Laporan Hardware In The Loop

Nama : Hudzaifah Afif Al Fatih Nasution

NIM : 13219031

Nama : Muhammad Ali

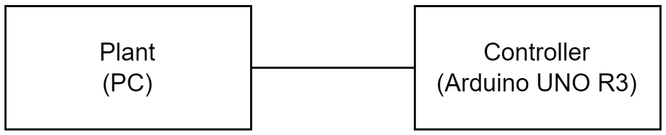
NIM : 13219017

Contents

Daftar Gambar

1 Perancangan Hardware

Skema rangkaian & penjelasan



Gambar 1 Top Level

Komponen yang digunakan adalah sebuah PC dengan distribusi Python terinstall dan sebuah microcontroller ATMega328p dalam board Arduino UNO R3 yang dihubungkan melalui USB.

PC digunakan sebagai implementasi dari suatu plant digital dan Arduino digunakan sebagai implementasi kendali PID digital. Baud rate dan sampling time dari kedua komponen ini harus bernilai sama. Dalam percobaan ini digunakan baud rate 9600 dan sampling time 0.05 s.

2 Perancangan Software

2.1 Komunikasi Data

- Format data seria
- Format data Pengendali ke simulator
- Format data Simulator ke pengendali
- Perhitungan kecepatan komunikasi

Catatan:

Sampling rate pada sistem kendali maksimum dibatasi oleh kecepatan komunikasi data. Hitunglah kecepatan maksimum sampling rate yang dimungkinkan.

Asumsi kecepatan serial 9600 bps, jumlah data yang dikirim tergantung format data yang dipakai.

Komunikasi dilakukan antara plant dan controller dengan saling bertukar paket string yang dikirimkan melalui serial. Seluruh paket data berupa string yang diawali oleh karakter I dan diakhiri oleh karakter F.

Controller mengirimkan paket string dengan format dengan sintaks Python

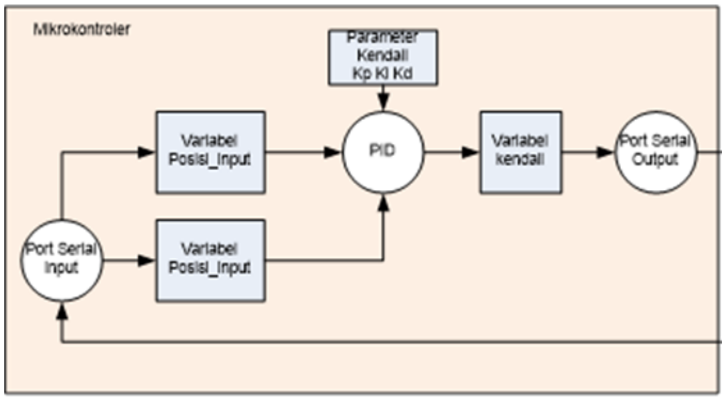
```
"I"+str(data)+"F"
```

dimana data adalah output dari controller berupa angka floating point dengan ketelitian 6 angka dibelakang titik koma.

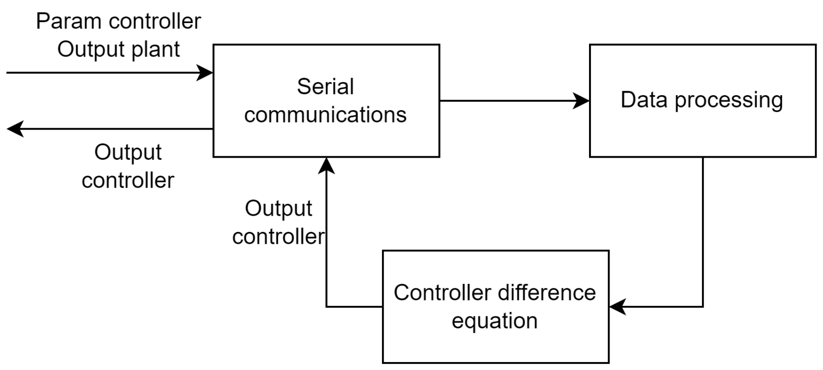
Plant/PC memiliki beberapa tipe paket data dengan awalan yang berbeda-beda sesuai dengan jenis data yang dikirimkan.

Format paket	Isi paket
`"IS"+str(sp)+"F\n"`	Set point
`"IP"+str(Kp)+"F\n"`	Kp
`"IN"+str(Ki)+"F\n"`	Ki
`"ID"+str(Kd)+"F\n"`	Kd
`"ICF\n"`	Mengecek apakah semua paket awal telah diterima controller
`"IV"+str(y[0])+"F\n"`	Output dari plant
`"ILF\n"`	Selesai mengirim output dari plant

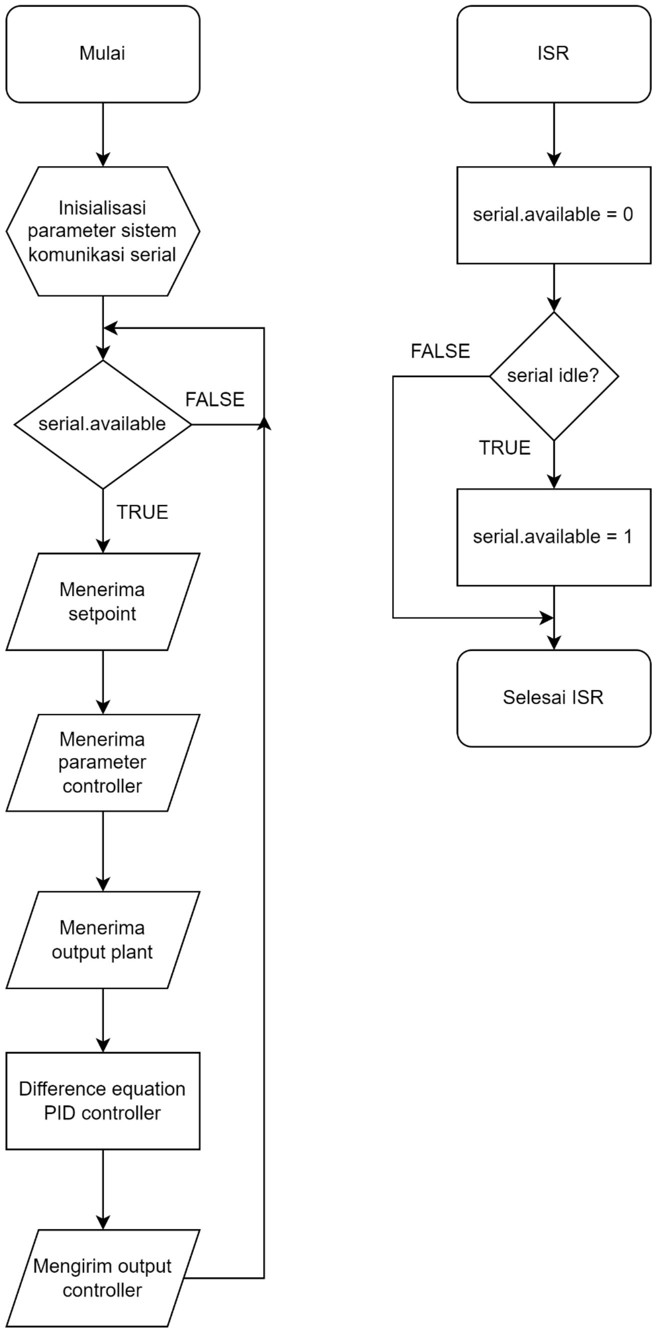
2.2 Pengendali



Gambar 2 Diagram blok perangkat lunak di pengendali



Gambar 3 Data Flow Diagram Microcontroller



Gambar 4 Flowchart Microcontroller

2.2.1 Port Serial

Port serial disetting dengan baud rate 9600. Tidak bisa digunakan baud rate yang terlalu besar karena akan menyebabkan transmisi data semakin tidak stabil. Digunakan timeout serial sebesar 1 detik.

2.2.2 PID

Digunakan persamaan PID digital

G(z) = Kp + Ki*Ts/(z-1)+D*(z-1)/Ts

Dengan penyederhanaan diperoleh

G(z) = [(2*Kp*Ts + Ki*Ts*Ts +4*Kd) + (2*Ki*Ts*Ts-8*Kd)*z^-1 + (4*Kd-2*Kp*Ts+Ki*Ts*Ts)/(2*Ts)*z^-2]/[1-z^2]

Dari fungsi transfer diatas didapati implementasi persamaan diferensial pada sintaks C++ sebagai berikut

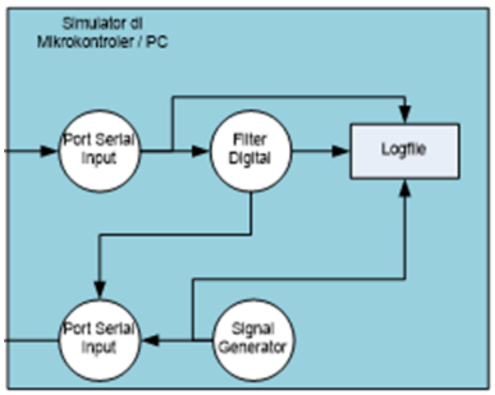
y[0] = ((2*Kp*Ts + Ki*Ts*Ts +4*Kd)*x[0] + (2*Ki*Ts*Ts-8*Kd)*x[1] + (4*Kd-2*Kp*Ts+Ki*Ts*Ts)*x[2])/(Ts*2) + y[2];

Input dan output disimpan dalam buffer dimana x adalah input pada t, y[0] adalah output pada t, y[2] adalah output pada t-2. Ts disini adalah sampling time yang bernilai 0.05

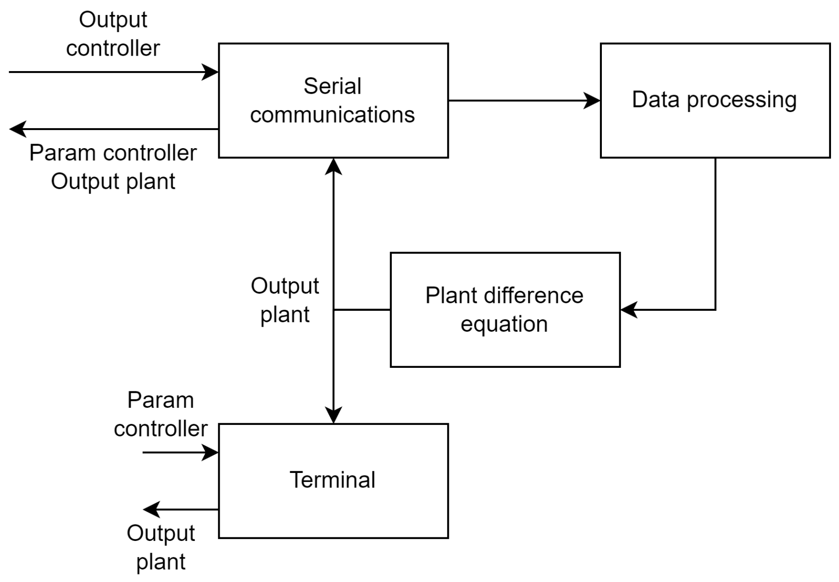
2.2.3 Parameter Kendali

Parameter kendali akan diterima dari PC, digunakan nilai default Kp = 1000, Ki = 0.5, dan Kd = 0.5

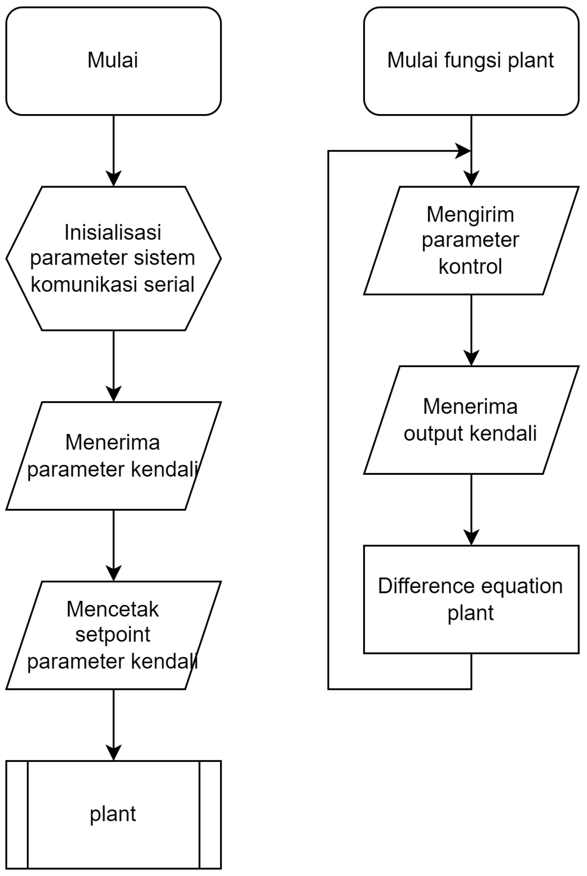
2.3 Simulator



Gambar 5 Diagram blok perangkat lunak di simulator



Gambar 6 Data Flow Diagram Simulator


Mulai
Inisialisasi parameter sistem komunikasi serial
Menerima parameter kendali
Mencetak setpoint parameter kendali
plant
Mulai fungsi plant
Mengirim parameter kontrol
Menerima output kendali
Difference equation plant

Gambar 7 Flowchart Simulator

Diagram blok perangkat lunak di simulator dalam format Data Flow Diagram

Diagram alir perangkat lunak di simulator

2.3.1 Filter Digital

Sistem menggunakan model pengendali kecepatan motor dari praktikum sistem kendali dengan fungsi transfer domain Laplace.

G(s) = K/(T*s + 1)

Koefisien K dapat dicari dari menentukan besar steady state dari respons unit step sistem open loop. Koefisien T dapat dicari dengan mencari waktu dimana besar dari respons unit step sistem openloop sebesar 1-   1/e dari besar ketika steady state

G(s) = 0.9/(120.5*s + 1)

Fungsi ini dapat diubah dengan menggunakan transfer domain-z dengan menggunakan transform bilinar

s = 2/Ts (z-1)/(z+1)

didapati fungsi transfer domain-z

G(z) = 3.734e-4/(z-0.9996)

Fungsi transfer ini dapat diubah menjadi persamaan differens sebagai implementasi filter digital

```
y[0] = A*x - B[1]*y[1]
```

Dimana y[0] adalah y saat n, x adalah x saat n, y[1] adalah y saat n-1.

Parameter filter digital disimpan dalam array berikut

A = 3.374e-4

B = [1, -0.9996]

2.3.2 Port Serial

Port serial disetting sama dengan controller dengan baud rate 9600. Tidak bisa digunakan baud rate yang terlalu besar karena akan menyebabkan transmisi data semakin tidak stabil. Digunakan timeout serial sebesar 1 detik.

2.3.3 Signal Generator

Sinyal generator dilakukan dengan menginisialisasi variabel setpoint dengan nilai 1 di awal program sebagai implementasi step response.

2.3.4 Mekanisme Log

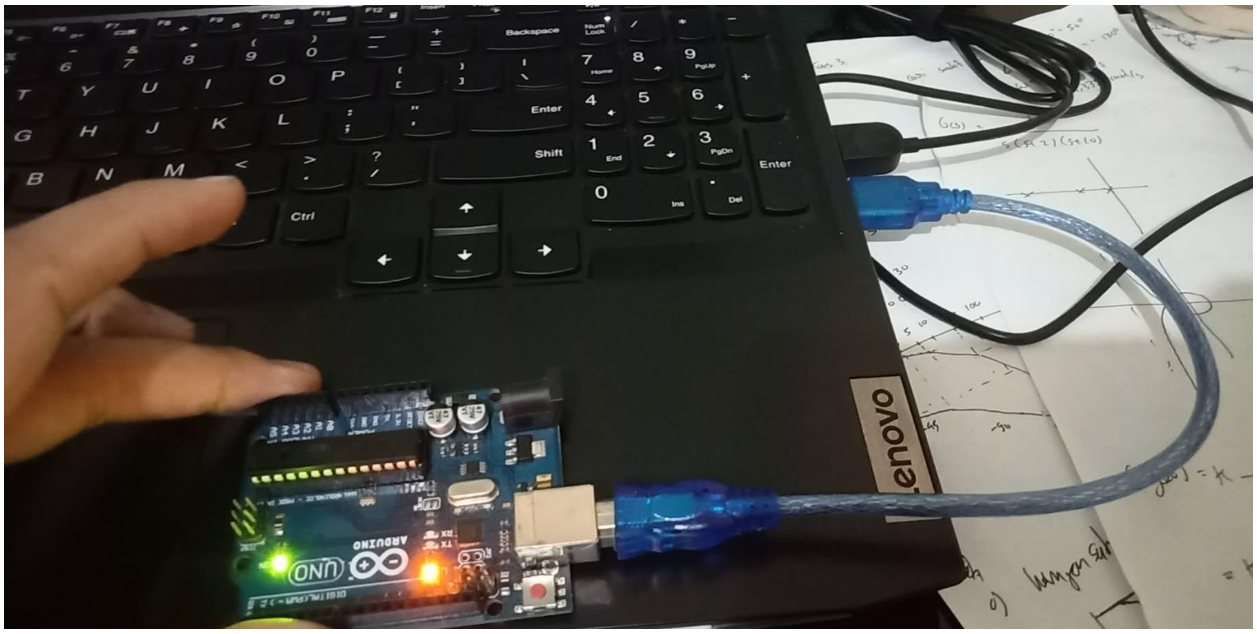
Log data dilakukan dengan mencetak dalam format CSV di terminal

3 Implementasi Hardware

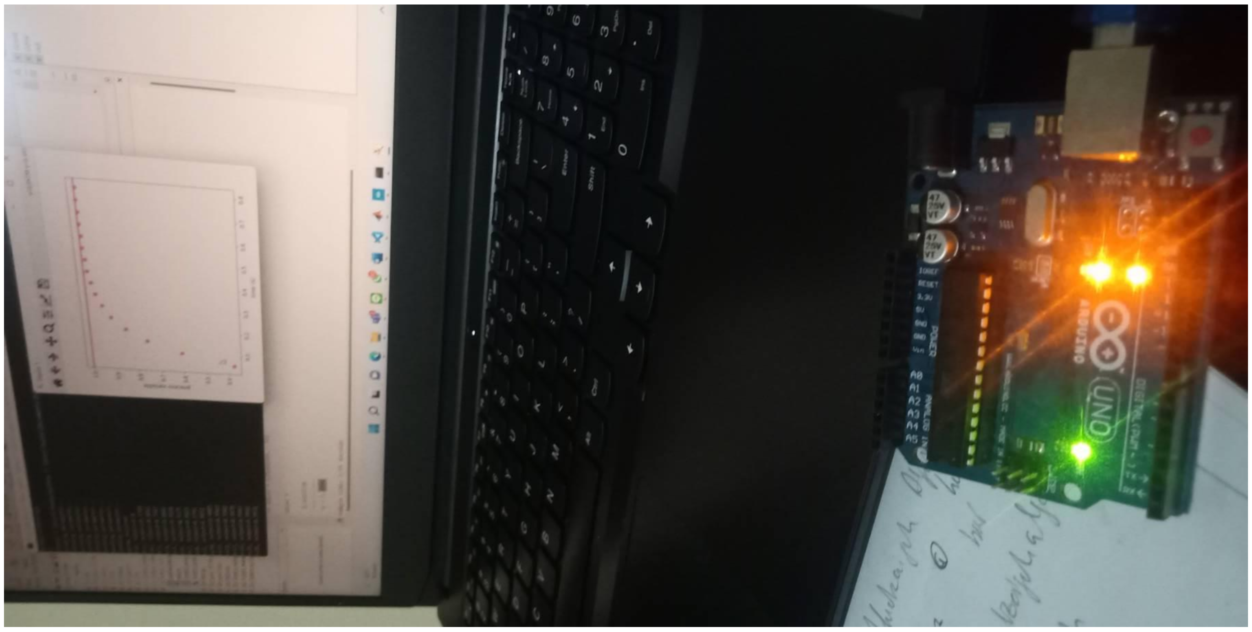
Teknis penyambungan hardware

Hardware dihubungkan dengan konektor USB ke Arduino UNO

Foto perangkat



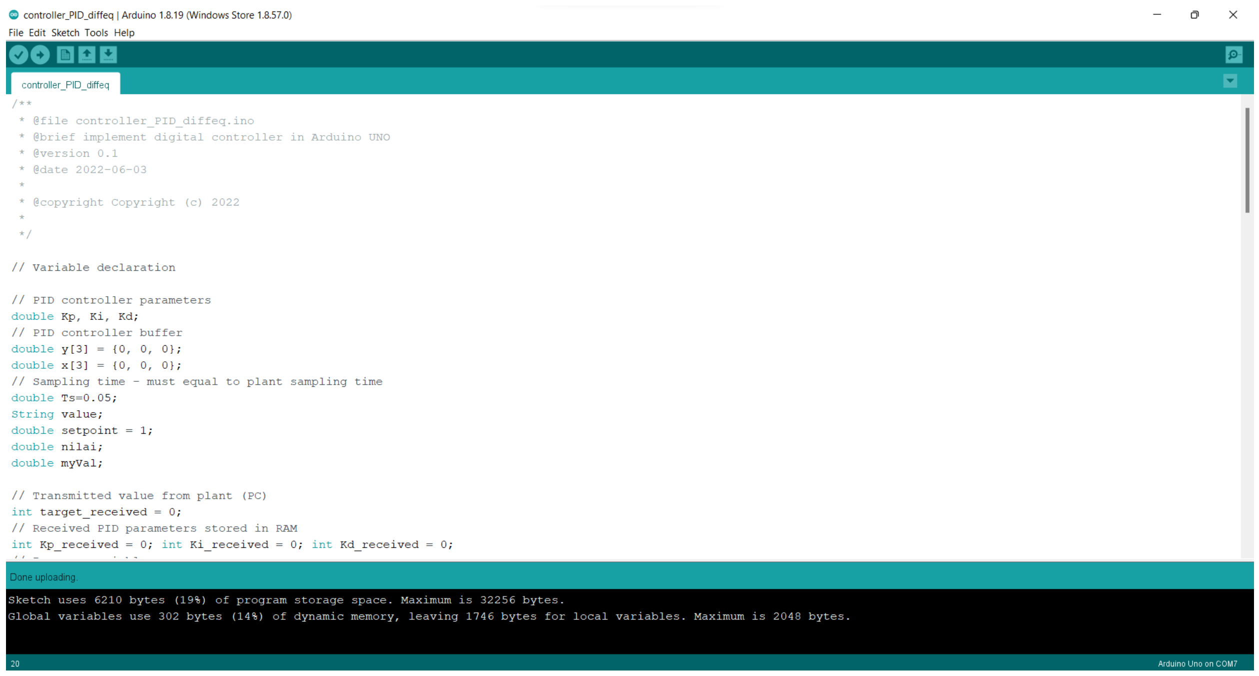
Gambar 8 Foto Perangkat (1)



Gambar 9 Foto Perangkat (2)

4 Implementasi Software

4.1 Pengendali


controller_PID_diffeq | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
controller_PID_diffeq
Done uploading.
Arduino Uno on COM7

```
/**
 * @file controller_PID_diffeq.ino
 * @brief implement digital controller in Arduino UNO
 * @version 0.1
 * @date 2022-06-03
 *
 * @copyright Copyright (c) 2022
 *
 */

// Variable declaration

// PID controller parameters
double Kp, Ki, Kd;
// PID controller buffer
double y[3] = {0, 0, 0};
double x[3] = {0, 0, 0};
// Sampling time - must equal to plant sampling time
double Ts=0.05;
String value;
double setpoint = 1;
double nilai;
double myVal;

// Transmitted value from plant (PC)
int target_received = 0;
// Received PID parameters stored in RAM
int Kp_received = 0; int Ki_received = 0; int Kd_received = 0;
```

```
Sketch uses 6210 bytes (19%) of program storage space. Maximum is 32256 bytes.
Global variables use 302 bytes (14%) of dynamic memory, leaving 1746 bytes for local variables. Maximum is 2048 bytes.
```

Gambar 10 Implementasi Software Pengendali

Program di upload ke Arduino. Perlu diperhatikan bahwa karena komunikasi serial digunakan untuk transfer antara PC dan Arduino, maka serial monitor pada Arduino IDE tidak dapat digunakan selama simulasi hardware-in-loop berjalan.

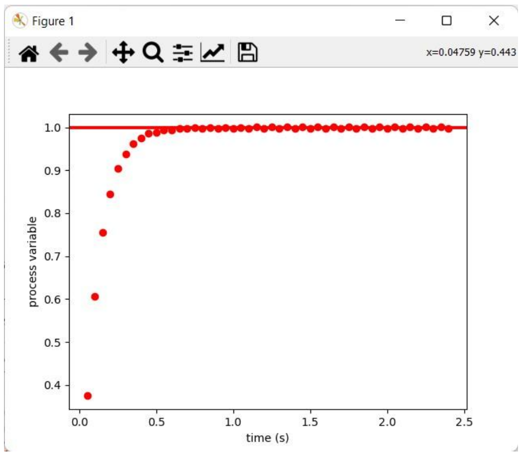
4.2 Simulator

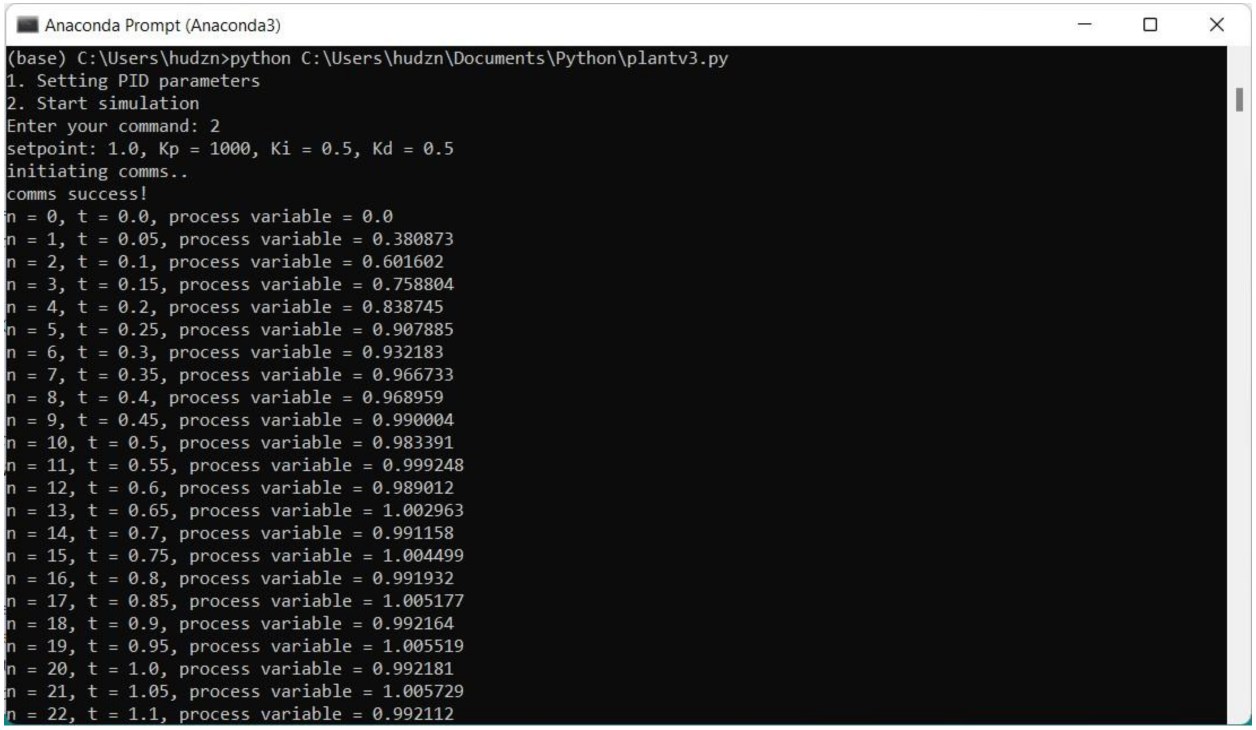
Untuk nilai parameter kendali berikut didapati output

Kp = 1000

Ki = 0.5

Kd = 0.5





```
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 1000, Ki = 0.5, Kd = 0.5
initiating comms..
comms success!
n = 0, t = 0.0, process variable = 0.0
n = 1, t = 0.05, process variable = 0.380873
n = 2, t = 0.1, process variable = 0.601602
n = 3, t = 0.15, process variable = 0.758804
n = 4, t = 0.2, process variable = 0.838745
n = 5, t = 0.25, process variable = 0.907885
n = 6, t = 0.3, process variable = 0.932183
n = 7, t = 0.35, process variable = 0.966733
n = 8, t = 0.4, process variable = 0.968959
n = 9, t = 0.45, process variable = 0.990004
n = 10, t = 0.5, process variable = 0.983391
n = 11, t = 0.55, process variable = 0.999248
n = 12, t = 0.6, process variable = 0.989012
n = 13, t = 0.65, process variable = 1.002963
n = 14, t = 0.7, process variable = 0.991158
n = 15, t = 0.75, process variable = 1.004499
n = 16, t = 0.8, process variable = 0.991932
n = 17, t = 0.85, process variable = 1.005177
n = 18, t = 0.9, process variable = 0.992164
n = 19, t = 0.95, process variable = 1.005519
n = 20, t = 1.0, process variable = 0.992181
n = 21, t = 1.05, process variable = 1.005729
n = 22, t = 1.1, process variable = 0.992112
```

Gambar 11 Implementasi Software Simulator

Dapat dilihat bahwa sistem berhasil menampilkan output hardware in loop dan melakukan logging data.

Dalam simulator dapat pula dilakukan perubahan dari parameter kendali Kp, Ki, dan Kd untuk mensimulasikan penyimpanan dari variabel kendali di RAM.

5 Pengujian

5.1 Filter Digital

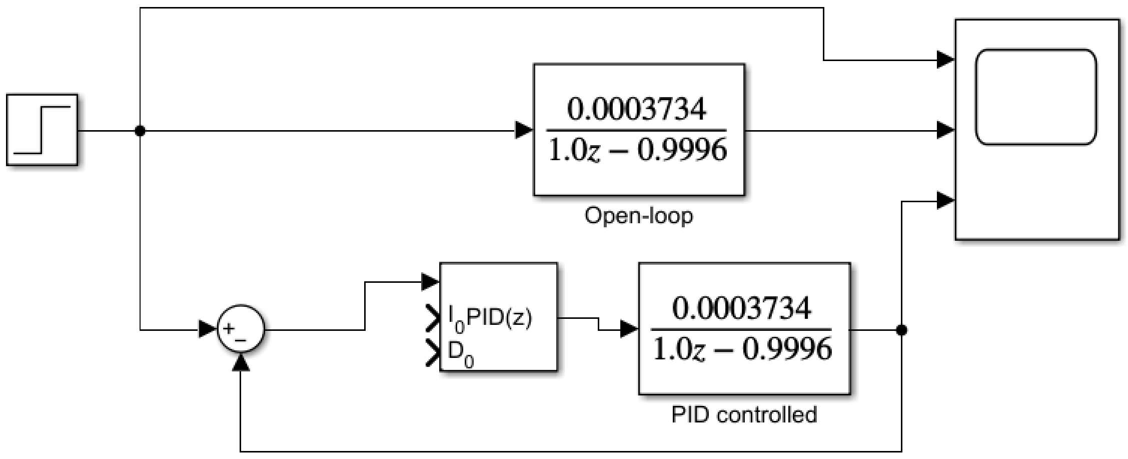
Filter digital diuji dengan software MATLAB dengan menggunakan perintah transformasi bilinear

```
>> Gcont = tf([0.9], [120.5 1])


Gcont =


     0.9
  -----------
  120.5 s + 1


Continuous-time transfer function.
```

```
>> Gdisc = c2d(Gcont, .05)


Gdisc =


  0.0003734

  ----------

  z - 0.9996


Sample time: 0.05 seconds

Discrete-time transfer function.
```

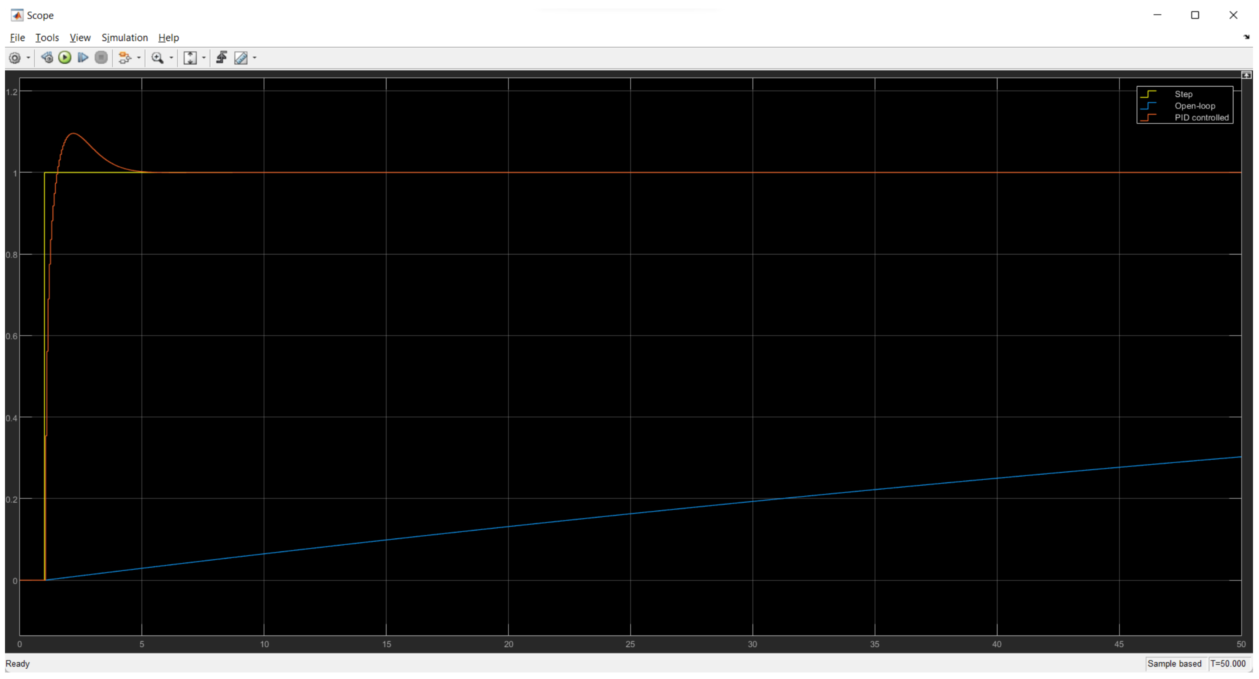
Dapat dilihat bahwa dihasilkan fungsi filter yang sama seperti sebelumnya.

Selanjutnya sistem filter digital disimulasikan dalam Simulink



Gambar 12 Diagram blok filter digital dan Respon filter digital terhadap input step

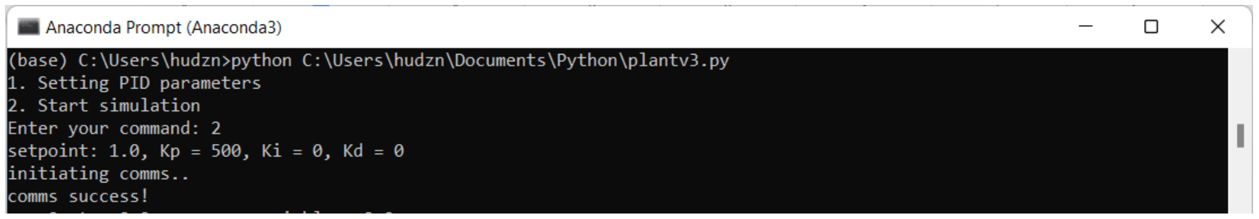
Didapati hasil simulasi sebagai berikut, bisa dilihat bahwa sistem menghasilkan tipe respons yang sesuai.



Gambar 13 Respon filter digital terhadap input step

5.2 Komunikasi Data

Dapat dilihat bahwa komunikasi berhasil dilakukan antar PC dan Arduino



```
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 500, Ki = 0, Kd = 0
initiating comms..
comms success!
```

Gambar 13 Komunikasi Data

5.3 Pengendali

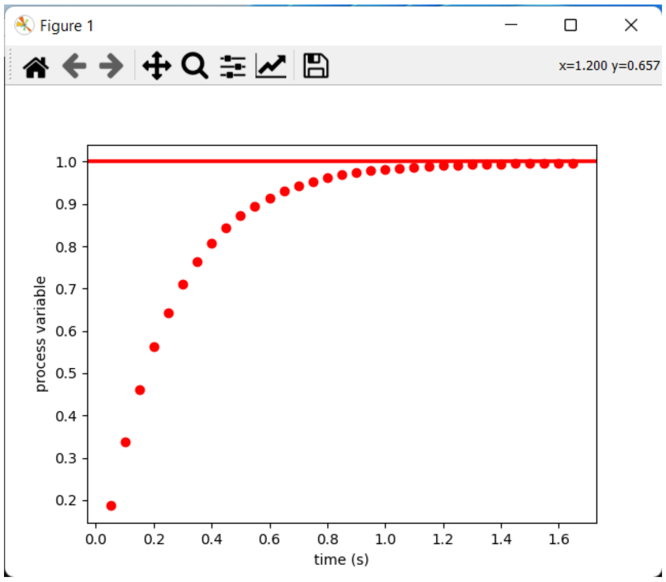
5.3.1 Pengendali P

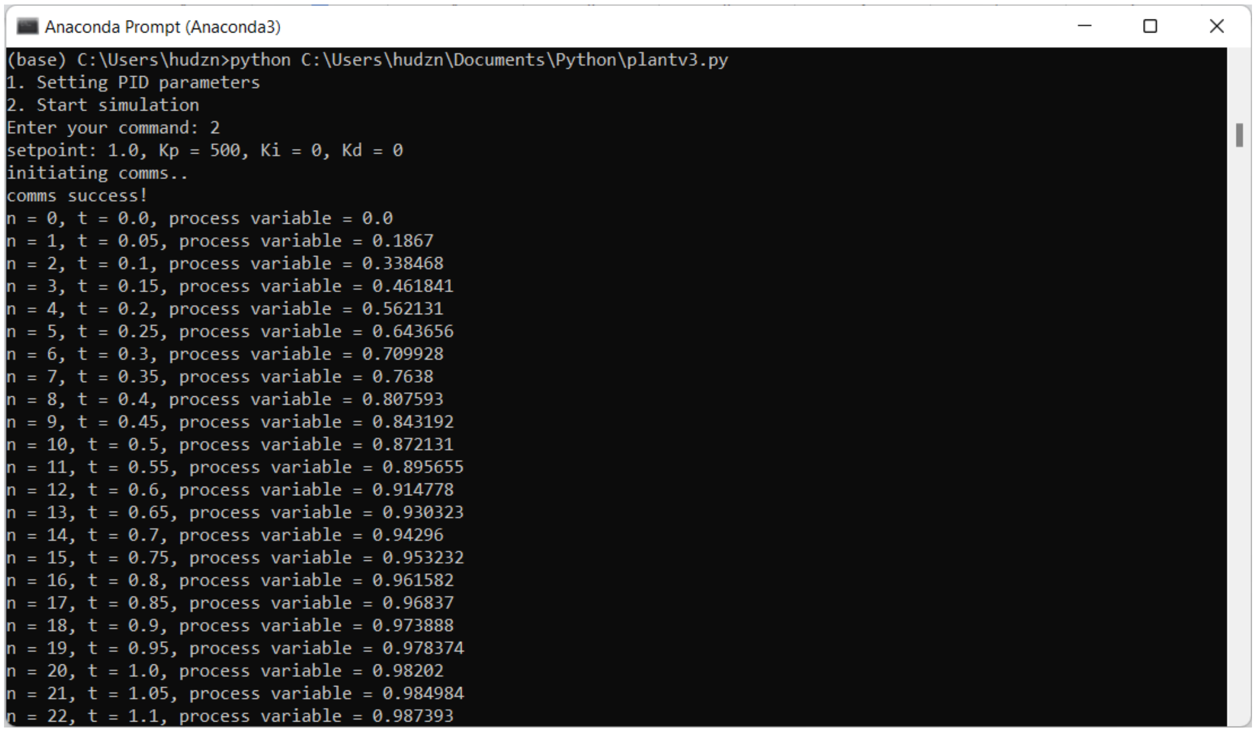
Untuk nilai parameter kendali berikut didapati output

Kp = 500

Ki = 0

Kd = 0





```
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 500, Ki = 0, Kd = 0
initiating comms..
comms success!
n = 0, t = 0.0, process variable = 0.0
n = 1, t = 0.05, process variable = 0.1867
n = 2, t = 0.1, process variable = 0.338468
n = 3, t = 0.15, process variable = 0.461841
n = 4, t = 0.2, process variable = 0.562131
n = 5, t = 0.25, process variable = 0.643656
n = 6, t = 0.3, process variable = 0.709928
n = 7, t = 0.35, process variable = 0.7638
n = 8, t = 0.4, process variable = 0.807593
n = 9, t = 0.45, process variable = 0.843192
n = 10, t = 0.5, process variable = 0.872131
n = 11, t = 0.55, process variable = 0.895655
n = 12, t = 0.6, process variable = 0.914778
n = 13, t = 0.65, process variable = 0.930323
n = 14, t = 0.7, process variable = 0.94296
n = 15, t = 0.75, process variable = 0.953232
n = 16, t = 0.8, process variable = 0.961582
n = 17, t = 0.85, process variable = 0.96837
n = 18, t = 0.9, process variable = 0.973888
n = 19, t = 0.95, process variable = 0.978374
n = 20, t = 1.0, process variable = 0.98202
n = 21, t = 1.05, process variable = 0.984984
n = 22, t = 1.1, process variable = 0.987393
```

Gambar 14 Hasil simulasi pengujian pengendali Proporsional

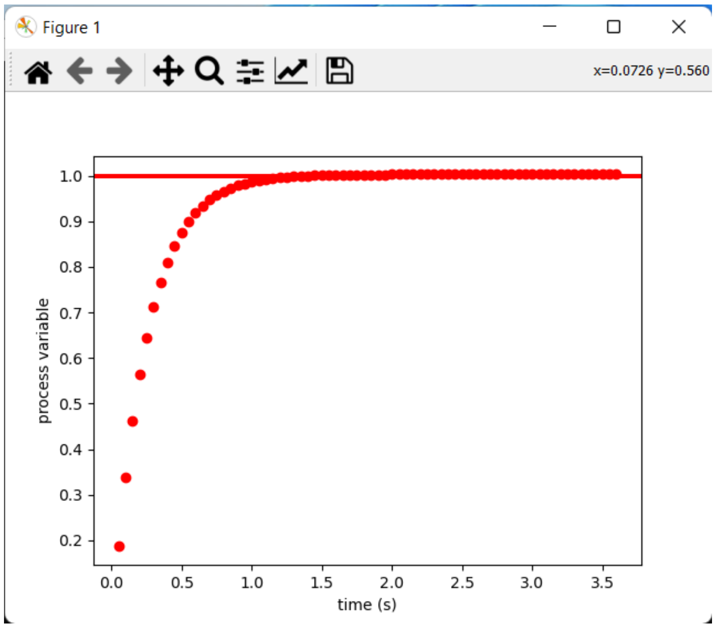
5.3.2 Pengendali PI

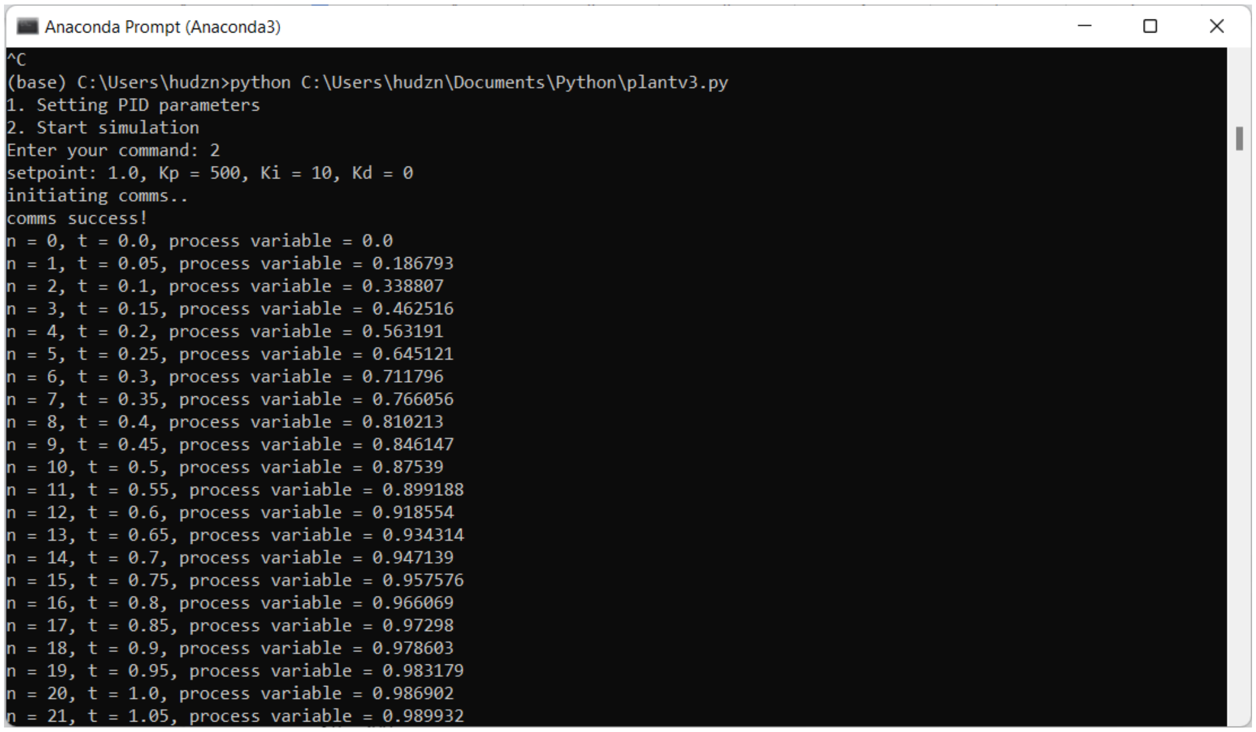
Untuk nilai parameter kendali berikut didapati output

Kp = 500

Ki = 10

Kd = 0





```
^C
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 500, Ki = 10, Kd = 0
initiating comms..
comms success!
n = 0, t = 0.0, process variable = 0.0
n = 1, t = 0.05, process variable = 0.186793
n = 2, t = 0.1, process variable = 0.338807
n = 3, t = 0.15, process variable = 0.462516
n = 4, t = 0.2, process variable = 0.563191
n = 5, t = 0.25, process variable = 0.645121
n = 6, t = 0.3, process variable = 0.711796
n = 7, t = 0.35, process variable = 0.766056
n = 8, t = 0.4, process variable = 0.810213
n = 9, t = 0.45, process variable = 0.846147
n = 10, t = 0.5, process variable = 0.87539
n = 11, t = 0.55, process variable = 0.899188
n = 12, t = 0.6, process variable = 0.918554
n = 13, t = 0.65, process variable = 0.934314
n = 14, t = 0.7, process variable = 0.947139
n = 15, t = 0.75, process variable = 0.957576
n = 16, t = 0.8, process variable = 0.966069
n = 17, t = 0.85, process variable = 0.97298
n = 18, t = 0.9, process variable = 0.978603
n = 19, t = 0.95, process variable = 0.983179
n = 20, t = 1.0, process variable = 0.986902
n = 21, t = 1.05, process variable = 0.989932
```

Gambar 15 Hasil simulasi pengujian pengendali Proporsional Integral *15*

Dapat dilihat bahwa dengan menambah komponen integral, steady state error dari kendali menjadi lebih kecil

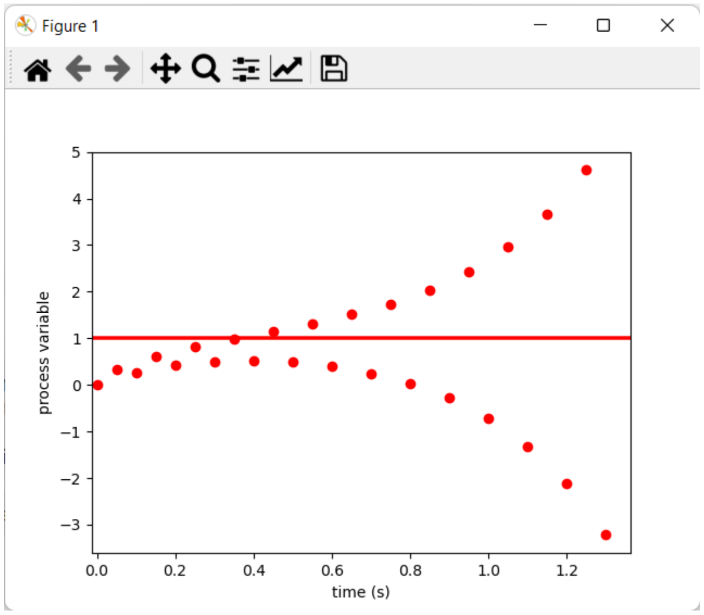
5.3.3 Pengendali PD

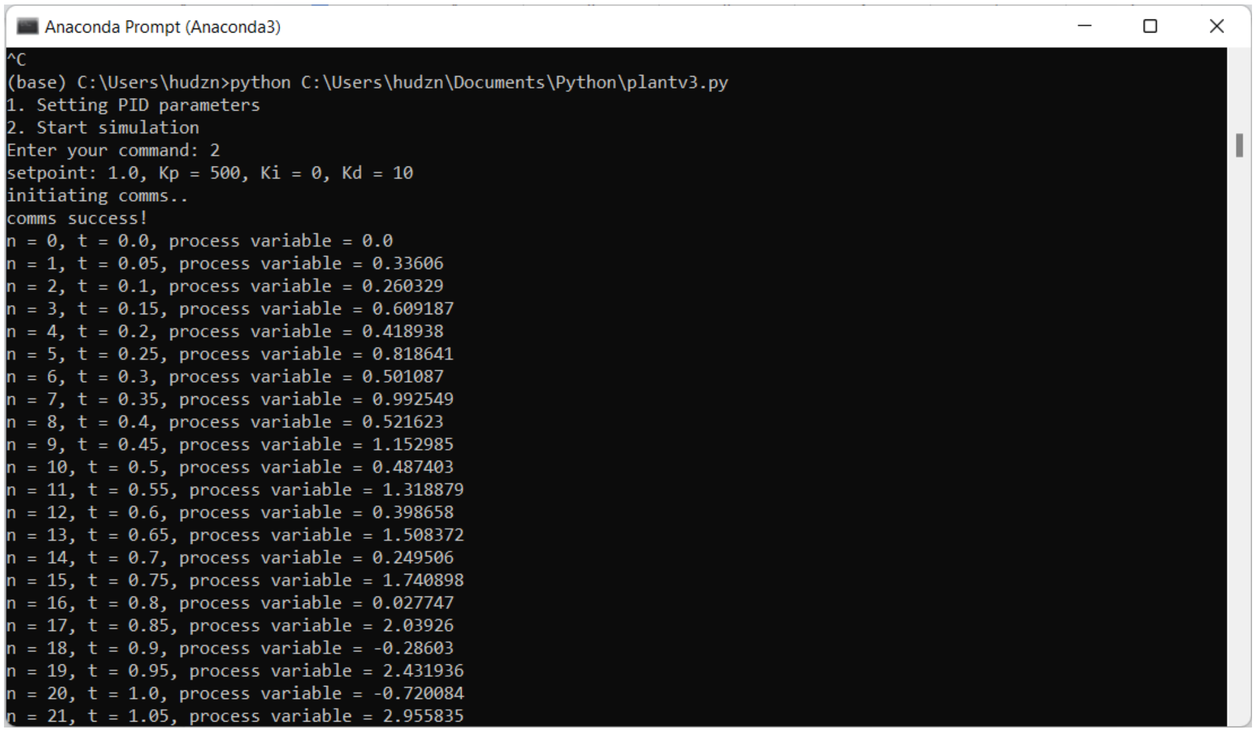
Untuk nilai parameter kendali berikut didapati output

Kp = 500

Ki = 0

Kd = 10





```
^C
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 500, Ki = 0, Kd = 10
initiating comms..
comms success!
n = 0, t = 0.0, process variable = 0.0
n = 1, t = 0.05, process variable = 0.33606
n = 2, t = 0.1, process variable = 0.260329
n = 3, t = 0.15, process variable = 0.609187
n = 4, t = 0.2, process variable = 0.418938
n = 5, t = 0.25, process variable = 0.818641
n = 6, t = 0.3, process variable = 0.501087
n = 7, t = 0.35, process variable = 0.992549
n = 8, t = 0.4, process variable = 0.521623
n = 9, t = 0.45, process variable = 1.152985
n = 10, t = 0.5, process variable = 0.487403
n = 11, t = 0.55, process variable = 1.318879
n = 12, t = 0.6, process variable = 0.398658
n = 13, t = 0.65, process variable = 1.508372
n = 14, t = 0.7, process variable = 0.249506
n = 15, t = 0.75, process variable = 1.740898
n = 16, t = 0.8, process variable = 0.027747
n = 17, t = 0.85, process variable = 2.03926
n = 18, t = 0.9, process variable = -0.28603
n = 19, t = 0.95, process variable = 2.431936
n = 20, t = 1.0, process variable = -0.720084
n = 21, t = 1.05, process variable = 2.955835
```

Gambar 15 Hasil simulasi pengujian pengendali Proporsional Derivatif

Dapat dilihat bahwa output sistem menjadi tidak stabil

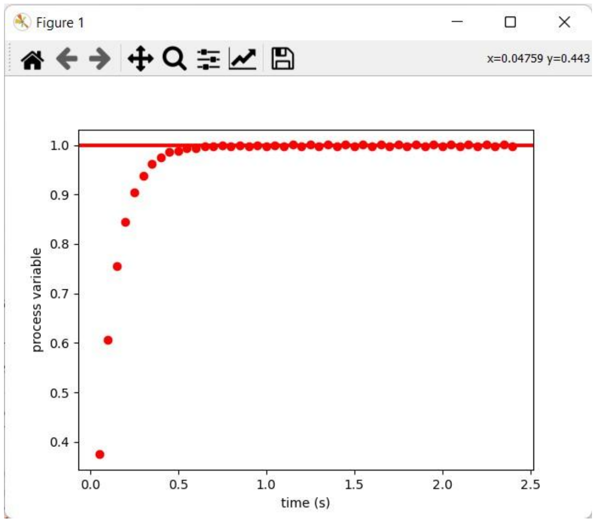
5.3.4 Pengendali PID

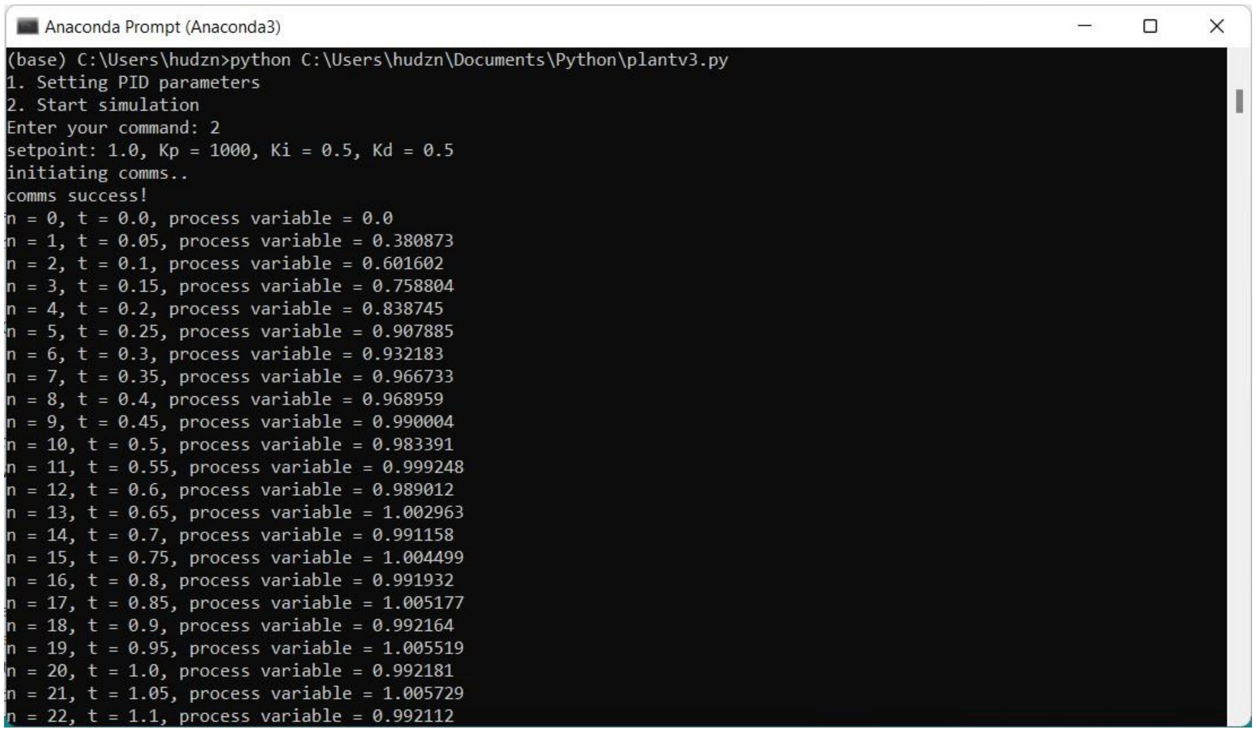
Untuk nilai parameter kendali berikut didapati output

Kp = 1000

Ki = 0.5

Kd = 0.5





```
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 1000, Ki = 0.5, Kd = 0.5
initiating comms..
comms success!
n = 0, t = 0.0, process variable = 0.0
n = 1, t = 0.05, process variable = 0.380873
n = 2, t = 0.1, process variable = 0.601602
n = 3, t = 0.15, process variable = 0.758804
n = 4, t = 0.2, process variable = 0.838745
n = 5, t = 0.25, process variable = 0.907885
n = 6, t = 0.3, process variable = 0.932183
n = 7, t = 0.35, process variable = 0.966733
n = 8, t = 0.4, process variable = 0.968959
n = 9, t = 0.45, process variable = 0.990004
n = 10, t = 0.5, process variable = 0.983391
n = 11, t = 0.55, process variable = 0.999248
n = 12, t = 0.6, process variable = 0.989012
n = 13, t = 0.65, process variable = 1.002963
n = 14, t = 0.7, process variable = 0.991158
n = 15, t = 0.75, process variable = 1.004499
n = 16, t = 0.8, process variable = 0.991932
n = 17, t = 0.85, process variable = 1.005177
n = 18, t = 0.9, process variable = 0.992164
n = 19, t = 0.95, process variable = 1.005519
n = 20, t = 1.0, process variable = 0.992181
n = 21, t = 1.05, process variable = 1.005729
n = 22, t = 1.1, process variable = 0.992112
```

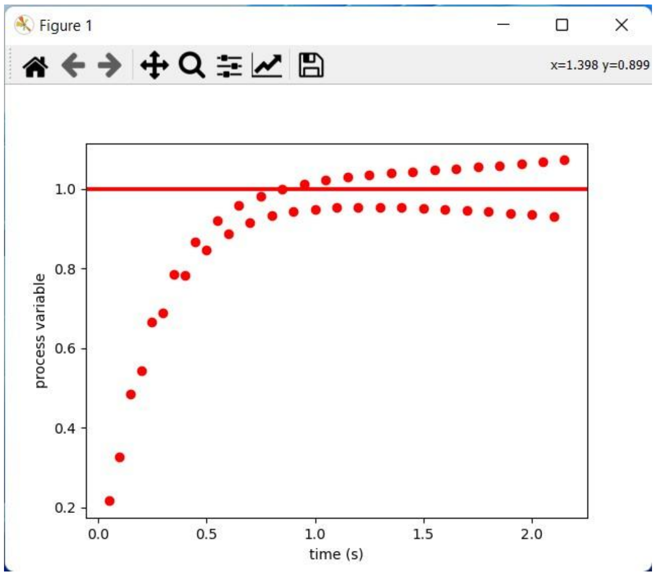
Gambar 16 Hasil simulasi pengujian pengendali Proporsional Integral Derivatif

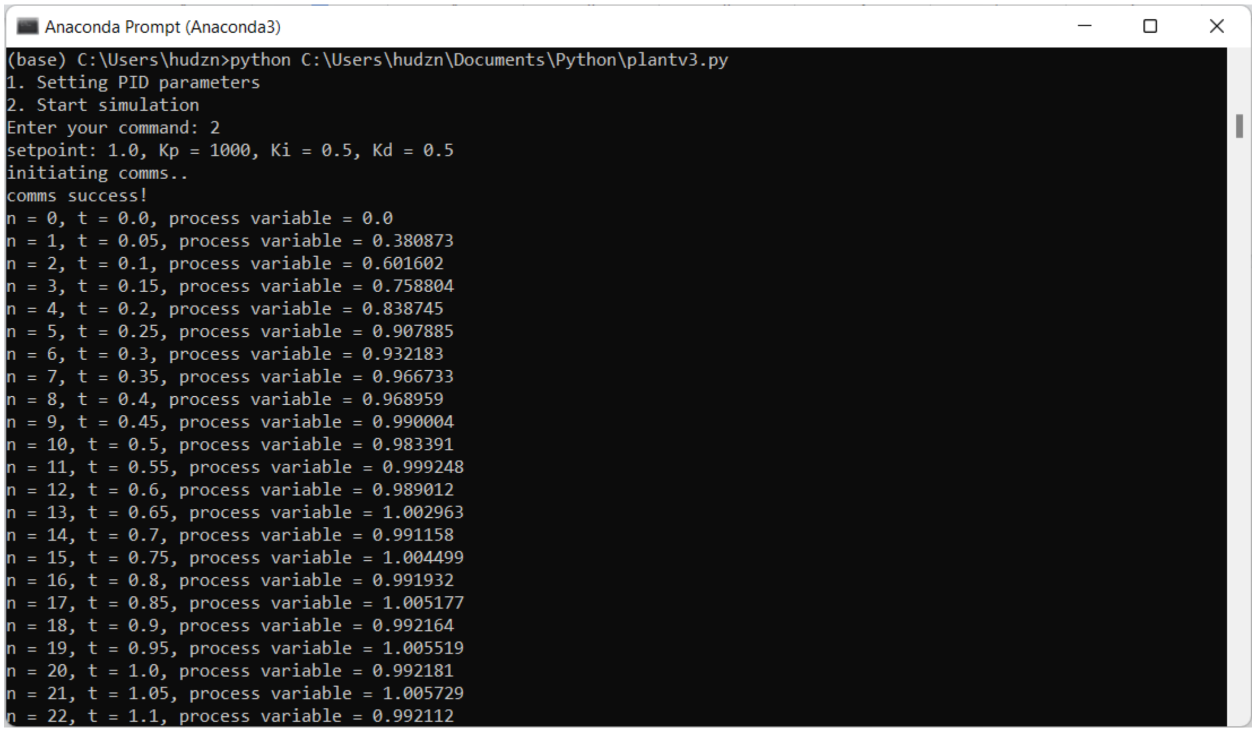
Untuk perubahan nilai parameter kendali berikut didapati output

Kp = 500

Ki = 5

Kd = 2




Anaconda Prompt (Anaconda3)

```
(base) C:\Users\hudzn>python C:\Users\hudzn\Documents\Python\plantv3.py
1. Setting PID parameters
2. Start simulation
Enter your command: 2
setpoint: 1.0, Kp = 1000, Ki = 0.5, Kd = 0.5
initiating comms..
comms success!
n = 0, t = 0.0, process variable = 0.0
n = 1, t = 0.05, process variable = 0.380873
n = 2, t = 0.1, process variable = 0.601602
n = 3, t = 0.15, process variable = 0.758804
n = 4, t = 0.2, process variable = 0.838745
n = 5, t = 0.25, process variable = 0.907885
n = 6, t = 0.3, process variable = 0.932183
n = 7, t = 0.35, process variable = 0.966733
n = 8, t = 0.4, process variable = 0.968959
n = 9, t = 0.45, process variable = 0.990004
n = 10, t = 0.5, process variable = 0.983391
n = 11, t = 0.55, process variable = 0.999248
n = 12, t = 0.6, process variable = 0.989012
n = 13, t = 0.65, process variable = 1.002963
n = 14, t = 0.7, process variable = 0.991158
n = 15, t = 0.75, process variable = 1.004499
n = 16, t = 0.8, process variable = 0.991932
n = 17, t = 0.85, process variable = 1.005177
n = 18, t = 0.9, process variable = 0.992164
n = 19, t = 0.95, process variable = 1.005519
n = 20, t = 1.0, process variable = 0.992181
n = 21, t = 1.05, process variable = 1.005729
n = 22, t = 1.1, process variable = 0.992112
```

Dapat dilihat bahwa dengan penambahan komponen kendali integral, sistem menjadi berosilasi tidak stabil.

6 Kesimpulan

Dapat dilihat bahwa sistem berhasil dimodelkan dengan filter digital. Namun perlu diperhatikan bahwa dengan melakukan sampling dapat merubah batas kestabilan sistem. Sistem awalnya stabil mutlak pada domain Laplace dapat menjadi tidak stabil akibat penambahan kendali proporsional atau integral.

Dapat dilihat dari hasil pengujian bahwa berhasil diimplementasikan kendali digital pada Arduino dan simulasi pada PC.

Dengan mengubah parameter-parameter kendali Kp, Ki, maupun Kd, dapat dilihat bahwa respons sistem berubah, dengan meningkatkan Kp, respons sistem semakin cepat dan error mengecil. Dengan meningkatkan Ki, error sistem mengecil namun mengakibatkan ketidakstabilan, dengan menambah Kd respons transient sistem dalam bentuk overshoot dapat dikurangi.

7 Referensi

[1] Waskita, et al., *Petunjuk Praktikum Sistem Mikroprosesor*, STEI ITB, Bandung, 2022.

[2] Adel S. Sedra dan Kennet C. Smith, *Microelectronic Circuits*, Oxford University Press, USA, 1997.

[3] Arief Syaichu dan Anisa Izzaty, *Petunjuk Praktikum Sistem Kendali*, LDTE, Bandung, 2022.

[4] Ogata, K, *Modern Control Engineering*, Prentice Hall, USA, 2010.

8 Lampiran

8.1 Source Code

Source code dilampirkan di dokumen dan di github (share ke waskita@gmail.com)

Source code dapat diakses di repository https://github.com/hudzaifahalfatih/HIL.git

Kode plant dan simulator (PC)

```
"""

Implement HIL plant with animated graph output

"""



# Importing Libraries

import serial

import time
import matplotlib.pyplot as plt
from tkinter import *


# Declare Arduino
# Note: larger baud rate create instability
arduino = serial.Serial(port='COM7', baudrate=9600, timeout= 1)


# Create plot figure
plt.ion()
fig = plt.figure()
x_graph = list()
y_graph = list()


# System variables
sp = 1.0     # set point
Ts = 0.05    # sampling time


# default discrete time PID controller parameters
Kp = 1000
Ki = 0.5
Kd = 0.5


# Communication variables
value_sent = 0
value = 0
receive = 0
```

```
# Validating variables
stop = 0
param_send = 0
start_plant = False


def find_between( s, first, last ):
    try:
        start = s.index( first ) + len( first )
        end = s.index( last, start )
        return s[start:end]
    except ValueError:
        return ""


def plant(Ts, sp, Kp, Ki, Kd):
    global param_send, value, value_sent, receive, stop
    # System variables
    pv = 0.0    # process variable
    n = 0       # discrete time n = t/Ts

    # Plant buffer variables
    y = [0.0, 0.0, 0.0] # plant output
    x = 0.0             # plant input

    # plant discrete time transfer function coef
    A = 0.0003734       # H(z) = A(z)/B(z)
    B = [1.0, -0.9996]

    print("initiating comms..")

    # while parameter are not yet sent, send system and controller
variables

    while(param_send == 0):

        # print("sending set point..")

        # send setpoint

        bufferwrite = "IS"+str(sp)+"F\n"

        arduino.write(bufferwrite.encode())

        time.sleep(0.01)

        # print("sending PID parameters..")

        # send Kp

        bufferwrite = "IP"+str(Kp)+"F\n"

        arduino.write(bufferwrite.encode())

        time.sleep(0.01)


        # send Ki

        bufferwrite = "IN"+str(Ki)+"F\n"

        arduino.write(bufferwrite.encode())

        time.sleep(0.01)


        # send Kd

        bufferwrite = "ID"+str(Kd)+"F\n"

        arduino.write(bufferwrite.encode())

        time.sleep(0.01)
        #ask for confirmation if all parameter is received

        bufferwrite = "ICF\n"

        arduino.write(bufferwrite.encode())


        if(arduino.in_waiting > 0) :

            strreceive_byte = arduino.readline()

            strreceive_str = strreceive_byte.decode("utf-8")

            if("transmission_success" in strreceive_str) :

                param_send = 1

                print("comms success!")




    while (not(stop)):

        if(receive):

            # print result to terminal

            print(f'n = {n}, t = {round(n*Ts, 4)}, process variable =
{y[0]}')

            # plot result to graph

            plt.scatter(n*Ts, y[0], color='r')

            plt.axhline(y = sp, color = 'r', linestyle = '-')

            plt.show()

            plt.xlabel("time (s)")

            plt.ylabel("process variable")

            plt.pause(0.0001)


            # update buffer value
        y[2] = y[1]

        y[1] = y[0]

        x = value


        # difference equation

        y[0] = A*x - B[1]*y[1]

        y[0] = round(y[0], 6)

        receive=0


        n=n+1

        value_sent=0


    while (value_sent==0) :

        bufferwrite = "IV"+str(y[0])+"F\n"

        arduino.write(bufferwrite.encode())

        if(arduino.in_waiting>0) :

            strreceive_byte = arduino.readline()

            strreceive_str = strreceive_byte.decode("utf-8")

            if("value_received" in strreceive_str) :

                value_sent=1


    bufferwrite = "ILF\n"

    arduino.write(bufferwrite.encode())

    arduino.flush()


    if (arduino.in_waiting > 0) :
            strreceive_byte = arduino.readline()

            strreceive_str = strreceive_byte.decode("utf-8")

            if("I" in strreceive_str) :

                value_str = find_between(strreceive_str,"I","F")

                value = float(value_str)

                receive=1



while (not(start_plant)):

    #print("1. Setting set point")

    print("1. Setting PID parameters")

    print("2. Start simulation")

    comm = int(input("Enter your command: "))

    if (comm == 1):

        Kp = input("New Kp: ")

        Ki = input("New Ki: ")

        Kd = input("New Kd: ")

    elif (comm == 2):

        start_plant = True

        print(f'setpoint: {sp}, Kp = {Kp}, Ki = {Ki}, Kd = {Kd}')

plant(Ts, sp, Kp, Ki, Kd)
```

Kode Controller (Arduino)

```
/**
```

```
 * @file controller_PID_diffeq.ino

 * @brief implement digital controller in Arduino UNO

 * @version 0.1

 * @date 2022-06-03

 *

 * @copyright Copyright (c) 2022

 *

 */


// Variable declaration


// PID controller parameters

double Kp, Ki, Kd;

// PID controller buffer

double y[3] = {0, 0, 0};

double x[3] = {0, 0, 0};

// Sampling time - must equal to plant sampling time

double Ts=0.05;

String value;

double setpoint = 1;

double nilai;

double myVal;


// Transmitted value from plant (PC)

int target_received = 0;

// Received PID parameters stored in RAM
```

```
int Kp_received = 0; int Ki_received = 0; int Kd_received = 0;

// Process variables

int process_start = 0;

int value_received = 0;


void setup() {

  // Begin serial communication

  Serial.begin(9600);


}


void loop() {

  // Serial.available value changed by interrupt when serial
communication is available

  if(Serial.available()) {

    // Parsing packet

    // Packet begins with indicator of data type

    // Packet ends with char 'F' indicating EOP


    if(Serial.read() == 'I') {

      // reading bufferwrite string transmitted via serial communication

      char bufferwrite = Serial.read();

      // receiving controller parameters

      // receiving setpoint

      if(bufferwrite == 'S') {

        setpoint = Serial.parseFloat(SKIP_ALL);
        if(Serial.read() == 'F') {
          target_received=1;
        }
      }
      // receiving Kp
      else if(bufferwrite == 'P') {
        Kp = Serial.parseFloat(SKIP_ALL);
        if(Serial.read() == 'F') {
          Kp_received = 1;
        }
      }
      // receiving Ki
      else if(bufferwrite == 'N') {
        Ki = Serial.parseFloat(SKIP_ALL);
        if(Serial.read() == 'F') {
          Ki_received = 1;
        }
      }
      // receiving Kd
      else if(bufferwrite == 'D') {
        Kd = Serial.parseFloat(SKIP_ALL);
        if(Serial.read() == 'F') {
          Kd_received = 1;
        }
      }
      // Finish receiving
```

```
      else if(bufferwrite == 'C') {
        if(Serial.read() == 'F') {
          if(target_received == 1 && Kp_received == 1 && Kd_received ==
1 && Ki_received == 1) {
          Serial.print("transmission_success");
          Serial.flush();
          }
        }
      }
      // Calculating controller output
      else if(bufferwrite == 'V') {
        float output = Serial.parseFloat(SKIP_ALL);
        Serial.print("value_received");
        Serial.flush();
        if(Serial.read() == 'F' && value_received == 0) {
          // Update buffer
          x[2] = x[1];
          x[1] = x[0];
          y[2] = y[1];
          y[1] = y[0];
          x[0] = setpoint - output;
          // PID controller difference equation
          y[0] = ( (2*Kp*Ts + Ki*Ts*Ts +4*Kd)*x[0] +
(2*Ki*Ts*Ts-8*Kd)*x[1] + (4*Kd-2*Kp*Ts+Ki*Ts*Ts)*x[2])/(Ts*2) + y[2];
          value_received = 1;
        }
      }
      // requesting for plant output
      else if(bufferwrite == 'L') {
        if(Serial.read() == 'F') {
          value_received = 0;
          // Send controller output to plant
          Serial.print("I");
          Serial.print(y[0], 6);
          Serial.print("F");
          Serial.flush();
        }
      }
    }
  }
  delay(5);
}
```