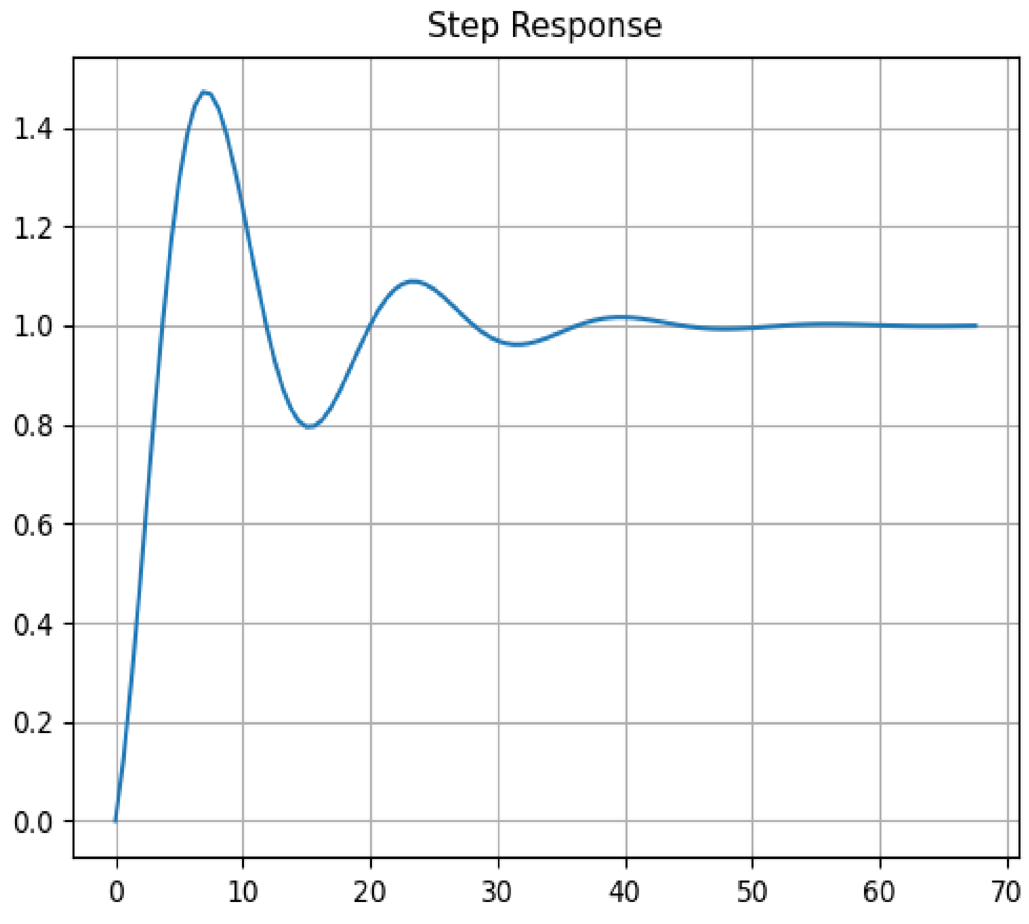
Control System

Shamim Al Mamun

Student ID:251244

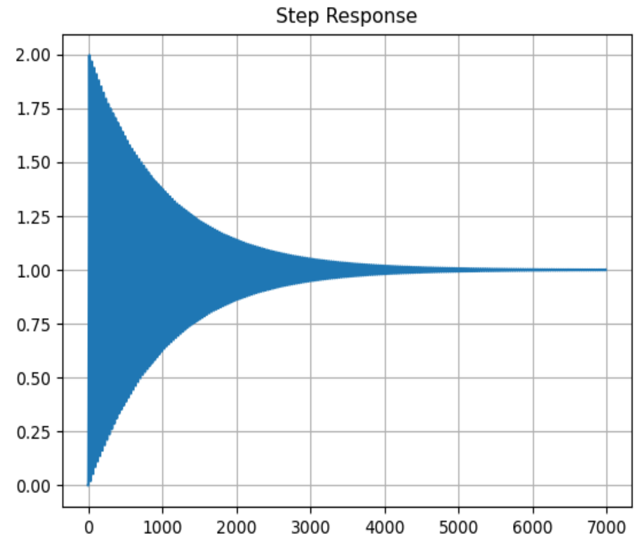
Frequency Response of Air Heater Control system

```
import numpy as np
import matplotlib.pyplot as plt
import control
# Process Parameters
Kh = 3.5
theta_t = 22
theta_d = 2

# Transfer Function Process
num_p = np. array ([Kh])
den_p = np. array ([theta_t , 1])
Hp1 = control.tf(num_p , den_p)


# Transfer Function PI Controller
Kp = 1
Ti = 1
num_c = np.array ([Kp*Ti, Kp])
den_c = np.array ([Ti , 0])
Hc = control.tf(num_c, den_c)
print ('Hc(s) =', Hc)

# The Loop Transfer function
L = control.series(Hc, Hp1)
print ('L(s) =', L)

# Tracking transfer function
T = control.feedback(L,1)
print ('T(s) =', T)

#Step Response Feedback System (Tracking System)
t, y = control.step_response(T)
plt.figure()
plt.plot(t,y)
plt.title("Step Response")
plt.grid()
```

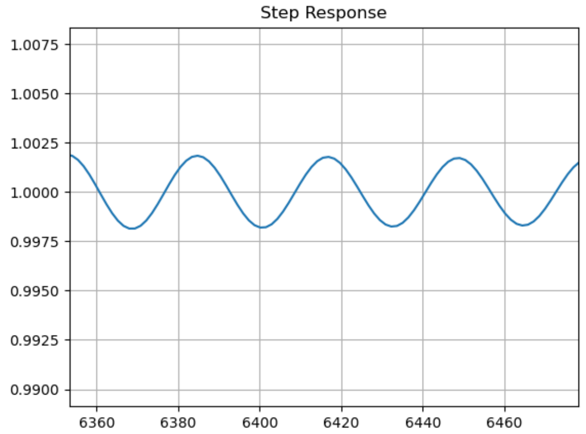


Ziegler–Nichols Frequency Response method to Find PI(D) parameters in Air heater system

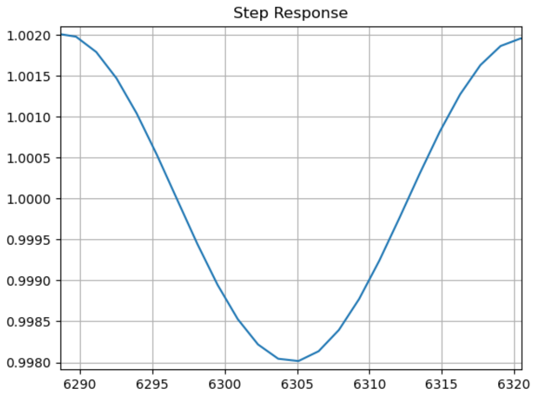
```python
import numpy as np
import matplotlib.pyplot as plt
import control
# Process Parameters
Kh = 3.5
theta_t = 22
theta_d = 2

# Transfer Function Process
num_p = np. array ([Kh])
den_p = np. array ([theta_t , 1])
Hp1 = control.tf(num_p , den_p)

N = 5 # Time Delay - Order of the Approximation
[num_pade,den_pade] = control.pade(theta_d, N)
Hp_pade = control.tf(num_pade,den_pade);
#print ('Hp_pade(s) =', Hp_pade)
Hp = control.series(Hp1, Hp_pade);

# Transfer Function PI Controller
Kp = 0.22
Ti = 0.9
num_c = np.array ([Kp*Ti, Kp])
den_c = np.array ([Ti , 0])
Hc = control.tf(num_c, den_c)
print ('Hc(s) =', Hc)

# The Loop Transfer function
L = control.series(Hc, Hp)
print ('L(s) =', L)

# Tracking transfer function
T = control.feedback(L,1)
print ('T(s) =', T)

#Step Response Feedback System (Tracking System)
t, y = control.step_response(T)
plt.figure()
plt.plot(t,y)
plt.title("Step Response")
plt.grid()

```



Zoom



Zoom



Tc=32; Kc=0.22

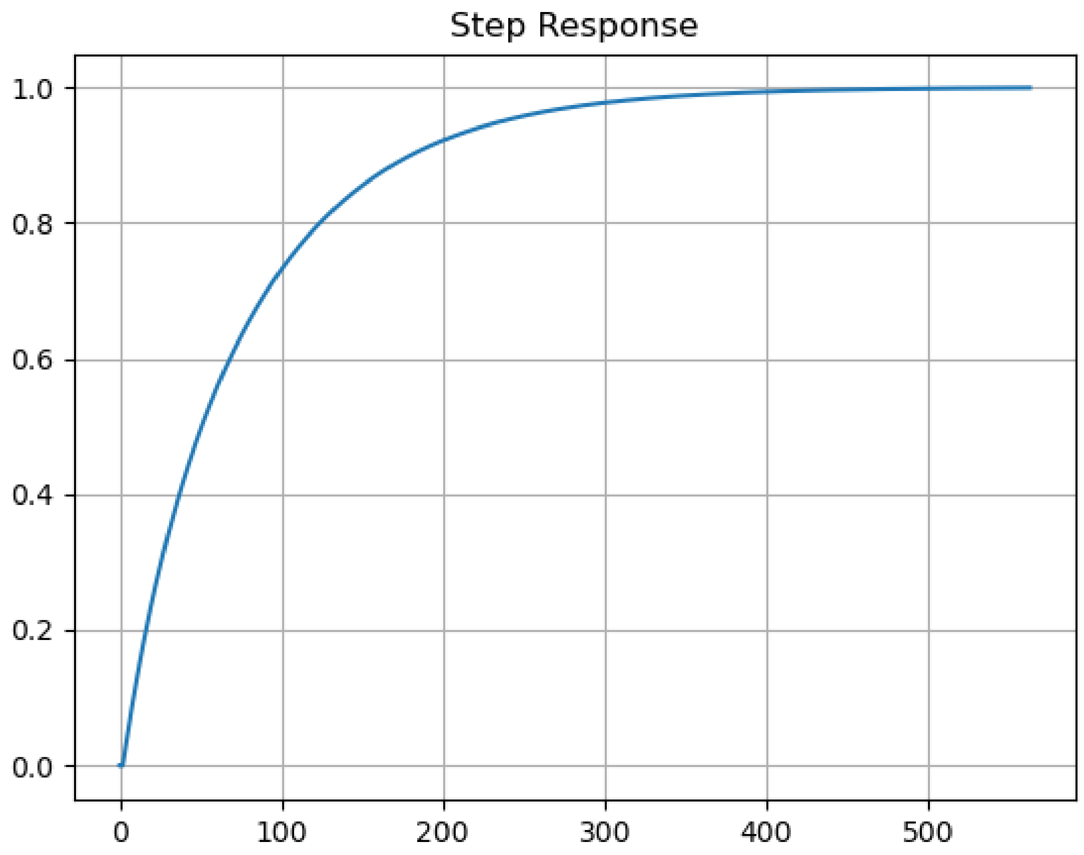
PID Parameter for PI controller:

Kp= 0.45 X 0.22=**0.099**

Ti = 32/1.2 = **26.67**

Implementation of New PI Parameter to the Transfer function

```python
import numpy as np
import matplotlib.pyplot as plt
import control
# Process Parameters
Kh = 3.5
theta_t = 22
theta_d = 2

# Transfer Function Process
num_p = np. array ([Kh])
den_p = np. array ([theta_t , 1])
Hp1 = control.tf(num_p , den_p)

N = 5 # Time Delay - Order of the Approximation
[num_pade,den_pade] = control.pade(theta_d, N)
Hp_pade = control.tf(num_pade,den_pade);
#print ('Hp_pade(s) =', Hp_pade)
Hp = control.series(Hp1, Hp_pade);

# Transfer Function PI Controller
Kp = 0.099
Ti = 26.67
num_c = np.array ([Kp*Ti, Kp])
den_c = np.array ([Ti , 0])
Hc = control.tf(num_c, den_c)
print ('Hc(s) =', Hc)

# The Loop Transfer function
L = control.series(Hc, Hp)
print ('L(s) =', L)

# Tracking transfer function
T = control.feedback(L,1)
print ('T(s) =', T)

#Step Response Feedback System (Tracking System)
t, y = control.step_response(T)
plt.figure()
plt.plot(t,y)
plt.title("Step Response")
plt.grid()
```



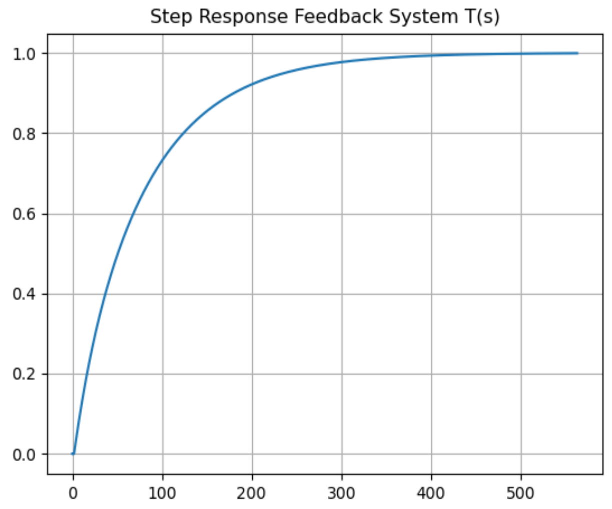
Stability Analysis

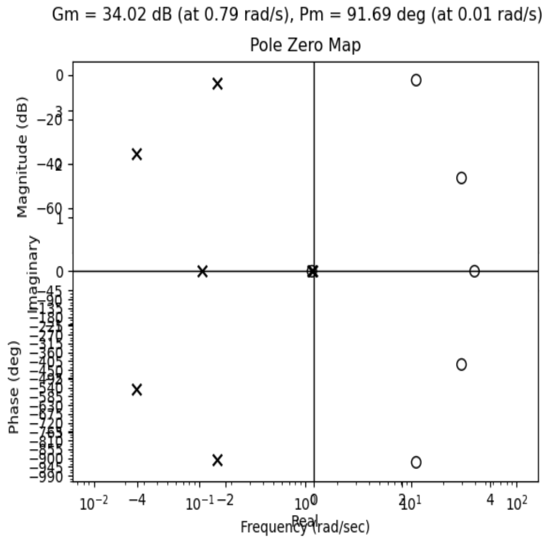
Asymptotically stable system: $\omega_c < \omega_{180}$
Marginally stable system: $\omega_c = \omega_{180}$
Unstable system: $\omega_c > \omega_{180}$

```python
import numpy as np
import matplotlib.pyplot as plt
import control
# Process Parameters
Kh = 3.5
theta_t = 22
theta_d = 2

# Transfer Function Process
num_p = np. array ([Kh])
den_p = np. array ([theta_t , 1])
Hp1 = control.tf(num_p , den_p)

N = 5 # Time Delay - Order of the Approximation
[num_pade,den_pade] = control.pade(theta_d, N)
Hp_pade = control.tf(num_pade,den_pade);
Hp = control.series(Hp1, Hp_pade);

# Transfer Function PI Controller
Kp = 0.099
Ti = 26.67
num_c = np.array ([Kp*Ti, Kp])
den_c = np.array ([Ti , 0])
Hc = control.tf(num_c, den_c)
print ('Hc(s) =', Hc)

# The Loop Transfer function
L = control.series(Hc, Hp)
print ('L(s) =', L)

# Tracking transfer function
T = control.feedback(L,1)
print ('T(s) =', T)
```

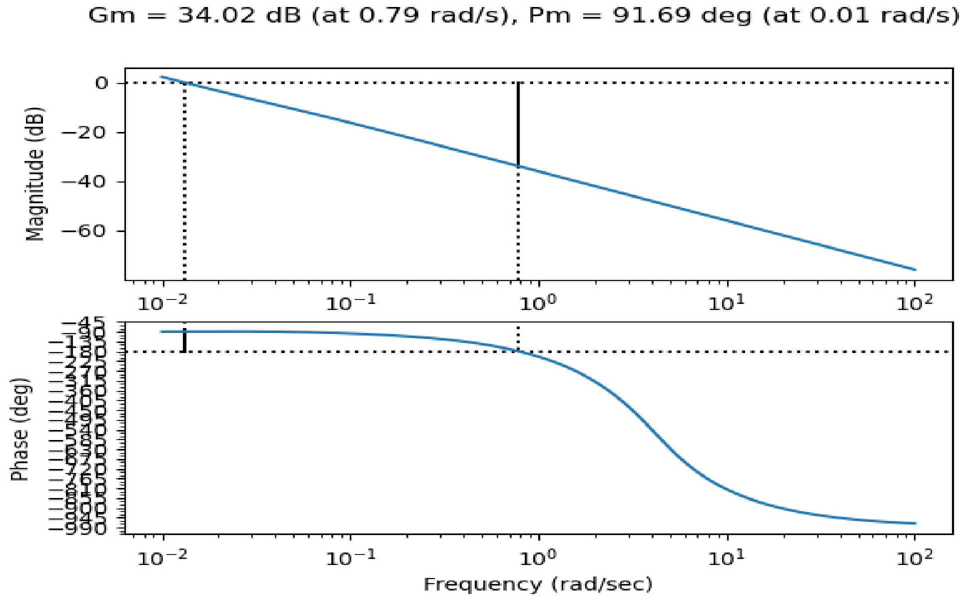
```python
# Sensitivity transfer function
S = 1 - T
print ('S(s) =', S)

# Step Response Feedback System (Tracking System)
t, y = control.step_response(T)
plt.figure(1)
plt.plot(t,y)
plt.title("Step Response Feedback System T(s)")
plt.grid()

# Bode Diagram with Stability Margins
plt.figure(2)
control.bode(L, dB=True, deg=True, margins=True)

# Poles and Zeros
plt.figure(3)
control.pzmap(T)
p = control.pole(T)
z = control.zero(T)
print("poles = ", p)
print("zero= ", z)

# Calculating stability margins and crossover frequencies
gm , pm , w180 , wc = control.margin(L)
# Convert gm to Decibel
gmdb = 20 * np.log10(gm)
print("wc =", f'{wc:.2f}', "rad/s")
print("w180 =", f'{w180:.2f}', "rad/s")
print("GM =", f'{gm:.2f}')
print("GM =", f'{gmdb:.2f}', "dB")
print("PM =", f'{pm:.2f}', "deg")

# Find when Sysem is Marginally Stable (Kritical Gain - Kc)
Kc = Kp*gm
print("Kc=", f'{Kc:.2f}')
```

```
poles =  [-2.19278871+3.51803301j -2.19278871-3.51803301j
-4.02425149+2.19288771j
 -4.02425149-2.19288771j -2.53361796+0.j         -0.04973633+0.j
 -0.01226985+0.j         ]
zero=  [ 2.3246743 +3.57102292j  2.3246743 -3.57102292j  3.6467386 +0.j
  3.3519564 +1.74266142j  3.3519564 -1.74266142j -0.03749531+0.j        ]
wc = 0.01 rad/s
w180 = 0.79 rad/s
GM = 50.21
GM = 34.02 dB
PM = 91.69 deg
Kc= 4.97
```



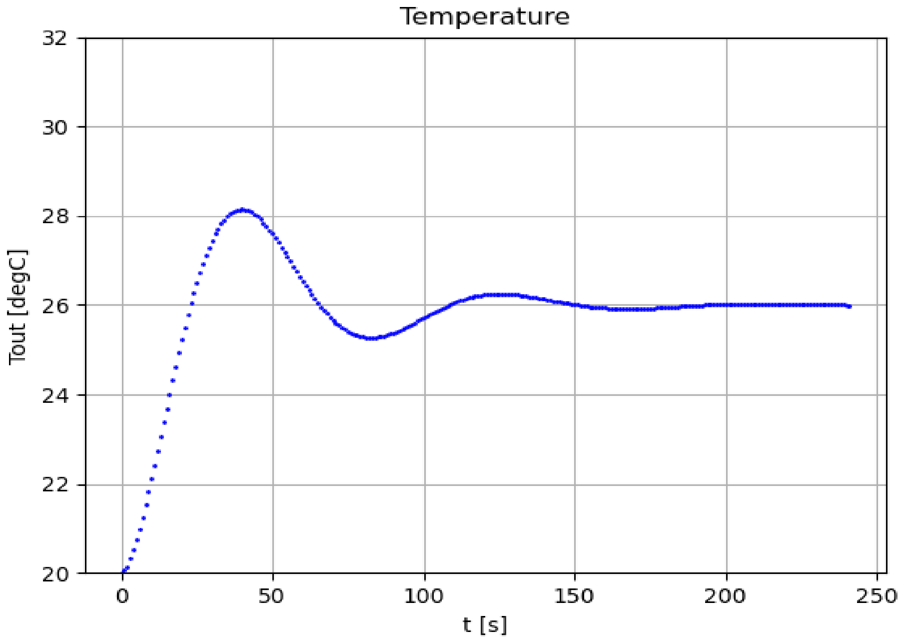


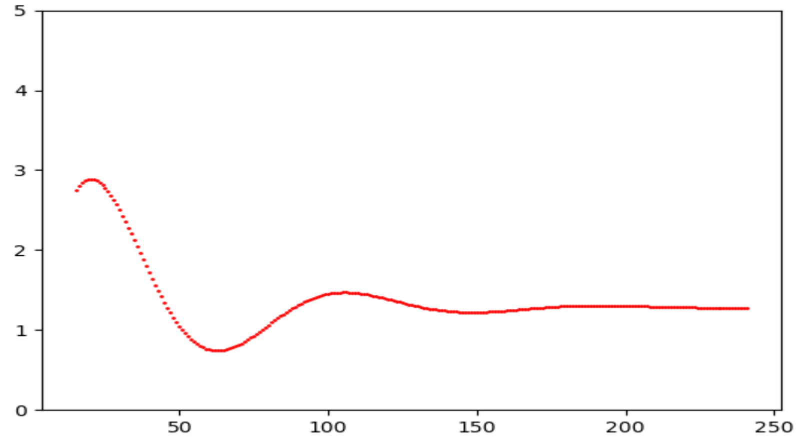
Kp=0.099
Ti = 26.67



simulation of a mathematical model of the Air Heater system

```python
# Air Heater System
import numpy as np
import time
import matplotlib.pyplot as plt
# Model Parameters
Kh = 3.5
theta_t = 22
theta_d = 2
Tenv = 21.5

# Simulation Parameters
Ts = 0.1 # Sampling Time
Tstop = 400 # End of Simulation Time
N = int(Tstop/Ts) # Simulation length
Tout = np.zeros(N+2) # Initialization the Tout vector
Tout[0] = 20 # Initial Vaue

# PI Controller Settings
Kp = 0.099
Ti = 26.67
r = 26 # Reference value [degC]
e = np.zeros(N+2) # Initialization
u = np.zeros(N+2) # Initialization

t = np.arange(0,Tstop+2*Ts,Ts) #Create the Time Series

# Formatting the appearance of the Plot
plt.figure(1)
plt.title('Control Signal')
plt.xlabel('t [s]')
plt.ylabel('u [V]')
plt.grid()
plt.figure(2)
plt.title('Temperature')
plt.xlabel('t [s]')
plt.ylabel('Tout [degC]')
plt.grid()

# Simulation
try:
    for k in range(N+1):
    # Controller
        e[k] = r - Tout[k]
        u[k] = u[k-1] + Kp*(e[k] - e[k-1]) + (Kp/Ti)*e[k] #PI Controller
        if u[k]>5:
            u[k] = 5
            # Process Model
        Tout[k+1] = Tout[k] + (Ts/theta_t) * (-Tout[k] + Kh*u[int(k-theta_d/Ts)] + Tenv)
        print("t = %2.1f, u = %3.2f, Tout = %3.1f" %(t[k], u[k], Tout[k+1]))
        if k%10 == 0: #Update Plot only every second
            # Plot Control Signal
            plt.figure(1)
            plt.plot(t[k],u[k], '-o', markersize=1, color='red')
            plt.ylim(0, 5)
            plt.show()
            plt.pause(Ts)
            # Plot Temperature
            plt.figure(2)
            plt.plot(t[k],Tout[k+1], '-o', markersize=1, color='blue')
            plt.ylim(20, 32)
            plt.show()
            plt.pause(Ts)
        time.sleep(Ts)
except KeyboardInterrupt:
    pass
print("Program Finished")
```

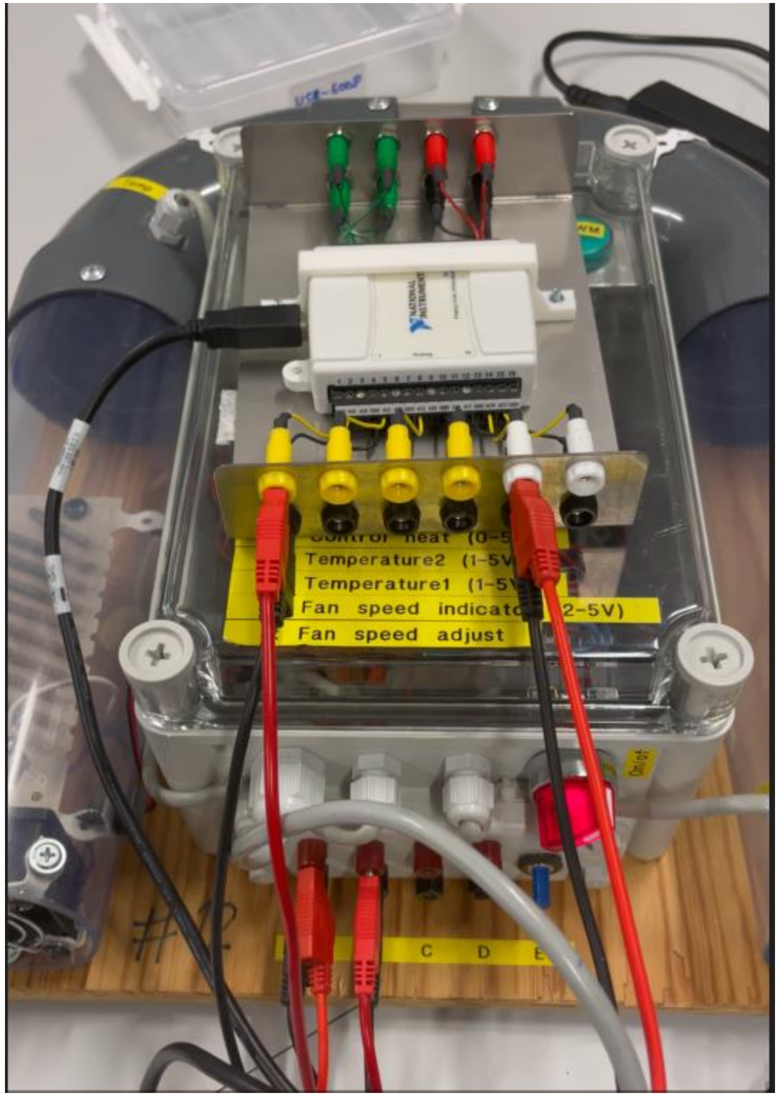




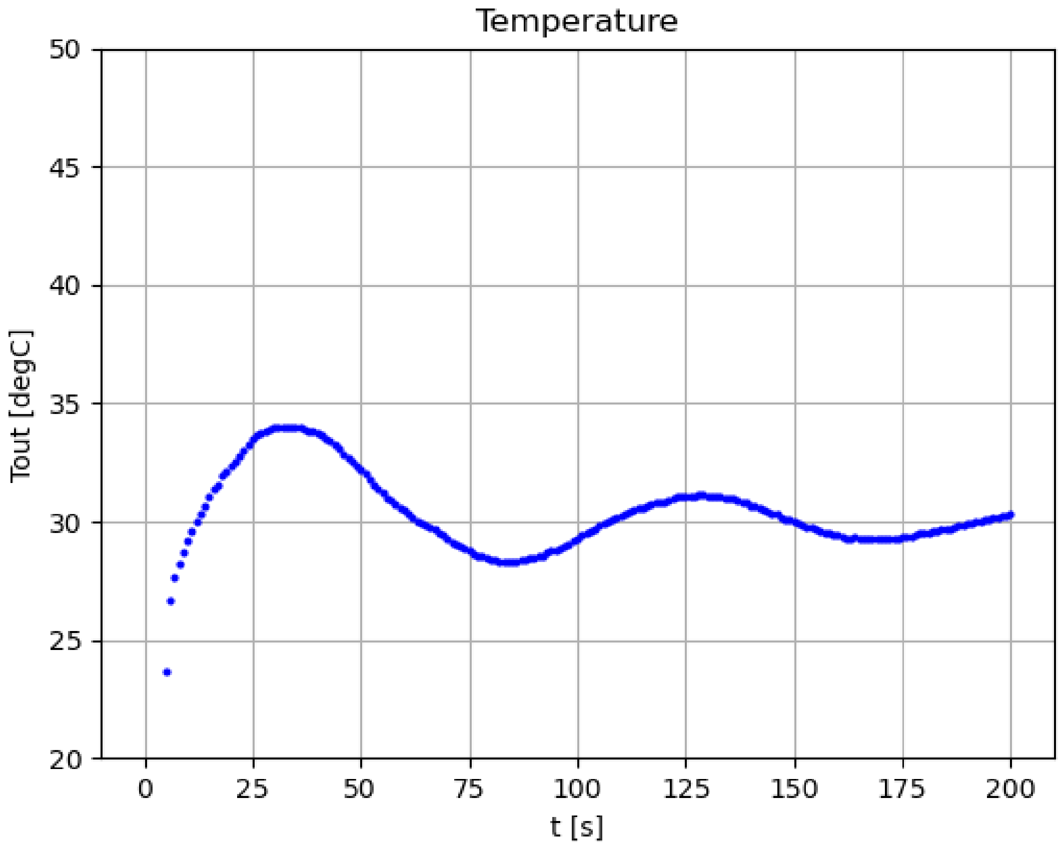
simulation of a real Air Heater

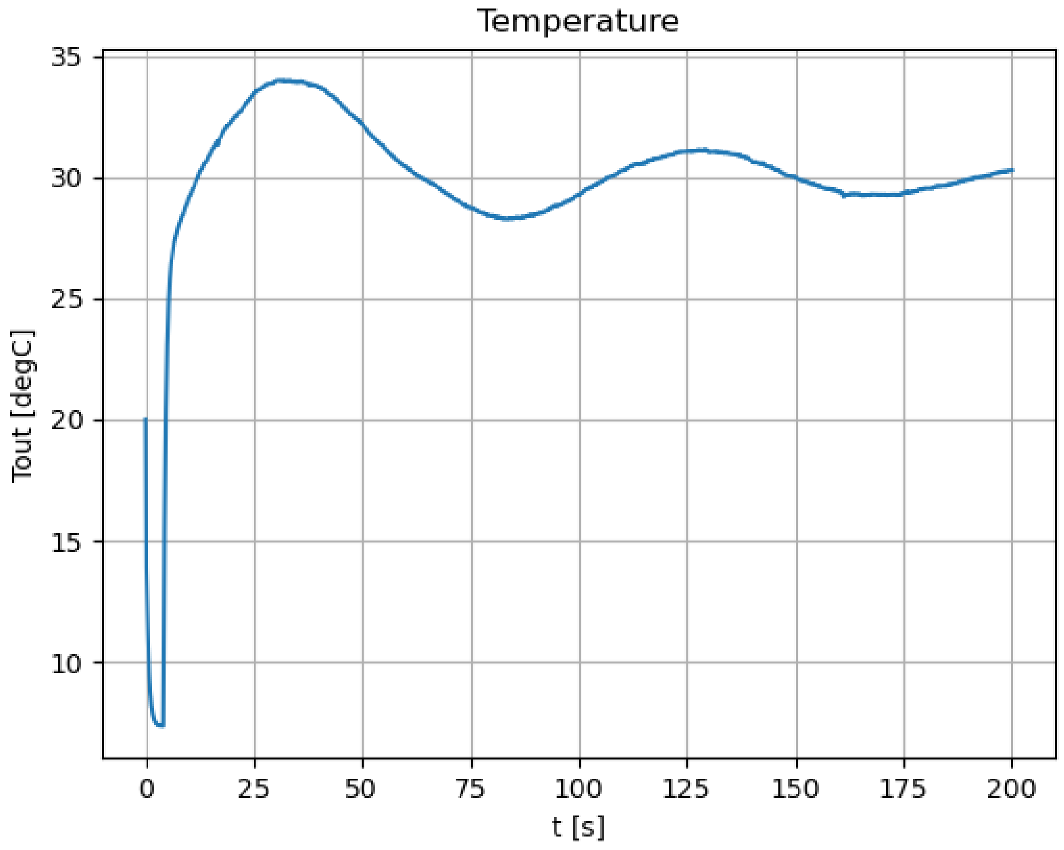
```python
import numpy as np
import time
import matplotlib.pyplot as plt
import nidaqmx
from nidaqmx.constants import (TerminalConfiguration)
task_ai = nidaqmx.Task()
task_ai.ai_channels.add_ai_voltage_chan("daq1/ai0", terminal_config=TerminalConfiguration.RSE)
task_ai.start()
task_ao = nidaqmx.Task()
task_ao.ao_channels.add_ao_voltage_chan('daq1/ao0','mychannel',0,5)
task_ao.start()

# Control System Parameters
Ts = 0.1 # Sampling Time
Tstop = 200
N = int(Tstop/Ts)
Tout = np.zeros(N+2) # Initialization the Tout vector
Tout[0] = 20 # Initial Value
Tf = 0.5 #Lowpass Filter

# PI Controller Settings
Kp = 0.099
Ti = 26.67
r = 30 # Reference value [degC]

e = np.zeros(N+2) # Initialization
u = np.zeros(N+2) # Initialization
t = np.arange(0,Tstop+2*Ts,Ts) #Create the Time Series

# Formatting the appearance of the Plot
plt.figure(1)
plt.title('Control Signal')
plt.xlabel('t [s]')
plt.ylabel('u [V]')
plt.grid()
plt.figure(2)
plt.title('Temperature')
plt.xlabel('t [s]')
plt.ylabel('Tout [degC]')
```

```python
plt.grid()
def scaling(x, x1, x2, y1, y2):
    y = y1 + (x-x1)*(y2-y1)/(x2-x1)
    return y
def lowpass(u, y_prev, Tf, Ts):
    a = Ts/(Tf+Ts)
    y = (1-a)*y_prev + a*u
    y_prev = y
    return y

# Control System Loop
for k in range(N+1):
# Controller
    e[k] = r - Tout[k]
    u[k] = u[k-1] + Kp*(e[k] - e[k-1]) + (Kp/Ti)*e[k] #PI Controller
    if u[k]<0:
        u[k] = 0
    if u[k]>5:
        u[k] = 5
    task_ao.write(u[k])
    # Process Model
    ToutVolt = task_ai.read()
    if ToutVolt<1:
        ToutVolt = 1
    if ToutVolt>5:
        ToutVolt = 5
    Tout[k+1] = scaling(ToutVolt, 1, 5, 0, 50)
    Tout[k+1] = lowpass(Tout[k+1], Tout[k], Tf, Ts)
    print("t = %2.1f, u = %3.2f, Tout = %3.1f" %(t[k], u[k], Tout[k+1]))

    if k%10 == 0: #Update Plot every second
    # Plot Control Signal
        plt.figure(1)
        plt.plot(t[k],u[k], '-o', markersize=2, color='red')
        plt.ylim(0, 5)
        plt.show()
        plt.pause(Ts)
        # Plot Temperature
        plt.figure(2)
        plt.plot(t[k],Tout[k+1], '-o', markersize=2, color='blue')
        plt.ylim(20, 50)
        plt.show()
        plt.pause(Ts)
    time.sleep(Ts)
plt.figure(3)
plt.plot(t, Tout)
plt.title('Temperature')
plt.xlabel('t [s]')
```



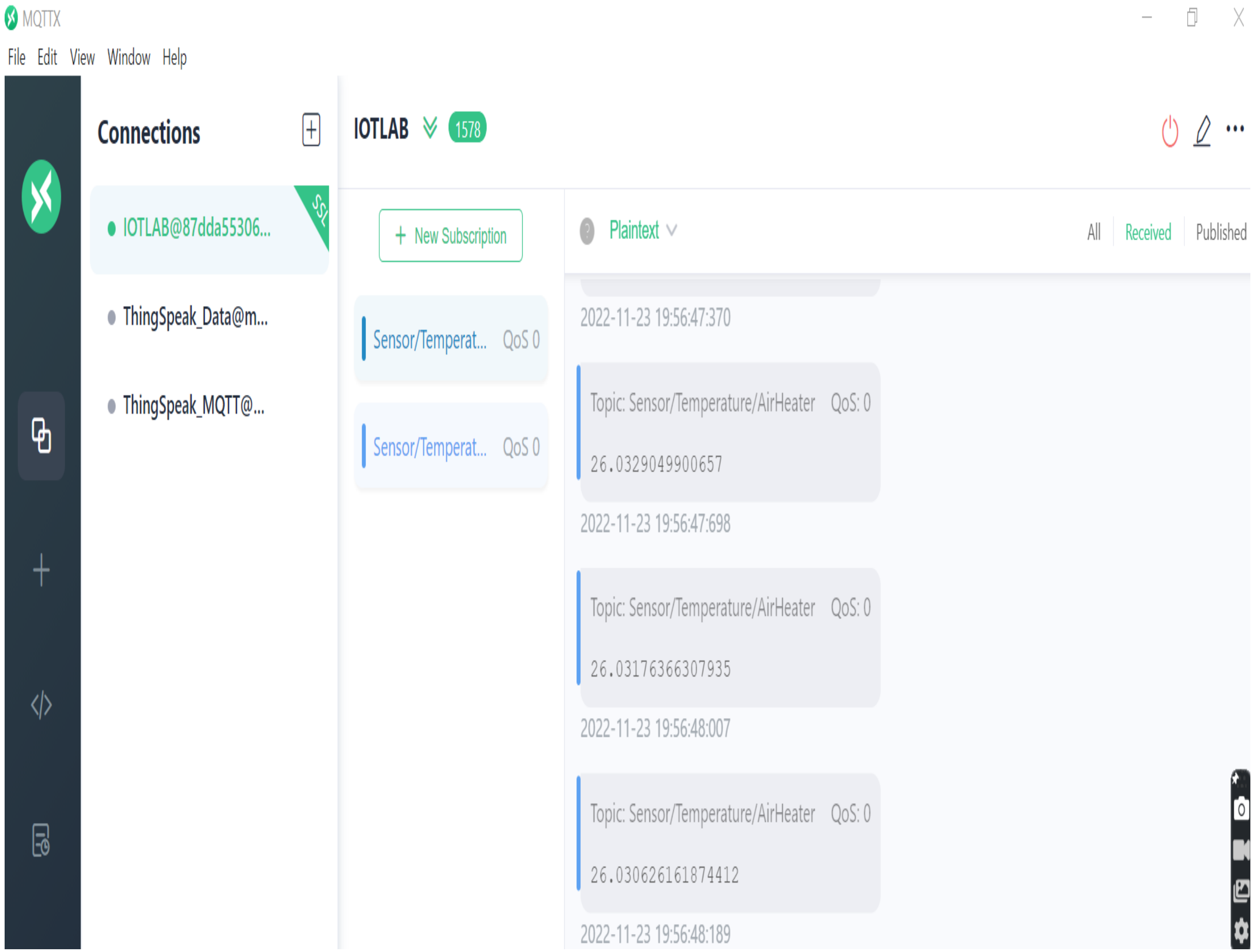
simulation of a real Air Heater





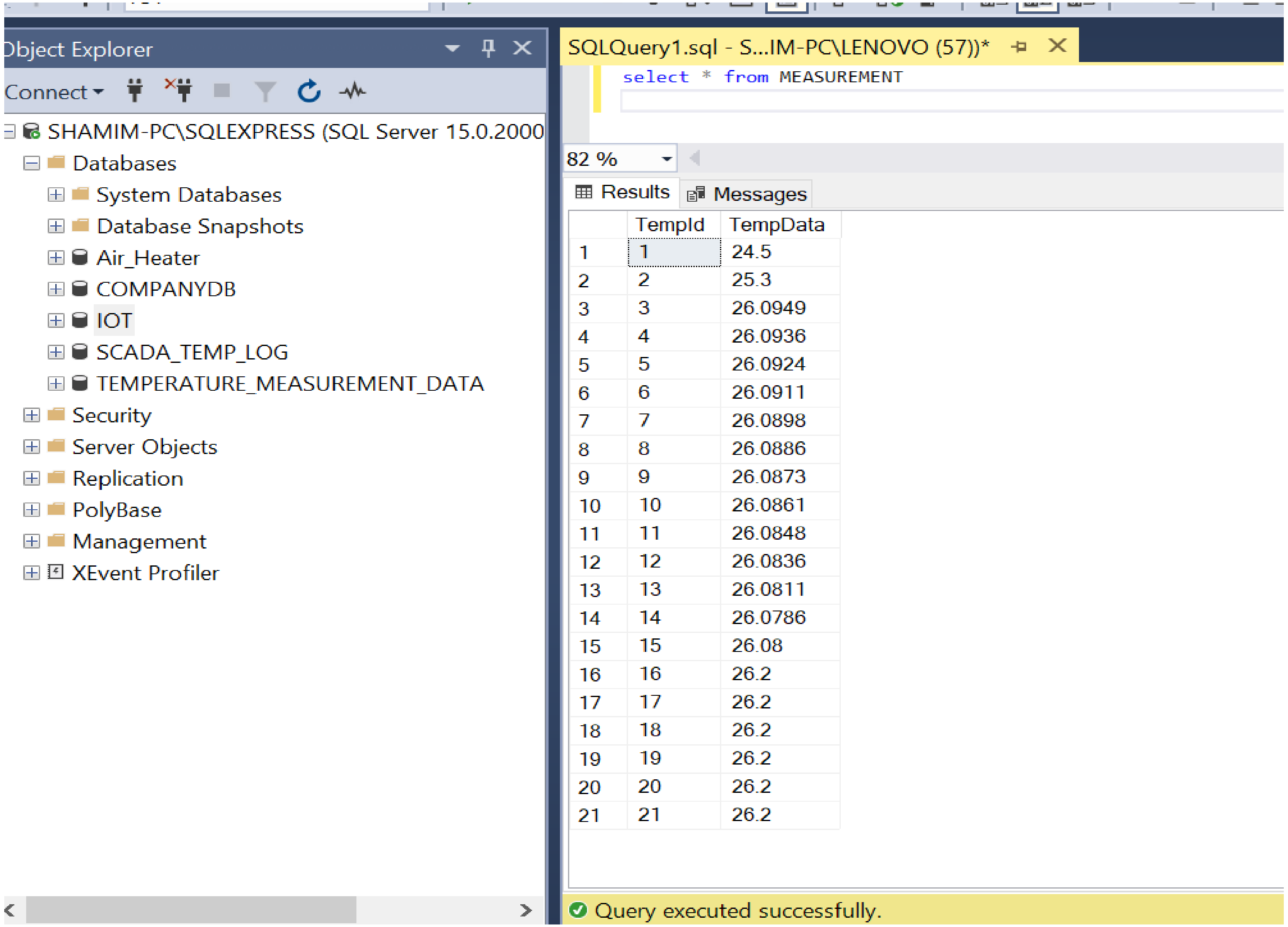
Publish Temperature data to MQTT

```python
import numpy as np
import time
import matplotlib.pyplot as plt
import paho.mqtt.client as mqtt

# MQTT connection Parameter
brokerAddress = "87dda553061c43adbe54872a0430dc70.s1.eu.hivemq.cloud"
userName = "joyelshamim"
passWord = "rjpcbl@2012"
topic = "Sensor/Temperature/AirHeater"

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))
# create the client
client = mqtt.Client()
client.on_connect = on_connect
client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

# Model Parameters
Kh = 3.5
theta_t = 22
theta_d = 2
Tenv = 21.5

# Simulation Parameters
Ts = 0.1 # Sampling Time
Tstop = 400 # End of Simulation Time
N = int(Tstop/Ts) # Simulation length
Tout = np.zeros(N+2) # Initialization the Tout vector
Tout[0] = 20 # Initial Vaue
wait= 15

# PI Controller Settings
Kp = 0.099
Ti = 26.67
r = 26 # Reference value [degC]
e = np.zeros(N+2) # Initialization
u = np.zeros(N+2) # Initialization

t = np.arange(0,Tstop+2*Ts,Ts) #Create the Time Series

# Formatting the appearance of the Plot
plt.figure(1)
plt.title('Control Signal')
plt.xlabel('t [s]')
plt.ylabel('u [V]')
plt.grid()

plt.figure(2)
plt.title('Temperature')
plt.xlabel('t [s]')
plt.ylabel('Tout [degC]')
plt.grid()

# Simulation
try:
    for k in range(N+1):
    # Controller
        e[k] = r - Tout[k]
        u[k] = u[k-1] + Kp*(e[k] - e[k-1]) + (Kp/Ti)*e[k] #PI Controller
        if u[k]>5:
            u[k] = 5
            # Process Model
        Tout[k+1] = Tout[k] + (Ts/theta_t) * (-Tout[k] + Kh*u[int(k-theta_d/Ts)] + Tenv)
        print("t = %2.1f, u = %3.2f, Tout = %3.1f" %(t[k], u[k], Tout[k+1]))

        if k%10 == 0: #Update Plot only every second
            # Plot Control Signal
            plt.figure(1)
            plt.plot(t[k],u[k], '-o', markersize=1, color='red')
            plt.ylim(0, 5)
            plt.show()
            plt.pause(Ts)
            # Plot Temperature
            plt.figure(2)
            plt.plot(t[k],Tout[k+1], '-o', markersize=1, color='blue')
            plt.ylim(20, 32)
            plt.show()
            plt.pause(Ts)
        client.publish(topic, Tout[k+1])
        time.sleep(Ts)
except KeyboardInterrupt:
    pass
print("Program Finished")
```



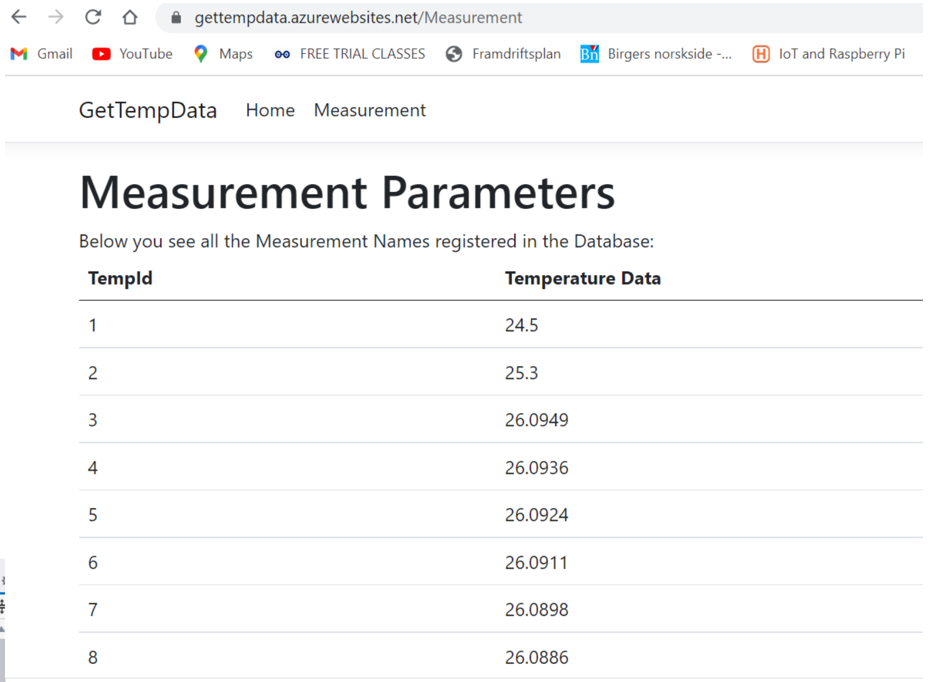
Publish Temperature data to SQL

```python
passWord = "rjpcbl@2012"
topic = "Sensor/Temperature/AirHeater"

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " + msg.

# create the client
client = mqtt.Client()
client.on_message = on_message
client.on_connect = on_connect

client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

print(client.subscribe(topic))
print(client.on_message)
print(client.on_connect)

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "INSERT INTO MEASUREMENT (TempData) VALUES (?)"
```



	TempId	TempData
1	1	24.5
2	2	25.3
3	3	26.0949
4	4	26.0936
5	5	26.0924
6	6	26.0911
7	7	26.0898
8	8	26.0886
9	9	26.0873
10	10	26.0861
11	11	26.0848
12	12	26.0836
13	13	26.0811
14	14	26.0786
15	15	26.08
16	16	26.2
17	17	26.2
18	18	26.2
19	19	26.2
20	20	26.2
21	21	26.2

ASP.NET Core Web Application for Monitoring Data


gettempdata.azurewebsites.net/Measurement

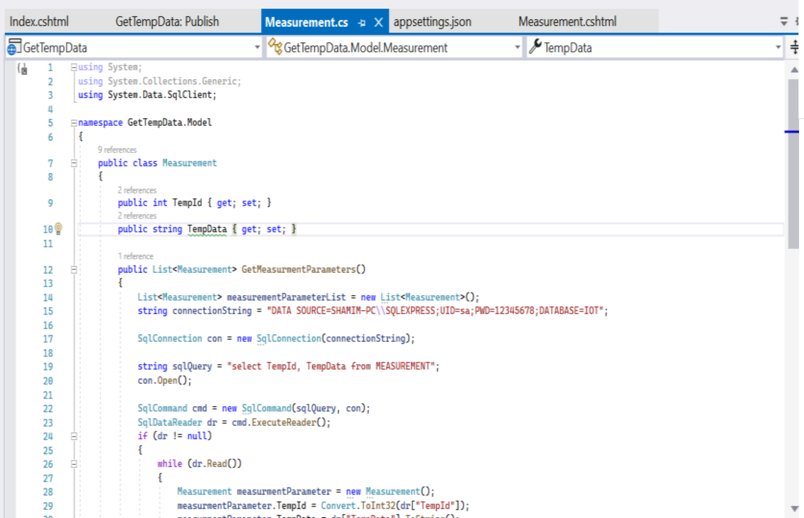
Gmail YouTube Maps FREE TRIAL CLASSES Framdriftsplan Birgers norskside -... IoT and Raspberry Pi

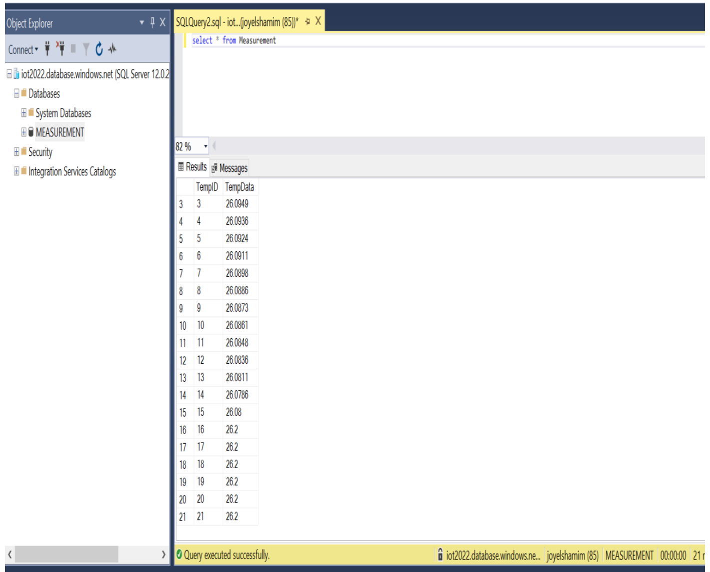
GetTempData Home Measurement

Measurement Parameters

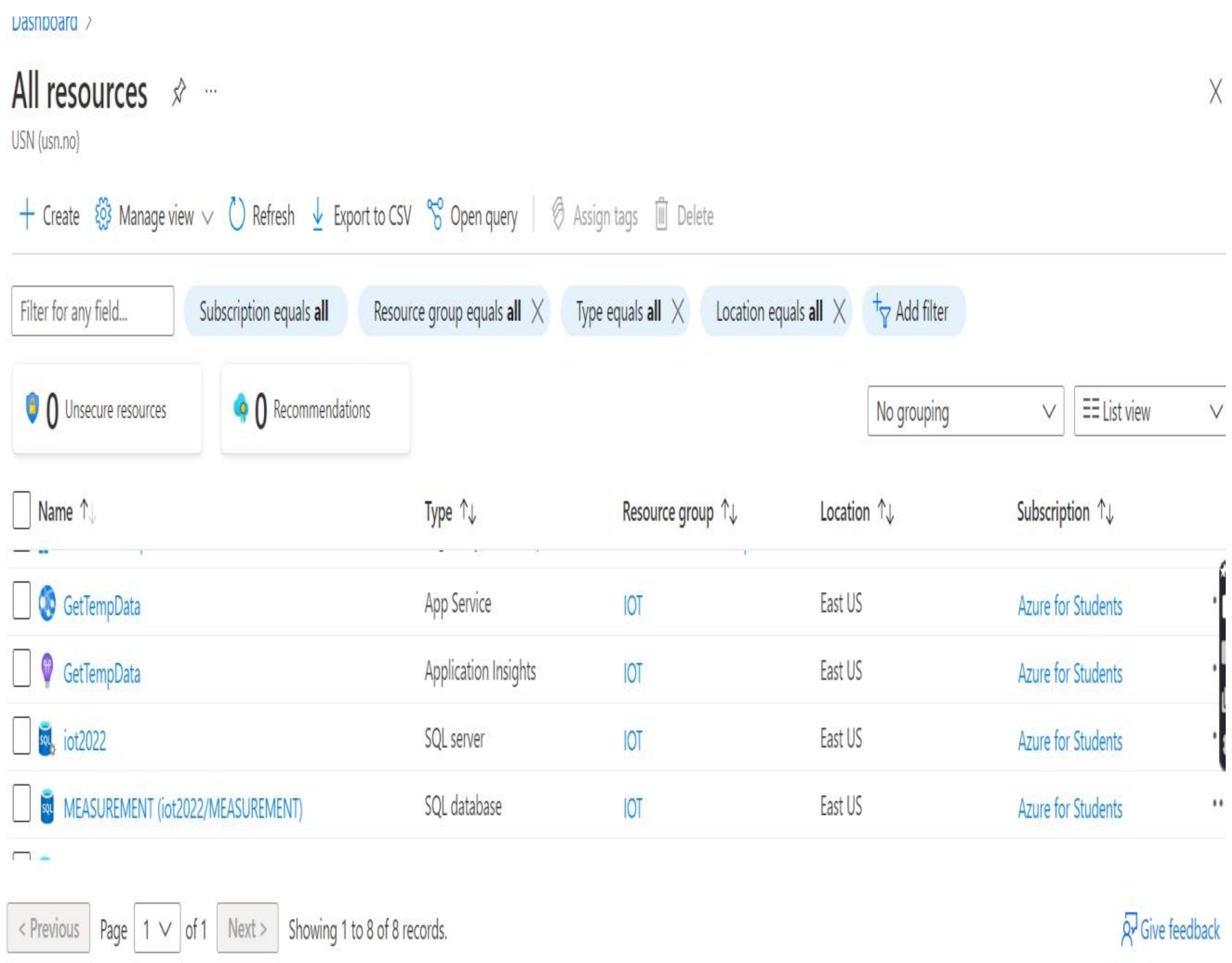
Below you see all the Measurement Names registered in the Database:

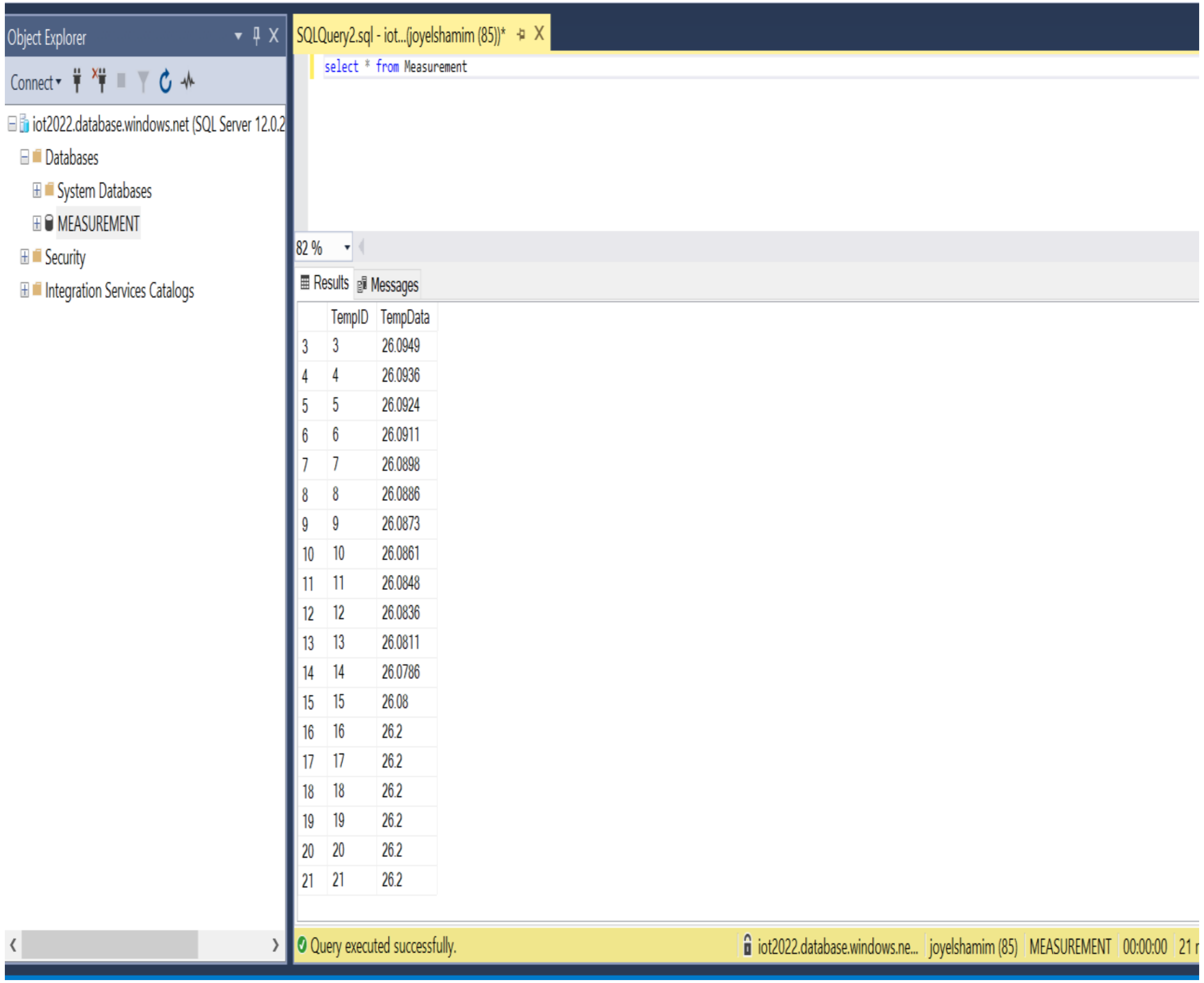
TempId	Temperature Data
1	24.5
2	25.3
3	26.0949
4	26.0936
5	26.0924
6	26.0911
7	26.0898
8	26.0886





Deploy system to Microsoft Azure





THANK YOU