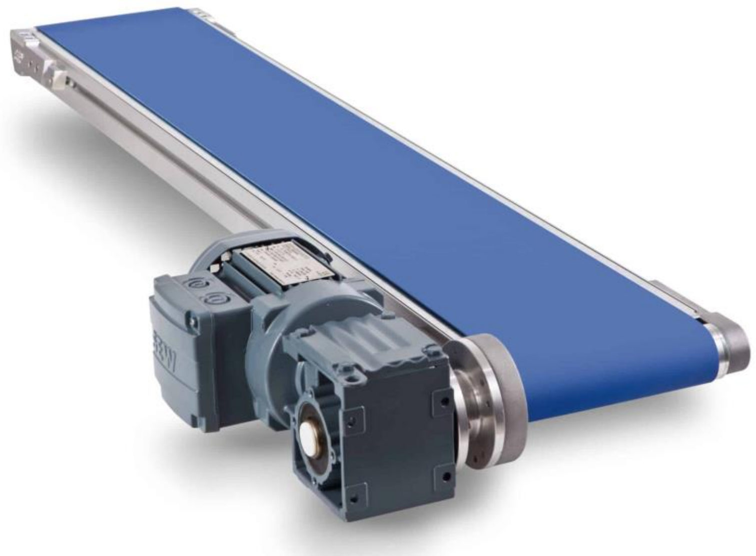
Design and Simulation of an Industrial Conveyor Belt

Yagiz Yagmur

Process Specifications

- Let us consider a factory that manufactures bottled drinks. In an industrial beverage bottling plant, a high-speed conveyor belt system is essential for efficient production and packaging processes. This conveyor belt system is designed to transport cases of bottled drinks from the production line to the wrapping and logistics area.



Process Specifications

- The length of one conveyor is 10 meters and the width is 0.5 meters.
- The desired speed of one conveyor is 2 m/s.
- One conveyor carries three cases of bottled drinks, each weighting 10kg. Conveyor belt has a density of 2 kg/m^2. So we calculate the weight of the belt

$$m = d \times l \times w \times 2 = 2 \times 5 \times 0.5 \times 2 = 10kg$$

Summing up the conveyor and the load mass, total mass is 40kg.

Actuator Specifications

- Required Torque:

$$F = m.g = 40 \times 9.81 = 392.4\ N$$
$$\tau = F \times r = 392.4 \times 0.1 = 39.24\ Nm$$

Safety Margin: $1.2 \times 39.24 = 47.09\ Nm$

- Required Speed:

$$rpm = \frac{60 \times V}{2 \times \pi \times r} = \frac{60 \times 2}{2 \times \pi \times 0.1} = 191\ rpm$$

Safety Margin: $1.1 \times 191 = 210\ rpm$

Actuator: AC Permanent Magnet Servomotor

- Siemens SIMATICS S-1FK7:
  - Permanent Magnet Synchronous Motor
  - Max Torque: 10.5 Nm
  - Max Speed: 6400 rpm
  - Gear Ratio: $\mathrm{N} = \frac{47.09}{10.5} = 4.48 \approx 4.5$
  - Max Angular Speed: $\omega_{max} = \frac{rpm \times 2 \times \pi}{60 \times N} = \frac{6400 \times 2\pi}{60 \times 4.5} \cong 150\ rad/sec$

Transfer Function of The Process

- Torque Constant: K_t = 1.38 Nm/A
- Back emf Constant: K_e = 0.025 V.sec/rad
- Winding Resistance: R = 4.67 ohm
- Motor Inductance: L = 35 mH
- Motor Inertia: J_m = 3.2 x 10^{-4} kg.m^2
- Load Inertia: $J_L = mr^2 = 40\,.\,(0.01) = 0.4\ \text{kg.}m^2$
- Reflected Load Inertia: $J_{ref} = \frac{0.4}{4.5^2} = 0.019\ \text{kg.}m^2$
- Total Inertia: $J_{tot} = J_m + J_{ref} \cong 0.02\ \text{kg.}m^2$

Transfer Function of The Process

- Output Angular Velocity / Input Voltage of the Motor:

$$V(t) = L\frac{dI}{dt} + RI + K_e w$$

$$J_{tot}\frac{dw}{dt} = K_t I - B_m w$$

(Bm is negligible): $I = \frac{J_{tot}}{K_t}\frac{dw}{dt}$

Substitute I on eq.1: $\frac{J_{tot}L}{K_t}\frac{d^2w}{dt^2} + \frac{J_{tot}R}{K_t}\frac{dw}{dt} + K_e w$

Transfer Function of The Process

Laplace Transfer: $V(s) = \frac{J_{tot}L}{K_t}\ \Omega(s)\ s^2 + \frac{J_{tot}R}{K_t}\ \Omega(s)\ s + K_e\ \Omega(s)$

$$G(s) = \frac{\Omega(s)}{V(s)} = \frac{K_t}{JLs^2 + JRs + K_t K_e}$$

$$G(s) = \frac{1.38}{0.0007s^2 + 0.1s + 0.0048}$$

Controller: PLC – Siemens S7-1200

- Receives speed, torque and encoder values from the driver; transmits speed and torque values to driver.
- Siemens S7-1200:
  - High speed processing
  - Digital and Analog I/Os
  - Integrated Profinet/Profibus communication
- Siemens TIA Portal programming
- Motor Driver: SIMATICS S120
  - Rise time < 3ms

PI Control

- Setpoint: 2 m/s = 20 rad/sec
- Settling Time: 1s
- Actuator Limitation: $K_p e_{max} \leq w_{max} = 20K_p \leq 150 = K_p \leq 7.5$
- TF Order Reduction using MATLAB:

```
% Define the transfer function
num = 1.38;
den = [7e-4 0.1 4.83e-3];
sys = tf(num, den);
% Convert the transfer function to state-space representation
[A, B, C, D] = tf2ss(num, den);
% Balance the realization of the state-space model
sys_ss = ss(A, B, C, D);
sys_ss_balanced = balreal(sys_ss);
% Reduce the order of the state-space model
sys_ss_reduced = modred(sys_ss_balanced, [2], 'truncate');
% Convert the reduced state-space model back to a transfer function
[num_reduced, den_reduced] = ss2tf(sys_ss_reduced.A, sys_ss_reduced.B, ...
    sys_ss_reduced.C, sys_ss_reduced.D);
% Display the reduced transfer function
G_reduced = tf(num_reduced, den_reduced)
```

- $G(s) = \frac{1.38}{0.0007s^2+0.1s+0.0048} \rightarrow \frac{13.8}{s+0.048}$

PI Control

$$0.07\,\dot{w} + 0.0035\,w = K_p e + \int_0^t K_I e\,d\tau$$

$$0.07\,\ddot{w} + \left(0.0035 + K_p\right)\dot{w} + K_I w = K_I w_d$$

$$0.07\,\lambda^2 + \left(0.0035 + K_p\right)\lambda + K_I = 0$$

$$\lambda = \frac{-\left(0.0035 + K_p\right) \pm \sqrt{\left(0.0035 + K_p\right)^2 - 4\,(0.07)K_I}}{2\,(0.07)}$$

PI Control

$$t_s \leq 1s \cong 5\tau$$
$$\tau \leq 0.2$$

$$\frac{0.14}{\left(0.0035 + K_p\right)} \leq 0.2 \rightarrow K_p \geq 0.7$$

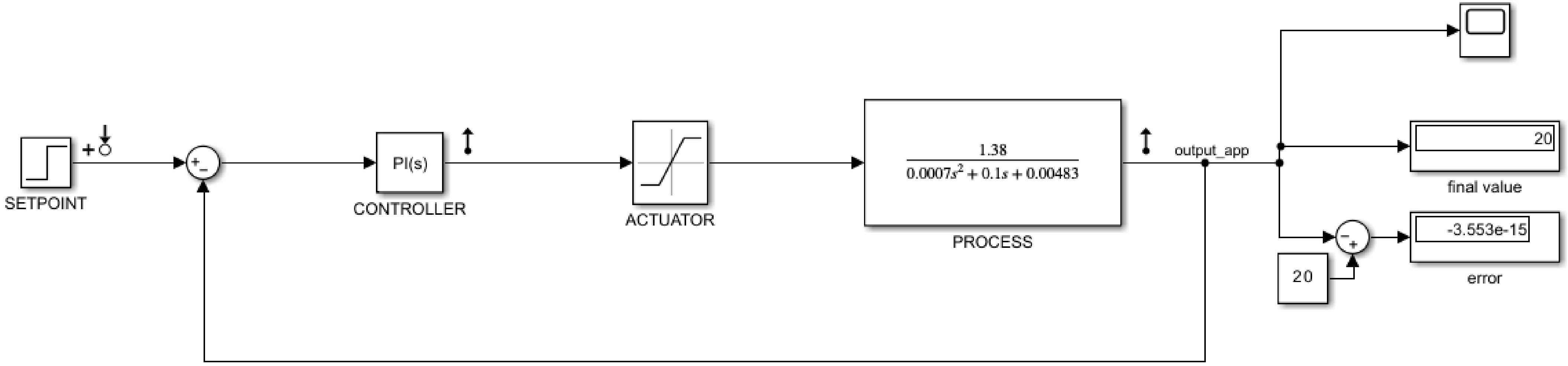
$$Kp = 1$$

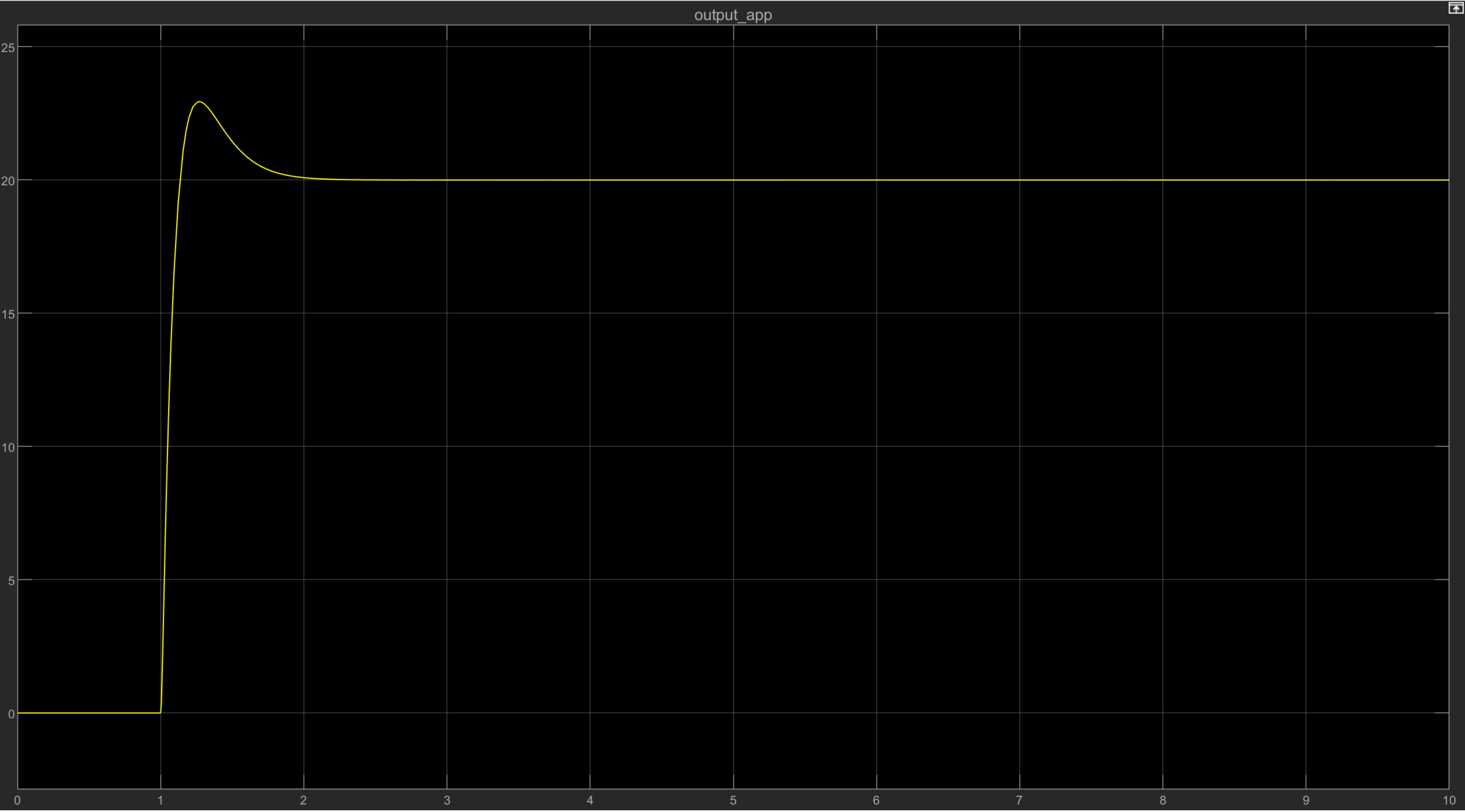
$$\left(0.0035 + K_p\right)^2 - 4\,(0.07)K_I < 0 \rightarrow K_I > 3.57$$

$$K_I = 3.6$$

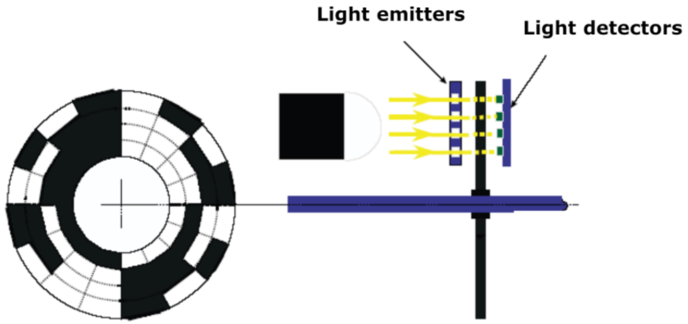
Application Layer Simulation (1)

- SETPOINT: Generates a step input with the magnitude of 20 (desired angular speed) at t=1.
- CONTROLLER: PI controller with the constants calculated.
- ACTUATOR: Limits the input according to the physical constraints of the motor (max speed = 150 rad/sec).
- PROCESS: Transfer function of the process.




output_app
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10

Sensor: Absolute Encoder



- Siemens AM24DQI
  - Optical Encoder
  - 24 bit resolution (16777216 pulse)
  - Accuracy: ± 0.01 rad/sec
  - Response Time: t < 1ms (We will consider t = 1ms)
  - Transfer Function:

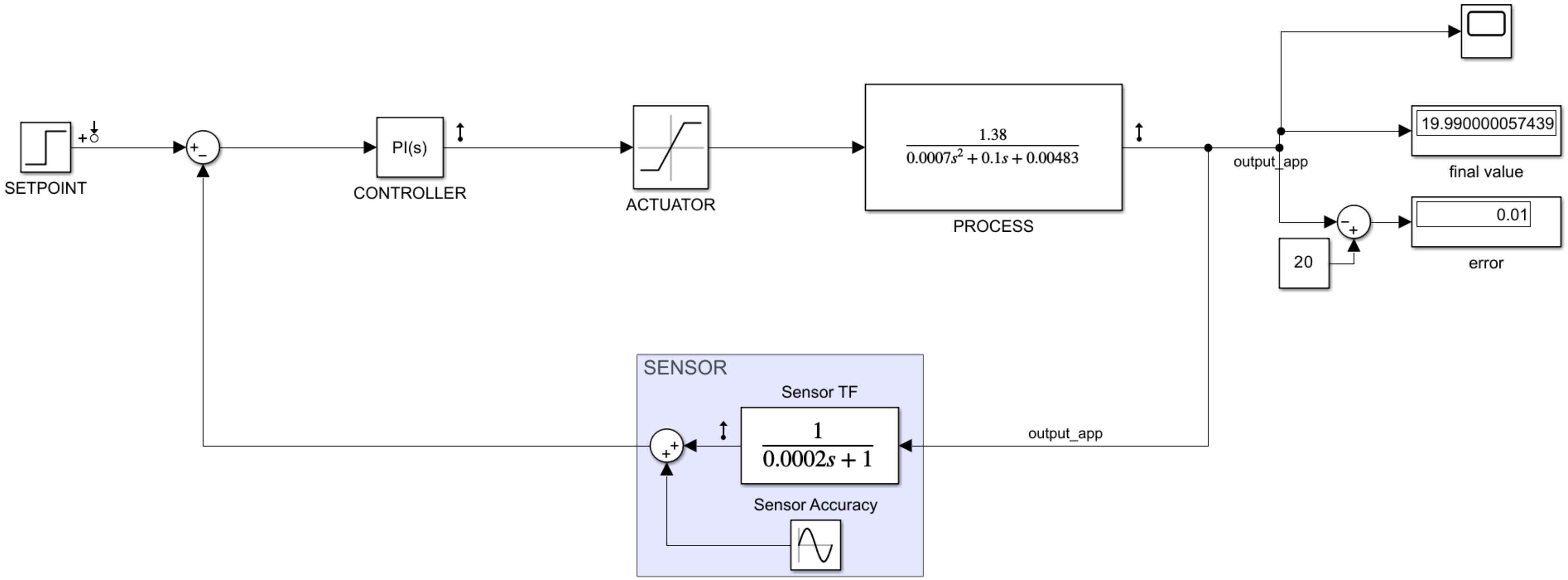
$$\tau \cong \frac{t}{5} = 0.0002s$$

$$H(s) = \frac{1}{0.0002s + 1}$$

Application Layer Simulation (2)

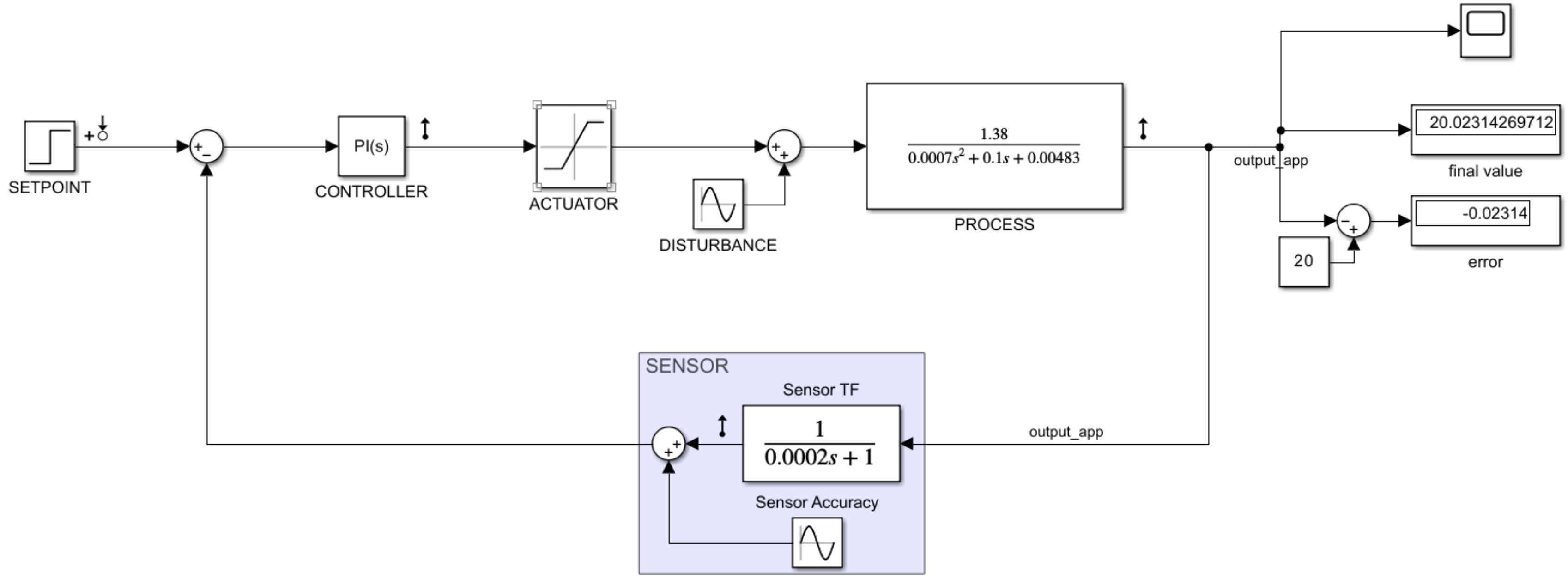
- SENSOR: Sensor transfer function and accuracy (0.01) on feedback.

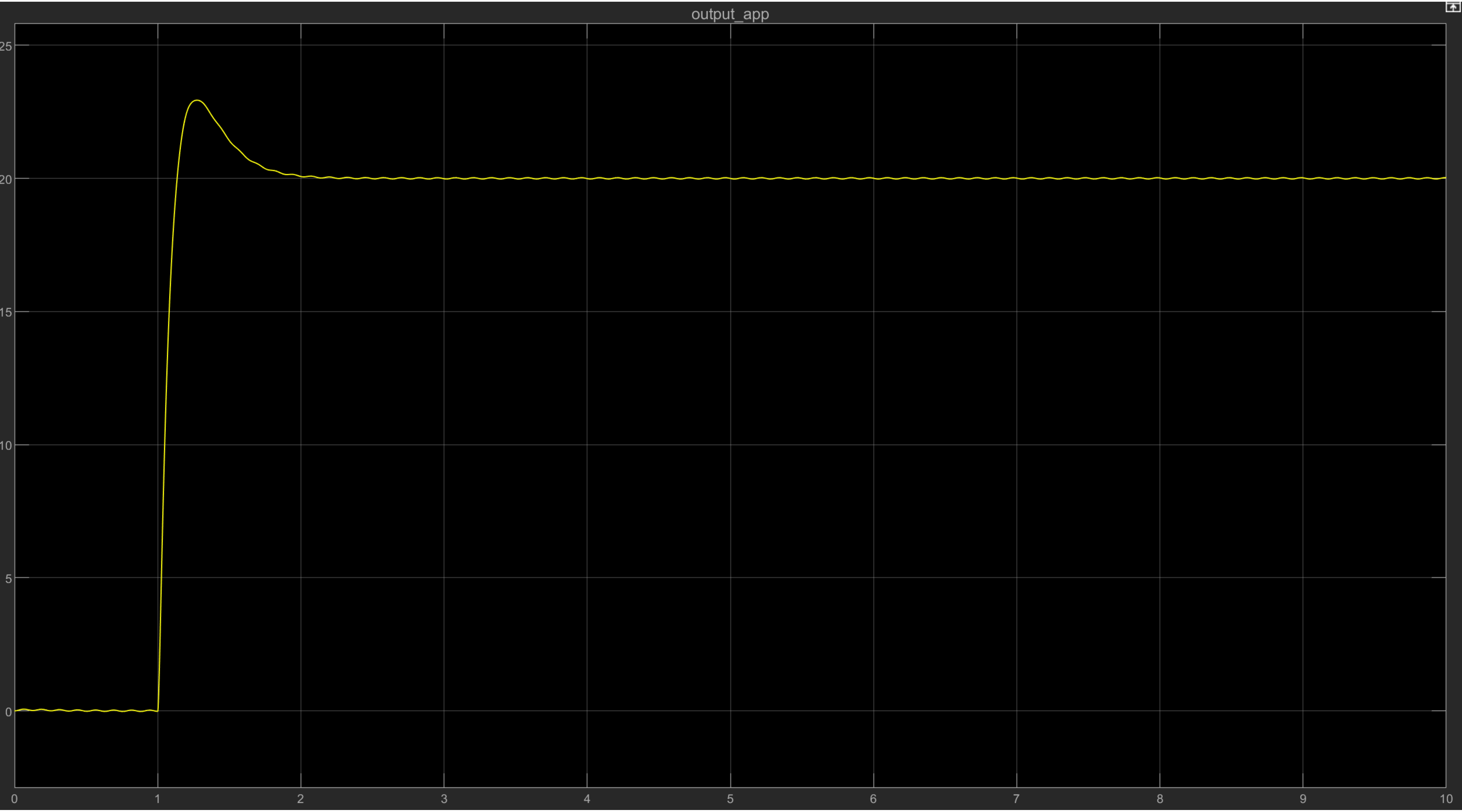
Addition of the sensor characteristics to the simulation slightly increases the steady state error due to small miscalculations of the sensor. Since the time constant of the sensor is very low, the delay is negligible and does not affect the process.



Application Layer Simulation (3): Disturbances

- The process gets affected from the noise, vibration, temperature and power fluctuations and many more factors.
- In order to simulate this disturbances, we add a disturbance block between the actuator and the process(50Hz, amplitude = 0.1, bias = 0.05).
- With the addition of disturbances, steady state error is even higher and we can see small fluctuations on the output.




output_app
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10

Network Layer: Profibus

- Fiber optic data communication between PLCs, sensors, actuators, HMIs.
  - Profibus DP (Decentralized Peripherals)
  - Profibus PA (Process Automation)
- Bandwidth: 12 Mbps / 9.6 Kbps
- Max. Buffer Size: 244 bytes = 1952 bits
- Up to 126 nodes

Data Rates

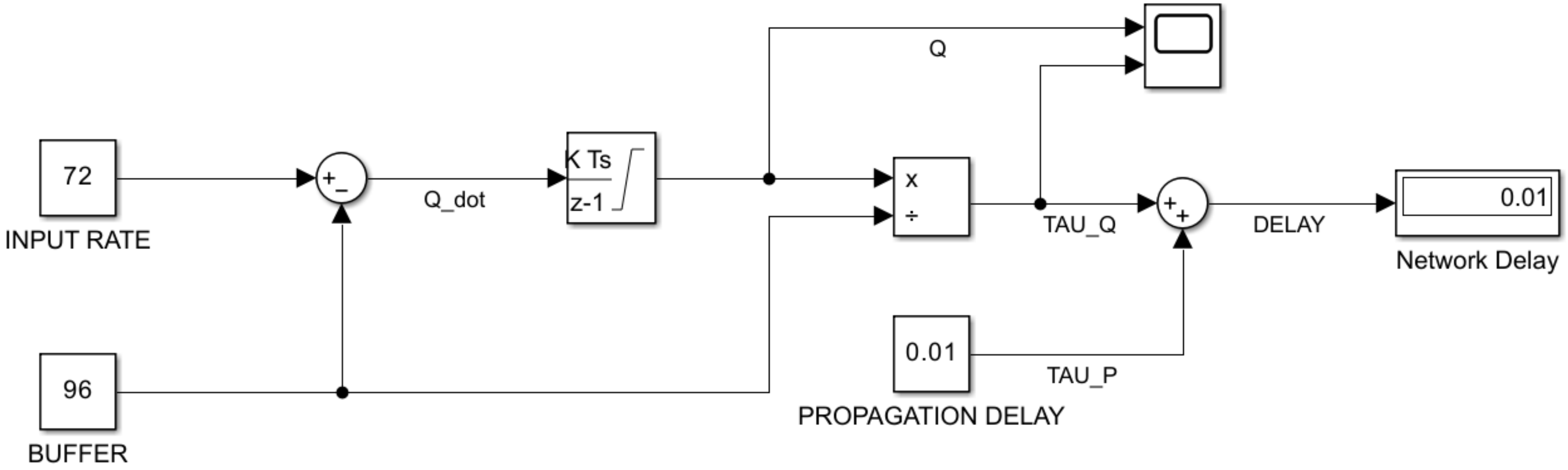
- From PLC to Driver:
  - Address (8) + Control (8) + Data Length(8) + Speed (16) + Torque (16) + Command (8) + Error checking (8) = 72 bit
  - 1 kHz rate: 72 x 1000 = 72 kbps
- From Driver to PLC:
  - Same as above with extra 24 bit encoder data: 96 bit
  - 1 kHz rate: 96 x 1000 = 96 kbps

Network Layer Simulation (1)

$$\dot{q} = \mathrm{r} - \mathrm{c}$$

$$\tau_q = \frac{q}{c}$$

- Bandwidth and buffer selected 9.6 Mbps and 96 kbps respectively in order to avoid queues.

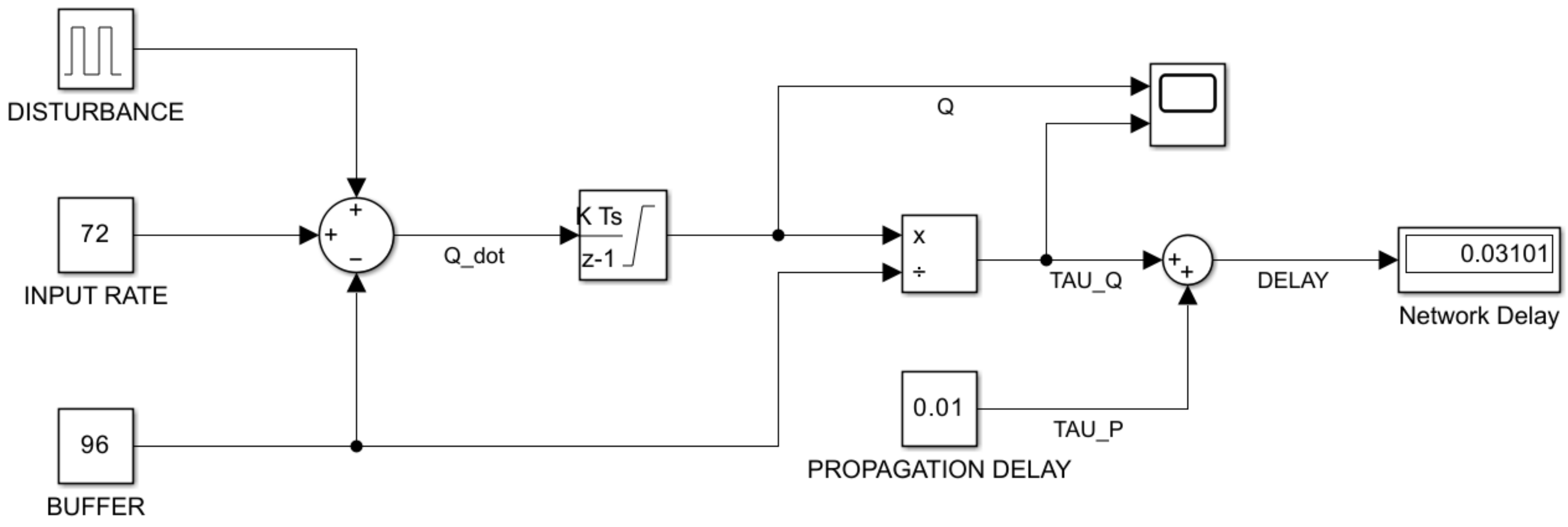


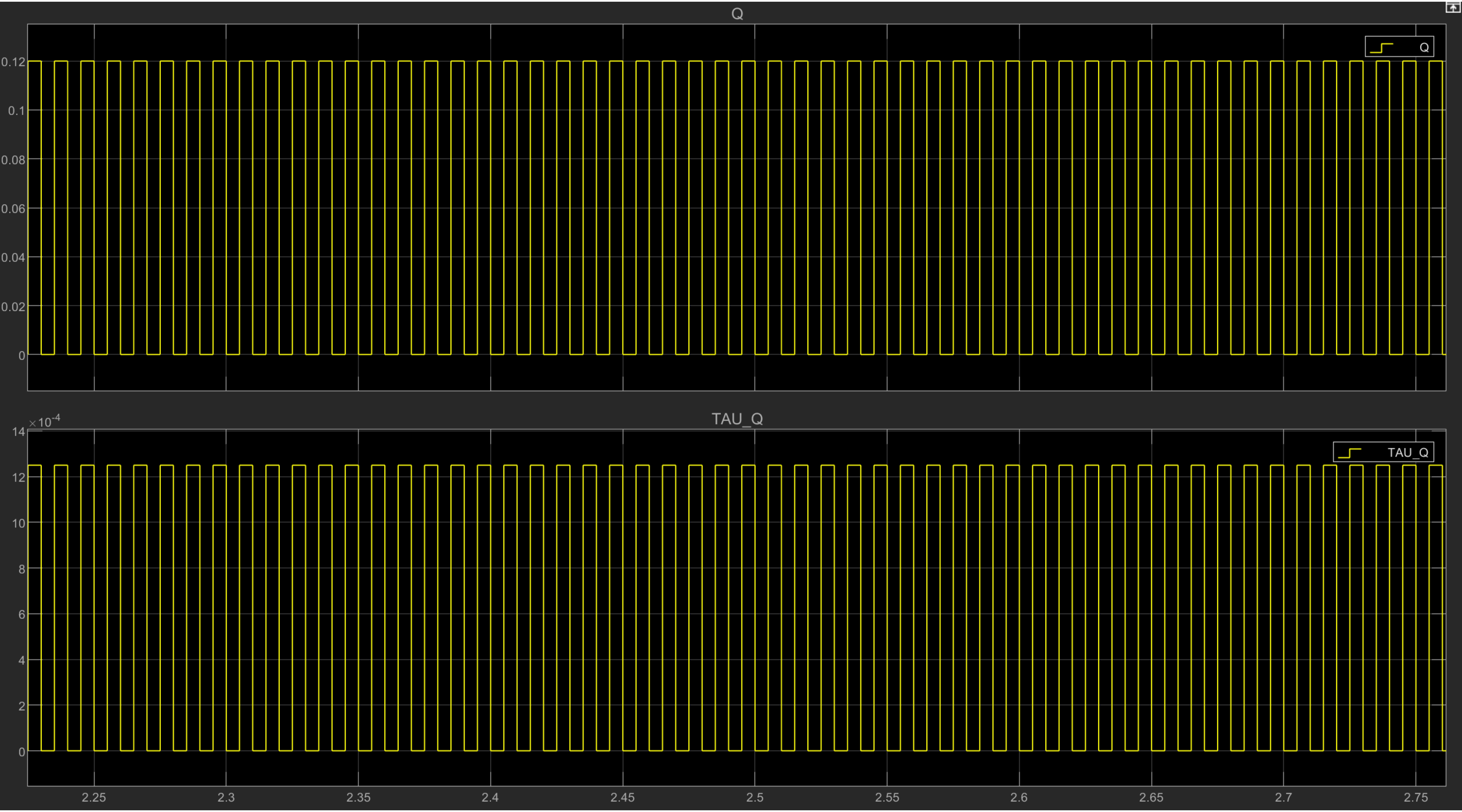
Network Layer Simulation (2)

$$\dot{q} = \mathrm{r} - \mathrm{c}$$

$$\tau_q = \frac{q}{c}$$

- Possible disturbances occuring on the bus are simulated.



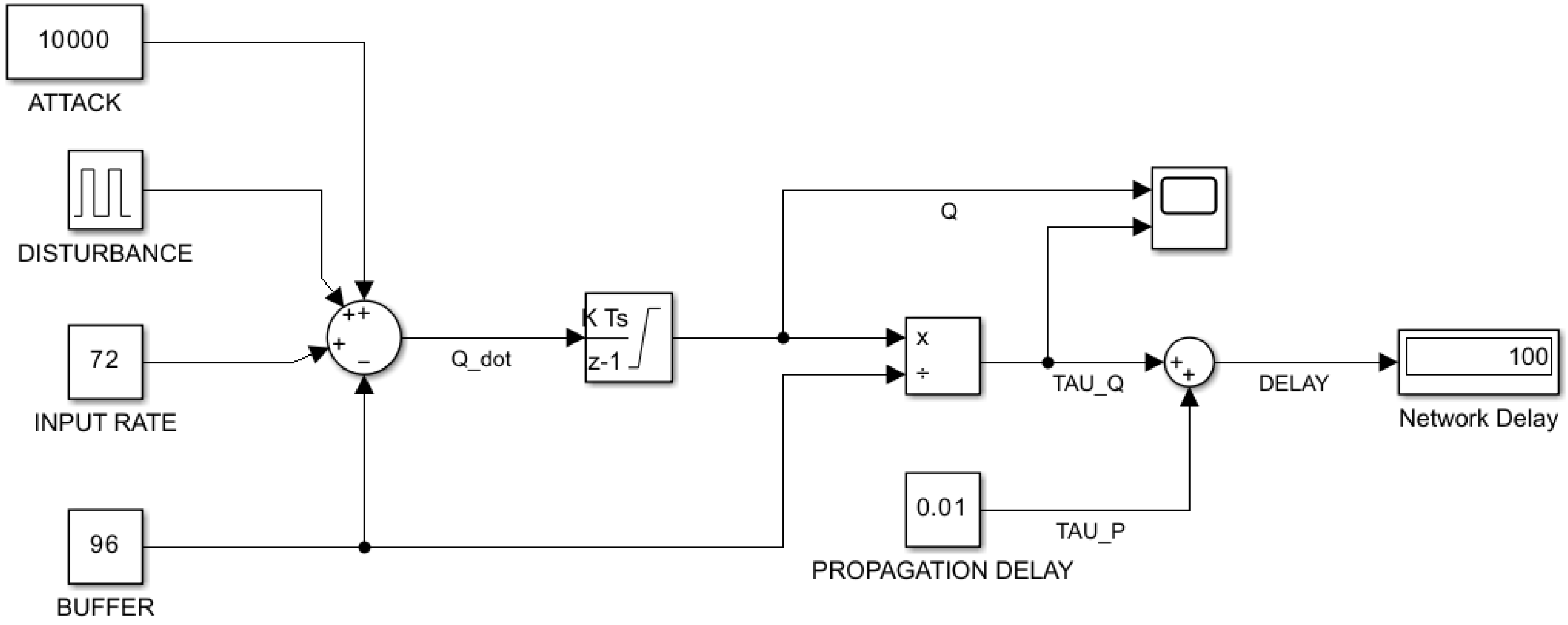

Q
Q
0.12
0.1
0.08
0.06
0.04
0.02
0
TAU_Q
TAU_Q
×10^-4
14
12
10
8
6
4
2
0
2.25
2.3
2.35
2.4
2.45
2.5
2.55
2.6
2.65
2.7
2.75

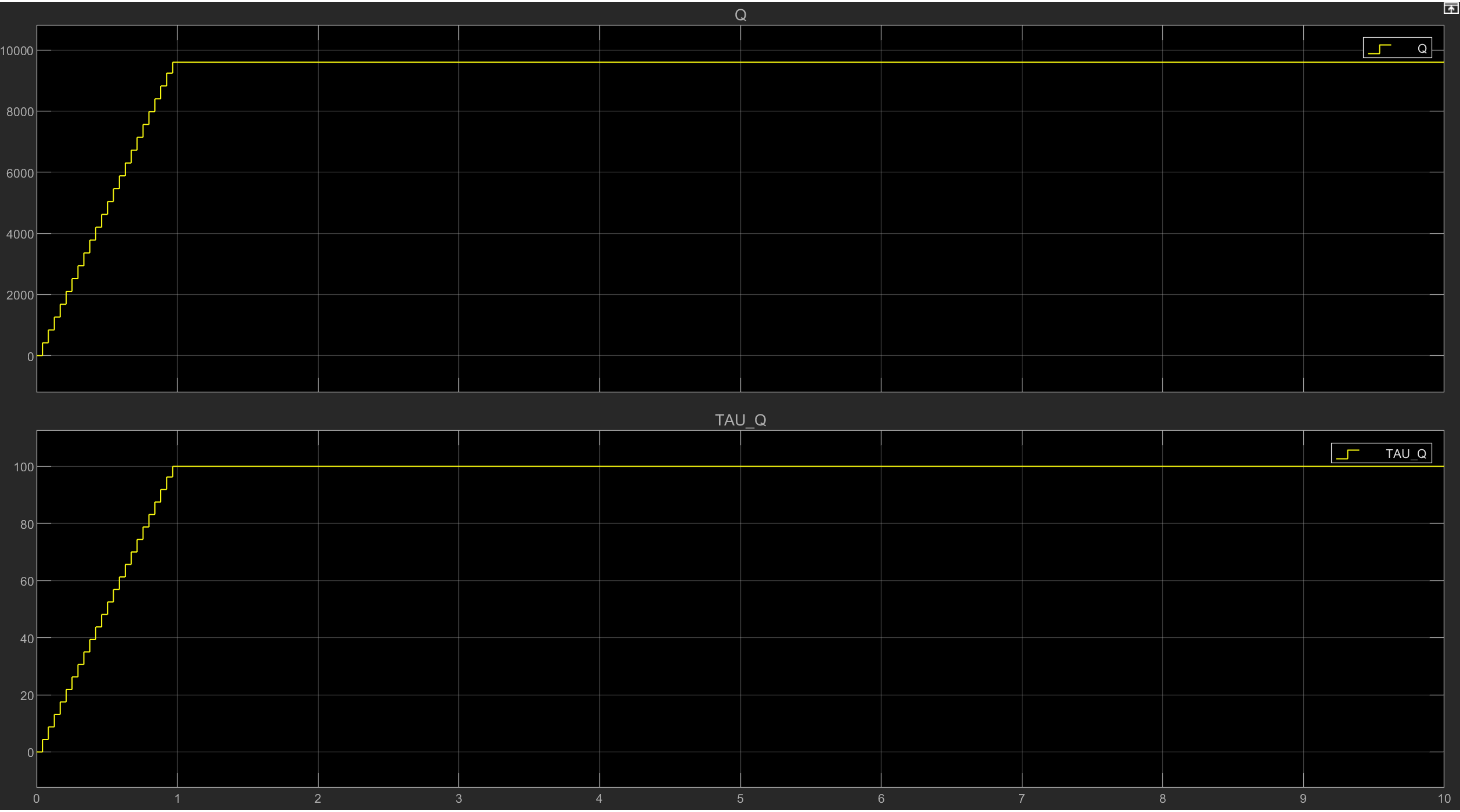
Network Layer Simulation (3)

$$\dot{q} = \mathrm{r} - \mathrm{c}$$

$$\tau_q = \frac{q}{c}$$

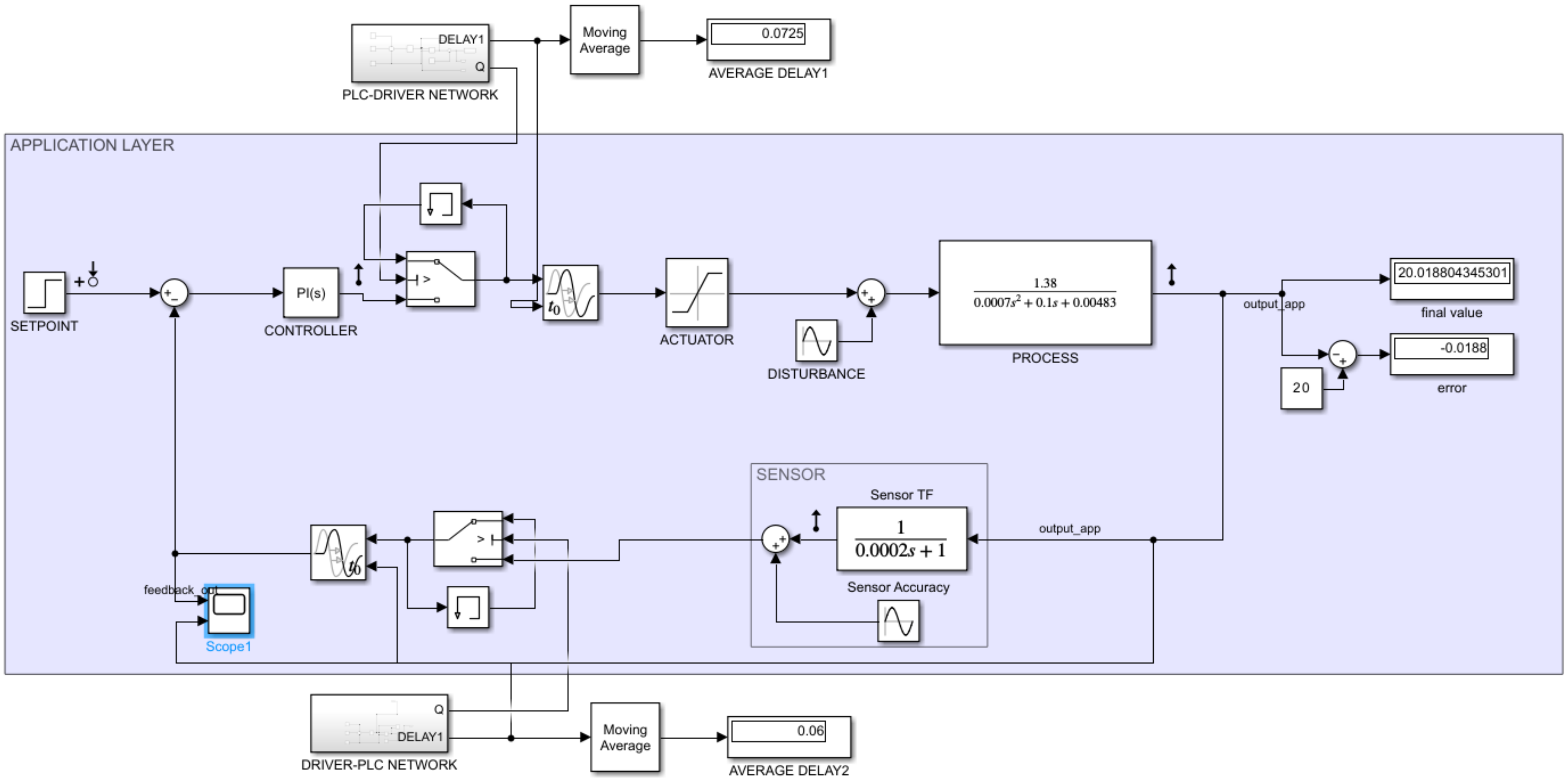
- Network should be prepared in case of a hacker attack, where unnecessary data is sent on the bus to create queues and fill up to bandwidth which results in package loss. Package loss makes the system unstable.

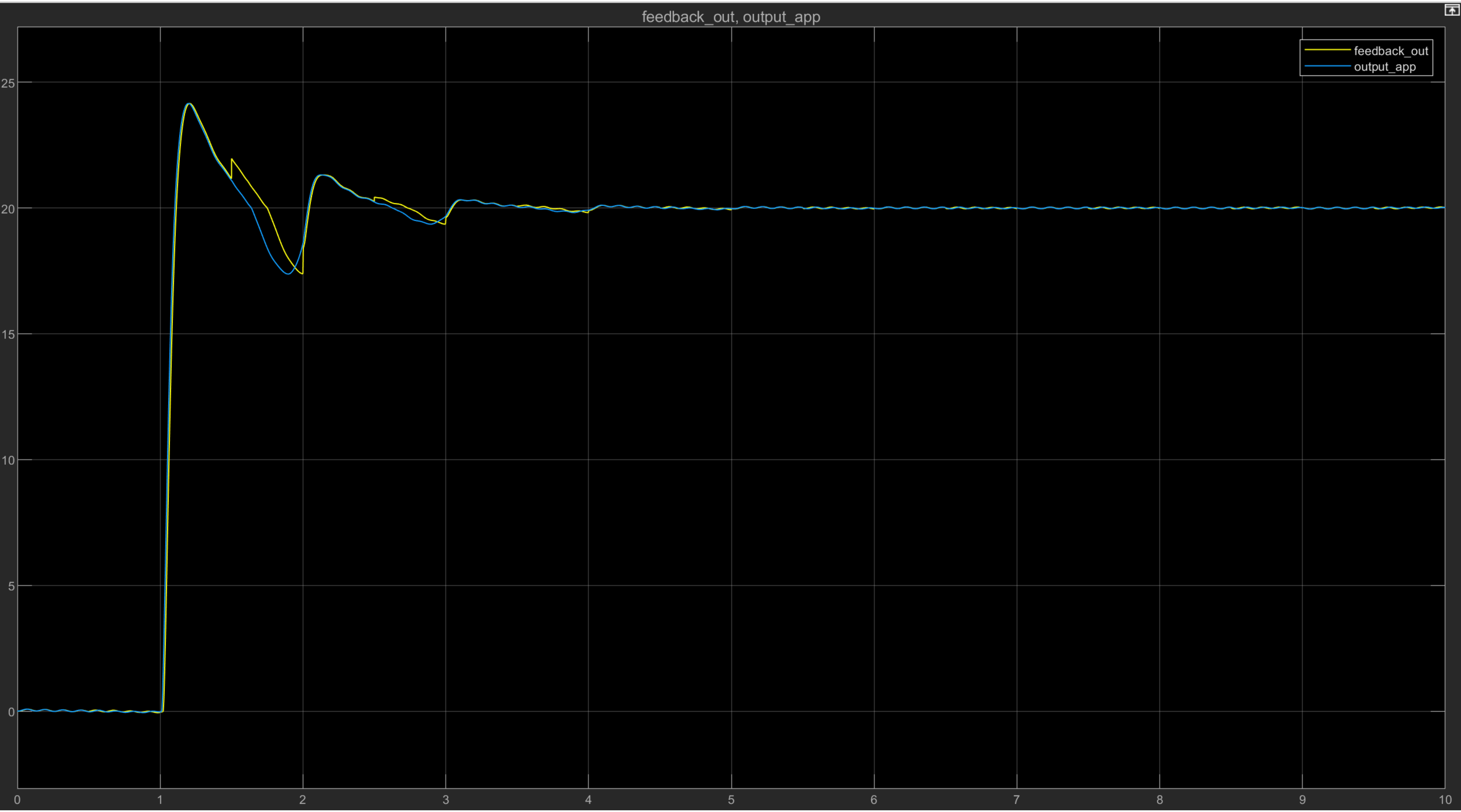




Q
10000
8000
6000
4000
2000
0
Q
TAU_Q
100
80
60
40
20
0
TAU_Q
0
1
2
3
4
5
6
7
8
9
10

Multilayer Simulation (One Process)

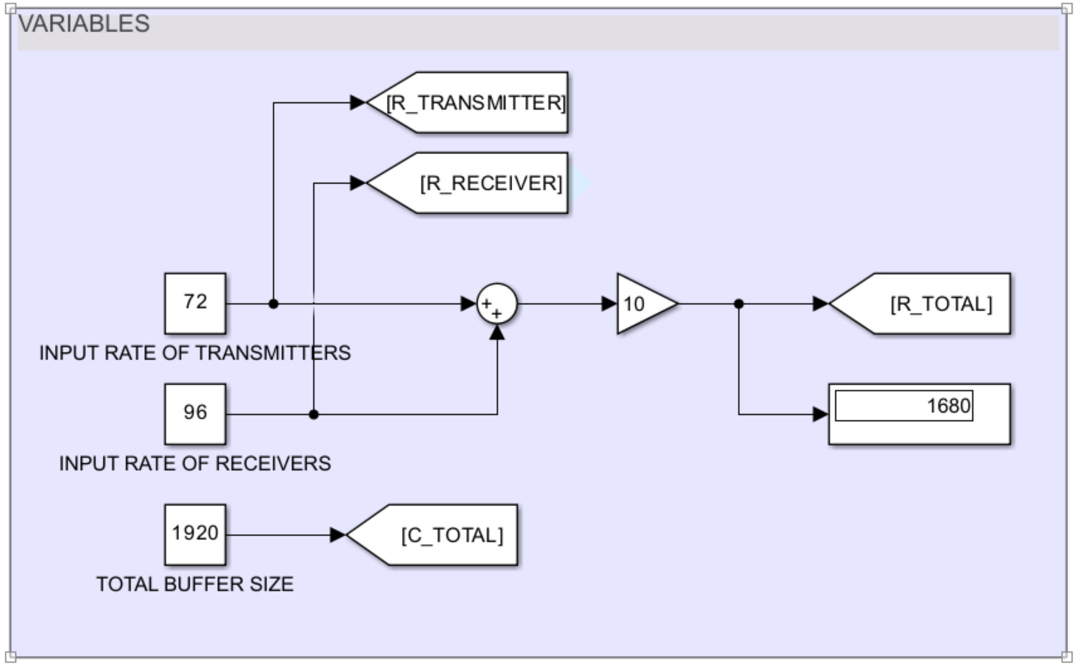




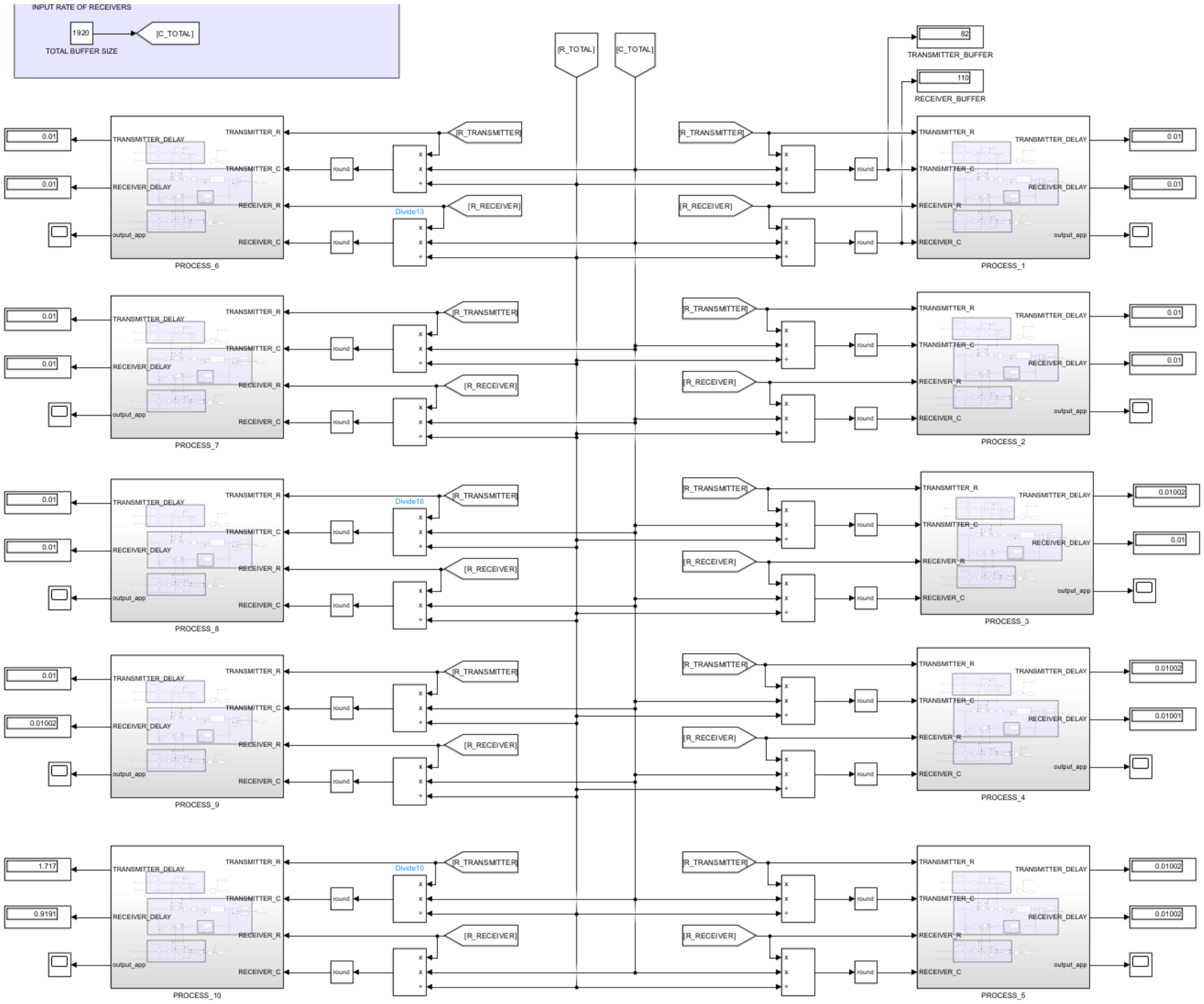
feedback_out, output_app
feedback_out
output_app
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10

Multilayer Simulation (Multiple Process)

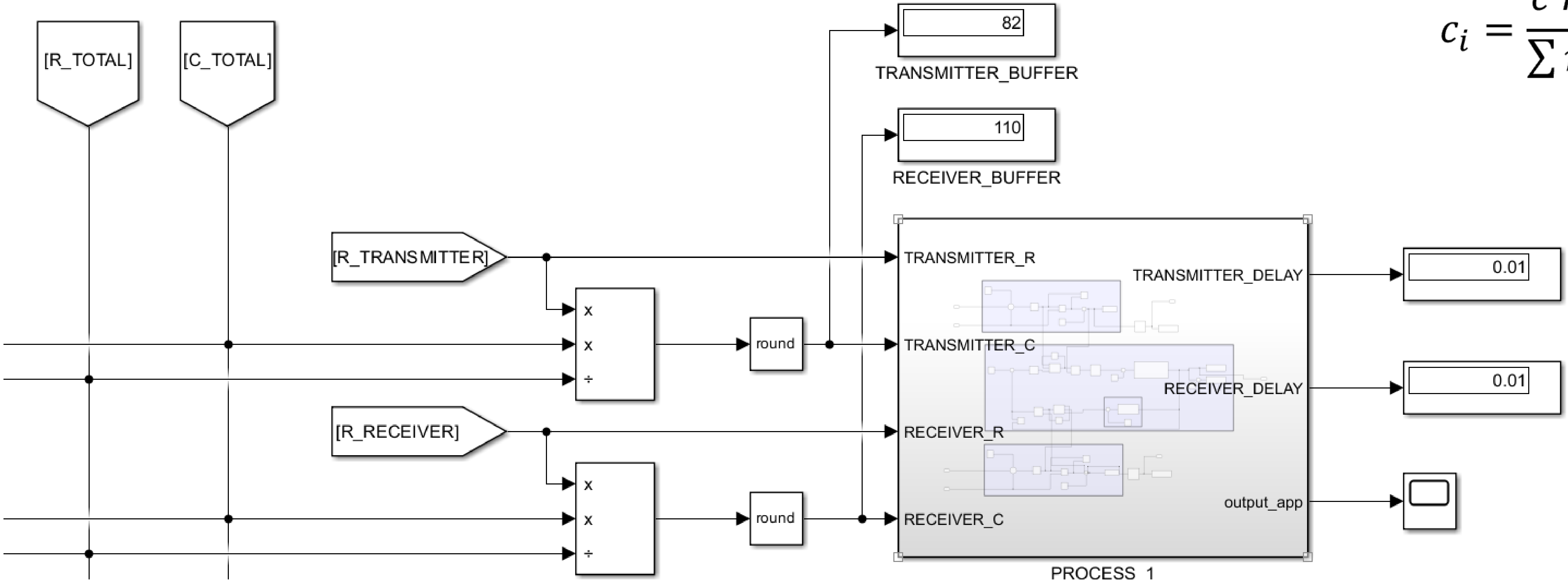
- Total Buffer Size: 1920 kbps
- Input Rate of a Single Process: 72 + 96 = 168 kbps
- With at least 10% safety margin, let's offer 192 kbps buffer for each process.
- In this case we can control 10 different processes simultenously if there is very low disturbance on network. If we want a more reliable communication, we need to lower the no. of processes in order to increase buffer size.



$$c_i = \frac{c\, r_i}{\sum r_j}$$



$$c_i = \frac{c\, r_i}{\sum r_j}$$



$$c_i = \frac{c\, r_i}{\sum r_j}$$