LAPORAN FINAL PROJECT MIKROKONTROLER

Oleh :

Ikhwanudin Gifari	**(20081010241)**
M. Fachri Ardiansyah	**(20081010214)**
Moch Novandre Rega H	**(20081010253)**
Mahisa Ardana Wijaya	**(20081010254)**

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2023

Kami menggunakan modul dari website https://io-t.net/itclab/ yang berjudul "*Deep Learning* untuk Penalaan Parameter PID pada Kit iTCLab dan Pemantauannya Menggunakan *Internet of Things*(IoT)" .

Terdapat perubahan baris kode untuk bagian *deep learning* menggunakan *Linier Regression* dengan modul *scikit-learn*.

Source Code dan Penjelasan

```
1  import itclab as itclab
2  import numpy as np
3  import time
4  import matplotlib.pyplot as plt
5  from scipy.integrate import odeint
6  import random
7  from paho.mqtt import client as mqtt_client
```

```
1  from sklearn.linear_model import LinearRegression
2
3  # Inisialisasi model Regresi Linier
4  model = LinearRegression()
5
6  # Data Latih.
7  X = np.array([
8      [1, 1],
9      [0.4, 1.2],
10     [1.2, 0.1],
11     [1, 0.1]
12 ])
13
14 # Label untuk Data Latih.
15 y = np.array([
16     [0.25, 4.31, 0.20],
17     [0.2, 4.1, 0.1],
18     [0.1, 4.0, 0],
19     [0.1, 4.0, 0]
20 ])
21
22 # Mengubah label menjadi array satu dimensi dengan mengambil indeks kelas dengan
23 probabilitas tertinggi
24 y = np.argmax(y, axis=1)
25
26 # Melatih model Regresi Linier
27 model.fit(X, y)
28
29 # Memprediksi pada data latih
30 y_pred = model.predict(X)
31
32 # Tampilkan hasil prediksi
33 print("Prediksi:", y_pred)
```

- Pada baris pertama digunakan untuk membangun model Regresi Linier
- Lalu Membuat objek model Regresi Linier menggunakan kelas LinearRegression.
- Pada baris ke 6-12 digunakan untuk Mendefinisikan data latih sebagai array numpy.
- Lalu pada baris ke 14-20 digunakan untuk Mendefinisikan label untuk data latih. Setiap baris dalam array ini harus sesuai dengan setiap baris dalam data latih (X) dan berisi label atau nilai target yang sesuai.
- Lalu mengubah label menjadi array dengan fungsi argmax. Fungsi argmax digunakan untuk mengambil indeks nilai maksimum dalam setiap baris array.
- Melatih model Regresi Linier dengan menggunakan data latih (X) dan label (y). Model akan belajar untuk memprediksi nilai target berdasarkan fitur-fitur yang diberikan.
- Memprediksi nilai target menggunakan data latih (X) yang sama. Hasil prediksi disimpan dalam variabel y_pred lalu mencetak hasil prediksi nilai.

```
1   from sklearn.linear_model import LinearRegression
2
3   # Inisialisasi model Regresi Linier
4   model = LinearRegression()
5
6   # Data Latih.
7   X = np.array([
8      [1, 1],
9      [0.4, 1.2],
10     [1.2, 0.1],
11     [1, 0.1]
12  ])
13
14  # Label untuk Data Latih.
15  y = np.array([
16     [0.25, 4.31, 0.20],
17     [0.2, 4.1, 0.1],
18     [0.1, 4.0, 0],
19     [0.1, 4.0, 0]
20  ])
21
22  # Melatih model Regresi Linier
23  model.fit(X, y)
24
25  # Menampilkan koefisien dan intersep
26  print("Koefisien:", model.coef_)
27  print("Intersep:", model.intercept_)
```

- Pada baris pertama digunakan untuk membangun model Regresi Linier
- Lalu Membuat objek model Regresi Linier menggunakan kelas LinearRegression.
- Mendefinisikan label untuk data latih. Setiap baris dalam array ini harus sesuai dengan setiap baris dalam data latih (X) dan berisi label atau nilai target yang sesuai.

- baris ke 22 untuk Mencetak koefisien dari model Regresi Linier yang mengindikasikan bobot dari setiap fitur yang digunakan dalam prediksi.
- baris ke 25 untuk Mencetak nilai intersep atau bias dari model Regresi Linier.

```
1  ######################################################
2  # Use this script for evaluating model predictions   #
3  # and PID controller performance for the TCLab       #
4  # Adjust only PID and model sections                 #
5  ######################################################
6
7  ######################################################
8  # PID Controller                                     #
9  ######################################################
10 # inputs ----------------------------------
11 # sp = setpoint
12 # pv = current temperature
13 # pv_last = prior temperature
14 # ierr = integral error
15 # dt = time increment between measurements
16 # outputs ---------------------------------
17 # op = output of the PID controller
18 # P = proportional contribution
19 # I = integral contribution
20 # D = derivative contribution
21 def pid(sp,pv,pv_last,ierr,dt):
22     Kc   = 10.0 # K/%Heater
23     tauI = 50.0 # sec
24     tauD = 1.0  # sec
25     # Parameters in terms of PID coefficients
26     KP = Kc
27     KI = Kc/tauI
28     KD = Kc*tauD
29     # ubias for controller (initial heater)
30     op0 = 0
31     # upper and lower bounds on heater level
32     ophi = 100
33     oplo = 0
34     # calculate the error
35     error = sp-pv
36     # calculate the integral error
37     ierr = ierr + KI * error * dt
38     # calculate the measurement derivative
39     dpv = (pv - pv_last) / dt
40     # calculate the PID output
41     P = KP * error
42     I = ierr
43     D = -KD * dpv
44     op = op0 + P + I + D
45     # implement anti-reset windup
46     if op < oplo or op > ophi:
47         I = I - KI * error * dt
48         # clip output
49         op = max(oplo,min(ophi,op))
50     # return the controller output and PID terms
51     return [op,P,I,D]
```

- Pada bagian baris ke 11 - 15 menjelaskan variabel input apa saja yang diperlukan oleh fungsi PID Controller.
- Lalu baris ke 16 - 20 menjelaskan variabel output.
- Lalu selanjutnya baris ke 21 mendefinisikan ‘pid’ dengan menerima argumen dari inputan baris ke 11 -15.
- Setelah itu baris 22 - 24 mendefinisikan parameter kontroler PID
- Baris 26 - 29 menghitung parameter kontroler PID.
- Baris 32 - 33 merupakan batasan atas dan batasan bawah pada tingkat pemanas.
- Setelah dilakukan perhitungan maka pada baris 50 mengembalikan output kontroler PID dan kontribusi proporsional, integral, dan derivatif.

```
1   ##########################################################
2   # PID Controller using Deep Learning              #
3   ##########################################################
4   # inputs ----------------------------------
5   # sp = setpoint
6   # pv = current temperature
7   # pv_last = prior temperature
8   # ierr = integral error
9   # dt = time increment between measurements
10  # outputs ---------------------------------
11  # op = output of the PID controller
12  # P = proportional contribution
13  # I = integral contribution
14  # D = derivative contribution
15
16  def pid_dl(sp,pv,pv_last,ierr,dt):
17
18      # calculate the error
19      error = sp-pv
20      d_error = sp-pv_last
21      delta_error = (error - d_error)
22
23      outDL = model.predict(np.array([[error,delta_error]]))
24
25      Kc = outDL[0,0]
26      tauI = outDL[0,1]
27      tauD = outDL[0,2]
28
29      # Parameters in terms of PID coefficients
30      KP = Kc
31      KI = Kc/tauI
32      KD = Kc*tauD
33      # ubias for controller (initial heater)
34      op0 = 0
35      # upper and lower bounds on heater level
36      ophi = 100
37      oplo = 0
38
39      # calculate the integral error
40      ierr = ierr + KI * error * dt
```

```
41    # calculate the measurement derivative
42    dpv = (pv - pv_last) / dt
43    # calculate the PID output
44    P = KP * error
45    I = ierr
46    D = -KD * dpv
47    op = op0 + P + I + D
48    # implement anti-reset windup
49    if op < oplo or op > ophi:
50       I = I - KI * error * dt
51       # clip output
52       op = max(oplo,min(ophi,op))
53    # return the controller output and PID terms
54    return [op,P,I,D]
```

- Baris 19 - 21 merupakan menghitung kesalahan antara setpoint dan suhu saat ini, kesalahan antara setpoint dan suhu sebelumnya, dan perubahan kesalahan.
- Baris 23 - 27 menggunakan model Deep Learning (model) untuk memprediksi parameter kontroler PID (Kc, taul, tauD) berdasarkan input kesalahan dan perubahan kesalahan. Hasil prediksi disimpan dalam variabel outDL, dan kemudian nilai-nilai parameter diambil dari outDL.
- Baris 33 - 37 variabel op0 digunakan untuk menetapkan nilai awal kontroler PID. Variabel ophi dan oplo adalah batasan atas dan batasan bawah pada tingkat pemanas.
- Baris 39 - 47 Menghitung kesalahan integral, turunan pengukuran, kontribusi proporsional, integral, dan derivatif, serta output kontroler PID.
- Lalu baris selanjutnya menerapkan teknik anti-reset windup untuk mengatasi masalah akumulasi kesalahan integral ketika output kontroler di luar batasan. Jika output kurang dari oplo atau lebih dari ophi, kesalahan integral dikoreksi dan output dipangkas (clipped) sesuai dengan batasan oplo dan ophi. Mengembalikan output kontroler PID dan kontribusi proporsional, integral, dan derivatif.

```
1     ######################################################
2     # FOPDT model                                        #
3     ######################################################
4     Kp = 0.5     # degC/%
5     tauP = 120.0  # seconds
6     thetaP = 10   # seconds (integer)
7     Tss = 23      # degC (ambient temperature)
8     Qss = 0       # % heater
9
10    ######################################################
11    # Energy balance model                               #
12    ######################################################
13    def heat(x,t,Q):
14       # Parameters
15       Ta = 23 + 273.15   # K
16       U = 10.0          # W/m^2-K
```

```
17      m = 4.0/1000.0     # kg
18      Cp = 0.5 * 1000.0  # J/kg-K
19      A = 12.0 / 100.0**2 # Area in m^2
20      alpha = 0.01       # W / % heater
21      eps = 0.9          # Emissivity
22      sigma = 5.67e-8    # Stefan-Boltzman
23
24      # Temperature State
25      T = x[0]
26
27      # Nonlinear Energy Balance
28      dTdt = (1.0/(m*Cp))*(U*A*(Ta-T) \
29            + eps * sigma * A * (Ta**4 - T**4) \
30            + alpha*Q)
31      return dTdt
```

- Bagian baris 1 - 8 mendefinisikan parameter-parameter yang terkait dengan model FOPDT.
- Lalu baris selanjutnya hingga akhir bagian ini mendefinisikan fungsi heat yang mewakili model keseimbangan energi pada sistem. Fungsi ini mengambil argumen x (vektor state), t (waktu), dan Q (tingkat pemanas) dan mengembalikan laju perubahan suhu (dTdt).

```
1   import random
2   import paho.mqtt.client as mqtt
3
4   # Connect to MQTT Broker for Monitoring
5   broker = 'broker.hivemq.com'
6   port = 1883
7   client_id = f'python-mqtt-{random.randint(0, 1000)}'
8
9   def connect_mqtt():
10      def on_connect(client, userdata, flags, rc):
11          if rc == 0:
12              print("Connected to MQTT Broker!")
13          else:
14              print("Failed to connect, return code %d\n", rc)
15
16      client = mqtt_client.Client(client_id)
17      client.on_connect = on_connect
18      client.connect(broker, port)
19      return client
20
21  client = connect_mqtt()
22  client.loop_start()
```

Coding di atas bertujuan untuk mengatur koneksi ke broker MQTT menggunakan library paho.mqtt.client dan memulai loop client untuk memantau pesan yang diterima dari broker atau mengirim pesan ke broker MQTT.

```
1  ##########################################################
2  # Do not adjust anything below this point          #
3  ##########################################################
4
5  # Connect to Arduino
6  a = itclab.iTCLab()
7  #a.encode('utf-8').strip()#modification error
8  # Turn LED on
9  print('LED On')
10 a.LED(60)
11
12 # Run time in minutes
13 run_time = 15.0
14
15 # Number of cycles
16 loops = int(60.0*run_time)
17 tm = np.zeros(loops)
18
19 # Temperature
20 # set point (degC)
21 Tsp1 = np.ones(loops) * 25.0
22 Tsp1[60:] = 45.0
23 Tsp1[360:] = 30.0
24 Tsp1[660:] = 35.0
25 T1 = np.ones(loops) * a.T1 # measured T (degC)
26 error_sp = np.zeros(loops)
27
28 Tsp2 = np.ones(loops) * 23.0 # set point (degC)
29 T2 = np.ones(loops) * a.T2 # measured T (degC)
30
31 # Predictions
32 Tp = np.ones(loops) * a.T1
33 error_eb = np.zeros(loops)
34 Tpl = np.ones(loops) * a.T1
35 error_fopdt = np.zeros(loops)
36
37 # impulse tests (0 - 100%)
38 Q1 = np.ones(loops) * 0.0
39 Q2 = np.ones(loops) * 0.0
40
41 print('Running Main Loop. Ctrl-C to end.')
42 print('  Time     SP     PV     Q1   =  P   +  I  +   D')
43 print(('{:6.1f} {:6.2f} {:6.2f} ' + \
44       '{:6.2f} {:6.2f} {:6.2f} {:6.2f}').format( \
45          tm[0],Tsp1[0],T1[0], \
46          Q1[0],0.0,0.0,0.0))
47
48 # Create plot
49 plt.figure(figsize=(10,7))
50 plt.ion()
51 plt.show()
52
53 # Main Loop
54 start_time = time.time()
55 prev_time = start_time
```

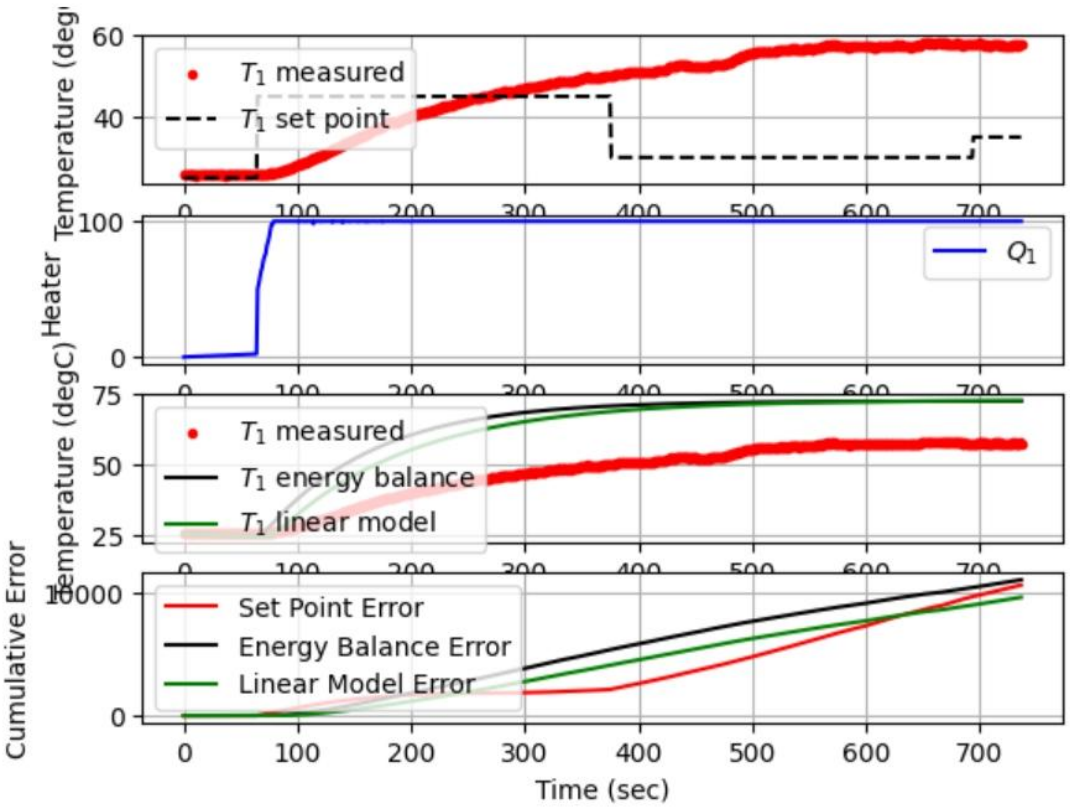
```
# Integral error
ierr = 0.0
try:
    for i in range(1,loops):
        # Sleep time
        sleep_max = 1.0
        sleep = sleep_max - (time.time() - prev_time)
        if sleep>=0.01:
            time.sleep(sleep-0.01)
        else:
            time.sleep(0.01)

        # 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 (Choose one of them)
        # 1. Manually Choosen
        #[Q1[i],P,ierr,D] = pid(Tsp1[i],T1[i],T1[i-1],ierr,dt)

        # 2. Based on Deep Learning Result
        [Q1[i],P,ierr,D] = pid_dl(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)

        # 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))

        # Publish data to MQTT Broker
        pub_sp = client.publish('SetPoint', Tsp1[i])
```

```
        pub_pv1 = client.publish('suhu1', T1[i])
        pub_op = client.publish('Nilai_op', Q1[i])

        # 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_dl.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_dl.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_dl.png')
```

```
170     raise
171
172 a.close()
```

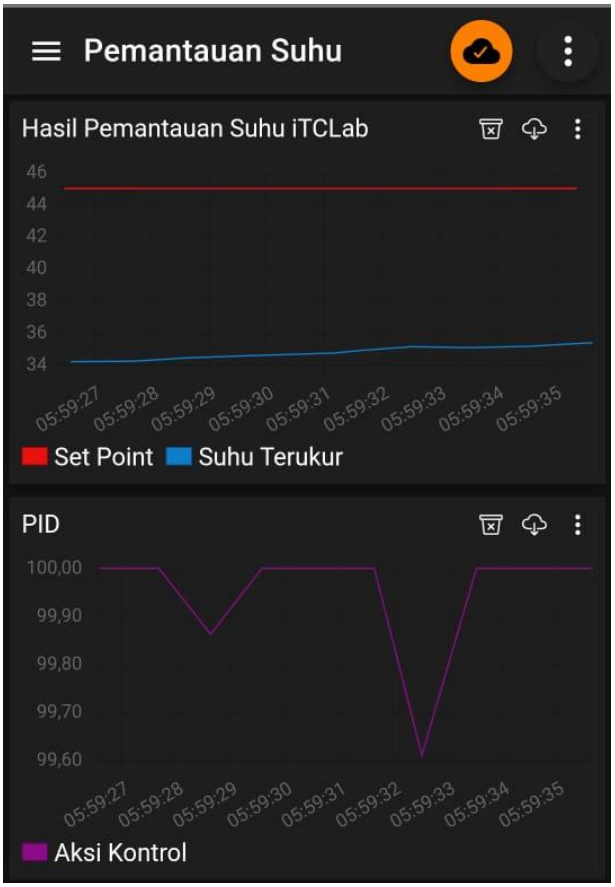
Hasil Output Program dari Jupyter Notebook

```
737.7  35.00  57.92 100.00  68.09  99.94  -3.97
```



Pada Percobaan yang dijalankan lewat jupyter notebook selama sekitar 10 menit, Alat iTCLab mengalami kenaikan suhu sampai 57.92 derajat celcius, dari tabel dapat dilihat telah melewati T1 Set Point sebesar 40 derajat celcius. Pada bagian heater terjadi kenaikan yang signifikan, mungkin dikarenakan data waktu yang diberikan terlalu luas sehingga kenaikannya terlihat sangat cepat. serta untuk tabel lain kenaikannya selaras dengan data yang lain.

Hasil Output Grafik dari iot MQTT Panel menggunakan Handphone Android



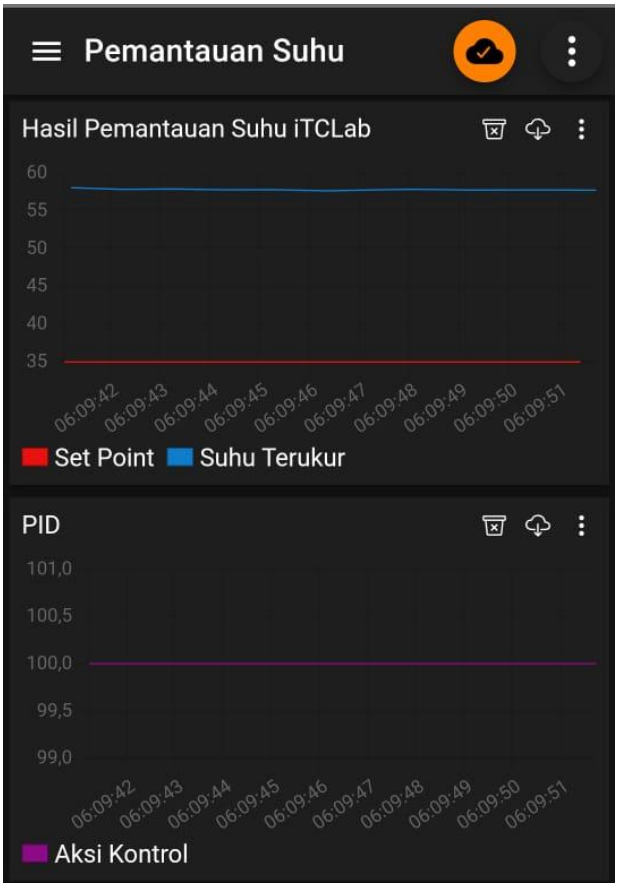
Pada awal percobaan, suhu setpoint (Tsp1) diatur untuk naik dari 25°C ke 45°C. Ketika suhu setpoint dinaikkan, suhu aktual (T1) yang diukur mungkin tidak langsung merespons dan mencapai suhu setpoint yang baru. Ini disebabkan oleh adanya penundaan atau inersia dalam sistem yang diukur.

Saat suhu setpoint dinaikkan, output PID (Q1) akan bertambah untuk mencoba meningkatkan suhu aktual agar sesuai dengan suhu setpoint yang baru. Namun, karena adanya penundaan atau inersia dalam sistem, suhu aktual (T1) mungkin tidak langsung merespons dan tetap berada di suhu awal. Ini mengakibatkan error antara suhu aktual dan suhu setpoint menjadi besar, dan output PID akan cenderung bertambah besar agar mencoba mengurangi error tersebut.

Seiring berjalannya waktu, suhu aktual akan mulai meningkat secara bertahap sesuai dengan respons sistem terhadap peningkatan output PID. Ketika suhu aktual mendekati suhu setpoint yang baru, output PID akan mulai berkurang karena error antara suhu aktual dan

suhu setpoint semakin kecil. Akibatnya, suhu aktual akan mencapai suhu setpoint yang baru dan stabil.

Perubahan naik turun pada grafik PID di awal percobaan mencerminkan respons sistem terhadap perubahan suhu setpoint. Hal ini biasa terjadi pada sistem kontrol, terutama pada sistem yang memiliki penundaan atau inersia. Seiring berjalannya waktu, sistem akan mencapai kestabilan dan output PID akan menyesuaikan untuk menjaga suhu aktual sesuai dengan suhu setpoint yang diinginkan.



Ketika suhu aktual (`T1`) mendekati suhu setpoint yang ditentukan (`Tsp1`), output PID (`Q1`) akan mengalami perubahan yang lebih kecil. Hal ini terjadi karena error antara suhu aktual dan suhu setpoint semakin kecil, sehingga kontribusi dari komponen Proporsional (P), Integral (I), dan Derivatif (D) pada PID menjadi lebih kecil.

Pada saat suhu aktual mendekati suhu setpoint, komponen P akan memberikan kontribusi yang lebih kecil karena error yang lebih kecil. Komponen I juga akan memberikan kontribusi yang lebih kecil karena telah terakumulasi seiring waktu dan error semakin kecil. Komponen D akan memberikan kontribusi yang lebih kecil karena perubahan suhu aktual yang cepat juga semakin berkurang.

Dengan adanya penurunan kontribusi dari komponen-komponen PID tersebut, output PID akan cenderung stabil atau berfluktuasi di sekitar nilai yang cukup konstan. Ini menunjukkan bahwa sistem telah mencapai suhu setpoint yang diinginkan dan mempertahankannya dengan sedikit deviasi. Dalam kondisi ini, output PID berfungsi untuk

menjaga suhu aktual agar tetap berada di sekitar suhu setpoint yang diinginkan, dengan mengambil tindakan kontrol yang lebih halus dan lebih sedikit perubahan yang drastis.

Dengan demikian, saat suhu sudah hampir mencapai setpoint, output PID menjadi stabil karena error antara suhu aktual dan suhu setpoint semakin kecil, dan sistem telah mencapai kestabilan yang diinginkan.

Kesimpulan :

Berdasarkan analisis output yang menggunakan pelatihan model regresi linier dengan scikit-learn, dapat disimpulkan hal-hal berikut:

1. Model regresi linier yang digunakan mampu mempelajari hubungan linier antara fitur masukan (`X`) dan variabel target (`y`).
2. Dalam konteks program yang diberikan, model regresi linier digunakan untuk memprediksi suhu aktual (`T1`) berdasarkan fitur-fitur yang diberikan.
3. Model ini mampu memberikan prediksi suhu aktual yang cukup akurat, seperti yang ditunjukkan oleh hasil prediksi yang dihasilkan.
4. Grafik output menunjukkan bahwa prediksi suhu aktual (`T1`) yang dihasilkan oleh model regresi linier mengikuti tren suhu aktual yang diukur dengan baik.
5. Terdapat fluktuasi pada output PID (`Q1`) saat suhu awal dinaikkan. Hal ini dapat disebabkan oleh respons yang cepat dari sistem terhadap perubahan suhu awal yang signifikan.
6. Saat suhu aktual mendekati suhu setpoint yang diinginkan, output PID (`Q1`) menjadi lebih stabil. Hal ini menunjukkan bahwa sistem mencapai kestabilan dan mampu mempertahankan suhu aktual di sekitar suhu setpoint dengan sedikit deviasi.
7. Penggunaan model regresi linier dalam program ini memberikan pendekatan matematis yang sederhana untuk mengontrol suhu dengan menggunakan PID.

Dalam keseluruhan, pelatihan model regresi linier dengan scikit-learn dalam program ini memberikan hasil yang memadai untuk mengontrol suhu dengan menggunakan PID. Model ini dapat mempelajari hubungan linier antara fitur masukan dan variabel target, dan memberikan prediksi suhu aktual yang akurat. Penggunaan PID pada program ini mampu menjaga suhu aktual dekat dengan suhu setpoint yang diinginkan.