LAPORAN FINAL PROJECT MIKROKONTROLER
PENGUJIAN KENDALI PID ITCLAB ARDUINO DAN PYTHON DENGAN PENAMBAHAN FITUR INDIKATOR LED SERTA ANALISIS GRAFIK

Disusun Oleh :

Al Danny Rian Wibisono (20081010010)

Syahrul Hidayat (20081010076)

Humam Maulana T. R. (20081010084)

Daffa Risky Pratama (19081010052)

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2023

KATA PENGANTAR

Puji dan syukur kami haturkan kepada Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan tugas akhir mata kuliah Mikrokontroler ini dengan baik dan lancar. Laporan berjudul "Laporan Final Project Mikrokontroler" ini dibuat untuk memenuhi Ujian Akhir Semester mata kuliah Mikrokontroler yang akan dilaksanakan pada hari Selasa, 06 Juni 2023.

Laporan yang telah kami buat tentu tidak terlepas dari kesalahan dan kekurangan, baik dari segi teknis penulisan, maupun isi materi yang sedang dibahas. Oleh karena itu kami selaku penulis dari laporan ini bersedia untuk menerima berbagai komentar, saran, dan kritik dari pembaca sekalian untuk kedepannya akan kami jadikan sebagai pedoman dan evaluasi.

Demikian laporan tugas akhir ini kami buat, semoga ini dapat diterima sebagai bahan penilaian bagi kami dan semoga laporan ini dapat bermanfaat bagi siapapun yang membacanya.

Surabaya, 05 Juni 2023

Tim Penulis

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kendali PID (Proporsional-Integral-Derivative) merupakan metode umum yang digunakan dalam sistem kontrol otomatis untuk mencapai kendali yang optimal. Dalam pengembangan perangkat keras dan lunak, Arduino dan Python sering digunakan sebagai platform untuk mengimplementasikan kendali PID dan menghubungkan perangkat elektronik. Arduino adalah papan mikrokontroler yang populer dan terkenal karena kemudahan penggunaannya. Di sisi lain, Python merupakan bahasa pemrograman yang kuat dan fleksibel untuk mengembangkan aplikasi kontrol dan pemrosesan data.

Dalam implementasi kendali PID menggunakan platform Arduino dan Python, penting untuk memiliki indikator visual guna memantau dan mengevaluasi kinerja sistem. Salah satu indikator visual yang umum digunakan adalah lampu LED. Dengan penambahan fitur indikator LED pada sistem kendali PID, pengguna dapat melihat secara langsung respons sistem terhadap perubahan input atau set point yang diberikan.

Selain itu, analisis grafik juga memainkan peran penting dalam evaluasi sistem kendali PID. Grafik menyediakan informasi visual yang komprehensif tentang kinerja sistem, termasuk respons terhadap perubahan set point, over-damping, under-damping, atau overshoot. Melalui analisis grafik, masalah dapat diidentifikasi dan tuning parameter PID dapat ditingkatkan untuk mencapai kinerja yang lebih baik.

Dalam laporan ini, dilakukan pengujian kendali PID menggunakan ITCLAB (Integrated Teaching and Control Laboratory) yang mengintegrasikan Arduino dan Python. Pengujian mencakup penambahan fitur indikator LED pada sistem kendali PID serta analisis grafik untuk mengevaluasi kinerja sistem yang diimplementasikan.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah yang dapat diajukan adalah:

1. Bagaimana penerapan kendali PID menggunakan platform Arduino dan Python dengan penambahan fitur indikator LED pada sistem kendali PID?
2. Bagaimana analisis grafik dapat digunakan untuk mengevaluasi kinerja sistem kendali PID yang diimplementasikan?

1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Mengimplementasikan kendali PID menggunakan platform Arduino dan Python dengan penambahan fitur indikator LED pada sistem kendali PID.
2. Menganalisis grafik untuk mengevaluasi kinerja sistem kendali PID yang diimplementasikan.

1.4 Manfaat

Penerapan kendali PID menggunakan platform Arduino dan Python dengan penambahan fitur indikator LED serta analisis grafik memiliki manfaat sebagai berikut:

1. Meningkatkan pemahaman tentang penggunaan kendali PID dalam sistem kontrol otomatis.
2. Menunjukkan kemampuan dan keandalan penggunaan platform Arduino dan Python dalam mengembangkan sistem kendali.
3. Memberikan indikator visual yang membantu pengguna dalam memantau dan mengevaluasi kinerja sistem secara real-time.
4. Memungkinkan identifikasi masalah dan peningkatan tuning parameter PID untuk mencapai kinerja sistem yang lebih baik.
5. Memberikan kontribusi dalam pengembangan sistem kendali yang efektif dan efisien.

BAB II
TINJAUAN PUSTAKA

2.1 Kendali PID dalam Sistem Kontrol Otomatis

Kendali PID (Proporsional-Integral-Derivative) adalah metode yang umum digunakan dalam sistem kontrol otomatis. Metode ini menggabungkan tiga komponen utama, yaitu proporsional, integral, dan diferensial, untuk mencapai kendali yang optimal. Penelitian sebelumnya telah mengungkapkan keefektifan dan kehandalan kendali PID dalam berbagai aplikasi.. Metode ini dapat meningkatkan kestabilan, presisi, dan respons sistem kontrol.

2.2 Platform Arduino dan Python dalam Pengembangan Sistem Kontrol

Dalam pengembangan perangkat keras dan lunak, Arduino dan Python menjadi platform yang sering digunakan untuk mengimplementasikan sistem kontrol dan menghubungkan perangkat elektronik. Arduino, sebagai papan mikrokontroler populer, ditandai dengan kemudahan penggunaan, fleksibilitas, dan dukungan komunitas yang besar. Di sisi lain, Python merupakan bahasa pemrograman yang kuat dan fleksibel untuk mengembangkan aplikasi kontrol dan pemrosesan data.

2.3 Indikator LED sebagai Alat Visual dalam Sistem Kontrol

Indikator visual memainkan peran penting dalam pengembangan sistem kontrol. Salah satu indikator visual yang umum digunakan adalah lampu LED. Penelitian sebelumnya telah menunjukkan bahwa penambahan fitur indikator LED pada sistem kendali PID memungkinkan pengguna atau operator untuk memantau respons sistem secara langsung terhadap perubahan input atau set point yang diberikan. Hal ini membantu dalam memonitor kinerja sistem secara visual dan mendeteksi masalah potensial dengan lebih cepat.

2.4 Analisis Grafik dalam Evaluasi Sistem Kendali PID

Analisis grafik merupakan alat penting dalam evaluasi sistem kendali PID. Grafik memberikan informasi visual yang lebih komprehensif tentang kinerja sistem, termasuk respons terhadap perubahan set point, over-damping, under-damping, atau overshoot. Dengan menganalisis grafik, masalah dalam sistem kendali dapat diidentifikasi, dan parameter PID dapat ditingkatkan untuk mencapai kinerja yang lebih baik. Penelitian sebelumnya telah mengaplikasikan analisis grafik dalam peningkatan sistem kendali PID dan menunjukkan keberhasilan dalam meningkatkan performa sistem.

BAB III
PENGUJIAN DAN ANALISIS PERCOBAAN

3.1 Kode Program Arduino

Sebelum menjalankan program Python, terlebih dahulu kita siapkan program Arduino untuk diupload ke kit iTCLab. Berikut adalah kode program Arduino yang akan diupload di kit iTCLab:

```
/*
  iTCLab Internet-Based Temperature Control Lab Firmware
  Jeffrey Kantor, Initial Version
  John Hedengren, Modified
  Oct 2017
  Basuki Rahmat, Modified
  April 2022

  This firmware is loaded into the Internet-Based Temperature Control
Laboratory ESP32 to
  provide a high level interface to the Internet-Based Temperature
Control Lab. The firmware
  scans the serial port looking for case-insensitive commands:

  Q1        set Heater 1, range 0 to 100% subject to limit (0-255 int)
  Q2        set Heater 2, range 0 to 100% subject to limit (0-255 int)
  T1        get Temperature T1, returns deg C as string
  T2        get Temperature T2, returns dec C as string
  VER       get firmware version string
  X         stop, enter sleep mode

  Limits on the heater can be configured with the constants below.
*/

#include <Arduino.h>

// constants
const String vers = "1.04";     // version of this firmware
const int baud = 115200;        // serial baud rate
const char sp = ' ';            // command separator
const char nl = '\n';           // command terminator

// pin numbers corresponding to signals on the iTCLab Shield
const int pinT1   = 34;         // T1
const int pinT2   = 35;         // T2
const int pinQ1   = 32;         // Q1
const int pinQ2   = 33;         // Q2
const int pinLED  = 26;         // LED

// setting PWM properties
const int freq = 5000; //5000
const int ledChannel = 0;
const int Q1Channel = 1;
const int Q2Channel = 2;
const int resolutionLedChannel = 8; //Resolution 8, 10, 12, 15
```

```
const int resolutionQ1Channel = 8; //Resolution 8, 10, 12, 15
const int resolutionQ2Channel = 8; //Resolution 8, 10, 12, 15

const double batas_suhu_atas = 59;

// global variables
char Buffer[64];                // buffer for parsing serial input
String cmd;                     // command
double pv = 0;                  // pin value
float level;                    // LED level (0-100%)
double Q1 = 0;                  // value written to Q1 pin
double Q2 = 0;                  // value written to Q2 pin
int iwrite = 0;                 // integer value for writing
float dwrite = 0;               // float value for writing
int n = 10;                     // number of samples for each temperature
measurement

void parseSerial(void) {
  int ByteCount = Serial.readBytesUntil(nl,Buffer,sizeof(Buffer));
  String read_ = String(Buffer);
  memset(Buffer,0,sizeof(Buffer));

  // separate command from associated data
  int idx = read_.indexOf(sp);
  cmd = read_.substring(0,idx);
  cmd.trim();
  cmd.toUpperCase();

  // extract data. toInt() returns 0 on error
  String data = read_.substring(idx+1);
  data.trim();
  pv = data.toFloat();
}

// Q1_max = 100%
// Q2_max = 100%

void dispatchCommand(void) {
  if (cmd == "Q1") {
    Q1 = max(0.0, min(25.0, pv));
    iwrite = int(Q1 * 2.0); // 10.? max
    iwrite = max(0, min(255, iwrite));
    ledcWrite(Q1Channel,iwrite);
    Serial.println(Q1);
  }
  else if (cmd == "Q2") {
    Q2 = max(0.0, min(25.0, pv));
    iwrite = int(Q2 * 2.0); // 10.? max
    iwrite = max(0, min(255, iwrite));
    ledcWrite(Q2Channel,iwrite);
    Serial.println(Q2);
  }
  else if (cmd == "T1") {
    float mV = 0.0;
    float degC = 0.0;
    for (int i = 0; i < n; i++) {
        mV = (float) analogRead(pinT1) * 0.322265625;
        degC = degC + mV/10.0;
    }
    degC = degC / float(n);
```

```
    Serial.println(degC);
  }
  else if (cmd == "T2") {
    float mV = 0.0;
    float degC = 0.0;
    for (int i = 0; i < n; i++) {
         mV = (float) analogRead(pinT2) * 0.322265625;
         degC = degC + mV/10.0;
   }
    degC = degC / float(n);
    Serial.println(degC);
  }
  else if ((cmd == "V") or (cmd == "VER")) {
    Serial.println("TCLab Firmware Version " + vers);
  }
  else if (cmd == "LED") {
    level = max(0.0, min(100.0, pv));
    iwrite = int(level * 0.5);
    iwrite = max(0, min(50, iwrite));
    ledcWrite(ledChannel, iwrite);
    Serial.println(level);
  }
  else if (cmd == "X") {
    ledcWrite(Q1Channel,0);
    ledcWrite(Q2Channel,0);
    Serial.println("Stop");
  }
}

// check temperature and shut-off heaters if above high limit
void checkTemp(void) {
    float mV = (float) analogRead(pinT1) * 0.322265625;
    //float degC = (mV - 500.0)/10.0;
    float degC = mV/10.0;
    if (degC >= batas_suhu_atas) {
      Q1 = 0.0;
      Q2 = 0.0;
      ledcWrite(Q1Channel,0);
      ledcWrite(Q2Channel,0);
      //Serial.println("High Temp 1 (> batas_suhu_atas): ");
      Serial.println(degC);
    }
    mV = (float) analogRead(pinT2) * 0.322265625;
    //degC = (mV - 500.0)/10.0;
    degC = mV/10.0;
    if (degC >= batas_suhu_atas) {
      Q1 = 0.0;
      Q2 = 0.0;
      ledcWrite(Q1Channel,0);
      ledcWrite(Q2Channel,0);
      //Serial.println("High Temp 2 (> batas_suhu_atas): ");
      Serial.println(degC);
    }
}

// arduino startup
void setup() {
  //analogReference(EXTERNAL);
  Serial.begin(baud);
```

```
  while (!Serial) {
    ; // wait for serial port to connect.
  }

  // configure pinQ1 PWM functionalitites
  ledcSetup(Q1Channel, freq, resolutionQ1Channel);

  // attach the channel to the pinQ1 to be controlled
  ledcAttachPin(pinQ1, Q1Channel);

  // configure pinQ2 PWM functionalitites
  ledcSetup(Q2Channel, freq, resolutionQ2Channel);

  // attach the channel to the pinQ2 to be controlled
  ledcAttachPin(pinQ2, Q2Channel);

  // configure pinLED PWM functionalitites
  ledcSetup(ledChannel, freq, resolutionLedChannel);

  // attach the channel to the pinLED to be controlled
  ledcAttachPin(pinLED, ledChannel);

  ledcWrite(Q1Channel,0);
  ledcWrite(Q2Channel,0);
}

// arduino main event loop
void loop() {
  parseSerial();
  dispatchCommand();
  checkTemp();
}
```

Program Arduino di atas digunakan agar kit iTCLab dapat melakukan kontrol dasar terhadap sensor-sensor yang terdapat pada kit iTCLab seperti membaca sensor suhu untuk mendapatkan data suhu dan memberikan output ke pemanas.

3.2 Program Python

Selanjutnya kita menggunakan Python notebook untuk menjalankan pengujian kit iTCLab. Namun sebelum itu, kita perlu menyiapkan modul itclab.py supaya program Python notebook kita nanti bisa mengontrol kit iTCLab. Berikut adalah kode program itclab.py:

```
import sys
import time
import numpy as np
try:
    import serial
except:
    import pip
    pip.main(['install','pyserial'])
    import serial
from serial.tools import list_ports
```

```
class iTCLab(object):

    def __init__(self, port=None, baud=115200):
        port = self.findPort()
        print('Opening connection')
        self.sp = serial.Serial(port=port, baudrate=baud, timeout=2)
        self.sp.flushInput()
        self.sp.flushOutput()
        time.sleep(3)
        print('iTCLab connected via Arduino on port ' + port)

    def findPort(self):
        found = False
        for port in list(list_ports.comports()):
            # Arduino Uno
            if port[2].startswith('USB VID:PID=16D0:0613'):
                port = port[0]
                found = True
            # Arduino HDuino
            if port[2].startswith('USB VID:PID=1A86:7523'):
                port = port[0]
                found = True
            # Arduino Leonardo
            if port[2].startswith('USB VID:PID=2341:8036'):
                port = port[0]
                found = True
            # Arduino ESP32
            if port[2].startswith('USB VID:PID=10C4:EA60'):
                port = port[0]
                found = True
            # Arduino ESP32 - Tipe yg berbeda
            if port[2].startswith('USB VID:PID=1A86:55D4'):
                port = port[0]
                found = True
        if (not found):
            print('Arduino COM port not found')
            print('Please ensure that the USB cable is connected')
            print('--- Printing Serial Ports ---')
            for port in list(serial.tools.list_ports.comports()):
                print(port[0] + ' ' + port[1] + ' ' + port[2])
            print('For Windows:')
            print('  Open device manager, select "Ports (COM & LPT)"')
            print('  Look for COM port of Arduino such as COM4')
            print('For MacOS:')
            print('  Open terminal and type: ls /dev/*.')
            print('  Search for /dev/tty.usbmodem* or
/dev/tty.usbserial*. The port number is *.')
            print('For Linux')
            print('  Open terminal and type: ls /dev/tty*')
            print('  Search for /dev/ttyUSB* or /dev/ttyACM*. The port
number is *.')
            print('')
            port = input('Input port: ')
            # or hard-code it here
            #port = 'COM3' # for Windows
            #port = '/dev/tty.wchusbserial1410' # for MacOS
        return port

    def stop(self):
```

```
        return self.read('X')

    def version(self):
        return self.read('VER')

    @property
    def T1(self):
        self._T1 = float(self.read('T1'))
        return self._T1

    @property
    def T2(self):
        self._T2 = float(self.read('T2'))
        return self._T2

    def LED(self,pwm):
        pwm = max(0.0,min(100.0,pwm))/2.0
        self.write('LED',pwm)
        return pwm

    def Q1(self,pwm):
        pwm = max(0.0,min(100.0,pwm))
        self.write('Q1',pwm)
        return pwm

    def Q2(self,pwm):
        pwm = max(0.0,min(100.0,pwm))
        self.write('Q2',pwm)
        return pwm

    # save txt file with data and set point
    # t = time
    # u1,u2 = heaters
    # y1,y2 = tempeatures
    # sp1,sp2 = setpoints
    def save_txt(self,t,u1,u2,y1,y2,sp1,sp2):
        data = np.vstack((t,u1,u2,y1,y2,sp1,sp2))  # vertical stack
        data = data.T                 # transpose data
        top = 'Time (sec), Heater 1 (%), Heater 2 (%), ' \
          + 'Temperature 1 (degC), Temperature 2 (degC), ' \
          + 'Set Point 1 (degC), Set Point 2 (degC)'
        np.savetxt('data.txt',data,delimiter=',',header=top,comments='')

    def read(self,cmd):
        cmd_str = self.build_cmd_str(cmd,'')
        try:
            self.sp.write(cmd_str.encode())
            self.sp.flush()
        except Exception:
            return None
        return self.sp.readline().decode('UTF-8').replace("\r\n", "")

    def write(self,cmd,pwm):
        cmd_str = self.build_cmd_str(cmd,(pwm,))
        try:
            self.sp.write(cmd_str.encode())
            self.sp.flush()
        except:
            return None
        return self.sp.readline().decode('UTF-8').replace("\r\n", "")
```

```
    def build_cmd_str(self,cmd, args=None):
        """
        Build a command string that can be sent to the arduino.

        Input:
            cmd (str): the command to send to the arduino, must not
                contain a % character
            args (iterable): the arguments to send to the command
        """
        if args:
            args = ' '.join(map(str, args))
        else:
            args = ''
        return "{cmd} {args}\n".format(cmd=cmd, args=args)

    def close(self):
        try:
            self.sp.close()
            print('Arduino disconnected successfully')
        except:
            print('Problems disconnecting from Arduino.')
            print('Please unplug and reconnect Arduino.')
        return True
```

Setelah memiliki modul itclab.py, di folder yang sama buat Python notebook yang berisi kode untuk menjalankan percobaan yang akan dilakukan. Berikut adalah kode Python notebook yang dibuat:

```
import itclab
import numpy as np
import time
import matplotlib.pyplot as plt
from scipy.integrate import odeint

######################################################
# Use this script for evaluating model predictions   #
# and PID controller performance for the TCLab       #
# Adjust only PID and model sections                 #
######################################################

######################################################
# PID Controller                                     #
######################################################
# inputs -----------------------------------
# sp = setpoint
# pv = current temperature
# pv_last = prior temperature
# ierr = integral error
# dt = time increment between measurements
# outputs ----------------------------------
# op = output of the PID controller
# P = proportional contribution
# I = integral contribution
# D = derivative contribution
```

```
def pid(sp,pv,pv_last,ierr,dt):
    Kc   = 10.0 # K/%Heater
    tauI = 50.0 # sec
    tauD = 1.0  # sec
    # Parameters in terms of PID coefficients
    KP = Kc
    KI = Kc/tauI
    KD = Kc*tauD
    # ubias for controller (initial heater)
    op0 = 0
    # upper and lower bounds on heater level
    ophi = 100
    oplo = 0
    # calculate the error
    error = sp-pv
    # calculate the integral error
    ierr = ierr + KI * error * dt
    # calculate the measurement derivative
    dpv = (pv - pv_last) / dt
    # calculate the PID output
    P = KP * error
    I = ierr
    D = -KD * dpv
    op = op0 + P + I + D
    # implement anti-reset windup
    if op < oplo or op > ophi:
        I = I - KI * error * dt
        # clip output
        op = max(oplo,min(ophi,op))
    # return the controller output and PID terms
    return [op,P,I,D]

########################################################
# FOPDT model                                          #
########################################################
Kp = 0.5      # degC/%
tauP = 120.0  # seconds
thetaP = 10   # seconds (integer)
Tss = 23      # degC (ambient temperature)
Qss = 0       # % heater

########################################################
# Energy balance model                                 #
########################################################
def heat(x,t,Q):
    # Parameters
    Ta = 23 + 273.15   # K
    U = 10.0           # W/m^2-K
    m = 4.0/1000.0     # kg
    Cp = 0.5 * 1000.0  # J/kg-K
    A = 12.0 / 100.0**2 # Area in m^2
    alpha = 0.01       # W / % heater
    eps = 0.9          # Emissivity
    sigma = 5.67e-8    # Stefan-Boltzman

    # Temperature State
    T = x[0]

    # Nonlinear Energy Balance
    dTdt = (1.0/(m*Cp))*(U*A*(Ta-T) \
```

```
            + eps * sigma * A * (Ta**4 - T**4) \
            + alpha*Q)
    return dTdt

######################################################
# Do not adjust anything below this point            #
######################################################

# Connect to Arduino
a = itclab.iTCLab()

# Turn LED on
print('LED On')
a.LED(100)

# Run time in minutes
run_time = 15.0

# Number of cycles
loops = int(60.0*run_time)
tm = np.zeros(loops)

# Temperature
# set point (degC)
Tsp1 = np.ones(loops) * 25.0
Tsp1[10:] = 40.0
# Tsp1[180:] = 40.0
# Tsp1[300:] = 35.0
T1 = np.ones(loops) * a.T1 # measured T (degC)
error_sp = np.zeros(loops)

Tsp2 = np.ones(loops) * 23.0 # set point (degC)
T2 = np.ones(loops) * a.T2 # measured T (degC)

# Predictions
Tp = np.ones(loops) * a.T1
error_eb = np.zeros(loops)
Tpl = np.ones(loops) * a.T1
error_fopdt = np.zeros(loops)

# impulse tests (0 - 100%)
Q1 = np.ones(loops) * 0.0
Q2 = np.ones(loops) * 0.0

print('Running Main Loop. Ctrl-C to end.')
print('  Time     SP     PV     Q1   =  P   +  I  +   D')
print(('{:6.1f} {:6.2f} {:6.2f} ' + \
       '{:6.2f} {:6.2f} {:6.2f} {:6.2f}').format( \
           tm[0],Tsp1[0],T1[0], \
           Q1[0],0.0,0.0,0.0))

# Create plot
plt.figure(figsize=(20,14))
plt.ion()
plt.show()

# Main Loop
start_time = time.time()
prev_time = start_time
# Integral error
```

```
ierr = 0.0
try:
    for i in range(1,loops):
        # Sleep time
        sleep_max = 1.0
        sleep = sleep_max - (time.time() - prev_time)
        if sleep>=0.01:
            time.sleep(sleep-0.01)
        else:
            time.sleep(0.01)

        # Record time and change in time
        t = time.time()
        dt = t - prev_time
        prev_time = t
        tm[i] = t - start_time

        # Read temperatures in Kelvin
        T1[i] = a.T1
        T2[i] = a.T2

        # Simulate one time step with Energy Balance
        Tnext = odeint(heat,Tp[i-1]+273.15,[0,dt],args=(Q1[i-1],))
        Tp[i] = Tnext[1]-273.15

        # Simulate one time step with linear FOPDT model
        z = np.exp(-dt/tauP)
        Tpl[i] = (Tpl[i-1]-Tss) * z \
                 + (Q1[max(0,i-int(thetaP)-1)]-Qss)*(1-z)*Kp \
                 + Tss

        # Calculate PID output
        [Q1[i],P,ierr,D] = pid(Tsp1[i],T1[i],T1[i-1],ierr,dt)

        # Start setpoint error accumulation after 1 minute (60 seconds)
        if i>=60:
            error_eb[i] = error_eb[i-1] + abs(Tp[i]-T1[i])
            error_fopdt[i] = error_fopdt[i-1] + abs(Tpl[i]-T1[i])
            error_sp[i] = error_sp[i-1] + abs(Tsp1[i]-T1[i])

        # Write output (0-100)
        a.Q1(Q1[i])
        a.Q2(0.0)

        if(Q1[i] == 0):
            a.LED(0)
        else:
            a.LED(100)

        # Print line of data
        print(('{:6.1f} {:6.2f} {:6.2f} ' + \
              '{:6.2f} {:6.2f} {:6.2f} {:6.2f}').format( \
                  tm[i],Tsp1[i],T1[i], \
                  Q1[i],P,ierr,D))

        # Plot
        plt.clf()
        ax=plt.subplot(4,1,1)
        ax.grid()
        plt.plot(tm[0:i],T1[0:i],'r.',label=r'$T_1$ measured')
```

```
        plt.plot(tm[0:i],Tsp1[0:i],'k--',label=r'$T_1$ set point')
        plt.ylabel('Temperature (degC)')
        plt.legend(loc=2)
        ax=plt.subplot(4,1,2)
        ax.grid()
        plt.plot(tm[0:i],Q1[0:i],'b-',label=r'$Q_1$')
        plt.ylabel('Heater')
        plt.legend(loc='best')
        ax=plt.subplot(4,1,3)
        ax.grid()
        plt.plot(tm[0:i],T1[0:i],'r.',label=r'$T_1$ measured')
        plt.plot(tm[0:i],Tp[0:i],'k-',label=r'$T_1$ energy balance')
        plt.plot(tm[0:i],Tpl[0:i],'g-',label=r'$T_1$ linear model')
        plt.ylabel('Temperature (degC)')
        plt.legend(loc=2)
        ax=plt.subplot(4,1,4)
        ax.grid()
        plt.plot(tm[0:i],error_sp[0:i],'r-',label='Set Point Error')
        plt.plot(tm[0:i],error_eb[0:i],'k-',label='Energy Balance Error')
        plt.plot(tm[0:i],error_fopdt[0:i],'g-',label='Linear Model
Error')
        plt.ylabel('Cumulative Error')
        plt.legend(loc='best')
        plt.xlabel('Time (sec)')
        plt.draw()
        plt.pause(0.05)

    # Turn off heaters
    a.Q1(0)
    a.Q2(0)
    # Save figure
    plt.savefig('test_PID.png')

# Allow user to end loop with Ctrl-C
except KeyboardInterrupt:
    # Disconnect from Arduino
    a.Q1(0)
    a.Q2(0)
    print('Shutting down')
    a.close()
    plt.savefig('test_PID.png')

# Make sure serial connection still closes when there's an error
except:
    # Disconnect from Arduino
    a.Q1(0)
    a.Q2(0)
    print('Error: Shutting down')
    a.close()
    plt.savefig('test_PID.png')
    raise

a.close()
```

Pada kode di atas, percobaan yang akan kami lakukan adalah dengan menjalankan kit iTCLab selama 15 menit dengan set point 40 derajat celsius. Set point baru dimulai ketika kit iTCLab berjalan setelah 10 detik.

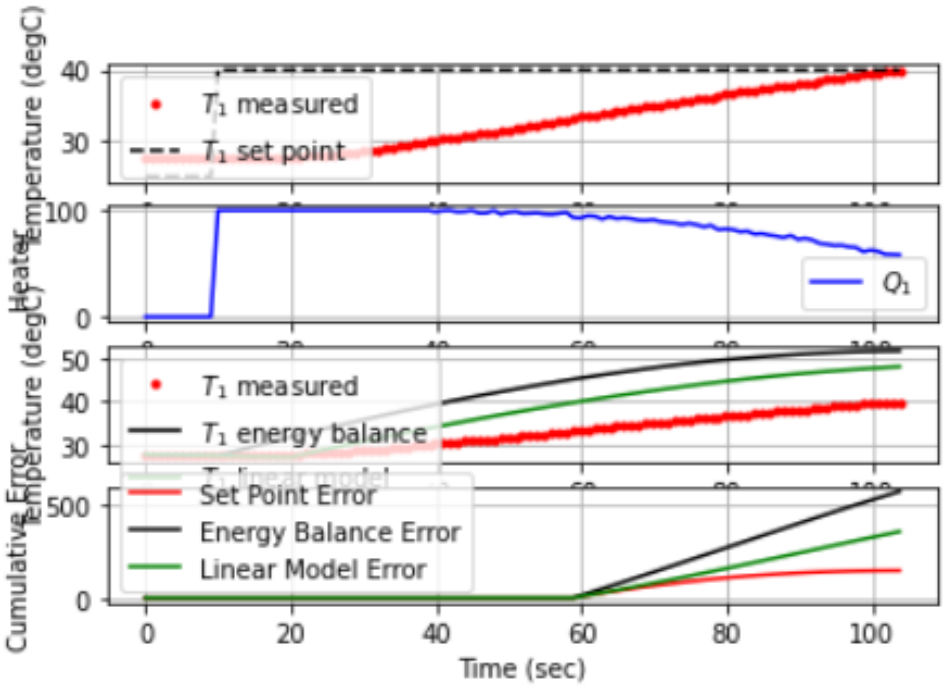
Pada kode di atas juga sudah dilakukan modifikasi di mana lampu LED akan menyala ketika pemanas juga menyala. Dengan adanya fitur ini kita bisa melihat apakah pemanas sedang aktif atau tidak. Namun perlu diingat bahwa ketika lampu indikator mati, bukan berarti suhu pemanas sudah normal lagi karena lampu hanya mengindikasikan bahwa pemanas sudah tidak aktif untuk memanaskan pemanas lagi.

3.3 Analisis Hasil

Dari percobaan yang sudah kami lakukan, kami menganalisa *rising time*, *overshoot*, *settling time*, dan *steady state error* dari grafik yang kami dapatkan.

3.3.1 Rising Time

Rising time adalah waktu yang dibutuhkan oleh sistem untuk mencapai *set point* yang ditentukan pertama kali setelah perubahan input atau *set point*. *Rising time* mengukur seberapa cepat sistem merespon dan mendekati *set point* yang diinginkan. Semakin pendek *rising time*, semakin cepat sistem mencapai *set point*. Di percobaan ini, kami mengatur *set point* di 40 derajat celsius dan berikut adalah grafik *rising time*-nya.



105.8 40.00 40.14 54.25 -1.40 57.44 -1.79

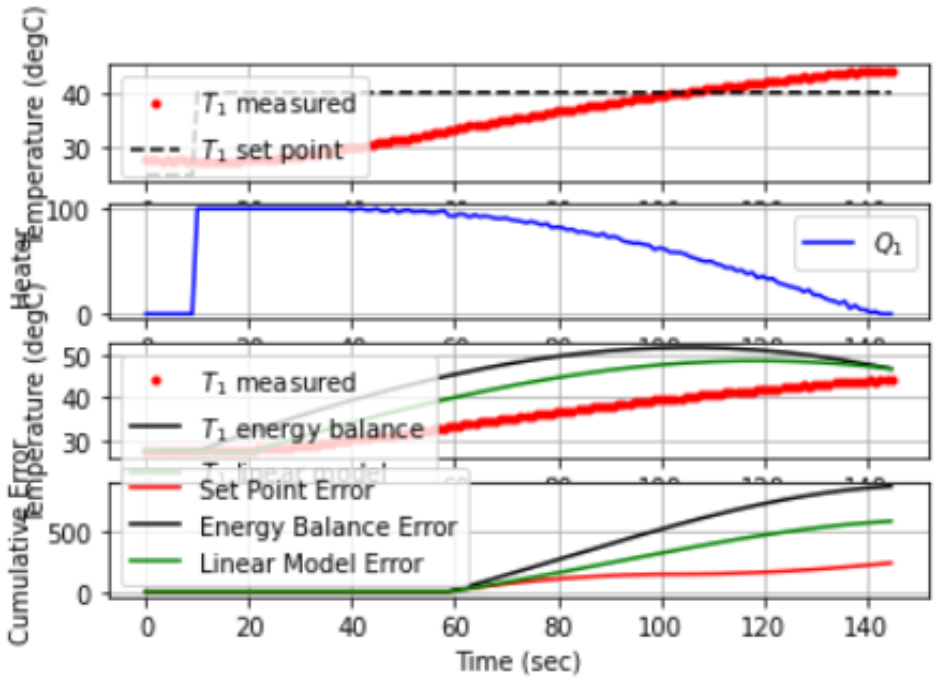
Gambar 1. Grafik Rising time

Grafik pada gambar 1 menunjukkan perubahan suhu sistem seiring waktu. Set point diatur pada suhu 40 derajat Celsius, dan grafik menggambarkan bagaimana sistem mencapai set point tersebut dari kondisi awal. *Rising time* pada grafik ini adalah waktu yang diperlukan sistem untuk mencapai suhu 40 derajat Celsius.

Berdasarkan grafik yang disajikan, dapat dilihat bahwa *set point* diatur pada detik ke-10. Sistem kemudian membutuhkan waktu sekitar 95 detik (detik ke-105) untuk mencapai suhu 40,14 derajat Celsius, yang mendekati set point yang diinginkan. Oleh karena itu, *rising time* dari kit iTCLab dalam percobaan ini adalah 95 detik.

3.3.2 Overshoot

Overshoot adalah kondisi ketika respon sistem melebihi nilai *set point* yang sudah ditentukan. Pada umumnya, overshoot dalam sistem kontrol adalah hasil dari ketidakseimbangan antara proporsional, integral, dan komponen diferensial dalam kendali PID. Ketika respons sistem bergerak melewati *set point,* ini dapat menyebabkan ketidakstabilan atau perubahan yang berlebihan dalam sistem.



146.7 40.00 44.14 0.00 -41.40 40.97 -0.81

Gambar 2. Grafik Overshoot

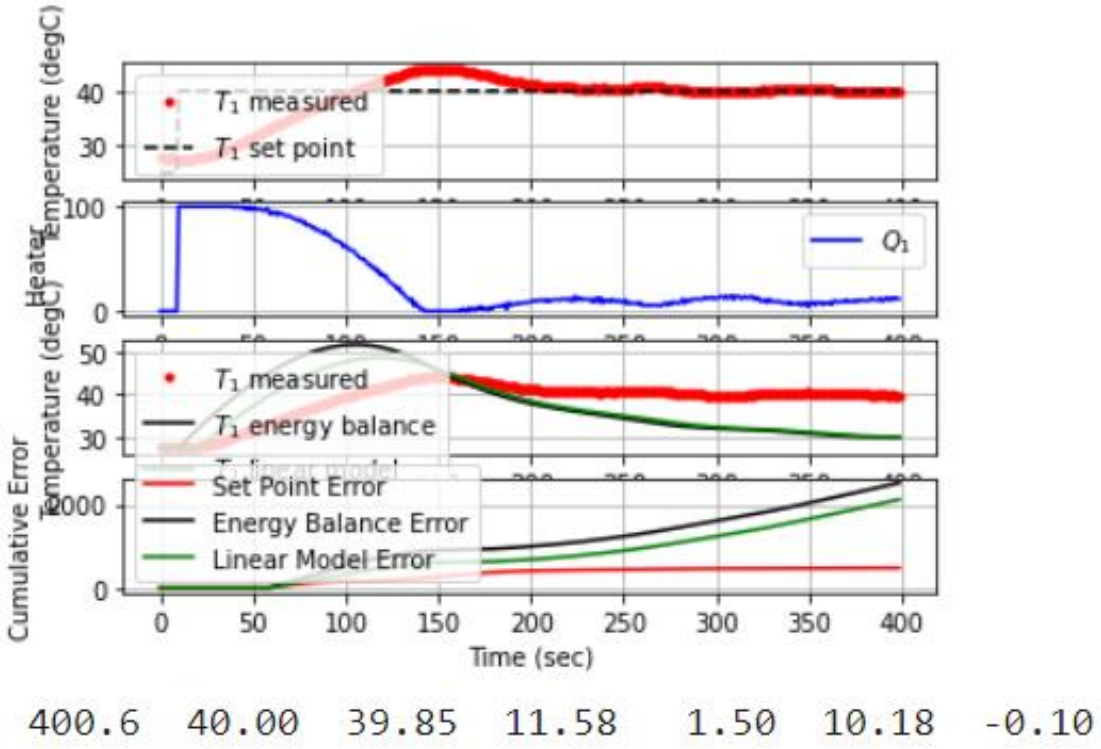
Pada gambar 2 menunjukkan bahwa respon sistem terhadap suhu melebihi nilai *set point* yang telah ditentukan, dan mencapai angka 44.14 derajat Celcius. Overshoot ini

menunjukkan bahwa sistem melewati batas yang diinginkan dan mencapai nilai yang lebih tinggi dari yang seharusnya.

Analisis lebih lanjut menyebutkan bahwa overshoot pada percobaan tersebut sebesar 10% dari set point. Hal ini berarti selisih antara nilai maksimum yang dicapai oleh respon sistem (yaitu 44.14 derajat Celsius) dan set point adalah sebesar 10% dari set point tersebut (4.14 derajat Celsius).

3.3.3 Settling Time

Settling time adalah waktu yang dibutuhkan oleh sistem untuk mencapai keadaan stabil di sekitar nilai *set point*. *Settling time* adalah salah satu parameter penting dalam evaluasi kinerja sistem kendali, karena menunjukkan seberapa cepat sistem mencapai stabilitas setelah mengalami perubahan input.



Gambar 3. Grafik Settling time

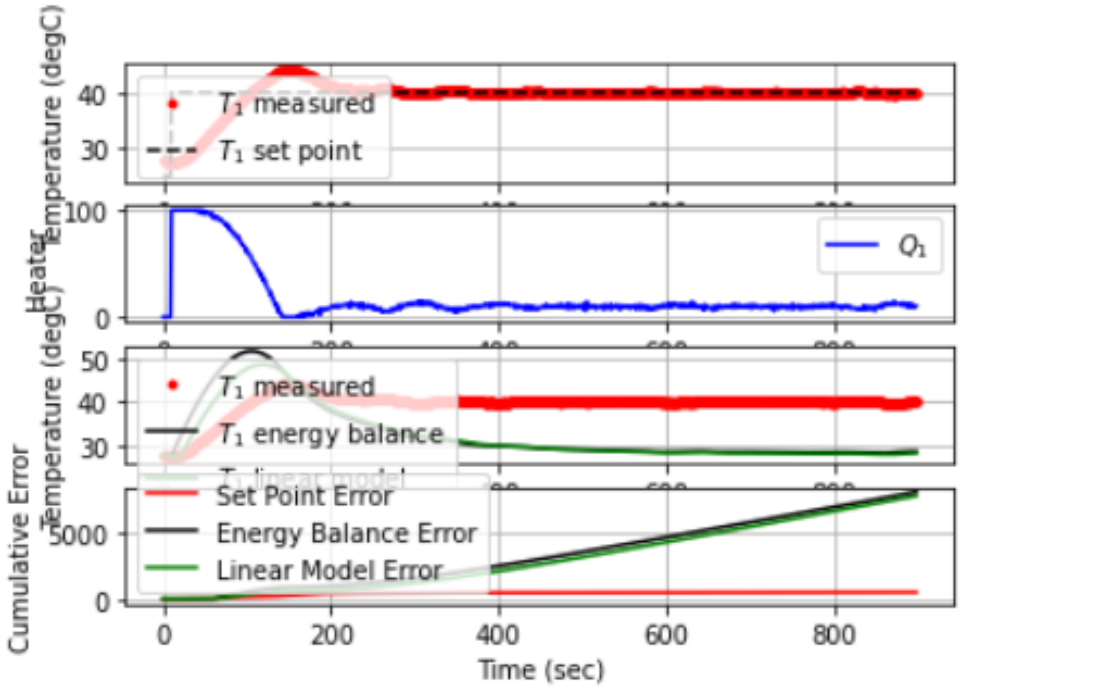
Pada gambar 3, menunjukkan grafik yang menggambarkan pergerakan sistem dari titik puncak overshoot hingga mencapai kestabilan di sekitar *set point.* Pada detik ke-208 sistem mencapai suhu 40.86 derajat Celsius, yang mendekati nilai set point yang diinginkan. Namun, pergerakan suhu masih mengalami fluktuasi dalam rentang 39.71 hingga 40.71 derajat Celsius.

Setelah detik ke-400, sistem mulai menunjukkan kestabilan dengan suhu yang lebih konsisten di sekitar set point. Oleh karena itu, dalam pengujian ini, settling time dapat

dikatakan terjadi selama 292 detik, yaitu waktu yang dibutuhkan sistem untuk mencapai kestabilan setelah melalui fase awal yang melibatkan fluktuasi suhu.

3.3.4 Steady State Error

Steady state error adalah besarnya perbedaan antara nilai yang diinginkan (*set point*) dengan nilai yang dicapai oleh sistem pada kondisi stabil. Pada gambar 4, menunjukkan grafik atau data yang menunjukkan steady state error pada sistem yang diuji.



899.4 40.00 40.04 10.35 -0.40 10.55 0.20

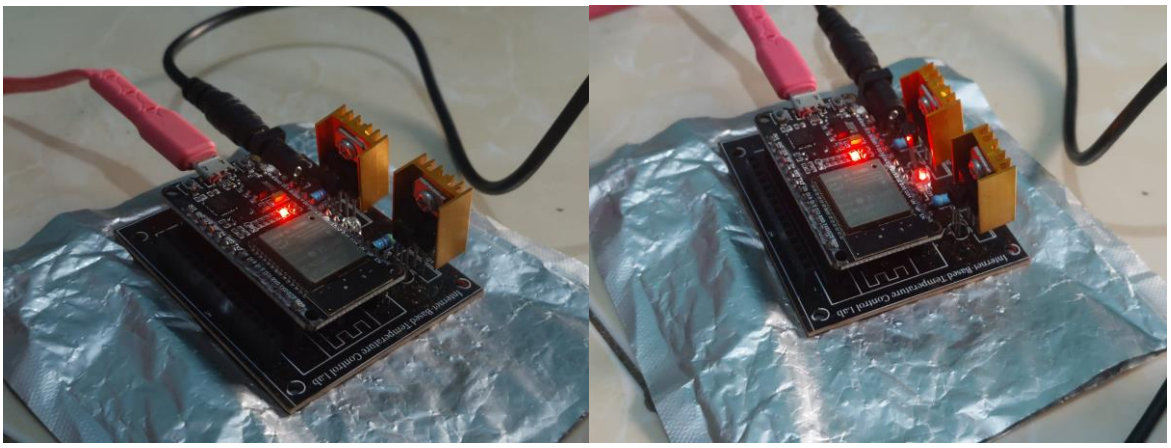
Gambar 4. Steady state error

Dari hasil pengujian yang dilakukan, ketika sistem mencapai kondisi stabil, bacaan suhu berada dalam rentang 39.93 - 40.09 derajat Celsius. Artinya, perbedaan antara *set point* yang diinginkan dan nilai yang dicapai oleh sistem adalah sebesar 0.175% - 0.225% dari nilai *set point.* Dalam hal ini, suhu adalah parameter yang diuji, namun konsep *steady state error* dapat diterapkan pada berbagai jenis sistem kontrol.

Analisis ini menunjukkan bahwa sistem yang diuji memiliki tingkat *steady state error* yang relatif kecil, yang mengindikasikan kinerja yang baik dalam mencapai *set point.* Semakin kecil nilai *steady state error*, semakin dekat nilai yang dicapai oleh sistem dengan nilai yang diinginkan. Oleh karena itu, hasil pengujian menunjukkan bahwa sistem yang diuji memiliki tingkat keakuratan dan kestabilan yang memadai dalam mencapai kondisi stabil.

3.3.5 Lampu Indikator

Fitur lampu indikator pada kit iTCLab kami tambahkan dengan tujuan untuk memberikan informasi visual yang jelas tentang status aktivitas pemanas. Dengan melihat lampu indikator, pengguna dapat dengan mudah mengetahui apakah pemanas sedang aktif atau tidak tanpa harus melakukan pengecekan langsung pada perangkat atau melalui antarmuka software.



Gambar 5. Lampu indikator kit mati (kiri), lampu indikator kit nyala (kanan)

Lampu indikator pada kit iTCLab berfungsi sebagai penanda visual yang menggambarkan kondisi pemanas seperti pada Gambar 5. Ketika pemanas sedang tidak aktif, lampu indikator akan mati, menunjukkan bahwa tidak ada aliran listrik yang mengalir ke pemanas. Sebaliknya, ketika pemanas sedang aktif, lampu indikator akan menyala, menandakan bahwa aliran listrik sedang mengalir ke pemanas. Dengan demikian, pengguna dapat dengan cepat dan mudah mengetahui apakah pemanas sedang bekerja atau tidak hanya dengan melihat lampu indikator pada kit iTCLab.

BAB IV
PENUTUP

4.1 Kesimpulan

1. Dari analisis *rising time,* ditemukan bahwa kit iTCLab membutuhkan waktu sekitar 95 detik untuk mencapai *set point* suhu 40 derajat Celsius. Hal ini menunjukkan sistem mampu merespons dan mendekati *set point* dengan waktu yang relatif cepat.
2. Terdapat overshoot sebesar 10% dari *set point* pada sistem yang diuji. Overshoot ini menunjukkan adanya ketidakseimbangan antara proporsional, integral, dan komponen diferensial dalam kendali PID yang digunakan pada kit iTCLab.
3. *Settling time* pada kit iTCLab ditemukan sekitar 292 detik. Hal ini mengindikasikan waktu yang dibutuhkan agar sistem mencapai keadaan stabil di sekitar *set point.*
4. *Steady state error* pada sistem yang diuji memiliki tingkat yang relatif kecil, yaitu sekitar 0.175% - 0.225% dari nilai *set point.* Hal ini menunjukkan kinerja yang baik dalam mencapai nilai *set point* secara akurat dan stabil.
5. Penambahan fitur lampu indikator pada kit iTCLab memberikan keuntungan dalam memberikan informasi visual yang jelas tentang status aktivitas pemanas. Lampu indikator yang mati menunjukkan ketiadaan aliran listrik ke pemanas, sedangkan lampu indikator yang menyala menandakan adanya aliran listrik.

4.2 Saran

1. Untuk mengurangi overshoot, disarankan untuk melakukan penyesuaian pada tuning parameter PID pada kit iTCLab, untuk memperoleh respon sistem yang lebih stabil.
2. Dalam pengembangan kit iTCLab, perlu dipertimbangkan peningkatan performa sistem dalam mencapai kestabilan lebih cepat.
3. Diperlukan evaluasi terhadap fitur lampu indikator pada kit iTCLab untuk memastikan kehandalan dan keakuratannya. Pengembangan lebih lanjut pada fitur ini dapat mencakup tambahan informasi visual yang lebih detail atau integrasi dengan antarmuka software untuk pemantauan jarak jauh.
4. Dalam analisis sistem kendali, penting untuk memperhatikan faktor-faktor lain seperti noise, non-linearitas, dan gangguan yang dapat mempengaruhi performa sistem. Pengujian lebih lanjut dan peningkatan metode pengendalian dapat dilakukan untuk mengatasi kendala-kendala tersebut.