



# Pemrograman Internet of Things (IoT) dengan **Arduino** dan **Python**

Jilid  
2

Basuki Rahmat  
Muljono



# 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 Istrumentasi 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.

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

## **JILID 2**

**Basuki Rahmat  
Muljono**



**PENERBIT CV.EUREKA MEDIA AKSARA**

# **PEMROGRAMAN INTERNET OF THINGS (IOT) DENGAN ARDUINO DAN PYTHON JILID 2**

**Penulis** : Basuki Rahmat  
Muljono

**Desain Sampul** : Eri Setiawan

**Tata Letak** : Ayu May Lisa

**ISBN** : 978-623-120-596-4 (no.jil.lengkap)  
978-623-120-952-8 (jil.2)

Diterbitkan oleh : **EUREKA MEDIA AKSARA, JUNI 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 2. 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 atau muljono@dsn.dinus.ac.id. Terimakasih.

Semarang, Juli 2024  
Tim Penulis.

## DAFTAR ISI

<b>PRAKATA .....</b>	<b>iii</b>
<b>DAFTAR ISI.....</b>	<b>iv</b>
<b>DAFTAR GAMBAR.....</b>	<b>v</b>
<b>DAFTAR TABEL .....</b>	<b>viii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
A. Pengantar Buku Jilid 2 .....	1
B. Perkembangan Artificial Intelligence .....	2
C. Perkembangan Machine Learning dan Deep Learning .....	3
D. Pengembangan Framework Deep Learning .....	16
E. Penalaan Parameter PID dengan Sistem Cerdas .....	28
<b>BAB 2 PEMROGRAMAN DEEP LEARNING.....</b>	<b>29</b>
A. Pemrograman Deep Learning dengan Python .....	29
B. Pemrograman Gerbang XOR.....	30
C. Penalaan Parameter PID dengan Deep Learning .....	37
<b>BAB 3 PEMROGRAMAN IoT LANJUTAN.....</b>	<b>51</b>
A. Pengendalian PID-iTCLab dengan Deep Learning .....	51
B. Pemantauan PID-iTCLab-Deep Learning Via IoT .....	81
<b>BAB 4 PENUTUP .....</b>	<b>121</b>
<b>DAFTAR PUSTAKA.....</b>	<b>122</b>
<b>TENTANG PENULIS.....</b>	<b>125</b>

## DAFTAR GAMBAR

Gambar 1. 1.	Perkembangan Artificial Intelligence, Machine Learning dan Deep Learning [10].....	5
Gambar 1. 2.	Deep Learning bagian dari Machine Learning, keduanya bagian dari Artificial Intelligence [10] .....	6
Gambar 1. 3.	Arsitektur jaringan DBN [15] .....	10
Gambar 1. 4.	Contoh CNN dalam visi komputer [15] .....	12
Gambar 1. 5.	Arsitektur CNN [15].....	14
Gambar 1. 6.	Logo TensorFlow .....	17
Gambar 1. 7.	Logo Keras .....	18
Gambar 1. 8.	Logo PyTorch.....	20
Gambar 1. 9.	Logo Caffe .....	20
Gambar 1. 10.	Logo Sonnet .....	21
Gambar 1. 11.	Logo MXNet.....	22
Gambar 1. 12.	Logo Gluon .....	23
Gambar 1. 13.	Logo Swift .....	24
Gambar 1. 14.	Logo Chainer.....	24
Gambar 1. 15.	Logo DL4J.....	25
Gambar 1. 16.	Logo ONNX .....	26
Gambar 1. 17.	Logo Darknet. ....	26
Gambar 1. 18.	Logo YOLO. ....	27
Gambar 2. 1.	Arsitektur Deep Learning untuk menirukan Gerbang Logika XOR .....	31
Gambar 2. 2.	Model, bobot dan bias awal.....	33
Gambar 2. 3.	Proses pelatihan Deep Learning menggunakan Keras.....	34
Gambar 2. 4.	Keluaran hasil prediksi XOR menggunakan Deep Learning dengan Keras .....	35
Gambar 2. 5.	Perbandingan prediksi Keras dan target.....	36
Gambar 2. 6.	Kesalahan hasil prediksi Keras .....	36
Gambar 2. 7.	Proses penalaan nilai $K_c$ , $\tau_I$ dan $\tau_D$ pada pengendali PID menggunakan Deep Learning .....	37
Gambar 2. 8.	Arsitektur Deep Learning untuk penyesuaian nilai $K_c$ , $\tau_I$ dan $\tau_D$ .....	38
Gambar 2. 9.	Model Deep Learning .....	40

Gambar 2. 10. Bobot-bobot Deep Learning .....	40
Gambar 2. 11. Proses pelatihan Deep Learning.....	41
Gambar 2. 12. Contoh hasil prediksi, keluaran Deep Learning ....	41
Gambar 2. 13. Contoh hasil parameter PID keluaran Deep Learning.....	42
Gambar 2. 14. Contoh hasil proses penalaan nilai Kc, tI dan tD berbasis Deep Learning .....	43
Gambar 2. 15. Hasil proses penalaan nilai Kc, tI dan tD pada pengendali PID berbasis Deep Learning terhadap keluaran sistem .....	47
Gambar 2. 16. Hasil proses penalaan nilai Kc, tI dan tD yang lain pada pengendali PID berbasis Deep Learning terhadap keluaran sistem.....	49
Gambar 3. 1. Proses penalaan nilai Kc, tI dan tD pada pengendali PID menggunakan Deep Learning pada Kit iTCLab.....	52
Gambar 3. 2. Arsitektur Deep Learning untuk penyesuaian nilai Kc, tI dan tD pada Sistem Kendali PID-iTCLab.....	52
Gambar 3. 3. Proses upload Program Deep_PID_iTCLab.ino.....	60
Gambar 3. 4. Program Deep_PID_iTCLab.ino berhasil di-upload .....	61
Gambar 3. 5. Contoh hasil konfigurasi model dan bobotnya .....	68
Gambar 3. 6. Contoh proses pelatihan Deep Learning .....	69
Gambar 3. 7. Contoh hasil proses penalaan nilai Kc, tI dan tD pada pengendali PID menggunakan Deep Learning pada Kit iTCLab.....	80
Gambar 3. 8. Contoh hasil pengendalian PID menggunakan Deep Learning pada Kit iTCLab.....	80
Gambar 3. 9. Pemantauan hasil proses penalaan nilai Kc, tI dan tD pada pengendali PID menggunakan Deep Learning pada Kit iTCLab melalui IoT .....	83
Gambar 3. 10. Proses upload Program Deep_PID_iTCLab_IoT.ino .....	91

Gambar 3. 11. Program Deep_PID_iTCLab_IoT.ino berhasil di-upload .....	92
Gambar 3. 12. Contoh hasil konfigurasi model dan bobotnya .....	99
Gambar 3. 13. Contoh proses pelatihan Deep Learning.....	101
Gambar 3. 14. Terhubung ke Broker MQTT .....	107
Gambar 3. 15. Contoh hasil proses penalaan nilai $K_c$ , $\tau_I$ dan $\tau_D$ pada pengendali PID menggunakan Deep Learning pada Kit iTCLab .....	113
Gambar 3. 16. Contoh hasil pengendalian PID menggunakan Deep Learning pada Kit iTCLab... ..	114
Gambar 3. 17. Pengaturan di IoT MQTT Panel .....	115
Gambar 3. 18. Pengaturan di IoT MQTT Panel (Lanjutan) .....	116
Gambar 3. 19. Pengaturan di IoT MQTT Panel (Lanjutan) .....	117
Gambar 3. 20. Pengaturan di IoT MQTT Panel (Lanjutan) .....	118
Gambar 3. 21. Contoh pemantauan hasil pengendalian PID menggunakan Deep Learning pada Kit iTCLab melalui IoT.....	119

## **DAFTAR TABEL**

Tabel 2. 1. Data Latih Gerbang Logika XOR. ....	30
---	----

# BAB

# 1 | PENDAHULUAN

## A. Pengantar Buku Jilid 2

Pada Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 1 sebelumnya, telah dibahas mulai Konsep dan Teknologi Internet of Things (IoT) serta persiapan apa saja yang dibutuhkan agar bisa bereksperimen dengan IoT. Kemudian dijelaskan Mikrokontroler DOIT ESP32 Devkit V1 dan diperkenalkan Kit *Internet-Based Temperatue Control Lab* (iTCLab). Selanjutnya dilakukan *review* Sistem Kendali Proporsional Integral dan Derivatif (PID), dan bagaimana membuat program PID dengan Arduino dan Python. Terakhir bagaimana membuat program IoT untuk pemantauan dan pengendalian suhu PID-iTCLab menggunakan Arduino dan Python. Termasuk bagaimana konsep penalaan parameter PID menggunakan metode Ziegler-Nichols, serta bagaimana teknik penalaan parameter PID secara *trial* berbasis IoT.

Pada Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 2 ini, akan digabungkan teknologi IoT dengan *Artificial Intelligence* (AI). Terutama intinya dipergunakan untuk proses penalaan parameter PID menggunakan *Deep Learning* bagian dari *Machine Learning* bagian dari AI, dan proses pemantauan hasil pengendaliannya melalui IoT.

Pada Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 2 ini, akan dijelaskan Perkembangan *Artificial Intelligence*, Perkembangan *Machine Learning* dan *Deep*

*Learning*, Pengembangan *Framework Deep Learning*, Penalaan Parameter PID dengan Sistem Cerdas, Pemrograman *Deep Learning* dengan Python, Pemrograman Gerbang XOR, Penalaan Parameter PID dengan *Deep Learning*, Pengendalian PID-iTCLab dengan *Deep Learning*, dan terakhir bagaimana Pemantauan PID-iTCLab-*Deep Learning* via IoT.

## B. Perkembangan Artificial Intelligence

*Artificial Intelligence* (AI) atau Kecerdasan Buatan didefinisikan sebagai situasi di mana mesin dapat mensimulasikan pikiran manusia dalam pembelajaran dan analisis, yang dengan demikian dapat bekerja dalam pemecahan masalah [1]. Dengan kata lain situasi dimana mesin seolah bisa menirukan bagaimana manusia berpikir, belajar, menganalisis dan mengambil keputusan untuk memecahkan masalah. AI juga didefinisikan sebagai studi tentang "agen cerdas" yaitu, agen atau perangkat apa pun yang dapat merasakan dan memahami lingkungannya sehingga dapat mengambil tindakan yang tepat agar bisa memaksimalkan peluangnya dalam mencapai tujuannya [1].

AI mulai dikutip pada tahun 1940-an dalam studi tentang apakah mesin dapat membuat sebuah keputusan [2]. Pada tahun 1970-an, pengembangan solusi AI terapan dimulai, dengan studi di berbagai bidang pengetahuan. Pada tahun 2010-an, studi dan aplikasi AI telah mencakup beberapa fungsi sektor publik, kesehatan masyarakat, transportasi, pendidikan, keamanan, komunikasi, dan bahkan angkatan bersenjata [2].

Dalam perkembangannya, AI juga sudah banyak diintegrasikan dengan teknologi *Internet of Things* (IoT). Tinjauan tantangan dan solusi dalam implementasi AI, IoT, dan Blockchain dalam Industri Konstruksi telah dilakukan [3] Penggabungan AI dan IoT juga dipergunakan untuk pengamanan jaringan IoT. Terutama serangan Denial-of-Service Terdistribusi (DDoS) pada jaringan IoT. DDoS adalah salah satu serangan cyber yang paling merusak dan menantang. Karena jumlah pengguna IoT tumbuh secara eksponensial akibat

peningkatan perangkat IoT selama beberapa tahun terakhir. Akibatnya, serangan DDoS menjadi serangan paling menonjol karena perangkat IoT yang rentan menjadi korbannya [4].

Pemanfaatan AI dan IoT di bidang pertanian juga telah banyak dilakukan, diantaranya: Sistem irigasi otomatis untuk pertanian pintar menggunakan AI dan jaringan IoT yang Didukung 6G [5], aplikasi berbasis IoT dan AI dalam berbagai aktivitas pra-panen, panen, dan pasca panen pada pertanian pintar [6], dan lain-lain.

Pemanfaatan AI dan IoT di bidang kesehatan juga telah banyak dilakukan, diantaranya: sistem pemantauan kesehatan berbasis IoT dengan AI di *Edge Computing* untuk Kota Cerdas [7], pertimbangan permasalahan keamanan dan privasi berbasis AI digerakkan IoT untuk perawatan kesehatan pintar [8], pemangkasan dinamis berbasis *Reinforcement Learning* untuk inferensi terdistribusi melalui AI yang dapat dijelaskan dalam sistem IoT perawatan kesehatan juga telah dilakukan [9], dan lain-lain. Dalam Buku ini AI dan IoT akan digabungkan untuk penalaan parameter PID dan memantau hasil pengendaliannya melalui jarak jauh selama tersedia koneksi internet.

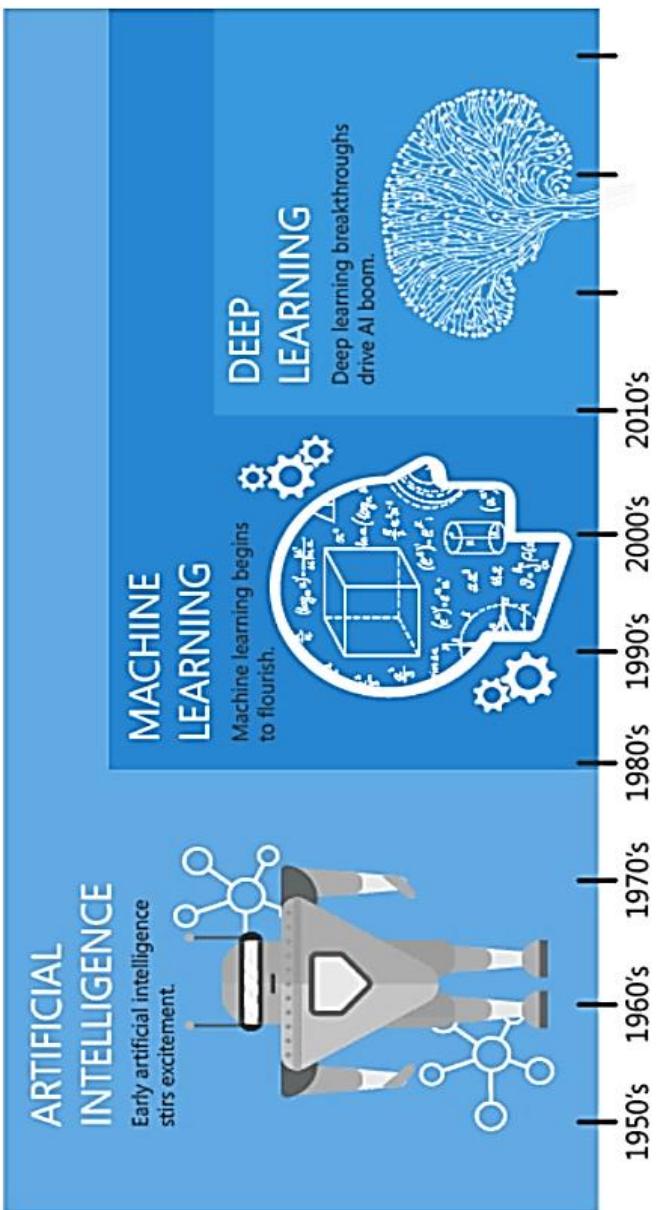
## C. Perkembangan Machine Learning dan Deep Learning

### 1. Machine Learning

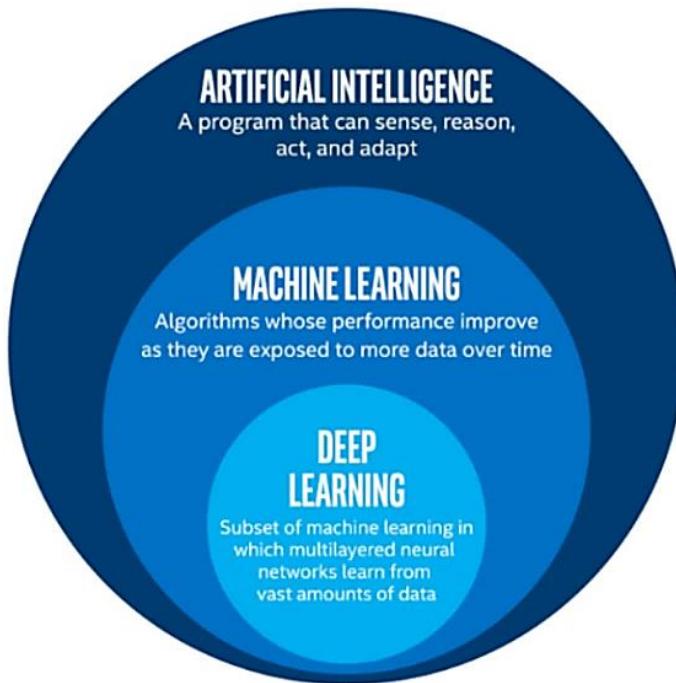
*Machine Learning* merupakan bagian dari Kecerdasan Buatan atau AI. Dengan kata lain, perkembangan lebih lanjut dari AI melahirkan *Machine Learning*. Perkembangan lebih lanjut dari *Machine Learning* melahirkan *Deep Learning*. Secara umum perkembangan AI, *Machine Learning* dan *Deep Learning* dari tahun ke tahun diperlihatkan pada Gambar 1.1 [10]. Hubungan antara AI, *Machine Learning* dan *Deep Learning* diperlihatkan pada Gambar 1.2 [10].

*Machine Learning* sendiri merupakan teknik yang memungkinkan komputer "belajar" dari data-data yang disediakan tanpa pemrograman yang menyeluruh dan eksplisit dari setiap masalah [11]. Ini bertujuan memodelkan hubungan yang mendalam terhadap input data dan

merekonstruksi skema pengetahuan. Hasil dari proses pembelajaran dapat digunakan untuk estimasi, prediksi, dan klasifikasi [11].



Gambar 1.1. Perkembangan Artificial Intelligence, Machine Learning dan Deep Learning [10]



Gambar 1. 2. Deep Learning bagian dari Machine Learning, keduanya bagian dari Artificial Intelligence [10]

## 2. Deep Learning

*Deep Learning* telah menjadi tantangan untuk didefinisikan oleh banyak orang dan para peneliti karena telah berubah bentuk secara perlahan selama dekade terakhir. Satu definisi yang berguna menetapkan bahwa *Deep Learning* berkaitan dengan "Jaringan Sel Saraf (*Neural Network*) dengan lebih dari dua lapisan." Namun jika dari definisi ini, maka seolah-olah *Deep Learning* sudah ada sejak tahun 1980-an (lihat Gambar 1.1). Padahal *Deep Learning* dianggap baru muncul pada tahun 2006, setelah Geoffrey Hinton memperkenalkan salah satu varian Jaringan Sel Saraf Tiruan atau biasa disebut Jaringan Saraf Tiruan (*Artificial Neural Network*) yang disebut *Deep Belief Nets* [12]. Ide untuk melatih model Jaringan Saraf ini adalah dengan melatih dua lapisan kemudian ditambahkan satu lapisan diatasnya,

kemudian dilatih hanya lapisan teratas dan begitu seterusnya. Dengan strategi ini didapat pelatihan model Jaringan Saraf yang memiliki lapisan lebih banyak dari model-model sebelumnya. Tulisan Geoffrey Hinton ini merupakan awal populernya istilah *Deep Learning* untuk membedakan arsitektur Jaringan Saraf Tiruan dengan banyak lapisan.

Setelah istilah *Deep Learning* populer, *Deep Learning* belum menjadi daya tarik yang besar bagi para peneliti karena Jaringan Saraf Tiruan dengan banyak lapisan memiliki kompleksitas algoritma yang besar, sehingga membutuhkan komputer dengan spesifikasi tinggi, dan tidak efisien secara komputasi saat itu. Hingga pada tahun 2009 Andrew Y. Ng dkk memperkenalkan penggunaan *Graphics Processing Unit* (GPU) untuk *Deep Learning* [13]. Dengan penggunaan GPU Jaringan Saraf Tiruan dapat berjalan lebih cepat dibanding dengan menggunakan *Central Processing Unit* (CPU). Dengan tersedianya *hardware* yang memadai perkembangan *Deep Learning* mulai pesat, dan menghasilkan produk-produk yang dapat kita nikmati saat ini seperti pengenalan wajah, *self-driving car*, pengenalan suara, dan lain lain.

Perkembangan Jaringan Saraf sudah jauh melampaui arsitektur gaya jaringan sebelumnya (diiringi dengan lebih banyak kekuatan pemrosesan) sebelum menunjukkan hasil spektakuler yang terlihat dalam beberapa tahun terakhir. Berikut ini adalah beberapa aspek dalam evolusi Jaringan Saraf ini, yang menjadi ciri khas dari *Deep Learning*:

- a. Lebih banyak neuron daripada jaringan sebelumnya.
- b. Cara yang lebih kompleks untuk menghubungkan lapisan/neuron di Jaringan Saraf.
- c. Ledakan dalam jumlah daya komputasi yang tersedia untuk pelatihan.
- d. Ekstraksi fitur otomatis.

*Deep Learning* mencoba memodelkan abstraksi data skala besar dengan menggunakan arsitektur *Deep Neural Networks* (DNNs) multi lapisan, sehingga mereka dapat memahami gambar, suara, dan teks [14]. *Deep Learning* umumnya memiliki dua sifat [14]:

- a. terdiri dari beberapa lapisan unit pemrosesan nonlinear, dan
- b. presentasi fitur pada setiap lapisan menggunakan proses pembelajaran yang diawasi atau tidak diawasi.

Selanjutnya, pembahasan tentang *Deep Learning*, tidak dapat dipisahkan dari *Deep Network* itu sendiri yang membahas tentang arsitektur jaringannya. Arsitektur utama *Deep Network* yang terkenal dan banyak digunakan di berbagai bidang penerapan, yaitu [15]:

- a. *Unsupervised Pretrained Networks* (UPNs)
- b. *Convolutional Neural Networks* (CNNs)
- c. *Recurrent Neural Networks*, dan
- d. *Recursive Neural Networks*.

Masing-masing secara ringkas diuraikan sebagai berikut.

**a. *Unsupervised Pretrained Networks* (UPNs)**

Dalam kelompok ini, setidaknya terdapat tiga arsitektur spesifik, yaitu [15]:

- 1) *Autoencoders*
- 2) *Deep Belief Networks* (DBNs), dan
- 3) *Generative Adversarial Networks* (GANs).

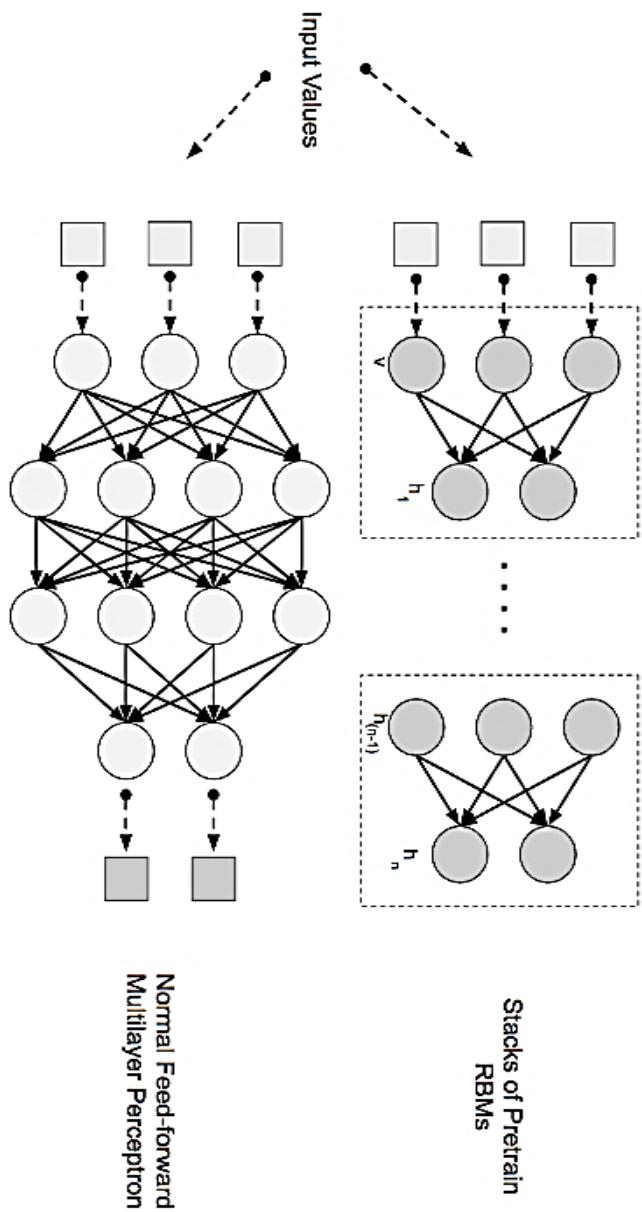
***Autoencoders***

*Autoencoder* adalah varian dari Jaringan Saraf umpan-maju yang memiliki bias ekstra untuk menghitung kesalahan merekonstruksi input asli. Setelah pelatihan, *autoencoder* kemudian digunakan sebagai Jaringan Saraf umpan-maju normal untuk aktivasi. Ini adalah bentuk ekstraksi fitur yang tidak diawasi karena Jaringan Saraf hanya menggunakan input asli untuk pembelajaran bobot daripada *backpropagation*, yang memiliki label. *Deep Network* dapat menggunakan

*Restricted Boltzmann Machines* (RBMs) atau *autoencoder* sebagai blok penyusun untuk jaringan yang lebih besar (namun jarang ada satu jaringan yang menggunakan keduanya).

#### ***Deep Belief Networks (DBNs)***

DBN terdiri dari lapisan *Restricted Boltzmann Machines* (RBMs) untuk fase pra-terlatihan dan kemudian jaringan umpan-maju untuk fase *fine-tune*. Gambar 1.3 menunjukkan arsitektur jaringan DBN [15].



Gambar 1. 3. Arsitektur jaringan DBN [15]

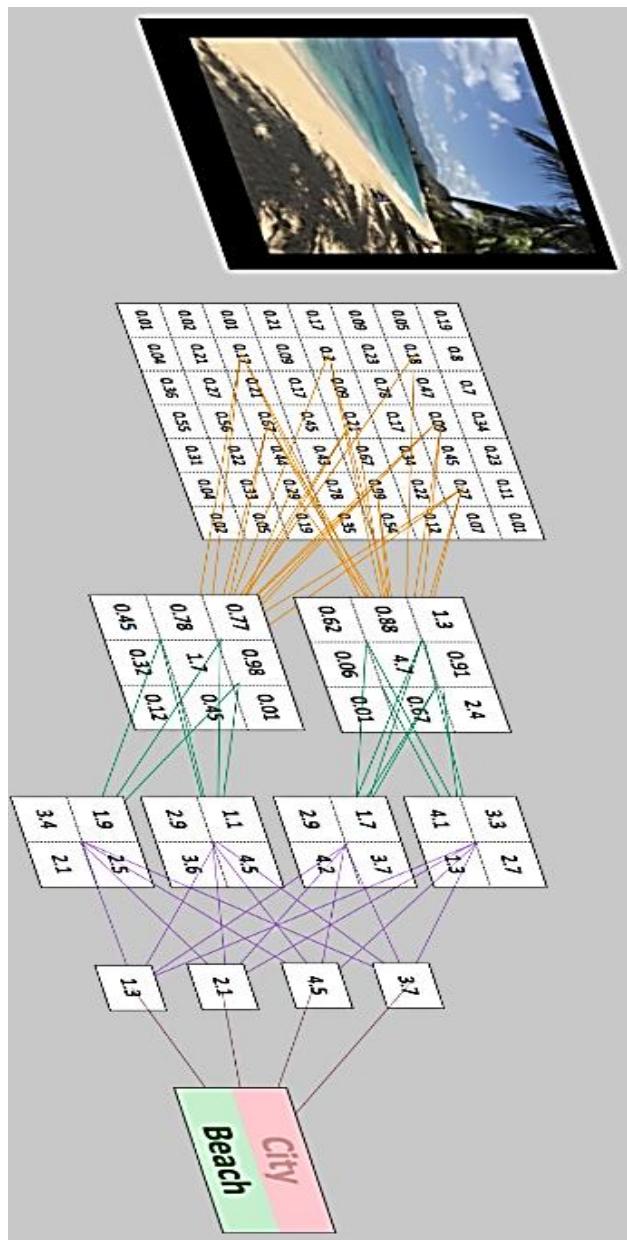
### ***Generative Adversarial Networks (GANs)***

GAN adalah contoh jaringan yang menggunakan pembelajaran tanpa pengawasan untuk melatih dua model secara paralel. Aspek kunci dari GAN (dan model generatif pada umumnya) adalah bagaimana mereka menggunakan jumlah parameter yang secara signifikan lebih kecil dari normal sehubungan dengan jumlah data yang dilatih dalam jaringan. Jaringan dipaksa untuk secara efisien mewakili data pelatihan, membuatnya lebih efektif dalam menghasilkan data yang mirip dengan data pelatihan.

### **b. *Convolutional Neural Networks (CNNs)***

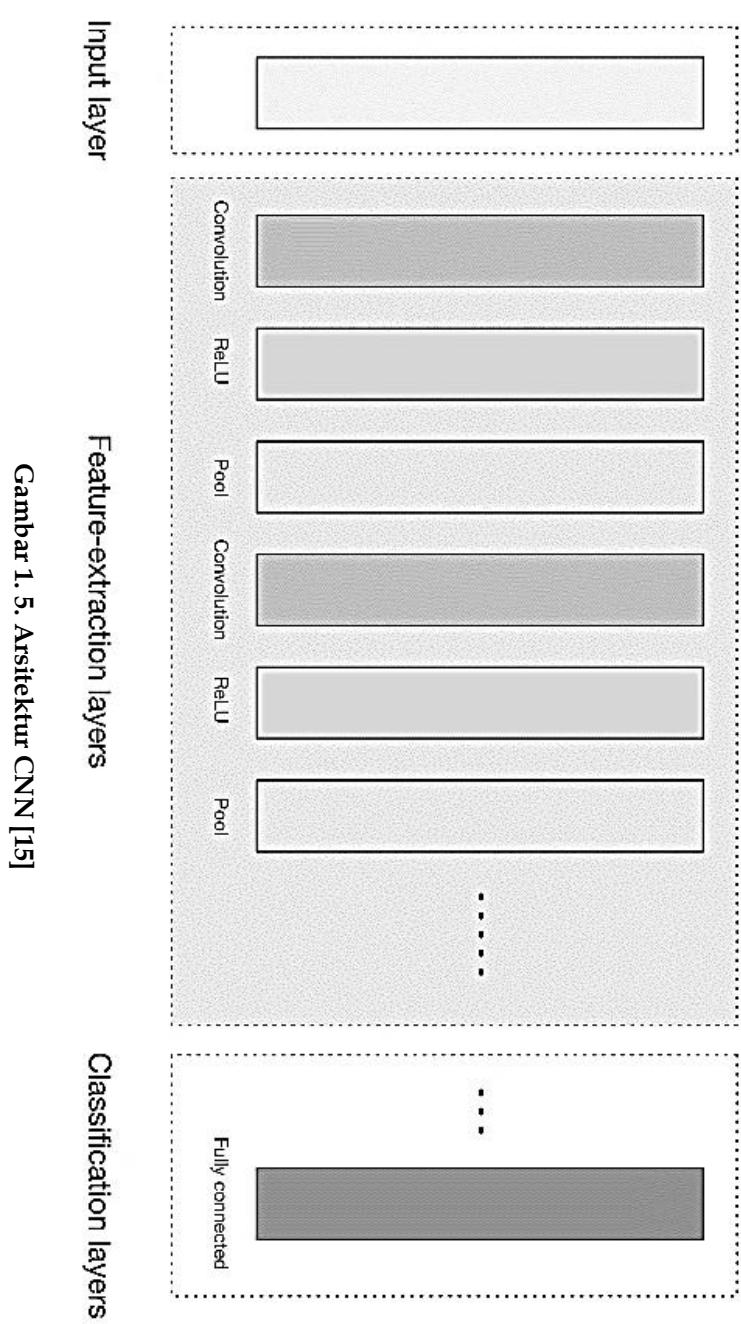
Tujuan dari CNN adalah mempelajari fitur tingkat tinggi dalam data melalui konvolusi. CNN sangat cocok untuk pengenalan objek melalui gambar dan memegang kompetisi klasifikasi gambar teratas secara konsisten. CNN dapat mengidentifikasi wajah, individu, rambu jalan, platipus, dan banyak aspek lain dari data visual. Untuk analisis teks, CNN tumpang tindih dengan *Optical Character Recognition* (OCR), karena CNN juga mampu menganalisis kata-kata sebagai unit textual diskrit. CNN juga pandai menganalisis suara.

Kemampuan CNN dalam pengenalan gambar adalah salah satu alasan utama mengapa dunia mengakui kehebatan *Deep Learning*. Seperti yang diilustrasikan pada Gambar 1.4, CNN sangat bagus dalam mendapatkan fitur-fitur dalam posisi tegak dan agak bagus dalam posisi invarian rotasi dari data gambar mentah [15].



Gambar 1.4. Contoh CNN dalam visi komputer [15]

CNN mengubah data input dari lapisan input melalui semua lapisan yang terhubung ke satu set skor kelas yang diberikan oleh lapisan output. Ada banyak variasi arsitektur CNN, tetapi umumnya mereka didasarkan pada pola lapisan, seperti yang ditunjukkan pada Gambar 1.5 [15].



Gambar 1. 5. Arsitektur CNN [15]

Gambar 1.5 menggambarkan tiga kelompok utama:

- 1) Lapisan input
- 2) Lapisan ekstraksi fitur, dan
- 3) Lapisan klasifikasi

**Lapisan input** menerima input tiga dimensi umumnya dalam bentuk spasial dari ukuran gambar (lebar  $\times$  tinggi) dan memiliki kedalaman yang mewakili saluran warna (umumnya tiga, untuk saluran warna Red Green Blue (RGB)).

**Lapisan ekstraksi fitur** memiliki pola pengulangan urutan yang umum:

- 1) Lapisan konvolusi
- 2) Lapisan (fungsi aktivasi) *Rectified Linear Unit* (ReLU), dan
- 3) Lapisan pooling.

Lapisan-lapisan ini menemukan sejumlah fitur dalam gambar dan secara progresif membangun fitur tingkat tinggi.

**Lapisan klasifikasi** memiliki satu atau lebih lapisan yang terhubung sepenuhnya untuk mengambil fitur tingkat tinggi dan menghasilkan probabilitas atau skor kelas. Lapisan-lapisan ini sepenuhnya terhubung ke semua neuron di lapisan sebelumnya. Keluaran dari lapisan-lapisan ini biasanya menghasilkan keluaran dua dimensi dari dimensi  $[b \times N]$ , di mana b adalah jumlah contoh-contoh dalam *mini-batch* dan N adalah jumlah kelas dalam penyusunan skor.

#### c. *Recurrent Neural Networks*

*Recurrent Neural Networks* berada dalam keluarga Jaringan Saraf umpan-maju. Jaringan yang memungkinkan komputasi paralel dan berurutan. Jaringan ini serupa dengan otak manusia, yang memiliki jaringan umpan balik besar dari neuron yang terhubung. Otak adalah panutan yang luar biasa karena dapat memecahkan banyak masalah yang belum dapat diselesaikan oleh mesin saat ini. Jaringan ini mengambil

setiap vektor dari urutan vektor input dan memodelkannya satu per satu. Ini memungkinkan jaringan untuk mempertahankan status, sementara ia memodelkan setiap vektor input melintasi jendela vektor input. Memodelkan dimensi waktu adalah ciri khas dari jaringan ini.

#### d. *Recursive Neural Networks*

*Recursive Neural Networks*, seperti halnya *Recurrent Neural Networks*, dapat menangani input variabel panjang. Bedanya *Recurrent Neural Networks* memiliki kemampuan untuk memodelkan struktur hierarki dalam dataset pelatihan. Lalu apa yang menarik dari jaringan ini? Secara umum, gambar memiliki pemandangan yang terdiri dari banyak objek. Adegan-adegan yang mendekonstruksi sering kali merupakan domain masalah yang menarik. Sifat rekursif dari dekonstruksi inilah yang menantang, karena bukan hanya permasalahan identifikasi objek dalam adegan saja, tetapi juga bagaimana agar objek-objek terkait dapat membentuk adegan.

Arsitektur jaringan ini terdiri dari matriks berbobot bersama dan struktur pohon biner yang memungkinkan jaringan rekursif mempelajari berbagai urutan kata atau bagian gambar. Ini berguna sebagai parser kalimat dan adegan. Jaringan ini menggunakan variasi *backpropagation* yang disebut *backpropagation through structure* (BPTS). Umpam-umpam maju terjadi dari bawah ke atas, dan *backpropagation* adalah dari atas ke bawah. Bisa dianggap, bahwa tujuan adalah seperti bagian atas pohon, sedangkan inputnya adalah bagian bawahnya.

### D. Pengembangan Framework Deep Learning

Pemrograman *Deep Learning* menggunakan Bahasa Pemrograman Python, sesuai dengan arsitektur-arsitektur *Deep Network*, saat ini tersedia banyak pilihan *framework*, antara lain:

1. TensorFlow
2. Keras
3. PyTorch
4. Caffe
5. Sonnet
6. MXNet
7. Gluon
8. Swift
9. Chainer
10. DL4J
11. ONNX
12. Darknet
13. Darkflow
14. YOLO

Masing-masing dijelaskan secara ringkas sebagai berikut. Sebagian besar diambil dari referensi [16]. Sedangkan untuk mempelajari lebih mendalam, silahkan dipelajari tersendiri. Meskipun tidak dibahas semua *framework*. Kami hanya memilih yang cukup populer menurut kami.

### 1. TensorFlow



Gambar 1. 6. Logo TensorFlow

TensorFlow adalah *framework* yang dibuat dan dikembangkan oleh Google yang digunakan untuk merancang, membangun, dan melatih model *Deep Learning*. *Framework* ini bisa dibilang paling populer saat ini. Gmail, Uber, Airbnb, Nvidia, dan banyak merek terkenal lainnya menggunakanannya.

Python adalah bahasa klien yang paling nyaman untuk bekerja dengan TensorFlow. Namun, ada juga antarmuka eksperimental yang tersedia di JavaScript, C++, Java dan Go, C # dan Julia. TensorFlow tidak hanya memperhitungkan klaster komputasi yang kuat tetapi juga kemampuan untuk

menjalankan model pada platform seluler seperti iOS dan Android.

TensorFlow membutuhkan banyak pengkodean. Itu tidak akan menghasilkan AI yang kuat dalam semalam, tetapi bisa jadi alat bantu penelitian *Deep Learning* agar sedikit kurang rumit. Perlu berpikir hati-hati tentang arsitektur Jaringan Saraf, penilaian dengan benar terhadap dimensi dan volume data input dan outputnya, agar dihasilkan karya AI yang hebat.

TensorFlow beroperasi dengan grafik perhitungan statis. Yaitu, pertama-tama didefinisikan grafiknya, lalu dijalankan perhitungannya. Jika diperlukan, dibuat perubahan pada arsitekturnya, dan dilatih kembali modelnya. Pendekatan seperti itu dipilih demi efisiensi. Tentunya perhitungan perbaikan dalam proses pembelajaran tanpa harus kehilangan kecepatan belajarnya. Dalam hal ini, pesaing utama TensorFlow adalah PyTorch.

TensorFlow berguna untuk membuat dan bereksperimen dengan arsitektur *Deep Learning*. Formulasinya nyaman untuk integrasi data seperti memasukkan grafik, tabel SQL, dan gambar secara bersamaan. Hal itu didukung oleh Google yang menjamin akan tetap ada untuk sementara waktu. Oleh karena itu masuk akal untuk menginvestasikan waktu dan sumber daya untuk mempelajarinya.

## 2. Keras



Gambar 1. 7. Logo Keras

Keras adalah *framework Machine Learning* yang mungkin bisa menjadi teman baru ketika bekerja dengan banyak data, terutama jika mengikuti *state-of-the-art* dari AI:

*Deep Learning*. Keras dapat digunakan sebagai API tingkat tinggi di atas *library* tingkat bawah populer lainnya seperti Theano dan CNTK selain Tensorflow. Penciptaan model *Deep Learning* yang sangat besar di Keras direduksi menjadi fungsi garis tunggal. Tetapi strategi ini membuat Keras menjadi lingkungan yang kurang dapat dikonfigurasi dibandingkan dengan framework tingkat rendah.

Keras adalah framework *Deep Learning* yang terbaik bagi mereka yang baru memulai. Ini ideal untuk belajar dan membuat prototipe konsep-konsep sederhana, untuk memahami inti dari berbagai model dan proses pembelajaran mereka.

Keras adalah API yang ditulis dengan indah. Sifat fungsional API membantu sepenuhnya dan sebagai jalan keluar menghasilkan aplikasi yang lebih eksotis. Keras tidak memblokir akses ke *framework* tingkat bawah. Keras menghasilkan kode yang jauh lebih mudah dibaca dan ringkas. Keras dengan model API serialisasi / deserialisasi, panggilan balik, dan streaming data menggunakan generator Python, sangatlah matang. Namun, Keras tidak dapat dibandingkan dengan Tensorflow karena mereka berada pada level abstraksi yang berbeda.

Tensorflow ada di Level Bawah. Di sinilah *framework* seperti MXNet, Theano, dan PyTorch. Ini adalah tingkat di mana operasi matematika seperti perkalian matriks-matriks umum, dan Jaringan Saraf primitif seperti operasi konvolusi diimplementasikan.

Keras ada di level yang lebih tinggi. Pada level ini, level primitif bawah digunakan untuk mengimplementasikan abstraksi Jaringan Saraf seperti lapisan-lapisan dan model. Secara umum, pada level ini, API memiliki manfaat lainnya seperti penghematan model dan pelatihan model.

### 3. PyTorch



Gambar 1. 8. Logo PyTorch

Alat perangkat lunak utama untuk *Deep Learning* setelah Tensorflow adalah PyTorch. *Framework* PyTorch dikembangkan untuk layanan Facebook tetapi sudah digunakan untuk menyelesaikan tugas-tugasnya sendiri oleh perusahaan seperti Twitter dan Salesforce.

Tidak seperti TensorFlow, *library* PyTorch beroperasi dengan grafik yang diperbarui secara dinamis. Ini berarti dimungkinkan dibuat perubahan pada arsitektur ketika proses sedang berlangsung. Di PyTorch, bisa digunakan *debugger* standar, seperti pdb atau PyCharm.

PyTorch memiliki proses pelatihan Jaringan Saraf yang sederhana dan jelas. Pada saat yang sama, PyTorch mendukung paralelisme data dan model pembelajaran terdistribusi, dan juga berisi banyak model pra-terlatih. PyTorch jauh lebih cocok untuk proyek kecil dan pembuatan prototipe. Namun ketika dibutuhkan solusi lintas platform, maka TensorFlow sepertinya pilihan yang lebih cocok.

### 4. Caffe



Gambar 1. 9. Logo Caffe

Caffe (*Convolutional Architecture for Fast Feature Embedding*) adalah *framework Deep Learning* yang mendukung banyak antarmuka seperti C, C++, Python, MATLAB serta Antarmuka Baris Perintah.

Caffe mendukung berbagai jenis arsitektur *Deep Learning* yang diarahkan pada segmentasi gambar dan klasifikasi gambar. Ini dikembangkan oleh *Berkeley AI Research* (BAIR) dan oleh kontributor komunitas.

Namun, Caffe tidak mendukung lapisan jaringan granularitas yang baik seperti pada TensorFlow. Mengingat arsitekturnya, dukungan keseluruhan terhadap jaringan berulang dan pemodelan bahasanya cukup buruk, dan pembangunan tipe lapisan kompleks harus dilakukan dalam bahasa tingkat rendah.

Caffe adalah jaringan *Deep Learning* yang populer untuk pengenalan gambar. Ini digunakan oleh para akademisi dan perusahaan pemula (*startup*) tetapi juga beberapa perusahaan besar seperti Yahoo. Caffe juga dapat melakukan komputasi pada *Central Processing Unit* (CPU) dan *Graphics Processing Unit* (GPU).

## 5. Sonnet



Gambar 1. 10. Logo Sonnet

Sonnet adalah *framework Deep Learning* yang dibangun di atas TensorFlow. Ini dirancang untuk membuat Jaringan Saraf dengan arsitektur yang kompleks oleh perusahaan terkenal dunia DeepMind. *Library* Sonnet berorientasi objek tingkat tinggi yang menghasilkan abstraksi ketika mengembangkan Jaringan Saraf atau algoritma *Machine Learning* lainnya.

Ide Sonnet adalah untuk membangun objek Python primer yang sesuai dengan bagian tertentu dari Jaringan Saraf. Selanjutnya, objek-objek ini terhubung secara independen ke komputasi grafik TensorFlow. Pemisahan proses pembuatan objek dan menghubungkannya dengan

grafik, dapat menyederhanakan desain arsitektur tingkat tinggi.

Keuntungan utama dari Sonnet, adalah dapat digunakan untuk mereproduksi penelitian-penelitian seperti yang telah ditunjukkan oleh makalah-makalah DeepMind. Hal ini tentunya DeepMind akan lebih memilih menggunakan Sonnet sendiri, daripada menggunakan framework-framework Deep Learning lainnya.

## 6. MXNet



Gambar 1. 11. Logo MXNet

MXNet adalah *framework Deep Learning* yang sangat skalabel yang dapat digunakan pada berbagai perangkat. Meskipun tampaknya tidak banyak digunakan dibandingkan dengan TensorFlow. Pertumbuhan MXNet kemungkinan banyak didorong oleh perkembangan proyek-proyek Apache.

*Framework* ini awalnya mendukung sejumlah besar bahasa (C++, Python, R, Julia, JavaScript, Scala, Go, dan bahkan Perl). Penekanan utamanya pada kenyataan bahwa framework ini paralel sangat efektif pada banyak GPU dan banyak mesin. Ini, khususnya, telah ditunjukkan oleh karyanya di Amazon *Web Services*.

MXNet mendukung banyak GPU (dengan perhitungan yang dioptimalkan dan pengalihan konteks cepat). Kode bersih dan mudah dirawat (Python, R, Scala, dan API lainnya). Kemampuan pemecahan masalah yang cepat (penting, bagi pemula yang belajar Deep Learning).

Meskipun tidak sepopuler TensorFlow, MXNet memiliki dokumentasi terperinci dan mudah digunakan. Dengan kemampuan untuk memilih antara gaya

pemrograman imperatif dan simbolis, menjadikannya kandidat *framework* yang baik bagi para pemula dan programmer yang berpengalaman.

## 7. Gluon



**Gambar 1. 12. Logo Gluon**

Gluon adalah *framework Deep Learning* yang luar biasa yang dapat digunakan untuk membuat model sederhana maupun canggih. Kekhususan proyek Gluon adalah antarmuka yang fleksibel yang menyederhanakan prototipe, membangun dan melatih model Deep Learning tanpa mengorbankan kecepatan belajar.

Gluon didasarkan pada MXNet dan menawarkan API sederhana yang menyederhanakan pembuatan model Deep Learning. Mirip dengan PyTorch, *framework* Gluon mendukung pekerjaan dengan grafik dinamis. Penggabungan Gluon dengan MXNet akan menghasilkan kinerja yang tinggi. Dari perspektif ini, Gluon terlihat seperti alternatif yang sangat menarik dari Keras untuk komputasi terdistribusi.

Gluon dapat mendefinisikan Jaringan Saraf menggunakan kode sederhana, jelas, dan ringkas. Ini menyatakan algoritma pelatihan dan model Jaringan Saraf, sehingga memberikan fleksibilitas dalam proses pengembangan tanpa mengorbankan kinerja. Gluon memungkinkan untuk mendefinisikan model Jaringan Saraf yang dinamis, artinya mereka dapat dibangun dengan cepat, dengan struktur apa pun, dan menggunakan aliran kontrol asli Python.

## 8. Swift



Gambar 1. 13. Logo Swift

Bagi sebagian programmer, ketika mendengar Swift, mungkin yang dipikirkan adalah pengembangan aplikasi berbasis iOS atau MacOS. Bagi yang tertarik dengan *Deep Learning*, maka seharusnya tidak asing dengan istilah Swift for Tensorflow (disingkat S4TF). Dengan integrasi langsung dengan bahasa pemrograman umum, S4TF memungkinkan menjadi algoritma yang lebih kuat untuk diekspresikan dalam penyelesaian masalah. Dengan menggunakan TensorFlow, API Swift memberi akses transparan ke semua operator TensorFlow tingkat rendah.

## 9. Chainer



Gambar 1. 14. Logo Chainer

Sampai munculnya DyNet di CMU, dan PyTorch di Facebook, Chainer adalah *framework* Jaringan Saraf terkemuka untuk grafik komputasi dinamis atau jaringan yang memungkinkan input dengan panjang berbeda-beda. Fitur yang populer untuk tugas-tugas *Neuro-linguistic programming* (NLP).

Kode ini ditulis dalam Python murni di atas *library* Numpy dan CuPy. Chainer adalah *framework* pertama yang menggunakan model arsitektur dinamis (seperti pada PyTorch).

Chainer beberapa kali memegang rekor mengalahkan *framework* yang lain (TensorFlow, MxNet dan CNTK) dalam hal efektivitas penskalaan ketika masalah pemodelan diselesaikan oleh Jaringan Saraf. Dengan tolok ukur kecepatan proses, Chainer lebih cepat daripada *framework* yang berorientasi Python lainnya. Terutama dengan kinerja pusat data berbasis GPU, Chainer memiliki kecepatan proses yang lebih baik daripada TensorFlow. TensorFlow sendiri dioptimalkan untuk arsitektur TPU.

## 10. DL4J



Gambar 1. 15. Logo DL4J

Bagi penggemar pemrograman Java, seharusnya tidak asing dengan DL4J (kependekan dari *Deep Learning for Java*). Pelatihan Jaringan Saraf dalam DL4J dilakukan secara paralel melalui iterasi menggunakan klaster. Proses ini didukung oleh arsitektur Hadoop dan Spark.

## 11. ONNX



Gambar 1. 16. Logo ONNX

Proyek ONNX lahir dari kolaborasi Microsoft dan Facebook dalam hal pencarian format terbuka untuk presentasi model *Deep Learning*. ONNX menyederhanakan proses transfer model antar algoritma yang bekerja dengan kecerdasan buatan. ONNX memungkinkan model untuk dilatih dalam satu *framework* dan ditransfer ke *framework* yang lain untuk menghasilkan kesimpulan. Model ONNX saat ini didukung oleh Caffe2, Microsoft *Cognitive Toolkit*, MXNet, PyTorch, konektor ke banyak *framework* dan library umum lainnya.

## 12. Darknet



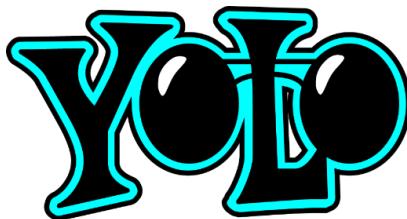
Gambar 1. 17. Logo Darknet.

Darknet adalah kerangka kerja Jaringan Saraf *open source* yang ditulis dalam bahasa pemrograman C dan CUDA. Cepat, mudah dipasang, dan mendukung komputasi CPU dan GPU.

### 13. Darkflow

Darkflow adalah *library Deep Learning* yang menerjemahkan *library* Darknet ke TensorFlow. Darkflow dapat digunakan dalam aplikasi lain yang mendukung bahasa pemrograman Python sebagai alternatif jika suatu perangkat tidak mendukung CUDA dan cuDNN. Melalui konfigurasi yang sederhana, adanya file cfg merupakan kode untuk model sedangkan *weights* merupakan bobot hasil pelatihan yang dapat digunakan untuk melakukan *transfer learning*.

### 14. YOLO



Gambar 1. 18. Logo YOLO.

*You only look once* (YOLO) adalah sistem deteksi objek *real-time* yang canggih. Pada Pascal Titan X dapat memproses gambar pada 30 FPS dan memiliki pemetaan sebesar 57,9% pada COCO *test-dev*. YOLOv3 menggunakan beberapa trik untuk meningkatkan pelatihan dan meningkatkan kinerja, termasuk: prediksi multi-skala, pengklasifikasi *backbone* yang lebih baik, dan banyak lagi.

Jika mayoritas sistem deteksi menggunakan pengklasifikasian atau *localizer* untuk melakukan deteksi dengan menerapkan model ke gambar di beberapa lokasi dan skala dan memberi nilai pada gambar sebagai bahan untuk pendekstrian. YOLO menggunakan pendekatan yang berbeda, yakni menerapkan Jaringan Saraf tunggal pada keseluruhan gambar. Jaringan ini akan membagi gambar menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas. Untuk setiap kotak wilayah

pembatas ditimbang probabilitasnya untuk diklasifikasi sebagai objek atau bukan. YOLO memiliki arsitektur CNN.

### E. Penalaan Parameter PID dengan Sistem Cerdas

Telah dijelaskan pada Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python Jilid 1, bahwa terdapat tiga parameter penentu keberhasilan proses pada sistem kendali Proporsional Integral dan Derivatif (PID), 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.

Saat ini telah banyak dilakukan penelitian penalaan parameter sistem kendali PID menggunakan sistem cerdas maupun *Machine Learning*. Beberapa diantaranya dapat disebutkan, antara lain: penalaan pengontrol PID secara *real-time* berbasis Jaringan Syaraf Tiruan [17], penalaan pengontrol PID menggunakan *Deep Reinforcement Learning* [18], penalaan pengontrol PID menggunakan algoritma metaheuristik (*Genetic Algorithms*, *Ant Colony Algorithms*, *Artificial Bee Colony Algorithms*, *Bat Algorithms*, *Bacteria Foraging Optimization*, *Particle Swarm Optimization*, *Cuckoo Optimization Algorithm*, *Simulated Annealing*, *Grev Wolf Optimization*, *Krill Herd Algorithm*, *Whale Optimizaiton Algorithm*) [19], dan lain-lain. Yang sudah dilakukan oleh Tim Penulis sendiri, yaitu penalaan pengontrol PID menggunakan *Deep Learning* [20]. Ini juga nantinya yang akan menjadi contoh penerapan dari Buku ini.

# BAB

# 2

# PEMROGRAMAN

# DEEP LEARNING

## A. Pemrograman Deep Learning dengan Python

Pembahasan mengenai Pemrograman *Deep Learning* dengan Python telah diuraikan secara panjang lebar dan juga sudah dilengkapi dengan contoh-contoh penerapannya, pada buku kami yang lain yang berjudul Pemrograman *Deep Learning* dengan Python [21].

Buku Pemrograman *Deep Learning* dengan Python ini untuk versi *e-Book* bisa diunduh pada alamat berikut ini:

1. [https://github.com/bsrahmat/ebook-03/blob/main/EBook03\\_DL.pdf](https://github.com/bsrahmat/ebook-03/blob/main/EBook03_DL.pdf)
2. <https://www.academia.edu/117612880>
3. <https://www.researchgate.net/publication/379872216>

Sedangkan jika menginginkan buku tersebut dalam versi cetak (bisa untuk hadiah, sebagai *handout workshop* atau pelatihan, dan lain-lain) bisa dibeli pada alamat berikut ini: <https://shopee.co.id/product/78709625/9069755032?smtt=0.78711099-1649659869.9>

*Source Code* contoh program pelengkap buku tersebut, juga dapat diunduh pada alamat berikut:

<https://github.com/bsrahmat/dl>

Dan jangan lupa, untuk mensiasati buku tersebut dapat mengunduh format bibtex-nya pada alamat:

[https://github.com/bsrahmat/ebook-03/blob/main/EBook03\\_DL.bib](https://github.com/bsrahmat/ebook-03/blob/main/EBook03_DL.bib). Terimakasih.

## B. Pemrograman Gerbang XOR

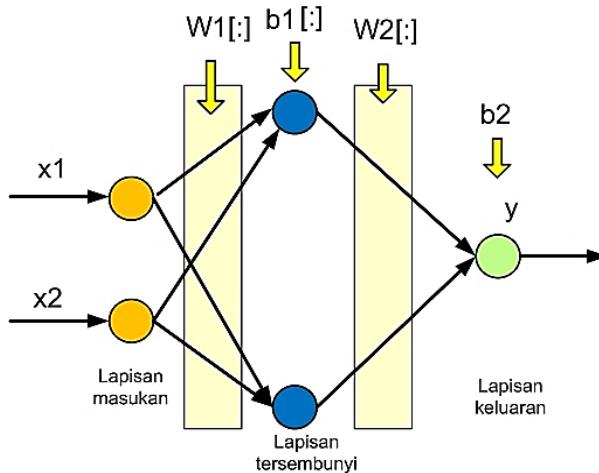
Salah satu contoh program *Deep Learning* pada buku Pemrograman *Deep Learning* dengan Python [21], yaitu Pemrograman Gerbang XOR. Dalam contoh ini, *Deep Learning* dipergunakan untuk memprediksi atau menirukan cara kerja mesin Gerbang Logika XOR. Program sederhana ini sebagai inspirasi bagaimana nantinya *Deep Learning* mendapatkan parameter pengendali PID terbaik secara otomatis yang dijelaskan pada pembahasan berikutnya.

Pemrograman Gerbang XOR, datasetnya mengikuti Tabel kebenaran dari Gerbang XOR. Pada saat proses pelatihan, *Deep Learning* akan mempelajari bagaimana pola masukan keluaran berdasarkan Data Latih Gerbang Logika XOR. Selanjutnya, pada saat pengujian, diberikan masukan sebarang, dia akan memprediksi keluarannya. Data Latih proses pelatihan Gerbang Logika XOR, seperti diperlihatkan pada Tabel 2.1.

**Tabel 2. 1. Data Latih Gerbang Logika XOR.**

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Arsitektur *Deep Learning* dengan dua masukan dan satu keluaran sesuai dengan *dataset* (Data Latih dan Data Uji) yang digunakan. Rancangan arsitektur *Deep Learning* diperlihatkan pada Gambar 2.1 [21].



**Gambar 2. 1. Arsitektur Deep Learning untuk menirukan Gerbang Logika XOR**

Pemrograman Python untuk menyelesaikan permasalahan Gerbang Logika XOR menggunakan *Deep Learning* dengan *framework* Keras, seperti dijabarkan pada program berikut.

Pertama siapkan data latih yang digunakan.

```
# Pasangan data latih
```

```
XOR_X = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])
```

```
XOR_Y = np.array([
    [0],
    [1],
    [1],
    [0]
])
```

Panggil library yang dibutuhkan, inisialisasi konstruktor, dan pembuatan struktur *Deep Learning*.

```
import numpy as np

# Impor `Sequential` dari` keras.models`
from keras.models import Sequential

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

# Inisialisasi konstruktor
model = Sequential()

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

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

# Tambahkan lapisan keluaran
model.add(Dense(1, activation='sigmoid'))
```

Selanjutnya, ketikkan skrip berikut ini, untuk model *Deep Learning*-nya, dapatkan bobot-bobot dan bias awal.

```
# Bentuk keluaran model
model.output_shape

# Ringkasan model
model.summary()

# Konfigurasi model
```

```

model.get_config()

# Buat daftar semua tensor bobot
model.get_weights()

```

Jika dijalankan, maka hasilnya seperti terlihat pada Gambar 2.2.

```

Model: "sequential_1"
-----  

Layer (type)           Output Shape        Param #
-----  

dense_3 (Dense)        (None, 2)           6  

-----  

dense_4 (Dense)        (None, 2)           6  

-----  

dense_5 (Dense)        (None, 1)           3  

-----  

Total params: 15
Trainable params: 15
Non-trainable params: 0
-----  

Out[7]: [array([[-0.00594807, -1.1232877 ],
               [-1.211806 ,  0.38877928]], dtype=float32),
         array([0., 0.], dtype=float32),
         array([[ 0.6797166 , -0.9488363 ],
               [-0.70436096,  0.28295314]], dtype=float32),
         array([0., 0.], dtype=float32),
         array([[ 0.59368217],
               [-0.5525098 ]], dtype=float32),
         array([0.], dtype=float32)]

```

**Gambar 2. 2. Model, bobot dan bias awal**

Untuk pelatihan *Deep Learning* silahkan ketikkan skrip berikut.

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(XOR_X,    XOR_Y,epochs=100000,    batch_size=1,
          verbose=1)

```

Jika dijalankan, hasilnya seperti terlihat pada Gambar 2.3.

```
Epoch 99992/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.6884 - accuracy: 0.5333
Epoch 99993/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.4075 - accuracy: 0.9000
Epoch 99994/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.4013 - accuracy: 0.7333
Epoch 99995/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.2857 - accuracy: 0.9000
Epoch 99996/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.4072 - accuracy: 0.9000
Epoch 99997/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.3319 - accuracy: 0.8333
Epoch 99998/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.6616 - accuracy: 0.5333
Epoch 99999/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.6885 - accuracy: 0.5333
Epoch 100000/1000000
4/4 [=====] - 0s 2ms/step - loss: 0.6884 - accuracy: 0.5333
```

```
Out[8]: <tensorflow.python.keras.callbacks.History at 0x23f21b40e50>
```

Gambar 2. 3. Proses pelatihan Deep Learning menggunakan Keras

Untuk menguji coba *Deep Learning* menggunakan Keras ini, silahkan diberikan masukan sebarang, misalkan diberikan masukan yang sama seperti data latihnya. Silahkan ketikkan skrip berikut.

```
Hasil_Prediksi_Keras = model.predict(XOR_X)
print(Hasil_Prediksi_Keras)
```

Jika dijalankan, hasilnya seperti terlihat pada Gambar 2.4.

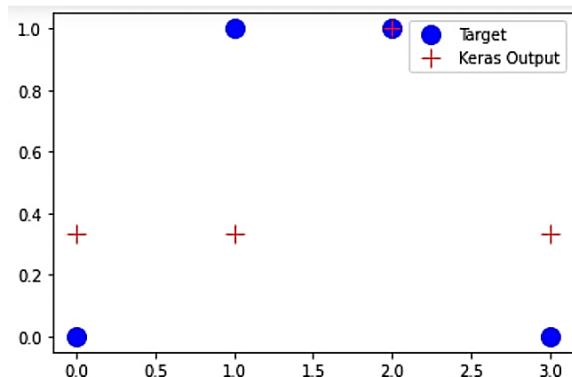
```
[[ 0.3335482]
 [ 0.3335482]
 [ 0.9997659]
 [ 0.3335482]]
```

**Gambar 2. 4. Keluaran hasil prediksi XOR menggunakan Deep Learning dengan Keras**

Selanjutnya untuk pengeplotan keluaran hasil prediksi XOR dengan Keras dibandingkan dengan target berupa keluaran yang diinginkan dari gerbang Logika XOR, silahkan ketikkan skrip berikut ini.

```
import matplotlib.pyplot as plt
plt.plot(XOR_Y, 'bo', label='Target', linewidth=2,
markersize=12)
plt.plot(Hasil_Prediksi_Keras, 'r+', label='Keras Output',
linewidth=2, markersize=12)
plt.legend(loc='upper right')
plt.show()
```

Jika dijalankan, hasilnya seperti terlihat pada Gambar 2.5.



**Gambar 2. 5. Perbandingan prediksi Keras dan target**

Selanjutnya untuk menghitung kesalahan prediksi XOR dengan Keras, digunakan *Mean Square Error* (MSE) dan *Root Mean Square Error* (RMSE). Berikut ini skrip python-nya.

```
from sklearn.metrics import mean_squared_error
from math import sqrt
mse2 = mean_squared_error(XOR_Y, Hasil_Prediksi_Keras)
rmse2 = sqrt(mean_squared_error(XOR_Y, Hasil_Prediksi_Keras))
print('MSE =',mse2)
print('RMSE =',rmse2)
```

Jika dijalankan, maka hasilnya seperti terlihat pada Gambar 2.6.

```
MSE = 0.1666667149924983
RMSE = 0.4082483496506732
```

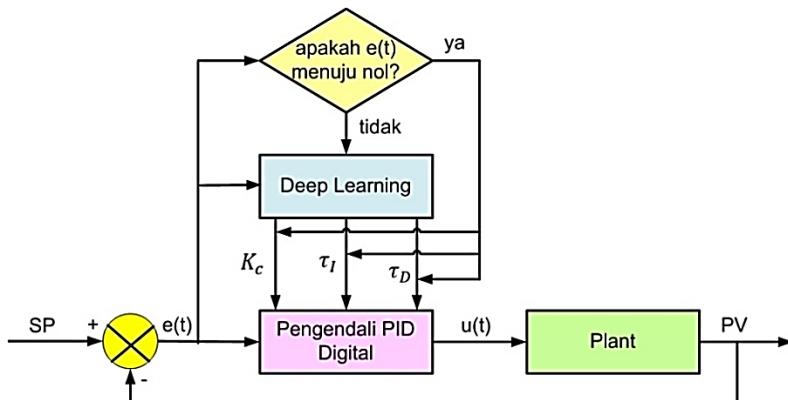
**Gambar 2. 6. Kesalahan hasil prediksi Keras**

Program Prediksi Gerbang XOR tersebut secara lengkap, bisa diunduh pada alamat berikut:

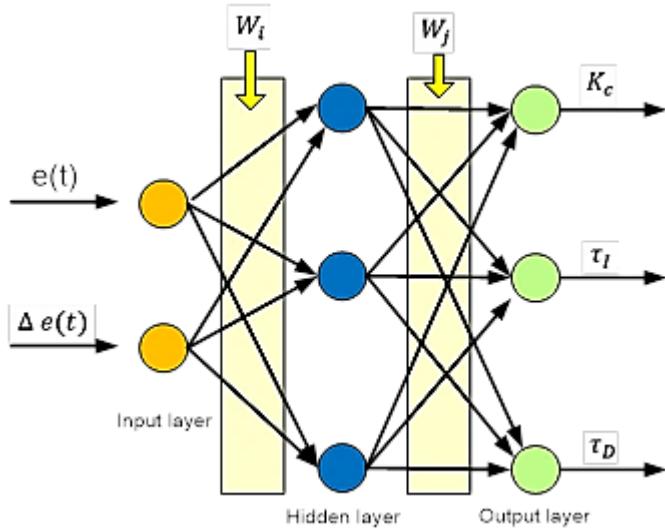
[https://github.com/bsrahmat/itclab-11/blob/main/XOR\\_Gate\\_Programming\\_using\\_Deep\\_Learning.ipynb](https://github.com/bsrahmat/itclab-11/blob/main/XOR_Gate_Programming_using_Deep_Learning.ipynb)

### C. Penalaan Parameter PID dengan Deep Learning

Selanjutnya akan diuraikan penalaan parameter PID dengan Deep Learning. Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* diperlihatkan seperti pada Gambar 2.7. Sistemnya membutuhkan mekanisme bagaimana melakukan penyesuaian nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$ , berdasarkan pembacaan *error*  $e(t)$  dan *delta\_error*  $\Delta e(t)$ . Pembacaan *error* dan *delta\_error* digunakan untuk memutuskan apakah perlu dilakukan pengubahan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  atau tidak. Jika *error* sudah konvergen kearah nol, maka nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  yang sudah ada ada dipertahankan. Namun jika masih jauh dari konvergen kearah nol, maka perlu dijalankan pengubahan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  dengan cara *update* bobot *Deep Learning*. Proses diulang-ulang sampai sistem pengendalian selesai. Arsitektur *Deep Learning* untuk keperluan ini diperlihatkan pada Gambar 2.8.



Gambar 2. 7. Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning



**Gambar 2. 8. Arsitektur Deep Learning untuk penyesuaian nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$**

Contoh implementasi pemrograman penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  berbasis *Deep Learning* menggunakan bahasa pemrograman Python diperlihatkan pada skrip-skrip program berikut ini.

```

import numpy as np # For matrix math
import matplotlib.pyplot as plt # For plotting

import sys # For printing

# Data Latih
# Misalkan error dan delta_error ideal untuk membangkitkan
# gain PID  $K_c$ ,  $\tau_I$  dan  $\tau_D$ , sebagai berikut:

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

```

```
[1.2, 0.1],  
[1, 0.1]  
])  
  
# Label untuk Data Latih.  
y = np.array([  
    [0.25, 4.31, 0.20],  
    [0.2, 4.1, 0.1],  
    [0.1, 4.0, 0],  
    [0.1, 4.0, 0]  
])  
  
# Impor `Sequential` dari` keras.models`  
from keras.models import Sequential  
  
# Impor `Dense` dari` keras.layers`  
from keras.layers import Dense  
  
# Inisialisasi konstruktor  
model = Sequential()  
  
# Tambahkan lapisan masukan  
model.add(Dense(2, activation='sigmoid', input_shape=(2,)))  
  
# Tambahkan satu lapisan tersembunyi  
model.add(Dense(3, activation='sigmoid'))  
  
# Tambahkan lapisan keluaran  
model.add(Dense(3, activation='sigmoid'))  
  
# Bentuk keluaran model  
model.output_shape  
  
# Ringkasan model  
model.summary()
```

```
# Konfigurasi model
model.get_config()

# Buat daftar semua tensor bobot
model.get_weights()
```

Sementara model *Deep Learning* seperti terlihat pada Gambar 2.9.

```
Model: "sequential_3"
-----
Layer (type)          Output Shape       Param #
dense_9 (Dense)      (None, 2)           6
dense_10 (Dense)     (None, 3)           9
dense_11 (Dense)     (None, 3)           12
-----
Total params: 27
Trainable params: 27
Non-trainable params: 0
```

### Gambar 2. 9. Model Deep Learning

Sedangkan keluaran berupa bobot-bobot Deep Learning seperti diperlihatkan pada Gambar 2.10.

```
[132]: [array([[-1.0747703 ,  0.13438284],
   [-0.3540696 ,  0.89655507]], dtype=float32),
 array([0., 0.], dtype=float32),
 array([[-0.93595296, -0.42637336, -0.8650211 ],
   [-0.655743 , -0.68840504, -0.01161814]], dtype=float32),
 array([0., 0., 0.], dtype=float32),
 array([[-0.7498157 , -0.29627323, -0.36188388],
   [ 0.01366854, -0.21947813,  0.9318299 ],
   [-0.59939027,  0.19210243, -0.5551851 ]], dtype=float32),
 array([0., 0., 0.], dtype=float32)]
```

### Gambar 2. 10. Bobot-bobot Deep Learning

Untuk pelatihan Deep Learning silahkan ketikkan skrip berikut.

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(X, y, epochs=100, batch_size=1, verbose=1)
```

Proses pelatihan seperti diperlihatkan pada Gambar 2.11.

```
Epoch 1/100
4/4 [=====] - 0s 2ms/step - loss: 0.7517 - accuracy: 0.0000e+00
Epoch 2/100
4/4 [=====] - 0s 2ms/step - loss: 0.7375 - accuracy: 0.0000e+00
Epoch 3/100
4/4 [=====] - 0s 4ms/step - loss: 0.7235 - accuracy: 0.2500
Epoch 4/100
4/4 [=====] - 0s 3ms/step - loss: 0.7095 - accuracy: 0.5000
Epoch 5/100
4/4 [=====] - 0s 2ms/step - loss: 0.6956 - accuracy: 1.0000
Epoch 6/100
4/4 [=====] - 0s 3ms/step - loss: 0.6817 - accuracy: 1.0000
Epoch 7/100
4/4 [=====] - 0s 2ms/step - loss: 0.6680 - accuracy: 1.0000
```

### Gambar 2. 11. Proses pelatihan Deep Learning

Untuk menampilkan hasil prediksi, silahkan ketikkan skrip berikut ini.

```
Hasil_Prediksi_Keras = model.predict(X)
print(Hasil_Prediksi_Keras)
```

Hasilnya seperti diperlihatkan pada Gambar 2.12.

```
[[ 0.2116262  0.7111814  0.26767278]
 [ 0.21551791  0.70976496  0.26997936]
 [ 0.21004573  0.71002096  0.27128503]
 [ 0.21138996  0.70934224  0.27257684]]
```

### Gambar 2. 12. Contoh hasil prediksi, keluaran Deep Learning

Silahkan dicoba diberi masukan  $e(t)$  sembarang.

```
# Pengujian ke-1
ujicoba1 = np.array([
    [1, 1]
])

outDL = model.predict(ujicoba1)

result_Kc = outDL[0,0]
result_tauI = outDL[0,1]
result_tauD = outDL[0,2]
```

Dari pengujian ke-1 di atas diperoleh hasil parameter PID seperti terlihat pada Gambar 2.13.

[140]: result\_Kc

[140]: 0.2116262

[141]: result\_tauI

[141]: 0.7111814

[142]: result\_tauD

[142]: 0.26767278

x

**Gambar 2. 13. Contoh hasil parameter PID keluaran Deep Learning**

Dari contoh proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID berbasis *Deep Learning* ini, diperoleh hasil  $K_c = 0.2116262$ ,  $\tau_I = 0.7111814$ , dan  $\tau_D = 0.26767278$ . Selanjutnya ketikkan skrip program berikut untuk mendapatkan gambaran visualisasinya.

```

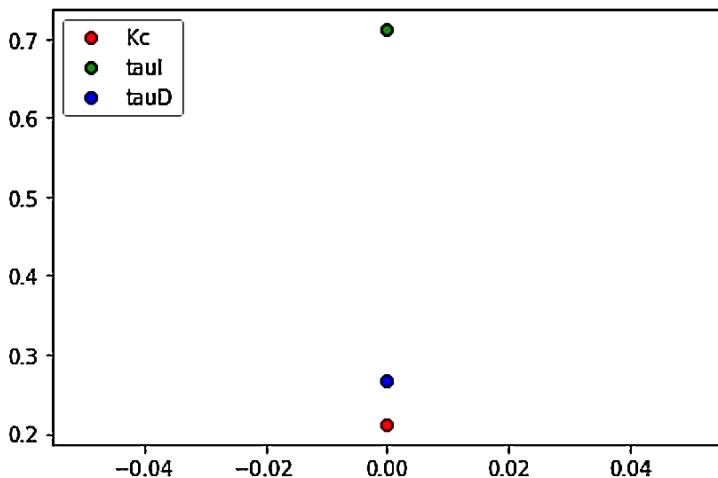
# Visualize
plt.plot(result_Kc, 'ro', label='Kc')
plt.plot(result_tauI, 'go', label='tauI')
plt.plot(result_tauD, 'bo', label='tauD')

#plt.xlabel('Kc, tauI, tauD');
#plt.legend((result_Kc, result_tauI, result_tauD), ('Kc', 'tauI',
#'tauD'))

plt.legend(loc='upper left')
#pylab.ylim(-1.5, 2.0)
plt.show()

```

Maka hasilnya seperti terlihat pada Gambar 2.14.



**Gambar 2. 14. Contoh hasil proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  berbasis Deep Learning**

Nilai-nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  hasil proses penalaan berbasis *Deep Learning* ini, selanjutnya dilihat pengaruhnya terhadap keberhasilan pengendalian. Ketikkan skrip program berikut ini.

```

import numpy as np
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 = 50.0 # 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

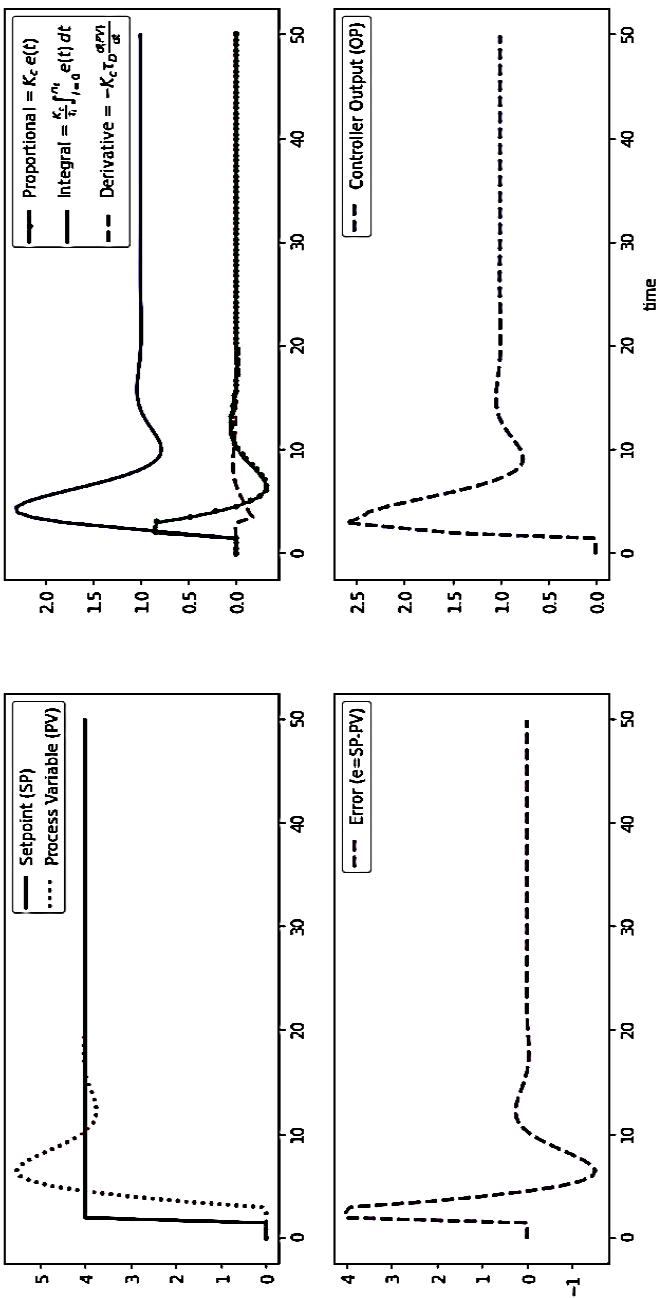
# plot PID response
plt.figure(1,figsize=(15,7))
plt.subplot(2,2,1)
plt.plot(t,SP,'k-',linewidth=2,label='Setpoint (SP)')
plt.plot(t,PV,'r-',linewidth=2,label='Process Variable (PV)')
plt.legend(loc='best')
plt.subplot(2,2,2)
plt.plot(t,P,'g-',linewidth=2,label=r'Proportional = $K_c \cdot e(t)$')
plt.plot(t,I,'b-',linewidth=2,label=r'Integral = $\frac{K_c}{\tau_I} \int_{i=0}^{n_t} e(t) \cdot dt$')
plt.plot(t,D,'r--',linewidth=2,label=r'Derivative = $-K_c \cdot \tau_D \frac{d(PV)}{dt}$')
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=result_Kc,min=-
0.2,max=1.0,step=0.05)
tauI_slide    =
wg.FloatSlider(value=result_tauI,min=0.01,max=5.0,step=0.1
)
tauD_slide    =
wg.FloatSlider(value=result_tauD,min=0.0,max=1.0,step=0.1)
wg.interact(pidPlot,      Kc=Kc_slide,      tauI=tauI_slide,
tauD=tauD_slide)
```

Hasilnya, dari proses pengendalian menggunakan pengendali PID melalui proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  menggunakan Deep Learning diperlihatkan pada Gambar 2.15. Terlihat sistem kendali PID-Deep Learning bekerja dengan baik, ditunjukkan dengan *error* menuju nol.

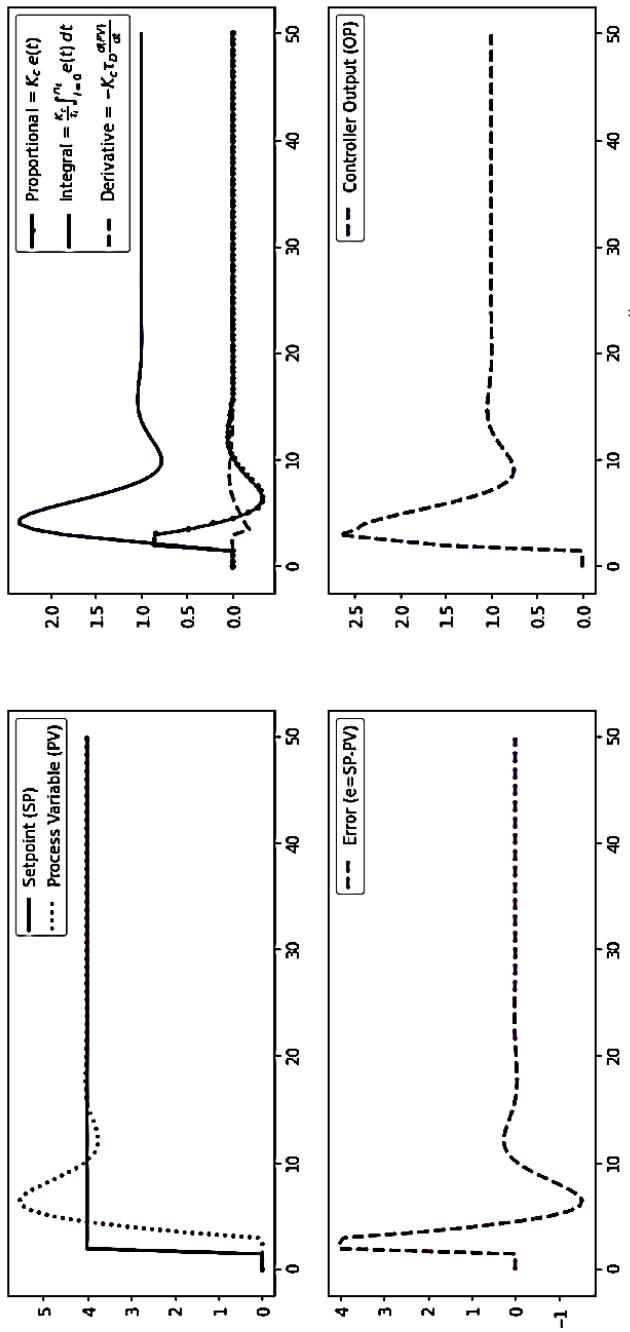


Gambar 2. 15. Hasil proses penilaian nilai  $K_c$ ,  $\tau_i$  dan  $\tau_D$  pada pengendali PID berbasis Deep Learning terhadap keluaran sistem

Selanjutnya, dengan cara yang sama, bisa dicoba dengan cara diberikan masukan  $e(t)$  sembarang yang lain. Misalkan seperti skrip berikut.

```
# Pengujian ke-2
ujicoba2 = np.array([
    [0.4, 1.2]
])
```

Setelah diproses dengan cara yang sama seperti cara pengujian ke-1, maka diperoleh hasil simulasi pengendalian PID-Deep Learning seperti terlihat pada Gambar 2.16. Terlihat sistem kendali PID-Deep Learning bekerja dengan baik, ditunjukkan dengan *error* menuju nol.



Gambar 2. 16. Hasil proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  yang lain pada pengendali PID berbasis Deep Learning terhadap keluaran sistem

Program penalaan parameter PID dengan *Deep learning* tersebut (Deep\_PID.ipynb) secara lengkap, dapat diunduh pada alamat berikut:

[https://github.com/bsrahmat/itclab-12/blob/main/Deep\\_PID.ipynb](https://github.com/bsrahmat/itclab-12/blob/main/Deep_PID.ipynb)

# BAB

# 3

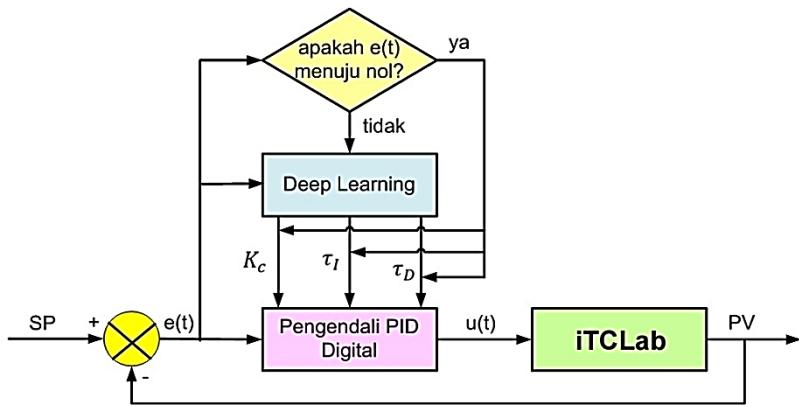
# PEMROGRAMAN

# IOT LANJUTAN

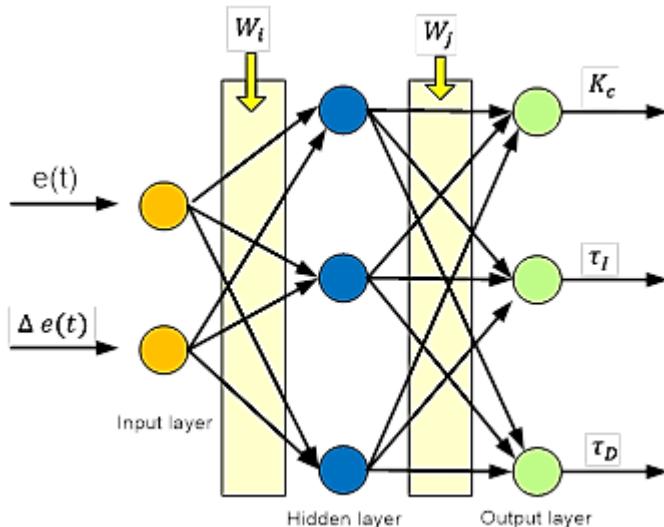
## A. Pengendalian PID-iTCLab dengan Deep Learning

Bab ini, merupakan inti dari Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python Jilid 2. Idenya kurang lebih sama dengan yang telah diuraikan sebelumnya, yaitu pada sub bab penalaan parameter PID dengan Deep Learning. Bedanya, jika yang sebelumnya hanya dalam bentuk simulasi, maka yang sekarang diterapkan langsung pada Kit iTCLab.

Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab mirip seperti yang telah dijelaskan sebelumnya (Gambar 2.7), yaitu diperlihatkan pada Gambar 3.1. Sistemnya membutuhkan mekanisme bagaimana melakukan penyesuaian nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$ , berdasarkan pembacaan *error*  $e(t)$  dan *delta\_error*  $\Delta e(t)$ . Pembacaan *error* dan *delta\_error* digunakan untuk memutuskan apakah perlu dilakukan pengubahan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  atau tidak. Jika *error* sudah konvergen kearah nol, maka nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  yang sudah ada dipertahankan. Namun jika masih jauh dari konvergen kearah nol, maka perlu dijalankan pengubahan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  dengan cara *update* bobot *Deep Learning*. Proses diulang-ulang sampai sistem pengendalian selesai. Gambaran proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab ini seperti diperlihatkan pada Gambar 3.1. Dan arsitektur Deep Learning yang digunakan ditunjukkan pada Gambar 3.2.



**Gambar 3. 1.** Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning pada Kit iTCLab



**Gambar 3. 2.** Arsitektur Deep Learning untuk penyesuaian nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada Sistem Kendali PID-iTCLab

Realisasinya disederhanakan, dengan cara disiapkan data latih sesuai dengan kebutuhan tersebut. Pasangan data input-output masing-masing terdiri dari error, error delta, dan tiga parameter PID. Data latih diperoleh dari kondisi sistem pengendalian PID yang ideal. Pasangan data terbaik diperoleh

dari beberapa kali eksperimen. Beberapa pasangan data latih ideal yang menghasilkan kinerja kontrol PID terbaik, dipilih.

Selanjutnya, dengan memanfaatkan pasangan data pelatihan yang ideal ini, proses pelatihan Deep Learning dijalankan untuk mendapatkan bobot yang paling terbaik. Bobot pelatihan ini kemudian digunakan sebagai mekanisme penyetelan otomatis dari Deep Learning, yang bertujuan untuk meningkatkan kinerja pengendali PID. Dengan sistem ini, proses auto-tuning beroperasi. Selanjutnya sinyal kontrol yang dihasilkan oleh pengontrol PID mengendalikan plant iTCLab sehingga diperoleh keluaran suhu yang sesuai dengan SetPoint yang diinginkan. Proses pengendalian ini berlanjut hingga dihentikan. Selanjutnya dilakukan analisis terhadap hasil pengendalian.

Berikut ini, file-file program yang dibutuhkan untuk proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab:

1. Program **Deep\_PID\_iTCLab.ino**  
[https://github.com/bsrahmat/itclab-13/blob/main/Deep\\_PID\\_iTCLab.ino](https://github.com/bsrahmat/itclab-13/blob/main/Deep_PID_iTCLab.ino)
2. Program **Deep\_PID\_iTCLab.ipynb**  
[https://github.com/bsrahmat/itclab-13/blob/main/Deep\\_PID\\_iTCLab.ipynb](https://github.com/bsrahmat/itclab-13/blob/main/Deep_PID_iTCLab.ipynb)
3. Program **itclab.py**  
<https://github.com/bsrahmat/itclab-13/blob/main/itclab.py>
4. Contoh hasil pengendaliannya bisa dilihat pada:  
[https://github.com/bsrahmat/itclab-13/blob/main/Deep\\_PID\\_iTCLab.pdf](https://github.com/bsrahmat/itclab-13/blob/main/Deep_PID_iTCLab.pdf)

Silahkan disiapkan ketiga program di atas. Program **Deep\_PID\_iTCLab.ino** silahkan di-upload ke Kit iTCLab terlebih dahulu. Berikut ini secara lengkap, program **Deep\_PID\_iTCLab.ino**.

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

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

const double batas_suhu_atas = 59;

// global variables
char Buffer[64];          // buffer for parsing serial input
String cmd;                // command
double pv = 0;              // pin value
float level;               // LED level (0-100%)
double Q1 = 0;              // value written to Q1 pin
double Q2 = 0;              // value written to Q2 pin
int iwrite = 0;             // integer value for writing
float dwrite = 0;            // float value for writing
int n = 10;                 // number of samples for each
temperature measurement

void parseSerial(void) {
    int ByteCount = 0;
    Serial.readBytesUntil(nl,Buffer,sizeof(Buffer));
    String read_ = String(Buffer);
    memset(Buffer,0,sizeof(Buffer));
}

```

```

// separate command from associated data
int idx = read_.indexOf(sp);
cmd = read_.substring(0,idx);
cmd.trim();
cmd.toUpperCase();

// extract data.ToInt() returns 0 on error
String data = read_.substring(idx+1);
data.trim();
pv = data.toFloat();
}

// Q1_max = 100%
// Q2_max = 100%

void dispatchCommand(void) {
    if (cmd == "Q1") {
        Q1 = max(0.0, min(25.0, pv));
        iwrite = int(Q1 * 2.0); // 10.? max
        iwrite = max(0, min(255, iwrite));
        ledcWrite(Q1Channel,iwrite);
        Serial.println(Q1);
    }
    else if (cmd == "Q2") {
        Q2 = max(0.0, min(25.0, pv));
        iwrite = int(Q2 * 2.0); // 10.? max
        iwrite = max(0, min(255, iwrite));
        ledcWrite(Q2Channel,iwrite);
        Serial.println(Q2);
    }
    else if (cmd == "T1") {
        float mV = 0.0;
        float degC = 0.0;
        for (int i = 0; i < n; i++) {
            mV = (float) analogRead(pinT1) * 0.322265625;

```

```

        degC = degC + mV/10.0;
    }
    degC = degC / float(n);

    Serial.println(degC);
}

else if (cmd == "T2") {
    float mV = 0.0;
    float degC = 0.0;
    for (int i = 0; i < n; i++) {
        mV = (float) analogRead(pinT2) * 0.322265625;
        degC = degC + mV/10.0;
    }
    degC = degC / float(n);
    Serial.println(degC);
}
else if ((cmd == "V") or (cmd == "VER")) {
    Serial.println("TCLab Firmware Version " + vers);
}
else if (cmd == "LED") {
    level = max(0.0, min(100.0, pv));
    iwrite = int(level * 0.5);
    iwrite = max(0, min(50, iwrite));
    ledcWrite(ledChannel, iwrite);
    Serial.println(level);
}
else if (cmd == "X") {
    ledcWrite(Q1Channel,0);
    ledcWrite(Q2Channel,0);
    Serial.println("Stop");
}
}

// check temperature and shut-off heaters if above high limit
void checkTemp(void) {
    float mV = (float) analogRead(pinT1) * 0.322265625;

```

```

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

// arduino startup
void setup() {
    //analogReference(EXTERNAL);
    Serial.begin(baud);
    while (!Serial) {
        ; // wait for serial port to connect.
    }

    // configure pinQ1 PWM functionalitites
    ledcSetup(Q1Channel, freq, resolutionQ1Channel);

    // attach the channel to the pinQ1 to be controlled
    ledcAttachPin(pinQ1, Q1Channel);
}

```

```
// configure pinQ2 PWM functionalitites
ledcSetup(Q2Channel, freq, resolutionQ2Channel);

// attach the channel to the pinQ2 to be controlled
ledcAttachPin(pinQ2, Q2Channel);

// configure pinLED PWM functionalitites
ledcSetup(ledChannel, freq, resolutionLedChannel);

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

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

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

Proses upload program dan ketika berhasil di-upload seperti terlihat pada Gambar 3.3 dan Gambar 3.4.

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

- Title Bar:** Deep\_PID\_iTCLab | Arduino IDE 2.3.1
- File Menu:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for back, forward, search, and refresh.
- Sketch Name:** Deep\_PID\_iTCLab.ino
- Code Area:** Displays the source code for the sketch. The code includes a header section and a footer section with version information and authorship.
- Output Area:** Shows the upload progress:
  - Hash of data verified.
  - Compressed 8192 bytes to 47
  - Writing at 0x00000000
  - Uploading...
- Status Bar:** Indicated the line number (Ln 7, Col 13), the board (DOIT ESP32 DEVKIT V1), the port (COM4), and the serial connection status (♂ 2).

Gambar 3.3. Proses upload Program Deep\_PID\_iTCLab.ino

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

- Title Bar:** Deep\_PID\_iTCLab | Arduino IDE 2.3.1
- File Menu:** File Edit Sketch Tools Help
- Sketch Name:** Deep\_PID\_iTCLab.ino
- Sketch Type:** DOIT ESP32 DEVKIT V1
- Code Content:**

```
/*  
 * iTCLab Internet-Based Temperature Control Lab Firmware  
 * Jeffrey Kantor, Initial Version  
 * John Hedengren, Modified  
 * Oct 2017  
 * Basuki Rahmat, Modified  
 * April 2022  
 */
```
- Output Window:** Shows the message "Leaving... Hard resetting via RTS pin..."
- Status Bar:** Ln 7, Col 13 DOIT ESP32 DEVKIT V1 on COM4

Gambar 3.4. Program Deep\_PID\_iTCLab.ino berhasil di-upload

Selanjutnya, program **itclab.py** dan **Deep\_PID\_iTCLab.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(list_ports.comports()):
            # Arduino Uno
            if port[2].startswith('USB VID:PID=16D0:0613'):
                port = port[0]
                found = True
            # Arduino HDUino
```

```

if port[2].startswith('USB VID:PID=1A86:7523'):
    port = port[0]
    found = True
# Arduino Leonardo
if port[2].startswith('USB VID:PID=2341:8036'):
    port = port[0]
    found = True
# Arduino ESP32
if port[2].startswith('USB VID:PID=10C4:EA60'):
    port = port[0]
    found = True
# Arduino ESP32 - Tipe yg berbeda
if port[2].startswith('USB VID:PID=1A86:55D4'):
    port = port[0]
    found = True
if (not found):
    print('Arduino COM port not found')
    print('Please ensure that the USB cable is connected')
    print('--- Printing Serial Ports ---')
    for port in list(serial.tools.list_ports.comports()):
        print(port[0] + ' ' + port[1] + ' ' + port[2])
    print('For Windows:')
    print(' Open device manager, select "Ports (COM & LPT)"')
    print(' Look for COM port of Arduino such as COM4')
    print('For MacOS:')
    print(' Open terminal and type: ls /dev/*.')
    print(' Search for /dev/tty.usbmodem* or /dev/tty.usbserial*. The port number is *.')
    print('For Linux')
    print(' Open terminal and type: ls /dev/tty*')
    print(' Search for /dev/ttUSB* or /dev/ttACM*.
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 **Deep\_PID\_iTCLab.ipynb**, untuk menguji proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab, programnya sebagai berikut.

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

```
# Machine Learning - Building Datasets and Model
# Impor `Sequential` dari` keras.models`
from keras.models import Sequential

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

# Inisialisasi konstruktor
model = Sequential()

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

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

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

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

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

```

# Bentuk keluaran model
model.output_shape

# Ringkasan model
model.summary()

# Konfigurasi model
model.get_config()

# Buat daftar semua tensor bobot
model.get_weights()

```

Coba dijalankan program di atas, seharusnya hasilnya berupa konfigurasi model dan bobotnya sebagai berikut.

```

Model: "sequential_2"
-----
Layer (type)          Output Shape       Param #
=====
dense_6 (Dense)      (None, 2)           6
dense_7 (Dense)      (None, 3)           9
dense_8 (Dense)      (None, 3)           12
=====
Total params: 27
Trainable params: 27
Non-trainable params: 0
-----
[19]: [array([[6.5393162e-01, 6.7713737e-02],
   [5.0425529e-04, 1.0704283e+00]], dtype=float32),
       array([0., 0.], dtype=float32),

```

**Gambar 3. 5. Contoh hasil konfigurasi model dan bobotnya**

Berikutnya ketikkan program untuk menjalankan proses pelatihan Deep Learning. Misalkan digunakan epoch 100, sebagai berikut.

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(X, y, epochs=100, batch_size=1, verbose=1)

```

Coba dijalankan program di atas, contoh hasilnya sebagai berikut.

```
Epoch 1/100
4/4 [=====] - 0s 2ms/step - loss: 1.2397 - accuracy: 0.0000e+00
Epoch 2/100
4/4 [=====] - 0s 2ms/step - loss: 1.2222 - accuracy: 0.0000e+00
Epoch 3/100
4/4 [=====] - 0s 3ms/step - loss: 1.2049 - accuracy: 0.0000e+00
Epoch 4/100
4/4 [=====] - 0s 3ms/step - loss: 1.1876 - accuracy: 0.0000e+00
Epoch 5/100
4/4 [=====] - 0s 3ms/step - loss: 1.1704 - accuracy: 0.5000
Epoch 6/100
4/4 [=====] - 0s 3ms/step - loss: 1.1532 - accuracy: 0.7500
Epoch 7/100
4/4 [=====] - 0s 4ms/step - loss: 1.1362 - accuracy: 1.0000
Epoch 8/100
4/4 [=====] - 0s 4ms/step - loss: 1.1191 - accuracy: 1.0000
```

Gambar 3.6. Contoh proses pelatihan Deep Learning

Selanjutnya dituliskan fungsi untuk pengendali PID-nya, sebagai berikut.

```
#####
#####  
##### Use this script for evaluating model predictions #####  
##### and PID controller performance for the TCLab #####  
##### Adjust only PID and model sections #####  
#####  
#####  
##### PID Controller #####  
#####  
#####  
# inputs -----  
# sp = setpoint  
# pv = current temperature  
# pv_last = prior temperature  
# ierr = integral error  
# dt = time increment between measurements  
# outputs -----  
# op = output of the PID controller  
# P = proportional contribution  
# I = integral contribution  
# D = derivative contribution  
def pid(sp,pv,pv_last,ierr,dt):  
    Kc = 10.0 # K/ %Heater  
    tauI = 50.0 # sec  
    tauD = 1.0 # sec  
    # Parameters in terms of PID coefficients  
    KP = Kc  
    KI = Kc/tauI  
    KD = Kc*tauD
```

```

# ubias for controller (initial heater)
op0 = 0
# upper and lower bounds on heater level
ophi = 100
oplo = 0
# calculate the error
error = sp-pv
# calculate the integral error
ierr = ierr + KI * error * dt
# calculate the measurement derivative
dpv = (pv - pv_last) / dt
# calculate the PID output
P = KP * error
I = ierr
D = -KD * dpv
op = op0 + P + I + D
# implement anti-reset windup
if op < oplo or op > ophi:
    I = I - KI * error * dt
    # clip output
    op = max(oplo,min(ophi,op))
# return the controller output and PID terms
return [op,P,I,D]

```

Selanjutnya dituliskan fungsi untuk pengendali PID menggunakan Deep Learning, sebagai berikut.

```

#####
#####
# PID Controller using Deep Learning      #
#####
#####

# 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_dl(sp,pv,pv_last,ierr,dt):

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

    outDL = model.predict(np.array([[error,delta_error]]))

    Kc = outDL[0,0]
    tauI = outDL[0,1]
    tauD = outDL[0,2]

    # Parameters in terms of PID coefficients
    KP = Kc
    KI = Kc/tauI
    KD = Kc*tauD
    # ubias for controller (initial heater)
    op0 = 0
    # upper and lower bounds on heater level
    ophi = 100
    oplo = 0

    # calculate the integral error
    ierr = ierr + KI * error * dt
    # calculate the measurement derivative

```

```

dpv = (pv - pv_last) / dt
# calculate the PID output
P = KP * error
I = ierr
D = -KD * dpv
op = op0 + P + I + D
# implement anti-reset windup
if op < oplo or op > ophi:
    I = I - KI * error * dt
    # clip output
    op = max(oplo,min(ophi,op))
# return the controller output and PID terms
return [op,P,I,D]

```

Selanjutnya diketikkan fungsi untuk model *First Order Plus Dead Time* (FOPDT) dan model keseimbangan energi (*Energy balance*) sebagai pembanding.

```

#####
#####
# FOPDT model           #
#####
#####

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

#####
#####
# Energy balance model   #
#####
#####
```

```

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

    # Temperature State
    T = x[0]

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

```

Selanjutnya silahkan diketikkan program utama proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab, sebagai berikut.

```

#####
#####
# Do not adjust anything below this point      #
#####
#
# Connect to Arduino
a = itclab.iTCLab()
#a.encode('utf-8').strip()#modification error
# Turn LED on

```

```

print('LED On')
a.LED(100)

# Run time in minutes
run_time = 15.0

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

# Temperature
# set point (degC)
Tsp1 = np.ones(loops) * 25.0
Tsp1[60:] = 45.0
Tsp1[360:] = 30.0
Tsp1[660:] = 35.0
T1 = np.ones(loops) * a.T1 # measured T (degC)
error_sp = np.zeros(loops)

Tsp2 = np.ones(loops) * 23.0 # set point (degC)
T2 = np.ones(loops) * a.T2 # measured T (degC)

# Predictions
Tp = np.ones(loops) * a.T1
error_eb = np.zeros(loops)
Tpl = np.ones(loops) * a.T1
error_fopdt = np.zeros(loops)

# impulse tests (0 - 100%)
Q1 = np.ones(loops) * 0.0
Q2 = np.ones(loops) * 0.0

print('Running Main Loop. Ctrl-C to end.')
print(' Time SP PV Q1 = P + I + D')
print(({:6.1f} {:6.2f} {:6.2f} {:6.2f} ' + \
'{:6.2f} {:6.2f} {:6.2f} {:6.2f'}).format( \

```

```

tm[0],Tsp1[0],T1[0], \
Q1[0],0.0,0.0,0.0))

# Create plot
plt.figure(figsize=(10,7))
plt.ion()
plt.show()

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

        # Record time and change in time
        t = time.time()
        dt = t - prev_time
        prev_time = t
        tm[i] = t - start_time

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

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

```

```

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

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

# 2. Based on Deep Learning Result
[Q1[i],P(ierr,D) = pid_dl(Tsp1[i],T1[i],T1[i-1],ierr,dt)

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

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

# Print line of data
print('{{:6.1f} {:6.2f} {:6.2f} {:6.2f} ' + \
'{:6.2f} {:6.2f} {:6.2f} {: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_dl.png')

# Allow user to end loop with Ctrl-C
except KeyboardInterrupt:

```

```
# Disconnect from Arduino
a.Q1(0)
a.Q2(0)
print('Shutting down')
a.close()
plt.savefig('test_PID_dl.png')

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

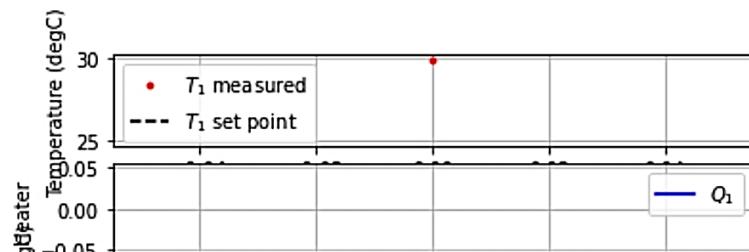
a.close()
```

Silahkan dijalankan, perhatikan hasil pengendalian yang dihasilkan. Contoh hasil pengendaliannya, seperti terlihat pada Gambar 3.7 dan Gambar 3.8.

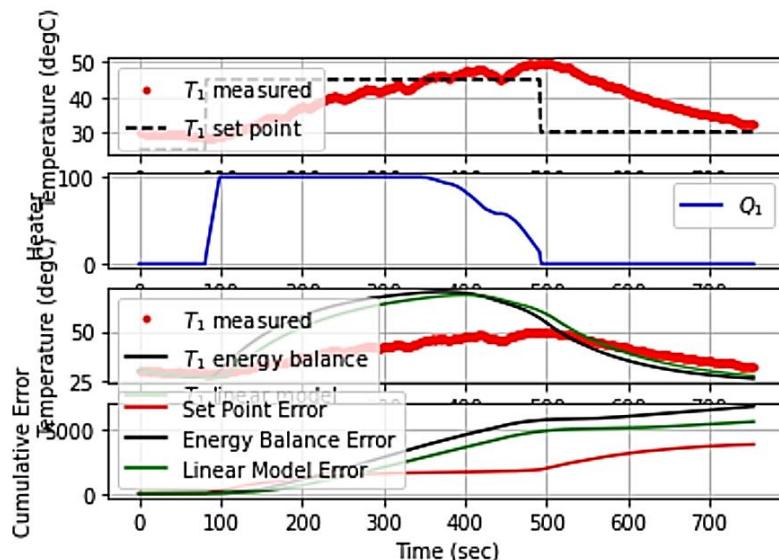
Opening connection  
 iTCLab connected via Arduino on port COM3  
 LED On  
 Running Main Loop. Ctrl-C to end.

Time	SP	PV	Q1	= P	+ I	+ D
0.0	25.00	29.92	0.00	0.00	0.00	0.00
1.0	25.00	29.86	0.00	-0.93	0.00	0.00

<Figure size 720x504 with 0 Axes>



Gambar 3. 7. Contoh hasil proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning pada Kit iTCLab



Gambar 3. 8. Contoh hasil pengendalian PID menggunakan Deep Learning pada Kit iTCLab

Contoh hasil pengendalian secara lengkap bisa dilihat pada:  
[https://github.com/bsrahmat/itclab-13/blob/main/Deep\\_PID\\_iTCLab.pdf](https://github.com/bsrahmat/itclab-13/blob/main/Deep_PID_iTCLab.pdf)

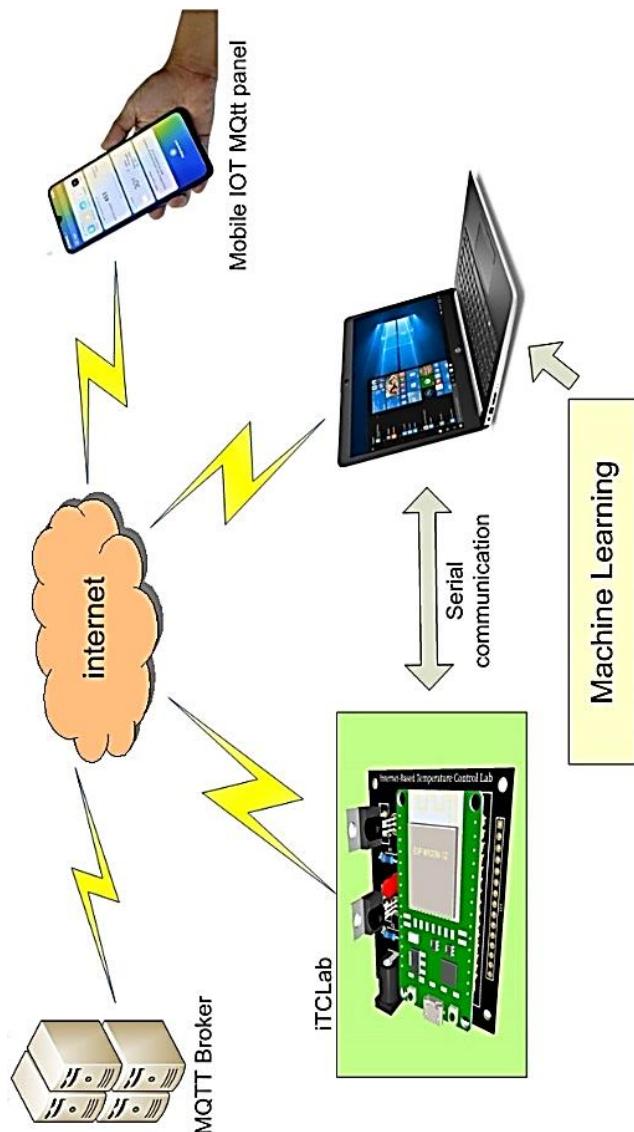
## B. Pemantauan PID-iTCLab-Deep Learning Via IoT

Selanjutnya, hasil eksperimen sebelumnya ditingkatkan, dengan cara dipantau hasil pengendaliannya, melalui Internet of Things (IoT). Proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab mirip seperti yang telah dijelaskan sebelumnya (Gambar 3.1). Sistemnya membutuhkan mekanisme bagaimana melakukan penyesuaian nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$ , berdasarkan pembacaan *error*  $e(t)$  dan *delta\_error*  $\Delta e(t)$ . Pembacaan *error* dan *delta\_error* digunakan untuk memutuskan apakah perlu dilakukan pengubahan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  atau tidak. Jika *error* sudah konvergen kearah nol, maka nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  yang sudah ada dipertahankan. Namun jika masih jauh dari konvergen kearah nol, maka perlu dijalankan pengubahan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  dengan cara *update* bobot *Deep Learning*. Proses diulang-ulang sampai sistem pengendalian selesai.

Juga telah dijelaskan sebelumnya, bahwa realisasinya disederhanakan. Dengan cara disiapkan data latih sesuai dengan kebutuhan tersebut. Pasangan data input-output masing-masing terdiri dari *error*, *error delta*, dan tiga parameter PID. Data latih diperoleh dari kondisi sistem pengendalian PID yang ideal. Pasangan data terbaik diperoleh dari beberapa kali eksperimen. Beberapa pasangan data latih ideal yang menghasilkan kinerja kontrol PID terbaik, dipilih.

Selanjutnya, dengan memanfaatkan pasangan data pelatihan yang ideal ini, proses pelatihan Deep Learning dijalankan untuk mendapatkan bobot yang paling terbaik. Bobot pelatihan ini kemudian digunakan sebagai mekanisme penyetelan otomatis dari Deep Learning, yang bertujuan untuk meningkatkan kinerja pengendali PID. Dengan sistem ini, proses auto-tuning beroperasi. Selanjutnya sinyal kontrol yang dihasilkan oleh pengontrol PID mengendalikan plant iTCLab

sehingga diperoleh keluaran suhu yang sesuai dengan SetPoint yang diinginkan. Proses pengendalian ini berlanjut hingga dihentikan. Selanjutnya dilakukan analisis terhadap hasil pengendalian. Bedanya dengan eksperimen kali ini, yaitu hasil proses pengendaliannya dapat dipantau dari jarak jauh melalui IoT. Arsitektur pemantauan hasil proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning pada Kit iTCLab melalui IoT, diperlihatkan pada Gambar 3.9.



Gambar 3. 9. Pemantauan hasil proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning pada Kit iTCLab melalui IoT

Berikut ini, file-file program yang dibutuhkan untuk melakukan pemantauan hasil proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning pada Kit iTCLab melalui IoT:

1. Program **Deep\_PID\_iTCLab\_IoT.ino**

[https://github.com/bsrahmat/itclab-14/blob/main/Deep\\_PID\\_iTCLab\\_IoT.ino](https://github.com/bsrahmat/itclab-14/blob/main/Deep_PID_iTCLab_IoT.ino)

2. Program **Deep\_PID\_iTCLab\_IoT.ipynb**

[https://github.com/bsrahmat/itclab-14/blob/main/Deep\\_PID\\_iTCLab\\_IoT.ipynb](https://github.com/bsrahmat/itclab-14/blob/main/Deep_PID_iTCLab_IoT.ipynb)

3. Program **itclab.py**

<https://github.com/bsrahmat/itclab-14/blob/main/itclab.py>

4. Contoh hasil pengendaliannya bisa dilihat pada:

[https://github.com/bsrahmat/itclab-14/blob/main/Deep\\_PID\\_iTCLab\\_IoT.pdf](https://github.com/bsrahmat/itclab-14/blob/main/Deep_PID_iTCLab_IoT.pdf)

Silahkan disiapkan ketiga program di atas. Program **Deep\_PID\_iTCLab\_IoT.ino** silahkan di-upload ke Kit iTCLab terlebih dahulu. Berikut ini secara lengkap, program **Deep\_PID\_iTCLab\_IoT.ino**. Proses *upload* program dan ketika berhasil di-*upload* seperti terlihat pada Gambar 3.10 dan Gambar 3.11.

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

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

```

const int resolutionQ1Channel = 8; //Resolution 8, 10, 12, 15
const int resolutionQ2Channel = 8; //Resolution 8, 10, 12, 15

const double batas_suhu_atas = 59;

// global variables
char Buffer[64];           // buffer for parsing serial input
String cmd;                 // command
double pv = 0;              // pin value
float level;                // LED level (0-100%)
double Q1 = 0;               // value written to Q1 pin
double Q2 = 0;               // value written to Q2 pin
int iwrite = 0;              // integer value for writing
float dwrite = 0;             // float value for writing
int n = 10;                  // number of samples for each
temperature measurement

void parseSerial(void) {
    int ByteCount = 0;
    Serial.readBytesUntil(nl,Buffer,sizeof(Buffer));
    String read_ = String(Buffer);
    memset(Buffer,0,sizeof(Buffer));

    // separate command from associated data
    int idx = read_.indexOf(sp);
    cmd = read_.substring(0,idx);
    cmd.trim();
    cmd.toUpperCase();

    // extract data.ToInt() returns 0 on error
    String data = read_.substring(idx+1);
    data.trim();
    pv = data.toFloat();
}

// Q1_max = 100%

```

```

// Q2_max = 100%

void dispatchCommand(void) {
    if (cmd == "Q1") {
        Q1 = max(0.0, min(25.0, pv));
        iwrite = int(Q1 * 2.0); // 10.? max
        iwrite = max(0, min(255, iwrite));
        ledcWrite(Q1Channel,iwrite);
        Serial.println(Q1);
    }
    else if (cmd == "Q2") {
        Q2 = max(0.0, min(25.0, pv));
        iwrite = int(Q2 * 2.0); // 10.? max
        iwrite = max(0, min(255, iwrite));
        ledcWrite(Q2Channel,iwrite);
        Serial.println(Q2);
    }
    else if (cmd == "T1") {
        float mV = 0.0;
        float degC = 0.0;
        for (int i = 0; i < n; i++) {
            mV = (float) analogRead(pinT1) * 0.322265625;
            degC = degC + mV/10.0;
        }
        degC = degC / float(n);

        Serial.println(degC);
    }
    else if (cmd == "T2") {
        float mV = 0.0;
        float degC = 0.0;
        for (int i = 0; i < n; i++) {
            mV = (float) analogRead(pinT2) * 0.322265625;
            degC = degC + mV/10.0;
        }
        degC = degC / float(n);
    }
}

```

```

    Serial.println(degC);
}

else if ((cmd == "V") or (cmd == "VER")) {
    Serial.println("TCLab Firmware Version " + vers);
}

else if (cmd == "LED") {
    level = max(0.0, min(100.0, pv));
    iwrite = int(level * 0.5);
    iwrite = max(0, min(50, iwrite));
    ledcWrite(ledChannel, iwrite);
    Serial.println(level);
}

else if (cmd == "X") {
    ledcWrite(Q1Channel,0);
    ledcWrite(Q2Channel,0);
    Serial.println("Stop");
}

}

// check temperature and shut-off heaters if above high limit
void checkTemp(void) {
    float mV = (float) analogRead(pinT1) * 0.322265625;
    //float degC = (mV - 500.0)/10.0;
    float degC = mV/10.0;
    if (degC >= batas_suhu_atas) {
        Q1 = 0.0;
        Q2 = 0.0;
        ledcWrite(Q1Channel,0);
        ledcWrite(Q2Channel,0);
        //Serial.println("High Temp 1 (> batas_suhu_atas): ");
        Serial.println(degC);
    }

    mV = (float) analogRead(pinT2) * 0.322265625;
    //degC = (mV - 500.0)/10.0;
    degC = mV/10.0;
    if (degC >= batas_suhu_atas) {

```

```
Q1 = 0.0;
Q2 = 0.0;
ledcWrite(Q1Channel,0);
ledcWrite(Q2Channel,0);
//Serial.println("High Temp 2 (> batas_suhu_atas): ");
Serial.println(degC);
}

}

// arduino startup
void setup() {
//analogReference(EXTERNAL);
Serial.begin(baud);
while (!Serial) {
; // wait for serial port to connect.
}

// configure pinQ1 PWM functionalitites
ledcSetup(Q1Channel, freq, resolutionQ1Channel);

// attach the channel to the pinQ1 to be controlled
ledcAttachPin(pinQ1, Q1Channel);

// configure pinQ2 PWM functionalitites
ledcSetup(Q2Channel, freq, resolutionQ2Channel);

// attach the channel to the pinQ2 to be controlled
ledcAttachPin(pinQ2, Q2Channel);

// configure pinLED PWM functionalitites
ledcSetup(ledChannel, freq, resolutionLedChannel);

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

ledcWrite(Q1Channel,0);
```

```
ledcWrite(Q2Channel,0);
}

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

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

- Title Bar:** Deep\_PID\_iTCLab\_IoT | Arduino IDE 2.3.1
- File Menu:** File Edit Sketch Tools Help
- Sketch Name:** Deep\_PID\_iTCLab\_IoTino
- Code Area:** The code is for a main event loop. Lines 189-195 are shown:

```
189 // arduino main event loop
190 void loop() {
191     parseSerial();
192     dispatchCommand();
193     checkTemp();
194 }
```
- Upload Buttons:** A yellow arrow pointing right, a blue circular icon with a gear, and a green circular icon with a checkmark.
- Output Window:** Shows the upload process:

```
Writing at 0x00002a05... (40 %)
Writing at 0x0002f51c... (50 %)
Writing at 0x000349d0... (60 %)
Writing at 0x0003befc Uploading...
```
- Status Bar:** Ln 194, Col 2 DOIT ESP32 DEVKIT V1 on COM13 ⚡ 2

Gambar 3.10. Proses upload Program Deep\_PID\_iTCLab\_IoT.ino

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

- File Menu:** File Edit Sketch Tools Help
- Sketch Name:** Deep\_PID\_iTCLab\_IoT.ino
- Board:** DOIT ESP32 DEVKIT V1
- Port:** COM13
- Upload Buttons:** Checkmark, Refresh, Upload, and a downward arrow.
- Code Area:** The code for `Deep_PID_iTCLab_IoT.ino` is displayed, showing the main loop and various helper functions. Lines 189 through 195 are highlighted.

```
Deep_PID_iTCLab_IoT.ino
189 // arduino main event loop
190 void loop() {
191     parseSerial();
192     dispatchCommand();
193     checkTemp();
194 }
195
```

- Output Area:** The output window shows the following text:

```
Output
Hash or data verified.
...
Leaving...
Hard resetting via RTS pin...
```
- Bottom Status:** Ln 194, Col 2 DOIT ESP32 DEVKIT V1 on COM13 2

Gambar 3.11. Program Deep\_PID\_iTCLab\_IoT.ino berhasil di-upload

Selanjutnya, program **itclab.py** dan **Deep\_PID\_iTCLab\_IoT.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(list_ports.comports()):
            # Arduino Uno
            if port[2].startswith('USB VID:PID=16D0:0613'):
                port = port[0]
                found = True
            # Arduino HDUino
            if port[2].startswith('USB VID:PID=1A86:7523'):
```

```

port = port[0]
found = True
# Arduino Leonardo
if port[2].startswith('USB VID:PID=2341:8036'):
    port = port[0]
    found = True
# Arduino ESP32
if port[2].startswith('USB VID:PID=10C4:EA60'):
    port = port[0]
    found = True
# Arduino ESP32 - Tipe yg berbeda
if port[2].startswith('USB VID:PID=1A86:55D4'):
    port = port[0]
    found = True
if (not found):
    print('Arduino COM port not found')
    print('Please ensure that the USB cable is connected')
    print('--- Printing Serial Ports ---')
    for port in list(serial.tools.list_ports.comports()):
        print(port[0] + ' ' + port[1] + ' ' + port[2])
    print('For Windows:')
    print(' Open device manager, select "Ports (COM & LPT)"')
    print(' Look for COM port of Arduino such as COM4')
    print('For MacOS:')
    print(' Open terminal and type: ls /dev/*.')
    print(' Search for /dev/tty.usbmodem* or /dev/tty.usbserial*. The port number is *.')
    print('For Linux')
    print(' Open terminal and type: ls /dev/tty*')
    print(' Search for /dev/ttyUSB* or /dev/ttyACM*.
The port number is *.')
    print('')
    port = input('Input port: ')
    # or hard-code it here
    #port = 'COM3' # for Windows

```

```

#port = '/dev/tty.wchusbserial1410' # for MacOS
return port

def stop(self):
    return self.read('X')

def version(self):
    return self.read('VER')

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

@property
def T2(self):
    self._T2 = float(self.read('T2'))
    return self._T2

def LED(self,pwm):
    pwm = max(0.0,min(100.0,pwm))/2.0
    self.write('LED',pwm)
    return pwm

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

def Q2(self,pwm):
    pwm = max(0.0,min(100.0,pwm))
    self.write('Q2',pwm)
    return pwm

# save txt file with data and set point
# t = time

```

```

# u1,u2 = heaters
# y1,y2 = tempeatures
# sp1,sp2 = setpoints
def save_txt(self,t,u1,u2,y1,y2,sp1,sp2):
    data = np.vstack((t,u1,u2,y1,y2,sp1,sp2)) # vertical stack
    data = data.T          # transpose data
    top = 'Time (sec), Heater 1 (%), Heater 2 (%), '\
        + 'Temperature 1 (degC), Temperature 2 (degC), '\
        + 'Set Point 1 (degC), Set Point 2 (degC)'

    np.savetxt('data.txt',data,delimiter=',',header=top,comments="")

def read(self,cmd):
    cmd_str = self.build_cmd_str(cmd,"")
    try:
        self.sp.write(cmd_str.encode())
        self.sp.flush()
    except Exception:
        return None
    return self.sp.readline().decode('UTF-8').replace("\r\n",
"")

def write(self,cmd,pwm):
    cmd_str = self.build_cmd_str(cmd,(pwm,))
    try:
        self.sp.write(cmd_str.encode())
        self.sp.flush()
    except:
        return None
    return self.sp.readline().decode('UTF-8').replace("\r\n",
"")

def build_cmd_str(self,cmd, args=None):
    """
    Build a command string that can be sent to the arduino.

```

```

Input:
    cmd (str): the command to send to the arduino, must
not
        contain a % character
    args (iterable): the arguments to send to the command
    """
if args:
    args = ''.join(map(str, args))
else:
    args = ""
return "{cmd} {args}\n".format(cmd=cmd, args=args)

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

```

Selanjutnya, program Python Jupyter Notebook, dengan nama file program **Deep\_PID\_iTCLab\_IoT.ipynb**, untuk menguji pemantauan hasil proses penalaan nilai Kc, tI dan tD pada pengendali PID menggunakan Deep Learning pada Kit iTCLab melalui IoT, programnya sebagai berikut.

```

import itclab
import numpy as np
import time
import matplotlib.pyplot as plt
from scipy.integrate import odeint
import random
from paho.mqtt import client as mqtt_client

```

```
# Machine Learning - Building Datasets and Model
# Impor `Sequential` dari` keras.models`
from keras.models import Sequential

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

# Inisialisasi konstruktor
model = Sequential()

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

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

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

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

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

```
# Bentuk keluaran model  
model.output_shape  
  
# Ringkasan model  
model.summary()  
  
# Konfigurasi model  
model.get_config()  
  
# Buat daftar semua tensor bobot  
model.get_weights()
```

Coba dijalankan program di atas, seharusnya hasilnya berupa konfigurasi model dan bobotnya sebagai berikut.

```
Model: "sequential"
-----
Layer (type)          Output Shape         Param #
=====
dense (Dense)        (None, 2)            6
dense_1 (Dense)      (None, 3)            9
dense_2 (Dense)      (None, 3)            12
=====
Total params: 27
Trainable params: 27
Non-trainable params: 0
```

Gambar 3.12. Contoh hasil konfigurasi model dan bobotnya

Selanjutnya ketikkan program untuk menjalankan proses pelatihan Deep Learning. Misalkan digunakan epoch 10, sebagai berikut.

```
model.compile(loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy'])

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

Coba dijalankan program di atas, contoh hasilnya sebagai berikut.

```
Epoch 1/10
4/4 [=====] - 2s 19ms/step - loss: 1.1343 - accuracy: 0.0000e+00
Epoch 2/10
4/4 [=====] - 0s 2ms/step - loss: 1.1146 - accuracy: 0.0000e+00
Epoch 3/10
4/4 [=====] - 0s 2ms/step - loss: 1.0950 - accuracy: 0.0000e+00
Epoch 4/10
4/4 [=====] - 0s 4ms/step - loss: 1.0756 - accuracy: 0.0000e+00
Epoch 5/10
4/4 [=====] - 0s 4ms/step - loss: 1.0561 - accuracy: 0.0000e+00
Epoch 6/10
4/4 [=====] - 0s 2ms/step - loss: 1.0367 - accuracy: 0.0000e+00
Epoch 7/10
4/4 [=====] - 0s 2ms/step - loss: 1.0173 - accuracy: 0.0000e+00
Epoch 8/10
```

Gambar 3.13. Contoh proses pelatihan Deep Learning

Selanjutnya dituliskan fungsi untuk pengendali PID-nya, sebagai berikut.

```
#####
#####  
#### # Use this script for evaluating model predictions #  
#### # and PID controller performance for the TCLab #  
#### # Adjust only PID and model sections #  
#####  
####  
  
#####
#####  
#### # PID Controller #  
#####  
####  
#### # inputs -----  
#### # sp = setpoint  
#### # pv = current temperature  
#### # pv_last = prior temperature  
#### # ierr = integral error  
#### # dt = time increment between measurements  
#### # outputs -----  
#### # op = output of the PID controller  
#### # P = proportional contribution  
#### # I = integral contribution  
#### # D = derivative contribution  
def pid(sp,pv,pv_last,ierr,dt):  
    Kc = 10.0 # K/ %Heater  
    tauI = 50.0 # sec  
    tauD = 1.0 # sec  
    # Parameters in terms of PID coefficients  
    KP = Kc  
    KI = Kc/tauI  
    KD = Kc*tauD  
    # ubias for controller (initial heater)
```

```

op0 = 0
# upper and lower bounds on heater level
ophi = 100
oplo = 0
# calculate the error
error = sp-pv
# calculate the integral error
ierr = ierr + KI * error * dt
# calculate the measurement derivative
dpv = (pv - pv_last) / dt
# calculate the PID output
P = KP * error
I = ierr
D = -KD * dpv
op = op0 + P + I + D
# implement anti-reset windup
if op < oplo or op > ophi:
    I = I - KI * error * dt
    # clip output
    op = max(oplo,min(ophi,op))
# return the controller output and PID terms
return [op,P,I,D]

```

Selanjutnya dituliskan fungsi untuk pengendali PID menggunakan Deep Learning, sebagai berikut.

```

#####
######
# PID Controller using Deep Learning      #
######
#####

# 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_dl(sp,pv,pv_last,ierr,dt):

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

    outDL = model.predict(np.array([[error,delta_error]]))

    Kc = outDL[0,0]
    tauI = outDL[0,1]
    tauD = outDL[0,2]

    # Parameters in terms of PID coefficients
    KP = Kc
    KI = Kc/tauI
    KD = Kc*tauD
    # ubias for controller (initial heater)
    op0 = 0
    # upper and lower bounds on heater level
    ophi = 100
    oplo = 0

    # calculate the integral error
    ierr = ierr + KI * error * dt
    # calculate the measurement derivative
    dpv = (pv - pv_last) / dt

```

```

# calculate the PID output
P = KP * error
I = ierr
D = -KD * dpv
op = op0 + P + I + D
# implement anti-reset windup
if op < oplo or op > ophi:
    I = I - KI * error * dt
    # clip output
    op = max(oplo,min(ophi,op))
# return the controller output and PID terms
return [op,P,I,D]

```

Selanjutnya diketikkan fungsi untuk model *First Order Plus Dead Time* (FOPDT) dan model keseimbangan energi (*Energy balance*) sebagai pembanding.

```

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

#####
#####
# Energy balance model      #
#####
#####

```

```

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

    # Temperature State
    T = x[0]

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

```

Selanjutnya, bagian penting dari program ini. Ketikkan program agar sistem dapat terhubung ke *MQTT Broker*. Seperti telah dijelaskan di Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python Jilid 1, digunakan *MQTT Broker* hivemq.com. Berikut ini programnya.

```

# Connect to MQTT Broker for Monitoring
broker = 'broker.hivemq.com'
port = 1883
client_id = f'python-mqtt-{random.randint(0, 1000)}'

def connect_mqtt():
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")

```

```

else:
    print("Failed to connect, return code %d\n", rc)

client = mqtt_client.Client(client_id)
client.on_connect = on_connect
client.connect(broker, port)
return client

client = connect_mqtt()
client.loop_start()

```

Silahkan dijalankan, jika berhasil terhubung, terlihat seperti pada Gambar 3.14.

```

        print("Connected to MQTT Broker!")
else:
    print("Failed to connect, return code %d\n", rc)

client = mqtt_client.Client(client_id)
client.on_connect = on_connect
client.connect(broker, port)
return client

client = connect_mqtt()
client.loop_start()

```

Connected to MQTT Broker!

**Gambar 3. 14. Terhubung ke Broker MQTT**

Selanjutnya silahkan diketikkan program utama pemantauan proses penalaan nilai  $K_c$ ,  $T_I$  dan  $T_D$  pada pengendali PID menggunakan *Deep Learning* pada Kit iTCLab melalui IoT, sebagai berikut.

```

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

```

# Connect to Arduino
a = itclab.iTCLab()
#a.encode('utf-8').strip()#modification error
# Turn LED on
print('LED On')
a.LED(100)

# Run time in minutes
run_time = 15.0

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

# Temperature
# set point (degC)
Tsp1 = np.ones(loops) * 25.0
Tsp1[60:] = 45.0
Tsp1[360:] = 30.0
Tsp1[660:] = 35.0
T1 = np.ones(loops) * a.T1 # measured T (degC)
error_sp = np.zeros(loops)

Tsp2 = np.ones(loops) * 23.0 # set point (degC)
T2 = np.ones(loops) * a.T2 # measured T (degC)

# Predictions
Tp = np.ones(loops) * a.T1
error_eb = np.zeros(loops)
Tpl = np.ones(loops) * a.T1
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] = (Tpl[i-1]-Tss) * z \
+ (Q1[max(0,i-int(thetaP)-1)]-Qss)*(1-z)*Kp \
+ Tss

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

# 2. Based on Deep Learning Result
[Q1[i],P,ierr,D] = pid_dl(Tsp1[i],T1[i],T1[i-1],ierr,dt)

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

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

# Print line of data
print('{:6.1f} {:6.2f} {:6.2f} ' + \
'{:6.2f} {:6.2f} {:6.2f} {:6.2f}').format( \
tm[i],Tsp1[i],T1[i], \
Q1[i],P,ierr,D))

```

```

# Publish data to MQTT Broker
pub_sp = client.publish('SetPoint', Tsp1[i])
pub_pv1 = client.publish('Suhu1', T1[i])
pub_op = client.publish('Nilai_op', Q1[i])

# Plot
plt.clf()
ax=plt.subplot(4,1,1)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'r.',label=r'$T_1$ measured')
plt.plot(tm[0:i],Tsp1[0:i],'k--',label=r'$T_1$ set point')
plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(4,1,2)
ax.grid()
plt.plot(tm[0:i],Q1[0:i],'b-',label=r'$Q_1$')
plt.ylabel('Heater')
plt.legend(loc='best')
ax=plt.subplot(4,1,3)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'r.',label=r'$T_1$ measured')
plt.plot(tm[0:i],Tp[0:i],'k-',label=r'$T_1$ energy balance')
plt.plot(tm[0:i],TpI[0:i],'g-',label=r'$T_1$ linear model')
plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(4,1,4)
ax.grid()
plt.plot(tm[0:i],error_sp[0:i],'r-',label='Set Point Error')
plt.plot(tm[0:i],error_eb[0:i],'k-',label='Energy Balance Error')
plt.plot(tm[0:i],error_fopdt[0:i],'g-',label='Linear Model Error')
plt.ylabel('Cumulative Error')
plt.legend(loc='best')
plt.xlabel('Time (sec)')
plt.draw()

```

```

plt.pause(0.05)

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

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

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

a.close()

```

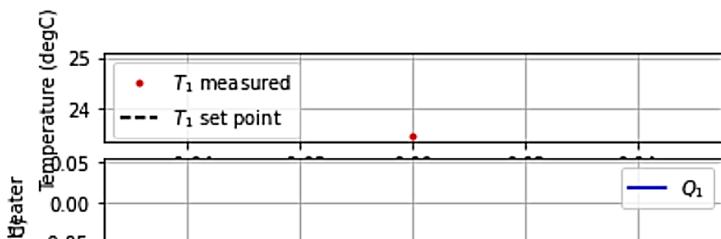
Dari program utama di atas, terdapat bagian **publish data**, yang nantinya harus disesuaikan dengan pengaturan di IoT MQTT Panel di Ponsel. (Cara penggunaan IoT MQTT Panel di Ponsel sudah dijelaskan pada Buku Pemrograman Internet of Things (IoT) dengan Arduino dan Python” Jilid 1)

Berikut ini, bagian yang dimaksud:

```
# Publish data to MQTT Broker
pub_sp = client.publish('SetPoint', Tsp1[i])
pub_pv1 = client.publish('Suhu1', T1[i])
pub_op = client.publish('Nilai_op', Q1[i])
```

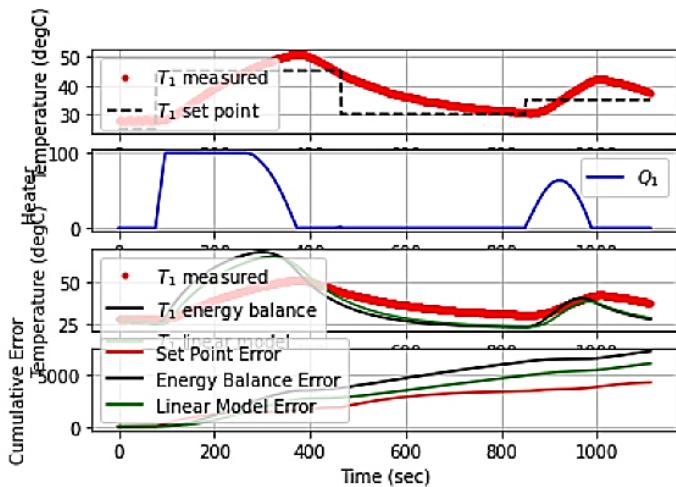
Setelah itu, silahkan dijalankan, perhatikan hasil pengendalian yang dihasilkan. Contoh hasil pengendaliannya, seperti terlihat pada Gambar 3.15 dan Gambar 3.16.

```
Opening connection
iTCLab connected via Arduino on port COM9
LED On
Running Main Loop. Ctrl-C to end.
Time      SP      PV      Q1      = P      + I      + D
0.0   25.00  23.46    0.00    0.00    0.00    0.00
<Figure size 720x504 with 0 Axes>
1.0   25.00  23.49    2.91    0.98    1.95   -0.01
```



Gambar 3. 15. Contoh hasil proses penalaan nilai K<sub>c</sub>, τ<sub>I</sub> dan τ<sub>D</sub> pada pengendali PID menggunakan Deep Learning pada Kit iTCLab

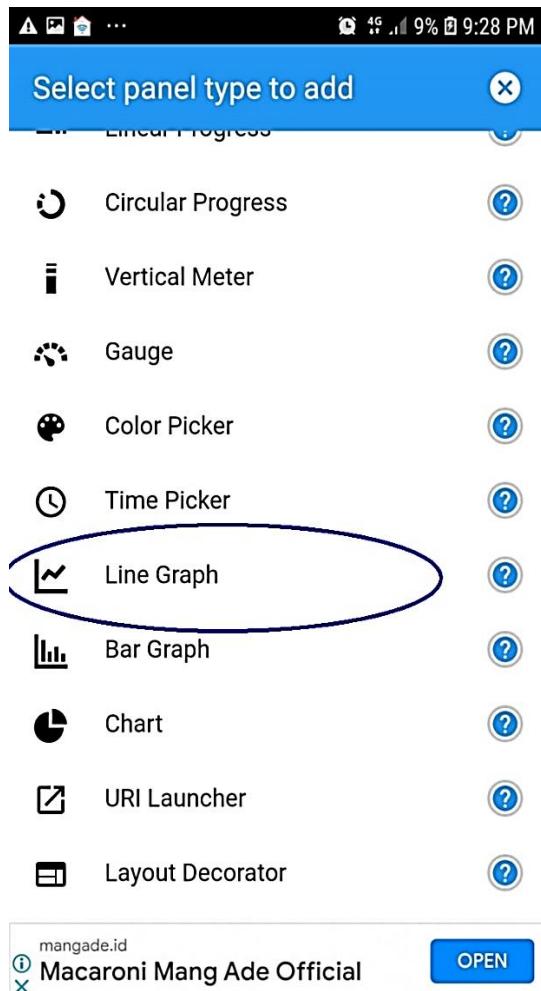
1113.3 35.00 37.59 0.00 -0.52 0.80 -0.00



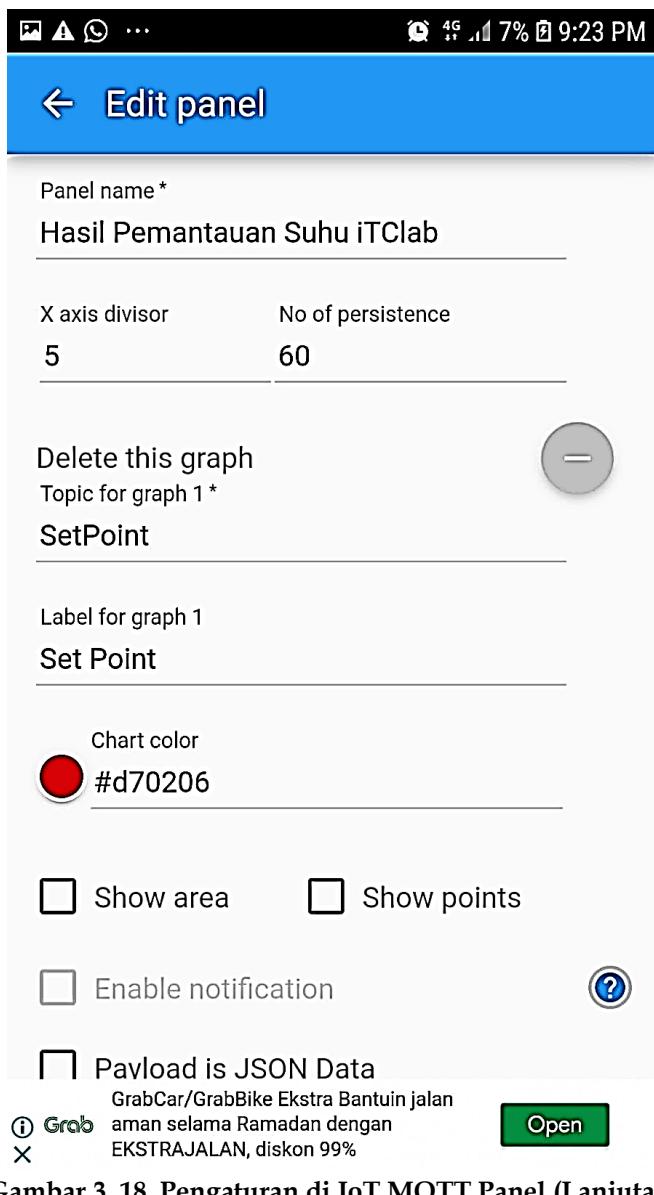
Gambar 3. 16. Contoh hasil pengendalian PID menggunakan Deep Learning pada Kit iTCLab

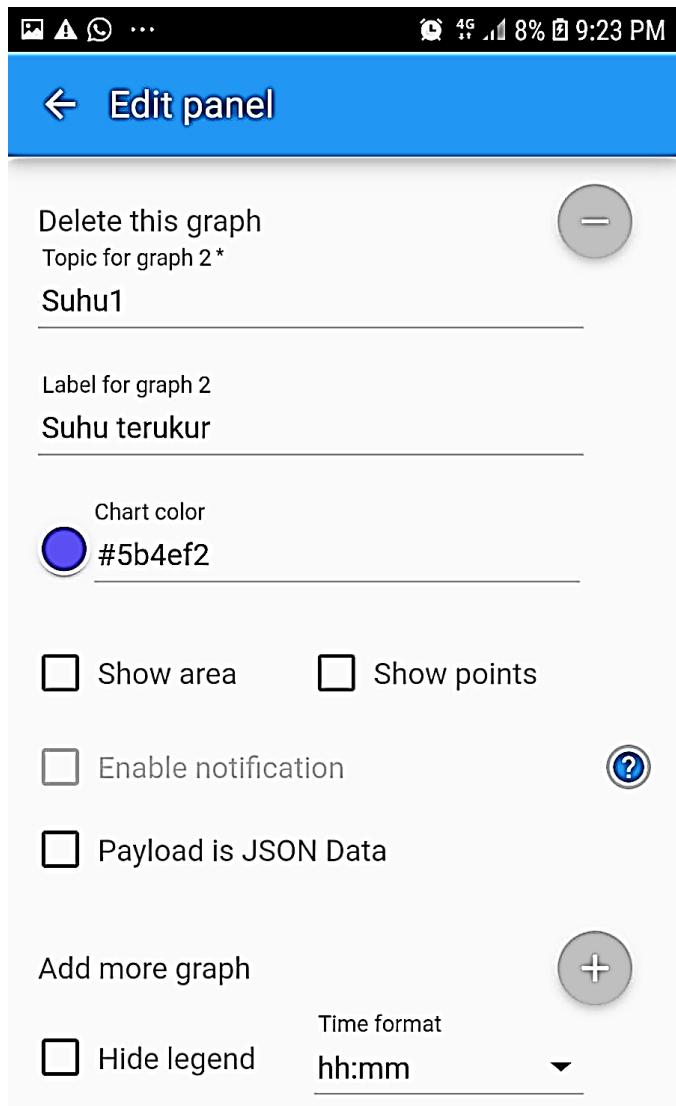
Contoh hasil pengendalian secara lengkap bisa dilihat pada:  
[https://github.com/bsrahmat/itclab-14/blob/main/Deep\\_PID\\_iTCLab\\_IoT.pdf](https://github.com/bsrahmat/itclab-14/blob/main/Deep_PID_iTCLab_IoT.pdf)

Selanjutnya, pengaturan pada Ponsel Android, menggunakan IoT MQTT Panel sebagai berikut:

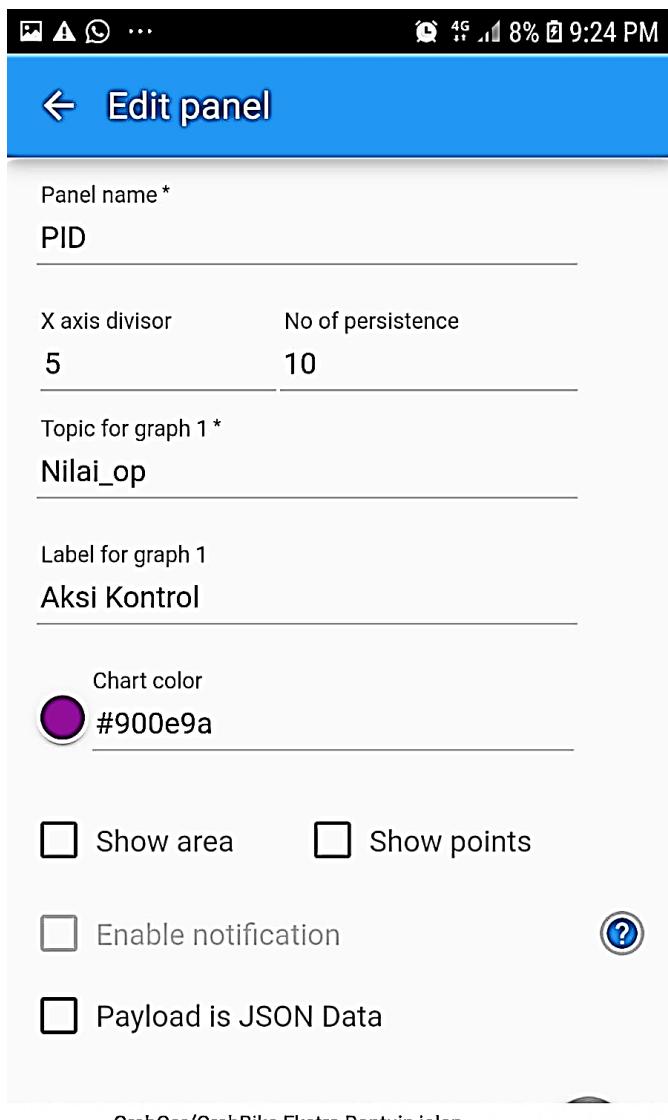


Gambar 3. 17. Pengaturan di IoT MQTT Panel



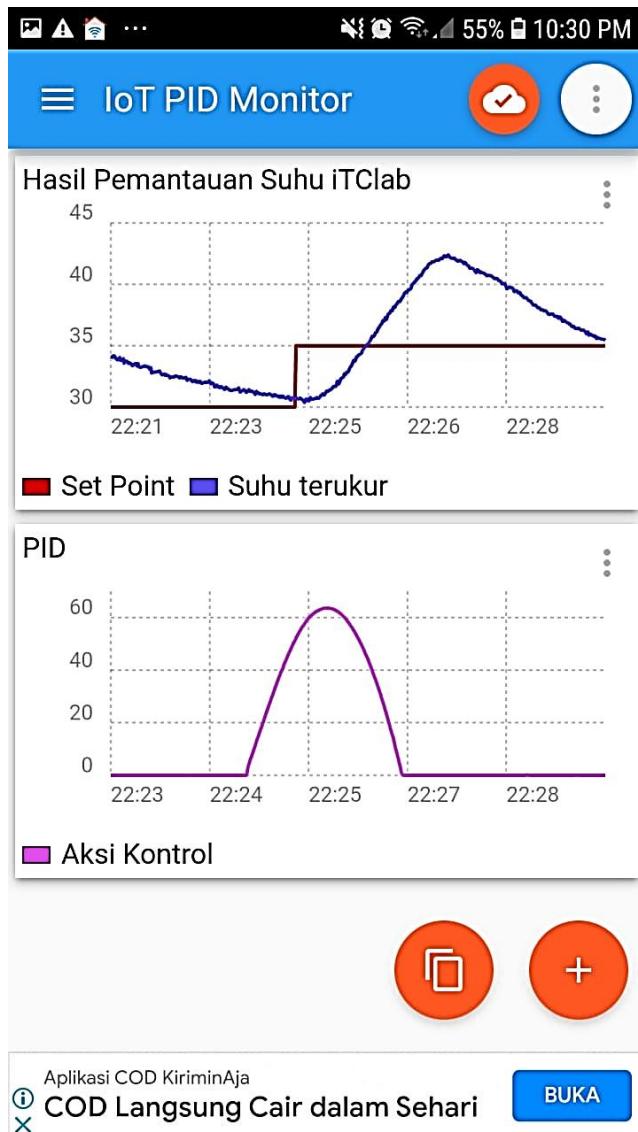


Gambar 3. 19. Pengaturan di IoT MQTT Panel (Lanjutan)



Gambar 3. 20. Pengaturan di IoT MQTT Panel (Lanjutan)

Jika berhasil terhubung ke Broker MQTT, maka contoh hasil pemantauannya seperti terlihat pada Gambar 3.21.



Gambar 3. 21. Contoh pemantauan hasil pengendalian PID menggunakan Deep Learning pada Kit iTCLab melalui IoT

Bagaimana Gaes? Apakah hasil pemantauan proses penalaan nilai  $K_c$ ,  $\tau_I$  dan  $\tau_D$  pada pengendali PID menggunakan Deep Learning pada Kit iTCLab melalui IoT, hasilnya kurang lebih sama seperti pada Gambar 3.21? Jika berhasil. Alhamdulillaah, anda memang layak dapat bintang. Dan berhasil menyelesaikan tantangan terakhir dari Buku Pemrograman IoT dengan Arduino dan Python Jilid 2 ini.

Apakah anda masih tertarik dengan tantangan berikutnya? Tantangan berikutnya, insyaAllaah kita akan melakukan eksperimen lanjutan Pemrograman Internet of Things (IoT) dengan Kit iTCLab. Dimana pemantauan dan pengendalian PID-iTCLab berbasis IoT dapat dijalankan melalui Blynk IoT Web Dashboard dan Blynk IoT Mobile Apps. InsyaAllaah kita akan bertemu kembali pada Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 3. Barakallaah!

# BAB

# 4

# PENUTUP

Telah diuraikan dalam Buku Referensi Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 2 ini. Dimulai dari Perkembangan Artificial Intelligence, Perkembangan Machine Learning dan Deep Learning, Pengembangan Framework Deep Learning, Penalaan Parameter PID dengan Sistem Cerdas, Pemrograman Deep Learning dengan Python, Pemrograman Gerbang XOR, Penalaan Parameter PID dengan Deep Learning, Pengendalian PID-iTCLab dengan Deep Learning, dan diakhiri dengan Pemantauan PID-iTCLab-Deep Learning Via IoT.

Semoga Buku ini bermanfaat bagi siapa saja yang ingin melakukan penelitian di bidang IoT dan AI. Selanjutnya, untuk eksperimen lanjutan Pemrograman Internet of Things (IoT) dengan Kit iTCLab. Dimana pemantauan dan pengendalian PID-iTCLab berbasis IoT dapat dijalankan melalui Blynk IoT Web Dashboard dan Blynk IoT Mobile Apps, insyaAllaah bisa dibaca pada Pemrograman Internet of Things (IoT) dengan Arduino dan Python, Jilid 3. Jika ingin berkomunikasi dengan penulis, bisa melalui alamat email: basukirahmat.if@upnjatim.ac.id atau muljono@dsn.dinus.ac.id. Salam sehat dan sukses selalu.

Juli 2024

Tim Penulis

## DAFTAR PUSTAKA

- [1] G. Rong, A. Mendez, E. B. Assi, B. Zhao, and M. Sawan, "Artificial Intelligence in Healthcare: Review and Prediction Case Studies," *Engineering*, 2020, doi: <https://doi.org/10.1016/j.eng.2019.08.015>.
- [2] W. G. de Sousa, E. R. P. de Melo, P. H. D. S. Bermejo, R. A. S. Farias, and A. O. Gomes, "How and where is artificial intelligence in the public sector going? A literature review and research agenda," *Gov Inf Q*, vol. 36, no. 4, p. 101392, 2019, doi: <https://doi.org/10.1016/j.giq.2019.07.004>.
- [3] V. V Prabhakar, C. S. Belarmin Xavier, and K. M. Abubeker, "A Review on Challenges and Solutions in the Implementation of Ai, IoT and Blockchain in Construction Industry," *Mater Today Proc*, 2023, doi: <https://doi.org/10.1016/j.matpr.2023.03.535>.
- [4] B. Bala and S. Behal, "AI techniques for IoT-based DDoS attack detection: Taxonomies, comprehensive review and research challenges," *Comput Sci Rev*, vol. 52, p. 100631, 2024, doi: <https://doi.org/10.1016/j.cosrev.2024.100631>.
- [5] S. R *et al.*, "A novel autonomous irrigation system for smart agriculture using AI and 6G enabled IoT network," *Micropoproess Microsyst*, vol. 101, p. 104905, 2023, doi: <https://doi.org/10.1016/j.micpro.2023.104905>.
- [6] R. Kumar Kasera, S. Gour, and T. Acharjee, "A comprehensive survey on IoT and AI based applications in different pre-harvest, during-harvest and post-harvest activities of smart agriculture," *Comput Electron Agric*, vol. 216, p. 108522, 2024, doi: <https://doi.org/10.1016/j.compag.2023.108522>.
- [7] V. K. Rathi *et al.*, "An edge AI-enabled IoT healthcare monitoring system for smart cities," *Computers & Electrical Engineering*, vol. 96, p. 107524, 2021, doi: <https://doi.org/10.1016/j.compeleceng.2021.107524>.

- [8] I. Keshta, "AI-driven IoT for smart health care: Security and privacy issues," *Inform Med Unlocked*, vol. 30, p. 100903, 2022, doi: <https://doi.org/10.1016/j.imu.2022.100903>.
- [9] E. Baccour, A. Erbad, A. Mohamed, M. Hamdi, and M. Guizani, "Reinforcement learning-based dynamic pruning for distributed inference via explainable AI in healthcare IoT systems," *Future Generation Computer Systems*, vol. 155, pp. 1-17, 2024, doi: <https://doi.org/10.1016/j.future.2024.01.021>.
- [10] A. Oppermann, "Artificial Intelligence vs. Machine Learning vs. Deep Learning," DeepLearning Academy, 2019.
- [11] T. Meng, X. Jing, Z. Yan, and W. Pedrycz, "A survey on machine learning for data fusion," *Information Fusion*, vol. 57, pp. 115-129, 2020, doi: <https://doi.org/10.1016/j.inffus.2019.12.001>.
- [12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527-1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.
- [13] R. Raina, A. Madhavan, and A. Y. Ng, "Large-Scale Deep Unsupervised Learning Using Graphics Processors," in *Proceedings of the 26th Annual International Conference on Machine Learning*, in ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 873-880. doi: 10.1145/1553374.1553486.
- [14] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory dsp]," *IEEE Signal Process Mag*, vol. 28, no. 1, pp. 145-154, 2010.
- [15] J. P. Adam Gibson, "Deep Learning," *Deep Learning*. O'Reilly Media, Inc., 2017.
- [16] O. Kharkovyna, "Top 10 Best Deep Learning Frameworks in 2019." Towards Data Science, 2019.
- [17] T. Bestwick and K. V Camarda, "Artificial Neural Network-Based Real-Time PID Controller Tuning," in *33rd European Symposium on Computer Aided Process Engineering*, vol. 52, A.

- C. Kokossis, M. C. Georgiadis, and E. Pistikopoulos, Eds., in Computer Aided Chemical Engineering, vol. 52. , Elsevier, 2023, pp. 1609–1614. doi: <https://doi.org/10.1016/B978-0-443-15274-0.50256-0>.
- [18] N. P. Lawrence, M. G. Forbes, P. D. Loewen, D. G. McClement, J. U. Backström, and R. B. Gopaluni, “Deep reinforcement learning with shallow controllers: An experimental application to PID tuning,” *Control Eng Pract*, vol. 121, p. 105046, 2022, doi: <https://doi.org/10.1016/j.conengprac.2021.105046>.
- [19] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, and B. M. Khammas, “Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems,” *Heliyon*, vol. 8, no. 5, p. e09399, 2022, doi: <https://doi.org/10.1016/j.heliyon.2022.e09399>.
- [20] B. Rahmat *et al.*, “ITCLab PID Control Tuning Using Deep Learning,” in *Proceeding - IEEE 9th Information Technology International Seminar, ITIS 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: [10.1109/ITIS59651.2023.10420130](https://doi.org/10.1109/ITIS59651.2023.10420130).
- [21] B. Rahmat, *Pemrograman Deep Learning dengan Python*. CV. Indomedia Pustaka, 2021.

## 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 Istrumentasi 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 dan Pemrosesan Bahasa Alami.