



Dr. Basuki Rahmat, S.Si, MT.
Dr. Ir. Minto Waluyo, M.M.
Ir. Tuhu Agung Rachmanto, M.T.

PENYIAPAN BROKER MQTT

**SEBAGAI PENUNJANG LAYANAN SISTEM KENDALI
BERBASIS INTERNET OF THINGS (IoT)**





Tentang Penulis



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



Dr. Ir. Minto Waluyo, M.M., adalah Dosen Teknik Industri, Fakultas Teknik, Universitas Pembangunan Nasional "Veteran" Jawa Timur. Dia menerima gelar Sarjana S-1 Teknik kimia Universitas Pembangunan Nasional "Veteran" Jawa Timur, Magister Manajemen UNKRIS Jakarta, Doktor Manajemen Unair Surabaya. Minat penelitiannya adalah di bidang Manajemen Perusahaan Industri.



Ir. Tuhu Agung Rachmanto, M.T., adalah Dosen Teknik Lingkungan, Fakultas Teknik, Universitas Pembangunan Nasional "Veteran" Jawa Timur. Dia menerima gelar Sarjana Teknik Kimia dari Fakultas Teknik Universitas Pembangunan Nasional "Veteran" Jawa Timur pada tahun 1986 dan menerima gelar Magister Teknik Kimia dari Institut Teknologi Sepuluh Nopember pada tahun 1999. Minat penelitiannya adalah di bidang Waste water treatment and Chemical Proses reaction Instrumentation.

PENYIAPAN BROKER MQTT SEBAGAI PENUNJANG LAYANAN SISTEM KENDALI BERBASIS INTERNET OF THINGS (IoT)

Dr. Basuki Rahmat, S.Si., M.T.

Dr. Ir. Minto Waluyo, M.M.

Ir. Tuhu Agung Rachmanto, M.T.



eureka
media aksara

PENERBIT CV. EUREKA MEDIA AKSARA

**PENYIAPAN BROKER MQTT
SEBAGAI PENUNJANG LAYANAN
SISTEM KENDALI BERBASIS
INTERNET OF THINGS (IoT)**

Penulis : Dr. Basuki Rahmat, S.Si., M.T.
Dr. Ir. Minto Waluyo, M.M.
Ir. Tuhu Agung Rachmanto, M.T.

Desain Sampul : Eri Setiawan

Tata Letak : Meilita Anggie Nurlatifah

ISBN : 978-623-487-302-3

Diterbitkan oleh : **EUREKA MEDIA AKSARA,
NOVEMBER 2022
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 : 2022

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.

KATA PENGANTAR

Saat ini telah terjadi perkembangan yang sangat pesat hasil kolaborasi konsep dan teknologi baru yang dibawa oleh Internet of Things (IoT) dan cloud computing di seluruh dunia. Banyak lapisan masyarakat secara bertahap bergerak menuju masyarakat modern yang cerdas. Teknologi ini secara bertahap merambah ke hampir semua bidang, dari teknologi yang paling sederhana hingga teknologi yang sangat kompleks.

Sistem dasar IoT terdiri dari 3 hal yaitu: perangkat keras/fisik (things), koneksi internet, dan *Cloud data center* sebagai tempat untuk menyimpan atau menjalankan aplikasinya. Pada *cloud data center* bisa diinstal *Broker Message Queuing Telemetry Transport* (MQTT) sebagai entitas perantara yang memungkinkan klien MQTT untuk berkomunikasi.

Meskipun masih jauh dari sempurna, semoga buku ini tetap bisa menjadi salah satu buku referensi bagi para peneliti yang ingin mengembangkan aplikasi kendali berbasis IoT.

Terima kasih

DAFTAR ISI

KATA PENGANTAR.....	iii
DAFTAR ISI.....	iv
BAB 1 PENDAHULUAN	1
A. Konsep dan Teknologi Internet of Things (IoT)	1
B. Apa itu IoT	2
C. Aplikasi IoT.....	3
D. Sistem IoT.....	5
E. Protokol MQTT.....	5
BAB 2 PENYIAPAN BROKER MQTT	8
A. Penyiapan Virtual Privat Server (VPS) dan Sistem Operasi	8
B. Instalasi dan Pengaturan Web Server.....	8
C. Instalasi dan Pengaturan Web Broker	10
D. Pengaturan NodeRed.....	26
BAB 3 PENGUJIAN SISTEM KENDALI SUHU.....	36
A. Penyiapan Sistem Kendali	36
B. Pengujian Sistem Kendali Suhu Berbasis IoT	78
BAB 4 PENGUJIAN SISTEM SMART FARMING.....	86
A. Sistem <i>Smart Farming</i> Berbasis IoT.....	86
B. Pengujian Sistem <i>Smart Farming</i> Berbasis IoT.....	90
BAB 5 PENUTUP	92
DAFTAR PUSTAKA.....	93
TENTANG PENULIS.....	94

UCAPAN TERIMA KASIH

Terimakasih kepada Direktorat Sumber Daya Direktorat Jenderal Pendidikan Tinggi Kementerian Pendidikan, Kebudayaan, Riset dan Teknologi yang telah memberikan dana penelitian sesuai dengan Kontrak Penelitian Tahun Jamak Program Penelitian Terapan Unggulan Perguruan Tinggi Nomor: 12 / UN63.8 / LT - Kontrak / VII / 2021, tanggal 15 Juli 2021, dan Nomor: 09/UN63.8/LT-Kontrak/III/2022, tanggal 16 Maret 2022.



**PENYIAPAN BROKER MQTT
SEBAGAI PENUNJANG LAYANAN
SISTEM KENDALI BERBASIS
INTERNET OF THINGS (IoT)**

Dr. Basuki Rahmat, S.Si., M.T.

Dr. Ir. Minto Waluyo, M.M.

Ir. Tuhu Agung Rachmanto, M.T.



BAB 1

PENDAHULUAN

A. Konsep dan Teknologi Internet of Things (IoT)

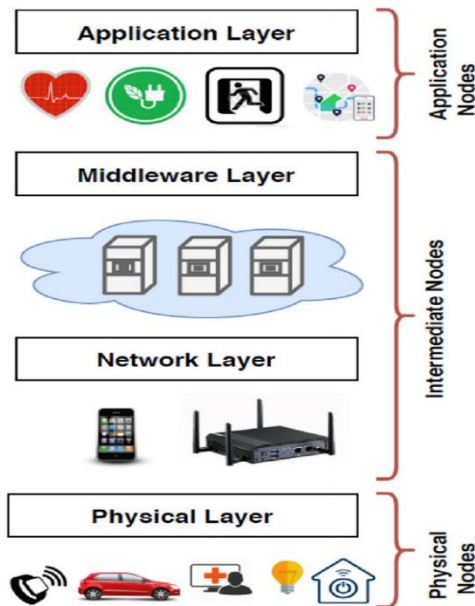
Konsep dan teknologi Internet of Things (IoT) sudah menjadi kebutuhan yang sangat penting di era industri 4.0 dan Society 5.0 saat ini. Dimana sistem dasar IoT terdiri dari 3 hal yaitu: perangkat keras / fisik (things), koneksi internet, dan *cloud data center* sebagai tempat untuk menyimpan atau menjalankan aplikasinya. Pada *cloud data center* bisa diinstal *Broker Message Queuing Telemetry Transport* (MQTT) sebagai entitas perantara yang memungkinkan klien MQTT untuk berkomunikasi. Namun saat ini yang menyediakan layanan Cloud IoT hanya didominasi oleh pemain dari luar negeri. Barangkali bisa disebutkan yang cukup populer saat ini antara lain thingspeak.com dan cloudmqtt.com.

Banyak pemain IoT saat ini yang tergantung kepada kedua layanan Cloud IoT ini. [Thingspeak.com](https://thingspeak.com) masih gratis namun hanya terbatas untuk layanan pemantauan saja. Untuk layanan pengendalian belum kami temukan cara penggunaannya. Sedangkan cloudmqtt.com, bisa digunakan untuk pemantauan dan pengendalian jarak jauh melalui koneksi internet. Hanya saja, saat ini, layanan gratis dari cloud ini sudah tidak disediakan lagi. Sehingga banyak pemain dan praktisi IoT terutama dari kalangan dosen dan peneliti atau pun mahasiswa yang kebingungan. Penelitian ini dimaksudkan sebagai solusi dari kebingungan atau kesulitan ini. Penelitian ini mengusulkan pembuatan Cloud IoT yang menyediakan layanan Broker MQTT

yang bisa menyediakan layanan gratis untuk para pengguna IoT Indonesia maupun dunia.

B. Apa itu IoT

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



Gambar 1.1. Arsitektur IoT [2]

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

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

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

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

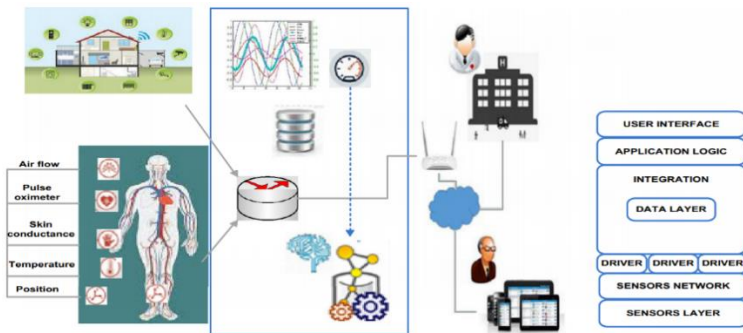
C. Aplikasi IoT

Banyak sekali contoh penerapan teknologi IoT, beberapa contohnya:

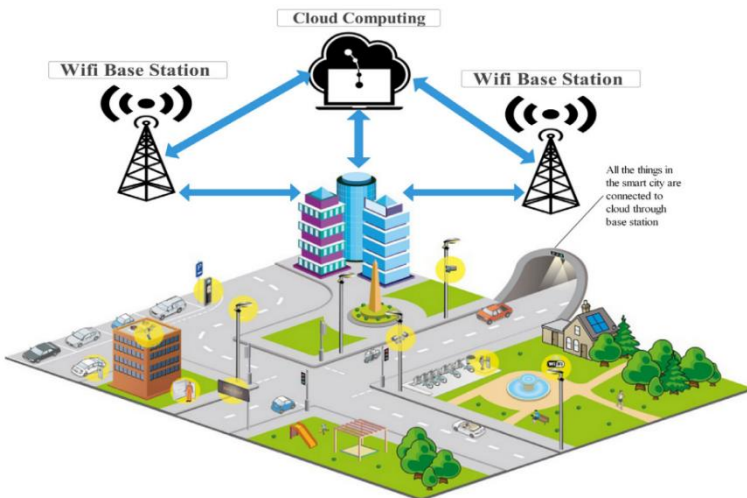
1. *Smart Home* (sistem keamanan rumah berbasis internet, dapat mengetahui keadaan rumah serta mengontrol peralatan rumah tangga melalui jaringan internet).
2. *Smart Farming* (sistem pertanian cerdas berbasis internet, untuk pemantauan dan pengendalian kualitas air dan tanah

- pertanian serta pertumbuhan tanaman melalui jaringan internet).
3. *Internet industry* (pemantauan dan pengendalian peralatan serta proses di industri).
 4. Kesehatan (pemantauan kondisi kesehatan seseorang).
 5. Transportasi (manajemen dan informasi lalu lintas).
 6. Dan lain-lain.

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



Gambar 1.2. Contoh arsitektur berbasis IoT untuk bidang kesehatan [3]



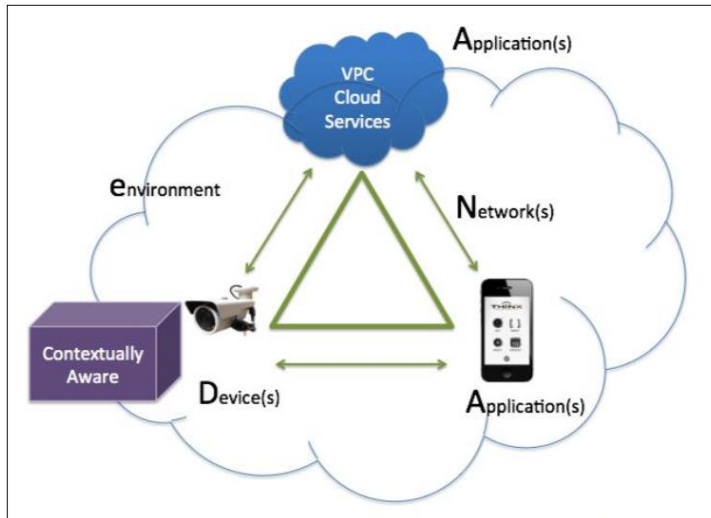
Gambar 1.3. Contoh penerapan IoT pada Smart City [4]

D. Sistem IoT

Sistem dasar dari IoT terdiri dari 3 hal, yaitu:

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

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



Gambar 1.4. Sistem IoT [5]

E. Protokol MQTT

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

Beberapa istilah penting terkait MQTT, antara lain:

Broker - Broker adalah server yang mendistribusikan informasi kepada klien yang terhubung ke server.

Client (Klien) - Perangkat yang terhubung ke broker untuk mengirim atau menerima informasi.

Topic (Topik) - Nama tentang pesan itu. Klien mempublikasikan, berlangganan, atau melakukan keduanya untuk suatu topik.

Publish (Terbitkan) - Klien yang mengirim informasi ke broker untuk dibagikan kepada klien yang tertarik berdasarkan nama topik.

Subscribe (Berlangganan) - Klien memberi tahu broker topik apa yang mereka minati. Ketika klien berlangganan suatu topik, pesan apa pun yang diterbitkan ke broker didistribusikan ke pelanggan topik itu. Klien juga dapat berhenti berlangganan untuk berhenti menerima pesan dari broker tentang topik itu.

QoS (*Quality of Service*) - Kualitas Layanan. Setiap koneksi dapat menentukan kualitas layanan ke broker dengan nilai integer mulai dari 0-2. QoS tidak mempengaruhi penanganan transmisi data TCP, hanya antara klien MQTT.

1. 0 menentukan paling banyak sekali, atau sekali dan hanya sekali tanpa memerlukan pemberitahuan pengiriman. Ini sering disebut sebagai *fire* (api) dan *forget* (lupa).
2. 1 menentukan setidaknya satu kali. Pesan itu dikirim beberapa kali sampai sebuah pengakuan diterima, dikenal sebaliknya sebagai pengiriman yang diakui.
3. 2 menentukan tepat sekali. Klien pengirim dan penerima menggunakan jabat tangan dua tingkat untuk memastikan hanya satu salinan pesan yang diterima, yang dikenal sebagai pengiriman yang terjamin.

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

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

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

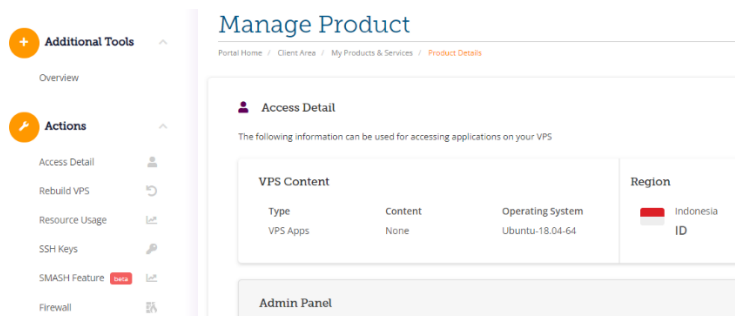
BAB

2

PENYIAPAN BROKER MQTT

A. Penyiapan Virtual Privat Server (VPS) dan Sistem Operasi

Langkah pertama untuk pembuatan Broker MQTT adalah penyiapan Virtual Privat Server (VPS). Setup dan Konfigurasi VPS dengan cara pemanfaatan fitur XCube pada VPS X. XCube adalah kumpulan template VPS yang siap dipilih dan diinstall sesuai kebutuhan. Tak perlu install satu-persatu, XCube diciptakan untuk mempercepat proses setup. Termasuk untuk keperluan proses instalasi sistem operasinya. Kami menggunakan sistem operasi Ubuntu-18.04-64, seperti terlihat pada Gambar 2.1.



Gambar 2.1. Sistem operasi VPS

B. Instalasi dan Pengaturan Web Server

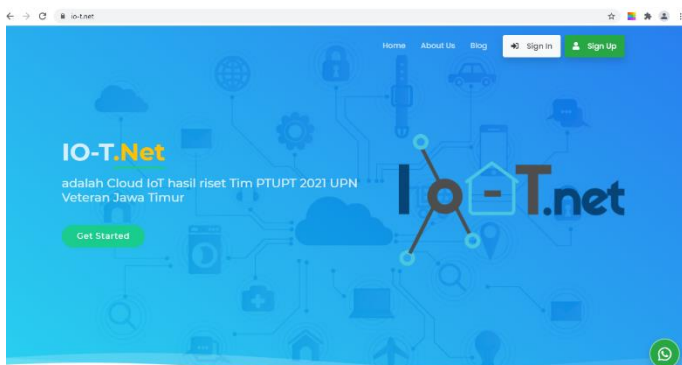
Kebutuhan selanjutnya selain sistem operasi yang harus dipersiapkan agar Broker MQTT bisa berjalan di browser, termasuk nantinya untuk keperluan penyiapan aplikasi web dan

database, dan kemudahan pengelolaan administratornya, maka perlu diinstal dan diatur (di-setting) antara lain: Lamp, MySQL, Webmin, Phpmyadmin, pemindahan ke htdocs, dan penggunaan SSL agar lebih aman. Tahap-tahap ini relatif mudah, dan semua sudah berjalan dengan baik.

Alamat Broker MQTT yang kami kembangkan, yaitu i-ot.net (untuk versi stabil), dan io-t.net untuk keperluan *trial* dan eksperimen serta proses-proses pembelajaran yang kami lakukan. Saat ini kedua alamat ini sudah bisa diakses di alamat <https://i-ot.net> dan <https://io-t.net>. Berikut ini kedua tampilan dari kedua alamat web untuk keperluan layanan Broker MQTT yang kami kembangkan.



Gambar 2.2. Tampilan alamat Broker MQTT versi stabil



Gambar 2.3. Tampilan alamat Broker MQTT versi trial dan eksperimen

C. Instalasi dan Pengaturan Web Broker

Kebutuhan selanjutnya adalah penyiapan Web Broker dengan protokol MQTT. Untuk keperluan ini, kami gunakan Mosquitto. Tahap ini menurut kami merupakan tahap yang paling sulit dan menantang. Berikut ini penjelasan bagaimana menyiapkan IoT server dengan menggunakan stack Linux, yang terdiri dari Mosquitto, pengecekan database MySQL, NodeRED, dan Grafana.

Install Tool & Library

Masuk dulu ke root@io-t.net menggunakan putty.

Update & upgrade ubuntu,

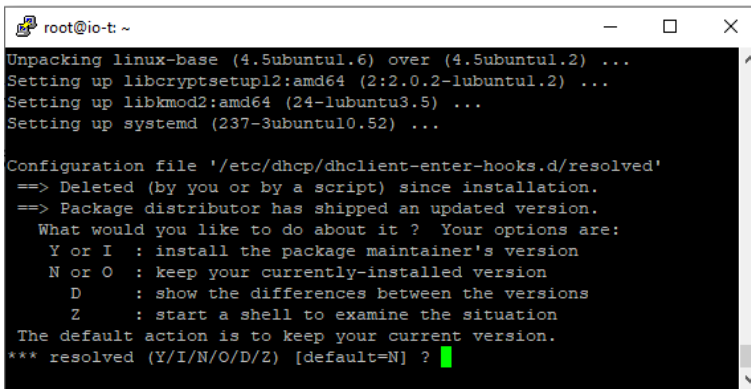
Perintahnya:

```
$ sudo apt-get update && sudo apt-get upgrade -y
```

Silahkan ditunggu, sampai proses update dan upgrade selesai. Jika ada pertanyaan:

resolved (Y/I/N/O/D/Z) [default=N] ?

Seperti terlihat pada Gambar 2.4.



```
root@io-t: ~
Unpacking linux-base (4.5ubuntu1.6) over (4.5ubuntu1.2) ...
Setting up libcryptsetup12:amd64 (2:2.0.2-1ubuntu1.2) ...
Setting up libkmod2:amd64 (24-1ubuntu3.5) ...
Setting up systemd (237-3ubuntu10.52) ...

Configuration file '/etc/dhcp/dhclient-enter-hooks.d/resolved'
==> Deleted (by you or by a script) since installation.
==> Package distributor has shipped an updated version.
What would you like to do about it ? Your options are:
  Y or I : install the package maintainer's version
  N or O : keep your currently-installed version
  D      : show the differences between the versions
  Z      : start a shell to examine the situation
The default action is to keep your current version.
*** resolved (Y/I/N/O/D/Z) [default=N] ?
```

Gambar 2.4. Pertanyaan resolved

Pilihannya adalah:

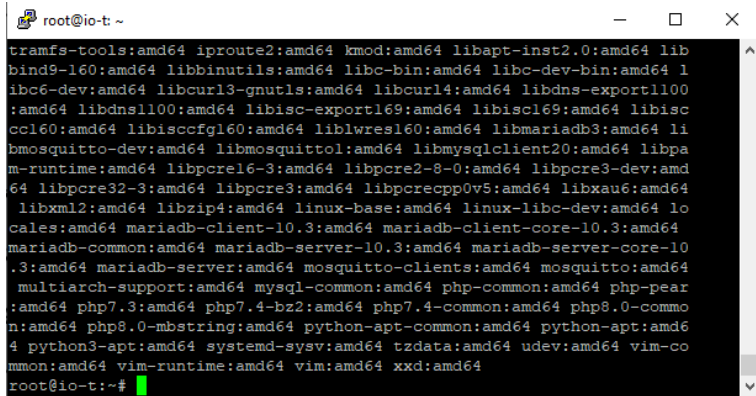
Y or I : instal versi pengelola paket

N or O : pertahankan versi yang saat ini anda instal

D : tunjukkan perbedaan antar versi

Z : mulai *shell* untuk memeriksa *sit*.

Silahkan dipilih N (default-nya). Silahkan ditunggu sampai proses selesai, seperti terlihat pada Gambar 2.5.



```
root@io-t: ~  
tramfs-tools:amd64 iproute2:amd64 kmod:amd64 libapt-inst2.0:amd64 lib  
bind9-160:amd64 libbinutils:amd64 libc-bin:amd64 libc-dev-bin:amd64 l  
ibc6-dev:amd64 libcurl3-gnutls:amd64 libcurl4:amd64 libdns-export1100  
:amd64 libdns1100:amd64 libisc-export169:amd64 libisc169:amd64 libisc  
cc160:amd64 libisccfg160:amd64 liblwres160:amd64 libmariadb3:amd64 li  
bmosquitto-dev:amd64 libmosquitto:amd64 libmysqlclient20:amd64 libpa  
m-runtime:amd64 libpcre16-3:amd64 libpcre2-8-0:amd64 libpcre3-dev:amd  
64 libpcre32-3:amd64 libpcre3:amd64 libpcrecpp0v5:amd64 libxau6:amd64  
libxml2:amd64 libzip4:amd64 linux-base:amd64 linux-libc-dev:amd64 lo  
cales:amd64 mariadb-client-10.3:amd64 mariadb-client-core-10.3:amd64  
mariadb-common:amd64 mariadb-server-10.3:amd64 mariadb-server-core-10  
.3:amd64 mariadb-server:amd64 mosquitto-clients:amd64 mosquitto:amd64  
multiarch-support:amd64 mysql-common:amd64 php-common:amd64 php-pear  
:amd64 php7.3:amd64 php7.4-bz2:amd64 php7.4-common:amd64 php8.0-commo  
n:amd64 php8.0-mbstring:amd64 python-apt-common:amd64 python-apt:amd6  
4 python3-apt:amd64 systemd-sysv:amd64 tzdata:amd64 udev:amd64 vim-co  
mmon:amd64 vim-runtime:amd64 vim:amd64 xxd:amd64  
root@io-t:~#
```

Gambar 2.5. Proses update dan upgrade selesai

Install Mosquitto

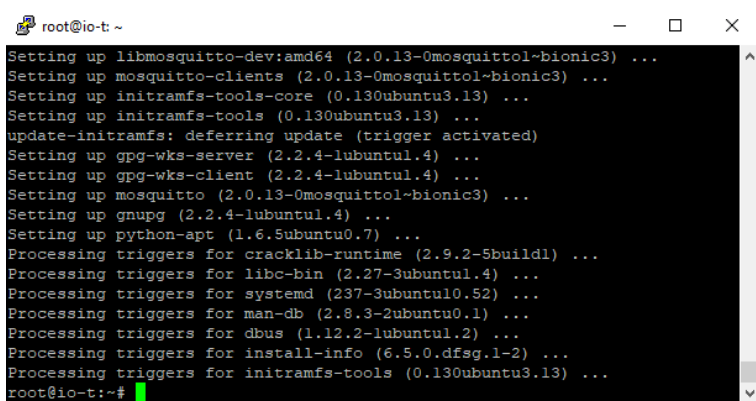
Install Mosquitto MQTT Broker, perintahnya:

```
$ sudo apt-get install mosquitto mosquitto-clients
```

Sampai ada pertanyaan

Do you want to continue? [Y/n]

Silahkan tekan Y untuk melanjutkan. Tunggu sampai proses instalasi Mosquitto selesai, seperti terlihat pada Gambar 2.6.



```
root@io-t: ~  
Setting up libmosquitto-dev:amd64 (2.0.13-0mosquitto~bionic3) ...  
Setting up mosquitto-clients (2.0.13-0mosquitto~bionic3) ...  
Setting up initramfs-tools-core (0.130ubuntu3.13) ...  
Setting up initramfs-tools (0.130ubuntu3.13) ...  
update-initramfs: deferring update (trigger activated)  
Setting up gpg-wks-server (2.2.4-lubuntu1.4) ...  
Setting up gpg-wks-client (2.2.4-lubuntu1.4) ...  
Setting up mosquitto (2.0.13-0mosquitto~bionic3) ...  
Setting up gnupg (2.2.4-lubuntu1.4) ...  
Setting up python-apt (1.6.5ubuntu0.7) ...  
Processing triggers for cracklib-runtime (2.9.2-5build1) ...  
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...  
Processing triggers for systemd (237-3ubuntu10.52) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for dbus (1.12.2-lubuntu1.2) ...  
Processing triggers for install-info (6.5.0.dfsg.1-2) ...  
Processing triggers for initramfs-tools (0.130ubuntu3.13) ...  
root@io-t:~#
```

Gambar 2.6. Proses instalasi Mosquitto selesai

Test Mosquitto

Subscribe dengan topic test, perintahnya:

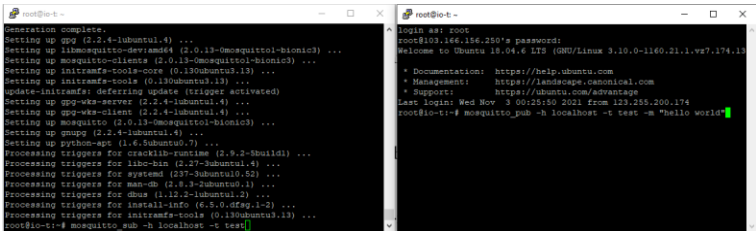
```
$ mosquitto_sub -h localhost -t test
```

Buka terminal baru untuk publish message Hello World!

pada topic test, perintahnya:

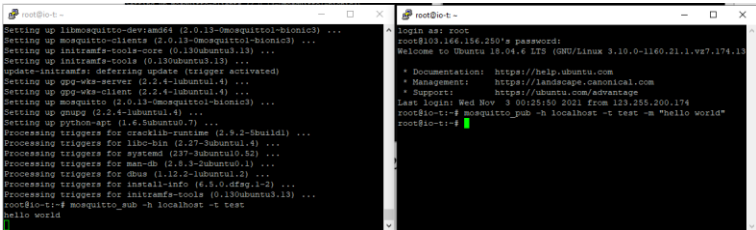
```
$ mosquitto_pub -h localhost -t test -m "hello world"
```

Seperti terlihat pada Gambar 2.7.



Gambar 2.7. Test Mosquitto

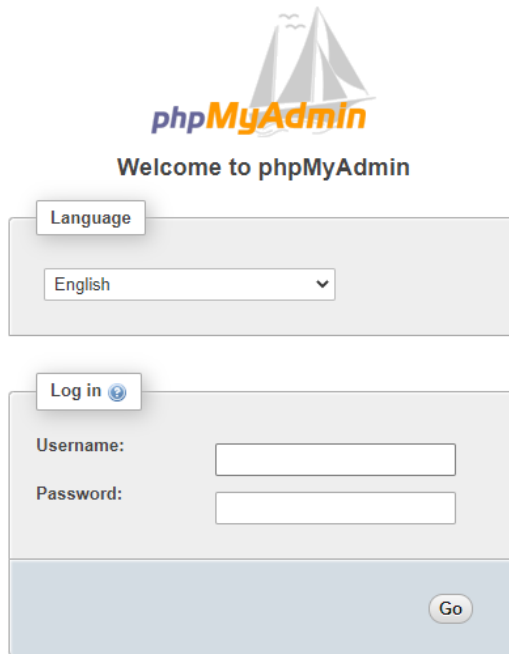
Hasilnya seperti terlihat pada Gambar 2.8.



Gambar 2.8. Hasil test Mosquitto

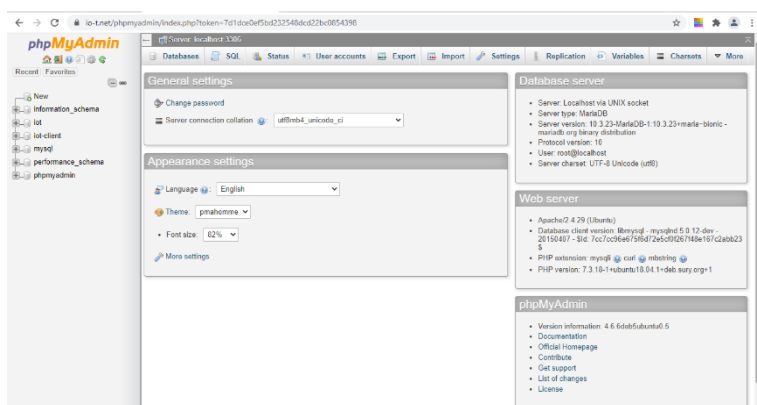
Cek database MySQL

Selanjutnya silahkan dilakukan pengecekan database mysql yang sudah diinstal sebelumnya melalui phpmyadmin pada alamat <https://io-t.net/phpmyadmin> seperti terlihat pada Gambar 2.9.



Gambar 2.9. Pengecekan database MySql melalui phpmyadmin

Silahkan dimasukkan username dan password saat proses instalasi, jika sudah berhasil masuk, maka tampilannya seperti terlihat pada Gambar 2.10.



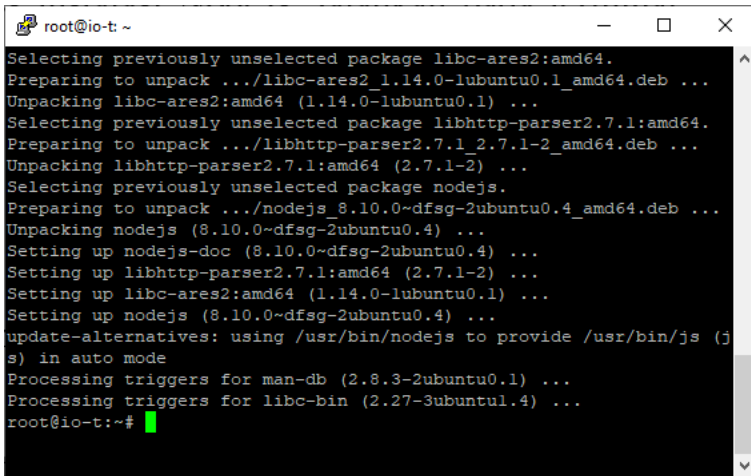
Gambar 2.10. Pengaturan database melalui phpmyadmin

Install NodeJs

Selanjutnya silahkan dilakukan proses instalasi NodeJs, jalankan pada terminal, perintahnya:

```
$ sudo apt install nodejs
```

Sampai ada pertanyaan: Do you want to continue? [Y/n], silahkan tekan Y untuk melanjutkan. Tunggu sampai proses instalasi NodeJs selesai. Jika sukses, maka tampilannya seperti terlihat pada Gambar 2.11.



```
root@io-t: ~  
Selecting previously unselected package libc-ares2:amd64.  
Preparing to unpack .../libc-ares2_1.14.0-1ubuntu0.1_amd64.deb ...  
Unpacking libc-ares2:amd64 (1.14.0-1ubuntu0.1) ...  
Selecting previously unselected package libhttp-parser2.7.1:amd64.  
Preparing to unpack .../libhttp-parser2.7.1_2.7.1-2_amd64.deb ...  
Unpacking libhttp-parser2.7.1:amd64 (2.7.1-2) ...  
Selecting previously unselected package nodejs.  
Preparing to unpack .../nodejs_8.10.0~dfsg-2ubuntu0.4_amd64.deb ...  
Unpacking nodejs (8.10.0~dfsg-2ubuntu0.4) ...  
Setting up nodejs-doc (8.10.0~dfsg-2ubuntu0.4) ...  
Setting up libhttp-parser2.7.1:amd64 (2.7.1-2) ...  
Setting up libc-ares2:amd64 (1.14.0-1ubuntu0.1) ...  
Setting up nodejs (8.10.0~dfsg-2ubuntu0.4) ...  
update-alternatives: using /usr/bin/nodejs to provide /usr/bin/js (js)  
in auto mode  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...  
root@io-t:~#
```

Gambar 2. 11. Proses instalasi NodeJs sukses

Install NPM

Selanjutnya silahkan melakukan proses instalasi NPM, jalankan pada terminal, perintahnya:

```
$ sudo apt install npm
```

Sampai ada pertanyaan: Do you want to continue? [Y/n], silahkan tekan Y untuk melanjutkan. Tunggu sampai proses instalasi NPM selesai. Jika sukses, maka tampilannya seperti terlihat pada Gambar 2.12.

```
root@io-t: ~  
Setting up gcc-7 (7.5.0-3ubuntu1~18.04) ...  
Setting up g++-7 (7.5.0-3ubuntu1~18.04) ...  
Setting up gcc (4:7.4.0-1ubuntu2.3) ...  
Setting up node-glob (7.1.2-4) ...  
Setting up g++ (4:7.4.0-1ubuntu2.3) ...  
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++  
) in auto mode  
Setting up build-essential (12.4ubuntu1) ...  
Setting up node-rimraf (2.6.2-1) ...  
Setting up node-read-package-json (1.2.4-1) ...  
Setting up node-fstream (1.0.10-1ubuntu0.18.04.1) ...  
Setting up node-fstream-ignore (0.0.6-2) ...  
Setting up node-tar (2.2.1-1) ...  
Setting up node-gyp (3.6.2-1ubuntu1) ...  
Setting up npm (3.5.2-0ubuntu4) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...  
root@io-t:~#
```

Gambar 2.12. Proses instalasi NPM sukses

Check NodeJs & NPM

Selanjutnya silahkan dilakukan pengecekan NodeJs dan NPM yang sudah diinstal. Perintahnya sbb:

\$ nodejs -v

\$ npm -v

Contoh hasil pengecekan seperti diperlihatkan pada Gambar 2.13.

```
root@io-t: ~  
) in auto mode  
Setting up build-essential (12.4ubuntu1) ...  
Setting up node-rimraf (2.6.2-1) ...  
Setting up node-read-package-json (1.2.4-1) ...  
Setting up node-fstream (1.0.10-1ubuntu0.18.04.1) ...  
Setting up node-fstream-ignore (0.0.6-2) ...  
Setting up node-tar (2.2.1-1) ...  
Setting up node-gyp (3.6.2-1ubuntu1) ...  
Setting up npm (3.5.2-0ubuntu4) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...  
root@io-t:~# nodejs -v  
v8.10.0  
root@io-t:~# npm -v  
3.5.2  
root@io-t:~#
```

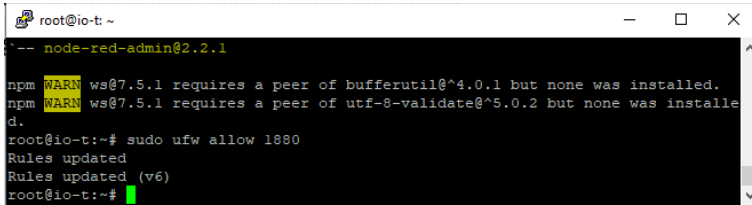
Gambar 2.13. Hasil pengecekan NodeJs dan NPM

Install NodeRed

Selanjutnya silahkan dilakukan proses instalasi NodeRed via NPM. Sebelumnya, NodeRed menggunakan port:1880 sebagai port *default*. Jadi, izinkan port untuk digunakan oleh layanan dengan menggunakan perintah berikut:

```
$ sudo ufw allow 1880
```

Hasilnya seperti terlihat pada Gambar 2.14.



```
root@io-t: ~  
-- node-red-admin@2.2.1  
npm WARN ws@7.5.1 requires a peer of bufferutil@^4.0.1 but none was installed.  
npm WARN ws@7.5.1 requires a peer of utf-8-validate@^5.0.2 but none was installed.  
root@io-t:~# sudo ufw allow 1880  
Rules updated  
Rules updated (v6)  
root@io-t:~#
```

Gambar 2.14. Izinkan port 1880

Selanjutnya silahkan dilakukan proses instalasi NodeRed via NPM, perintahnya sebagai berikut:

```
$ sudo npm install -g --unsafe-perm node-red node-red-admin
```

Keterangan:

- g : Ini berarti bahwa paket tersebut diinstal secara global dan akan tersedia untuk Node.js
- unsafe-perm : Ini mengesampingkan kesalahan selama waktu instalasi. Pada dasarnya peralihan ID pengguna/grup ditekan ketika skrip paket dijalankan, pengguna non-root akan dapat menginstal paket.
- Node-red-admin : Ini akan menginstal modul admin yang memberi kita beberapa alat administrasi tambahan untuk Node-RED.

Maka akan tampil seperti diperlihatkan pada Gambar 2.15. tunggu sampai proses selesai.


```
root@io-t: ~
0.0", "npm": "3.5.2"})
WARN engine node-red@2.1.3: wanted: {"node": ">=12"} (current: {"node": "8.10.0", "
loadDep:bcrypt -> get - |#####-----|
WARN engine fs-extra@10.0.0: wanted: {"node": ">=12"} (current: {"node": "8.10.0",
loadDep:bcrypt -> 304 / |#####-----|
WARN engine semver@7.3.5: wanted: {"node": ">=10"} (current: {"node": "8.10.0", "np
m": "3.5.2"})
WARN engine bcrypt@5.0.1: wanted: {"node": ">= 10.0.0"} (current: {"node": "8.10.0
loadDep:universalify -> a \ |#####-----|
WARN engine universalify@2.0.0: wanted: {"node": ">= 10.0.0"} (current: {"node": "
loadDep:tar -> 304 | |#####-----|
WARN engine tar@6.1.11: wanted: {"node": ">= 10"} (current: {"node": "8.10.0", "npm
loadDep:abbrev - |#####-----|
WARN engine gauge@3.0.1: wanted: {"node": ">=10"} (current: {"node": "8.10.0", "npm
": "3.5.2"})
WARN engine are-we-there-yet@2.0.0: wanted: {"node": ">=10"} (current: {"node": "8
loadDep:util-deprecate -> \ |#####-----|
```

Gambar 2.15. Proses instalasi NodeRed

Jika instalasi NodeRed gagal, silahkan diperhatikan pesan *Warning* yang muncul. Contohnya jika misalnya pesan peringatan yang muncul seperti berikut ini:

```
WARN engine node-red@2.1.3: wanted: {"node": ">=12"}
(current: {"node": "8.10.0", "npm": "3.5.2"})
```

Itu karena yang dibutuhkan saat instalasi, adalah Nodejs versi 12 keatas. Sementara yang ada saat ini adalah versi 8.10.0. Maka perlu dilakukan update Nodejs terlebih dahulu.

Cara update Nodejs menggunakan NPM (*Node Package Manager*), sebagai berikut:

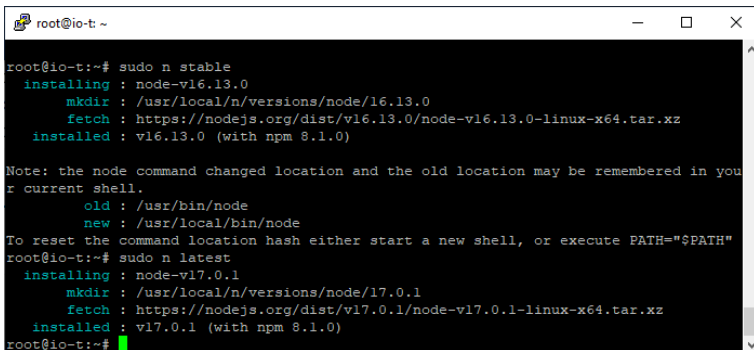
Dengan menambahkan modul *n*, dapat dikelola versi Node.js secara interaktif.

1. Pertama, bersihkan cache npm:
\$ npm cache bersih -f
2. Instal *n*, manajer versi Node:
\$ npm install -g n
3. Dengan modul *n* terpasang, dapat digunakan untuk instal versi stabil terbaru:
\$ sudo n stable

Instal rilis terbaru:

```
$ sudo n latest
```

Hasilnya seperti terlihat pada Gambar 2.16.



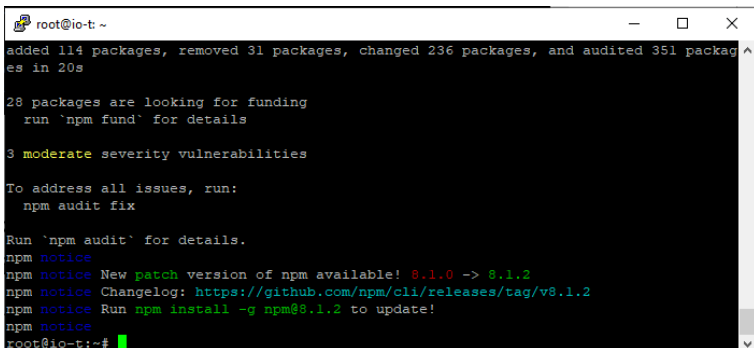
```
root@io-ti:~# sudo n stable
installing : node-v16.13.0
mkdir : /usr/local/n/versions/node/16.13.0
fetch : https://nodejs.org/dist/v16.13.0/node-v16.13.0-linux-x64.tar.xz
installed : v16.13.0 (with npm 8.1.0)

Note: the node command changed location and the old location may be remembered in your current shell.
old : /usr/bin/node
new : /usr/local/bin/node
To reset the command location hash either start a new shell, or execute PATH="$PATH"
root@io-ti:~# sudo n latest
installing : node-v17.0.1
mkdir : /usr/local/n/versions/node/17.0.1
fetch : https://nodejs.org/dist/v17.0.1/node-v17.0.1-linux-x64.tar.xz
installed : v17.0.1 (with npm 8.1.0)
root@io-ti:~#
```

Gambar 2.16. Node-v17.01

Selanjutnya silahkan dilakukan proses instalasi ulang NodeRed via NPM, perintahnya sebagai berikut:
\$ sudo npm install -g --unsafe-perm node-red node-red-admin

Seharusnya berbeda dengan hasil sebelumnya, setelah proses instalasi selesai, silahkan diperhatikan apakah hasilnya seperti terlihat pada Gambar 2.17.



```
added 114 packages, removed 31 packages, changed 236 packages, and audited 351 packages in 20s

28 packages are looking for funding
  run `npm fund` for details

3 moderate severity vulnerabilities

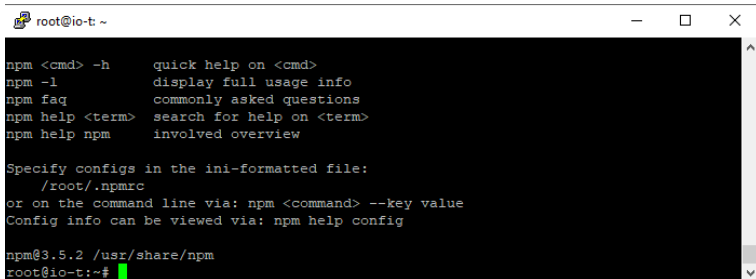
To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New patch version of npm available! 8.1.0 -> 8.1.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.1.2
npm notice Run npm install -g npm@8.1.2 to update!
npm notice
root@io-ti:~#
```

Gambar 2.17. Setelah proses instalasi NodeRed

Silahkan diperhatikan pesan yang muncul, silahkan dipenuhi:
\$ npm audit fix

Hasilnya seharusnya seperti terlihat pada Gambar 2.18.

A terminal window titled 'root@io-t: ~' with standard window controls. It displays the output of the 'npm help' command. The output lists various help options: 'npm <cmd> -h' for quick help, 'npm -l' for full usage info, 'npm faq' for common questions, 'npm help <term>' for searching help, and 'npm help npm' for an overview. It also explains how to specify configurations in an ini-formatted file or via command-line flags, and provides the path to the npmrc file. The prompt at the bottom is 'root@io-t:~#'.

```
root@io-t: ~
npm <cmd> -h      quick help on <cmd>
npm -l            display full usage info
npm faq           commonly asked questions
npm help <term>   search for help on <term>
npm help npm      involved overview

Specify configs in the ini-formatted file:
  /root/.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@3.5.2 /usr/share/npm
root@io-t:~#
```

Gambar 2.18. Hasil perbaikan audit

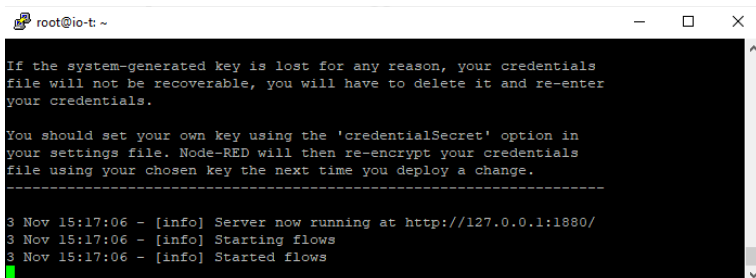
Test NodeRed

Uji instalasi NodeRed.

Jadi sekarang kita sudah siap dan bisa mulai menggunakan NodeRed, silahkan diketikkan perintah berikut:

\$ node-red

Perhatikan, apakah terlihat proses seperti terlihat pada Gambar 2.19.

A terminal window titled 'root@io-t: ~' with standard window controls. It shows the output of the 'node-red' command. The output includes a warning about system-generated keys, instructions on setting a custom key, and three status messages: 'Server now running at http://127.0.0.1:1880/', 'Starting flows', and 'Started flows'. The prompt at the bottom is 'root@io-t:~#'.

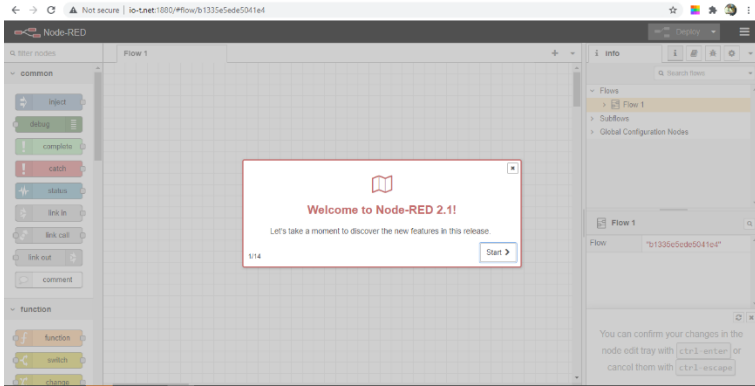
```
root@io-t: ~
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----
3 Nov 15:17:06 - [info] Server now running at http://127.0.0.1:1880/
3 Nov 15:17:06 - [info] Starting flows
3 Nov 15:17:06 - [info] Started flows

root@io-t:~#
```

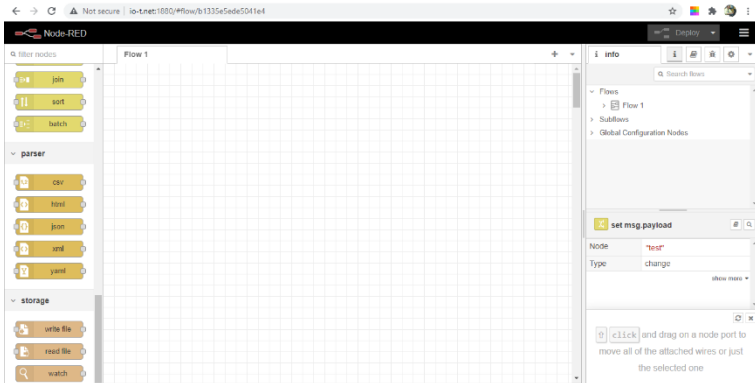
Gambar 2.19. NodeRed sedang berjalan

Selanjutnya, tinggal dibuka browser dan silahkan diakses alamat <http://io-t.net:1880>. Seharusnya akan tampil seperti pada Gambar 2.20.



Gambar 2. 20. Tampilan NodeRed

Silahkan tekan tombol *Start* untuk memulai. Boleh diikuti *tour* untuk mengetahui fitur-fitur NodeRed, sampai NodeRed siap digunakan. Seperti diperlihatkan pada Gambar 2.21.



Gambar 2. 21. NodeRed siap digunakan

Install Grafana

Grafana adalah alat visualisasi dan pemantauan data *opensource* yang terintegrasi dengan data kompleks dari sumber seperti Prometheus, InfluxDB, Graphite, dan Elasticsearch. Grafana memungkinkan kita membuat lansiran, notifikasi, dan filter ad-hoc untuk data kita sekaligus mempermudah kolaborasi dengan rekan tim kita melalui fitur berbagi bawaan.

Langkah-langkah Instalasi Grafana

Pada langkah ini, kita akan menginstal Grafana ke server Ubuntu 18.04. Kita dapat menginstal Grafana baik dengan mengunduhnya langsung dari situs resminya atau melalui repositori APT. Karena repositori APT memudahkan untuk menginstal dan mengelola pembaruan Grafana, akan digunakan metode itu dalam riset ini.

Meskipun Grafana tersedia di repositori resmi paket Ubuntu 18.04, versi Grafana mungkin bukan yang terbaru, jadi gunakan repositori resmi Grafana.

Unduh kunci Grafana GPG dengan `wget`, lalu pipa output ke `apt-key`. Ini akan menambahkan kunci ke daftar kunci tepercaya instalasi APT kita, yang memungkinkan kita mengunduh dan memverifikasi paket Grafana yang ditandatangani GPG. Perintahnya:

```
$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

Dalam perintah ini, opsi `-q` mematikan pesan pembaruan status untuk `wget`, dan `-O` mengeluarkan file yang diunduh ke terminal. Kedua opsi ini memastikan bahwa hanya konten file yang diunduh yang disalurkan ke `apt-key`.

Selanjutnya, kita tambahkan repositori Grafana ke sumber APT:

```
$ sudo add-apt-repository  
"deb https://packages.grafana.com/oss/deb stable main"
```

Refresh cache APT kita untuk memperbarui daftar paket kita:

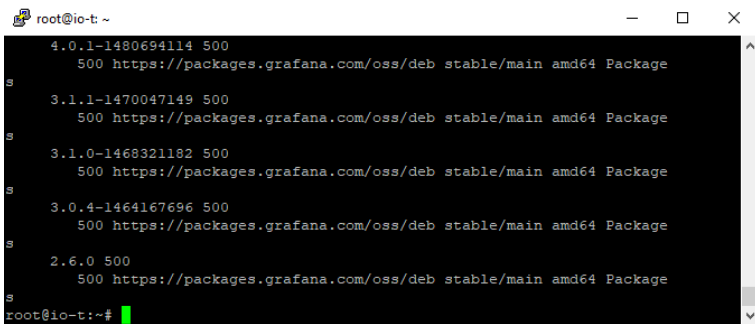
```
$ sudo apt update
```

Selanjutnya, pastikan Grafana akan diinstal dari repositori Grafana:

```
$ apt-cache policy grafana
```

Output dari perintah ini memberi tahu versi Grafana yang akan diinstal, dan dari mana akan diambil paketnya. Verifikasi bahwa kandidat instalasi di bagian atas daftar akan berasal dari

repositori resmi Grafana di <https://packages.grafana.com/oss/deb>. Seperti terlihat pada Gambar 2.22.



```
root@io-t: ~  
4.0.1-1480694114 500  
500 https://packages.grafana.com/oss/deb stable/main amd64 Package  
s  
3.1.1-1470047149 500  
500 https://packages.grafana.com/oss/deb stable/main amd64 Package  
s  
3.1.0-1468321182 500  
500 https://packages.grafana.com/oss/deb stable/main amd64 Package  
s  
3.0.4-1464167696 500  
500 https://packages.grafana.com/oss/deb stable/main amd64 Package  
s  
2.6.0 500  
500 https://packages.grafana.com/oss/deb stable/main amd64 Package  
s  
root@io-t:~#
```

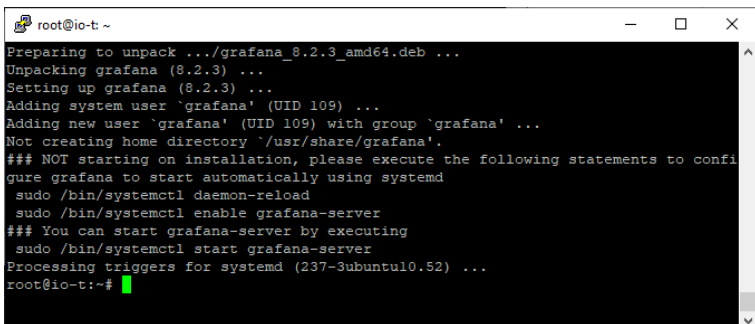
Gambar 2. 22. Sumber paket Grafana

Sekarang dapat dilanjutkan dengan instalasi Grafana:

\$ sudo apt install grafana

Silahkan ditunggu sampai proses instalasi Grafana selesai.

Jika sukses, maka tampilannya seperti terlihat pada Gambar 2.23.



```
root@io-t: ~  
Preparing to unpack .../grafana_8.2.3_amd64.deb ...  
Unpacking grafana (8.2.3) ...  
Setting up grafana (8.2.3) ...  
Adding system user `grafana' (UID 109) ...  
Adding new user `grafana' (UID 109) with group `grafana' ...  
Not creating home directory `/usr/share/grafana'.  
### NOT starting on installation, please execute the following statements to confi  
gure grafana to start automatically using systemd  
sudo /bin/systemctl daemon-reload  
sudo /bin/systemctl enable grafana-server  
### You can start grafana-server by executing  
sudo /bin/systemctl start grafana-server  
Processing triggers for systemd (237-3ubuntu10.52) ...  
root@io-t:~#
```

Gambar 2.23. Grafana sukses diinstal


Setelah Grafana diinstal, gunakan systemctl untuk memulai server Grafana:

\$ sudo systemctl start grafana-server

Selanjutnya, verifikasi bahwa Grafana berjalan dengan memeriksa status layanan:

\$ sudo systemctl status grafana-server

Seharusnya akan tampil seperti terlihat pada Gambar 2.24.

```
 root@io-t ~
```

```
● grafana-server.service - Grafana instance  
Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)  
Active: active (running) since Wed 2021-11-03 22:57:44 UTC; 44s ago  
Docs: http://docs.grafana.org  
Main PID: 1431 (grafana-server)  
Tasks: 7 (limit: 60)  
CGroup: /system.slice/grafana-server.service  
└─1431 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pid=
```

```
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
Nov 03 22:57:44 io-t.net grafana-server[1431]: t=2021-11-03T22:57:44+0000 lvl=info  
~  
lines 1-9/19 (END)
```

Gambar 2.24. Status Grafana-server

Keluaran ini berisi informasi tentang proses Grafana, termasuk statusnya, Pengenal Proses Utama atau *Main Process Identifier* (PID), dan lainnya. aktif (*running*) menunjukkan bahwa proses berjalan dengan benar.

Terakhir, aktifkan layanan untuk memulai Grafana secara otomatis saat boot:

```
$ sudo systemctl enable grafana-server
```

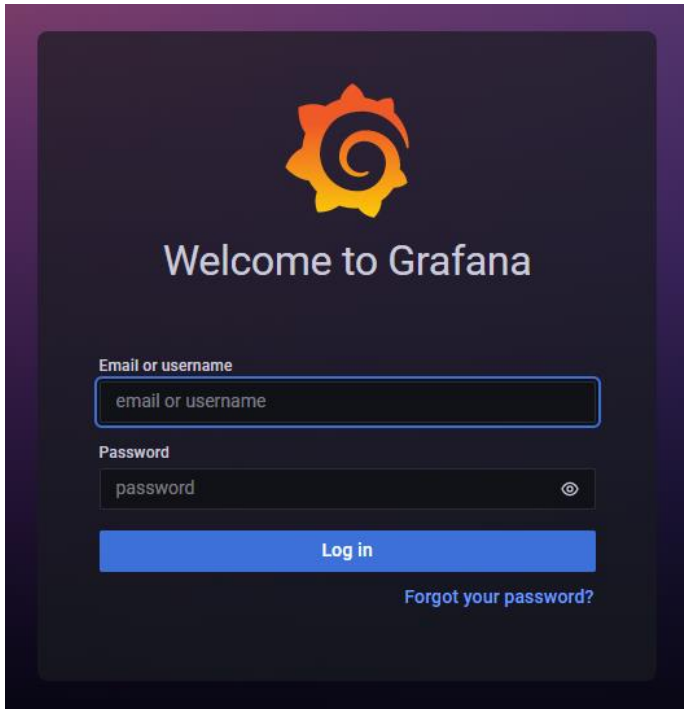
Maka seharusnya akan tampil seperti diperlihatkan pada Gambar 2.25.

```
root@io-t: ~  
login as: root  
root@103.166.156.250's password:  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 3.10.0-1160.21.1.vz7.174.13 x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/advantage  
  
Last login: Wed Nov  3 11:29:11 2021 from 123.255.200.174  
root@io-t:~# sudo systemctl enable grafana-server  
Synchronizing state of grafana-server.service with SysV service script with /lib  
/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable grafana-server  
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.servi  
ce -> /usr/lib/systemd/system/grafana-server.service.  
root@io-t:~#
```

Gambar 2.25. Enable Grafana-server

Grafana sekarang terinstal dan siap digunakan.

Cek Grafana *user interface* pada URL `http://io-t.net:3000`. Seharusnya akan muncul tampilan Grafana seperti terlihat pada Gambar 2.26.



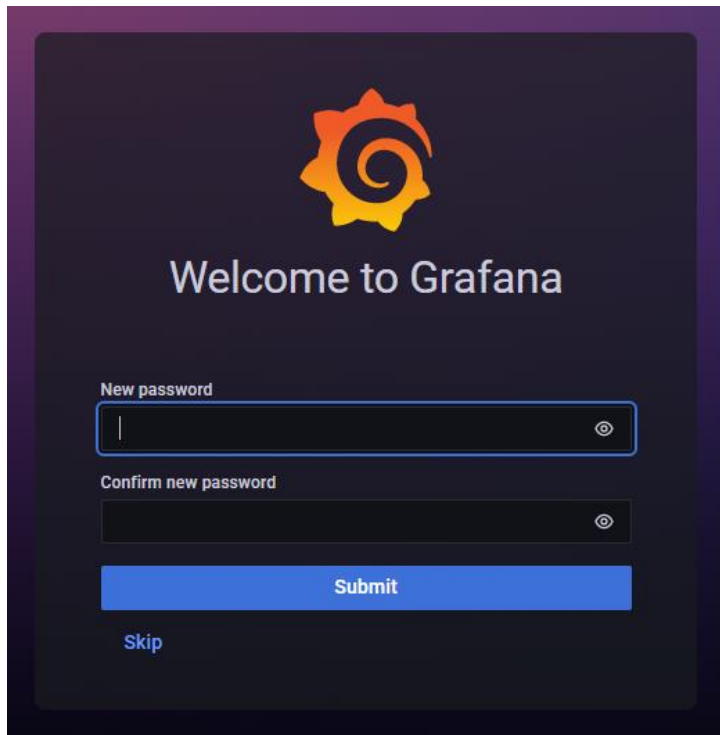
Gambar 2.26. Login Grafana

Silahkan login ke Grafana dengan menggunakan:

Username : admin

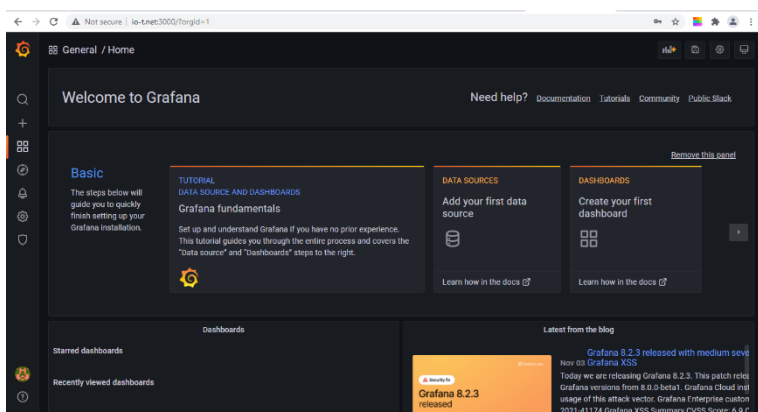
Password : admin

Setelah itu akan diminta untuk mengganti *password*, seperti terlihat pada Gambar 2.27.



Gambar 2.27. Isian mengganti password

Silahkan diganti dengan *password* yang diinginkan. Setelah itu akan muncul *dashboard* Grafana, seperti terlihat pada Gambar 2.28.



Gambar 2.28. Dashboard Grafana

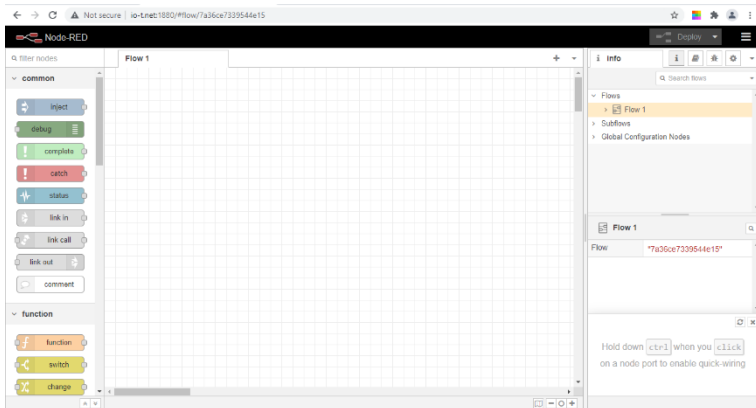
Sampai di sini, Alhamdulillah, telah terlampaui tahap yang paling sulit, dalam proses penyiapan Broker MQTT yang kami kembangkan. Selanjutnya tinggal dilakukan pengaturan-pengaturan menggunakan NodeRed, MySql, dan Grafana, dalam manajemen permintaan *Publish* dan *Subscribe* dari para pengguna Broker MQTT, melalui protokol MQTT Mosquitto.

D. Pengaturan NodeRed

Silahkan jalankan NodeRed seperti yang telah dijelaskan sebelumnya:

```
$ node-red
```

Kemudian silahkan dijalankan melalui browser menggunakan alamat: <http://io-t.net:1880>. Maka akan tampil seperti terlihat pada Gambar 2.29.

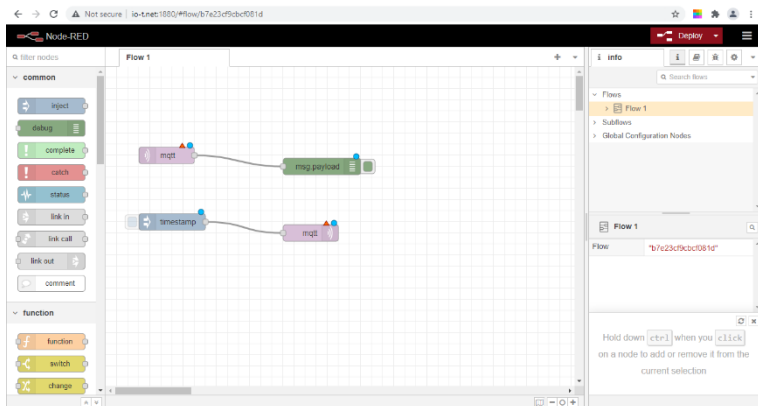


Gambar 2.29. Tampilan awal NodeRed

Siapkan NodeRed sebagai Data Flow

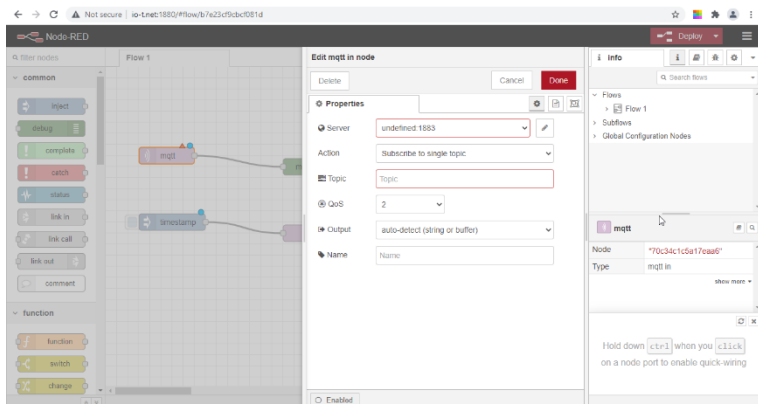
Selanjutnya pada NodeRed UI kita akan test untuk terhubung ke MQTT Mosquitto yang telah kita install sebelumnya. Kita buat *flow* seperti langkah-langkah berikut:

Letakkan mqtt in dan mqtt out, kemudian masing-masing tambahkan inject dan Debug. Mqtt in terhubung ke debug (msg payload), dan Inject (timestamp) terhubung ke mqtt out. Inject ketika diletakkan akan berubah menjadi timestamp, dan debug menjadi msg payload, seperti terlihat pada Gambar 2.30.

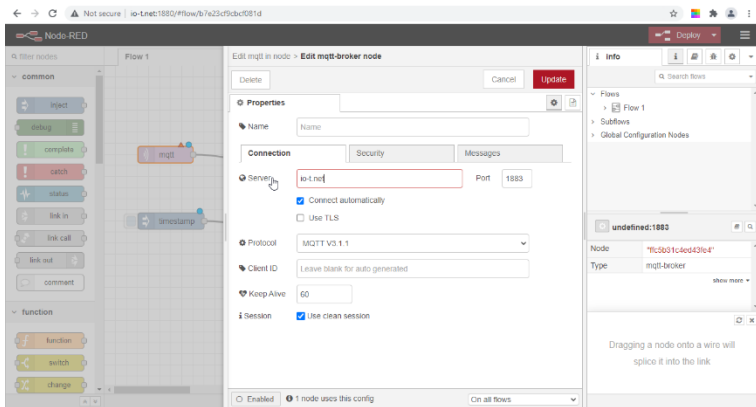


Gambar 2.30. Penyiapan NodeRed sebagai Data Flow (1)

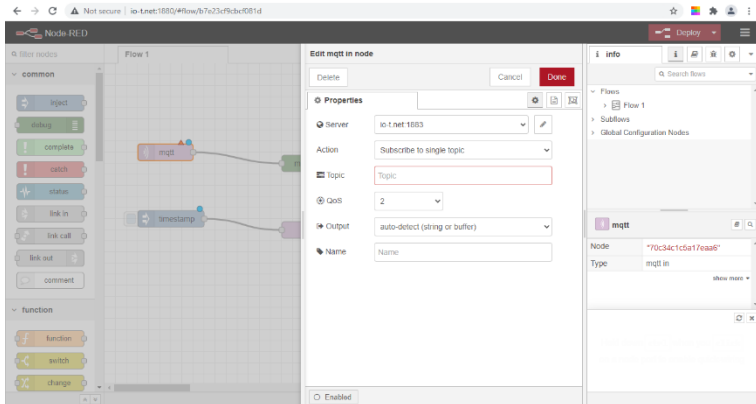
Selanjutnya, pada mqtt in, silahkan diklik dua kali. Dan silahkan diisi sesuai server IoT. Dalam penelitian ini, io-t.net, seperti terlihat pada Gambar 2.31 dan Gambar 2.32. Dan silahkan ditekan tombol Update.



Gambar 2. 31. Penyiapan NodeRed sebagai Data Flow (2)

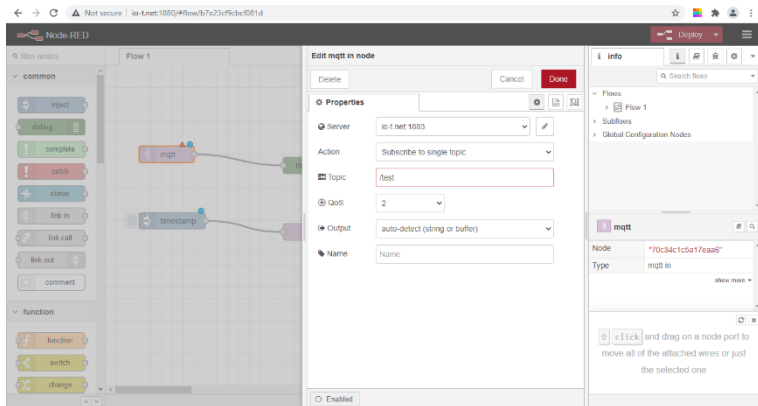


Gambar 2.32. Penyiapan NodeRed sebagai Data Flow (3)



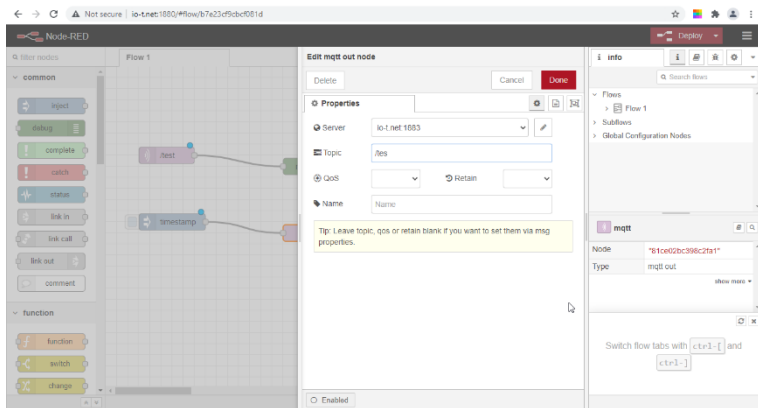
Gambar 2.33. Penyiapan NodeRed sebagai Data Flow (4)

Silahkan isikan Topic dengan /test, seperti terlihat pada Gambar 2.34. Dan tekan Done.



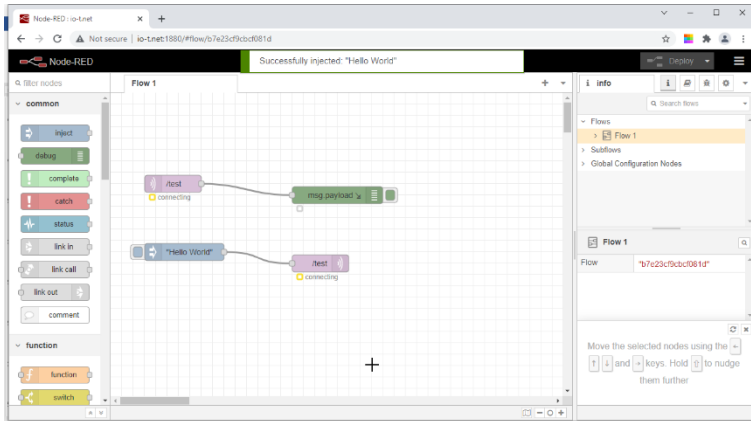
Gambar 2.34. Penyiapan NodeRed sebagai Data Flow (5)

Langkah selanjutnya, silahkan ditekan pada bagian mqtt out. Dan silahkan Topic diisi yang sama, yaitu /test. Dan tekan tombol Done. Seperti terlihat pada Gambar 2.35.



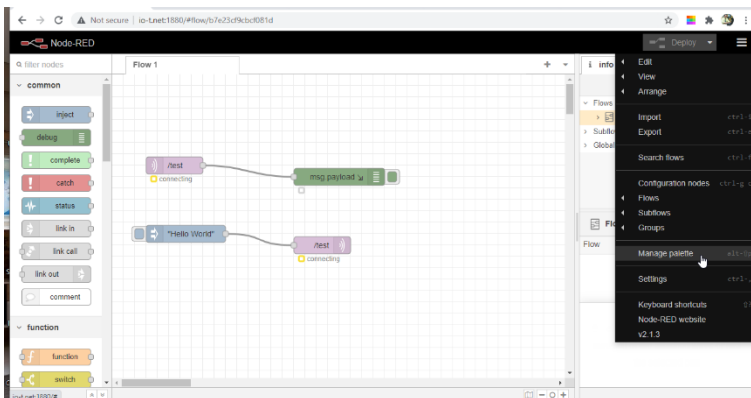
Gambar 2.35. Penyiapan NodeRed sebagai Data Flow (6)

Selanjutnya silahkan ditekan pada timestamp, sisi ujung kiri. Akan keluar pesan: *Warning: node has undeployed changes*. Silahkan ditekan tombol Deploy pada kanan atas. Sampai muncul tulisan: *Successfully deployed*. Selanjutnya, silahkan dilakukan pengujian dengan melakukan proses inject pesan tadi. Dengan cara menekan tombol di ujung timestamp (terdapat tulisan *Hello World*). Jika berhasil akan terlihat pesan sukses, seperti pada Gambar 2.36.

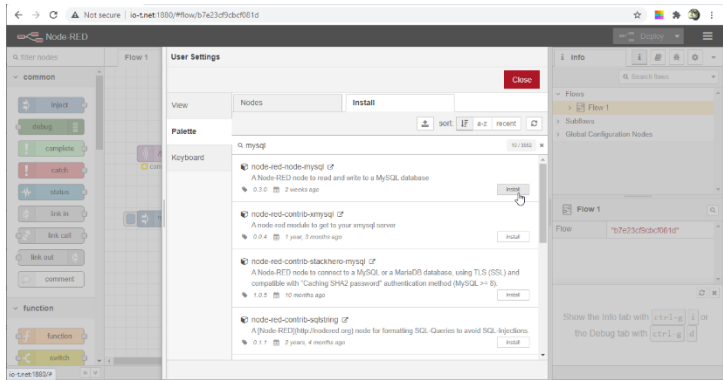


Gambar 2.36. Penyiapan NodeRed sebagai Data Flow (7)

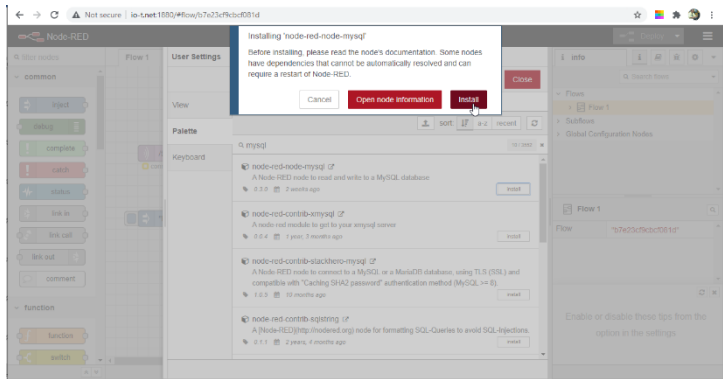
Selanjutnya kita install MySQL extension pada NodeRed UI. Dari menu yang ada di sebelah kanan atas, silahkan ditekan, dan dipilih *Manage Pallette*, seperti pada Gambar 2.37. Dan silahkan diikuti proses instalasi selanjutnya.



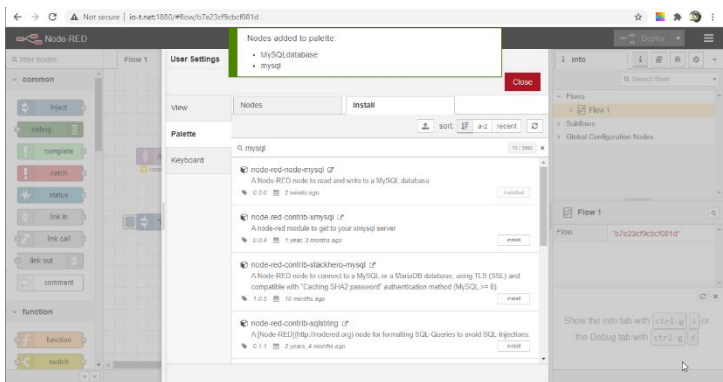
Gambar 2.37. Instalasi MySQL extension pada NodeRed



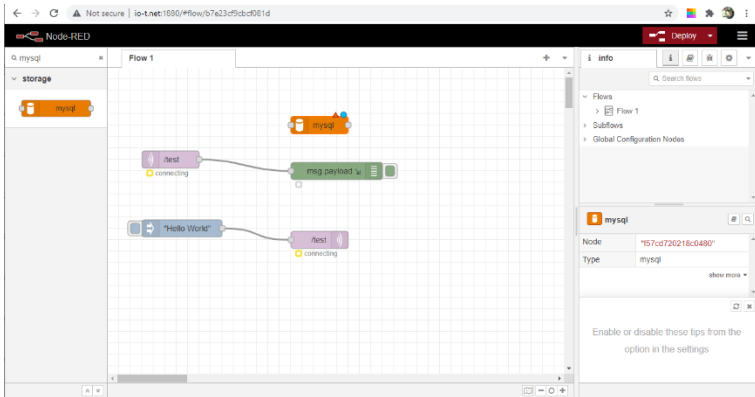
Gambar 2.38. Instalasi MySQL extension pada NodeRed (1)



Gambar 2.39. Instalasi MySQL extension pada NodeRed (2)

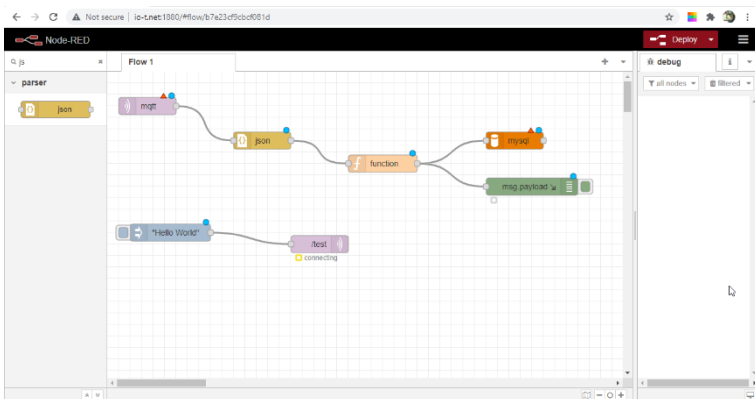


Gambar 2.40. Instalasi MySQL extension pada NodeRed (3)



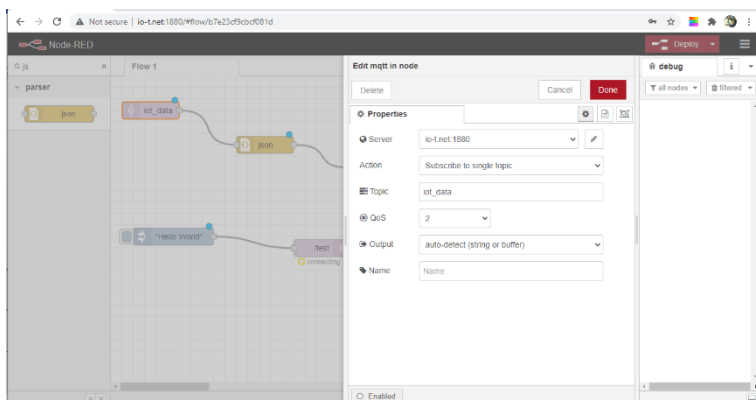
Gambar 2.41. Instalasi MySQL extension pada NodeRed (4)

Selanjutnya, kita buat *flow* untuk *subscribe* MQTT dan *Insert* ke MySQL, seperti terlihat pada gambar-gambar berikut. Yang perlu ditambahkan komponen *function*, *json* dan *mqtt in*.



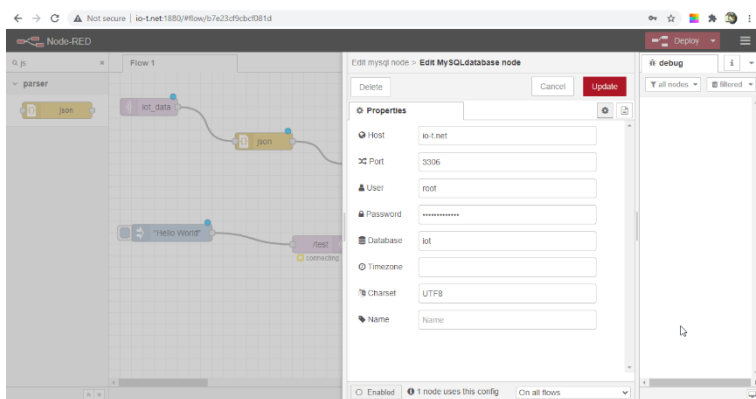
Gambar 2.42. Flow untuk subscribe MQTT dan Insert ke MySQL

Selanjutnya, silahkan tekan pada bagian *mqtt in*, dan silahkan diisikan dengan nama host dan topiknya.



Gambar 2.43. Flow untuk subscribe MQTT dan Insert ke MySQL (1)

Selanjutnya silahkan ditekan komponen Mysql, silahkan diisi dengan database, dan user password, yang digunakan yang ada di server io-t.net.



Gambar 2.44. Flow untuk subscribe MQTT dan Insert ke MySQL (2)

Contoh pengisian function, sesuai dengan tabel yang ada di server, sbb:

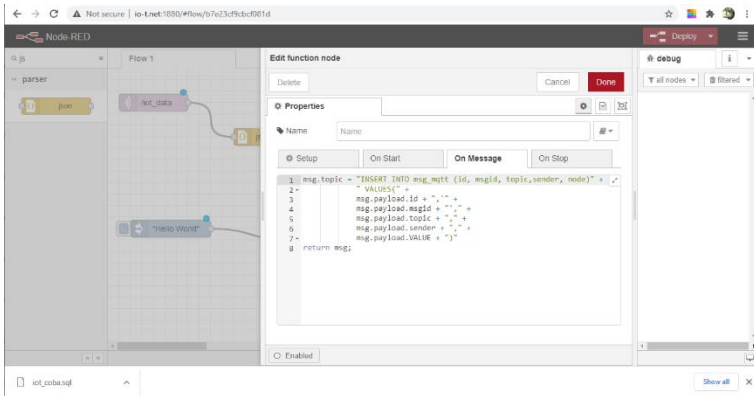
```
msg.topic = "INSERT INTO msg_mqtt (id, msgid, topic, sender,
node)" +
" VALUES(" +
msg.payload.id + "," +
```

```

msg.payload.msgid + "," +
msg.payload.topic + "," +
msg.payload.sender + "," +
msg.payload.VALUE + ")"
return msg;

```

Dimasukkan ke komponen function, seperti terlihat pada Gambar 2.45.



Gambar 2.45. Flow untuk subscribe MQTT dan Insert ke MySQL (3)

Dan contoh yang diisikan di Inject, JSON Sample untuk block inject, sbb:

```

{
  "id":1,
  "msgid":"test",
  "topic":suhu,
  "sender":gue,
  "node":1
}

```

Seperti terlihat pada Gambar 2.46. Dan silahkan tekan tombol Deploy.

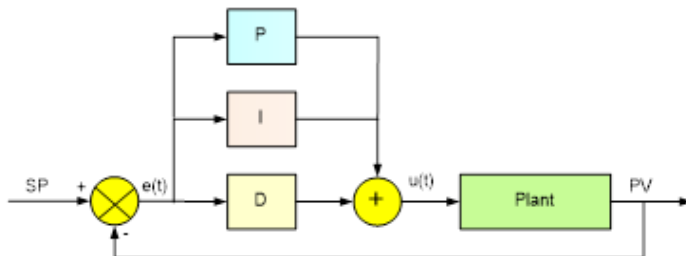
BAB 3

PENGUJIAN SISTEM KENDALI SUHU

A. Penyiapan Sistem Kendali

1. Sistem Kendali PID

Sistem Kendali di industri yang paling terkenal adalah Sistem Kendali Proporsional Integral dan Derivatif (PID). Diagram blok Sistem Kendali PID seperti diperlihatkan pada Gambar 3.1. PID menggabungkan tiga aksi kendali proporsional, integral dan derivatif. Masing-masing aksi kendali ini mempunyai keunggulan-keunggulan tertentu, dimana aksi kendali proporsional mempunyai keunggulan *rise time* yang sangat cepat, aksi kendali integral mempunyai keunggulan untuk memperkecil *error*, dan aksi kendali derivatif mempunyai keunggulan untuk memperkecil *error* atau meredam *overshoot*. Tujuan penggabungan ketiga aksi kendali ini agar dihasilkan keluaran dengan *risetime* yang cepat dan *error* yang kecil.



Gambar 3.1. Sistem Kendali PID

Pengendali PID dalam kerjanya secara otomatis menyesuaikan keluaran kendali berdasarkan perbedaan antara *Setpoint* (SP) dan variabel proses yang terukur (PV), sebagai *error* pengendalian $e(t)$. Nilai keluaran Pengendali $u(t)$ ditransfer sebagai masukan sistem. Masing-masing hubungan yang digunakan seperti terlihat pada Persamaan (3.1) dan (3.2) [7].

$$e(t) = SP - PV \quad (3.1)$$

$$u(t) = u_{bias} + K_c e(t) + \frac{K_c}{\tau_I} \int_0^t e(t) dt - K_c \tau_D \frac{d(PV)}{dt} \quad (3.2)$$

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

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

Selanjutnya, untuk keperluan implementasi digunakan Pengendali PID Diskrit. Pengendali digital diimplementasikan dengan periode sampling diskrit. Bentuk terpisah dari persamaan PID diperlukan untuk memperkirakan integral dan derivatif dari *error*. Modifikasi ini menggantikan bentuk kontinyu integral dengan penjumlahan *error* dan menggunakan Δt sebagai waktu

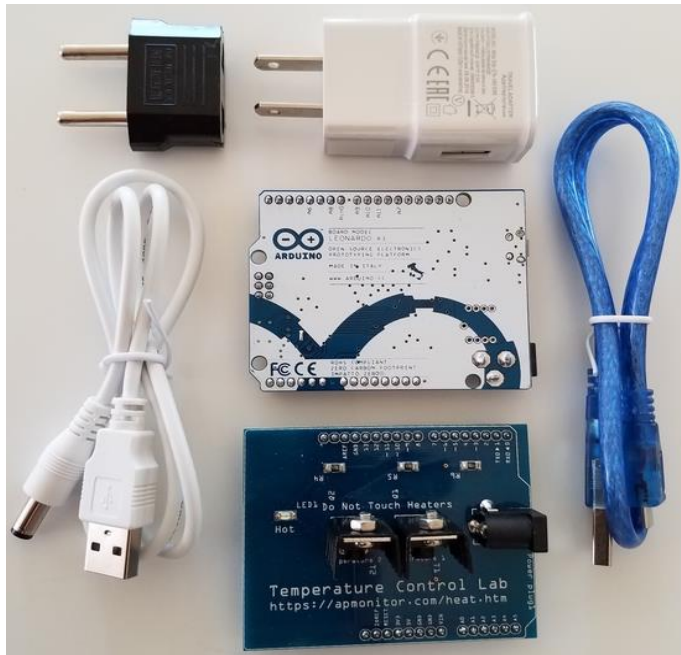
antara contoh pengambilan sampel dan n_t sebagai jumlah contoh pengambilan sampel. Ini juga menggantikan derivatif dengan versi turunannya atau metode lain yang disaring untuk memperkirakan kemiringan instan (PV). Persamaan (3.2) jika dinyatakan kedalam bentuk digital seperti diperlihatkan pada Persamaan (3.3) [7].

$$u(t) = u_{bias} + K_c e(t) + \frac{K_c}{\tau_I} \sum_{i=1}^{n_t} e_i(t) \Delta t - K_c \tau_D \frac{PV_{n_t} - PV_{n_t-1}}{\Delta t} \quad (3.3)$$

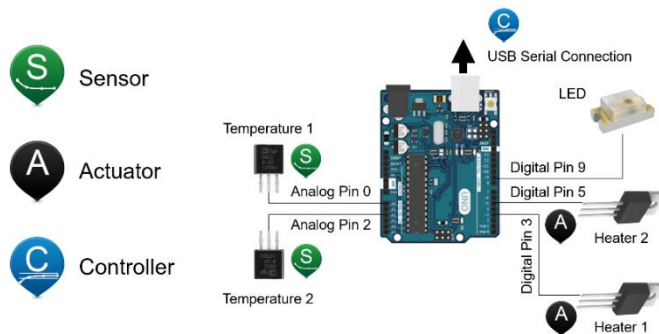
Dari Persamaan (3.3) dapat dilihat tiga parameter penentu keberhasilan proses kendali, yaitu gain K_c , konstanta waktu integral τ_I dan konstanta waktu derivatif τ_D . Proses pencarian atau pengaturan atau penalaan agar diperoleh nilai K_c , τ_I dan τ_D terbaik ini umumnya disebut dengan proses penalaan atau *tuning*.

2. Sistem Kendali Berbasis TCLab

Selanjutnya, untuk keperluan uji-coba sistem kendali menggunakan plant riil, digunakan Lab Kit TCLab. TCLab adalah modul hardware kendali suhu umpan balik dengan Arduino, LED, dua pemanas, dan dua sensor suhu. Keluaran daya pemanas disesuaikan untuk mempertahankan *setpoint* suhu yang diinginkan. Energi panas dari pemanas ditransfer melalui konduksi, konveksi, dan radiasi ke sensor suhu. Panas juga dipindahkan dari perangkat ke lingkungan. TCLab juga dapat digunakan untuk identifikasi model dan pengembangan pengendali. Bentuk TCLab seperti diperlihatkan pada Gambar 3.2 berikut. Gambar skematiknya diperlihatkan pada Gambar 3.3. Informasi lebih lanjut mengenai Lab Kit TCLab ini dapat diakses pada alamat: <http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl>.



Gambar 3.2. TCLab



Gambar 3. 3. Skematik TCLab

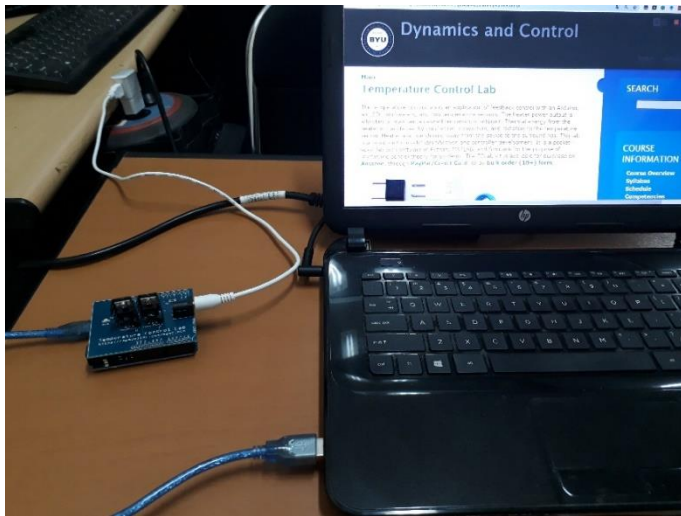
a. Pengaturan dan Instalasi

TCLab dapat bekerja dengan Bahasa Pemrograman MATLAB maupun Python. Paket tambahannya harus diinstal terlebih dahulu agar TCLab dapat digunakan. MATLAB membutuhkan paket dukungan Arduino.

Python membutuhkan paket tambahan seperti paket tclab yang dapat diinstal dengan *pip* melalui Anaconda Prompt.

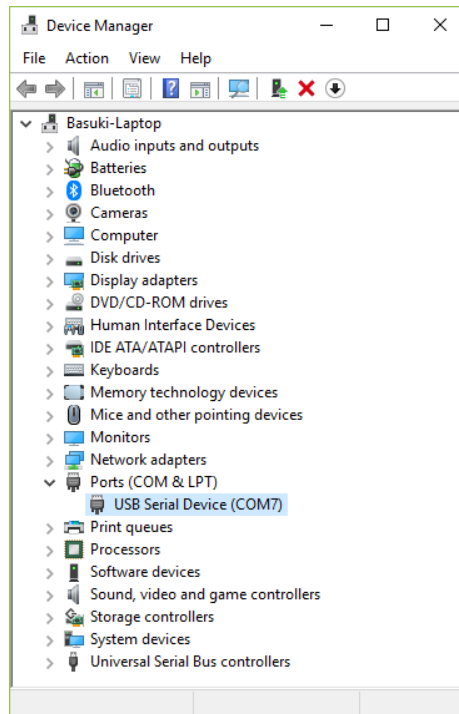
```
>> pip install tclab
```

Untuk menguji apakah Lab Kit TCLab dapat bekerja dengan baik, silahkan hardware TCLab dihubungkan ke komputer (laptop). Power untuk pemanas juga dapat dicolokkan ke listrik. Seperti diperlihatkan pada Gambar 3.4.



Gambar 3.4. Hubungan TCLab ke komputer

Setelah hardware TCLab ditancapkan ke port USB komputer, silahkan cek di *Device Manager*, apakah port *USB Serial Device* (COM berapa) sudah muncul di Ports (COM & LPT). Seperti diperlihatkan pada Gambar 3.5, pada contoh ini muncul di COM 7.



Gambar 3.5. Tampilan USB Serial pada Device Manager

Selanjutnya bisa diuji dengan menjalankan skrip python berikut.

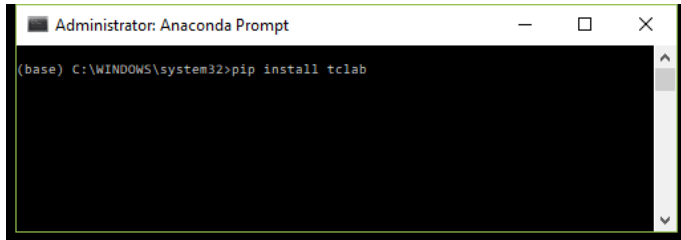
```
import tclab
import time
# Connect to Arduino
a = tclab.TCLab()
print('LED On')
a.LED(100)
# Pause for 1 second
time.sleep(1.0)
print('LED Off')
a.LED(0)
a.close()
```

Jika *library* tclab belum diinstal akan muncul error seperti terlihat pada Gambar 3.6.

```
-----  
AttributeError                                Traceback (most recent call last)  
<ipython-input-3-611ab67d351a> in <module>  
    2 import time  
    3 # Connect to Arduino  
----> 4 a = tclab.TCLab()  
      5 print('LED On')  
      6 a.LED(100)  
  
AttributeError: module 'tclab' has no attribute 'TCLab'
```

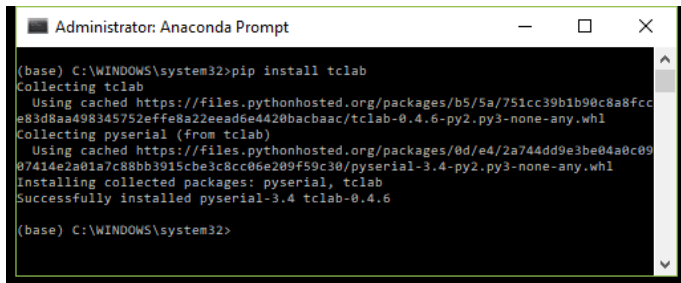
Gambar 3.6. Error jika tclab belum terinstal

Silahkan diinstal *library* tclab melalui Anaconda Prompt seperti pada Gambar 3.7.



Gambar 3.7. Instalasi library tclab

Jika tclab berhasil diinstal, maka akan muncul pesan sukses seperti terlihat pada Gambar 3.8.



Gambar 3.8. TCLab berhasil diinstal

Selanjutnya, jalankan lagi skrip di atas. Jika berhasil akan tampil seperti terlihat pada Gambar 3.9.

```
TCLab version 0.4.6
Arduino Leonardo connected on port COM7 at 115200 baud.
TCLab Firmware Version 1.01.
LED On
LED Off
TCLab disconnected successfully.
```

Gambar 3.9. Pengujian TCLab berhasil

Untuk menguji pemanasnya, ada dua pemanas (Q1 dan Q2) bisa diketikkan skrip python berikut ini.

```
import tclab
import numpy as np
import time
# Connect to Arduino
a = tclab.TCLab()
# Temperatures
print('Temperatures')
print('Temperature 1: ' + str(a.T1) + ' degC')
print('Temperature 2: ' + str(a.T2) + ' degC')
# Turn LED on
print('LED On')
a.LED(100)
# Turn on Heaters (0-100%)
print('Turn On Heaters (Q1=90%, Q2=80%)')
a.Q1(90.0)
a.Q2(80.0)
# Sleep (sec)
time.sleep(30.0)
# Turn Off Heaters
print('Turn Off Heaters')
a.Q1(0.0)
a.Q2(0.0)
# Temperatures
print('Temperatures')
print('Temperature 1: ' + str(a.T1) + ' degC')
print('Temperature 2: ' + str(a.T2) + ' degC')
a.close()
```

Jika dijalankan, akan dihasilkan keluaran seperti terlihat pada Gambar 3.10.

```
TCLab version 0.4.6
Arduino Leonardo connected on port COM7 at 115200 baud.
TCLab Firmware Version 1.01.
Temperatures
Temperature 1: 31.86 degC
Temperature 2: 31.82 degC
LED On
Turn On Heaters (Q1=90%, Q2=80%)
Turn Off Heaters
Temperatures
Temperature 1: 37.21 degC
Temperature 2: 34.11 degC
TCLab disconnected successfully.
```

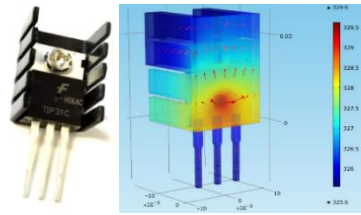
Gambar 3.10. Keluaran pengujian pemanas

b. Pemodelan Dinamis Sistem Pemanas

Tujuan: Menurunkan model transien nonlinear berdasarkan hukum keseimbangan energi (*energy balance*) dan dicocokkan dengan sistem orde satu linier atau orde dua linier. Kemudian hasil prediksi model dibandingkan dengan data transien riil keluaran dari hardware TCLab.

1) Pemodelan Dinamis Keseimbangan Energi

Pemodelan pertama adalah menurunkan model dinamis sistem dengan menggunakan nilai perkiraan parameter. Tiga elemen penting untuk kendali lup tertutup TCLab yaitu perangkat pengukuran (sensor suhu termistor), aktuator (transistor), dan kemampuan untuk melakukan kendali terkomputerisasi (dengan antarmuka USB). Pada keluaran maksimum, transistor melepas daya 1 watt pada keluaran pemanas 100%. Massa transistor dan *heat sink* dengan sirip adalah 4 gram. Diperlihatkan pada Gambar 3.11.



Gambar 3.11. Pemanas TCLab

Pemanas TCLab memiliki kapasitas panas 500 J/kgK. Luas permukaan pemanas dan sensor sekitar 12 cm². Koefisien perpindahan panas konvektif untuk udara diam adalah sekitar 10 W/m²K. Perpindahan panas yang dihasilkan oleh transistor melalui perangkat, utamanya disebabkan oleh faktor konveksi, selain faktor radiasi. Perpindahan panas radiatif dapat dimasukkan kedalam model untuk menentukan fraksi panas yang hilang akibat konveksi dan radiasi panas. Perpindahan panas ditingkatkan dengan kopling termal yang menghubungkan dua komponen. Secara lengkap karakteristik pemanas TCLab seperti yang terdata pada Tabel 3.1.

Tabel 3.1. Karakteristik pemanas TCLab

Quantity	Value
Initial temperature (T_0)	296.15 K (23°C)
Ambient temperature (T_∞)	296.15 K (23°C)
Heater output (Q)	0 to 1 W
Heater factor (α)	0.01 W/(% heater)
Heat capacity (C_p)	500 J/kg-K
Surface Area (A)	1.2x10 ⁻³ m ² (12 cm ²)
Mass (m)	0.004 kg (4 gm)
Overall Heat Transfer Coefficient (U)	10 W/m ² -K
Emissivity (ϵ)	0.9
Stefan Boltzmann Constant (σ)	5.67x10 ⁻⁸ W/m ² -K ⁴

Selanjutnya dilakukan pemodelan dinamis, respons dinamis antara daya masukan ke transistor dan suhu yang dirasakan oleh termistor. Digunakan keseimbangan energi untuk memulai penurunan, seperti pada Persamaan (3.4) [8].

$$mc_p \frac{dT}{dt} = \sum \dot{h}_{in} - \sum \dot{h}_{out} + Q \quad (3.4)$$

Istilah-istilah yang diperlukan untuk keperluan ini diperluas atau disederhanakan. Keseimbangan energi penuh didalamnya sudah termasuk istilah konveksi dan radiasi. Persamaan (3.4) diperluas menjadi Persamaan (3.5) [8].

$$mc_p \frac{dT}{dt} = UA(T_{\infty} - T) + \epsilon \sigma A(T_{\infty}^4 - T^4) + \alpha Q \quad (3.5)$$

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

Pemrograman python untuk menguji keluaran model keseimbangan energi, bisa diketikkan skrip berikut.

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

# define 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]

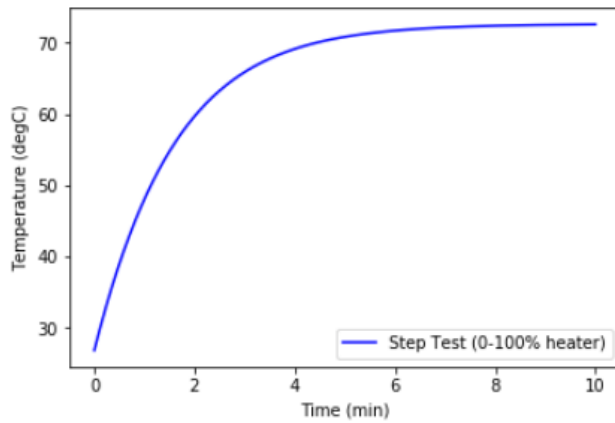
    # Nonlinear Energy Balance
    dTdt = (1.0/(m*Cp))*(U*A*(Ta-T) \
        + eps * sigma * A * (Ta**4 - T**4) \
        + alpha*Q)
    return dTdt

Q = 100.0 # Percent Heater (0-100%)
T0 = 23.0 + 273.15 # Initial temperature
n = 60*10+1 # Number of second time points (10min)
time = np.linspace(0,n-1,n) # Time vector
T = odeint(heat,300.0,time,args=(Q,)) # Integrate ODE

# Plot results
plt.figure(1)
plt.plot(time/60.0,T-273.15,'b-')
plt.ylabel('Temperature (degC)')
plt.xlabel('Time (min)')
plt.legend(['Step Test (0-100% heater)'])
plt.show()

```

Jika dijalankan, hasilnya seperti terlihat pada Gambar 3.12.



Gambar 3.12. Respon step model keseimbangan energi

Dari keluaran model keseimbangan energi berupa transien nonlinear dapat diamati apakah pengaruh perpindahan radiasi panas cukup signifikan. Nantinya apakah respon suhu ini sesuai dengan keluaran model orde satu atau orde dua, berdasarkan nilai-nilai parameter yang tidak pasti pada model berbasis fisis. Tujuan akhirnya nanti apakah model-model ini cukup membantu prediksi suhu jika dibandingkan dengan data riil keluaran langsung dari hardware TCLab.

2) Pemodelan Dinamis Orde Satu Plus Waktu-Tunda

Pemodelan kedua sistem pemanas TCLab ini dicoba didekati dengan sistem linear Model Orde Satu Plus Waktu-Tunda atau *First-Order Plus Dead-Time* (FOPDT). Sistem linear orde satu dengan waktu tunda ini adalah deskripsi empiris umum dari banyak proses dinamis yang stabil. FOPDT digunakan untuk mendapatkan konstanta penalaan pengendali awal.

Skrip python dengan *widget* interaktif berikut menunjukkan efek dari tiga parameter yang dapat ditata atau disesuaikan dalam persamaan FOPDT. Tiga parameter FOPDT yang dimaksud adalah gain K_p , konstanta waktu τ_p , dan waktu tunda θ_p .

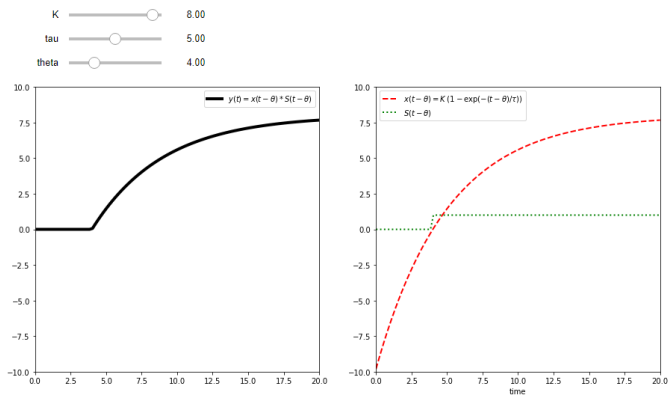
```
import ipywidgets as wg
from IPython.display import display
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt

def fopdtPlot(K,tau,theta):
    n = 100 # time points to plot
    t = np.linspace(0,20,100) # create time vector
    # create 0 -> 1 step at t=theta
    delay = np.empty_like(t)
    for i in range(n):
        if t[i] < theta:
            delay[i] = 0.0
        else:
            delay[i] = 1.0
    # calculate response to step input
    x = K * (1.0-np.exp(-(t-theta)/tau))
    y = x * delay
    # plot response
    plt.figure(1,figsize=(15,7))
    plt.subplot(1,2,1)
    plt.plot(t,y,'k-',linewidth=4,label=r'$y(t)=x(t-\theta)*S(t-\theta)$')
    plt.legend(loc='best')
    plt.ylim([-10,10])
    plt.xlim([0,20])
    plt.subplot(1,2,2)
    plt.plot(t,x,'r--',linewidth=2,label=r'$x(t-\theta)=K\cdot(1-\exp(-(t-\theta)/\tau))$')
    plt.plot(t,delay,'g:',linewidth=2,label=r'$S(t-\theta)$')
```

```
plt.xlabel('time')
plt.legend(loc='best')
plt.ylim([-10,10])
plt.xlim([0,20])
```

```
K_slide = wg.FloatSlider(value=8.0,min=-
10.0,max=10.0,step=0.1)
tau_slide =
wg.FloatSlider(value=5.0,min=0.1,max=10.0,step=0.1)
theta_slide =
wg.FloatSlider(value=4.0,min=0.1,max=15.0,step=0.1)
wg.interact(fopdtPlot, K=K_slide, tau=tau_slide,
theta=theta_slide)
```

Jika dijalankan, maka hasilnya seperti terlihat pada Gambar 3.13.



Gambar 3.13. Pengaruh perubahan parameter FOPDT

Setelah mendapatkan pemahaman intuitif tentang bagaimana parameter-parameter gain K_p , konstanta waktu τ_p , dan waktu tunda θ_p mempengaruhi respon step, penting untuk memahami persamaan matematis FOPDT. Persamaannya diuraikan pada penjelasan berikut ini. Setelah mendapatkan pemahaman intuitif tentang bagaimana

parameter-parameter gain K_p , konstanta waktu τ_p , dan waktu tunda θ_p mempengaruhi respon step, penting untuk memahami persamaan matematis FOPDT. Persamaannya diuraikan pada penjelasan berikut ini.

Persamaan FOPDT dinyatakan dalam bentuk [8]:

$$\tau_p \frac{dy(t)}{dt} = -y(t) + K_p u(t - \theta_p) \quad (3.6)$$

Persamaan (3.6) memiliki variabel keluaran $y(t)$ dan masukan $u(t)$ dan tiga parameter yang tidak diketahui yaitu gain proses K_p , konstanta waktu τ_p , dan waktu tunda θ_p .

Gain proses K_p .

Gain proses adalah perubahan keluaran y yang diinduksi oleh perubahan satuan pada masukan u . Gain proses dihitung dengan cara mengevaluasi perubahan dalam $y(t)$ dibagi dengan perubahan $u(t)$ pada kondisi awal dan akhir kondisi tunak (steady state) [8].

$$K_p = \frac{\Delta y}{\Delta u} = \frac{y_{ss2} - y_{ss1}}{u_{ss2} - u_{ss1}} \quad (3.7)$$

Gain proses mempengaruhi besarnya respon, terlepas dari kecepatan respon.

Konstanta waktu τ_p .

Diberikan perubahan $u(t) = \Delta u$, solusi untuk diferensial orde satu linier (tanpa waktu tunda) menjadi [8]:

$$y(t) = \left(e^{-\frac{t}{\tau_p}} \right) y(0) + \left(1 - e^{-\frac{t}{\tau_p}} \right) K_p \Delta u \quad (3.8)$$

Jika kondisi awal $y(0)=0$ dan pada $t=\tau_p$, maka solusinya disederhanakan sebagai berikut [8]:

$$y(\tau_p) = \left(1 - e^{-\frac{\tau_p}{\tau_p}} \right) K_p \Delta u = (1 - e^{-1}) K_p \Delta u = 0.632 K_p \Delta u \quad (3.9)$$

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

Waktu tunda θ_p .

Waktu tunda dinyatakan sebagai pergeseran waktu yang terdapat didalam variabel input $u(t)$ [8].

$$u(t - \theta_p) \quad (3.10)$$

Misalkan sinyal masukan itu adalah berupa fungsi *step* yang biasanya berubah dari 0 ke 1 pada waktu $t=0$ tetapi pergeseran ini tertunda 5 detik. Fungsi masukan $u(t)$ dan fungsi keluaran $y(t)$ digeser waktunya 5 detik. Solusi untuk persamaan diferensial orde satu dengan waktu tunda diperoleh dengan mengganti semua variabel t dengan $t-\theta_p$ dan menerapkan hasil kondisional berdasarkan waktu tunda θ_p [8].

$$y(t < \theta_p) = y(0) \quad (3.11)$$

$$y(t \geq \theta_p) = \left(e^{-\frac{(t-\theta_p)}{\tau_p}} \right) y(0) + \left(1 - e^{-\frac{(t-\theta_p)}{\tau_p}} \right) K_p \Delta u \quad (3.12)$$

3) Pemodelan Dinamis Orde Dua Plus Waktu-Tunda

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

Persamaannya [8]:

$$\tau_s^2 \frac{d^2 y}{dt^2} + 2\zeta \tau_s \frac{dy}{dt} + y = K_p u(t - \theta_p) \quad (3.13)$$

Persamaan (3.13) memiliki keluaran $y(t)$ dan masukan $u(t)$ dan empat parameter yang tidak diketahui. Keempat parameter tersebut adalah gain K_p ,

faktor redaman ζ , konstanta waktu orde dua τ_s , dan waktu tunda θ_p .

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

Berikut ini skrip python untuk membangkitkan data simulasi keluaran dari Model SOPDT.

```
# Generate process data as data.txt
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# define process model (to generate process data)
def process(y,t,n,u,Kp,taup):
    # arguments
    # y[n] = outputs
    # t = time
    # n = order of the system
    # u = input value
    # Kp = process gain
    # taup = process time constant

    # equations for higher order system
    dydt = np.zeros(n)
    # calculate derivative
    dydt[0] = (-y[0] + Kp * u)/(taup/n)
    for i in range(1,n):
        dydt[i] = (-y[i] + y[i-1])/(taup/n)
    return dydt

# specify number of steps
```

```

ns = 50
# define time points
t = np.linspace(0,40,ns+1)
delta_t = t[1]-t[0]
# define input vector
u = np.zeros(ns+1)
u[5:20] = 1.0
u[20:30] = 0.1
u[30:] = 0.5

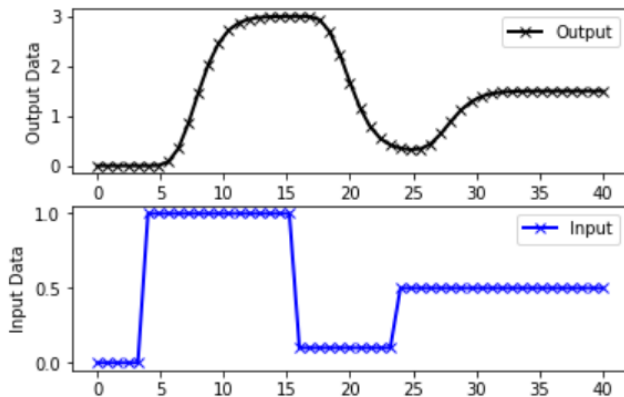
# use this function or replace yp with real process data
def sim_process_data():
    # higher order process
    n=10    # order
    Kp=3.0  # gain
    taup=5.0 # time constant
    # storage for predictions or data
    yp = np.zeros(ns+1) # process
    for i in range(1,ns+1):
        if i==1:
            yp0 = np.zeros(n)
            ts = [delta_t*(i-1),delta_t*i]
            y = odeint(process,yp0,ts,args=(n,u[i],Kp,taup))
            yp0 = y[-1]
            yp[i] = y[1][n-1]
    return yp
yp = sim_process_data()

# Construct results and save data file
# Column 1 = time
# Column 2 = input
# Column 3 = output
data = np.vstack((t,u,yp)) # vertical stack
data = data.T             # transpose data
np.savetxt('data.txt',data,delimiter=',')

```

```
# plot results
plt.figure()
plt.subplot(2,1,1)
plt.plot(t,yp,'kx-',linewidth=2,label='Output')
plt.ylabel('Output Data')
plt.legend(loc='best')
plt.subplot(2,1,2)
plt.plot(t,u,'bx-',linewidth=2)
plt.legend(['Input'],loc='best')
plt.ylabel('Input Data')
plt.show()
```

Jika dijalankan, hasilnya seperti terlihat pada Gambar 3.14.



Gambar 3.14. Keluaran Model SOPDT

c. Pengujian Model Panas

Selanjutnya untuk menguji model-model pemanas yang telah dijelaskan, dibandingkan dengan data riil keluaran dari hardware TCLab. Model-model yang dimaksud yaitu model keseimbangan energi, model orde satu linier dengan waktu tunda, dan model orde dua linier dengan waktu tunda. Hasil prediksi model selanjutnya dibandingkan dengan data transien riil keluaran dari hardware TCLab.

Berikut ini skrip python untuk membandingkan keluaran model keseimbangan energi (energy balance) dan orde satu linier dengan waktu tunda (FOPDT), dengan data riil keluaran dari hardware TCLab.

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

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

# define 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

# Connect to Arduino
a = tclab.TCLab()

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

# Run time in minutes
run_time = 10.0

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

# Temperature (K)
Tsp1 = np.ones(loops) * 23.0 # set point (degC)
T1 = np.ones(loops) * a.T1 # measured T (degC)

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
Q1[10:110] = 50.0 # step up for 100 sec
Q1[200:300] = 90.0 # step up for 100 sec
Q1[400:500] = 70.0 # step up for 100 sec

```

```

print('Running Main Loop. Ctrl-C to end.')
print(' Time  Q1  Q2  T1  T2')
print('{:6.1f} {:6.2f} {:6.2f} {:6.2f} {:6.2f}'.format(tm[0], \
                                                    Q1[0], \
                                                    Q2[0], \
                                                    T1[0], \
                                                    T2[0]))

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

# Main Loop
start_time = time.time()
prev_time = start_time
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
error_eb[i] = error_eb[i-1] + abs(Tp[i]-T1[i])

# Simulate one time step with 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
error_fopdt[i] = error_fopdt[i-1] + abs(Tpl[i]-T1[i])

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

# Print line of data
print('{:6.1f} {:6.2f} {:6.2f} {:6.2f} {:6.2f}'.format(tm[i],
\
Q1[i], \
Q2[i], \
T1[i], \
T2[i]))

# Plot
plt.clf()
ax=plt.subplot(3,1,1)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'ro',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'$T_1$ FOPDT')
plt.plot(tm[0:i],T2[0:i],'bx',label=r'$T_2$ measured')

```

```

plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(3,1,2)
ax.grid()
plt.plot(tm[0:i],error_eb[0:i],'k-',label='Energy
Balance')
plt.plot(tm[0:i],error_fopdt[0:i],'g:',label='Linear')
plt.ylabel('Cumulative Error')
plt.legend(loc='best')
ax=plt.subplot(3,1,3)
ax.grid()
plt.plot(tm[0:i],Q1[0:i],'r-',label=r'$Q_1$')
plt.plot(tm[0:i],Q2[0:i],'b:',label=r'$Q_2$')
plt.ylabel('Heaters')
plt.xlabel('Time (sec)')
plt.legend(loc='best')
plt.draw()
plt.pause(0.05)

# Turn off heaters
a.Q1(0)
a.Q2(0)
# Save text file

a.save_txt(tm[0:i],Q1[0:i],Q2[0:i],T1[0:i],T2[0:i],Tsp1[0:i]
,Tsp2[0:i])
# Save figure
plt.savefig('test_Models.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()

```

```

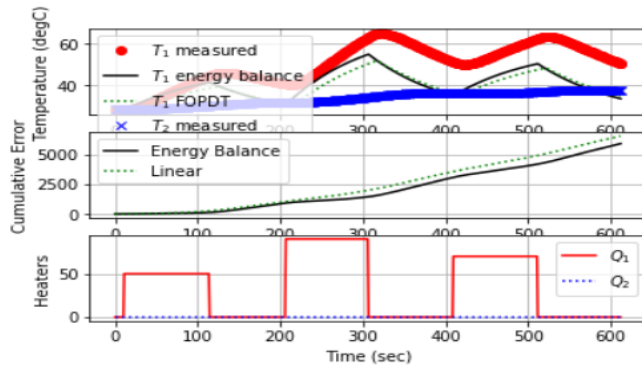
a.save_txt(tm[0:i],Q1[0:i],Q2[0:i],T1[0:i],T2[0:i],Tsp1[0:i]
,Tsp2[0:i])
plt.savefig('test_Models.png')

# 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()

a.save_txt(tm[0:i],Q1[0:i],Q2[0:i],T1[0:i],T2[0:i],Tsp1[0:i]
,Tsp2[0:i])
plt.savefig('test_Models.png')
raise

```

Jika dijalankan maka hasilnya seperti terlihat pada Gambar 3.15.



Gambar 3.15. Perbandingan keluaran model keseimbangan energi, FOPDT dan hardware TClab

Dan berikut ini skrip python untuk membandingkan keluaran model orde dua linier dengan waktu tunda (SOPDT) dengan data riil keluaran dari hardware TCLab.

```
import tclab
import numpy as np
import time
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import random

# Second order model of TCLab
# initial parameter guesses
Kp = 0.2
taus = 50.0
zeta = 1.2
# magnitude of step
M = 80
# overdamped 2nd order step response
def model(y0,t,M,Kp,taus,zeta):
    # y0 = initial y
    # t = time
    # M = magnitude of the step
    # Kp = gain
    # taus = second order time constant
    # zeta = damping factor (zeta>1 for overdamped)
    a = np.exp(-zeta*t/taus)
    b = np.sqrt(zeta**2-1.0)
    c = (t/taus)*b
    y = Kp * M * (1.0 - a * (np.cosh(c)+(zeta/b)*np.sinh(c)))
    + y0
    return y

# define objective for optimizer
def objective(p,tm,y meas):
    # p = optimization parameters
```

```

Kp = p[0]
taus = p[1]
zeta = p[2]
# tm = time points
# ymeas = measurements
# ypred = predicted values
n = np.size(tm)
ypred = np.ones(n)*ymeas[0]
for i in range(1,n):
    ypred[i] = model(ymeas[0],tm[i],M,Kp,taus,zeta)
sse = sum((ymeas-ypred)**2)
# penalize bound violation
if taus<10.0:
    sse = sse + 100.0 * (10.0-taus)**2
if taus>200.0:
    sse = sse + 100.0 * (200.0-taus)**2
if zeta<=1.1:
    sse = sse + 1e6 * (1.0-zeta)**2
if zeta>=5.0:
    sse = sse + 1e6 * (5.0-zeta)**2
return sse

# Connect to Arduino
a = tclab.TCLab()
# Get Version
print(a.version)
# Turn LED on
print('LED On')
a.LED(100)

# Run time in minutes
run_time = 5.0

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

```

```

z = np.zeros(loops)

# Temperature (K)
T1 = np.ones(loops) * a.T1 # measured T (degC)
T1p = np.ones(loops) * a.T1 # predicted T (degC)

# step test (0 - 100%)
Q1 = np.ones(loops) * 0.0
Q1[1:] = M # magnitude of the step

print('Running Main Loop. Ctrl-C to end.')
print(' Time Kp  taus  zeta')
print('{:6.1f}          {:6.2f}          {:6.2f}'
      '{:6.2f}'.format(tm[0],Kp,taus,zeta))

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

# Main Loop
start_time = time.time()
prev_time = start_time
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)
        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

# Estimate parameters after 15 cycles and every 3
steps
if i>=15 and (np.mod(i,3)==0):
    # randomize guess values
    r = random.random()-0.5 # random number -0.5 to
0.5
    Kp = Kp + r*0.05
    taus = taus + r*1.0
    zeta = zeta + r*0.01
    p0=[Kp,taus,zeta] # initial parameters
    solution = minimize(objective,p0,args=(tm[0:i+1],T1[0:i+1]))
    p = solution.x
    Kp = p[0]
    taus = max(10.0,min(200.0,p[1])) # clip to >10,
<=200
    zeta = max(1.1,min(5.0,p[2])) # clip to >=1.1, <=5

# Update 2nd order prediction
for j in range(1,i+1):
    T1p[j] = model(T1p[0],tm[j],M,Kp,taus,zeta)

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

# Print line of data
print('{:6.1f}           {:6.2f}           {:6.2f}
{:6.2f}'.format(tm[i],Kp,taus,zeta))

# Plot

```

```

plt.clf()
ax=plt.subplot(2,1,1)
ax.grid()
plt.plot(tm[0:i],T1p[0:i],'k-',label=r'$T_1 \setminus$, Pred$')
plt.plot(tm[0:i],T1[0:i],'ro',label=r'$T_1 \setminus$, Meas$')
plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(2,1,2)
ax.grid()
plt.plot(tm[0:i],Q1[0:i],'b-',label=r'$Q_1$')
plt.ylabel('Heaters')
plt.xlabel('Time (sec)')
plt.legend(loc='best')
plt.draw()
plt.pause(0.05)

# Turn off heaters
a.Q1(0)
a.Q2(0)
# Save text file

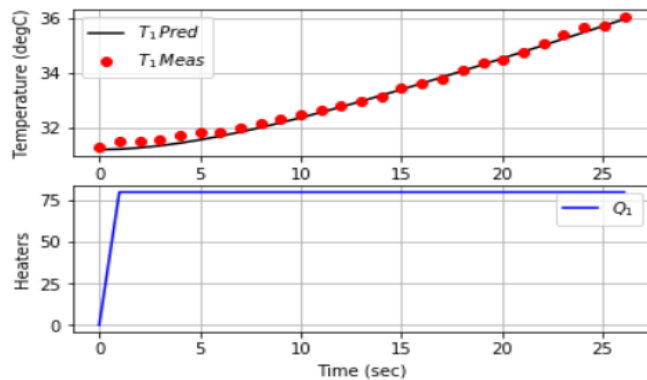
a.save_txt(tm[0:i],Q1[0:i],z[0:i],T1[0:i],T1e[0:i],z[0:i],z[0:i])
# Save figure
plt.savefig('test_Second_Order.png')

# Allow user to end loop with Ctrl-C
except KeyboardInterrupt:
    # Disconnect from Arduino
    a.Q1(0)
    a.Q2(0)
    print('Shutting down')
    a.close()
    a.save_txt(tm[0:i],Q1[0:i],z[0:i],T1[0:i],z[0:i],z[0:i],z[0:i])
    plt.savefig('test_Heaters.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()
    a.save_txt(tm[0:i],Q1[0:i],z[0:i],T1[0:i],z[0:i],z[0:i],z[0:i])
    plt.savefig('test_Second_Order.png')
    raise
```

Jika dijalankan maka hasilnya seperti terlihat pada Gambar 3.16.

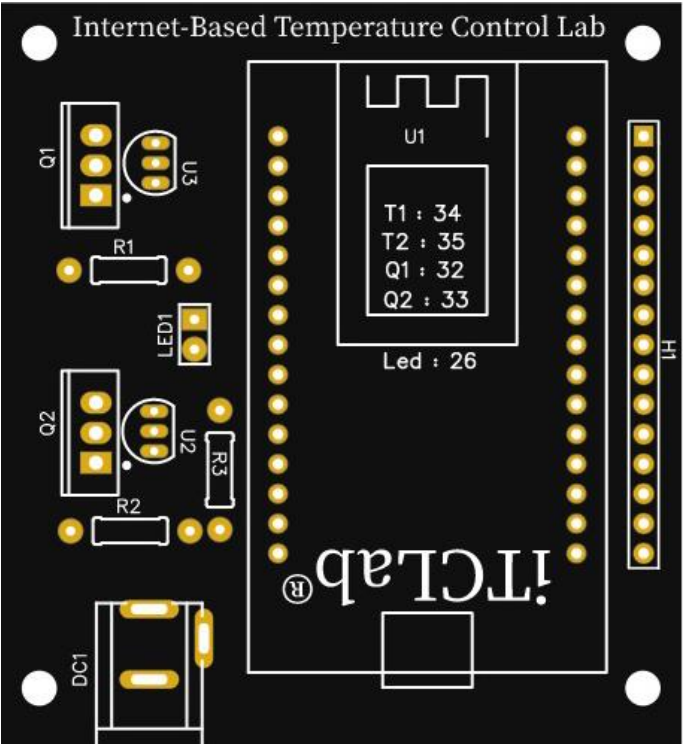


Gambar 3.16. Perbandingan keluaran model SOPDT dan hardware TCLab

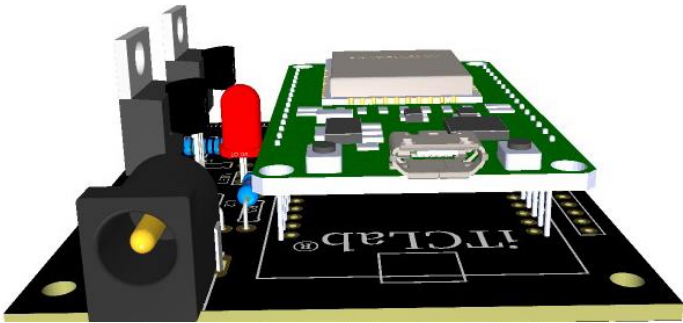
3. Sistem Kendali Berbasis iTCLab

Terinspirasi Kit *Temperature Control Lab* (TCLab) Produk Brigham Young University (BYU) seperti yang telah dijelaskan sebelumnya. Kebutuhan Kit berupa Miniatur Sistem Kendali Suhu dalam Saku, yang bisa digunakan untuk Paket Pembelajaran IoT Praktis, Pengenalan Sistem IoT, Pemrograman IoT, dan Praktek Sistem Kendali Berbasis IoT, maka kami kembangkan iTCLab (internet-Based TCLab). Berikut ini gambaran sistem iTCLab yang kami rancang dan

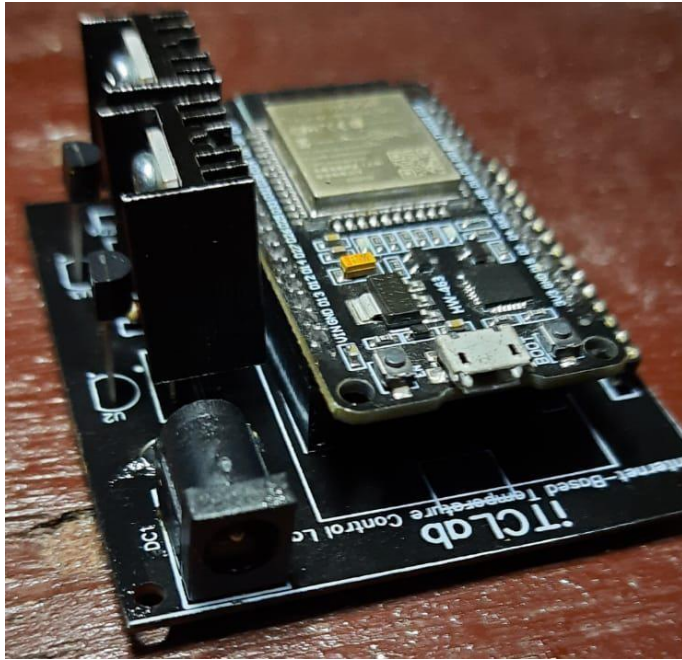
yang sudah menjadi Kit IoT, untuk menguji Broker Web atau Broker MQTT yang kami kembangkan.



Gambar 3.17. Rancangan iTCLab



Gambar 3.18. Rancangan iTCLab (Lanjutan)



Gambar 3.19. Kit iTCLab

Berikut ini contoh program pengujian sistem iTCLab berbasis Kendali PID yang sudah kami lakukan.

```
import itclab
import numpy as np
import time
import matplotlib.pyplot as plt
from scipy.integrate import odeint
#####
# Use this script for evaluating model predictions #
# and PID controller performance for the TCLab #
# Adjust only PID and model sections #
#####
# PID Controller #
#####
# inputs -----
# sp = setpoint
# pv = current temperature
```

```

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

```

```

        op = max(oplo,min(ophi,op))
    # return the controller output and PID terms
    return [op,P,I,D]

#####
# FOPDT model                                     #
#####
Kp = 0.5    # degC/%
tauP = 120.0 # seconds
thetaP = 10  # seconds (integer)
Tss = 23    # degC (ambient temperature)
Qss = 0     # % heater
#####
# Energy balance model                             #
#####
def heat(x,t,Q):
    # Parameters
    Ta = 23 + 273.15 # K
    U = 10.0         # W/m^2-K
    m = 4.0/1000.0   # kg
    Cp = 0.5 * 1000.0 # J/kg-K
    A = 12.0 / 100.0**2 # Area in m^2
    alpha = 0.01     # W / % heater
    eps = 0.9        # Emissivity
    sigma = 5.67e-8  # Stefan-Boltzman

    # Temperature State
    T = x[0]

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

#####

```

```

# Do not adjust anything below this point      #
#####
# Connect to Kit iTClab
a = itclab.iTCLab()

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

# Run time in minutes
run_time = 15.0

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

# Temperature
# set point (degC)
Tsp1 = np.ones(loops) * 25.0
Tsp1[60:] = 50.0
Tsp1[360:] = 30.0
Tsp1[660:] = 40.0
T1 = np.ones(loops) * a.T1 # measured T (degC)
error_sp = np.zeros(loops)

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

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

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

```



```

Q2 = np.ones(loops) * 0.0

print('Running Main Loop. Ctrl-C to end.')
print(' Time   SP   PV   Q1   = P   + I   +   D')
print('{:6.1f} {:6.2f} {:6.2f} ' + \
      '{:6.2f} {:6.2f} {:6.2f} {:6.2f}').format( \
      tm[0],Tsp1[0],T1[0], \
      Q1[0],0.0,0.0,0.0))

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

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

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

        # Read temperatures in Kelvin

```

```

T1[i] = a.T1
T2[i] = a.T2

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

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

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

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

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

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

# Plot
plt.clf()

```

```

ax=plt.subplot(4,1,1)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'.r',label=r'$T_1$ measured')
plt.plot(tm[0:i],Tsp1[0:i],'k--',label=r'$T_1$ set point')
plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(4,1,2)
ax.grid()
plt.plot(tm[0:i],Q1[0:i],'.b-',label=r'$Q_1$')
plt.ylabel('Heater')
plt.legend(loc='best')
ax=plt.subplot(4,1,3)
ax.grid()
plt.plot(tm[0:i],T1[0:i],'.r',label=r'$T_1$ measured')
plt.plot(tm[0:i],Tp[0:i],'k-',label=r'$T_1$          energy
balance')
plt.plot(tm[0:i],Tpl[0:i],'.g-',label=r'$T_1$          linear
model')
plt.ylabel('Temperature (degC)')
plt.legend(loc=2)
ax=plt.subplot(4,1,4)
ax.grid()
plt.plot(tm[0:i],error_sp[0:i],'.r-',label='Set Point Error')
plt.plot(tm[0:i],error_eb[0:i],'k-',label='Energy Balance
Error')
plt.plot(tm[0:i],error_fopdt[0:i],'.g-',label='Linear
Model Error')
plt.ylabel('Cumulative Error')
plt.legend(loc='best')
plt.xlabel('Time (sec)')
plt.draw()
plt.pause(0.05)

# Turn off heaters
a.Q1(0)
a.Q2(0)

```

```

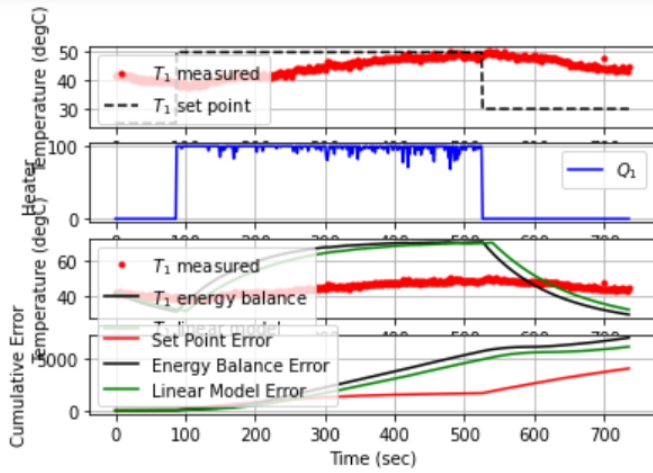
# Save figure
plt.savefig('test_PID.png')

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

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

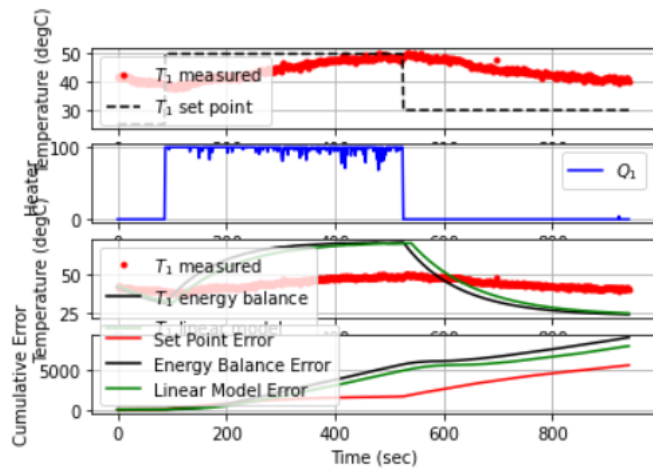
```

Dan contoh-contoh hasil pengujian Kit iTCLab untuk Kendali PID, seperti diperlihatkan pada Gambar 3.19 dan Gambar 3.20.



738.4 30.00 43.71 0.00 -137.10 89.63 2.32

Gambar 3.20. Hasil pengujian Kit iTCLab

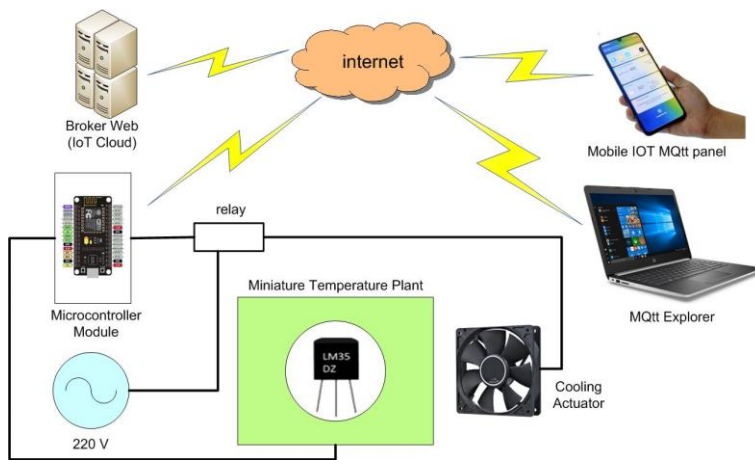


944.5 30.00 40.85 0.00 -108.50 87.19 0.98
 Shutting down
 Arduino disconnected successfully
 Arduino disconnected successfully

Gambar 3.21. Hasil pengujian Kit iTCLab (Lanjutan)

B. Pengujian Sistem Kendali Suhu Berbasis IoT

Seperti pada penjelasan sebelumnya, sistem dasar IoT terdiri dari 3 hal, yaitu: perangkat keras / fisik (things), koneksi internet, dan Cloud pusat data IoT sebagai tempat penyimpanan atau menjalankan aplikasi. Pada Cloud IoT sudah terinstal Broker MQTT, seperti telah dijelaskan sebelumnya, yang berada pada alamat i-ot.net dan io-t.net. Dan untuk menguji kemampuannya, akan dilakukan pemantauan dan pengendalian suhu. Penggunaan layanan Broker MQTT ini, terlebih dahulu harus memiliki akun di web i-ot.net. Dengan akun tersebut akan diperoleh data-data yang diperlukan untuk pengaturan secara remote melalui koneksi internet. Selanjutnya data-data tersebut digunakan untuk pengaturan pada program mikrokontroler. Gambaran arsitektur Sistem Pemantauan dan Pengendalian Suhu, seperti yang ditunjukkan pada Gambar. 3.21.

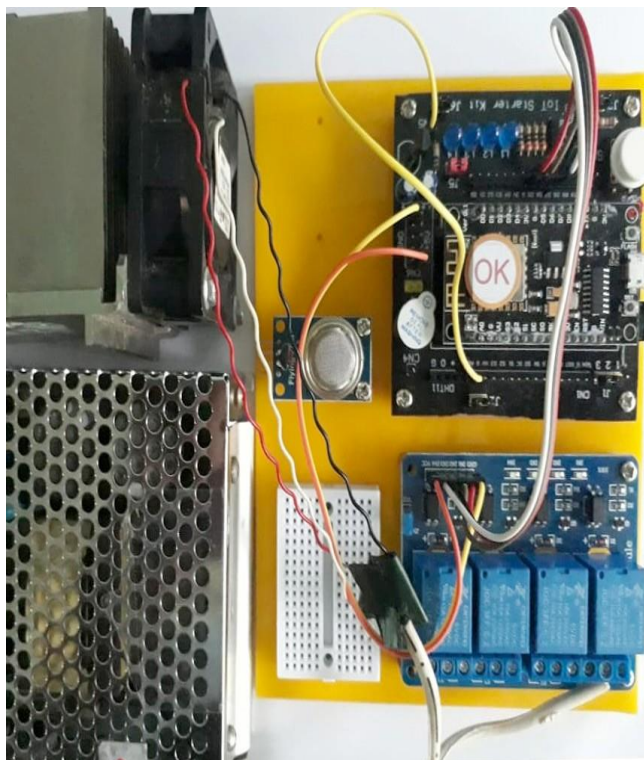


Gambar 3.22. Arsitektur Sistem Pemantauan dan Pengendalian Suhu Berbasis IoT

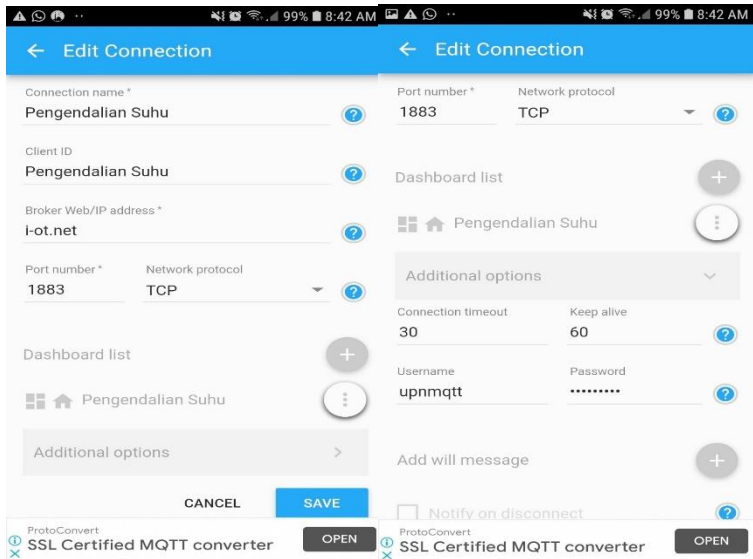
Pada Gambar 3.21, Sistem Pemantauan dan Pengendalian Suhu dirancang untuk digunakan sebagai solusi pemantauan dan pengendalian suhu secara sederhana. Modul mikrokontroler yang digunakan adalah NodeMCU. Dengan rangkaian pendukungnya, sistem ini dilengkapi dengan sensor LM35. Berdasarkan pembacaan sensor suhu ini, digunakan

untuk memutuskan apakah suhu yang terukur ini sudah sesuai dengan nilai suhu yang diinginkan atau tidak. Sistem pengendaliannya yaitu dengan cara menyalakan dan mematikan kipas lamanya sesuai kebutuhan. Pengendalian ini bisa dilakukan secara otomatis, atau dengan penekanan tombol di ponsel melalui koneksi internet. Sambungan internet harus tersedia antara perangkat mikrokontroler yang dilengkapi dengan rangkaian sensor LM35, i-ot.net sebagai Broker MQTT yang kami kembangkan dan aplikasi IoT MQTT Panel pada ponsel.

Perangkat Keras Sistem Pemantauan dan Pengendalian Suhu ini seperti diperlihatkan pada Gambar 3.22. Mikrokontroler yang digunakan pada rangkaian yaitu NodeMCU dilengkapi dengan sensor LM35. Kemudian juga dilengkapi dengan modul relay, power dan kipas yang digunakan untuk mengendalikan suhu. Kemudian untuk pengaturan pada IoT MQTT Panel sesuai dengan data yang didapat dari broker web atau Broker MQTT i-ot.net yang kami kembangkan, contohnya seperti diperlihatkan pada Gambar 3.23. Di mana pengaturan yang terdiri dari *Connection Name* dan *Client ID* dapat diberi nama sebarang, misalnya Pengendalian Suhu. Broker Web / IP Address diisi dengan i-ot.net. Nomor port dan protokol jaringan masing-masing dimasukkan dengan 1883 dan TCP. Kemudian masukkan pengguna dan kata sandi seperti yang telah diberikan oleh broker.



Gambar 3.23. Perangkat keras Sistem Pemantauan dan Pengendalian Suhu



Gambar 3.24. Pengaturan di IoT MQTT Panel

Selanjutnya dilakukan pemrograman yang harus ditanam pada mikrokontroler NodeMCU disesuaikan dengan pengaturan yang telah disediakan oleh broker web atau Broker MQTT, seperti pada Gambar 3.23. Berikut ini contoh program mikrokontroler sederhana untuk Sistem Pemantauan dan Pengendalian Suhu.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

String Topic;
String Payload;

const char* ssid = "ingat_Allah"; // Tergantung wifi yang
digunakan
const char* password = "B15m1l14h_XXX"; // Password wifi

#define IN_1 D5 // Kipas
int outputpin = A0; // sensor LM35
```

```

int analogValue;
float millivolts,celsius;

#define mqttServer "i-ot.net"
#define mqttPort 1883
#define mqttUser "upnmqtt"
#define mqttPassword "20upnmqtt"

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

void receivedCallback(char* topic, byte* payload, unsigned int
length) {

/* we got '1' -> Kipas nyala */
if ((char)payload[0] == '1') {
    digitalWrite(IN_1, HIGH);
    Serial.println("Kipas nyala");
}

/* we got '2' -> Kipas mati */
if ((char)payload[0] == '2') {
    digitalWrite(IN_1, LOW);
    Serial.println("Kipas mati");
}
}

void setup() {
    Serial.begin(115200);
    delay(10);
    pinMode(IN_1, OUTPUT);
    digitalWrite(IN_1, LOW);

    // Connect to WiFi network
    Serial.println();

```

```

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

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

server.begin();
Serial.println("Server started");

Serial.print("Use this URL to connect: ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");

// Connect to Server IoT (CloudMQTT)

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

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

    if (client.connect("ESP32Client", mqttUser, mqttPassword )) {

        Serial.println("connected");
        Serial.print("Message received: ");
    } else {
        Serial.print("failed with state ");
        Serial.print(client.state());
        delay(2000);
    }
}

```

```

    }
    client.subscribe("kipas");
  }
}

void loop() {
  char hasil[4];
  client.loop();
  analogValue = analogRead(outputpin);
  millivolts = (analogValue/1024.0) * 3300;
  celsius = millivolts/10;

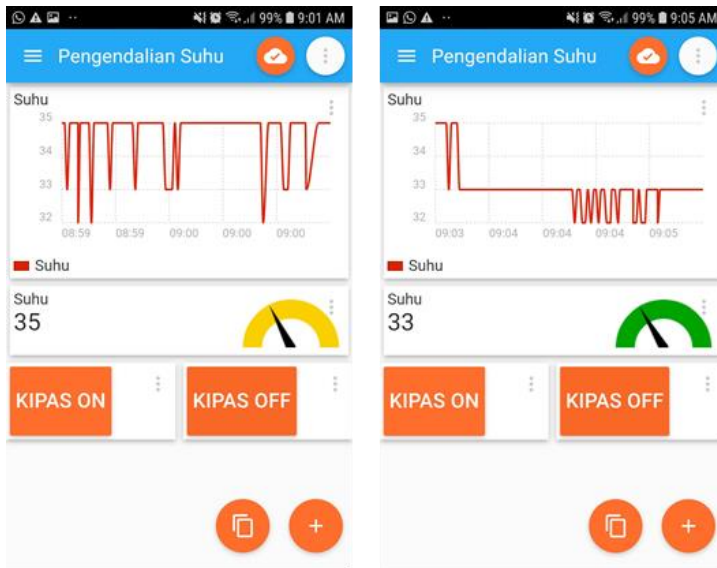
  Serial.print("Temperature: ");
  Serial.print(celsius);
  Serial.print(" Celcius ");
  Serial.println(" send to i-ot.net");

  dtostrf(celsius, 1, 0, hasil);
  client.publish("Suhu",hasil);

  delay (1000);
}

```

Setelah dilakukan proses kompilasi dan upload ke mikrokontroler NodeMCU, selanjutnya dilakukan pengujian. Contoh hasil pemantauan dan pengendalian suhu pada IoT MQTT Panel ditunjukkan pada Gambar 3.24. Tampak dari Gambar 3.24, terjadi kenaikan maupun penurunan suhu, akibat dari penekanan tombol kipas (On/Off).



Gambar 3.25. Contoh hasil pemantauan dan pengendalian suhu di IoT MQTT Panel

Dengan demikian, telah diuji Broker MQTT i-ot.net yang kami kembangkan sebagai Cloud Internet of Things (IoT), untuk Sistem Pemantauan dan Pengendalian Suhu. Dari hasil pengujian tersebut, dapat ditunjukkan bahwa i-ot.net bisa digunakan sebagai alternatif Broker MQTT untuk sistem pengendalian dan pemantauan plant secara umum, melalui ponsel menggunakan IoT MQTT Panel.

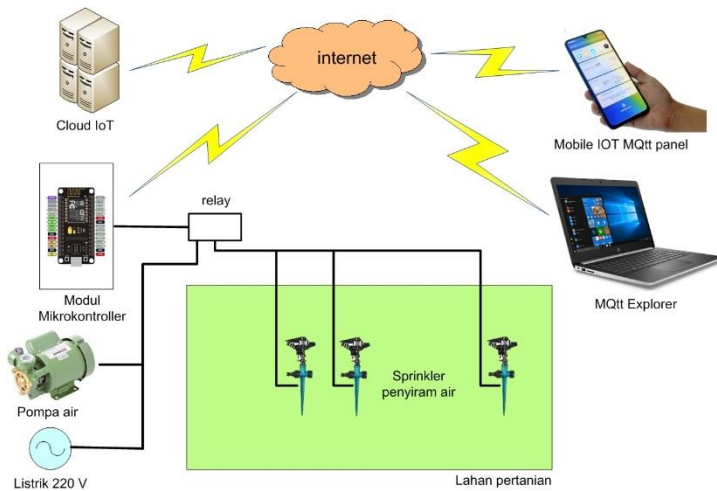
BAB 4

PENGUJIAN SISTEM SMART FARMING

A. Sistem *Smart Farming* Berbasis IoT

Layanan Broker MQTT i-ot.net yang kami kembangkan selanjutnya akan diuji pada sistem *Smart Farming*. *Smart Farming* Berbasis Internet of Things (IoT) adalah solusi pemantauan dan pengendalian jarak jauh terhadap kualitas tanah, Ph, Suhu, kelembaban, dan lain-lain yang dibutuhkan di pertanian, melalui koneksi internet. Program pemantauan dan pengendalian pada *Smart Farming* ini ditanam didalam mikrokontroller. Berdasarkan hasil pembacaan sensor, misalnya sensor kelembaban, digunakan untuk memutuskan apakah lahan pertanian perlu disiram atau tidak. Penyiraman bisa dilakukan secara otomatis, maupun melalui penekanan tombol di ponsel melalui koneksi internet. Koneksi internet harus terhubung antara device mikrokontroller yang sudah dilengkapi sensor, Broker MQTT dan aplikasi IoT MQTT Panel di ponsel.

Contoh penerapan sistem *Smart Farming* untuk penyiraman otomatis maupun secara mobile menggunakan ponsel melalui koneksi Internet of Thing (IoT), berdasarkan pembacaan sensor kelembaban tanah, seperti digambarkan pada Gambar 4.1.



Gambar 4.1. Smart Farming Berbasis IoT

Program *Smart Farming* yang harus ditanam di mikrokontroller, seperti terlihat pada script program berikut ini.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "wifi@yah";
const char* password = "H4n1s4_sby";

#define mqttServer "i-ot.net"
#define mqttPort 1883
#define mqttUser "upnmqtt"
#define mqttPassword "20upnmqtt"

WiFiClient espClient;
PubSubClient client(espClient);
const int relay=4;
int j;
int c;
int batas_atas;
int batas_bawah;
```

```

void receivedCallback(char* topic, byte* payload, unsigned int
length) {
  Serial.print("Message received: ");
  Serial.println(topic);

  Serial.print("payload: ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  /* we got '1' -> on */
  // if (topic=="tombol2")
  // {
  if (((char)payload[0] == '1') && (c==0)) {
    digitalWrite(relay, LOW);
  } else if (((char)payload[0] == '0') && (c==0)) {
    /* we got '0' -> on */
    digitalWrite(relay, HIGH);
  }
  else if ((char)payload[0] == '5') {
    /* we got '0' -> on */
    c=1;
  }
  else if ((char)payload[0] == '6') {
    /* we got '0' -> on */
    c=0;
  }
  // }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  digitalWrite(relay,HIGH);
}

```



```

batas_atas=500;
batas_bawah=300;
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.println("Connecting to WiFi..");
}
Serial.println("Connected to the WiFi network");
pinMode(relay,OUTPUT);

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

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

  if (client.connect("ESP32Client", mqttUser, mqttPassword )) {

    Serial.println("connected");

  } else {

    Serial.print("failed with state ");
    Serial.print(client.state());
    delay(2000);
  }
}
client.subscribe("tombol2");
client.subscribe("control");
j=10;
c=1;
}

void loop() {
  // put your main code here, to run repeatedly:
  float sensor;
  /* sensor= analogRead(A0);

```

```

Serial.println(sensor);
digitalWrite(relay,HIGH);
delay(2000);
digitalWrite(relay,LOW);
delay(2000);*/

char hasil[4];
client.loop();

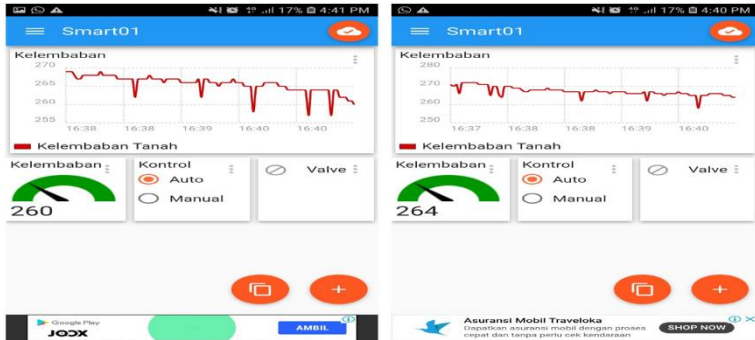
if (j==10)
{
  sensor= analogRead(A0);
  Serial.println(sensor);
  dtostrf(sensor, 1, 0, hasil);
  client.publish("Kelembaban",hasil);
  j=0;
  if (c==1)
  {
    if (sensor<batas_bawah)
    {
      digitalWrite(relay, HIGH);
    }
    if (sensor>batas_atas)
    {
      digitalWrite(relay, LOW);
    }
  }
  j++;
  delay (200);
}

```

B. Pengujian Sistem *Smart Farming* Berbasis IoT

Sesuai dengan gambaran sistem *Smart Farming* Berbasis IoT pada Gambar 4.1, berikut ini contoh hasil-hasil pengujian sistem yang sudah kami lakukan. Hasil pengujian pengendalian

di ponsel dan di lahan pertanian seperti diperlihatkan pada gambar-gambar berikut ini.



Gambar 4.2. Hasil Pengujian Smart Farming



Gambar 4.3. Hasil pengujian Smart Farming di Lahan Pertanian



Gambar 4.4. Hasil pengujian Smart Farming di Lahan Pertanian (Lanjutan)

BAB 5 | PENUTUP

Telah diuraikan dalam Buku Referensi ini bagaimana menyiapkan Cloud Internet of Things (IoT) yang diinstal Broker *Message Queuing Telemetry Transport* (MQTT) sebagai entitas perantara yang memungkinkan klien MQTT untuk berkomunikasi. Broker MQTT yang kami siapkan berbasis open source Mosquitto. Serta hasil-hasil pengujiannya menggunakan Plant Kendali Suhu dan *Smart Farming* berbasis IoT.

DAFTAR PUSTAKA

- [1] R. K. Chahal, N. Kumar, and S. Batra, "Trust management in social Internet of Things: A taxonomy, open issues, and challenges," *Comput. Commun.*, vol. 150, pp. 13–46, 2020.
- [2] S. Ravidas, A. Lekidis, F. Paci, and N. Zannone, "Access control in Internet-of-Things: A survey," *J. Netw. Comput. Appl.*, vol. 144, pp. 79–101, 2019.
- [3] S. Zeadally and O. Bello, "Harnessing the power of Internet of Things based connectivity to improve healthcare," *Internet of Things*, p. 100074, 2019.
- [4] M. Gheisari, G. Wang, and S. Chen, "An Edge Computing-enhanced Internet of Things Framework for Privacy-preserving in Smart City," *Comput. Electr. Eng.*, vol. 81, p. 106504, 2020.
- [5] anonymous, "mCLOUD IoT Platform Services," *mthinx.com*, 2019.
- [6] M. Kashyap, V. Sharma, and N. Gupta, "Taking MQTT and NodeMcu to IOT: Communication in Internet of Things," *Procedia Comput. Sci.*, vol. 132, pp. 1611–1618, 2018.
- [7] B. Y. U. BYU, "Apmonitor.com," *Proportional Integral Derivative (PID) Control*. 2018.
- [8] B. Y. U. BYU, "Apmonitor.com," *Temperature Control Lab*. 2018.

TENTANG PENULIS

PENULIS 1

Dr. Basuki Rahmat, S.Si., M.T.



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

PENULIS 2

Dr. Ir. Minto Waluyo, M.M.



Minto Waluyo, adalah Dosen Teknik Industri, Fakultas Teknik, Universitas Pembangunan Nasional “Veteran” Jawa Timur. Dia menerima gelar Sarjana S-1 Teknik kimia Universitas Pembangunan Nasional “Veteran” Jawa Timur, Magister Manajemen UNKRIS Jakarta, Doktor Manajemen Unair Surabaya. Minat penelitiannya adalah di bidang Manajemen Perusahaan Industri

PENULIS 3

Ir. Tuhu Agung Rachmanto, M.T.



Tuhu Agung Rachmanto, adalah Dosen Teknik Lingkungan, Fakultas Teknik, Universitas Pembangunan Nasional “Veteran” Jawa Timur. Dia menerima gelar Sarjana Teknik Kimia dari Fakultas Teknik Universitas Pembangunan Nasional “Veteran” Jawa Timur pada tahun 1986 dan menerima gelar Magister Teknik Kimia dari Institut Teknologi Sepuluh Nopember pada tahun 1999. Minat penelitiannya adalah di bidang Waste water treatment and Chemical Proses reaction Instrumentation.