

Basuki Rahmat  
Muljono



# Pemrograman Internet of Things (IoT)

DENGAN ARDUINO DAN PYTHON

Jilid 1



## Tentang Penulis



**Basuki Rahmat**, adalah Dosen Program Studi S2 Teknologi Informasi, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional "Veteran" Jawa Timur. Beliau menerima gelar Sarjana Fisika Bidang Instrumentasi dari Institut Teknologi Sepuluh Nopember Surabaya pada tahun 1995, menerima gelar Magister Teknik Program Instrumenasi dan Kontrol Institut Teknologi Bandung pada tahun 2000, dan menerima gelar Doktor Teknik Elektro Bidang Jaringan Cerdas Multimedia dari Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2018. Minat penelitiannya adalah di bidang komputasi cerdas, kendali cerdas, komputer visi, drone, robotika, pemrograman PHP, arduino dan python.



**Muljono**, adalah Dosen Program Studi Doktor Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro Semarang. Beliau menerima gelar Sarjana Matematika dari Universitas Diponegoro Semarang pada tahun 1996, menerima gelar Magister Komputer dari STTIBI Jakarta pada tahun 2001, dan menerima gelar Doktor Teknik Elektro Bidang Jaringan Cerdas Multimedia dari Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2016. Minat penelitiannya adalah di bidang Kecerdasan Buatan, Data Mining, Pemrosesan Bahasa Alami dan Rekayasa Perangkat Lunak.



Anggota IKAPI  
No. 225 UTE/2021

0858 5343 1992

eurekamediaaksara@gmail.com

Jl. Banjaran RT.20 RW.10

Bojongsari - Purbalingga 53362

ISBN 978-623-129-596-4 (no p. terangkai)

ISBN 978-623-129-597-1 (p.11)



9 78623 205971

# **PEMROGRAMAN INTERNET OF THINGS (IoT) DENGAN ARDUINO DAN PYTHON**

## **JILID 1**

**Basuki Rahmat  
Muljono**



**PENERBIT CV.EUREKA MEDIA AKSARA**

**PEMROGRAMAN INTERNET OF THINGS (IoT) DENGAN  
ARDUINO DAN PYTHON  
JILID 1**

**Penulis** : Basuki Rahmat  
Muljono

**Desain Sampul** : Eri Setiawan

**Tata Letak** : Husnun Nur Afifah

**ISBN** : 978-623-120-596-4 (no.jil.lengkap)  
978-623-120-597-1 (jil.1)

Diterbitkan oleh : **EUREKA MEDIA AKSARA, APRIL 2024**  
**ANGGOTA IKAPI JAWA TENGAH**  
**NO. 225/JTE/2021**

**Redaksi:**

Jalan Banjaran, Desa Banjaran RT 20 RW 10 Kecamatan Bojongsari  
Kabupaten Purbalingga Telp. 0858-5343-1992  
Surel : eurekamediaaksara@gmail.com  
Cetakan Pertama : 2024

**All right reserved**

Hak Cipta dilindungi undang-undang  
Dilarang memperbanyak atau memindahkan sebagian atau seluruh  
isi buku ini dalam bentuk apapun dan dengan cara apapun,  
termasuk memfotokopi, merekam, atau dengan teknik perekaman  
lainnya tanpa seizin tertulis dari penerbit.

## PRAKATA

Segala puji bagi Allah S.W.T yang telah melimpahkan rahmat, hidayah dan pertolongan-Nya sehingga kami dapat menyelesaikan buku yang berjudul "Pemrograman Internet of Things (IoT) dengan Arduino dan Python" Jilid 1. Sholawat serta salam untuk Baginda Nabi Muhammad SAW, semoga kelak kita mendapatkan syafaat beliau di hari akhir.

Semoga buku ini bisa menjadi salah satu Buku Referensi untuk para dosen, mahasiswa, guru, pelajar, dan peneliti, serta siapa saja yang ingin mempelajari dan mengembangkan penelitian tentang Pemrograman Berbasis Internet of Things (IoT).

Kami menyadari bahwa buku ini masih banyak kekurangan. Kritik, saran dan diskusi lebih lanjut, serta peluang kerjasama riset, dan lain-lain, bisa disampaikan melalui alamat email: [basukirahmat.if@upnjatim.ac.id](mailto:basukirahmat.if@upnjatim.ac.id) dan [muljono@dsn.dinus.ac.id](mailto:muljono@dsn.dinus.ac.id). Terimakasih.

Surabaya, Maret 2024

Tim Penulis

## DAFTAR ISI

<b>PRAKATA .....</b>	<b>iii</b>
<b>DAFTAR ISI.....</b>	<b>iv</b>
<b>DAFTAR GAMBAR.....</b>	<b>vi</b>
<b>DAFTAR TABEL .....</b>	<b>x</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
A. Konsep dan Teknologi <i>Internet of Things</i> (IoT).....	1
B. Sistem IoT.....	3
C. Aplikasi IoT.....	6
D. IoT dengan MQTT Broker .....	6
E. IoT Mobile dengan IoT MQTT Panel.....	9
<b>BAB 2 BERKENALAN DENGAN MIKROKONTROLER</b>	
<b>DOIT ESP32 DEVKIT V1 .....</b>	<b>11</b>
A. Hardware DOIT ESP32 DEVKIT V1 .....	11
B. Konfigurasi Pin DOIT ESP32 DEVKIT V1.....	13
C. Pemrograman Mikrokontroler ESP32 .....	15
<b>BAB 3 BERKENALAN DENGAN KIT iTCLAB.....</b>	<b>24</b>
A. Konsep dan Teknologi iTCLab.....	24
B. Bagaimana Mendapatkan Kit iTCLab .....	26
C. Pemodelan Dinamis Sistem Pemanas.....	28
D. Pemrograman Dasar Kit iTCLab dengan Arduino .....	33
E. Pemrograman Dasar Kit iTCLab dengan Python.....	53
<b>BAB 4 REVIEW SISTEM KENDALI PID .....</b>	<b>68</b>
A. Konsep Sistem Kendali PID.....	68
B. Proses Penalaan Parameter PID .....	70
<b>BAB 5 PEMROGRAMAN PID.....</b>	<b>74</b>
A. Pemrograman PID Dasar .....	74
B. Pemrograman PID-iTCLab dengan Arduino.....	77
C. Pemrograman PID-iTCLab dengan Arduino-Python..	84
D. Pemrograman PID-iTCLab dengan Arduino-Python GUI .....	98
<b>BAB 6 PEMROGRAMAN IOT .....</b>	<b>104</b>
A. Review Sistem Dasar IoT .....	104
B. Pemrograman IoT On/Off.....	105

C. Pemrograman Pemantauan PID-iTCLab dengan IoT .....	119
D. Pemrograman Pengendalian PID-iTCLab dengan IoT .....	131
<b>BAB 7 PENUTUP .....</b>	<b>151</b>
<b>DAFTAR PUSTAKA .....</b>	<b>152</b>
<b>TENTANG PENULIS .....</b>	<b>155</b>

## DAFTAR GAMBAR

Gambar 1. 1.	Arsitektur IoT [2].....	2
Gambar 1. 2.	Contoh arsitektur berbasis IoT untuk bidang kesehatan [3] .....	4
Gambar 1. 3.	Contoh penerapan IoT pada Smart City [4] .....	5
Gambar 1. 4.	Sistem IoT [5] .....	6
Gambar 1. 5.	hivemq.com .....	8
Gambar 1. 6.	Free Public MQTT Broker hivemq.....	9
Gambar 1. 7.	IoT MQTT Panel .....	10
Gambar 1. 8.	IoT MQTT Panel siap digunakan.....	10
Gambar 2. 1.	Mikrokontroller DOIT ESP32 DEVKIT V1.....	12
Gambar 2. 2.	Konfigurasi Pin DOIT ESP32 DEVKIT V1 .....	13
Gambar 2. 3.	Konfigurasi Pin mikrokontroller DOIT ESP32 DEVKIT V1 .....	14
Gambar 2. 4.	Download Arduino IDE 2.3.1.....	16
Gambar 2. 5.	Persiapan Instalasi Arduino IDE IDE 2.3.1 .....	17
Gambar 2. 6.	Tekan Next.....	17
Gambar 2. 7.	Lokasi Instalasi .....	18
Gambar 2. 8.	Proses Instalasi .....	18
Gambar 2. 9.	Proses Instalasi Selesai.....	19
Gambar 2. 10.	Download library otomatis tunggu sampai selesai .....	19
Gambar 2. 11.	Arduino IDE 2.3.1 siap digunakan .....	20
Gambar 2. 12.	Pengaturan Board ESP32 .....	20
Gambar 2. 13.	Ketik dan Pilih ESP32.....	21
Gambar 2. 14.	Proses Instalasi Board ESP32.....	22
Gambar 2. 15.	Instalasi Board ESP32 Sukses .....	22
Gambar 2. 16.	Pemilihan Board DOIT ESP32 DEVKIT V1.....	23
Gambar 2. 17.	Board DOIT ESP32 DEVKIT V1 siap digunakan..	23
Gambar 3. 1.	Kit iTCLab Hasil Riset Dosen Kampus Bela Negara.....	25
Gambar 3. 2.	Perbedaan iTCLab dan TCLab Kampus BYU.....	26
Gambar 3. 3.	i-ot.net .....	27
Gambar 3. 4.	iTCLab di i-ot.net .....	27
Gambar 3. 5.	iTCLab di Shopee .....	28

Gambar 3. 6.	Pemanas iTCLab .....	29
Gambar 3. 7.	iTCLab terhubung ke Laptop .....	33
Gambar 3. 8.	Pengaturan File - Preferences .....	34
Gambar 3. 9.	Pengaturan File - Preferences (Lanjutan) .....	34
Gambar 3. 10.	Pilih Port iTCLab .....	35
Gambar 3. 11.	Pilih Upload Speed 115200.....	36
Gambar 3. 12.	LED ESP32.....	37
Gambar 3. 13.	Simpan Program Blink .....	38
Gambar 3. 14.	Program Blink .....	38
Gambar 3. 15.	Proses Upload Sukses.....	39
Gambar 3. 16.	LED berkedip (blink).....	39
Gambar 3. 17.	Gambaran kinerja Kit TCLab BYU .....	40
Gambar 3. 18.	Proses upload Program iTCLab_Testing .....	46
Gambar 3. 19.	Proses upload Program iTCLab_Testing Berhasil	46
Gambar 3. 20.	Tools – Serial Monitor .....	47
Gambar 3. 21.	Hasil Pembacaan Suhu iTCLab .....	48
Gambar 3. 22.	Menuju Batas Atas suhu iTCLab .....	48
Gambar 3. 23.	Diagram Siklus Kerja.....	50
Gambar 3. 24.	Proses upload Program PWM_Testing .....	52
Gambar 3. 25.	Upload Program PWM_Testing selesai .....	52
Gambar 3. 26.	Perubahan Kecerahan dari LED Kit iTCLab Sesuai Siklus Kerja PWM .....	53
Gambar 3. 27.	Proses upload Program arduino_python.ino.....	60
Gambar 3. 28.	Program arduino_python.ino berhasil di-upload	61
Gambar 3. 29.	Hasil pengujian Kit iTCLab dengan Python .....	66
Gambar 3. 30.	Hasil pengujian Kit iTCLab dengan Python Lanjutan.....	67
Gambar 4. 1.	Sistem Kendali PID .....	68
Gambar 4. 2.	Respon Sistem terhadap Masukan Fungsi Step	71
Gambar 4. 3.	Respon Sistem Berbentuk Kurva S.....	71
Gambar 4. 4.	Sistem Lup-Tertutup dengan Pengendali Proporsional .....	72
Gambar 4. 5.	Periode $P_{cr}$ dari Osilasi Keluaran Sistem .....	72
Gambar 5. 1.	Simulasi Penalaan Parameter PID.....	74

Gambar 5. 2.	Contoh hasil proses penalaan nilai parameter Kc, τI dan τD pengendali PID terhadap keluaran sistem.....	77
Gambar 5. 3.	Proses upload program PID_Arduino.ino .....	83
Gambar 5. 4.	Proses upload PID_Arduino.ino sukses.....	83
Gambar 5. 5.	Hasil pengendalian PID pada Kit iTCLab.....	84
Gambar 5. 6.	Proses Upload PID_Python.ino.....	90
Gambar 5. 7.	Proses Upload PID_Python.ino Sukses .....	91
Gambar 5. 8.	Contoh hasil pengendalian PID pada Kit iTCLab menggunakan Python .....	98
Gambar 5. 9.	Penyesuaian dengan Kit iTCLab.....	99
Gambar 5. 10.	Penyesuaian semua TCLab diganti dengan iTCLab.....	100
Gambar 5. 11.	Tampilan Program demo.ipynb.....	100
Gambar 5. 12.	Contoh Hasil Pengendalian Demo PID Python GUI .....	101
Gambar 5. 13.	Copy Program Demo PID untuk PID-iTCLab....	102
Gambar 5. 14.	Penyesuaian dari Demo PID ke PID-iTCLab.....	103
Gambar 5. 15.	Contoh Hasil Pengendalian PID-iTCLab Python GUI .....	103
Gambar 6. 1.	Cara Install Library Arduino.....	112
Gambar 6. 2.	Tombol Install library PubSubClient.....	112
Gambar 6. 3.	Library PubSubClient berhasil diinstall .....	113
Gambar 6. 4.	Proses upload program IoT_OnOff.ino.....	113
Gambar 6. 5.	Upload program IoT_OnOff.ino sukses .....	114
Gambar 6. 6.	Pengaturan IoT MQTT Panel di Ponsel (a) .....	115
Gambar 6. 7.	Pengaturan IoT MQTT Panel di Ponsel (b) .....	116
Gambar 6. 8.	Pengaturan IoT MQTT Panel di Ponsel (c) .....	117
Gambar 6. 9.	Contoh hasil pemantauan dan pengendalian Kit iTCLab secara On/Off via IoT MQTT Panel di Ponsel.....	118
Gambar 6. 10.	Proses upload program IoT_Monitor.ino.....	127
Gambar 6. 11.	Proses upload program IoT_Monitor.ino sukses	127
Gambar 6. 12.	Contoh hasil pemantauan suhu PID-iTCLab melalui tampilan Serial Monitor di laptop.....	128
Gambar 6. 13.	Pengaturan di IoT MQTT Panel (a) .....	129

Gambar 6. 14.	Pengaturan di IoT MQTT Panel (b).....	129
Gambar 6. 15.	Pengaturan di IoT MQTT Panel (c) .....	130
Gambar 6. 16.	Contoh hasil pemantauan PID-iTCLab via IoT .....	131
Gambar 6. 17.	Arsitektur sistem pengendalian suhu menggunakan PID-iTCLab via IoT .....	132
Gambar 6. 18.	Konsep penalaan parameter PID-iTCLab berbasis IoT melalui Ponsel .....	133
Gambar 6. 19.	Proses upload program IoT_Control.ino .....	142
Gambar 6. 20.	Proses upload program IoT_Control.ino sukses.....	143
Gambar 6. 21.	Contoh hasil pengendalian suhu PID-iTCLab melalui tampilan Serial Monitor.....	143
Gambar 6. 22.	Pengaturan di IoT MQTT Panel (a).....	144
Gambar 6. 23.	Pengaturan di IoT MQTT Panel (b).....	145
Gambar 6. 24.	Pengaturan di IoT MQTT Panel (c) .....	146
Gambar 6. 25.	Pengaturan di IoT MQTT Panel (d).....	147
Gambar 6. 26.	Pengaturan di IoT MQTT Panel (e) .....	148
Gambar 6. 27.	Pengaturan di IoT MQTT Panel (f).....	149
Gambar 6. 28.	Contoh hasil pengendalian suhu PID-iTCLab via IoT .....	150

## **DAFTAR TABEL**

Tabel 2. 1. Hardware DOIT ESP32 DEVKIT V1 .....	12
Tabel 3. 1. Karakteristik pemanas iTCLab.....	29
Tabel 4. 1. Aturan Ziegler-Nichols Metode Pertama .....	71
Tabel 4. 2. Aturan Ziegler-Nichols metode kedua.....	73

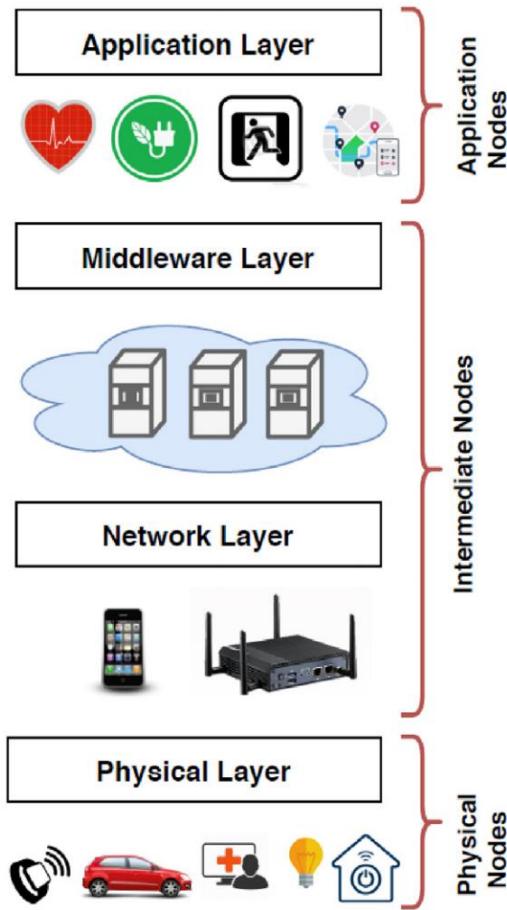
# BAB

# 1

# PENDAHULUAN

## A. Konsep dan Teknologi *Internet of Things* (IoT)

*Internet of Things* (IoT) adalah area yang muncul di mana milyaran objek pintar saling berhubungan satu sama lain menggunakan internet untuk berbagi data dan sumber daya [1]. Teknologi IoT memungkinkan benda-benda di sekitar kita saling terhubung dengan jaringan internet. Dimana setiap benda yang terhubung dengan internet bisa diakses kapan saja dan dimana saja. Contohnya, dari jarak jauh kita bisa menghidup-matikan peralatan di rumah (lampu, televisi, kompor, pemanas, dan lain-lain) selama peralatan terhubung ke *cloud* IoT dan tersedia koneksi internet. Secara umum arsitektur IoT terdiri dari *Application Layer*, *Middleware Layer*, *Network Layer*, dan *Physical Layer*, seperti diperlihatkan pada Gambar 1.1 [2].



Gambar 1. 1. Arsitektur IoT [2]

**Lapisan Aplikasi (Application Layer):** bertujuan untuk menyediakan layanan kepada pengguna akhir. Lapisan ini terdiri dari simpul aplikasi yang menangani logika aplikasi serta semantik data dan presentasi. Simpul ini menerima data dari *middleware* dan memprosesnya tergantung pada persyaratan pengguna akhir dan jenis layanan yang disediakan. Selain itu, lapisan aplikasi mencakup *Application Programming Interface* (API) untuk memfasilitasi komunikasi dengan *middleware* dan antarmuka pengguna yang digunakan pengguna akhir untuk mengakses layanan.

**Lapisan Middleware (Middleware Layer):** untuk memastikan konektivitas dan interoperabilitas dalam ekosistem IoT. Ini terdiri dari simpul menengah yang memproses data yang diterima dari lapisan bawah dan meneruskannya ke lapisan aplikasi.

**Lapisan Jaringan (Network Layer):** untuk mendukung jaringan dan transfer data antar simpul. Lapisan jaringan mengimplementasikan protokol komunikasi yang diperlukan untuk pertukaran data dalam ekosistem IoT.

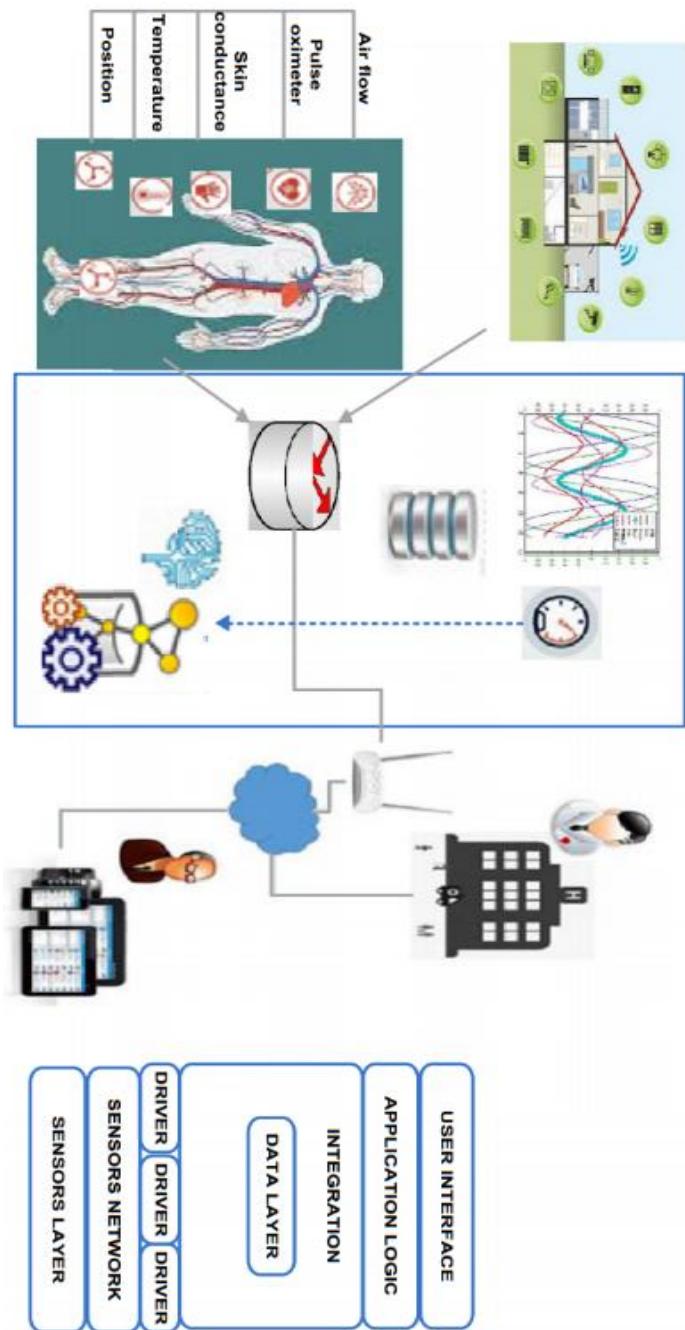
**Lapisan Fisik (Physical Layer):** untuk mengkarakterisasi kemampuan penginderaan dan kontrol dari sistem IoT. Lapisan ini terdiri dari simpul fisik seperti sensor dan aktuator yang merasakan lingkungan dan berinteraksi dengannya dalam menanggapi perubahan atau permintaan pengguna. Node ini menghasilkan sumber daya (merasakan data) yang dilewatkan ke simpul aplikasi melalui jaringan dan lapisan *middleware*.

## B. Sistem IoT

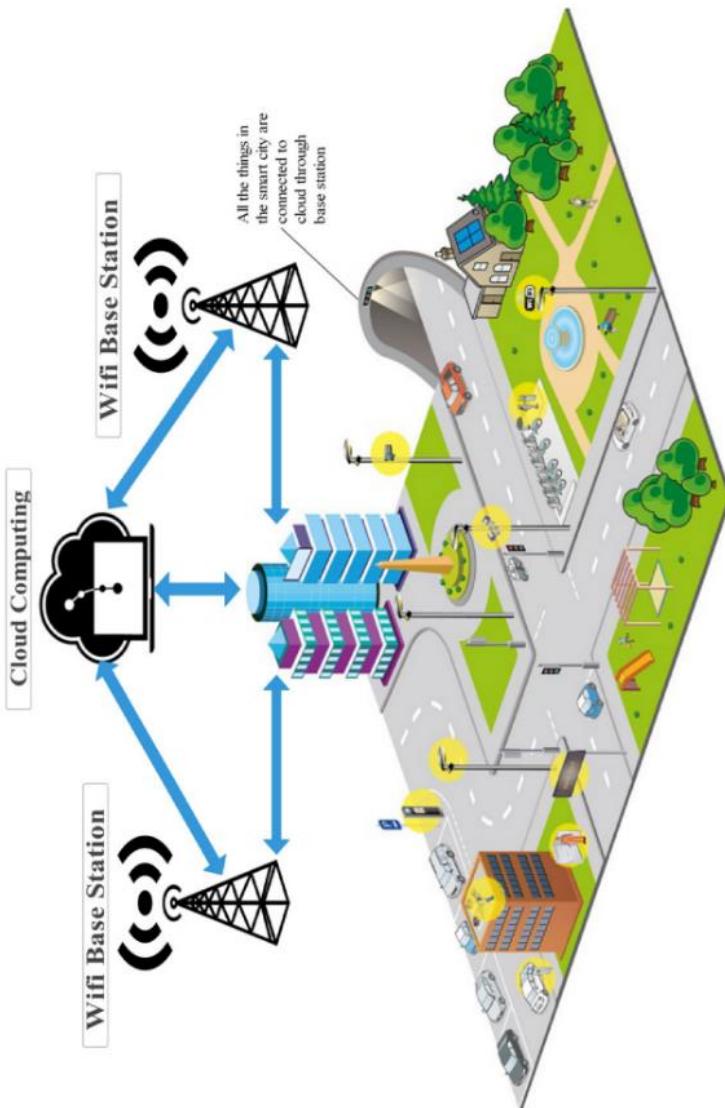
Banyak sekali contoh penerapan teknologi IoT, beberapa contohnya:

1. *Smart Home* (sistem keamanan rumah berbasis internet, dapat mengetahui keadaan rumah serta mengontrol peralatan rumah tangga melalui jaringan internet).
2. *Smart Farming* (sistem pertanian cerdas berbasis internet, untuk pemantauan dan pengendalian kualitas air dan tanah pertanian serta pertumbuhan tanaman melalui jaringan internet).
3. *Internet industry* (pemantauan dan pengendalian peralatan serta proses di industri).
4. Kesehatan (pemantauan kondisi kesehatan seseorang).
5. Transportasi (majemen dan informasi lalulintas).

Gambaran contoh penerapan IoT untuk bidang kesehatan dan *Smart City* diperlihatkan pada Gambar 1.2 dan Gambar 1.3.



Gambar 1. 2. Contoh arsitektur berbasis IoT untuk bidang kesehatan [3]



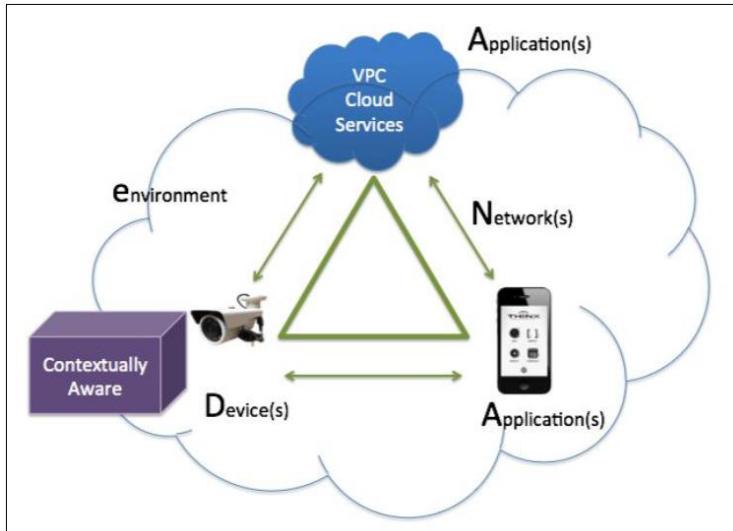
Gambar 1. 3. Contoh penerapan IoT pada Smart City [4]

### C. Aplikasi IoT

Sistem dasar dari IoT terdiri dari 3 hal, yaitu:

1. Hardware/fisik (*Things*).
2. Koneksi internet.
3. *Cloud data center* sebagai tempat untuk menyimpan atau menjalankan aplikasinya.

Masing-masing seperti diperlihatkan pada Gambar 1.4 [5].



Gambar 1. 4. Sistem IoT [5]

### D. IoT dengan MQTT Broker

MQTT kependekan dari *Message Queuing Telemetry Transport*, yaitu protokol pesan berbasis penerbitan standar berlangganan *International Organization for Standardization* (ISO). MQTT adalah protokol yang digunakan dalam IoT untuk transmisi data. Lapisan Aplikasi dan protokol MQTT merupakan lapisan paling atas dari TCP / IP. MQTT merupakan protokol berbasis penerbit dan pelanggan yang memungkinkan beberapa perangkat berkomunikasi satu sama lain melalui jaringan nirkabel [6].

Beberapa istilah penting terkait MQTT, antara lain:

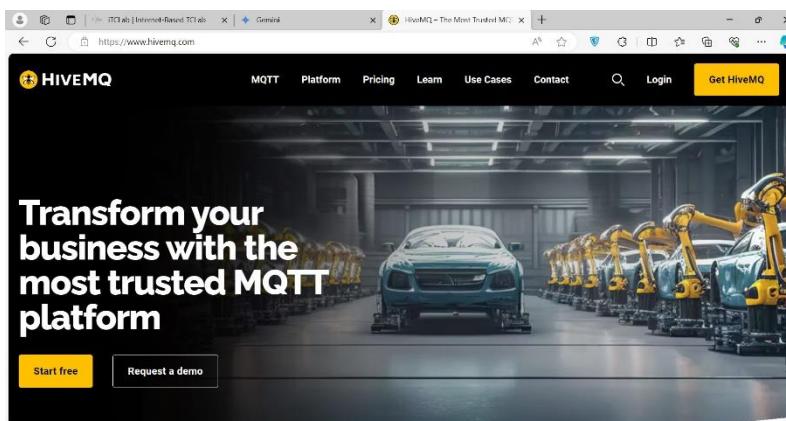
1. *Broker* - Broker adalah server yang mendistribusikan informasi kepada klien yang terhubung ke server.
2. *Client* (Klien) - Perangkat yang terhubung ke broker untuk mengirim atau menerima informasi.
3. *Topic* (Topik) - Nama tentang pesan itu. Klien mempublikasikan, berlangganan, atau melakukan keduanya untuk suatu topik.
4. *Publish* (Terbitkan) - Klien yang mengirim informasi ke broker untuk dibagikan kepada klien yang tertarik berdasarkan nama topik.
5. *Subscribe* (Berlangganan) - Klien memberi tahu broker topik apa yang mereka minati. Ketika klien berlangganan suatu topik, pesan apa pun yang diterbitkan ke broker didistribusikan ke pelanggan topik itu. Klien juga dapat berhenti berlangganan untuk berhenti menerima pesan dari broker tentang topik itu.
6. *QoS* (*Quality of Service*) - Kualitas Layanan. Setiap koneksi dapat menentukan kualitas layanan ke broker dengan nilai integer mulai dari 0-2. QoS tidak mempengaruhi penanganan transmisi data TCP, hanya antara klien MQTT.
  - a. 0 menentukan paling banyak sekali, atau sekali dan hanya sekali tanpa memerlukan pemberitahuan pengiriman. Ini sering disebut sebagai *fire* (api) dan *forget* (lupa).
  - b. 1 menentukan setidaknya satu kali. Pesan itu dikirim beberapa kali sampai sebuah pengakuan diterima, dikenal sebaliknya sebagai pengiriman yang diakui.
  - c. 2 menentukan tepat sekali. Klien pengirim dan penerima menggunakan jabat tangan dua tingkat untuk memastikan hanya satu salinan pesan yang diterima, yang dikenal sebagai pengiriman yang terjamin.

Di MQTT, penerbit dan pelanggan (atau klien) tidak perlu saling mengenal identitas satu sama lain. MQTT mengirimkan informasi dari sumber ke tujuan dan diimplementasikan pada lapisan TCP. MQTT lebih cocok untuk simpul IoT yang memiliki kemampuan dan aset terbatas. Setiap koneksi MQTT

mempertimbangkan dua jenis agen: yang pertama adalah klien MQTT dan yang lainnya adalah *server* broker MQTT. Informasi yang dikirimkan oleh protokol dikenal sebagai pesan aplikasi. Klien MQTT mengacu pada perangkat atau objek yang terhubung ke jaringan yang mengambil bagian dalam komunikasi atau pertukaran pesan melalui MQTT. Klien MQTT disebut sebagai penerbit dan pelanggan. Penerbit dapat mengirim pesan aplikasi dan pelanggan dapat meminta pesan aplikasi itu untuk mendapatkan informasi yang terkait dengan pesan itu. Pialang memungkinkan klien yang berbeda untuk terhubung satu sama lain. Ini mengakui dan mentransmisikan pesan aplikasi di antara berbagai klien yang terkait dengannya. Klien MQTT dapat berupa sensor, perangkat seluler, dll.

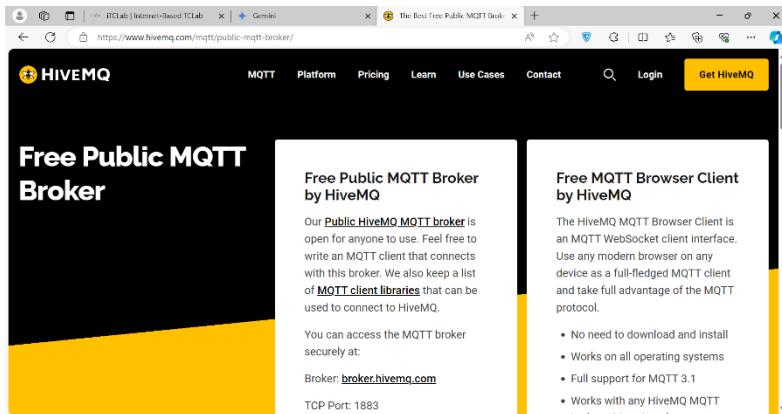
Sederhananya adalah MQTT adalah protokol untuk menyampaikan pesan dari server ke klien maupun sebaliknya. Kelebihan dari MQTT antara lain: mengirim pesan secepat mungkin, menimbulkan *encoding* dan *decoding* data, dan memanfaatkan *storage* (penyimpanan) sekecil mungkin.

Selanjutnya, seperti telah disebutkan pada sistem IoT, untuk membuat projek berbasis IoT kita membutuhkan *cloud data center* yang berfungsi sebagai *MQTT Broker*. Salah satu *MQTT Broker* yang bisa dimanfaatkan adalah *hivemq* (<https://www.hivemq.com>). Tampilan halaman website *hivemq* seperti diperlihatkan pada Gambar 1.5.



Gambar 1. 5. [hivemq.com](https://www.hivemq.com)

Untuk memanfaatkan *MQTT Broker* hivemq, bisa menggunakan fitur yang bersifat terbuka atau umum (*open* atau *public*). Seperti diperlihatkan pada Gambar 1.6.



Gambar 1. 6. Free Public MQTT Broker hivemq

Dari Gambar 1.6 terlihat, ingat konfigurasi yang nantinya kita butuhkan saat bekerja dengan sistem IoT. Sesuai dengan konfigurasi berikut:

Broker	:	broker.hivemq.com
TCP Port	:	1883
Websocket Port	:	8000
TLS TCP Port	:	8883
TLS Websocket Port	:	8884

## E. IoT Mobile dengan IoT MQTT Panel

Selanjutnya untuk bekerja dengan IoT baik untuk pemantauan maupun pengendalian hasil pembacaan sensor dari jarak jauh, melalui ponsel, dibutuhkan *Android Apps IoT MQTT Panel*. Silahkan dicari IoT MQTT Panel di *Play Store* dengan tampilan seperti diperlihatkan pada Gambar 1.7.



Gambar 1. 7. IoT MQTT Panel

Silahkan diinstall di ponsel, aplikasi IoT MQTT Panel, sampai aplikasi siap digunakan.



Gambar 1. 8. IoT MQTT Panel siap digunakan

# BAB

# 2

## BERKENALAN DENGAN MIKROKONTROLER DOIT ESP32 DEVKIT V1

### A. Hardware DOIT ESP32 DEVKIT V1

DOIT ESP32 DevKit V1 adalah salah satu papan pengembangan yang dibuat oleh DOIT untuk mengevaluasi modul ESP-WROOM-32. Papan ini menggunakan mikrokontroler ESP32 yang menawarkan dukungan Wifi, Bluetooth, Ethernet, dan Daya Rendah dalam satu chip tunggal [7].

Platform Espressif 32: ESP32 adalah rangkaian mikrokontroler sistem-pada-chip (SoC) dengan biaya rendah dan konsumsi daya rendah, yang terintegrasi dengan Wi-Fi dan Bluetooth. ESP32 dilengkapi dengan modul sakelar antena, balun RF, penguat daya, penguat penerima low-noise, filter, dan modul manajemen daya [7]. Papan ini dirancang untuk proyek dan aplikasi *Internet of Things* (IoT).



**Gambar 2. 1. Mikrokontroller DOIT ESP32 DEVKIT V1**

**Tabel 2. 1. Hardware DOIT ESP32 DEVKIT V1**

Microcontroller	ESP32
Frequency	240MHz
Flash	4MB
RAM	320KB
Vendor	DOIT

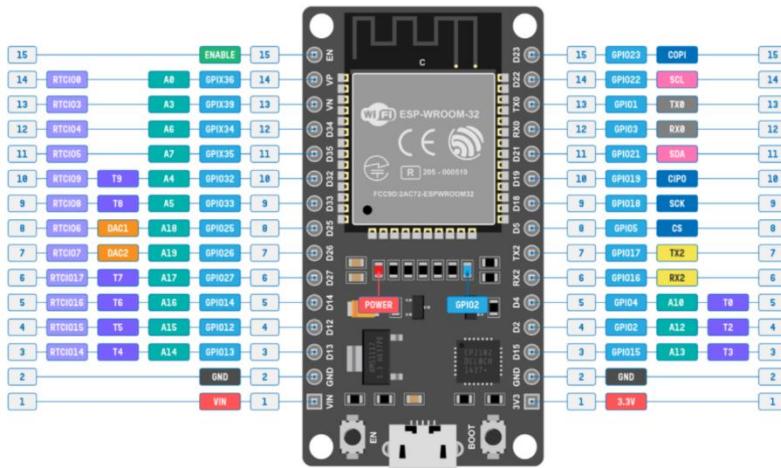
#### **Spesifikasi:**

Mikrokontroler	: Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6
Tegangan Operasi	: 3.3 V
Tegangan Input	: 7-12 V
Pin I/O Digital (DIO)	: 25
Pin Input Analog (ADC)	: 6
Pin Output Analog (DAC)	: 2
UART	: 3
SPI	: 2
I2C	: 3
Memori Flash	: 4 MB

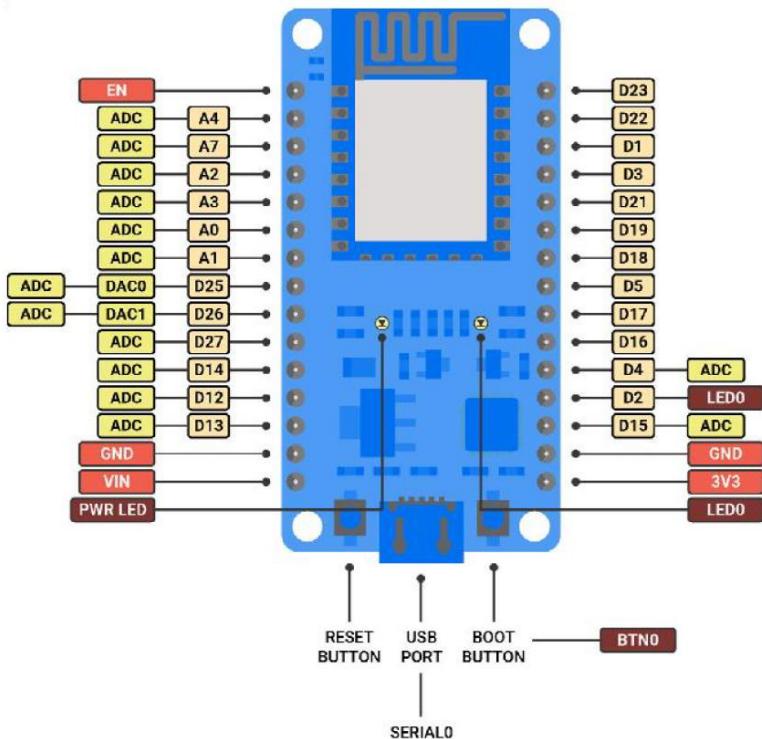
SRAM	: 520 KB
Kecepatan Clock	: 240 MHz
Wi-Fi	: IEEE 802.11 b/g/n/e/i:
* Saklar TR terintegrasi, balun, LNA, penguat daya dan jaringan pencocokan	
* Autentikasi WEP atau WPA/WPA2, atau jaringan terbuka	

## B. Konfigurasi Pin DOIT ESP32 DEVKIT V1

DOIT ESP32 DevKit V1 papan pengembangan yang berbasis SoC Wi-Fi ESP32 populer dari Espressif. Memiliki diagram pinout yang indah yang dibuat oleh CIRCUITSTATE diperlihatkan pada Gambar 2.2 dan Gambar 2.3 [8].



Gambar 2. 2. Konfigurasi Pin DOIT ESP32 DEVKIT V1



**Gambar 2. 3. Konfigurasi Pin mikrokontroller DOIT ESP32 DEVKIT V1**

### Deskripsi Pinout

DOIT ESP32 DevKit V1 memiliki susunan pin yang komprehensif untuk berbagai fungsi. Berikut ini ringkasan singkat pinout-nya:

- Pin GPIO :** Papan ini menyediakan banyak pin General Purpose Input/Output (GPIO) yang dapat digunakan untuk berbagai fungsi input/output digital. Pin ini juga mendukung fungsi seperti PWM, I2C, SPI, dan lainnya.
- Pin Analog Input :** Beberapa pin pada ESP32 DevKit V1 dapat membaca sinyal analog, sehingga cocok untuk dihubungkan dengan sensor analog.
- Pin 3.3V dan GND :** Digunakan untuk memberi daya pada komponen atau sensor eksternal.

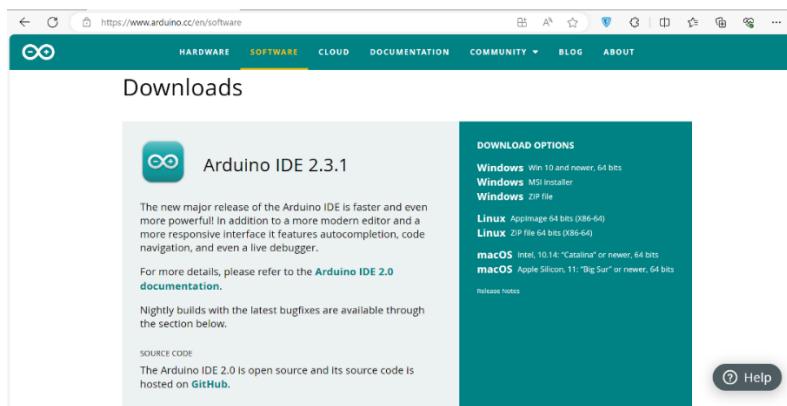
4. **Pin 5V dan GND** : Papan ini juga dapat menyediakan output 5V, yang berguna untuk memberi daya pada modul eksternal yang membutuhkan lebih banyak daya.
5. **Pin VIN** : Ini adalah pin input voltage, yang dapat digunakan untuk memberi daya pada papan saat tidak menggunakan koneksi USB.
6. **Pin EN** : Ini adalah pin enable. Digunakan untuk mereset mikrokontroler.
7. **Pin TX/RX** : Pin ini digunakan untuk komunikasi serial.
8. **Antarmuka SPI** : Papan ini memiliki pin untuk komunikasi SPI, memungkinkan transfer data cepat dengan periferal seperti layar atau memori flash.
9. **Antarmuka I2C** : ESP32 DevKit V1 mendukung komunikasi I2C, yang banyak digunakan untuk menghubungkan sensor dan periferal lainnya.
10. **Pin Sensor Sentuh** : Beberapa GPIO dapat digunakan sebagai input sentuh kapasitif, menawarkan antarmuka untuk perangkat input berbasis sentuh.
11. **Pin VP/VN** : Ini adalah pin untuk sensor efek Hall internal.
12. **Bridge USB-ke-UART** : Fitur ini penting untuk memprogram ESP32 menggunakan kabel USB dan juga untuk komunikasi serial dengan komputer atau perangkat host USB lainnya.

Fleksibilitas papan ini dengan berbagai protokol dan antarmuka membuatnya ideal untuk berbagai aplikasi IoT dan sistem tertanam (*embedded system*).

## C. Pemrograman Mikrokontroller ESP32

### 1. Penyiapan Arduino IDE

Selanjutnya untuk proyek pengembangan aplikasi berbasis mikrokontroller, dibutuhkan Arduino IDE. Silahkan *download* Arduino IDE melalui alamat <https://www.arduino.cc/en/software>. Saat buku ini ditulis, versi terbarunya adalah Arduino IDE 2.3.1.

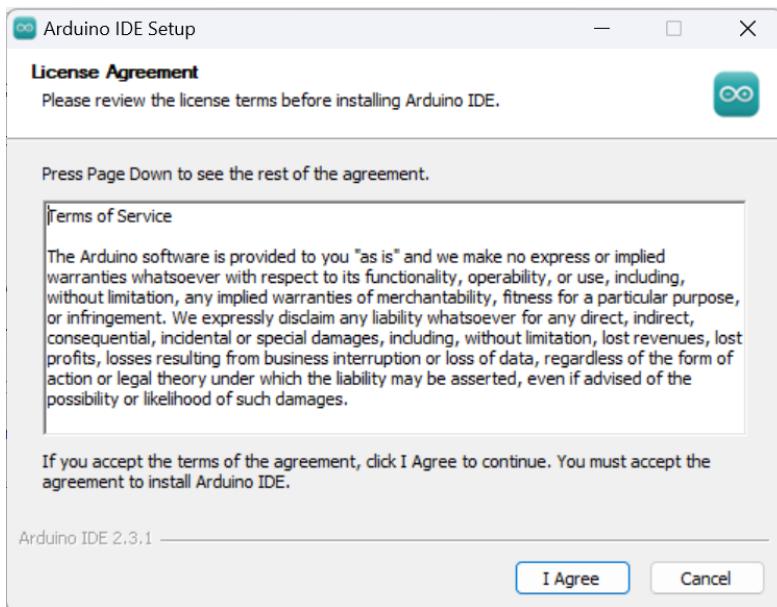


Gambar 2. 4. Download Arduino IDE 2.3.1

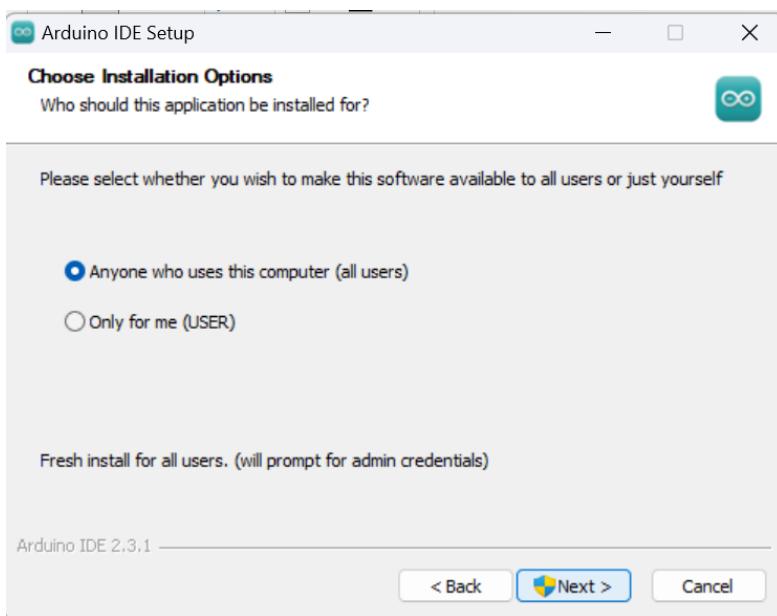
Silahkan *download*, seperti terlihat pada Gambar 2.4. Atau untuk versi Windows 64bit, bisa secara langsung *download* Arduino IDE 2.3.1 pada link berikut ini:

[https://downloads.arduino.cc/arduino-ide/arduino-ide\\_2.3.1\\_Windows\\_64bit.exe](https://downloads.arduino.cc/arduino-ide/arduino-ide_2.3.1_Windows_64bit.exe)

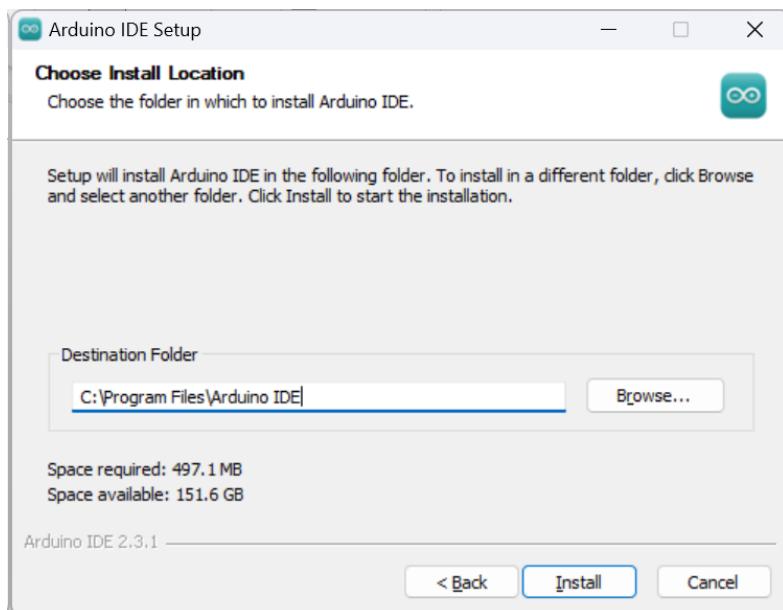
Setelah di-*download* silahkan diinstall seperti biasanya. Tinggal diikuti prosesnya, seperti proses instalasi aplikasi pada umumnya. Seperti terlihat pada gambar-gambar berikut ini.



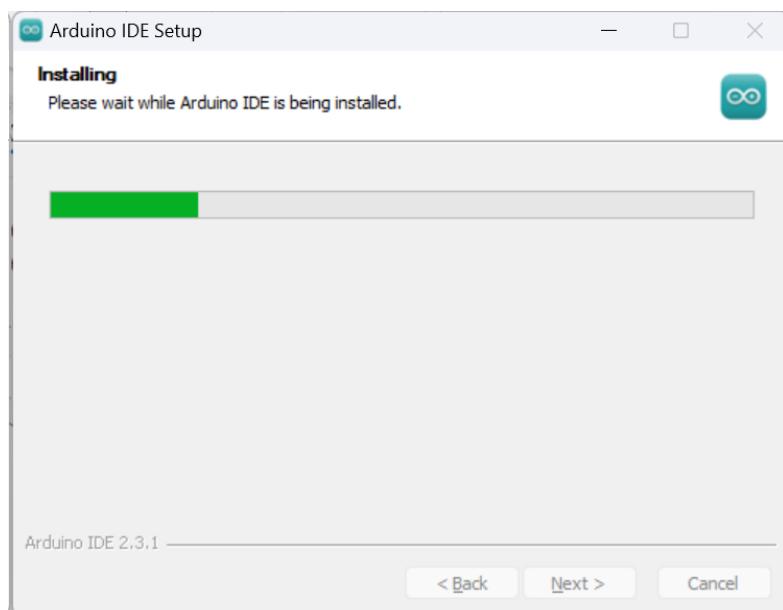
Gambar 2. 5. Persiapan Instalasi Arduino IDE IDE 2.3.1



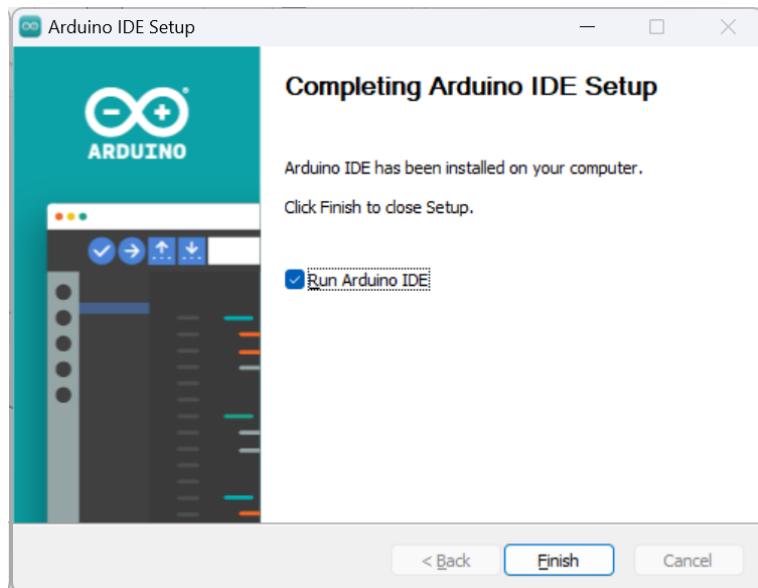
Gambar 2. 6. Tekan Next



Gambar 2. 7. Lokasi Instalasi

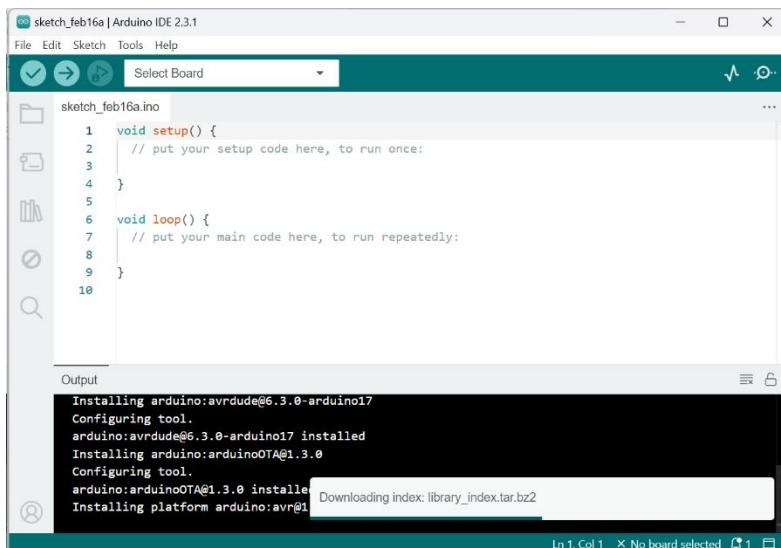


Gambar 2. 8. Proses Instalasi

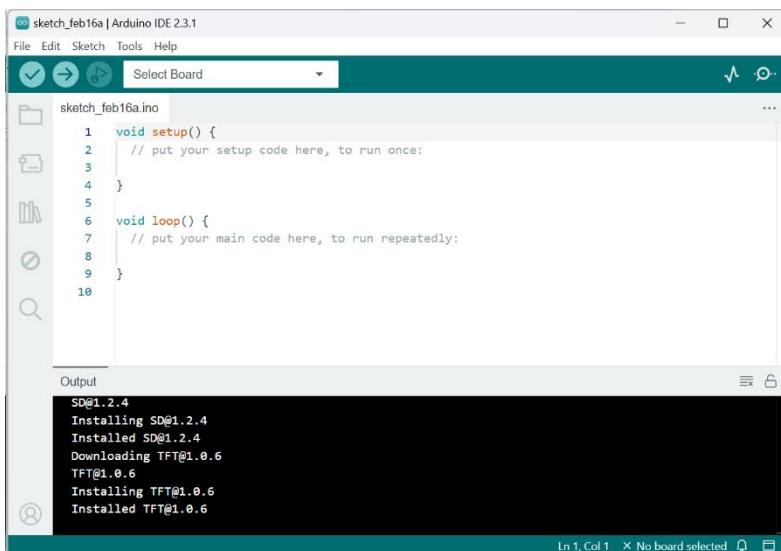


Gambar 2. 9. Proses Instalasi Selesai

Tekan tombol Finish, jika Run Arduino IDE dicentang, maka langsung jalan programnya. Tunggu sampai *download library* otomatis selesai.



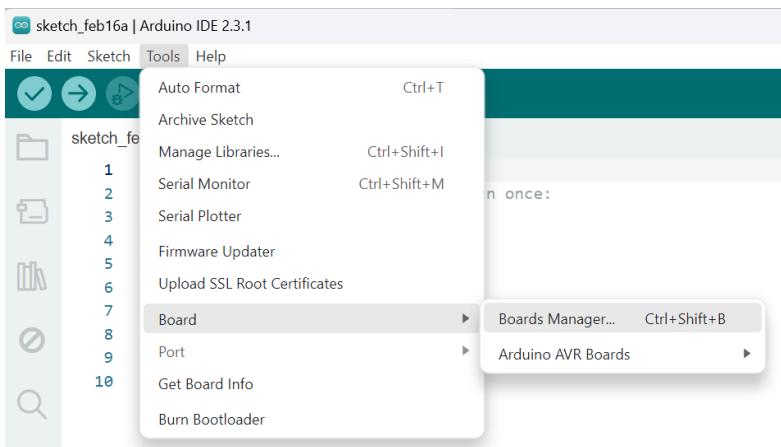
Gambar 2. 10. Download library otomatis tunggu sampai selesai



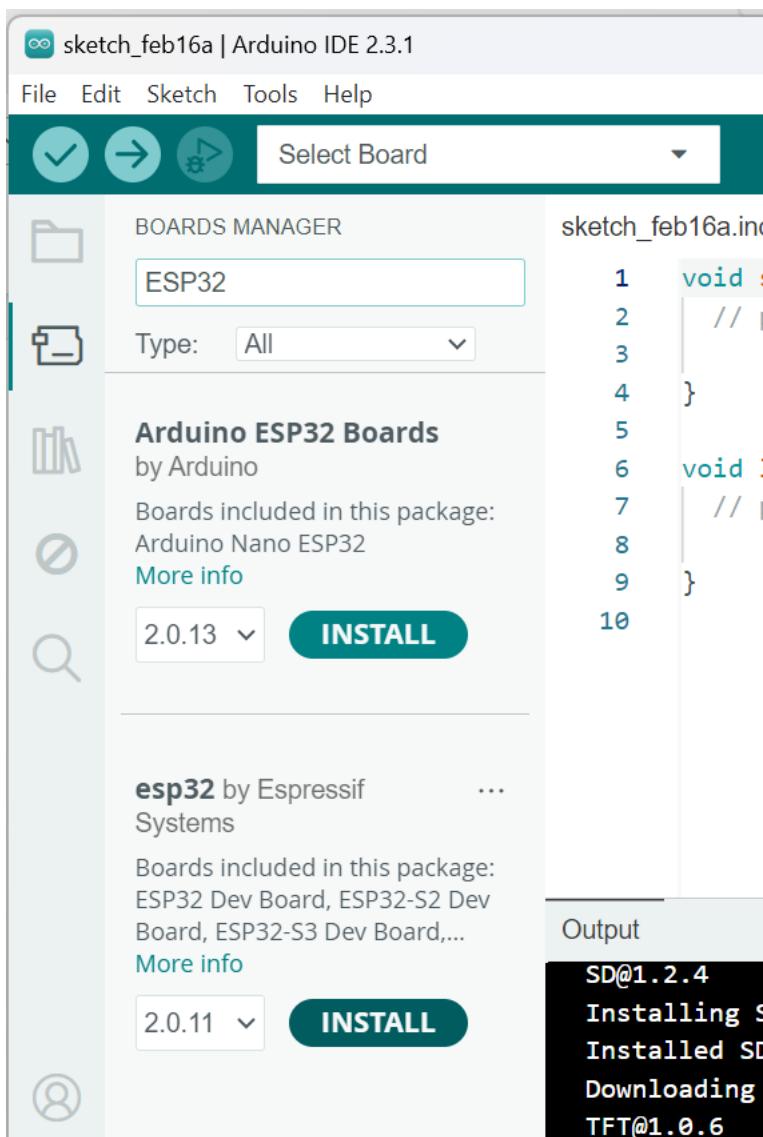
Gambar 2. 11. Arduino IDE 2.3.1 siap digunakan

## 2. Penyiapan Board DOIT ESP32 DEVKIT V1

Pengaturan Board Mikrokontroller ESP32, dengan cara menekan Menu Tools – Board – Boards Manager.

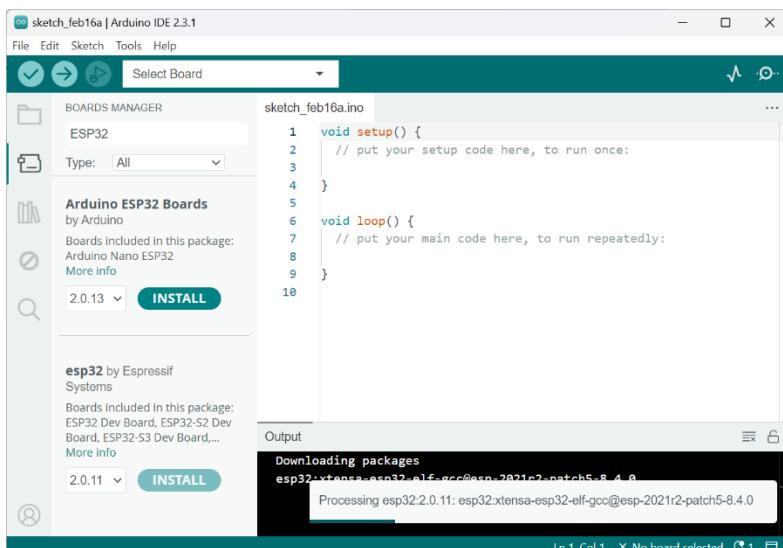


Gambar 2. 12. Pengaturan Board ESP32

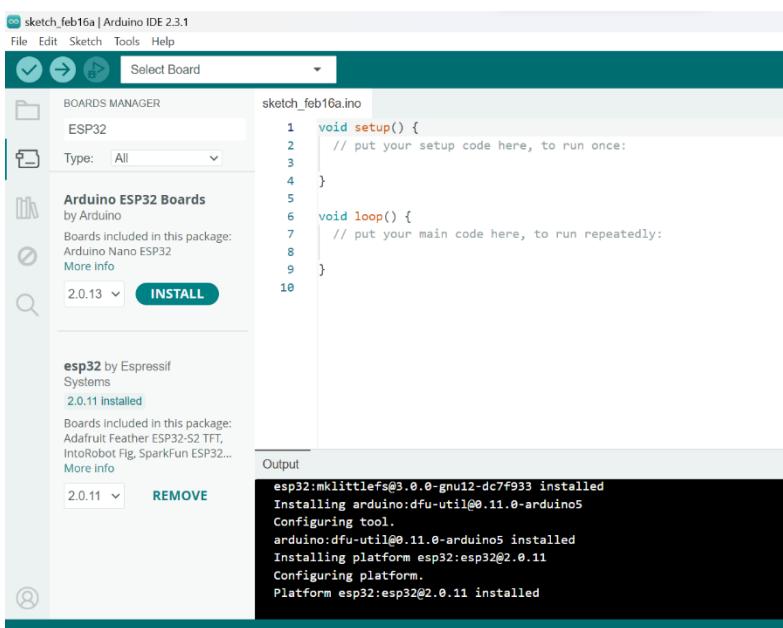


Gambar 2. 13. Ketik dan Pilih ESP32

Silahkan dipilih ESP32, lalu tekan tombol INSTALL. Tunggu sampai proses instalasi selesai.

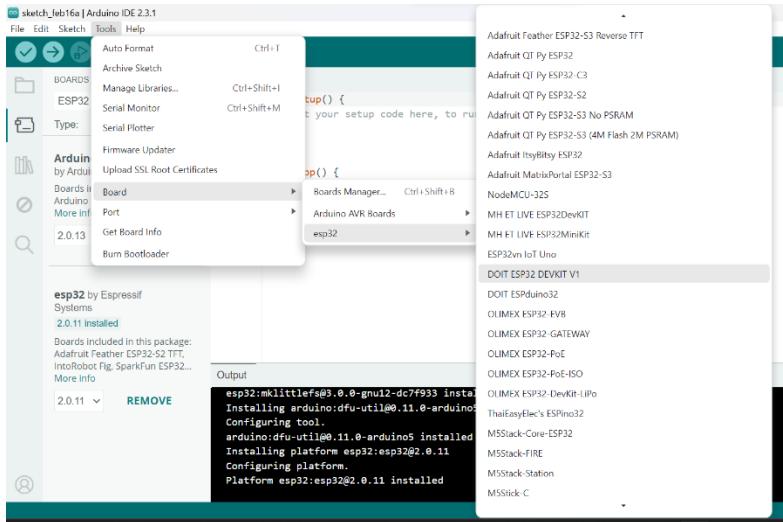


Gambar 2. 14. Proses Instalasi Board ESP32

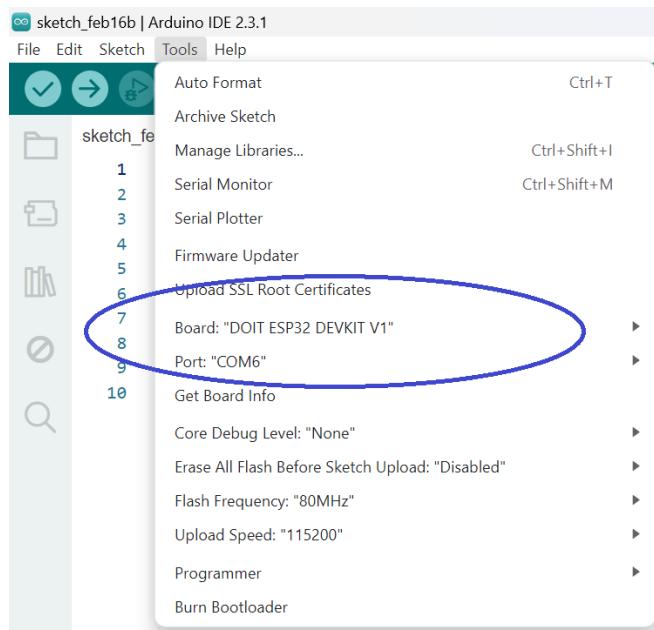


Gambar 2. 15. Instalasi Board ESP32 Sukses

Selanjutnya cek Board ESP32, pada bagian Tools – Board – ESP32, pilih DOIT ESP32 DEVKIT V1.



Gambar 2. 16. Pemilihan Board DOIT ESP32 DEVKIT V1



Gambar 2. 17. Board DOIT ESP32 DEVKIT V1 siap digunakan

# BAB

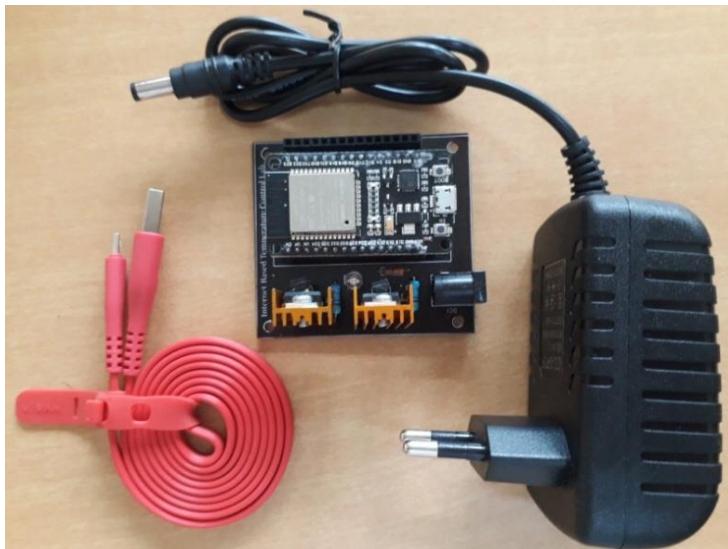
# 3

## BERKENALAN DENGAN KIT ITCLAB

### A. Konsep dan Teknologi iTCLab

Di Era Industry 5.0 dan Society 5.0, kecepatan, ketepatan, dan kecerdasan, memegang peranan sangat penting dalam Proses Pengambilan Keputusan. Kerjasama dengan segala potensi yang kita miliki, terutama di bidang Teknologi Informasi, Internet of Things (IoT), Jaringan, Kontrol, dan Sistem Cerdas, memungkinkan untuk diterapkan di segala bidang kehidupan.

Salah satu wujud penguatan terhadap penguasaan pengetahuan dan teknologi di Era Industry 5.0 dan Society 5.0, antara lain dengan penguasaan teknologi Berbasis Kecerdasan Buatan dan Internet of Things (IoT). Kit *Internet-Based Temperature Control Lab* (iTCLab) merupakan salah satu sarana untuk menunjang hal tersebut.



**Gambar 3. 1. Kit iTCLab Hasil Riset Dosen Kampus Bela Negara**

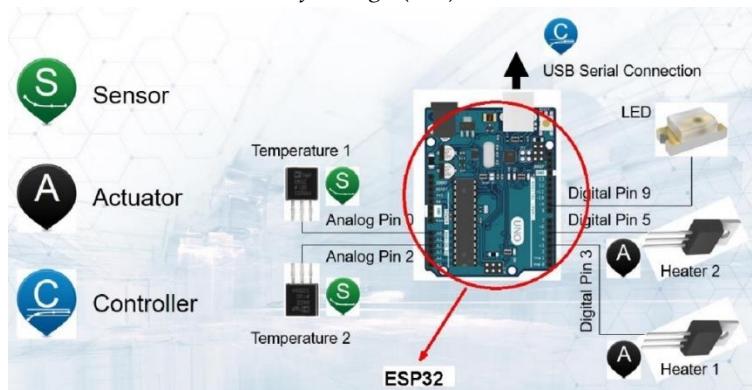
iTCLab - *Internet-Based Temperature Control Lab*. Kit kontrol suhu untuk aplikasi kontrol umpan balik dengan Mikrokontroller ESP32, LED, dua pemanas, dan dua sensor suhu. Keluaran daya pemanas disesuaikan untuk mempertahankan setpoint suhu yang diinginkan. Energi panas dari pemanas ditransfer secara konduksi, konveksi, dan radiasi ke sensor suhu. Panas juga dipindahkan dari perangkat ke lingkungan [9], [10], [11], [12].

#### **Kit iTCLab ini:**

1. Terinspirasi TCLab Produk BYU (Brigham Young University). Salah satu kampus swasta di Provo, Utah Amerika Serikat (<https://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl>)
2. Miniatur Sistem Kendali dalam Saku.
3. Paket Pembelajaran IoT Praktis.
4. Pengenalan Sistem IoT.
5. Pemrograman IoT.
6. Praktek Sistem Kendali Berbasis IoT.

7. Bisa digunakan untuk belajar Dinamika Sistem dan Sistem Kontrol.
8. Bisa digunakan untuk belajar Pemrograman Arduino dan Python.
9. Bisa digunakan untuk belajar Pemrograman Machine Learning.
10. Dan lain-lain.

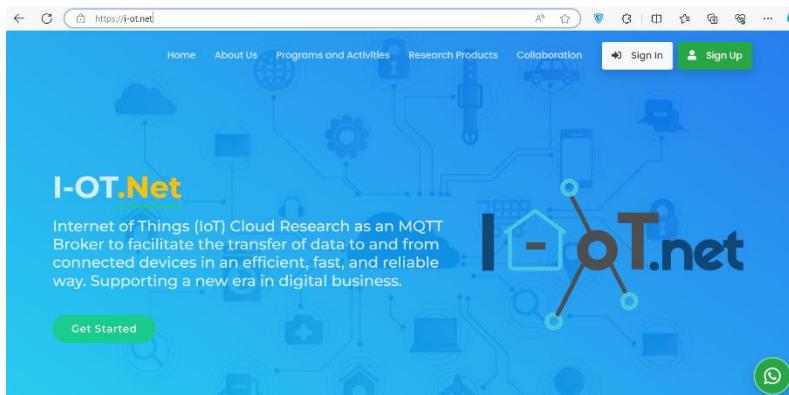
Perbedaan mendasar antara Kit iTCLab dengan Kit TCLab yang diproduksi oleh Kampus BYU, adalah dengan mengganti Mikrokontroller Arduino Uno dengan ESP32. Dengan menggunakan ESP32 ini, maka iTCLab memiliki kemampuan untuk koneksi ke *Internet of Things* (IoT).



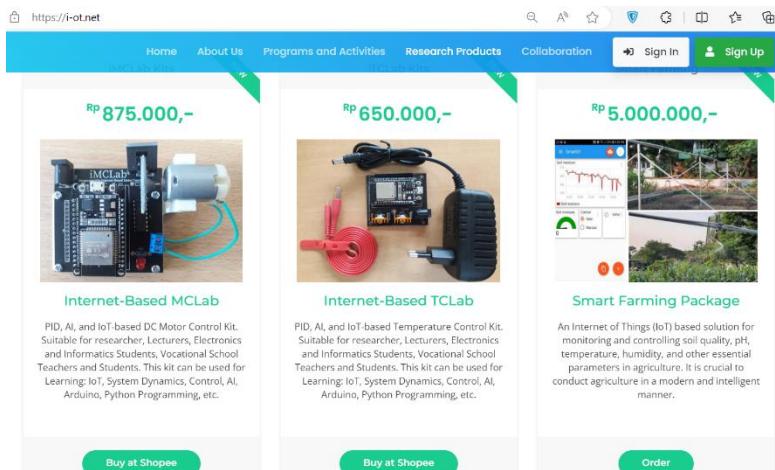
**Gambar 3. 2. Perbedaan iTCLab dan TCLab Kampus BYU**

## B. Bagaimana Mendapatkan Kit iTCLab

Cara mendapatkan Kit iTCLab ini, bisa melalui alamat <https://i-ot.net>. IO-T.Net adalah kumpulan Para Peneliti dengan Kompetensi di Bidang Elektronika, Komputer, Informatika, Jaringan, Instrumentasi & Kontrol, Sistem Informasi, dan Sains Data.

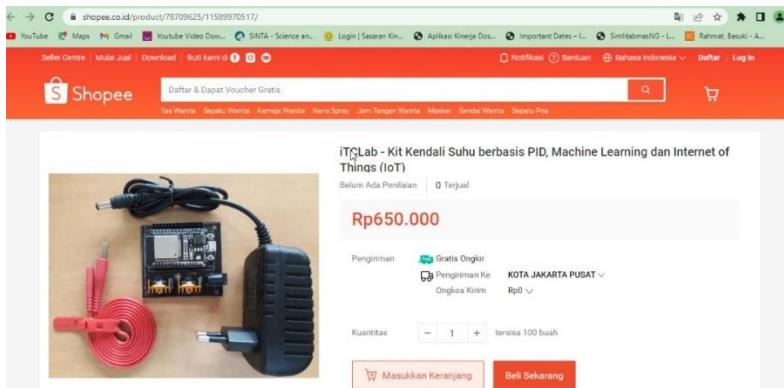


Gambar 3. 3. i-ot.net



Gambar 3. 4. iTCLab di i-ot.net

Cara lain mendapatkan Kit iTCLab juga bisa langsung melalui alamat Shopee berikut:  
<https://shopee.co.id/product/78709625/11589970517/>.



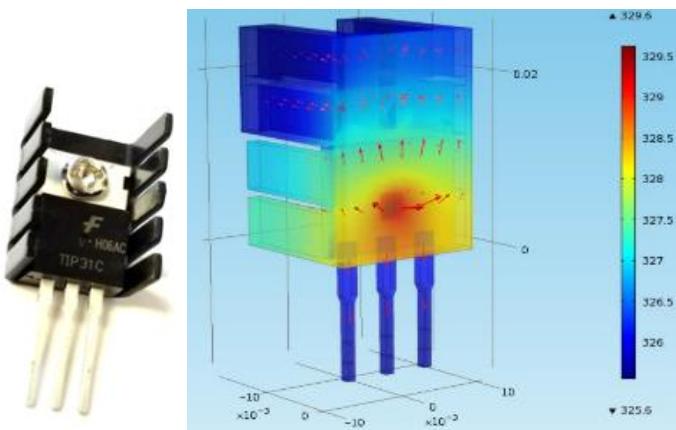
Gambar 3. 5. iTCLab di Shopee

### C. Pemodelan Dinamis Sistem Pemanas

Tujuan: Menurunkan model transien nonlinear berdasarkan hukum keseimbangan energi (*energy balance*) dan dicocokkan dengan sistem orde satu linier atau orde dua linier.

#### 1. Pemodelan Dinamis Keseimbangan Energi

Pemodelan pertama adalah menurunkan model dinamis sistem dengan menggunakan nilai perkiraan parameter. Tiga elemen penting untuk kendali lop tertutup iTCLab yaitu perangkat pengukuran (sensor suhu), aktuator (transistor), dan kemampuan untuk melakukan kendali terkomputerisasi (dengan antarmuka USB). Pada keluaran maksimum, transistor melepas daya kurang lebih 1 Watt pada keluaran pemanas 100%. Massa transistor dan heat sink dengan sirip adalah kurang lebih 4 gram. Diperlihatkan pada Gambar 3.6.



**Gambar 3. 6. Pemanas iTCLab**

Pemanas iTCLab memiliki kapasitas panas  $500 \text{ J/kgK}$ . Luas permukaan pemanas dan sensor sekitar  $1.2 \text{ cm}^2$ . Koefisien perpindahan panas konvektif untuk udara diam adalah sekitar  $10 \text{ W/m}^2\text{K}$ . Perpindahan panas yang dihasilkan oleh transistor melalui perangkat, utamanya disebabkan oleh faktor konveksi, selain faktor radiasi. Perpindahan panas radiatif dapat dimasukkan kedalam model untuk menentukan fraksi panas yang hilang akibat konveksi dan radiasi panas. Perpindahan panas ditingkatkan dengan kopling termal yang menghubungkan dua komponen. Secara lengkap karakteristik pemanas iTCLab seperti yang terdata pada Tabel 3.1.

**Tabel 3. 1. Karakteristik pemanas iTCLab**

<i>Quantity</i>	<i>Value</i>
Initial temperature ( $T_0$ )	$300.15 \text{ K (27}^\circ\text{C)}$
Ambient temperature ( $T_\infty$ )	$300.15 \text{ K (27}^\circ\text{C)}$
Heater output ( $Q$ )	0 to 1 W
Heater factor ( $\alpha$ )	$0.01 \text{ W/(\% heater)}$
Heat capacity ( $C_p$ )	$500 \text{ J/kg-K}$
Surface Area ( $A$ )	$0.12 \times 10^{-3} \text{ m}^2 (1.2 \text{ cm}^2)$
Mass ( $m$ )	$0.004 \text{ kg (4 gm)}$

<i>Quantity</i>	<i>Value</i>
Overall Heat Transfer Coefficient (U)	10 W/m <sup>2</sup> -K
Emissivity ( $\epsilon$ )	0.9
Stefan Boltzmann Constant ( $\sigma$ )	5.67x10 <sup>-8</sup> W/m <sup>2</sup> -K <sup>4</sup>

Selanjutnya dilakukan pemodelan dinamis, respons dinamis antara daya masukan ke transistor dan suhu yang dirasakan oleh termistor. Digunakan keseimbangan energi untuk memulai penurunan, seperti pada Persamaan (3.1) [13].

$$Mc_p \frac{dT}{dt} = \sum \dot{h}_{in} - \sum \dot{h}_{out} + Q \quad (3.1)$$

Istilah-istilah yang diperlukan untuk keperluan ini diperluas atau disederhanakan. Keseimbangan energi penuh didalamnya sudah termasuk istilah konveksi dan radiasi. Persamaan (3.1) diperluas menjadi Persamaan (3.2) [13].

$$Mc_p \frac{dT}{dt} = UA(T_{\infty} - T) + \epsilon \sigma A(T_{\infty}^4 - T^4) + \alpha Q \quad (3.2)$$

Dimana m adalah massa,  $c_p$  adalah kapasitas panas, T adalah suhu, U adalah koefisien perpindahan panas, A adalah area,  $T_{\infty}$  adalah suhu sekitar,  $\epsilon = 0.9$  adalah emisivitas,  $\sigma = 5.67 \times 10^{-8}$  W/m<sup>2</sup>K<sup>4</sup> adalah konstanta Stefan-Boltzmann, dan Q adalah persentase keluaran pemanas. Parameter  $\alpha$  adalah faktor yang menghubungkan keluaran pemanas (0-100%) dengan daya yang didisipasi oleh transistor dalam Watt. Persamaan ini dapat dikembangkan untuk simulasi respons suhu yang dinamis karena adanya dorongan (mati, hidup, mati) pada keluaran pemanas. Biarkan pemanas hidup untuk waktu yang cukup untuk mengamati kondisi yang hampir stabil.

## 2. Pemodelan Dinamis Orde Satu Plus Waktu-Tunda

Pemodelan kedua sistem pemanas iTCLab ini dicoba didekati dengan sistem linear Model Orde Satu Plus Waktu-Tunda atau *First-Order Plus Dead-Time* (FOPDT). Sistem linear

orde satu dengan waktu tunda ini adalah deskripsi empiris umum dari banyak proses dinamis yang stabil. FOPDT digunakan untuk mendapatkan konstanta penalaan pengendali awal.

Parameter-parameter yang mempengaruhi keluaran dari sistem linear orde satu plus waktu tunda ini terdiri dari parameter gain  $K_p$ , konstanta waktu  $\tau_p$ , dan waktu tunda  $\theta_p$ . Persamaan FOPDT dinyatakan dalam bentuk [13]:

$$\tau_p \frac{dy(t)}{dt} = -y(t) + K_p u(t - \theta_p) \quad (3.3)$$

Persamaan (3.3) memiliki variabel keluaran  $y(t)$  dan masukan  $u(t)$  dan tiga parameter yang tidak diketahui yaitu gain proses  $K_p$ , konstanta waktu  $\tau_p$ , dan waktu tunda  $\theta_p$ .

#### a. Gain proses $K_p$ .

Gain proses adalah perubahan keluaran  $y$  yang diinduksi oleh perubahan satuan pada masukan  $u$ . Gain proses dihitung dengan cara mengevaluasi perubahan dalam  $y(t)$  dibagi dengan perubahan  $u(t)$  pada kondisi awal dan akhir kondisi tunak (*steady state*) [13].

$$K_p = \frac{\Delta y}{\Delta u} = \frac{y_{ss2} - y_{ss1}}{u_{ss2} - u_{ss1}} \quad (3.4)$$

Gain proses mempengaruhi besarnya respon, terlepas dari kecepatan respon.

#### b. Konstanta waktu $\tau_p$ .

Diberikan perubahan  $u(t) = \Delta u$ , solusi untuk diferensial orde satu linier (tanpa waktu tunda) menjadi [13]:

$$y(t) = \left( e^{-\frac{t}{\tau_p}} \right) y(0) + \left( 1 - e^{-\frac{t}{\tau_p}} \right) K_p \Delta u \quad (3.5)$$

Jika kondisi awal  $y(0)=0$  dan pada  $t=\tau_p$ , maka solusinya disederhanakan sebagai berikut [13]:

$$y(\tau_p) = \left( 1 - e^{-\frac{\tau_p}{\tau_p}} \right) K_p \Delta u = (1 - e^{-1}) K_p \Delta u = 0.632 K_p \Delta u \quad (3.6)$$

Maka konstanta waktu proses adalah jumlah waktu yang diperlukan untuk mencapai keluaran ( $1 - \exp(-1)$ ) atau 63.2% menuju kondisi tunak. Konstanta waktu proses mempengaruhi kecepatan respon.

### c. Waktu tunda $\theta_p$ .

Waktu tunda dinyatakan sebagai pergeseran waktu yang terdapat didalam variabel input  $u(t)$  [13]:

$$u(t - \theta_p) \quad (3.7)$$

Misalkan sinyal masukan itu adalah berupa fungsi *step* yang biasanya berubah dari 0 ke 1 pada waktu  $t=0$  tetapi pergeseran ini tertunda 5 detik. Fungsi masukan  $u(t)$  dan fungsi keluaran  $y(t)$  digeser waktunya 5 detik. Solusi untuk persamaan diferensial orde satu dengan waktu tunda diperoleh dengan mengganti semua variabel  $t$  dengan  $t-\theta_p$  dan menerapkan hasil kondisional berdasarkan waktu tunda  $\theta_p$  [13]:

$$y(t < \theta_p) = y(0) \quad (3.8)$$

$$Y(t \geq \theta_p) = \left( e^{-\frac{(t-\theta_p)}{\tau_p}} \right) y(0) + \left( 1 - e^{-\frac{(t-\theta_p)}{\tau_p}} \right) K_p \Delta u \quad (3.9)$$

## 3. Pemodelan Dinamis Orde Dua Plus Waktu-Tunda

Pemodelan ketiga sistem pemanas iTCLab ini dicoba didekati dengan sistem linear Model Orde Dua Plus Waktu-Tunda atau *Second-Order Plus Dead-Time* (SOPDT). Sistem linear orde dua dengan waktu tunda ini adalah deskripsi empiris dari proses dinamis yang berpotensi berosilasi.

Persamaannya [13]:

$$\tau_s^2 \frac{d^2y}{dt^2} + 2\zeta\tau_s \frac{dy}{dt} + y = K_p u(t - \theta_p) \quad (3.10)$$

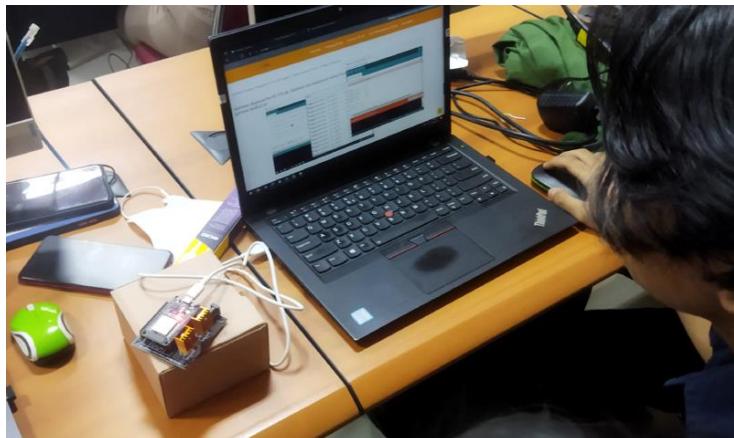
Persamaan (3.10) memiliki keluaran  $y(t)$  dan masukan  $u(t)$  dan empat parameter yang tidak diketahui. Keempat parameter tersebut adalah gain  $K_p$ , faktor redaman  $\zeta$ , konstanta waktu orde dua  $\tau_s$ , dan waktu tunda  $\theta_p$ .

Alternatif untuk pendekatan penyesuaian grafik respon model adalah dengan penggunaan optimasi agar keluaran model SOPDT sesuai dengan data riil. Tujuannya untuk meminimalkan jumlah kesalahan kuadrat yang menyebabkan penyimpangan model SOPDT dari data. Algoritma optimasi mengubah parameter untuk mencocokkan data pada titik waktu yang ditentukan.

## D. Pemrograman Dasar Kit iTCLab dengan Arduino

### 1. Pengaturan Preferences

Selanjutnya, silahkan hubungkan Kit iTCLab ke komputer.

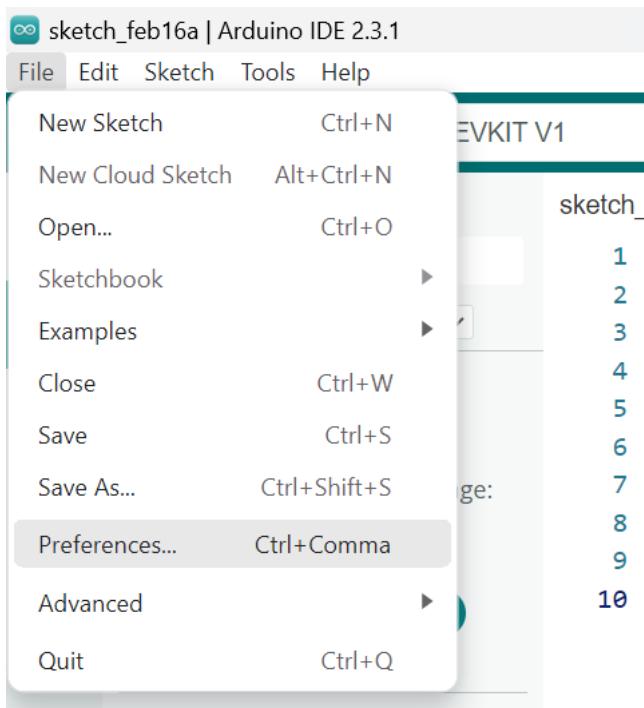


Gambar 3. 7. iTCLab terhubung ke Laptop

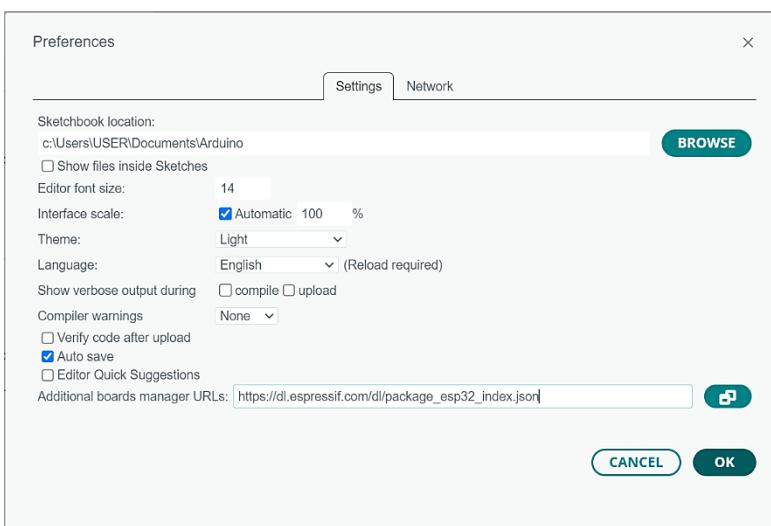
#### Pengaturan File - Preferences:

Kit iTCLab menggunakan Mikrokontroller ESP32. Silahkan di-copy dan di-paste, pada bagian *File - Preferences*, alamat berikut ini:

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

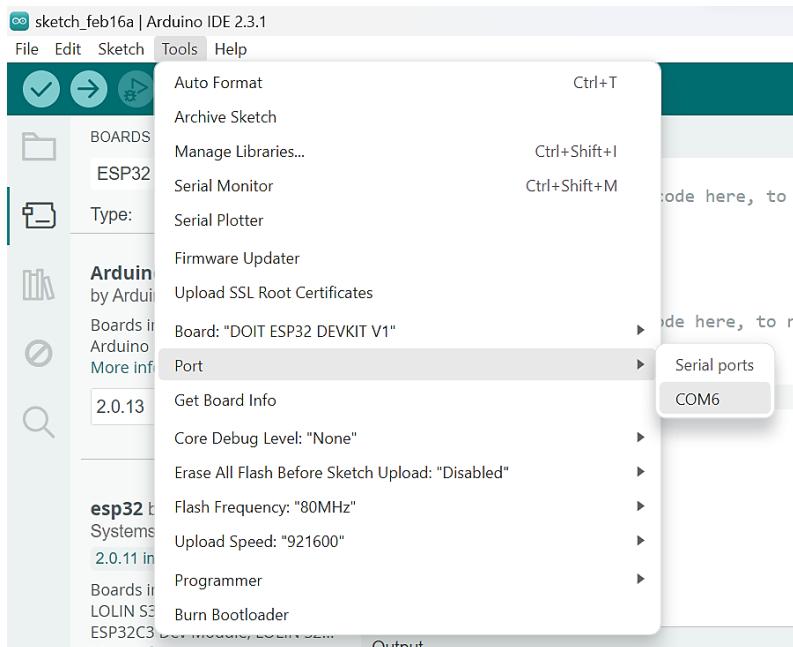


Gambar 3. 8. Pengaturan File - Preferences



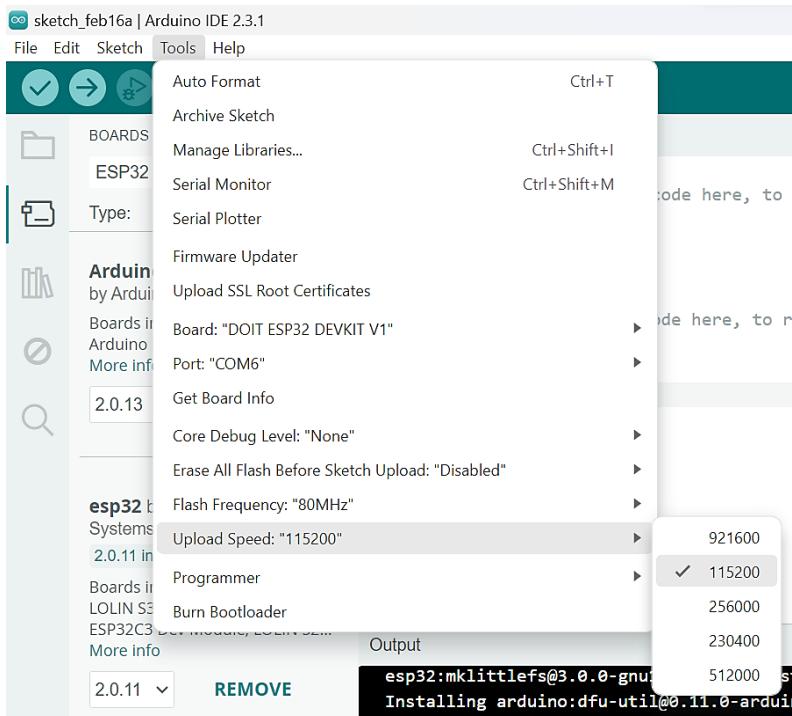
Gambar 3. 9. Pengaturan File - Preferences (Lanjutan)

Selanjutnya, silahkan pilih Port sesuai yang muncul, ketika Kit iTCLab dalam keadaan terhubung ke komputer (laptop).



Gambar 3. 10. Pilih Port iTCLab

Selanjutnya, silahkan pilih kecepatan upload, di angka 115200.

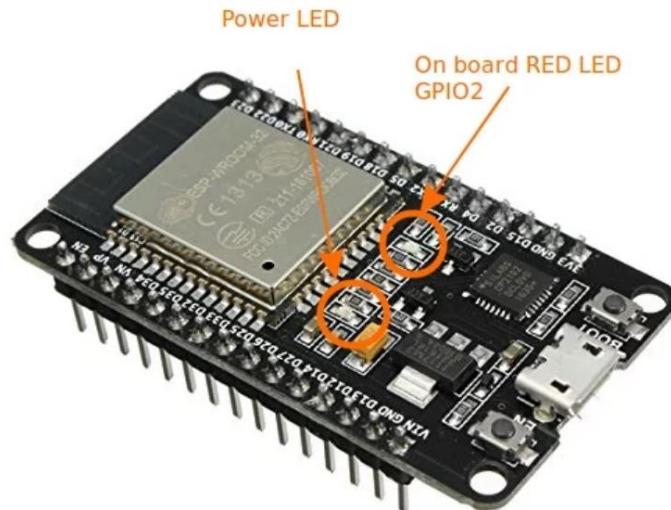


Gambar 3. 11. Pilih Upload Speed 115200

## 2. Program Pertama Blink

ESP32 adalah perangkat Internet of Things (IoT) yang dilengkapi dengan prosesor inti ganda (dual core CPU), Wi-Fi, dan Bluetooth. Dalam contoh ini, kita akan memulai dengan contoh sederhana yaitu membuat Program Lampu LED berkedip (blink) di ESP32.

ESP32 DevKit V1 memiliki Lampu LED merah bawaan yang terhubung ke pin GPIO2. Proses inti ganda (dual core CPU): Memiliki dua prosesor yang dapat bekerja secara bersamaan untuk meningkatkan kinerja. GPIO (*General Purpose Input/Output*): Pin pada ESP32 yang dapat dikonfigurasi untuk berbagai fungsi, termasuk mengontrol LED.



Gambar 3. 12. LED ESP32

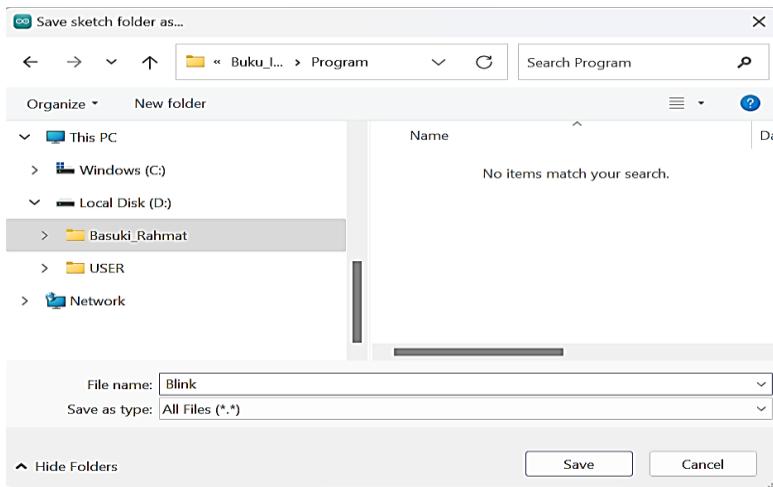
Berikut ini contoh Program Blink sederhana:

```
#define LED 2

void setup() {
    // Set pin mode
    pinMode(LED,OUTPUT);
}

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

Silahkan *Copy – Paste* ke Arduino IDE, dengan cara select All program sebelumnya, ganti dengan Program Blink ini. Silahkan disimpan dengan nama Blink.



Gambar 3. 13. Simpan Program Blink

Silahkan upload ke Kit iTCLab, dengan menekan tombol Upload, tunggu sampai selesai, dan lihat hasilnya.

A screenshot of the Arduino IDE version 2.3.1. The title bar says "Blink | Arduino IDE 2.3.1". The menu bar includes File, Edit, Sketch, Tools, and Help. A toolbar with icons for checkmark, upload, and settings is visible. The central workspace shows a sketch named "Blink" with the following code:

```
1 #define LED 2
2
3 void setup() {
4     // Set pin mode
5     pinMode(LED,OUTPUT);
6 }
7
8 void loop() {
9     delay(500);
10    digitalWrite(LED,HIGH);
11    delay(500);
12    digitalWrite(LED,LOW);
13 }
```

Gambar 3. 14. Program Blink

```
1 // Define LED 2
2
3 void setup() {
4     // Set pin mode
5     pinMode(LED,OUTPUT);
6 }
7
8 void loop() {
9     delay(500);
10    digitalWrite(LED,HIGH);
11    delay(500);
12    digitalWrite(LED,LOW);
13 }
14
```

Output

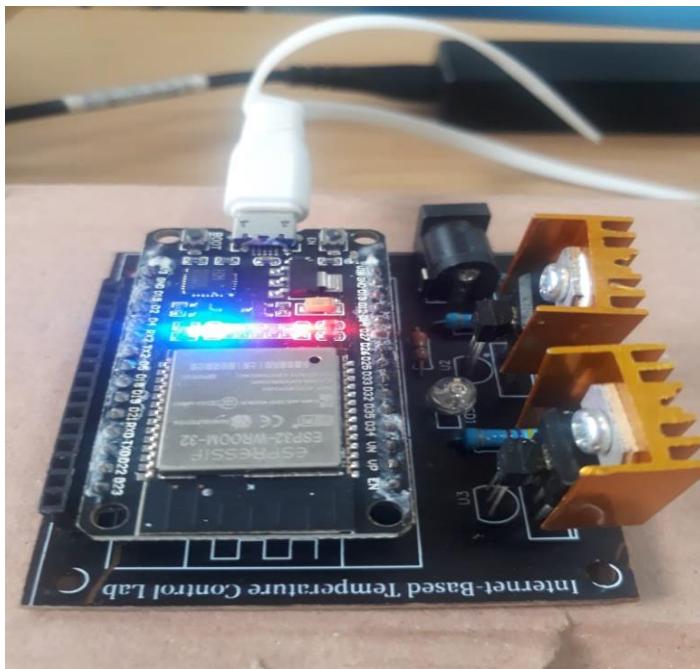
```
Writing at 0x00003f00... (87 %)
Writing at 0x00004700... (100 %)
Wrote 217168 Bytes (138427 compressed) at 0x00010000 in 11.7 seconds (effective 162.5 kbit/s)...
Hash of data verified.
```

Leaving...
Hard resetting via RTS pin...

Done uploading.

Gambar 3. 15. Proses Upload Sukses

Silahkan lihat hasilnya, pada Kit iTCLab. Seharusnya LED iTCLab akan berkedip.



Gambar 3. 16. LED berkedip (blink)

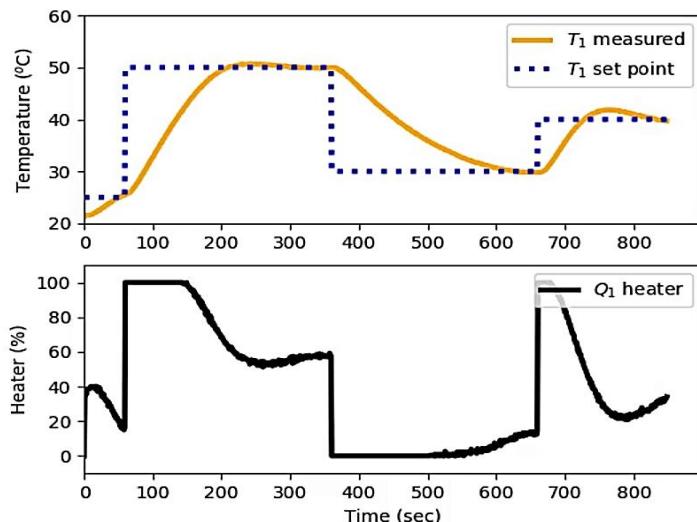
### 3. Program Pengujian iTCLab Sederhana

Selanjutnya, akan dibuat Program Pengujian iTCLab Sederhana. Namun sebelumnya, perlu diperhatikan, batasan kemampuan dari Kit iTCLab.

#### Batas Suhu Atas iTCLab

Kemampuan Kit iTCLab untuk Batas Suhu Atas adalah pada Suhu **60 derajad Celsius**. Oleh karena itu, didalam berekspimen dengan menggunakan Kit iTCLab ini, tidak boleh melebihi Batas Suhu Atas ini. **Pelanggaran akan ketentuan ini, dikhawatirkan akan menyebabkan kerusakan (terbakarnya) komponen.**

Walaupun batas atas 60 derajad Celsius, namun sudah cukup untuk berekspimen dengan Kit ini. Dan sudah memenuhi untuk melihat kinerja dari sebuah metode pengendalian. Misalnya pengendalian menggunakan Proporsional Integral dan Deratif (PID). Atau pun untuk melihat pengaruh penalaan (tuning) terhadap parameter PID menggunakan metode Machine Learning. Gambaran kemampuan Kit iTCLab ini, bisa dilihat dari gambaran penampilan kinerja TCLab dari BYU, seperti terlihat pada simulasi berikut ini [13].



Gambar 3. 17. Gambaran kinerja Kit TCLab BYU

## Coding Batasan Suhu Atas

Perlu dibuat batasan agar Kit iTCLab selalu bekerja di wilayah aman. Tidak boleh melebihi batas atas 60 derajad Celsius. Berikut ini contoh *script* program arduino yang harus ditambahkan setiap kali bereksperimen dengan Kit ini. Pada *Loop*, ditambahkan, jika mencapai batas atas yang ditentukan (boleh diturunkan sedikit, misalnya 55 derajad celsius), maka *heater* (pemanas) harus dimatikan (Off).

```
const float upper_temperature_limit = 55;

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);
}
```

Berikut ini, secara lengkap, Program Pengujian iTCLab Sederhana (iTCLab\_Testing.ino). **iTCLab\_Testing.ino** adalah program pengujian Kit iTCLab sederhana. Suhunya dinaikkan secara bertahap sampai batas atas yang diinginkan, yaitu 55 derajad celsius.

```
/*
 * Program : iTCLab_Testing
 * By      : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI,
 *           Assoc. Prof. Dr. Muljono, S.Si, M.Kom, et al
 * Pro. Team : i-ot.net, io-t.net
 * R. Group  : Intelligent Control, Robotics and Automation Systems Research Group
 * Univ.     : Universitas Pembangunan Nasional "Veteran" Jawa Timur
 * Country   : Indonesia
 */

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

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

```

```

Serial.print("°C");
Serial.print(" ~ "); // separator Celsius - Fahrenheit
Serial.print(cel1); // print temperature T2 in Celsius
Serial.println("°C");
}

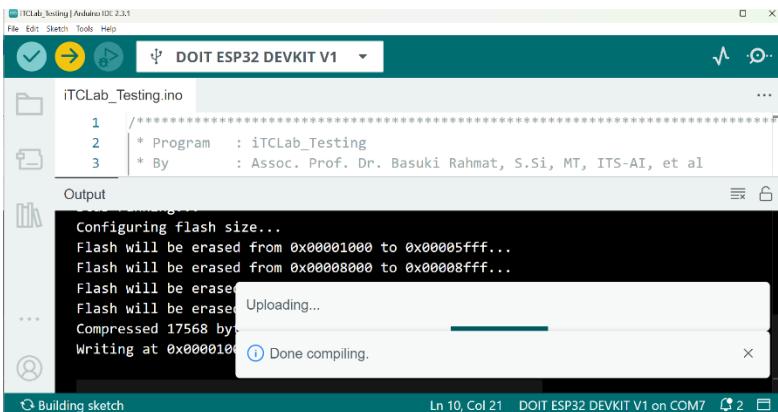
void loop() {
// put your main code here, to run repeatedly:
cektemp();
if (cel > upper_temperature_limit){
Q1off();
ledon();
}
else {
Q1on();
ledoff();
}
if (cel1 > upper_temperature_limit){
Q2off();
ledon();
}
else {
Q2on();
ledoff();
}
delay (100);
}

```

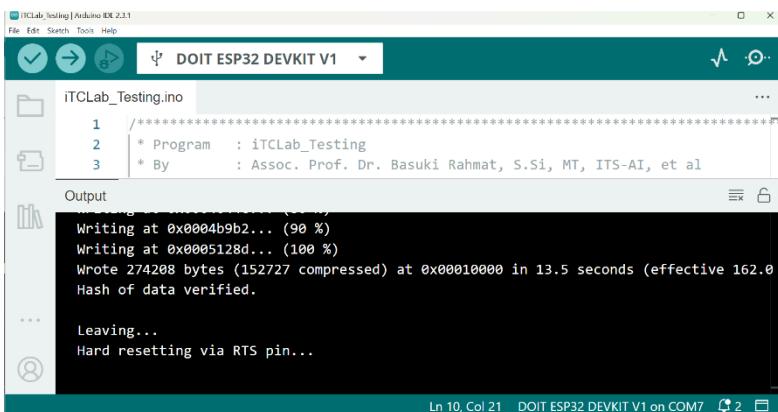
Program **iTCLab\_Testing.ino** di atas, juga bisa di-*download* di alamat:

[https://github.com/bsrahmat/itclab-01/blob/main/iTCLab\\_Testing.ino](https://github.com/bsrahmat/itclab-01/blob/main/iTCLab_Testing.ino)

Silahkan program di atas di-*upload* ke Kit iTCLab, tunggu sampai proses selesai.

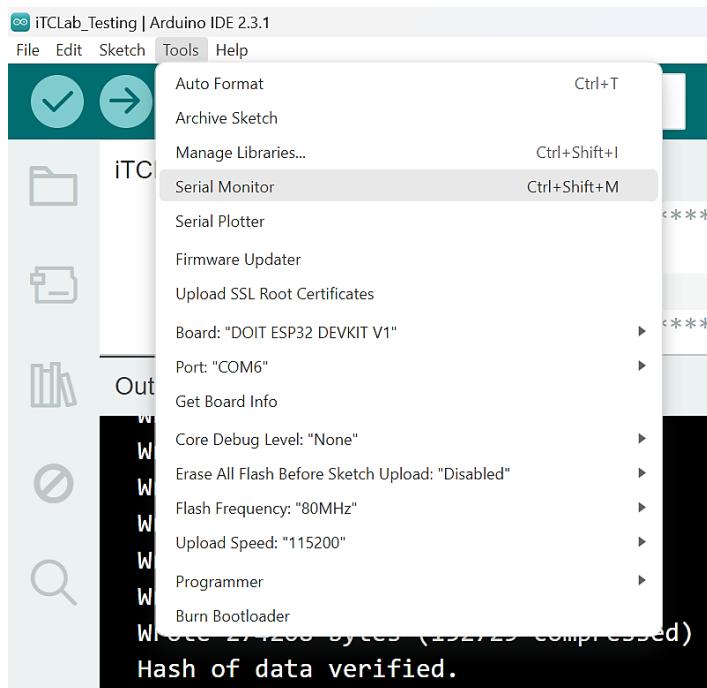


Gambar 3. 18. Proses upload Program iTCLab\_Testing



Gambar 3. 19. Proses upload Program iTCLab\_Testing Berhasil

Setelah berhasil di-upload dengan sempurna. Silahkan cek hasilnya pada Serial Monitor. Dilakukan dengan cara menekan tombol Tools – Serial Monitor.



Gambar 3. 20. Tools - Serial Monitor

Jangan lupa, sesuaikan baudrate pada 115200. Jika sudah berhasil tampil hasil pembacaan suhu seperti terlihat pada Gambar 3.21. Jika ingin suhunya terus naik sampai batas atas (55 derajad celsius), maka jangan lupa Power Adaptor dihubungkan ke listrik. Sehingga seharusnya suhu terus meningkat sampai batas atas. Seperti terlihat pada Gambar 3.22.

Gambar 3. 21. Hasil Pembacaan Suhu iTCLab

Gambar 3. 22. Menulis Batas Atas suhu iTCLab

Komponen pemanas akan panas sekali. Jangan disentuh.

#### 4. Program Pengujian PWM iTCLab

Pengujian berikutnya, disiapkan Program PWM Testing.

### Pengertian PWM (Pulse Width Modulation)

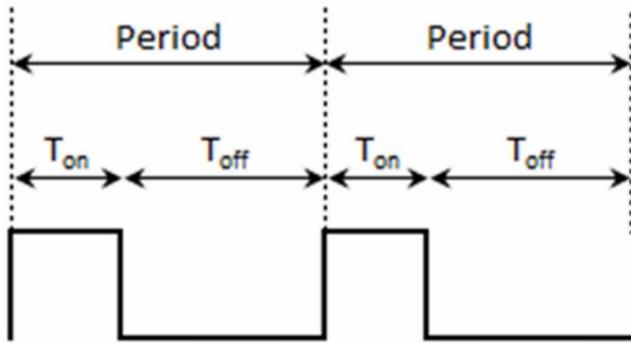
PWM atau *Pulse Width Modulation* atau dalam bahasa Indonesia dapat diterjemahkan menjadi Modulasi Lebar Pulsa. Jadi pada dasarnya, PWM adalah suatu teknik modulasi yang mengubah lebar pulsa (pulse width) dengan nilai frekuensi dan amplitudo yang tetap. PWM dapat dianggap sebagai kebalikan dari ADC (*Analog to Digital Converter*) yang mengkonversi sinyal Analog ke Digital,

PWM ini digunakan menghasilkan sinyal analog dari perangkat Digital (Kit iTCLab).

Sinyal PWM akan tetap ON untuk waktu tertentu dan kemudian terhenti atau OFF selama sisa periodenya. Yang membuat PWM ini istimewa dan lebih bermanfaat adalah kita dapat menetapkan berapa lama kondisi ON harus bertahan dengan cara mengendalikan siklus kerja atau *Duty Cycle* PWM.

Siklus kerja (*duty cycle*) adalah konsep penting yang menggambarkan proporsi waktu sinyal berada dalam keadaan aktif dibandingkan dengan periode totalnya. Konsep ini banyak digunakan dalam berbagai aplikasi, mulai dari modulasi lebar pulsa (pulse-width modulation) dalam sistem kontrol hingga modulasi sinyal dalam telekomunikasi. Istilah "siklus kerja" digunakan dalam elektronika dan teknik untuk menyatakan rasio waktu sistem, perangkat, atau komponen aktif atau dalam keadaan "on" dibandingkan dengan seluruh periode operasi.

Biasanya dinyatakan sebagai persentase dan menunjukkan persentase waktu sistem aktif selama satu siklus lengkap. Siklus kerja dalam konteks sinyal dan bentuk gelombang listrik mengacu pada persentase waktu sinyal berada dalam keadaan "tinggi" (sering dinyatakan sebagai "aktif" atau ON) relatif terhadap durasinya secara keseluruhan. Kondisi yang sinyalnya selalu dalam kondisi ON disebut sebagai 100% *Duty Cycle* (Siklus Kerja 100%), sedangkan kondisi yang sinyalnya selalu dalam kondisi OFF (mati) disebut dengan 0% *Duty Cycle* (Siklus Kerja 0%). Gambaran Diagram Siklus Kerja atau *Duty Cycle Diagram* diperlihatkan pada Gambar 3.23 [14].



Gambar 3. 23. Diagram Siklus Kerja

Rumus untuk menghitung siklus kerja atau *duty cycle* dapat ditunjukkan seperti persamaan di bawah ini [14]:

$$Periode = \frac{1}{Frekuensi} \quad (3.11)$$

$$Periode = T_{on} + T_{off} \quad (3.12)$$

$$Siklus\ Kerja = \left( \frac{T_{on}}{T_{on}+T_{off}} \right) \times 100\% \quad (3.13)$$

Berikut ini, secara lengkap, Program Pengujian *Pulse Width Modulation* (PWM) iTCLab Sederhana (PWM\_Testing.ino).

```
/*
* Program : PWM_Testing
* By      : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI,
*           Assoc. Prof. Dr. Muljono, S.Si, M.Kom, et al
* Pro. Team : i-ot.net, io-t.net
* R. Group : Intelligent Control, Robotics and Automation Systems Research Group
* Univ.    : Universitas Pembangunan Nasional "Veteran" Jawa Timur
* Country  : Indonesia
***** */
```

```

// 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 functionalities
    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);
    }
}

```

Program **PWM\_Testing.ino** di atas, juga bisa diunduh di alamat:  
[https://github.com/bsrahmat/itclab-02/blob/main/PWM\\_Testing.ino](https://github.com/bsrahmat/itclab-02/blob/main/PWM_Testing.ino)

Silahkan program di atas di-upload ke Kit iTCLab, tunggu sampai proses selesai.

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** PWM\_Testing | Arduino IDE 2.3.1
- Sketch Name:** PWM\_Testing.ino
- Board:** DOIT ESP32 DEVKIT V1
- Serial Monitor Output:**

```
Writing at 0x000292ff... (44 s)
Writing at 0x0002e85c... (5 - s)
Writing at 0x00036e... Uploading...
```
- Status Bar:** indexing: 34/49, Ln 37, Col 41, DOIT ESP32 DEVKIT V1 on COM7, 2

Gambar 3. 24. Proses upload Program PWM\_Testing

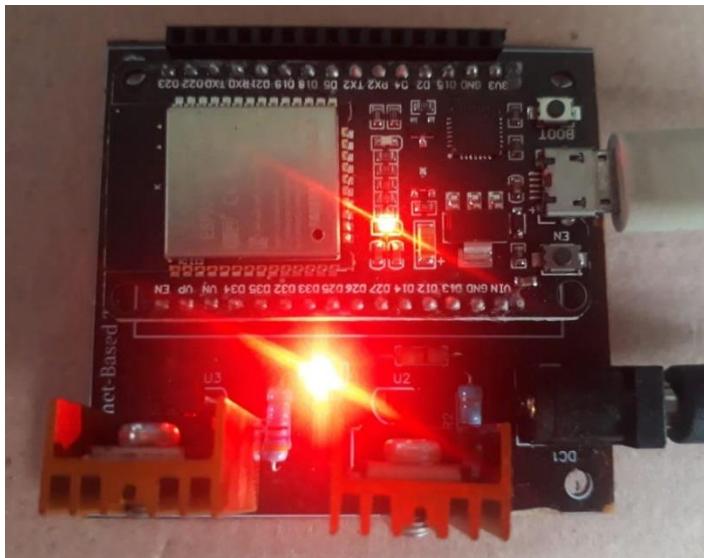
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** PWM\_Testing | Arduino IDE 2.3.1
- Sketch Name:** PWM\_Testing.ino
- Board:** DOIT ESP32 DEVKIT V1
- Serial Monitor Output:**

```
Leaving...
Hard resetting via RTS pin...
```
- Status Bar:** Ln 37, Col 41, DOIT ESP32 DEVKIT V1 on COM7, 2

Gambar 3. 25. Upload Program PWM\_Testing selesai

Silahkan cek hasilnya dengan melihat perubahan kecerahan dari LED Kit iTCLab. LED yang ditambahkan pada Kit iTCLab, bukan LED yang ada pada mikrokontroller ESP32 (saat mencoba program Blink).



**Gambar 3. 26. Perubahan Kecerahan dari LED Kit iTCLab Sesuai Siklus Kerja PWM**

### E. Pemrograman Dasar Kit iTCLab dengan Python

Pemrograman Kit iTCLab berikutnya, digunakan Bahasa Pemrograman Python, yang digabungkan dengan Bahasa Pemrograman Arduino.

#### Sumber daya TCLab Brigham Young University (BYU)

Seperi telah disebutkan sebelumnya, bahwa lahirnya Kit iTCLab terinspirasi oleh Kit TCLab produk dari kampus BYU. Secara umum, semua sumber daya TCLab hasil riset kampus BYU, masih bisa digunakan untuk berekspeten dengan Kit iTCLab. Tentunya dengan penyesuaian seperlunya. Terutama yang berkaitan dengan pengaturan penggunaan mikrokontroller ESP32. Diantaranya perubahan pin dari mikrokontroller ESP32 yang digunakan. Kit TCLab BYU menggunakan mikrokontroller Arduino UNO.

Berikut ini, sumber daya TCLab Brigham Young University (BYU), yang masih bisa digunakan, dalam berekspeten dengan Kit iTCLab:

1. Belajar Pemrograman Python  
[https://github.com/APMonitor/begin\\_python](https://github.com/APMonitor/begin_python)
2. Program TCLab Master  
<https://apmonitor.com/pdc/uploads/Main/tclab.zip>
3. Program Arduino Master  
<https://github.com/APMonitor/arduino>
4. Program TCLab Jupyter Master  
[https://github.com/evertoncolling/tclab\\_jupyter](https://github.com/evertoncolling/tclab_jupyter)

Semua sumber daya di atas, agar bisa digunakan dalam bereksperimen dengan Kit iTCLab, maka program **tclab.py** harus diganti dengan **itclab.py**. File itclab.py, dapat di-*download* di sini:

<https://github.com/bsrahmat/itclab-03/blob/main/itclab.py>

Kemudian, perubahan pin mikrokontroller juga harus dilakukan. Berikut ini, adalah nomor pin yang sesuai dengan sinyal pada iTCLab, yang menggunakan mikrokontroller ESP32, yaitu:

1. pinT1 = 34, digunakan untuk Sensor Suhu T1.
2. pinT2 = 35, digunakan untuk Sensor Suhu T2.
3. pinQ1 = 32, digunakan untuk Pemanas Q1.
4. pinQ2 = 33, digunakan untuk Pemanas Q2.
5. pinLED = 26, digunakan untuk LED.

Selanjutnya, akan dilakukan pengujian iTCLab dengan program sederhana menggunakan Python Jupyter Notebook. Program ini akan digunakan bersama dengan program Arduino yang sebelumnya harus sudah tertanam pada Kit iTCLab. Komunikasi keduanya melalui Port Serial.

Berikut ini, file program yang dibutuhkan untuk pengujian iTCLab menggunakan Python:

1. Program arduino\_python.ino  
[https://github.com/bsrahmat/itclab-03/blob/main/arduino\\_python.ino](https://github.com/bsrahmat/itclab-03/blob/main/arduino_python.ino)

2. Program python\_testing.ipynb  
[https://github.com/bsrahmat/itclab-03/blob/main/python\\_testing.ipynb](https://github.com/bsrahmat/itclab-03/blob/main/python_testing.ipynb)
3. Program itclab.py  
<https://github.com/bsrahmat/itclab-03/blob/main/itclab.py>

Silahkan disiapkan ketiga program di atas. Program **arduino\_python.ino** silahkan di-upload ke Kit iTCLab terlebih dahulu.

Berikut ini secara lengkap, program **arduino\_python.ino**. Proses *upload* program dan ketika berhasil *di-upload* seperti terlihat pada Gambar 3.27 dan Gambar 3.28.

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

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

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

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

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

```

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

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

//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
}

```

```

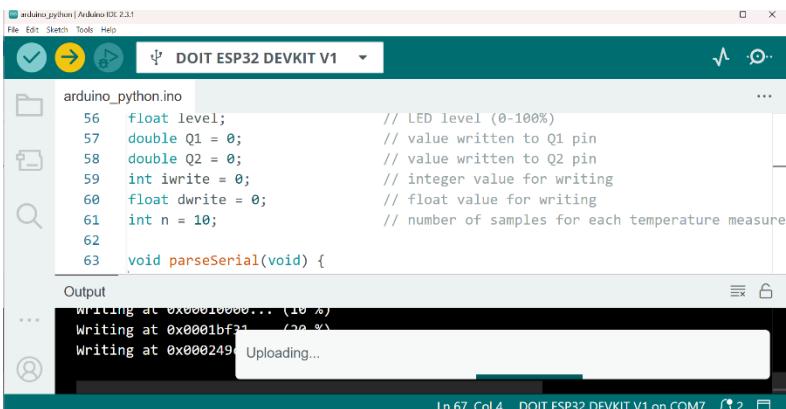
ledcSetup(ledChannel, freq, resolutionLedChannel);

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

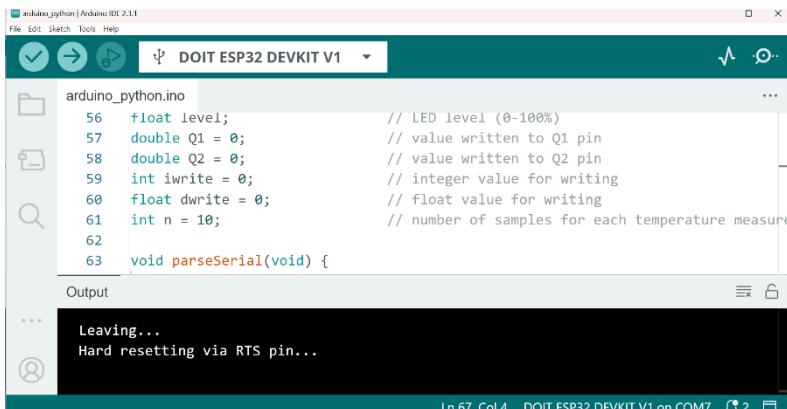
ledcWrite(Q1Channel,0);
ledcWrite(Q2Channel,0);
}

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

```



Gambar 3. 27. Proses upload Program arduino\_python.ino



Gambar 3. 28. Program arduino\_python.ino berhasil di-upload

Selanjutnya, program **itclab.py** dan **python\_testing.ipynb** silahkan diletakkan di *folder* yang sama.

Berikut ini, isi program **itclab.py**:

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

class iTCLab(object):

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

```

def findPort(self):
    found = False
    for port in list(serial.tools.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 ')
        print('/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

```

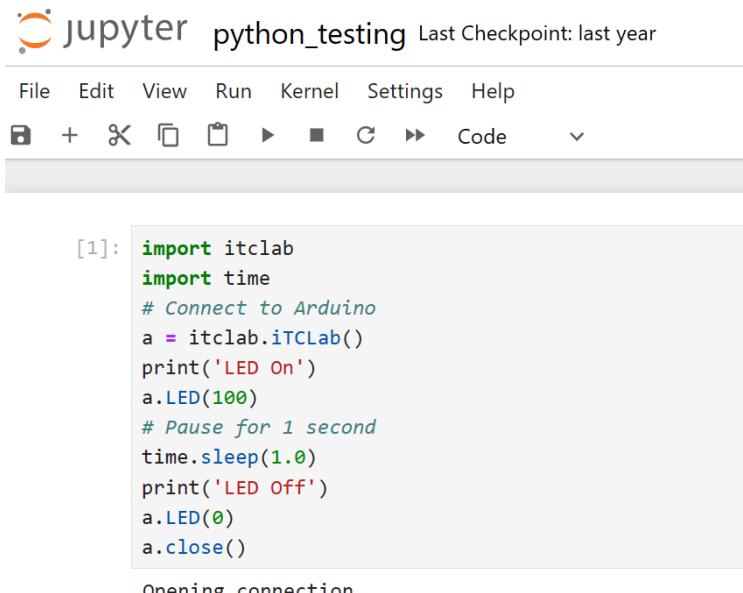
Selanjutnya, program Python Jupyter Notebook, dengan nama file program **python\_testing.ipynb**, untuk menguji Kit iTCLab, kodingnya sebagai berikut.

```

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

```

Silahkan dijalankan di lingkungan Jupyter notebook, dengan kondisi Kit iTCLab terhubung ke PC atau Laptop (dengan program **arduino\_python.ino** sudah tertanam di dalamnya). Maka seharusnya hasilnya seperti terlihat pada Gambar 3.29.



The screenshot shows a Jupyter Notebook interface. The title bar says "jupyter python\_testing Last Checkpoint: last year". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. The code cell [1] contains the following Python script:

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

Opening connection
iTCLab connected via Arduino on port COM7
LED On
LED Off
Arduino disconnected successfully
```

**Gambar 3. 29. Hasil pengujian Kit iTCLab dengan Python**

Boleh dicoba dengan program kedua sebagai berikut:

```
import itclab
import time

# Connect to Arduino
a = itclab.iTCLab()

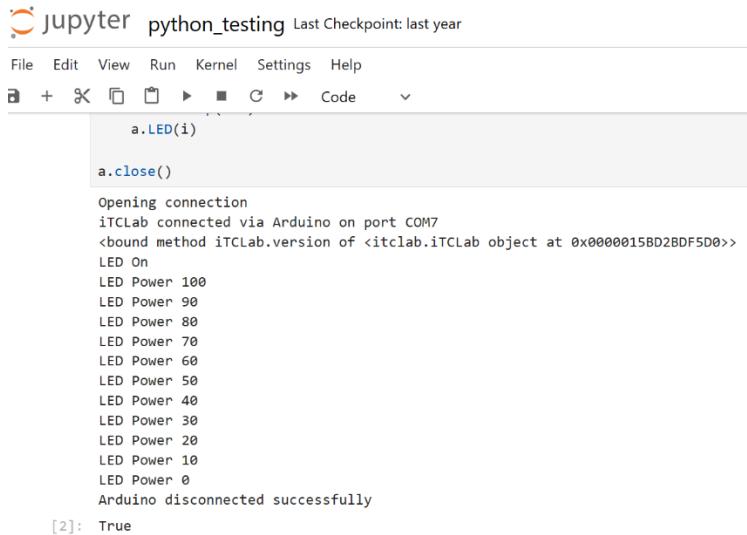
# Get Version
print(a.version)

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

```
# Taper LED off
for i in range(100,-1,-10):
    print('LED Power ' + str(i))
    time.sleep(0.5)
    a.LED(i)

a.close()
```

Maka seharusnya hasilnya, seperti terlihat pada Gambar 3.30. Silahkan sambil dilihat perubahan kecerahan LED yang ada pada Kit iTCLab.



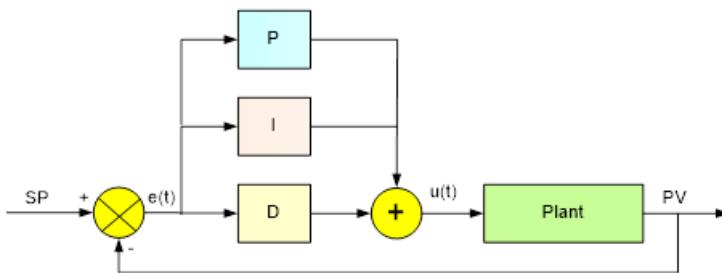
```
jupyter python_testing Last Checkpoint: last year
File Edit View Run Kernel Settings Help
a + X □ ▶ ■ C ▷ Code ▾
a.LED(i)
a.close()
Opening connection
iTCLab connected via Arduino on port COM7
<bound method iTCLab.version of <ipclab.iTCLab object at 0x0000015BD2BDF5D0>>
LED On
LED Power 100
LED Power 90
LED Power 80
LED Power 70
LED Power 60
LED Power 50
LED Power 40
LED Power 30
LED Power 20
LED Power 10
LED Power 0
Arduino disconnected successfully
[2]: True
```

**Gambar 3. 30. Hasil pengujian Kit iTCLab dengan Python Lanjutan**

# BAB 4 | REVIEW SISTEM KENDALI PID

## A. Konsep Sistem Kendali PID

Sistem Kendali di industri yang paling terkenal adalah Sistem Kendali Proporsional Integral dan Derivatif (PID). Diagram blok Sistem Kendali PID seperti diperlihatkan pada Gambar 4.1. PID menggabungkan tiga aksi kendali proporsional, integral dan derivatif. Masing-masing aksi kendali ini mempunyai keunggulan-keunggulan tertentu, dimana aksi kendali proporsional mempunyai keunggulan *rise time* yang sangat cepat, aksi kendali integral mempunyai keunggulan untuk memperkecil *error*, dan aksi kendali derivatif mempunyai keunggulan untuk memperkecil *error* atau meredam *overshoot*. Tujuan penggabungan ketiga aksi kendali ini agar dihasilkan keluaran dengan *risetime* yang cepat dan *error* yang kecil.



Gambar 4. 1. Sistem Kendali PID

Pengendali PID dalam kerjanya secara otomatis menyesuaikan keluaran kendali berdasarkan perbedaan antara *Setpoint* (SP) dan variabel proses yang terukur (PV), sebagai *error*

pengendalian  $e(t)$ . Nilai keluaran Pengendali  $u(t)$  ditransfer sebagai masukan sistem. Masing-masing hubungan yang digunakan seperti terlihat pada Persamaan (4.1) dan (4.2) [15]:

$$e(t) = SP - PV \quad (4.1)$$

$$u(t) = u_{bias} + K_c e(t) + \frac{K_c}{\tau_I} \int_0^t e(t) dt - K_c \tau_D \frac{d(PV)}{dt} \quad (4.2)$$

Istilah **ubias** adalah konstanta yang biasanya diatur ke nilai  $u(t)$  ketika pengendali pertama beralih dari mode manual ke mode otomatis. Ini memberikan transfer "*bumpless*" jika kesalahannya nol ketika pengendali dihidupkan. Ketiga nilai penalaan atau *tuning* untuk pengendali PID adalah gain  $K_c$ , konstanta waktu integral  $\tau_I$ , dan konstanta waktu derivatif  $\tau_D$ . Nilai  $K_c$  adalah penguat *error* proporsional dan istilah integral membuat pengendali lebih agresif dalam menanggapi *error* dari *Setpoint*. Konstanta waktu integral  $\tau_I$  (juga dikenal sebagai waktu reset integral) harus positif dan memiliki satuan waktu. Karena  $\tau_I$  semakin kecil, istilah integralnya lebih besar karena  $\tau_I$  berada di penyebut. Konstanta waktu derivatif  $\tau_D$  juga memiliki satuan waktu dan harus positif.

*Setpoint* (SP) adalah nilai target, dan variabel proses (PV) adalah nilai terukur yang mungkin menyimpang dari nilai yang diinginkan. Kesalahan dari *setpoint* adalah perbedaan antara SP dan PV dan didefinisikan sebagai *error*,  $e(t) = SP - PV$ .

Selanjutnya, untuk keperluan imlementasi digunakan Pengendali PID Diskrit. Pengendali digital diimplementasikan dengan periode sampling diskrit. Bentuk terpisah dari persamaan PID diperlukan untuk memperkirakan integral dan derivatif dari *error*. Modifikasi ini menggantikan bentuk kontinyu integral dengan penjumlahan *error* dan menggunakan  $\Delta t$  sebagai waktu antara contoh pengambilan sampel dan  $n_t$  sebagai jumlah contoh pengambilan sampel. Ini juga menggantikan derivatif dengan versi turunannya atau metode lain yang disaring untuk memperkirakan kemiringan instan (PV). Persamaan (4.2) jika dinyatakan kedalam bentuk digital seperti diperlihatkan pada Persamaan (4.3) [15]:

$$u(t) = u_{bias} + K_c e(t) + \frac{K_c}{\tau_I} \sum_{i=1}^{n_t} e_i(t) \Delta t - K_c \tau_D \frac{PV_{n_t} - PV_{n_t-1}}{\Delta t} \quad (4.3)$$

Dari Persamaan (4.3) dapat dilihat tiga parameter penentu keberhasilan proses kendali, yaitu gain  $K_c$ , konstanta waktu integral  $\tau_I$  dan konstanta waktu derivatif  $\tau_D$ . Proses pencarian atau pengaturan atau penalaan agar diperoleh nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  terbaik ini umumnya disebut dengan proses penalaan atau *tuning*. Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID memberikan peluang pengaturan secara manual atau *trial and error* maupun pengaturan secara terprogram menggunakan bantuan algoritma sistem cerdas.

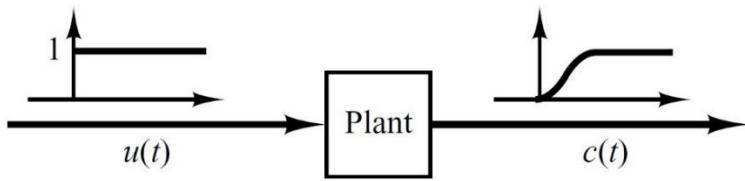
## B. Proses Penalaan Parameter PID

Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menjadi tantangan tersendiri di dunia kontrol. Banyak metode telah dikembangkan selama bertahun-tahun untuk menjawab permasalahan mengenai bagaimana menyetel pengendali PID. Tetapi yang paling terkenal adalah metode Ziegler-Nichols.

Proses pemilihan parameter pengendali untuk memenuhi spesifikasi kinerja yang ditentukan dikenal sebagai penyetelan atau penalaan pengendali (*controller tuning*). Ziegler dan Nichols mengusulkan aturan untuk menentukan nilai  $K_c$ ,  $\tau_I$ , dan  $\tau_D$  berdasarkan karakteristik respon transien dari sebuah sistem. Terdapat dua metode dalam aturan penalaan Ziegler-Nichols, yang disebut sebagai metode pertama dan metode kedua [16].

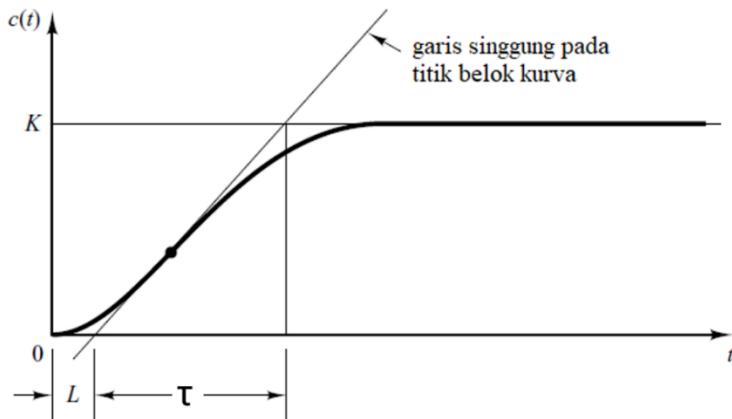
### Metode Pertama

Pada metode pertama, tanggapan atau respon sistem atau plant terhadap masukan satu satuan fungsi step ditentukan secara eksperimental, seperti yang ditunjukkan pada Gambar 4.2. Jika respon yang dihasilkan melalui eksperimen ataupun dari simulasi dinamik sistem menunjukkan kurva berbentuk S, maka metode ini dapat diterapkan [16].



**Gambar 4. 2. Respon Sistem terhadap Masukan Fungsi Step**

Kurva berbentuk S pada Gambar 4.2, dapat dicirikan oleh dua konstanta yaitu waktu tunda  $L$  dan konstanta waktu  $\tau$ . Kedua konstanta ini ditentukan berdasarkan garis singgung pada titik belok kurva S, perpotongan garis singgung dengan sumbu waktu dan garis  $c(t) = K$  menghasilkan konstanta  $L$  dan  $\tau$  [16].



**Gambar 4. 3. Respon Sistem Berbentuk Kurva S**

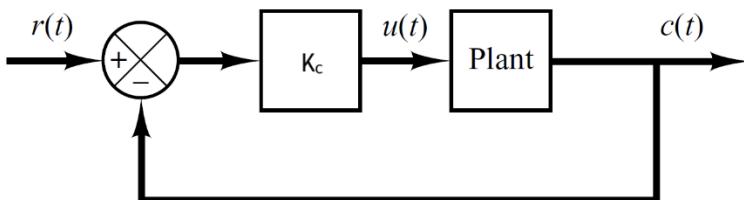
Parameter  $K_c$ ,  $\tau_I$ , dan  $\tau_D$  yang diperoleh dari metode pertama Ziegler-Nichols ditentukan berdasarkan Tabel 4.1.

**Tabel 4. 1. Aturan Ziegler-Nichols Metode Pertama**

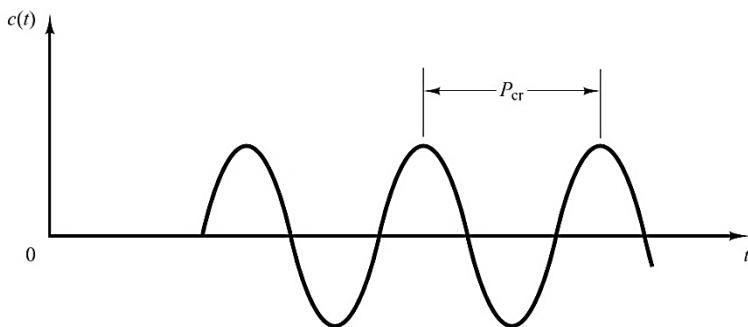
Pengendali	$K_c$	$\tau_I$	$\tau_D$
P	$\frac{\tau}{L}$	$\infty$	0
PI	$0.9 \frac{\tau}{L}$	$\frac{L}{0.3}$	0
PID	$1.2 \frac{\tau}{L}$	$2L$	$0.5L$

## Metode Kedua

Pada metode kedua Ziegler-Nichols, mula-mula diatur  $\tau_I = \infty$  dan  $\tau_D = 0$ , sehingga sistem hanya bekerja dengan pengendali proporsional seperti yang ditunjukkan pada Gambar 4.4. Nilai  $K_c$  ditingkatkan dari 0 ke nilai kritis  $K_{cr}$  sehingga diperoleh keluaran yang mulai berosilasi dengan amplitudo yang konstan secara terus menerus. Nilai penguat kritis  $K_{cr}$  dan periode  $P_{cr}$  yang bersesuaian ditentukan secara eksperimental seperti yang diperlihatkan pada Gambar 4.5. Tabel 4.2 menunjukkan rumusan yang diajukan Ziegler-Nichols untuk digunakan dalam menentukan parameter  $K_c$ ,  $\tau_I$ , dan  $\tau_D$ .



Gambar 4. 4. Sistem Lup-Tertutup dengan Pengendali Proporsional



Gambar 4. 5. Periode  $P_{cr}$  dari Osilasi Keluaran Sistem

**Tabel 4. 2. Aturan Ziegler-Nichols metode kedua**

Pengendali	$K_c$	$\tau_I$	$\tau_D$
P	$0.5K_{cr}$	$\infty$	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Metode penalaan Ziegler-Nichols bertujuan untuk mencapai maksimum overshoot 25% terhadap masukan step [16]. Nilai parameter  $K_c$ ,  $\tau_I$ , dan  $\tau_D$  berdasarkan pada respon step sistem atau plant secara eksperimental atau berdasarkan nilai  $K_c$  yang dihasilkan dalam kestabilan marginal bila hanya aksi kendali proporsional yang digunakan.

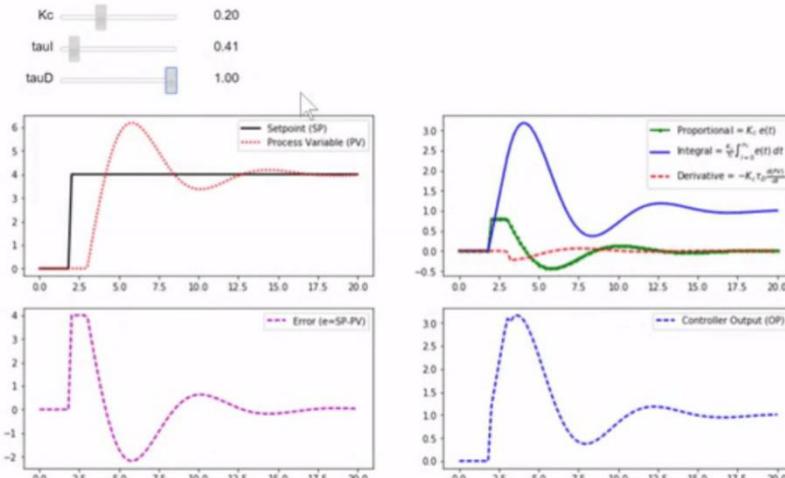
# BAB |

# 5 |

# PEMROGRAMAN PID

## A. Pemrograman PID Dasar

Setelah mengetahui bagaimana Sistem Kendali Proporsional Integral dan Derivatif (PID) bekerja, selanjutnya dapat dikembangkan bagaimana teknik pemrograman PID. Pengaruh perubahan nilai parameter gain  $K_c$ , konstanta waktu integral  $\tau_I$  dan konstanta waktu derivatif  $\tau_D$ , terhadap kinerja Sistem Kendali PID, dapat dilihat dari simulasi berikut [15].



Gambar 5. 1. Simulasi Penalaan Parameter PID

Untuk merealisasikan bagaimana proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  dari pengendali PID, berikut ini program simulasi sistem kendali PID. Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  dilakukan secara manual. Pemrograman Python untuk simulasi

pengendalian menggunakan pengendali PID dengan proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  secara manual diperlihatkan pada skrip program berikut.

```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.integrate import odeint
import ipywidgets as wg
from IPython.display import display
n = 100 # time points to plot
tf = 20.0 # final time
SP_start = 2.0 # time of set point change

def process(y,t,u):
    Kp = 4.0
    taup = 3.0
    thetap = 1.0
    if t<(thetap+SP_start):
        dydt = 0.0 # time delay
    else:
        dydt = (1.0/taup) * (-y + Kp * u)
    return dydt

def pidPlot(Kc,tauI,tauD):
    t = np.linspace(0,tf,n) # create time vector
    P= np.zeros(n)      # initialize proportional term
    I = np.zeros(n)      # initialize integral term
    D = np.zeros(n)      # initialize derivative term
    e = np.zeros(n)      # initialize error
    OP = np.zeros(n)     # initialize controller output
    PV = np.zeros(n)     # initialize process variable
    SP = np.zeros(n)     # initialize setpoint
    SP_step = int(SP_start/(tf/(n-1))+1) # setpoint start
    SP[0:SP_step] = 0.0 # define setpoint
    SP[SP_step:n] = 4.0 # step up
    y0 = 0.0            # initial condition
    # loop through all time steps
    for i in range(1,n):
```

```

# simulate process for one time step
ts = [t[i-1],t[i]]      # time interval
y = odeint(process,y0,ts,args=(OP[i-1],)) # compute next step
y0 = y[1]                # record new initial condition
# calculate new OP with PID
PV[i] = y[1]              # record PV
e[i] = SP[i] - PV[i]      # calculate error = SP - PV
dt = t[i] - t[i-1]        # calculate time step
P[i] = Kc * e[i]          # calculate proportional term
I[i] = I[i-1] + (Kc/taul) * e[i] * dt # calculate integral term
D[i] = -Kc * tauD * (PV[i]-PV[i-1])/dt # calculate derivative term
OP[i] = P[i] + I[i] + D[i] # calculate new controller output

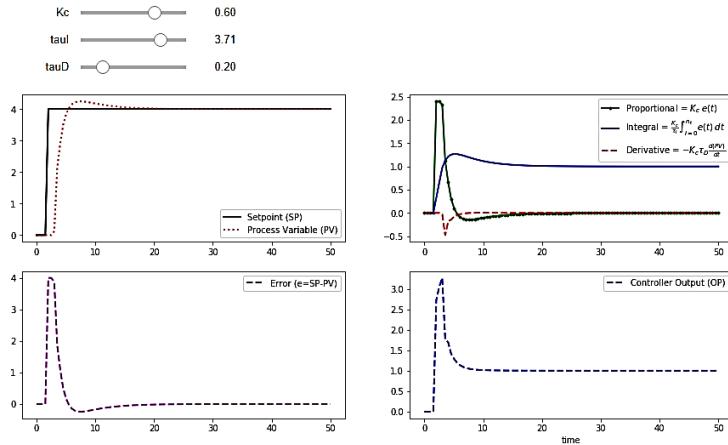
# plot PID response
plt.figure(1,figsize=(15,7))
plt.subplot(2,2,1)
plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')
plt.plot(t,PV,'r:',linewidth=2,label='Process Variable (PV)')
plt.legend(loc='best')
plt.subplot(2,2,2)
plt.plot(t,P,'g.-',linewidth=2,label=r'Proportional = $K_c \cdot e(t)$')
plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\frac{K_c}{\tau_I} \int_{t_0}^t e(t) dt$')
plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c / \tau_D \frac{d(PV)}{dt}$')
plt.legend(loc='best')
plt.subplot(2,2,3)
plt.plot(t,e,'m--',linewidth=2,label='Error (e=SP-PV)')
plt.legend(loc='best')
plt.subplot(2,2,4)
plt.plot(t,OP,'b--',linewidth=2,label='Controller Output (OP)')
plt.legend(loc='best')
plt.xlabel('time')

Kc_slide = wg.FloatSlider(value=0.1,min=-0.2,max=1.0,step=0.05)
taul_slide = wg.FloatSlider(value=4.0,min=0.01,max=5.0,step=0.1)
tauD_slide = wg.FloatSlider(value=0.0,min=0.0,max=1.0,step=0.1)
wg.interact(pidPlot,           Kc=Kc_slide,           tauI=taul_slide,
tauD=tauD_slide)

```

Program simulasi PID di atas ([pid\\_simulation.ipynb](#)), juga bisa diunduh di alamat berikut:  
[https://github.com/bsrahmat/itclab-04/blob/main/pid\\_simulation.ipynb](https://github.com/bsrahmat/itclab-04/blob/main/pid_simulation.ipynb)

Jika dijalankan, hasilnya seperti terlihat pada Gambar 5.2.



**Gambar 5. 2. Contoh hasil proses penalaan nilai parameter  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pengendali PID terhadap keluaran sistem**

## B. Pemrograman PID-iTCLab dengan Arduino

Seperti yang telah dibahas pada Bab 4 sebelumnya, bahwa pengendali PID dalam kerjanya secara otomatis menyesuaikan keluaran kendali berdasarkan perbedaan antara *Setpoint* (SP) dan variabel proses yang terukur (PV), sebagai *error* pengendalian  $e(t)$ . Nilai keluaran Pengendali  $u(t)$  ditransfer sebagai masukan sistem. Persamaan yang digunakan, seperti yang diperlihatkan pada Persamaan (4.1). Selanjutnya, sesuai dengan cara kerja pengendali PID tersebut, dibuat program PID yang direalisasikan pada Kit iTCLab menggunakan bahasa pemrograman Arduino.

Berikut ini, skrip program pengendali PID pada Kit iTCLab menggunakan program Arduino (**PID\_Arduino.ino**).

```
/*
 * Program : PID-iTCLab Programming Using Arduino
 * By      : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI,
 *           Assoc. Prof. Dr. Muljono, S.Si, M.Kom, et al
 * Pro. Team : i-ot.net, io-t.net
 * R. Group : Intelligent Control, Robotics and Automation Systems Research Group
 * Univ.    : Universitas Pembangunan Nasional "Veteran" Jawa Timur
 * Country  : Indonesia
 */

#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, op0, 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);

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

```

```

cel1 = degC1/10;

Serial.print("Temperature T1: ");
Serial.print(cel); // print temperature T1 in Celsius
Serial.print("°C");
Serial.print(" ~ "); // separator Celsius-Fahrenheit
Serial.print("Temperature T2: ");
Serial.print(cell1); // 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();
}
}

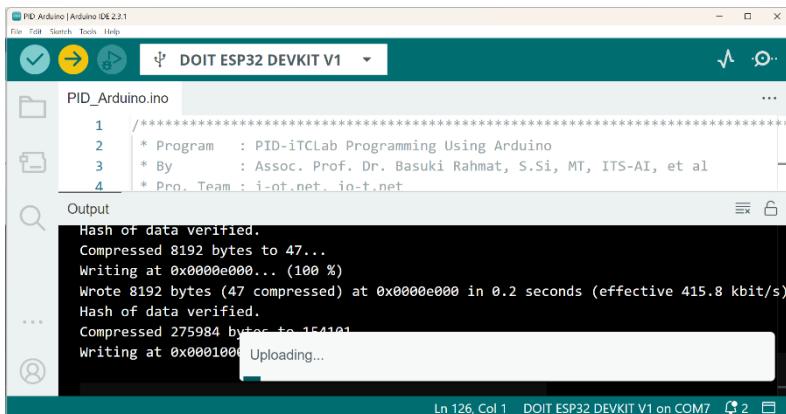
pv = cel; // Temperature T1
dt = (new_ts - ts) / 1000.0;
ts = new_ts;
op = pid(sp,pv,pv_last,ierr,dt);
ledcWrite(Q1Channel,op);
pv_last = pv;

delay (200);
}
}

```

Program **PID\_Arduino.ino** tersebut juga dapat diunduh pada alamat berikut:  
[https://github.com/bsrahmat/itclab-05/blob/main/PID\\_Arduino.ino](https://github.com/bsrahmat/itclab-05/blob/main/PID_Arduino.ino)

Silahkan program tersebut di-*upload* ke Kit iTCLab sesuai pengaturan (*board*, port, dll) yang dijelaskan pada Bab sebelumnya. Tunggu sampai proses *upload* selesai. Seperti terlihat pada Gambar 5.3 dan Gambar 5.4 berikut.

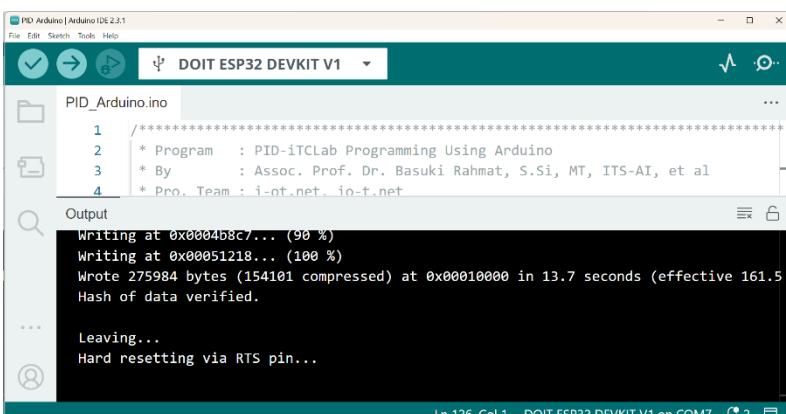


The screenshot shows the Arduino IDE interface with the title bar "PID\_Arduino | Arduino IDE 2.3.1". The central workspace displays the code for "PID\_Arduino.ino". The code includes a header with the program name, author, and team information. Below the code, the "Output" tab is active, showing the progress of the upload. The output window displays the following text:

```
Hash of data verified.  
Compressed 8192 bytes to 47...  
Writing at 0x0000e000... (100 %)  
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.2 seconds (effective 415.8 kbit/s)  
Hash of data verified.  
Compressed 275984 bytes to 154101  
Writing at 0x00010000 Uploading...
```

At the bottom right of the output window, there is a progress bar indicating the upload progress. The status bar at the bottom of the IDE shows "Ln 126, Col 1 DOIT ESP32 DEVKIT V1 on COM7".

Gambar 5. 3. Proses upload program PID\_Arduino.ino



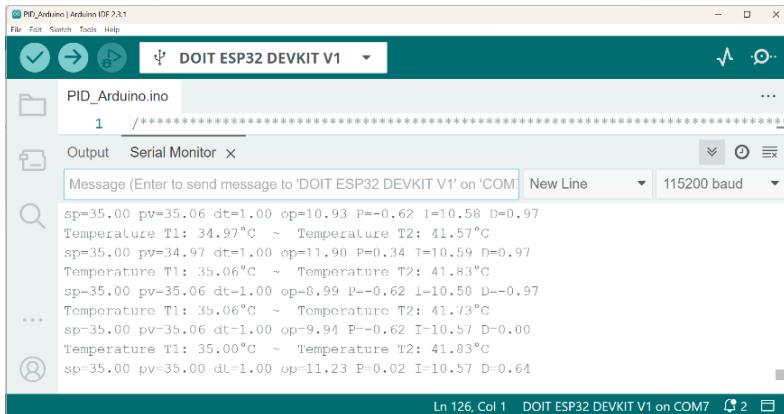
This screenshot shows the Arduino IDE after the upload process has completed successfully. The "Output" tab is still active, displaying the final upload results. The output window shows:

```
Writing at 0x0000458c... (90 %)  
Writing at 0x00051218... (100 %)  
Wrote 275984 bytes (154101 compressed) at 0x00010000 in 13.7 seconds (effective 161.5 kbit/s)  
Hash of data verified.  
Leaving...  
Hard resetting via RTS pin...
```

The progress bar in the output window is now fully green, indicating completion. The status bar at the bottom right shows "Ln 126, Col 1 DOIT ESP32 DEVKIT V1 on COM7".

Gambar 5. 4. Proses upload PID\_Arduino.ino sukses

Setelah proses *upload* program **PID\_Arduino.ino** ke Kit iTCLab berhasil. Silahkan cek hasilnya melalui serial monitor. Jika power adaptor terhubung ke listrik, seharusnya hasilnya terlihat seperti pada Gambar 5.5.



The screenshot shows the Arduino IDE interface with the title bar "PID\_Arduino | Arduino IDE 2.3.1". The main window displays the code for "PID\_Arduino.ino". Below the code editor is the "Serial Monitor" tab, which is active. The monitor window shows the following text output:

```
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM' New Line 115200 baud
sp=35.00 pv=35.06 dt=1.00 op=-0.62 I=10.58 D=0.97
Temperature T1: 34.97°C ~ Temperature T2: 41.57°C
sp=35.00 pv=34.97 dt=1.00 op=11.90 P=0.34 I=10.59 D=0.97
Temperature T1: 35.06°C ~ Temperature T2: 41.83°C
sp=35.00 pv=35.06 dt=1.00 op=8.99 P=-0.62 I=10.58 D=-0.97
Temperatura T1: 35.06°C ~ Temperatura T2: 41.73°C
sp=35.00 pv=35.06 dt=1.00 op=9.94 P=-0.62 I=10.57 D=0.00
Temperature T1: 35.00°C ~ Temperature T2: 41.83°C
sp=35.00 pv=35.00 dt=1.00 op=11.23 P=0.02 I=10.57 D=0.64
```

At the bottom of the monitor window, it says "Ln 126, Col 1 DOIT ESP32 DEVKIT V1 on COM".

Gambar 5.5. Hasil pengendalian PID pada Kit iTCLab

Dari Gambar 5.5, terlihat hasil pembacaan Sensor Suhu T1 dan T2. Karena memang terdapat 2 (dua) sensor suhu, dan 2 (dua) *heater* atau pemanas yang digunakan pada Kit iTCLab. Dalam contoh pemrograman PID dengan arduino ini, kita fokus mengendalikan salah satu saja. Dalam contoh ini, pada *heater* 1 (pemanas 1), yang terbaca oleh Sensor T1. Sedangkan *heater* 2 atau pemanas 2 tidak kita kendalikan. Tampak dari hasil di atas, pembacaan sensor T1 menuju nilai Set Point (SP) yang kita harapkan. Sedangkan pembacaan sensor T2, tidak kita kendalikan. Membaca apa adanya akibat pengaruh pemanasan *heater* 2.

### C. Pemrograman PID-iTCLab dengan Arduino-Python

Jika pada program **PID\_Arduino.ino** menampilkan hasil seperti terlihat pada Gambar 5.5, maka agak sulit dilakukan proses analisis hasil pengendalian tersebut. Sehingga dibutuhkan hasil tampilan setidaknya dalam bentuk grafik. Solusi yang paling mudah, tentunya hasil pengendalian ditampilkan pada program Python.

Pada bab sebelumnya, telah berhasil dilakukan penggabungan bahasa pemrograman arduino dan Python menggunakan Kit iTCLab. Di sub bab ini, akan dikembangkan untuk pengendalian PID.

Seperti biasa, untuk keperluan ini, terlebih dahulu disiapkan program arduino yang di-upload ke Kit iTCLab. Kemudian disiapkan program Python Jupyter Notebook untuk menjalankan sistem kendali PID. Harapannya nanti dapat menampilkan hasil pengendaliannya dalam bentuk grafik. Jangan lupa juga. Seperti yang telah dijelaskan di bab sebelumnya, di folder yang sama dengan program python, ditaruh file program **itclab.py** agar iTCLab bisa dieksekusi.

Berikut ini program **PID\_Python.ino**.

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

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

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

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

```
*/
```

```

#include <Arduino.h>

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

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

// setting PWM properties
const int freq = 5000; //5000
const int ledChannel = 0;
const int Q1Channel = 1;
const int Q2Channel = 2;
const int resolutionLedChannel = 8; //Resolution 8,10,12,15
const int resolutionQ1Channel = 8; //Resolution 8,10,12,15
const int resolutionQ2Channel = 8; //Resolution 8,10,12,15

const double 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);
}

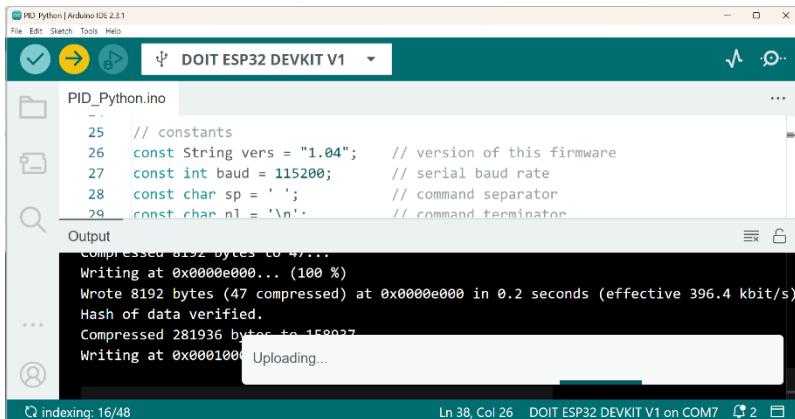
// arduino main event loop
void loop() {
    parseSerial();
    dispatchCommand();
    checkTemp();
}

```

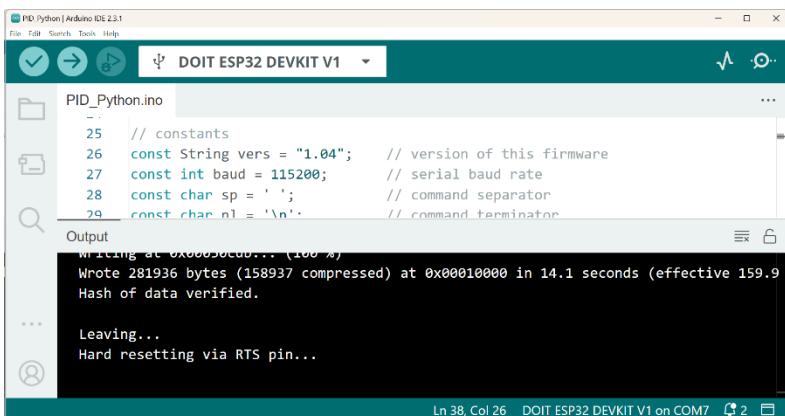
Program **PID\_Python.ino** tersebut juga dapat diunduh pada alamat berikut:

[https://github.com/bsrahmat/itclab-06/blob/main/PID\\_Python.ino](https://github.com/bsrahmat/itclab-06/blob/main/PID_Python.ino)

Silahkan *di-upload* ke Kit iTCLab, sampai berhasil seperti terlihat pada Gambar 5.6 dan Gambar 5.7.



**Gambar 5. 6. Proses Upload PID\_Python.ino**



Gambar 5. 7. Proses Upload PID\_Python.ino Sukses

Berikut ini program PID\_Python.ipynb.

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

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

#####
#####
# PID Controller                                     #
#####
#####
# inputs -----
# sp = setpoint
# pv = current temperature
# pv_last = prior temperature
# ierr = integral error
```

```

# dt = time increment between measurements
# outputs -----
# op = output of the PID controller
# P = proportional contribution
# I = integral contribution
# D = derivative contribution
def pid(sp,pv,pv_last,ierr,dt):
    Kc = 10.0 # K/%Heater
    tauI = 50.0 # sec
    tauD = 1.0 # sec
    # Parameters in terms of PID coefficients
    KP = Kc
    KI = Kc/tauI
    KD = Kc*tauD
    # ubias for controller (initial heater)
    op0 = 0
    # upper and lower bounds on heater level
    ophi = 100
    opl0 = 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 < opl0 or op > ophi:
        I = I - KI * error * dt
        # clip output
        op = max(opl0,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                         #
#####
######
# Parameters
Ta = 23 + 273.15 # K
U = 10.0        # W/m^2-K
m = 4.0/1000.0  # kg
Cp = 0.5 * 1000.0 # J/kg-K
A = 12.0 / 100.0**2 # Area in m^2
alpha = 0.01      # W / % heater
eps = 0.9        # Emissivity
sigma = 5.67e-8   # Stefan-Boltzman

# Temperature State
T = x[0]

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

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

```

```

# Connect to Arduino
a = itclab.iTCLab()

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

# Run time in minutes
run_time = 15.0

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

# Temperature
# set point (degC)
Tsp1 = np.ones(loops) * 25.0
Tsp1[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}').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] = (Tp[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} '.format( \
    ':6.2f' '{:6.2f}' '{:6.2f}' '{:6.2f}').format( \
    tm[i],Tsp1[i],T1[i], \
    Q1[i],P,ierr,D))

# Plot
plt.clf()
ax=plt.subplot(4,1,1)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'r.',label=r'T_1$ measured')
plt.plot(tm[0:i],Tsp1[0:i],'k--',label=r'T_1$ set point')
plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(4,1,2)
ax.grid()
plt.plot(tm[0:i],Q1[0:i],'b-',label=r'Q_1$')
plt.ylabel('Heater')
plt.legend(loc='best')
ax=plt.subplot(4,1,3)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'r.',label=r'T_1$ measured')
plt.plot(tm[0:i],Tp[0:i],'k-',label=r'T_1$ energy balance')
plt.plot(tm[0:i],Tpl[0:i],'g-',label=r'T_1$ linear model')

```

```

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

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

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

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

a.close()

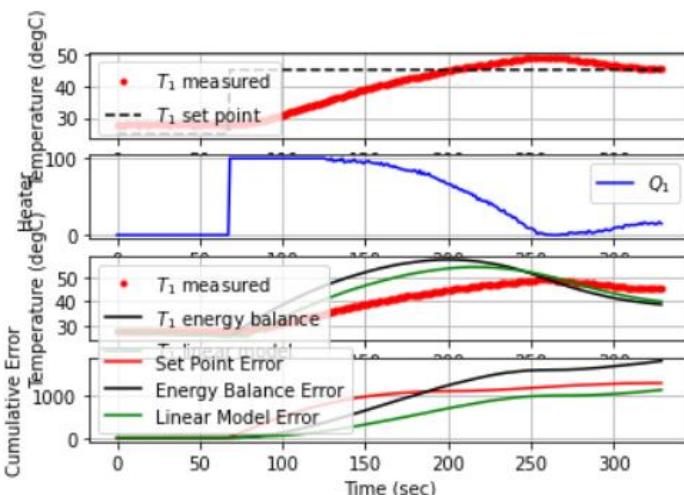
```

Program **PID\_Python.ipynb** di atas juga bisa diunduh melalui alamat berikut:

[https://github.com/bsrahmat/itclab-06/blob/main/PID\\_Python.ipynb](https://github.com/bsrahmat/itclab-06/blob/main/PID_Python.ipynb)

Silahkan dijalankan melalui Jupyter Notebook, dengan file program **itclab.py** sudah berada pada folder yang sama. Jika power adaptor dalam keadaan terhubung listrik, seharusnya hasil pengendaliannya terlihat seperti pada Gambar 5.8.

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

**Gambar 5.8. Contoh hasil pengendalian PID pada Kit iTCLab menggunakan Python**

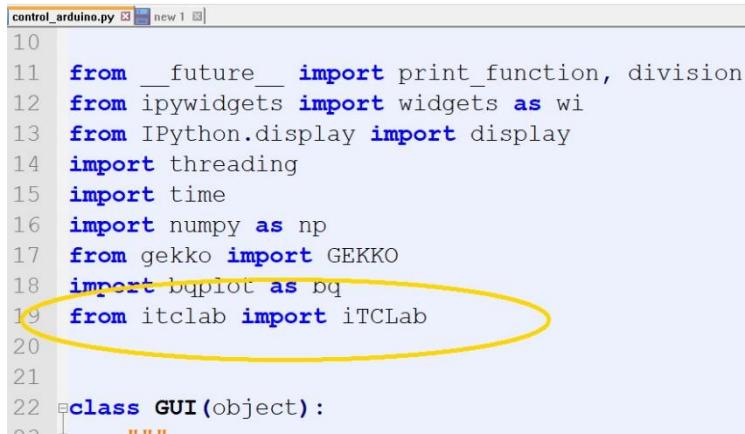
#### D. Pemrograman PID-iTCLab dengan Arduino-Python GUI

Variasi lain dari pemrograman PID pada Kit iTCLab menggunakan Python, dapat memanfaatkan PID versi *Graphical User Interface* (GUI) yang telah dikembangkan oleh Kampus Brigham Young University (BYU). Pemrograman PID-iTCLab dengan Arduino-Python GUI, seperti biasa membutuhkan

program arduino yang harus di-*upload* ke Kit iTCLab dan program Python. Program arduino yang harus di-*upload* ke Kit iTCLab masih sama dengan Pemrograman PID-iTCLab dengan Arduino-Python sebelumnya, yaitu menggunakan program **PID\_Python.ino**. Sedangkan yang berbeda, yaitu program Python Jupyter Notebook-nya.

Program Python Jupyter Notebook-nya silahkan diunduh program PID-GUI yang dikembangkan oleh Kampus BYU. Berikut ini link *download*-nya: [https://github.com/everttoncolling/tclab\\_jupyter](https://github.com/everttoncolling/tclab_jupyter)

Setelah *download* dari link tersebut, silahkan diekstrak. File modul program **itclab.py** jangan lupa dimasukkan kedalam folder hasil ekstrak tersebut. Bukalah program **control\_arduino.py**, silahkan diedit misalnya menggunakan Notepad+. Silahkan diganti from tclab import TCLab menjadi from itclab import iTCLab. Sehingga menjadi seperti terlihat pada Gambar 5.9.



```
control_arduino.py new 1
10
11     from __future__ import print_function, division
12     from ipywidgets import widgets as wi
13     from IPython.display import display
14     import threading
15     import time
16     import numpy as np
17     from gekko import GEKKO
18     import bqplot as bq
19     from itclab import iTCLab
20
21
22 class GUI(object):
23     """
```

Gambar 5. 9. Penyesuaian dengan Kit iTCLab

Dan gantilah semua TCLab menjadi iTCLab, contohnya seperti terlihat pada Gambar 5.10.

```

new 1 control_arduino.py OPEN LOOP
996
997     #####
998     #####
999     #####
1000 def _work_man(self):
1001     try:
1002         a = iTCLab()
1003     except:
1004         a.close()
1005         a = iTCLab()
1006
1007         # Parater to start each cycle
1008         self._Tc0 = np.array([
1009             a.T1 + 273.15,
1010             a.T2 + 273.15
1011         ])
1012

```

**Gambar 5. 10. Penyesuaian semua TCLab diganti dengan iTCLab**

Selanjutnya, silahkan dibuka file program **demo.ipynb**, menggunakan Python Jupyter Notebook. Maka akan tampil seperti pada Gambar 5.11.

```

In [1]: import control_demo as cd
        demo = cd.GUI()

In [2]: demo.app()

In [3]: demo.config()

```

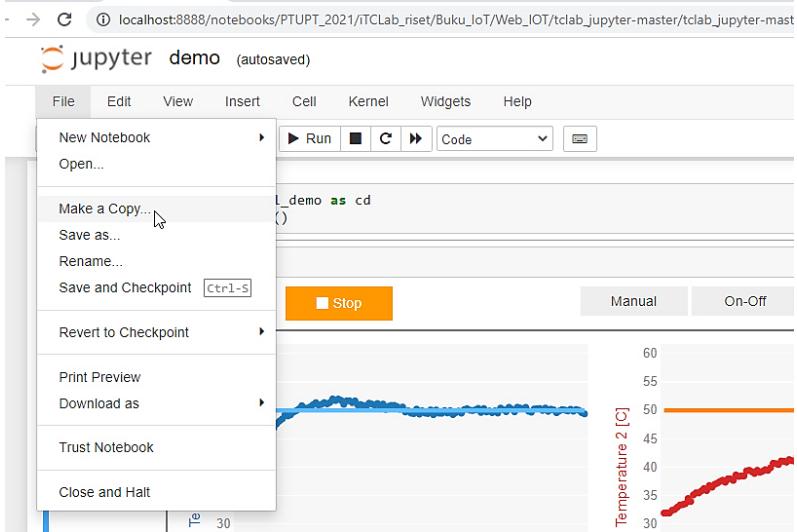
**Gambar 5. 11. Tampilan Program demo.ipynb**

Silahkan masing-masing program pada cell dijalankan. Jika misal ada *error* karena ada modul atau library yang belum diinstall, silahkan diinstall terlebih dahulu. Biasanya perintahnya dari *Command Prompt* dengan menggunakan hak akses Administrator, yaitu: **pip install nama\_modul**. Selanjutnya jika tidak ada *error*. Silahkan jalankan setiap cell, dan arahkan Tab masing-masing ke Tab PID. Hasil tampilannya seharusnya terlihat seperti pada Gambar 5.12.



Gambar 5.12. Contoh Hasil Pengendalian Demo PID Python GUI

Setelah berhasil pada program Demo PID di atas, selanjutnya, silahkan program Demo tersebut di-copy dan diberi nama lain. Misalnya **PID\_iTCLab.ipynb**. Seperti terlihat pada Gambar 5.13.



Gambar 5. 13. Copy Program Demo PID untuk PID-iTCLab

Kemudian, silahkan ganti tulisan control\_demo dengan control\_arduino. Demikian juga, demo = cd.GUI() silahkan diganti dengan run\_control = cd.GUI(), dan demo.app() silahkan diganti dengan run\_control.app(), dan demo.config() silahkan diganti dengan run\_control.config(). Seperti terlihat pada Gambar 5.14.

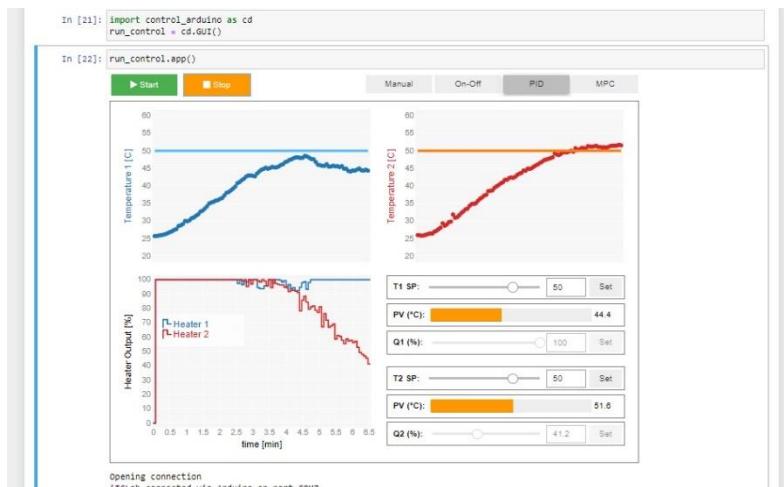
The screenshot shows a Jupyter Notebook interface. In the code editor, two cells are visible:

```
In [16]: import control_arduino as cd  
run_control = cd.GUI()  
In [17]: run_control.app()
```

Cell [17] has been run, and its output is displayed below. It shows a graphical user interface for PID control. The interface includes two temperature graphs: "Temperature 1 [C]" (blue line) and "Temperature 2 [C]" (orange line), both showing an increasing trend over time. Below the graphs are four sliders labeled T1 SP, P1 (\*C), Q1 (%), T2 SP, P2 (\*C), and Q2 (%). The T1 SP and T2 SP sliders are set to 50. The P1 (\*C) and P2 (\*C) sliders are set to 44.4 and 51.6 respectively. The Q1 (%) and Q2 (%) sliders are set to 100 and 41.2 respectively. At the bottom left, it says "Opening connection iTCLab connected via Arduino on port COM7".

Gambar 5. 14. Penyesuaian dari Demo PID ke PID-iTCLab

Kemudian, dalam posisi Kit iTCLab menggunakan kabel data terhubung ke USB laptop atau PC, dan kondisi Power Adaptor ditancap ke colokan listrik. Serta program arduino **PID\_Python.ino** sudah tertanam pada Kit iTCLab. Selanjutnya, silahkan dijalankan masing-masing *cell* program di atas. Tab diarahkan posisinya pada PID. Jika berhasil maka tampilannya, seharusnya seperti terlihat pada Gambar 5.15.



Gambar 5. 15. Contoh Hasil Pengendalian PID-iTCLab Python GUI

# BAB |

# 6 |

# PEMROGRAMAN

# IoT

## A. Review Sistem Dasar IoT

Inilah inti dari Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python ini. Jika Bab 1 sampai dengan Bab 5 sudah dikuasai dengan baik, maka sekaranglah saatnya belajar jurus paling ampuh dari Pemrograman IoT untuk Level Jilid 1 ini.

Seperti telah dijelaskan pada Bab 1, bahwa sistem dasar dari IoT terdiri dari 3 hal, yaitu:

1. Hardware/fisik (*Things*).
2. Koneksi internet, dan
3. *Cloud data center* sebagai tempat untuk menyimpan atau menjalankan aplikasinya.

Dengan demikian kebutuhannya adalah: Cloud IoT sebagai MQTT Broker, kemudian *Device*, dan Aplikasi yang berjalan di Ponsel. Dalam buku ini, MQTT Broker digunakan [hivemq.com](http://hivemq.com), *Device* digunakan Kit iTCLab, dan untuk aplikasi di Ponsel digunakan **IoT MQTT Panel**.

MQTT Broker dari hivemq, digunakan konfigurasi sebagai berikut:

Broker	:	broker.hivemq.com
TCP Port	:	1883
WebSocket Port	:	8000
TLS TCP Port	:	8883
TLS WebSocket Port	:	8884

## B. Pemrograman IoT On/Off

Pemrograman IoT yang pertama, digunakan untuk pemantauan dan pengendalian suhu Kit iTCLab secara On/Off melalui IoT menggunakan Arduino. Serta pemantauannya melalui Ponsel menggunakan IoT MQTT Panel. Program arduino yang dibutuhkan, yaitu program **IoT\_OnOff.ino**. Dan dibutuhkan pengaturan **IoT MQTT Panel** di Ponsel.

Berikut ini program **IoT\_OnOff.ino**.

```
/*
 * Program : iTCLab Kit temperature monitoring and control On/Off via IoT
 * By      : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI,
 *           Assoc. Prof. Dr. Muljono, S.Si, M.Kom, et al
 * Pro. Team : i-ot.net, io-t.net
 * R. Group : Intelligent Control, Robotics and Automation Systems Research Group
 * Univ.    : Universitas Pembangunan Nasional "Veteran" Jawa Timur
 * Country  : Indonesia
 */

#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:

    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)
client.setServer(mqttServer, mqttPort);
client.setCallback(receivedCallback);

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

    if (client.connect("iTCLab Temperature On/Off")) {

```

```

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_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 Celsius-Fahrenheit
    Serial.print(cel1); // print temperature T2 in Celsius
    Serial.println("°C");
}

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() {
    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);

Serial.print("Temperature T2: ");
Serial.print(cel1);
Serial.print(" Celcius ");
Serial.println(" send to Broker MQTT");

dtostrf(cel1, 1, 0, suhu2);
client.publish("Suhu2",suhu2);

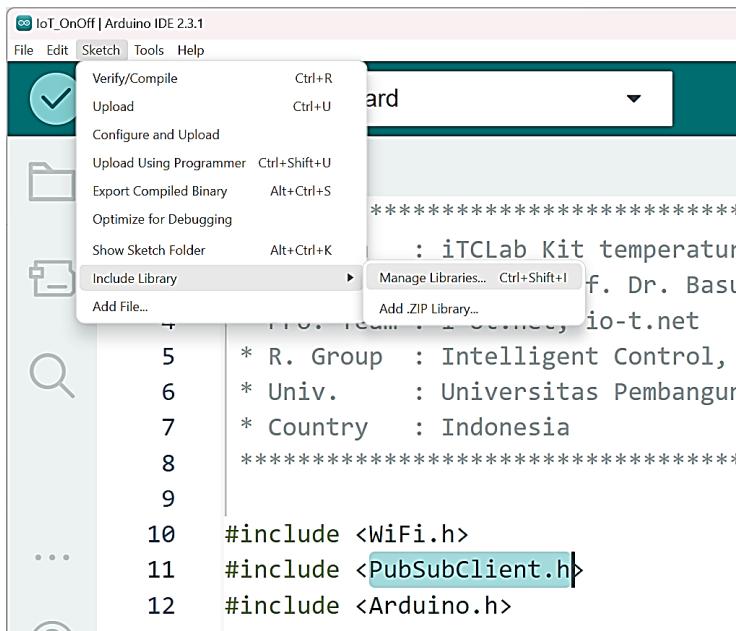
delay (200);

}
```

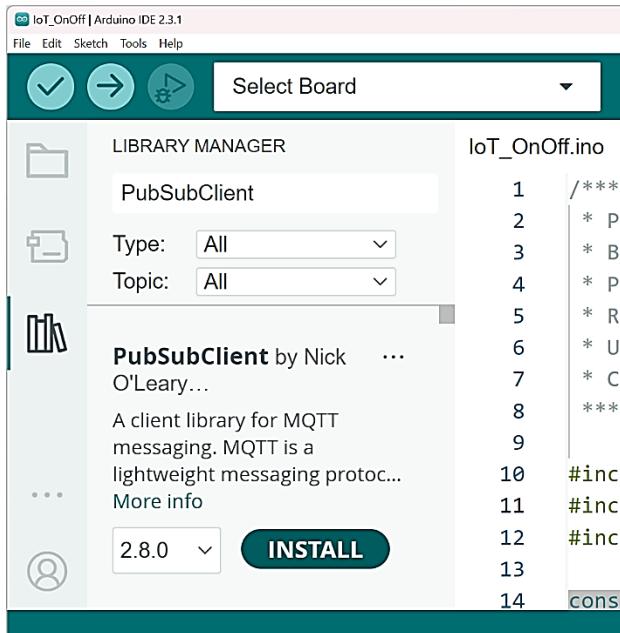
Program **IoT\_OnOff.ino** tersebut juga bisa diunduh di alamat berikut:

[https://github.com/bsrahmat/itclab-08/blob/main/IoT\\_OnOff.ino](https://github.com/bsrahmat/itclab-08/blob/main/IoT_OnOff.ino)

Program **IoT\_OnOff.ino** silahkan di-*upload* ke Kit iTCLab. Jika ada *error*, biasanya hanya perlu di-*install library* yang dibutuhkan, seperti PubSubClient. Silahkan di-*install* terlebih dahulu. Cara *install library* yang dibutuhkan cukup mudah. Tekan Menu *Sketch, include library, Manage Libraries*. Seperti ditunjukkan pada Gambar 6.1 dan Gambar 6.2. Silahkan diketikkan dan dipilih *library* yang dibutuhkan.

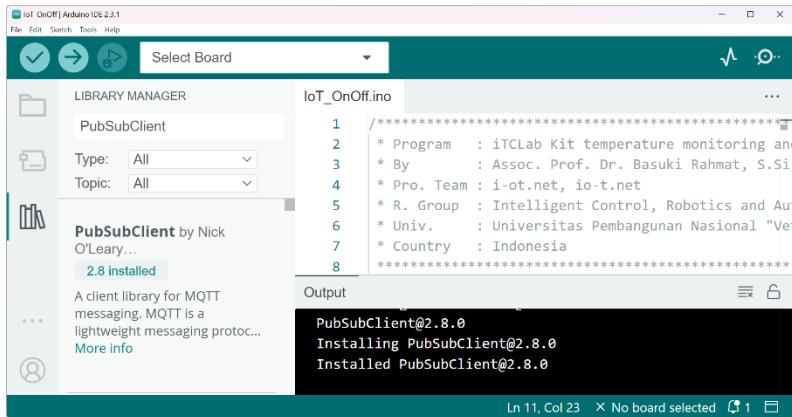


Gambar 6. 1. Cara Install Library Arduino



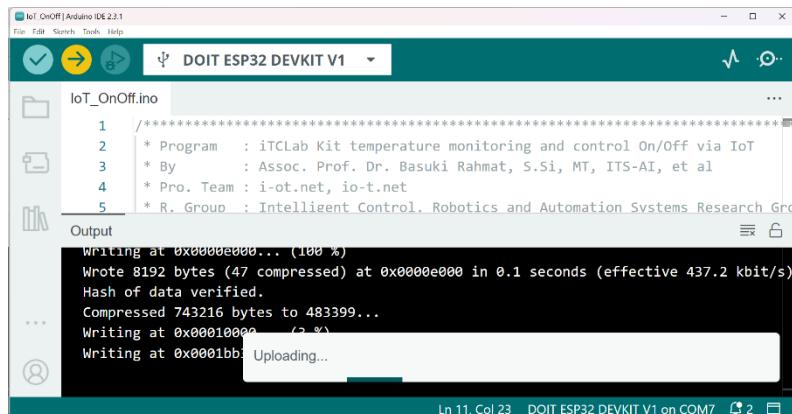
Gambar 6. 2. Tombol Install library PubSubClient

Silahkan tekan tombol *Install*, tunggu sampai proses instalasi selesai. Jika berhasil, terlihat seperti pada Gambar 6.3.



Gambar 6. 3. Library PubSubClient berhasil diinstall

Selanjutnya, silahkan *upload* program **IoT\_OnOff.ino** ke Kit iTCLab. Tunggu sampai proses *upload* selesai. Seperti terlihat pada Gambar 6.4 dan Gambar 6.5.



Gambar 6. 4. Proses upload program IoT\_OnOff.ino

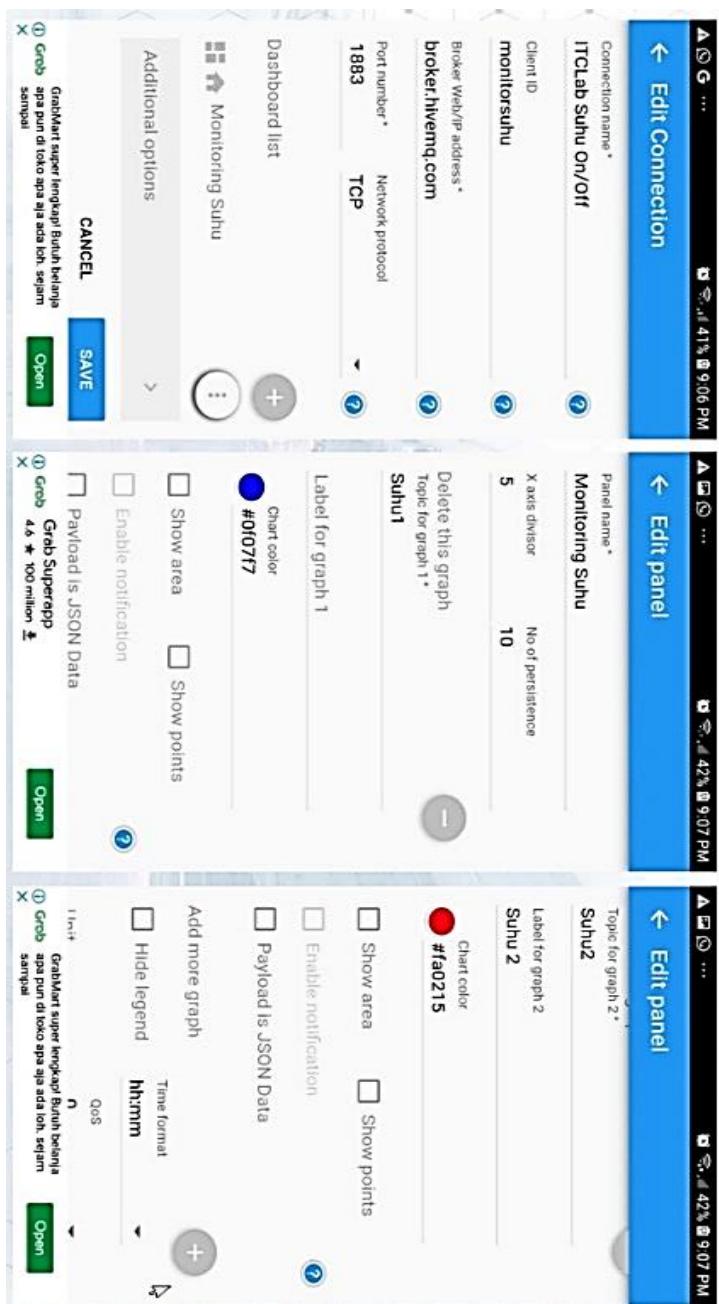
The screenshot shows the Arduino IDE interface with the title bar "IoT OnOff | Arduino IDE 2.3.1". The menu bar includes File, Edit, Sketch, Tools, Help, and a language selection dropdown. The toolbar contains icons for Open, Save, Run, Stop, and Refresh. The main area displays the code for "IoT\_OnOff.ino" and the "Output" window. The code is a C++ program with comments about the project's purpose and team. The "Output" window shows the progress of the upload: "Writing at 0x000c282d... (100 %)" followed by "Wrote 743216 bytes (483399 compressed) at 0x00010000 in 42.8 seconds (effective 139.1 Hash of data verified.)". Below this, it says "Leaving..." and "Hard resetting via RTS pin...". At the bottom right of the IDE, there is status information: "Ln 11, Col 23 DOIT ESP32 DEVKIT V1 on COM7" and a small icon.

**Gambar 6.5. Upload program IoT\_OnOff.ino sukses**

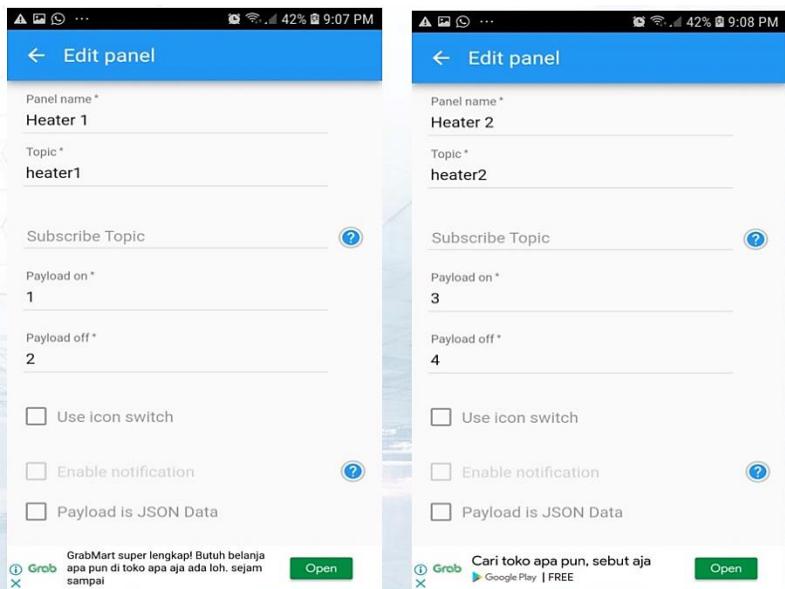
Langkah selanjutnya, silahkan dilakukan pengaturan pada aplikasi **IoT MQTT Panel** di Ponsel. Panel-panel yang dibutuhkan dan cara pengaturannya, silahkan diikuti sesuai Gambar 6.6 sampai dengan Gambar 6.8.



Gambar 6. Pengaturan IoT MQTT Panel di Ponsel (a)

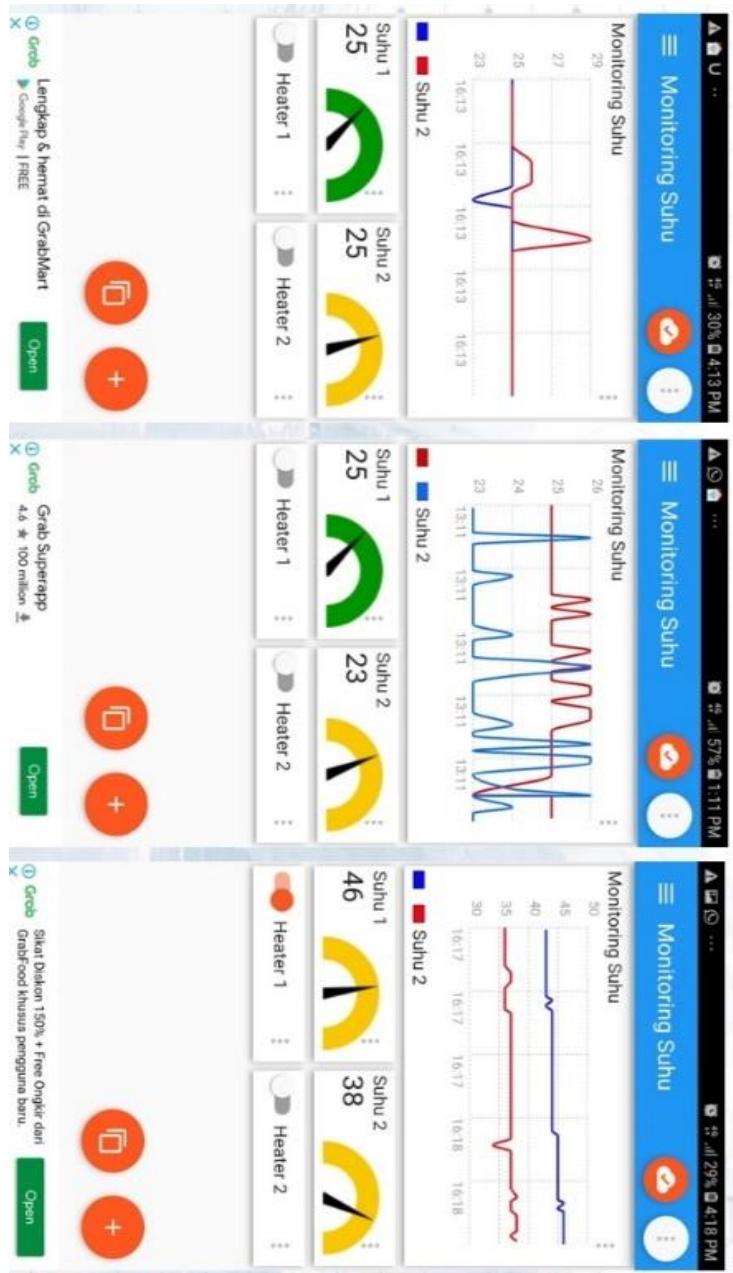


Gambar 6.7. Pengaturan IoT MQTT Panel di Ponsel (b)



**Gambar 6. 8. Pengaturan IoT MQTT Panel di Ponsel (c)**

Jika semua pengaturan telah disesuaikan, seperti pada Gambar 6.6 sampai dengan Gambar 6.8, selanjutnya tinggal dicoba untuk terhubung ke MQTT Broker hivemq.com. Jika berhasil terhubung ke MQTT Broker, maka contoh hasilnya seperti terlihat pada Gambar 6.9.



Gambar 6.9. Contoh hasil pemantauan dan pengendalian Kit iTCLab secara On/Off via IoT MQTT

Panel di Ponsel

Bagaimana? Apakah hasil pemantauan dan pengendalian Kit iTCLab secara On/Off di Ponsel anda, hasilnya kurang lebih sama seperti pada Gambar 6.9 di atas? Jika berhasil. Alhamdulillaah, anda memang layak dapat bintang 1. Tantangan bisa dilanjutkan ke program selanjutnya yang lebih menarik.

### C. Pemrograman Pemantauan PID-iTCLab dengan IoT

Tantangan selanjutnya, dibuat program Pemantauan PID-iTCLab dengan IoT. Program arduino yang dibutuhkan, yaitu **IoT\_Monitor.ino**.

Berikut ini program **IoT\_Monitor.ino**.

```
/*
* Program : PID-iTCLab Monitoring Using IoT
* By      : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI,
*           Assoc. Prof. Dr. Muljono, S.Si, M.Kom, et al
* Pro. Team : i-ot.net, io-t.net
* R. Group : Intelligent Control, Robotics and Automation Systems Research Group
* Univ.    : Universitas Pembangunan Nasional "Veteran" Jawa Timur
* Country  : Indonesia
***** */

#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
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);

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

    Serial.print("Temperature T1: ");
    Serial.print(cel); // print temperature T1 in Celsius
    Serial.print("°C");
    Serial.print(" ~ "); // separator Celsius-Fahrenheit
    Serial.print("Temperature T2: ");
    Serial.print(cel1); // print 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);
    client.publish("Suhu2",suhu2);

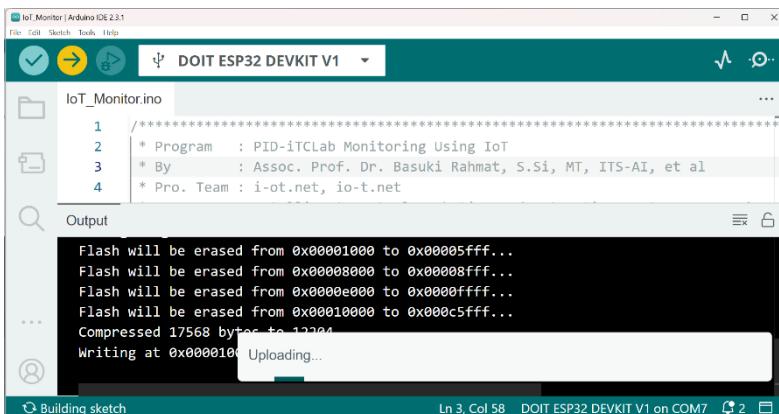
    delay (200);
}
}

```

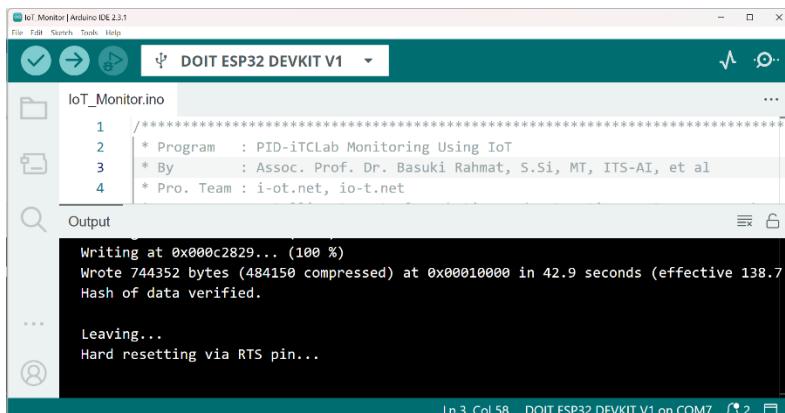
Program **IoT\_Monitor.ino** tersebut bisa juga diunduh melalui alamat berikut:

[https://github.com/bsrahmat/itclab-09/blob/main/IoT\\_Monitor.ino](https://github.com/bsrahmat/itclab-09/blob/main/IoT_Monitor.ino)

Silahkan program IoT\_Monitor.ino tersebut di-upload ke Kit iTCLab. Jika berhasil seperti terlihat pada Gambar 6.10 dan Gambar 6.11.

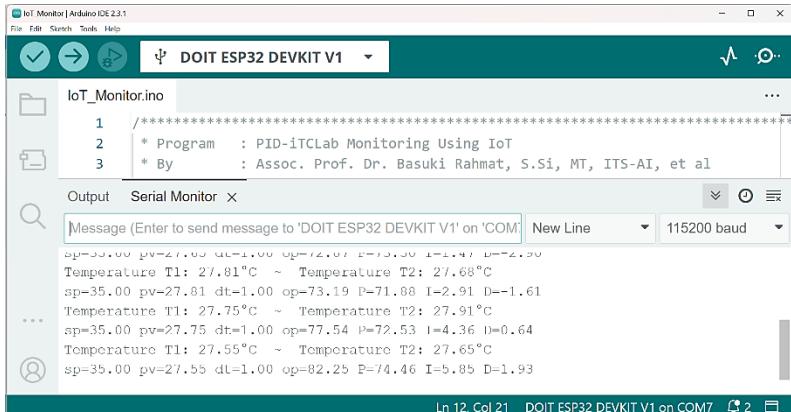


Gambar 6. 10. Proses upload program IoT\_Monitor.ino



Gambar 6. 11. Proses upload program IoT\_Monitor.ino sukses

Setelah program **IoT\_Monitor.ino** berhasil ter-upload, silahkan periksa koneksi ke wifi dan password wifi yang digunakan. Jika berhasil terhubung, silahkan cek hasilnya melalui Serial Monitor. Seharusnya akan tampil seperti terlihat pada Gambar 6.12.

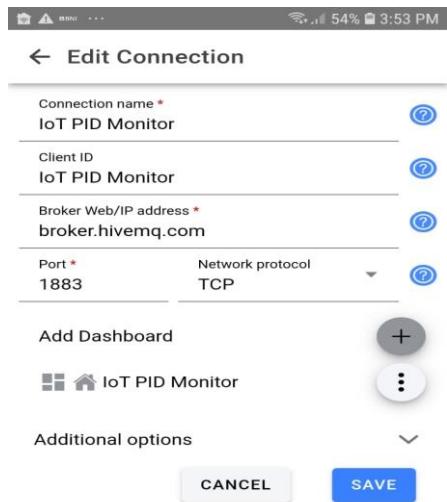


The screenshot shows the Arduino IDE Serial Monitor window titled "DOIT ESP32 DEVKIT V1". The code in the editor is for "IoT\_Monitor.ino" and includes comments about the program being a PID-iTCLab Monitoring Using IoT and being developed by Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI, et al. The serial monitor displays temperature data from two sensors, T1 and T2, with values such as 27.81°C and 27.68°C. The baud rate is set to 115200 baud.

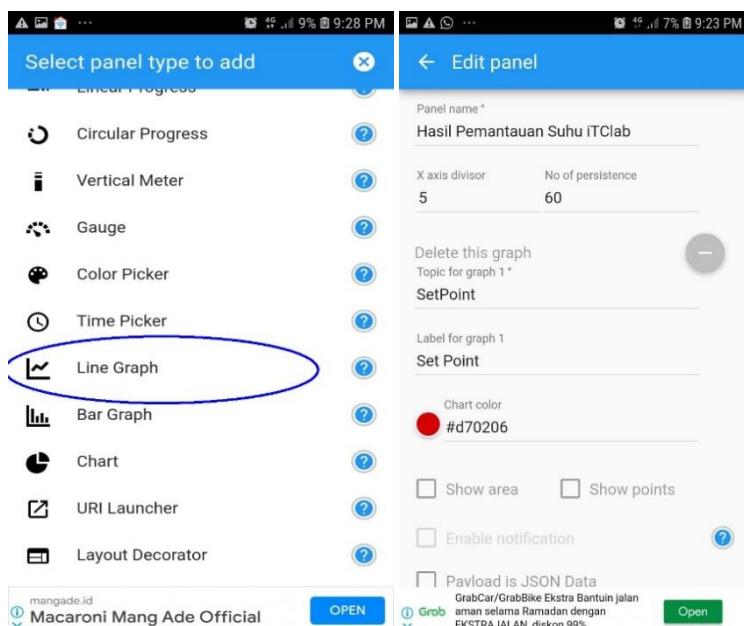
```
IoT_Monitor | Arduino IDE 2.3.1
File Edit Search Tools Help
DOIT ESP32 DEVKIT V1
IoT_Monitor.ino
1 //*****
2 /* Program : PID-iTCLab Monitoring Using IoT
3 * By : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI, et al
Output Serial Monitor X
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM1') New Line 115200 baud
sp=35.00 pv=27.00 dt=1.00 op=72.00 I=1.47 D=-2.30
Temperature T1: 27.81°C ~ Temperature T2: 27.68°C
sp=35.00 pv=27.81 dt=1.00 op=73.19 P=71.00 I=2.91 D=-1.61
Temperature T1: 27.75°C ~ Temperature T2: 27.91°C
sp=35.00 pv=27.75 dt=1.00 op=77.54 P=72.53 I=4.36 D=0.64
Temperature T1: 27.55°C ~ Temperature T2: 27.65°C
sp=35.00 pv=27.55 dt=1.00 op=82.25 P=74.46 I=5.85 D=1.93
Ln 12, Col 21 DOIT ESP32 DEVKIT V1 on COM1 2
```

**Gambar 6. 12. Contoh hasil pemantauan suhu PID-iTCLab melalui tampilan Serial Monitor di laptop**

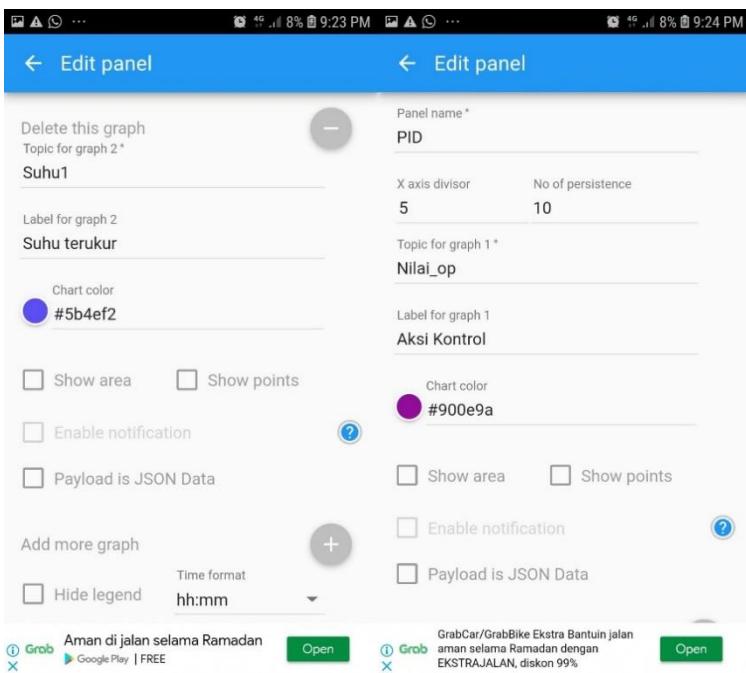
Langkah selanjutnya, silahkan dilakukan pengaturan pada aplikasi **IoT MQTT Panel** di Ponsel. Panel-panel yang dibutuhkan dan cara pengaturannya, silahkan diikuti sesuai gambar-gambar yang ada pada Gambar 6.13 sampai dengan Gambar 6.15.



Gambar 6. 13. Pengaturan di IoT MQTT Panel (a)



Gambar 6. 14. Pengaturan di IoT MQTT Panel (b)



Gambar 6. 15. Pengaturan di IoT MQTT Panel (c)

Jika semua pengaturan telah disesuaikan, seperti pada Gambar 6.13 sampai dengan Gambar 6.15, selanjutnya tinggal dicoba untuk terhubung ke MQTT Broker hivemq.com. Jika berhasil terhubung ke MQTT Broker, maka contoh hasilnya seperti terlihat pada Gambar 6.16.

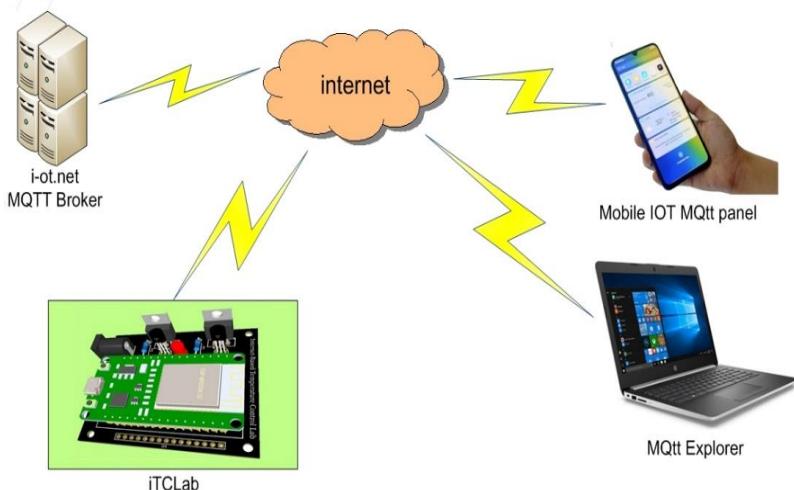


Gambar 6. 16. Contoh hasil pemantauan PID-iTCLab via IoT

Bagaimana? Apakah hasil pemantauan PID-iTCLab di Ponsel anda, hasilnya kurang lebih sama seperti pada Gambar 6.16 di atas? Jika berhasil. Alhamdulillaah, anda memang layak dapat bintang 2. Selanjutnya, tinggal menuju tantangan terakhir yang paling menarik dari Buku Jilid 1 ini. Bukan hanya pemantauan PID-iTCLab via IoT, tetapi juga pengendaliannya.

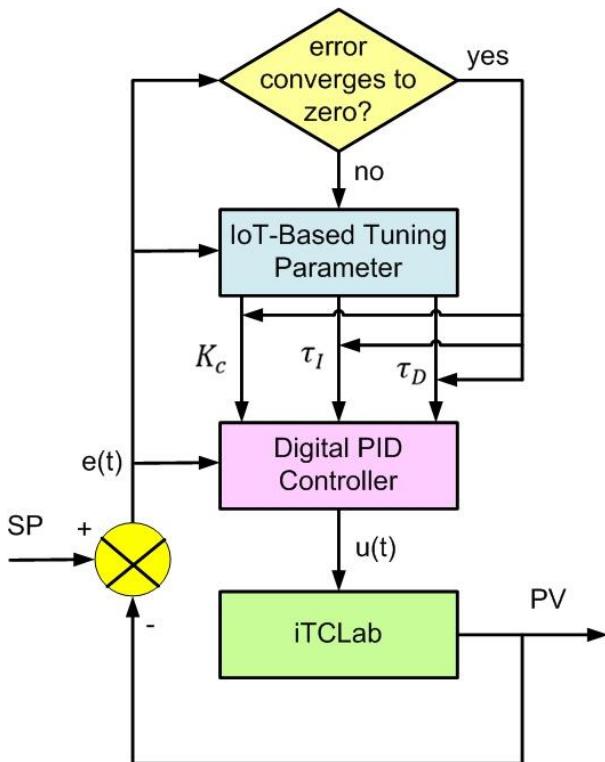
#### D. Pemrograman Pengendalian PID-iTCLab dengan IoT

Tantangan terakhir dari Buku Pemrograman IoT dengan Arduino dan Python Jilid 1 ini, dibuat program Pengendalian PID-iTCLab dengan IoT. Arsitektur sistem pengendalian suhu menggunakan PID-iTCLab via IoT seperti diperlihatkan pada Gambar 6.17.



**Gambar 6. 17. Arsitektur sistem pengendalian suhu menggunakan PID-iTCLab via IoT**

Konsep pengendalian suhu PID-iTCLab berbasis IoT secara manual melalui Ponsel, didasarkan pada keluaran suhu yang terukur, akibat penggunaan parameter PID yang digunakan. Jika parameter PID ( $K_c$ ,  $\tau_I$ , dan  $\tau_D$ ) sudah memberikan respon sistem yang bagus, ditunjukkan dengan keluaran suhu menuju SetPoint, atau dengan kata lain *error* menuju nol. Maka tidak perlu dilakukan perubahan parameter. Tetapi jika *error* tidak menuju nol, maka parameter PID bisa diubah-ubah melalui Ponsel. Konsep pengaturan ini diperlihatkan pada Gambar 6.18.



Gambar 6. 18. Konsep penalaan parameter PID-iTCLab berbasis IoT melalui Ponsel

Selanjutnya, program arduino yang dibutuhkan untuk pengendalian suhu PID-iTCLab berbasis IoT melalui Ponsel, yaitu **IoT\_Control.ino**.

Berikut ini program **IoT\_Control.ino**.

```
/*****
* Program : PID-iTCLab Controlling Using IoT
* By      : Assoc. Prof. Dr. Basuki Rahmat, S.Si, MT, ITS-AI,
*           Assoc. Prof. Dr. Muljono, S.Si, M.Kom, et al
* Pro. Team : i-ot.net, io-t.net
* R. Group : Intelligent Control, Robotics and Automation Systems Research Group
* Univ.    : Universitas Pembangunan Nasional "Veteran" Jawa Timur
* Country  : Indonesia
*****/

#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;
float KP, KI, KD, op0, ophi, oplo, error, dpv;

float sp = 30, //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

int autoSet = 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);
    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 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 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;

}

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 < oplow) || (op > ohigh)) {
    I = I - KI * error * dt;
    // clip output
    op = max(oplow, min(ohigh, 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 loop() {
new_ts = millis();

if (new_ts - ts > 1000) {

```

```
char suhu1[4];
char suhu2[4];
char Nilai_op[4];
char Tampil_SP[4];
char Tampil_Kc[4];
char Tampil_tauI[4];
char Tampil_tauD[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);
Serial.println("<----->");

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

dtostrf(cel1, 1, 0, suhu2);
client.publish("Suhu2",suhu2);

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

dtostrf(Kc, 1, 0, Tampil_Kc);
client.publish("Tampil_Kc",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
}
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;

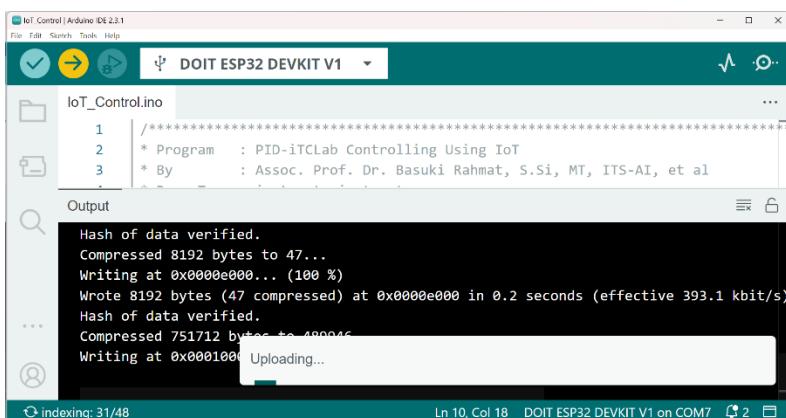
dtostrf(op, 1, 0, Nilai_op);
client.publish("Nilai_op",Nilai_op);
}
}

```

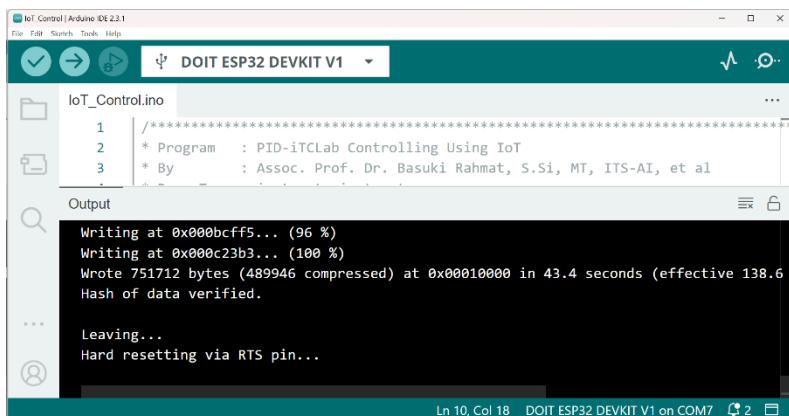
Program **IoT\_Control.ino** tersebut bisa juga diunduh melalui alamat berikut:

[https://github.com/bsrahmat/itclab-10/blob/main/IoT\\_Control.ino](https://github.com/bsrahmat/itclab-10/blob/main/IoT_Control.ino)

Silahkan program **IoT\_Control.ino** tersebut di-upload ke Kit iTCLab. Jika berhasil, maka akan terlihat seperti pada Gambar 6.19 dan Gambar 6.20.



**Gambar 6. 19. Proses upload program IoT\_Control.ino**



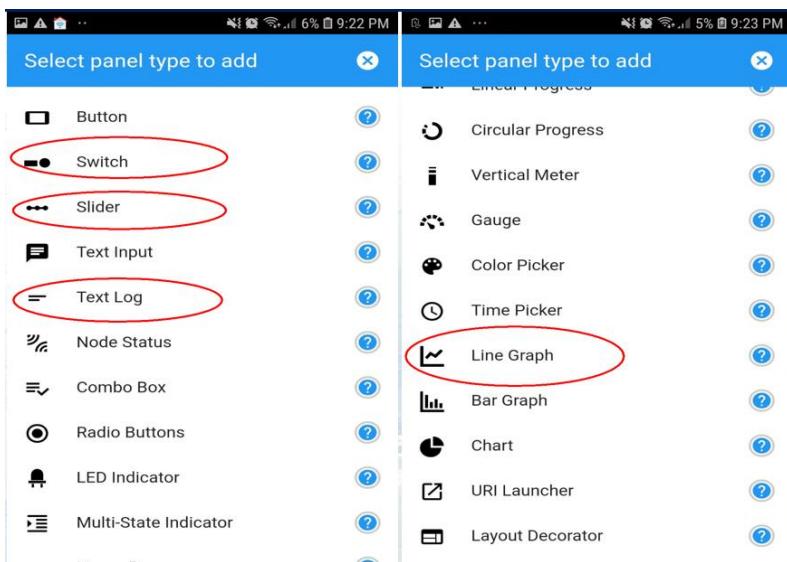
Gambar 6. 20. Proses upload program IoT\_Control.ino sukses

Setelah program **IoT\_Control.ino** berhasil ter-upload, silahkan periksa koneksi ke wifi dan password wifi yang digunakan. Jika berhasil terhubung, silahkan cek hasilnya melalui Serial Monitor. Seharusnya akan tampil seperti terlihat pada Gambar 6.21.

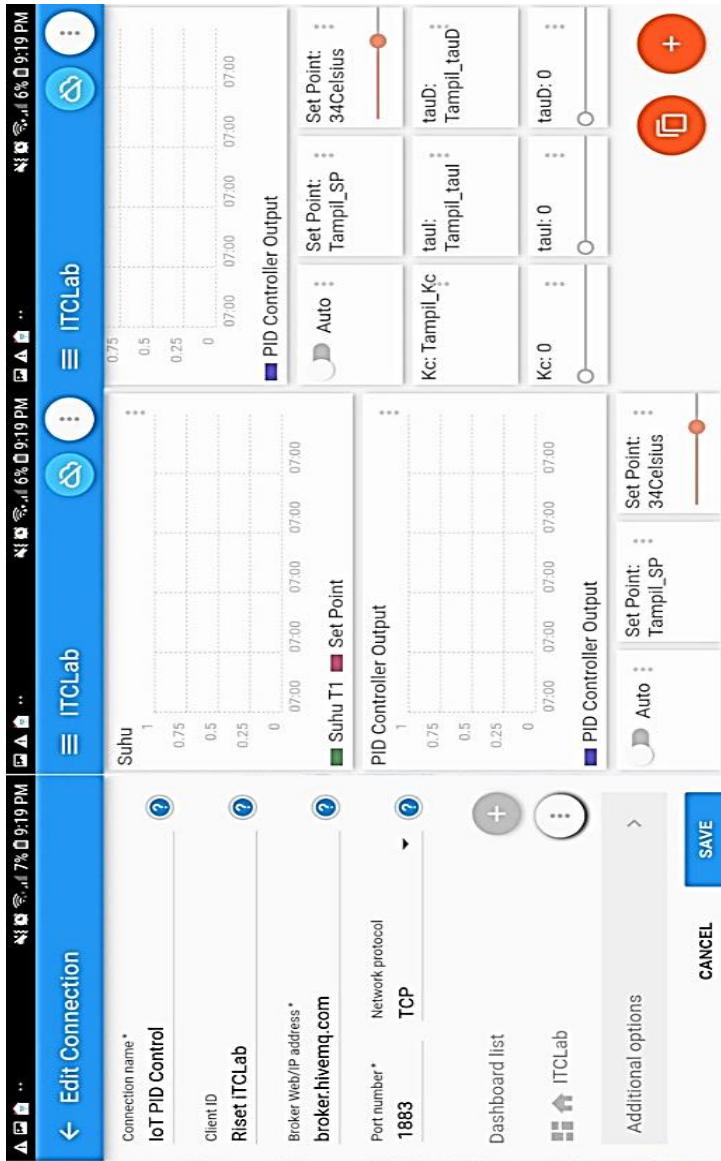


Gambar 6. 21. Contoh hasil pengendalian suhu PID-iTCLab melalui tampilan Serial Monitor

Langkah selanjutnya, silahkan dilakukan pengaturan pada aplikasi **IoT MQTT Panel** di Ponsel. Panel-panel yang dibutuhkan dan cara pengaturannya, silahkan diikuti sesuai gambar-gambar yang ada pada Gambar 6.22 sampai dengan Gambar 6.27 berikut ini.



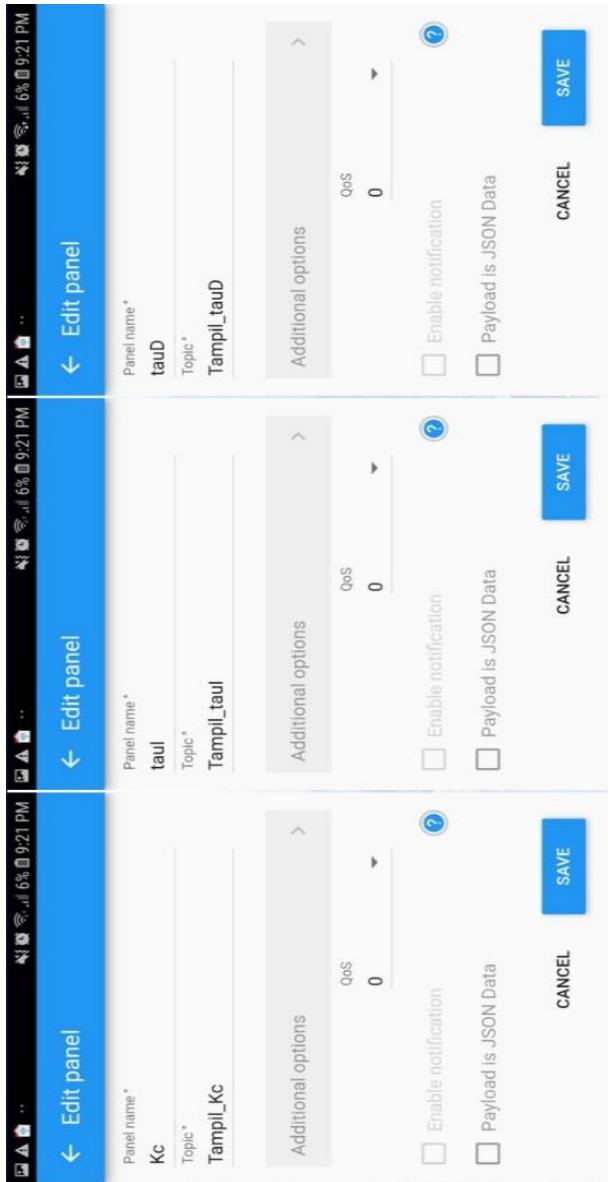
Gambar 6. 22. Pengaturan di IoT MQTT Panel (a)



Gambar 6.23. Pengaturan di IoT MQTT Panel (b)

Edit panel	
Panel name *	KC
Topic *	Nilai_KC
Subscribe Topic	Nilai_KC
Payload min *	0
Payload max *	60
Unit	
<input type="checkbox"/> Enable notification	
Slider Orientation	Horizontal
<input type="checkbox"/> Payload is JSON Data	
<input type="checkbox"/> Confirm before publish	
Edit panel	
Panel name *	tauD
Topic *	Nilai_tauD
Subscribe Topic	Nilai_tauD
Payload min *	0
Payload max *	60
Unit	
<input type="checkbox"/> Enable notification	
Slider Orientation	Horizontal
<input type="checkbox"/> Payload is JSON Data	
<input type="checkbox"/> Confirm before publish	
Edit panel	
Panel name *	tauI
Topic *	Nilai_tauI
Subscribe topic	Nilai_tauI
Payload min *	0
Payload max *	60
Unit	
<input type="checkbox"/> Enable notification	
Slider Orientation	Horizontal
<input type="checkbox"/> Payload is JSON Data	
<input type="checkbox"/> Confirm before publish	

Gambar 6.24. Pengaturan di IoT MQTT Panel (c)



Gambar 6.25. Pengaturan di IoT MQTT Panel (d)

< Edit panel

Panel name \*

Topic \*

Label for graph 2

Additional options

QoS

Show area
 Show points

Payload min \* 
 Payload max \*

Unit

Slider Orientation

QoS

CANCEL
**SAVE**

< Edit panel

Panel name \*

Topic \*

Label for graph 2

Additional options

QoS

Show area
 Show points

Payload min \* 
 Payload max \*

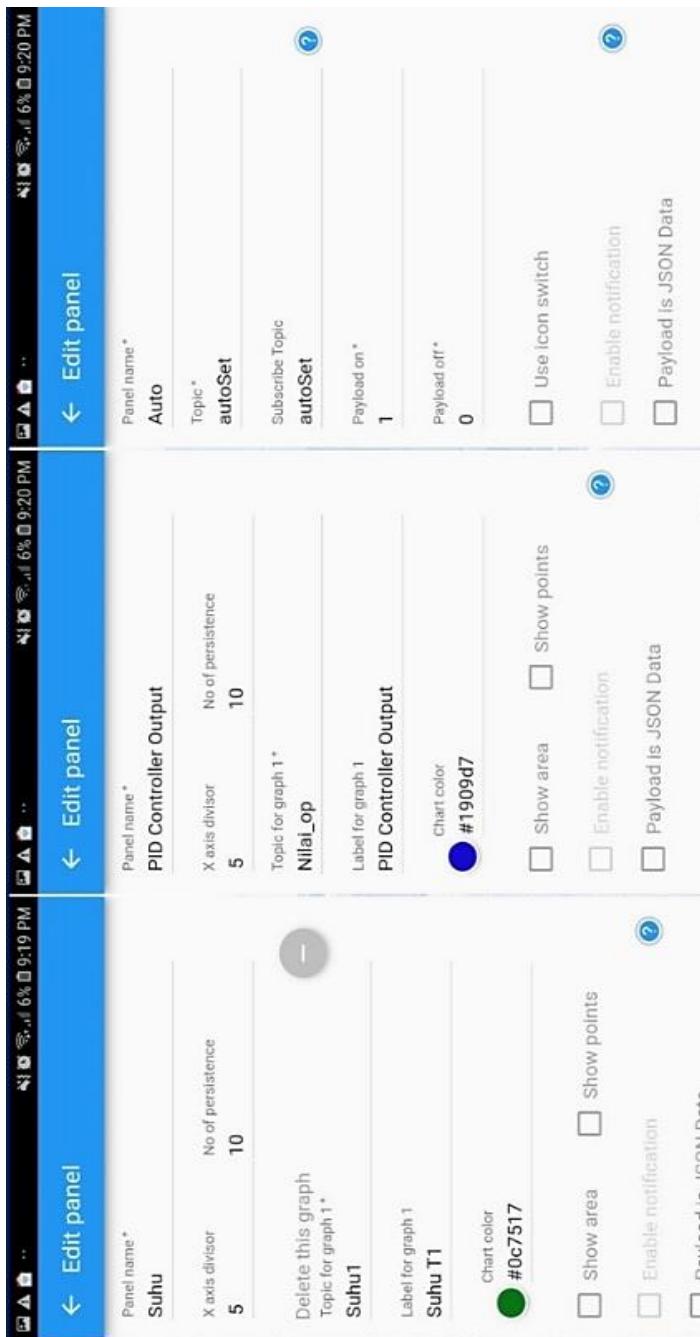
Unit

Slider Orientation

QoS

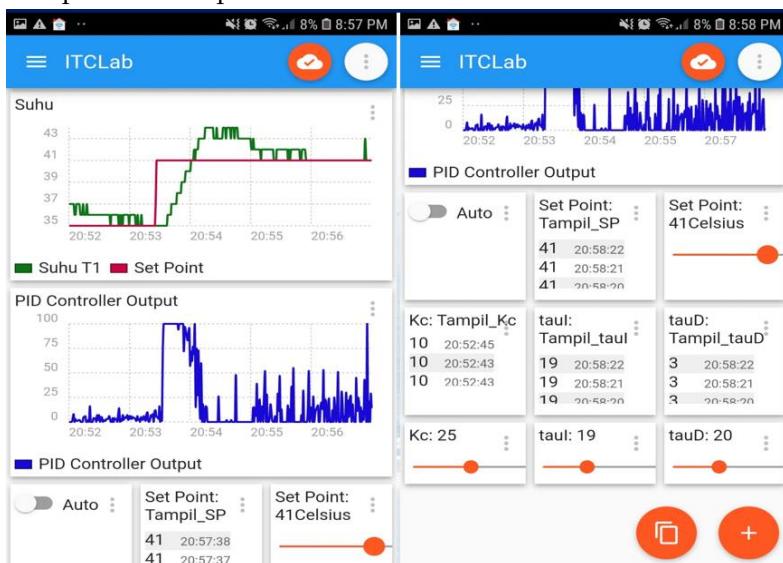
CANCEL
**SAVE**

Gambar 6.26. Pengaturan di IoT MQTT Panel (e)



Gambar 6.27. Pengaturan di IoT MQTT Panel (f)

Jika semua pengaturan telah disesuaikan, seperti pada Gambar 6.22 sampai dengan Gambar 6.27, selanjutnya tinggal dicoba untuk terhubung ke MQTT Broker hivemq.com. Jika berhasil terhubung ke MQTT Broker, maka contoh hasilnya seperti terlihat pada Gambar 6.28.



Gambar 6.28. Contoh hasil pengendalian suhu PID-iTCLab via IoT

Bagaimana? Apakah hasil pengendalian suhu PID-iTCLab di Ponsel anda, hasilnya kurang lebih sama seperti pada Gambar 6.28? Jika berhasil. Alhamdulillaah, anda memang layak dapat bintang 3. Dan berhasil menyelesaikan tantangan terakhir dari Buku Pemrograman IoT dengan Arduino dan Python Jilid 1 ini.

Apakah anda masih tertarik dengan tantangan berikutnya? Tantangan berikutnya, yaitu berupa penggabungan teknologi IoT dengan *Artificial intelligence* (AI). Teknik penggabungan IoT dan AI, insyaAllaah bisa dibaca pada Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 2. InsyaAllaah kita akan bertemu kembali pada Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 2. Barakallaah!

# BAB

# 7

# PENUTUP

Telah diuraikan dalam Buku Referensi Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 1 ini. Dimulai dari Konsep dan Teknologi *Internet of Things* (IoT) serta persiapan apa saja yang dibutuhkan agar bisa bereksperimen dengan IoT. Kemudian diperkenalkan Mikrokontroler DOIT ESP32 Devkit V1 dan Kit *Internet-Based Temperatue Control Lab* (iTCLab). Kemudian dilakukan *review* Sistem Kendali Proporsional Integral dan Derivatif (PID), dan bagaimana membuat program PID dengan Arduino dan Python. Terakhir bagaimana membuat program IoT untuk pengendalian suhu PID-iTCLab menggunakan Arduino dan Python.

Semoga Buku ini bermanfaat bagi siapa saja yang ingin melakukan penelitian di bidang IoT. Selanjutnya, untuk penggabungan teknologi IoT dengan *Artificial intelligence* (AI), insyaAllaah bisa dibaca pada Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 2. Jika ingin berkomunikasi dengan penulis, bisa melalui alamat email: [basukirahmat.if@upnjatim.ac.id](mailto:basukirahmat.if@upnjatim.ac.id) dan [muljono@dsn.dinus.ac.id](mailto:muljono@dsn.dinus.ac.id). Salam sehat dan sukses selalu.

Maret 2024

Tim Penulis

## DAFTAR PUSTAKA

- [1] R. K. Chahal, N. Kumar, and S. Batra, 'Trust management in social Internet of Things: A taxonomy, open issues, and challenges', *Comput Commun*, vol. 150, pp. 13–46, 2020, doi: <https://doi.org/10.1016/j.comcom.2019.10.034>.
- [2] S. Ravidas, A. Lekidis, F. Paci, and N. Zannone, 'Access control in Internet-of-Things: A survey', *Journal of Network and Computer Applications*, vol. 144, pp. 79–101, 2019, doi: <https://doi.org/10.1016/j.jnca.2019.06.017>.
- [3] S. Zeadally and O. Bello, 'Harnessing the power of Internet of Things based connectivity to improve healthcare', *Internet of Things*, p. 100074, 2019, doi: <https://doi.org/10.1016/j.iot.2019.100074>.
- [4] M. Gheisari, G. Wang, and S. Chen, 'An Edge Computing-enhanced Internet of Things Framework for Privacy-preserving in Smart City', *Computers & Electrical Engineering*, vol. 81, p. 106504, 2020, doi: <https://doi.org/10.1016/j.compeleceng.2019.106504>.
- [5] anonymous, 'mCLOUD IoT Platform Services', *mthinx.com*, 2019.
- [6] M. Kashyap, V. Sharma, and N. Gupta, 'Taking MQTT and NodeMcu to IOT: Communication in Internet of Things', *Procedia Comput Sci*, vol. 132, pp. 1611–1618, 2018, doi: <https://doi.org/10.1016/j.procs.2018.05.126>.
- [7] Espressif, 'ESP32Series Datasheet'. 2024. [Online]. Available: [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)
- [8] V. Mohanan, 'DOIT ESP32 DevKit V1 Wi-Fi Development Board – Pinout Diagram & Arduino Reference'. 2022. [Online]. Available: <https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/>

- [9] B. Rahmat *et al.*, ‘iTCLab PID Control Tuning Using Deep Learning’, in *2023 IEEE 9th Information Technology International Seminar (ITIS)*, 2023, pp. 1–4. doi: 10.1109/ITIS59651.2023.10420130.
- [10] B. Rahmat, M. Waluyo, and T. A. Rachmanto, ‘Temperature Monitoring via the Internet of Things Using PID-iTCLab’, *Nusantara Science and Technology Proceedings*, vol. 2023, no. 33, pp. 197–203, 2023, [Online]. Available: <https://nstproceeding.com/index.php/nuscientech/article/view/939>
- [11] B. Rahmat, M. Waluyo, and T. A. Rachmanto, ‘On/Off Temperature Monitoring and Control via the Internet of Things Using iTCLab Kit’, *Nusantara Science and Technology Proceedings*, vol. 2023, no. 33, pp. 147–152, May 2023, doi: 10.11594/nstp.2023.3325.
- [12] B. Rahmat *et al.*, ‘iTCLab Temperature Monitoring and Control System Based on PID and Internet of Things (IoT)’, *IGI Global*, pp. 199–210, 2023, doi: <https://doi.org/10.4018/978-1-6684-5629-3.ch012>.
- [13] B. Y. U. BYU, ‘Apmonitor.com’, *Temperature Control Lab*. 2018. [Online]. Available: <http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl>
- [14] M. Maung, M. Latt, and C. Nwe, ‘DC Motor Angular Position Control using PID Controller with Friction Compensation’, *International Journal of Scientific and Research Publications (IJSRP)*, vol. 8, Mar. 2018, doi: 10.29322/IJSRP.8.11.2018.p8321.
- [15] B. Y. U. BYU, ‘Apmonitor.com’, *Proportional Integral Derivative (PID) Control*. 2018. [Online]. Available: <http://apmonitor.com/pdc/index.php/Main/ProportionalIntegralDerivative>

- [16] R. P. Batni, 'Modern control engineering: K. Ogata. 836 pages, diagrams, 6 × 9 in. Englewood Cliffs, N.J., Prentice-Hall, 1970. Price, 18.50.', *J Franklin Inst*, vol. 293, no. 1, pp. 70–71, 1972, doi: [https://doi.org/10.1016/0016-0032\(72\)90146-9](https://doi.org/10.1016/0016-0032(72)90146-9).

## TENTANG PENULIS



**Basuki Rahmat**, adalah Dosen Program Studi S2 Teknologi Informasi, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional “Veteran” Jawa Timur. Beliau menerima gelar Sarjana Fisika Bidang Instrumentasi dari Institut Teknologi Sepuluh Nopember Surabaya pada tahun 1995, menerima gelar Magister Teknik Program Instrumentasi dan Kontrol Institut Teknologi Bandung pada tahun 2000, dan menerima gelar Doktor Teknik Elektro Bidang Jaringan Cerdas Multimedia dari Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2018. Minat penelitiannya adalah di bidang komputasi cerdas, kendali cerdas, komputer visi, drone, robotika, pemrograman PHP, arduino dan python.



**Muljono**, adalah Dosen Program Studi Doktor Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro Semarang. Beliau menerima gelar Sarjana Matematika dari Universitas Diponegoro Semarang pada tahun 1996, menerima gelar Magister Komputer dari STTIBI Jakarta pada tahun 2001, dan menerima gelar Doktor Teknik Elektro Bidang Jaringan Cerdas Multimedia dari Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2016. Minat penelitiannya adalah di bidang Kecerdasan Buatan, Data Mining, Pemrosesan Bahasa Alami dan Rekayasa Perangkat Lunak.