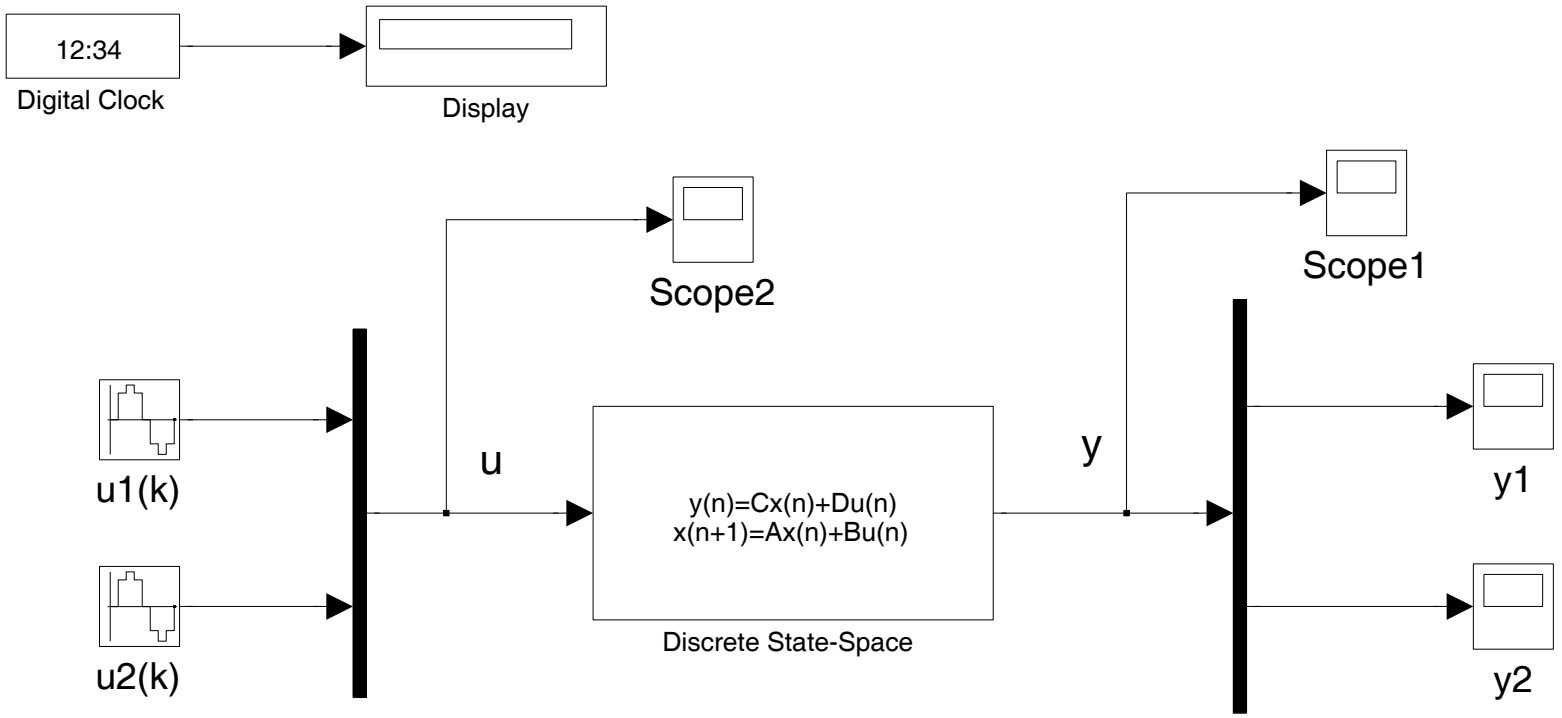
```
A  = [1.4000   -0.7100    0.1540   -0.0120
      1.0000         0         0         0
      0         1.0000         0         0
      0         0         1.0000         0];


B = [0         0
     0.9734    0
     0         0
    -0.8299    1.0000];


C = [1 0 0 0
     0 1 0 1];


D = [0  0
     0  0];


x0 = [0.2 0 0.55 0]

Ts = 0.01;
```


12:34
Digital Clock
Display
Scope2
Scope1
u1(k)
u2(k)
u
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
y
y1
y2

Esercizio

Dato il sistema nello spazio degli stati a tempo discreto descritto dalle matrici assegnate A, B, C e D, con tempo di campionamento $T_s = 0.01s$, e per una durata della simulazione di 10*s*, determinare:

1. Il banco di osservatori con ingressi non noti (UIO) per l'isolamento dei guasti a gradino di ampiezza unitaria che iniziano a 5*s* sui sensori di ingresso; si esegua il progetto degli UIO in Matlab e lo schema della simulazione in Simulink.

2. Il banco di filtri di Kalman per l'isolamento dei guasti a gradino di ampiezza unitaria che iniziano a 5*s* sui sensori d'uscita; i rumori sui sensori sono descritti da processi gaussiani incorrelati a media nulla e varianze secondo i valori:

   ```
   q1 = (0.05)^2; % Rumore su u1
   q2 = (0.03)^2; % Rumore su u2
   r1 = (0.15)^2; % Rumore su y1
   r2 = (0.2)^2;  % Rumore su y2
   ```

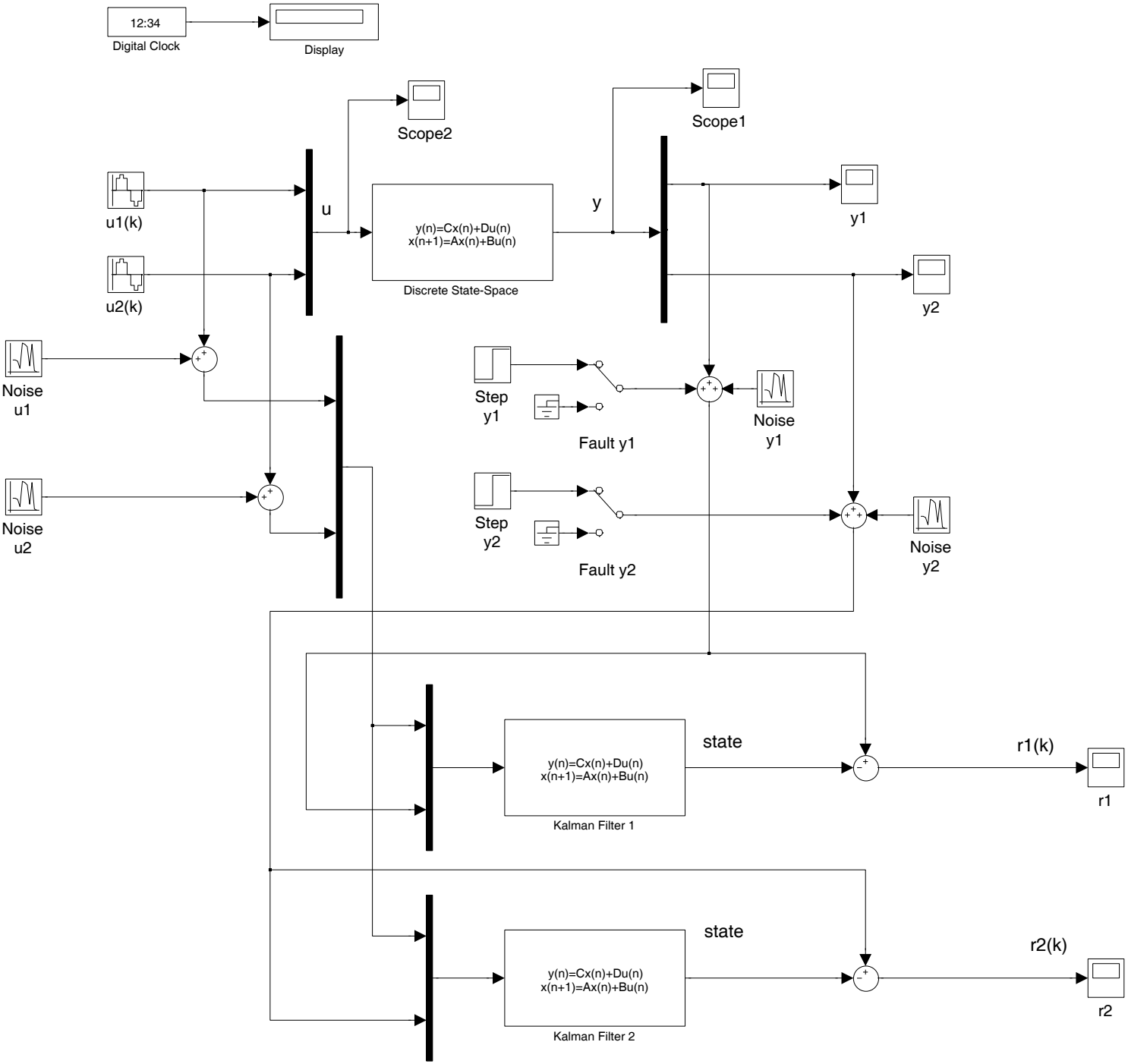
   Si esegua il progetto dei filtri di Kalman in Matlab e lo schema della simulazione in Simulink.

3. Il banco di reti neurali e sistemi fuzzy con 1 ingresso ed 1 uscita in grado di isolare i guasti sui sensori di ingresso e di uscita. Si utilizzi un tempo di campionamento $T_s = 0.05s$ per una durata della simulazione di almeno 300*s*. Si esegua il training in Matlab e gli schemi per le simulazioni in Simulink.

```
%%%
%%% KF conditions
%%%

rank(ctrb(A,B(:,1)))
rank(ctrb(A,B(:,2)))
rank(obsv(A,C(1,:)))
rank(obsv(A,C(2,:)))

%%%
%%% UIO conditions
%%%

E1 = B(:,1);
H1 = E1*pinv(C*E1);
rank(E1)
rank(C*E1)
rank(obsv(A-H1*C*A,C))

E2 = B(:,2);
H2 = E2*pinv(C*E2);
rank(E2)
rank(C*E2)
rank(obsv(A-H2*C*A,C))
```

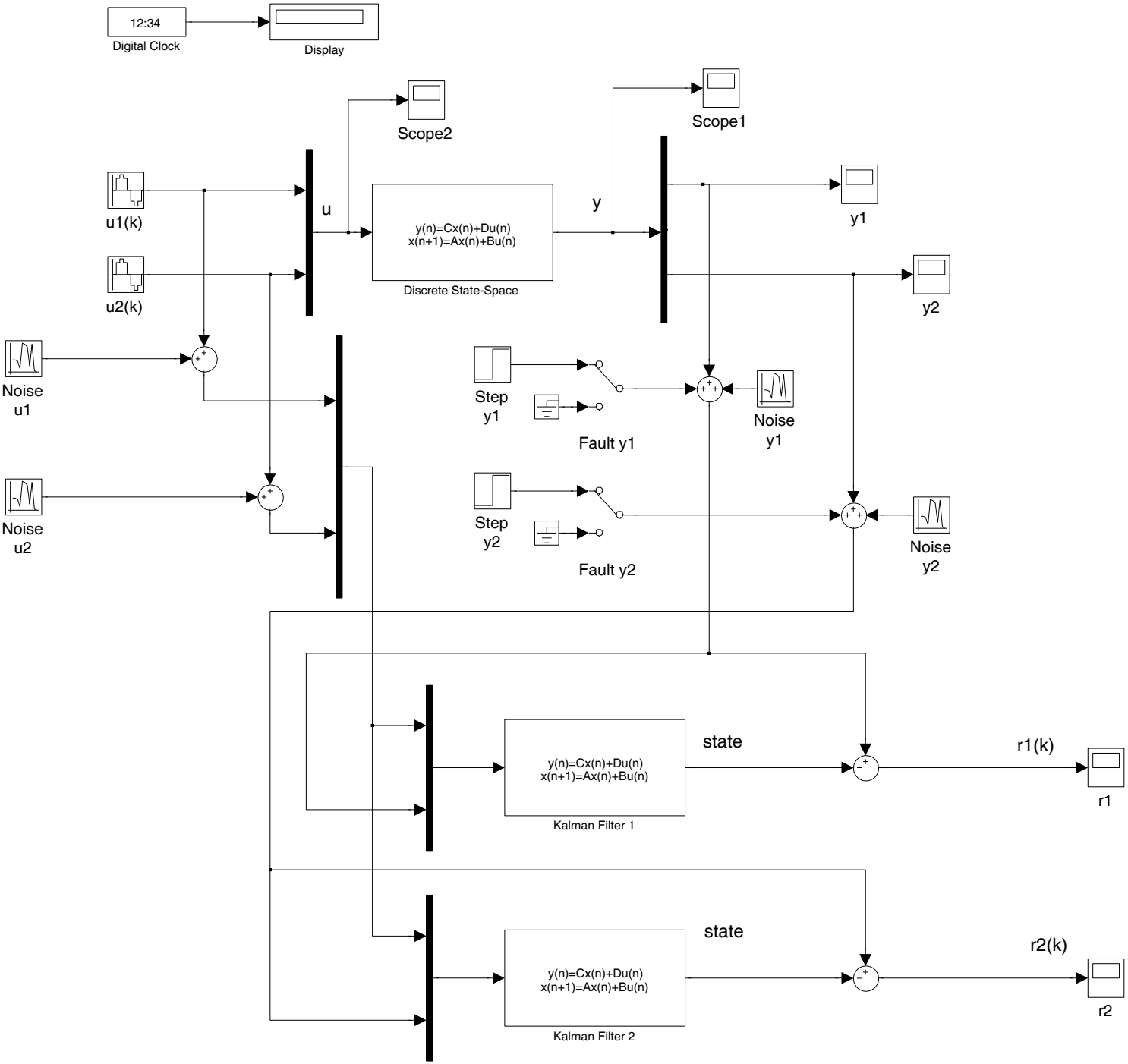
```
%%%
%%% UIO design for isolation of input faults u1 & u2
%%%

v = [0.1 0.2 0.3 0.4];

%%% UIO decoupled from the 1st input: see matrix Buio1!

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(4);     % Dimension of the matrix A: 4 states!
Duio1 = [zeros(4,2) H1]; % 4 outputs and 4 inputs

%%% UIO decoupled from the 2nd input: see matrix Buio2!

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;
```

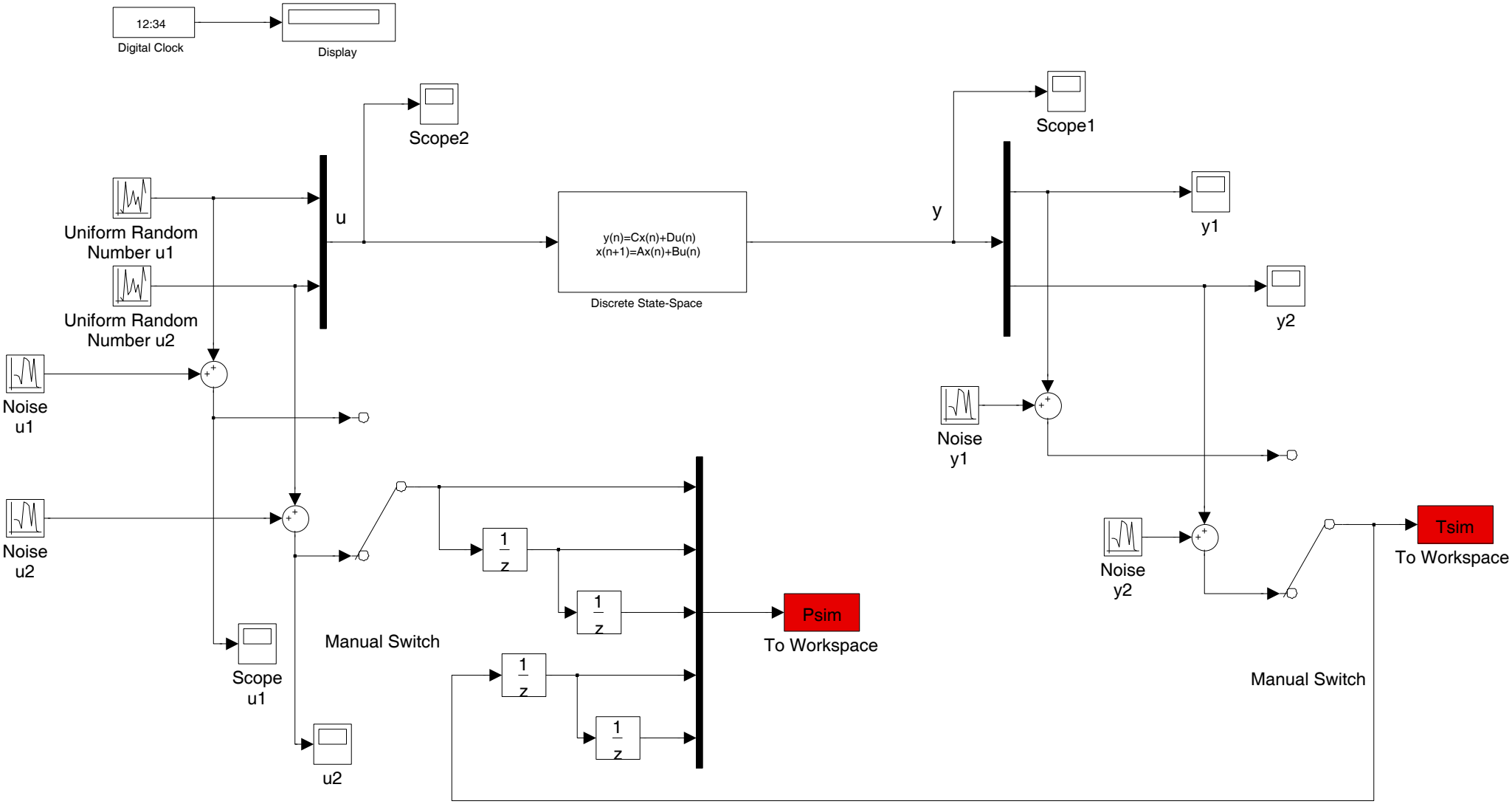
```
Buio2 = [T2*B K2];
Cuio2 = eye(4); % Dimension of the matrix A: 4 states!
Duio2 = [zeros(4,2) H2]; % 4 outputs and 4 inputs

return
```


12:34
Digital Clock
Display
Scope2
Scope1
u1(k)
u2(k)
u
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
y
y1
y2
Noise
u1
Noise
u2
Step
y1
Fault y1
Noise
y1
Step
y2
Fault y2
Noise
y2
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Kalman Filter 1
state
r1(k)
r1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Kalman Filter 2
state
r2(k)
r2

```
%%%
%%% Kalman Filter design example for FDI
%%%

q1 = (0.05)^2; % Noise on the first input u1
q2 = (0.03)^2; % Noise on the second input u2
Q = diag([q1 q2]);

r1 = (0.15)^2; % Noise on the first output y1
r2 = (0.2)^2;  % Noise on the second output y2

%%%
%%% Check the controllability of (A,B) and the
%%% observability of (A,C), which must be equal
%%% to the dimension of A
%%%

c1 = C(1,:); % KF for the first output!

[Pkf1,Ekf1,Kkft1] = dare(A',c1',B*Q*B',r1); % Dual CARE
                                           % for KF1
Kkf1 = Kkft1';

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

Akf1 = A - Kkf1 * c1;
Bkf1 = [B Kkf1];
Ckf1 = c1;
Dkf1 = zeros(1,3); % 1 output and 3 inputs


c2 = C(2,:); % KF for the second output!

[Pkf2,Ekf2,Kkft2] = dare(A',c2',B*Q*B',r2); % Dual CARE
                                            % for KF2
Kkf2 = Kkft2';

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

Akf2 = A - Kkf2 * c2;
Bkf2 = [B Kkf2];
Ckf2 = c2;
```

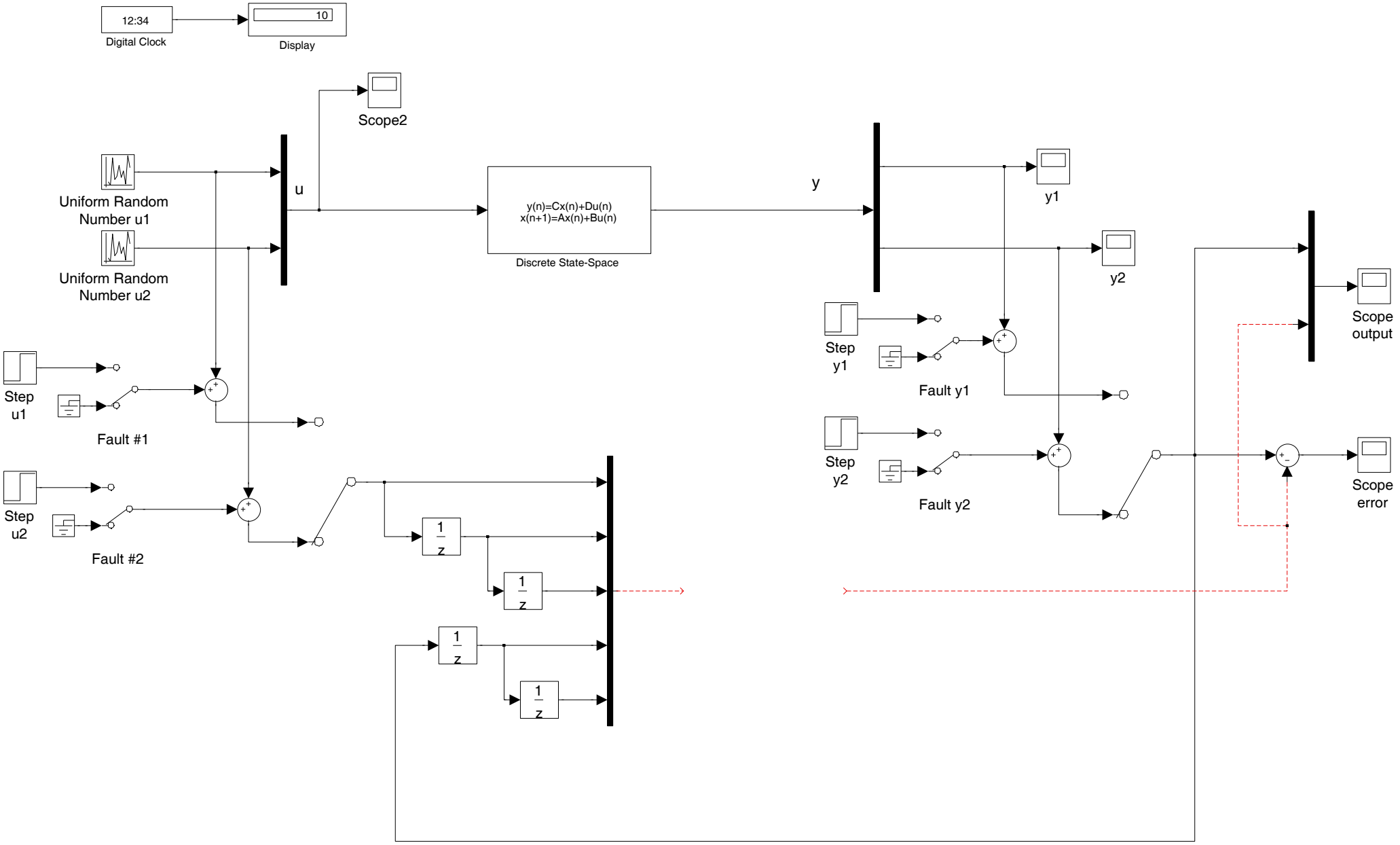
```
Dkf2 = zeros(1,3); % 1 output and 3 inputs

return
```

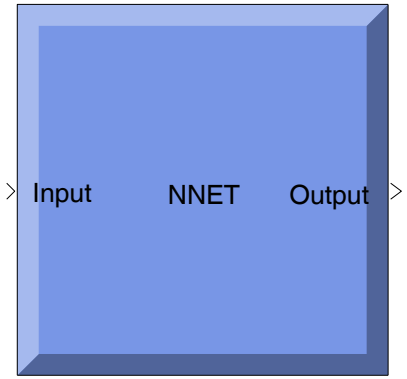

12:34
Digital Clock
Display
Scope2
Scope1
u1(k)
u2(k)
u
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
y
y1
y2
Noise
u1
Noise
u2
Step
y1
Fault y1
Noise
y1
Step
y2
Fault y2
Noise
y2
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Kalman Filter 1
state
r1(k)
r1
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Kalman Filter 2
state
r2(k)
r2


12:34
Digital Clock
Display
Scope2
Uniform Random Number u1
Uniform Random Number u2
u
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
y
Scope1
y1
y2
Noise u1
Noise u2
Noise y1
Noise y2
Scope u1
u2
Manual Switch
1/z
Psim
To Workspace
Tsim
To Workspace
Manual Switch

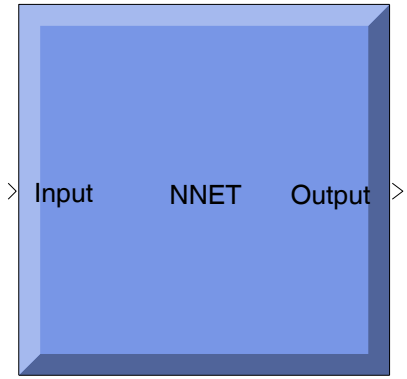
```
%%%
%%% File "train_net3.m": neural network training and generation
%%%

% 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
```

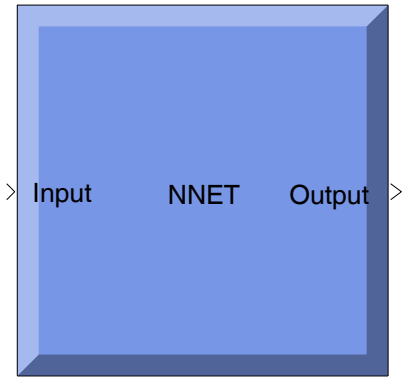

12:34
Digital Clock
10
Display
Scope2
Uniform Random
Number u1
Uniform Random
Number u2
u
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
y
y1
y2
Step
u1
Fault #1
Step
u2
Fault #2
Step
y1
Fault y1
Step
y2
Fault y2
Scope
output
Scope
error
1
z
1
z
1
z
1
z



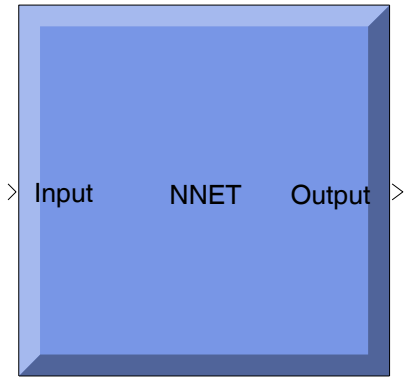
NN u1 y1



NN u1 y2

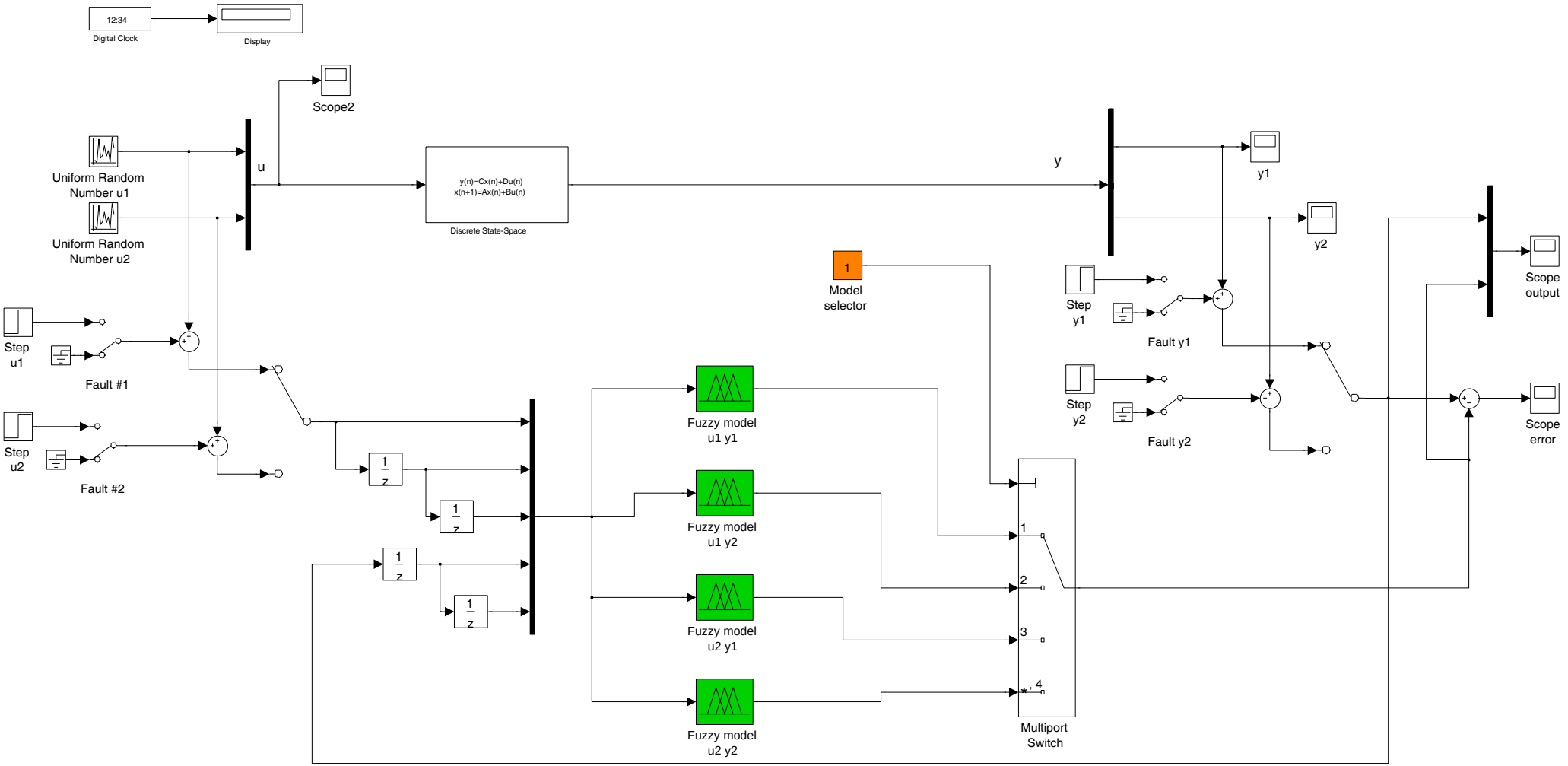


NN u2 y1



NN u2 y2

```
%%%
%%% 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
Scope2
Uniform Random Number u1
Uniform Random Number u2
u
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
y
y1
y2
Scope output
Scope error
Step u1
Fault #1
Step u2
Fault #2
1
Model selector
Step y1
Fault y1
Step y2
Fault y2
1/z
Fuzzy model u1 y1
Fuzzy model u1 y2
Fuzzy model u2 y1
Fuzzy model u2 y2
1
2
3
4
Multiport Switch