LAPORAN MIKROKONTROLLER

Oleh :

DAFFA TUNGGA WISESA

NPM 21081010243

PROGRAM STUDI INFORMATIKA FAKULTAS ILMU KOMPUTER UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2024

A. Program Blink:

```
#define LED 2

void setup() {
 pinMode(LED,OUTPUT);
} void loop() {  delay(500);
 digitalWrite(LED,HIGH);
delay(500);
 digitalWrite(LED,LOW);
}
```

Kode ini adalah program sederhana untuk mengendalikan LED menggunakan **Arduino**. Berikut adalah penjelasan detail untuk setiap bagian dari kode:

1. Pendefinisian Konsta

```
#define LED 2
```

•Baris ini mendefinisikan konstanta bernama `LED` dengan nilai `2`.

•Konstanta `LED` digunakan untuk merepresentasikan pin **digital** nomor 2 pada board Arduino.

2. Fungsi `setup()`

```
void setup() {  pinMode(LED,OUTPUT);
```

•Fungsi `setup()` dijalankan satu kali saat Arduino pertama kali dinyalakan atau di-reset.

•`pinMode(LED,OUTPUT);` mengatur pin 2 (yang didefinisikan dengan `LED`) sebagai **output**, sehingga dapat mengontrol perangkat seperti LED.

3. Fungsi `loop()`

```
void loop() {  delay(500);  digitalWrite(LED,HIGH);  delay(500);
 digitalWrite(LED,LOW);
}
```

Fungsi `loop()` dijalankan terus-menerus selama Arduino aktif. •**`delay(500)`**`;` : Menunggu selama 500 milidetik (0.5 detik) sebelum menjalankan instruksi berikutnya.

•**`digitalWrite(LED,HIGH);`** **:** Mengirim sinyal HIGH ke pin 2, yang menyalakan LED.

•**`delay(500);`** : Menunggu selama 500 milidetik (0.5 detik) lagi.

•**digitalWrite(LED,LOW)**`;` : Mengirim sinyal LOW ke pin 2, yang mematikan LED.

OutPut

LED yang terhubung ke pin 2 akan menyala selama 0.5 detik, kemudian mati selama 0.5 detik, dan pola ini akan terus diulang selama Arduino menyala.

Prinsip Kerja

•`HIGH`: Menghubungkan pin ke tegangan tinggi (biasanya 5V), sehingga LED menyala.

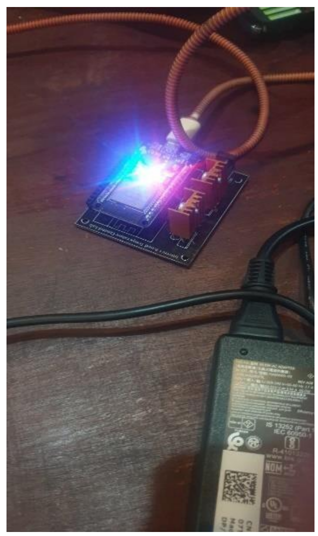
•`LOW`: Menghubungkan pin ke ground (0V), sehingga LED mati.

•Fungsi `delay()` menambahkan jeda untuk menghasilkan efek nyala-mati secara teratur.

Kegunaan

Program ini cocok untuk membuat LED berkedip (blinking) sederhana, yang sering digunakan sebagai contoh dasar pemrograman Arduino.

Foto Hasil



B. ITCLab-1:

```
#include <Arduino.h>

// constants
const int baud = 115200;     // serial baud rate

// 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

float cel, cel1, degC, degC1;
const float upper_temperature_limit = 55;

// global variables
float Q1 = 0;         // value written to Q1 pin float Q2 = 0;
// value written to Q2 pin int iwrite_max = 255;      // integer
value for writing int iwrite_min = 0;       // integer value for
writing
 void setup() {
 // put your setup code here, to run once:
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);
ledcWrite(ledChannel,0);
}
void Q1on(){
  ledcWrite(Q1Channel,iwrite_max/255*100);
  //Q1 = iwrite_max/255*100;
  //Serial.println(Q1);
}
void Q1off(){
  ledcWrite(Q1Channel,iwrite_min/255*100);
  //Q1 = iwrite_min/255*100;
  //Serial.println(Q1);
}
void Q2on(){
  ledcWrite(Q2Channel,iwrite_max/255*100);
  //Q2 = iwrite_max/255*100;
  //Serial.println(Q2);
```

```
}
void Q2off(){
  ledcWrite(Q2Channel,iwrite_min/255*100);
  //Q2 = iwrite_min/255*100;
  //Serial.println(Q2);
}
void ledon(){
  ledcWrite(ledChannel,iwrite_max);
}
void ledoff(){
  ledcWrite(ledChannel,iwrite_min);
}
void cektemp(){
 degC = analogRead(pinT1) * 0.322265625 ;  // use for 3.3v AREF   cel =
degC/10;
 degC1 = analogRead(pinT2) * 0.322265625 ;  // use for 3.3v AREF   cel1 =
degC1/10;

 Serial.print("Temperature: ");
 Serial.print(cel);  // print the temperature T1 in Celsius
 Serial.print("°C");
 Serial.print(" ~ "); // separator between Celsius and Fahrenheit
 Serial.print(cel1);  // print the temperature T2 in Celsius
 Serial.println("°C");
}
void loop() {
 // put your main code here, to run repeatedly:  cektemp();
 if (cel > upper_temperature_limit){
Q1off();    ledon();
 } else {
Q1on();
ledoff();
 }
 if (cel1 > upper_temperature_limit){
Q2off();    ledon();
 } else {
Q2on();
ledoff();
 }
 delay (100);
}
```

Kode ini adalah program berbasis Arduino yang menggunakan **iTCLab Shield** untuk mengukur suhu melalui sensor suhu (T1 dan T2), mengendalikan output PWM (Q1 dan Q2), dan LED. Berikut adalah penjelasan detail dari kode tersebut:

1. Header dan Konstanta

```
#include <Arduino.h>
```

Header ini digunakan dalam platform Arduino IDE untuk memastikan kompatibilitas fungsi.

```
const int baud = 115200;    // serial baud rate
```

Kecepatan komunikasi serial (baud rate) diatur ke 115200 bps untuk pengiriman data melalui Serial Monitor.

```
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
```

•Pin **T1** dan **T2** adalah input analog untuk membaca suhu dari sensor.

•Pin **Q1**, **Q2**, dan **LED** adalah output digital dengan fungsi PWM.

```
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
```

•Frekuensi PWM diatur ke 5 kHz dengan resolusi 8-bit (nilai 0-255).

•**upper_temperature_limit** adalah batas suhu maksimum untuk tindakan otomatis.

2. Variabel Global

```
float Q1 = 0;         // value written to Q1 pin float Q2 = 0;
// value written to Q2 pin int iwrite_max = 255;     // integer
value for writing int iwrite_min = 0;      // integer value for
writing
```

Variabel ini digunakan untuk mengontrol dan menyimpan data PWM serta nilai suhu.

3. Fungsi setup()

```
void setup() {
 // put your setup code here, to run once:
Serial.begin(baud);  while (!Serial) {
  ; // wait for serial port to connect.  }
```

Memulai komunikasi serial dengan baud rate 115200 dan menunggu koneksi serial.

```
 // 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);
 ledcWrite(ledChannel,0);
}
```

Pin Q1, Q2, dan LED dikonfigurasi sebagai output PWM menggunakan kanal masing-masing.

4. Fungsi Pengendalian

Menghidupkan/Mematikan Q1, Q2, dan LED:

```
void Q1on(){
  ledcWrite(Q1Channel,iwrite_max/255*100);
  //Q1 = iwrite_max/255*100;
  //Serial.println(Q1);
}
void Q1off(){
  ledcWrite(Q1Channel,iwrite_min/255*100);
  //Q1 = iwrite_min/255*100;
  //Serial.println(Q1);
}
void Q2on(){
  ledcWrite(Q2Channel,iwrite_max/255*100);
  //Q2 = iwrite_max/255*100;
  //Serial.println(Q2);
}
void Q2off(){
  ledcWrite(Q2Channel,iwrite_min/255*100);
  //Q2 = iwrite_min/255*100;
  //Serial.println(Q2);
}
void ledon(){
  ledcWrite(ledChannel,iwrite_max);
}
void ledoff(){
  ledcWrite(ledChannel,iwrite_min);
}
```

Fungsi ini digunakan untuk mengontrol status PWM (0 untuk mati, nilai maksimum untuk hidup). **Membaca Suhu dari Sensor:**

```
void cektemp(){
 degC = analogRead(pinT1) * 0.322265625 ;  // use for 3.3v AREF   cel = degC/10;
 degC1 = analogRead(pinT2) * 0.322265625 ;  // use for 3.3v AREF   cel1 = degC1/10;

 Serial.print("Temperature: ");
 Serial.print(cel);  // print the temperature T1 in Celsius
 Serial.print("°C");
 Serial.print(" ~ "); // separator between Celsius and Fahrenheit
```

```
 Serial.print(cel1); // print the temperature T2 in Celsius
 Serial.println("°C");
}
```

`analogRead` membaca nilai ADC dari pin sensor suhu dan dikonversi ke derajat Celsius menggunakan faktor skala.

5. Fungsi `loop()`:

```
void loop() {
 // put your main code here, to run repeatedly:  cektemp();
 if (cel > upper_temperature_limit){
Q1off();    ledon();
 }  else {
Q1on();
ledoff();
 }
 if (cel1 > upper_temperature_limit){
Q2off();    ledon();
 }  else {
Q2on();
ledoff();
 }
 delay (100);
}
```

Fungsi ini:

- Membaca suhu T1 dan T2 menggunakan `cektemp()`.
- Jika suhu T1 melebihi batas, Q1 dimatikan, dan LED dihidupkan.
- Jika suhu T2 melebihi batas, Q2 dimatikan, dan LED dihidupkan.
- Jika suhu berada di bawah batas, Q1 dan Q2 dinyalakan, dan LED dimatikan.

Output

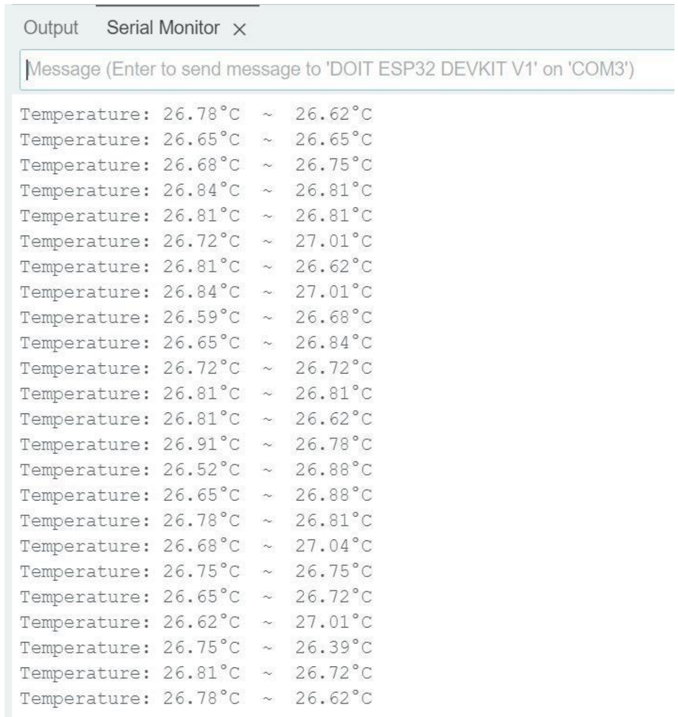
- Serial Monitor akan menampilkan suhu T1 dan T2 secara real-time.
- Jika suhu melebihi 55°C:
  - Output Q1 atau Q2 akan dimatikan.
  - LED akan menyala sebagai indikator peringatan.
- Jika suhu di bawah 55°C:
  - Output Q1 dan Q2 akan hidup.
  - LED akan mati.

Kegunaan

Kode ini dapat digunakan dalam sistem kontrol suhu otomatis seperti pendingin, pemanas, atau peringatan suhu tinggi.

Screenshot

Hasil



C. ITCLab-2

```
// the number of the LED pin const
int ledPin = 26;

// setting PWM properties
const int freq = 5000; const int
ledChannel = 0; const int
resolution = 8;

void setup(){
 // configure LED PWM functionalitites  ledcSetup(ledChannel,
freq, resolution);

 // attach the channel to the GPIO to be controlled  ledcAttachPin(ledPin,
ledChannel);
}
void loop(){
 // increase the LED brightness
 for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
  // changing the LED brightness with PWM
ledcWrite(ledChannel, dutyCycle);   delay(20);
 }

 // decrease the LED brightness
 for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
  // changing the LED brightness with PWM
ledcWrite(ledChannel, dutyCycle);    delay(20);

 }
}
```

Kode ini adalah program sederhana untuk mengontrol kecerahan sebuah LED menggunakan PWM (Pulse Width Modulation) pada ESP32 atau mikrokontroler serupa. Berikut adalah penjelasan per bagian:

1. Variabel Global

```
const int ledPin = 26;
```

Menentukan pin GPIO yang terhubung ke LED. Dalam hal ini, pin 26.

2. Properti PWM

```
const int freq = 5000; const
int ledChannel = 0; const
int resolution = 8;
```

- Frekuensi PWM yang akan digunakan, yaitu 5000 Hz.
- Saluran PWM yang digunakan (ESP32 mendukung beberapa saluran PWM).
- Resolusi PWM dalam bit, yaitu 8-bit (nilai PWM antara 0 hingga 255).

3. Fungsi `setup()`

Konfigurasi PWM pada LED:

```
ledcSetup(ledChannel, freq, resolution);
```

Mengatur properti PWM (frekuensi dan resolusi) untuk saluran yang ditentukan.

Melampirkan Saluran PWM ke GPIO:

```
ledcAttachPin(ledPin, ledChannel);
```

Mengaitkan saluran PWM ke pin GPIO yang terhubung dengan LED.

4. **Fungsi `loop()`**

Tujuan: Mengontrol LED untuk secara bertahap menjadi lebih terang dan kemudian lebih redup dalam sebuah siklus. **Meningkatkan Kecerahan LED**

```
for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
  // changing the LED brightness with PWM
ledcWrite(ledChannel, dutyCycle);    delay(20);
 }
```

Proses:

- `for` loop menaikkan nilai `dutyCycle` dari 0 ke 255.
- `ledcWrite(ledChannel, dutyCycle);` : Menulis nilai PWM ke saluran yang ditentukan untuk mengubah tingkat kecerahan LED.
- `delay(20);` : Menambahkan jeda 20 ms antara setiap perubahan kecerahan agar transisinya halus.

Menurunkan Kecerahan LED

```
for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
  // changing the LED brightness with PWM
  ledcWrite(ledChannel, dutyCycle);      delay(20);
 }
```

Proses:

- `for` loop menurunkan nilai `dutyCycle` dari 255 ke 0.
- Sama seperti sebelumnya, `ledcWrite` mengatur nilai PWM untuk membuat LED perlahan menjadi lebih redup.

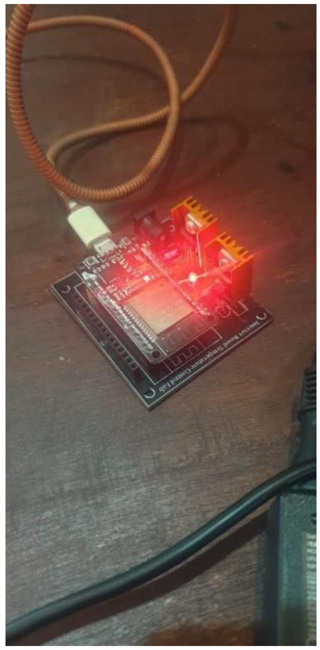
Output

- LED akan perlahan menjadi lebih terang hingga mencapai kecerahan maksimum (nilai PWM = 255).
- Setelah itu, LED akan perlahan menjadi lebih redup hingga mati (nilai PWM = 0).
- Siklus ini akan berulang terus-menerus.

Kegunaan

Kode ini menunjukkan bagaimana menggunakan fitur PWM pada ESP32 untuk menghasilkan efek fading pada LED. Nilai resolusi 8-bit memberikan tingkat kontrol hingga 256 langkah (0-255), dan loop dalam kode memberikan perubahan bertahap pada kecerahan LED dengan efek yang terlihat halus.

Hasil Foto



D. ITCLab-3

Di ITCLab ke-3 ada tiga kode yaitu, kode Arduino, Python, dan Notebook Jupyter.

```
#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

//Q1 32 - T1 34
//Q2 33 - T2 35

// 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 upper_temperature_limit = 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 >= upper_temperature_limit) {      Q1 =
0.0;
```

```
    Q2 = 0.0;
    ledcWrite(Q1Channel,0);     ledcWrite(Q2Channel,0);
    //Serial.println("High Temp 1 (> upper_temperature_limit): ");
    Serial.println(degC);
  }
  mV = (float) analogRead(pinT2) * 0.322265625;
  //degC = (mV - 500.0)/10.0;    degC
= mV/10.0;
  if (degC >= upper_temperature_limit) {
    Q1 = 0.0;      Q2 =
0.0;
    ledcWrite(Q1Channel,0);     ledcWrite(Q2Channel,0);
    //Serial.println("High Temp 2 (> upper_temperature_limit): ");
    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();
}
```

Kode ini adalah program firmware untuk **Arduino** yang digunakan untuk mengontrol dan memonitor perangkat keras melalui komunikasi serial. Program ini mengontrol elemen seperti LED dan dua saluran kontrol (Q1 dan Q2), serta membaca dua sensor suhu (T1 dan T2). Berikut penjelasan detail setiap bagian kode:

1. **Konstanta dan Pin**
   - `vers`: Versi firmware.
   - `baud`: Kecepatan komunikasi serial (115200 bps).
   - `sp` dan `nl`: Separator (`sp` = spasi) dan terminator (`nl` = newline) untuk memproses perintah serial.
   - Pin:
     - `pinT1`, `pinT2`: Pin input untuk sensor suhu.
     - `pinQ1`, `pinQ2`: Pin output PWM untuk kontrol. `pinLED`: Pin output PWM untuk kontrol LED.
   - PWM:
     - Frekuensi dan resolusi saluran PWM untuk LED (`ledChannel`) dan kontrol (`Q1Channel`, `Q2Channel`).
2. **Variabel Global**
   - `Buffer`: Buffer untuk menyimpan data input dari serial.
   - `cmd` dan `pv`: Menyimpan perintah dan nilai yang diproses dari input serial.
   - `Q1`, `Q2`: Nilai kontrol output PWM untuk saluran Q1 dan Q2.
   - `level`: Nilai kontrol untuk LED (0-100%).
   - `n`: Jumlah pengukuran untuk menghitung rata-rata suhu dari sensor.
3. **Fungsi**

   `parseSerial()`
   - Membaca data serial hingga ditemukan terminator (`\n`).
   - Memisahkan data input menjadi:
     - `cmd`: Perintah.
     - `pv`: Data numerik (misal, tingkat PWM atau nilai suhu).

   `dispatchCommand()`
   - Mengeksekusi perintah berdasarkan nilai `cmd`:
     - Q1/Q2: Mengatur output PWM untuk Q1 atau Q2, dengan batas nilai 025%.
     - T1/T2: Membaca suhu dari sensor T1/T2 dalam derajat Celsius.
     - V/VER: Mengirim versi firmware melalui serial.
     - LED: Mengontrol intensitas LED (0-100%).

- X: Mematikan Q1 dan Q2.

`checkTemp()`

- Memeriksa suhu dari T1 dan T2:
  - Jika suhu melebihi batas (`upper_temperature_limit = 59`), Q1 dan Q2 dimatikan untuk keamanan. **`setup()`**
- Inisialisasi:
  - Serial komunikasi.
  - Saluran PWM untuk Q1, Q2, dan LED.
  - Menyambungkan saluran PWM ke pin masing-masing.

`loop()`

- Fungsi utama yang terus berjalan:
  - Membaca perintah serial (`parseSerial()`).
  - Mengeksekusi perintah sesuai input (`dispatchCommand()`).
  - Memantau suhu dan mematikan kontrol jika suhu terlalu tinggi (`checkTemp()`).

```
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):         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
```

Kode ini merupakan implementasi kelas Python yang digunakan untuk berkomunikasi dengan perangkat keras berbasis Arduino melalui port serial. Perangkat ini tampaknya digunakan untuk pengendalian dan pemantauan suhu serta pengaturan elemen pemanas (heater). Berikut adalah penjelasan rinci:

1. Impor dan Instalasi Modul

- `serial`: Modul untuk komunikasi serial. Jika belum terinstal, skrip akan mencoba menginstalnya menggunakan `pip`.
- `numpy`: Digunakan untuk mengolah data (termasuk menyimpan data dalam bentuk file teks).

2. **Kelas `ITCLab`**

Kelas ini menangani komunikasi dengan Arduino melalui port serial. Berikut penjelasan metode dan atributnya:

Inisialisasi (`__init__`)

- Menghubungkan ke perangkat Arduino melalui port serial.
- Memanggil metode `findPort` untuk mendeteksi port serial yang terhubung ke Arduino.
- Mengatur baud rate komunikasi (default: 115200).
- Menunggu beberapa saat agar perangkat siap (`time.sleep(3)`).

Metode `findPort`

- Mendeteksi port serial Arduino berdasarkan VID (Vendor ID) dan PID (Product ID) yang umum digunakan oleh perangkat Arduino.
- Jika tidak ditemukan port yang cocok, pengguna akan diminta memasukkan port secara manual.
- Memberikan panduan cara menemukan port di Windows, MacOS, dan Linux.

Metode `stop` dan `version`

- `stop()`: Mengirimkan perintah "X" untuk menghentikan perangkat.
- `version()`: Mengirimkan perintah "VER" untuk mendapatkan versi firmware Arduino.

Properti `T1` dan `T2`

- Membaca suhu dari sensor suhu pada Arduino: `T1`: Membaca suhu dari sensor pertama.
   `T2`: Membaca suhu dari sensor kedua.

Metode untuk Mengontrol Perangkat

- `LED(pwm)`: Mengontrol LED pada Arduino dengan nilai PWM (0-100).
- `Q1(pwm)`: Mengontrol pemanas pertama dengan nilai PWM (0-100).
- `Q2(pwm)`: Mengontrol pemanas kedua dengan nilai PWM (0-100).

Metode `save_txt`

- Menyimpan data dalam file teks (`data.txt`).
- Data yang disimpan meliputi waktu, pengaturan heater, suhu dari sensor, dan setpoint (nilai target suhu).

Metode `read` dan `write`

- `read(cmd)`: Mengirimkan perintah ke Arduino dan membaca balasan.

- `write(cmd, pwm)`: Mengirimkan perintah beserta nilai (PWM) ke Arduino dan membaca balasan. **Metode `build_cmd_str`**
- Membuat string perintah yang dikirim ke Arduino.
- Format perintah: `{cmd} {args}\n` (misalnya: "Q1 50\n").

Metode `close`

- Menutup koneksi serial dengan Arduino.
- Menampilkan pesan jika koneksi berhasil atau jika ada masalah saat memutuskan koneksi.


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Opening connection\n",
      "iTCLab connected via Arduino on port COM9\n",
      "LED On\n",
      "LED Off\n",
      "Arduino disconnected successfully\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "True"
      ]
     },
     "execution_count": 4,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "import itclab\n",
    "import time\n",
    "# Connect to Arduino\n",
    "a = itclab.iTCLab()\n",
    "print('LED On')\n",
    "a.LED(100)\n",
    "# Pause for 1 second\n",
    "time.sleep(1.0)\n",
    "print('LED Off')\n",
    "a.LED(0)\n",
    "a.close()"
```

```
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 2,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
```

```
   "text": [
    "Opening connection\n",
    "iTCLab connected via Arduino on port COM9\n",
    "<bound method iTCLab.version of <itclab.iTCLab object at 0x0000023679324520>>\n",
    "LED On\n",
    "LED Power 100\n",
    "LED Power 90\n",
    "LED Power 80\n",
    "LED Power 70\n",
    "LED Power 60\n",
    "LED Power 50\n",
    "LED Power 40\n",
    "LED Power 30\n",
    "LED Power 20\n",
    "LED Power 10\n",
    "LED Power 0\n",
    "Arduino disconnected successfully\n"
   ]
  },
  {
   "data": {
    "text/plain": [
     "True"
    ]
   },
   "execution_count": 2,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "import itclab\n",
  "import time\n",
  "\n",
  "# Connect to Arduino\n",
  "a = itclab.iTCLab()\n",
  "\n",
  "# Get Version\n",
  "print(a.version)\n",
  "\n",
  "# Turn LED on\n",
  "print('LED On')\n",
  "a.LED(100)\n",
  "\n",
  "# Taper LED off\n",
  "for i in range(100,-1,-10):\n",
  "    print('LED Power ' + str(i))\n",
  "    time.sleep(0.5)\n",
  "    a.LED(i)\n",
  "\n",
  "a.close()"
 ]
},
```

```
{
 "cell_type": "code",
 "execution_count": 10,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
```

[redacted]

```
    "text": [
     "Arduino disconnected successfully\n"
    ]
   },
   {
    "data": {
     "text/plain": [
      "True"
     ]
    },
    "execution_count": 10,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "a.close()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
 }
],
"metadata": {
 "anaconda-cloud": {},
 "kernelspec": {
  "display_name": "Python 3 (ipykernel)",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.11.6"
 },
 "widgets": {
  "state": {
   "43a770e6b0524d899f4a1126019a0c2f": {
    "views": [
     {
      "cell_index": 1
     }
```

```
    ]
   }
  },
  "version": "1.2.0"
 }
},
"nbformat": 4,
"nbformat_minor": 4
```

```
}
```

Kode ini adalah kode Notebook Jupyter dalam format JSON, berisi skrip Python yang digunakan untuk mengontrol perangkat Arduino melalui komunikasi serial dengan modul Python. Berikut adalah penjelasan mendetail untuk setiap bagian dari kode ini:

1. Cell1: Mengontrol LED Sederhana

```
"import itclab\n",
   "import time\n",
   "# Connect to Arduino\n",
   "a = itclab.iTCLab()\n",
   "print('LED On')\n",
   "a.LED(100)\n",
   "# Pause for 1 second\n",
   "time.sleep(1.0)\n",
   "print('LED Off')\n",
   "a.LED(0)\n",     "a.close()"
```

- Impor Library: `itclab` adalah library yang mendukung komunikasi dengan perangkat Arduino, sementara `time` digunakan untuk mengatur jeda waktu.
- Koneksi ke Arduino: Objek `a` adalah instance dari kelas `iTCLab`, yang menginisialisasi koneksi serial dengan Arduino.
- Mengontrol LED:
  - Menyalakan LED dengan nilai PWM maksimum (100%).
  - Menunggu selama 1 detik (`time.sleep(1.0)`). Mematikan LED dengan nilai PWM 0%.
- Menutup Koneksi: Fungsi `a.close()` menutup koneksi serial ke Arduino.

2. Cell 2: Variasi LED dan Pembacaan Versi

```
"import itclab\n",
   "import time\n",
   "\n",
   "# Connect to Arduino\n",
   "a = itclab.iTCLab()\n",
   "\n",
   "# Get Version\n",
   "print(a.version)\n",
   "\n",
   "# Turn LED on\n",
   "print('LED On')\n",
   "a.LED(100)\n",
   "\n",
   "# Taper LED off\n",
   "for i in range(100,-1,-10):\n",
   "    print('LED Power ' + str(i))\n",
   "    time.sleep(0.5)\n",
   "    a.LED(i)\n",
   "\n",
```

```
"a.close()"
```

- Impor Library: Sama seperti cell sebelumnya.
- Membaca Versi Firmware:
  - `print(a.version)` digunakan untuk mendapatkan versi firmware Arduino.
  - Hasilnya mungkin berupa string seperti "`iTCLab v1.0`".
- Menyalakan LED: LED dinyalakan dengan nilai PWM 100%.
- Variasi Kekuatan LED:
  - Loop `for i in range(100, -1, -10)` menurunkan kekuatan PWM dari 100 ke 0 dengan langkah 10.
  - Pada setiap langkah, kekuatan LED dicetak, dan fungsi `time.sleep(0.5)` digunakan untuk menunggu 0,5 detik sebelum menyesuaikan nilai PWM berikutnya.
- Menutup Koneksi: `a.close()` menutup koneksi ke Arduino.

3. Cell 3: Menutup Koneksi

```
"a.close()"
```

Menutup koneksi serial ke Arduino. Jika ada koneksi terbuka dari cell sebelumnya, fungsi ini memastikan tidak ada konflik saat menggunakan Arduino lagi.

E. ITCLab-4


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "4d63608e",
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "%matplotlib inline\n",
    "import matplotlib.pyplot as plt\n",
    "from scipy.integrate import odeint\n",
    "import ipywidgets as wg\n",
    "from IPython.display import display"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "id": "7382d42e",
   "metadata": {},
```

```
 "outputs": [
  {
   "data": {
    "application/vnd.jupyter.widget-view+json": {
     "model_id": "18f0584c06094245be5bfad0a176d921",
     "version_major": 2,
     "version_minor": 0
```

```
     },
     "text/plain": [
      "interactive(children=(FloatSlider(value=0.1, description='Kc', max=1.0, min=-0.2, step=0.05), FloatSlider(valu…"
     ]
    },
    "metadata": {},
    "output_type": "display_data"
   },
   {
    "data": {
     "text/plain": [
      "<function __main__.pidPlot(Kc, tauI, tauD)>"
     ]
    },
    "execution_count": 4,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "n = 100 # time points to plot\n",
   "tf = 20.0 # final time\n",
   "SP_start = 2.0 # time of set point change\n",
   "\n",
   "def process(y,t,u):\n",
   "    Kp = 4.0\n",
   "    taup = 3.0\n",
   "    thetap = 1.0\n",
   "    if t<(thetap+SP_start):\n",
   "        dydt = 0.0  # time delay\n",
   "    else:\n",
   "        dydt = (1.0/taup) * (-y + Kp * u)\n",
   "    return dydt\n",
   "\n",
   "def pidPlot(Kc,tauI,tauD):\n",
   "    t = np.linspace(0,tf,n) # create time vector\n",
   "    P= np.zeros(n)          # initialize proportional term\n",
   "    I = np.zeros(n)         # initialize integral term\n",
   "    D = np.zeros(n)         # initialize derivative term\n",
   "    e = np.zeros(n)         # initialize error\n",
   "    OP = np.zeros(n)        # initialize controller output\n",
   "    PV = np.zeros(n)        # initialize process variable\n",      "    SP = np.zeros(n)        #
initialize setpoint\n",
   "    SP_step = int(SP_start/(tf/(n-1))+1) # setpoint start\n",
   "    SP[0:SP_step] = 0.0     # define setpoint\n",
   "    SP[SP_step:n] = 4.0     # step up\n",
   "    y0 = 0.0                # initial condition\n",
   "    # loop through all time steps\n",
   "    for i in range(1,n):\n",
   "        # simulate process for one time step\n",
   "        ts = [t[i-1],t[i]]         # time interval\n",
   "        y = odeint(process,y0,ts,args=(OP[i-1],))  # compute next step\n",
   "        y0 = y[1]                  # record new initial condition\n",
```

```
"        # calculate new OP with PID\n",
"        PV[i] = y[1]               # record PV\n",
"        e[i] = SP[i] - PV[i]       # calculate error = SP - PV\n",
"        dt = t[i] - t[i-1]         # calculate time step\n",
"        P[i] = Kc * e[i]           # calculate proportional term\n",
"        I[i] = I[i-1] + (Kc/tauI) * e[i] * dt  # calculate integral term\n",
"        D[i] = -Kc * tauD * (PV[i]-PV[i-1])/dt # calculate derivative term\n",
```

```
    "        OP[i] = P[i] + I[i] + D[i] # calculate new controller output\n",     "        \n",
    "    # plot PID response\n",
    "    plt.figure(1,figsize=(15,7))\n",
    "    plt.subplot(2,2,1)\n",
    "    plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')\n",
    "    plt.plot(t,PV,'r:',linewidth=2,label='Process Variable (PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,2)\n",
    "    plt.plot(t,P,'g.-',linewidth=2,label=r'Proportional = $K_c \\; e(t)$')\n",
    "    plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\\frac{K_c}{\\tau_I} \\int_{i=0}^{n_t} e(t) \\; dt $')\n",
    "    plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c \\tau_D \\frac{d(PV)}{dt}$')
\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,3)\n",
    "    plt.plot(t,e,'m--',linewidth=2,label='Error (e=SP-PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,4)\n",
    "    plt.plot(t,OP,'b--',linewidth=2,label='Controller Output (OP)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.xlabel('time')\n",
    "    \n",
    "Kc_slide = wg.FloatSlider(value=0.1,min=-0.2,max=1.0,step=0.05)\n",
    "tauI_slide = wg.FloatSlider(value=4.0,min=0.01,max=5.0,step=0.1)\n",     "tauD_slide =
wg.FloatSlider(value=0.0,min=0.0,max=1.0,step=0.1)\n",
    "wg.interact(pidPlot, Kc=Kc_slide, tauI=tauI_slide, tauD=tauD_slide)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "9d5b3b16",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"   },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.6"
  }
```

 },
 "nbformat": 4,
 "nbformat_minor": 5
}

Kode yang diberikan adalah sebuah Notebook Jupyter dalam format JSON, yang dirancang untuk simulasi kontrol PID (Proportional-Integral-Derivative) menggunakan Python. Berikut penjelasan dari isi notebook:

1. **Impor Modul**

   Di cell pertama, modul berikut diimpor:

   - `numpy`: Untuk manipulasi array dan perhitungan numerik.
   - `matplotlib`: Untuk membuat plot dan visualisasi.
   - `scipy.integrate.odeint`: Untuk menyelesaikan persamaan diferensial secara numerik.
   - `ipywidgets`: Untuk membuat widget interaktif.
   - `IPython.display`: Untuk menampilkan elemen interaktif.

2. **Fungsi untuk Model Proses**

   - Parameter Proses:
     - `Kp`: Gain (penguatan) proses.
     - `taup`: Waktu tunda proses.
     - `thetap`: Waktu mati (dead time)
   - Fungsi ini menggunakan persamaan diferensial pertama untuk merepresentasikan respons proses terhadap input kontrol `u`. Ada penundaan waktu mati sebelum proses bereaksi terhadap perubahan input.

3. **Fungsi Simulasi PID**

   Fungsi pidPlot(Kc, tauI, tauD) melakukan simulasi respons kontrol PID:

   - Parameter:
     - `Kc`: Penguatan proporsional.
     - `tauI`: Konstanta waktu integral.
     - `tauD`: Konstanta waktu derivatif.
   - Langkah-langkah:
     1) Inisialisasi variabel untuk waktu, error, keluaran kontrol, dan respons proses.
     2) Setpoint (SP):
        - Awalnya 0.
        - Berubah menjadi 4 setelah waktu tertentu.
     3) Untuk setiap langkah waktu:

- Solusi persamaan diferensial numerik menggunakan odeint untuk menghitung respons proses.
- Menghitung kontribusi PID:
  - P: Proporsional terhadap error.
  - I: Integral dari error terhadap waktu.
  - D: Derivatif dari perubahan proses
- Menentukan output kontrol `OP` sebagai kombinasi P, I, dan D.

4) Menampilkan 4 subplot:
   - Setpoint vs. respons proses (PV).
   - Komponen P, I, D.
   - Error.
   - Output kontrol (OP).

4. Widget Interaktif

Bagian akhir kode menggunakan modul ipywidgets untuk membuat slider interaktif untuk parameter PID:

- `Kc_slide`: Slider untuk nilai `Kc`.
- `tauI_slide`: Slider untuk nilai `tauI`.
- `tauD_slide`: Slider untuk nilai `tauD`.

Widget interaktif ini memungkinkan pengguna untuk mengubah parameter PID secara real-time dan melihat pengaruhnya pada respons sistem menggunakan fungsi `wg.interact`.

Output

Ketika kode ini dijalankan di Jupyter Notebook:

- Visualisasi Simulasi PID:
  - Pengguna dapat melihat pengaruh parameter PID terhadap respons sistem.
  - Grafik menunjukkan hubungan antara setpoint, error, output kontrol, dan respons proses.
- Interaktivitas:
  - Pengguna dapat menyesuaikan parameter PID (Kc, tauI, tauD) dengan slider untuk mengeksplorasi kinerja kontrol.

Kegunaan

Kode ini digunakan untuk:

- Belajar Konsep PID: Memahami bagaimana pengontrol PID bekerja.
- Simulasi: Mengetes dan menyetel parameter PID sebelum diterapkan pada sistem nyata.
- Pengajaran: Memberikan cara visual dan interaktif untuk memahami kontrol PID.

F. ITCLab-5

```
#include <Arduino.h>

// constants
const int baud = 115200;    // serial baud rate

// 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
 float cel, cel1, degC, degC1;
float P, I, D, Kc, tauI, tauD;
float KP, KI, KD, opo, ophi, oplo, error, dpv; float
sp = 35, //set point  pv = 0,     //current
temperature  pv_last = 0,  //prior temperature  ierr
= 0,   //integral error
dt = 0,     //time between measurements  op = 0;
//PID controller output  unsigned long ts = 0,
new_ts = 0; //timestamp  const float
upper_temperature_limit = 58;

// global variables
float Q1 = 0;          // value written to Q1 pin  float Q2 = 0;
// value written to Q2 pin  int iwrite_value = 25;     // integer
value for writing  int iwrite_led = 255;      // integer value for
writing  int iwrite_min = 0;      // integer value for writing
 void setup() {
 // put your setup code here, to run once:

 ts = millis();

 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);
```

[redacted]

```
 // 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);
ledcWrite(ledChannel,0);
}
void Q1on(){
   ledcWrite(Q1Channel,iwrite_value);
   //Serial.println(Q1);
}
void Q1off(){
   ledcWrite(Q1Channel,iwrite_min);
   //Serial.println(Q1);
}
void Q2on(){
   ledcWrite(Q2Channel,iwrite_value);
   //Serial.println(Q2);
}
void Q2off(){
   ledcWrite(Q2Channel,iwrite_min);
   //Serial.println(Q2);
}
void ledon(){
   ledcWrite(ledChannel,iwrite_led);
}
void ledoff(){
   ledcWrite(ledChannel,iwrite_min);
}
void cektemp(){
 degC = analogRead(pinT1) * 0.322265625 ;  // use for 3.3v AREF   cel =
degC/10;
 degC1 = analogRead(pinT2) * 0.322265625 ;  // use for 3.3v AREF   cel1 =
degC1/10;

 Serial.print("Temperature T1: ");
 Serial.print(cel);  // print the temperature T1 in Celsius
 Serial.print("°C");
 Serial.print(" ~ "); // separator between Celsius and Fahrenheit
 Serial.print("Temperature T2: ");
 Serial.print(cel1);  // print the temperature T2 in Celsius
 Serial.println("°C");
} float pid(float sp, float pv, float pv_last, float& ierr, float dt) {  float
Kc = 10.0; // K / %Heater   float tauI = 50.0; // sec
```

```
 float tauD = 1.0; // sec  //
PID coefficients  float KP
= Kc;  float KI = Kc / tauI;
float KD = Kc*tauD;
 // upper and lower bounds on heater level
float ophi = 100;  float oplo = 0;  // calculate
the error  float error = sp - pv;
 // calculate the integral error  ierr =
ierr + KI * error * dt;  // calculate the
measurement derivative  float dpv = (pv -
pv_last) / dt;  // calculate the PID output
 float P = KP * error; //proportional contribution
float I = ierr; //integral contribution  float D = -KD *
dpv; //derivative contribution  float op = P + I + D;
 // implement anti-reset windup  if
((op < oplo) || (op > ophi)) {
   I = I - KI * error * dt;
   // clip output
   op = max(oplo, min(ophi, op));
 }
 ierr = I;
 Serial.println("sp="+String(sp) + " pv=" + String(pv) + " dt=" + String(dt) + " op=" +
String(op) + " P=" + String(P) + " I=" + String(I) + " D=" + String(D));  return op;
} void loop() {
new_ts = millis();
 if (new_ts - ts > 1000) {

 // put your main code here, to run repeatedly:  cektemp();
 if (cel > upper_temperature_limit){
Q1off();    ledon();
 } else {
Q1on();
ledoff();
 }
 if (cel1 > upper_temperature_limit){
Q2off();    ledon();
 } else {
Q2on();
ledoff();
 }  pv = cel;  // Temperature T1
dt = (new_ts - ts) / 1000.0;  ts =
new_ts;
 op = pid(sp,pv,pv_last,ierr,dt);
ledcWrite(Q1Channel,op);  pv_last = pv;
```

```
 delay (200);
 }
}
```

Kode di atas merupakan program untuk mikrokontroler berbasis Arduino, digunakan untuk mengontrol suhu menggunakan metode PID (Proportional-Integral-Derivative) pada perangkat keras tertentu. Berikut adalah penjelasan terperinci:

1. **Konfigurasi Awal**

   Konstanta dan Pin:
   - `baud`: Kecepatan komunikasi serial dengan PC (115200 bps).
   - `pinT1` dan `pinT2`: Pin input analog untuk membaca suhu dari sensor suhu T1 dan T2.
   - `pinQ1` dan `pinQ2`: Pin output PWM untuk kontrol aktuator terkait (misalnya, pemanas).
   - `pinLED`: Pin output PWM untuk kontrol LED.

   Properti PWM:
   - `freq`: Frekuensi PWM (5 kHz).
   - `resolution`: Resolusi PWM (8 bit, nilai 0-255).

2. **Fungsi `setup`**
   - Mengatur komunikasi serial dan inisialisasi PWM untuk pin kontrol (Q1, Q2, LED).
   - Semua kanal PWM (Q1Channel, Q2Channel, ledChannel) diatur dengan frekuensi dan resolusi yang sama, kemudian dihubungkan ke pin masingmasing.

3. **Fungsi untuk Kontrol Aktuator**
   - `Q1on` / `Q1off`, `Q2on` / `Q2off`: Mengatur keluaran PWM ke pin Q1 dan Q2, masing-masing untuk menghidupkan dan mematikan aktuator.
   - `ledon` / `ledoff`: Menghidupkan atau mematikan LED dengan mengatur keluaran PWM.

4. **Fungsi untuk Membaca Suhu** `cektemp`

   :
   - Membaca nilai analog dari sensor suhu (T1 dan T2) melalui pinT1 dan pinT2.
   - Mengkonversi nilai analog menjadi suhu dalam derajat Celsius menggunakan faktor skala `0.322265625`.

- Mencetak suhu T1 dan T2 ke serial monitor.

5. **Fungsi PID**
    - Fungsi PID mengontrol keluaran aktuator berdasarkan error (selisih antara set point `sp` dan suhu aktual `pv`).
    - Komponen PID:
        - Proportional (`P`): Berbanding lurus dengan error.
        - Integral (`I`): Akumulasi error untuk memperbaiki kesalahan jangka panjang.
        - Derivative (`D`): Mengurangi efek perubahan cepat pada error.
    - Membatasi output (anti-reset windup) agar tetap dalam rentang oplo hingga ophi.

6. Fungsi `loop`

Dieksekusi secara terus-menerus untuk membaca suhu, menghitung keluaran PID, dan mengontrol aktuator.

- Membaca Suhu:
    - Memanggil `cektemp` untuk membaca suhu T1 (`cel`) dan T2 (`cel1`).
- Logika Kontrol:
    - Jika suhu T1 atau T2 melebihi batas atas (`upper_temperature_limit`), aktuator dimatikan (`Q1off` atau `Q2off`) dan LED dihidupkan.
    - Jika tidak, aktuator dihidupkan dan LED dimatikan.
- PID Control:
    - Nilai suhu T1 (`pv`) digunakan untuk perhitungan PID.
    - PID menghasilkan nilai keluaran (`op`) yang kemudian digunakan untuk mengatur keluaran PWM di Q1.
- Pengaturan Waktu:
    - `dt` dihitung sebagai waktu antara pengukuran, digunakan dalam perhitungan PID.

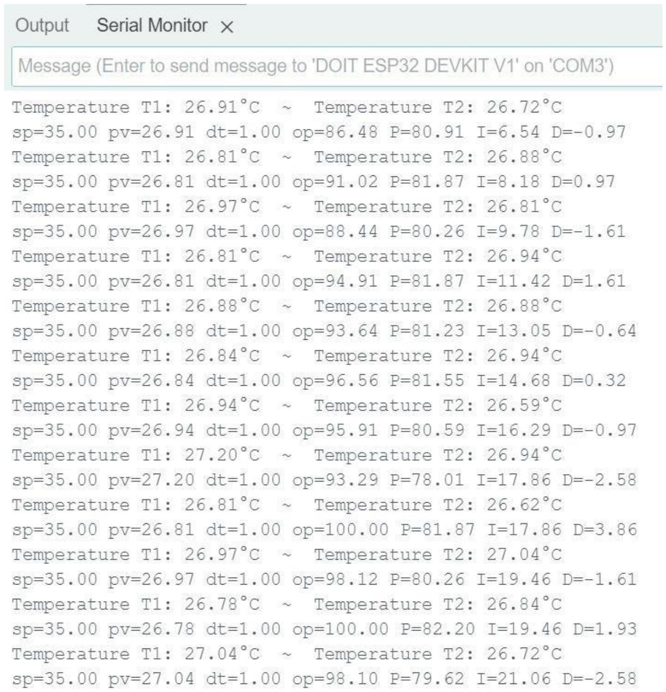
Kegunaan

Kode ini bertujuan untuk:

- Membaca suhu dari sensor (T1 dan T2).
- Menggunakan metode PID untuk mengontrol aktuator berbasis suhu (Q1).
- Menjaga suhu dalam batas tertentu dengan logika kontrol sederhana.

- Memberikan umpan balik melalui LED dan serial monitor untuk memantau kinerja sistem.

Hasil Screenshot



Output   Serial Monitor ×

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM3')

```
Temperature T1: 26.91°C  ~  Temperature T2: 26.72°C
sp=35.00 pv=26.91 dt=1.00 op=86.48 P=80.91 I=6.54 D=-0.97
Temperature T1: 26.81°C  ~  Temperature T2: 26.88°C
sp=35.00 pv=26.81 dt=1.00 op=91.02 P=81.87 I=8.18 D=0.97
Temperature T1: 26.97°C  ~  Temperature T2: 26.81°C
sp=35.00 pv=26.97 dt=1.00 op=88.44 P=80.26 I=9.78 D=-1.61
Temperature T1: 26.81°C  ~  Temperature T2: 26.94°C
sp=35.00 pv=26.81 dt=1.00 op=94.91 P=81.87 I=11.42 D=1.61
Temperature T1: 26.88°C  ~  Temperature T2: 26.88°C
sp=35.00 pv=26.88 dt=1.00 op=93.64 P=81.23 I=13.05 D=-0.64
Temperature T1: 26.84°C  ~  Temperature T2: 26.94°C
sp=35.00 pv=26.84 dt=1.00 op=96.56 P=81.55 I=14.68 D=0.32
Temperature T1: 26.94°C  ~  Temperature T2: 26.59°C
sp=35.00 pv=26.94 dt=1.00 op=95.91 P=80.59 I=16.29 D=-0.97
Temperature T1: 27.20°C  ~  Temperature T2: 26.94°C
sp=35.00 pv=27.20 dt=1.00 op=93.29 P=78.01 I=17.86 D=-2.58
Temperature T1: 26.81°C  ~  Temperature T2: 26.62°C
sp=35.00 pv=26.81 dt=1.00 op=100.00 P=81.87 I=17.86 D=3.86
Temperature T1: 26.97°C  ~  Temperature T2: 27.04°C
sp=35.00 pv=26.97 dt=1.00 op=98.12 P=80.26 I=19.46 D=-1.61
Temperature T1: 26.78°C  ~  Temperature T2: 26.84°C
sp=35.00 pv=26.78 dt=1.00 op=100.00 P=82.20 I=19.46 D=1.93
Temperature T1: 27.04°C  ~  Temperature T2: 26.72°C
sp=35.00 pv=27.04 dt=1.00 op=98.10 P=79.62 I=21.06 D=-2.58
```

G. ITCLab-6

Di ITCLab ke-6 ada tiga kode yaitu, kode Arduino, Python, dan Notebook Jupyter.

```
#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 upper_temperature_limit = 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);
```

[redacted]

```
 }
 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 >= upper_temperature_limit) {
    Q1 = 0.0;      Q2 =
0.0;
    ledcWrite(Q1Channel,0);      ledcWrite(Q2Channel,0);
    //Serial.println("High Temp 1 (> upper_temperature_limit): ");
    Serial.println(degC);
  }
  mV = (float) analogRead(pinT2) * 0.322265625;
  //degC = (mV - 500.0)/10.0;    degC
= mV/10.0;
  if (degC >= upper_temperature_limit) {
    Q1 = 0.0;      Q2 =
0.0;
    ledcWrite(Q1Channel,0);      ledcWrite(Q2Channel,0);
    //Serial.println("High Temp 2 (> upper_temperature_limit): ");
    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();
}
```

Kode ini adalah firmware untuk Arduino yang digunakan untuk mengontrol dan memantau perangkat berbasis iTCLab Shield. Berikut adalah penjelasan elemen-elemen utama dalam kode ini:

1. **Konstanta dan Konfigurasi**
   - `vers`: Menyimpan versi firmware.
   - `baud`: Menentukan kecepatan baud untuk komunikasi serial (115200 bps).
   - `sp` dan `nl`: Separator dan terminator untuk parsing perintah serial.
   - Pin konfigurasi:
     - Pin analog `pinT1` dan `pinT2` untuk membaca suhu dari sensor.
     - Pin PWM `pinQ1`, `pinQ2`, dan `pinLED` untuk mengontrol output sinyal (heater dan LED).
   - Konfigurasi PWM:
     - Frekuensi PWM diatur ke 5000 Hz dengan resolusi 8 bit.
     - Tiga kanal PWM digunakan: satu untuk LED, dua untuk output Q1 dan Q2.
   - Batas suhu maksimal (`upper_temperature_limit`): Digunakan untuk pengamanan agar perangkat mati jika suhu terlalu tinggi.

2. **Variabel Global**
   - `Buffer`: Menyimpan input serial yang diterima.
   - `cmd`: Perintah yang diterima melalui serial.
   - `pv`: Nilai yang dikaitkan dengan perintah.
   - `level, Q1, Q2`: Nilai yang akan diterapkan pada LED dan heater (Q1, Q2).
   - `n`: Jumlah sampel untuk pengukuran suhu rata-rata.

3. **Fungsi Parsing Serial** `parseSerial()`
   :
   - Membaca data serial hingga menemukan terminator (`nl`).
   - Memisahkan perintah dan data (dengan menggunakan separator `sp`).
   - Mengubah perintah menjadi huruf besar untuk standardisasi.
   - Mengonversi data menjadi nilai numerik (`toFloat()`).

4. **Fungsi Eksekusi Perintah**
   `dispatchCommand()`:
   - Mengatur respons terhadap perintah yang diterima:
     - `Q1` dan `Q2`: Mengontrol nilai heater dalam kisaran 0-25%. Nilai ini dikonversi ke skala PWM (0-255).
     - `T1` dan `T2`: Membaca suhu dari sensor pada pin T1 atau T2, menghitung nilai rata-rata, dan mencetaknya ke serial.
     - `V` atau `VER`: Menampilkan versi firmware.
     - `LED`: Mengatur tingkat kecerahan LED dalam kisaran 0-100%.
     - `X`: Menghentikan output Q1 dan Q2 (mematikan heater).

5. **Fungsi Pengamanan Suhu**
   `checkTemp()`:
   - Membaca suhu dari sensor T1 dan T2.
   - Jika suhu melebihi batas (`upper_temperature_limit`), output heater (Q1 dan Q2) dimatikan untuk mencegah kerusakan.

6. **Fungsi `setup()`**
   - Menginisialisasi komunikasi serial dengan baud rate yang ditentukan.
   - Mengatur kanal PWM untuk Q1, Q2, dan LED serta mengaitkannya ke pin masing-masing.
   - Memastikan semua output PWM dimulai dari 0 (mati).

7. **Fungsi `loop()`**
   - `parseSerial()`: Membaca dan memproses perintah dari serial.
   - `dispatchCommand()`: Mengeksekusi perintah yang diterima.
   - `checkTemp()`: Memantau suhu dan mengambil tindakan pengamanan jika suhu melebihi batas.

```
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
```

[redacted]

```
  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
```

Kode ini adalah implementasi Python yang dirancang untuk mengontrol perangkat keras yang terhubung ke Arduino, seperti LED, pemanas (heaters), dan sensor suhu (temperatures), melalui komunikasi serial. Kelas utama dalam kode ini disebut `iTCLab`.

1. Import Module

```
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
```

- Mengimpor modul standar Python dan modul tambahan:
  - `serial`: Untuk komunikasi serial.
  - `numpy`: Untuk manipulasi data numerik, digunakan untuk menyimpan data dalam bentuk file.
- Jika modul `serial` belum terinstal, akan diinstal secara otomatis menggunakan `pip`.

2. Kelas iTCLab

Kelas ini merepresentasikan perangkat keras yang dikendalikan melalui Arduino. **a) Konstruktor (`__init__`)**

```
def __init__(   ,    =    ,    =115200):
        =    .        ()
        ('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)
```

- Port: Menentukan port serial yang digunakan oleh Arduino.
- Baudrate: Kecepatan komunikasi serial (115200 bps).
- Memanggil metode `findPort` untuk mendeteksi port otomatis.
- Menginisialisasi komunikasi serial dengan Arduino.
- `time.sleep(3)`: Memberikan waktu bagi Arduino untuk siap.

b) Pendeteksian Port (`findPort`)

```
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
```

- Tujuan: Menemukan port Arduino secara otomatis berdasarkan ID perangkat (VID:PID).
- Jika tidak ditemukan, memberikan petunjuk kepada pengguna untuk menginput port secara manual.

c) Metode Utama

- Komunikasi Dasar
  - `read(cmd)`: Membaca data dari Arduino berdasarkan perintah (`cmd`).
  - `write(cmd, pwm)`: Mengirimkan perintah dengan nilai parameter PWM.
- Kontrol Perangkat
  - `LED(pwm)`: Mengatur intensitas LED (dalam skala 0-100%).

- Q1(pwm) dan Q2(pwm): Mengatur tingkat pemanasan heater 1 dan 2 (dalam skala 0-100%).

- Pengambilan Data Sensor
  - T1 dan T2: Membaca suhu dari sensor suhu 1 dan 2.
- Fungsi Lain
  - stop(): Menghentikan operasi.
  - version(): Membaca versi firmware Arduino.

d) Penyimpanan Data (save_txt)

```
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='')
```

Menyimpan data waktu, pemanasan, suhu, dan setpoint dalam format CSV (data.txt) untuk analisis lebih lanjut.

e) Penutupan Koneksi

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

Menutup koneksi serial dengan Arduino dengan aman.

```
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
```

[redacted]

[redacted]

```
    I = ierr     D = -KD * dpv
op = opo + 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()
```

[redacted]

[redacted]

```
# Temperature
# set point (degC)
Tsp1 = np.ones(loops) * 25.0
Tsp1[60:] = 45.0
Tsp1[360:] = 30.0
Tsp1[660:] = 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=(10,7)) 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
```

[redacted]

[redacted]

```
    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)

    # 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')
```

[redacted]

[redacted]

```
    # 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()
```

Kode ini adalah implementasi simulasi dan pengendalian sistem menggunakan **PID controller** (Proportional-Integral-Derivative) pada perangkat keras **TCLab** (Temperature Control Laboratory). Tujuan utamanya adalah untuk mengevaluasi performa model sistem dan pengendalian PID dalam mengatur suhu berdasarkan setpoint yang berubah seiring waktu. Berikut adalah penjelasan detail dari setiap bagian:

1. **Import Library**
    - `itclab`: Berisi antarmuka untuk mengendalikan perangkat keras TCLab (misalnya, membaca suhu dan mengontrol heater).
    - `numpy`, `time`, dan `matplotlib.pyplot`: Digunakan untuk komputasi numerik, pengukuran waktu, dan visualisasi data.
    - `scipy.integrate.odeint`: Digunakan untuk menyelesaikan persamaan diferensial numerik.

2. **Fungsi PID**

    Fungsi `pid` adalah implementasi dari PID controller.
    - Input:
        - `sp` (setpoint): Suhu target.
        - `pv` (process variable): Suhu saat ini.

- `pv_last`: Suhu sebelumnya.
- `ierr`: Kesalahan integral sebelumnya. `dt`: Interval waktu.

- Output:
  - `op`: Output pengendali (persentase daya heater).
  - `P, I, D`: Kontribusi masing-masing komponen PID.
- Parameter:
  - `Kc, tauI, tauD`: Parameter pengendali PID (gain, waktu integral, waktu derivatif).
- Logika:
  - Menghitung kesalahan (`error`), kesalahan integral (`ierr`), dan derivatif perubahan suhu (`dpv`).
  - Menghitung kontribusi P, I, D, dan hasil akhirnya (`op`).
  - Anti-reset windup: Menjaga agar `op` tetap dalam batas 0-100%.

3. Model Sistem

- FOPDT Model (First Order Plus Dead Time): Parameter:
  - o `Kp`: Gain proses. o `tauP`: Konstanta waktu.
  - o `thetaP`: Waktu mati (dead time).

   Simulasi suhu linier berdasarkan model matematika.
- Energy Balance Model:
  - Persamaan termal non-linier berdasarkan energi masuk/keluar (konduksi, konveksi, radiasi).
  - Fungsi `heat` menyelesaikan laju perubahan suhu dengan metode `odeint`.

4. Loop Utama

- Inisialisasi:
  - `a = itclab.iTCLab()`: Menghubungkan ke perangkat TCLab.
  - `loops`: Jumlah iterasi berdasarkan waktu simulasi.
  - `Tsp1, T1, Q1`: Setpoint, suhu aktual, dan output heater.
- Iterasi (Setiap Detik):
  - Membaca suhu aktual dari TCLab.

- Menjalankan simulasi model linier (FOPDT) dan non-linier (energy balance).
- Menghitung output PID berdasarkan suhu aktual dan setpoint.
- Memperbarui cumulative error (kesalahan kumulatif).
- Mencetak data dan memperbarui plot secara real-time.

- Penanganan Akhir:
    - Jika program dihentikan (Ctrl-C atau error), heater dimatikan dan koneksi ditutup.

5. **Visualisasi**

   Membuat empat subplot:

   - Temperatur: Suhu aktual vs setpoint.
   - Heater: Output daya heater.
   - Prediksi Model: Perbandingan antara suhu aktual, model energi, dan model linier.
   - Kesalahan Kumulatif: Kesalahan antara suhu aktual, prediksi, dan setpoint.

H. ITCLab-7

Di ITCLab ke-7 ada 2 kode yaitu, kode Arduino dan Python.

```
#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 upper_temperature_limit = 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 >= upper_temperature_limit) {
     Q1 = 0.0;      Q2 =
0.0;
     ledcWrite(Q1Channel,0);      ledcWrite(Q2Channel,0);
     //Serial.println("High Temp 1 (> upper_temperature_limit): ");
     Serial.println(degC);
   }
   mV = (float) analogRead(pinT2) * 0.322265625;
   //degC = (mV - 500.0)/10.0;    degC
= mV/10.0;
   if (degC >= upper_temperature_limit) {
     Q1 = 0.0;      Q2 =
0.0;
     ledcWrite(Q1Channel,0);      ledcWrite(Q2Channel,0);
     //Serial.println("High Temp 2 (> upper_temperature_limit): ");
     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();
}
```

Kode ini adalah program untuk Arduino yang dirancang untuk mengontrol perangkat keras, seperti **heater**, **LED**, dan **sensor suhu**, dengan menggunakan modul PWM (Pulse Width Modulation) pada papan Arduino yang kompatibel, seperti ESP32. Program ini juga menyediakan antarmuka serial untuk menerima perintah dan memberikan respon yang sesuai. Berikut adalah penjelasan terperinci:

1. **Konstanta dan Pin Konfigurasi**
   - `vers`: Versi firmware (v1.04).
   - `baud`: Kecepatan komunikasi serial (115200 bps).
   - `sp` dan `nl`: Separator dan terminator untuk parsing perintah.
   - `pinT1`, `pinT2`, `pinQ1`, `pinQ2`, `pinLED`: Nomor pin untuk sensor suhu (T1, T2), heater (Q1, Q2), dan LED.
   - PWM Properties:
     - Frekuensi PWM diatur pada 5000 Hz.
     - Resolusi PWM adalah 8-bit (nilai antara 0-255).
     - Terdapat tiga channel PWM untuk Q1, Q2, dan LED.

2. **Variabel Global**
   - Buffer: Buffer untuk menyimpan input serial.
   - cmd: Perintah yang diterima.
   - pv: Nilai data dari perintah.
   - Q1, Q2: Nilai output PWM ke heater Q1 dan Q2.
   - `level`: Level intensitas LED dalam persen.
   - `n`: Jumlah sampel untuk pembacaan rata-rata suhu.

3. **Fungsi Utama**

   `parseSerial()`

- Membaca input serial hingga menemukan karakter `nl` (`\n`).
- Memisahkan input menjadi perintah (`cmd`) dan data (`pv`).
- Mengubah perintah menjadi huruf besar untuk konsistensi.

`dispatchCommand()`

- Mengeksekusi perintah berdasarkan nilai `cmd`. Perintah yang didukung:
  - Q1: Mengontrol nilai PWM untuk heater Q1. Nilai maksimum dibatasi pada 25% (50 PWM dari 255).
  - Q2: Mengontrol nilai PWM untuk heater Q2 dengan cara serupa.
  - T1 dan T2: Membaca suhu dari sensor analog (T1 atau T2) dan mengembalikan nilai rata-rata dalam derajat Celsius.
  - V atau VER: Mengembalikan versi firmware.
  - LED: Mengontrol intensitas LED (0-100%).
  - X: Mematikan heater (Q1 dan Q2) dengan menulis nilai PWM 0.

`checkTemp()`

- Membaca suhu dari sensor T1 dan T2.
- Jika suhu melebihi batas atas (59°C), heater Q1 dan Q2 dimatikan untuk mencegah overheating.

`setup()`

- Menginisialisasi komunikasi serial.
- Mengatur channel PWM untuk heater dan LED, serta menghubungkannya ke pin masing-masing.
- Menulis nilai awal 0 ke semua channel PWM.

`loop()`

- Mengulangi tiga fungsi utama:
  - `parseSerial()`: Membaca perintah dari serial.
  - `dispatchCommand()`: Menjalankan perintah yang diterima.
  - `checkTemp()`: Memastikan suhu aman.

4. Detail Perintah

- Perintah Q1/Q2:
  - Nilai input (pv) dibatasi antara 0-25%.
  - Nilai PWM dihitung sebagai pv * 2.0 untuk menghasilkan skala 0-50 (mewakili hingga 25% dari 255).

- Perintah T1/T2:
  - Sensor analog membaca nilai dalam miliVolt (mV), di mana 1 unit ADC setara dengan 0.322 mV.
  - Suhu dihitung sebagai mV / 10.0 dalam Celsius.
- Perintah LED:
  - Level intensitas LED (0-100%) dikonversi ke nilai PWM skala 0-50.
- Keamanan:
  - Jika suhu melebihi 59°C, semua heater dimatikan dengan menulis 0 ke channel PWM heater.

5. **Penggunaan**
   - Hubungkan Arduino ke komputer atau perangkat lain.
   - Kirim perintah melalui antarmuka serial (misalnya, `Q1 10` untuk mengatur Q1 ke 10%).
   - Perangkat akan merespon perintah sesuai logika dalam fungsi `dispatchCommand()`.

```
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
```

[redacted]

[redacted]

```
= '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
```

Kode ini adalah implementasi sebuah kelas Python yang berfungsi untuk mengontrol perangkat berbasis Arduino melalui komunikasi serial. Berikut adalah penjelasan detail dari masing-masing bagian kode:

1. Import Library

```
mport 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
```

Kode ini mengimpor modul standar Python seperti `sys` dan `time`, bersama dengan modul `numpy` untuk pengolahan data. Jika modul `serial` (dari pustaka `pyserial`) tidak diinstal, maka kode akan mencoba menginstalnya secara otomatis.

2. Kelas iTCLab

Kelas ini mendefinisikan metode untuk berkomunikasi dengan Arduino menggunakan port serial.

Inisialisasi (`__init__`)

```
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)
```

- Mencari port serial Arduino menggunakan metode `findPort`.
- Membuka koneksi serial pada port yang ditemukan dengan baud rate 115200.
- Membersihkan buffer input/output serial.
- Menunggu 3 detik agar perangkat siap. **Metode findPort**

```
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')
```

```
                ('--- 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
```

- Mendeteksi port serial Arduino berdasarkan VID:PID USB yang terdaftar.
- Jika tidak ditemukan, meminta pengguna untuk memasukkan nama port secara manual.

Metode Pembacaan dan Penulisan

- `read`: Membaca data dari Arduino berdasarkan perintah (`cmd`) yang dikirim.
- `write`: Menulis data ke Arduino, seperti mengatur nilai PWM pada perangkat keras (misalnya, pemanas atau LED).

Properti dan Metode untuk Kontrol

- `T1` dan `T2`: Membaca suhu dari sensor 1 dan 2.
- `LED`: Mengontrol intensitas LED (0-100%).
- `Q1` dan `Q2`: Mengontrol pemanas (heater) 1 dan 2 menggunakan PWM (0100%).

Fungsi Penyimpanan Data

```
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='')
```

- Menyimpan data pengukuran ke file teks (`data.txt`) dalam format CSV.
- Data yang disimpan mencakup waktu, nilai heater, suhu, dan setpoint.

Menutup Koneksi

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

- Menutup koneksi serial secara aman.
- Memberikan notifikasi jika terjadi masalah saat memutuskan koneksi.

Kegunaan

Kode ini dirancang untuk eksperimen atau pengendalian perangkat berbasis Arduino, seperti pengendalian suhu menggunakan heater dan LED, yang mungkin digunakan dalam aplikasi laboratorium atau pembelajaran IoT.

I. ITCLab-8

```
#include <WiFi.h>
#include <PubSubClient.h> #include
<Arduino.h>

const char* ssid = "wifi_name"; // Enter your WiFi name
const char* password =  "wifi_password"; // Enter WiFi password
#define mqttServer "broker.hivemq.com"
#define mqttPort 1883

WiFiServer server(80);
WiFiClient espClient;
PubSubClient client(espClient);

String Topic;
String Payload;

// constants
const int baud = 115200;     // serial baud rate

// 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

float cel, cel1, degC, degC1;
const float upper_temperature_limit = 58;

// global variables
float Q1 = 0;          // value written to Q1 pin float Q2 = 0;
// value written to Q2 pin int iwrite_value = 25;      // integer
value for writing int iwrite_min = 0;       // integer value for
writing
 void setup() {
 // put your setup code here, to run once:
```

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

 // configure pinQ1 PWM functionalitites
        (         ,    ,                 );

 // 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);
ledcWrite(ledChannel,0);

 // Connect to WiFi network
 Serial.println();
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
delay(500);    Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 // Connect to Server IoT (CloudMQTT)  client.setServer(mqttServer,
mqttPort);  client.setCallback(receivedCallback);

 while (!client.connected()) {
  Serial.println("Connecting to CLoud IoT ...");
```

```
//    if (client.connect("ESP32Client", mqttUser, mqttPassword )) {
if (client.connect("iTCLab Suhu On/Off")) {
```

```
        .       ("connected");
        .     ("Message received: ");

  } else {
        .     ("failed with state ");
     .    (      .    ());          (2000);   }
       .          ("heater1bas");
     .          ("heater2bas");
 }
} void     (){
        (          ,          );
  //Q1 = iwrite_value/255*100;
  //Serial.println(Q1);
}
```

```
void Q1off(){
   ledcWrite(Q1Channel,iwrite_min);
   //Q1 = iwrite_min/255*100;
   //Serial.println(Q1);
}
void Q2on(){
   ledcWrite(Q2Channel,iwrite_value);
   //Q2 = iwrite_value/255*100;
   //Serial.println(Q2);
}
void Q2off(){
   ledcWrite(Q2Channel,iwrite_min);
   //Q2 = iwrite_min/255*100;
   //Serial.println(Q2);
}
void ledon(){
   ledcWrite(ledChannel,iwrite_value);
}
void ledoff(){
   ledcWrite(ledChannel,iwrite_min);
}
void cektemp(){
  degC = analogRead(pinT1) * 0.322265625 ;   // use for 3.3v AREF   cel =
degC/10;
  degC1 = analogRead(pinT2) * 0.322265625 ;   // use for 3.3v AREF   cel1 =
degC1/10;

  Serial.print("Temperature: ");
  Serial.print(cel);  // print the temperature T1 in Celsius
  Serial.print("°C");
  Serial.print(" ~ "); // separator between Celsius and Fahrenheit
  Serial.print(cel1);  // print the temperature T2 in Celsius
  Serial.println("°C");
}
void receivedCallback(char* topic, byte* payload, unsigned int length) {
```

```
/* we got '1' -> Q1_on */ if
((char)      [0] == '1') {
       ();
       .      ("Q1 On");
 }

/* we got '2' -> Q1_off */ if
((char)      [0] == '2') {
       ();
       .      ("Q1 Off");
 }

/* we got '3' -> Q2_on */ if
((char)      [0] == '3') {
    ();
       .      ("Q2 On");
 }

/* we got '4' -> Q2_off */
```

```
 if ((char)payload[0] == '4') {
      Q2off();
      Serial.println("Q2 Off");
 }
} void loop() {
char suhu1[4];
char suhu2[4];
client.loop();

 // put your main code here, to run repeatedly:   cektemp();
 if (cel > upper_temperature_limit){
Q1off();     ledon();
 }  else {
Q1on();
ledoff();
 }
 if (cel1 > upper_temperature_limit){
Q2off();     ledon();
 }  else {
Q2on();
ledoff();
 }
 delay (100);


 Serial.print("Temperature T1: ");
 Serial.print(cel);
 Serial.print(" Celcius ");
 Serial.println(" send to Broker MQTT");

 dtostrf(cel, 1, 0, suhu1);   client.publish("Suhu1",suhu1);

 delay (200);
```

```
  .       ("Temperature T2: ");
  .       (     );
  .       (" Celcius ");
  .          (" send to Broker MQTT");
    (     , 1, 0,          );
 .          ("Suhu2",          );
  (200);
```

}

Kode ini adalah program untuk ESP32 yang mengontrol pemanas menggunakan sensor suhu dan mengirim data suhu ke broker MQTT (Message Queuing Telemetry Transport). Berikut adalah penjelasan untuk setiap bagian:

1. **Header dan Konstanta**
   - Library yang digunakan:
     - `WiFi.h`: Untuk koneksi WiFi.
     - `PubSubClient.h`: Untuk komunikasi dengan broker MQTT. `Arduino.h`: Fungsi dasar Arduino.
   - Konstanta WiFi dan MQTT:
     - `ssid` dan `password`: Nama dan kata sandi WiFi untuk koneksi internet.
     - `mqttServer` dan `mqttPort`: Server dan port broker MQTT (`broker.hivemq.com`).
   - Konstanta lainnya:
     - `baud`: Kecepatan baud untuk komunikasi serial (115200 bps).
     - Pin untuk sensor suhu, pemanas, dan LED (`pinT1`, `pinT2`, `pinQ1`, `pinQ2`, `pinLED`).
     - PWM properties (frekuensi dan resolusi PWM).

2. **Variabel Global**
   - `cel`, `cel1`: Variabel untuk menyimpan suhu dari sensor T1 dan T2.
   - `upper_temperature_limit`: Batas suhu atas (58°C) untuk mengontrol pemanas.
   - Variabel `Q1`, `Q2`: Nilai output PWM ke pemanas.
   - `iwrite_value`, `iwrite_min`: Nilai PWM (dalam skala 0-255) untuk mengontrol intensitas.

3. **Fungsi `setup()`**
   - Koneksi Serial:
     - Menginisialisasi komunikasi serial pada baud rate 115200.

- Konfigurasi PWM:
    - Mengatur properti PWM (frekuensi dan resolusi) untuk pin pemanas (`pinQ1`, `pinQ2`) dan LED (`pinLED`).
    - Mengikat saluran PWM dengan pin terkait. Menetapkan nilai awal PWM ke 0 (mati).
- Koneksi WiFi:
     Menghubungkan ESP32 ke jaringan WiFi dengan `WiFi.begin`. Looping hingga koneksi berhasil (`WiFi.status()`).
- Koneksi ke Broker MQTT:
    - Mengatur server MQTT dan callback untuk menerima pesan. Looping hingga koneksi berhasil.
    - Berlangganan ke topik `heater1bas` dan `heater2bas`.

4. **Fungsi Kontrol**
    - Kontrol Pemanas dan LED:
        - `Q1on()`, `Q1off()`, `Q2on()`, `Q2off()`: Menghidupkan/mematikan pemanas dengan menulis nilai PWM.
        - `ledon()`, `ledoff()`: Menghidupkan/mematikan LED.

5. **Fungsi `cektemp()`**
    - Membaca nilai analog dari sensor suhu (`pinT1`, `pinT2`), mengkonversi ke suhu dalam °C.
    - Menggunakan konstanta `0.322265625` untuk kalibrasi berdasarkan AREF 3.3V.
    - Menampilkan suhu di serial monitor.

6. **Callback MQTT**
   `receivedCallback(char* topic, byte* payload, unsigned int length)`
    - Mengevaluasi pesan yang diterima dari broker MQTT.
    - Pesan berupa angka yang menentukan tindakan:
        - `'1'`: Menghidupkan `Q1`. `'2'`: Mematikan `Q1`.
        - `'3'`: Menghidupkan `Q2`.
        - `'4'`: Mematikan `Q2`.

7. **Fungsi `loop()`** Loop Utama:
    - Memeriksa suhu sensor menggunakan `cektemp()`.
    - Membandingkan suhu dengan batas atas: Jika suhu > batas: ○ Matikan pemanas, nyalakan LED.
         Jika suhu <= batas:
            ○ Hidupkan pemanas, matikan LED.
    - Mengirim data suhu (`cel` dan `cel1`) ke broker MQTT pada topik `Suhu1` dan `Suhu2`.
    - Penundaan (`delay`) untuk interval pengiriman data.

Kegunaan

Kode ini digunakan untuk:

- Mengontrol pemanas berdasarkan suhu.
- Mengirim data suhu ke broker MQTT untuk monitoring.
- Menerima perintah kontrol manual dari broker MQTT.

J. ITCLab-9

```
#include <WiFi.h>
#include <PubSubClient.h> #include
<Arduino.h>

const char* ssid = "wifi_name"; // Enter your WiFi name
const char* password =  "wifi_password"; // Enter WiFi password
#define mqttServer "broker.hivemq.com"
#define mqttPort 1883

WiFiServer server(80);
WiFiClient espClient;
PubSubClient client(espClient);

String Topic;
String Payload;

// constants
const int baud = 115200;     // serial baud rate

// 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

float cel, cel1, degC, degC1; float P, I,
D, Kc, tauI, tauD;
float KP, KI, KD, op0, ophi, oplo, error, dpv; float
sp = 35, //set point  pv = 0,     //current
temperature
```

```
      = 0, //prior temperature     = 0,
//integral error
```

```
dt = 0,     //time between measurements  op = 0;
//PID controller output unsigned long ts = 0,
new_ts = 0; //timestamp const float
upper_temperature_limit = 58;

// global variables
float Q1 = 0;          // value written to Q1 pin float Q2 = 0;
// value written to Q2 pin int iwrite_value = 25;     // integer
value for writing int iwrite_led = 255;      // integer value for
writing int iwrite_min = 0;       // integer value for writing
 void setup() {
 // put your setup code here, to run once:

 ts = millis();

 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);
ledcWrite(ledChannel,0);

 // Connect to WiFi network
 Serial.println();
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
delay(500);    Serial.print(".");
 }
 Serial.println("");
```

```
Serial.println("WiFi connected");
// Connect to Server IoT (CloudMQTT)
```

```
. ( , ); . ( );
```

```
 while (!client.connected()) {
  Serial.println("Connecting to MQTT Broker ...");

   if (client.connect("PID-iTCLab Monitoring Using IoT...")) {
    Serial.println("connected");
    Serial.print("Message received: ");

  } else {
    Serial.print("failed with state ");
Serial.print(client.state());     delay(2000);
  }
  //client.subscribe("heater1");
  //client.subscribe("heater2");
 }
}
void Q1on(){
  ledcWrite(Q1Channel,iwrite_value);
  //Q1 = iwrite_value/255*100;
  //Serial.println(Q1);
}
void Q1off(){
  ledcWrite(Q1Channel,iwrite_min);
  //Q1 = iwrite_min/255*100;
  //Serial.println(Q1);
}
void Q2on(){
  ledcWrite(Q2Channel,iwrite_value);
  //Q2 = iwrite_value/255*100;
  //Serial.println(Q2);
}
void Q2off(){
  ledcWrite(Q2Channel,iwrite_min);
  //Q2 = iwrite_min/255*100;
  //Serial.println(Q2);
}
void ledon(){
  ledcWrite(ledChannel,iwrite_led);
}
void ledoff(){
  ledcWrite(ledChannel,iwrite_min);
}
void cektemp(){
 degC = analogRead(pinT1) * 0.322265625 ;  // use for 3.3v AREF   cel =
degC/10;
 degC1 = analogRead(pinT2) * 0.322265625 ;  // use for 3.3v AREF   cel1 =
degC1/10;

 Serial.print("Temperature T1: ");
  Serial.print(cel);  // print the temperature T1 in Celsius
```

```
.    ("°C");
.    (" ~ "); // separator between Celsius and Fahrenheit
```

```
  Serial.print("Temperature T2: ");
  Serial.print(cel1);  // print the temperature T2 in Celsius
  Serial.println("°C");
}
float pid(float sp, float pv, float pv_last, float& ierr, float dt) {  float Kc
= 10.0; // K / %Heater   float tauI = 50.0; // sec   float tauD = 1.0;  // sec
// PID coefficients   float KP = Kc;   float KI = Kc / tauI;   float KD =
Kc*tauD;
 // upper and lower bounds on heater level
float ophi = 100;  float oplo = 0;  // calculate
the error   float error = sp - pv;
 // calculate the integral error   ierr =
ierr + KI * error * dt;   // calculate the
measurement derivative   float dpv = (pv -
pv_last) / dt;  // calculate the PID output
 float P = KP * error; //proportional contribution
float I = ierr; //integral contribution   float D = -KD *
dpv; //derivative contribution   float op = P + I + D;
 // implement anti-reset windup   if
((op < oplo) || (op > ophi)) {
   I = I - KI * error * dt;
   // clip output
   op = max(oplo, min(ophi, op));
 }
 ierr = I;
 Serial.println("sp="+String(sp) + " pv=" + String(pv) + " dt=" + String(dt) + " op=" +
String(op) + " P=" + String(P) + " I=" + String(I) + " D=" + String(D));  return op;
}

void receivedCallback(char* topic, byte* payload, unsigned int length) {
/* we got '1' -> Q1_on */  if
((char)payload[0] == '1') {
Q1on();
      Serial.println("Q1 On");
 }

/* we got '2' -> Q1_off */  if
((char)payload[0] == '2') {
      Q1off();
      Serial.println("Q1 Off");
 }

/* we got '3' -> Q2_on */  if
((char)payload[0] == '3') {
      Q2on();
      Serial.println("Q2 On");
   }
```

```
/* we got '4' -> Q2_off */
```

```
 if ((char)payload[0] == '4') {
     Q2off();
     Serial.println("Q2 Off");
 }
} void loop() {
new_ts = millis();
 if (new_ts - ts > 1000) {

 char suhu1[4];  char
suhu2[4];  char
SetPoint[4];  char
Nilai_op[4];  char
Nilai_P[4];  char
Nilai_I[4];  char
Nilai_D[4];

 client.loop();

 // put your main code here, to run repeatedly:   cektemp();
 if (cel > upper_temperature_limit){
Q1off();    ledon();
 }  else {
Q1on();
ledoff();
 }
 if (cel1 > upper_temperature_limit){
Q2off();    ledon();
 }  else {
Q2on();
ledoff();
 }
 //delay (100);
  pv = cel;  // Temperature T1   dt
= (new_ts - ts) / 1000.0;   ts =
new_ts;
 op = pid(sp,pv,pv_last,ierr,dt);
ledcWrite(Q1Channel,op);  pv_last = pv;

 dtostrf(cel, 1, 0, suhu1);  client.publish("Suhu1",suhu1);

 dtostrf(sp, 1, 0, SetPoint);
 client.publish("SetPoint",SetPoint);

 dtostrf(op, 1, 0, Nilai_op);
 client.publish("Nilai_op",Nilai_op);
  delay (200);
 dtostrf(cel1, 1, 0, suhu2);
```

```
.        ("Suhu2",       );
```

```
delay (200);
 }
}
```

Kode ini merupakan implementasi dari sistem IoT berbasis ESP32 yang mengontrol suhu menggunakan PID (Proportional-Integral-Derivative) controller. Kode memanfaatkan WiFi untuk koneksi internet, PubSubClient untuk komunikasi MQTT (Message Queuing Telemetry Transport), serta pengukuran suhu melalui pin analog dan pengendalian perangkat keras menggunakan PWM (Pulse Width Modulation). Berikut adalah penjelasan rinci setiap bagian:

1. **Header File dan Konstanta**
   - Library:
     - `WiFi.h` dan `PubSubClient.h` digunakan untuk koneksi WiFi dan komunikasi MQTT.
     - `Arduino.h` adalah library dasar Arduino.
   - Konstanta WiFi dan MQTT:
     - `ssid` dan `password`: Nama dan password jaringan WiFi.
     - `mqttServer` dan `mqttPort`: Server MQTT (HiveMQ) dan port (1883).

2. **Pin Konfigurasi dan PWM**
   - Pin Konfigurasi:
     - Pin `T1` dan `T2`: Sensor suhu analog.
     - Pin `Q1`, `Q2`: Perangkat keluaran yang dikendalikan (misal: pemanas).
     - Pin `LED`: Indikator visual.
   - PWM Konfigurasi:
     - Frekuensi PWM `freq` ditetapkan pada 5000 Hz.
     - Resolusi PWM ditetapkan pada 8 bit.

3. **Variabel Global**
   - Variabel seperti `cel`, `sp`, `pv`, `ierr`, dan `op` digunakan untuk menghitung suhu, setpoint, error, dan output PID.

- `upper_temperature_limit`: Batas suhu atas, digunakan untuk menghidupkan/mematikan pemanas.

4. Fungsi `setup()`

- Menginisialisasi:
  - Serial komunikasi untuk debugging.

- PWM pada pin `Q1`, `Q2`, dan `LED`.

- Koneksi WiFi:
  - Menghubungkan ESP32 ke WiFi dengan SSID dan password.
- Koneksi MQTT:
  - ESP32 dihubungkan ke broker MQTT, dan callback `receivedCallback` ditetapkan untuk menangani pesan masuk.

5. Fungsi Utama Fungsi Kontrol:

- `Q1on(),Q1off(),Q2on(),Q2off()`: Menyalakan atau mematikan pin PWM.
- `ledon(,ledoff()`: Menyalakan atau mematikan LED. **Pembacaan Suhu:**
- `cektemp()`: Membaca suhu dari pin analog (T1 dan T2) dan mencetak nilainya ke serial monitor.

Fungsi PID:

- `pid()`: Menghitung output berdasarkan kontrol PID: Error: Selisih antara setpoint dan suhu saat ini.
  - Proportional: Mengalikan error dengan `Kc`.
  - Integral: Mengakumulasikan error untuk respon lambat.
  - Derivative: Memperkirakan perubahan error untuk respon cepat.

Callback MQTT:

- `receivedCallback()`: Mengolah pesan masuk dari broker MQTT. Pesan dapat berupa:
  - `'1'`: Menyalakan `Q1`. `'2'`: Mematikan `Q1`.
  - `'3'`: Menyalakan `Q2`.
  - `'4'`: Mematikan `Q2`.

6. Fungsi `loop()`

- Pemrosesan Periodik:
  - Memeriksa waktu menggunakan `millis()` untuk iterasi PID setiap 1 detik.
- Pengendalian Suhu:

- Jika suhu melebihi batas, pemanas dimatikan dan LED menyala. Jika suhu normal, pemanas dinyalakan dan LED mati.
- PID Kontrol:
  - Menghitung output PID untuk menyesuaikan keluaran PWM.
- Komunikasi MQTT:
  - Mengirim data suhu (`T1`, `T2`), setpoint, dan nilai PID ke broker MQTT.

Kegunaan

Kode ini cocok untuk aplikasi pengendalian suhu seperti inkubator atau oven pintar. Sistemnya modular, mendukung komunikasi IoT melalui MQTT, dan mengintegrasikan algoritma PID untuk pengendalian suhu yang presisi.

K. ITCLab-10

```
#include <WiFi.h>
#include <PubSubClient.h> #include
<Arduino.h>

const char* ssid = "wifi_name"; // Enter your WiFi name
const char* password =  "wifi_password"; // Enter WiFi password
#define mqttServer "broker.hivemq.com"
#define mqttPort 1883

WiFiServer server(80);
WiFiClient espClient;
PubSubClient client(espClient);

String Topic;
String Payload;

// constants
const int baud = 115200;     // serial baud rate

// 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
 float   ,    ,      ,      ;
float  , ,  ;
float   ,   ,   ,    ,    ,    ,    ,   ;

float   = 30, //set point    = 0,
//current temperature       = 0,  //prior
temperature     = 0,   //integral error    =
```

```
0,    //time between measurements    = 0;
//PID controller output

int       = 0; // autoSet = 1 otomatis sesuai Default
//float Kc = 0;
```

```
//float tauI = 0;
//float tauD = 0;

// Default = autoset = 1 otomatis sesuai Default
float Kc = 10.0; // K / %Heater  float tauI = 50.0; //
sec  float tauD = 1.0; // sec
 unsigned long ts = 0, new_ts = 0; //timestamp
const float upper_temperature_limit = 58;

// global variables
float Q1 = 0;         // value written to Q1 pin float Q2 = 0;
// value written to Q2 pin int iwrite_value = 25;     // integer
value for writing int iwrite_led = 255;      // integer value for
writing int iwrite_min = 0;       // integer value for writing
 void setup() {
 // put your setup code here, to run once:
  ts = millis();
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);
```

```
       (         ,0);

// Connect to WiFi network
    .     ();
    .     ();
    .    ("Connecting to ");
    .     (   );
   .    (   ,        );

while (    .    () !=                ) {
```

```
    (500);       .    (".");
}
    .     ("");
```

```
 Serial.println("WiFi connected");
 // Connect to Server IoT (CloudMQTT)   client.setServer(mqttServer,
mqttPort);  client.setCallback(receivedCallback);

 while (!client.connected()) {
   Serial.println("Connecting to MQTT Broker ...");

   if (client.connect("PID-iTCLab Controlling Using IoT...")) {
     Serial.println("connected");
     Serial.print("Message received: ");

   } else {
     Serial.print("failed with state ");
Serial.print(client.state());      delay(1000);
   }
   client.subscribe("autoSet");    client.subscribe("SetPoint");
client.subscribe("Nilai_Kc");    client.subscribe("Nilai_tauI");
client.subscribe("Nilai_tauD");
 }
}
void Q1on(){
   ledcWrite(Q1Channel,iwrite_value);
   //Q1 = iwrite_value/255*100;
   //Serial.println(Q1);
}
void Q1off(){
   ledcWrite(Q1Channel,iwrite_min);
   //Q1 = iwrite_min/255*100;
   //Serial.println(Q1);
}
void Q2on(){
   ledcWrite(Q2Channel,iwrite_value);
   //Q2 = iwrite_value/255*100;
   //Serial.println(Q2);
}
void Q2off(){
   ledcWrite(Q2Channel,iwrite_min);
```

```
  //Q2 = iwrite_min/255*100;
  //Serial.println(Q2);
} void
     (){
          (           ,          );
} void      (){
        (           ,          ); } void
       (){
     =           (     ) * 0.322265625 ;  // use for 3.3v AREF
```

```
  cel = degC/10;
  degC1 = analogRead(pinT2) * 0.322265625 ;  // use for 3.3v AREF   cel1 =
degC1/10;

}
float pid(float sp, float Kc, float tauI, float tauD, float pv, float pv_last, float& ierr, float dt) {
 // PID coefficients
float KP = Kc;  float KI =
Kc / tauI;  float KD =
Kc*tauD;
 // upper and lower bounds on heater level
float ophi = 100;  float oplo = 0;  // calculate
the error  float error = sp - pv;
 // calculate the integral error   ierr =
ierr + KI * error * dt;   // calculate the
measurement derivative  float dpv = (pv -
pv_last) / dt;  // calculate the PID output
 float P = KP * error; //proportional contribution
float I = ierr; //integral contribution  float D = -KD *
dpv; //derivative contribution  float op = P + I + D;
 // implement anti-reset windup  if
((op < oplo) || (op > ophi)) {
   I = I - KI * error * dt;
   // clip output
   op = max(oplo, min(ophi, op));
  }
  ierr = I;
  Serial.println("sp="+String(sp) + " pv=" + String(pv) + " dt=" + String(dt) + " op=" +
String(op) + " P=" + String(P) + " I=" + String(I) + " D=" + String(D));  return op;
}
void receivedCallback(char* topic, byte* payload, unsigned int length) {  Topic =
topic;  char autoS[60];  int i;
 for (i=0;i<length;i++){    autoS[i] =
payload[i];
  }
  autoS[i] = '\0';
  Payload = String(autoS);
```

```
} void    (){
      =     ();

 if (       -   > 1000) {
 char      [4];
char      [4];  char
       [4];  char
         [4];  char
         [4];  char
          [4];  char
           [4];
```

```
client.loop();

// put your main code here, to run repeatedly:   cektemp();
if (cel > upper_temperature_limit){
Q1off();    ledon();
} else {
Q1on();
ledoff();
}
if (cel1 > upper_temperature_limit){
Q2off();    ledon();
} else {
Q2on();
ledoff();
}    if(Topic=="autoSet"){
autoSet=Payload.toInt();
}  if(Topic=="Nilai_Kc"){
Kc=Payload.toFloat();
}  if(Topic=="Nilai_tauI"){
tauI=Payload.toFloat();
}  if(Topic=="Nilai_tauD"){
tauD=Payload.toFloat()/6;
}  if(Topic=="SetPoint"){
sp=Payload.toFloat();
}
Serial.println("<------------------------->");
Serial.print("autoSet: ");
Serial.println(autoSet);
Serial.print("SetPoint: ");
Serial.println(sp);
Serial.print("Nilai_Kc: ");
Serial.println(Kc);
Serial.print("Nilai_tauI: ");  Serial.println(tauI);
Serial.print("Nilai_tauD: ");
Serial.println(tauD);
```

```
  .      ("<-------------------------->");
  (   , 1, 0,       );
.       ("Suhu1",      );

 (    , 1, 0,       );       .       ("Suhu2",      );
  (   , 1, 0,            );
.       ("Tampil_SP",           );
```

```
 ( , 1, 0, );
. ("Tampil_Kc", );
```

```
 dtostrf(tauI, 1, 0, Tampil_tauI);
 client.publish("Tampil_tauI",Tampil_tauI);

 dtostrf(tauD, 1, 0, Tampil_tauD);
 client.publish("Tampil_tauD",Tampil_tauD);
  if(autoSet==1){
sp = 35;
   Kc = 10.0; // K / %Heater
tauI = 50.0; // sec      tauD = 1.0;
// sec

 }else if(autoSet == 0){
  // bisa diubah2, sesuai yg muncul terakhir, dan setelah diubah2
 }     pv = cel;  // Temperature T1
dt = (new_ts - ts) / 1000.0;      ts =
new_ts;

   op = pid(sp,Kc,tauI,tauD,pv,pv_last,ierr,dt); // PID Process

   ledcWrite(Q1Channel,op);      pv_last =
pv;

   dtostrf(op, 1, 0, Nilai_op);
   client.publish("Nilai_op",Nilai_op);
 }
}
```

Kode di atas merupakan program untuk mengontrol suhu menggunakan PID (Proportional-Integral-Derivative) dengan bantuan ESP32 yang terhubung ke jaringan WiFi dan MQTT (Message Queuing Telemetry Transport) untuk komunikasi IoT.

1. Fungsi Utama

- Koneksi WiFi dan MQTT:
  - Program menghubungkan ESP32 ke jaringan WiFi menggunakan kredensial `ssid` dan `password`.
  - Menggunakan broker MQTT (`broker.hivemq.com`) pada port 1883 untuk komunikasi data.
  - Topik MQTT yang digunakan mencakup:
    - o `autoSet` untuk mengatur mode otomatis. o `SetPoint` untuk menetapkan suhu target.

- Nilai_Kc, Nilai_tauI, dan Nilai_tauD untuk parameter PID.

- Konfigurasi Pin dan PWM:
  - Pin tertentu (Q1, Q2, LED) dikonfigurasi untuk kontrol PWM (Pulse Width Modulation) dengan frekuensi 5000 Hz dan resolusi 8 bit.
  - Pin T1 dan T2 digunakan untuk membaca suhu dari sensor analog.
- Logika Kontrol Suhu:
  - Suhu diukur menggunakan fungsi `cektemp()` yang membaca nilai analog dari sensor suhu.
  - Jika suhu melebihi batas tertentu (`upper_temperature_limit`), pemanas (Q1 dan Q2) dimatikan, dan LED menyala sebagai indikator.
- Kontrol PID:
  - Fungsi `pid()` digunakan untuk menghitung nilai kontrol berdasarkan suhu target (SetPoint) dan suhu saat ini (Process Variable, `pv`).
  - Parameter PID (Kc, tauI, tauD) digunakan untuk menyesuaikan respons kontrol.
  - Nilai keluaran PID (`op`) menentukan intensitas pemanas melalui PWM.
- Komunikasi MQTT:
  - Callback MQTT (`receivedCallback`) memproses pesan masuk untuk memperbarui nilai `autoSet`, `SetPoint`, atau parameter PID.
  - Data suhu dan parameter kontrol dikirimkan kembali ke broker MQTT untuk pemantauan.
- Fungsi AutoSet:
  - Jika `autoSet` bernilai 1, program menggunakan parameter default:
    - SetPoint = 35
    - Kc = 10.0
    - tauI = 50.0
    - tauD = 1.0
  - Jika `autoSet` bernilai 0, parameter dapat diatur secara manual melalui MQTT.

2. Rincian Variabel

- PID Variables:
    - `sp` (SetPoint): Suhu target.
    - `pv` (Process Variable): Suhu saat ini. `Kc`, `tauI`, `tauD`: Parameter kontrol PID.
    - `op`: Output PID untuk mengontrol pemanas.
- Global Variables:
    - `cel`, `cel1`: Suhu yang dibaca dari sensor (dalam Celsius).
    - `ierr`: Kesalahan integral untuk kontrol PID.
    - `ts`, `new_ts`: Penanda waktu untuk menghitung durasi antara pengukuran.

Kegunaan

Program ini merupakan implementasi sistem kontrol suhu berbasis IoT yang fleksibel. Dengan kombinasi PID dan MQTT, pengguna dapat mengatur dan memantau sistem dari jarak jauh.

L. ITCLab-11

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "49922eca",
   "metadata": {},
   "source": [
    "# XOR Gate Programming using Deep Learning"
   ]
  },
  {
   "cell_type": "raw",
   "id": "6e239ded",
   "metadata": {},
   "source": [
    "By: IO-T.NET Team (https://io-t.net/itclab)"
   ]
  },
```

```
{
 "cell_type": "code",
 "execution_count": 1,
 "id": "4aeddbdf",
 "metadata": {},
 "outputs": [],
 "source": [
  "# Library yg dibutuhkan\n",
  "import numpy as np"
 ]
},
{
 "cell_type": "code",
 "execution_count": 2,
 "id": "312606b1",
 "metadata": {},
 "outputs": [],
 "source": [
  "# Pasangan data latih\n",
  "\n",
  "XOR_X = np.array([\n",
  "    [0, 0],\n",
  "    [0, 1],\n",
  "    [1, 0],\n",
  "    [1, 1]\n",
  "])\n",
```

[redacted]

[redacted]

```
    "### Arsitektur Deep Learning\n",
    "Arsitektur Deep Learning dengan Dua Masukan dan Satu Keluaran"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "cf5a0863",
   "metadata": {},
   "source": [
    "![arsitektur_DL](Gerbang_XOR.jpg)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "2b875363",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "WARNING:tensorflow:From
C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\site-
packages\\keras\\src\\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.\n",
      "\n",
      "WARNING:tensorflow:From
C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\site-
packages\\keras\\src\\backend.py:873: The name tf.get_default_graph is deprecated. Please use
tf.compat.v1.get_default_graph instead.\n",
      "\n"
     ]
    }
   ],
   "source": [
    "# Impor `Sequential` dari` keras.models`\n",
    "from keras.models import Sequential\n",     "\n",
    "# Impor `Dense` dari` keras.layers`\n",
    "from keras.layers import Dense\n",
    "\n",
```

```
"# Inisialisasi konstruktor\n",
"model = Sequential()\n",
"\n",
"# Tambahkan lapisan masukan \n",
"model.add(Dense(2, activation='sigmoid', input_shape=(2,)))\n",
```

[redacted]

[redacted]

```
awal."    ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "id": "c72c87b8",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Model: \"sequential\"\n",
      "_________________________________________________________________\n",
      " Layer (type)                Output Shape              Param #   \n",
      "=================================================================\n",
      " dense (Dense)               (None, 2)                 6         \n",
      "                                                                 \n",
      " dense_1 (Dense)             (None, 2)                 6         \n",
      "                                                                 \n",
      " dense_2 (Dense)             (None, 1)                 3         \n",
      "                                                                 \n",
      "=================================================================\n",
      "Total params: 15 (60.00 Byte)\n",
      "Trainable params: 15 (60.00 Byte)\n",
      "Non-trainable params: 0 (0.00 Byte)\n",
      "_________________________________________________________________\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "[array([[ 0.88105714,  1.120653  ],\n",
       "        [-1.0473598 ,  0.37149537]], dtype=float32),\n",
       " array([0., 0.], dtype=float32),\n",
       " array([[ 0.55942726, -0.02052712],\n",
       "        [-1.1376439 ,  0.07940733]], dtype=float32),\n",
       " array([0., 0.], dtype=float32),\n",
       " array([[ 0.2318461 ],\n",
       "        [-0.25704885]], dtype=float32),\n",
```

```
    " array([0.], dtype=float32)]"
   ]
  },
  "execution_count": 4,
  "metadata": {},
  "output_type": "execute_result"
```

```
   ]
  },
  {
   "cell_type": "markdown",
   "id": "57d2a1b5",
   "metadata": {},
   "source": [
    "Untuk pelatihan Deep Learning silahkan ketikkan skrip berikut."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "id": "ed4683b0",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "WARNING:tensorflow:From
C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\sitepackages\\keras\\src\\optimizers\\__init__.p
y:309: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.\n",
      "\n",
      "Epoch 1/1000\n",
      "WARNING:tensorflow:From
```

```
C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\sitepackages\\keras\\src\\utils\\tf_utils.py:492:
The name tf.ragged.RaggedTensorValue is deprecated.
Please use tf.compat.v1.ragged.RaggedTensorValue instead.\n",
    "\n",
    "WARNING:tensorflow:From
C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\sitepackages\\keras\\src\\engine\\base_layer_uti
ls.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.\n",
    "\n",
    "4/4 [==============================] - 1s 5ms/step - loss: 0.6940 - accuracy: 0.5000\n",      "Epoch 2/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6940 - accuracy: 0.5000\n",      "Epoch 3/1000\n",
    "4/4 [==============================] - 0s 5ms/step - loss: 0.6939 - accuracy: 0.5000\n",      "Epoch 4/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 5/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6938 - accuracy: 0.5000\n",
    "Epoch 6/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",
    "Epoch 7/1000\n",
                                                            0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",
```

```
"4/4 [==============================] - - loss: 0.6936 - accuracy: 0.5000\n",
"4/4 [==============================] -
```

```
"Epoch 11/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 12/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6937 - accuracy: 0.5000\n",
"Epoch 13/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6937 - accuracy: 0.5000\n",
"Epoch 14/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6937 - accuracy: 0.5000\n",
"Epoch 15/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 16/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 17/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6938 - accuracy: 0.5000\n",      "Epoch 18/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 19/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 20/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6937 - accuracy: 0.5000\n",
"Epoch 21/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 22/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 23/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 24/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 25/1000\n",
"4/4 [==============================] - 0s 7ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 26/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 27/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 28/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 29/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6938 - accuracy: 0.5000\n",      "Epoch 30/1000\n",
"4/4 [==============================] - 0s 924us/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 31/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 32/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 33/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 34/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 35/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 36/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 37/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 38/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n", "Epoch 39/1000\n",
```

```
"Epoch 41/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6938 - accuracy: 0.5000\n",
"Epoch 42/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6936 - accuracy: 0.5000\n",      "Epoch 43/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6937 - accuracy: 0.5000\n",      "Epoch 44/1000\n",
```

```
"4/4 [==============================] -
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"4/4 [==============================] -
```

```
"4/4 [==============================] - 0s 6ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 45/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 46/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",
"Epoch 47/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 48/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 49/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 50/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 51/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 52/1000\n",
"4/4 [==============================] - 0s 345us/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 53/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 54/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 55/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 56/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 57/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6936 - accuracy: 0.5000 \n",       "Epoch 58/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 59/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 60/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 61/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 62/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",       "Epoch 63/1000\n",
"4/4 [==============================] - 0s 7ms/step - loss: 0.6937 - accuracy: 0.5000\n",       "Epoch 64/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 65/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 66/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 67/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 68/1000\n",
                                        0s 0s/step - loss: 0.6936 - accuracy: 0.5000\n",
```

```
"Epoch 72/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 73/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 74/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 75/1000\n",
"4/4 [==============================] - 0s 758us/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 76/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 77/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 78/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 79/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 80/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 81/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6937 - accuracy: 0.5000\n",
"Epoch 82/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 83/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 84/1000\n",
```

```
"4/4 [==============================] -
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"4/4 [==============================] -
```

```
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",       "Epoch 85/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 86/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 87/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",
"Epoch 88/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 89/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 90/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 91/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 92/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",       "Epoch 93/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 94/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 95/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",
"Epoch 96/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 97/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 98/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",       "Epoch 99/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n", "Epoch 100/1000
```

```
"4/4 [==============================] -
```

```
                              \n",
"4/4 [==============================] -                                                          \n",
```

```
"Epoch 102/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 103/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 104/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 105/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 106/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 107/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 108/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 109/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 110/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",      "Epoch 111/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 112/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 113/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",
"Epoch 114/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 115/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 116/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 117/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 118/1000\n",
"4/4 [==============================] - 0s 8ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 119/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 120/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
"Epoch 121/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 122/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 123/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 124/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 125/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 126/1000\n",
"4/4 [==============================] - 0s 993us/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 127/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 128/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 129/1000\n",
                                                  0s 0s/step - loss: 0.6934 - accuracy: 0.5000\n",
```

```
"Epoch 133/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 134/1000\n",
```

```
"4/4 [==============================] -                    -                    - accuracy: 0.5000\n",
                              \n",
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
                               \n",
"4/4 [==============================] -
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 135/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6935 - accuracy: 0.5000\n",
"Epoch 136/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 137/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 138/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 139/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 140/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 141/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 142/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 143/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 144/1000\n",
"4/4 [==============================] - 0s 822us/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 145/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 146/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 147/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 148/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 149/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 150/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 151/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6936 - accuracy: 0.5000\n",     "Epoch 152/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 153/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 154/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 155/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000 \n",     "Epoch 156/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 157/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000 \n",     "Epoch 158/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 159/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",     "Epoch 160/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6935 - accuracy: 0.5000\n", "Epoch 161/1000
```

```
"Epoch 166/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 167/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6934 - accuracy: 0.5000\n",     "Epoch 168/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 169/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5000\n",
                                    \n",
    "4/4 [==============================] -
```

```
    "Epoch 170/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",
    "Epoch 171/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 172/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 173/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 174/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 175/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 176/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6934 - accuracy: 0.5000\n",
    "Epoch 177/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 178/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 179/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 180/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000 \n",      "Epoch 181/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.5000\n",
    "Epoch 182/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6934 - accuracy: 0.5000\n",
    "Epoch 183/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6934 - accuracy: 0.5000\n",
    "Epoch 184/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 185/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 186/1000\n",
    "4/4 [==============================] - 0s 8ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 187/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 188/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000 \n",
    "Epoch 189/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6934 - accuracy: 0.5000\n",
    "Epoch 190/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 191/1000\n",
0s 0s/step - loss: 0.6933 - accuracy: 0.5000\n",
```

```
    "4/4 [==============================] -
```

```
"4/4 [==============================] - 0s 3ms/step -  - accuracy: 0.5000\n",
 \n",
"4/4 [==============================] -  \n",
```

```
 \n",
"4/4 [==============================] -  -  -  \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                                   - accuracy: 0.5000\n",
                          \n",
"4/4 [==============================] -                                                               \n",
```

```
"Epoch 194/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 195/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 196/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 197/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 198/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 199/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 200/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6935 - accuracy: 0.5000\n",      "Epoch 201/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 202/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 203/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 204/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 205/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 206/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 207/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6933 - accuracy: 0.5000 \n",      "Epoch 208/1000\n",
"4/4 [==============================] - 0s 336us/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 209/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 210/1000\n",
"4/4 [==============================] - 0s 8ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 211/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 212/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 213/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 214/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 215/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 216/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 217/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 218/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 219/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000 \n",      "Epoch 220/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 221/1000\n",
                                                0s 3ms/step - loss: 0.6934 - accuracy: 0.5000
```

```
                          \n",
"4/4 [==============================] -                  -              -                  \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                     - accuracy: 0.5000\n",
                  \n",
"4/4 [==============================] -                                             \n",
```

```
                  \n",
"4/4 [==============================] -               -             -              \n",
```

```
"4/4 [==============================] - os 3ms/step -                                   - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                                                                \n",
```

```
"Epoch 227/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 228/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 229/1000\n",
"4/4 [==============================] - os 5ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 230/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 231/1000\n",
"4/4 [==============================] - os os/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 232/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 233/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6935 - accuracy: 0.5000\n",    "Epoch 234/1000\n",
"4/4 [==============================] - os 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 235/1000\n",
"4/4 [==============================] - os 890us/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 236/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 237/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 238/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 239/1000\n",
"4/4 [==============================] - os 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 240/1000\n",
"4/4 [==============================] - os 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 241/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 242/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 243/1000\n",
"4/4 [==============================] - os os/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 244/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 245/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 246/1000\n",
"4/4 [==============================] - os 5ms/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 247/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",    "Epoch 248/1000\n",
"4/4 [==============================] - os 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 249/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 250/1000\n",
"4/4 [==============================] - os 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",    "Epoch 251/1000\n",
                                                  os os/step - loss: 0.6934 - accuracy: 0.5000
```

```
                              \n",
"4/4 [==============================] -                    -                -                 \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                    - accuracy: 0.5000\n",
                  \n",
"4/4 [==============================] -                                                       \n",
```

```
                  \n",
"4/4 [==============================] -              -            -            \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                                   - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                                                                    \n",
```

```
"Epoch 255/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 256/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 257/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 258/1000\n",
"4/4 [==============================] - 0s 8ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 259/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 260/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 261/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6933 - accuracy: 0.5000 \n",      "Epoch 262/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 263/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 264/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000 \n",
"Epoch 265/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 266/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 267/1000\n",
"4/4 [==============================] - 0s 381us/step - loss: 0.6932 - accuracy: 0.5000\n",      "Epoch 268/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 269/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 270/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 271/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.5000\n",
"Epoch 272/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 273/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 274/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 275/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.5000\n",      "Epoch 276/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",
"Epoch 277/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 278/1000\n",
"4/4 [==============================] - 0s 504us/step - loss: 0.6932 - accuracy: 0.5000\n",      "Epoch 279/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 280/1000\n",
"4/4 [==============================] - 0s 633us/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 281/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5000\n",      "Epoch 282/1000\n",
                                                   0s 0s/step - loss: 0.6933 - accuracy: 0.5000
```

```
                              \n",
"4/4 [==============================] -                     -                -                  \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                          - accuracy: 0.5000\n",
                   \n",
"4/4 [==============================] -                                                  \n",
```

```
                   \n",
"4/4 [==============================] -                   -              -                 \n",
```

```
                                \n",
    "4/4 [==============================] -                                  - loss: 0.6932 - accuracy: 0.2500\n",
                                \n",
    "4/4 [==============================] -                  -               -                   \n",
                                \n",
    "4/4 [==============================] -                  -               -                   \n",
                                \n",
    "4/4 [==============================] -                                                      \n",
```

```
    "Epoch 288/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",      "Epoch 289/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 290/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6934 - accuracy: 0.5000\n",      "Epoch 291/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000\n",
    "Epoch 292/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.2500\n",
    "Epoch 293/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 294/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500 \n",      "Epoch 295/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 296/1000\n",
    "4/4 [==============================] - 0s 357us/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 297/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
    "Epoch 298/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
    "Epoch 299/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500 \n",      "Epoch 300/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 301/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 302/1000\n",
    "4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 303/1000\n",
    "4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n",
    "Epoch 304/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.2500\n",
    "Epoch 305/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
    "Epoch 306/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",      "Epoch 307/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.2500\n",      "Epoch 308/1000\n",
    "4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 309/1000\n",
```

```
    "4/4 [==============================] -                  -                  - accuracy: 0.2500\n",
                                \n",
    "4/4 [==============================] -                  -                  - accuracy: 0.2500\n",
                                \n",
    "4/4 [==============================] -                  -                  - accuracy: 0.2500\n",
                                \n",
    "4/4 [==============================] -                  -                  - accuracy: 0.2500\n",
                                \n",
    "4/4 [==============================] -                  -                  - accuracy: 0.2500\n",
                                \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                    - accuracy: 0.2500\n",
              \n",
"4/4 [==============================] -               -             - accuracy: 0.2500\n",
              \n",
"4/4 [==============================] -                                          \n",
```

```
              \n",
"4/4 [==============================] -               -           -             \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                    - accuracy: 0.2500\n",
                \n",
"4/4 [==============================] -              -             - accuracy: 0.2500\n",
                \n",
"4/4 [==============================] -                                        \n",
```

```
"Epoch 317/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 318/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 319/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 320/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 321/1000\n",
"4/4 [==============================] - 0s 836us/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 322/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 323/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 324/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 325/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 326/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 327/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 328/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 329/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 330/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 331/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 332/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 333/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 334/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 335/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 336/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 337/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500 \n",
"Epoch 338/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 339/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 340/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 341/1000\n",
"4/4 [==============================] - 0s 828us/step - loss: 0.6933 - accuracy: 0.2500\n",     "Epoch 342/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 343/1000\n",
                                                0s 2ms/step - loss: 0.6933 - accuracy: 0.2500
```

```
                \n",
"4/4 [==============================] -              -           -               \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                              - accuracy: 0.2500\n",
                         \n",
"4/4 [==============================] -                      -                     - accuracy: 0.2500\n",
                         \n",
"4/4 [==============================] -                                                       \n",
```

```
"Epoch 350/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 351/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500 \n",
"Epoch 352/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 353/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 354/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 355/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 356/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6933 - accuracy: 0.2500\n",    "Epoch 357/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 358/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",    "Epoch 359/1000\n",
"4/4 [==============================] - 0s 593us/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 360/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 361/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6933 - accuracy: 0.2500\n",    "Epoch 362/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 363/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 364/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500 \n",
"Epoch 365/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 366/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 367/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6933 - accuracy: 0.2500\n",    "Epoch 368/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",    "Epoch 369/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500 \n",    "Epoch 370/1000\n",
```

```
                         \n",
"4/4 [==============================] -                      -            -                 \n",
```

```
"4/4 [==============================] - 0s 3ms/step -                    - accuracy: 0.2500\n",
              \n",
"4/4 [==============================] -               -                 - accuracy: 0.2500\n",
              \n",
"4/4 [==============================] -                                                    \n",
```

```
              \n",
"4/4 [==============================] -               -            -                       \n",
```

```
"4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
                   \n",
"4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
                   \n",
"4/4 [==============================] -                                                     \n",
```

```
                   \n",
"4/4 [==============================] -                    -               -                \n",
```

```
"4/4 [==============================] -                                -                              - accuracy: 0.2500\n",
                                \n",
"4/4 [==============================] -                                -                              - accuracy: 0.2500\n",
                                \n",
"4/4 [==============================] -                                                                                  \n",
```

```
"Epoch 378/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 379/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 380/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 381/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 382/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",       "Epoch 383/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 384/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 385/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 386/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 387/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500 \n",       "Epoch 388/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 389/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 390/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 391/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 392/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 393/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 394/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 395/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 396/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",       "Epoch 397/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 398/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 399/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 400/1000\n",
"4/4 [==============================] - 0s 863us/step - loss: 0.6933 - accuracy: 0.2500\n",       "Epoch 401/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 402/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 403/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 404/1000\n",
                                                  0s 3ms/step - loss: 0.6932 - accuracy: 0.2500
```

```
"4/4 [==============================] -
```

```
"4/4 [==============================] -                    -                    - accuracy: 0.2500\n",
                    \n",
"4/4 [==============================] -                    -                    - accuracy: 0.2500\n",
                    \n",
"4/4 [==============================] -                                                            \n",
```

```
"Epoch 411/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 412/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 413/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 414/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 415/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 416/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500 \n",       "Epoch 417/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 418/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 419/1000\n",
"4/4 [==============================] - 0s 503us/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 420/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 421/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 422/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 423/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 424/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",       "Epoch 425/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 426/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 427/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 428/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 429/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 430/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",       "Epoch 431/1000\n",
```

```
"4/4 [==============================] -
```

```
"4/4 [==============================] -                    -                    - accuracy: 0.2500\n",
                    \n",
"4/4 [==============================] -                    -                    - accuracy: 0.2500\n",
                    \n",
"4/4 [==============================] -                                          \n",
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 440/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 441/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 442/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 443/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 444/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 445/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 446/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 447/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 448/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 449/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 450/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 451/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 452/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 453/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 454/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 455/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 456/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 457/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 458/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",     "Epoch 459/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.2500\n",
"Epoch 460/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 461/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 462/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500 \n",     "Epoch 463/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 464/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",     "Epoch 465/1000\n",
```

```
"4/4 [==============================] -
```

```
    "4/4 [==============================] -                          -                        - accuracy: 0.2500\n",
                         \n",
    "4/4 [==============================] -                          -                        - accuracy: 0.2500\n",
                         \n",
    "4/4 [==============================] -                                                                      \n",
```

```
    "4/4 [==============================] - 0s 527us/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 466/1000
```

```
    "4/4 [==============================] -
```

```
                    \n",
 "4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
                    \n",
 "4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
                    \n",
 "4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
                    \n",
 "4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
                    \n",
 "4/4 [==============================] -                    -                  - accuracy: 0.2500\n",
```

```
 "Epoch 472/1000\n",
 "4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 473/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
 "Epoch 474/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
 "Epoch 475/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
 "Epoch 476/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 477/1000\n",
 "4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",   "Epoch 478/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 479/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
 "Epoch 480/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
 "Epoch 481/1000\n",
 "4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",
 "Epoch 482/1000\n",
 "4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 483/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 484/1000\n",
 "4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500 \n",   "Epoch 485/1000\n",
 "4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 486/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
 "Epoch 487/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
 "Epoch 488/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
 "Epoch 489/1000\n",
 "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 490/1000\n",
 "4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.2500\n",   "Epoch 491/1000\n",
```

```
                    \n",
 "4/4 [==============================] -                         - loss: 0.6931 - accuracy: 0.2500\n",
                    \n",
```

```
"4/4 [==============================] -
\n",
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 492/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 493/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 494/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n", "Epoch 495/1000\n",
0s 743us/step - loss: 0.6931 - accuracy: 0.2500\n",
```

```
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 499/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 500/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 501/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 502/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 503/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 504/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 505/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 506/1000\n",
"4/4 [==============================] - 0s 929us/step - loss: 0.6932 - accuracy: 0.2500\n", "Epoch 507/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 508/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 509/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 510/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 511/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6932 - accuracy: 0.2500\n", "Epoch 512/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 513/1000\n",
"4/4 [==============================] - 0s 503us/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 514/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 515/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 516/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 517/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 518/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 519/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 520/1000\n",
"4/4 [==============================] - 0s 693us/step - loss: 0.6931 - accuracy: 0.2500\n", "Epoch 521/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 522/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n", "Epoch 523/1000\n",
0s 965us/step - loss: 0.6931 - accuracy: 0.2500\n",
```

```
\n",
"4/4 [==============================] -    -    -    \n",
\n",
"4/4 [==============================] -    -    -    \n",
\n",
"4/4 [==============================] -    -    -    \n",
\n",
"4/4 [==============================] -    -    -    \n",
```

```
"4/4 [==============================] -
                        \n",
```

```
"Epoch 533/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 534/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6931 - accuracy: 0.5000 \n",
"Epoch 535/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 536/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.5000\n",
"Epoch 537/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6932 - accuracy: 0.2500\n",      "Epoch 538/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 539/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 540/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 541/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 542/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 543/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 544/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 545/1000\n",
"4/4 [==============================] - 0s 472us/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 546/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 547/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.2500\n",
"Epoch 548/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",
```

```
                        \n",
"4/4 [==============================] -                -            -               \n",
                        \n",
"4/4 [==============================] -                -            -               \n",
                        \n",
"4/4 [==============================] -                -            -               \n",
                        \n",
"4/4 [==============================] -                -            -               \n",
```

```
"4/4 [==============================] -
\n",
```

```
"Epoch 549/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000\n",
"Epoch 550/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",     "Epoch 551/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",     "Epoch 552/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",     "Epoch 553/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000\n",     "Epoch 554/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 555/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 556/1000\n",
0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
```

```
\n",
"4/4 [==============================] -    -    -    \n",
\n",
"4/4 [==============================] -    -    -    \n",
\n",
"4/4 [==============================] -    -    -    \n",
\n",
"4/4 [==============================] -    -    -    \n",
```

```
"4/4 [==============================] -                                   \n",
                  \n",
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5000\n",      "Epoch 560/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.2500\n",      "Epoch 561/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 562/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 563/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000 \n",
"Epoch 564/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 565/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 566/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 567/1000\n",
"4/4 [==============================] - 0s 8ms/step - loss: 0.6931 - accuracy: 0.2500\n",      "Epoch 568/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 569/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 570/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 571/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.2500\n",      "Epoch 572/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6930 - accuracy: 0.2500\n",      "Epoch 573/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.2500\n",      "Epoch 574/1000\n",
"4/4 [==============================] - 0s 792us/step - loss: 0.6930 - accuracy: 0.2500\n",      "Epoch 575/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 576/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 577/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 578/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 579/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 580/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 581/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 582/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 583/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.2500\n",
"Epoch 584/1000\n",
0s 2ms/step - loss: 0.6932 - accuracy: 0.2500\n",
```

```
                  \n",
"4/4 [==============================] -             -           -             \n",
                  \n",
"4/4 [==============================] -             -           -             \n",
                  \n",
"4/4 [==============================] -             -           -             \n",
                  \n",
"4/4 [==============================] -             -           - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                   \n",
                              \n",
```

```
"Epoch 591/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 592/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 593/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 594/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 595/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 596/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 597/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 598/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 599/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 600/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 601/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 602/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 603/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 604/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 605/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 606/1000\n",
"4/4 [==============================] - 0s 604us/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 607/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 608/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 609/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 610/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 611/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 612/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 613/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000 \n",      "Epoch 614/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 615/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 616/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000\n",
```

```
                              \n",
"4/4 [==============================] -                    -                -                  \n",
                              \n",
"4/4 [==============================] -                    -                -                  \n",
                              \n",
"4/4 [==============================] -                    -                -                  \n",
                              \n",
"4/4 [==============================] - 0s 3ms/step -                          - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                            \n",
                                \n",

"Epoch 617/1000\n",
                                                          0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",

"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 621/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000 \n",    "Epoch 622/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 623/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 624/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 625/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 626/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 627/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 628/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 629/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 630/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 631/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 632/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 633/1000\n",
"4/4 [==============================] - 0s 7ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 634/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 635/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 636/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 637/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 638/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 639/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 640/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",    "Epoch 641/1000\n",
"4/4 [==============================] - 0s 840us/step - loss: 0.6931 - accuracy: 0.5000\n",    "Epoch 642/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 643/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 644/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 645/1000\n",
                                                          0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",

                                \n",
"4/4 [==============================] -                    -                -                  \n",
                                \n",
"4/4 [==============================] -                    -                -                  \n",
                                \n",
"4/4 [==============================] -                    -                -                  \n",
                                \n",
"4/4 [==============================] - 0s 3ms/step -                          - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                        \n",
                    \n",
```

```
                    \n",
"4/4 [==============================] -              -          -              \n",
                    \n",
"4/4 [==============================] -              -          -              \n",
                    \n",
"4/4 [==============================] -              -          -              \n",
                    \n",
"4/4 [==============================] - 0s 3ms/step -            - accuracy: 0.5000\n",
```

```
                                \n",
    "4/4 [==============================] -                         -                        - accuracy: 0.5000\n",
                                \n",
    "4/4 [==============================] -
```

```
    "Epoch 652/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 653/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 654/1000\n",
    "4/4 [==============================] - 0s 6ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 655/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 656/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6930 - accuracy: 0.5000\n",
    "Epoch 657/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
    "Epoch 658/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
    "Epoch 659/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 660/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 661/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000 \n",      "Epoch 662/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
    "Epoch 663/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
    "Epoch 664/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000 \n",
    "Epoch 665/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 666/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 667/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",      "Epoch 668/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 669/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
    "Epoch 670/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5000\n",
    "Epoch 671/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6930 - accuracy: 0.5000\n",
    "Epoch 672/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 673/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 674/1000\n",
    "4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 675/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 676/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",
    "Epoch 677/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
    "Epoch 678/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 679/1000\n",
                                                       0s 245us/step - loss: 0.6930 - accuracy: 0.5000\n",
```

```
                                \n",
```

```
"4/4 [==============================] -                                                    \n",
                              \n",
```

```
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 682/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 683/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 684/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 685/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 686/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 687/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 688/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 689/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 690/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 691/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 692/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 693/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 694/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 695/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 696/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 697/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 698/1000\n",
"4/4 [==============================] - 0s 521us/step - loss: 0.6931 - accuracy: 0.5000\n",    "Epoch 699/1000\n",
"4/4 [==============================] - 0s 6ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 700/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 701/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 702/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 703/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5000\n",
"Epoch 704/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 705/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 706/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 707/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 708/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000 \n",    "Epoch 709/1000\n",
                                                  0s 0s/step - loss: 0.6929 - accuracy: 0.5000
```

```
                              \n",
"4/4 [==============================] - 0s 3ms/step -                          - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                  \n",
                    \n",
```

```
"Epoch 713/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 714/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 715/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 716/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 717/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 718/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 719/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 720/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 721/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 722/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 723/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 724/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 725/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 726/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 727/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 728/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 729/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 730/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 731/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 732/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 733/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 734/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 735/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 736/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 737/1000\n",
0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",
```

```
                    \n",
"4/4 [==============================] - 0s 3ms/step -                          - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                    \n",
                              \n",
```

```
                              \n",
"4/4 [==============================] -                        -                    - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                        -                    - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] - 0s 3ms/step -                               - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                        \n",
                        \n",
```

```
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 743/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 744/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 745/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 746/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 747/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 748/1000\n",
"4/4 [==============================] - 0s 128us/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 749/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 750/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 751/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 752/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 753/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 754/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 755/1000\n",
"4/4 [==============================] - 0s 521us/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 756/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000 \n",      "Epoch 757/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 758/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 759/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 760/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 761/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 762/1000\n",
"4/4 [==============================] - 0s 7ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 763/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 764/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 765/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 766/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 767/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 768/1000\n",
                                                       0s 0s/step - loss: 0.6929 - accuracy: 0.5000
```

```
                        \n",
"4/4 [==============================] -                  -              -                 \n",
                        \n",
"4/4 [==============================] -                  -                 - accuracy: 0.5000\n",
                        \n",
"4/4 [==============================] -                  -                 - accuracy: 0.5000\n",
                        \n",
"4/4 [==============================] -                  -                 - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                    \n",
                              \n",
```

```
                              \n",
"4/4 [==============================] -                    -                  -                  \n",
                              \n",
"4/4 [==============================] -                    -                  - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                    -                  - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                    -                  - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                    \n",
                         \n",
```

```
"Epoch 773/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 774/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 775/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 776/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 777/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 778/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 779/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 780/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 781/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 782/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 783/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 784/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",      "Epoch 785/1000\n",
"4/4 [==============================] - 0s 894us/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 786/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000 \n",      "Epoch 787/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 788/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 789/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 790/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 791/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 792/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 793/1000\n",
"4/4 [==============================] - 0s 223us/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 794/1000\n",
"4/4 [==============================] - 0s 521us/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 795/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 796/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 797/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 798/1000\n",
                                            0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
```

```
                         \n",
"4/4 [==============================] -                  -              -                  \n",
                         \n",
"4/4 [==============================] -                  -                - accuracy: 0.5000\n",
                         \n",
"4/4 [==============================] -                  -                - accuracy: 0.5000\n",
                         \n",
"4/4 [==============================] -                  -                - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                  \n",
                    \n",
```

```
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 804/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 805/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 806/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 807/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 808/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 809/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",    "Epoch 810/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 811/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 812/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6930 - accuracy: 0.5000\n",
"Epoch 813/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 814/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 815/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 816/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 817/1000\n",
"4/4 [==============================] - 0s 263us/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 818/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 819/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 820/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 821/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",    "Epoch 822/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 823/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 824/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 825/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 826/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",
"Epoch 827/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 828/1000\n",
0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
```

```
                    \n",
"4/4 [==============================] -               -           -               \n",
                    \n",
"4/4 [==============================] -               -             - accuracy: 0.5000\n",
                    \n",
"4/4 [==============================] -               -             - accuracy: 0.5000\n",
                    \n",
"4/4 [==============================] -               -             - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                                                              \n",
                             \n",
```

```
                             \n",
"4/4 [==============================] -                     -                     -                   \n",
                             \n",
"4/4 [==============================] -                     -                     - accuracy: 0.5000\n",
                             \n",
"4/4 [==============================] -                     -                     - accuracy: 0.5000\n",
                             \n",
"4/4 [==============================] -                     -                     - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -
                              \n",
```

```
"Epoch 838/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 839/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 840/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",      "Epoch 841/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 842/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 843/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 844/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 845/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 846/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 847/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 848/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 849/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 850/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 851/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 852/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 853/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",
"Epoch 854/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",
"Epoch 855/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 856/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 857/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 858/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 859/1000\n",
                                                 0s 0s/step - loss: 0.6929 - accuracy: 0.5000\n",
```

```
                              \n",
"4/4 [==============================] -                    -               -                  \n",
                              \n",
"4/4 [==============================] -                    -                 - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                    -                 - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                    -                 - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -
                                      \n",
```

```
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000 \n",      "Epoch 865/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 866/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 867/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 868/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 869/1000\n",
"4/4 [==============================] - 0s 418us/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 870/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 871/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 872/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 873/1000\n",
"4/4 [==============================] - 0s 709us/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 874/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 875/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 876/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 877/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 878/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 879/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 880/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.7500\n",
"Epoch 881/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.7500\n",
"Epoch 882/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 883/1000\n",
"4/4 [==============================] - 0s 847us/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 884/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6929 - accuracy: 0.5000\n",      "Epoch 885/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.7500\n",      "Epoch 886/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.7500\n",
"Epoch 887/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 888/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 889/1000\n",
0s 0s/step - loss: 0.6928 - accuracy: 0.5000\n",
```

```
                                      \n",
"4/4 [==============================] -                    -                -                  \n",
                                      \n",
"4/4 [==============================] -                    -                - accuracy: 0.5000\n",
                                      \n",
"4/4 [==============================] -                    -                - accuracy: 0.5000\n",
                                      \n",
"4/4 [==============================] -                    -                - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -
                              \n",
```

```
"Epoch 899/1000\n",
"4/4 [==============================] - 0s 610us/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 900/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 901/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 902/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 903/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6927 - accuracy: 0.5000\n",      "Epoch 904/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 905/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 906/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 907/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 908/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 909/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 910/1000\n",
"4/4 [==============================] - 0s 345us/step - loss: 0.6928 - accuracy: 0.5000\n",      "Epoch 911/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",      "Epoch 912/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",      "Epoch 913/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 914/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.7500\n",
"Epoch 915/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",
"Epoch 916/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5000\n",      "Epoch 917/1000\n",
```

```
                              \n",
"4/4 [==============================] -                    -               -                  \n",
                              \n",
"4/4 [==============================] -                    -                 - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                    -                 - accuracy: 0.5000\n",
                              \n",
"4/4 [==============================] -                    -                 - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -
\n",
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",       "Epoch 918/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",       "Epoch 919/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6928 - accuracy: 0.5000\n",       "Epoch 920/1000\n",
0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
```

```
\n",
"4/4 [==============================] -          -          -          \n",
\n",
"4/4 [==============================] -          -          - accuracy: 0.5000\n",
\n",
"4/4 [==============================] -          -          - accuracy: 0.5000\n",
\n",
"4/4 [==============================] -          -          - accuracy: 0.5000\n",
```

```
"4/4 [==============================] -                    -                -
                    \n",
```

```
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",    "Epoch 926/1000\n",
"4/4 [==============================] - 0s 7ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 927/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 928/1000\n",
"4/4 [==============================] - 0s 669us/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 929/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 930/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 931/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6929 - accuracy: 0.5000 \n",    "Epoch 932/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 933/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 934/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",    "Epoch 935/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 936/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6927 - accuracy: 0.5000 \n",    "Epoch 937/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 938/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6928 - accuracy: 0.5000 \n",    "Epoch 939/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 940/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.7500\n",
"Epoch 941/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.7500\n",
"Epoch 942/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5000\n",
"Epoch 943/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 944/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 945/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 946/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 947/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 948/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 949/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.7500\n",
"Epoch 950/1000\n",
0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",
```

```
                    \n",
"4/4 [==============================] -                    -                -                \n",
                    \n",
"4/4 [==============================] -                    -                -                \n",
                    \n",
"4/4 [==============================] -                    -                -                \n",
                    \n",
"4/4 [==============================] -                    -                -                \n",
```

```
"4/4 [==============================] -                    -                    -
                              \n",
```

```
"Epoch 960/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 961/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",
"Epoch 962/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 963/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 964/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 965/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 966/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 967/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 968/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 969/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 970/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 971/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 972/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 973/1000\n",
"4/4 [==============================] - 0s 4ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 974/1000\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 975/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 976/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 977/1000\n",
"4/4 [==============================] - 0s 836us/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 978/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 979/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.7500\n",    "Epoch 980/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5000\n",    "Epoch 981/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 982/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 983/1000\n",
                                  0s 337us/step - loss: 0.6927 - accuracy: 0.5000\n",
```

```
"4/4 [==============================] - 0s 858us/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 987/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",    "Epoch 988/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 989/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6926 - accuracy: 0.5000\n",
"Epoch 990/1000\n",
```

"4/4 [==============================] - - -
\n",

"4/4 [==============================] - 0s 1ms/step - loss: 0.6928 - accuracy: 0.5000\n",
"Epoch 991/1000\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.6927 - accuracy: 0.7500\n", "Epoch 992/1000\n",
"4/4 [==============================] - 0s 1ms/step - loss: 0.6927 - accuracy: 0.5000\n", "Epoch 993/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n", "Epoch 994/1000\n",
"4/4 [==============================] - 0s 777us/step - loss: 0.6927 - accuracy: 0.5000\n", "Epoch 995/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n",
"Epoch 996/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n", "Epoch 997/1000\n",
"4/4 [==============================] - 0s 418us/step - loss: 0.6926 - accuracy: 0.5000\n", "Epoch 998/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.5000\n", "Epoch 999/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.6926 - accuracy: 0.5000\n", "Epoch 1000/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 0.6926 - accuracy: 0.5000\n"
]
},
{
"data": {
"text/plain": [
"<keras.src.callbacks.History at 0x284e0b99c50>"
]
},
"execution_count": 5,
"metadata": {},
"output_type": "execute_result"
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
"Epoch 627/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
"Epoch 628/1000\n",
"4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
"Epoch 629/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
"Epoch 630/1000\n",
"4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:

```
                                    \n",
     "4/4 [==============================] -                                -                         -
```

```
     "Epoch 632/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
     "Epoch 633/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
     "Epoch 634/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
     "Epoch 635/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
     "Epoch 636/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
     "Epoch 637/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
     "Epoch 638/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 639/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 640/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 641/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
     "Epoch 642/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
     "Epoch 643/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
     "Epoch 644/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
     "Epoch 645/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
     "Epoch 646/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
     "Epoch 647/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
```

```
                                   \n",
```

```
     "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                                \n",
```

```
1.0000\n",
     "Epoch 648/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
     "Epoch 649/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
     "Epoch 650/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
```

```
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4016e-08 - accuracy:
1.0000\n",
     "Epoch 655/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4016e-08 - accuracy:
1.0000\n",
     "Epoch 656/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4017e-08 - accuracy:
1.0000\n",
     "Epoch 657/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4017e-08 - accuracy:
1.0000\n",
     "Epoch 658/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
     "Epoch 659/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
```

```
                                \n",
     "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                                \n",
```

```
    "4/4 [==============================] -                    -                  -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                  -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                  -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "Epoch 660/1000\n",
    "4/4 [==============================] - 0s 5ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 661/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 662/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 663/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 664/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 665/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 666/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 667/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
```

```
                        \n",
    "4/4 [==============================] -                    -                  -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                  -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                  -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] -                    -                -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] -                    -                -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] -                    -                -08 - accuracy:
1.0000\n",
                      \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 675/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 676/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 677/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 678/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 679/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 680/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 681/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 682/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 683/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
    "Epoch 684/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
```

```
                      \n",
    "4/4 [==============================] -                    -                -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] -                    -                -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] -                    -                -08 - accuracy:
1.0000\n",
                      \n",
```

```
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                          \n",
```

```
    "Epoch 685/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
    "Epoch 686/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
    "Epoch 687/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
    "Epoch 688/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
    "Epoch 689/1000\n",
    "4/4 [==============================] - ETA: 0s - loss: 1.4587e-09 - accuracy: 1.00 - 0s 2ms/step - loss: 1.4013e-
08 - accuracy: 1.0000\n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
    "Epoch 695/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4015e-08 - accuracy:
1.0000\n",
    "Epoch 696/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4015e-08 - accuracy:
1.0000\n",
    "Epoch 697/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4015e-08 - accuracy:
1.0000\n",
```

```
                          \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                          \n",
```

```
     "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                     \n",
     "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                     \n",
     "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                     \n",
```

```
     "Epoch 698/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
     "Epoch 699/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
     "Epoch 700/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
     "Epoch 701/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
     "Epoch 702/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
     "Epoch 703/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
     "Epoch 704/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
     "Epoch 705/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
     "Epoch 706/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
     "Epoch 707/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
```

```
                     \n",
     "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                     \n",
     "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                     \n",
     "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                     \n",
```

```
    "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                        \n",
```

```
                        \n",
    "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                    -                 -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                                \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                                \n",
    "4/4 [==============================] -                        -                        -08 - accuracy:
1.0000\n",
                                \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 715/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 716/1000\n",
    "4/4 [==============================] - 0s 1ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 717/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 718/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 719/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 720/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 721/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 722/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 723/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
```

```
                                \n",
    "4/4 [==============================] -                        -                        -08 - accuracy:
1.0000\n",
                                \n",
    "4/4 [==============================] -                        -                        -08 - accuracy:
1.0000\n",
                                \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                                \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                       -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] - 0s 2ms/step -                       -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] -                  -                  -08 - accuracy:
1.0000\n",
                         \n",
```

```
1.0000\n",
    "Epoch 724/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 725/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 726/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 727/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
    "Epoch 735/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
    "Epoch 736/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
    "Epoch 737/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
```

```
                         \n",
    "4/4 [==============================] -                  -                  -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] -                  -                  -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] - 0s 2ms/step -                       -08 - accuracy:
1.0000\n",
                         \n",
```

```
     "4/4 [==============================] - 0s 2ms/step -                                    -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] - 0s 2ms/step -                                    -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] -                     -                        -08 - accuracy:
1.0000\n",
                                \n",
```

```
1.0000\n",
     "Epoch 738/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
     "Epoch 739/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4014e-08 - accuracy:
1.0000\n",
     "Epoch 740/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
     "Epoch 741/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
     "Epoch 742/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
     "Epoch 743/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
     "Epoch 744/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
     "Epoch 745/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
     "Epoch 746/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
     "Epoch 747/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
```

```
                                \n",
     "4/4 [==============================] -                     -                        -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] -                     -                        -08 - accuracy:
1.0000\n",
                                \n",
     "4/4 [==============================] - 0s 2ms/step -                                    -08 - accuracy:
1.0000\n",
                                \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 755/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 756/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 757/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 758/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 759/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 760/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 761/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 762/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
```

```
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                          -                       -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "Epoch 763/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 764/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 765/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 766/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 767/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
```

```
                              \n",
    "4/4 [==============================] -                          -                       -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                          -                       -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                              -08 - accuracy:
1.0000\n",
                                          \n",
    "4/4 [==============================] - 0s 2ms/step -                                              -08 - accuracy:
1.0000\n",
                                          \n",
    "4/4 [==============================] - 0s 3ms/step -                                              -08 - accuracy:
1.0000\n",
                                          \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
    "Epoch 775/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
    "Epoch 776/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
    "Epoch 777/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
    "Epoch 778/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
    "Epoch 779/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4013e-08 - accuracy:
1.0000\n",
    "Epoch 780/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 781/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 782/1000\n"
   ]
  },
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
```

```
                                          \n",
    "4/4 [==============================] -                    -                               -08 - accuracy:
1.0000\n",
                                          \n",
    "4/4 [==============================] -                    -                               -08 - accuracy:
1.0000\n",
                                          \n",
    "4/4 [==============================] - 0s 2ms/step -                                              -08 - accuracy:
1.0000\n",
                                          \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                              -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] - 0s 2ms/step -                              -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] - 0s 3ms/step -                              -08 - accuracy:
1.0000\n",
                          \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 783/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 784/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 785/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 793/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 794/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 795/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 796/1000\n",
```

```
                          \n",
    "4/4 [==============================] -                  -                -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                  -                -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                  -                -08 - accuracy:
1.0000\n",
                          \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                  -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] - 0s 2ms/step -                                  -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] - 0s 3ms/step -                                  -08 - accuracy:
1.0000\n",
                         \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 797/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 798/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 799/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 800/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 801/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 802/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 803/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 804/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 805/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
```

```
                         \n",
    "4/4 [==============================] -                  -                  -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] -                  -                  -08 - accuracy:
1.0000\n",
                         \n",
    "4/4 [==============================] -                  -                  -08 - accuracy:
1.0000\n",
                         \n",
```

```
     "4/4 [==============================] - 0s 2ms/step -                                             -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] - 0s 2ms/step -                                             -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] - 0s 3ms/step -                                             -08 - accuracy:
1.0000\n",
                                          \n",
```

```
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
     "Epoch 813/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
     "Epoch 814/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
     "Epoch 815/1000\n",
     "4/4 [==============================] - 0s 7ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
     "Epoch 816/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 817/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 818/1000\n",
     "4/4 [==============================] - 0s 3ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 819/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4012e-08 - accuracy:
1.0000\n",
     "Epoch 820/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
     "Epoch 821/1000\n",
     "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
     "Epoch 822/1000\n",
```

```
                                          \n",
     "4/4 [==============================] -                    -                         -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] -                    -                         -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] -                    -                         -08 - accuracy:
1.0000\n",
                                          \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 3ms/step -                                    -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 823/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 824/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 825/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
```

```
                              \n",
    "4/4 [==============================] -                  -                 -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                  -                 -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                  -                 -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 3ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 833/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 834/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 835/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 836/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 837/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 838/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 839/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 840/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 841/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 842/1000\n",
```

```
                              \n",
    "4/4 [==============================] -                      -                 -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                      -                 -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                                   -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                        -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] - 0s 2ms/step -                        -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] - 0s 3ms/step -                        -08 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 843/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 844/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 845/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 853/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 854/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 855/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 856/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
```

```
                    \n",
    "4/4 [==============================] -              -              -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -              -              -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -              -              -08 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                         -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] - 0s 2ms/step -                         -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] - 0s 3ms/step -                         -08 - accuracy:
1.0000\n",
                    \n",
```

```
1.0000\n",
    "Epoch 857/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 858/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 859/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4010e-08 - accuracy:
1.0000\n",
    "Epoch 860/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4011e-08 - accuracy:
1.0000\n",
    "Epoch 861/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 862/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 863/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 864/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 865/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
```

```
                    \n",
    "4/4 [==============================] -                -                -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -                -                -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -                -                -08 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                          -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] - 0s 2ms/step -                          -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] - 0s 3ms/step -                          -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 873/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 874/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 875/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 876/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 877/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 878/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 879/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 880/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 881/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 882/1000\n",
```

```
                        \n",
    "4/4 [==============================] -                  -                -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                  -                -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                  -                -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                         -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] - 0s 2ms/step -                         -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] - 0s 3ms/step -                         -08 - accuracy:
1.0000\n",
                      \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 883/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 884/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 885/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
```

```
                      \n",
    "4/4 [==============================] -                   -                -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] -                   -                -08 - accuracy:
1.0000\n",
                      \n",
    "4/4 [==============================] -                   -                -08 - accuracy:
1.0000\n",
                      \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                          -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] - 0s 2ms/step -                          -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                 -                      -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 893/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 894/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 895/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 896/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 897/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 898/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 899/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 900/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
    "Epoch 901/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4009e-08 - accuracy:
1.0000\n",
```

```
                        \n",
    "4/4 [==============================] -                 -                      -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                 -                      -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                 -                      -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                          -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "Epoch 902/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 903/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 904/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 905/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3998e-08 - accuracy:
1.0000\n",
    "Epoch 913/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3999e-08 - accuracy:
1.0000\n",
    "Epoch 914/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3999e-08 - accuracy:
1.0000\n",
    "Epoch 915/1000\n",
```

```
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -                    -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - os 2ms/step -                    -o8 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -                    -                    -o8 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -                    -                    -o8 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - os 3ms/step - loss: 1.3999e-08 - accuracy:
1.0000\n",
    "Epoch 916/1000\n",
    "4/4 [==============================] - os 3ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 917/1000\n",
    "4/4 [==============================] - os 3ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 918/1000\n",
    "4/4 [==============================] - os 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 919/1000\n",
    "4/4 [==============================] - os 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 920/1000\n",
    "4/4 [==============================] - os 2ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 921/1000\n",
    "4/4 [==============================] - os 3ms/step - loss: 1.4001e-08 - accuracy:
1.0000\n",
    "Epoch 922/1000\n",
    "4/4 [==============================] - os 3ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 923/1000\n",
    "4/4 [==============================] - os 2ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 924/1000\n",
    "4/4 [==============================] - os 3ms/step - loss: 1.4002e-08 - accuracy:
1.0000\n",
    "Epoch 925/1000\n",
    "4/4 [==============================] - os 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
```

```
                    \n",
    "4/4 [==============================] -                    -                    -o8 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -                    -                    -o8 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -                    -                    -o8 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -               -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -               -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 933/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 934/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 935/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 936/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 937/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 938/1000\n"
   ]
  },
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
```

```
                              \n",
    "4/4 [==============================] -                    -               -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -               -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                    -               -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                    -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -               -               -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -               -               -08 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 939/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 940/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
    "Epoch 941/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 942/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
    "Epoch 943/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4008e-08 - accuracy:
1.0000\n",
```

```
                    \n",
    "4/4 [==============================] -               -               -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -               -               -08 - accuracy:
1.0000\n",
                    \n",
    "4/4 [==============================] -               -               -08 - accuracy:
1.0000\n",
                    \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3998e-08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                    -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -              -               -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 951/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 952/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 953/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 954/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 955/1000\n",
    "4/4 [==============================] - ETA: 0s - loss: 2.3377e-08 - accuracy: 1.00 - 0s
5ms/step - loss: 1.3998e-08 - accuracy: 1.0000\n",
    "Epoch 956/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3998e-08 - accuracy:
1.0000\n",
    "Epoch 957/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3998e-08 - accuracy:
1.0000\n",
    "Epoch 958/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3999e-08 - accuracy:
1.0000\n",
    "Epoch 959/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3999e-08 - accuracy:
1.0000\n",
    "Epoch 960/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3999e-08 - accuracy:
```

```
                              \n",
    "4/4 [==============================] -              -               -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -              -               -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -              -               -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                              -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] - 0s 2ms/step -                              -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                     -                      -08 - accuracy:
1.0000\n",
                              \n",
```

```
1.0000\n",
    "Epoch 961/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 962/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
    "Epoch 963/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4000e-08 - accuracy:
1.0000\n",
```

```
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 971/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 972/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 973/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4003e-08 - accuracy:
1.0000\n",
    "Epoch 974/1000\n",
```

```
                              \n",
    "4/4 [==============================] -                     -                      -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                     -                      -08 - accuracy:
1.0000\n",
                              \n",
    "4/4 [==============================] -                     -                      -08 - accuracy:
1.0000\n",
                              \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                        -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] - 0s 2ms/step -                        -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                  -                    -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 975/1000\n",
    "4/4 [==============================] - 0s 4ms/step - loss: 1.4004e-08 - accuracy:
1.0000\n",
    "Epoch 976/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 977/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 978/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 979/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4005e-08 - accuracy:
1.0000\n",
    "Epoch 980/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 981/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 982/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.4006e-08 - accuracy:
1.0000\n",
    "Epoch 983/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.4007e-08 - accuracy:
1.0000\n",
```

```
                        \n",
    "4/4 [==============================] -                  -                    -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                  -                    -08 - accuracy:
1.0000\n",
                        \n",
    "4/4 [==============================] -                  -                    -08 - accuracy:
1.0000\n",
                        \n",
```

```
    "4/4 [==============================] - 0s 2ms/step -                              -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] - 0s 2ms/step -                              -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                      -                      -08 - accuracy:
1.0000\n",
                          \n",
```

```
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 991/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 992/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 993/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 994/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 995/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 996/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3996e-08 - accuracy:
1.0000\n",
    "Epoch 997/1000\n",
    "4/4 [==============================] - 0s 3ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 998/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 999/1000\n",
    "4/4 [==============================] - 0s 2ms/step - loss: 1.3997e-08 - accuracy:
1.0000\n",
    "Epoch 1000/1000\n",
```

```
                          \n",
    "4/4 [==============================] -                      -                      -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                      -                      -08 - accuracy:
1.0000\n",
                          \n",
    "4/4 [==============================] -                      -                      -08 - accuracy:
1.0000\n",
                          \n",
```

```
     "4/4 [==============================] - 0s 2ms/step -                                             -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] - 0s 2ms/step -                                             -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] -                                    -                       -08 - accuracy:
1.0000\n",
                                          \n",
```

```
     "4/4 [==============================] - 0s 3ms/step - loss: 1.3998e-08 - accuracy: 1.0000\n"
    ]
   },
   {
    "data": {
     "text/plain": [
      "<keras.callbacks.History at 0x27484c2fe20>"
     ]
    },
    "execution_count": 13,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "model.compile(loss='binary_crossentropy',\n",
   "              optimizer='adam',\n",
   "              metrics=['accuracy'])\n",
   "                   \n",
   "model.fit(XOR_X, XOR_Y,epochs=1000, batch_size=1, verbose=1)"
```

```
  },
```

```
                                          \n",
     "4/4 [==============================] -                                    -                       -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] -                                    -                       -08 - accuracy:
1.0000\n",
                                          \n",
     "4/4 [==============================] -                                    -                       -08 - accuracy:
1.0000\n",
                                          \n",
```

[redacted]

[redacted]

"text": [
"1/1 [==============================] - 0s 76ms/step\n",
"[[0.5029386]\n",
" [0.49986482]\n",
" [0.5019195]\n",
" [0.49771422]]\n"
]
}
],
"source": [
"Hasil_Prediksi_Keras = model.predict(XOR_X)\n",
"print(Hasil_Prediksi_Keras)"
]
},
{
"cell_type": "code",
"execution_count": 7,
"id": "52977d62",
"metadata": {},
"outputs": [
{
"data": { "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAiMAAAGdCAYAAADAAnMpAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLj
iwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8g+/7EAAAACXBIWXMAAA9hAAAPYQG0P6dpAAAuJklEQVR4nO3dfXRU9Z3H8c8kkAQ
ISXCBJOCU8KDgY3iOgfWIbjBVikLtig/HIFV5KLJocDVRSMiyGKuA9GjkySJ297igtGALGMXUWMVYakIQlYAi1JgLPtBBIgM
HP3j9mMDnlgJmTyyyTv1zn30Nz5/e5853079zO/+d07NsuyLAEAABgSYr0AAADQsRFGAACAUYQRAABgFGEEAAAYRRgBAABGEUY
AAIBRhBEAAGAUYQQAABjVyXQBvnC5XDp69Ki6d+8um81muhwAAODy7J08uRJ9enTRyEhjY9/BEUYOXroqOx2u+kyAABAMxw5c
kSXX355048HRRjp3r27JPeLiYqKMlwNAADwRVVVlex2u+c83pigCCN1X81ERUURRgAACDIXO2LBBFYAAAGAUYQQAABhFGAEAAEY
FxZwRAEDrsixL58+fl9PpNF0K2rDQ0FB16tTpkpfdIIwAABrspLs6EhAz0+hqa2h+7QRgBWtC5c9LWrVJZmXTqlBQZKQ0ZIk2YIHXubL06BKvWPq5qa2tVXl6um
pqalt842p2uXbsqPj5eYWFhzd4GYQRoAUePSqtWSStWSMePS506Sabz+fNSr17SrFnSjBlSnz6mqo0WwMHFcuVwuHTx4UKG
hoerTp4/CwsK42SQaZFmWamtrdfz4cR08eFBXXHFFkzc2cRrdeFBXXHFFkzc2a4rNsiyrhetrcVVVVYqOjpbD4eDSXrQ5hYXAoaFSl
y7SH/4gjRvXWtUhWJk6rs6cOaODBw+qX79+6tq1a1a7O2wQhhx1JTU6NDhw6pf//+ioiI8HrM1/M3IyPAJSgslMaPl1wu99IUp1O
qqXG3376dQILGtYXjqjmfcBkh7JiaOxritY0WqAPokI4edX9y9eWEUaeu7R13uPuPsDFwrW46qwUB08WFq82B1EJHcAOXfO/a/kX
r94sbtdYaGZOtE2+R1G/vSnP2nixxInq06ePbDabNm/e3OR6xcXFGj58uMLDwzVgwAAVFBQ0uV8AALA/p0i5C+7BJUWu0d6kx8jfTS3pQdYNHZSvnGOeJ1W9/NzMVnlf+HXtC2o98bBGEtEUyNC2sKcH6kPu/o3+Krnk/nb8D0cPWuXLrZl1S/2kiZKfwhI1ZHNuCvugTLEH6YQj6kEg5P5FMWO/DAcbcQURBtOFv0s0JIjJ9EnxNF+I0R42RCSC7KKsJ3zJTT92xydjb+GidrANqORvv7LPTc+0jKBYrUK6gQgEqSK1ilF0INTpaZfvthn7zuWqnNuYUOK1yC2GHIOBFbU+9q2tBcxvwGqpzkBQkJfHkVhVmLTA3LHwD5FamWGUMK4zLpqA3QrM5XFIm4hOP1LlfXpfvgSOi4JlhAW5KJKotnEGbwLOzvHTe93D2gt+PB9Shwz4dEjp7U4SDQSZMrG1aurk3TC8cm0Rns7Gn5Wbd+nhD3GgeLAzZESaORU0JCKtn8zmO4m0mYyfdsM09fRxbffzNH0Ubnud
9OcOnVKpaWlKiotleS+dLeotNSTMjMzM5WWluZpP3PmTH377bd68sknVVZWpldeeUVvvvmmHn/88ZZ5BYABnTu77yQZGtq8/iE
h7v7cGhs/FGzHVTCM5OzYsUMPPvigJk+erOuuu05xcXH67rvvvLtpv8ODB9eZ4XPj38OHDtW/fPg0aNKjeUndXs6dO/PLxz7wO
4x89tlnGjZsmIYNGZJSk9P17Bhw5SVlSXJfnVD4e/+vfvr61bt2r79uu1KTEzU0qVL9eqrr3JZL4LejBnu+4j4ebsH1iV39+e2
S/o/YB7wE03FVN5JzKQI9knPFFVfod7/7nUpLS7V79927dd999cvlwHfKcOXO0obds2LVu2TF9//bVWtKbB2xvG32DxHD81Ysb3

2rA4cOKAHH3ywBUoHzOrTx/3jZCEhvp846tr+4Q/8NgcaFkzHVTCM5Cxbtkw9evTQmDFjNHHiRKWmpmr48OEX7Td27FitXLlSy
5YtU2JiovLz8/X44497ff2SmpqqLVu26L333tOoUaNowwo36MUXX1S/fvo8bZYuXart27fLbrd7PsSjPn61F7hEhYXuOole7Fb
YdZ9c//AH6aabWqo8BClTx1Xdr/Y29AusDTl61P1bMzU1/t34rK7usrLgCeaPPPKIysrK9NFHH5kupU1p6pjx9fzdJi/tBYLJu
HHuN9T5878fsg4NdX/aq/vE2Lu3tGCBux1BBL4IluMqmEZy/LVkyRLt3r1b33zzjV566SW9/vrrmjp1qumy2iVGRoAWdO6c+wZ
OZWXuyxW7d3dPzrv9diarovla87jyd2SkTnscIbz77rtVWFiokydPasCAAZozZ45mzpxpuqw2pyVGRggjAACP5oYRyf2VzerVo
iuvuC9PDg11hw+Xyx1Qevd2zxGZPr1tj4APyoRRgJ+nxEAQMfQp4/7R++eeYRQviHMAIAaFGdO7t/NZhfpIavMAKAACMIo
w
AAACjCCMAgMAoL3dPIikvN1oJ2jjCCAAgMMrLpZwcwgguijACAACMIowAAILegw8+qEmTJnmt27hxoyIiIrRo6VIzRTVhy5Yt
u
ummm9S9e3d17dpVooaN8vzGmz8WLlyooUOHtnh9kvu35mJiYgKy7QsRRgAA7c6rr76q+++/XytWrNC8efOatY1z5861cFVuL73
oku688o6NHTtWf/7zn/X555/rnnvuocyZM/XEE08E5DnbOsIIAKBdef755zVnzhytX79e6ZN86x/++23NXz4cEVERGjAgAHKy
cnR+fPnPY/bbDatWLFCd9xxh7p166Vr7yer7CwUKNHj1a3btoUExOjsWPH6tChQw3
WduTIEc2bN0+PPfaYnn32WV199dUaNGiQ5s2bpxdeeEFLy7Vn//8ZokNjoxs3rxZNpvN83hOT052794tm8omm83mGV2pey23
3
XabunTpogEDBmjjx01eNdtsNv3jH//wrCstLZXNZtN3332nwsJCTZs2TQ6Hw7PthQsX+vufwmeEEQBAu/HUUo9poaJF2rJliyZ
PnuxZ/9FHHyktLU1z587VV199pVWrVmndunavHixV/+FCxdq8uTJ2rNnj37+85/L5XLp8ssv11tvvvaWvvvpKWVlZevrpp/Xmm
29Kks6fP69Jkybppptuoueff66ioiJNnz7dExgutHHjRpo7d67BEZAZM2YoMjJS//M/+PTa5oyZYrmzZuna665RuXXl5SovL9e
UKVM8jy9YsEB33XWXdu/erfvvvvv1/33HOP9u7d69O2x4wZ0o+XLlysqKsqz7UCOO2nAHVgBA85WXN3611TEmJJ978NiY93Ly3gnX
eo dtvv62CggLdcsstXo/l5OQo1IyPD86u7AwYMoKJFi/Tkko8qOzvbo+6+++7zGk2p61unf//+Kioq0optvvqm7775bVVVVcjgc+sl
PfqKBAwdKkq666qpGa9y/f7+io6o6MV38BrDgsLo4ABA7R//36fXm+XLloUGRmpTp06KS4urt7j7j/
730kl555ZWLbjssLEzRodGy2WwNbrulEUY
s8BoJcTqdOnPmjGpqatS1a1dJosiRI+ttNy8vRp1dbWeiaNXnbZZXrwwQeVmpqq8ePHKyUlXfffXeDYaO1J
Scn1/u7tLTUTDEXQRgBADTfjBnSHXco/FhJiTuIrFkjDR/ecJsWPGn37/wf//wf//jHeuedd9S9e3dJoqlTp5STk6O
f/vSn9fr98Jdmu3Xr5vXY+vXr9c9cQTT2jpoqVKTk5W9+7d9cIcLLL3jmdUjSa6+pn/7t39t39Tfn6+NmzzY0Pnz52v79u264YYYb6j3Xl
VdeKYfDoaNHj6rPBT9dXFtbqwMHDujmm2+WJIWEhiyLK82LTWpNiTEPUvjh9sP1IRdXxBGAADN58vXLMOHNx5GWli/f
v3o4Yc
fegJJfn6+unfvruHDh2vfnoaNGiQX9vbsWOHxoHzowZo1/84heedQcOHKjXbwxZo1/84heedQcOHKjXbwxZo1/84heedQcOHKjXbwxZo1
qVL611yvHLlSlVXV+vee++VJPXq1UsnT55UdXW1JyRdOLIRFhmp9RaPZZ9PZZ/P2ffvqpotLSvP4eNmyYZ9uSVF5erh49eh9ZbGBFY
AQLtit9tVWFioYOY8eOKTU1VaQ4sNDTDD8UT4v4vSBEmpvvVvVVVVcjgc+sl+/V3VtGmqmOKdZ+0BAXwAzqAzr1pepe3qc9ZfqUD7ZbGBFY
B4/ePCgMjMzVVRUpEOHUm9997T119/3ei8kR/96Ed6vnntmnzzGj3Gm4gJJVPa2PWSZ0TX5LgAzxMtsDMqKyvTgQMHtDgaN2ZdKkSJElJSUnq2r
W
rnn76aRo4cEBvvPFGvXuRJCQk6ODBgyotLdWJEydo9uxZ2NvvfWW1q5dq/379ys7O1s7d+7U048+KkaNGiQ7Ha7Fi5cqK+//
lpbt26tF44SEhJ06tQpFRQU6MSJE6qpqpfH5v4HfrCDgcDgsSZbD4TBdCgCoa6sDn+7a++uor6/Tp05e+seJiy5Lc/wbY1KlTrTv
vvNNr3V//+lfriiuusG644QbL4XBY+fn51pgxY6wuXbpYUVFR1ujR063Vq1d72kuyNm5kLWNM2fOWA8++KAVHR1txcTEWLN
mz bIyMjKsxMREy7Isq6Kiwpo0aZIVHx9vhYWFFWf369bOysp2PZZL1vv/22deONNirdunWzIiIrBEjRlhr166t127Tpk3WoEG
DrC5dulg/+clPrNWrV1s/PHWfOXPGuuuuu6yYmBhLkvXaa695XkteFa695XkteXp41fvx4Kzw83EpISLA2bNjgte2PP/7Yuu6666yiAjrx
htvtN566y1LknXxw4EFPm5kzZ1r/9E//ZEmysrOzG3wtTRozvp6/bf9fdJtWVVVWl6Oh0ORwORVFmS4HANqtM2fO6Oxv6ODBg+rf
v7/
XXIpmKSmRRoyQiotb7WsauNlsNlsNm3atKneXWkDoaljxtfzN0XKK6mAQAERny8+z4ibeCeGx1NEMzA8EIYA
QAER
nx8i93QDOobX9MAAACjCCMAgHqCbZgf5rTEsUIYAQB4dO7cWZICe4MrtCt1xordsMczBkBAHiEhoYqJiZGx44dkR17dpV
Npv
NcFVoiyzLUk1NjY4dO6aYmBiFhoY2e1uEEQCAl7qfjK8LJEBTYmJiPMdMcxFGAABebDab4uPj1bt3b6O/5Iq2r3PnzpcoIlKHM
AIAaFBoaGiLnGiAi2ECKwAAMIowAgAAjCKMAAAAowgjAADAKMIIAAAwijACAACMIowAAACjCCMAAMAowggAADCKM
AIAAIwijAA
AAKMIIwAAwCjCCAAAMIowAgAAjCKMAAAAowgjAADAKMIIAAAwijACAACMIowAAACjCCMAAMAowggAADCKMAIAA
IwijAAAAKMII
wAAwKhmhZG8vDwlJCQoIiJCSUlJ2rlzZ5Ptly9frsGDB6tLly6y2+16/PHHdebMmWYVDAAA2he/w8iGDRuUnp6u7OxslZSUK
DE
xUampqTp27FiD7d944w1lZGQoOztbe/fu1a9//Wtt2LBBTz/99CUXDwAAgp/fYWTZsmV65JFHNG3aNF199dVauXKlunbtqrVr1
zbY/pNPPPtHYsWN13333KSEhQbfeeqqvuvffei46mAACAjsGvMFJbW6W6i4mKlpKR8v4GQEKWkpKioqKjBPmPmGjFFFxcbEnfHz77b
f atm2bbr/99kaf5+zZs6qqqvJaAABA+9TJn8YnpyQ0+lUbGys1/rY2FiVlZU12Oe+++7TiRMn9M///M+yLEvnz5/XzJkzm/yaJ

jc3Vzk5Of6UBgAAglTAr6YpLCzUs88+q1deeUUlJSX63e9+p61bt2rRokWN9snMzJTD4fAsR44cCXSZAADAEL9GRnr27KnQoF
B
VVlZ6ra+srFRcXFyDfRYsWKAHHnhADz/8sCTpuuuuU3V1taZPn65nnnlGISH181B4eLjCw8P9KQoAAAQpvoZGwsLCNGLECB
UUF
HjWuVwuFRQUKDk5ucE+NTU19QJHaGioJMmyLH/rBQAA7YxfIyOSlJ6erqlTp2rkyJEaPXqol19frurqak2bNk2SlJaWpr59+yo
3N1eSNHHiRC1btkzDhg1TUlKSvvnmGy1YsEATJ070hBIAANBx+R1GpkyZouPhPjysrKosVFRUaOnSo8vPzPZNaDx8+7DUSMn/+
f
NlsNs2fP19/+9vf1KtXLo2cOFGLFy9uuVcBAACClsoKgu9KqqqqFBodLYfDoaioKNPlAAAAH/h6/ua3aQAAgFGEEQAAYYBRhBA
A
AGEUYAQAARhFGAACAUYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYA
QAARhFGAACAUYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhF
GAACAUYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAU
YQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAAB
gFGEEAAAYRRgBAABGNSuM5OXlKSEhQREREUpKStLOnTubbbP+Pf/xDs2fPVnx8vMLDw3XllVdq27ZtzSoYAACoL5387bbbw
walp
6dr5cqVSkpKovLly5Wamqp9+/apd+/apd+/e9drX1tZq/Pjx6t27tzZu3Ki+ffvqoKFDiomJaYn6AQBAkLNZlmX5oyEpKUmjRo3Syy+
/LElyuVyy2+2aM2eOMjIy6rVfuXXKlXnjhBZWVlalz587NKrKqkrRodFyFyOByKiopq1jYAAEDr8vX87dfXNLW1tSouSUolZKSsr3G
wgJUUpKioqKihrs8/vf/17JycmaPXu2YmNjde211+rZZ5+Vo+ls9HnOnj2rqqowrwUAALRPfoWREyKjY21mt9gbGysKio9bGysKio
qGuzz7bffauPGjXI6ndq2bZsWLFigpUuX6j//8z+bhfZ7c3FxFRod7Frvd7k+ZAAAgiAT8ahqXy6XevXXt7orVGjFihKZMMmaJnn
nlGK1eubLRPZmamHA6HZzly5Eigyw8QAAIb4NYG1Z8+eCg0NVWVlpdf6yspKcXFNdgnPj5enTt3VmhoqGfdVddpYqKCtX
W1io
sLKxen/DwcIWHh/tTGgAACFJ+jYyEhYVpxIgRKigo8KxzuVwqqChQcnJyg33Gjh2rb775Ri6Xy6b775Ri6Xy7Nu//79io+PbzCIAACAjsXvr
2nSo9O1Zsoavf7669q7d69mzZql6upqTZs2TZKUlpamzMxMT/tZs2bpf//3fV37lzt379fW7du1bPPPPqvZs2e33KsAAABBy+/
7jEyZMkXHjx9x9XVlaWKioqNHToUNToUOXn53smtR4+fFghId9kXLvdrnfnfffVePP/64rr/+evXt21t21dz587VUo891XKvAgAABC2/7zNiA
vcZAQAg+ATkPiMAAAAtjTACAAACMIowAAACjCCMAAMAowggggAAADCKMAIAIAAwijAAAAAKMIIwAAAAAowggAAADCqY4eR8nJp4UL3vwAAwjE8h+GGEDh
pIFA4AMUA
qENvF8RRoBA4KQBIFiogfcrwggAADCKMAIAAIwijAAAAKM6mS4g4MrLG/8erKG/8erKTE+9+GxMe7FwAAEBDtP4ysWuWmNemNO
URx5p/LH
sbPcsYwBoDXyAQgfU/sPIjBnSHXco/FhJiTuIrFkjDR/ecBv+T43GcNJAIPABCoHQxt+vbJZlWQHbegupqqpSdHSoHA6HoqKi
W
m7DJSXSiBFScXHjYQRozMKFF8z9pNIWTBhpysZOGLx+gCLm4kKH3K1/P3+1/ZAQIFEbdEAi+hInhw/kABf+o8fcrwgjQXJwoA
AS
LNv5+xaW9AADKMIIAAAwijACAACM6thhJD5eyspq/K4HAAAQcv7dzW7ez1Fb0Qc+AAAEKLRTHTuMAIHCSQOIBQQABwAcoBEIbe
L/q2Dc9A
wAAAePr+btjzxkBAADGEUYAAIBRhBEAAGAUYQQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACA
UYQRAABgFGE
EAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYFSzwkheXp6SEhIUERGhpKQk7dy506d+69evl81m06RJ
k5rzawEAQDvkdxjZsGGD0tPTlZWVpZKSEg0ZMkSpqak6duxYk/0OHjyoJ598Ujff3CywAADQPvkdRpYtW6YZM2Zo2rRpuvrqq7Vy5
ApKK1eu2bdvv/PXDd+7vb04VhLjeXp5PhAzD7ZdY+55LDCzgrwR8ykhtLSxapAmfyqHfH+qjvmzmnKBMW+g0LrYuKkSPSUlRRUNV
iT//WvunF8pMAgAA2yYDkByBIIIFSEXVwFQnG9ONEVVU9DEuaeZmBdH44SDyhZOg3AuvLX5Lnd2TwZrHbLAE6rSDYpmcB6aH8vZWeYDbVIx
aRrvvvmA2jlA6Ro5Dp/kVAvCloIUxIfVAGAA2AfQALQ5ssFZ4QjGyLfv6E/EZOKoA0
DRYX1+UkO3BhjSGe5U6Lxy5EOrhmqbpfjAm4AhPKF11FE6wUPEbdWxGzWEJoJxb4mlBC2ZkEAjJNrBfIGGkDaV1s/SkjNbMe6dlzgLHXUanGjmWZ7/6jYwmVfrrZaYXVeI2mVyDPMnvd55Cqk6Hu7OsN/bFfJpp8oWqOnX8fUGnJz+Rov6ioP+RuOSKHz+GnbePu6z7pVNKUAP0jrHkXzbZgSwSlfMyGfg16EcMUgHuHEU8vcrm4Vu+hw/4TSDjJC8ahHr7oF4JD
w9n9enMeFSCe/KU4YPwRUmZQUESXlzoEg==

TVW
w8AADomv8PIlClTdPz4cWVlZamiokJDhw5Vfn6+Z1Lr4cOHFRLCjVoBAIBvbJZlWaaLuJiqqipFRofL4XAoKirKdDkAAMAHvp
6
/GcIAAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAABgF
GEEAAAYRRgBA
ABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAABgFGEEAA
AYRRgBAABGEUY
AAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAABgFGEEAAAYRRgB
AABGEUYAAIBRh
BEAAGAUYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAABgFGEEAAAYRRgBAABGE
UYAAIBRhBEAAGA
UYQQAABhFGAEAAEYRRgAAgFGEEQAAYBRhBAAAGNWsMJKXl6eEhARhFREQ0KSlJO3fubLTtmjVrdOONN6pHjx7qoaOH
UlJSmmwPA
AA6Fr/DyIYNG5Senq7s7GyVlJQ0MTFRqampOnbsWIPtCwsLde+99+qDDz5QUVGR7Ha7br31V3tb3+75OIBAEDws1mWZfn
TISk
pSaNGjdLLL78sSXK5XLLb7ZozZ44yMjIu2t/pdKpHx56+eWXlZaW5tNzVlVVKTo6Wg6HQ1FRUf6UCwAADPH1/O3XyEhtba
2Ki
4uVkpLy/QZCQpSSkqKioiKftlFTU6Nz587snpsssua7TN2bNnVVVV5bUAAID2ya8wcuLECTmdTsXGxnqtj42Nqpp
9S
nTx+vQHOh3NxcRUdHexa73e5PmQAAIIi06tUozz33nzz33nNavX69NmzYpIiKioXaZmZlyOBye5ciRI61YJQAAaE2d/Gncs2dPhYaGq
rKyomt9ZWWl4uLimuy7ZMkSPffcc3r//r//fd1/fXXN9k2PDxc4eHh/pQGAACClF8jI2FhYRoxYoQKCg086iwulKCpScnNxov+e
ff16LFi1Sfn6+Ro4c2fxqAQBAu+PXyIgkpaena+rUqRo05cqRGjx6Gx6t5cuXq7q6WtOmWkpaWlqW/fvsrNzZUk/fKXv1RWVpbee
OMNJSQkeOaWREZGKjIysgVfCgAACEZ+h5EpEpU6bo+PHjysrTKaotrZWP/vZz7y/3Z8x9B/s6N/oPkj6ta
2k52
drYULF15a9QAAIOj5fZ8RE7jRCPCAAAwScg9xkBAABoaYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGA
EAAEYRR
gAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGA
EAAEYRRgAAgFG
EEQAAYBRhBAAAGEUYAQAARhFGAACAUYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYR
RgAAgFGEEQAAY
BRhBAAAGEUYAQAARhFGAACAUYQRAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABhFGAEAAEYRRgAAgF
GEEQAAYBRhBAA
AGEUYAQAARhFGAACAUYQRAABgFGEEAAAY1cloASacOydt3SqVlUmnTkmRkdKQIdKECVLzqarAwMg8NrSubBDhZGjR6
VVq6QVK
6Tjx6VOnSSbTb1s5fx5qVcvadYsUCANDy2uK50GZZltU6T9V8VVVVi06lOlsPhUFRVluPhUFRUVLO2UVgo0ZWonT4tOZ2N
tws
NlbpokQ7wB2ncuGY9FQAAbVJrnwt9PX93iDkYS+PFSTU3TO19yP15T425fWNga1QEAEHht+Vzq1EhQ7yHtYT3SPIoaPuFOhyuRdf1LW94
w53fwAAgllbPxc2K4zk5eUpISFBERERSkpKUk7Os6jVsTFs6ds/9ZbbnIkCGKiIjQddd23btjWr2OZYtco9HOXrzl2aNLml9O8
pCwCA1tLWz4V+h5n5ENGzYoT1PT1pbd5V8iHlTK7AHkEeLHNgzPn9i72dnZtO2SiioKFIy7oSPFYz9IekS69o56nQWRlSk8ahTrtWeVWpJSU1MbS9SbkrK9xsICVFKSoqKiooo2Z4nqZ
8+eVVVVVldfSHGVl7kuWLkVoqHs7AAAEo2A4F/oVRk6cOCGno6nY2Fiv9bGxsaqoqGiwz6KnoqGiwToVFhV/tJSk3N1fRodGexW63+1Omx6l
T7munLoVIiHTy5KVtAwAU4LhXNgmr6bJzMyUw+bMyUw+HwLEeOHGnWdiIj3TdxuRQul9S9+6VtAwAAU4LhXOjXwE3Pnjo
qyspKr
/WVlZWKi4trsE9cXJf7SUpPDxc4eHh/pTWoCFD3HeX88Dfe7USTuxROp3s7AAAEo2A4F/o1MhIWFqYRIoaooKDAs87lcqmg0EDJyckN9
kl
OTvZqLonbt29vtH1LmjDBfVvbS9G7t3T77S1TDwAArSoYzoV+fo2Tnp6uuNWvW6PXuWvW6PXXX9fevXs1a9YoaZKktLQoZWZ
me
trPnTtX+fn5WrpoqcrKyrRw4UJ99tlnevTRRvuVTSic2f3/fVDQ5vXPyTE3Z8fzwMABKtgOBf6HUamTJmTJmiJUuWKCsrSoOHDlV
paany8/M9k1QPHz6s8vJyT/sxY8obTfeoOrVq5WYmKiNGdzq8+bNuvbaa1vuvuVTRhxgz3/fVD/HylISFSt27S9OmBqQsAgNbS1
s+FHeKH8urux+/rbXBDQtzL++9LN93kf7oAALQ1JsF/FDeD4wbJ23f7k53FxumqkuBBBEAQHvSls+FHSKMSO7/CGVlovz530o/
kCQ1fwdW9x+ld29pwQJ3O4IAKC9aavnwg7xNc2Fzp2Ttm1z7+iTz7iT9g8Z4p4pGRVAEBHoBrnQl/P3xoyjAAAgMBjzggAA

AgKhBEAAGAUYQQAABhFGAEAAEYRRgAAgFF+/WqvKXUX/FRVVRmuBAAA+KruvH2xC3eDIoycPHlSkmS32w1XAgAA/H
Xy5ElFRoc
3+nhQ3GfE5XLp6NGj6t69u2w2W4ttt6qqSna7XUeOHOH+JRfBvvIP+8t37Cvfsa98x77yXSD3lWVZOnnypPro6aOQJn6lLyhGR
kJCQnT55ZcHbPtRUVEcrD5iX/mH/eU79pXv2Fe+Y1/5LlD7qqkRkTpMYAUAAEYRRgAAgFEdOoyEh4crOztb4eHhpktp89hX/
mF
/+Y595Tv2le/YV75rC/sqKCawAgCA9qtDj4wAAADzCCMAAMAowggAADCKMAIAAIxq92EkLy9PCQkJioiIUFJSknbu3Nlk+7fe
e
ktDhgxRRESErrvuOm3btq2VKjXPn321bto62Ww2ryUiIqIVqzXnT3/6kyZOnKg+ffrIZrNp8+bNF+1TWFio4cOHKzw8XIMGDd
K
6desCXmdb4O++KiwsrHdc2Ww2VVRUtE7BBuXm5mrUqFHq3r27evfurUmTJmnfvnoX7dcR37Oas6866nvWihUrdP3113tuaJac
n
Kx33nmnyT4mjql2HUY2bNig9PRoZWdnq6SkRImJiUpNTdWxY8cabP/JJ5/o3nvv1UMPPaRdu3ZpoqRJmjRpkr744otWrrz1+bu
vJPfd+srLyz3LoUOHWrFic6qrq5WYmKi8vDyf2h88eFATJkzQzTffrNLSUj322GN6+OGH9e677wa4UvP83Vd19u3b53Vs9e7dO
oAVthoffvihZs+erU8//VTbt2/XuXPndOutt6q6urrRPh31Pas5+orqmO9Zl19+uZ577jkVFxfrs88+0y233KI777xTX375ZYP
tjR1TVjs2evRoa/bs2Z6/nU6n1adPHys3N7fB9nfffbc1YcIEr20pK0aeMq6ZrZoXhPdDwxGaOXOmNm3apKuvvvvrrBtqaOqaD41V6oPc
n
JyV7JesyYMbrqqquoatUqLVq0yGBlCGaDBw/W4MGDPX+PGTNG1GBw4c0Isvvqj/+q//MlhZ65o9e7a++OILffzxx6ZLafN83Vcd+

```
8E4gSg1pjx1VUVJS6dOliqKrgMXr06A51XD366KPasmWLPvjgA11++eVNtu2071l1/NlXF+pI71lhYWEaNGiQRowYodzcXCUmJ
upXv/pVg21NHVPtN0yEhYVpxIgRKig08KxzuVwqKCh09Luy5ORkr/aStH379kbbtxfN2VcXcjqd2rNnj+Lj4wNVZtDqqMdVSyk
tLeoQx5VlWXr00Ue1adMm/fGPfiT//vov2qejHlvN2VcX6sjvWS6XS2fPnm3wMWPHVECnxxq2fv16Kzw83Fq3bp311VdfWdOn
T          7diYmKsi00Ky7Is64EHHrAyMjI87Xfs2GF16tTJWrJkibV3714rOzvb6ty5s7Vnzx5TL6HV+LuvcnJyrHfffdc6cOCAVVxcbN1
zzz1WRESE9eWXX5p6Ca3m5MmT1q5du6xdu3ZZkqxly5ZZu3btsg4dOmRZlmVlZGRYDzzwgKf9t99+a3Xt2tX693//d2vv3r1WX
l6eFRoaauXn55t6Ca3G33314osvWps3b7a+/vpra8+ePdbcuXOtkJAQ6/333zf1ElrNrFmzrOjoaKuwsNAqLy/3LDU1NZ42vGe
5NWdfddT3rIyMDOvDDz+0Dh48aH3++edWRkaGZbPZrPfee8+yrLZzTLXrMGJZlvXSSy9ZP/rRj6ywsDBr9OjR1qeffup57Kabb
rKmTp3q1f7NN9+0rrzySissLMy65pprrK1bt7Zyxeb4s68ee+wxT9vY2Fjr9ttvtopKSgxU3frqLj+9cKnbP1OnTrVuuummen2
GDh1qhYWFWQMGDLBee+21Vq/bBH/31S9/+Utr4MCBVkREhXZZZdZ48aNs/74xz+aKb6VNbSfJHkdK7xnuTVnX3XU96yf/
/znV
r9+/aywsDCrV69e1r/8y794g0hltZ1jymZZlhXYsRcAAIDGtds5IwAAIDgQRgAAgFGEEQAAYBRhBAAAGEUYAQAARhFGAACA
UYQ
RAABgFGEEAAAYRRgBAABGEUYAAIBRhBEAAGAUYQQAABj1f947F4hDOeJrAAAAAElFTkSuQmCC",
      "text/plain": [
       "<Figure size 640x480 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "import matplotlib.pyplot as plt\n",
    "plt.plot(XOR_Y, 'bo', label='Target', linewidth=2, markersize=12)\n",
    "plt.plot(Hasil_Prediksi_Keras, 'r+', label='Keras Output', linewidth=2, markersize=12)\n",
    "plt.legend(loc='upper right')\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "id": "a8e13d18",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "MSE = 0.24972152203319542\n",
      "RMSE = 0.4997214444399954\n"
     ]
    }
```

```
  ],
  "source": [
   "from sklearn.metrics import mean_squared_error\n",
   "from math import sqrt\n",
   "mse2  = mean_squared_error(XOR_Y, Hasil_Prediksi_Keras)\n",
   "rmse2 = sqrt(mean_squared_error(XOR_Y, Hasil_Prediksi_Keras))\n",
   "print('MSE =',mse2)\n",
   "print('RMSE =',rmse2)"
  ]
```

```
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.11.6"
 }
```

```
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

M. ITCLab-12

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# PID Parameter Tuning Using Deep Learning\n"
   ]
  },
  {
   "cell_type": "raw",
   "metadata": {},
   "source": [
    "By: IO-T.NET Team (https://io-t.net/itclab)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "![PID_Deep_Learning](Deep_PID.jpg)\n",
    "Proses penalaan nilai Kc, τI dan τD pada\n",
    "pengendali PID menggunakan Deep Learning"    ]
```

```
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Data Latih\n",
  "\n",
  "Misalkan error dan delta_error ideal untuk membangkitkan gain PID Kc, tauI dan tauD, sebagai berikut: "
 ]
},
{
 "cell_type": "code",
 "execution_count": 43,
 "metadata": {},
 "outputs": [],
 "source": [
  "# Data Latih.\n",
  "X = np.array([\n",
  "    [1, 1],\n",
  "    [0.4, 1.2],\n",
  "    [1.2, 0.1],\n",
```

```
  "    [1, 0.1]\n",
  "])\n",
  "\n",
  "# Label untuk Data Latih.\n",
  "y = np.array([\n",
  "    [0.25, 4.31, 0.20],\n",
  "    [0.2, 4.1, 0.1],\n",
  "    [0.1, 4.0, 0],\n",
  "    [0.1, 4.0, 0]\n",
  "])"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Arsitektur Deep Learning\n",
  "Arsitektur Deep Learning dengan Dua Masukan dan Tiga Keluaran"
 ]
},
```

```
  "model = Sequential()\n",
  "\n",
  "# Tambahkan lapisan masukan \n",
  "model.add(Dense(2, activation='sigmoid', input_shape=(2,)))\n",      "\n",
  "# Tambahkan satu lapisan tersembunyi\n",
  "model.add(Dense(3, activation='sigmoid'))\n",
  "\n",
  "# Tambahkan lapisan keluaran\n",
  "model.add(Dense(3, activation='sigmoid'))"
 ]
},
{
 "cell_type": "code",
 "execution_count": 45,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Model: \"sequential_1\"\n",
    "_________________________________________________________________\n",
    " Layer (type)                Output Shape              Param #   \n",
    "=================================================================\n",
    " dense_3 (Dense)             (None, 2)                 6         \n",
    "                                                                 \n",
    " dense_4 (Dense)             (None, 3)                 9         \n",
    "                                                                 \n",
    " dense_5 (Dense)             (None, 3)                 12        \n",
    "                                                                 \n",
    "=================================================================\n",
    "Total params: 27 (108.00 Byte)\n",
    "Trainable params: 27 (108.00 Byte)\n",
    "Non-trainable params: 0 (0.00 Byte)\n",
    "_________________________________________________________________\n"
   ]
  },
  {
   "data": {
    "text/plain": [
     "[array([[ 1.0956396 , -0.40167177],\n",
     "        [-0.1714741 , -0.14249814]], dtype=float32),\n",
     " array([0., 0.], dtype=float32),\n",
     " array([[ 0.6540258 ,  1.0830467 ,  0.44223344],\n",
     "        [ 0.26533806, -0.691223  ,  0.9030155 ]], dtype=float32),\n",
     " array([0., 0., 0.], dtype=float32),\n",
     " array([[ 0.26789904,  0.15584779, -0.4230957 ],\n",
     "        [-0.9043672 ,  0.6550486 , -0.40327096],\n",
     "        [ 0.6617336 , -0.674325  ,  0.31031728]], dtype=float32),\n",
```

```
   "\n",
   "# Ringkasan model\n",
   "model.summary()\n",
   "\n",
   "# Konfigurasi model\n",
   "model.get_config()\n",
   "\n",
   "# Buat daftar semua tensor bobot \n",
   "model.get_weights()"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Untuk pelatihan Deep Learning silahkan ketikkan skrip berikut."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 46,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Epoch 1/100\n",
     "4/4 [==============================] - 1s 2ms/step - loss: 0.5775 - accuracy: 0.7500\n",      "Epoch 2/100\n",
     "4/4 [==============================] - 0s 0s/step - loss: 0.5559 - accuracy: 1.0000\n",      "Epoch 3/100\n",
     "4/4 [==============================] - 0s 0s/step - loss: 0.5344 - accuracy: 1.0000\n",      "Epoch 4/100\n",
     "4/4 [==============================] - 0s 0s/step - loss: 0.5128 - accuracy: 1.0000\n",      "Epoch 5/100\n",
     "4/4 [==============================] - 0s 0s/step - loss: 0.4914 - accuracy: 1.0000\n",
     "Epoch 6/100\n",
     "4/4 [==============================] - 0s 0s/step - loss: 0.4700 - accuracy: 1.0000\n",      "Epoch 7/100\n",
     "4/4 [==============================] - 0s 4ms/step - loss: 0.4487 - accuracy: 1.0000\n",      "Epoch 8/100\n",
     "4/4 [==============================] - 0s 292us/step - loss: 0.4274 - accuracy: 1.0000\n",      "Epoch 9/100\n",
     "4/4 [==============================] - 0s 5ms/step - loss: 0.4061 - accuracy: 1.0000\n",      "Epoch 10/100\n",
     "4/4 [==============================] - 0s 5ms/step - loss: 0.3849 - accuracy: 1.0000\n",      "Epoch 11/100\n",
     "4/4 [==============================] - 0s 5ms/step - loss: 0.3636 - accuracy: 1.0000\n",      "Epoch 12/100\n",
     "4/4 [==============================] - 0s 5ms/step - loss: 0.3425 - accuracy: 1.0000\n", "Epoch 13/100\n",
```

```
"Epoch 14/100\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.3004 - accuracy: 1.0000\n",
"Epoch 15/100\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.2794 - accuracy: 1.0000\n",      "Epoch 16/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.2585 - accuracy: 1.0000\n",      "Epoch 17/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.2376 - accuracy: 1.0000\n",      "Epoch 18/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.2167 - accuracy: 1.0000\n",      "Epoch 19/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.1959 - accuracy: 1.0000\n",      "Epoch 20/100\n",
"4/4 [==============================] - 0s 0s/step - loss: 0.1752 - accuracy: 1.0000\n",      "Epoch 21/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.1545 - accuracy: 1.0000\n",
"Epoch 22/100\n",
"4/4 [==============================] - 0s 3ms/step - loss: 0.1339 - accuracy: 1.0000\n",      "Epoch 23/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.1133 - accuracy: 1.0000\n",      "Epoch 24/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.0928 - accuracy: 1.0000\n",      "Epoch 25/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.0723 - accuracy: 1.0000\n",
"Epoch 26/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.0518 - accuracy: 1.0000\n",
"Epoch 27/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.0315 - accuracy: 1.0000\n",
"Epoch 28/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: 0.0111 - accuracy: 1.0000\n",      "Epoch 29/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.0091 - accuracy: 1.0000\n",      "Epoch 30/100\n",
"4/4 [==============================] - 0s 3ms/step - loss: -0.0294 - accuracy: 1.0000\n",      "Epoch 31/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.0496 - accuracy: 1.0000\n",      "Epoch 32/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.0697 - accuracy: 1.0000\n",
"Epoch 33/100\n",
"4/4 [==============================] - 0s 6ms/step - loss: -0.0898 - accuracy: 1.0000\n",
"Epoch 34/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.1098 - accuracy: 1.0000\n",
"Epoch 35/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.1297 - accuracy: 1.0000\n",      "Epoch 36/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.1497 - accuracy: 1.0000\n",      "Epoch 37/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.1696 - accuracy: 1.0000\n",      "Epoch 38/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.1895 - accuracy: 1.0000\n",      "Epoch 39/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.2092 - accuracy: 1.0000\n",      "Epoch 40/100\n",
"4/4 [==============================] - 0s 3ms/step - loss: -0.2289 - accuracy: 1.0000\n",      "Epoch 41/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.2485 - accuracy: 1.0000\n",
"Epoch 42/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.2682 - accuracy: 1.0000\n",
```

```
"Epoch 44/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.3073 - accuracy: 1.0000\n",      "Epoch 45/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.3268 - accuracy: 1.0000\n",      "Epoch 46/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.3462 - accuracy: 1.0000\n",
"Epoch 47/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.3655 - accuracy: 1.0000\n",      "Epoch 48/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.3848 - accuracy: 1.0000\n",
"Epoch 49/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.4041 - accuracy: 1.0000\n",      "Epoch 50/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.4234 - accuracy: 1.0000\n",      "Epoch 51/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.4425 - accuracy: 1.0000\n",      "Epoch 52/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.4616 - accuracy: 1.0000\n",      "Epoch 53/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.4807 - accuracy: 1.0000\n",      "Epoch 54/100\n",
"4/4 [==============================] - 0s 1ms/step - loss: -0.4998 - accuracy: 1.0000\n",      "Epoch 55/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.5187 - accuracy: 1.0000\n",      "Epoch 56/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.5377 - accuracy: 1.0000\n",      "Epoch 57/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.5565 - accuracy: 1.0000\n",      "Epoch 58/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.5754 - accuracy: 1.0000\n",      "Epoch 59/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.5941 - accuracy: 1.0000\n",      "Epoch 60/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.6129 - accuracy: 1.0000\n",      "Epoch 61/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.6315 - accuracy: 1.0000\n",
"Epoch 62/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.6502 - accuracy: 1.0000\n",      "Epoch 63/100\n",
"4/4 [==============================] - 0s 2ms/step - loss: -0.6688 - accuracy: 1.0000\n",      "Epoch 64/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.6873 - accuracy: 1.0000\n",      "Epoch 65/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.7059 - accuracy: 1.0000\n",      "Epoch 66/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.7243 - accuracy: 1.0000\n",      "Epoch 67/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.7427 - accuracy: 1.0000\n",
"Epoch 68/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.7611 - accuracy: 1.0000\n",      "Epoch 69/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.7794 - accuracy: 1.0000\n",      "Epoch 70/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.7977 - accuracy: 1.0000\n",      "Epoch 71/100\n",
"4/4 [==============================] - 0s 10ms/step - loss: -0.8160 - accuracy: 1.0000\n",      "Epoch 72/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.8341 - accuracy: 1.0000\n",      "Epoch 73/100\n",
    "4/4 [==============================] - 0s 5ms/step - loss: -0.8523 - accuracy: 1.0000\n",
```

```
"Epoch 75/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.8885 - accuracy: 1.0000\n",
"Epoch 76/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.9065 - accuracy: 1.0000\n",       "Epoch 77/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.9244 - accuracy: 1.0000\n",       "Epoch 78/100\n",
"4/4 [==============================] - 0s 3ms/step - loss: -0.9424 - accuracy: 1.0000\n",       "Epoch 79/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.9602 - accuracy: 1.0000\n",       "Epoch 80/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -0.9781 - accuracy: 1.0000\n",       "Epoch 81/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -0.9960 - accuracy: 1.0000\n",
"Epoch 82/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.0137 - accuracy: 1.0000\n",       "Epoch 83/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.0314 - accuracy: 1.0000\n",       "Epoch 84/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.0492 - accuracy: 1.0000\n",       "Epoch 85/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.0668 - accuracy: 1.0000\n",       "Epoch 86/100\n",
"4/4 [==============================] - 0s 4ms/step - loss: -1.0843 - accuracy: 1.0000\n",       "Epoch 87/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.1019 - accuracy: 1.0000\n",       "Epoch 88/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.1195 - accuracy: 1.0000\n",
"Epoch 89/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.1369 - accuracy: 1.0000\n",       "Epoch 90/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.1545 - accuracy: 1.0000\n",       "Epoch 91/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.1718 - accuracy: 1.0000\n",       "Epoch 92/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.1892 - accuracy: 1.0000\n",       "Epoch 93/100\n",
"4/4 [==============================] - 0s 1ms/step - loss: -1.2066 - accuracy: 1.0000\n",
"Epoch 94/100\n",
"4/4 [==============================] - 0s 4ms/step - loss: -1.2238 - accuracy: 1.0000\n",       "Epoch 95/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.2412 - accuracy: 1.0000\n",       "Epoch 96/100\n",
"4/4 [==============================] - 0s 838us/step - loss: -1.2584 - accuracy: 1.0000\n",       "Epoch 97/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.2756 - accuracy: 1.0000\n",       "Epoch 98/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.2928 - accuracy: 1.0000\n",       "Epoch 99/100\n",
"4/4 [==============================] - 0s 0s/step - loss: -1.3099 - accuracy: 1.0000\n",       "Epoch 100/100\n",
"4/4 [==============================] - 0s 5ms/step - loss: -1.3270 - accuracy: 1.0000\n"
```

```
   "                    \n",
   "model.fit(X, y,epochs=100, batch_size=1, verbose=1)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 47,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "1/1 [==============================] - 0s 47ms/step\n",
     "[[0.24453239 0.8341355  0.18436413]\n",
     " [0.25178796 0.82828206 0.18918379]\n",
     " [0.24446805 0.8346314  0.18357222]\n",
     " [0.24636793 0.83322346 0.18463148]]\n"
    ]
   }
  ],
  "source": [
   "Hasil_Prediksi_Keras = model.predict(X)\n",
   "print(Hasil_Prediksi_Keras)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "### Dicoba diberi masukan e(t) sembarang\n",
   "Pengujian ke-1"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 48,
  "metadata": {},
  "outputs": [],
  "source": [
   "ujicoba1 = np.array([\n",
```

[redacted]

[redacted]

```
 "source": [
  "ujicoba1"
 ]
},
{
 "cell_type": "code",
 "execution_count": 50,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "1/1 [==============================] - 0s 63ms/step\n"
   ]
  }
 ],
 "source": [
  "outDL = model.predict(ujicoba1)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 51,
 "metadata": {},
 "outputs": [
```

```
   {
    "data": {
     "text/plain": [
      "array([[0.24453239, 0.8341355 , 0.18436413]], dtype=float32)"
     ]
    },
    "execution_count": 51,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "outDL"
  ]
 },
```

```
     ]
    },
    "execution_count": 53,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "result_Kc"
  ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 54,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "0.8341355"
    ]
   },
   "execution_count": 54,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "result_tauI"
 ]
},
{
 "cell_type": "code",
 "execution_count": 55,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "0.18436413"
    ]
```

iwgaHRocHM6Ly9tYXRwbG90bGliLm9yZy8g+/7EAAAACXBIWXMAAA9hAAAPYQGoP6dpAAAr4klEQVR4nO3dfXTU1Z3H
8c/MQBI
iZMACSUgGg40PWCUumBhrVtDUuOsqNGVJ8SEhVVxFLTTaBXxIVjoan2qToyhdFo7WHiXIZtc9i4Vuo/GgRtBwbKGAVg0kw
UwgK
hlESdiZu39kGRhJMAMzuUzyfp3zO5g7987v+7snx/nk/h7GYYwxAgAAsMRpuwAAADCwEUYAAIBVhBEAAGAVYQQAAFhFG
AEAAFY
RRgAAgFWEEQAAYBVhBAAAWDXIdgG9EQgE9Nlnn2nYsGFyOBy2ywEAAL1gjNH+/fs1ZswYOZo9r3/ERBj57LPP5PF4bJcBA
ABOQ
FNTk9LTo3t8PSbCyLBhwyR1HUxSUpLlagAAQG/4fD55PJ7g53hPYiKMHD41k5SURBgBACDGfNclFlzACgAArCKMAAAAqwgj
AAD
Aqpi4ZqQ3/H6/Dho6ZLuMmDZ48GC5XC7bZQAABph+EUa++uorNTc3yxhju5SY5nA4lJ6erqFDh90uBQAwgMR8GPH7/Wpu
blZiY
qJGjRrFQ9FOkDFGe/fuVXNzs8aPH88KCQCgz8R8GDlo6JCMMR01apSGDBliu5yYNmrUKO3cuVOHDhoijAAA+ky/uYCVFZG
Txxw
CAGyI+ZURALHLH/BrXeM6texvUeqwVOWOzZXLyaocMNAQRgBYUbO1RvPWzFOzrznYlp6UrqprlRwfoHFygDotX5zmua
k+fiSX
Z3o6qtd//r9tisC+q2arTWasXJGSBCRpN2+3ZqxcoZqpkTIypKlTpRtu6Po3I6OrPUpmz56t6dOnh7StStWrV
KCQkJ+uUvfxm1/QK2+QN+zVszTobH3op/uG3+mvnyB/iDABgoCCM1NdKMGVJz6F9zqz2KgeRo//Zv/6Ybb7xRL7zwgu65
5
54+2Sdgw7rGdcesiBzNyKjJ16R1jev6sCoANg3sMOL3S/PmLl5/z5UT9l8+STT+ruu/WihUrVFJSIkkBAJ68skndfb
ZZys+Pl5jx47Vo48+GtU6gL7Qsr8lovoAxL6BfQHrunXhrogczXRgczRipqamr35/Xf//3fuuqqq4LttiixtotKlS
/WrX/1Kl19+uVpaWrRt27ao1ADopdRhqRHtByDewwotLLv7x62y9y9Mv/vd7/T666+rtrZW
V15guWZWfevevfjWX5guuugiZWRkqLy8XF999VWwfevW/ro6dI+oBjRstuvyyxu/WNddco/3790osSj7V
HvidwfoFWzVyltKSokPbopHStmrmK54wAA8zADiMul1TV9RfaMYk+OM8+VlYFkAhQ1Rb
Wxu1/
QK2FZxfoJ3zduqt4rfoSsEreqv4Le2Yt4MgAgxAAzuMSFJBgbRqlZQW+heaotO72gui/z9Gj8ejuro67dmzR/n5+ers7NSCBQv
oz//8z/rNb36j7du36/3339eyZcuiXgvQl1xOl6ZkTNGsC2dpSsYUTsoAA9TAvoD1sII
Cadqorrtm
Wlq6rhHJzY3qisi3paenq
66uTlOnTlV+fr7Wrl2rQYMGqaysTJ999plSU1N1++2391k9AAD0FYcx3T1k49Ti8/nk
dTj
yoHTt2aNy4cUp
ISLBUYf/AXAIAIul4n99H4zQNAACwijACAACsIowAAACrCCMAAMAqwggAALDqhMLI4sWLlZGRoYSEBGVnZ2vDhg3H7V
9ZWalzz
z1XQ4YMkcfjoc9//nMdPHjwhAoGAAD9S9hhpLq6WgWjRs1ceJE5efna8+ePd32f+WVV7Rw4UjaatWrZsma
q rq3XfffeddPEAACD2hR1GnnnmGc2ZMoclJSWaMGGClixZosTERC1fvrzb/u+9955+8IMf6IYbblBGRoauvvpqzZo16ztXUwAAw
MAQVhjp7OxUQoOD8vLyjryByBo6m8vDzV19d3O+ayyy5TQoNDMHz89a9/1RtvvKG///u73E/HROd8vl8IRsAAOifwgojbW1t8v
v
9Sk5ODmlPTk6W1+vtdswNN9yghx3Px6ss846S1OmTDnuaZqKigq53ZX3c46+QP+q
O5vypQpmj9/fsy8LwAAoRL1u2nq6uro2GOP6fnnn9fGjRtVU1Oj1atX65FHHHulxzKJFi9Te3h7cmpqaolpjzdYaZVVRlaOpLU3V
DzQ2a+tJUZVRlqGZrTVT3CwAAwgwjIoeOlMvlUmtraaoh7a2urUlJSqumaabb9att96qCy+8UD/6oY/o2GOPqaKiQ0FAo
Nsx8fHxSkpKCtmipWZrjWasnNI+27fbs1YOSMqgWT27Nl6++23VVVVJYfDIoe3bt+uWHHjNGTIEJ177rmqq
qo
KGdfdqsfo6dM1e/bsiNcIAEBfCSuMxMXFKTs7W7W1tcG2QCCg2tpa5ebmdjvmwIEDcjpD38bl6roLw/YDvPwBv+atmSfTzWPuD7fNXzNf/gBPDYvMaI1Tkp64QJfkmcVUOcurZaYXPsWDNe2c3yzg8kqp+iNHewaztNGXzY4NO3vdKsTEXVy4n/JT/xt1TUijHjnJ6R4aKE

9vmr5kf8VM2VVVVysnJoZw5c9TSoqKWlhalp6crPT1dr732mrZs2aKysjLdd999WrlyZUT3DQDAqWZQuANKSotVXFysyZMn
Kys
rS5WVlTpw4IBKSkokSUVFRUpLS1NFRYUk6brrrtMzzzyjiy++WNnZ2frooo/14IMP6rrrrguGElvWNa47ZkXkaEZGTb4mrWtcp
ykZUyK2X7fbrbi4OCUmJoasKD3ooEPB/x43bpzq6+u1cuVKzZw5M2L7BgDgVBN2GCksLNTevXtVVlYmr9erzMxMrVmzJnhRa
2N
jY8hKyAMPPCCHw6EHHnhAu3fv1qhRo3TddfpoUcfjdxRnKCW/SoR7XeyFi9erOXLl6uxsVHffPONOjs7lZmZ2Sf7BgDAlrDDi
CTdddddduuuuu7p9ra6uLnQHgwapvLxc5eXlJ7KrqEodlhrRfidjxYoYoVuvfee/XLX/5SOTk5GjZsmJ566imtX78+2MfpdB5zauv
QoUNRrwoAgGgaoN9Nkzs2V+lJ6XLIoe3rDjnkSfIod2xuxPcdFxcnv//ItSjvvvuLrvsMs2dO1cXX3yzxzj77bG3fvj1kzKhRo
9TScmSVxu/3a/PmzRGvDQCAvjSgw4jL6V6Bz+ufKa+Srmckb+2JSMjQ+vvxXr9fOnPH68MPPP9TatWv1tam8Pvy1v
Se
f6MEHH9QHH3wQMubKK6/U6tWrtXr1am3btk133HGH9u3bF/HaAAD0SwM6jEhSwfkfkFWjVzldKSoklaW0o5a5PStWrmKhWcXx
CV/d577
71yuVyaMGGCRooapfz8fBUUFKiwsFDZ2dn6/PPPPNXfu3JAxP/3pT1VcXKyioiJdccUVOvPMMzV16tSo1AcAQF9xGNv31/aCz+e
T2+1We3v7Mc8cOXjwwoHbs2KFx48YpISHhhPfhD/i1rnGdWva3KHVYqnLH5kZlReRUFqm5BABAOv7n99FO6/ALW/sjldEXo9lo
AA
NA7A/4oDQAAsIswAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMPL//H6prk569d
Wuf4/6Qt2
omDJliubPnx+V93U4HHI4HIqPj1daWpquu+461dTURHxfAABEAmFEUk2oq3XBD178ZGV3tsWjOnDlqaWnQ9u3b9e
///
u+aMGGCfvKTn+i2226zXRoAAMcY8GGkpaaMUNqbg5t3727qzoagWT27Nl6++23VVVVVFzF2L59u2655RaNGzdOQ4YMob
nnnqu
qqqqQcd2tpkyfPl2zZ88OaUtMTFRKSorSo9N16aWX6oknntCvf/1rLV26VH/4wx8ifoAAAJyEAR1G/H5p3jjJ/H5p37fkT
9lUVVUpJycnuILRotKi9PRopaen67XXXXtOWLVtUVlam++67TytXrozIPouLizVixAizwAhOtjkD+lt71607dkXkaMZITU1d/aZ
Midx+3W634uLigisYhz300EPB/x43bpzq6+u1cuVKzZw586T36XQ6dc4552jnzpon/V4AAETSgA4jLS2R7XeyFi9erOXLl6uxs
VHffPONOjs7lZmZGbH3N8bI4XBE7PoAAIiEAX2ah0OAoAlEAX2aJjU1sv1OxVK0VK3Tvvvvfqllt+tuoe9//NFHKikpUWdnZ7CPo+mU+dY5pUO
HDvXq/f1+v/7yl7/7yl79o3LhxEaobAIBCTNaBXRnJzpfTorotvu7uxOHoej3j3N/L7ZWdZZZo7d26d26wbfv6/v27SFjR
ooapZajlmn8fr8b96sqVOnfuf+XnrpJX355355Zf68Y9/HIQqAQCInAG9MuJySe++X2dTdo3YMR1PXVWLtI1qMDK1fvv147d+5dW
bxo8frw8//FBr167Vq7O7Vt2zbccdrdv3zHv/fXAX5bXXI8aq5uVnvv/++6Xt6NnVtyN4xH/XNb6P6aEZRoxHAQCgcCIgeIQeH9Nyu
w4ePKgdO3Z3O3Zo3LhxSdhdrOdHh+HpF8uosrzHTyANQVbqQGdBK4uAZuhXdcLabGpqfs3W9nvd2Nu+8YWt
QAAFh
FGAEAAFYRRgAAgFX9JozEwE1BpzmzEABgQ8yHEdfEgP/33GPTcWIF/c78
OcrK

AAH2uX4QRqeuveh5hDgBA7OHPYAAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABg1QmFkcWLFysjIoMJCQnKzs7Whgo
beuw7ZcoUO
RyOY7Zrr732hIsGAAD9R9hhpLq6WqWlpSovL9fGjRs1ceJE5efna8+ePd32r6mpUUtLS3DbvHmzXC6X/vEf//GkiwcAALEv7DD
yzDPPaM6cOSopKdGECROoZMkSJSYmavny5d32P/3oo5WSkhLc/ud//keJiYmEEQAAICnMMNLZ2amGhgbl5eUdeQOnU3l5e
aqvr
+/Veyxbtkw/+clPdNppp/XYp6OjKzTX6/X6fL8nJySHtycnJ8nq93zl+w4YN2rx5s2699dbj9quoqJDb7Q5
uHo8nnDIBAEAM6dO7aZYtW6YLL7xQWVlZx+23aNEitbe3B7empqY+qhAAAPS1sL6bZuTIkXK5XGptbQ1pb21tVUpKynHHH
jhwQ
CtWrNDDDz/8nfuJj49Xfhx8OKUBAIAYFdbKSFxcnCZNmqTa2tpgWyAQUG1trXXXXlNHR4duuummmE6sUAADoS2
F/a29
paamKi4s1efJkZWVlqbKyUgcOHFBJSYkkqaioSGlpaaqoqAgZt2zZMk2fPl3f+973IlM5AADoF8IOI4WFhdq7d6/Kysrk9XqVm
ZmpNWvWBC9qbWxslNZuuDy8ccf65133tHvf//7yFQNAAD6DYcxxtgu4rv4fD653W61t7crKSnJdjkAAKAXevv5zXfTAAAAq
wg
jAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAq
wgjAADAK
sIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAAD
AKsIIAAC
wijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAADAKsIIA
ACwijACA
ACsOqEwsnjxYmVkZCghIUHZ2dnasGHDcfvv27dPd955p1JTUxUfH69zzjlHb7zxxgkVDAAA+pdB4Q6orq5WaWmplixZouzsb
FV
WVio/P18ff/yxRo8efUz/zs5O/fCHP9Too0aO+fHgk6gcAADHOYYWx4QzIzs7WJZdcoueeeeeo6SFAgE5PF4d
Pfdd2vhwoXH9F+yZImeeuopbdu2TYMHDz6hIno+n9xut9rb25WUlHRC7wEAAPpWbz+/wzpN09n9nZXl5R15T
qq+ v73bMf/3XfyknJod33nmnkpOT9f3vf1+PPfaY/H5OLsGAAD9VFinadra2uT3+5WcnBzSnpycrG3btnU75q9//avefj
XrjjTf06aefau7cuTpo6JDKy8Hu7HdPRoaGOjo7gzz6fL5wyAQBADIn63TSBQEC jR4/Wv/7rv2rSpEkqLCzU/fffryVLvQ4pqK
iQm63O7h5PJ5olwkAACwJK4yMHDlSLpdLra2tIe2traı KSUnpdkxqaqrOOeccuVyuYNv5558vr9erzs7Obscs cWrRI7e3twa2pq
SmcMgEAQAwJK4zExcVpoqRJqq2tDbYFAgHV1tYqJyen2zE/+MEP9OmnnyoQCATbPvnkE6WmpiouLq7bMfHx8UpKSgrZAA
BA/xT
2aZrSolItXbpUL73okrZu3a077rhDBw4cUElJiSSpqKhIixYtCva/44479MUX2jevHn65JNPtHr1aj322GO68847I3cUAAAgZ
0X9nJHCwkLt3btXZWVl8nq9ynysM1Jo1a4IXtTY2NsrpPJJxPB6P1q5dq5///Oe66KKLlJaWpnnz5mnBggWROwwoAABCzwn7OiA
o
8ZwQAgNgTleeMAAAARBphBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRR
gAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFW
EEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAY
BVhBAAAWEUYAQAAVhFGAACAVYQRAABg1QmFkcWLFysjIoMJCQnKzs7Whgobeuz74osvyuFwhGwJCQknXDAAAOhfwg
4j1dXVKio
tVXl5uTZu3KiJEyYqPz9fe/bs6XFMUlKSWlpagtvOnTtPuUGAAANB/hB1GnnnmmGc2ZMocl SWaMGGClixZosTERC1fvrzHMQ6HQykpKcEtOTn5pIoGAAD9R1hhpLOzUw0NDcrLyzvyBk6n8vLyVF9f3+O4/fv368wzz5THM9O4r776SmeccYY8PMs8oujkwYkeTgjQYpzRcvWyhW8oLqXlYc45mSgkNIS+rIdOz47jS
ykpKcEtOTn5pIoGAAD9R1hhpLOzUwoNDcrLyzvyBk6n8vLyVF9f3+O4r776SmeccYY8Ho+mTZumP//5z8ftdHh3w+X8gG
AAD
6p7DCSFtbm/x+/zErG8nJyfJ6vd2OOffcc7V8+XK9/vrr+u1vf6tAIKApU1Pe6noqJCbrc7uHk8nnDKBAAAMSTqd9Pk5
OSoqKhImZmZuuKK1RTU6NRoobp17/+dY9jFi1apPb29uDW1NQU7TIBAIAlg8LpPHLkSLlcLrW2toaot7a2KiUlpVfvMXjwYF1
88cX69NNPe+wTHx+v+Pj4c4EoDAAAxKqyVkbi4NEm1tbXAtkAgoNraWuWkZ5PTqfgU1NTwKgUAAP1SWCsjkl
RaW qri4mJNnjxZWVlZqqys1IEDB1RSUiJJKioqUlpamioqKiRJDz/8sC699FsFKdffbZ2rdvn5566int2rVLt956a2SPBAAAxKSwwoh
hYaH27t2rsrIyeb1eZWZmas2aNcGLWhsbG+VoHllw+fLLLLzVnzhx5vV6NGDFCkyZNonvvvacjEyZE7igAAEDMchhjjOoivovP5
5Pb7VZ7e7uSkpJsGxkpJslwMAAHqht5/ffDcNAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCC
AAAsIo
wAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAA

sIowAgAAr
CKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAAsIowAg
AArCKMAAA
AqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALDqhMLI4sWLlZGRoYSEBGVnZ2vDhg29GrdixQo5HA5Nnz
79RHYLA
AD6obDDSHV1tUpLS1VeXq6NGzdq4sSJys/P1549e447bufOnbr33nuVm5t7wsUCAID+J+ww8swzz2jOnDkqKSnRhAkTtGTJEi
U
mJmr58uU9jvH7/brxxhv1oEMP6cwzzzypggEAQP8SVhjp7OxUQoOD8vLyjryBo6m8vDzV19f3OO7VV1f3OO7hhx/W6NGjdcstt/RqPxod
H fL5fCEbAADon8IKI21tbfL7/UpOTg5pTo5Oltfr7XbMO++8o2XLlmnpoqW93k9FRYXcbndw83k9FRYXcbndw83g84ZQJAABiSFTvptm/f79uvvl
mLV26VCNHjuz1uEWLFqm9vT24NTU1RbFKAABg06BwO08cOVIuloutraoh7a2trUpJSTmm//bt27Vz505o5dd911wbZAINC14oG
D9
PHHH+uss846Zlx8fLzi4+PDKQ0AAMSosFZG4uLiNGnSJNXW1gbbbA0oGAamtrlZOTcoz/8847T5s2bdJHH3oU3K6//npNnTpV
H33
oEadfAABAeCsjklRaWqri4mJNnjxZWVlZqqys1IEDB1RSUiJJKioqUlpamioqKpSQkDvf//7IeOHDx8uSceoAwCAgSnsMFJYW
Ki9e/eqrKxMXq9XmZmZWrNmTfCi1sbGRjmdPgVAADojsMYY2wX8V18Pp/cbrfa29uVlJRkuxwAANALvf38gZkDAABYRRgB
AAB
WEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAA
BYRRgBAABWEUYAA
IBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBA
ABWEUYAAIBVhBE
AAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEU
YAAIBVhBEAAGAVY
QQAAFh1QmFk8eLFysjIUEJCgrKzs7Vhw4Ye+9bU1Gjy5MkaPny4TjvtNGVmZurll18+4YIBAED/EnYYqa6uVmtqqrKzs7TjffffnfTz/9dN1///2qr6/Xn/7oJ5WUlKikpERr16496eIBAEDscxhjTDgDsrOzdckll+i5556TJAUCCAXk8Ht199iauHB1rA7ZyeMmdJt//j4eMXHx4dTGgAAiFFhrYzExcVpoqRJq2tDbYFAgH
V1
tYqJyen1+8TCATUodERzq4BAEA/FdbKiCSVlpaquLhYkydPVlZWliorK3XgwAGVlJRIkoqKipSWlqaKigpJXdd/TJ48WWeddZY
6Ojroxhtv6OWXX9YLL7wQ2SMBAAAxKewwUlhYqL1796qsrExer1eZmZlas2ZN8KLWxsZGOZ1Hluwd2ZGdZ0HFlwOHDiguXPnqrm5WUO
GDNF55
52n3/72tyosLIzcUQAAgJgV9nNGbOA5IwAAxJ6oPGcEAAAgoggjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALC
KMAI
AAKwijAAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsGmS7AAADmN8vrVsntbRIqalSbq7k
ctmuC
kAfI4wAsKOmRp03T2puPtKWni5VVUkFBfbqAtDnOEoDoO/V1EgzZ0QGEUnavburvabGTloArCCMAOhbfn/Xiogxx752uG3+
/K5
+AAYEwgiAvrVu3bErIkczRmpq6uoHYEAgjADoWyotkeoHIOYRRgDordTUyPYDEPMIIwD6Vm5u110ozDkf3rzscksfT1Q/AgEAY
A
dC3XK6u23elYwPJ4Z8rK3neCDCAEEYA9L2CAmnVKiktLbQ9Pb2rneeMAAMKDzoDYEdBgTRtGk9gBUAYAWCRyyVNmWK7
CgCWcZo
GAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQ
RAABg1QmFkcWLF
ysjIoMJCQnKzs7Whgobeuy7dOlS5ebmasSIERoxYoTy8vKO2x8AAAwsYYeR6upqlZaWqry8XBs3btTEiROVn5+vPXv2dNu/rq5
Os2bNoltvvaX6+np5PB5dffXXV2r179okXDwAAYp/DGGCPCdna1LLrlEzz33nCQpEAjI4/Ho7/rvvvN/vi4gRI/Tcc

8+pqKioV/vo+Xxyu91qb29XUlJSOOUCAABLevv5HdbKSGdnpxoaGpSXl3fkDZxO5eXlqb6+vlfv8fXXX+vQ0UM6/fTTe+zTodE
hn88XsgEAgP4prDDS1tYmv9+v5OTkkPbk5GR5vd5evceCBQs0ZsyYkEDzbRUVFXK73cHN4/GEUyYAAIghfX03zeOPP64VK1boP
/7jP5SQkNBjvoWLFqm9vT24NTU19WGVAACgLwoKp/PIkSPlcrnU2toaot7a2qqUlJTjjn366af1+OOP6w9/+IMuuuii4/aNj49
XfHx8OKUBAIAYFdbKSFxcnCZNmqqTa2tpgWyAQUG1trXJycn0c9+STT+qRRx7RmjVrNHNHny5BOvFkC/4vdLdXXSq692/ev3264Ig
A1hrYxIUmlpqYqLizV58mRlZWWpsrJSBw4cUElJiSSpqKhIaWlpqqiokCQ98cQTKisroyuvvKKMjIzgstVHny5BOvFkC/4vdLdXXSq692/ev3264Ig
JTYoob57U3HykLT1dqqqSCgrs1QWg74UdRg0LC7V3716VlZXJ6/UqMzNTa9asCV7U2tjYKKfzyILLCy+8oM7OTs2YMSPkfcrLy
/Uv//IvJ1c9gJhUUyPNmCF9+8ECu3d3ta9aRSABBpKwnzNiA88ZAfoPv1/KyAhdETmaw9G1QrJjh+Ry9WlpACIsKs8ZAYCTtW5
dz0FE6lotaWrq6gdgYCCMAOhTLS2R7Qcg9hFGAPSp1NTI9gMQ+wgjAPpUbm7XNSEOR/evOxySx9PVD8DAQBgBoKdcrq7bd6VjA
8nhnysruXgVGEgIIwD6XEFB1+27aWmh7enp3NYLDERhP2cEACKh0ECaNq3rrpmWlq5rHRJzWREBBiLCCABrXC5pyhTbVQCwjdM
0AADAKsIIACwijACACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwKqYeAKrMUaS5PP5LFcCAAB66/Dn9uP8
Z7ERBjZv3+/JMnj8ViuBAAAhGv//1vyu9o9vu4w3xVXTgBQEC4A4tmBuxzOvZ+Pc52s3cqm5+9mlr0cSfNeOxwzAxq4h+XGW4rDoirs8J5pOZ
9g3nuG8xzKGOM9u/frzFjxsjsp7PnKkJhYGXE6nUpPT7ddxikl2SmJX/Q+wlz3Dea5bzDPfYN5PuJ4KyKHcQErAACCwijCsI
0zEmPj4eJWXlys+Pt52fcec9o3mOe+wTz3Deb5xMTEBawAAKD/YmUEAAYRRgBAABWEUYAAIBVhBEAAGAVYeQU9MUXX+jGG2
UUlKShg8frltuuUVfffXVccccPHhQd955p773ve9p6NCh+vGPf6zW1tu+37++edKT0+Xw+HQvn37onAEs8zbNk
sfj0ZAhQ3T++eerqq0o2odySlm8eLEyWMjJKUkJCg7Oxobxdiw4bj9X3vtNY0dOOB
vL09/+ctfonkIMSGS83roCEtWLBAF+154oU/TR7vvtGnGTNR9GRnud9+aZvZ4apjjMEp55prrzGWXXWZeffdd82f/vQn87OOOlKsXEU
o4077//vlm3bp05zzZax4Zs4475vbbbbcej8fU1NSX6+TNR/9VqW8jvvrZZKkRVWRXRVF+uyzz6aYrrjETJ
PfvYzU1dXZ7Zv325efllM2TIEPPss89G9SYOGt58uGltbe22/Gb+SsOGt58uGltbe22/7vvvvmtcLpd58sknzZ
8wDDzxgBg8ebDZt2hTs8/jjj5uOuvmSWzpnS9/xsnTcqamSuq4bwn7pmTRpkx4Ba9gnGzSDDJn56xkhX2rfoq7ruW5l2f
/hwn8gBzHCu/PSE17+NhS+vXZ+xXdkyr9sa2bdbLzovXJfd+BH9/qLCPAoX/n3Z9tUnI+EQV4nVW7DCK9QVTJJBtcKdyXf
oS7MrXQa9vYfkCM15X3/njQDXcbhH0hSC+uzuHOODdMYKgUuOQGbcpHjNiAs0dHW7B5EEc+v7tYfx1mG3Ypkx6/+W3FzZo4DqJnXhx/Ce
jXQ/g6aAUz4BOtqL5ypnOOKo/2nWVWtaBcFxPw4TVtlcVRrLlq4QS2fVQlbWVqS7gxf0czaU1YwCjQ/YHmA+45ozpKjJqTM5aVVnU0Bdz
Hx8V+sAQ5sVm2hOqJ+4Mrz6fWUe6JVHbHvzJCqPM6wZvZTWCzJl7ZC/1rtTqaCFYZF+PbFEzUaxWrHyLgXJBuSOVRcNMSlCbBFQIrK
RvMhpJgj6m5gm1bPnR4GodwdpQHq0XTe9+jNBnV/gZWSaJiXXwbzNK+OI9Zz8I0JNfVc5wSYTn5c3Zs+kDpZqNKJi7OXXaWdJlWWwBDe
VeDBB1SGxXZcw4CdVL2kdXHNAP9lMdH2NL3ZSRKyM4g4NPHbkkTxFFOh/oFlXNHuGQJN0MLOdB3V8KK79Gt3XTn5oZYRr0yAt4SJbbYbc
Lmj9AG3+3uS8hHc94rN4SB+5HmCnlBNzaXLklXPfHmWYoGPMqMJF4VPZe1UnMQoczVu0PR3SbpAVd7b1ThXqRQ4IprdApEYGvrFRTbSc
R6PpWsoUNUJN6g1WWq1CKTtr+Hcbj4Sm5d4zNhF4O7QK3kmsWqgLmC9rQ2uhF6uHJiVPOmM9G9CRapWOKpxJEp4QY5HOL4a0DNg+CVdkU
sJNQYTdhGvYeGVAwZGJHRhXcB++v2YK8QZzLB3dz/Z//88Pj8DSR8Q6uBwTfoFpa4w6fuNjnsvdvBR8hGk6+M04f66k7ZH9jPoJgNiz
8Qk1a6GtGKk/jhtPYNcc5Aw6YzjbQw7H4IA2IUo0GNOFmY5xY+kXfn/6gPqDXt3ZqvOFFdj/7WF3QqSrzfm3ShSJMRrqsm+QpiBj1cxTqF
fLM4lPu1vFdAFLXWrNkbeyYCqR28EbwbN/7eT4rrNOwPYiBx0FoiA+FZmZYXW2UcNQWj9Rq1yxX+ZOwlaV1gqeiDtzxG3mfA71lEHM4z0tXiAe
sweyPQ9zs5eJLR8KWTlXaALiMGgmWWwF2dXEMN0k/kBOfBCcJMajZC2sTC/5lmc/7vRPoXFmwsDGaVU+3YWMv75e4uGXmBaBAX+ZoEsSb
wnMWcgRAYMgHtA8EQdmoqKVtpHL0P7bcpAYVUtIt2Vr+m9GvVU1F3lc+e3nQbW3ogwcM2kiOzeqvbdngaEhaMHFBbSzcjn7QrKM7sXr
oCF1KzIYIqvEyUBzNqGNnwYJU4v7mj6d1BFq+ZRqT3SM8Dg9O5RsQmn9AQypQxCKBaw5RbL7oiUqaMw97Kbw4DxMYI53UoxrPgpkQ6HNPsD
GIn6zgeIkcaBrbZjbTu6C6zhMXEfY9RZbH6BHVyKZVXt4AyvTCZWa7HUh/F5Ezfh6dLXe2h1vcJfhmnE9Ruz9b4qE4OaH6VEVpQTkcv99p8
Ps7Y3ek2L0fuEjjNhyYSv7yTSDQ0WuYwFw0RUVzF3/Z0aBDfs3aCQfHJuXNHJ39rHYLBjfbmuDY0fkFTnnDwUwKuJZk8zmPb6lXlQa7kR
LBiMqQrWMbF1jNHcXGRVmOJyQ6FFa3VJcFMNPdyNLzmJH+7u3UqF4aVZ1TPyT3ljmxhLmcUjUzTfWWKQeMkYVtCVbVVpvtnFsHcQ/DxNqE
SNNKKS4ZD28mqxKfYiJUtCFCQ9Y5rH2gJZaIxgqyKcXFNs5hPZXZjvhS9EEu8PsY32uaBGvU0Mb3cQf5Uj2FNTcq3eXS5kuUt4G5cwGxeI
AtPN0/N4PnmRFNZxZIpmuWfMpxDCtqrwYN6dCt/yI/XkG9Vk8q+U2dsR4fT0rsWB5Gk4n8ptkzfOp6Wc0AwP42jzxX2IESXwuU6DfjD5RY
cFG7aXKT+UGSwvnqV9XOo2Yh6dRpVYRfGv/xb8f0Ev0HVaNlhT3B/SmEpUGTxHybWM6o+JgUDEXUC+vj26kWs6eQ7UWeVJkMIB7nYkDkZhxw
+Xfj1MxJYmn0pOWnOWtPcwe/f+aOhs/t53r/90oPhhvaHsBHjhbQyNnvRm9iRaA3E9mKHO6G7gHEEuN9UlNmFsM+DLfpNYHuNVpHVw+bHAZY

```
"#plt.xlabel('Kc, tauI, tauD');\n",
"#plt.legend((result_Kc, result_tauI, result_tauD), ('Kc', 'tauI', 'tauD'))\n",     "\n",
"plt.legend(loc='upper left')\n",
"#pylab.ylim(-1.5, 2.0)\n",
    "plt.show()"
```

```
berikut\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 57,
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import matplotlib.pyplot as plt\n",
    "from scipy.integrate import odeint\n",
    "import ipywidgets as wg\n",
    "from IPython.display import display"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 58,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "application/vnd.jupyter.widget-view+json": {
       "model_id": "e9a2f7a951b94ddfb51842e4dafee081",
       "version_major": 2,
       "version_minor": 0
      },
      "text/plain": [
       "interactive(children=(FloatSlider(value=0.24453239142894745, description='Kc', max=0.7335971742868423, min=-0.…"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    },
    {
     "data": {
      "text/plain": [
       "<function __main__.pidPlot(Kc, tauI, tauD)>"
      ]
     },
     "execution_count": 58,
     "metadata": {},
     "output_type": "execute_result"
    }
```

```
  ],
  "source": [
   "n = 100 # time points to plot\n",
   "tf = 50.0 # final time\n",
   "SP_start = 2.0 # time of set point change\n",
   "\n",
   "def process(y,t,u):\n",
   "    Kp = 4.0\n",
```

[redacted]

[redacted]

```
    "\n",
    "def pidPlot(Kc,tauI,tauD):\n",
    "    t = np.linspace(0,tf,n) # create time vector\n",
    "    P= np.zeros(n)          # initialize proportional term\n",
    "    I = np.zeros(n)         # initialize integral term\n",
    "    D = np.zeros(n)         # initialize derivative term\n",
    "    e = np.zeros(n)         # initialize error\n",
    "    OP = np.zeros(n)        # initialize controller output\n",
    "    PV = np.zeros(n)        # initialize process variable\n",
    "    SP = np.zeros(n)        # initialize setpoint\n",
    "    SP_step = int(SP_start/(tf/(n-1))+1) # setpoint start\n",
    "    SP[0:SP_step] = 0.0     # define setpoint\n",
    "    SP[SP_step:n] = 4.0     # step up\n",
    "    y0 = 0.0                # initial condition\n",
    "    # loop through all time steps\n",
    "    for i in range(1,n):\n",
    "        # simulate process for one time step\n",
    "        ts = [t[i-1],t[i]]         # time interval\n",
    "        y = odeint(process,y0,ts,args=(OP[i-1],))  # compute next step\n",
    "        y0 = y[1]                  # record new initial condition\n",
    "        # calculate new OP with PID\n",
    "        PV[i] = y[1]               # record PV\n",
    "        e[i] = SP[i] - PV[i]       # calculate error = SP - PV\n",
    "        dt = t[i] - t[i-1]         # calculate time step\n",
    "        P[i] = Kc * e[i]           # calculate proportional term\n",
    "        I[i] = I[i-1] + (Kc/tauI) * e[i] * dt  # calculate integral term\n",
    "        D[i] = -Kc * tauD * (PV[i]-PV[i-1])/dt # calculate derivative term\n",
    "        OP[i] = P[i] + I[i] + D[i] # calculate new controller output\n",     "        \n",
    "    # plot PID response\n",
    "    plt.figure(1,figsize=(15,7))\n",
    "    plt.subplot(2,2,1)\n",
    "    plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')\n",
    "    plt.plot(t,PV,'r:',linewidth=2,label='Process Variable (PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,2)\n",
    "    plt.plot(t,P,'g.-',linewidth=2,label=r'Proportional = $K_c \\; e(t)$')\n",
    "    plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\\frac{K_c}{\\tau_I} \\int_{i=0}^{n_t} e(t) \\; dt $')\n",
    "    plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c \\tau_D \\frac{d(PV)}{dt}$')
\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,3)\n",
    "    plt.plot(t,e,'m--',linewidth=2,label='Error (e=SP-PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,4)\n",
    "    plt.plot(t,OP,'b--',linewidth=2,label='Controller Output (OP)')\n",
    "    plt.legend(loc='best')\n",
```

```
"    plt.xlabel('time')\n",
"    \n",
"Kc_slide = result_Kc\n",
"tauI_slide = result_tauI\n",
"tauD_slide = result_tauD\n",
    "wg.interact(pidPlot, Kc=Kc_slide, tauI=tauI_slide, tauD=tauD_slide)"
```

```
   "Pengujian ke-2"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 59,
  "metadata": {},
  "outputs": [],
  "source": [
   "ujicoba2 = np.array([\n",
   "    [0.4, 1.2]\n",
   "])"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 60,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array([[0.4, 1.2]])"
     ]
    },
    "execution_count": 60,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "ujicoba2"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 61,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
```

```
    "output_type": "stream",
    "text": [
     "1/1 [==============================] - 0s 16ms/step\n"
    ]
   }
  ],
  "source": [
   "outDL = model.predict(ujicoba2)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 62,
```

```
     ]
    },
    "execution_count": 62,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "outDL"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 63,
  "metadata": {},
  "outputs": [],
  "source": [
   "result_Kc   = outDL[0,0]\n",
   "result_tauI = outDL[0,1]\n",
   "result_tauD = outDL[0,2]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 64,
  "metadata": {},
  "outputs": [
   {
    "data": {
```

```
    "text/plain": [
     "0.25178796"
    ]
   },
   "execution_count": 64,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "result_Kc"
 ]
},
{
 "cell_type": "code",
 "execution_count": 65,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "0.82828206"
    ]
   },
   "execution_count": 65,
   "metadata": {},
```

},
{
"cell_type": "code",
"execution_count": 66,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0.18918379"
]
},
"execution_count": 66,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"result_tauD"
]
},
{
"cell_type": "code",
"execution_count": 67,
"metadata": {},
"outputs": [
{
"data": {   "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAiMAAAGdCAYAAADAAnMpAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLj
iwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8g+/7EAAAACXBIWXMAAA9hAAAPYQGoP6dpAAAr30lEQVR4nO3df3RU9Z3/8ddkIBO
iJKBAEpLB0CKKP4gLEmPNV9DUuOta2JQlxR8JKcIC2aqHRLqQKWXVr/FWbHEVxWThaeypBNrvuWSy4jcaDGkXDoYUCWimQBDO
GQQJaEzn+8fWUZGEszATD6Z5Pk4554wn/l87n3fe3KYVz73xxziMMUYAAACWxNguAAAA9G+EEQAAYBVhBAAAWEUYAQAAVhFGA
AVYQRAABgFWEEAABYRRgBAABWDbBdQHf4/X59+umnGjx4sBwBwhjdPjwYYocVIxMV3PfoRFGPnooo/ldrttlwEAA
E5DQ0ODotLSunw/KsLI4MGDJXXsTEJCguvqABAd3i9Xrnd7sDneFeiuowPzTkJBAGAEAIMp81yUWXMAKAACsIowAAACrCCM
AAMCqqLhmpDt8ParP+OHTmu4yoNnDgQDmdTttlAD6mT4R8kssNcCGEZGYZams88l
+Lj4zV8+HAeinaajDE6ePCgGhsbNWbMGGZIAA9JurDyLFjx2SMofDhwzVooCDb5US14cOHa+/evTp27BhhBADQY/rMBazj
5jiEAwIa0nxkBEL18fp821W9S0+EmpUD/hvCCAAArobDljmJUD+hvCCAArqnZWaeygZyYYBiRpKoT1dmjJUD+hvCCAArqnZWaeGGhWr0NgbaohLKT1dmjJFuvXWjp/p6R3tETJr1ixNmzYtG3
dunWKi4vTL3/5y4YaFMjr5VvzjbYs2LJLPzx8EQH9BGKmqkqZPlxqD/oLT/vod7REMJCf693//d9222kw4cPSxa
O+99/bINgEbNtVvOmlG5ERGRg3Bm2q39SDSVQGwqX+HEZ9PWWcFFi/XFcc4/XfcfO6e/0Sx+MJBJIxY8Zoo1KBBq6uj
uCy+8UC6XS6NGGjdIvfvGLiNYBB0vcXXv1Sm0EsSjm0hi7xY3+p7Xgw4Ad5SmUu4G1gn7GYNqtBcDnpr5wQuU83hsh5jLNXfGt5vyFpwrOnQc/ovJ0eAX
otO3x1k9gnEptcpM+9hhd7HFOdRR04QxmHqxm+/Jx4HEQn0Bt5hE/nVoDo77LTBN6K
PsNQSdr4CU/LxT5R5gS5bTE1+/r8yyoZnMhN6pIDR3mwoW+W9FkMbAcZNixe6t0Kd4PV/8vRyxXo1tvmG1Nu8LMrAdWu
PnUUwJMSpSNmsTy4zoZO+c3Qns5TtL4tLPTpBA1kDU1FCHAbtXbvAAAmX7cfctC9X3TZApKOKzNxf1C5a+yt
ZxbTWNkxr5MOTTDZmYJZ4MPjtP3o3ByEgCWSHWUNpQ1n4zG5awPQZ8H43+WkrD8LvtiXaT
+9bj19PFIHJ2E9TzUxbEKJAi0j5uJOHOkDeWW6W1hbZsNWHm85ILTaNuO+OMeDzvCeJyfG1JEwZKyzPK25xGcbP7cH3vTFDnKiL0v7Lh2d
/rYPfjpLE7gqtuqNkiiLgPBlQN55i1xjTNdQ0y8YDwlNmhsJ8L4dxPpC0nS7UI2Mj0lTjbd5yO0aPvl7wi
TO0q3fdc7Xj0i4t3jl1p66GwwoRoIhkBIK5
+Zh6JR8Rwu6TsBxVLwTgeM2anaZobsCtAqiXJtNmwT7J/MTE0EfOUmHH9Li3ajQd33fAnjvM1E4I0oOEkhqBYmJ6XmQ
qLtjVIfptVp7ewyrXtKp8VuAsXbcWdvjpvXGO20y4fWAuSGwsyYKpWrG/E9vPYq6vPiDSLVUTHFtJ8tPjvVf
kLi6TUrZqTrJkKiaShPcm6pJmnbhFvl4oqwx1M3WcWnHdjVLR4T+FcbvvqJXEo90TVTP8Db3qeD6N0cc/VcrAy5q
AJCi0SXbgM+AYdumf6/Ql3QmYzZYyZPdaGmWvqEXx4FfEb4hd+0YL4DXXzZCUCtmLhO79LUhKB3ZjDh7d1E
PQkFv9tSwDD7Ey+gP1I4RV/+pi/MoKjzOyk5bhHoNKdmGf5ZI/p5aK7mDBUWzNEXR9KX0wZtDYELVKxVkAhIDEq+5rt
Z3JlXNvIwZqdaPBvBMPJQ3tQy2yDD/kk6cjOW4L2dFnAjj0vnuRl80rxF0lVyYmHoYTi9VPNOPM1I81Yu5k1+u0g3X
Gh4ZJGvfexKrm6FzF7/76K+u0Dqx9VbBEPHDZFjtBDzJ+Cs5Qot0hO+FSfWixBBQDW7gMN29uTU2LOvNGJLHNPDqOmQ
sO+g3tEL8KgO1W4vQTqTBzdsdyGMyw1ZjhLo51Uw44Sr4LzQGlIdYpUAmMT30AAbjNNdjr9n83H6KuEDBqcojUl3OaqOxqVnvTnv2u03v
K/Iu3YCkpEYm9e7v+v2ssTnVqXk/0o00NQ0HDXK1UB3djvpjwT

tt96qlpYWXXvttTLG6K9//avmzZt3ytMoZWVlevDBBoMpDQAARKmI301TU1OjRx55RM8++6y2bNmiqqoqrV+/Xg8//HCXYopKS
tTa2hpYGhoaIlomAACwJKSZkWHDhsnpdKq5uTmovbm5WcnJyZ2OWbpoqe644w7deeedkqTLL79cR44cody5c3X//fcrJubkPOR

ydroaJFUbNeAAAiJeTvpikuLlZhYaEmTpyoSZMmqby8XEeOHFFRUZEkqaCgQKmpqSorK5Mk3XLLLXrqqado5ZVXKjMzU59
88om
WLl2qW265JRBKbKvaWaWFGxaqodsYaEtLSFPFTRV8gygAABEW8jUj+fn5evLJJ7Vs2TJlZGRo69at2rBhQ+Ci1vr6ejU1NQX6P
/DAA7r33nv1wAMPaNy4cZo9e7Zyc3P1/PPPh28vzkDVzipNXzs9KIhIon7vfk1fO11VO6vCvs1Zs2bprbfeUkVFhRwOhxwOh3b
v3q3Zs2dr9OjRGjRokMaOHauKioqgcZ3NekybNk2zZs0Ke4oAAPSU0/rW3rvvvvlt33313p+/V1NQEb2DAAJWWlqqotPRoNhVR
P r9PCzcslNHJp4uMjBxyaNGGRZo6dmpYv9q8oqJCH3/8sS677DI99NBDkqShQ4cqLS1Nr7zyis4991y9++67mjt3rlJSUjRjxoy
wbRsAgN7mtMJIX7GpftNJMyInMjJq8DZoU/omTU6fHLbtJiYmKjY2VvHx8UEX/p54O/PooaNVW1urtWvXEkYAAH1avw4jTY
ebv rtTCP3O1PLly7V69WrV19fr66+/Vnt7uzIyMnpk2wAA2NKvv7U3ZXBKWPudiTVr1ui+++7T7Nmz9frrr2vr1q0oqKipSe3t7oE9
MTMxJdyAdO3Ys4rUBABBJ/TqMZI/KVlpCmhxydPq+Qw65E9zKHpUd9m3HxsbK5/vm9uF33nlH11xzjRYsWKArr7xSF154oXb
v3 hooZvjw4UEXB/t8Pm3fvj3stQEAoJP6dRhxxjhVcVPHHSvfDiTHX5ffVB7Wi1ePS09P1/vvv6+9e/eqpaVFY8aMoYcffqiNGzf
q448/1tKlS/XBBx8Ejbn++uu1fv16rV+/Xrt27dL8+fN16NChsNcGAEBP6tdhRJLyLsnTuhnrlJqQGtSelpCmdTPWRew5I/fdd
5+cTqfGjRun4cOHKzc3V3l5ecrPz1dmZqY+++wzLViwIGjMj3/8YxUWFqqgoEDXXXedzj//8FE2ZMiUi9QEAoFMcpjc9BrULXq9
XiYmJam1tVUJCQtB7R48e1Z49ezR69GjFxcWd9jZ8fp/KastUf6hei+ewNaAIIjMiPRm4TqWAABIp/78PlG/vpvmRM4YZ1hv
3
wUAAN3T7o/TAAAAuwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsII//H55NqaqSX
X+74ecI
X6kbE5MmTtWjRoois1+FwyOFwyOVyKTU1VbfccouqqqrCvioAAMKBMCKpqkpKT5emTJFuvbXjZ3p6R3somjNnjpqamrR792
79x
3/8h8aNG6cf/ehHmjt3ru3SAAA4Sb8PI1VVovTpUmNjcPv+/R3tkQgks2bNoltvvaWKiorALMbu3bs1e/ZsjR49WoMGDdLYsWN
VUVERNK6z2ZRpo6Zp1qxZQW3x8fFKTk5WWlqarr76aj322GN6/vnntXLlSv3+978P/w4BAHAG+nUY8fmkhQulzr63+HjbokX
hP
2VTUVGhrKyswAxGU1OTotLSlJaWpldeeUU7duzQsmXL9POf/1xr164NyzYLCws1dOhQTtcAHqdfv2tvZs2nTwjciJjpIaGjn6
TJ4dvu4mJiYqNjQ3MYBz34IMPBv49evRo1dbWau3atZoxY8Y8ZbzMmJkYXXXXSR9u7de8brAgAgnPp1GGlqCm+/M7V8+XKtXr1a
9 fX1+vrrr9Xe3q6MjIywd8YI4fDEbb1AQAQDv36NE1KSnj7nYk1a9bovvvvuo+zZs/X6669r69atKioqUnt7e6BPTEyMzLfOKRo
7dqxb6/f5fPrzn/+soaNHh7VuAADOVL+eGcnOltLSOi5e7ey6EYejY312dmTrKCsrU1ZWlu6+++5A2+7du89omy+99JLWrVtt
T58uJpOmKbx+Xzavn27pkyZ8p3be/HFF/XFF1/ohz/8YRiqBwAgfPr1zIjTKf3mNx3/7ky/7x9//fbbO3TrlqIv1rE4i8+eeNG
oaMyYMfrwww1cePNGffzx0q6dKk++OCD0o3+91q9fr/Xr12vXrl2aP3++Dho6aP3++Dh06dNK6v/rqK3k8HjU2Nuq9997T4sWLNW/e
P
M2fP79bwQUAgJ7Ur8OIJOXlSevWSameWXmR2e5999onp9Opce7GafjwcrcNr4rNzVVeXp79/OVmZmpz77LGgiWRJJ+/OM
fq7CwUAUFBbruuuto/vnndxouVq5cqZSUFF1wwQXKy8vTjh07VFlZqWefTYYyOwMAwBlwmG9fhNALeb1eJSYmqrW1VQkJCU
HvH
T16VHv27NHooaMVFxd32tvw+Trummlq6rhGbGuGjiUVm4jiUAANKpP79P1K+vGTmRoxne23cBAED39PvTNAAAwC7CCA
AAsIo
wAgAArCKMAAAAq/pMGImCm4J6PY4hAMCGqA8jzv+bv+PYmAsEzKHjuhqGx4WY5vq3iKwyayXgc4yb3QEgvJM0hOrNddzuQrGQWs
ppNmUqZ6SmQjbDx8m6N/aW0vn+zQjxcMcGFXKT49GaTzGtD6RWzWaYy8NKD5eAKk7Xydz+lULWrwoPD6n+b5wx+mAkCvPgcyQdsXaLKq9IvRp9Vt/0bVgFxXxFqOeL/rCtmdjSAhpVywt6+n13NdHKhEQrPixjVQKHJbfPydeXgb4Ma8VVgsUqVMrNvmrlY6Z8RCKr8y8N2LHMwbfGf7qcPcsZFlUnNWpmlHdZI0GdXGoZbLrgyjs5egKh3l7+LL75JpxSZ2D6lqPDI6+IhGgo3ZlmBVFtHjfd9OX+1ePvgSp0oZ8P0I7JFr2MDeKmlNdJRDd3jf8+Nvve3jtapbt5NvTfWwQvEE8zmaxdtVrbVEn9OfPOhgQBCYVvtoEO4i84a4Nn4UPo+dPKd9u8PMVDm+f2W8UP3WcYP2Ulo5PG8q0OU8RDHDRYyBJtBLKQ2N/ySStCvBt23D9Q8cfRwdZtqOKR7uWjSAbyTNZThPa22GhbBCEcdCZw5QDu2rgSfkXM7KJeB+IqBQEIF19kSe1eVaVKpN0xpNdMFJLyZRQ3AjaB1p8bPrYL4BFzLjkaw5YCsutD2HbIQmfcsgHs6lqiKovxhBkBYZNdjSADm6m2VASbbsSmPQAh7rxhNgCRrCs+lp3x0V3/jnhMhTZsOqpJrGYcDRD2bwFc7wb2pk0P+xpr7CkrSEvrj4oxaYVaQoJzi5anu7+ZR8hvpWZD2kBC/vqsd/NMJPKTLo3vw5e3MmqLrK/TyTxBoaknRLuFwtp0TtXVu2Vgsh+N9wTfoJjAYqOzqwGfB5dbTb8wjGB09EDDTqj+IijMvTTqA1WCKtkIihD6D/ZemmDDP7DjXsOV7zl2+XdD+Jx2xg86obC6RlDMTH2+/9izZ66+LmvnijW4Ju+Lr+AWM6iVMUhq8jtZxJPljsyf1rPumRdbdcNNT1uUfV89rpPWJr8K0fF6k/WK0fHoQXiF2I/t6lyhwe+wPxUb6mIlp/3yp6emK7WCWTL8FYUDNB7EGbnkwfVXDlJhZlJOPkYbg7VrUXd05RFT7aGRZlGhu6ByhMekdm+tFeDC4IkzBEfHw9+gxfk6nDnb0FbOV46Gcf8/bglrcq0f1LHqL1v9L6FMeCyE+dwymO+L2RyRqmxTHvzj7JrFJ6kGLWoHbsBV9yDWg/3R1ohOfF8rcsVUI9/UzKIU++GqQU/ikMF4fBaYjmLxsffjvbHzG/Rm/n8XUXpRy7eQezmuJ9+MvCrAhoj1F9KwUx/kbIrEWtngupvjd6NCNGfSLSrHfWaSS1L3j/lnR+RM89Lw9uUu/NoRvCJXTGk8jDIJNR3OVp/1uMUbfPy75/88J9wgvZWuHfGfyjWA9f+FnY7L3+R9bSQKF3R3ZeGl4TwKOHKHUC68f7HY9ttA+3iypqhRmbCy3v7CChoiLgLn/o/CjYkIQN0u/rD5iTeMUVyBf1NbVXnFMxpPn75ea68uhRNrOu9Q7yxCJ2rHVvhTxyEuTBqIPcE5ncCVRmIxH+EQWiBdm0o5D/Z8H7HRm7PK0gmq+/f3drLM0vQ2RGXCBfIRVn0+GQ/lnGwDoNeAJsncq5KeOHS81d3ljTOWljfLXkx+DE1V6/ghcnyyTWRHMLZrxcqntsBHLZ/PkZ9iFCMXhk2wT7VHn9Hof90NN1/DzQcZwtQzxoTHhZnN/YiFHfV9/Sm/MNNA3m5rOwjbWv6mnN2DYUGXoXf8rYphwXvUhj1g0s4V1/f/RzGldTQEv0p8AdrswYgP9Eov8zkJJwyMXy0hOjXEqihhhtiS4WHYA1fEAh22eYXxtmJ2d8sGWyIYw1i7k6pYbc6o9ZjhwkQZxmfffzMmmr47m7dLHB0ehwMVy9T5Qf2ZvTDz8kQrAnA9wXRGZ+LU9nJ23/jG/xXlz7vw3Gh8MUTemBNdvJKG/QQNqERQpk9R22VP8IIBaKkeQyZrzjCZdh5zlwr9sD3JmfG2DGBuSYEwTR+SvyU8gZ1f9b8oUVWVHcWCzk3YBfpHh+3rm3emgtjB4EbN98NqcZsvuaFJ/pjg3eNhiJ54w6Lx9EWjDR6QwxLF5M7vO8Vy9tZgRHyDzJumT7hZUvjFn6j98kXG5skKLD1MsqaKJ5nA5mlNOQbRBeWbySXBlRzDcvsWbgNDdQQi7y0lrOtqV0e39kfkmfcFFGc3EtNNdvsA5sE6Pu0qzzC2BGexczYdOaUpRM5xF5QvdK+UkUZ2NBv+f4AgAMwn5ZdypbLzUZWU3vXiR3Oar/eHFjAkjxqo9TG1S/mcFb8vNBJ9B+6AaZYPDH/WY6cA+mCKibMqmp3TK0F1CGhcFNDSmJDtN04JrdpvEYU2ohQHjhJrDhhzEF6Ghc8Z+G8GULPDWQb+cybz+kfhhDKEFlZQADBPy1tUFGD0xSB9u0sDcqFLSvHv5nsHCFhW9fdi96jHJdsFCJkXZHB/7jRhEHhhyd+hzyIQAO+mxFl8nmAmdsDGsu+nHDYakhVf4ALgvVbbx81ZTe/1cZ2EvLBTA0AyGoHobv2SlWKCcf09KQz/Tnm4FiQoPSmtvHxhEXdO4F7tt6KYEHz0aUdXx2G9Ctr/mJa1Dp5cE6DSEWa/tkoqCpkd0yuqVJIVJzfc2FNVyOPZeWcZ8T/mv+AaGJ6N4V/PmU3Rh6/Vb/sXrtyqL+A3BPRVdJ9QH9ITrgBTSsSrORQG7Pdzt2AxH0XHxaNLaPgB5j4m4dRh2vy9/6kBrQRFCPcQhmk2rrUe0QLZN8MJB3P4hrFOlO0hePuvvOuODvb2Ac6BlYb6y5hX6fDWy7ZpqXnHPiIcZ+PtKpSaNDTf6TWp5+r4UmhMXeWUDuopzF8QTD4u3PCOGZvf3rjaVW6AHaVOvuP3WChHjMf2TSeoNOXvj4Q+ZA4xoqzl3CTJeDF6Lpp0NgjuAgDJnvYcTdr0dSxR5nxvZgaSOdMMjzfK3ypjzZ1Tr0iLTvHrzDY1h5GtjwvYqG6w5rdrKkt0W3UX4L6vGyT7e8uEvprRGjj6Gqa+XPqCvoD4CxT2m0UvDOFgc2uGt3X+oYB7JzC/gpZP/e9EmwuzwxaJPMbW69ej3iPP4CMJ9t7eOXbiFa9/yD2EnxjLE0zGW+VS3Ay/Mnp4vO3Zo5+a/ycZpLn/JBjwsD56IxfcjwM1K7wJH0BXsjUUWxbqHVaXwHUbVJBdyy/1QQ8dm/Hv2rl0C8qPqjBu/1LDgfCrz/V1VrHn4qyTJPdfY2gZlw+vQkuPf1ZUmCg/IU+XFf+1KrTWIhlEqZwQo9/ywddZHzx/NQSGE/Om4NnpOnyxLJLzMDqCsR9X/eHG1GpQZcQ3fNiT2IuBxTVYFjQ+UF9UE1nj8WnXKddkFXwFHdhgfXzHsVp/7e9fXx/AJ7Cy0PA9+C4KoGKYEGADJI1/H+lLGh5SepjxSVODoKLb12AtVo4nvCgv4NGrEBPpiH8xn7rDjl1w+qpv6Rv9/MEADDsJKXuyGhpFZVEOuE4xmz3eA3N/ElDPOt1MGB++C1m4lBW9gOLIhIAACdHQzGmV/pNu4m+S6L3n7M28VEpEO3tbuZbpCT9dWHTWsdBqDvb0R7OI9ePvp8+cTWZdP/wAnOVO3v4xg4XMhVFhMKVTtH7OHG7p+BfA6yCjwyuXutd6wA4O0XEE7rBJnSvnGpd5Up0OiyDkqrG9aTmMT1QY4Fx3yGJOsi33Dt3nKHVuOPBiWW3S9NH3z7ZP8Ahyjvm98NPRoDKsnQxVa8b6m3BZx89ygAzF+s8Pgbss3AkKGgQCM4pYkrz57bx3A0Pc57afdL7HDmFzgXq1bn/H2a6AVsZh2aHyhN+lZs2N0TpM3hKT2ed2gvYGaEs43mxnkv/MVfN/81xq8CHH3P0KPQ/TYpgJ5EGGJPT3bdu6TFBH8Jrm/9avFqlRNv0uGXN+UXuXzv2zZZ++9jnD6cKQLLL7NUYvnjfwCVFV8xgPfY+mjw9s9tHLgWCr26gZE3Ac4KcC/Qrt30G8tBFwsPEz/1hvRoR8izr8SJTzqSQmf/0OrBVJsHhbmkRbJjKRaHjd7CYlYmXAIH+q5PeMCBFh3I+VdSIJXWLvRX/MZ2RAjm2hFqjK+r5GgKu7U5b+wXVvlfVzUEfZlPt/tG/JK8Am2Y7H1T2zN0KLQvmd1Ms51SjYKZ8NNCDTXGDZu/aDRf1eQWw+rEBB8HHANOiMOxbNBbGTqSDNhIpN2uE/nyg3UjTsnBm+uH6bc6eOjsNOQw3Cw4DgRxmCt+KDk0LC7xq8I6Zl/AUTX+7x3yXm56vwZhtqZiC1L6Trd45lAoIyYi3mF3AvXH1RO7Qk19VPzbF0yvhJxeDJeeLMAXu7Nxwhb2XwPMLMdwv0kMNQLAOAqwEu6NQkVaYwJ+M3yNyYLTMy9M8KnrQJdFuP0zUIUFwhrtV3f+zrGAkFhzJUsFDKkqCqGFDNtF6IFy1vq4R3yK7/t+/v6NfNfwZHuKWEiBy5RI6Pwam6XfyVM8mfGTDX/XhM2gsLmqlJ3d1upsJQGGqN3HrkbjRY5YL2a6ZxUGELnyWl+x6YJJIcQtT0p3O2yAMqjMW5Dx+8D9jV3HxanCDTJZgPYMBeLZ3B/MXS71sbQzbAsVsxDKfKhNHq47Wmas3ZfLr6nB0g4YrhPj1ZsYZzAQvi8Z1EXFJ+b0QHcDQzCdHmzc7j0/W3jzg9+pP/Y8KWD0d0TuZl4Mtc/+Zr6b8Awo/rn+CzWwR2TWQIeIEcqexmCfeqXSh+1IiWSsiDcUrMHcKsWH7TYmzsN7hdc/MxnB8AnArNZ2gEXe1Nv4dXGH/3s2+cHC3mLAIgrJATe7W4F4X8SS2m6TuvS43r4f2Lf1JGw7/Aje2X9JT9zrrmfNN8PiAkpv4lXfcOLhM9PVW3Ppc63XUHrB8RBDFfu62b3vuPxFNj7yQe72E2WqSqUepCKXBoONRLXNZN2Z4AOjfQaB0Ptt+LJQ0qrO3UXR3wbi9SZfPMpMxu5hLDAAx4DJfHvKNoc8+rehvi0pJuDA1RS6qIjkr3fAv9nNSuHYwjJzmD3DR6IJpzdQgjD7j7I1/4FOZxmuYyA4X8zrGyqlcPSFq6Mvdq1bg6VPgRYwc8XkFNvu7d3K8VP4Qb/a9AP5vgW6SZAB0JmYAYcEFtWydkBNXPfM7WybDeFG7SVdYdkffRBw3q4W7jUgVWnpifdZSW+ZKb2g5jG97Ugn72qcRBNMp2jSEvN6HI7JP9tOXGp8zLsSeM3Hxv5pW5SV6utiU65+3DJhVBv+n0bk6eGiNvCo8PYyPfZKa6B+3W5Tyu0q3Yzjtaks17YkVG9FudwfVK/46IHbfyzp+0IRvGSi4OkbNTSpAL9qzSDVMFVb6Gi+58OZyOSPmHyqJpUlGQK+Ynlwv3Rl4K3tAaDQv1R8hPBFKcbxmR62GawYn0gWhcjr6etu/5WOSd7t/X8yJtoB8CbPr+3NDvgYGo+m4BB8pTFp9tgUj/FJIHoIpq0lhU+RKdRSSVYb5gYX2/5MCMtqE9/SqHv3M35QeDNfWBPJIggTCuD8gYkq/Ip+Rxth3fHNZjajVdM0SbbVdCTo0kJ67+E4Cv3Yvr1b59fwuhU1WjCTR3Y5+X0ZWtvkhPNFbNrkLfH+6rZ9JytRtGsrZ7mUBoyDCyEmOkC+pU3yJtv47Mi5i7NN0CrBCvUw9i4j2zWPxNmhGJtC/81UNBnCoHvEW+v8pgKXf6Vc1P28sQT7xyIcbq/ko6aL9uRj8BSKN5QsxbJYmOs3BONHGbc7D2Q6ixx0ESkt+FV2yyjFPFN4QfAV/w5xxhQbk1yIq13VEGJiJvkh5EeMXg/mE/BJ0XWyvrkL6bD7v38jN1yYBAdeGa9C5NEDrjbsRkfIt+GZZpkeIj7oVuwXC+eSO2UsGbC7HD/3yUE9mPSlBvRyVWQsJlOnWIIzdAuYOwWNOUQGV+hkAQ5sVPx8LXtFyHDy5OIGtAFIbU3ESk5SClMNL/M2FlAjdxs8s0bzTlCvoaaQBdqzeIfe4DwfPYRNzb0pQyX9ZnT7pmP2+IJ+WUH+8c4HaYrAmZHxkALzG/CyiYcxfsHuk/vxwHgTMGRBkJPHfK0fk6KP7bXuSQuuPbNUzjwZDk/YkqTvmPGcN7lK9K08ImC6T8NU+GEO/KV9oYM33QT7bWfU4p6Mbi4LTLTU2JUPmK0ahvJg1BL0xf4Ld4wAAAABJRU5ErkJggg==

AAKwijAAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCK
MAIAAKwij
AAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAK
wijAAAAKs
IIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAA
AKtOK4wsX
75c6enpiouLU2ZmpjZv3nzK/ocOHdJdd92llJQUuVwuXXTRRXrttddOq2AAANC3DAh1QGVlpYqLi7VixQplZmaqvLxcubm5+u
i jjzRixIiT+re3t+v73/++RowYoXXr1ik1NVX79u3TkCFDwlE/AACIcg5jjAllQGZmpq666666io988wzkiS/3y+326177rlHS5YsO
an/ihUr9MQTT2jXrloaOHDgaRXp9Xp9Xq9XqVmJio1tZWJSQknNY6AABAz+ru53dIp2na29tVV1ennJycb1YQE6OcnBz2Oua1tZ2Oua///u
/lZWVpbvuuktJSUm67LLL9Mgjj8jn83W93nba2Nnm93qAFAADoTSGFkZaWFvl8PiUlJQW1JyUlePxDrmL3/5i9atWyefz6fXX
ntNS5cu1S9/+Uv967/+a5fbKSsrU2JiYmBxu92hlAkAAKJIxO+m8fv9GjFihP7t3/5NEyZMUH5+vu6/6//36tWLGiyzElJSVqbWo
NLAoNDZEuEwAAWBLSBazDhg2To+lUc3NzUHtzc7OSk5BJ5PB6t7crNjb2pDEulosulyuUog
AAQ
JQKaWYkNjZWEyZMUHV1daDN7/erurpaWVlZnY753ve+po8++UR+vz/Q9vHHHHyslJaXTIAIAAPqXkE/TFBcXa+XKlXrxxRe1c
+d OzZ8/XoeOHFFRUZEkqaCgQCUlJYH+8+fP1+eff66FCxfq448/1vr16/XII4/orrvuCt9eAACAqBXyc0by8/N18OBBBLVu2TB6PR
xkZGdqwYUPg0otb6+nrFxHyTcdxutzZu3Kif/vSnuuKKK5SamqqFCxdq8eLF4dsLAAAQtUJ+zogNPGcEAIDoE5HnjAAAAIQbYQ
QQ
AAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBE
AAGAVYQQAAFhFG
AEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVY
QQAAFhFGAEAAFY
RRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFh
FGAEAAFYRRgAAg
FWEEQAAYBVhBAAAWHVaYWT58uVKTo9XXFycMjMztzXnz5i77vvvDCC3I4HEFLXFzcaRcMAAD6lpDDSGVlpYqLi1VaWqot
W7Zo/Pj
xys3N1YEDB7ock5Qo+Kampsyb9++MyoaADoHSGHkaeeekpz5sxRUGRxobpxUrbVig+Pl6rV6/ucozD4VBycnJgSUpKOqozD4VBycnJgSUpKOqO
iA
QBA3xFSGGlvb1ddXZ1ycnK+WUFMjHJyclRbW9vluC+//FLnnXee3G63bk6dqj/96U+n3E5bW5u8Xm/QAgAA+qaQwkhLS4t8P
t9
JMxtJSUnyeDydjhk7dqxWr16tV1y6tV199Vb/5zW/k9/t1zTXXqLGxscsvLGxscvtlJWVKTExMbC43e5QygQAAFEk4nfTZGVlqaCgQBkZGbruu
utUVVWl4cOH6/nnn+9yTrLtV/3kMnuqyoRqaFXmLGxscvtlJWVKTExMbC43e5QygQAAFEk4nfTZGVlqaCgQBkZGbruu
utUVVWl4cOH6/nnn+9yTElJiVpbWwNLQoNDpMsEAACWDAil87Bhw+Ro0Xc3BzU3tzcrOTk5G6tY+DAgbryyivefdNn
H5XL
J5XKFUhoAAIhSIc2MxMbGasKECqurg6o+fi+VVdXKysrq1vr8Pl82rZtm1JSUkKrFAAA9EkhzYxIUnFxsQ0LCzVx4kRNmjRJ5
eXlOnLkiIqKiiRJBQUFSk1NVVlZmSTpoYceotVXX6oLL7xQhw4dohNPPKF9+/bpbzjvvDO+eAACAqBRyGMnPz9fgwebew1bNkye
Tw
eZWRkaMOGDYGLWuvr6xUT88+8Kf88EyxffKE5c+bI4/PsBhht59912NGzcufHsBAACilsMYY2wX8V28Xq8SExtqq8SExPV2tqqhIQ
E2
+UAAIBu6O7nN99NAwAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAA
KsIIwAAwCr
CCAAAsIowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAA
wCrCCAAAs
IowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCA
AAsIowAgA
ArCKMAAAAqwgjAADAKsIIAACwijACAACsOqowsnz5cqWnpysuLk6ZmZnavHlzt8atWbNGDodoDo6ZNO53NAgCAPijkMFJ
ZWani4
mKVlpZqy5YtGj9+vHJzc3XgwIFTjtu7d6/uu+8+ZWdnn3axxxACg7wk5jDz11FOaM2eiOioqKNG7cOK1YsULx8fFavXpi2N8Pp9
uu+o2Pfjggzr//PPPqGAAAANC3hBRG2tvbVdXp5ycnG9WEBOjnJycn2Foo4GRIzR79uzTrxQAAPRJAoLp3NLSIp/Pp
6SkpKD2pKQk7dq1q9Mxb7b/tlatWtW7d2z7ttbW1W1qa2sLvPZ6vaGUCQAAokh5+ZL3mCXL+Y8hLg1pm/7mjp/OaKydqd31jPNa17FvcQAQG2wAsb2SSVhxmWVAuLVVLB6E7Fp9PEl0mm1KhXmhSS1Xm6Pp/8gjsqH0ikKWNZWSaFDLUzZ3T5pYN1C5XSKr8vpA0N3qj8Vcd1PWh2uMjfo33T6sbV03s6gVPePkVS7J3RH9a8Wxw4iXYj+c+lrmdlXhqyXLOM5tLnIR9YOcCfGkxBYqUzIsDtDhpO
4ejQmuIj4wdR1CeiyAOF6QAAORtB2qMMDNlN57QfSrr5BoAFD3mtgmYY9ekBIRkmBxdPt4q1jKpP3vFAAoKhE76eOPsPKDW3qLPpgZ3BjpYkmNNWwL+FzrJARQPcLCzVx4kRNmjRJ5Ymn4ztz/2PD3fjvyUQYpOiv6VKgQRhsPbG5yyeKNkZL1NvlLy9XLOvNbGoDqqCVHv1fXYffw4c89NGYzWXm1CO8n2R22nI2F5DjUyDlX4rr2QZpKl9+9aeSiTCBg4jqpCmcu/uzzi+xtD1IF4PZcTzpBcPvv6KOS/dZN+T9TTTvfg0DojJNTM2kxDs76q5QP6lIuqD2KGpUkkwhKrmQyG+0EsoJ/gEKqd61j2jd5+/2WHc7fZnfhhdKaDFedHaDUnmbrEnupqv+v/xZKN9qPsD6DJF37xb1HS6jpBBs5sUVkfWRDCYPS4wvHcO/mRrLy7vMq1k3cF+HzcJjbxkZxO15kSRQX1p5VK9v3r30uKLXHQ
QwogkFRcXq7CwUBMnTtSkSZNUXl6uIoeOqKioSJJUFCg1NRlZlKMBxKSJJUUFCgRUlZWVKS4uTpdddlnQ+CFDhkjSfu6+DB

g1q2b
Jk8Ho8yMjKoYcOGwEWt9fX1ionhwa4AAKB7HMYYY7uI7+L1epWYmKjW1lYlJCTYLgcAAHRDdz+/mcIAAABWEUYAAIBVh
BEAAGA
VYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIB
VhBEAAGAVYQQAA
FhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAA
GAVYQQAAFhFGAE
AAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVYQ
QAAFhFGAEAAFYRR
gAAgFWnFUaWL1+u9PRoxcXFKTMzU5s3b+6yb1VVlSZOnKghQ4borLPOUkZGhl566aXTLhgAAPQtIYeRyspKFRcXq7SoVFu
2bNH
48eOVm5urAwcOdNr/nHPOf3336/a2lr98Y9/VFFRkYqKirx48YzLh4AAEQ/hzHGhDIgMzNTV111lZ555hlJkt/vl9vt1j333
KMlS5Zoax1/8zd/o5tvvvlkPP/xwt/p7vV4lJiaqtbVVCQkJoZQLAAAs6e7ndogzI+3t7aqr1NOTs43K4jJUU5Ojmpra79zvDF
G1dXV+uijj/T//67JfW1ubvF5voAIAAPqmkMJISouLfD6fkpKSgtqCjSI3fDB48WFu3btUH3ygX/ziFyoufouLlZNTU2X/UtKStTa2hpYGhoaeqJMAABgwYBQOg8bNkx
Op1PNzc1B7c3NzUpOTu5yXExMjC688EJJUkZGhnbu3KmysjyBCTSk7jBlRnbNk7gbmmUu4/vmYsjJNnjy5o/4ulosulyuU0ogAAQJQKaWYkNjZWEyZMCMVHVlYmqeP6j4kTJ+qC
ru
rpaWVlZ3V6P3+9XW1tbKJsGAAB9VEgzI5JUXFysw6UZMmTVJ5ebmOHDmioqIiSVJBQYFSU1NVVlYmqeP6j4kTJ+qC
Cy5
QW1ubXnvtNb3ookt67rnnwrsnAAAgKoUcRvLz83Xw4EEtW7ZMHo9HG7ZrBhQ+Ci1vr6esXEfDPhcuTIES1YsECNjYoaNGi
QL
r74Yv3mN79Rfn5++PYCAABErZCfM2IDzxkBACD6ROQ5IwAAAOFGGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAV
hFGAACAVYQ
RAABgFWEEAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYNsFoAgH7M55M2bZKamqSUFCk7W3I6bVcFoIc
RRgDYUVUlL
VwoNTZ+05aWJlVUSHl59uoCoOM4TQOg51VVSdOnBwcRSdq71+XtVSdOnBwcRSdq/v6O9qspOXQCsIIwA6Fk+X8eMiDEnv3e8bdGijn4A+gXCCIC
etWn
TyTMiJzJGamjo6AegXyCMAOhZTU3h7Qcg6hFGAPSlJTw9gMQ9QgjAHpWdnbHXTMOR+fvOxyS293RD0C/QBgBoLOcz07
bd6WTA
8nx1+XlPG8E6EcIIwB6Xl6etG6dlJoa3J6W1tHOc0SAfoJ8OPWdJw+6TQ1TcnA8sj6U8z
HWI5HqGdGUHKYMaPfKpR/TBkhn1tfHnFzaXbnqWyEWwP9AcC5Ckv23tkM5ZD+4N5h4Grf6ZhJb6Wo6wpJmOmp7KDrCUKNDMHLxgpqyDCyTXrxgCwY9AaT5uEQgnu8JzBCwz7AkLDuYP7ry6gaqgmGkPZ0ygsRpAlIhuaUiy
IXfOd7n82n0oKF65plnVFBQoK1ter1eJSYmqrW1VQkJCaGUCwAALOnu53dIMyPt7e2qq6tTTk7ONyuIiVFOTo5q6/6tTTk7ONyuIiVFOTo5qa2u7tY6vvvvp
Kx44doznnnNNln7a2Nnm93qAFAADoTSGFkZaWFvl8PiUlJyeyUlyePxdGsdixcv1siRI4MCzbeVlZUpMTExsLjd7lDKBBBAlf
D6ppkZ6+eWOnz6f7Y0A2NCjd9M8+uijmP6tAPq
XAaFoHjZsmP5Pam5bqm4Pam5ublZycfMqxTz75pB599FH9/e/aa7dc8992jJk
iXfOd7n82n0oKF65plnVFBQoK1er1eJSYmqrW1VQkJCaGUCwAALOnu53dIMyPt7e2qq6tTtk7ONyuIiVFOTo5qa2u7tY6vvvvp
Kx44doznnnNNln7a2Nnm93qAFAADoTSGFkZaWFvl8PiUlJQW1JyUlyePxdGsdixcv1siRI4MCzbeVlZUpMTExsLjd7lDKBBBAlf
D6ppkZ6+eWOnz6f7YoA2NCjd9M8+uijjWrNmjf7zP/9TcXFxXfYrKSlRa2trYGloaOjBKGHohKoqKT1dmjJFuvXWjp/
XAaFoHjZsmJxOp5qbm4Pam5ublZycfMqxTz75pB599FH9/e/aa7dc8992jJk
6c2ADovpJmR2NhYTZGwQdXV1YE2v9+v6upqZWVldTnu8ccf16Mq
muLhYK1eu1IsvvqidO3dq/vz5OnLkiIqKiiRJBQUFKikpCfR/7LHHHtHTpUq1evVrp6enyeDzyeDz68ssvw7cXAKLGpk1SY2PX7
xsjNTR09APQP4RomkaS8vPzdfDgQS1btkHD4KLW+vp6xcR8k3Gee+45tbe3a3a/ro6UHrKSot1b/8y7+cWfUAok5
TU3j7AYh+IT9nxAaeMwLoHTU1HRerfpc335QmT450NQAiKSLPGQGQ05xGAM5WdLaWlSL9+3HLb3dEPQP9AGAHQoA
aKi49/fD
iTHX5eXd/QDoD8QRgDouLy8jtt3U1P29PSuKoX6I9CvoAVAMIhL0+aOrXjrpmmJiklpePUDDMiQP9DGAFgjdPJRaoAOEo
DAAA
sI4wAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAAKsIIwAAAsrCCAAAsIqKJ7AaYyR1fBUxACIDsc/t49zjlKs
LI4
cOHJUlut9tyJQAIFSHDx9WYmYzHfFlV7A7/froo8/1eDBg+X49neO9zNer1dNdutisNDQ1KSEiwU6fxrHuGRznnsFx7hk
c52DGGB0+fFgjRo5FpjsR4UOLbSJfRqyQkMEExxcxITEwmotLSbJfRqyQkMEExxcxITEyMotLSbJfRqyQkJPCL3kM41j2D49wzOM49g+P8jVPNiBzHBawAAMAqwggAALCKM
BJlX
C6XSktL5XbJfS53GsewbHuWdwnHsGx/noRMUFrAAAAo9iZgQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhpBf6/PPPddNNNyN
khIUF

DhgzR7Nmz9eWXX55yzNGjR3XXXXfp3HPP1dlnn6of/vCHam5u7rTvZ599prSoNDkcDho6dCgCexAdInGc//CHP2jmzJlyu9oa
N
GiQLrnkElVUVER6V3qV5cuXKzo9XXFxccrMzNTmzZtP2f+VV17RxRdfrLi4OF1++eV67bXXgt43xmjZsmVKSUnRoEGDlJOToz
/
/+c+R3IWoEM7jfOzYMS1evFiXX365zjrrLIocOVIFBQX69NNPI70bUSHcv9MnmjdvnhwOh8rLy8NcdZQx6HVuuukmM378ePP
ee
++ZTZs2mQsvvNDMnDnzlGPmzZtn3G63qa6uNh9++KG5+uqrzTXXXNNp36lTp5q//du/NZLMF198EYE9iA6ROM6rVqoyP/n
JTox
NTY3ZvXu3eemll8ygQYPMoo8/Hend6RXWrFljYmNjzerVq82f/vQnM2fOHDNkyBDT3Nzcaf933nnHOJ1O8/jjj5sdO3aYBx54w
AwcONBs27YtoOfRRx81iYmJ5r/+67/MH/7wB/ODH/zAjB492nz99dc9tVu9TriP86FDhoxOTo6prKwou3btMrW1tWbSpElmwo
Q JPblbvVIkfqePq6qqMuPHjzcjR44ov/rVryK8J7obYaSX2bFjh5FkPvjggoDb7373O+NwOMz+/fs7HXPooCEzcOBA88orrwTad
u7caSSZ2traoL7PPvusue666ox1dXW/DiORPs4nWrBggZkyZUr4iu/FJk2aZO66667Aa5/PZoaOHGnKyso67T9jxgz88o3B7V
lZmaaf/qnfzLGGOP3+o1ycrJ54oknAu8fOnTIuFwu8/LLLodgD6JDuI9zZzZv3mwkmX379oWn6CgVqWPd2NhoUlNTzfbt28155
53X78MIp2l6mdraWgoZMkQTJo4MtOXk5CgmJkbvv/9+p2p6up07Ngx5eTkBNouvvhijijR01SrW1tYG2HTt26KGHHtKvf/3rU35
hUX8QyeP8ba2trTrnnHPCV3wv1d7errq6uqBjFBMTo5ycnFMeo9ra2qD+kpSbmxvo39DQIK/XG9THGyEwadKkwLY25byBMMT05ycnC6PT21tbVB/ScrNzQ3o37NnjzweT1CfxMREZWZmnvKY92WROM
6da
W1tlcPhoJAhQ8JSdzSK1LH2+/2644479LOf/UyXXnppZIqPMv37E6kX8ng8GjFiRFDbgAEDdM4558jj8XQ5JjY29qT/NJKSkgJ
j2traNHPmTD3xxBMaNWpURGqGjPJpE6zt/27rvvqrKyUnPnzg1L3b1ZSouLfD6fkpKSgtpPdXw8Hs8p+x//Gc06+7pIHOdvO3roq
BYvXqyZM2f26y97i9SxfuyxxzRgwAD95Cc/CX/RUYowokOWLFkih8NxymxmXXrloR235JSYkuueQS3X777RHbRm9g+zifaPv27Zo
6dapKSot144o39sg2gTN17NgxzZgxQ8YYPffc7bL6XPq6upUUVGhF154QQ6Hw3Y5vcYA2wXoF/fee69mzZp1yj7nn3++kpOTd
eDAgaD2v/71r/r888+VnJzc6Z2k5GS1t7fro0KFDQX+1Nzc3B8a88cYb2rZtm9atWyep4w4FSRo2bJjJjuv/9+Pfjgg6e5Z72L7eN
83I4dO3TDDTd07ty5euCBBo5rX6LNsGHD5HQ6T7qLf+x//x//2dzcrJSUlKA+GRkYaw+ekTiOB93PIs27dPb7zxR
r+eFZEic6w3bdqkAwcOBM1Q+3w+3XvvvSovL9fev/Mo5efs+d/wenXzDuxPRwvZFKwh2/MLKDz/8MNC2cePGbl1YuW7WdukDbrl27gi6s/OSTT
8y 2bdsCy+rVq4ok8+6773Z5VXhfFqnjbIkx27dvNyNGjDA/+9nPIrcDvdSkSKSZPM3XffHXjt8/lMamrqKS/2+/u///ugtqysrJMuY
H3yyScD77e2tnIBa5iPszHGtLe3m2nTpplLL73UHDhwIDKFR6FwH+vfwIDKFR6FwH+uWlpag/4u3bdtmRo4caRYvXmx27doVuR3p5QgjvdBN
91

```
mozw4cPN7fffrtpamoKLP3lP/c1a9YYl8tlXnjhBbNjxw4zd+5cM2TIEOPxeIwxxtxxxx1myZIlgf7vvPOOGTBggHnyySfNzpo
7TWlpaae39g4ZMsS8+uqr5o9//KOZOnUqt/aG+Ti3t7ebH/zgByYtLc1s3bo16He3ra3Nyj72FpH4nf427qYhjPRKn332mZk5c
6Y5++yzTUJCgikqKjKHDx8OvL9nzx4jybz55puBtq+//tosWLADBo61MTHx5t/+Id/ME1NTV1ugzASmeNcWlpqJJ2onHfeeT2
4Z3Y9/fTTZtSoUSY2NtZMmjTJvPfee4H3rrvuOlNYWBjUf+3ateaiiy4ysbGx5tJLLzXr168Pet/v95ulS5eapKQk43K5zA033
GA++uijntiVXi2cx/n473pny4m///1VuH+nv4owYozDmP+7eAAAAMAC7qYBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgA
AgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABY9f8BPYHTt1xjaqAAAAAAASUVORK5CYII=",
"text/plain": [
 "<Figure size 640x480 with 1 Axes>"
 ]
 },
 "metadata": {},
 "output_type": "display_data"
 }
 ],
 "source": [
 "# Visualize \n",
 "plt.plot(result_Kc, 'ro', label='Kc')\n",
 "plt.plot(result_tauI, 'go', label='tauI')\n",
 "plt.plot(result_tauD, 'bo', label='tauD')\n",
 "\n",
 "#plt.xlabel('Kc, tauI, tauD');\n",
 "#plt.legend((result_Kc, result_tauI, result_tauD), ('Kc', 'tauI', 'tauD'))\n", "\n",
 "plt.legend(loc='upper left')\n",
 "#pylab.ylim(-1.5, 2.0)\n",
 "plt.show()"
 ]
 },
 {
 "cell_type": "code",
 "execution_count": 68,
 "metadata": {},
 "outputs": [],
 "source": [
 "import numpy as np\n",
 "import matplotlib.pyplot as plt\n",
 "from scipy.integrate import odeint\n",
 "import ipywidgets as wg\n",
 "from IPython.display import display"
 ]
 },
 {
 "cell_type": "code",
 "execution_count": 69,
 "metadata": {},
 "outputs": [
 {
 "data": {
 "application/vnd.jupyter.widget-view+json": {
 "model_id": "8dbefe67155b4bdc84a972358f0a5fd5",
 "version_major": 2,
```

"version_minor": 0
},
"text/plain": [
"interactive(children=(FloatSlider(value=0.2517879605293274, description='Kc', max=0.7553638815879822, min=-0.2…"
]
},
"metadata": {},

```
    "data": {
     "text/plain": [
      "<function __main__.pidPlot(Kc, tauI, tauD)>"
     ]
    },
    "execution_count": 69,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "n = 100 # time points to plot\n",
   "tf = 50.0 # final time\n",
   "SP_start = 2.0 # time of set point change\n",
   "\n",
   "def process(y,t,u):\n",
   "    Kp = 4.0\n",
   "    taup = 3.0\n",
   "    thetap = 1.0\n",
   "    if t<(thetap+SP_start):\n",
   "        dydt = 0.0  # time delay\n",
   "    else:\n",
   "        dydt = (1.0/taup) * (-y + Kp * u)\n",
   "    return dydt\n",
   "\n",
   "def pidPlot(Kc,tauI,tauD):\n",
   "    t = np.linspace(0,tf,n) # create time vector\n",
   "    P= np.zeros(n)          # initialize proportional term\n",
   "    I = np.zeros(n)         # initialize integral term\n",
   "    D = np.zeros(n)         # initialize derivative term\n",
   "    e = np.zeros(n)         # initialize error\n",
   "    OP = np.zeros(n)        # initialize controller output\n",
   "    PV = np.zeros(n)        # initialize process variable\n",
   "    SP = np.zeros(n)        # initialize setpoint\n",
   "    SP_step = int(SP_start/(tf/(n-1))+1) # setpoint start\n",
   "    SP[0:SP_step] = 0.0     # define setpoint\n",
   "    SP[SP_step:n] = 4.0     # step up\n",
   "    y0 = 0.0                # initial condition\n",
   "    # loop through all time steps\n",
   "    for i in range(1,n):\n",
   "        # simulate process for one time step\n",
   "        ts = [t[i-1],t[i]]         # time interval\n",
   "        y = odeint(process,y0,ts,args=(OP[i-1],))  # compute next step\n",
   "        y0 = y[1]                  # record new initial condition\n",
   "        # calculate new OP with PID\n",
   "        PV[i] = y[1]               # record PV\n",
   "        e[i] = SP[i] - PV[i]       # calculate error = SP - PV\n",
   "        dt = t[i] - t[i-1]         # calculate time step\n",
   "        P[i] = Kc * e[i]           # calculate proportional term\n",
   "        I[i] = I[i-1] + (Kc/tauI) * e[i] * dt  # calculate integral term\n",
```

```
"        D[i] = -Kc * tauD * (PV[i]-PV[i-1])/dt # calculate derivative term\n",
"        OP[i] = P[i] + I[i] + D[i] # calculate new controller output\n",     "        \n",
"    # plot PID response\n",
"    plt.figure(1,figsize=(15,7))\n",
"    plt.subplot(2,2,1)\n",
"    plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')\n",
"    plt.plot(t,PV,'r:',linewidth=2,label='Process Variable (PV)')\n",
```

```
    "    plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\\frac{K_c}{\\tau_I} \\int_{i=0}^{n_t} e(t) \\; dt $')\n",
    "    plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c \\tau_D \\frac{d(PV)}{dt}$')
\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,3)\n",
    "    plt.plot(t,e,'m--',linewidth=2,label='Error (e=SP-PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,4)\n",
    "    plt.plot(t,OP,'b--',linewidth=2,label='Controller Output (OP)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.xlabel('time')\n",
    "    \n",
    "Kc_slide = result_Kc\n",
    "tauI_slide = result_tauI\n",
    "tauD_slide = result_tauD\n",
    "wg.interact(pidPlot, Kc=Kc_slide, tauI=tauI_slide, tauD=tauD_slide)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Dicoba diberi masukan e(t) sembarang\n",
    "Pengujian ke-3"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 70,
   "metadata": {},
   "outputs": [],
   "source": [
    "ujicoba3 = np.array([\n",
    "    [1.2, 0.1]\n",
    "])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 71,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([[1.2, 0.1]])"
      ]
     },
     "execution_count": 71,
```

```
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "ujicoba3"
 ]
},
```

```
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "1/1 [==============================] - 0s 31ms/step\n"
   ]
  }
 ],
 "source": [
  "outDL = model.predict(ujicoba3)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 73,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([[0.24446805, 0.8346314 , 0.18357222]], dtype=float32)"
    ]
   },
   "execution_count": 73,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "outDL"
 ]
},
{
 "cell_type": "code",
 "execution_count": 74,
 "metadata": {},
 "outputs": [],
 "source": [
```

```
   "result_Kc   = outDL[0,0]\n",
   "result_tauI = outDL[0,1]\n",
   "result_tauD = outDL[0,2]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 75,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "0.24446805"
     ]
    },
    "execution_count": 75,
    "metadata": {},
```

```
  "source": [
   "result_Kc"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 76,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "0.8346314"
     ]
    },
    "execution_count": 76,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "result_tauI"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 77,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "0.18357222"
     ]
    },
    "execution_count": 77,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "result_tauD"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 78,
  "metadata": {},
  "outputs": [
   {
    "data": {       "image/png":
```

"iVBORw0KGgoAAAANSUhEUgAAAiMAAAGdCAYAAADAAnMpAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLj
iwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8g+/7EAAAACXBIWXMAAA9hAAAPYQGoP6dpAAAr4klEQVR4nO3dfXTU1Z3H8c/MQ
iZMACSUgGg40PWCUumBhrVtDUuOsqNGVJ8SEhVVxFLTTaBXxIVjoan2qToyhdFo7WHiXIZtc9i4Vuo/GgRtBwbKGAVgokwUwgK
hlESdiZu39kGRhJMAMzuUzyfp3zO5g7987v+7snx/nk/h7GYYwxAgAAsMRpuwAAADCwEUYAAIBVhBEAAGAVYQQQQAAFhFGAEAAF
RRgAAgFWEEQAAYBVhBAAAWDXIdgG9EQgE9Nlnn2nYsGFyOBy2ywEAAL1gjjNH+/fs1ZswYOZo9r3/ERBj57LPP5PF4bJcBAABOQ
FNTk9LTo3t8PSbCyLBhwyR1HUxUrSUpLlagAAQG/4fD55PJ7g53hPYiKMMHD41k5SURBgACDGfNclFlzADgAArCKMAAAqwgjAAD
Aqpi4ZqQ3/H6/Dho6ZLuMmDZ48GC5XC7bBZQAABph+EUa++uorNTc3yxhju5ru5SY5nA4lJ6erqEDh9ouBQAwgMR8GPH7/Wpubl
ZCQoJSU1O5KVofGWO0bds2NTc368gjj+QICQCg38R8GOno6JAxRqmpqYqPj7ddTkxLTU3Vp59+qo6ODsIIAKDfHDAnsHJE5PtjDgEANhwwYQQAAMQmwggAALCKMAIAAKwijFji8/lUWFio3NxcJSUlKS0tTaeeeqqqq6sVCAQkSTU1NTr//POVnZ2tmTNnqra21nLVAABE3gF5NU3O5S/26/i0v/rXPm3f29ur2bNnKzc3V5WVlVq9erWqqqp09uzZqqmp0ZIlS+TxePTqq69q8ODBys3N1fvvv6/ExMQo/QQAAPQdjozsB4/Ho9dff11xcXHKy8vTpEmTVFlZqSRZs2ZpzZo1uvzyyzV48GBJUlJSkrZu3aq1a9faLBsAgO4OHEZ6eddNrlixQiUlJSorK1NBQYFuuOEGLVu2TAsWLNBbb71lu7ypU2gSHRERqgV0XRp6vYBQAkVRY43m5qaWlRVdddZXOPPNMbd68Wdu3b5fb7VZKSoo++ugjDR48WH/5y190xRVX6KGHHtLRRx+tN998U5mZmTrppJO0bt06SdJ1112nSZMmKT4+XjfeeKOuu+46jR07VqtWrZLX69WaNWt07LHHaufOnRo9erQqKys1bdo0Pfroo7rmmmv0wAMP6KqrrtJzzz2nzZs3a+vWrcrNzdXChQt1xBFHaPny5dq4caMaGhqUn5+vBx98UDk5OVqyZImWLVumjz/+WFVVVZozZ44efvhhTZgwQS+88ILGjBmjDz74QDfffLOef/55NTY2qqamRiUlJXrmmWc0YcIEvf/++5o2bZo++eQTLVq0SC+99JI2bdqkuXPn6rnnnlNpaanWr1+vk046SRs3btSvfvUrLV26VJ99tkWzZs3S888/r7lz5+qbb77RBRdcoJqaGl177bV6+eWX9dprr+myyy7Tiy++qPT0dJ1//vlauXKlrrrqKq1cuVLPP/+8Zs2apRdeeEGZmZk677zztGbNGnk8HrW1tWn58uW68cYbtXLlSqWnp+ucc87RypUrNXv2bLW0tOjpp5/W9ddfrxdeeEGZmZk644wztGrVKt100016/fXXtXHjRt1yyy1avny5MjIydOaZZ2rFihW66aabtHr1ar366qu67bbbtGLFCqWnp+vMM8/UW2+9pUsuuUTr1q3T888/r1tvvVUvvPCCMjMzdfrpp6u2tlbnnnuuNm3apGeeeUY33XSTnnvuOWVkZOiMM87Q6tWrdeaZZ6qhoUGPPvqobr31Vj3//PPKyMjQaaedpvr6es2YMUNr167VI488optvvlkvvvii0tPTdeqpp6q+vl4zZ87U66+/roceeki33nqrXnjhBWVkZOjUU0/Vxo0bdcopp+i1117TAw88oPXr1+vhhx/WHXfcoeeff17p6ek6+eSTtWbNGs2cOVNr167Vww8/rJtuuknPP/+8MjIy9PHHH+uiiy7SmjVrdO+99+rGG2/Uc889p4yMDB1//PFqaGjQqaeeqldffVUPPvig5s+fr+eee06ZmZk6+uijdcEFF2j16tW6//77NW/ePL3wwgvKzMzUkUceqUsvvVQvv/yyfvOb32jOnDl65plnNH78eB1yyCG65pprVFVVpV/96le6+uqr9fTTT+uwww7TxRdfrOrqat1zzz266qqr9NRTT+mQQw7RpZdeqrq6Ot11112aNWuWnnrqKR188MG6/PLLVVVVpdtvv10zZ87Uk08+qQMPPFBXXHGFqqqqVFpaqquvvlpPPPGEDjjgAF155ZWqqqpScXGxrrnmGj3++OPaf//9dcUVV6iqqkqXXXaZZs+ercd6YQ6XL5o0qVL1dOmaOnS/2//9u2Cv+1PaHPPfecr7eXHvO+8+J+Lz8+POsSP3/e6cFvJbXr+7t+i5W/PPzO+2x6Y/vPK/Hp6gcPwVLlL9KPf8L9dE5n//K/7S8Tr9pf8XdvC1z5vz/zO/a++kPjX8tA/d/7KsJHZ6vn/ys/9ytvPOzXKs6KH6uepU8+dZzHg5c+cz2e+/LRnjcfVF9e1LmVzpfmWefbZZOr9n9sv5ZnPflsUp/tm+79y1Y/+xTy9XvTOW9OQ59/vff+n6XYOfpI/8J7v/+e0YE//9+e9nSeTtY/Rf8v+hf8b8+e/2zXp/j7a9Kzj3DfnHP/77b/Zu2/5dGf3vL/+/h/8/+zzfOvN+u3e93+1j7s/r7vf2/fbbO/vu87r93v/z+z//Nbxb3Tb3nv//f2uW2fvb2zP/XP/n/dvd9++/jzvY/77Tbbb9/v7t/25/91/+dy+8/Ve+/+71n/9933/+c3/9+a/vf5+tZ5/3f+Z/q+f//7/1d//rv/f/+3f/nz//87/fvP3/9+/3/P+/3/f//3+d//3+d//3+d9+/f3/f3/s/9/z+/f//f+f9+d//P/v/1/+/3/+/+/3+f/33/v/vX///v/9f/+//33/v/5//fv3/9/b/3v/++r/f//+/v/v/v/3/f//f/3/v/f/9//f/7/f/3/v//f+/v//+/9/f/v//9/v//v/f/8/v//P/7//+/9/f//9/f/9/9//3/v///f/v//+9/v//f//9//v/9/f+//X7/3//7/3//+//7/3/9/v/v//+/f//v/8/

CAGyI+ZURALHLH/BrXeM6texvUeqwVOWOzZXLyaocMNAQRgBYUbO1RvPWzFOzrznYlp6UrqprqlRwfoHFygDotX5zmua
k+f1SX
Z306qtd//r9tisC+q2arTWasXJGSBCRpN2+3ZqxcoZqttZYqgyADYQRSaqpkTIypKlTpRtu6Po3I6OrPUpmz56t6dOnh7StWrV
KCQkJ+uUvfxm1/QK2+QN+zVszTobH30p/uG3+mvnyB/iDABg0CCM1NdKMGVJz6F902r27qz2KgeRo//Zv/6Ybb7xRL7zwgu65
5
54+2Sdgw7rGdcesiBzNyKjJ16R1jev6sCoANg3sMOL3S/PmSdo9LO1w2/z5UT9l8+STT+ruu+/WihUrVFJSIkkKBAJ68skndfb
ZZys+Pl5jx47Vo48+GtU6gL7Qsr8lovoAxL6BfQHrunXHrogczRipqamr35QpUSlhwYIFev755/Xf//3fuuqqq4LtixYtotKlS
/WrX/1Kl19+uVpaWrRt27a01ADopdRhqRHtByD2DewLLv7x62y9Mv/vd7/T666+rtrZWV155ZbB9//79qqqqonPPPPafi4mJ
JollnnaXLL788KnUAfSl3bK7Sk9K127e72+tGHHIoPSlduWNzLVQHwIaBfZomtZd/efW2X5guuugiZWRkqLy8XF999VWwfevW
r ero6AhZKQH6C5fTpaprqiR1BY+jHf658ppKnjcCDCADO4zk5krp6VJPTx51OCSPp6tfFKSlpamurk67d+/WNddco/3790sSj7V
Hv1dwfoFWzVyltKSokPbopHStmrmK54wAA8zADiMul1TV9RfaMYHk8M+VlV39ouSMM87Q22+/La/XGwwk48eP15AhQ1Rb
Wxu1/
QK2FZxfoJ3zduqt4rfoSsEreqv4Le2Yt4MgAgxAAzuMSFJBgbRqlZQW+heaotO72gui/z9Gj8ejuro67dmzR/n5+ers7NSCBQv
oz//8z/rNb36j7du36/3339eyZcuiXgvQl1xOl6ZkTNGsC2dpSsYUTsoAA9TAvoD1sIICadqorrtmWlq6rhHJzY3qisi3paenq
66uTlOnTlV+fr7Wrl2rQYMGqaysTJ999pq9TJ999plSU1N1++2391k9AD0FYcx3T1k49Ti8/nkdrvV3t6upKSkkNcOHjyoHTt2aNy4cUp
ISLBUYf/AXAIAIul4n99H4zQNAACwijACAAACsIowAAACrCCMAAMAqwggAALDqhMLI4sWLZlGRoYSEBBGvnZ2vvDhg3H7VDhg7H7V
9ZWalzz
z1XQ4YMkcfjoc9//nMdPHjwhAoGAAD9S9hhpLq6WsL6Wq9lpSovL9fGjRs1ceJE5efna8+ePd32f+WVV7Rw4UKVl5dr69atWrZsma
q rq3XfffeddPEAACD2hR1GnnnnmGc2ZMoclJSWaMGGClixZosTERC1fvrzb/u+9955TVCrzb/u+9955+8IMf6IYbblBGRoauvvpqzZo16ztXUwAAw
MAQVhjp7OxUQ0OD8vLyjryBo6m8m8vDzV19d3O+ayyy5TQ0NDMHz89a9/1RtvvKG///u/73E/HRod8vl8IRsAAOifwgojbW1t8v
v
9Sk5ODmlPTk6W1+vtdswNN9yghx9+WJdffrkGDx6ss846S1OmTDnuaZqZqKigq53e7g5vF4winzhPgDftXtrNORrm15V3c46+QP+q
O5vypQpmj9/fsy8LwAAoRL1u2nq6uro2GOP6fnnn6funn9fGjRtVU1Oj1atX65FHHuluxKJFKpaKBJFi9Te3h7cmpqaolpjzdYaZVRlaOpLU3V
DzQ2a+tJUZVRlqGZrTVT3CwAwgjIoeOlMvlUmtra0h7a2urUlJSuh3z3z4IMP6uabb9bs3btHH/6bt3bs+uWW27nYOSMqgWT27RuHjNGTIEJ177rmqq
qo
KGdfdqsfo6dM1e/bsiNcIAEBfCSuMxMXFKfrsMXFadKkSaqtrVVhYaGkrgtYa2trVW3//dXp/9+Xe3/N24/v/bcG1/R58/4Ne8NfNkdGwdh
9vmr5kf8VM2VVVVVysnJoZw5c9TSoqKWlhalp6crPT1dr732rs2aKyKysjOqWZQuANKSotVXFysyZMn
Kys
rS5WVlTpw4IBKSkokScXFxUpLS1NFRYUk6brrrtMzzzyjiy++WDk5Odq2bZsfr8++WNnZ2fro0/14IMP6rrrguGElvWNa47ZkXkaEZGTb4mrWtcp
ykZUyK2X7fbrbi4OCUmJoasKD30ooEPB/x43bpzq6+u1cuVKZZw5M2L7BgDgVBN2Ck5zkJnhRa
2N
jY8hKyAMPPCCHw6EHHnhAu3fv1qhR03tdddfpoUcfjdxRnKCW/SoR7XeyFi9erOXL6uxsVHffPONOjs7lZmZ2Sf7BgDAlrDDi
CTdddddduuuuu7p9ra6uLnQHgwapvLxc5eXlJ7KrqEodlhrfidxYoVuvfee/XLX/5SOTk5GjZsmJ566imtX78+2Mfpd8B5zauv
QoUNRrwoAgGgaoN9Nkzs2V+lJ6XLI0e3rDjnkSfIod2xuxPcdFxcnv//
9TScmSVxu/3a/PmzRGvDQCAvjSgw4jL6VLVLVNV13rHw7kBz+ufKaSrmckb+2JSMjQ+vXr9fOnTvVv1tam8ePH68MPPP9TatWv1y
Se
f6MEHH9QHH3wQMubKK6/U6tWrtXr1am3btk133HH9u3bF/HaAAD0SwM6jEhSwfkFWjVzldKSokLa05PStKLa05PStWrmKhWcXx
CV/d577
71yuVyaMGGCRooapfz8fBUUFKiwsFDZ2dn6/PPPNXfu3JAxP/3pT1VcXKyioiJdccUVOvPMMzV16tS01AcAQF9xGNv31/aCz+e
T2+1We3v7Mc8cOXjwoHbs2KFx48YpISHhhPfhD/i1rnGdWva3KnHV5ZYqnLH5kZlReRERUFqm5BABAOv7n99FO6ALW/sjldEXo9lo
AA
NA7A/40DQAAsIswAgAArCKMAAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMPL//H6prk569d
Wuf4/6Qt2
omDJliubPnx+V93U4HHI4HIqPj1daWpquu+461dTURHxfAABBEAmFEUk2NlJEhTZoq3XBD178ZGV3tsWjOnDlqaWnR9u3b9e
///
u+aMGGCfvKTn+i2226zXRoAAMcY8GGkpaUNqbg5t3727qzoagWT27Nl6++23VVVVFVzL2L59u2655RaNGzdOQ4YMmb
nnnqu
qqqqQcd2tpkyfPl2zZ88OaUtMTFRKSorS09a9N16aWX6oknntvCvf/1rLV26VH/4wx8ifoAAAJyEAR1G/H5p3jy8tPtw2f37kT
9lUVVUpJycnuILRotKi9PRopaen67XXXtOWLVtXCcSWtWAq+f/kwIt/vsOqjo/mKZEAWYo1AAUlbjadRUtdqlN3tHOcf/akV4vdO+x
Midx+3W634uLigisYhzooEPB/x43bpzq6+u1cuVKZZw5M2L7BgDgVBN2Ck5zkJnhRa586T36XQ6dc4552jnzpon/V4AAETSgA4jLS2R7R7XeyFi9erOXLl6uxs

VHffPONOjs7lZmZGbH3N8bI4XBE7PoAAIiEAX2aJjU1sv1Ox0oVK3Tvvffqlltu0e9//3t99NFHKikpUWdnZ7CP0+mU+dY5pUO
HDvXq/f1+v/7yl7903LhxEaobAICTNaBXRnJzpfTorotVu7tuxOHoej03N/L7jouLk/+0i1HeffddXXbZZZ07d26wbfv27SFjR
00apZajlmn8fr82b96sqVOnfuf+XnrpJX355Zf68Y9/HIHqAQCInAG9MuJySYdvWPn22YvDP1dWdvWLtIyMDK1fv147d+5UW1u
bx08frw8//FBr167VJ5980gcffFAffPBByJgrr7xSq1ev1urVq7Vt2zbdcccd2rdv3zHv/fXXX8vr9aq5uVnvv/+++FixY0Ntvv
1133HFHr4ILAAB9aUCHEUkqKJBWrZLS0kLb09O72gsKorPfe++Vy6XSxMmTNCoUaOUn5+vg0ICFRYWKjs7W59/W/v//nnIKokk
/fS
nP1VxcbGKiopoxRVX6Mwzz+w2XCxdulSpqak666yzVFBQ0C1btqi6ulrPP/98dA4GAICT4DDfvgjhFOTz+eR2u9Xe3q6kpKSQ1
w4ePKgdO3Zo3LhxSkhIOOF9+P1dd82otHRdI5KbG50okVNZpOYSAADp+J/fRAuWy91oWK7O27AACgdwb8aR0AAGAXYQ
QAAFh
FGAEAAFYRRgAAgFX9JozEwE1BpzzmEABgQ8yHEdf/33979GPTcWIOz6rK2FroN3TDACwKuZv7R0oaJASExO1d+9eDR48WE5nz
OcrK
wKBgPbu3avExEQNGhTzvxYAgBgS8586DodDqamp2rFjh3bt2mW7nR7nW7nJjmdDo1duxYvtkXANCnYj6MSF1fOjd+/HhO1ZykuLg4V
pY
AAH2uX4QRqeuveh5hDgBA7OHPYAAAVhBAAAWEUYAQAAVhFGAACAVYQRAABg1QmFkcWLFysJIoMJCQnKzs7Whgo
beuw7ZcoUO
RyOY7Zrr732hIsGAD9R9hhpLq6WqlpSovL9fGjRs1ceJE5efna8+ePd32r6mpUUtLS3DbsnWLXFKHmzXC6X/vEf//GkiwcAALEv7DD
yzDPPaM6cOSopKdGECROdjzzzisM/Sl/r/XSbKedkW2uu0xmZ7v5/MeIgJZ2QTMmBpU/xNW8rudqBfBz5+RHDH/aaBdu32SZBXZ3
aqvr
+/Veyxbtkw/+clPdNppp/XYXp/ZdPYoZtbqXzuttTOeuulmazaZAwEWAANE5pYGnUsdBZ3EouERpKHtw5vcvzRGI2beXqjCYopGC
+TAAA9PbPUcXT33HW6vn35JHd9nnmHkRyndU+QOn/L6wM6X0RbK1kgxYRPtVB1G2EmQNmEQVwzkGRk=3JDbX3PhVZ+
qq+ v73bMf/3XfyknJod33nmnkpOT9f3vf1+PPfaY/H5/OLsGAAD9VFinadra2uT3+5WcnBzSnpycrG3btnU75q9//+avefPNN3Xjjj
XrjjTf06aefau7cuTp06Hd6PRoaGOJ07gzz6fL5wyAQBADIn63TSBQECjR4/Wv/7rv2rSpEkqLCZU/fffryVLLvQ4pqK
iQm63O7h5PJ50lwkAACwJK4yMHDlSLpdLra2tIe2tra1KSUnpdkxqaqrOOeccuVyuyNv5558558vr9erzs7ObscsWrRI7e3two2q
SmcMgEAQAwJK4zExcVpoqRJqq2tDbYFAgHV1YYqJyen2zE/+MEP9OmnnyoE5yfGYK6rHx8UpKSgrZAA
BA/xT
2aZrSoItXbpUL73okrZu3a077rhDBw4cUElJjiSSpqKhIixYtDCva/44479MUXX2jevHn65JNPtHr1aj322GO68847kI3cUAAAAgZ
0X9nJHCwkLt3btXZWVl8nq9q9ysMM1J01a4lXtTY2NsrpPJB6P1q5dq5q5dq5///Oe66KKKLlJaWpnnz5nmnBggWROwoAABCzwn7OiA
o
8ZwQAgNgTleeMAAAARBphBAAAWEUYAQAAVhFGAACAVYQRAABg1QmFkcWLFysJIoMJCQnKzs7Whgo
beuz740svyuFwhGwJGwJCQknXDAAAOhfwg
4j1dXVKi0
tVXl5uTZu3KiJEycqPz9fe/bs6XFM9fmMUlKSWlpagtuubXbtOqmgAANB/hB1GnnnmmGc2ZMocIQ6SrvzHMQ6

HQ
ykpKcEtOTn5pIoGAAD9R1hhpLOzUwoNDcrLyzvyBk6n8vLyVF9f3+O4r776SmeccYY8Ho+mTZumP//5z8fdTodHh3w+X8gG
AAD
6p7DCSFtbm/x+/zErG8nJyfJ6vd2OOffcc7V8+XK9/vrr+u1vf6tAIKDLLrtMzc3NPe6noqJCbrc7uHk8nnDKBAAAMSTqd9Pk5
OSoqKhImZmZuuKKK1RTU6NRoobp17/+dY9jFi1apPb29uDW1NQU7TIBAIAlg8LpPHLkSLlcLrW2toaot7a2KiUlpVfvMXjwYF1
88cX69NNPe+wTHx+v+Pj4cEoDAAAxKqyVkbi4OE2aNEm1tbXBtkAgoNraWuXk5PTqPfx+vzZt2qTU1NTwKgUAAP1SWCsjkl
RaW   qri4mJNnjxZWVlZqqys1IEDB1RSUiJJKioqUlpamioqKiRJDz/8sC699FKdffbZ2rdvn5566int2rVLt956a2SPBAAAxKSwwoh
hYaH27t2rsrIyeb1eZWZmas2aNcGLWhsbG+VoHllw+fLLLZzVnzhx5vV6NGDFCkyZNonvvvacJEyZE7igAAEDMchhjOoivovP5
5Pb7VZ7e7uSkpJslwMAAHqht5/ffDcNAACwijACAACsIowAAAAAwqggAADAKsIIAACrCCMAAMAqwggAALCKMAIAAKwijAAAAAKsIIwAAwCrCC
AAAsIo
wAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAA
sIowAgAAr
CKMAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwCrCCAAAsIowAg
AArCKMAAA
AqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALDqhMLI4sWLlZGRoYSEBGVnZ2vDhg29GrdixQLA5Nnz
79RHYLA
AD6obDDSHV1tUpLS1VeXq6NGzdq4sSJys/P1549e447bufOnbr33nuVm5t7wsUCAID+J+ww8swzz2jOnDkqSnRhAkTtGTJEi
U
mJmr58uU9jvH7/brxhv10EMP6cwzzzzmPYVBnzXtttnbi8nmUrH8X6NGjdJJdvhc2XdC/Sgw82yU3lbYN9hdiuXZLbH3EcYXfH5 ... (truncated in view)
H fL5fCEbAADon8IKI21tbfL7/UpOTg5pT05Oltfr7XbMO++8o2XLlmnpoqW93k9FRYXcbndw83g84ZQJAABiSFTvptm/f79uvvl
mLV26VCNHjuz1uEWLFqm9vTq4NTU1RbFKAABg06BwO08cOVIuloutraoh7a2trUpJSTmm//bt27Vz505dd911wbZAZINC140G
D9
PHHH+uss846Zlx8fLzi4+PDKQ0AAMSosFZG4uLiNGnSJNW1gbbAoGAamtrlZOTcoz/8847T5s/8U3K6//npNnTpV
H33
oEadfAABAeCsjklRaWqri4mJNnjxZWVlZqqys1IEDB1RSUiJJKioqUlpamioqKpSQkKDvf//7IeOHDx8uScsoAwCAgSnsMFJYW
Ki9e/eqrKxMXq9XmZmZWrNmTfmZmZWrNmTfC11sbGRjmdPNgVAAD0jsMYY2wX8V18Pp/cbrfa29uVlJRkuxwAANALvf38frfa29uVlJRkuxwAANALvf38fgkDAABYRRgB
AAB
WEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAA
BYRRgBAABWEUYAA
IBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBA
ABWEUYAAIBVhBE
AAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQRAABgFWEEAABYRRgBAABWEU
YAAIBVhBEAAGDVC
YWRxYsXKyMjQwkJCcrOztaGDRt67FtTU6PJkydr+PDhOu2oo5o5SZmamXX375hAsGAAD9S9hhpLq6WqWlpSovL9fGjRs1ceJE5
ef
na8+ePd32P/3o03X//fervr5ef/rTn1RSUqKSkhKtXbv2pIsHAACxz2GMMeEMyM7O1iWXXXKLnnntOkhQIBOTxeHT33Xdr4cKF
v   XqPv/mbv9G1116rRx55pFf9fT6f3G632tvblZSUFE65AADAkt5+foe1MtLZ2amGhGhgbl5eUdeQOnU3l5eaqvr//O8cYY1dbW6uO
PP9bf/u3f9tivo6NDPp8vZAMAAP1TWGGkra1fycnMjJjycnyer09ero9jmtvb9fQ0oUMVFxena6v+9Vs8++6x++6MMf9ti/oqJCb
rc7uHk8nnDKBAAAMaRP7qYZNmyYPvr0I33wwQd69NFHVVpaqrq6uh77L1qoSO3t7cGtqampL8oEAAAWDAqn88iRI+Vyud
Ta2hr
S3traqpSUlB7HOZ1OnX322ZKkzMxMbd26VRUFZoyZUq3/ePj4xUfHx9OaQAAIEaFtTISFxenxenSZMmqba2NtgWCARUW1uurnJ
ycX
r9PIBBQRodHOLsGAAD9VFgrI5JUWlqq4uJiTZ48WVlZWaqsrNSBAwdUUlIiSSoqKlJaWpoqKiokdV3/MXnyZJ111lnq6OjG2+
8oZdfflkvvPBCZI8EAADEpLDDSGFhfofbu3auysysjJ5vV5lZmZzZzo1wYtaGxsb5XQeWXA5cOCA5s6dq+bmZgoZMkTnnXeefvvb3
6qwsDByRwEAAGJW2M8ZsYHnjnjAAAEHui8pwRAACASCOMAAAAAqwgjAADAKsIIAACwijACAACsIowAAACrCCMAAMAqw
ggAALCKMAI
AAKwijAAAAKsIIwAAwCrCCAAAsIowAgAArCKMAAAAqwgjAADAKsIIAACwijACAACsGmS7AAADmN8vrVsntbRIbRIqalSbq7k
ctmuC
kAfI4wAsKOmRpo3T2puPtKWni5VVVUkFBfbqAtDnOEoDoO/V1EgzZoQGEUnavburvabGTloArCCMAOhbfn/Xiogxx752uG3+
/K5
+AAYEwgiAvrVu3bErIkczRmpq6uoHYEAgjADoWyotkeoHIOYRRgDordTUy4ccdDEPMIIwD6Vm5u11ozDkf3rzsckscT1Q/AgEAY
A
dC3XK6u23elYwPJ4Z8rK3neCDCAEEYA9L2CAmnVKiktLbQ9Pb2rneeMAMKDzoDYEdBgTRtGk9gBUAYAWCRyyVNNmK7

CgCWcZo
GAABYRRgBAABWEUYAAIBVhBEAAGAVYQQAAFhFGAEAAFYRRgAAgFWEEQAAYBVhBAAAWEUYAQAAVhFGAACAVYQ
RAABg1QmFkcWLF
ysjIoMJCQnKzs7Whgobeuy7dOlS5ebmasSIERoxYoTy8vKO2x8AAAwsYYeR6upqlZaWqry8XBs3btTEiROVn5+vPXv2dNu/rq5
Os2bNoltvvaX6+np5PB5dffXV2r179okXDwAAYp/DGGPCGZCdna1LLrlEzz33nCQpEAjI4/Ho7rvv1sKFC79zvN/v14gRI/Tcc
8+pqKioV/vo+Xxyu91qb29XUlJSOOUCAABLevv5HdbKSGdnpxoaGpSXl3fkDZxO5eXl6Tq4nNsYz+aaG9FDV9emM4CnhQ+Xw6x6VrrjgSwdYbJiq8qn3m8E4PV5tr5T0/s0oqBq/GkJFCxsXtmhPP1pPyLVq7qlvZVSgc0xFCYfmWq8aTuKSBUfL9VjQMftgxmNJnWvTGRDdE2azm82mQxMyLTRKnpHeJ7xOiuGupS9yA03Q1fXGYtNXLa9bbD9/9lPz1j/f/Lh6n1kRHyWVd/RJJRUUF1jBWQt08nNuTfuAt/JCzrtKHt7Z5WQYs16r/xrr9Xeb/m
oP /7jP5SQkNBjvoWLFqm9vT24NTU19WGVAACgLwoKp/PIkSPlcrnU2toaot7a2qqUlJTjjn366af1+OOP6w9/+IMuuuii4/aNj49
XfHx8OKUBAIAYFdbKSFxcnCZNmqTa2tpgWyAQUG1trXJycnoc98QTT+jRRRxx7RmjVrNHny5BOvFkC/4vdLdXXSq692/ev3264
Ig
A1hrYxIUmlpqYqLizV58mRlZWWpsrJSBw4cUElJiSSpqKhIaWlpqqiokCQ98cQTKisroyuvvKKMjIzgtSVDhw7VoKFDI3g0AGJ
JTY0ob57U3HykLT1dqqqSCgrs1QWg74UdRg0LC7V3716VlZXJ6/UqMzNTa9asCV7U2tjYKKfzyILLCy+8oM7OTs2YMSPkfcrLy
/Uv//IvJ1c9gJhUUyPNmCF9+8ECu3d3ta9aRSABBpKwnzNiA88ZAfoPv1/KyAhdETmaw9G1QrJjh+Ry9WlpACIsKs8ZAYCTtW5
dz0FE6lotaWrq6gdgYCCMAOhTLS2R7Qcg9hFGAPSp1NTI9gMQ+wgjAPpUbm7XNSEOR/evOxySx9PVD8DAQBgBoKdcrq7b
d6VjA
8nhnysruXgVGEgIIwD6XEFB1+27aWmh7enp3NYLDERhP2cEACKh0ECaNq3rrpmWlq5rRHJzWREBBiLCCABrXC5pyhTbVQ
CwjdM
0AADAKsIIAACwijACAACsIowAAACrCCMAAMAqwggAALCKMAIAAKwijAAAAKsIIwAAwKqYeAKrMUaS5PP5LFcCAAB66/
Dn9uHP8
Z7ERBjZv3+/JMnj8ViuBAAAhGv//v1yu909vu4w3xVXTgGBQECfffaffZhgobJse3v3N8gPH5fPJ4PGpqalJSUpLtcv015rpvMM9
9g3nuG8xzKGOM9u/frzFjxsjp7PnKkJhYGXE6nUpPT7ddxiklKSmJSmJX/Q+wlz3Dea5bzDPfYN5PuJ4KyKHcQErAACwijACAACsI
0zEmPj4eJWXlys+Pt52Kfoec903mOe+wTz3Deb5xMTEBawAAKD/YmUEAABYRRgBAABWEUYAAIBVhBEAAGAVYeQU9MU
XX+jGG29
UUlKShg8frltuuUVfffXVcccPHhQd955p773ve9p6NCh+vGPf6zW1tZu+37++edKTo+Xw+HQvn37onAEsSEa8/zHP/5Rs2bNk

[redacted]

[redacted]

vL09/+ctfonkIMSGS83zooCEtWLBAF154oU477TSNGTNGRUVF+uyzz6J9GDEhor/TR7v99tvlcDhUWVkZ4apjjMEp55prrjETJ
04077//vlm3bpo5++yzzaxZs4475vbbbzcej8fU1taaDz/8oFx66aXmsssu67bvtGnTzN/93d8ZSebLL7+MwhHEhmjM87Jly8z
PfvYzU1dXZ7Zv325efvllM2TIEPPss89G+3BOCStWrDBxcXFm+fLl5s9//rOZM2eOGT58uGltbe22/7vvvmtcLpd58sknzZYtW
8wDDzxgBg8ebDZt2hTs8/jjxu3223+8z//o/zxj38o119/vRk3bpz55ptv+uqwTjmRnud9+/aZvLw8U11dbbZt22bq6+tNVla
WmTRpUl8e1ikpGr/Th9XU1JiJEyeaMWPGmF/96Ko5lkFLNyyYjwXyjyXzwwQfBtt/97nfG4XCY3bt3dztm3759ZvGwea11
14Ltm3dutVIMvX19SF9n3/+eXPFFVeY2traAR1Goj3PR5s7d66ZOnVq5Io/hWVlZZk777wz+LPf7zdjxowxwFRUV3fafOmufb
aaoPasrOzzT/9oz8ZY4wJBAImJSXFPPXUU8HX9+3bZ+Lj482rr74ahSOIDZGe5+s2LDBSDK7du2KTNExKlpz3dzcbNLSos8szmz
ZvNGWecMeDDCKdpTjH19fUaPny4Jk+eHGzLy8uTo+nU+vCxrux3ToNCgQ4cOKS8vL9h23nnnnaezYsaqvrw+2bdmyRQ8//LB+85v
fHPcLiwaCaM7zt7W3W3t3o+vooo+PXPGnqM7OTJUoNITMj9PpVF5eXo/zXF5e/X0/zU19fH9JfkvLz84P9d+zYIa/XG9LH7XYrOzv7uHPeno
rvT3t4uh8Oh4cOHR6TuWBStuQ4EArr55pvii8/8QhdccEFoi08xA/sT6RTk9Xo1evTokLZBwbwbp9NNPl9fr7XFMXFzcMf/TSE5
ODo7p6OjQrFmz9NRTT2ns2LFRqT2WRGuev+29997Tb3/7W/XW1brvttojUfSpra2uT3+9XcnJySPWSPvx5sfr9R63fzWSmu8
w8e1IIFCzRr1qwB/WVvoZrrJ554Q4MGDdLPfazyBcdowgjfWThwoVyOBzH3bZt2xa1/S9gatEjnn3Ptnn
zZk2bNk3l5eW6+uqr+2SfwMk6dOiQZs6cKWOMnXnjhBdvl9DsNDQ2qqqrSiy++KIfDYbucU8Yg2wUMFPcfcc49mz5593D5nnnnnmmU
lJStGfPnpD2//3f/9UXX3yhlJSUbselpkpKSs7NT+/btC/mrvbSpjmzTff+NTNLIkSN1/1/33366GHHHjrBIzu
12J7nw7Zs2aKrrrpKt912mx544IETOpZYM3LkSLlcLrW0tIS0t7S0yNhIETOpZYM3LkSLlcrmPu4upufg5LSUk5bv/D/7a2ti01NTWkT2ZmZgSrjx3RmOfDDgeRXbt26
co33xzQqyJSdOZ63bp1rNnT8gKtd/v1z333zKPPKykrt3LkzsgcRK2xhy+s/PDDD4Nta9eu7dWFlatXrw62bdq2NcrR/FVkD6
//dRs2rQpuC1fvtxIMu+9916PV/bHumjPd3xm5sxk9erT5xS9+EbEZDDBoEVlZWWu66K/iz3+vZyd4/PV4X3Z9GaZ2OM2bx5sxk9erT5xS9+EboDOEVlZWWu+66K/iz3+83aWlpx73Y7x/+4R9Cnnyc
o65gPXpp58Ovt7e3s4FrBGeZ2OM6ezsNNOnTzcXXHCB2bNnT3QKj2traQ/xdv2rTJjBkzxixYsMBs27YtegdyiiOMnIK
uueYac/HFF5v169ebd955x5x77rlm1/ofnZ955x4wfPz7kltPm5mZz7rnnmvXr1wfbbbr/9djN27Fjz5tvmg8//NDk7aYyJz
jxv2rTJjB0tytx0002mpaUluA2U/U/7mvWLHCxMfHmxdffNFs2bLF3HbbbbWb48OHG6/UaY4y5+eabzs+eabzcKFC4P93333n
Pm61bt5ry8vJub+od0dPny4ef318f/vQnM23aNG7jtfA8d3Z2muuvv96cddZZpq6uLuuvvvv96kp6ebjz76OR3t6Ojw8ow8xniqi8Tv9bdxNQxg5JX3++
edm1qxZZujQ0SYpKcmUlJSY/fv3B1/fvB1/sWOHkWTeutuYNs333xj5s6daoMLjPggj0gZ8Znn8vJyI+mYmY7Yw
zzujDI7Pr2WefNWPHjjVxcXEmKyvLvP/++P/++8XXXFCB2v27+kx5hKOCnMw6rm4lbLEYNLdRr+eLBBDNxUXlwjrm/3PGg5r1Y4WvuLvP/++8HXrrjiClNcXBzSf+KXleacc84xcXFZBdzMRLiLCrV6rRVGc8Oe
HPVVVeZjz/+uC8O5ZQWyXk+/Lve+3Xb07/9AFenf6W8jjBjjMOb/Lx4AAACwgLtpAACAVYQRAABgFWEEAABYRRgBAABWEEAABYRRgBAABWEEQAAYVvof7dPPnadPx2MAAAAASUVORK5CYII=",
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"# Visualize \n",
"plt.plot(result_Kc, 'ro', label='Kc')\n",
"plt.plot(result_tauI, 'go', label='tauI')\n",
"plt.plot(result_tauD, 'bo', label='tauD')\n",
"\n",
"#plt.xlabel('Kc, tauI, tauD');\n",
"#plt.legend((result_Kc, result_tauI, result_tauD), ('Kc', 'tauI', 'tauD'))\n", "\n",
"plt.legend(loc='upper left')\n",
"#pylab.ylim(-1.5, 2.0)\n",
"plt.show()"
]
},
{
"cell_type": "code",

```
"execution_count": 79,
"metadata": {},
"outputs": [],
"source": [
 "import numpy as np\n",
 "import matplotlib.pyplot as plt\n",
```

```
 "execution_count": 80,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "application/vnd.jupyter.widget-view+json": {
     "model_id": "aa927c4c0bb847219a82772a0dd9daa4",
     "version_major": 2,
     "version_minor": 0
    },
    "text/plain": [
     "interactive(children=(FloatSlider(value=0.24446804821491241, description='Kc', max=0.7334041446447372, min=-0.…"
    ]
   },
   "metadata": {},
   "output_type": "display_data"
  },
  {
   "data": {
    "text/plain": [
     "<function __main__.pidPlot(Kc, tauI, tauD)>"
    ]
   },
   "execution_count": 80,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "n = 100 # time points to plot\n",
  "tf = 50.0 # final time\n",
  "SP_start = 2.0 # time of set point change\n",
  "\n",
  "def process(y,t,u):\n",
  "    Kp = 4.0\n",
  "    taup = 3.0\n",
  "    thetap = 1.0\n",
  "    if t<(thetap+SP_start):\n",
  "        dydt = 0.0  # time delay\n",
  "    else:\n",
  "        dydt = (1.0/taup) * (-y + Kp * u)\n",
```

```
"    return dydt\n",
"\n",
"def pidPlot(Kc,tauI,tauD):\n",
"    t = np.linspace(0,tf,n) # create time vector\n",
"    P= np.zeros(n)          # initialize proportional term\n",
"    I = np.zeros(n)         # initialize integral term\n",
"    D = np.zeros(n)         # initialize derivative term\n",
"    e = np.zeros(n)         # initialize error\n",
"    OP = np.zeros(n)        # initialize controller output\n",
"    PV = np.zeros(n)        # initialize process variable\n",
"    SP = np.zeros(n)        # initialize setpoint\n",
"    SP_step = int(SP_start/(tf/(n-1))+1) # setpoint start\n",
```

```
    "        y = odeint(process,y0,ts,args=(OP[i-1],))  # compute next step\n",
    "        y0 = y[1]                  # record new initial condition\n",
    "        # calculate new OP with PID\n",
    "        PV[i] = y[1]               # record PV\n",
    "        e[i] = SP[i] - PV[i]       # calculate error = SP - PV\n",
    "        dt = t[i] - t[i-1]         # calculate time step\n",
    "        P[i] = Kc * e[i]           # calculate proportional term\n",
    "        I[i] = I[i-1] + (Kc/tauI) * e[i] * dt  # calculate integral term\n",
    "        D[i] = -Kc * tauD * (PV[i]-PV[i-1])/dt # calculate derivative term\n",
    "        OP[i] = P[i] + I[i] + D[i] # calculate new controller output\n",     "        \n",
    "    # plot PID response\n",
    "    plt.figure(1,figsize=(15,7))\n",
    "    plt.subplot(2,2,1)\n",
    "    plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')\n",
    "    plt.plot(t,PV,'r:',linewidth=2,label='Process Variable (PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,2)\n",
    "    plt.plot(t,P,'g.-',linewidth=2,label=r'Proportional = $K_c \\; e(t)$')\n",
    "    plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\\frac{K_c}{\\tau_I} \\int_{i=0}^{n_t} e(t) \\; dt $')\n",
    "    plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c \\tau_D \\frac{d(PV)}{dt}$')
\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,3)\n",
    "    plt.plot(t,e,'m--',linewidth=2,label='Error (e=SP-PV)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.subplot(2,2,4)\n",
    "    plt.plot(t,OP,'b--',linewidth=2,label='Controller Output (OP)')\n",
    "    plt.legend(loc='best')\n",
    "    plt.xlabel('time')\n",
    "    \n",
    "Kc_slide = result_Kc\n",
    "tauI_slide = result_tauI\n",
    "tauD_slide = result_tauD\n",
    "wg.interact(pidPlot, Kc=Kc_slide, tauI=tauI_slide, tauD=tauD_slide)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
```

"language": "python",
"name": "python3" },
"language_info": {
"codemirror_mode": {

```
  "pygments_lexer": "ipython3",
  "version": "3.11.6"
 }
},
"nbformat": 4,
"nbformat_minor": 4
}
```

N. ITCLab-13

Di ITCLab ke-13 ada tiga kode yaitu, kode Arduino, Python, dan Notebook Jupyter.

```
#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();
}
```

Kode ini adalah firmware untuk sebuah mikrokontroler berbasis Arduino. Firmware ini mengontrol perangkat keras yang tampaknya memiliki fungsi pemanas, pembaca suhu, dan LED. Berikut adalah penjelasan terperinci dari berbagai bagian kode:

1. **Konstanta dan Pin Setup**
    - Konstanta
        - `vers`: Versi firmware (`1.04`).
        - `baud`: Baud rate untuk komunikasi serial (`115200`).
        - `sp`: Separator untuk perintah serial (`spasi`).
        - `nl`: Terminator untuk perintah serial (`\n`).
    - Pin Setup
        - Pin Sensor Suhu (T1, T2): `pinT1` (34) dan `pinT2` (35).
        - Pin Output (Q1, Q2): `pinQ1` (32) dan `pinQ2` (33).
        - Pin LED: `pinLED` (26).
    - PWM Properties
        - Frekuensi PWM ditetapkan ke 5000 Hz.
        - Resolusi PWM untuk semua kanal adalah 8 bit (nilai maksimum 255).

2. **Variabel Global**
    - `Buffer`: Buffer untuk menyimpan data input serial.
    - `cmd`: Perintah yang diterima melalui serial.
    - `pv`: Nilai dari perintah serial (dalam bentuk angka).
    - `level`: Tingkat intensitas LED (0-100%).
    - `Q1`, `Q2`: Nilai output ke pin Q1 dan Q2 (maksimum 25).
    - `n`: Jumlah sampel untuk pengukuran suhu.

3. **Fungsi Utama**
    `parseSerial()`
    - Membaca data dari serial hingga karakter terminator (`nl`).
    - Memisahkan perintah (`cmd`) dan data nilai (`pv`) menggunakan pemisah (`sp`).
    - Data nilai dikonversi menjadi float.

`dispatchCommand()`

Menginterpretasikan perintah dan menjalankan fungsi terkait:

- `Q1`, `Q2`: Mengontrol output PWM untuk pin Q1 dan Q2 dengan nilai maksimum 25.
- `T1`, `T2`: Membaca nilai suhu dari sensor (T1 atau T2), mengonversinya ke derajat Celsius.
- `V` atau `VER`: Menampilkan versi firmware.
- `LED`: Mengontrol intensitas LED (0-100%).
- `X`: Menghentikan semua aktivitas (set Q1 dan Q2 ke nol).

`checkTemp()`

- Membaca suhu dari sensor T1 dan T2.
- Jika suhu melebihi batas (`batas_suhu_atas = 59` derajat Celsius), mematikan pemanas dengan mengatur Q1 dan Q2 ke nol.

4. Fungsi `setup()`

- Menginisialisasi komunikasi serial pada baud rate 115200.
- Menyiapkan fungsi PWM untuk pin Q1, Q2, dan LED.
- Mengatur semua output PWM awal ke 0.

5. Fungsi `loop()`

- Menjalankan tiga fungsi utama secara berulang:
  - `parseSerial()`: Membaca dan memproses perintah serial.
  - `dispatchCommand()`: Mengeksekusi perintah.
  - `checkTemp()`: Memeriksa suhu untuk perlindungan perangkat.

```
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
```

Kode ini adalah sebuah Python script yang digunakan untuk mengontrol perangkat Arduino melalui komunikasi serial. Perangkat Arduino yang dikontrol kemungkinan terhubung ke sistem fisik seperti laboratorium miniatur yang mengukur suhu, mengontrol pemanas, atau menjalankan LED, seperti pada eksperimen kontrol. Berikut adalah penjelasan detail dari setiap bagian kode:

1. Import dan Setup Library

```
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
```

- `serial`: Library untuk komunikasi serial.
- `list_ports`: Digunakan untuk mendeteksi port yang tersedia untuk perangkat yang terhubung.
- `time`: Digunakan untuk memberikan jeda waktu.
- `numpy`: Digunakan untuk memproses data numerik, seperti mengatur data dalam array dan menyimpannya ke file.
- `try-except`: Memastikan bahwa jika library serial tidak terinstal, akan diinstal secara otomatis menggunakan pip.

2. Kelas iTCLab

Kelas utama ini bertanggung jawab untuk mengontrol perangkat yang terhubung ke Arduino.

2.1. Konstruktor `__init__`

```
def __init__(self,        port=None, baud=115200):
```

- `port`: Port komunikasi serial. Jika tidak diberikan, akan dicari otomatis.
- `baud`: Baud rate untuk komunikasi (default: 115200).
- Membuka koneksi ke Arduino, menunggu selama 3 detik agar komunikasi siap, dan memberikan pesan sukses.

2.2. Metode `findPort`

```
def findPort(self):
```

- Mencari port yang sesuai dengan perangkat Arduino berdasarkan VID:PID (Vendor ID dan Product ID).
- Mendukung berbagai tipe Arduino seperti Uno, HDuino, Leonardo, dan ESP32.
- Jika port tidak ditemukan, memberikan panduan untuk menemukannya di berbagai sistem operasi (Windows, macOS, Linux).
-

2.3. Properti dan Fungsi Kontrol

- Properti `T1` dan `T2`: Mengembalikan suhu dari sensor 1 dan sensor 2.

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

- Fungsi `LED`, `Q1`, `Q2`: Mengontrol perangkat keras seperti LED dan pemanas (Heater 1 & Heater 2) dengan nilai PWM (0-100%).

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

2.4. Fungsi `save_txt`

```
def save_txt(self,t,u1,u2,y1,y2,sp1,sp2):
```

- Menyimpan data eksperimen ke file teks (`data.txt`).
- `t`: Waktu.
- `u1`, `u2`: Input ke pemanas.
- `y1`, `y2`: Suhu dari sensor.
- `sp1`, `sp2`: Set point (nilai target suhu).

2.5. Komunikasi Serial

- Fungsi `read`: Mengirim perintah ke Arduino dan membaca respon.

```
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", "")
```

- Fungsi `write`: Mengirim perintah dengan argumen (misalnya nilai PWM).

```
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", "")
```

- Fungsi `build_cmd_str`: Membuat string perintah dengan format tertentu untuk dikirim ke Arduino.

2.6. Fungsi Tambahan

- `version`: Mengembalikan versi firmware dari Arduino.
- `stop`: Menghentikan semua aktivitas.
- `close`: Menutup koneksi serial dengan Arduino.

```
import itclab import
numpy as np import time
import matplotlib.pyplot as plt from
scipy.integrate import odeint import random
# Machine Learning - Building Datasets and Model
# Impor `Sequential` dari` keras.models` from
keras.models import Sequential

# Impor `Dense` dari` keras.layers` from
keras.layers import Dense

# Inisialisasi konstruktor model =
Sequential()

# Tambahkan lapisan masukan
model.add(Dense(2, activation='sigmoid', input_shape=(2,)))
# Tambahkan satu lapisan tersembunyi model.add(Dense(3,
activation='sigmoid'))

# Tambahkan lapisan keluaran
model.add(Dense(3, activation='sigmoid'))

# Data Latih.
X = np.array([
    [1, 1],
    [0.4, 1.2],
    [1.2, 0.1],
    [1, 0.1]
])

# Label untuk Data Latih. y =
np.array([
    [0.25, 4.31, 0.20],
    [0.2, 4.1, 0.1],
    [0.1, 4.0, 0],
    [0.1, 4.0, 0]
])

# Bentuk keluaran model model.output_shape

# Ringkasan model model.summary()

# Konfigurasi model model.get_config()

# Buat daftar semua tensor bobot  model.get_weights()
model.compile(loss='binary_crossentropy',              optimizer='adam',
```

[redacted]

[redacted]

```
# 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
```

[redacted]

[redacted]

```
    # calculate the error      error =
sp-pv      d_error = sp-pv_last
    delta_error = (error - d_error)

    outDL = model.predict(np.array([[error,delta_error]]))
    Kc = outDL[0,0]      tauI =
outDL[0,1]      tauD =
outDL[0,2]

    # 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 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/%
```

[redacted]

[redacted]

```
    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()
#a.encode('utf-8').strip()#modification error
# 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[60:] = 45.0
Tsp1[360:] = 30.0
Tsp1[660:] = 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
```

[redacted]

[redacted]

[redacted]

[redacted]

```
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])
```

[redacted]

[redacted]

```
        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)
```

[redacted]

1. **Pembelajaran Mesin**

   Pada bagian ini, model pembelajaran mesin dibuat menggunakan Keras dengan struktur jaringan saraf tiruan (Artificial Neural Network). Model ini digunakan untuk memperkirakan parameter PID seperti Kc, tauI, dan tauD berdasarkan input dari kesalahan setpoint (error) dan perubahan kesalahan (delta_error).

   - Lapisan Input: Memiliki 2 neuron dengan fungsi aktivasi sigmoid.
   - Lapisan Tersembunyi: 3 neuron dengan fungsi aktivasi sigmoid.
   - Lapisan Keluaran: 3 neuron untuk memberikan output parameter PID.
   - Data Pelatihan (Training Data)
     - Input (X) adalah pasangan nilai untuk sistem kontrol.
     - Target/output (y) adalah nilai parameter PID.

2. **Pengendalian PID**

   Bagian ini berisi implementasi pengendali PID tradisional dan pengendali berbasis pembelajaran mesin:

   - PID Manual: Menggunakan parameter tetap seperti Kc, tauI, dan tauD.
   - PID Berbasis Deep Learning: Parameter PID dihitung oleh model pembelajaran mesin berdasarkan input dari kesalahan.

   Fungsi PID mengontrol keluaran (op) yang mengatur pemanas untuk menjaga suhu pada setpoint yang diinginkan.

3. **Simulasi Sistem Termal**

[redacted]

Sistem termal disimulasikan menggunakan model berikut:

- Persamaan Energi: Model termal nonlinier dengan parameter fisik seperti massa, luas permukaan, dan koefisien perpindahan panas.
- FOPDT Model: Model linier orde pertama untuk sistem termal dengan parameter seperti waktu tunda (thetaP) dan waktu konstanta (tauP).

Kegunaan

Kode menghasilkan grafik untuk menampilkan suhu aktual, setpoint, dan keluaran pengontrol dalam waktu nyata.

O. ITCLab-14

Di ITCLab ke-14 ada tiga kode yaitu, kode Arduino, Python, dan Notebook Jupyter.

```
#include <Arduino.h>

// constants
const          = "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();
}
```

Kode di atas merupakan firmware untuk sebuah perangkat berbasis Arduino yang menggunakan modul atau shield yang disebut **iTCLab Shield**. Firmware ini bertujuan untuk mengontrol perangkat keras tertentu seperti LED, sinyal output PWM, dan membaca sensor suhu (Thermocouple atau sensor analog lainnya). Berikut adalah penjelasan mendetail dari setiap bagian kode:

1. Konstanta dan Variabel Global

- Konstanta:
  - `vers`: Versi firmware ("1.04").
  - `baud`: Baud rate komunikasi serial (115200).
  - `sp` dan `nl`: Karakter pemisah dan terminator untuk parsing perintah serial.
  - Pin Numbers: Pin yang digunakan untuk sinyal (T1, T2, Q1, Q2, LED).
  - PWM Properties: Properti PWM seperti frekuensi (5000 Hz), kanal PWM (0-2), dan resolusi (8-bit).
- Variabel Global:
  - `Buffer`: Buffer untuk membaca data serial.
  - `cmd`: Perintah yang diterima dari input serial.
  - `pv`: Nilai yang diambil dari perintah serial.
  - `Q1`, `Q2`: Nilai yang ditulis ke pin Q1 dan Q2. `n`: Jumlah sampel untuk pembacaan suhu.

2. Fungsi parseSerial()

Fungsi ini membaca data dari Serial hingga karakter terminator (`\n`), lalu memisahkan perintah (`cmd`) dan data (`pv`).

3. Fungsi `dispatchCommand()`

Fungsi ini memproses perintah yang diterima dan mengambil tindakan sesuai jenis perintah:

- Q1 & Q2: Mengatur keluaran PWM ke pin Q1 dan Q2, dengan nilai diatur dalam rentang 0-25.
- T1 & T2: Membaca sensor suhu yang terhubung ke pin T1 dan T2, mengonversinya ke derajat Celcius, lalu mencetak hasilnya.
- VER: Menampilkan versi firmware.
- LED: Mengatur tingkat kecerahan LED dalam rentang 0-100%.
- X: Mematikan semua output (Q1, Q2).

4. Fungsi checkTemp()

Fungsi ini memeriksa suhu dari sensor T1 dan T2 untuk memastikan bahwa suhu tidak melebihi batas maksimum (`batas_suhu_atas` = 59°C). Jika suhu melebihi batas:

- Nilai Q1 dan Q2 diatur ke 0.

- Output PWM ke pin Q1 dan Q2 dimatikan.
- Suhu yang melampaui batas dicetak ke serial.

5. **Fungsi `setup()`**

   Fungsi inisialisasi Arduino:
   - Menyiapkan komunikasi serial pada baud rate yang ditentukan.
   - Mengatur kanal PWM untuk pin Q1, Q2, dan LED.
   - Melampirkan kanal PWM ke pin terkait.
   - Memastikan semua output dimatikan pada awal.

6. **Fungsi `loop()`**

   Fungsi utama yang dijalankan terus-menerus:
   - `parseSerial()`: Membaca dan mem-parsing perintah dari serial.
   - `dispatchCommand()`: Memproses dan menjalankan perintah yang diterima.
   - `checkTemp()`: Memantau suhu dan mematikan output jika suhu terlalu tinggi.

7. **Fungsi PWM dan Pengontrolan**

   PWM digunakan untuk mengontrol keluaran pin Q1, Q2, dan LED:
   - Nilai PWM dihitung berdasarkan input (`pv`) dan dikonversi ke skala 8-bit (0255).
   - Metode `ledcWrite(channel, value)` digunakan untuk menulis nilai PWM ke pin tertentu.

8. **Fungsi Utama dan Fitur:**
   - Kontrol Output (Q1/Q2): Mengontrol output dengan nilai rentang 0-25.
   - Pengukuran Suhu (T1/T2): Membaca nilai analog dan mengonversinya ke suhu.
   - Kontrol LED: Mengatur kecerahan LED dengan rentang 0-100%.
   - Proteksi Suhu: Memastikan perangkat tidak melampaui suhu yang aman.

```
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
```

Kode ini adalah implementasi dalam Python untuk berkomunikasi dengan perangkat Arduino yang terhubung melalui port serial. Kode ini ditujukan untuk mengontrol perangkat keras, seperti modul termal (heaters), LED, dan membaca data suhu dari sensor. Berikut adalah penjelasan detail setiap bagian:

1. Impor Modul

```
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
```

- `serial`: Digunakan untuk komunikasi serial antara Python dan perangkat Arduino.
- `numpy`: Digunakan untuk memproses dan menyimpan data.
- Jika `pyserial` tidak tersedia, modul akan diinstal secara otomatis.

2. Kelas `iTCLab`

Kelas ini menyediakan antarmuka untuk mengontrol perangkat keras yang terhubung ke Arduino.

- `__init__` Method
  - Memulai koneksi dengan Arduino melalui port serial.
  - Mencari port yang terhubung menggunakan metode `findPort`.
  - Menginisialisasi koneksi serial dengan baud rate default 115200 dan waktu tunggu 2 detik.
- `findPort` Method
  - Mendeteksi port yang terhubung dengan perangkat Arduino.

- Mencocokkan Vendor ID (VID) dan Product ID (PID) untuk berbagai tipe Arduino.
- Jika tidak ditemukan, meminta pengguna untuk memasukkan port secara manual.

3. **Kontrol dan Data**
   - Properti Suhu (`T1` dan `T2`)
     - `T1` dan `T2`: Membaca data suhu dari sensor melalui perintah `read`.
     - Mengembalikan nilai dalam tipe data float.
   - Kontrol Output (`Q1`, `Q2`, `LED`)
     - Metode ini mengontrol keluaran PWM (Pulse Width Modulation) dari heater atau LED:
       - `Q1`: Kontrol heater 1.
       - `Q2`: Kontrol heater 2.
       - `LED`: Kontrol intensitas LED.
   - Perintah Tambahan
     - `stop`: Mengirim perintah X untuk menghentikan operasi.
     - `version`: Membaca versi firmware Arduino.

4. **Penyimpanan Data**
   - `save_txt` Method
     - Menyimpan data ke file `data.txt` dalam format CSV.
     - Data yang disimpan:
       - Waktu (`t`)
       - Heater 1 dan Heater 2 output (`u1`, `u2`)
       - Suhu 1 dan 2 (`y1`, `y2`)
       - Set point untuk suhu 1 dan 2 (`sp1`, `sp2`).

5. **Komunikasi Serial**
   - `read` Method
     - Mengirim perintah ke Arduino dan membaca respon.
     - Respon di-decode dari byte ke string.
   - `write` Method
     - Mengirim perintah dengan parameter (misalnya nilai PWM).

- Menggunakan metode `build_cmd_str` untuk membangun format string perintah.

- `build_cmd_str` Method
  - Membentuk string perintah dengan format:
    Contoh: `Q1 50\n` untuk mengatur Heater 1 ke 50%.

6. Penanganan Koneksi

- `close` Method
  - Menutup koneksi serial dengan Arduino secara aman.

[redacted]

```
import itclab import
numpy as np import time
import matplotlib.pyplot as plt from
scipy.integrate import odeint import random
from paho.mqtt import client as mqtt_client # Machine
Learning - Building Datasets and Model
# Impor `Sequential` dari` keras.models` from
keras.models import Sequential

# Impor `Dense` dari` keras.layers` from
keras.layers import Dense

# Inisialisasi konstruktor model =
Sequential()

# Tambahkan lapisan masukan
model.add(Dense(2, activation='sigmoid', input_shape=(2,)))
# Tambahkan satu lapisan tersembunyi model.add(Dense(3,
activation='sigmoid'))

# Tambahkan lapisan keluaran
model.add(Dense(3, activation='sigmoid'))

# Data Latih.
X = np.array([
    [1, 1],
    [0.4, 1.2],
    [1.2, 0.1],
    [1, 0.1]
])

# Label untuk Data Latih. y =
np.array([
    [0.25, 4.31, 0.20],
    [0.2, 4.1, 0.1],
    [0.1, 4.0, 0],
    [0.1, 4.0, 0]
])

# Bentuk keluaran model model.output_shape

# Ringkasan model model.summary()

# Konfigurasi model model.get_config()

# Buat daftar semua tensor bobot  model.get_weights()
model.compile(loss='binary_crossentropy',              optimizer='adam',
```

[redacted]

[redacted]

[redacted]

[redacted]

```
# 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
```

[redacted]

[redacted]

```
    # calculate the error      error =
sp-pv      d_error = sp-pv_last
    delta_error = (error - d_error)

    outDL = model.predict(np.array([[error,delta_error]]))
    Kc = outDL[0,0]      tauI =
outDL[0,1]      tauD =
outDL[0,2]

    # 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 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/%
```

[redacted]

[redacted]

```
    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
# Connect to MQTT Broker for Monitoring broker
= 'broker.hivemq.com' port = 1883
client_id = f'python-mqtt-{random.randint(0, 1000)}'
 def connect_mqtt():
    def on_connect(client, userdata, flags, rc):         if rc == 0:
print("Connected to MQTT Broker!")         else:             print("Failed to
connect, return code %d\n", rc)

    client = mqtt_client.Client(client_id)
client.on_connect = on_connect
client.connect(broker, port)     return client
 client = connect_mqtt()
client.loop_start() Connected
to MQTT Broker!
######################################################
# Do not adjust anything below this point            #
######################################################
# Connect to Arduino a =
itclab.iTCLab()
#a.encode('utf-8').strip()#modification error
# Turn LED on print('LED
On')
a.LED(100)

# Run time in minutes run_time = 15.0
```

[redacted]

[redacted]

[redacted]

[redacted]

```
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=(10,7)) 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 \
```

```
        # 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)
```

```
# 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')     raise

a.close()
```

Kode ini merupakan kombinasi dari beberapa teknologi seperti Machine Learning, PID Controller, Simulasi Model Nonlinear dan FOPDT, serta komunikasi MQTT dengan perangkat keras melalui Arduino (iTCLab). Ini adalah penjelasan rincinya:

1. **Machine Learning**

   Bagian ini melibatkan pembuatan model Neural Network sederhana menggunakan Keras. Model ini digunakan untuk memprediksi parameter PID (Kc, τI, τD) berdasarkan data pelatihan.

   - Model Neural Network memiliki:
     - 1 lapisan masukan (2 neuron),
     - 1 lapisan tersembunyi (3 neuron),
     - 1 lapisan keluaran (3 neuron untuk menghasilkan parameter PID).
   - Data pelatihan (`X` dan `y`) dirancang untuk mengajarkan model memprediksi parameter PID berdasarkan error dan perubahan error.

2. **PID Controller**

- Controller PID digunakan untuk mengontrol suhu dengan parameter:
  - Proportional (P): Koreksi berdasarkan error saat ini.
  - Integral (I): Koreksi berdasarkan akumulasi error.
  - Derivative (D): Koreksi berdasarkan perubahan error.
- Ada dua metode:
  - PID Manual: Parameter PID diatur secara manual (`pid`).

- PID Deep Learning: Parameter PID diprediksi oleh model neural network (`pid_dl`).

3. **Model FOPDT dan Energi Nonlinear**
   - Model FOPDT (First Order Plus Dead Time):
     - Menggunakan parameter Kp, τP, dan θP untuk mensimulasikan sistem linier.
   - Model Energi Nonlinear:
     - Menghitung perubahan suhu berdasarkan energi konveksi, radiasi, dan daya pemanas.

4. **MQTT dan Arduino**
   - MQTT digunakan untuk komunikasi dengan broker (HiveMQ) untuk monitoring.
   - Arduino (iTCLab) digunakan sebagai perangkat keras untuk membaca suhu (`T1`, `T2`) dan mengontrol pemanas (`Q1`, `Q2`).

5. **Simulasi Utama**

   Simulasi mencakup langkah-langkah berikut:
   - Mengatur setpoint suhu:
     - Suhu target berubah pada waktu tertentu untuk menguji performa PID.
   - Menghitung output PID:
     - Baik manual maupun menggunakan model deep learning.
   - Memperbarui suhu:
     - Simulasi dilakukan menggunakan model energi nonlinear dan FOPDT.
   - Visualisasi:
     - Plot suhu aktual, setpoint, dan output kontrol.

6. **Kode Inti**

   Beberapa bagian penting dari kode:
   - Neural Network

```
model = Sequential()

# Tambahkan lapisan masukan
model.add(Dense(2, activation='sigmoid', input_shape=(2,)))
# Tambahkan satu lapisan tersembunyi model.add(Dense(3,
activation='sigmoid'))

# Tambahkan lapisan keluaran
model.add(Dense(3, activation='sigmoid'))

# Data Latih.
X = np.array([
    [1, 1],
    [0.4, 1.2],
    [1.2, 0.1],
    [1, 0.1]
])

# Label untuk Data Latih. y =
np.array([
    [0.25, 4.31, 0.20],
    [0.2, 4.1, 0.1],
    [0.1, 4.0, 0],
    [0.1, 4.0, 0]
])

# Bentuk keluaran model model.output_shape

# Ringkasan model model.summary()

# Konfigurasi model model.get_config()

# Buat daftar semua tensor bobot  model.get_weights()
model.compile(loss='binary_crossentropy',
optimizer='adam',               metrics=['accuracy'])

model.fit(X, y,epochs=10, batch_size=1, verbose=1)
```

- PID Controller Manual

```
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]
```

- PID Controller dengan Deep Learning

```
def pid_dl(sp,pv,pv_last,ierr,dt):

    # calculate the error     error =
sp-pv     d_error = sp-pv_last
    delta_error = (error - d_error)

    outDL = model.predict(np.array([[error,delta_error]]))
    Kc = outDL[0,0]     tauI =
outDL[0,1]     tauD =
outDL[0,2]

    # 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 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]
```

- Simulasi Nonlinier

```
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
```

P. Kode yang Ditingkatkan (Blink Pola Morse)

```
#define LED 2

void dot() {
 digitalWrite(LED, HIGH);  delay(200); //
Durasi titik  digitalWrite(LED, LOW);
delay(200);
}
void dash() {
 digitalWrite(LED, HIGH);  delay(600); //
Durasi garis  digitalWrite(LED, LOW);
delay(200);
}
void setup() {
 pinMode(LED, OUTPUT);
} void loop()
{
 // SOS: ... --- ...   dot();
dot(); dot(); // S
delay(400);
 dash(); dash(); dash(); // O   delay(400);
 dot(); dot(); dot(); // S
 delay(1000); // Jeda sebelum ulangi
}
```

Kode ini adalah sebuah program Arduino sederhana yang membuat lampu LED berkedip dalam pola kode Morse untuk huruf SOS, yaitu tiga titik (...), tiga garis (---), dan tiga titik lagi (...). Berikut adalah penjelasan rinci:

1. Pendefinisian Pin LED

```
#define LED 2
```

- `#define` digunakan untuk mendefinisikan konstanta.
- `LED` diatur ke pin digital nomor 2 pada papan Arduino, tempat LED terhubung.

2. Fungsi dot()

```
void dot() {
 digitalWrite(LED, HIGH);  delay(200); //
Durasi titik  digitalWrite(LED, LOW);
delay(200);
```

```
}
```

- Fungsi ini membuat LED menyala selama 200 milidetik (titik), lalu mati selama 200 milidetik.

3. **Fungsi dash()**

```
void dash() {
 digitalWrite(LED, HIGH);  delay(600); //
Durasi garis  digitalWrite(LED, LOW);
delay(200);
}
```

- Fungsi ini membuat LED menyala selama 600 milidetik (garis), lalu mati selama 200 milidetik.

4. **Fungsi setup()**

```
void setup() {
 pinMode(LED, OUTPUT);
}
```

- Fungsi ini dijalankan sekali saat perangkat Arduino dihidupkan.
- `pinMode(LED, OUTPUT)` mengatur pin nomor 2 sebagai output, memungkinkan pin mengontrol LED.

5. **Fungsi `loop()`**

```
void loop() {
 // SOS: ... --- ...  dot();
dot(); dot(); // S
delay(400);
 dash(); dash(); dash(); // O  delay(400);
 dot(); dot(); dot(); // S
 delay(1000); // Jeda sebelum ulangi
}
```

- Fungsi ini terus-menerus dijalankan setelah `setup()`.
- Pola kode Morse untuk SOS diimplementasikan: Tiga titik (`dot(); dot(); dot();`) untuk S. Jeda antar huruf sebesar 400 milidetik.
   - Tiga garis (`dash(); dash(); dash();`) untuk O. Jeda antar huruf sebesar 400 milidetik.
   - Tiga titik (`dot(); dot(); dot();`) untuk S.
   - Jeda antar pola sebesar 1000 milidetik sebelum mengulangi.

Kegunaan

- Program ini cocok untuk demonstrasi sinyal darurat SOS menggunakan kode Morse pada LED.
- SOS adalah sinyal universal untuk permintaan bantuan.