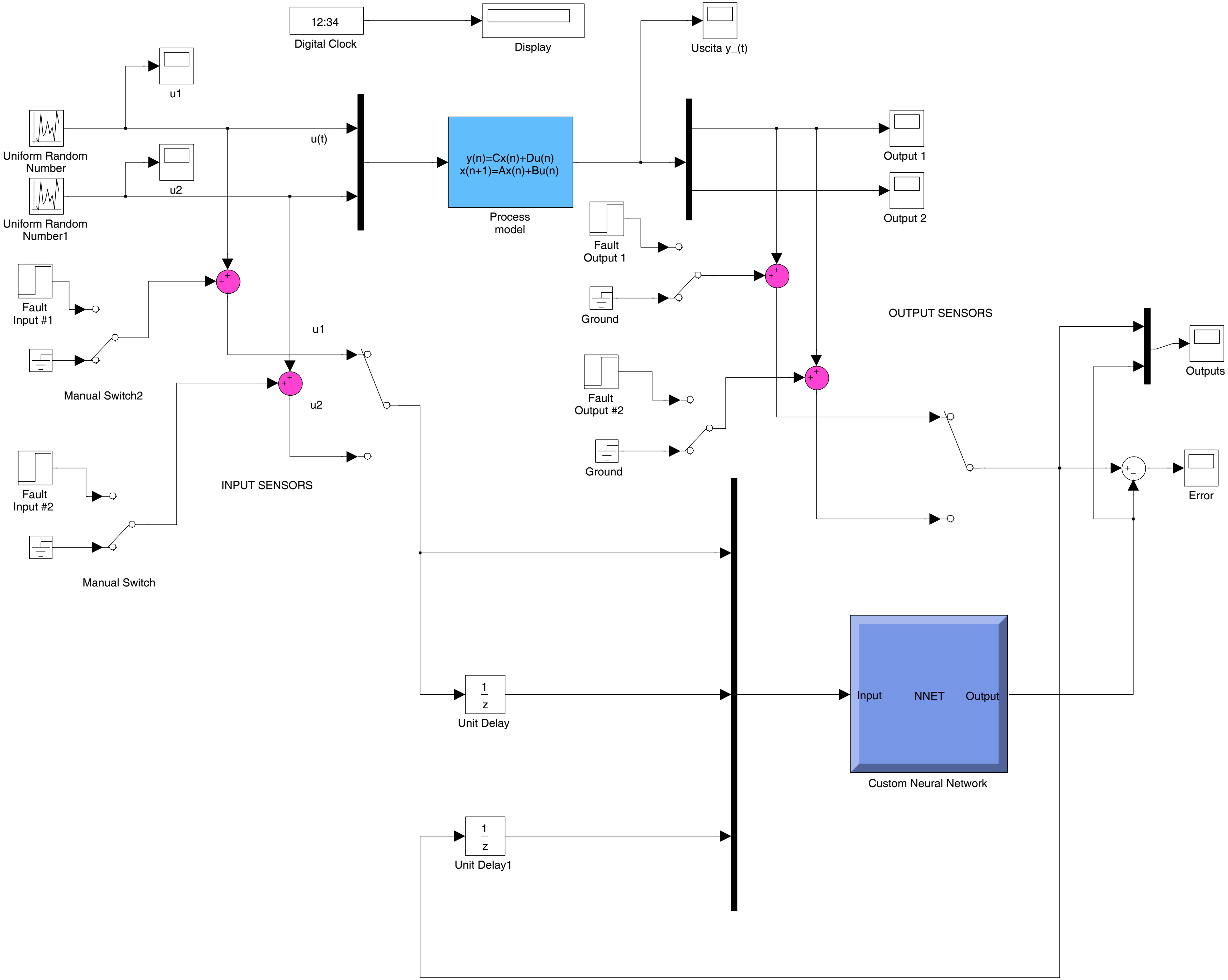

12:34
Digital Clock
Display
Uscita y_(t)
u1
Input #1
u(t)
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process
model
Output 1
u2
Input #2
Output 2
Noise
u1
Fault
Output 1
INPUT SENSORS
u1
Ground
Noise
y1
OUTPUT SENSORS
Noise
u2
u2
Fault
Output #2
Noise
y2
Ground
Output 1 + noise
Output 2 + noise
y1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Kalman Filter 1
r_1(t)
Errore Stima
Residuo Uscita #1
y2
y^(t)
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Kalman Filter 2
r_2(t)
Errore Stima
Residuo Uscita #2



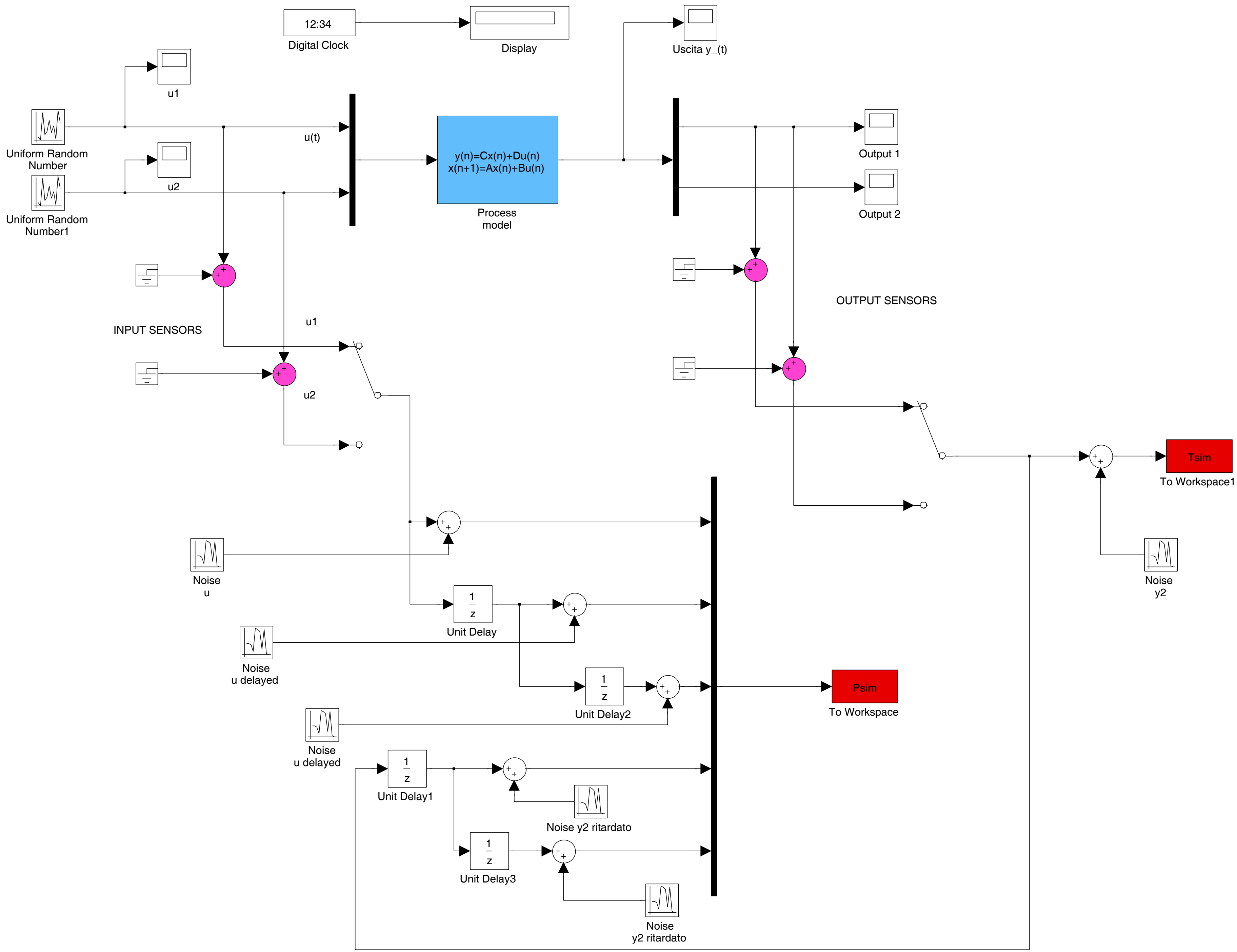
12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
u2
Uniform Random Number1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
OUTPUT SENSORS
INPUT SENSORS
u1
u2
Tsim
To Workspace1
Noise y2
Noise u
1
z
Unit Delay
Psim
To Workspace
Noise u delayed
1
z
Unit Delay1
Noise y2 ritardato

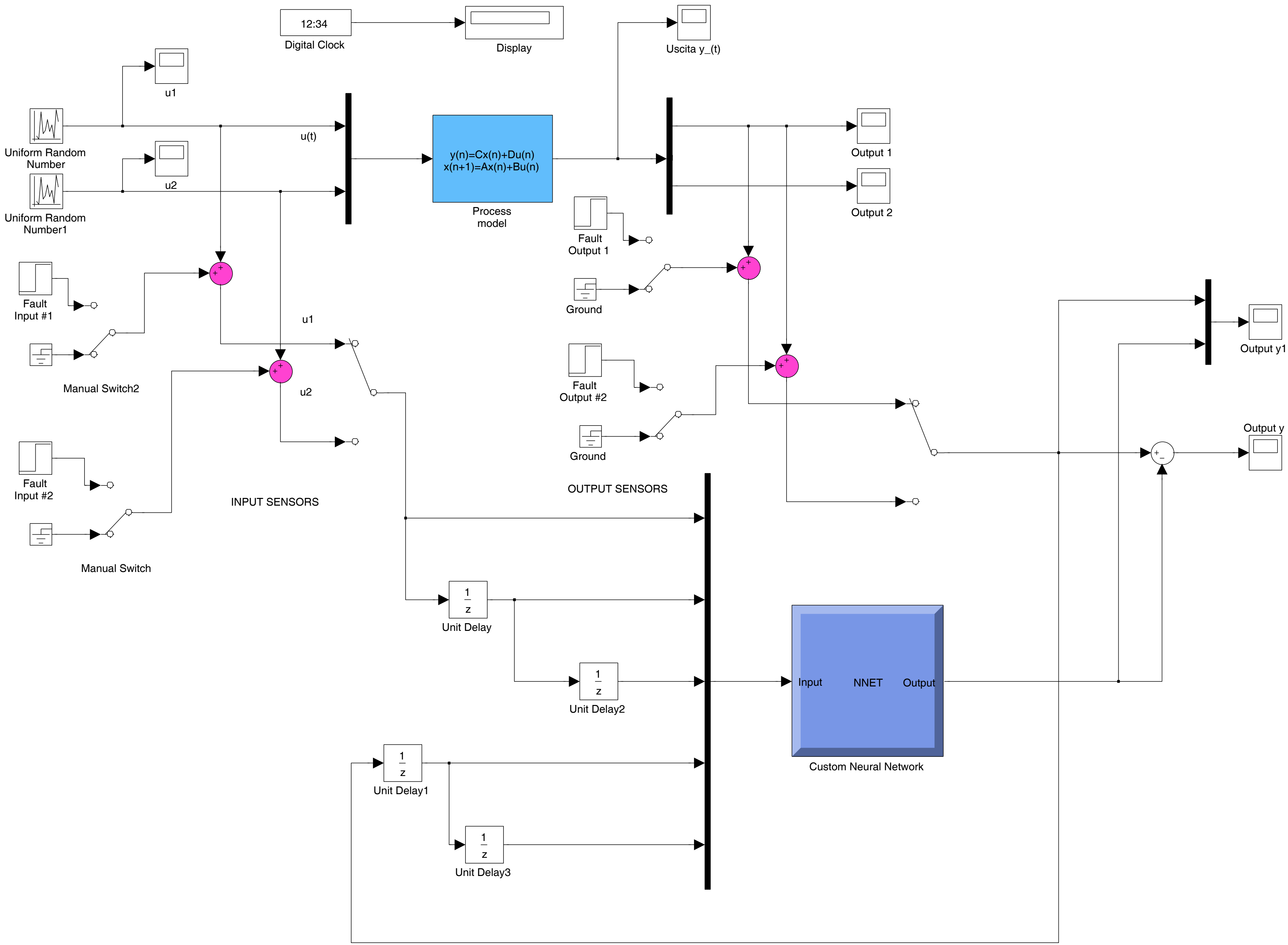
```
%%%
%%% File "train_net3.m": neural network training and generation
%%%

% Caricare Psim e Tsim;

P = Psim(1:round(size(Psim,1)/3),:)';
T = Tsim(1:round(size(Psim,1)/3),1)';

%%%
%%% Neural network parameters
%%%

Si = 4;  % Number of neurons in the input layer
Sh = 8;  % Number of neurons in the hidden layer
So = 1;  % Number of neurons in the output layer
         % It is equal to the rows of the matrix T

TFi = 'tansig'; % Sigmoidal tangent activation function
TFh = 'tansig';
TFo = 'purelin'; % Linear activation function

%BTF = 'traingdx'; % Function for the training of the
                   % backpropagation NN, default
BTF = 'trainlm';   % Levenberg-Marquardt backpropagation

BLF = 'learngdm';  % Backpropagation function
                   % weight/bias, default

PF  = 'mse';      % Performance function Mean Square Error
                  % default

PR  = minmax(P);  % Equal to: [min(P')' , max(P')'], it
                  % determines minimal and maximal values of
                  % inputs and output

val.P  = Psim(round(size(Psim,1)/3)+1:2*round(size(Psim,1)/3),:)';
                                                % validation data
val.T  = Tsim(round(size(Psim,1)/3)+1:2*round(size(Psim,1)/3),:)';
test.P = Psim(2*round(size(Psim,1)/3)+1:end,:)'; % test data
test.T = Tsim(2*round(size(Psim,1)/3)+1:end,:)';

%net = newff(P,T,[Si Sh So],{TFi TFh TFo},BTF,BLF,PF);
                                            % Note: it generates a NN
```

```
                                            % with 4 layers!!!
net = newff(P,T,[Si Sh],{TFi TFh TFo},BTF,BLF,PF);

%%%
%%% Parameters for the NN training
%%%

net.trainParam.epochs = 300;    % Number of epocs
net.trainParam.goal   = 1e-4;   % Value of the final error
net.trainParam.show   = 1;      % Show the plot after 1 epoch
net.trainParam.lr     = 0.05;   % Learning rate for trainlm function
net.trainParam.mc     = 0.9;    % Momentum constant: gradient value
                                % during the training phase: if 0 ->
                                % weights are changed only on the basis
                                % of the gradient; if 1 -> gradient
                                % function is completely neglected

net = train(net,P,T,[],[],val,test); % training function

%Ts = ...;   % Sampling time
             % NOTE: it should be equal to the sampling time used
             % for collecting the matrices Tsim and Psim!!!

net.sampleTime = Ts;

gensim(net,Ts); % It creates the neural network in Simulink

return
```

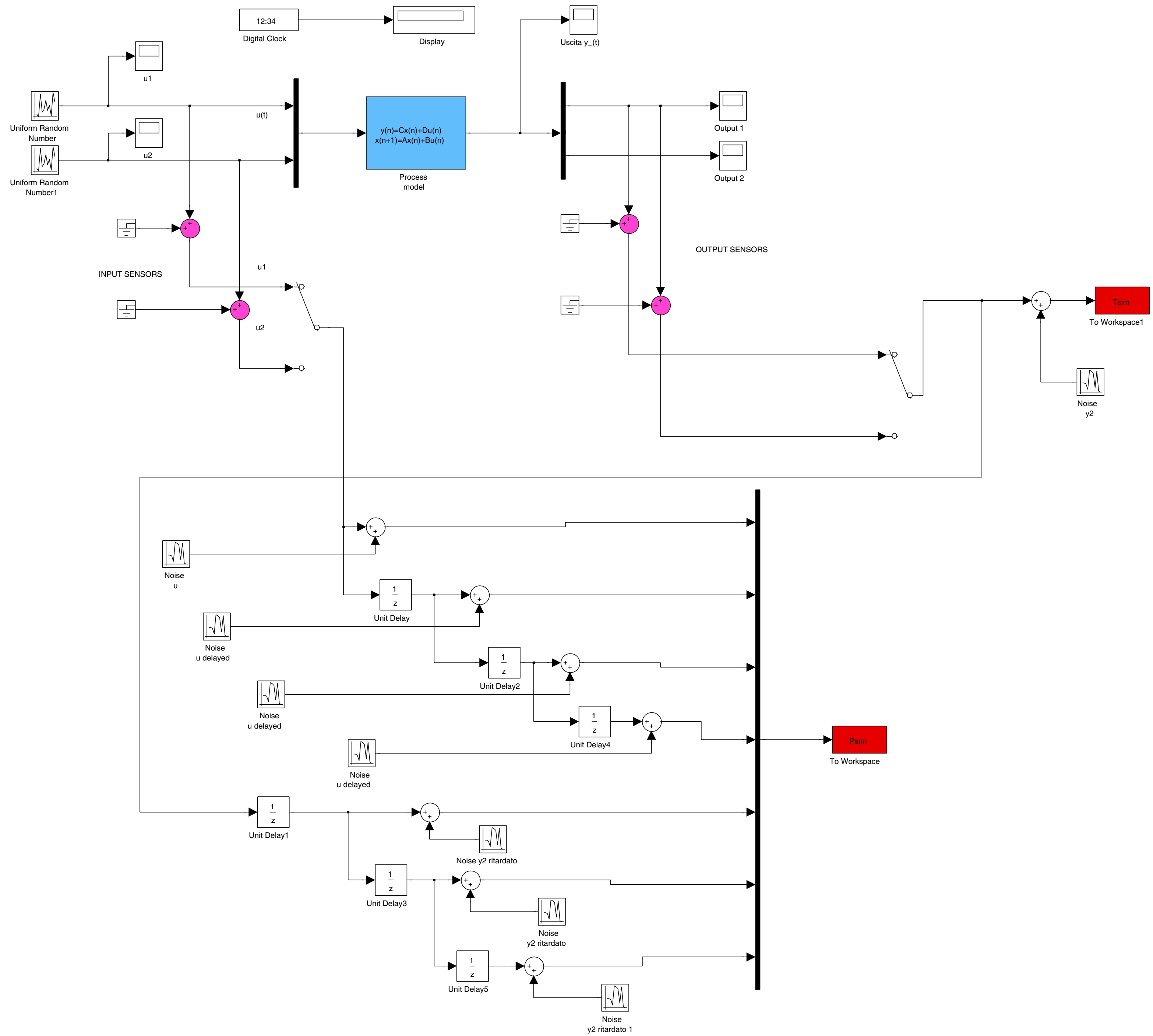


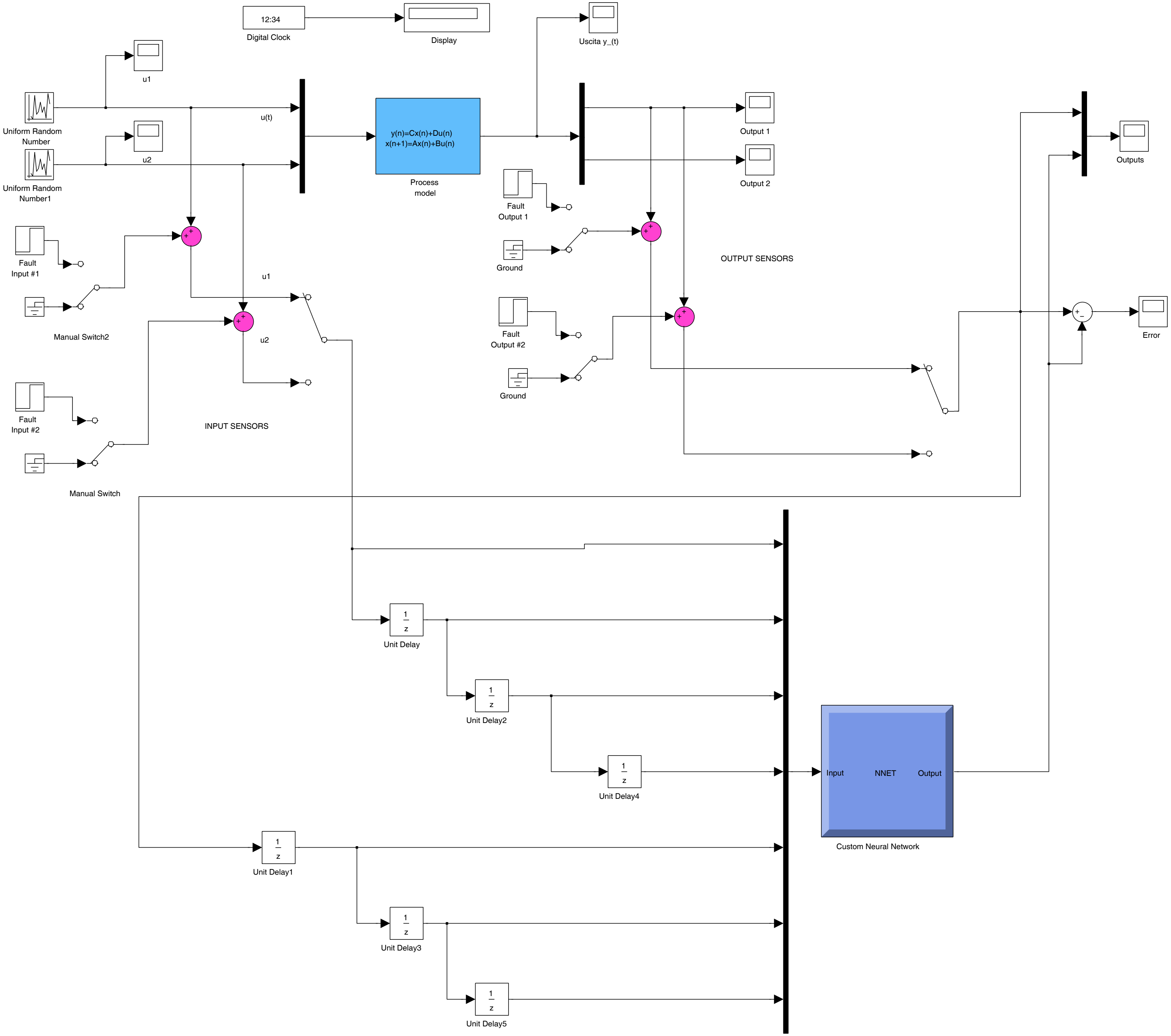
12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
u2
Uniform Random Number1
Output 2
Fault Output 1
Fault Input #1
Ground
OUTPUT SENSORS
u1
Outputs
Manual Switch2
Fault Output #2
u2
Ground
Error
Fault Input #2
INPUT SENSORS
Manual Switch
Input
NNET
Output
1
z
Unit Delay
Custom Neural Network
1
z
Unit Delay1


12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
u2
Uniform Random Number1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
OUTPUT SENSORS
INPUT SENSORS
u1
u2
Tsim
To Workspace1
Noise u
1
z
Unit Delay
Noise u delayed
Noise y2
1
z
Unit Delay2
Psim
To Workspace
Noise u delayed
1
z
Unit Delay1
Noise y2 ritardato
1
z
Unit Delay3
Noise y2 ritardato

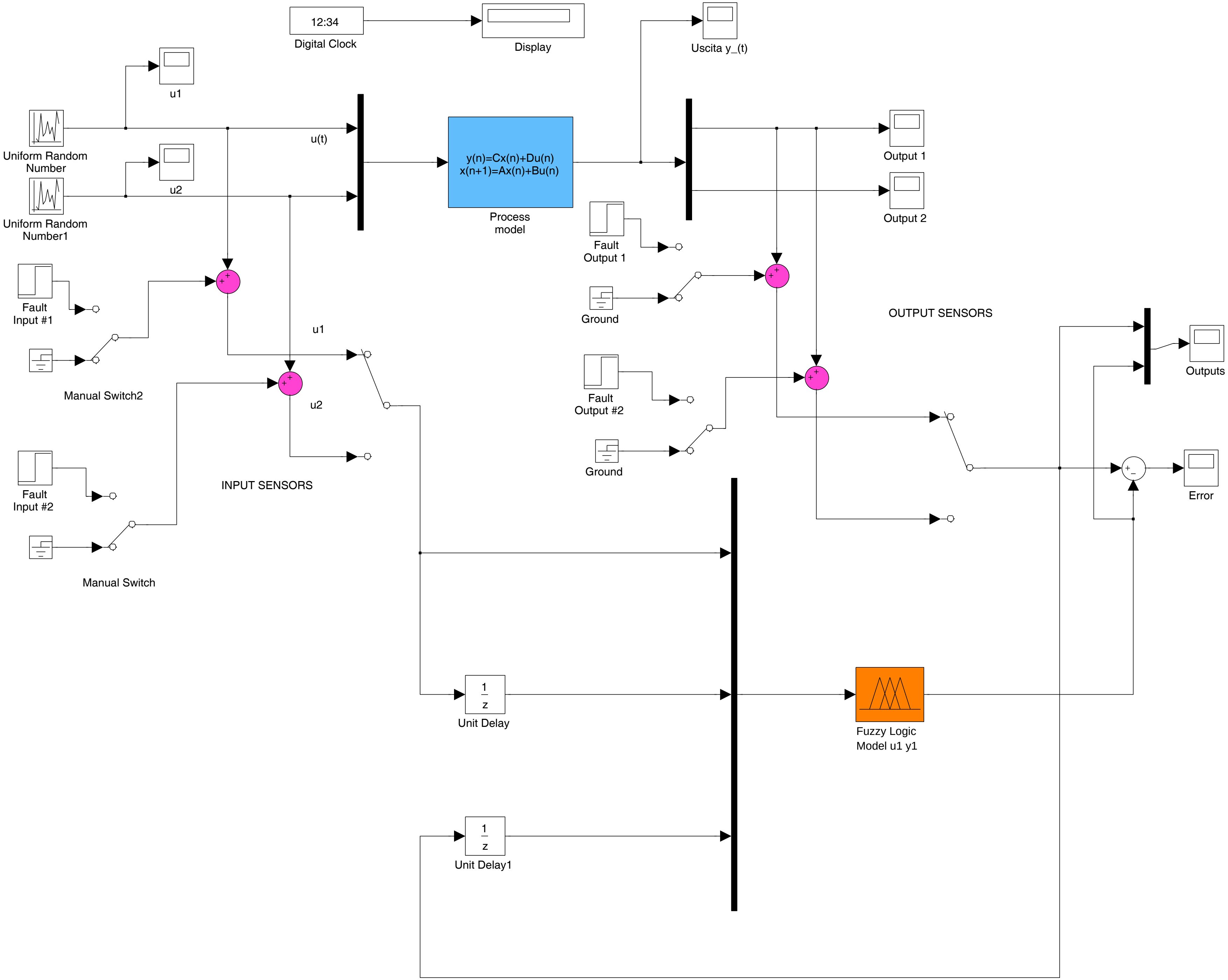


12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
u2
Uniform Random Number1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
Fault Output 1
Ground
Fault Input #1
u1
Manual Switch2
u2
Fault Output #2
Ground
Output y1
Output y
Fault Input #2
INPUT SENSORS
OUTPUT SENSORS
Manual Switch
1
z
Unit Delay
1
z
Unit Delay2
Input
NNET
Output
Custom Neural Network
1
z
Unit Delay1
1
z
Unit Delay3

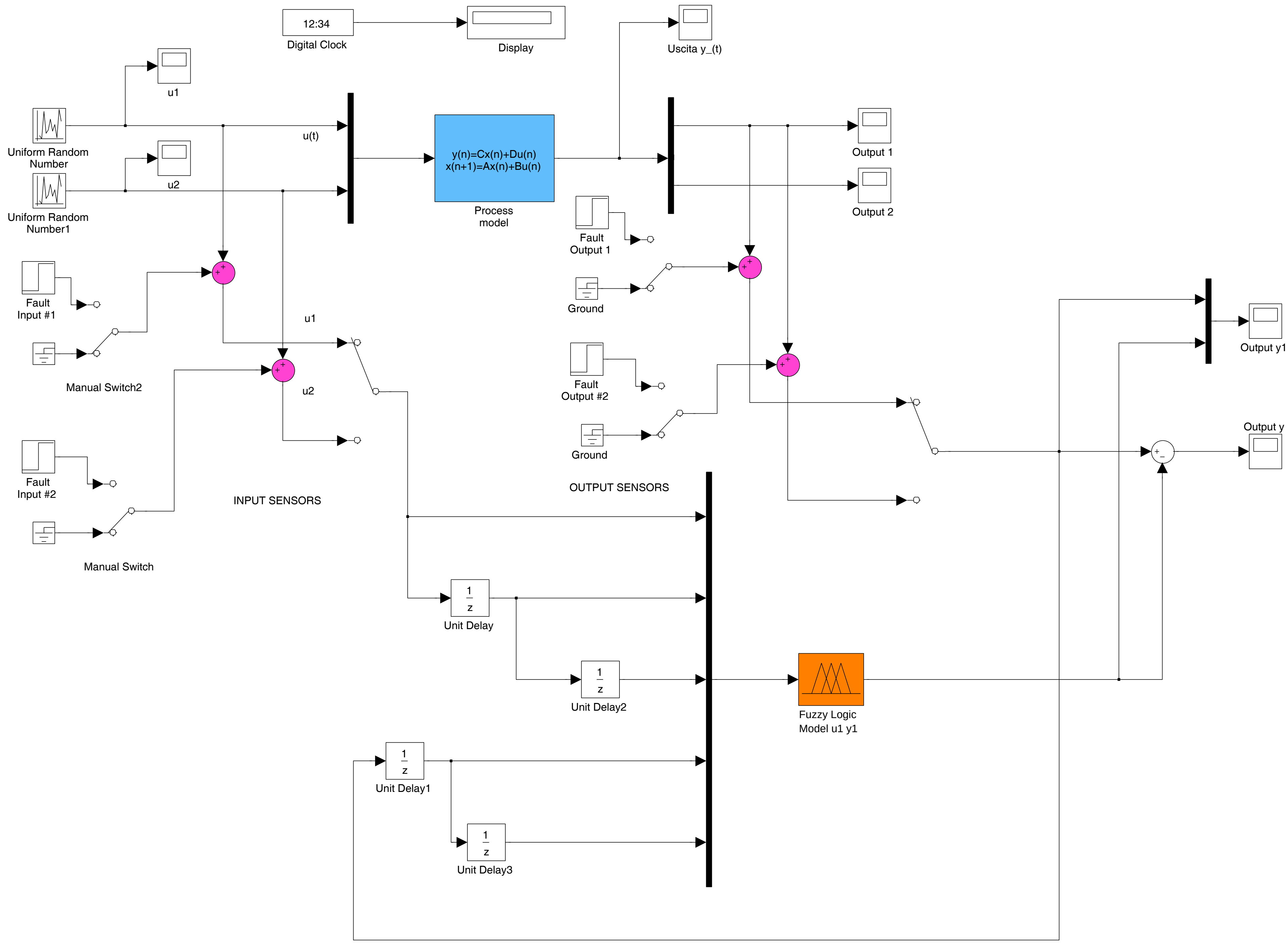

12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u2
Uniform Random Number1
u(t)
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
INPUT SENSORS
u1
u2
OUTPUT SENSORS
Tsim
To Workspace1
Noise y2
Noise u
Unit Delay
Noise u delayed
Unit Delay2
Noise u delayed
Unit Delay4
Noise u delayed
Psim
To Workspace
Unit Delay1
Noise y2 ritardato
Unit Delay3
Noise y2 ritardato
Unit Delay5
Noise y2 ritardato 1

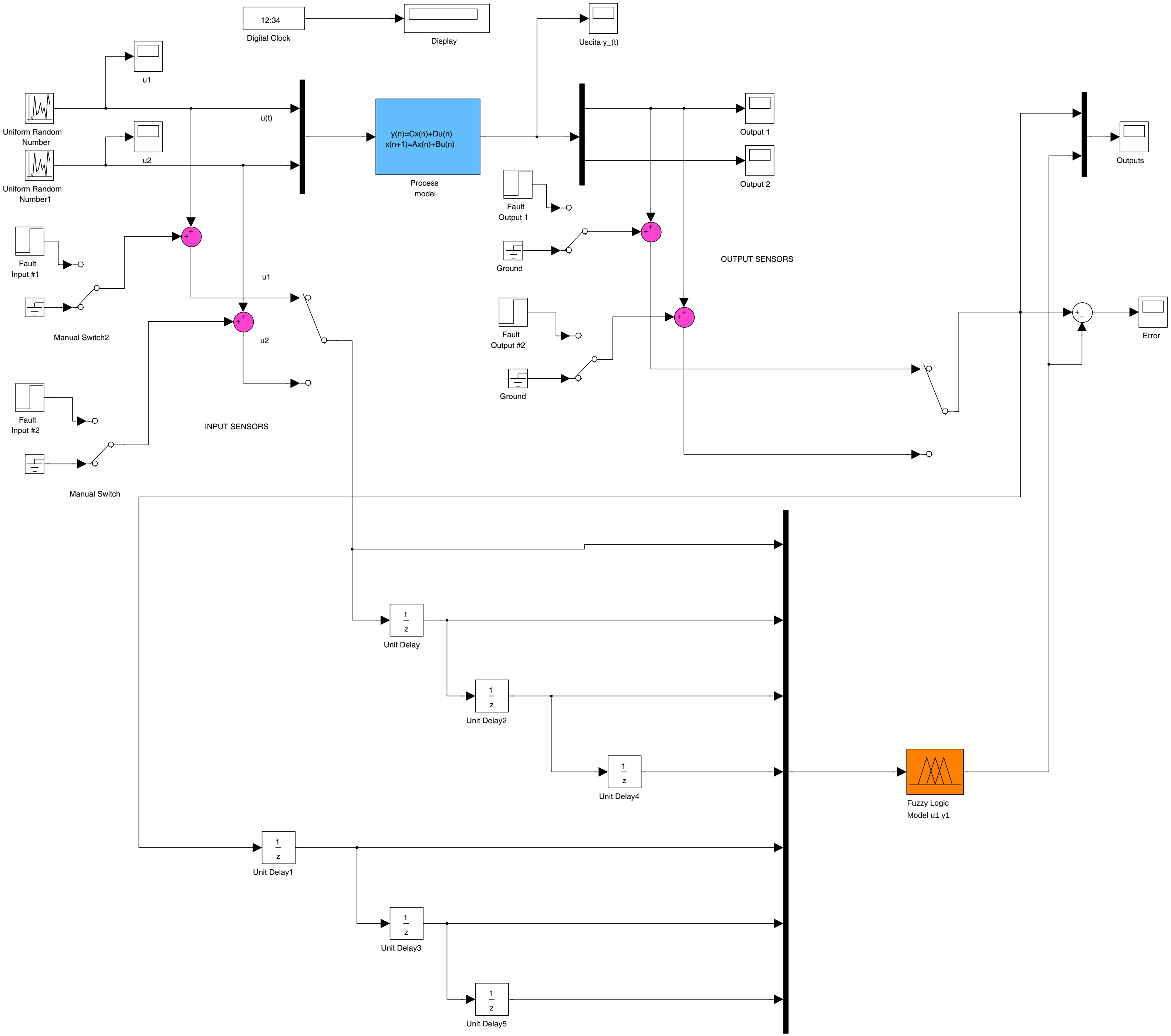

12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random
Number
u(t)
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process
model
Output 1
Output 2
Outputs
u2
Uniform Random
Number1
Fault
Output 1
Ground
Fault
Input #1
OUTPUT SENSORS
u1
Manual Switch2
u2
Fault
Output #2
Ground
Error
Fault
Input #2
INPUT SENSORS
Manual Switch
Unit Delay
Unit Delay2
Unit Delay4
Input
NNET
Output
Custom Neural Network
Unit Delay1
Unit Delay3
Unit Delay5

```
%%%
%%% File "gen_data_train_fuzzy.m": partiziona opportunamente i dati
%%%                                  per il training del sistema fuzzy
%%%

% Caricare P e T che devono essere nel workspace.
% UYtrain: dati di training
% UYval  : dati di validazione
% UYtest : dati di test

UYtrain = [Psim(1:round(size(Psim,1)/3),:)...
           Tsim(1:round(size(Psim,1)/3),1)];

UYval = [Psim(round(size(Psim,1)/3)+1:2*round(size(Psim,1)/3),:)...
       Tsim(round(size(Psim,1)/3)+1:2*round(size(Psim,1)/3),:)];

UYtest = [Psim(2*round(size(Psim,1)/3)+1:end,:)...
          Tsim(2*round(size(Psim,1)/3)+1:end,:)];

return
```



12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
u2
Uniform Random Number1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
Fault Output 1
Ground
Fault Input #1
Manual Switch2
u1
u2
OUTPUT SENSORS
Outputs
Fault Output #2
Ground
Error
Fault Input #2
INPUT SENSORS
Manual Switch
1
z
Unit Delay
Fuzzy Logic Model u1 y1
1
z
Unit Delay1


12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
u2
Uniform Random Number1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
Fault Output 1
Ground
Fault Input #1
Manual Switch2
u1
u2
Fault Output #2
Ground
Output y1
Output y
Fault Input #2
INPUT SENSORS
OUTPUT SENSORS
Manual Switch
1
z
Unit Delay
1
z
Unit Delay2
Fuzzy Logic Model u1 y1
1
z
Unit Delay1
1
z
Unit Delay3



12:34
Digital Clock
Display
Uscita y_(t)
u1
Uniform Random Number
u(t)
u2
Uniform Random Number1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Process model
Output 1
Output 2
Outputs
Fault Output 1
Ground
OUTPUT SENSORS
Fault Input #1
Manual Switch2
u1
u2
Fault Output #2
Ground
Error
Fault Input #2
INPUT SENSORS
Manual Switch
Unit Delay
Unit Delay2
Unit Delay4
Fuzzy Logic Model u1 y1
Unit Delay1
Unit Delay3
Unit Delay5