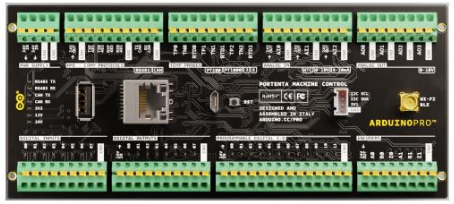
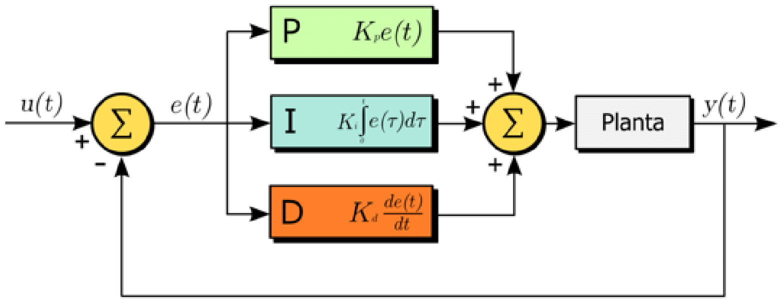
Training Course: Embedded Control System





Arduino with PID Controller

Suranaree University of Technology

Thanasak Wanglomklang| 2020

Agenda

- Open loop Control using Arduino
- Close loop Control using Arduino
- Practices

Open loop Control using Arduino

Basics Control Flow

Command

Arduino

Amplifier

Actuator

0 – 100 %

PWM (0 – 255 → 0-100%)

Digital (Low and High)

Analog (0 – 5 Volt)

DC Electrical

or

AC Electrical

Open loop Control using Arduino

Example Control

Control ON/OFF of LED

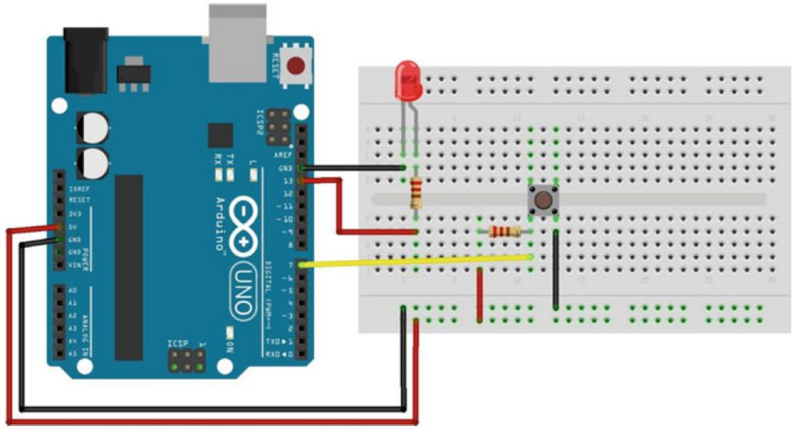
Application Ex. ➡ Control Pneumatic Solenoid valve

Control Lightness of LED

Application Ex. ➡ Control Speed DC motor

Open loop Control using Arduino

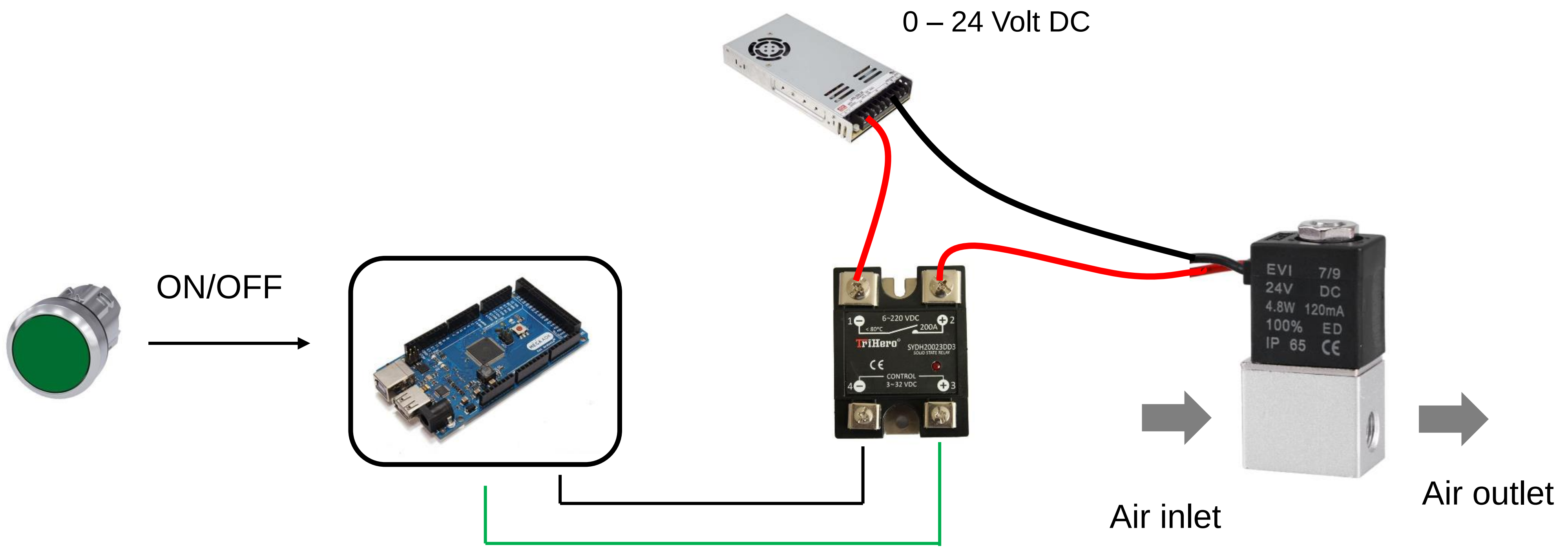
Control ON/OFF of LED



```
int led = 13;   // connect LED to pin 13
int pin = 7;    // connect pushbutton to pin 7
int value = 0; // variable to store the read value

void setup() {
  pinMode(led, OUTPUT);   // set pin 13 as output
  pinMode(pin, INPUT);    // set pin 7 as input
}
void loop() {
  value = digitalRead(pin); // set value equal to the pin 7 input
  digitalWrite(led, value); // set LED to the pushbutton value
}

```

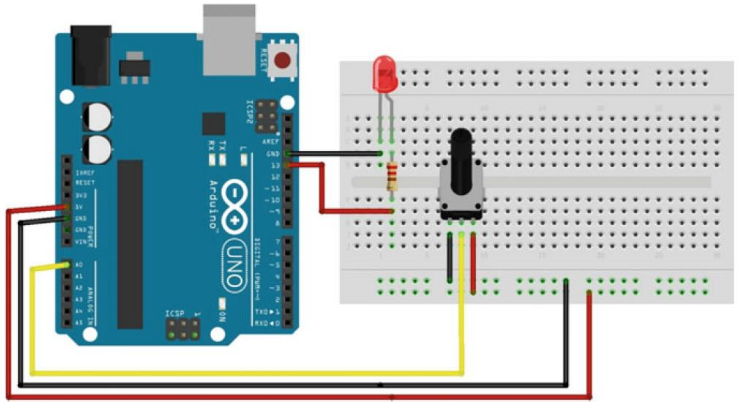
Open loop Control using Arduino

Control Pneumatic Solenoid valve



Open loop Control using Arduino

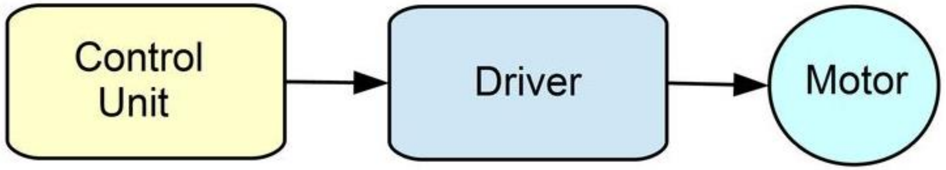
Control Lightness of LED

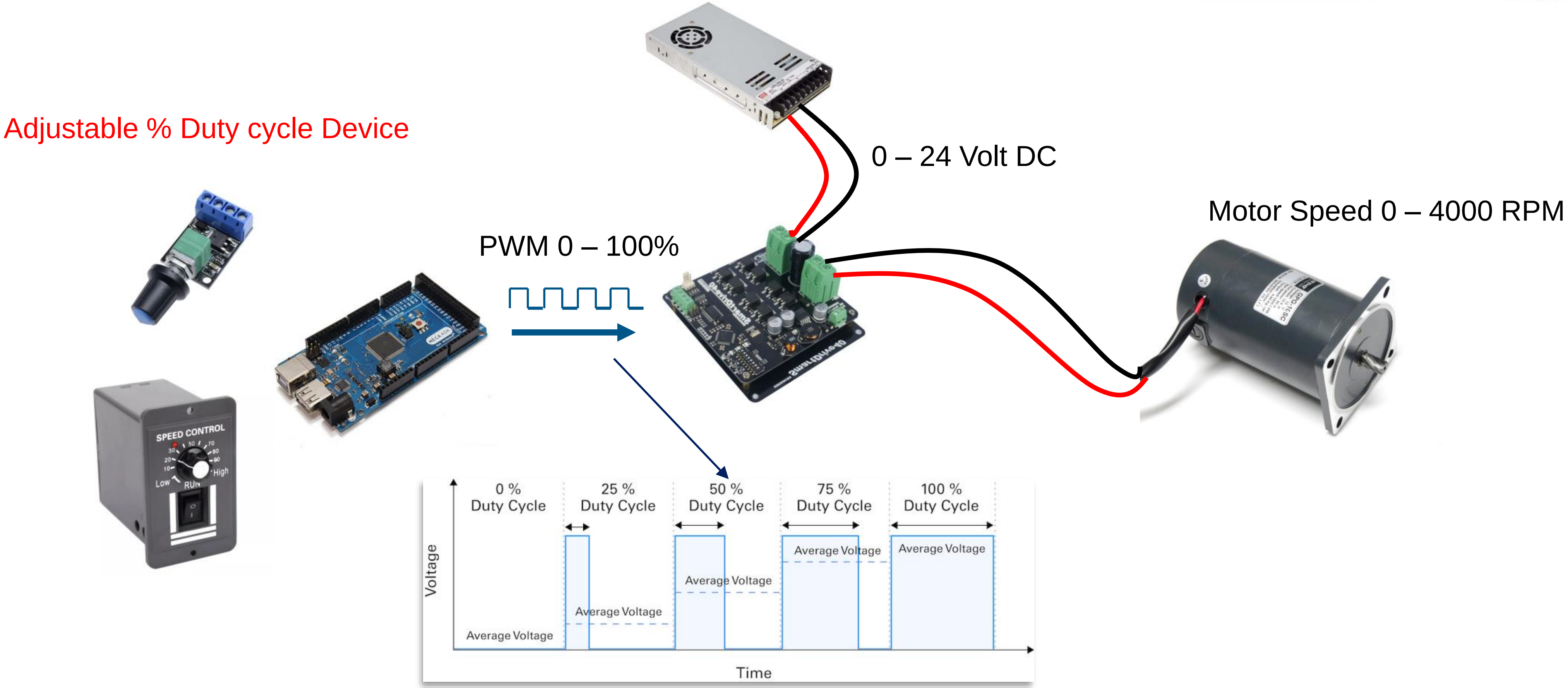


```
int led = 13;   // connect LED to pin 13
int pin = 0;    // potentiometer on analogy pin 0
int value = 0; // variable to store the read value

void setup() {
}
void loop() {
  value = analogRead(pin); // set value equal to the pin 0’s input
  value /= 4;                 // converts 0-1023 to 0-255
  analogWrite(led, value); // output PWM signal to LED
}

```

Open loop Control using Arduino

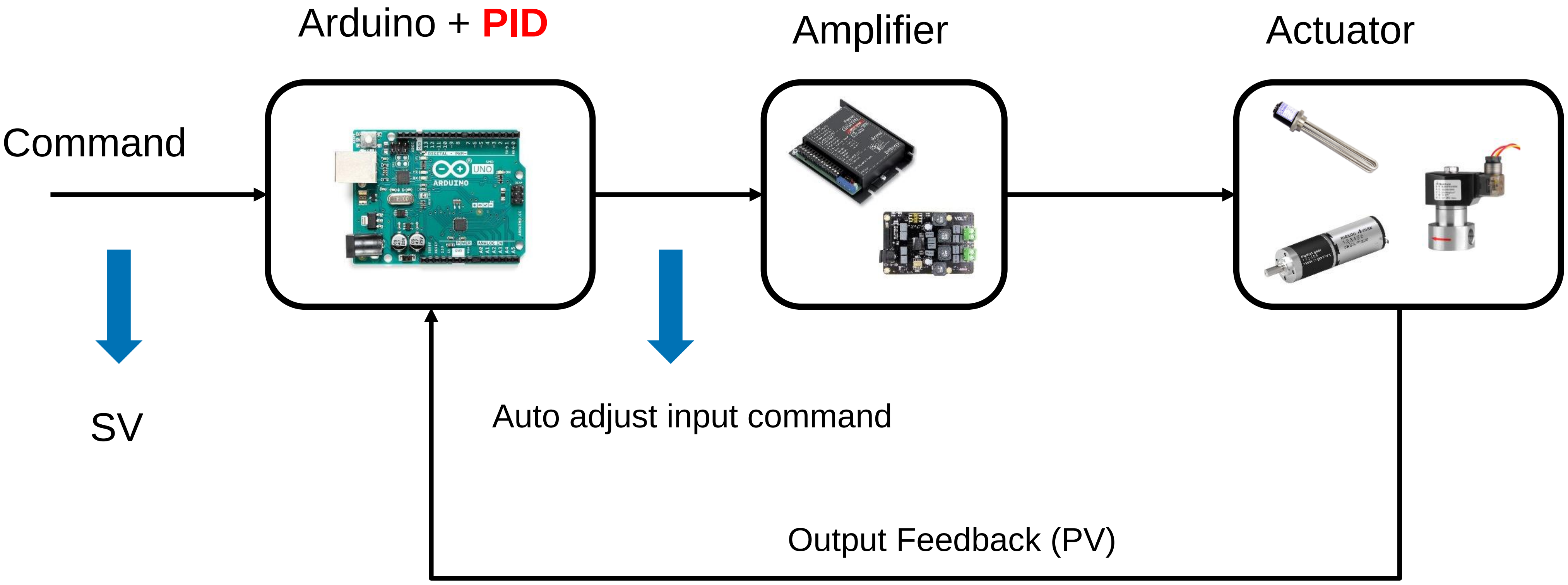
Control Speed DC motor





Close loop Control using Arduino

Basics Control Flow



Close loop Control using Arduino

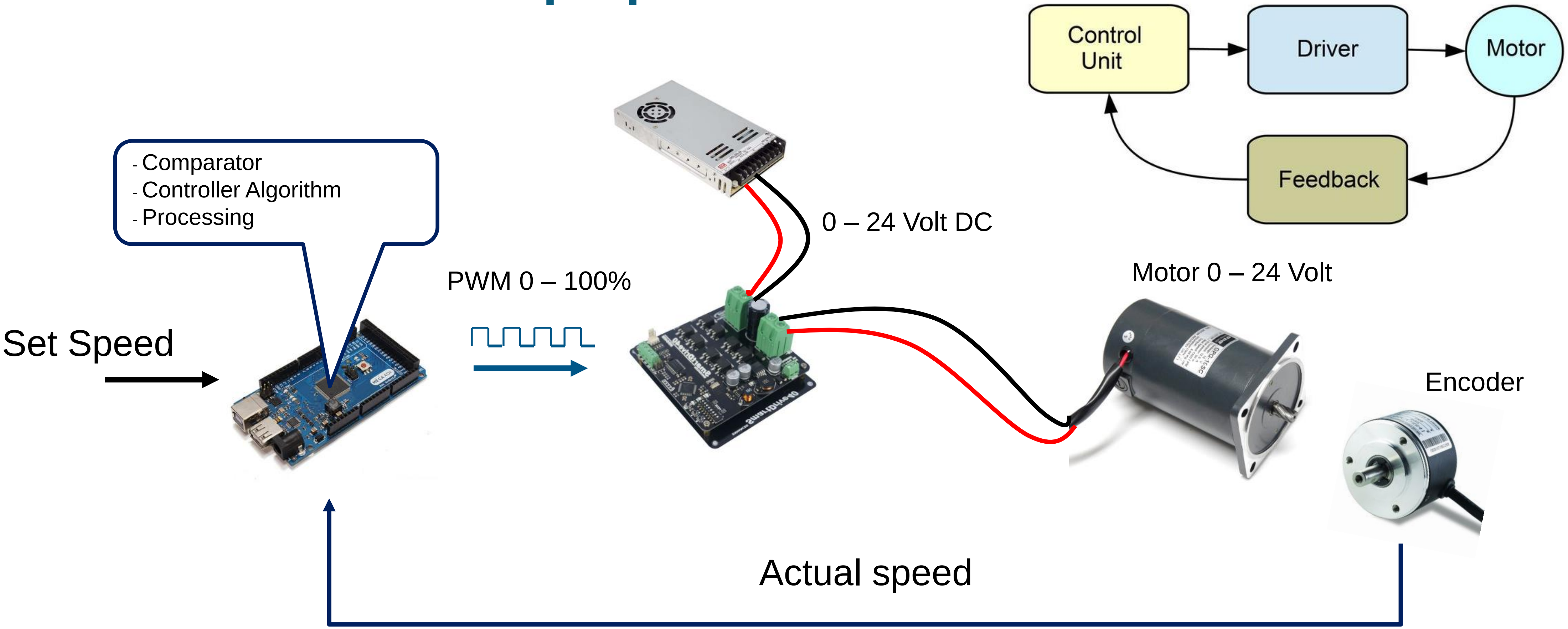
DC motor Close loop Speed control

Temperature Close loop control

Digital PID Controller

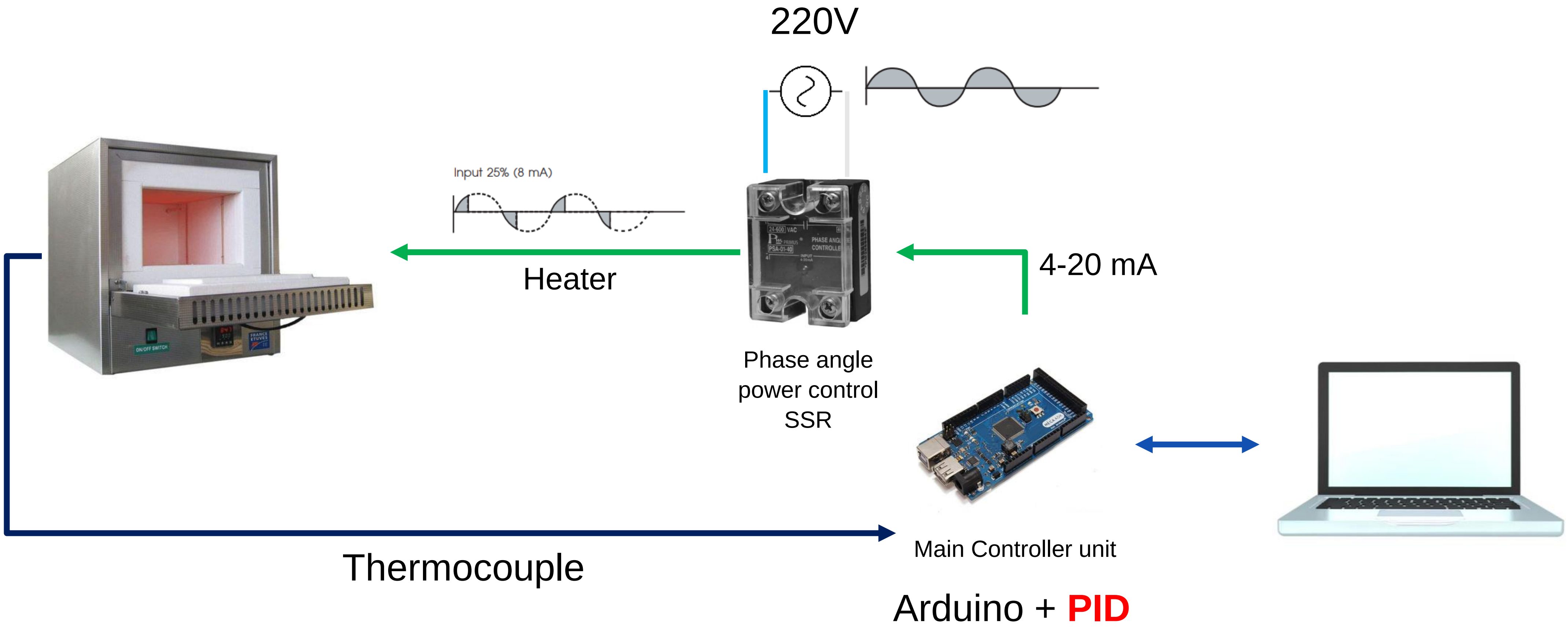
Close loop Control using Arduino

DC motor Close loop Speed control



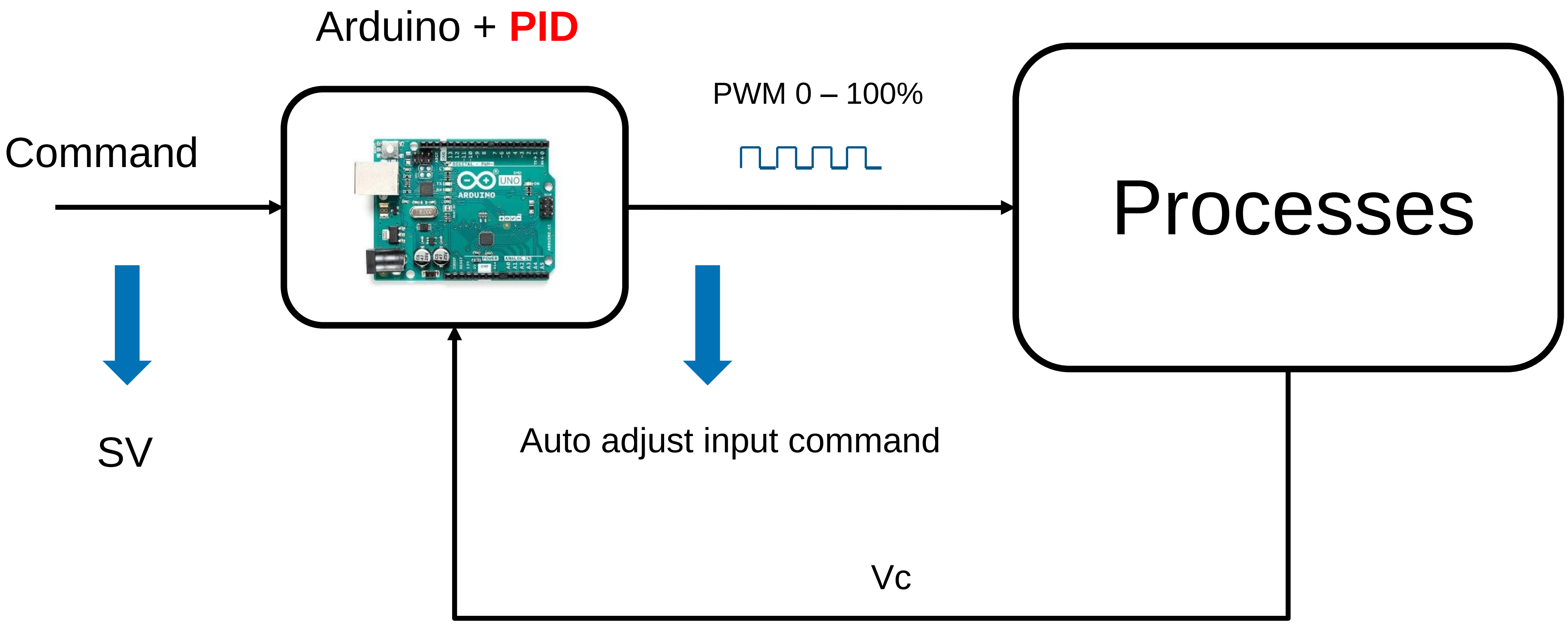
Close loop Control using Arduino

Temperature Close loop control



Close loop Control using Arduino

Digital PID Controller



Close loop Control using Arduino

Digital PID Controller

```
double sensed_output, control_signal;
double setpoint;
double Kp; //proportional gain
double Ki; //integral gain
double Kd; //derivative gain
int T; //sample time in milliseconds (ms)
unsigned long last_time;
double total_error, last_error;
int max_control;
int min_control;

void setup(){

}

void loop(){

 PID_Control(); //calls the PID function every T interval and outputs a control signal

}
```

Close loop Control using Arduino

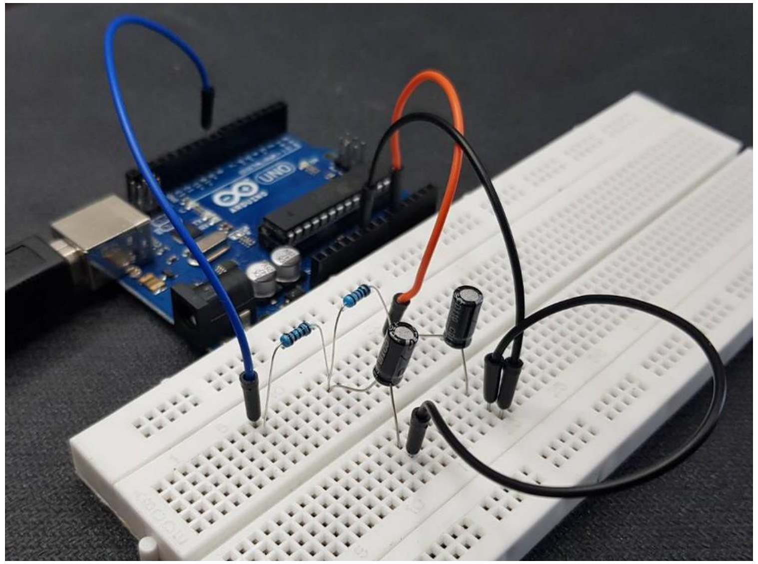
Digital PID Controller

```
void PID_Control(){

  unsigned long current_time = millis(); //returns the number of milliseconds passed since the Arduino started running the program

  int delta_time = current_time - last_time; //delta time interval

  if (delta_time >= T){

    double error = setpoint - sensed_output;

    total_error += error; //accumalates the error - integral term
    if (total_error >= max_control) total_error = max_control;
    else if (total_error <= min_control) total_error = min_control;

    double delta_error = error - last_error; //difference of error for derivative term

    control_signal = Kp*error + (Ki*T)*total_error + (Kd/T)*delta_error; //PID control compute
    if (control_signal >= max_control) control_signal = max_control;
    else if (control_signal <= min_control) control_signal = min_control;

    last_error = error;
    last_time = current_time;
    }
}
```

Practices

Open loop 2nd order RC circuit

Example **Voltage response control of RC circuit**



$$R_1 = R_2 \simeq 10\ k\Omega$$

$$C_1 = C_2 \simeq 100\ \mu F$$

$$\frac{V_C(s)}{V_{in}(s)} = \frac{1}{(R_1R_2C_1C_2)s^2 + (R_1C_1 + R_2C_2)s + 1}$$

$$\frac{V_C(s)}{V_{in}(s)} = \frac{1}{s^2 + 2s + 1}$$

Practices

Open loop 2nd order RC circuit

System analysis using python

```
from control.matlab import *

G = tf([1],[1,2,1])
print(G)
```

>>>

```
      1
-------------
s^2 + 2 s + 1
```

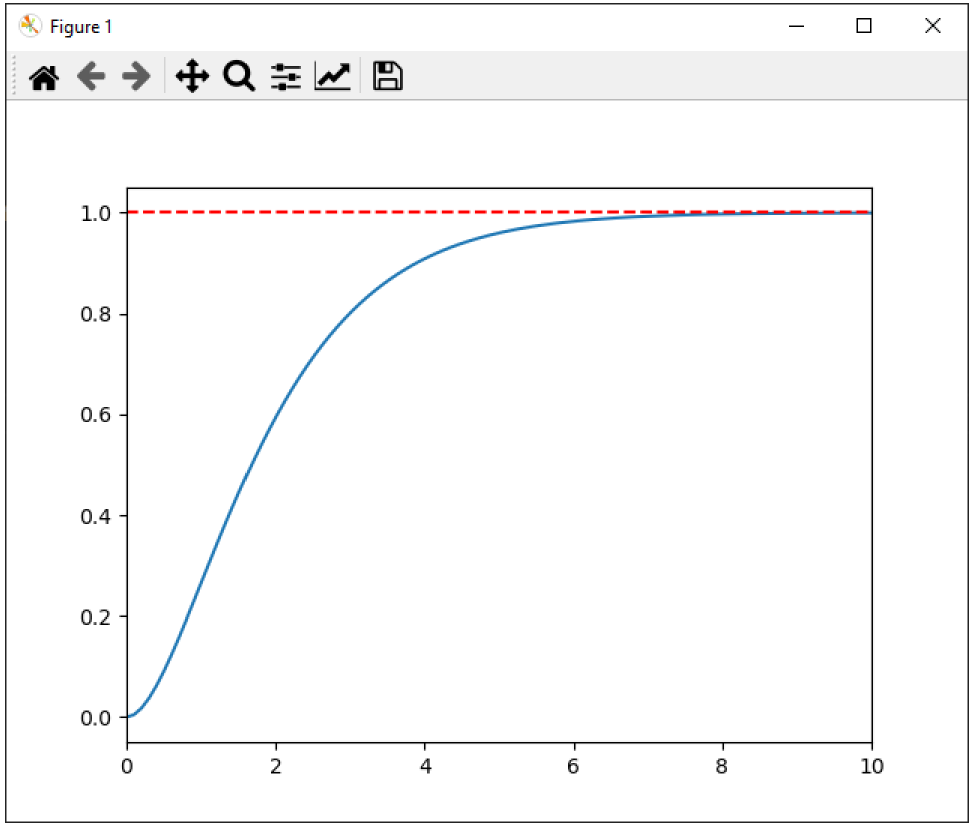
Practices

Open loop 2nd order RC circuit

System analysis using python

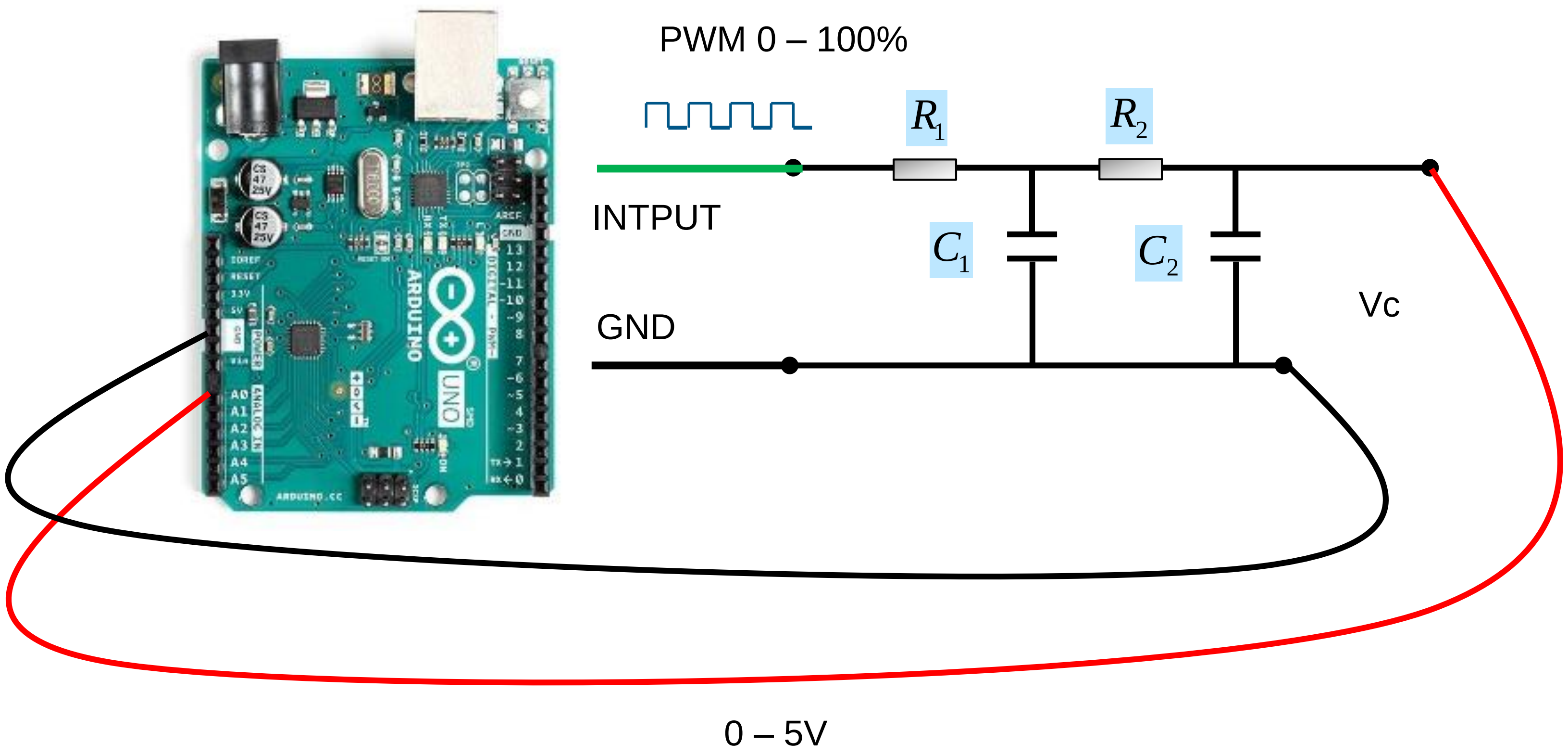
Step input

```
from control.matlab import *
import matplotlib.pyplot as plt
import numpy as np

G = tf([1],[1,2,1])
print(G)

val, time = step(G,10)
plt.plot(time,val)
plt.plot(time,np.ones(len(time)), '--r')
plt.xlim(0,10)
plt.show()
```



Practices

<u>Example</u> Voltage response control of RC circuit

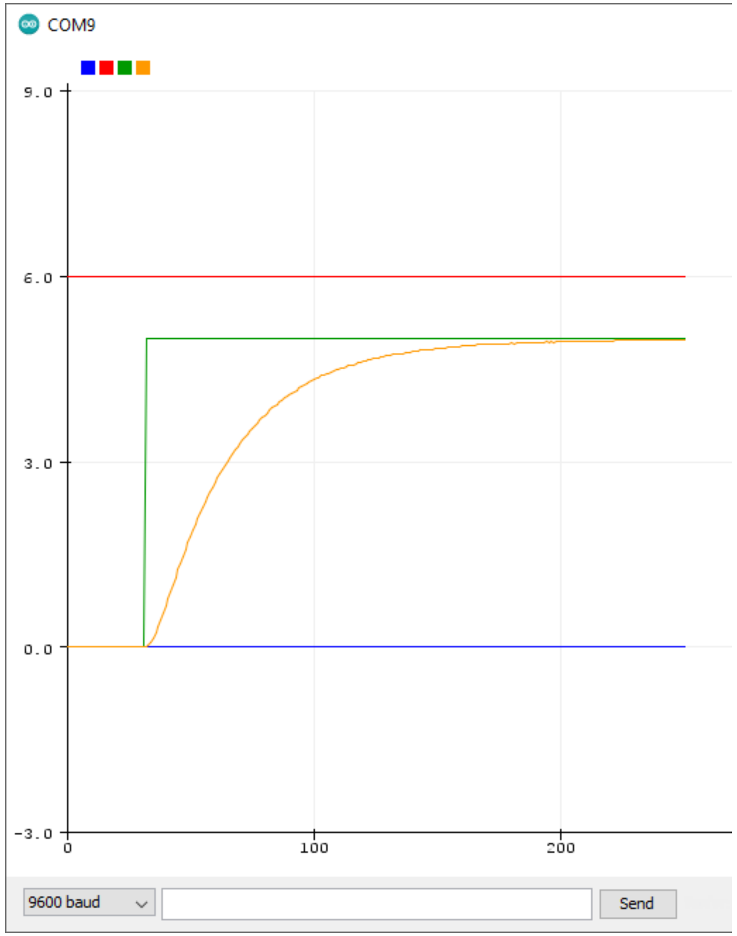


Practices

Time response of Step input

```
char get_serial;
String sum_serial;
float Vin;
float input;
float Vc;
void setup() {
  Serial.begin(9600);
  pinMode(9,OUTPUT);
}

void loop() {
  Vc = analogRead(A0)*5/1023.0;
  input = map(Vin,0,5,0,255);
  analogWrite(9,input);
  Serial.print(0);
  Serial.print("\t");
  Serial.print(6);
  Serial.print("\t");
  Serial.print(Vin);
  Serial.print("\t");
  Serial.println(Vc);
  serialread();
  delay(10);
}
```

```
void serialread() {
  while (Serial.available()) {
    get_serial = Serial.read();

    if (get_serial == 'A') {
      Vin = sum_serial.toFloat();
      sum_serial = "";
      break;
    }
    sum_serial += get_serial;
  }

}
```



Practices

Time response of Step input form python

#Vin#Vc#

Practices

Time response of Step input form python

```
char get_serial;
String sum_serial;
float Vin;
float input;
float Vc;
String sendtoPC;
void setup() {
  Serial.begin(9600);
  pinMode(9,OUTPUT);
}

void loop() {
  Vc = analogRead(A0)*5/1023.0;
  input = map(Vin,0,5,0,255);
  analogWrite(9,input);
  sendtoPC = "#" + String(Vin,2) + "#" + String(Vc,2) + "#";
  Serial.println(sendtoPC);
  serialread();
  //delay(10);
}
```

```
void serialread() {
  while (Serial.available()) {
    get_serial = Serial.read();

    if (get_serial == 'A') {
      Vin = sum_serial.toFloat();
      sum_serial = "";
      break;
    }
    sum_serial += get_serial;
  }
}
```

Practices

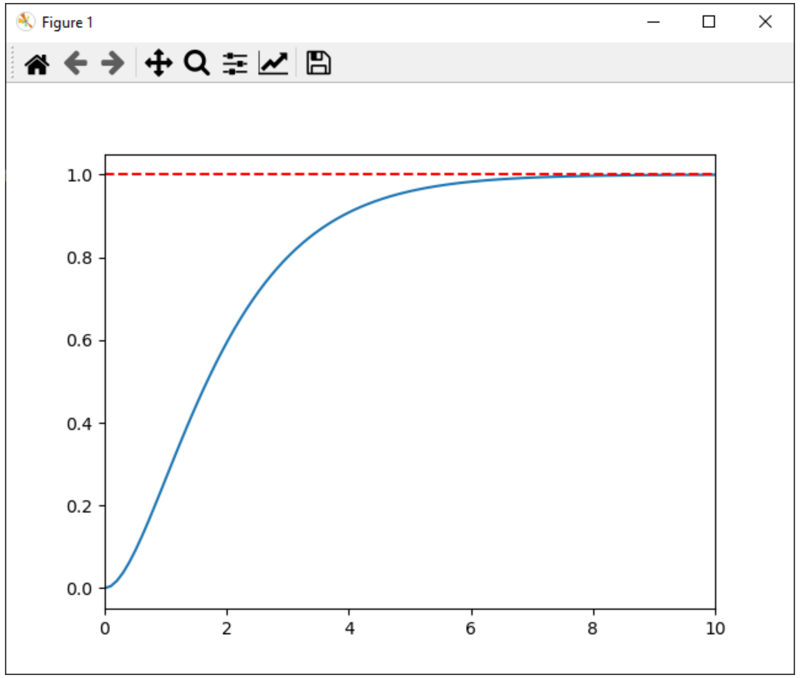
Time response of Step input form python

```python
import serial
import time
import matplotlib.pyplot as plt
import threading
import serial.tools.list_ports

# เริ่มต้นเชื่อมต่อ
port  = serial.tools.list_ports.comports()
comport = str(port[0])
ser = serial.Serial(comport[0:4], baudrate = '9600')
ser.flushInput()
ser.reset_input_buffer()

## อ่านข้อมูล
data = ""
def get_serial_data():
    global data, input_val, output_val
    while True:
        try:
            data = ser.readline().decode('utf-8', errors='replace')
            data = data.split("#")
            if len(data)==5:
                input_val = float(data[1])
                output_val = float(data[2])
        except KeyboardInterrupt:
            ser.close()
            pass
        except Exception as e:
            ser.close()
            pass
t1 = threading.Thread(target=get_serial_data)
t1.start()
```

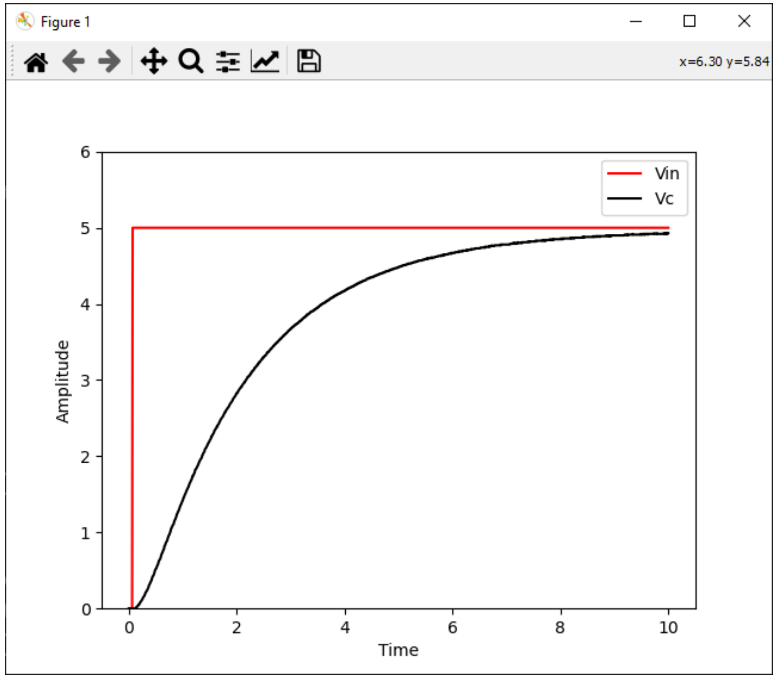
```python
i = 0
datat = []
datainput = []
dataoutput = []

sendinput = input('Vin = ')
ser.write(sendinput.encode())
while i<=10:
    datat.append(i)
    datainput.append(input_val)
    dataoutput.append(output_val)
    print('{:.2f}'.format(i), input_val, output_val)
    time.sleep(0.01)
    i+=0.01

plt.plot(datat,datainput, 'r')
plt.plot(datat,dataoutput, 'black')
plt.ylim(0,6)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend(['Vin','Vc'])
plt.show()

ser.close()
```

Practices

Time response of Step input form python



Simulation



Experiment

Practices

Time response of Step input form python

```
import serial
import time
import matplotlib.pyplot as plt
import threading
import serial.tools.list_ports
import threading

# เริ่มต้นเชื่อมต่อ
port  = serial.tools.list_ports.comports()
comport = str(port[0])
ser = serial.Serial(comport[0:4], baudrate = '9600')
ser.flushInput()
ser.reset_input_buffer()
```

Practices

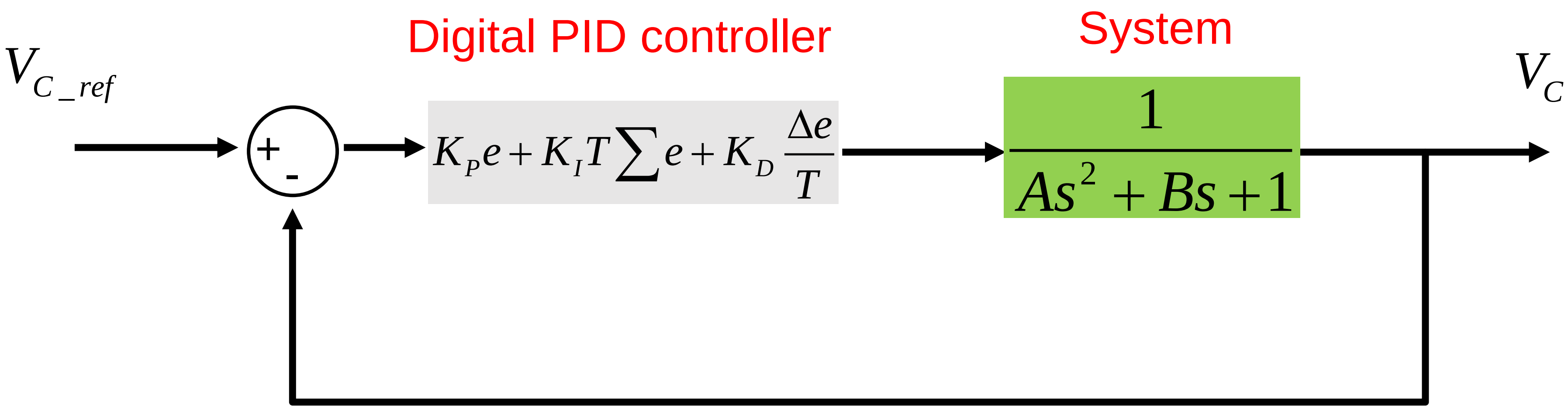
Time response of Step input form python

```python
## อ่านข้อมูล
data = ""
def get_serial_data():
    global data, input_val, output_val
    while True:
        try:
            data = ser.readline().decode('utf-8', errors='replace')
            data = data.split("#")

            if len(data)==4:
                input_val = float(data[1])
                output_val = float(data[2])

        except KeyboardInterrupt:
            ser.close()
            pass
        except Exception as e:
            ser.close()
            pass

t1 = threading.Thread(target=get_serial_data)
t1.start()
```

```python
i = 0
datat = []
datainput = []
dataoutput = []

sendinput = input('Vin = ')
ser.write(sendinput.encode())
while i<=10:
    datat.append(i)
    datainput.append(input_val)
    dataoutput.append(output_val)
    print('{:.2f}'.format(i), input_val, output_val)
    time.sleep(0.01)
    i+=0.01

plt.plot(datat,datainput, 'r')
plt.plot(datat,dataoutput, 'black')
plt.ylim(0,6)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend(['Vin','Vc'])
plt.show()

ser.close()
```

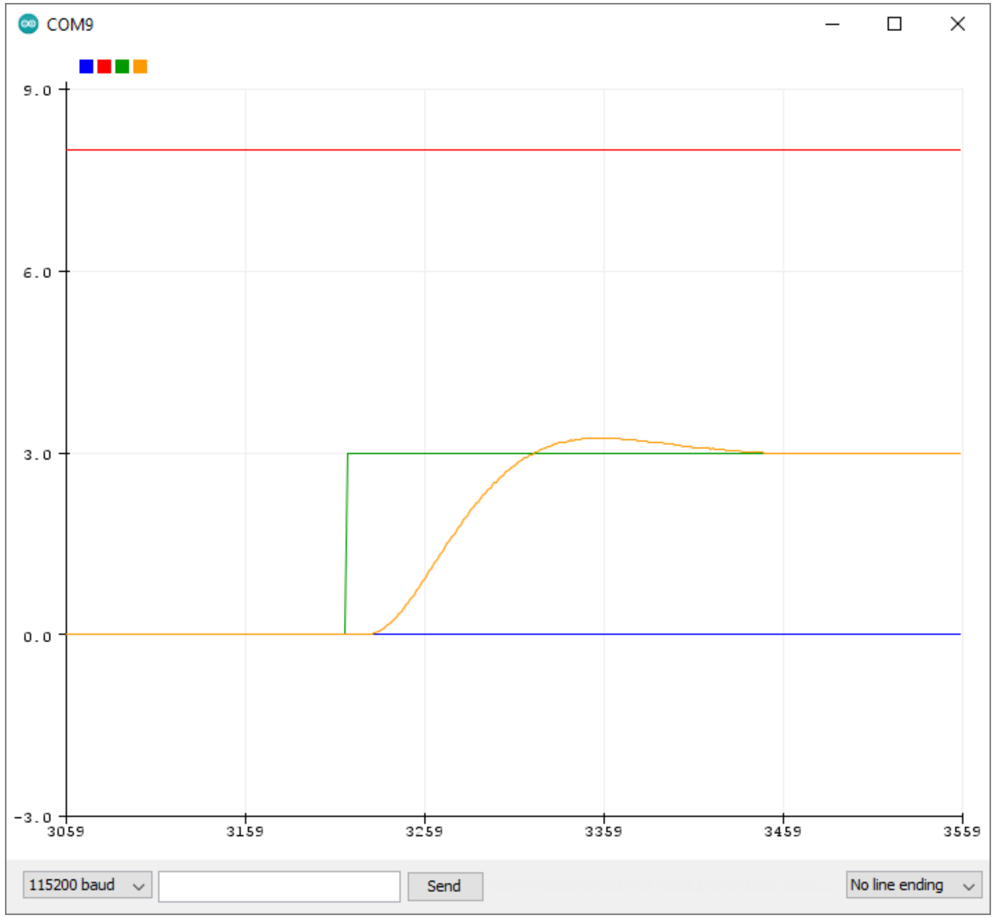
Practices

Close loop 2nd order RC circuit



Practices

Close loop 2nd order RC circuit



Kp = 15
Ki = 15
Kd = 0

Practices

Close loop 2nd order RC circuit

```
double kp = 0.0;
double ki = 0.0;
double kd = 0.0;
double SV, PV;
double control_signal, u;
unsigned long last_time, current_time;
double dT;
double last_error, error, sum_error, dE;
char get_serial;
String sum_serial;
float T = 0.1;
double upbound_control = 255;
double lowbound_control = 0;
String show_val;
```

```
void loop() {
  PV = analogRead(A0) * 5 / 1023.0;
  u = PID_comput();
  analogWrite(9, u);
//
//  show_val = "#" + String(SV,2) + "#" + String(PV,2) + "#" + String(u,2) + "#";
//  Serial.println(show_val);

    Serial.print(0);
    Serial.print("\t");
    Serial.print(8);
    Serial.print("\t");
    Serial.print(SV);
    Serial.print("\t");
    Serial.println(PV);

  serialread();
  delay(T*1000);
}
```

Practices

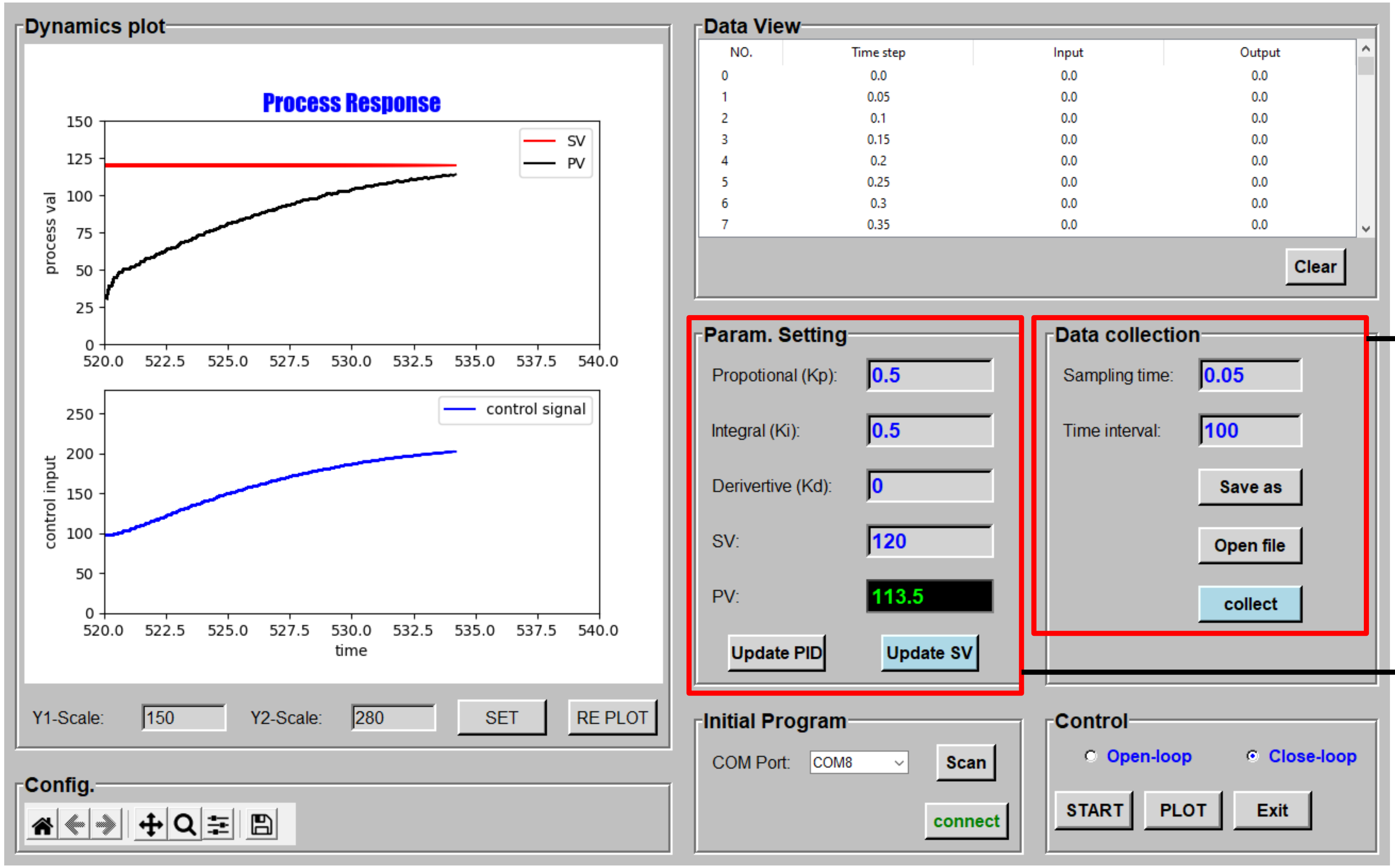
Close loop 2nd order RC circuit

```
double PID_comput() {
  current_time = millis();
  dT = (current_time - last_time) / 1000.0;
  if (dT >= T) {
    error = SV - PV;
    sum_error += error;
    dE = error - last_error;

    control_signal = kp * error + (ki * T * sum_error) + kd * (dE / T);

    if (control_signal >= upbound_control) {
      control_signal = upbound_control;
    }
    if (control_signal < lowbound_control) {
      control_signal = lowbound_control;
    }

    last_error = error;
    last_time = current_time;
  }
  return control_signal;
}
```

```
void serialread() {
  while (Serial.available()) {
    get_serial = Serial.read();

    if (get_serial == 'P') {
      kp = sum_serial.toFloat();
      sum_serial = "";
      break;
    }
    if (get_serial == 'I') {
      ki = sum_serial.toFloat();
      sum_serial = "";
      break;
    }
    if (get_serial == 'D') {
      kd = sum_serial.toFloat();
      sum_serial = "";
      break;
    }
    if (get_serial == 'S') {
      SV = sum_serial.toFloat();
      sum_serial = "";
      break;
    }

    sum_serial += get_serial;
  }

}
```

Practices

Developed software using python



Data logger

Updated parameter to Arduino

EX. 2P2I0D50S

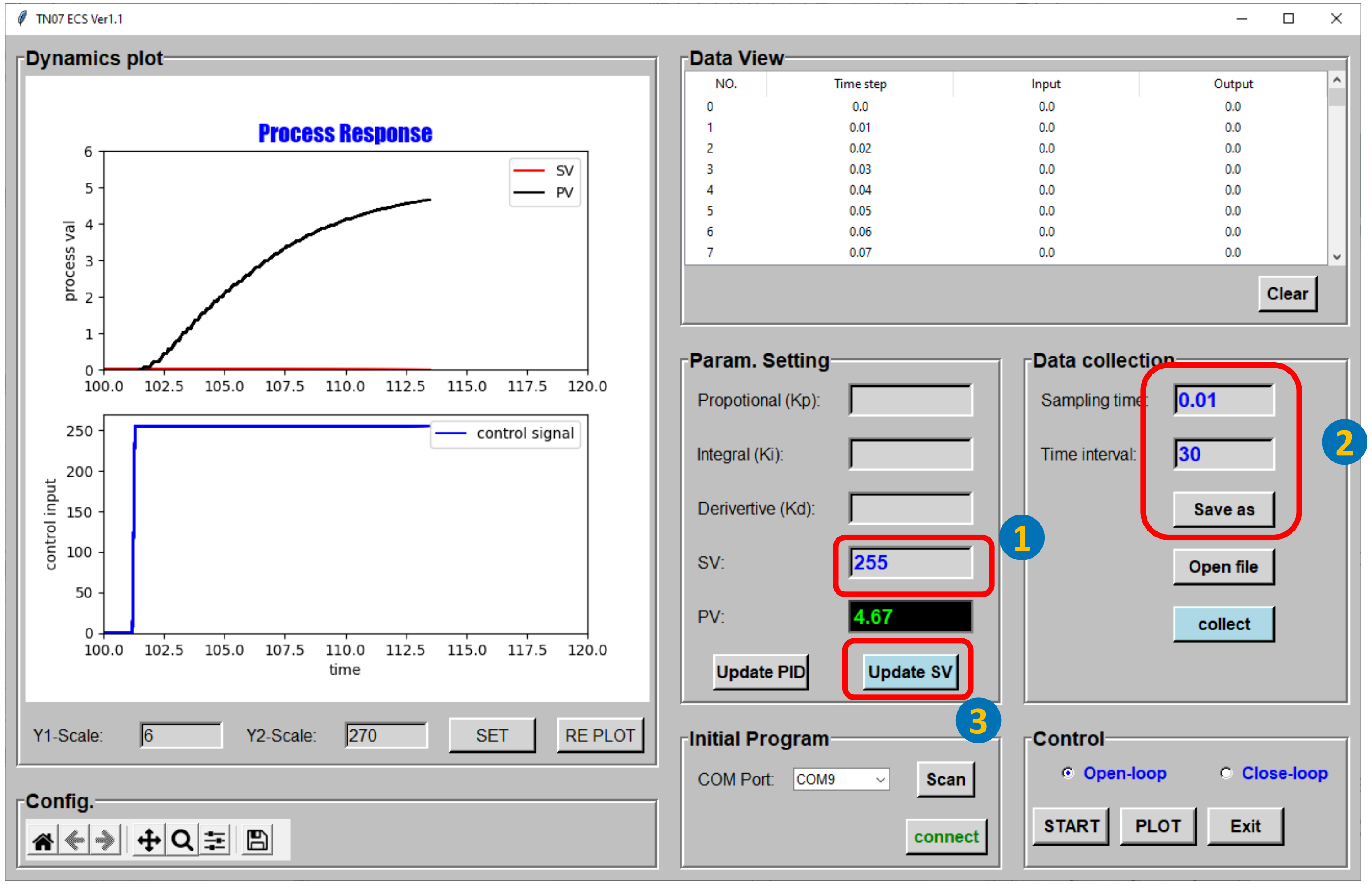
Practices

PID Controller Design Method 1

Process reaction curve

Practices

PID Controller Design Method 1 Process reaction curve



Practices

PID Controller Design Method 1 Process reaction curve

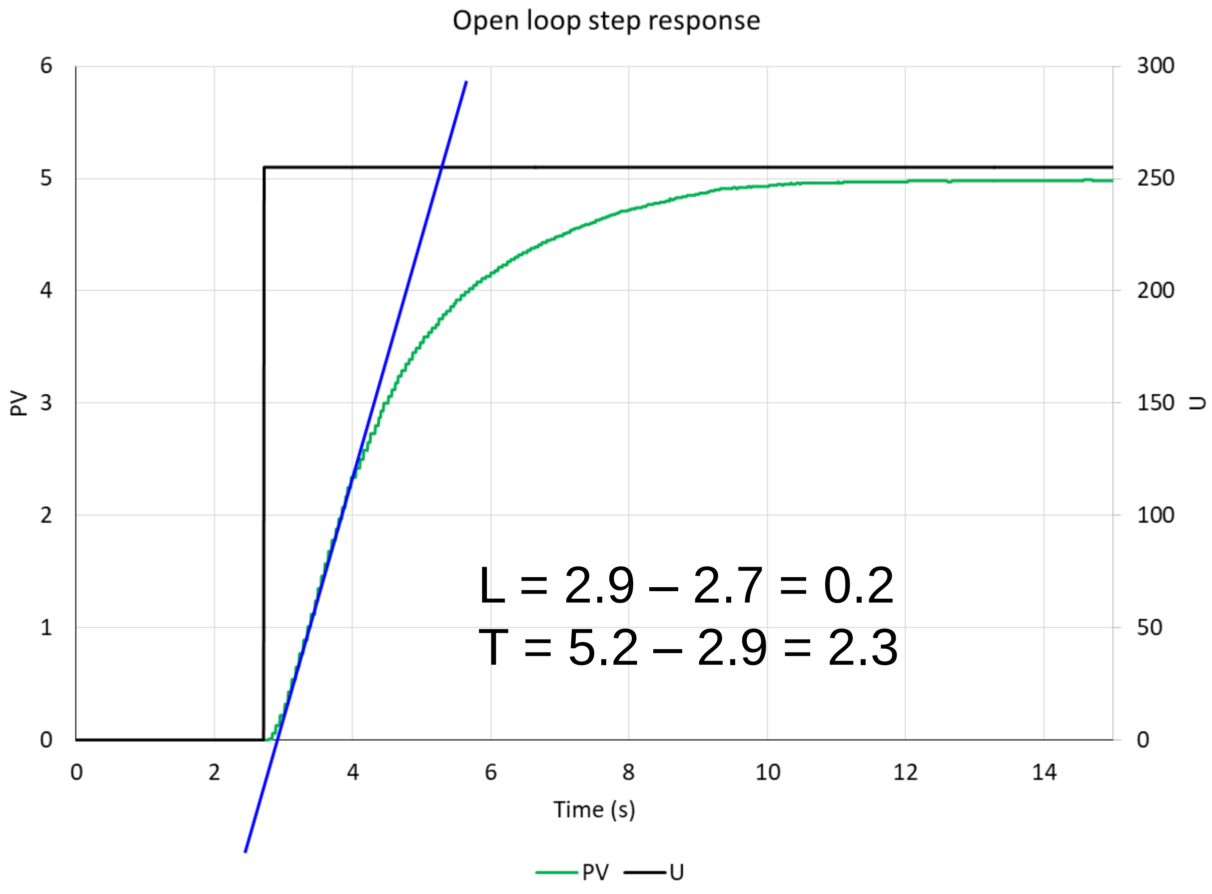


Table 8–1 Ziegler–Nichols Tuning Rule Based on Step Response of Plant (First Method)

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Using PI Controller

$$K_P = 0.9(\frac{2.3}{0.2}) = 10.35$$

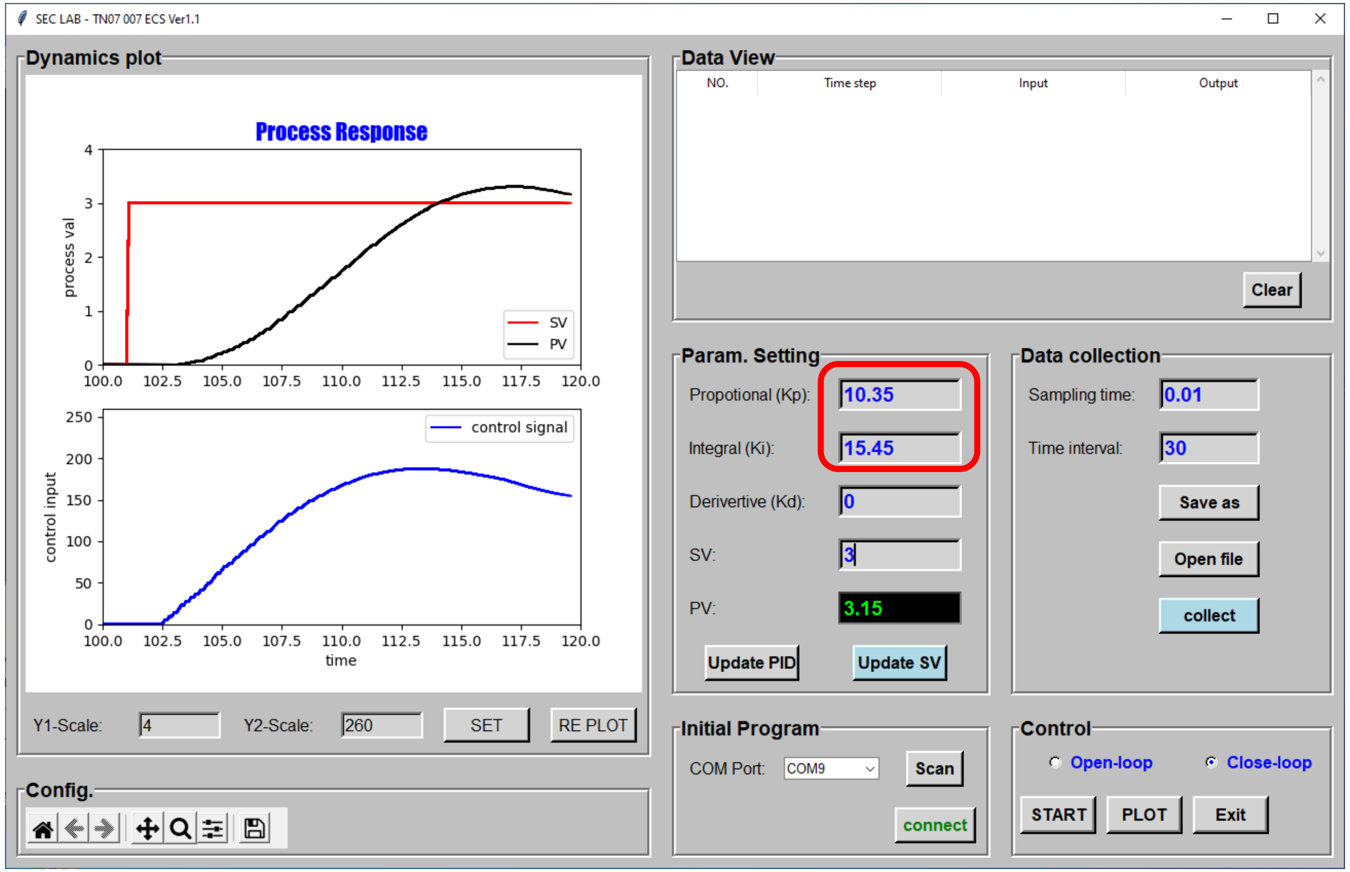
$$T_i = \frac{0.2}{0.3} = 0.67$$

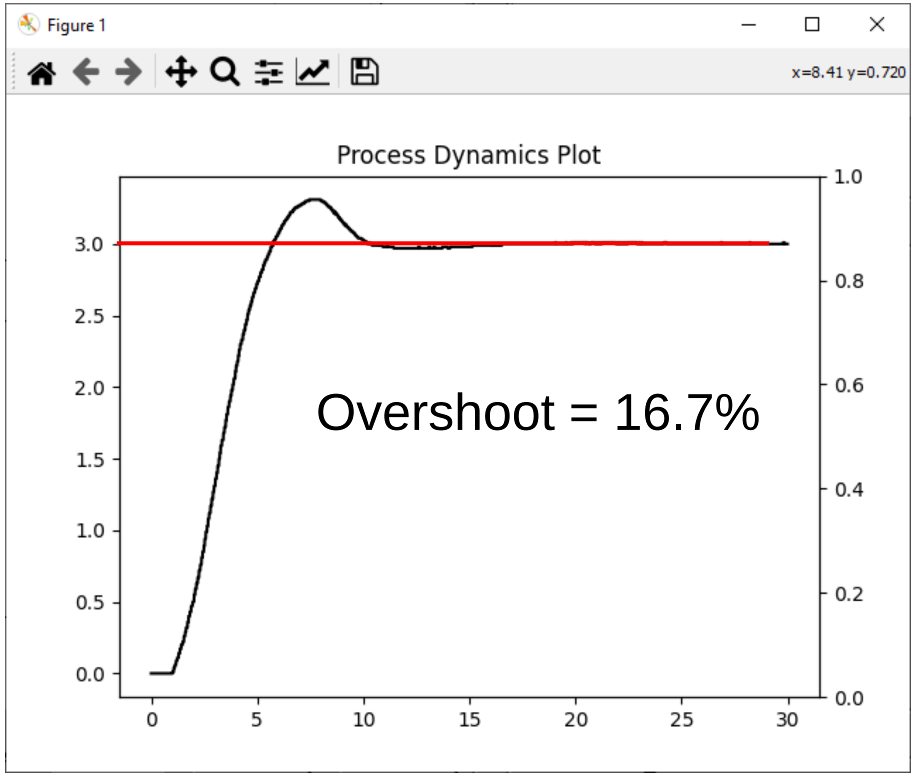
$$K_P = 10.35$$

$$K_I = \frac{10.35}{0.67} = 15.45$$

Practices

PID Controller Design Method 1 Process reaction curve





Practices

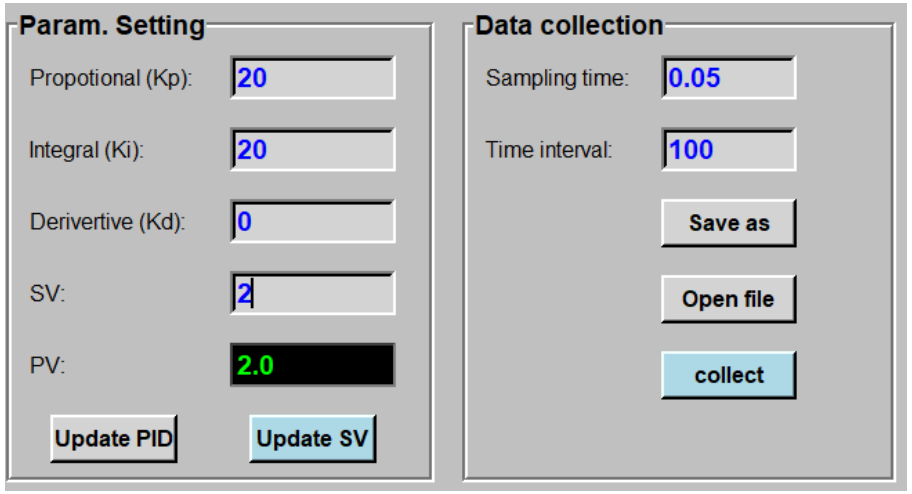
PID Controller Design Method 3

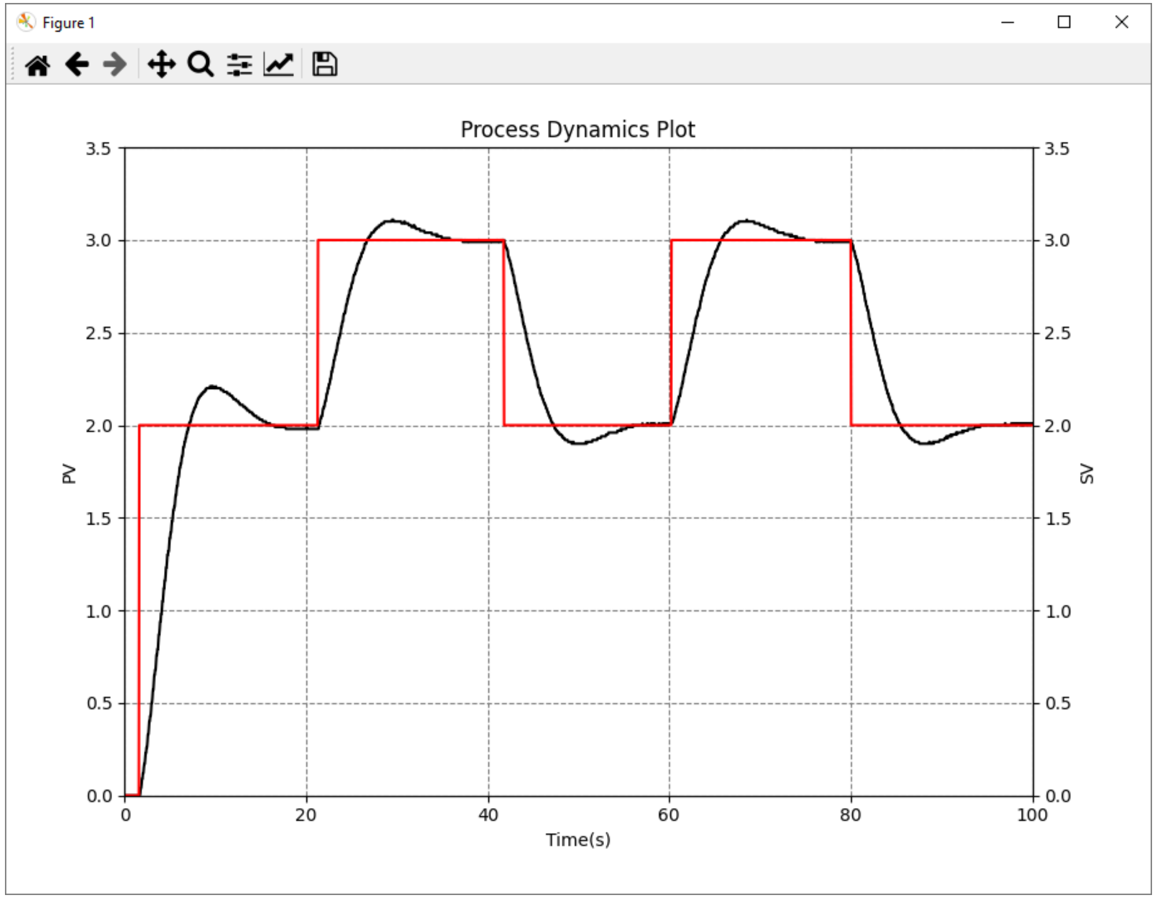
Model based tuning

Practices

- Initial Gauss Kp = 20, Ki = 20

- Collect data in desire operating range





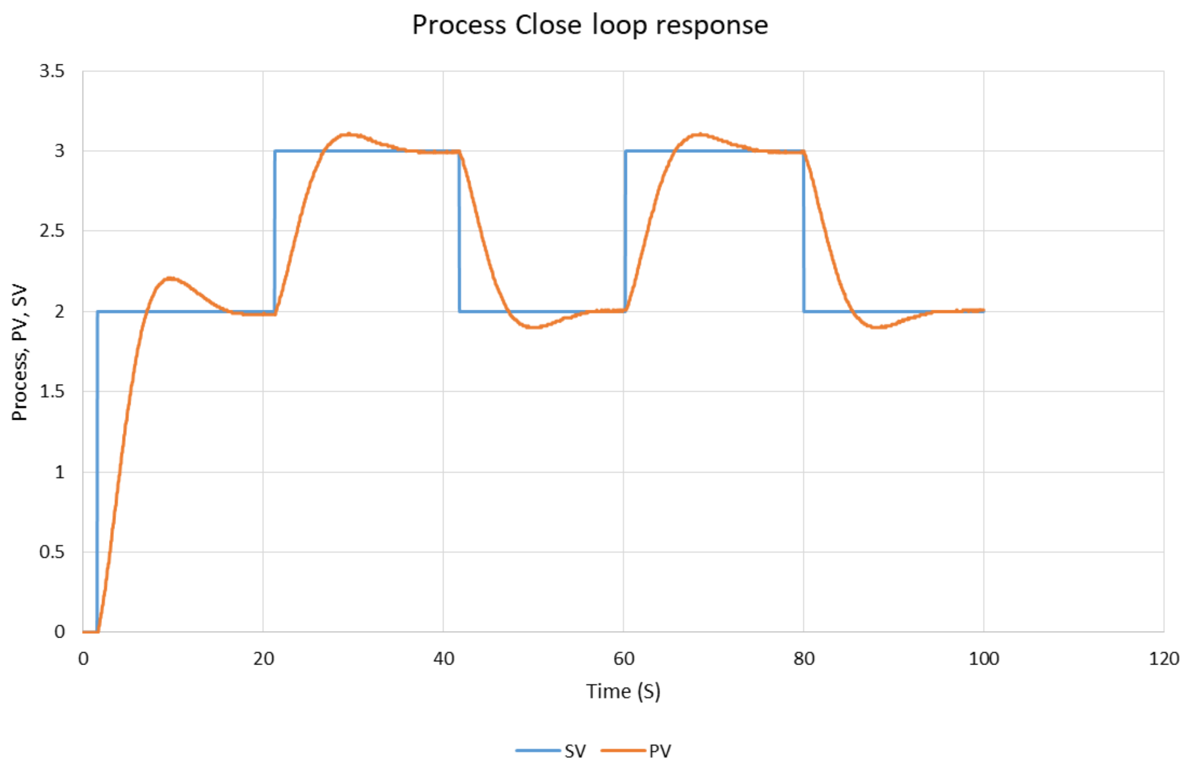
Practices

PID Controller Design Method 3 Model based tuning

Use for Model estimation

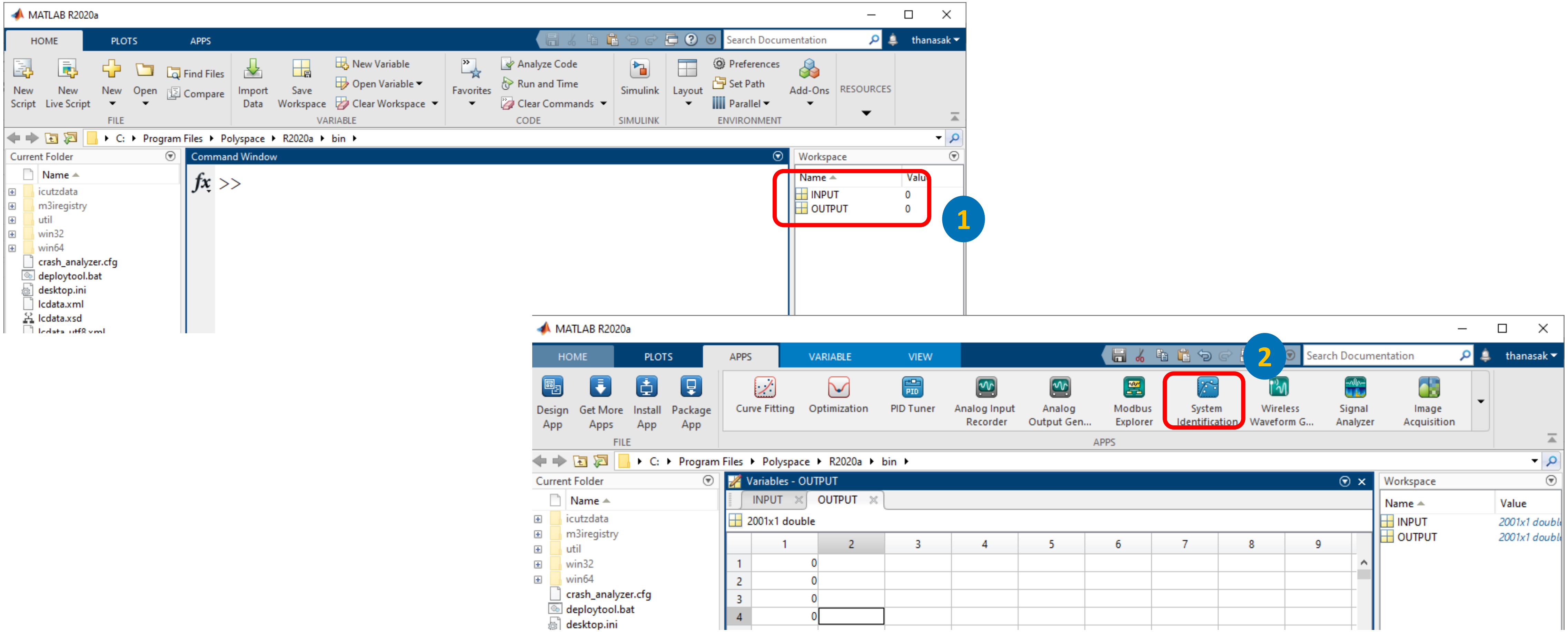
F42

	time	U	SV	PV
41	1.95	57.18	2	0.09
42	2	60.36	2	0.12
43	2.05	60.36	2	0.12
44	2.1	63.48	2	0.15
45	2.15	66.65	2	0.17
46	2.2	66.65	2	0.17
47	2.25	69.67	2	0.2
48	2.3	72.62	2	0.23
49	2.35	72.62	2	0.23
50	2.4	75.52	2	0.26
51	2.45	75.52	2	0.26
52	2.5	78.35	2	0.29
53	2.55	80.91	2	0.33
54	2.6	80.91	2	0.33
55	2.65	83.51	2	0.36
56	2.7	86.03	2	0.4
57	2.75	86.03	2	0.4



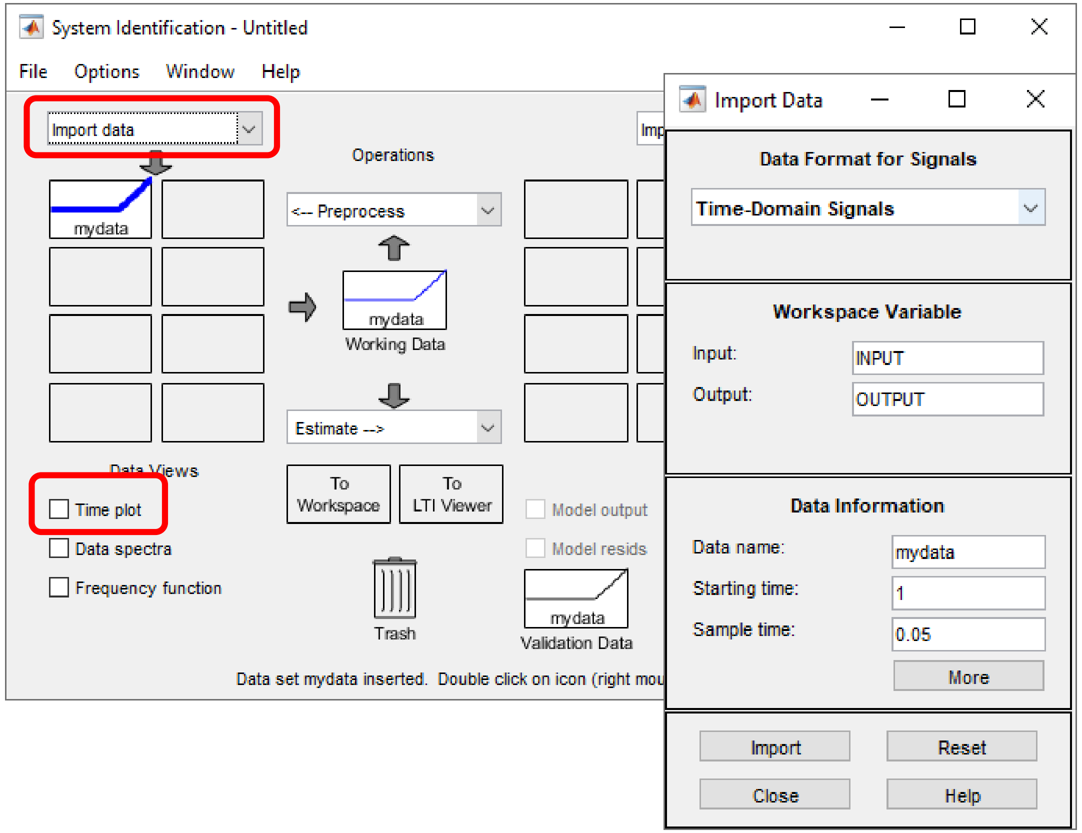
Practices

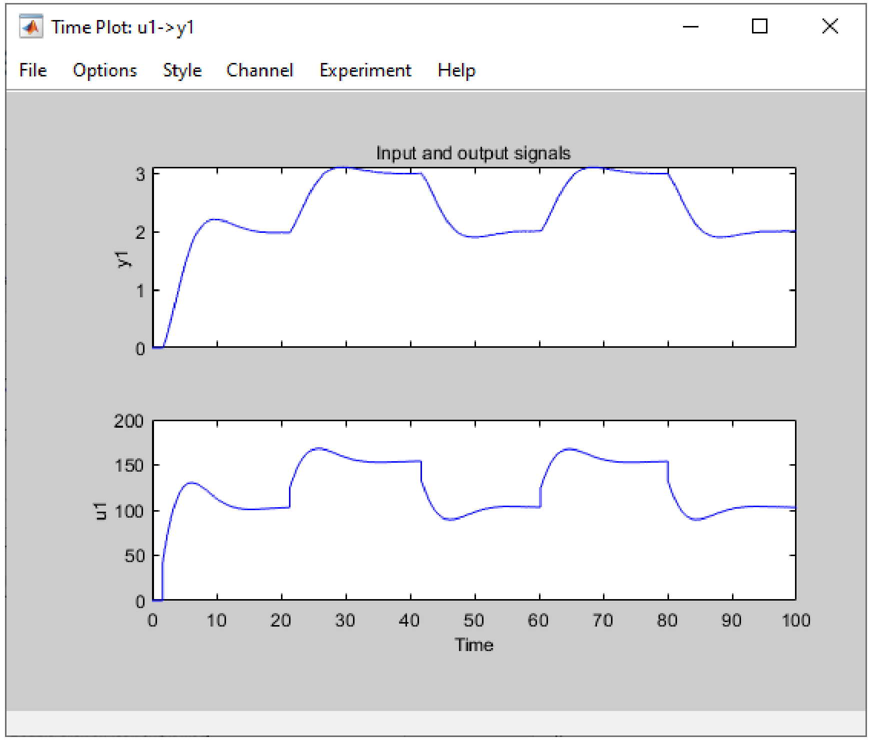
PID Controller Design Method 3 Model based tuning



Practices

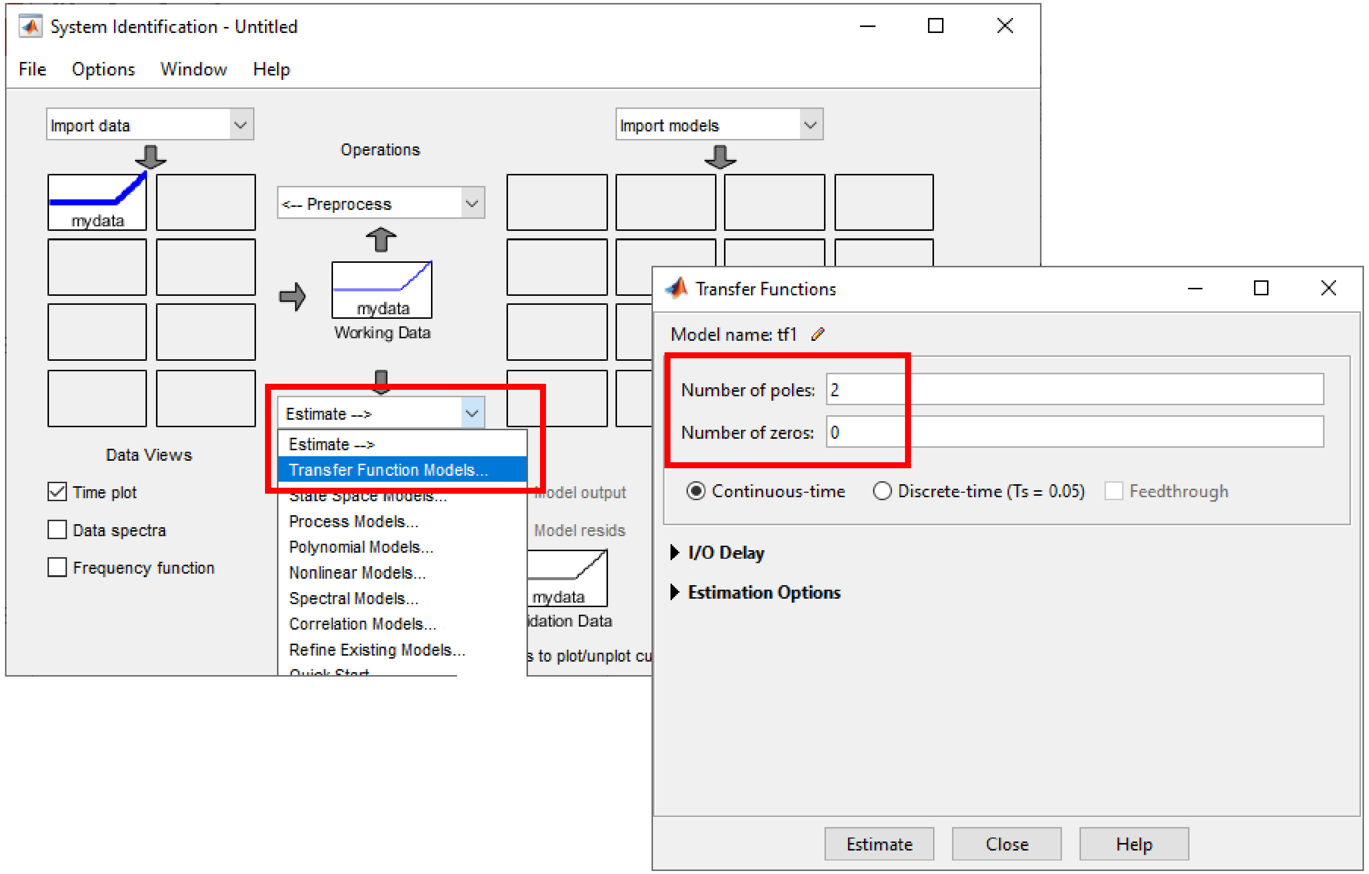
PID Controller Design Method 3 Model based tuning

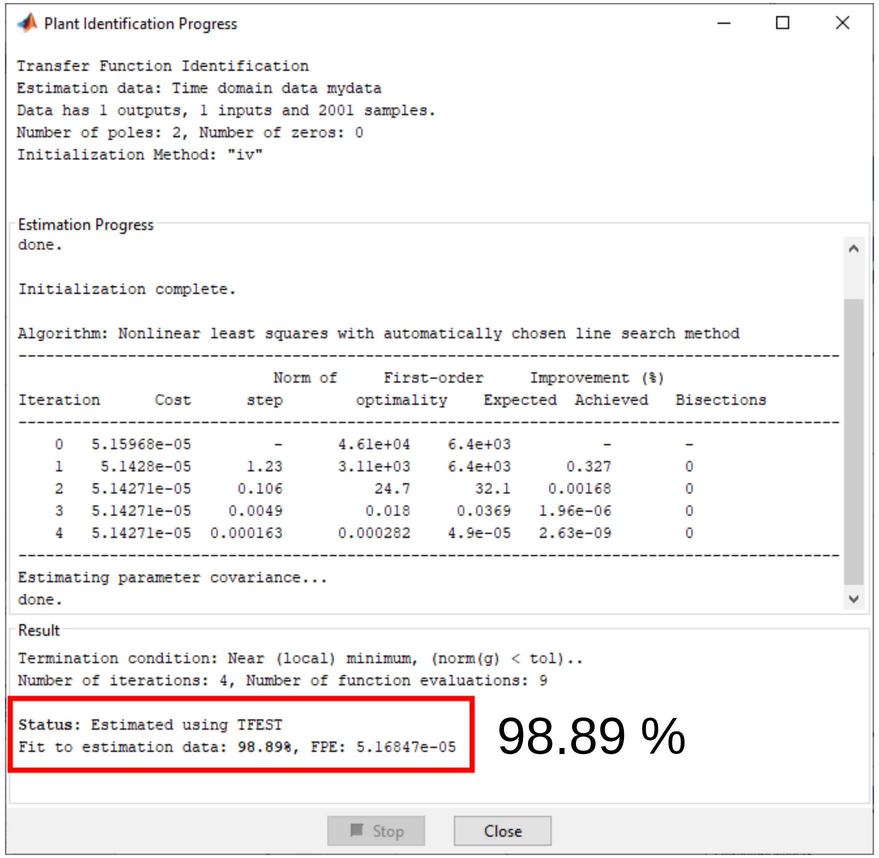




Practices

PID Controller Design Method 3 Model based tuning

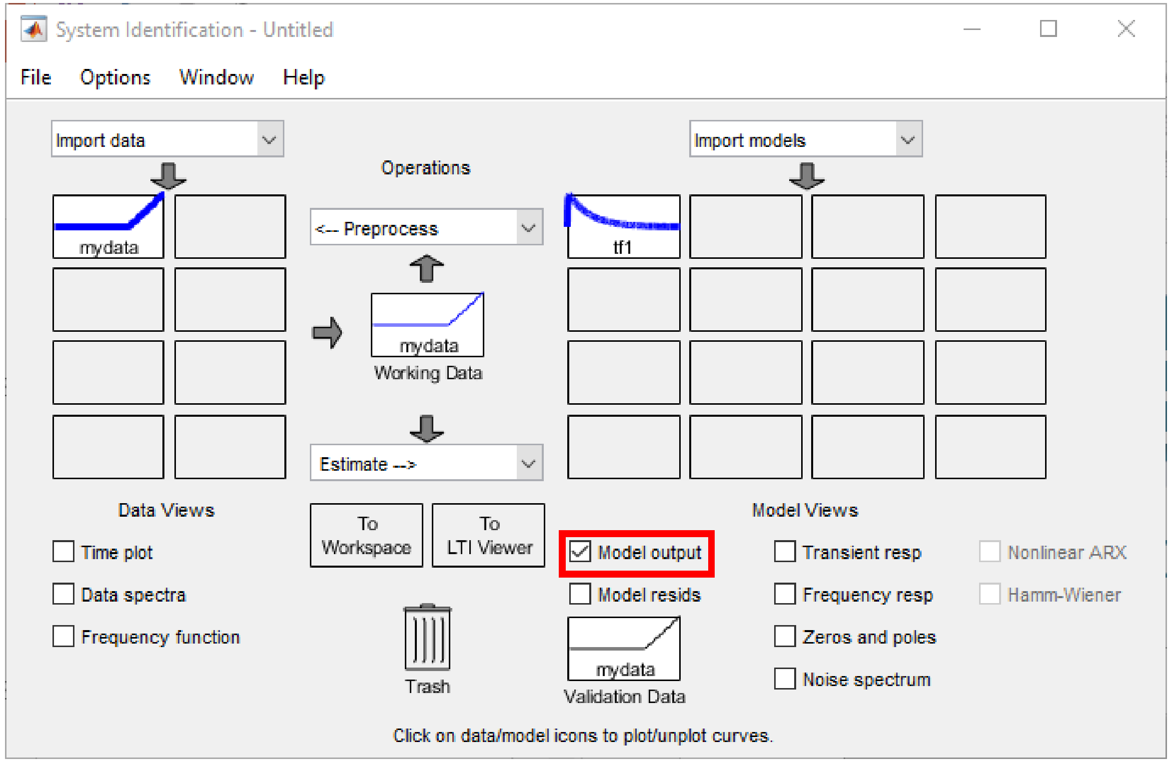


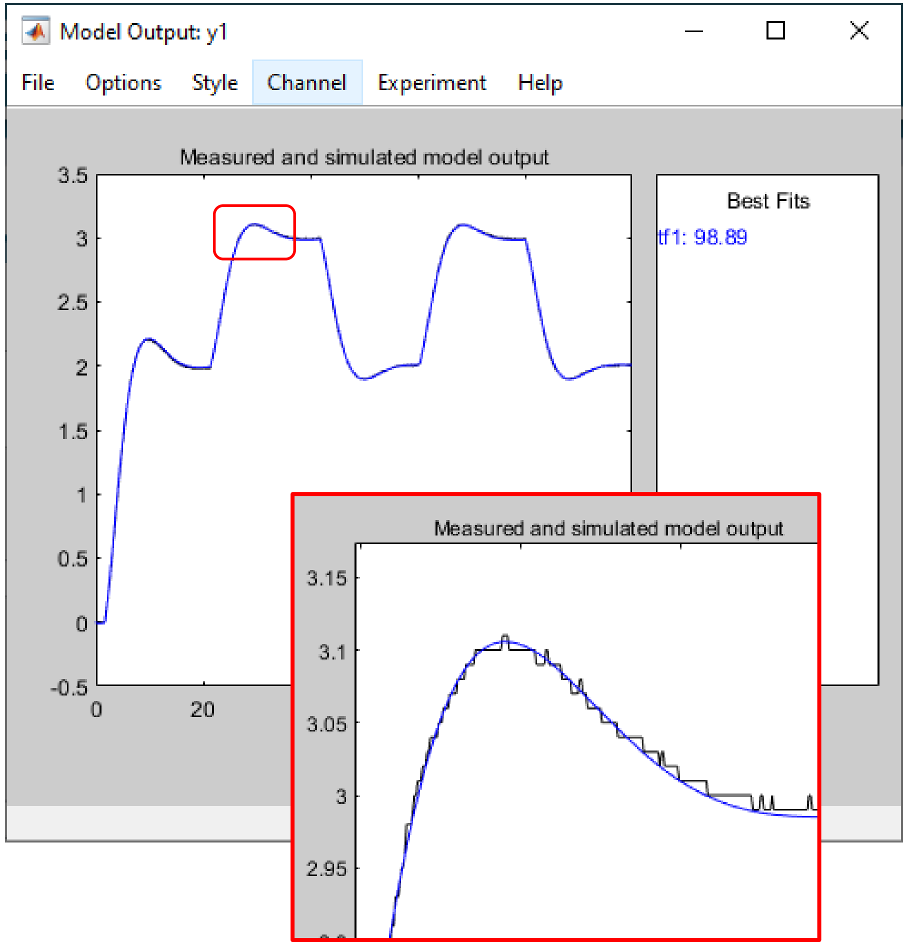


98.89 %

Practices

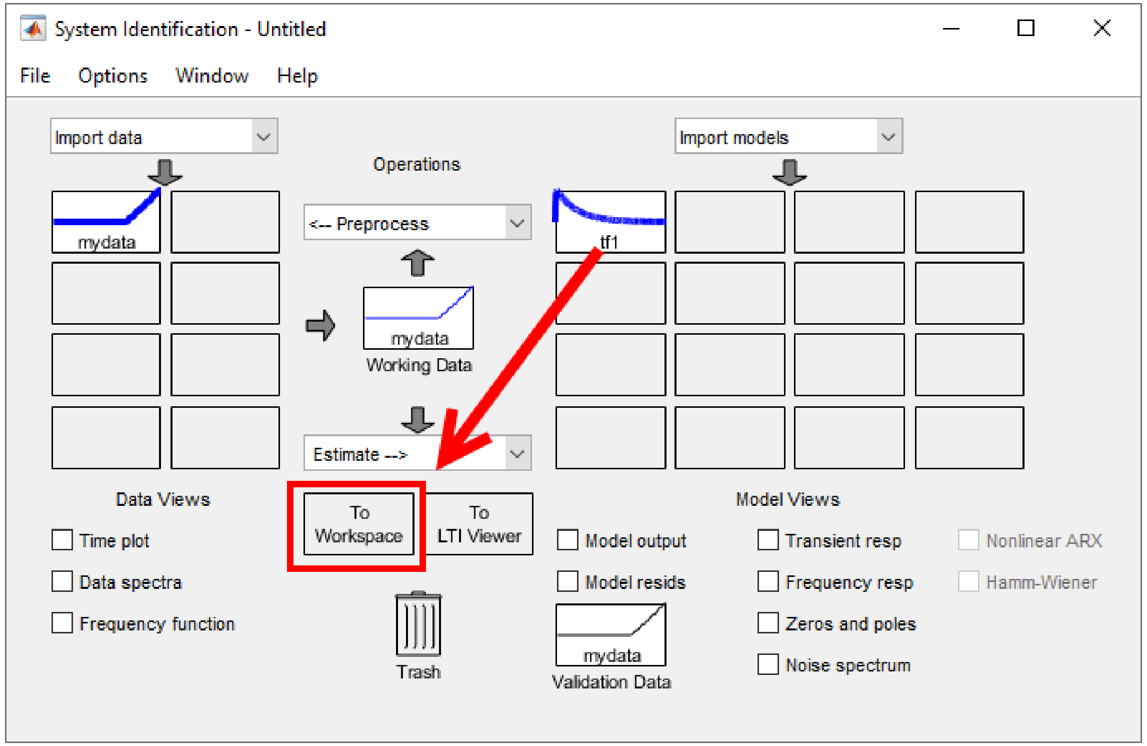
PID Controller Design Method 3 Model based tuning

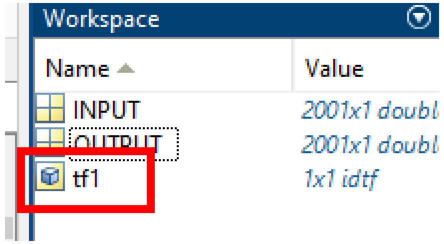




Practices

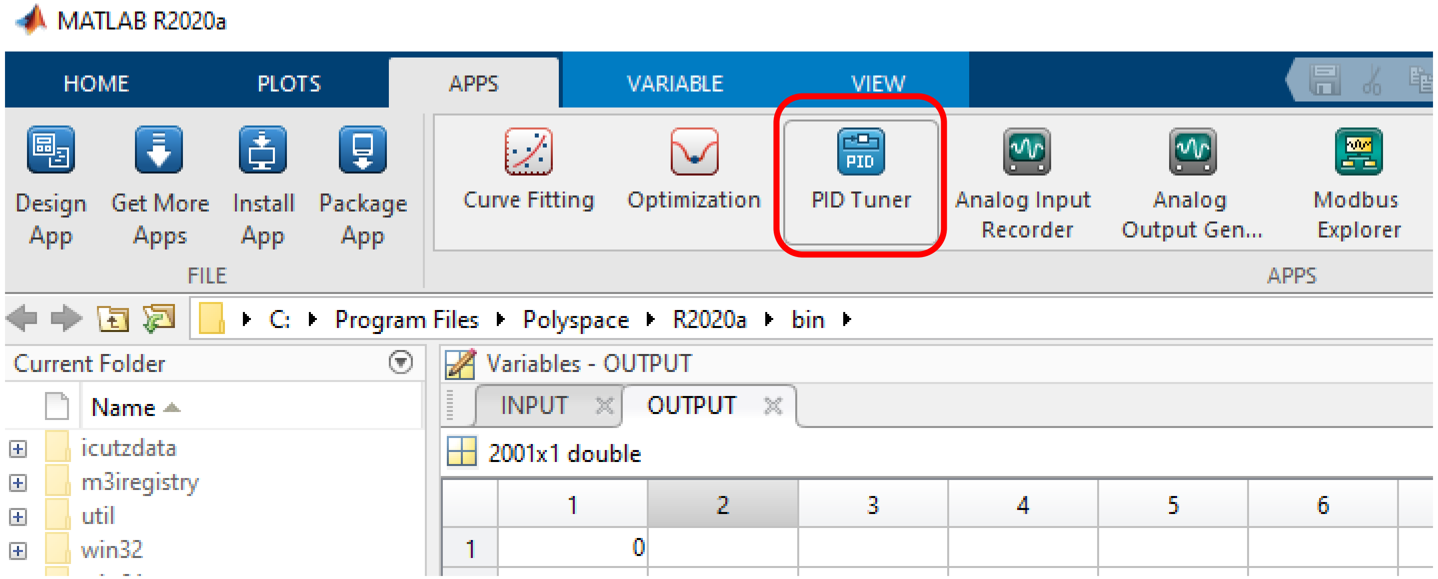
PID Controller Design Method 3 Model based tuning





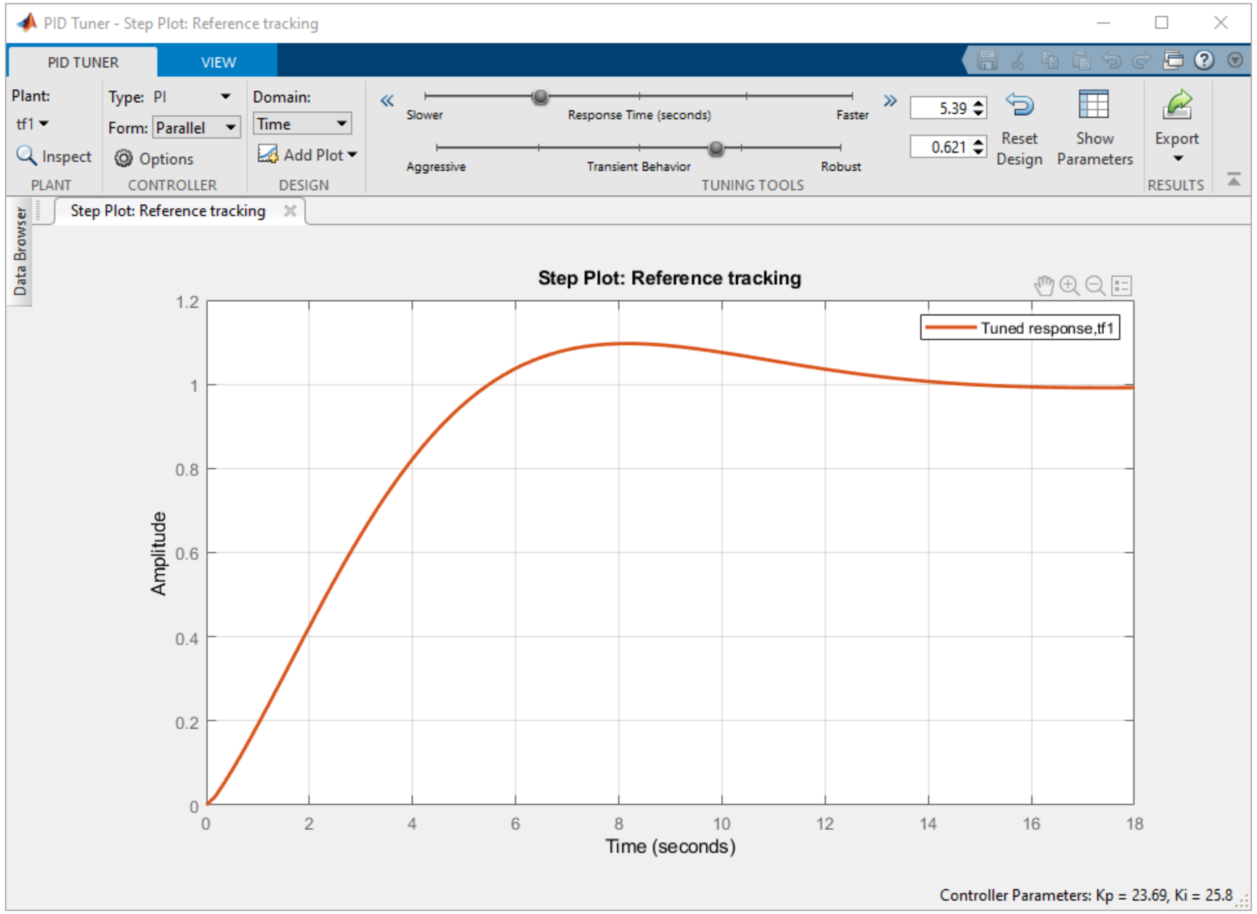
Practices

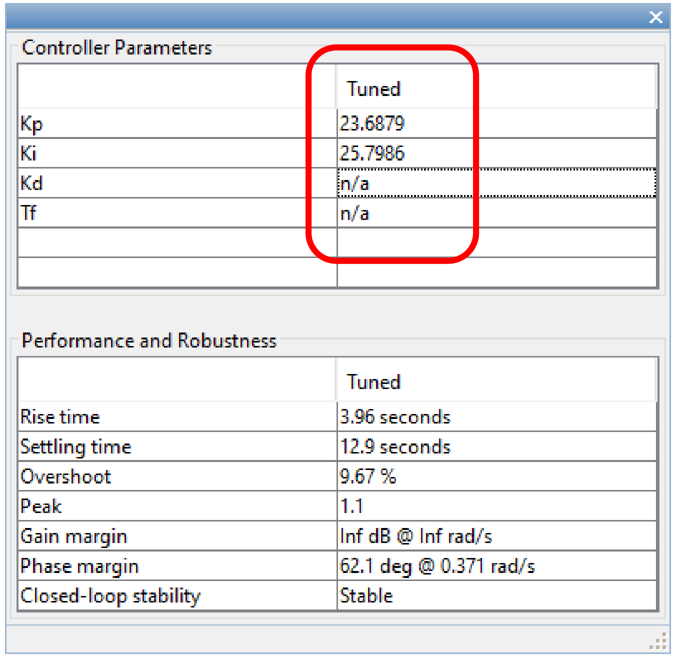
PID Controller Design Method 3 Model based tuning



Practices

PID Controller Design Method 3 Model based tuning





Controller Parameters

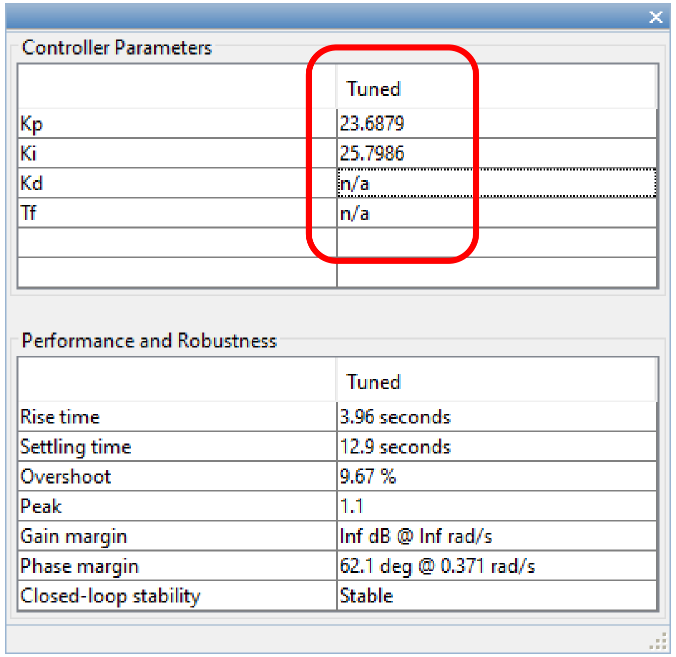
	Tuned
Kp	23.6879
Ki	25.7986
Kd	n/a
Tf	n/a

Performance and Robustness

	Tuned
Rise time	3.96 seconds
Settling time	12.9 seconds
Overshoot	9.67 %
Peak	1.1
Gain margin	Inf dB @ Inf rad/s
Phase margin	62.1 deg @ 0.371 rad/s
Closed-loop stability	Stable

Practices

PID Controller Design Method 3 Model based tuning



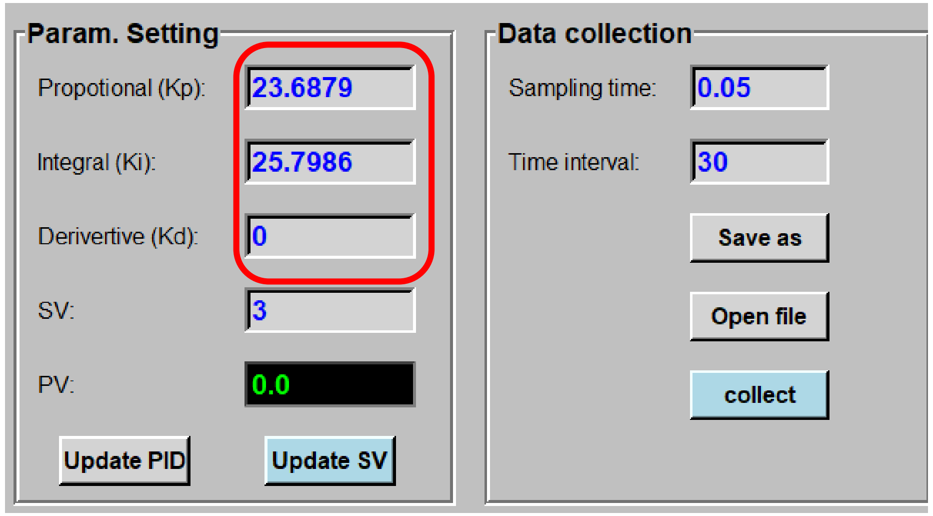
Controller Parameters

	Tuned
Kp	23.6879
Ki	25.7986
Kd	n/a
Tf	n/a

Performance and Robustness

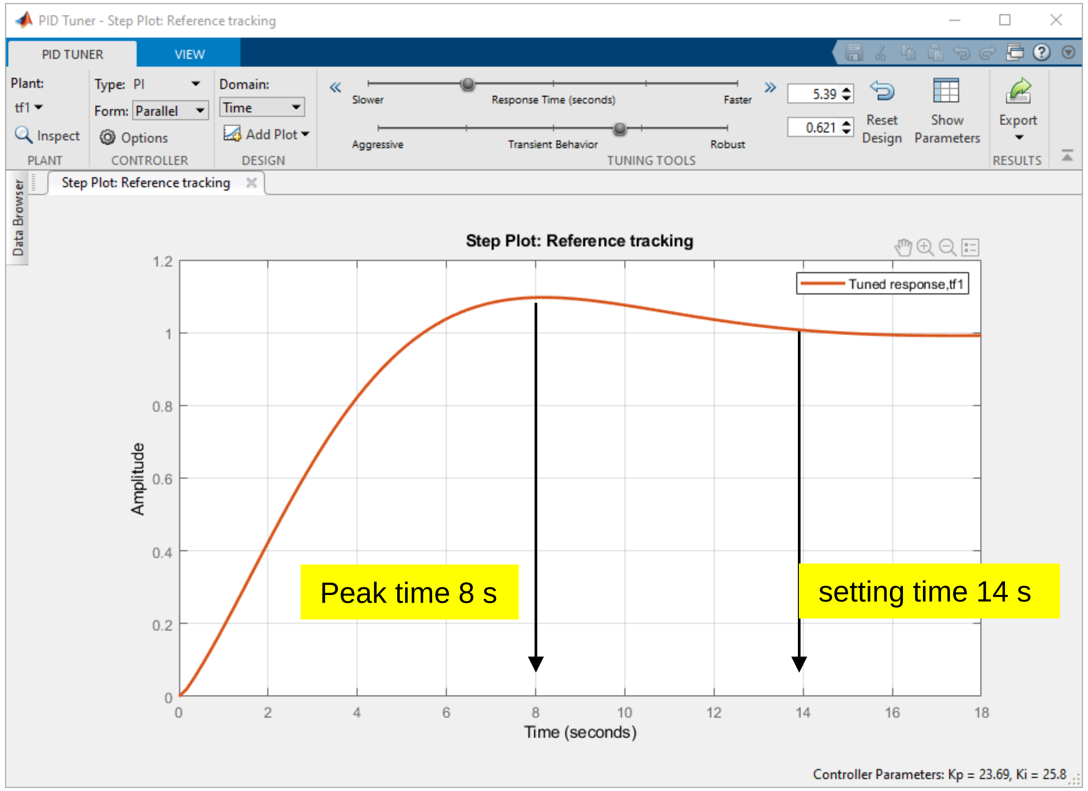
	Tuned
Rise time	3.96 seconds
Settling time	12.9 seconds
Overshoot	9.67 %
Peak	1.1
Gain margin	Inf dB @ Inf rad/s
Phase margin	62.1 deg @ 0.371 rad/s
Closed-loop stability	Stable

Controller testing

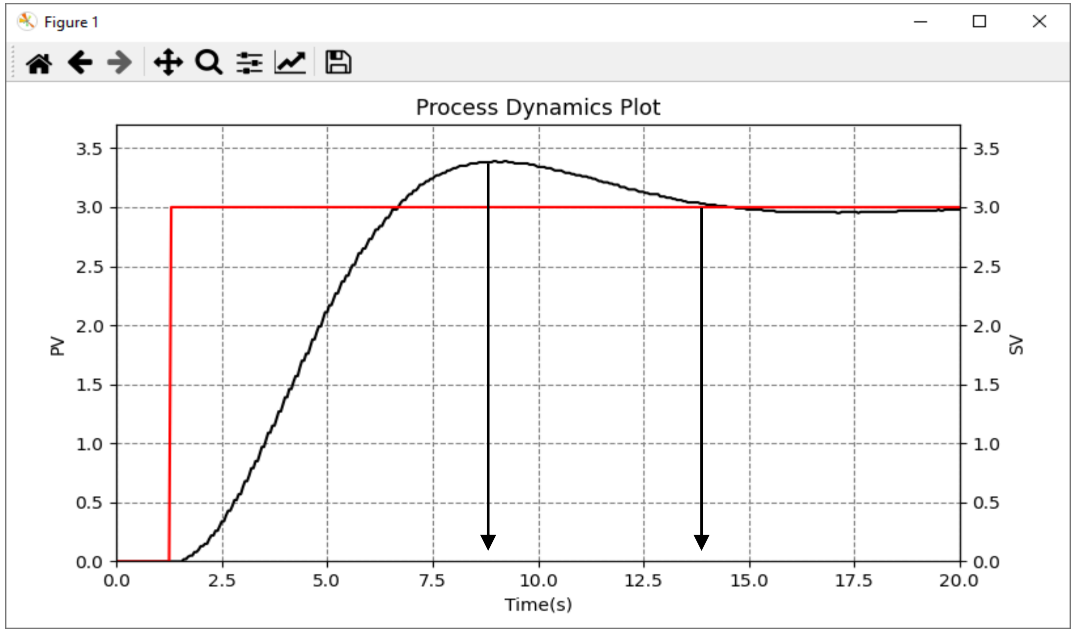


Practices

PID Controller Design Method 3 Model based tuning



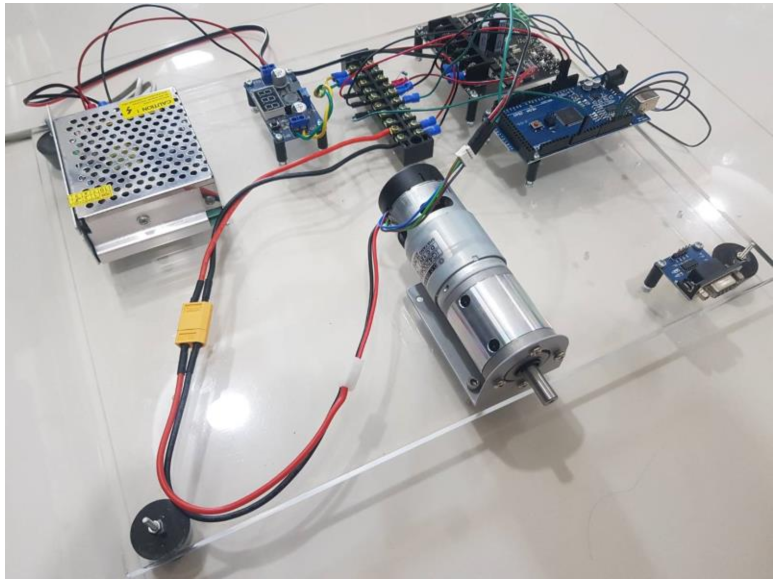
Design

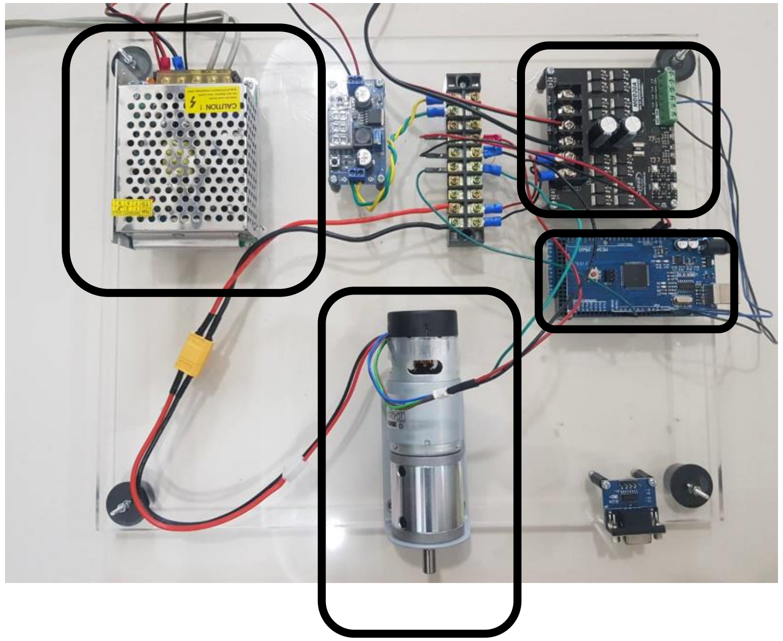


Testing

Practices

Practices Application



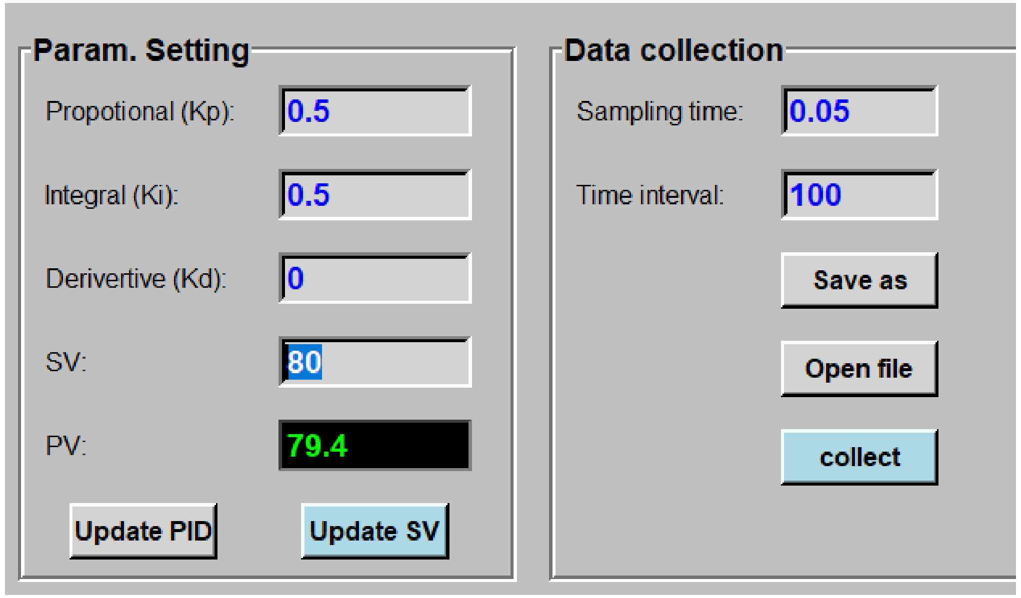


DC motor Speed Control

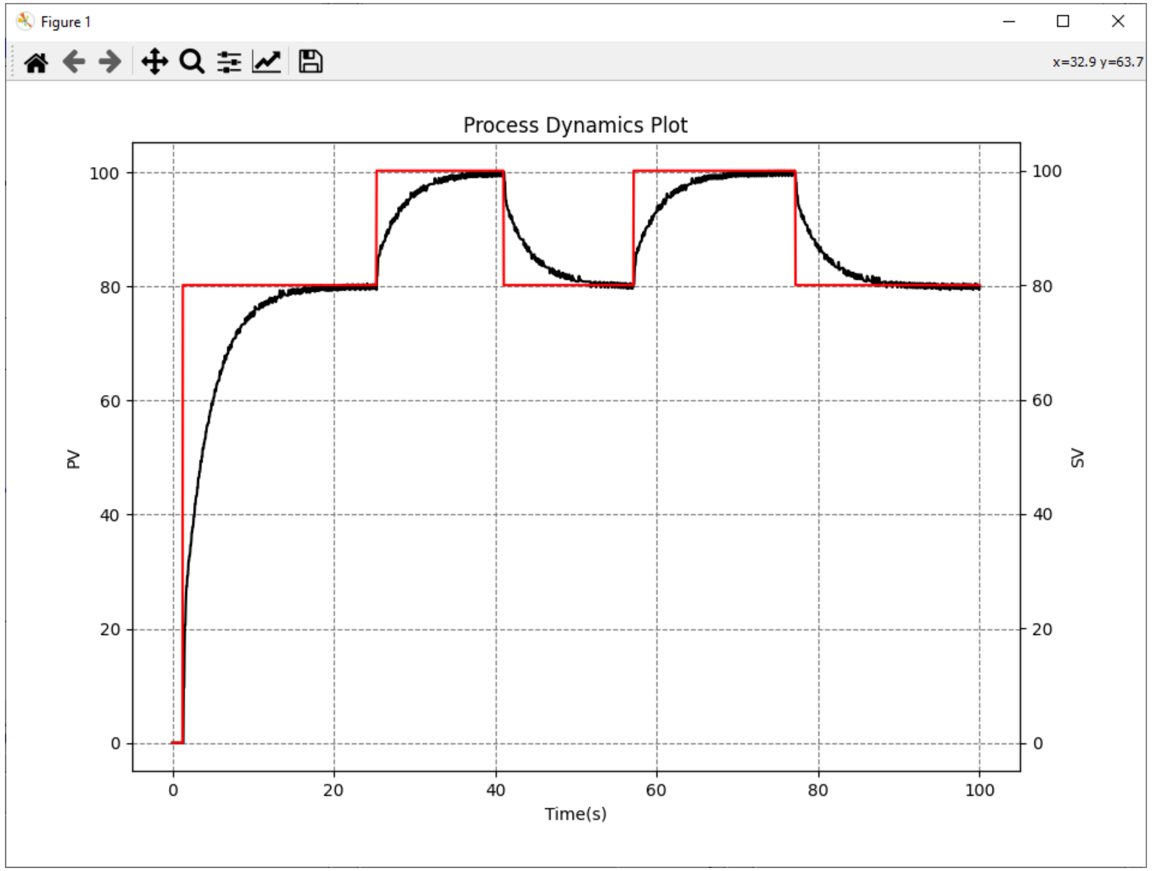
Practices

- Initial Gauss Kp = 0.5, Ki = 0.5

- Collect data in desire operating range

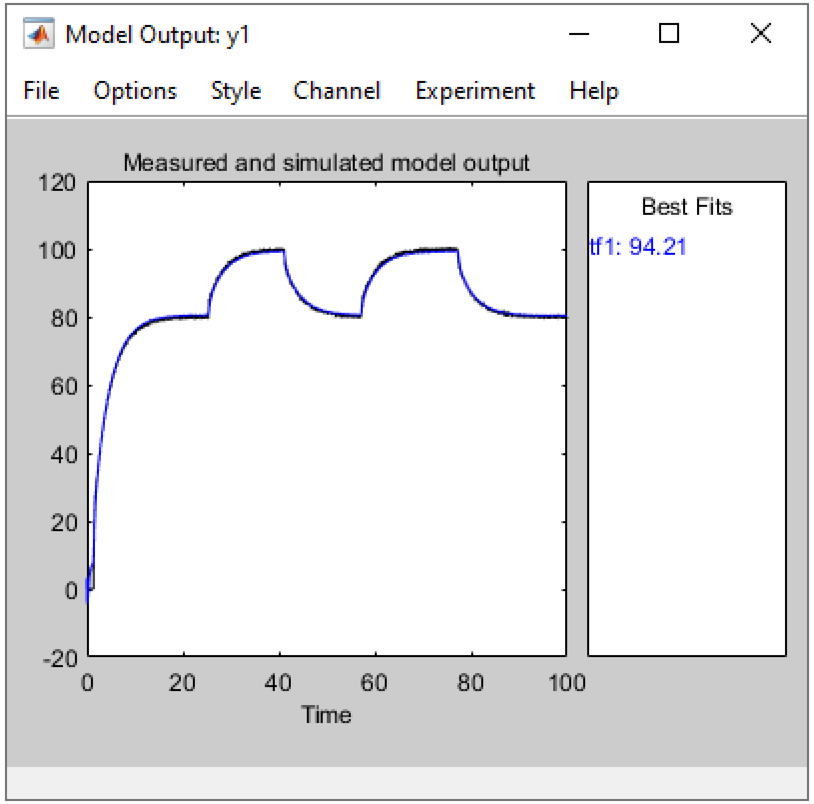


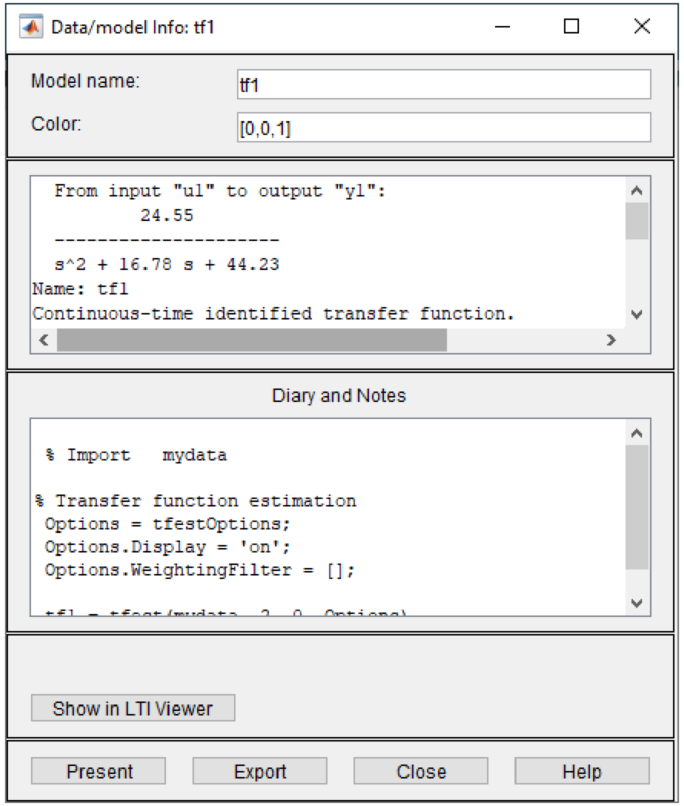
Experiment data



Practices

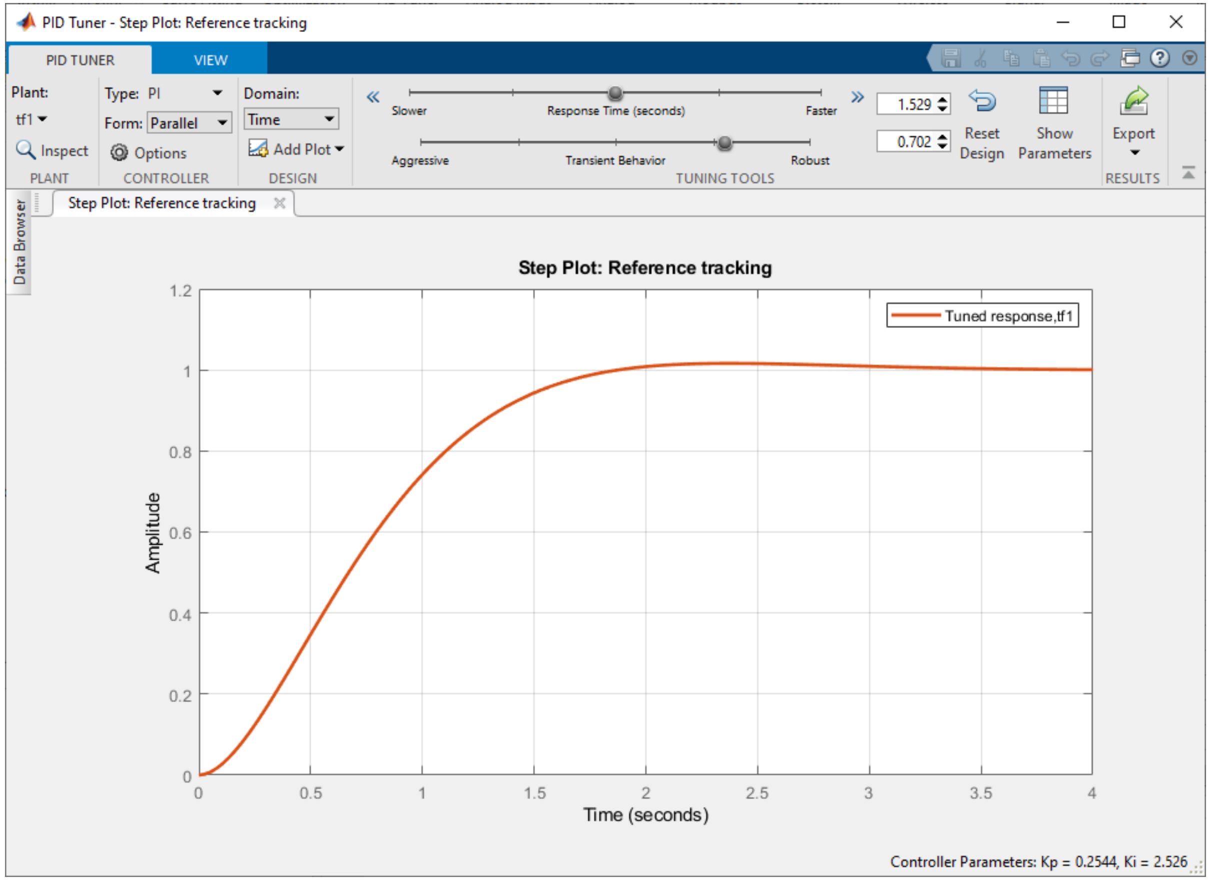
Best Fitted model 94.21 %





Practices

Controller Design



Controller Parameters

	Tuned
Kp	0.25439
Ki	2.5265
Kd	n/a
Tf	n/a

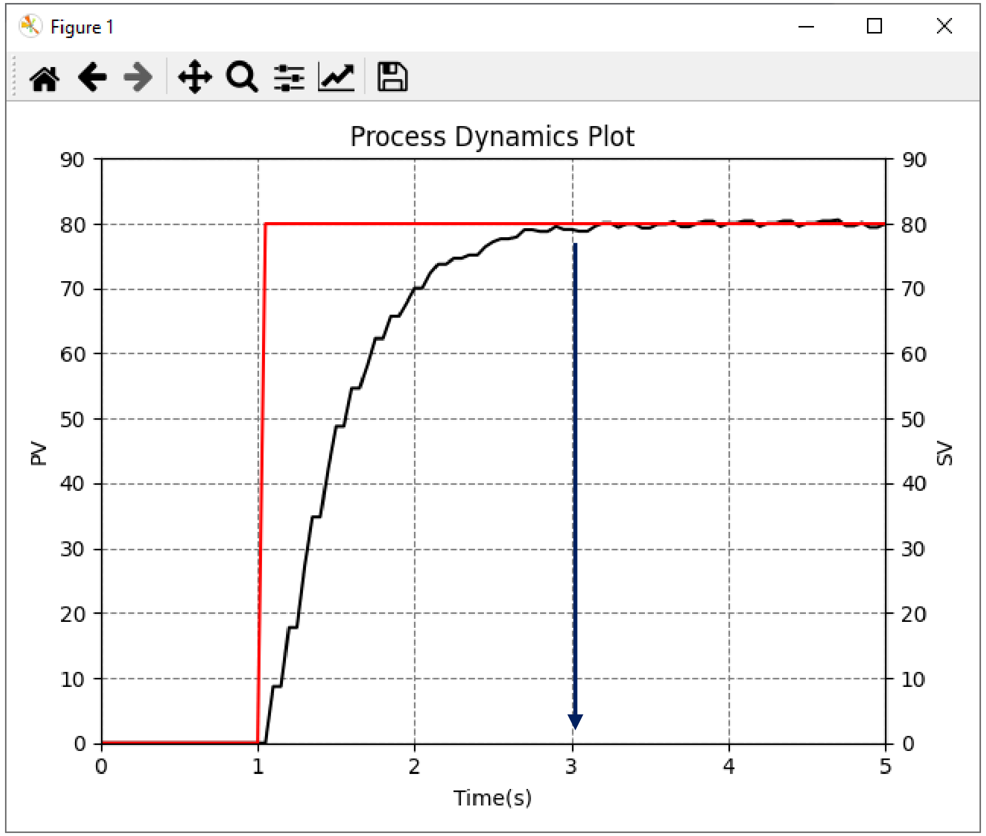
Performance and Robustness

	Tuned
Rise time	1.13 seconds
Settling time	1.7 seconds
Overshoot	1.62 %
Peak	1.02
Gain margin	Inf dB @ Inf rad/s
Phase margin	70.2 deg @ 1.31 rad/s
Closed-loop stability	Stable

Practices

Controller Design

Design



Testing