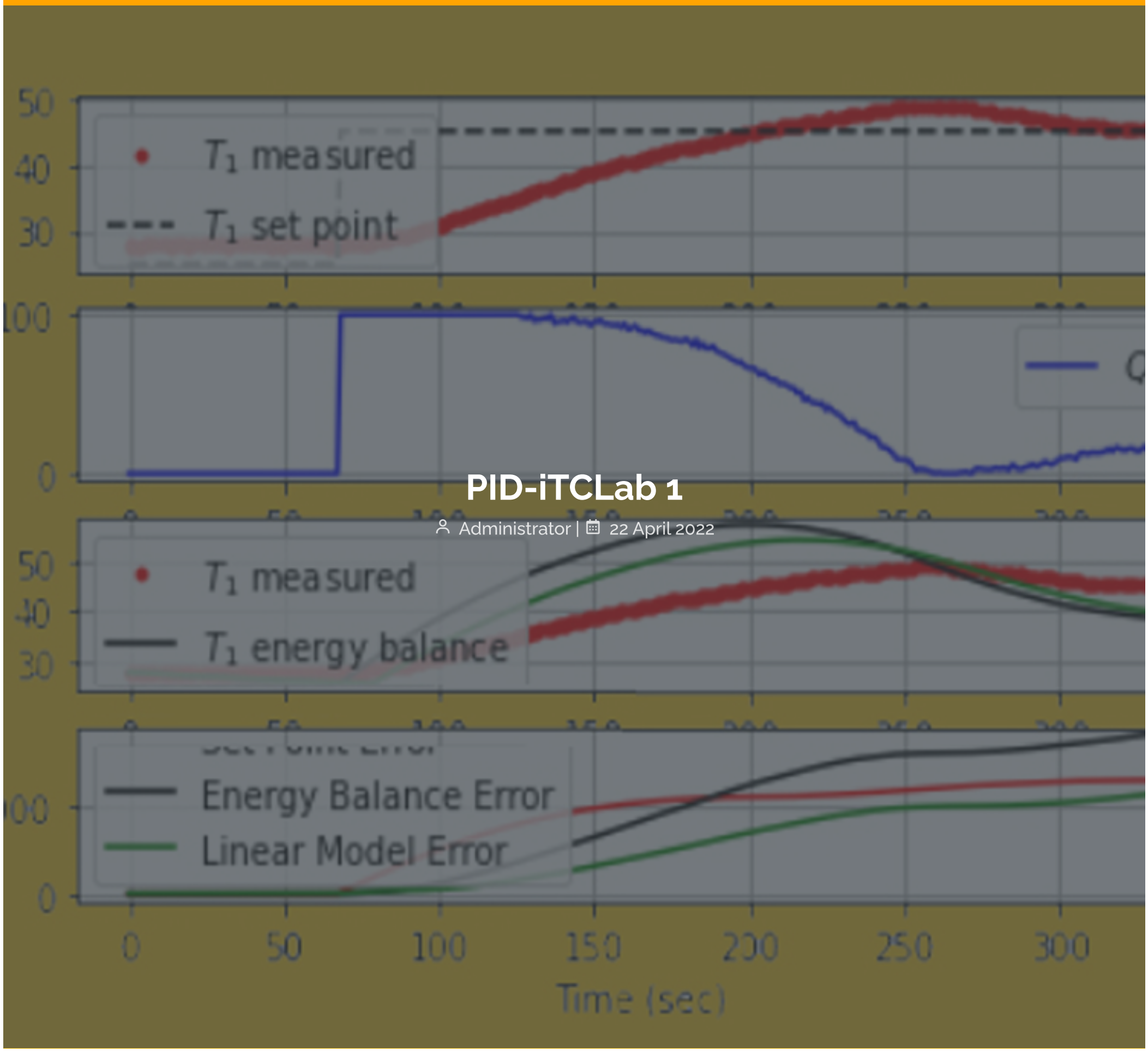
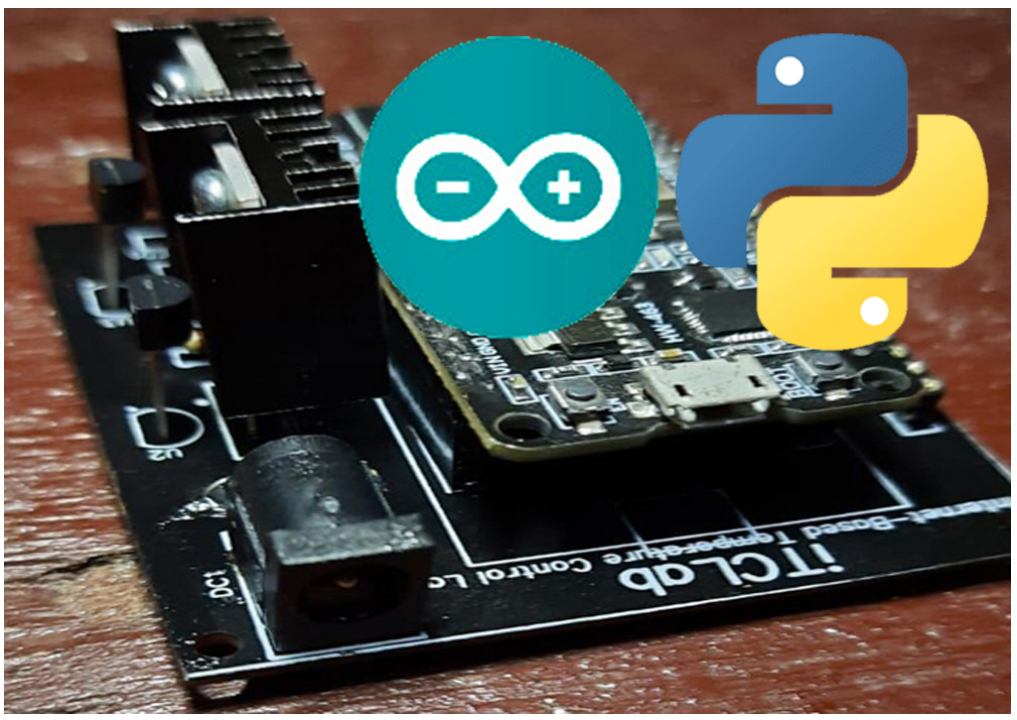
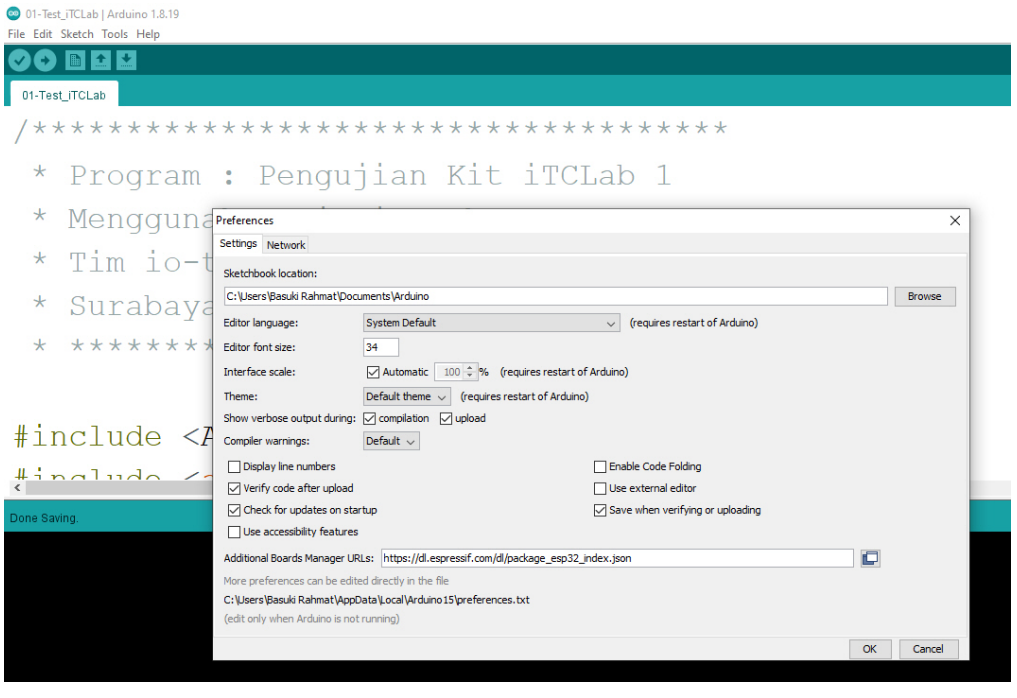


PID-iTCLab 1

Administrator | 22 April 2022

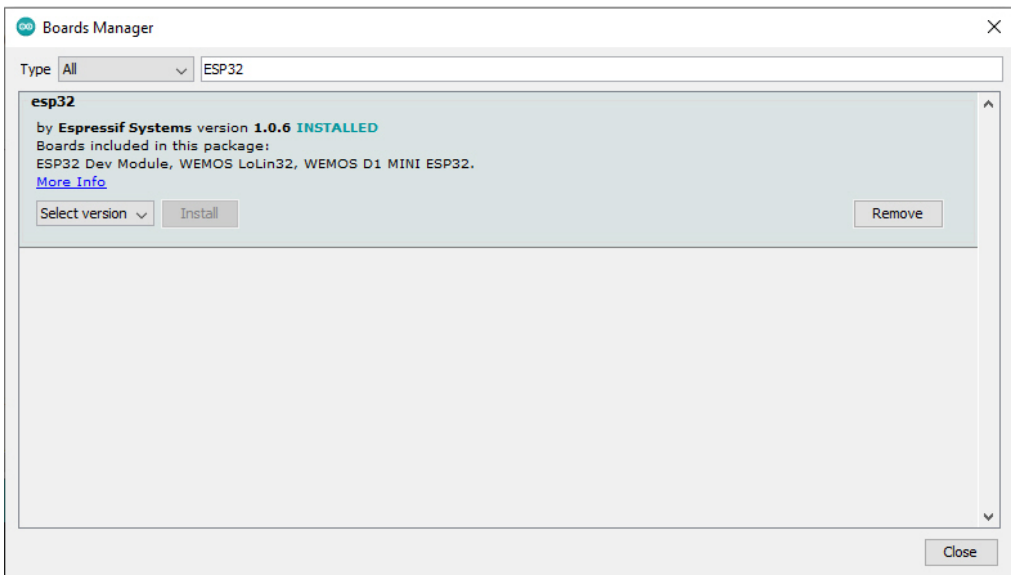
Pemrograman Kendali PID-iTCLab dengan Menggunakan Bahasa Pemrograman Arduino+Python





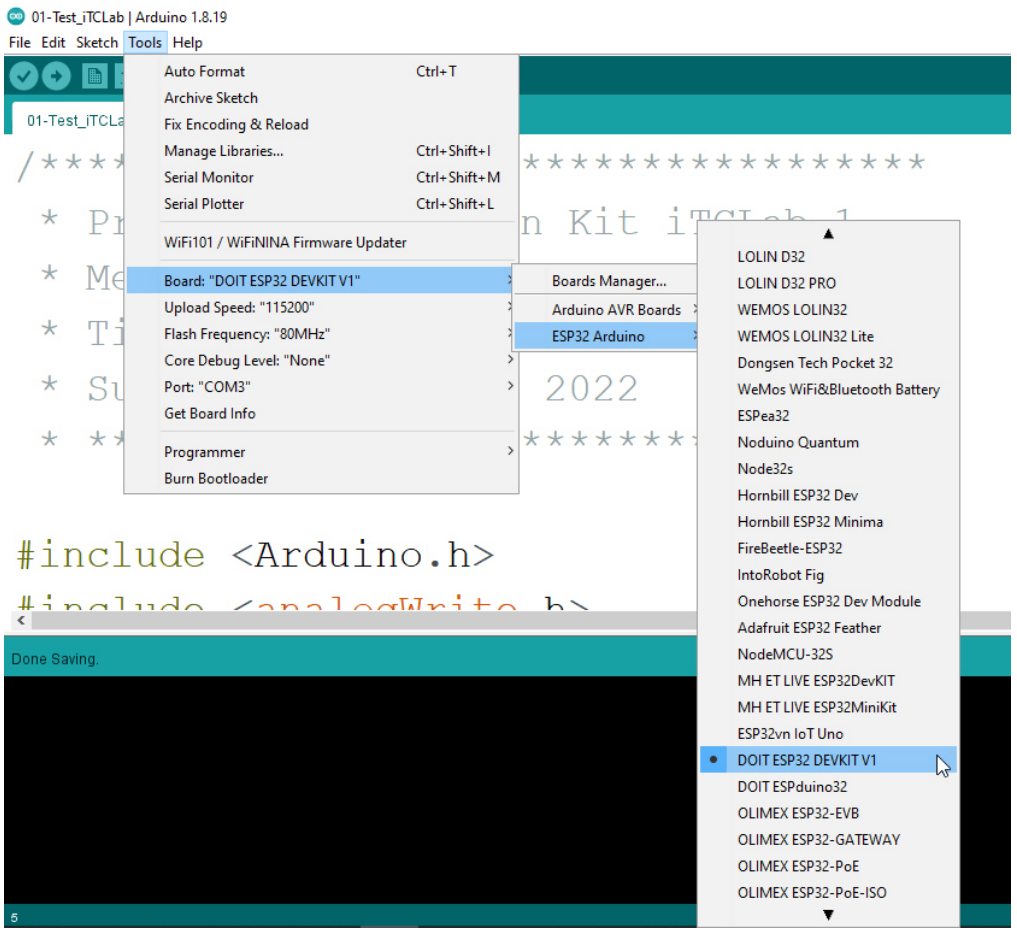
Pengaturan Board.

Kit iTCLab menggunakan Mikrokontroller ESP32. Jika belum muncul. Untuk menggunakan pertama kali , silahkan diinstall ESP32 di Board Manager.



Pilihan Board.

Selanjutnya, silahkan dipilih Board: DOIT ESP32 DEVKIT V1.



File Program yang dibutuhkan

File-file yang dibutuhkan agar bisa dijalankan pengujian Kit iTCLab menggunakan Bahasa Pemrograman Python Jupyter Notebook, yaitu:

1. Program yang harus diupload di Kit iTCLab (silahkan klik-kanan Save link as) (**05-iTCLab_PID.ino**).
2. File Program Modul (silahkan klik-kanan Save link as) **itclab.py**.
3. Script Program untuk menjalankan proses Pengujian Kit iTCLab menggunakan Python Jupyter Notebook (silahkan klik-kanan Save link as) (**iTCLab-PID-01.ipynb**).

```
  John Hedengren, Modified
  Oct 2017
  Basuki Rahmat, Modified
  April 2022

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

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

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

#include <Arduino.h>

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

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

//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 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;
```

```
  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();
}
```

File Program Modul itclab.py

File Program Modul itclab.py, harus ada, dan diletakkan di folder kerja yang sama dengan Program Python Jupyter Notebook.

```
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
```

```
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
```

Script Program untuk Pengujian Kit iTCLab (iTCLab-PID-01.ipynb)

Berikut ini Script Program untuk Pengujian Sistem Kendali PID dengan Kit iTCLab menggunakan Python Jupyter Notebook berikut ini. Dalam script Python Jupyter Notebook ini, hasil pengendalian PID-iTCLab dibandingkan dengan hasil dari **Pemodelan Dinamis Keseimbangan Energi (Energy Balance)**. Berikut ini codingnya.

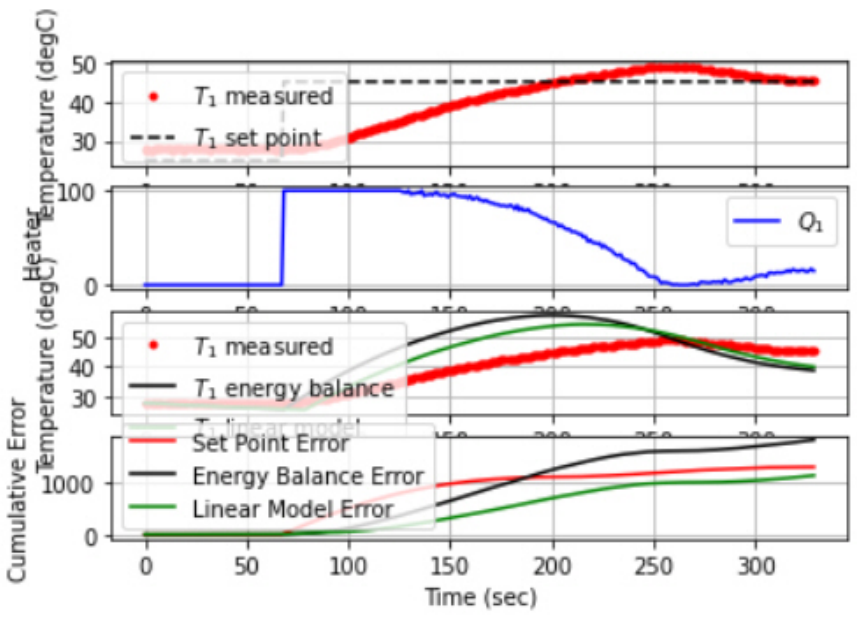
```
#########################################################
# Use this script for evaluating model predictions   #
# and PID controller performance for the TCLab       #
# Adjust only PID and model sections                 #
#########################################################

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

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

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

    # Temperature State
    T = x[0]

    # Nonlinear Energy Balance
    dTdt = (1.0/(m*Cp))*(U*A*(Ta-T) \
            + eps * sigma * A * (Ta**4 - T**4) \
            + alpha*Q)
    return dTdt

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

# Connect to Arduino
a = itclab.iTCLab()

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

# Run time in minutes
run_time = 15.0

# Number of cycles
```

```
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
        t = time.time()
        dt = t - prev_time
        prev_time = t
        tm[i] = t - start_time

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

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

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

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

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

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

        # 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)
```

```
        plt.legend(loc=2)
        ax=plt.subplot(4,1,4)
        ax.grid()
        plt.plot(tm[0:i],error_sp[0:i],'r-',label='Set Point Error')
        plt.plot(tm[0:i],error_eb[0:i],'k-',label='Energy Balance Error')
        plt.plot(tm[0:i],error_fopdt[0:i],'g-',label='Linear Model Error')
        plt.ylabel('Cumulative Error')
        plt.legend(loc='best')
        plt.xlabel('Time (sec)')
        plt.draw()
        plt.pause(0.05)

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

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

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

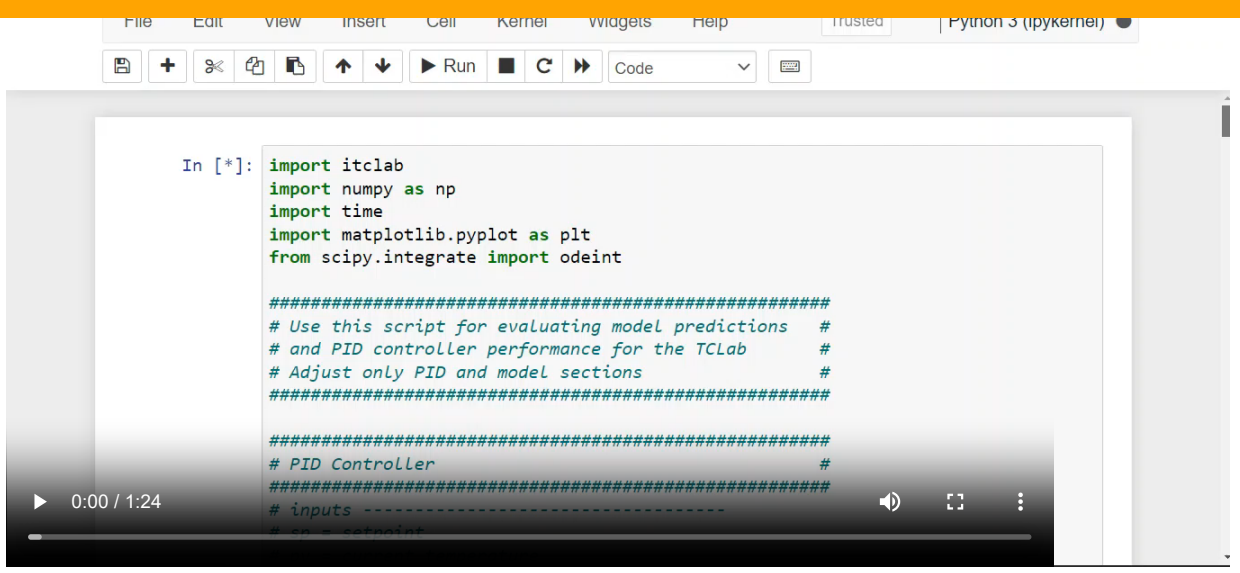
Silahkan dijalankan script program di atas, dalam kondisi Kit iTCLab terhubung ke laptop (PC). Power Adaptor silahkan ditancapkan ke colokan listrik. Maka jika berhasil, tunggu selama proses pemanasan bekerja. Hasilnya kurang lebih seperti terlihat pada gambar berikut.

```
329.9  45.00  45.20  17.22  -2.00  18.32   0.90
```




```
 331.0  45.00  45.26  15.13  -2.60  18.26  -0.53
Shutting down
Arduino disconnected successfully
Arduino disconnected successfully
```

Hasil pengendalian PID-iTCLab di atas dibandingkan dengan hasil dari Pemodelan Dinamis Keseimbangan Energi. Hasilnya sangat bagus. Hasil pengendalian PID-iTCLab menuju Set Point yang diharapkan. Pemodelan Dinamis Keseimbangan Energi juga menunjukkan kinerja yang bagus, ditunjukkan dengan pola dinamika yang mirip. Hasil pengendalian PID-iTCLab diperlihatkan pada video berikut.



Hasil pengendalian PID-iTCLab secara lengkap juga bisa dibaca pada file Pdf berikut : iTCLab-PID-01-JupyterNotebook.pdf.

Riset iTCLab

Kit iTCLab Test 1
Administrator | 10 April 2022

Kit iTCLab Test 2
Administrator | 10 April 2022

Riset Kendali PID Dasar
Administrator | 15 April 2022

Riset PWM iTCLab
Administrator | 16 April 2022

Arduino-Python iTCLab Test
Administrator | 16 April 2022

Arduino-Python PID Test
Administrator | 16 April 2022

Riset PID-iTCLab GUI
Administrator | 16 April 2022

Riset IoT Basic
Administrator | 17 April 2022

Riset IoT On/Off PWM
Administrator | 18 April 2022

Riset PID dengan Arduino
Administrator | 20 April 2022

Riset IoT PID Monitor
Administrator | 20 April 2022

Riset IoT PID Control
Administrator | 03 May 2022

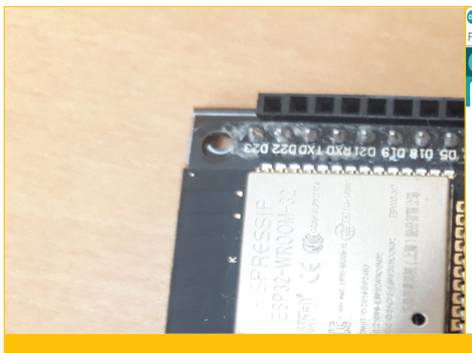
Riset Deep Learning - XOR
Administrator | 22 August 2022

Riset Deep Learning - PID
Administrator | 22 August 2022

Riset Deep - PID - iTCLab
Administrator | 22 August 2022

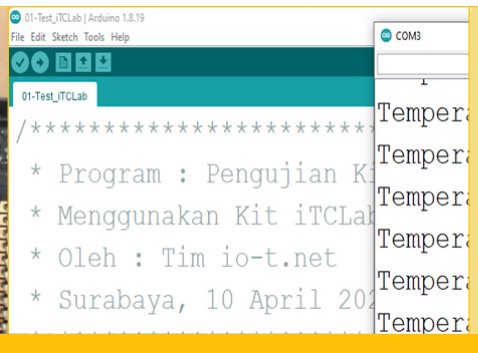
Riset Deep - PID - iTCLab - IoT

Artikel lainnya



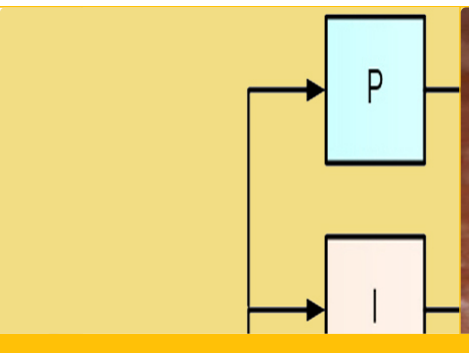
Pengu

PENGUJI



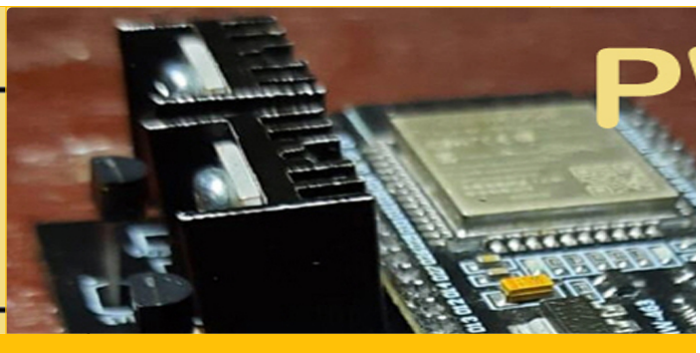
Pengu

PENGUJI



Menger

MENGENAL



Pengujian PWM

PROGRAM PENGUJIAN PWM...