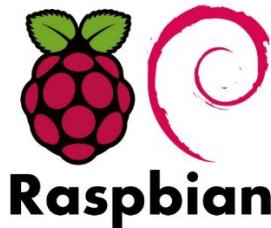


Internet of Things Menggunakan NodeMCU ESP8266-12E Untuk Pemula



i

Penulis: Dodit Suprianto, Rini Agustina, Tiffany Azhar Izzuddin

Daftar Isi

Daftar Isi

Kata Pengantar	v
BAB 1 Internet of Things	1
1.1 Pengantar IoT	1
1.2 Bagaimana IoT Bekerja.....	4
1.3 Mengapa IoT Penting	7
1.4 Keuntungan IoT Bagi Organisasi	8
1.5 Pro dan Kontra Teknologi IoT	9
1.6 Isu Berkaitan Dengan IoT	10
BAB 2 Arsitektur Internet of Things.....	12
2.1 Building Block Internet of Things	12
2.2 MQTT Messaging Protocol.....	13
BAB 3 Microcontroller Unit.....	19
3.1 NodeMCU ESP-12E Sebagai “Things”	19
3.1.1 Smart Devices (“Things”)	19
3.1.2 Jenis-jenis Microcontroller	21
3.1.3 Antarmuka Komunikasi Microcontroller	22
3.1.4 Microcontroller NodeMCU ESP-12E.....	25
3.2 Software Pendukung	30
3.3 Instalasi & Konfigurasi IDE MCU	30
3.4 Instalasi Aplikasi Fritzing	41
3.5 Aneka Projek Microcontroller.....	46
3.5.1 Running LED	46
3.5.2 Kontrol Remote IR Dengan VS1838.....	49
3.5.3 Suhu & Kelembaban Dengan DHT11	60
3.5.4 Intesitas Cahaya Dengan LDR	69

3.5.5 Pembacaan Kartu RFID Mifare RC522.....	72
3.5.6 Hitung Jarak Dengan Ultrasonic HC-SR04	79
BAB 4 Komunikasi MCU Dengan Socket TCP.....	84
4.1 Pemrograman Socket TCP	84
4.2 Rancangan “Things” NodeMCU	87
4.3 NodeMCU & Aplikasi Desktop C#	93
4.3.1 NodeMCU Sebagai Socket Server.....	94
4.3.2 NodeMCU Sebagai Socket Client	113
4.4 NodeMCU & Aplikasi Desktop Java.....	118
4.4.1 Rancangan NodeMCU Sebagai Socket Server.....	118
4.4.2 Antarmuka Aplikasi Desktop Java Swing.....	121
4.4.3 Program Kontrol Socket Client Java Swing	130
BAB 5 Edge Computing Blynk.....	155
5.1 Topologi IoT Server Jaringan Lokal	156
5.2 Kebutuhan Hardware & Software IoT Server Blynk	157
5.3 Software SDCard Formatter	160
5.4 Software Balena Etcher.....	163
5.5 Format Memori microSD	164
5.6 Burning Image Linux Raspbian Ke MicroSD	166
5.7 Instalasi Linux Raspbian Pada Raspberry Pi 3	169
5.8 Instalasi Server IoT Blynk	178
BAB 6 IoT Server Blynk	188
6.1 Blynk Client Android	188
6.2 Instalasi Aplikasi Blynk Client Android	189
6.3 Antarmuka Blynk Client Pada Android.....	194
6.4 Menghubungkan Smart Device NodeMCU Ke Blynk....	208
BAB 7 Server IoT Thingsboard	223

7.1 Instalasi Sistem Operasi Linux Ubuntu MATE.....	224
7.1.1 Burning Ubuntu Mate Ke CD	225
7.1.2 Burning Ubuntu Mate ke Flashdisk	228
7.2 Instalasi Linux Ubuntuk Mate ke Komputer PC/Laptop.	228
7.3 Install IoT Server Thingsboard Pada Ubuntu Mate	234
7.4 Manajemen Thingsboard.....	261
7.4.1 Manajemen Thingsboard.....	262
Penutup.....	265
Referensi.....	266
Biografi Penulis	267

Kata Pengantar

“Segala puji bagi Allah, kita memuji-Nya dan meminta pertolongan, pengampunan, dan petunjuk-Nya. Kita berlindung kepada Allah dari kejahatan diri kita dan keburukan amal kita. Barang siapa mendapat dari petunjuk Allah maka tidak akan ada yang menyesatkannya, dan barang siapa yang sesat maka tidak ada pemberi petunjuknya baginya. Aku bersaksi bahwa tidak ada Tuhan selain Allah dan bahwa Muhammad adalah hamba dan Rasul-Nya. Ya Allah, semoga doa dan keselamatan tercurah pada Muhammad dan keluarganya, dan sahabat dan siapa saja yang mendapat petunjuk hingga hari kiamat”

Buku ini membahas tentang Internet of Things pada tingkat dasar yang diperuntukkan bagi orang awam. Cakupan pembahasan IoT yang luas tidak memungkinkan bila tulisan IoT dirangkum menjadi satu buku.

Materi buku secara garis besar dibagi menjadi dua, yaitu mengulas Internet of Things secara teori dan praktik. Pembaca diharapkan bisa memahami *stage* apa saja yang ada di dalam arsitektur Internet of Things, memahami bagaimana Internet of Things bekerja, dan bagaimana hubungan pada tiap bagian di dalam arsitektur IoT. Sehingga diharapkan pembaca dikemudian hari dapat berkreasi mengembangkannya menjadi projek-projek Internet of Things yang lebih kompleks.

Pembahasan IoT sendiri melibatkan beberapa disiplin ilmu sekaligus, misalnya sistem kontrol dengan *microcontroller unit* (MCU), sistem operasi linux raspbian, jaringan komputer, routing (jika diperlukan), sistem dashboard Internet of Things dan bahasa pemrograman C. Oleh karena itu, pembaca diharapkan memiliki kemampuan dasar tentang komponen-komponen pendukung tersebut.

Bagi pembaca yang benar-benar awam tidak perlu khawatir karena buku ini bersifat “best-practice” yang dapat membimbing Anda setahap demi setahap mencoba setiap projek yang tersedia di dalam buku ini.

Saat ini IoT tidak hanya dikuasai oleh mahasiswa jurusan elektro saja, tetapi juga mahasiswa jurusan lain, siswa SMK, bahkan orang awam yang ingin memanfaatkannya. Banyak sekali kegunaan Internet of Things, misalnya sebagai alat otomasi, pengendali elektrik, pengendali mekanik, akuisisi data sensor, monitoring dan lain-lain. Arduino juga bisa disinergikan dengan alat lain melalui berbagai antarmuka seperti serial, wireless (wifi, bluetooth, infra red) dan lain-lain.

Akhir kata, semoga buku ini bisa membantu Anda untuk terus berkreasi secara produktif, menjadi ilmu yang bermanfaat bagi kita semua, Amin.

Malang, Februari 2021

BAB 1

Internet of Things

1.1 Pengantar IoT

Digitalisasi industri 4.0 yang melaju pesat, bergerak secara eksponensial berpengaruh pada semua aspek, antara lain sosial, ekonomi dan teknologi. Evolusi yang berkembang di dunia industri harus dipahami sebagai sebagai jalan untuk mencapai tujuan yang lebih penting, yaitu menciptakan kesejahteraan sosial. Revolusi industri tahap empat ditandai dengan berbagai teknologi yang menggabungkan dunia fisik, digital, dan biologis yang mempengaruhi semua disiplin ilmu, kehidupan perekonomian, aktivitas industri, dan kapasitas tenaga kerja. Adapun basis orientasi industri tahap empat adalah otomatisasi digitalisasi yang membuat proses produksi semakin efisien dengan kualitas produk lebih baik, sehingga meningkatkan kemampuan bersaing.

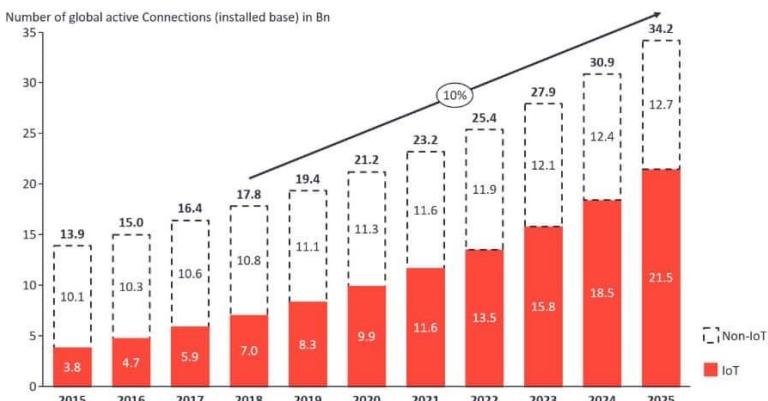
Beberapa teknologi yang berperan dalam industri 4.0, antara lain advanced robotic, additive manufacturing, augmented reality, simulation, horizontal/vertical integration, industrial internet, cloud, cyber security, big data dan analytics.



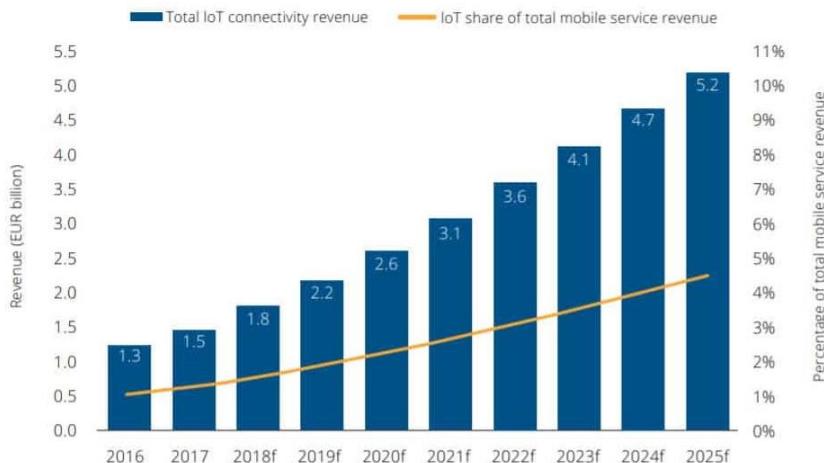
Tren Teknologi Era Evolusi Industri 4.0

Begitu pula dengan IoT (internet of things) yang merupakan bagian dari teknologi digitalisasi revolusi industri empat yang sedang tren saat ini dan akan datang. IoT bersifat interdisipliner, artinya IoT dapat melibatkan berbagai bidang ilmu sekaligus, misalnya elektronika digital & analog, sistem kontrol, perangkat lunak, sistem informasi, *database*, jaringan, *cyber security*, *cyber physical system*, *system on chip* dan lain-lain. Oleh karena itu, penting bagi kita untuk mempelajari IoT dengan pertimbangan berbagai tantangan dan peluang kedepan yang semakin kompleks, dimana persaingan tidak hanya dikuasai oleh orang yang hanya ahli di satu bidang. Sebagai contoh bidang elektronik atau bidang teknologi informasi saja, tetapi berbagai jenis bidang yang saling memberi kontribusi satu sama lain seperti yang dijelaskan pada bagian di atas.

Sejak dicetuskannya istilah IoT pada tahun 1999, internet of things (IoT) telah berubah dari sekadar visi menjadi realitas yang gamblang. Hal ini dapat dikaitkan dengan penggunaan secara luas Internet Protocol (IP), munculnya komputasi dimana-mana, dan kemajuan analisis data tingkat lanjut. Bahkan diperkirakan pada tahun 2025 terdapat 34,2 miliar perangkat yang terhubung ke IoT secara global dan pendapatan di bidang IoT mencapai 5,2 miliar Euro. Tentu hal ini akan menjadi peluang tersendiri bagi kita untuk turut andil dalam industri di bidang IoT.



Jumlah Perangkat Yang Terhubung Ke Jaringan Global
(Sumber: <https://www.comparitech.com/internet-providers/iot-statistics/>)



Total Pendapatan Bidan IoT

(Sumber: <https://www.comparitech.com/internet-providers/iot-statistics/>)

Secara ideal internet of things atau IoT adalah sistem perangkat komputasi yang saling terkait, misalnya mesin mekanika dan digital, benda, hewan atau orang yang dilengkapi dengan pengidentifikasi unik (UIDs - *Unique Identifier*) yang memiliki kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan interaksi manusia ke manusia atau manusia ke komputer. Penerapan “Things” dalam Internet of Things dapat berupa seseorang dengan implan monitor jantung, hewan ternak dengan transponder biochip, mobil yang memiliki sensor bawaan untuk memberitahu pengemudi ketika tekanan ban rendah atau normal atau objek buatan manusia lainnya yang dapat diberi alamat Internet Protocol (IP) dan dapat mentransfer data melalui jaringan.

Saat ini makin meningkat organisasi di berbagai jenis industri yang menggunakan IoT untuk mendukung efisiensi operasi, bagaimana agar lebih memahami pelanggan untuk meningkatkan layanan bagi pelanggan, membantu pengambilan keputusan dan meningkatkan nilai bisnis.

Interoperabilitas adalah salah satu aspek kunci dari IoT, berkontribusi terhadap popularitasnya yang semakin meningkat.

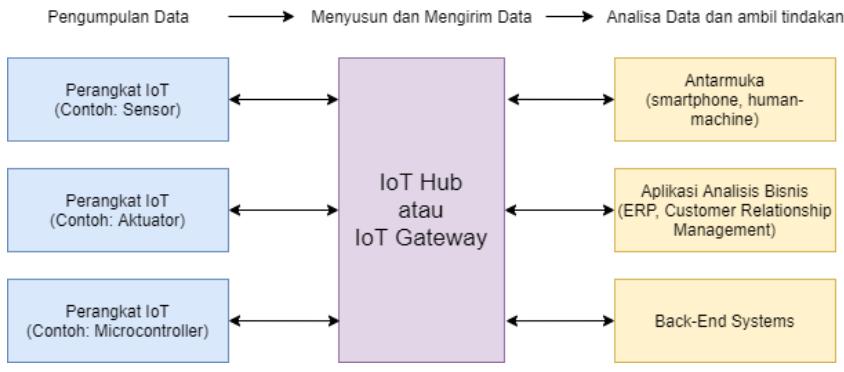
Perangkat yang terhubung ("things" dalam IoT) - memiliki kemampuan untuk mengumpulkan dan berbagi data dari lingkungan mereka dengan perangkat dan jaringan lain. Melalui analisis dan pemrosesan data, perangkat dapat melakukan fungsinya dengan sedikit atau tanpa perlu interaksi manusia.

Mengingat semakin banyaknya perangkat yang terhubung, IoT melanjutkan jalur evolusinya, menambahkan lapisan berbeda pada data yang sudah dibagikan dan diproses, dan memunculkan algoritma canggih yang menghasilkan peningkatan level otomatisasi. Karena beragamnya "things" yang dapat dihubungkan dengannya, IoT telah memungkinkan beragam aplikasi untuk dimanfaatkan oleh pengguna individu maupun industri.

1.2 Bagaimana IoT Bekerja

Ekosistem IoT terdiri dari perangkat pintar berkemampuan web yang menggunakan sistem tertanam (*embedded system*), seperti prosesor, sensor, dan perangkat keras komunikasi, untuk mengumpulkan, mengirim, dan bertindak berdasarkan data yang mereka peroleh dari lingkungan mereka. Perangkat IoT berbagi data sensor yang mereka kumpulkan dengan menghubungkan ke *gateway* IoT atau *edge device* lainnya, dimana data dikirim ke komputer *cloud* untuk dianalisis secara lokal. Terkadang, perangkat ini berkomunikasi dengan perangkat terkait lainnya dan bertindak berdasarkan informasi yang mereka dapatkan satu sama lain. Sebagian besar perangkat melakukan pekerjaan tanpa campur tangan manusia, meskipun orang masih dapat berinteraksi dengan perangkat tersebut, misalnya ketika saat mengkonfigurasinya, memberi mereka instruksi atau mengakses data. Konektivitas, jaringan dan protokol komunikasi yang digunakan pada perangkat yang mendukung web ini sangat tergantung pada spesifikasi aplikasi IoT yang digunakan.

IoT juga dapat menggunakan kecerdasan buatan (*artificial intelligence*) dan pembelajaran mesin (*machine learning*) untuk membantu membuat proses pengumpulan data lebih mudah dan lebih dinamis.



Ekosistem IoT Secara Umum

Perangkat IoT mengandung komponen-komponen berikut agar dapat beroperasi dengan benar sebagai bagian dari sistem IoT. Meskipun tidak semua komponen harus terpenuhi secara keseluruhan, antara lain:

- **Sensor**
Data pertama kali dikumpulkan dari lingkungan untuk sistem IoT dan mulai diproses. Data dikumpulkan oleh sensor dalam suatu perangkat yang dapat mengukur kejadian yang dapat diamati atau adanya perubahan pada lingkungan. Jenis data yang diukur oleh perangkat tergantung fungsinya, misalnya berupa denyut nadi seseorang berkaitan dengan pelacakan kebugaran atau jarak objek terdekat pada kendaraan otomotif.
- **Connection and Identification**
Data harus dikomunikasikan dari perangkat ke seluruh sistem IoT, baik itu ke komputer atau ke perangkat lain. Agar komunikasi ini memiliki makna, perangkat harus memiliki keberadaan unik yang dapat diidentifikasi di internet, yaitu alamat IP itu sendiri.
- **Actuator**
Sebagian besar perangkat IoT mampu melakukan fungsi utama mereka tanpa interaksi fisik dengan penggunanya. Perangkat IoT mampu mengambil tindakan berdasarkan data dari sensor mereka, kemudian memberi umpan balik melalui jaringan. Contohnya bola lampu pada "*smart home*" yang menyala atas perintah penggunaanya, bahkan ketika pengguna berada jauh dari lampu. Dengan cara yang sama, katup di "*smart*

manufacture" secara otomatis membuka atau menutup sesuai dengan data yang dikumpulkan oleh sensor pada sepanjang jalur produksi.

- **IoT Gateway**

Gateway IoT bertindak sebagai jembatan data dari berbagai perangkat untuk mencapai komputer *cloud*. Hal ini juga membantu menerjemahkan protokol yang berbeda dari berbagai perangkat IoT menjadi hanya satu protokol standar dan menyaring data tidak perlu yang dikumpulkan oleh perangkat.

- **Komputer Cloud**

Komputer Cloud adalah tempat semua data dari berbagai perangkat dikumpulkan dan tempat perangkat lunak dapat mencapai data ini untuk diproses. Karena sebagian besar pemrosesan data terjadi di cloud.

- **User Interface**

User interface (antarmuka aplikasi) berkomunikasi dengan pengguna data (yang telah dikumpulkan oleh perangkat) sehingga memungkinkan bagi *user* membuat perintah yang diperlukan untuk dieksekusi oleh perangkat.

Terdapat empat model konektivitas pada perangkat IoT:

- **Device-to-Device**

Model ini menggambarkan bagaimana dua atau lebih perangkat terhubung dan berkomunikasi secara langsung satu sama lain. Komunikasi antar perangkat biasanya dicapai melalui protokol, seperti Bluetooth, Z-Wave dan Zigbee. Model ini sering ditemukan pada perangkat *wearable* (perangkat yang dipakai di baju/badan, misalnya *smart watch*, *medical devices*, dan lain-lain) dan perangkat otomasi rumah, dimana paket data yang kecil dikomunikasikan dari satu perangkat ke perangkat lainnya.

- **Device-to-Cloud**

Banyak perangkat IoT terhubung ke cloud, sering kali dengan menggunakan antarmuka jaringan Ethernet atau Wi-Fi. Menghubungkan ke cloud memungkinkan bagi pengguna dan aplikasi terkait untuk mengakses perangkat, memungkinkan

untuk melakukan perintah secara jarak jauh serta melakukan *push update software* perangkat bila diperlukan.

- **Device-to-Gateway**
Sebelum terhubung ke cloud, perangkat IoT dapat berkomunikasi terlebih dahulu dengan perangkat gateway perantara. Gateway dapat menerjemahkan protokol dan menambahkan lapisan keamanan tambahan untuk seluruh sistem IoT. Dalam kasus *smart home*, misalnya, semua perangkat pintar dapat dihubungkan ke hub (gateway) yang membantu perangkat berbeda untuk bekerja bersama meskipun memiliki protokol koneksi yang berbeda.
- **Back-End Data-Sharing**
Adalah perpanjangan dari model *device-to-cloud*, model ini memungkinkan pengguna untuk mendapatkan akses ke dan menganalisis koleksi data dari *smart devices* berbeda. Perusahaan, misalnya, dapat menggunakan model ini untuk mengakses informasi dari semua perangkat yang bekerja di dalam gedung perusahaan sebagaimana diatur bersama di dalam cloud. Model ini juga membantu mengurangi masalah dengan portabilitas data.

1.3 Mengapa IoT Penting

Internet of Things membantu manusia hidup dan bekerja lebih cerdas, serta mendapatkan kontrol penuh atas kehidupan mereka. Selain menawarkan perangkat pintar untuk mengotomatisasi rumah, IoT sangat penting untuk bisnis. IoT memberikan bisnis waktu nyata (*real-time*) melihat ke dalam bagaimana sistem mereka benar-benar bekerja, memberikan wawasan tentang segala sesuatu mulai dari kinerja mesin hingga rantai pasokan dan operasi logistik. Kemampuan IoT memungkinkan perusahaan untuk mengotomatisasi proses dan mengurangi biaya tenaga kerja. Ini juga mengurangi pemborosan dan meningkatkan pengiriman layanan, membuatnya lebih murah untuk memproduksi dan mengirimkan barang, serta menawarkan transparansi dalam transaksi pelanggan. Dengan demikian, IoT adalah salah satu teknologi terpenting dalam kehidupan sehari-hari, dan IoT akan

terus meningkat seiring semakin banyaknya bisnis yang menyadari potensi perangkat yang terhubung agar tetap kompetitif.

1.4 Keuntungan IoT Bagi Organisasi

Internet of Things menawarkan beberapa manfaat bagi organisasi. Beberapa manfaat hanya khusus untuk industri tertentu dan beberapa lainnya berlaku untuk berbagai macam industri. Beberapa manfaat umum dari kemampuan IoT bagi bisnis antara lain:

- Memantau keseluruhan proses bisnis mereka
- Meningkatkan pengalaman pelanggan (customer experience)
- Menghemat waktu dan biaya
- Meningkatkan produktivitas karyawan
- Integrasi dan adaptasi model bisnis
- Membuat keputusan bisnis yang lebih baik
- Menghasilkan lebih banyak pendapatan

IoT mendorong perusahaan untuk berpikir ulang bagaimana cara mereka melakukan pendekatan bisnis dan memberi alat untuk meningkatkan strategi bisnis mereka. Saat ini, IoT paling banyak diterapkan pada organisasi manufaktur, transportasi dan utilitas, tidak menutup kemungkinan untuk bidang pertanian, infrastruktur dan industri otomasi rumah, yang mana IoT telah memimpin beberapa organisasi menuju transformasi digital.

Pada kasus lain, IoT dapat menguntungkan petani di bidang pertanian (agriculture) karena menjadikan pekerjaan mereka lebih mudah. Dengan sensor, dapat dikumpulkan data tentang curah hujan, kelembaban, suhu, kandungan tanah, serta faktor-faktor lain yang akan membantu mengotomatiskan teknik bercocok tanam.

Kemampuan memantau operasi infrastruktur di sekitarnya juga merupakan faktor yang dapat dibantu oleh IoT. Sensor dapat digunakan untuk memantau peristiwa atau perubahan dalam bangunan struktural, jembatan dan infrastruktur lainnya. Ini akan membawa manfaat seperti penghematan biaya dan waktu,

perubahan alur kerja menjadi lebih berkualitas dan alur kerja tanpa kertas.

Bisnis otomasi rumah dapat memanfaatkan IoT untuk memantau dan memanipulasi sistem mekanik dan listrik di sebuah gedung. Dalam skala yang lebih luas, kota pintar (*smart city*) dapat membantu warga mengurangi limbah dan konsumsi energi.

Kesimpulannya, IoT menyentuh setiap industri, termasuk bisnis di bidang kesehatan, keuangan, ritel, dan manufaktur.

1.5 Pro dan Kontra Teknologi IoT

Beberapa keuntungan IoT meliputi:

- Kemampuan untuk mengakses informasi dari mana saja kapan saja dari perangkat apa pun.
- Peningkatan komunikasi antar perangkat elektronik yang terhubung.
- Mentransfer paket data melalui jaringan yang terhubung akan menghemat waktu dan biaya.
- Mengotomasi tugas-tugas untuk membantu meningkatkan kualitas layanan bisnis dan mengurangi kebutuhan intervensi manusia.

Beberapa kelemahan IoT meliputi:

- Dengan meningkatnya jumlah perangkat yang terhubung dan lebih banyaknya informasi yang dibagikan antar perangkat maka potensi pencurian informasi rahasia oleh peretas juga meningkat.
- Perusahaan pada akhirnya berurusan dengan sejumlah besar - mungkin bahkan jutaan - perangkat IoT, mengumpulkan dan mengelola data dari semua perangkat tersebut akan menjadi tantangan.
- Jika terdapat *bug* dalam sistem maka kemungkinan setiap perangkat yang terhubung akan rusak.
- Karena tidak ada standar internasional mengenai kompatibilitas IoT maka sulit bagi perangkat dari produsen berbeda untuk berkomunikasi satu sama lain.

1.6 Isu Berkaitan Dengan IoT

IoT adalah teknologi yang relatif baru dan berkembang. Berikut ini adalah beberapa aspek dimana IoT terus menghadapi beberapa masalah.

- **Standar dan Regulasi**

Semakin banyaknya perangkat yang terhubung membuat standarisasi dan regulasi IoT menjadi urusan yang rumit. Masalah standardisasi dan regulasi berkisar dari masalah teknis hingga masalah hukum. Sebagai contoh, perangkat pintar berbeda dapat menggunakan berbagai protokol komunikasi nirkabel berbeda pula, misalnya Bluetooth, Wi-Fi, Zigbee, dan 5G yang menghambat komunikasi dalam sistem IoT. Di sisi lain, kurangnya regulasi yang menyoroti isu terkait internet saat ini akan menambah lapisan kerumitan lain untuk isu ini. Salah satu contoh adalah menentukan akuntabilitas: Jika ada cacat dan pelanggaran terkait penggunaan perangkat IoT, kurangnya regulasi membuat akuntabilitas sulit ditentukan. Standar dan peraturan memengaruhi kualitas layanan keseluruhan yang diberikan teknologi IoT, oleh karena itu menjadi perhatian semua pemangku kepentingan IoT, baik itu pengguna individu, produsen perangkat, atau organisasi yang mengintegrasikan teknologi ke dalam proses mereka.

- **Privasi**

Kesadaran privasi telah tumbuh seiring meningkatnya keragaman informasi pribadi yang dibagikan melalui internet. Dengan adanya IoT maka akan semakin rumit masalah ini karena luasnya jenis data yang direkam dan dibagikan melalui internet. Karena IoT bekerja lebih baik dengan mendapatkan data sedetil mungkin tentang lingkungan, terjadilah *trade-off* (untung rugi) antara privasi pengguna dan kualitas layanan. Menentukan titik-titik dimana pengumpulan data harus dibatasi, atau bahkan menghentikan pengumpulan data sekaligus karena masalah privasi pengguna juga sulit dicapai, apalagi sebagian besar sistem IoT bersifat otomatis.

- **Keamanan (Security)**

Masalah keamanan akan selalu muncul ketika menangani data dan informasi yang terlibat. IoT menambahkan tantangan

keamanannya sendiri dengan aksesnya ke berbagai informasi pribadi dan integrasinya yang erat ke dalam kegiatan individu dan organisasi. Karakteristik IoT ini menjadikan teknologi yang layak bagi penjahat cyber untuk ditarget. Selain itu, setiap pelanggaran, serangan, dan kerentanan pada satu perangkat IoT atau sistem akan melemahkan keamanan seluruh jaringan yang bersangkutan.

Ancaman keamanan lain yang terkait dengan teknologi IoT meliputi:

- Homogenitas perangkat pintar (*smart devices*) yang diproduksi secara massal, berarti penyebaran terhadap kerentanan yang mungkin sama bisa terjadi.
- Otomasi sistem IoT membuatnya lebih sulit untuk mendeteksi kerentanan dan pelanggaran karena kurangnya campur tangan manusia pada sistem.
- Lingkungan dimana perangkat IoT dikerahkan membuat perangkat ini rentan terhadap ancaman fisik yang tidak terduga, penyerang dapat merusak perangkat secara langsung.
- Interkoneksi sistem IoT menjadikan setiap bagian dari sistem menjadi jalan pelanggaran data dan serangan cyber, yang mana dapat menyebar ke seluruh jaringan yang terkena dampak.

BAB 2

Arsitektur Internet of Things

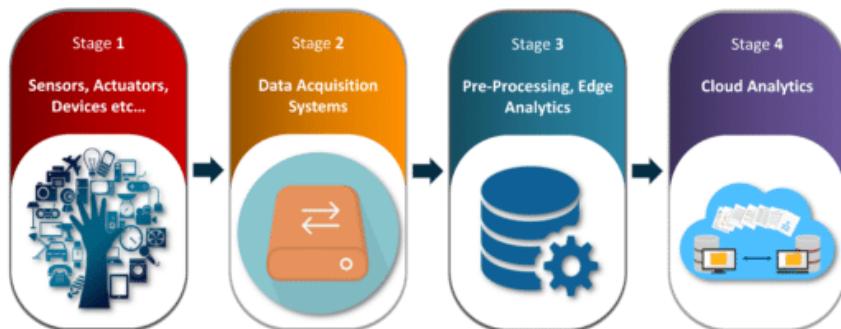
2.1 Building Block Internet of Things

Seperti yang diulas pada bab 1, Internet of Things (IoT) adalah jaringan perangkat yang dapat merasakan, mengakumulasi, dan mentransfer data melalui internet tanpa campur tangan manusia.

Apa yang membuat perangkat seperti hidup? Mereka bisa merasakan dan berkomunikasi satu sama lain. Bayangkan jika benda mati bisa merasakan dan berinteraksi satu sama lain tanpa campur tangan manusia tentu hal itu menakjubkan. Ini adalah konsep yang mendasari Internet of Things.

Sekarang Anda mungkin telah memahami bahwa IoT bukan hanya perangkat konsumen yang terhubung ke Internet. Faktanya, IoT adalah teknologi yang membangun sistem yang mampu merasakan dan merespons rangsangan secara mandiri dari dunia nyata tanpa campur tangan manusia. Oleh karena itu, kami perlu mengembangkan aliran proses untuk kerangka kerja yang pasti dimana solusi IoT dibangun.

Arsitektur IoT umumnya terdiri dari 4 tahap berikut:



IoT Architecture Building Block

(Sumber: <https://www.edureka.co/blog/what-is-iot/>)

Tahap 1 (Sensor / Aktuator):

Suatu hal dalam konteks “Internet of Things”, harus dilengkapi dengan sensor dan aktuator sehingga memberikan kemampuan untuk memancarkan, menerima dan memproses sinyal.

Tahap 2 (Sistem Akuisisi Data / Data Acquisition Systems):

Data dari sensor dimulai dalam bentuk analog yang perlu dikumpulkan dan diubah menjadi aliran digital untuk diproses lebih lanjut. Sistem akuisisi data melakukan fungsi agregasi dan konversi data ini.

Tahap 3 (Analisis Tepi / Edge Analytics):

Setelah data IoT diubah menjadi digital dan dikumpulkan, mungkin diperlukan pemrosesan lebih lanjut sebelum memasuki pusat data, di sinilah Edge Analytics masuk.

Tahap 4 (Cloud Analytics):

Data yang membutuhkan pemrosesan lebih mendalam diteruskan ke pusat data fisik atau sistem berbasis cloud.

2.2 MQTT Messaging Protocol

Tabel di bawah ini merupakan beberapa spesifikasi protokol pesan yang umum diterapkan pada platform Internet of Things

	DDS	AMQP	CoAP	MQTT	REST /HTTP	XMP
Transport	UDP/IP (unicast + multicast) TCP/IP	TCP/ IP	UDP/ IP	TCP/ IP	TCP/ IP	TCP/ IP
Interaction Model	Publish-and-Subscrib e, Request-Replay	Point-to-Point Message Exchange	Request-Replay (REST)	Publish-and-Subscrib e	Requ est-Repla y	Point-to-Point Messa ge

	DDS	AMQP	CoAP	MQTT	REST /HTTP	XMP
						Exch ange
Scope	Device-to-Device Device-to-Cloud Device-to-Cloud Cloud-to-Cloud	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud
Automatic Discovery	✓	-	✓	-	-	-
Content Awareness	Content-based Routing Queries	-	-	-	-	-
QoS	Extensive (20+)	Limited	Limited	Limited	-	-
Interoperability Level	Semantic	Structural	Semantic	Foundational	Semantic	Structural
Security	TLS, DTLS, DOS Security	TLS + SASL	DTLS	TLS	HTTP S	TLS + SASL
Data Prioritization	Transport Priorities	-	-	-	-	-
Fault Tolerance	Decentralized	Implementation-Specific	Decentralized	Broker is SpoF	Server is SpoF	Server is SpoF

Tabel Protocol Transport IoT
(Sumber <https://iot.eclipse.org/>)

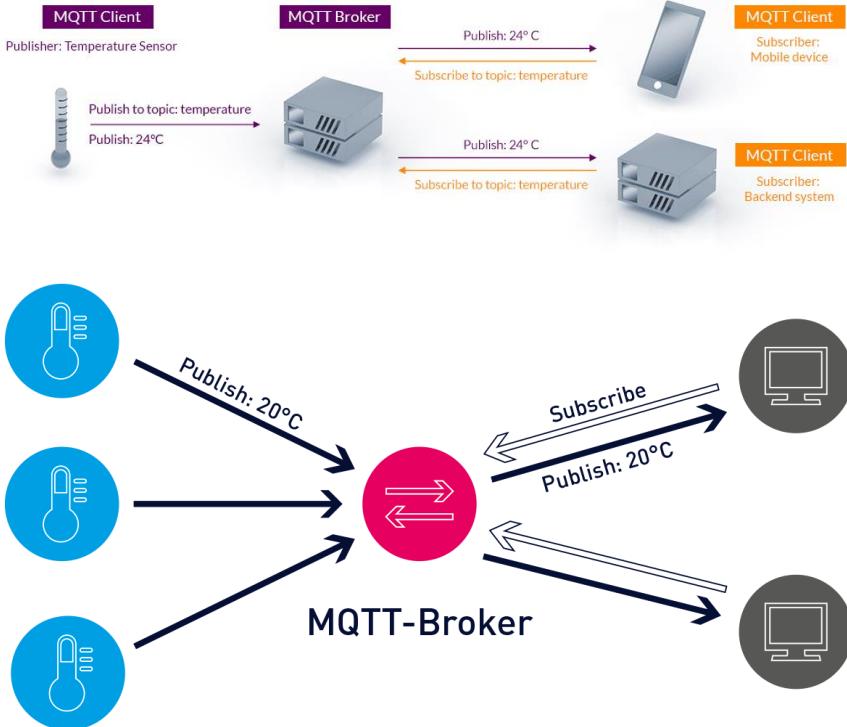
Seperti tampak pada tabel di atas, terdapat beberapa jenis protokol transport yang dapat diterapkan pada arsitektur IoT. Namun pada kesempatan ini kita hanya membahas protokol pesan MQTT (*Message Queuing Telemetry Transport*) karena MQTT merupakan protokol pesan terdepan dan paling banyak digunakan pada ekosistem IoT saat ini.

MQTT awalnya dibuat oleh Dr. Andy Stanford-Clark dan Arlen Nipper pada tahun 1999. Tujuan asli dari metode komunikasi ini adalah agar memungkinkan perangkat pemantauan yang digunakan dalam industri minyak dan gas untuk mengirim datanya ke server jarak jauh. Dalam banyak kasus, perangkat pemantauan seperti itu digunakan pada lokasi terpencil dimana segala jenis jalur darat, koneksi kabel, atau koneksi transmisi radio akan sulit, atau tidak mungkin untuk dibuat. Pada saat itu, satu-satunya pilihan untuk kasus semacam itu adalah komunikasi satelit yang sangat mahal karena dihitung berdasarkan seberapa banyak data yang digunakan. Dengan ribuan sensor di lapangan, industri membutuhkan bentuk komunikasi yang dapat memberikan data cukup handal untuk digunakan dengan menggunakan bandwidth minimal.

MQTT distandarisasi sebagai open source di bawah Organization for the Advancement of Structured Information Standards (OASIS) pada tahun 2013. OASIS masih mengelola standar MQTT.

MQTT berjalan di atas TCP / IP menggunakan topologi PUSH / SUBSCRIBE. Dalam arsitektur MQTT, terdapat dua jenis sistem: klien dan broker. Broker adalah server yang berkomunikasi dengan klien. Broker menerima komunikasi dari klien dan mengirimkan komunikasi tersebut ke klien lain. Klien tidak berkomunikasi secara langsung satu sama lain, melainkan terhubung ke broker. Setiap klien bisa jadi publisher/penerbit, subscriber/pelanggan, atau keduanya.

MQTT adalah protokol berbasis event-driven/peristiwa. Tidak ada transmisi data berkala atau berkelanjutan. Ini meminimalkan transmisi. Klien hanya mempublikasikan jika ada informasi yang akan dikirim, dan broker hanya mengirimkan informasi ke pelanggan (subscriber) ketika data baru tiba.



Arsitektur Publish/Subscribe MQTT
 (Sumber: <https://mqtt.org>)

Cara lain MQTT meminimalkan transmisinya adalah dengan mengkonstruksi pesan kecil (*small message*) yang ditentukan dengan ketat. Setiap pesan memiliki header tetap hanya 2 byte. Header opsional dapat digunakan tetapi akan meningkatkan ukuran pesan. Muatan pesan dibatasi hanya 256 MB.

Dengan tiga tingkat *Quality of Service* (QoS) yang berbeda memungkinkan perancang jaringan untuk memilih antara meminimalkan transmisi data dan memaksimalkan keandalan.

- QoS 0 - Menawarkan jumlah minimum transmisi data. Dengan level ini, setiap pesan dikirim ke pelanggan satu kali tanpa konfirmasi. Tidak ada cara untuk mengetahui apakah pelanggan menerima pesan tersebut. Metode ini terkadang disebut sebagai "*fire and forget*" atau "*at most once delivery*".

Karena tingkat ini mengasumsikan bahwa pengiriman selesai, pesan tidak disimpan untuk pengiriman ke klien terputus yang kemudian tersambung kembali.

- QoS 1 - Broker mencoba untuk menyampaikan pesan dan kemudian menunggu tanggapan konfirmasi dari subscriber. Jika konfirmasi tidak diterima dalam jangka waktu tertentu, pesan dikirim kembali. Dengan menggunakan metode ini, subscriber dapat menerima pesan lebih dari sekali jika broker tidak menerima pengakuan subscriber tepat waktu. Ini terkadang disebut sebagai "*at least once delivery*".
- QoS 2 - Klien dan broker menggunakan jabat tangan empat langkah untuk memastikan bahwa pesan diterima, dan hanya diterima sekali. Ini terkadang disebut sebagai "*exactly once delivery*".

QoS 0 mungkin merupakan pilihan terbaik untuk situasi dimana komunikasi dapat diandalkan tetapi terbatas. QoS 2 akan menjadi pilihan terbaik untuk situasi dimana komunikasi tidak dapat diandalkan, tetapi koneksi tidak terbatas pada sumberdaya. QoS 1 menyediakan solusi terbaik dari kedua pilihan di atas, tetapi mengharuskan aplikasi yang menerima data mengetahui cara menangani duplikasi.

Untuk QoS 1 dan QoS 2, pesan untuk klien yang dalam kondisi offline namun masih memiliki sesi persisten maka akan disimpan atau diantrikan. Jika klien kembali online, pesan ini dikirim ulang sesuai dengan level QoS yang diperlukan.

Pesan dalam MQTT dipublikasikan sebagai topik. Topik adalah struktur dalam hierarki yang menggunakan karakter garis miring (/) sebagai pembatas. Struktur ini menyerupai pohon direktori pada sistem file komputer. Struktur seperti "*sensor/OilandGas/Pressure*" memungkinkan subscriber untuk menentukan bahwa itu hanya boleh dikirim data dari klien yang mempublikasikan ke topik "*Pressure*", atau untuk pandangan yang lebih luas, bisa jadi semua data dari klien yang mempublikasikan ke *sensor/* topik *OilandGas*. Topik tidak dibuat secara eksplisit di MQTT. Jika broker menerima data yang dipublikasikan ke topik

yang saat ini tidak ada, topik tersebut dibuat begitu saja, dan klien dapat berlangganan ke topik baru.

Untuk menjaga footprint kecil, pesan yang diterima tidak disimpan pada broker kecuali mereka ditandai dengan bendera (flag/penanda) yang dipertahankan. Ini disebut pesan yang dipertahankan. Pengguna yang ingin menyimpan pesan yang diterima perlu menyimpannya di tempat lain di luar protokol MQTT. Ada satu pengecualian.

Sebagai protokol yang digerakkan oleh event/peristiwa, sangat mungkin, bahwa *subscriber* menerima sangat sedikit pesan untuk topik tertentu, bahkan dalam jangka waktu yang lama. Dalam struktur topik yang disebutkan sebelumnya, mungkin pesan ke topik *Pressure* hanya dikirim ketika sensor mendeteksi bahwa tekanan telah melebihi jumlah tertentu. Dengan asumsi bahwa apa pun yang dipantau oleh sensor itu tidak sering gagal, mungkin perlu berbulan-bulan, atau bahkan bertahun-tahun sebelum klien menerbitkan pesan ke topik itu.

Untuk memastikan bahwa *subscriber* baru menerima pesan dari suatu topik, broker dapat menyimpan pesan terakhir yang dikirim ke setiap topik. Ini disebut pesan yang dipertahankan. Setiap kali klien baru berlangganan ke suatu topik atau ketika klien yang ada kembali online, pesan yang dipertahankan dikirim ke pelanggan, dengan demikian memastikan bahwa langganan tersebut aktif, dan memiliki informasi terbaru.

Catatan

Bagaimana penggunaan protokol message MQTT pada platform IoT menurut penulis sebaiknya dibahas pada buku IoT tingkat lanjut karena untuk mempraktekannya perlu membangun infrastruktur IoT secara “create from scratch”. Kali ini kita hanya membahas IoT secara “best practie” untuk memberi pandangan secara umum IoT bagi pemula.

BAB 3

Microcontroller Unit

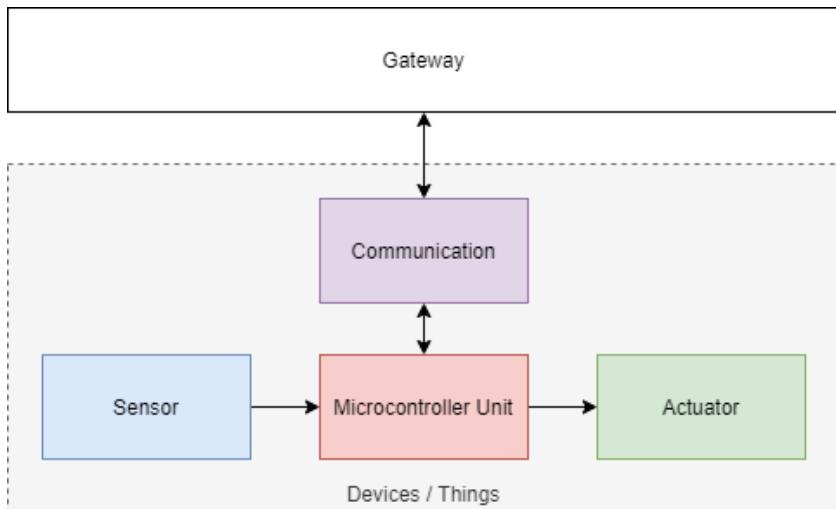
3.1 NodeMCU ESP-12E Sebagai “Things”

3.1.1 Smart Devices (“Things”)

Devices atau “things” umumnya berkaitan dengan pengambilan data yang bersumber dari lingkungan secara *real-time*. Tugas ini umumnya diemban oleh perangkat MCU (*Microcontroller Unit* - meskipun tidak selalu demikian) yang bertindak sebagai pengendali dan pemroses data. Untuk mendapatkan data dari lingkungan secara waktunya nyata, MCU dibantu oleh sensor. Sebaliknya, MCU juga dapat mengendalikan perangkat lain yang diistilahkan sebagai aktuator.

Definisi Sensor sendiri adalah alat yang berfungsi mengubah besaran fisis menjadi besaran listrik. Besaran fisis didapatkan dari fenomena di lingkungan, seperti panas, suara, intensitas cahaya, tekanan, kemiringan, magnetis, kelembaban dan gravitasi. Besaran fisis yang didapatkan oleh sensor diolah dan menghasilkan sinyal listrik berupa sinyal digital atau sinyal analog.

Sedangkan definisi Aktuator adalah alat yang berfungsi mengubah besaran listrik menjadi besaran fisis. Besaran fisis yang dimaksud adalah gerakan, cahaya, panas, tekanan atau magnetis. Aktuator menjadi bagian penting dari sistem kendali karena menentukan pergerakan dari sebuah proses. Aktuator digerakkan dengan kontrol dari mikrokontroller, komputer atau yang lainnya.



Hubungan Mikrokontroller, Sensor, Aktuator dan perangkat Komunikasi

Berikut ini penjelasan singkat hubungan tiap bagian antara mikrokontroller, sensor, aktuator dan perangkat komunikasi seperti tampak pada gambar di atas:

- Sensor adalah komponen elektronika yang berfungsi untuk mengubah besaran fisik (mekanis, magnetis, panas, sinar, dan kimia) menjadi besaran elektrik berupa tegangan, resistansi dan arus listrik. Data dari sensor kemudian diteruskan ke MCU untuk diproses lebih lanjut. MCU selain bertugas untuk memproses data juga melakukan konversi dari sinyal analog menjadi digital (proses ADC – Analog to Digital Converter).
- Mikrokontroler adalah sebuah chip yang berfungsi sebagai pengontrol rangkaian elektronik dan umumnya dapat menyimpan program di dalamnya, atau bersifat *embedded system*. Mikrokontroler terdiri dari CPU (*Central Processing Unit*), memori, I/O tertentu dan unit pendukung seperti *Analog-to-Digital Converter* (ADC) yang sudah terintegrasi didalamnya. Kelebihan utama dari mikrokontroler ialah tersedianya RAM dan peralatan I/O pendukung sehingga ukuran board mikrokontroler menjadi sangat ringkas.
- Aktuator adalah peralatan piranti keras yang mengubah sinyal perintah kontroler ke dalam parameter fisik yang biasanya

berupa mekanik, seperti perubahan posisi atau perubahan kecepatan. Pada dasarnya aktuator adalah *transduscer* karena merubah satu besaran fisik misalnya arus listrik kedalam besaran listrik fisik yang lain misalnya kecepatan rotasi motor listrik.

3.1.2 Jenis-jenis Microcontroller

Banyak jenis-jenis board atau modul mikrokontroler yang tersedia. Misalnya mikrokontroler berbasis chip AVR, ESP-XX (Expressive), MCS-51, PIC, ARM, dan lain-lain. Semuanya memiliki kelebihan dan kekurangan, baik ditinjau dari sisi harga, jumlah pin input/output, besaran memori, kecepatan pemrosesan dan lain-lain. Oleh karena itu pilihlah board mikrokontroler yang paling sesuai dengan kebutuhan Anda.

Berikut ini salah satu contoh perbandingan spesifikasi MCU seri ATmega:

Seri	Flash [kbytes]	RAM [bytes]	EEPROM [kbytes]	Pin I/O	Timer 16-bit	Timer 8-bit	UART	PWM	ADC 10-bit	SPI	ISP
ATmega8	8	1024	0.5	23	1	1	1	3	6/8	1	Ya
ATmega8535	8	512	0.5	32	2	2	1	4	8	1	Ya
ATmega16	16	1024	0.5	32	1	2	1	4	8	1	Ya
ATmega162	16	1024	0.5	35	2	2	2	6	8	1	Ya
ATmega32	32	2048	1	32	1	2	1	4	8	1	Ya
ATmega128	128	4096	4	53	2	2	2	8	8	1	Ya
ATTiny12	1	-	0.0625	6	-	1	-	-	-	-	Ya
ATTiny2313	2	128	0.125	18	1	1	1	4	-	1	Ya
ATTiny44	4	256	0.25	12	1	1	-	4	8	1	Ya
ATTiny84	8	512	0.5	12	1	1	-	4	8	1	Ya

Perbandingan spesifikasi mikrokontroler keluarga chip AVR

Di bawah ini merupakan sedikit contoh papan mikrokontroler berbasis Arduino dan Expressive/ESP.



Arduino Uno



Arduino Nano



Arduino Mega



Expressif ESP-01

Expressif ESP-12E

Expressive ESP-32

3.1.3 Antarmuka Komunikasi Microcontroller

Istilah antarmuka (*interfacing*) dalam mikrokontroler merupakan pin masukan atau pin luaran yang dapat berkomunikasi dengan perangkat eksternal lainnya. Dengan *interfacing* menjadikan mikrokontroler dapat saling bertukar data dengan sensor, aktuator, mikrokontroler, atau perangkat lainnya. Protokol komunikasi di kedua sisi tentunya harus disepakati jenisnya.

Terdapat beberapa protokol interface yang umum dipakai pada mikrokontroler, antara lain SPI (*Serial Peripheral Interface*), I₂C (*Inter-Integrated Circuit*), UART (*Universal Asynchronous Receiver-Transmitter*) dan masih banyak lagi tentunya. Dalam buku ini kita tidak membahas secara detil spesifikasi interface tersebut, Anda dapat menggalinya lebih dalam dari sumber-sumber lain karena topik tersebut cenderung ke bidang elektronika. Kita cukup mengetahui jenis interface yang akan digunakan dan bagaimana mempraktekkannya pada platform IoT.

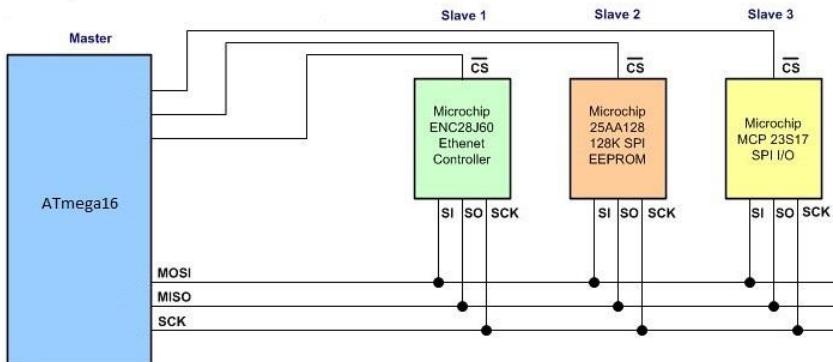
Berikut ini penjelasan singkat tentang ketiga interface tersebut:

- SPI (*Serial Peripheral Interface*)

Serial Peripheral Interface (SPI) adalah spesifikasi antarmuka komunikasi serial sinkron yang digunakan untuk komunikasi jarak pendek, terutama dalam sistem tertanam (*embedded systems*). Antarmuka dikembangkan oleh Motorola pada pertengahan 1980-an dan telah menjadi standar de facto.

Perangkat SPI berkomunikasi dalam mode *full-duplex* menggunakan arsitektur master-slave dengan satu master. Perangkat master menghasilkan bingkai untuk membaca dan

menulis. Beberapa perangkat budak didukung melalui pemilihan *slave select* secara individu (SS), kadang-kadang disebut *chip select* (CS), garis.



Implementasi Interface SPI antara Master dan Slave

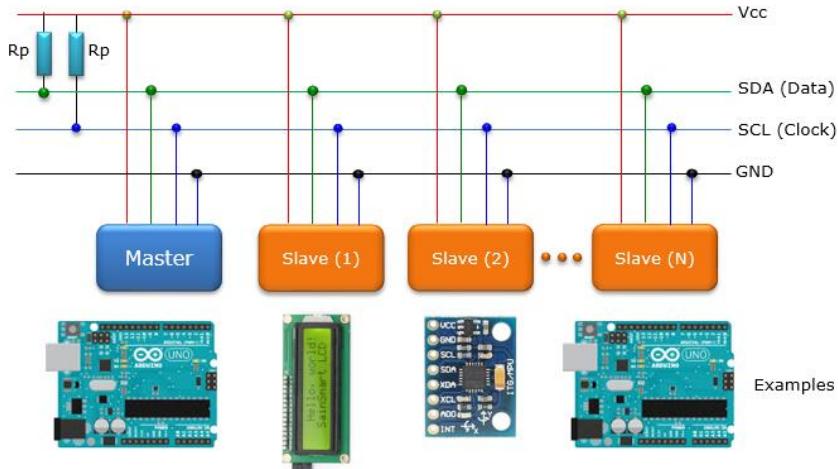
(Sumber: <http://christianto.tjahyadi.com/belajar-mikrokontroler/menggunakan-serial-peripheral-interface.html>)

- **I₂C (Inter-Integrated Circuit)**

I₂C adalah singkatan dari Inter-Integrated Circuit, sebuah protokol komunikasi serial yang dibuat oleh Philips Semiconductor (sekarang menjadi NXP Semiconductor). Itu dibuat dengan tujuan komunikasi antara chip berada di Papan Sirkuit Cetak (PCB) yang sama. Biasanya digunakan untuk menghubungkan IC kecepatan lambat ke mikroprosesor atau mikrokontroler. Ini adalah protokol master-slave, biasanya prosesor atau mikrokontroler adalah master dan chip lain seperti RTC, Sensor Suhu, EEPROM akan menjadi budak. Kita dapat memiliki banyak master dan banyak budak di bus I₂C yang sama. Oleh karena itu, I₂C adalah protokol multi-master, multi-slave. I₂C hanya membutuhkan dua kabel untuk bertukar data dan ground sebagai referensi, yaitu SDA (Serial Data), SCL (Serial Clock), dan GND (Ground).

Master adalah perangkat yang selalu memulai komunikasi dan menggerakkan Clock Line (SCL). Biasanya mikrokontroler atau mikroprosesor bertindak sebagai master yang perlu membaca data dari atau menulis data ke perangkat pendukung.

Perangkat Slave selalu merespon master dan tidak akan memulai komunikasi apa pun dengan sendirinya. Biasanya perangkat seperti EEPROM, LCD, RTC bertindak sebagai perangkat pendukung. Setiap perangkat slave akan memiliki alamat unik sehingga master dapat meminta data dari atau menulis data ke dalamnya.



Implementasi Interface I₂C antara Master dan Slave

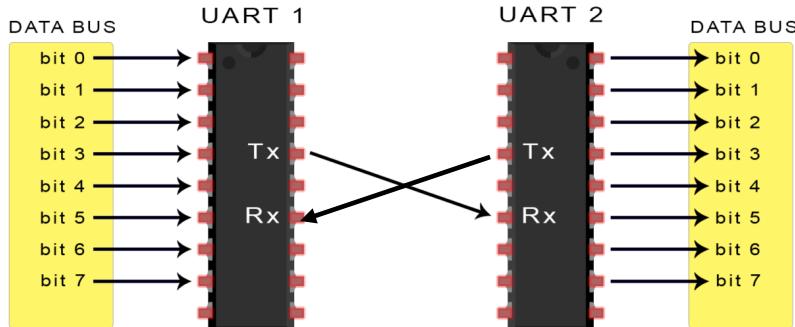
(Sumber: <https://2.bp.blogspot.com/>)

- **UART (*Universal Asynchronous Receiver-Transmitter*)**

Universal Asynchronous Receiver-Transmitter adalah perangkat komunikasi serial asinkron dimana format data dan kecepatan transmisi dapat dikonfigurasi. Level dan metode pensinyalan listrik ditangani oleh sirkuit driver di luar UART. UART biasanya merupakan individu (atau bagian dari) sirkuit terintegrasi (IC - integrated circuit) yang digunakan untuk komunikasi serial melalui komputer atau port serial perangkat periferal. Satu atau lebih perangkat UART biasanya terintegrasi dalam chip mikrokontroler.

Dalam komunikasi UART, dua UART berkomunikasi secara langsung satu sama lain. UART pemancar mengubah data paralel dari perangkat pengontrol seperti CPU menjadi bentuk serial, mentransmisikannya secara serial ke UART penerima, yang kemudian mengubah data serial kembali menjadi data

paralel untuk perangkat penerima. Hanya dua kabel yang dibutuhkan untuk mengirimkan data antara dua UART. Data mengalir dari pin Tx dari transmisi UART ke pin Rx dari UART penerima.



Komunikasi Antar IC UART

(Sumber: <https://www.circuitbasics.com/basics-uart-communication/>)

3.1.4 Microcontroller NodeMCU ESP-12E

Pada kesempatan ini kita akan membahas tentang mikrokontroler NodeMCU karena contoh-contoh projek yang akan dibuat nanti akan menggunakan mikrokontroller NodeMCU ESP-12E yang bertindak sebagai "Smart Devices / Things". Sedangkan jenis mikrokontroller lain seperti Arduino dan lain-lain dapat dipelajari dari sumber-sumber lain. Salah satu buku yang dapat menjadi rujukan adalah buku berjudul "Microcontroller Arduino Untuk Pemula (Disertai Contoh-contoh Projek Menarik)" yang diterbitkan oleh Jasakom.

NodeMCU (Node MicroController Unit) adalah perangkat lunak sumber terbuka (open source) dengan lingkungan pengembangan perangkat keras yang dibangun pada System-on-a-Chip (SoC) murah yang disebut ESP8266. ESP8266, dirancang dan diproduksi oleh Espressif Systems, berisi elemen penting dari komputer, antara lain: CPU, RAM, jaringan (WiFi), dan bahkan sistem operasi dan SDK modern. Itu menjadikannya pilihan yang sangat tepat untuk semua jenis proyek Internet of Things (IoT).

Namun, sebagai sebuah chip, ESP8266 juga sulit diakses dan digunakan bagi pemula. Anda harus menyolder kabel, dengan tegangan analog yang sesuai, ke pinnya untuk tugas-tugas paling sederhana seperti menyalakannya atau mengirim penekanan tombol ke "komputer" pada chip. Anda juga harus memprogramnya dalam instruksi mesin tingkat rendah yang dapat ditafsirkan oleh perangkat keras chip. Tingkat integrasi ini tidak menjadi masalah dengan menggunakan ESP8266 sebagai chip pengontrol tertanam dalam elektronik yang diproduksi secara massal. Ini merupakan beban besar bagi para penghobi, peretas, atau pelajar yang ingin bereksperimen dengannya dalam proyek IoT.

Berikut ini adalah spesifikasi NodeMCU dengan beberapa varian yang beredar:

	Official NodeMCU	NodeMCU Carrier Board	LoLin NodeMCU
Microcontroller	ESP-8266 32-bit	ESP-8266 32-bit	ESP-8266 32-bit
NodeMCU Model	Amica	Amica	Clone LoLin
NodeMCU Size	49mm x 26mm	49mm x 26mm	58mm x 32mm
Carrier Board Size	n/a	102mm x 51mm	n/a
Pin Spacing	0.9" (22.86mm)	0.9" (22.86mm)	1.1" (27.94mm)
Clock Speed	80 MHz	80 MHz	80 MHz
USB to Serial	CP2102	CP2102	CH340G
USB Connector	Micro USB	Micro USB	Micro USB
Operating Voltage	3.3V	3.3V	3.3V
Input Voltage	4.5V-10V	4.5V-10V	4.5V-10V
Flash Memory/SRAM	4 MB / 64 KB	4 MB / 64 KB	4 MB / 64 KB
Digital I/O Pins	11	11	11
Analog In Pins	1	1	1
ADC Range	0-3.3V	0-3.3V	0-3.3V
UART/SPI/I2C	1/1/2001	1/1/2001	1/1/2001
WiFi Built-In	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Temperature Range	-40C - 125C	-40C - 125C	-40C - 125C
Product Link		NodeMCU	NodeMCU

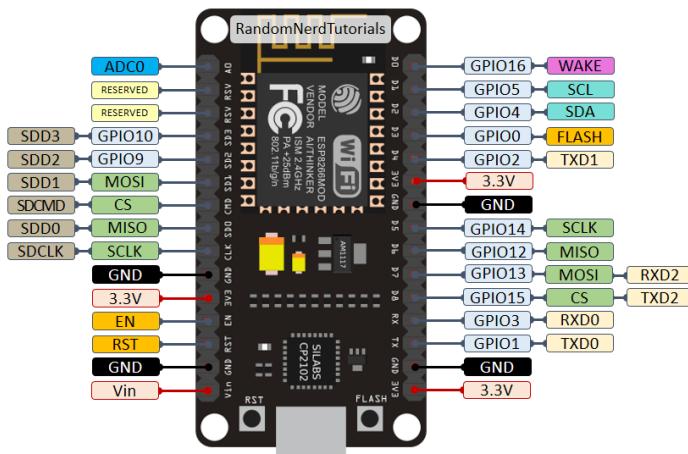


Papan Kit NodeMCU
ESP-12E Amica



Papan Kit NodeMCU
ESP-12E Lolin

Langkah selanjutnya agar dapat menggunakan mikrokontroler NodeMCU sebagai smart device atau “things” dalam IoT adalah memahami pin pada NodeMCU. Untuk mengetahui alamat pin dan fungsi-fungsi dasar pin dapat dilihat pada gambar di bawah. Pin-pin terdiri dari pin tegangan input (3,3 volt – 9 volt), pin tegangan output (3,3 volt), pin ground, pin ADC (analog to digital converter), pin komunikasi (I_2C , SPI, UART) dan pin GPIO (pin yang dapat difungsikan sebagai input atau output tergantung kode program yang menugaskan).



Peta pin Mikrokontroller NodeMCU ESP-12E
(Sumber: <https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>)

Pin-pin pada NodeMCU dapat dikustom sedemikian rupa sesuai kebutuhan projek yang akan dibangun. Pin yang semula berfungsi sebagai pin komunikasi dapat dialihkan menjadi pin input/output dan sebaliknya. Namun terdapat beberapa hal yang perlu diperhatikan bila Anda akan mengkustom pin dari fungsi utamanya, karena bila kita salah memfungsikannya akan menyebabkan aplikasi tidak berjalan dengan benar, misalnya sistem akan me-reset dengan sendirinya, sistem *hang* ditengah jalan sistem *reboot* sendiri, sensor tidak dapat membaca kondisi lingkungan luar dan sebagainya.

Tabel di bawah ini merupakan syarat dan kondisi penggunaan pin-pin pada NodeMCU.

- Label “OK” berwarna hijau menunjukkan bahwa pin aman digunakan sebagai input/output.
- Label “OK” berwarna kuning menunjukkan bahwa pin aman digunakan sebagai input/output, namun pada saat kondisi boot (NodeMCU dinyalakan pertamakali) akan bernilai HIGH.
- Label “pulled up” menunjukkan bahwa pin masih bisa digunakan sebagai input/output dengan syarat pin tersebut juga dihubungkan ke resistor, kemudian resistor dihubungkan ke VCC (tegangan positif).
- Label “pulled to ground” menunjukkan bahwa pin masih bisa digunakan sebagai input/output dengan syarat pin tersebut juga dihubungkan ke resistor, kemudian resistor dihubungkan ke Ground.
- Label berwarna merah menunjukkan pin tidak bisa digunakan sebagai input atau output namun tidak berlaku keduanya. Misalnya pin RX tidak dapat digunakan sebagai output, pin TX tidak dapat digunakan sebagai input, dan pin ADC tidak dapat digunakan sebagai output.
- Khusus pin D0 atau GPIO16 dapat digunakan sebagai input atau output, namun pin tersebut tidak dapat dikenai interrupt jika menjadi input dan tidak mendukung PWM jika sebagai output. Sebaliknya, pin D0/GPIO16 normal untuk digunakan.

Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

Syarat dan Kondisi Penggunaan Pin NodeMCU
 (Sumber: <https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>)

Bila pin-pin tersebut tidak digunakan sebagai input atau output, maka Anda bisa menggunakannya sebagai pin komunikasi. Pada komunikasi SPI, pin D5/GPIO14, D6/GPIO12, D7/GPIO13, dan D8/GPIO15 dipetakan sebagai pin SCLK, MISO, MOSI, dan CS. Dalam hal ini NodeMCU pada pin tersebut sebagai SPI Master. Begitu pula untuk jenis komunikasi I₂C, dimana pin D1/GPIO5 dipetakan sebagai SCL dan pin D2/GPIO4 dipetakan sebagai SDA. Sedangkan komunikasi UART menggunakan pin RX/GPIO3 dan TX/GPIO1

Sampai di sini dulu penjelasan singkat tentang mikrokontroler NodeMCU yang akan difungsikan sebagai smart device atau “things” pada platform IoT. Selanjutnya kita akan implementasikan

bagaimana pemanfaatan mikrokontroler NodeMCU pada IoT di pembahasan selanjutnya. Jika Anda masih belum paham mengenai NodeMCU maka jangan risau. Dalam buku ini akan dijelaskan setiap demi setiap sehingga Anda dapat mempraktekannya langsung.

3.2 Software Pendukung

Sebelum memulai projek Arduino maka kita harus meng-instal terlebih dahulu beberapa software pendukungnya, antara lain:

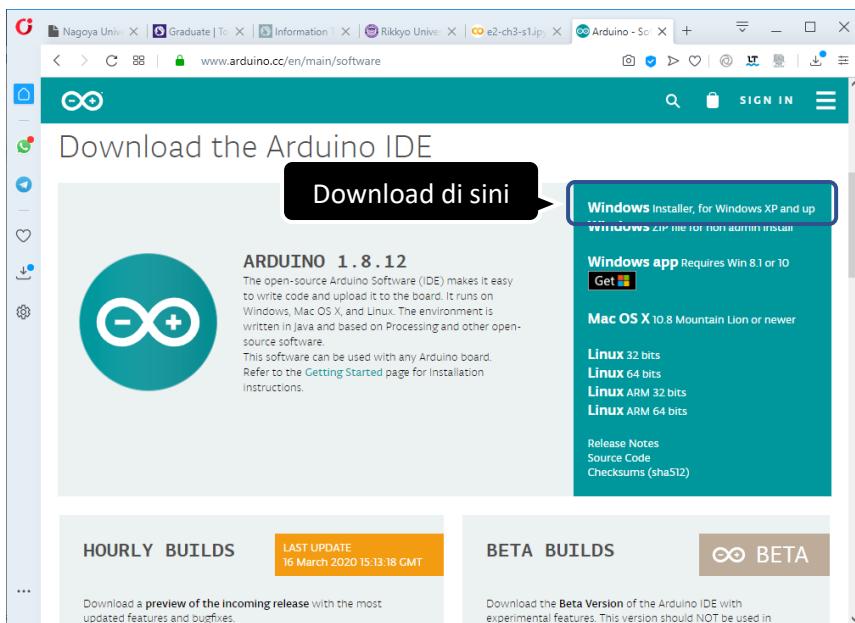
- Arduino IDE (Integrated Development Environment), adalah software yang berfungsi untuk menuliskan kode program, debugger kode program, dan sebagai compiler program. Dimana file hasil compile akan di-write ke chip Arduino, sehingga Arduino dapat berfungsi secara mandiri.
- Fritzing, adalah software alat bantu untuk mem-visualisasikan rancangan, pengabelan, peletakkan komponen secara software. Kemudian dari hasil rancangan Fritzing tersebut diimplementasikan pada kondisi sebenarnya.

3.3 Instalasi & Konfigurasi IDE MCU

Untuk memprogram microcontroller NodeMCU diperlukan *integrated development environment* Arduino (IDE Arduino) yang berfungsi sebagai *editor*, *debugger* dan *writer/flasher* program ke chipset NodeMCU ESP-12E.

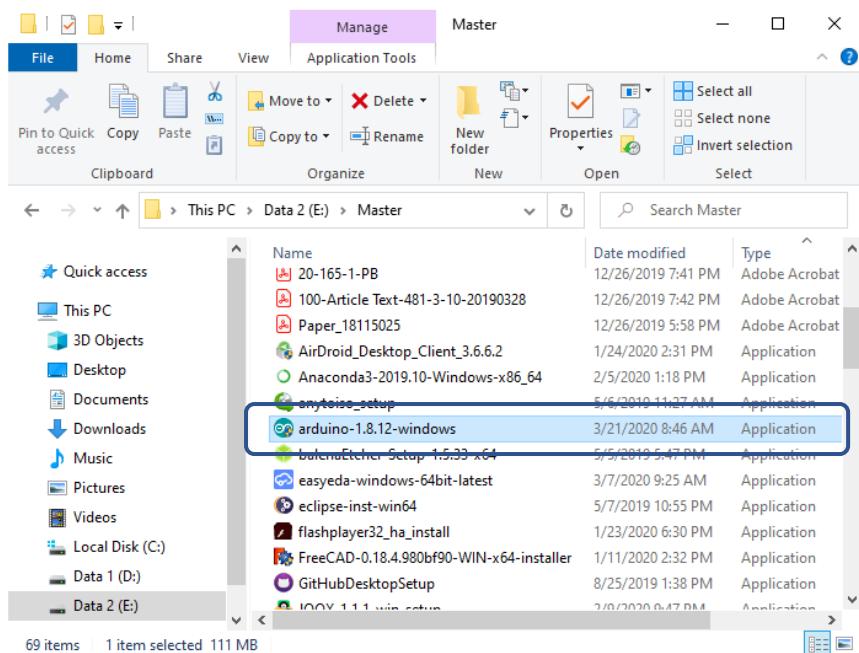
Berikut ini langkah instalasi dan konfigurasi aplikasi IDE MCU Arduino (IDE Arduino dapat juga digunakan sebagai aplikasi IDE NodeMCU):

- Silahkan download file Arduino IDE dari website berikut <https://www.Arduino.cc/en/Main/Software>, khususnya link berikut https://www.arduino.cc/download_handler.php?f=/arduino-1.8.12-windows.exe.



Alamat URL Download IDE Arduino

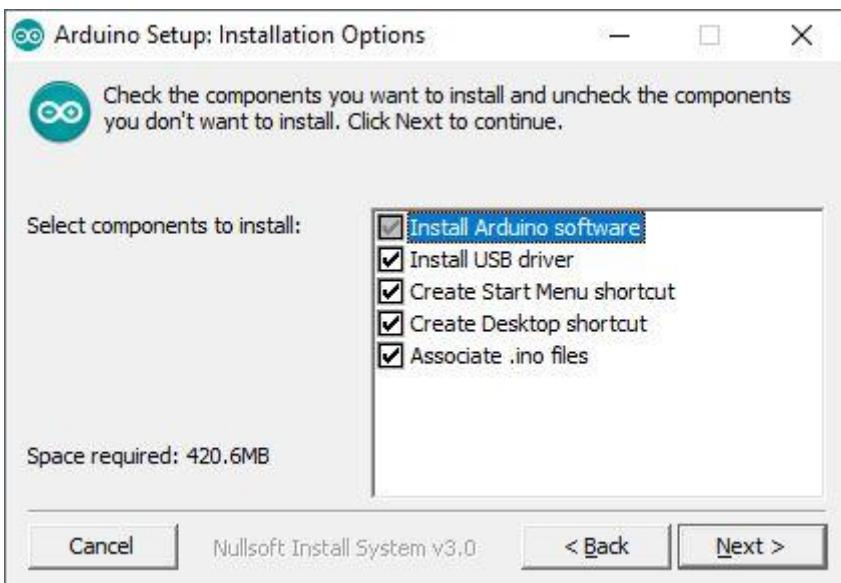
- Kemudian install file **Arduino-1.8.12-windows** di komputer Anda. Nama file yang di-download tidak harus sama antara yang tertera dibuku karena tergantung dari versi yang di-download saat ini.



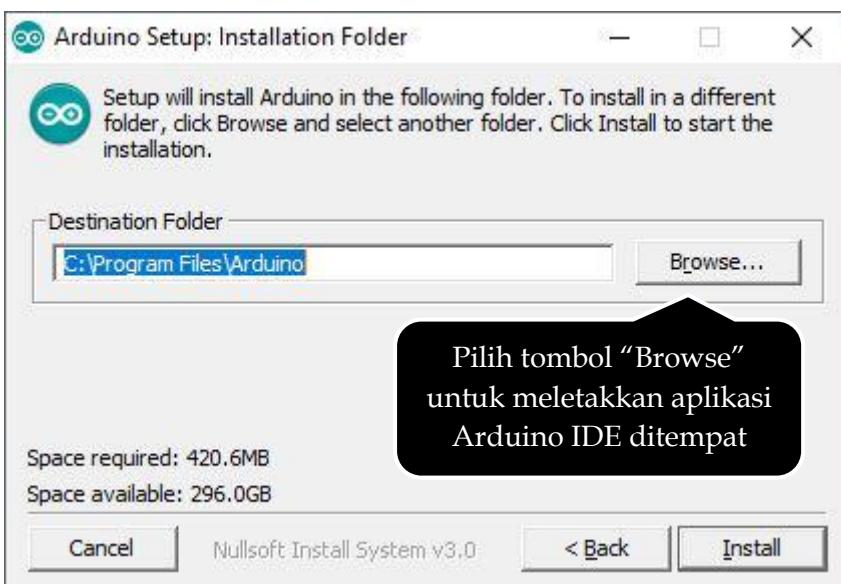
File Hasil Dowload arduino-1.8.2-widows



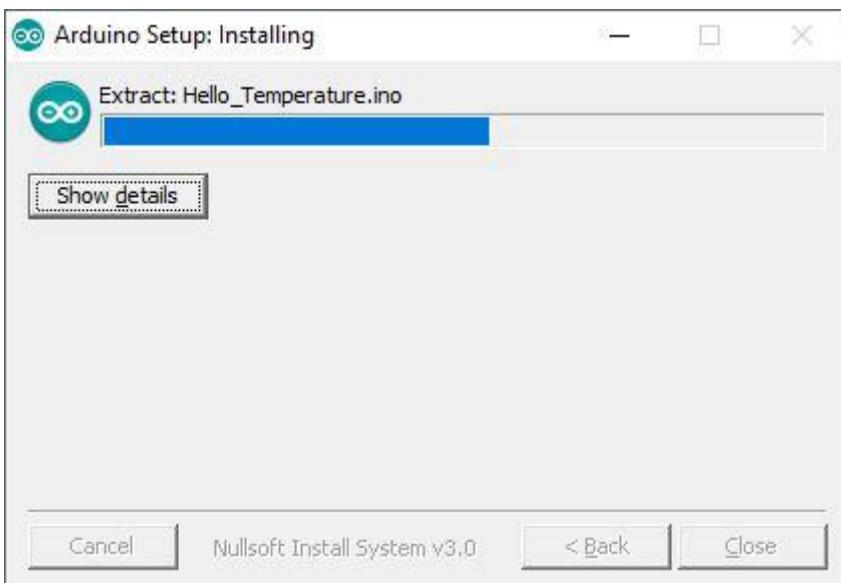
Pilih tombol **I Agree**



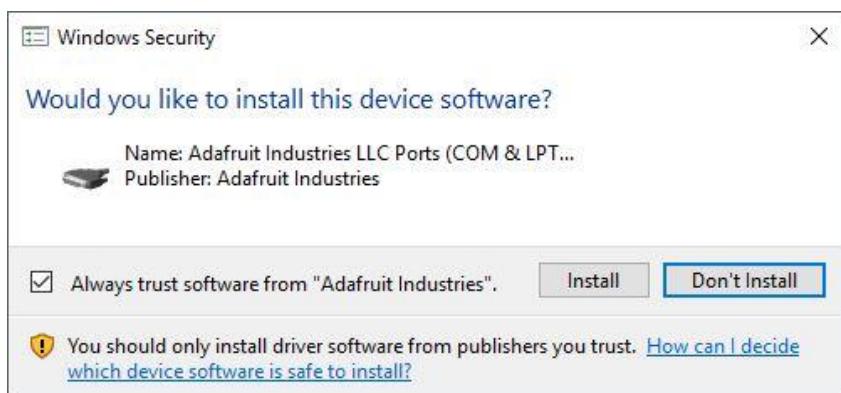
Pilih tombol **Next**



Pilih tombol **Install** untuk melanjutkannya



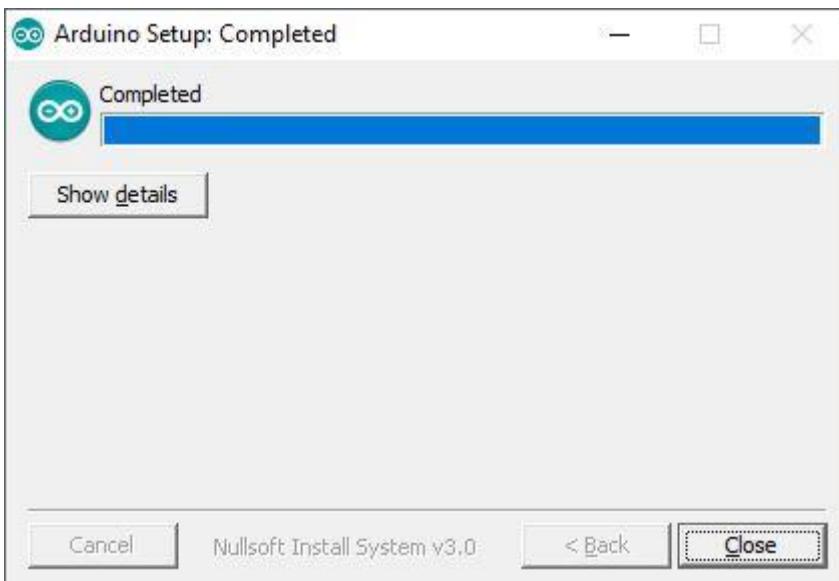
Tunggu proses instalasi sampai selesai



Pilih tombol **Install**



Pilih tombol **Install**

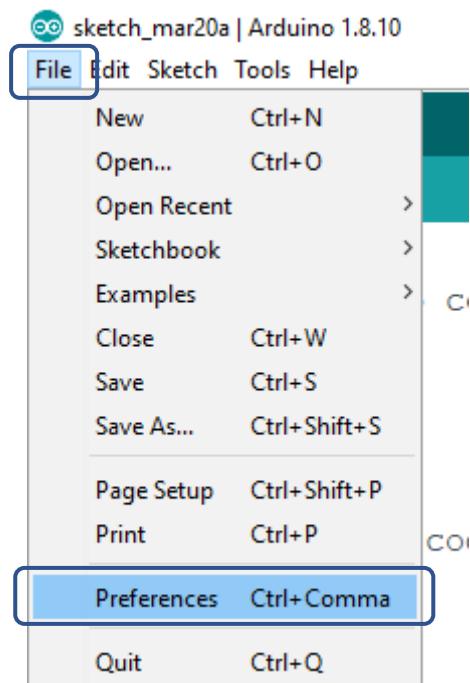


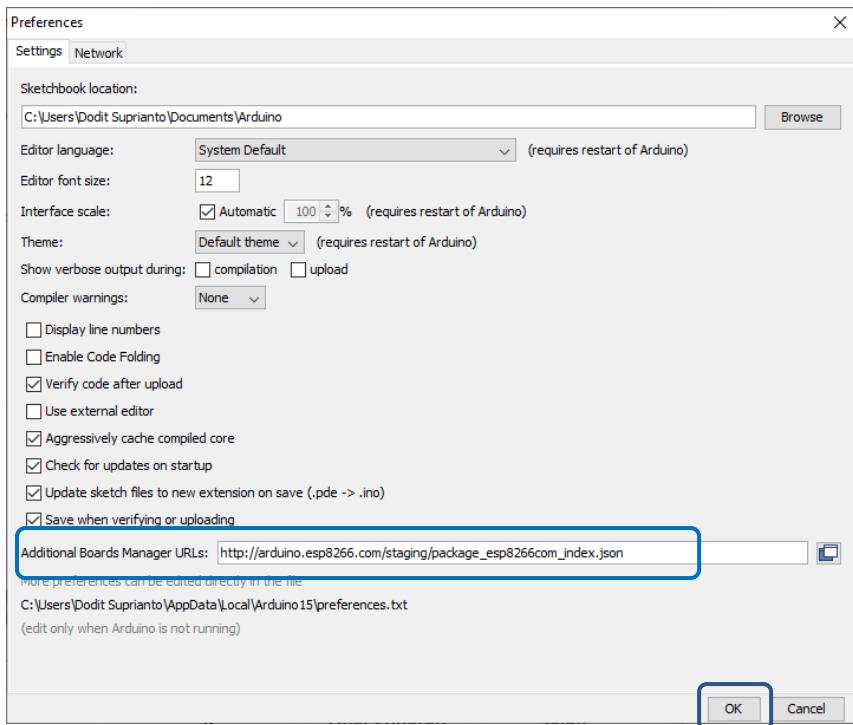
Proses instalasi selesai, pilih tombol **Close**

- Secara default aplikasi IDE Arduino hanya digunakan untuk memprogram board mikrokontroler Arduino sebagai target *writing*-nya, sehingga kita perlu menambahkan library board ESP8266 ke dalam IDE Arduino, agar board NodeMCU yang merupakan keluarga chipset ESP8266 dapat dikenali oleh IDE

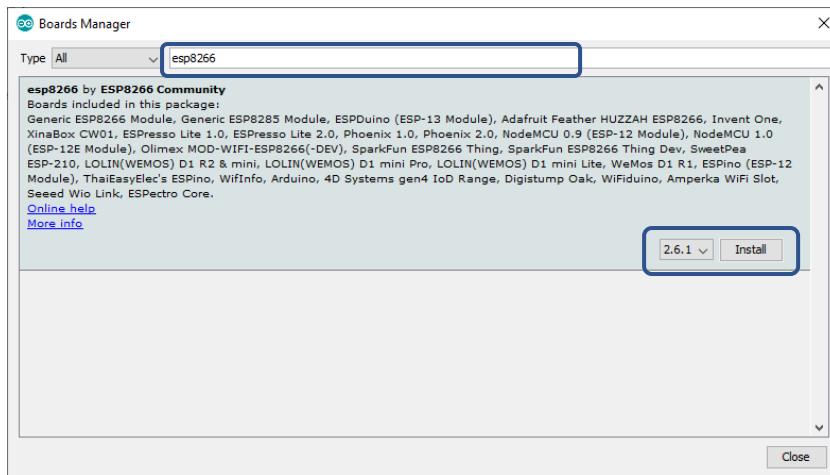
Arduino dan dapat diprogram dengan gaya bahasa C layaknya memprogram mikrokontroller Arduino.

- Untuk menambahkan library board ESP8266, jalankan aplikasi arduino IDE terlebih dahulu. Kemudian pilih menu “File → Preferences” pada kolom “Additional Boards Manager URLs”, kemudian tulis alamat website berikut ini “https://arduino.esp8266.com/stable/package_esp8266com_index.json”. Kemudian pilih tombol “OK”.

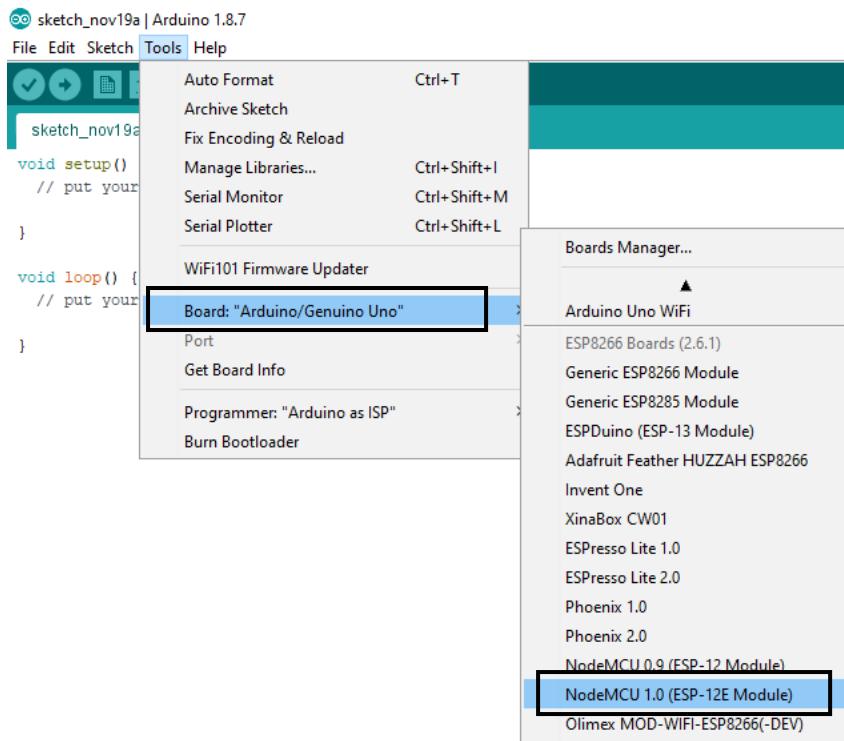


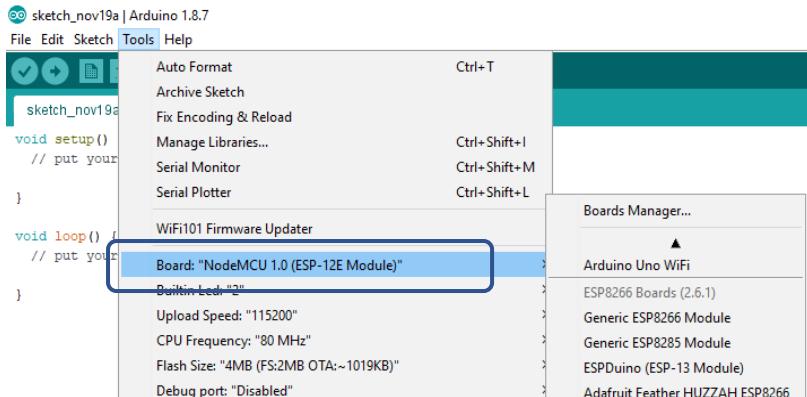


- Pilih menu “Tools → Board → Board Manager”. Tuliskan “esp8266” pada kolom pencarian. Pada daftar di bawahnya akan muncul tulisan “esp8266 by ESP8266 Community”, pilih dan klik tombol **Install** di sebelah kanan bawah. Pastikan saat melakukannya komputer telah terhubung dengan internet karena file pendukung akan di-download secara otomatis.

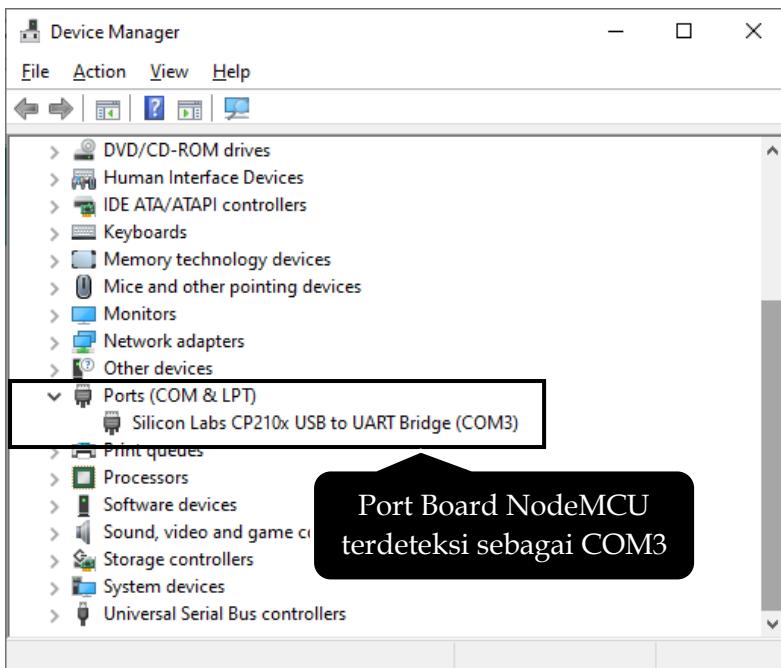


- Pilih menu Tools → Board Manager → NodeMCU 1.0 (ESP-12E Module)

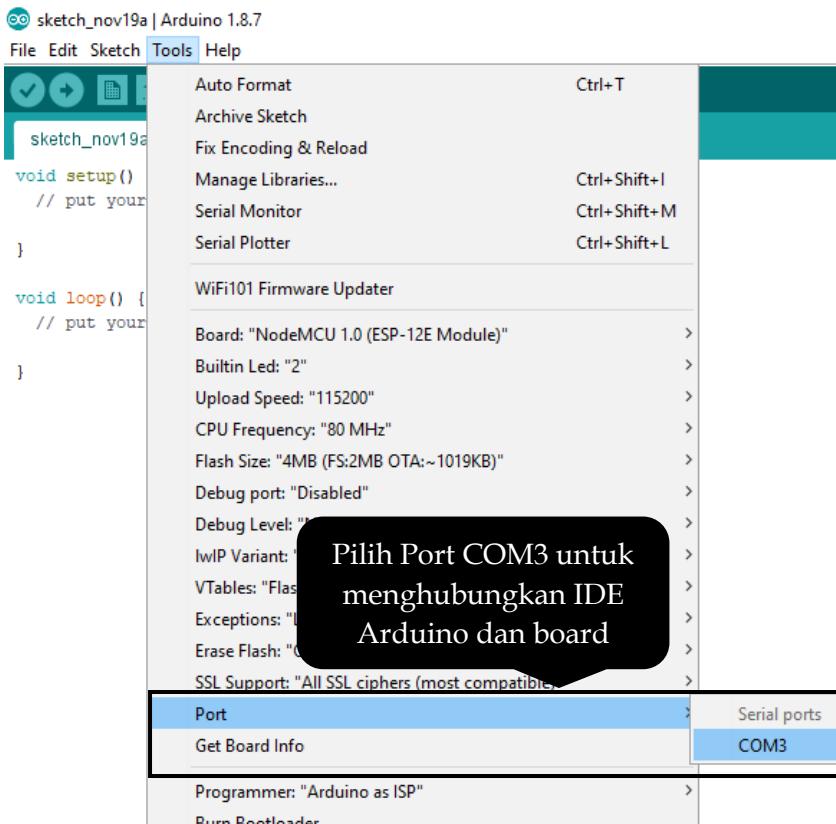




- Sekarang hubungkan board NodeMCU ke laptop melalui kabel data micro USB. Periksa apakah port USB telah aktif melalui **“Control Panel → Device Manager”**. Gambar di bawah menunjukkan *interface* antara NodeMCU dan laptop Windows 10 yang telah terhubung melalui Port COM3 (tidak selalu di port COM3, tergantung windows mendetectsinya).



- Buka aplikasi Arduino IDE dimenu “Tools → Port”, sesuaikan dengan nomor port yang terbaca pada sistem, dalam hal ini adalah Port COM3.



- Jalankan aplikasi dengan kode program standar seperti di bawah ini, kemudian pilih tombol centang di bagian toolbar kiri atas.

The screenshot shows the Arduino IDE interface. At the top, it says "sketch_nov19a | Arduino 1.8.7". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for upload, download, and other functions, with the "upload" icon circled in red. The main workspace shows a sketch named "sketch_nov19a" containing the following code:

```
void setup() {
  // put your setup code here, to run once:

}

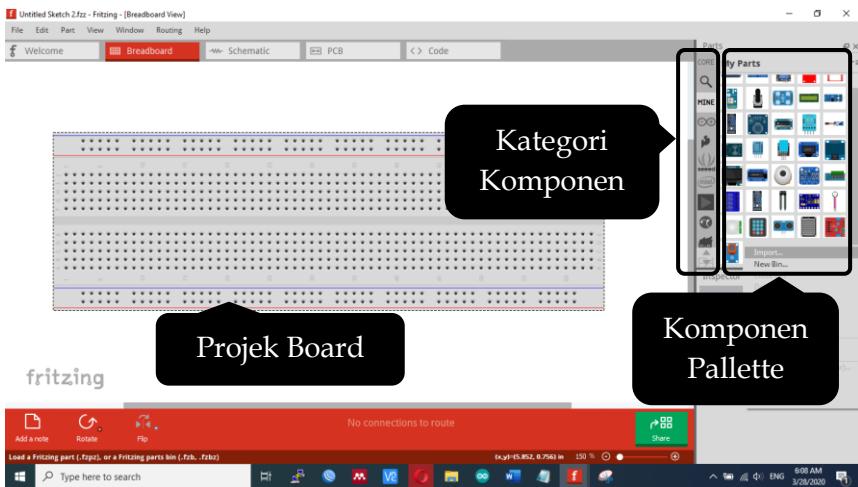
void loop() {
  // put your main code here, to run repeatedly:
}
```

At the bottom of the workspace, it says "Done compiling." A callout bubble points to this text with the text "Menunjukkan NodeMCU siap diprogram dengan IDE". The status bar at the bottom right shows "MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3".

Jika tidak ada pesan kesalahan maka instalasi sukses dan NodeMCU siap untuk diprogram.

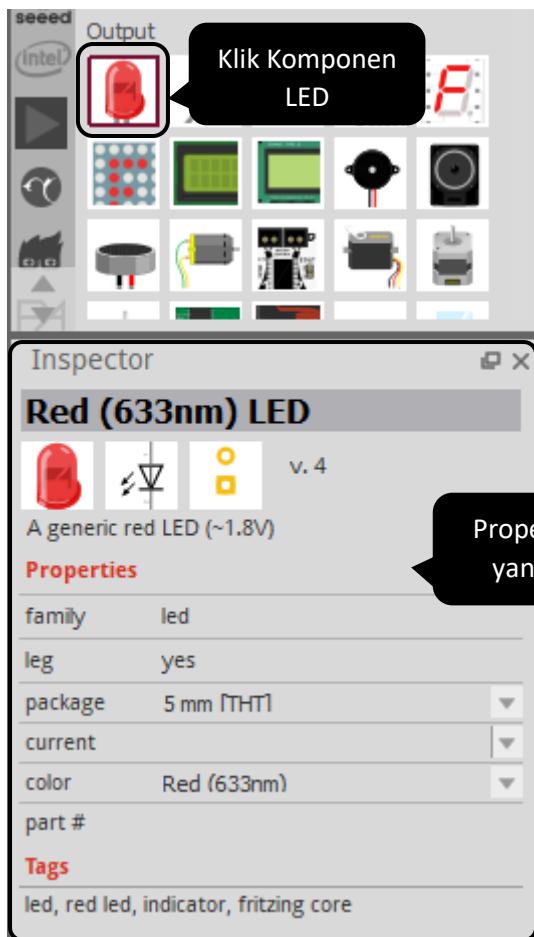
3.4 Instalasi Aplikasi Fritzing

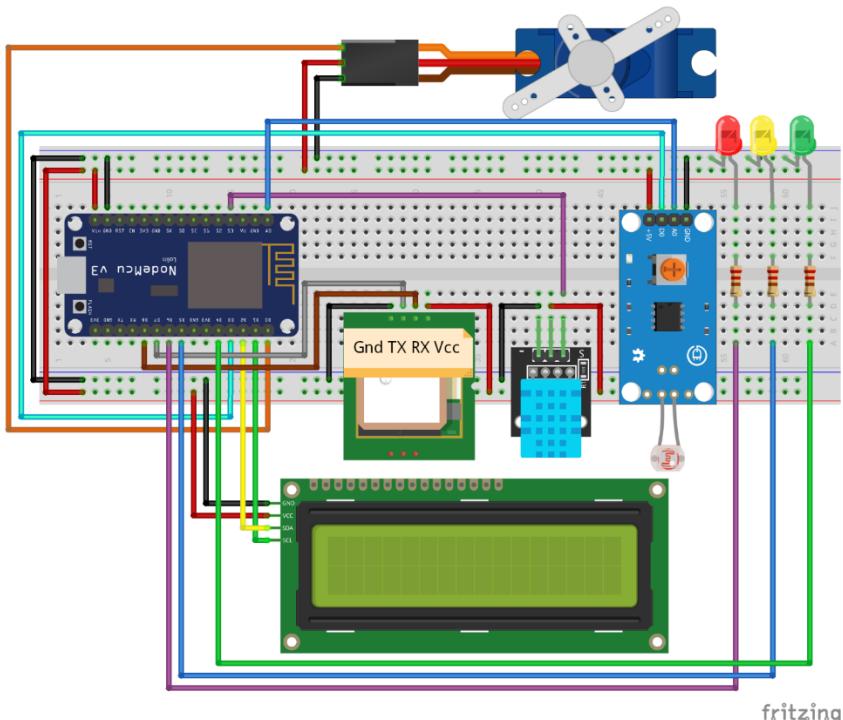
Untuk memudahkan perancangan sebuah model dan pengkabelan antar bagian, disarankan untuk menggunakan aplikasi Fritzing yang dapat di-download di <http://fritzing.org/download/0.8.7b/windows/fritzing.0.8.7b.pc.zip>. Jika gambar komponen tidak ditemukan pada aplikasi Fritzing maka Anda dapat melakukan pencarian dan mengimpor file komponen tersebut di internet. Adapun jenis file gambar komponen fritzing ditandai dengan file berekstensi .fzpz.



Untuk menempatkan komponen pada projek board, Anda tinggal melakukan drag-drop dari komponen pallette ke projek board. Untuk pengkabelan dari satu titik ke titik lainnya Anda tinggal mengklik sekali dimulai dari titik awal kemudian menyeret mouse sampai ke titik tujuan. Agar tampak rapi jalur pengkabelannya saat terjadi lekukan, pilih sembarang titik disepanjang kabel, kemudian gunakan **Shift+Mouse** untuk membentuk lekukan tegak lurus, atau gunakan **Ctrl+Shift+Mouse** pada titik disepanjang kabel untuk membentuk lekukan kurva.







Gambar rancangan di atas merupakan ilustrasi komponen secara visual. Anda harus memperhatikan kaki komponen fisik yang Anda miliki karena urutan pin setiap bracket komponen kadang berbeda tergantung produsennya. Kesalahan memposisikan kabel akan menyebabkan komponen tidak berfungsi, bahkan mengalami kerusakan yang fatal.

Cara membaca jalur pada projek board adalah sebagai berikut:

- Lubang-lubang mendatar yang ditandai garis warna biru adalah terhubung semua, digunakan sebagai jalur Ground.
- Lubang-lubang mendatar yang ditandai garis warna merah adalah terhubung semua, digunakan sebagai jalur tegangan Vcc.
- Lubang-lubang yang berada ditengah membentuk kolom terhubung semua, namun tidak terhubung dengan lubang-lubang pada kolom sebelahnya.
- Ditengah projek board terdapat sekat, artinya antara blok bagian atas dan bawah tidak terhubung.



3.5 Aneka Projek Microcontroller

Pada bagian ini kita akan memprogram perangkat keras microcontroller unit NodeMCU. Memahami bagaimana cara membaca kondisi lingkungan dengan sensor, mengendalikan actuator, menampilkan data pada LCD, memproses data pada MCU dan lain sebagainya.

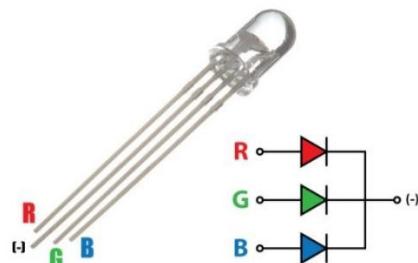
Bahasa pemrograman yang akan digunakan adalah C dan menggunakan IDE Arduino sebagai editornya. Sedangkan hasil akhirnya adalah membangun aplikasi *embedded system* secara mandiri, yaitu aplikasi yang bekerja pada sebuah chip melalui proses *writing/flashing* program ke dalam chip NodeMCU ESP-12E.

3.5.1 Running LED

LED (*Light Emitting Diode*) banyak digunakan sebagai indikator berlangsungnya proses, awal proses atau berakhirnya proses pada sistem *microcontroller* sehingga pengguna dapat mengetahui proses yang sedang terjadi. LED memiliki polaritas tegangan positif dan negatif. Biasanya posisi positif ditandai dengan kaki yang lebih panjang dari kaki negatif.



LED 5MM Satu Warna



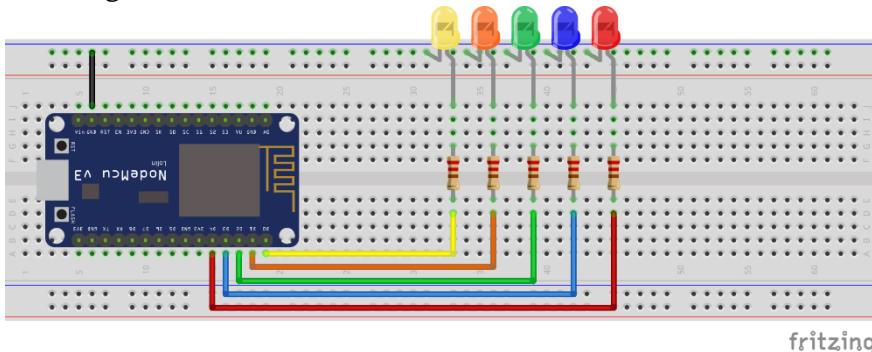
LED 5MM 3 Warna RGB

Tujuan projek ini adalah untuk membuat animasi LED dengan cara menghidupkan dan mematikan 5 LED secara bergantian selama 600 mikrodetik setiap LED-nya.

Kebutuhan Bahan

Bahan	Jumlah	Nilai	Keterangan
LED 5MM	5 pcs		Warna bebas
Resistor 1/4 watt	5 pcs	220 Ohm	

Rancangan



Kode Program

```
#define LED1 16
#define LED2 5
#define LED3 4
#define LED4 0
#define LED5 2
```

Kode Program

```
void setup() {  
    Serial.begin(9600);  
    pinMode(LED1, OUTPUT);  
    pinMode(LED2, OUTPUT);  
    pinMode(LED3, OUTPUT);  
    pinMode(LED4, OUTPUT);  
    pinMode(LED5, OUTPUT);  
}  
  
void loop() {  
    animasi_LED1();  
    //animasi_LED2();  
}  
  
void animasi_LED1() {  
    //LED 1 Hidup, lainnya mati  
    Serial.println("LED 1 Hidup, Lainnya Mati");  
    digitalWrite(LED1, HIGH);  
    digitalWrite(LED2, LOW);  
    digitalWrite(LED3, LOW);  
    digitalWrite(LED4, LOW);  
    digitalWrite(LED5, LOW);  
    delay(600);  
  
    //LED 2 Hidup, lainnya mati  
    Serial.println("LED 2 Hidup, Lainnya Mati");  
    digitalWrite(LED1, LOW);  
    digitalWrite(LED2, HIGH);  
    digitalWrite(LED3, LOW);  
    digitalWrite(LED4, LOW);  
    digitalWrite(LED5, LOW);  
    delay(600);  
  
    //LED 3 Hidup, lainnya mati  
    Serial.println("LED 3 Hidup, Lainnya Mati");  
    digitalWrite(LED1, LOW);  
    digitalWrite(LED2, LOW);  
    digitalWrite(LED3, HIGH);  
    digitalWrite(LED4, LOW);  
    digitalWrite(LED5, LOW);  
    delay(600);  
}
```

Kode Program

```
//LED 4 hidup, lainnya mati
Serial.println("LED 4 Hidup, Lainnya Mati");
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, HIGH);
digitalWrite(LED5, LOW);
delay(600);

//LED 5 hidup, lainnya mati
Serial.println("LED 5 Hidup, Lainnya Mati");
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED5, HIGH);
delay(600);

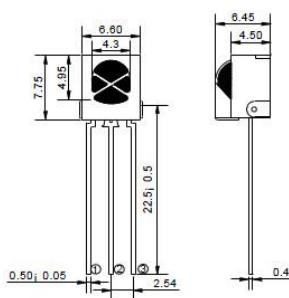
}

void animasi_LED2() {
    //kembangkan sendiri dengan berbagai animasi
    LED lainnya
}
```

3.5.2 Kontrol Remote IR Dengan VS1838

Sebelum memanfaatkan *Remote Infrared* sebagai pengendali jarak jauh, NodeMCU terlebih dahulu harus memahami kode setiap tombol remote. Setiap tombol remote infrared dipetakan terlebih dahulu oleh NodeMCU kemudian digunakan sesuai kebutuhan. Setiap jenis remote IR memiliki kode tombol berbeda-beda meskipun fungsinya sama.

Projek ini membutuhkan sebuah Remote IR sebagai pengirim kode tombol remote dan sensor IR Receiver seri TL1838 atau VS1838B sebagai penerima perintah remote.



IR Receiver Seri TL1838 atau VS1838B



① OUT
② GND
③ VCC

IR Receiver Pinout

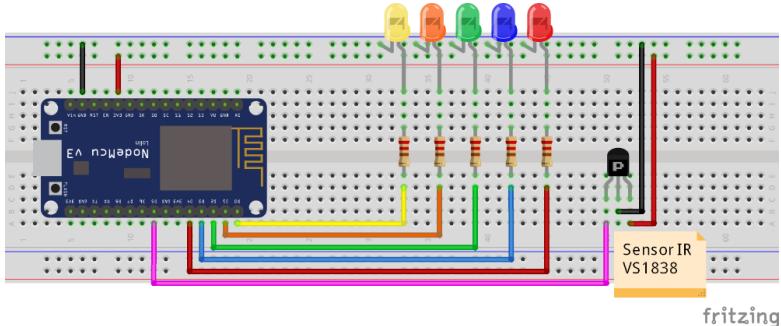


Remote IR Sebagai Transmitter

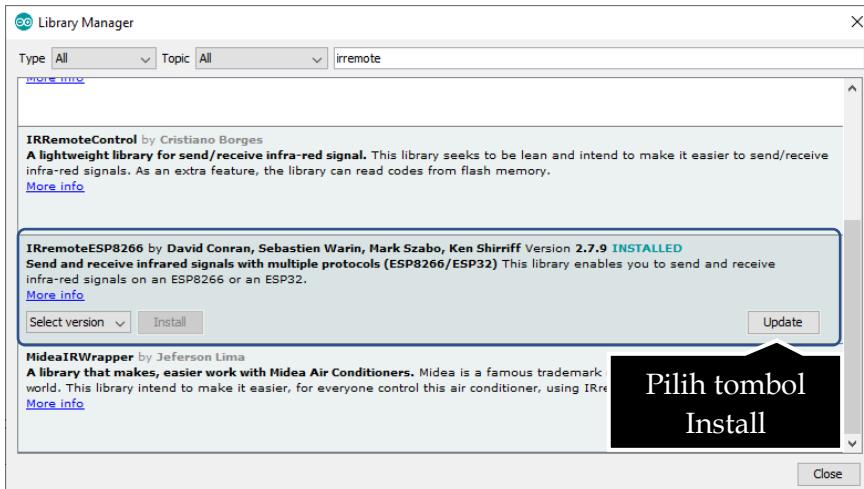
Kebutuhan Bahan

Bahan	Jumlah	Nilai	Keterangan
Sensor penerima (IR Photodiode)	1 pcs	TL1838 VS1838B	
Remote STB UseeTV IndieHome atau gunakan remote jenis lain.	1 pcs		
LED 5MM	5 pcs		Warna bebas
Resistor 1/4 watt	5 pcs	220 Ohm	

Rancangan



Projek remote IR membutuhkan library “**IRremoteESP8266**”. Untuk mendapatkannya, buka IDE Arduino, masuk ke menu Sketch → Include Library → Manage libraries. Pastikan bahwa komputer dalam kondisi terhubung ke internet, kemudian tuliskan pada field pencarian “irremote”. Dari daftar akan muncul beberapa pilihan library, pilih library “**IRremoteESP8266 by David...**”. Cara ini berlaku sama untuk penambahan library lainnya.



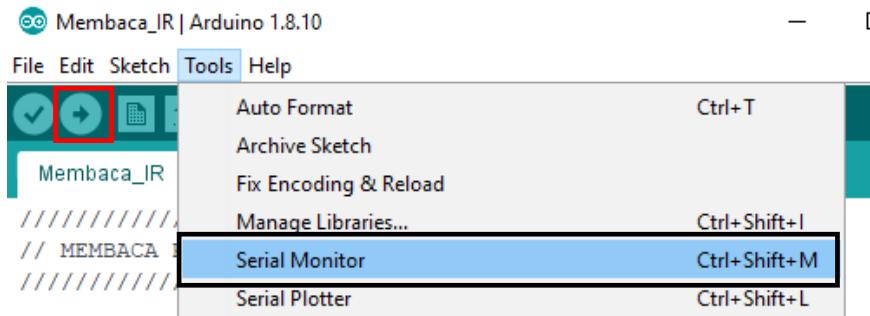
Kode Program

```
//////////  
// MEMBACA KEYPAD REMOTE IR //  
//////////  
  
#include <Arduino.h>  
#include <IRremoteESP8266.h>  
#include <IRrecv.h>  
#include <IRutils.h>  
  
const uint16_t pinIR = 14;  
String tombol = "";  
  
IRrecv PenerimaIR(pinIR);  
decode_results hasil;  
  
void setup() {  
    Serial.begin(9600);
```

Kode Program

```
PenerimaIR.enableIRIn();  
}  
  
void loop() {  
    if (PenerimaIR.decode(&hasil)) {  
        unsigned int nilaitombol = hasil.value;  
        tombol = String(nilaitombol);  
        Serial.println("Kode Tombol: " + tombol);  
        PenerimaIR.resume();  
    }  
    delay(100);  
}
```

Buka Serial Monitor pada menu Tools → Serial Monitor, kemudian jalankan aplikasinya



Menu Serial Monitor

Petakan setiap kode tombol remote dengan cara menembakkan remote IR ke sensor penerima IR. Sambil menekan setiap tombol pada remote IR, Anda mencatat kode tombol yang tertera pada Serial Monitor, seperti tampak pada gambar di bawah.

```

COM3
Send

Kode = 1303529910
Kode = 1303562550
Kode = 1303524300
Kode = 1303540110
Kode = 1303572750
Kode = 1303516140
Kode = 1303511550
Kode = 1303554390
Kode = 1303544190
Kode = 1303526340
Kode = 1303525830
Kode = 1303540110
Kode = 1303572750
Kode = 1303516140

Autoscroll No line ending 9600 baud Clear output

```

Hasil Pembacaan Tombol Remote IR UseeTV

Khusus remote IR UseeTV yang menjadi bahan percobaan diperoleh pemetaan kode tombol seperti tampak pada tabel di bawah. Sedangkan untuk remote yang berbeda akan menghasilkan pembacaan kode berbeda.

Tombol	Kode Tombol	Tombol	Kode Tombol
	Power 1303526340		Mute 1303525830
	Vol - 1303511550		Play/Pause 1303554390
	Vol + 1303544190		OK 1303540620
	Up 1303532460		Down 1303530420
	Left 1303550310		Right 1303544700
	Loop 1303552860		Menu 1303515630

Tombol	Kode Tombol	Tombol	Kode Tombol
	Home	1303527870	1
2	No. 2	1303562550	3
4	No. 4	1303540110	5
6	No. 6	1303516140	7
8	No. 8	1303564590	9
	Titik	1303542660	0
	Back	1303575300	

Pembacaan Kode Remote UseeTV

Dari tahap ini kita sudah bisa membaca kode tombol remote IR UseeTV. Selanjutnya kode tombol tersebut akan digunakan pada projek berikutnya.

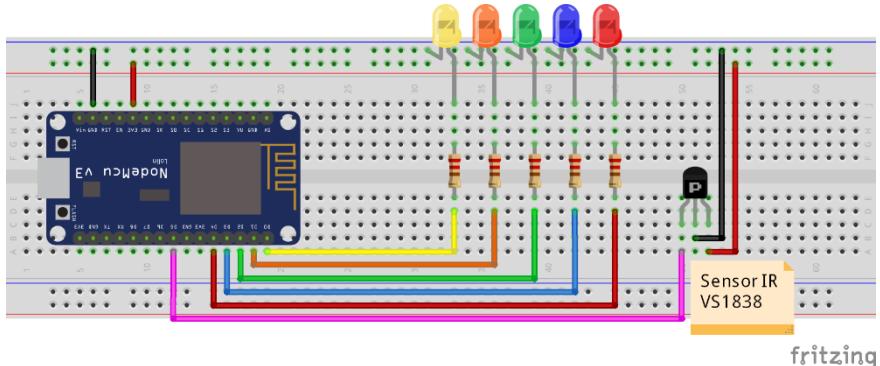
Berdasarkan pembacaan kode tombol remote IR, kita dapat mengembangkan menjadi lebih kompleks, yaitu menghidupkan LED melalui remote IR dengan skenario sebagai berikut:

- Tombol 1 (kode 1303529910) untuk memanggil fungsi animasi LED yang hidup dari kiri ke kanan.
- Tombol 2 (kode 1303562550) untuk memanggil fungsi animasi LED yang hidup dari kanan ke kiri.
- Tombol 3 (kode 1303524300) untuk memanggil fungsi animasi LED yang hidup dari tengah.

Kebutuhan Bahan

Bahan	Jumlah	Nilai	Keterangan
Resistor	6 pcs	220 Ohm	
LED 5MM	6 pcs		Warna bebas
Sensor IR Receiver	1 pcs	TL1838 / VS1838B	
Remote UseeTV	1 pcs		

Rancangan



Kode Program

```
//////////  

// Kontrol LED dengan Remote IR //  

//////////  
  

//Library  

#include <Arduino.h>  

#include <IRremoteESP8266.h>  

#include <IRrecv.h>  

#include <IRutils.h>  
  

//Pin LED  

#define LED1 16  

#define LED2 5  

#define LED3 4  

#define LED4 0
```

Kode Program

```
#define LED5 2

//Pin Sensor IR Receiver
const uint16_t pinIR = 14;
String tombol = "";

IRrecv PenerimaIR(pinIR);
decode_results hasil;

void setup() {
    //Baudrate serial monitor
    Serial.begin(9600);

    //Pin LED sebagai output
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(LED5, OUTPUT);

    PenerimaIR.enableIRIn();
}

void loop() {
    if (PenerimaIR.decode(&hasil)) {
        unsigned int nilaitombol = hasil.value;
        tombol = String(nilaitombol);

        Serial.println("Kode Tombol: " + tombol);

        //Fungsional kode tombol sesuaikan dengan
        kebutuhan
        //Kode tombol sesuaikan dengan remote kalian
        masing-masing
        if (tombol == "1303529910") {
            //kode tombol 1
            animasi_LED1();
        } else if (tombol == "1303562550") {
            //kode tombol 2
            animasi_LED2();
        } else if (tombol == "1303524300") {
```

Kode Program

```
//kode tombol 3
animasi_LED3();
}

PenerimaIR.resume();
}
delay(100);
}

void animasi_LED1() {
//LED 1 Hidup, lainnya mati
Serial.println("LED 1 Hidup, Lainnya Mati");
digitalWrite(LED1, HIGH);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED5, LOW);
delay(600);

//LED 2 Hidup, lainnya mati
Serial.println("LED 2 Hidup, Lainnya Mati");
digitalWrite(LED1, LOW);
digitalWrite(LED2, HIGH);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED5, LOW);
delay(600);

//LED 3 Hidup, lainnya mati
Serial.println("LED 3 Hidup, Lainnya Mati");
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, HIGH);
digitalWrite(LED4, LOW);
digitalWrite(LED5, LOW);
delay(600);

//LED 4 hidup, lainnya mati
Serial.println("LED 4 Hidup, Lainnya Mati");
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
```

Kode Program

```
digitalWrite(LED3, LOW);
digitalWrite(LED4, HIGH);
digitalWrite(LED5, LOW);
delay(600);

//LED 5 hidup, lainnya mati
Serial.println("LED 5 Hidup, Lainnya Mati");
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED5, HIGH);
delay(600);

}

void animasi_LED2() {
//LED 5 Hidup, lainnya mati
Serial.println("LED 1 Hidup, Lainnya Mati");
digitalWrite(LED5, HIGH);
digitalWrite(LED4, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED1, LOW);
delay(600);

//LED 4 Hidup, lainnya mati
Serial.println("LED 2 Hidup, Lainnya Mati");
digitalWrite(LED5, LOW);
digitalWrite(LED4, HIGH);
digitalWrite(LED3, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED1, LOW);
delay(600);

//LED 3 Hidup, lainnya mati
Serial.println("LED 3 Hidup, Lainnya Mati");
digitalWrite(LED5, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED3, HIGH);
digitalWrite(LED2, LOW);
digitalWrite(LED1, LOW);
```

Kode Program

```
delay(600);

//LED 2 hidup, lainnya mati
Serial.println("LED 4 Hidup, Lainnya Mati");
digitalWrite(LED5, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED2, HIGH);
digitalWrite(LED1, LOW);
delay(600);

//LED 1 hidup, lainnya mati
Serial.println("LED 5 Hidup, Lainnya Mati");
digitalWrite(LED5, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED1, HIGH);
delay(600);

}

void animasi_LED3() {
    Serial.println("LED 1 Hidup, Lainnya Mati");
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
    delay(600);

    Serial.println("LED 2 Hidup, Lainnya Mati");
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, HIGH);
    digitalWrite(LED5, LOW);
    delay(600);

    Serial.println("LED 3 Hidup, Lainnya Mati");
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
```

Kode Program

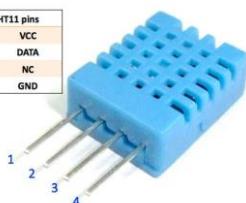
```
digitalWrite(LED3, HIGH);  
digitalWrite(LED4, HIGH);  
digitalWrite(LED5, HIGH);  
delay(600);  
  
Serial.println("LED 4 Hidup, Lainnya Mati");  
digitalWrite(LED1, LOW);  
digitalWrite(LED2, LOW);  
digitalWrite(LED3, LOW);  
digitalWrite(LED4, LOW);  
digitalWrite(LED5, LOW);  
delay(600);  
}
```

3.5.3 Suhu & Kelembaban Dengan DHT11

Dalam Tutorial Arduino ini kita akan belajar bagaimana menggunakan DHT11 atau DHT22 untuk mengukur suhu dan kelembaban. Sensor ini sangat populer untuk penggemar elektronik karena harganya murah tetapi tetap memberikan kinerja yang baik.

Dari sisi harga DHT22 lebih mahal daripada DHT11 dan DHT22 memiliki spesifikasi yang lebih baik. Rentang pengukuran suhu mulai dari -40°C hingga $+125^{\circ}\text{C}$ dengan derajat akurasi $\pm 0,5$, sedangkan rentang temperatur DHT11 adalah 0°C hingga 50°C dengan derajat akurasi ± 2 . Sensor DHT22 memiliki rentang pengukuran kelembaban yang lebih baik, mulai dari 0% hingga 100% dengan akurasi 2-5%, sementara rentang kelembaban DHT11 adalah 20% hingga 80% dengan akurasi 5%.

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



Sensor Kelembaban DHT11

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

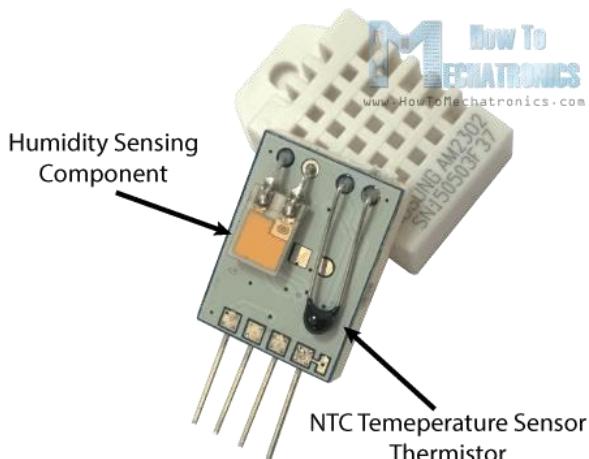


Sensor Kelembaban DHT22

Ada dua spesifikasi di mana DHT11 lebih baik daripada DHT22, yaitu laju pengambilan sampel datanya, untuk DHT11 adalah 1 Hz atau satu bacaan setiap detik, sedangkan tingkat sampling DHT22 adalah 0,5 Hz atau satu bacaan setiap dua detik dan juga DHT11 memiliki ukuran tubuh yang lebih kecil. Tegangan operasi kedua sensor adalah 3V hingga 5V, sedangkan arus maksimal yang digunakan saat mengukur adalah 2,5mA.

Sensor DHT11	Sensor DHT22
0 - 50°C / ± 2°C	<i>Temperature Range</i>
20 - 80% / ± 5%	<i>Humidity Range</i>
1Hz (one reading every second)	<i>Sampling Rate</i>
15.5mm x 12mm x 5.5mm	<i>Body Size</i>
3 - 5V	<i>Operating Voltage</i>
2.5mA	<i>Max Current During Measuring</i>

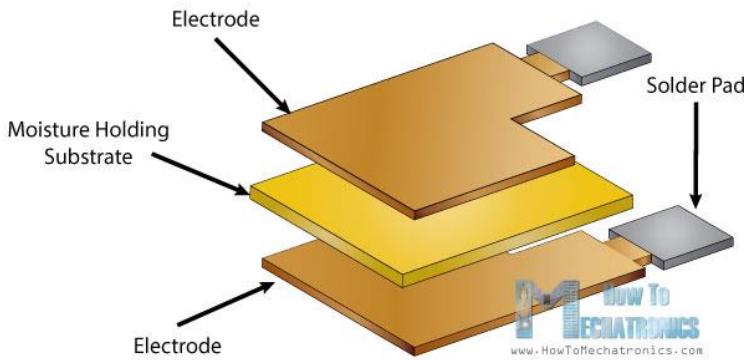
DHT11 dan DHT22 terdiri dari komponen penginderaan kelembaban, sensor suhu NTC (atau termistor) dan IC di sisi belakang sensor.



Komponen Sensor DHT22

(Sumber: <https://www.howtomechatronics.com>)

Untuk mengukur kelembaban, sensor DHT menggunakan komponen penginderaan kelembaban yang memiliki dua elektroda dengan substrat menahan kelembaban diantara dua sisi tersebut. Ketika kelembaban berubah, konduktivitas substrat berubah, atau resistensi antara elektroda-elektroda ini berubah. Perubahan resistansi ini diukur dan diproses oleh IC yang membuatnya siap dibaca oleh mikrokontroler.



Cara Kerja Sensor DHT11/DHT22

(Sumber: <https://howtomechatronics.com>)

Di sisi lain, untuk mengukur suhu, sensor ini menggunakan sensor suhu NTC atau termistor. Termistor adalah resistor variabel yang mengubah ketahanannya dengan perubahan suhu. Sensor ini dibuat dengan sintering (proses pemanasan material) bahan semikonduktif seperti keramik atau polimer untuk memberikan perubahan yang lebih besar pada hambatan hanya dengan perubahan suhu yang kecil. Istilah "NTC" negative Temperature Coeffesient berarti "Koefisien Suhu Negatif", yang berarti bahwa resistensi menurun dengan peningkatan suhu.

Projek yang akan dibuat cukup sederhana, yaitu:

- Membaca suhu dan kelembaban lingkungan yang ditangkap oleh sensor DHT11.
- Menampilkan nilai kelembaban dan suhu pada layar LCD.
- Jika nilai suhu mencapai rentang tertentu maka animasi LED dijalankan.

Aplikasi yang dibuat merupakan kombinasi projek animasi LED sebelumnya dengan menambahkan sensor DHT11 dan layar LCD.

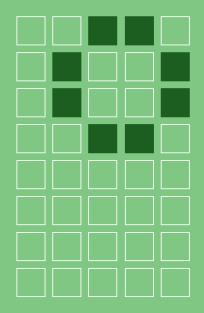
Kebutuhan Bahan

Bahan	Jumlah	Nilai	Keterangan
Resistor	5 pcs	220 ohm	
LED 5MM	5 pcs		Warna bebas
Sensor DHT11 / DHT22	1 pcs		
Modul LCD I ₂ C	1 pcs		

Rancangan

Kadang kita membutuhkan karakter simbol sesuai kebutuhan, namun tidak tersedia secara default. Untuk membuat bit karakter silahkan kunjungi halaman web berikut <https://maxpromer.github.io/LCD-Character-Creator/>. Sebagai contoh kita akan membuat simbol derajat °, dapat dibangkitkan seperti tampak berikut ini

Color Green Blue
Microcontroller Arduino
Interfacing Parallel I₂C
Data Type Binary Hex
Code



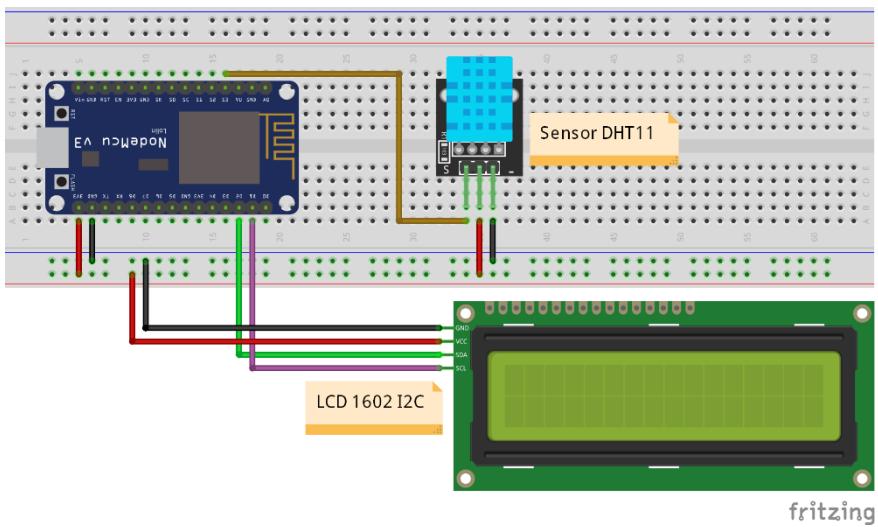
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 in PCF8574 by NXP and Set to 0x3F in PCF8574A by Ti
LiquidCrystal_I2C lcd(0x3F, 16, 2);

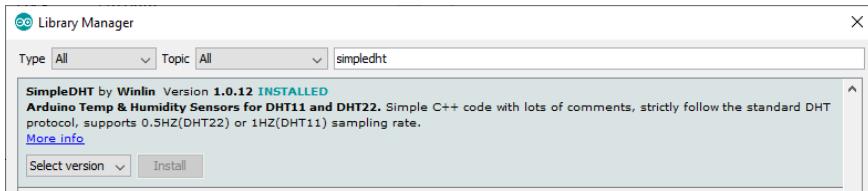
byte customChar[] = {
    B00110,
    B01001,
    B01001,
    B00110,
    B00000,
    B00000,
    B00000,
    B00000
};
```

Link

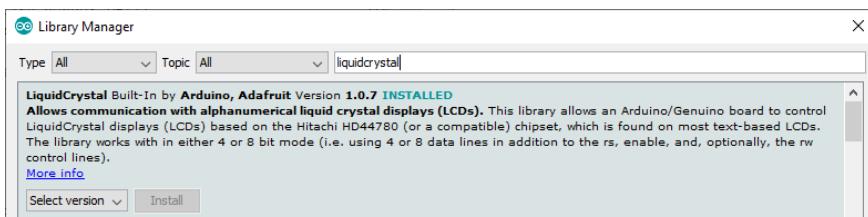
- Arduino LCD Circuit
- Arduino LCD I₂C Circuit
- Arduino LCD I₂C library



Library yang dibutuhkan dalam projek ini adalah "SimpleDHT" dan "LiquidCrystal_I2C". Seperti cara sebelumnya, dari IDE Arduino buka menu Sketch → Include Library → Manage libraries. Lakukan pencarian dengan kata kunci "simpaledht" dan "liquidcrystal" seperti tampak pada gambar di bawah ini.



Library SimpleDHT



Library LiquidCrystal_I2C

Kode Program

```
#include <SimpleDHT.h> //library sensor DHT11
```

Kode Program

```
#include <LiquidCrystal_I2C.h> //library LCD

#define LED1 16    // LED kuning
#define LED2 0     // LED Orange
#define LED3 2     // LED Hijau
#define LED4 14    // LED Biru
#define LED5 12    // LED Merah
#define pinDHT 10 // SD3 pin signal sensor DHT

byte temperature = 0;
byte humidity = 0;

LiquidCrystal_I2C lcd(0x27, 16, 2); //instan
LCD I2C
SimpleDHT11 dht11(pinDHT); //instan sensor
dht11

//membuat kustom karakter simbol derajat
//https://maxpromer.github.io/LCD-Character-Creator/
byte derajat[] = {
    B01110,
    B10001,
    B10001,
    B10001,
    B01110,
    B00000,
    B00000,
    B00000
};

void setup() {
    Serial.begin(9600);

    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(LED5, OUTPUT);
    pinMode(pinDHT, INPUT);
```

Kode Program

```
lcd.begin(); // Inisialisasi LCD
lcd.clear();
lcd.createChar(0, derajat);
lcd.setCursor(0, 0);
lcd.print("Humidity");
lcd.setCursor(0, 1);
lcd.print("Sensor");
delay(3000);
lcd.clear();
}

void loop() {
    KelembabanSuhu();

    //Agar nilai suhu dan kelembaban berubah
    //tempelkan jari ke DHT11 atau dekatkan
    dengan api kecil
    if (temperature >= 28)
    {
        LEDKiriKeKanan();
    }

    if (humidity > 60)
    {
        LEDKananKeKiri();
    }
}

void KelembabanSuhu()
{
    int err = SimpleDHTerrSuccess;

    if ((err = dht11.read(&temperature,
    &humidity, NULL)) != SimpleDHTerrSuccess)
    {
        Serial.print("Pembacaan DHT11 gagal,
err=");
        Serial.println(err);
        delay(1000);
        return;
    }
}
```

Kode Program

```
Serial.print("Sample OK: ");
Serial.print((int)temperature);

Serial.print(" *C, ");
Serial.print((int)humidity);
Serial.println(" H");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: " +
String((int)temperature));
lcd.write(0);
lcd.print("C ");
lcd.setCursor(0, 1);
lcd.print("Humi: " + String((int)humidity) +
"H");

delay(1500);
}

void LEDKiriKeKanan()
{
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
    delay(250);

    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
    delay(250);

    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
```

Kode Program

```
digitalWrite(LED5, LOW);
delay(250);

digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, HIGH);
digitalWrite(LED5, LOW);
delay(250);

digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
digitalWrite(LED5, HIGH);
delay(250);

}

void LEDKananKeKiri()
{
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, HIGH);
    delay(250);

    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, HIGH);
    digitalWrite(LED5, LOW);
    delay(250);

    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
    delay(250);
```

Kode Program

```
digitalWrite(LED1, LOW);  
digitalWrite(LED2, HIGH);  
digitalWrite(LED3, LOW);  
digitalWrite(LED4, LOW);  
digitalWrite(LED5, LOW);  
delay(250);  
  
digitalWrite(LED1, HIGH);  
digitalWrite(LED2, LOW);  
digitalWrite(LED3, LOW);  
digitalWrite(LED4, LOW);  
digitalWrite(LED5, LOW);  
delay(250);  
}
```

3.5.4 Intesitas Cahaya Dengan LDR

Sebuah LDR atau fotoresistor terbuat dari bahan semikonduktor dengan resistansi tinggi. Ia memiliki resistansi tinggi karena hanya ada sedikit elektron yang bebas dan dapat bergerak - sebagian besar elektron terkunci dalam kisi kristal dan tidak dapat bergerak.

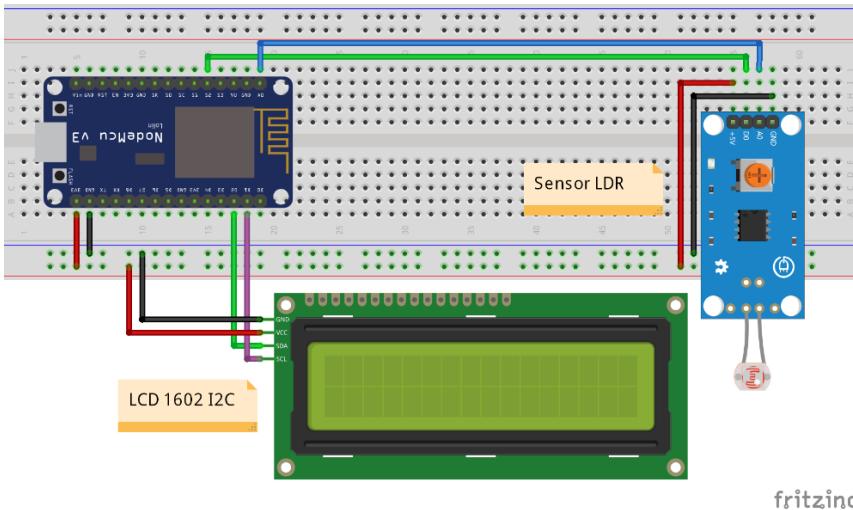
Saat cahaya jatuh ke semikonduktor, foton cahaya diserap oleh kisi semikonduktor dan sebagian energinya ditransfer ke elektron. Ini memberi beberapa energi yang cukup untuk melepaskan diri dari kisi kristal sehingga mereka kemudian dapat menghantarkan listrik. Ini menghasilkan penurunan resistansi semikonduktor.

Prosesnya progresif, semakin banyak cahaya yang menyinari semikonduktor LDR, semakin banyak elektron yang dilepaskan untuk menghantarkan listrik dan resistansi semakin turun.

Kebutuhan Bahan

Bahan	Jumlah	Nilai	Keterangan
Sensor LDR	1 pcs		
Modul LCD I ₂ C	1 pcs		

Rancangan



fritzing

Program yang akan dibangun adalah bagaimana mengetahui tingkat intensitas cahaya lingkungan yang ditangkap oleh sensor LDR. Pada module LDR terdapat dua luaran yaitu A0 dan D0. Pin A0 menghasilkan nilai luaran analog dengan rentang antara 0 sampai 1023, sedangkan pin D0 adalah luaran digital dengan kondisi HIGH (ada cahaya) atau LOW (tidak ada cahaya).

Nilai intensitas cahaya akan ditampilkan ke LCD. Sekaligus mengklasifikasi intensitas cahaya tersebut menjadi string redup, sedang dan terang. Redup bila nilai lebih dari 400, sedang bila nilai antara 200 sampai 400, dan terang bila nilai kurang dari 200.

Kode Program

```
#include <LiquidCrystal_I2C.h> //library LCD

#define AnalogLDR A0 // Intensitas cahaya,
analog: 0 - 1023
#define DigitalLDR 9 // Status terkena cahaya
atau tidak

LiquidCrystal_I2C lcd(0x27, 16, 2); //instan
LCD I2C
```

Kode Program

```
void setup() {  
    Serial.begin(9600);  
  
    // Set Input Sensor LDR  
    pinMode(DigitalLDR, INPUT);  
  
    lcd.begin(); // Inisialisasi LCD  
    lcd.clear();  
    lcd.setCursor(0, 0);  
    lcd.print("Sensor");  
    lcd.setCursor(0, 1);  
    lcd.print("Cahaya LDR");  
    delay(3000);  
    lcd.clear();  
}  
  
void loop() {  
    IntensitasCahaya();  
}  
  
void IntensitasCahaya() {  
    String intensitasCahaya = "";  
    int nilaiDigitalLDR =  
    digitalRead(DigitalLDR);  
  
    //nilai digital D0 dari LDR  
    if (nilaiDigitalLDR == HIGH) {  
        Serial.print("LDR terkana cahaya");  
    } else {  
        Serial.print("LDR tidak terkena cahaya");  
    }  
  
    //intensitas cahaya LDR, input analog 0-1023  
    int nilaiAnalogLDR = analogRead(AnalogLDR);  
  
    if (nilaiAnalogLDR < 200) {  
        Serial.println("Nilai Intensitas = " +  
        String(nilaiAnalogLDR));  
        Serial.println("Cahaya Terang");  
        intensitasCahaya = "Terang";  
    }  
}
```

Kode Program

```
 } else if (nilaiAnalogLDR >= 200 &&
nilaiAnalogLDR < 400) {
    Serial.println("Nilai Intensitas = " +
String(nilaiAnalogLDR));
    Serial.println("Cahaya Sedang");
    intensitasCahaya = "Sedang";
} else if (nilaiAnalogLDR >= 400) {
    Serial.println("Nilai Intensitas = " +
String(nilaiAnalogLDR));
    Serial.println("Cahaya Redup");
    intensitasCahaya = "Redup";
}

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("LDR: " + String(nilaiAnalogLDR));
lcd.setCursor(0, 1);
lcd.print(intensitasCahaya);

delay(1000);
}
```

3.5.5 Pembacaan Kartu RFID Mifare RC522

Kartu RFID (*Radio Frequency Identification*) berfungsi sebagai kunci digital karena setiap kartu RFID memiliki identifikasi unik yang disebut TAG. RFID menggunakan sistem identifikasi melalui transmisi gelombang radio, oleh karena itu dibutuhkan minimal dua buah perangkat yaitu yang TAG dan READER. Saat pemindaian data, READER membaca sinyal yang diberikan oleh RFID TAG.

TAG melekat pada obyek berbentuk kartu atau *key-chain* yang akan diidentifikasi oleh READER. TAG dapat berupa perangkat pasif atau aktif. TAG pasif artinya tanpa tenaga battery dan TAG aktif artinya menggunakan tenaga battery. TAG pasif lebih banyak digunakan karena murah dan mempunyai ukuran lebih kecil. RFID TAG dapat berupa perangkat read-only yang berarti hanya dapat dibaca saja atau perangkat read-write yang berarti dapat dibaca dan ditulis ulang untuk memperbarui data.

Terdapat dua jenis frekwensi kerja RFID yang banyak tersedia, yaitu 13,56MHz dan 125Khz, sehingga Anda perlu memperhatikan kesesuaian frekwensi antara RFID TAG dan RFID Reader yang dibutuhkan.



Kartu RFID



Key Chain RFID



TAG dan RFID Reader



Sensor RFID 13,56 Mhz

Skenario projek yang akan dirancang adalah sebagai berikut:

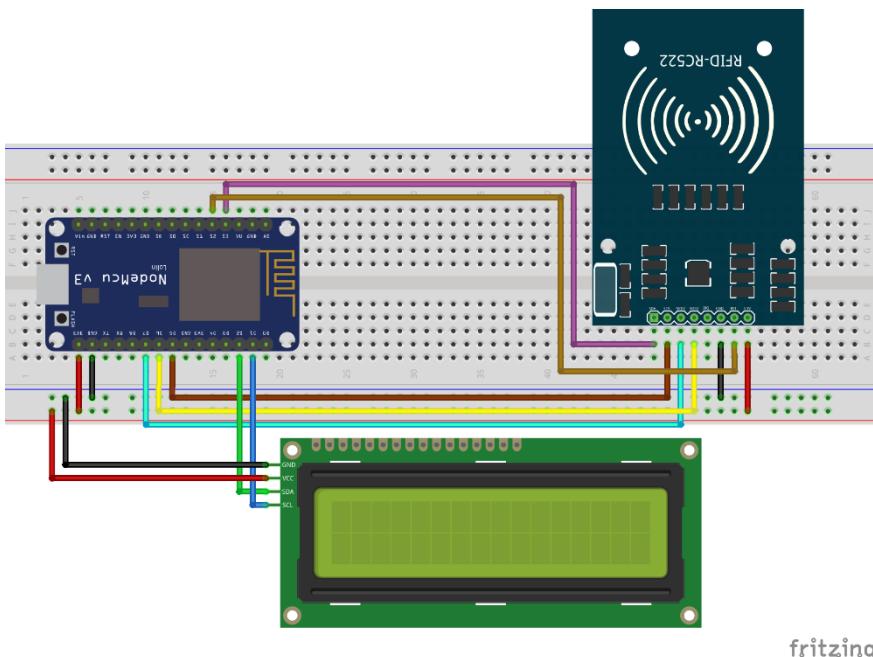
- Membaca TAG kartu RFID yang akan ditampilkan pada Serial Monitor.
- Kode TAG yang telah dibaca kemudian disimpan ke variabel melalui program.
- Untuk pengujian keabsahan TAG maka akan digunakan kartu RFID yang belum tercatat.
- Kartu RFID / Key Chain didekatkan dengan modul RFID Reader (Module MFRC522).
- Jika TAG pada kartu sesuai maka LED Hijau akan menyala dan pada LCD muncul pesan "Password Benar!"

- Jika TAG pada kartu tidak sesuai maka LED Merah akan menyala, buzzer speaker berbunyi dan pada LCD muncul pesan “Password Salah!”

Kebutuhan Bahan

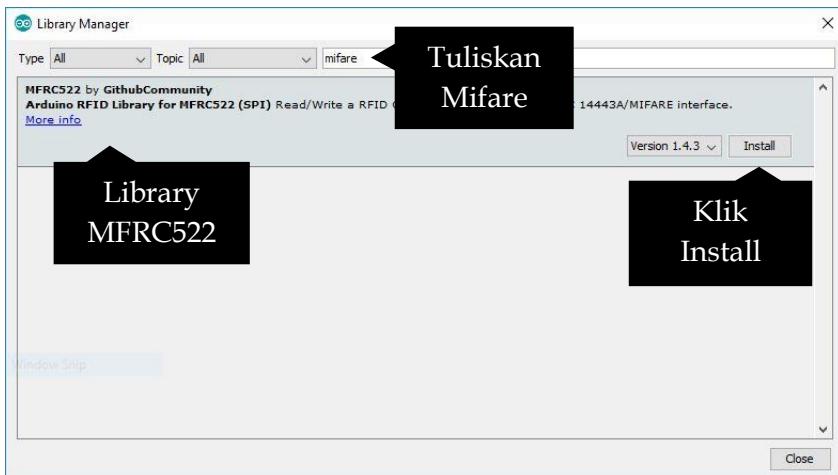
Bahan	Jumlah	Nilai	Keterangan
Module RFID Reader Mifare MFRC522	1 pcs	Frek. 13,56Mhz	
S50 Standard Blank Card	1 pcs	Frek. 13,56Mhz	
S50 Key Key Chain	1 pcs	Frek. 13,56Mhz	
LCD 1602 Serial I ₂ C	1 pcs		
LED 5MM	3 pcs		
Resistor	3 pcs	220 Ohm	

Rancangan



fritzing

Projek ini memerlukan library LiquidCrystal_I2C (telah diinstall pada projek sebelumnya) dan library MFRC522. Library MFRC522 dapat di-download di <https://github.com/miguelbalboa/rfid>, kemudian install file .zip-nya dari menu Sketch→Include Library→Add .zip Library. Libaray MFRC522 bisa juga di-install secara online melalui menu Sketch→Include Library→Manage libraries, kemudian cari library dengan kata kunci "mifare" seperti tampak pada gambar di bawah ini.



Menambahkan Library MFRC522

Kode Program

```
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>

#define pinReset 9 // Reset rfid
#define pinSS 10 // SS/SDA rfid

// Set alamat LCD ke 0x27 untuk ukuran tampilan
// 16 karakter 2 baris
LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(pinSS, pinReset);

//Sesuaikan TAG ID kartu Anda
String PasswordBenar[] = {"84206dcf",
"aa80a716", "Password3"};

void setup() {
  Serial.begin(9600);

  SPI.begin();
  mfrc522.PCD_Init();

  lcd.begin(); // inisialisasi LCD
  lcd.clear();
```

Kode Program

```
lcd.backlight(); // aktifkan backlight dan  
siap tampilan pesan  
lcd.setCursor(0, 0);  
lcd.print("    Masukkan      ");  
lcd.setCursor(0, 1);  
lcd.print("    Password      ");  
}  
  
void loop() {  
    if ( ! mfrc522.PICC_IsNewCardPresent() || !  
mfrc522.PICC_ReadCardSerial() )  
    {  
        delay(50);  
        Serial.println("RFID belum siap...");  
        return;  
    }  
  
    String content = "";  
    for (byte i = 0; i < mfrc522.uid.size; i++)  
    {  
  
content.concat(String(mfrc522.uid.uidByte[i] <  
0x10 ? "0" : ""));  
  
content.concat(String(mfrc522.uid.uidByte[i],  
HEX));  
    }  
  
    Serial.println("Kode TAG: " + content);  
    if (content != "")  
    {  
        //cek pembacaan TAG ID di serial monitor  
        //kemudian catat sebagai password  
        Serial.println("TAG ID : " + content);  
  
        boolean statusPassword = false;  
        // Cek keabsahan password  
        for (int i = 0; i < sizeof>PasswordBenar) -  
1; i++) {  
            if (PasswordBenar[i] == content)  
            {
```

Kode Program

```
//jika kartu RFID benar TAG 84206dcf
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(" Selamat      ");
lcd.setCursor(0, 1);
lcd.print(" Password Benar ");

//ganti status password menjadi true
statusPassword = true;
// jika password ditemukan maka
hentikan perulangan
    // tidak perlu meneruskan pencarian
password
    break;
} else
{
    statusPassword = false;
}
}

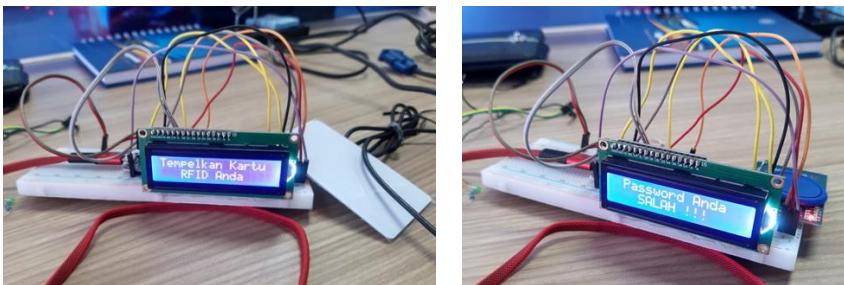
// beri pesan kesalahan sekali saja
// sehingga harus dikeluarkan dari loop
if (statusPassword == false) {
    //jika kartu RFID selain TAG 84206dcf
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Password Anda ");
    lcd.setCursor(0, 1);
    lcd.print(" SALAH !!!!");
}

//tunda 4 detik
delay(4000);

//beri pesan baru
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Tempelkan Kartu");
lcd.setCursor(0, 1);
lcd.print("   RFID Anda   ");
}
```

Kode Program
}

Hasil kurang lebih seperti berikut

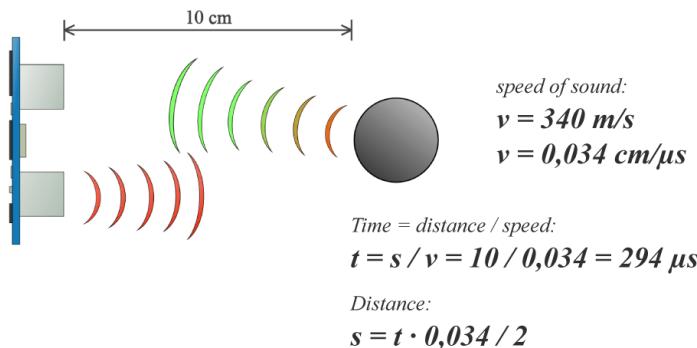


3.5.6 Hitung Jarak Dengan Ultrasonic HC-SR04

Sensor ultrasonic berkerja dengan cara memancarkan gelombang ultrasonic pada frekwensi 40.000 Hertz yang merambat di udara. Jika ditemukan obyek penghalang dijalurnya maka gelombang tersebut akan dipantulkan ke sensor penerima (Triger - Echo).

Jarak antara sensor dan obyek penghalang dapat dihitung berdasarkan waktu tempuh perambatan gelombang mulai sinyal dikirim, dipantulkan karena menabrak penghalang, sampai sinyal diterima kembali oleh sensor ultrasonic.

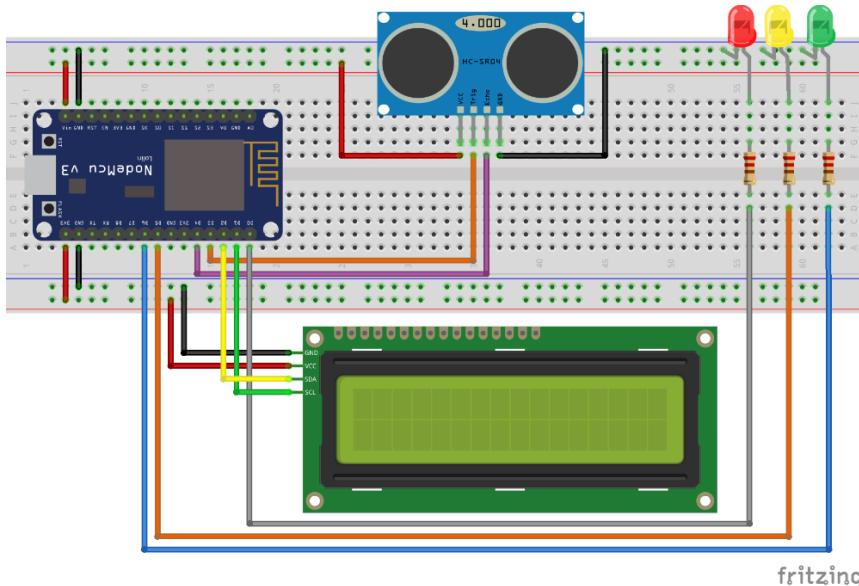
Misalnya, jika obyek berjarak 10 cm dari sensor dan kecepatan suara adalah 340 m/s atau $0,034 \text{ cm}/\mu\text{s}$ maka gelombang suara akan menempuh sekitar $294 \mu\text{s}$ (mikrodetik). Namun apa yang akan diperoleh dari pin Echo adalah dua kali lipat jumlahnya karena gelombang suara melaju ke depan dan memantul ke belakang. Jadi untuk mendapatkan jarak dalam satuan cm kita perlu mengalikan nilai waktu(t) perjalanan yang diterima dari pin echo sebesar $0,034$ dan membaginya dengan 2 .



Perhitungan Jarak antar Sensor dan Obyek Penghalang

Projek berikut ini cukup sederhana yaitu menghitung jarak antara sensor ultrasonic HC-SR04 dengan benda yang akan ditembak oleh gelombang ultrasonic. Setelah melalui proses perhitungan jarak, kemudian hasil jarak yang diperoleh akan ditampilkan ke LCD. Tampilan pada LCD berisi nilai jarak dalam satuan sentimeter dan inch. LCD menampilkan keterangan jika jarak kurang dari 30cm, jarak antara 30cm sampai 1m dan jarak lebih dari 1 meter. Ketiga rentang jarak tersebut akan memicu LED untuk hidup.

Rancangan



Kode Program

```
#include <LiquidCrystal_I2C.h>

#define pinTrigger 0      // pin trigger HC-SR04
#define pinEcho 2         // pin echo HC-SR04
#define pinLEDMerah 16    // pin LED Merah
#define pinLEDKuning 14   // pin LED Kuning
#define pinLEDHijau 12    // pin LED Hijau

// inisialisasi instance objek
LiquidCrystal_I2C lcd(0x27, 16, 2);

long duration;
int distanceCm, distanceInch;

void setup() {
    // baudrate serial monitor
    Serial.begin(9600);

    // mode pin
    pinMode(pinLEDMerah, OUTPUT);
    pinMode(pinLEDKuning, OUTPUT);
    pinMode(pinLEDHijau, OUTPUT);
    pinMode(pinTrigger, OUTPUT); // set pin
    trigger sebagai output
    pinMode(pinEcho, INPUT);    // set pin echo
    sebagai input

    // set semua LED mati
    digitalWrite(pinLEDMerah, LOW);
    digitalWrite(pinLEDKuning, LOW);
    digitalWrite(pinLEDHijau, LOW);

    // konfigurasi LCD1602 I2C
    // tampilkan tulisan awal
    lcd.begin();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Ukur Jarak");
    lcd.setCursor(0, 1);
    lcd.print("Ultrasonic");
    delay(3000);
```

Kode Program

```
lcd.clear();  
}  
  
void loop() {  
    HitungJarak();  
}  
  
void HitungJarak() {  
    // Membersihkan pin pinTrigger selama 2  
    microdetik  
    digitalWrite(pinTrigger, LOW);  
    delayMicroseconds(2);  
  
    // Set pinTrigger menjadi HIGH selama 10  
    microdetik  
    digitalWrite(pinTrigger, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(pinTrigger, LOW);  
  
    // Membaca pinEcho, mengembalikan waktu  
    perjalanan gelombang suara dalam mikrodetik  
    duration = pulseIn(pinEcho, HIGH);  
    distanceCm = duration * 0.034 / 2; // Jarak  
    dalam CM  
    distanceInch = duration * 0.0133 / 2; //  
    Jarak dalam INCH  
  
    String Jarak = "";  
    if (distanceCm <= 30) {  
        Jarak = "Jarak kurang dari atau sama dengan  
        30Cm";  
        digitalWrite(pinLEDMerah, HIGH);  
        digitalWrite(pinLEDKuning, LOW);  
        digitalWrite(pinLEDHijau, LOW);  
    } else if (distanceCm > 30 && distanceCm <=  
    100) {  
        Jarak = "Jarak lebih dari 30Cm tapi kurang  
        dari 1 Meter";  
        digitalWrite(pinLEDMerah, HIGH);  
        digitalWrite(pinLEDKuning, HIGH);  
        digitalWrite(pinLEDHijau, LOW);  
    }
```

Kode Program

```
 } else if (distanceCm > 100) {  
     Jarak = "Jarak lebih dari 1 meter";  
     digitalWrite(pinLEDMerah, HIGH);  
     digitalWrite(pinLEDKuning, HIGH);  
     digitalWrite(pinLEDHijau, HIGH);  
 }  
  
 Serial.println("Jarak: " + String(distanceCm)  
 + "Cm atau " + String(distanceInch) + "Inch");  
 Serial.println("Keterangan: " + Jarak);  
  
 lcd.clear();  
 // LCD baris pertama  
 lcd.setCursor(0, 0);  
 lcd.print("Jrk: " + String(distanceCm) + "Cm-  
 " + String(distanceInch) + "In");  
  
 // LCD baris kedua  
 lcd.setCursor(0, 1);  
 lcd.print("Jarak: " + Jarak);  
  
 delay(1000);  
 }
```

BAB 4

Komunikasi MCU Dengan Socket TCP

Pada bab ini akan dipelajari bagaimana mengkomunikasikan antara “Things/Smart Device” yang diwakili oleh microcontroller NodeMCU dengan berbagai Aplikasi dengan platform berbeda.

Komunikasi antara keduanya dijalin menggunakan socket TCP yang masih keluarga dari protokol tranpor http. Komunikasi ini tergolong tradisional karena belum menerapkan konsep manajemen pesan seperti *broker message* pada protokol MQTT. Namun socket TCP sampai saat ini masih sering digunakan untuk berbagai jenis keperluan, dimana sumberdaya tidak menjadi masalah karena kita ketahui bahwa protokol aplikasi http memiliki *overhead* sumberdaya yang cukup besar.

Rancangan projek disisi “Things” menggunakan board NodeMCU yang dibuat sekali, kemudian dikomunikasikan ke berbagai jenis aplikasi dengan platform berbeda seperti aplikasi desktop C#, aplikasi desktop Java, dan aplikasi Android.

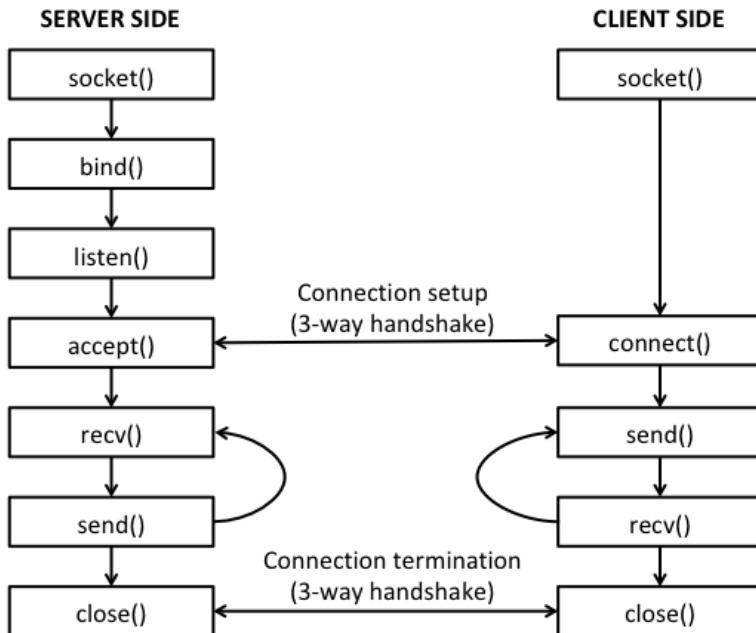
Disumsikan bahwa Anda telah memahami konsep-konsep dasar tentang jaringan komputer, termasuk routing jika diperlukan, semua mengacu pada protokol network model TCP/IP. Anda diharapkan telah menguasai bahasa pemrograman C#, Java dan Android, karena kita tidak mungkin membahas setiap bahasa pemrograman dalam buku ini.

4.1 Pemrograman Socket TCP

Pemrograman socket adalah cara menghubungkan dua node pada jaringan untuk berkomunikasi satu sama lain. Satu socket (node) mendengarkan pada port tertentu di sebuah IP, sementara socket lainnya menjangkau yang lain untuk membentuk koneksi. Server membentuk socket listener (pendengar) sementara klien menjangkau server.

Cara kerja socket, yang dapat dijelaskan sebagai berikut: Panggilan sistem **socket()** membuat soket baru. Di sisi server, **bind()** digunakan untuk mengasosiasikan socket dengan salah satu alamat

IP server dan nomor port yang akan didengarkan. Ketika ini selesai, socket diaktifkan dengan panggilan sistem **Listen()**.



Cara Kerja Socket

Panggilan sistem **accept()** memblokir hingga klien terhubung (menggunakan **connect()** pada akhirnya). Setelah ini, data dapat dikirim dan diterima menggunakan contoh ini **send()** dan **recv()** (**write()** dan **read()** juga akan berfungsi).

Perhatikan bahwa setelah koneksi dibuat, tidak ada perbedaan khusus antara *peers* dengan sebelum koneksi ketika satu menunggu koneksi dari yang lain. Kedua ujung dapat mengirim dan menerima data dalam urutan apa pun; prosedur yang tepat tentang bagaimana hal ini dilakukan biasanya ditentukan oleh protokol level aplikasi apa yang digunakan. Sebagai contoh, jika server adalah server web dan klien adalah browser web, biasanya klien mengirimkan permintaan dan server meresponsnya sesuai dengan protokol HTTP.

Ketika salah satu node ingin mengakhiri koneksi, ia menggunakan **close**, yang akan memulai jabat tangan tiga arah untuk penghentian koneksi.

Langkah-langkah untuk membuat socket TCP di sisi klien:

- Buat socket menggunakan fungsi `socket()`;
- Hubungkan socket ke alamat server menggunakan fungsi `connect()`;
- Mengirim dan menerima data melalui fungsi `read()` dan `write()`.
- Tutup koneksi dengan menggunakan fungsi `close()`.

Langkah-langkah dalam membangun socket TCP di sisi server:

- Buat socket dengan fungsi `socket()`;
- Ikat socket ke alamat menggunakan fungsi `bind()`;
- Dengarkan koneksi dengan fungsi `listening()`;
- Terima koneksi dengan panggilan sistem fungsi `accept()`. Panggilan ini biasanya memblokir hingga klien terhubung dengan server.
- Mengirim dan menerima data melalui `kirim()` dan `terima()`.
- Tutup koneksi dengan menggunakan fungsi `close()`.

Agar tidak timbul pemblokiran telalu lama oleh server ketika klien berusaha menjalin koneksi dengannya, namun karena terkendala suatu hal maka sistem (disediakan oleh API Socket) memberi batas waktu tertentu untuk putus dengan sendirinya (fungsi `timeout`). Sehingga ketika klien lain berusaha untuk menjalin hubungan dengan server dapat melakukannya.

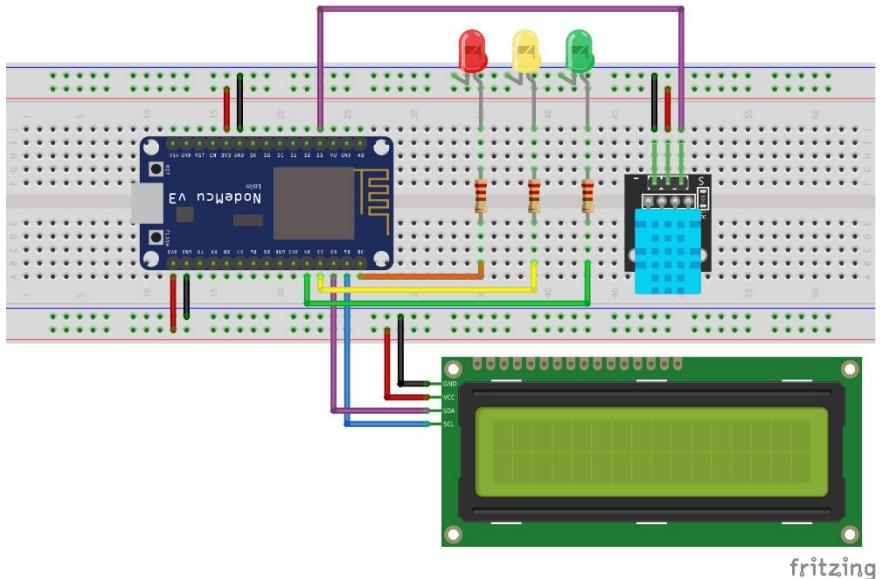
Optimasi lain yang dapat dilakukan oleh socket adalah dengan menggunakan `thread`, yaitu kemampuan untuk membagi kerja secara paralel antara server-socket dengan banyak client-socket. Otomatis hal ini akan mengurangi antrian *request* dari klien ke server karena *job/task* dapat ditangani secara bersamaan. Istilah ini disebut dengan *asynchronous socket*, sedangkan request klien yang dilaksanakan satu per satu oleh server dan tidak akan berpindah ke request klien lainnya selama request pertama belum selesai disebut dengan *synchronous socket*.

Berikut beberapa referensi tentang model client-server Socket TCP/IP yang layak menjadi sumber bacaan:

- <https://users.cs.duke.edu/~chase/cps196/slides/sockets.pdf>
- <https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf>
- <https://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html>

4.2 Rancangan “Things” NodeMCU

Rancangan



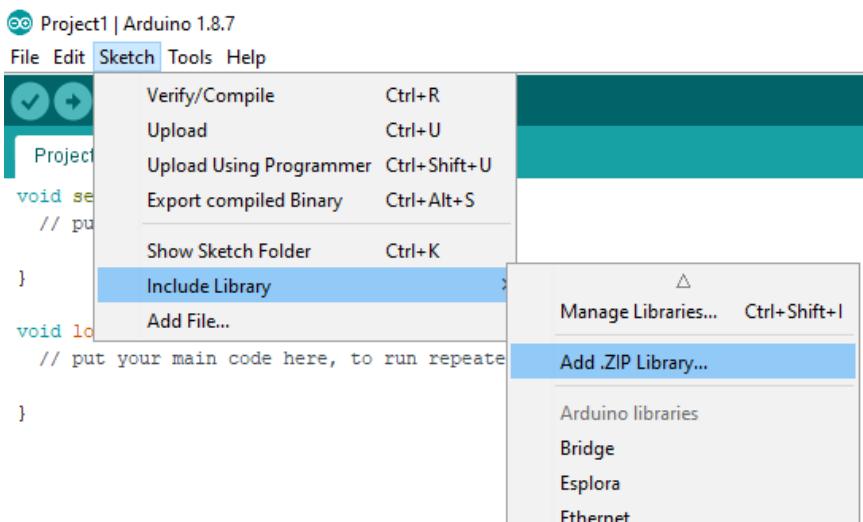
Buat project baru dari aplikasi IDE Arduino, simpan dengan nama “SocketNodeMCU”.

Projek ini membutuhkan beberapa library tambahan. Untuk menambahkan library ke IDE Arduino dengan 2 cara, yaitu:

- Cara pertama, install secara online dari internet melalui menu **Sketch → Include Library → Manage Libraries...**, tuliskan library yang dikehendaki kemudian pilih salah satu library untuk diinstal.

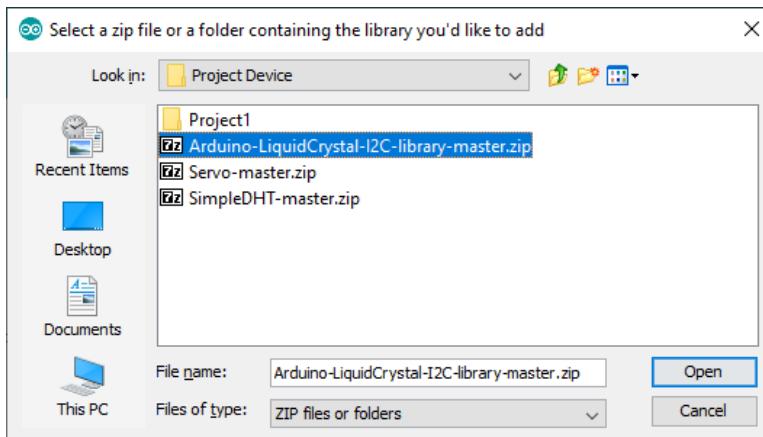


- Cara kedua, download file zip library terlebih dahulu, kemudian upload file zip tersebut melalui menu **Sketch** → **Include Library** → **Add ZIP Library**.

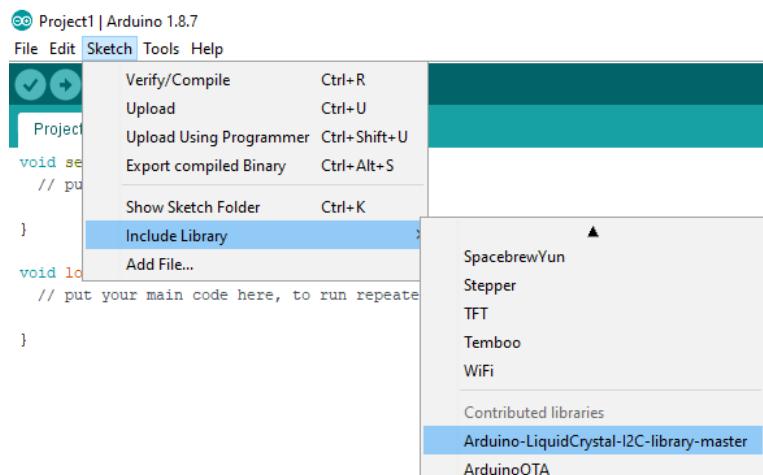


Berikut library yang dibutuhkan:

- Library Arduino-LiquidCrystal-I2C-library-master.zip, download di <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>.
- Library SimpleDHT.zip, download di <https://github.com/winlinvip/SimpleDHT>.



- Menambahkan library Arduino Liquid Crystal dan Servo ke program melalui menu **Sketch** → **Include Library** → *nama_library*. Tambahkan tiga library yang telah di-install sebelumnya dengan cara yang sama.



Berikut ini merupakan program standar microcontroller sebelum dikomunikasikan dengan perangkat lain. Pastikan bahwa semua program berjalan dengan benar.

Kode Program

```
#include <SimpleDHT.h> //library sensor DHT11
#include <LiquidCrystal_I2C.h> //library LCD

#define pinLedMerah 16 //D0 pin led merah
#define pinLedKuning 0 //D3 pin led kuning
#define pinLedHijau 2 //D4 pin led hijau
#define pinDHT 10 // SD3 pin signal sensor
DHT

//instan LCD I2C
LiquidCrystal_I2C lcd(0x27, 16, 2);

//instan sensor dht11
SimpleDHT11 dht11(pinDHT);

//membuat karakter derajat custom
//https://maxpromer.github.io/LCD-Character-Creator/
byte derajat[] = {
    B01110,
    B10001,
    B10001,
    B10001,
    B01110,
    B00000,
    B00000,
    B00000
};

void setup() {
    Serial.begin(9600);

    pinMode(pinLedMerah, OUTPUT);
    pinMode(pinLedKuning, OUTPUT);
    pinMode(pinLedHijau, OUTPUT);
    pinMode(pinDHT, INPUT);
```

Kode Program

```
lcd.begin(); // Inisialisasi LCD
lcd.createChar(0, derajat);
lcd.backlight(); // Menghidupkan backlight
lcd.setCursor(0, 0);
lcd.print("Project");
lcd.setCursor(0, 1);
lcd.print("Socket TCP");
delay(3000);
lcd.clear();
}

void loop() {
    LEDBerjalan();
    KelembabanSuhu();
}

void LEDBerjalan() {
    digitalWrite(pinLedMerah, HIGH);
    digitalWrite(pinLedKuning, LOW);
    digitalWrite(pinLedHijau, LOW);
    Serial.println("LED Merah Hidup ... ");
    delay(1000);

    digitalWrite(pinLedMerah, LOW);
    digitalWrite(pinLedKuning, HIGH);
    digitalWrite(pinLedHijau, LOW);
    Serial.println("LED Kuning Hidup ... ");
    delay(1000);

    digitalWrite(pinLedMerah, LOW);
    digitalWrite(pinLedKuning, LOW);
    digitalWrite(pinLedHijau, HIGH);
    Serial.println("LED Hijau Hidup ... ");
    delay(1000);

    Serial.println();
}

void KelembabanSuhu() {
    byte temperature = 0;
    byte humidity = 0;
```

Kode Program

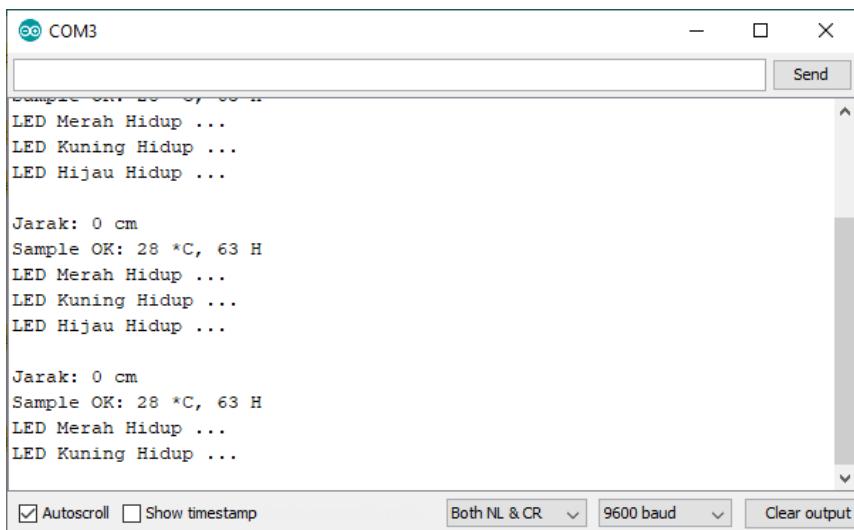
```
int err = SimpleDHTerrSuccess;
if ((err = dht11.read(&temperature,
&humidity, NULL)) != SimpleDHTerrSuccess)
{
    Serial.print("Pembacaan DHT11 gagal,
err=");
    Serial.println(err); delay(1000);
    return;
}

Serial.print("Sample OK: ");
Serial.print((int)temperature);
Serial.print(" *C, ");
Serial.print((int)humidity);
Serial.println(" H");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temperatur: " +
String((int)temperature));
lcd.write(0);
lcd.print("C ");
lcd.setCursor(0, 1);
lcd.print("Kelembaban: " +
String((int)humidity) + "H");

delay(1500);
}
```

Hasil pembacaan sensor dapat di-debug menggunakan Serial Monitor IDE Arduino di menu **Tools → Serial Monitor**.



Berdasarkan program dasar di atas akan dikembangkan menjadi aplikasi yang dapat berkomunikasi dengan perangkat lainnya menggunakan Socket TCP.

4.3 NodeMCU & Aplikasi Desktop C#

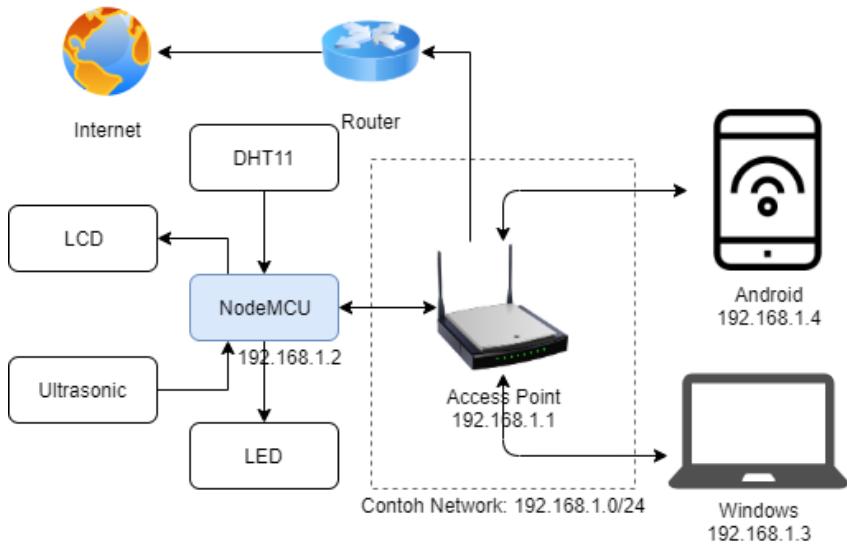
Projek selanjutnya adalah pengembangan dari “**SocketNodeMCU**” sebelumnya. Kita akan mengkomunikasikan NodeMCU dengan aplikasi pada komputer menggunakan Socket TCP.

Rancangan fisik microkontroler NodeMCU tetap mempertahankan rangkaian pada projek **SocketNodeMCU** sehingga tidak ada perubahan pengkabelan.

Topologi Jaringan

Alamat TCP pada gambar topologi adalah sekedar contoh. Dimana alamat IP Network-nya adalah 192.168.1.0/24 yang berarti ketersediaan alamat IP yang dapat digunakan adalah 254 (pelajari lebih lanjut mengenai subnet IP).

Oleh karena itu, untuk alamat alamat network disesuaikan dengan kondisi alamat IP DHCP router Anda masing-masing.



Topologi Jaringan

Di sini NodeMCU, Laptop dan Android diposisikan sebagai *station* yang terhubung ke *Access Point*. Alamat IP diset sebagai DHCP. Namun bila ditinjau dari sisi Socekt TCP maka NodeMCU diposisikan sebagai *Socket Server*, sedangkan Laptop dan Android diposisikan sebagai *Socket Client*.

4.3.1 NodeMCU Sebagai Socket Server

Tujuan projek ini adalah untuk mengendalikan NodeMCU dari aplikasi Laptop dan atau sebaliknya secara jarak jauh melalui jaringan Wi-Fi.

Adapun yang dapat dilakukan antara lain:

- Mengirim 2 baris teks string yang ditulis melalui 2 TextBox menggunakan aplikasi desktop C#.NET dari laptop. Kemudian menampilkan teks tersebut ke LCD NodeMCU secara jarak jauh.
- Menghidupkan atau mematikan LED Merah, LED Kuning dan LED Hijau yang terdapat pada NodeMCU melalui 3 jenis Radio Button menggunakan aplikasi desktop C#.NET dari laptop secara jarak jauh.

Terdapat satu tambahan library yang dibutuhkan pada program socket, yaitu “**ESP8266WiFi**”. Untuk menambahkannya lakukan dengan cara yang sama seperti sebelumnya.

Kode Program

NodeMCU Sebagai Server Socket
#include <ESP8266WiFi.h> #include <LiquidCrystal_I2C.h> #define pinLedMerah 16 // D0 pin led merah #define pinLedKuning 0 // D3 pin led kuning #define pinLedHijau 2 // D4 pin led hijau #define pinDHT 10 // SD3 pin signal sensor DHT //instan LCD I2C LiquidCrystal_I2C lcd(0x27, 16, 2); int port = 16375; // nomor port WiFiServer server(port); //Koneksi ke network Wifi AccessPoint //Sesuaikan SSID Wifi masing-masing const char *ssid = "Fanny 2004"; //Sesuaika password WiFi masing-masing const char *password = "fanny_200504"; void setup() { Serial.begin(9600); //pin LED pinMode(pinLedMerah, OUTPUT); pinMode(pinLedKuning, OUTPUT); pinMode(pinLedHijau, OUTPUT); KonfigurasiLCD(); KoneksiAP(); // inisialisasi koneksi ke AP }

NodeMCU Sebagai Server Socket

```
void loop()
{
    WiFiClient client = server.available();
    String pesan = "";
    if (client) {
        if (client.connected())
        {
            Serial.println("Client Connected");
        }

        while (client.connected()) {
            while (client.available() > 0) {
                //Serial.write(client.read());
                //membaca data dari client yang
terhubung
                delay(10);
                char c = client.read();
                pesan += String(c);
            }

            Serial.println(pesan);
            // filter string yang masuk untuk
menentukan status LED
            if (pesan == "Merah") {
                //LED merah hidup, lainnya mati
                digitalWrite(pinLedMerah, HIGH);
                digitalWrite(pinLedKuning, LOW);
                digitalWrite(pinLedHijau, LOW);
            } else if (pesan == "Kuning") {
                //LED kuning hidup, lainnya mati
                digitalWrite(pinLedMerah, LOW);
                digitalWrite(pinLedKuning, HIGH);
                digitalWrite(pinLedHijau, LOW);
            } else if (pesan == "Hijau") {
                //LED hijau hidup, lainnya mati
                digitalWrite(pinLedMerah, LOW);
                digitalWrite(pinLedKuning, LOW);
                digitalWrite(pinLedHijau, HIGH);
            } else if (getValue(pesan, '#', 0) ==
"LCD") {

```

NodeMCU Sebagai Server Socket

```
//Cek apakah kata sebelum karakter "#" pertama adalah "LCD"
    //Jika benar maka potong string sesuai delimiter "#" untuk baris 1 dan baris 2
    //Kemudian tampilkan ke LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(getValue(pesan, '#', 1));
    lcd.setCursor(0, 1);
    lcd.print(getValue(pesan, '#', 2));
}

client.stop(); //Putuskan koneksi dengan client
    Serial.println("Client disconnected");
}
}

void KonfigurasiLCD() {
    lcd.begin(); // Inisialisasi LCD
    lcd.backlight(); // Menghidupkan backlight
    lcd.setCursor(0, 0);
    lcd.print("Project");
    lcd.setCursor(0, 1);
    lcd.print("Socket TCP");
    delay(3000);
    lcd.clear();
}

void KoneksiAP() {
    // Konfigurasi WIFI
    // Koneksikan NodeMCU ke AP
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    // Menunggu koneksi dengan AP
    Serial.println("Connecting to Wifi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
}
```

NodeMCU Sebagai Server Socket

```
delay(500);
}

Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);

Serial.print("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
Serial.print("Open Telnet and connect to
IP:");
Serial.print(WiFi.localIP());
Serial.print(" on port ");
Serial.println(port);
}

// Function splitter/pemisah string berdasarkan
karakter
// Misal karakter khusus yang dipilih '#' sebagai pemisahnya
String getValue(String data, char separator,
int index)
{
    // Splitting string diantara penanda karakter
    "#"
    int found = 0;
    int strIndex[] = {0, -1};
    int maxIndex = data.length() - 2;

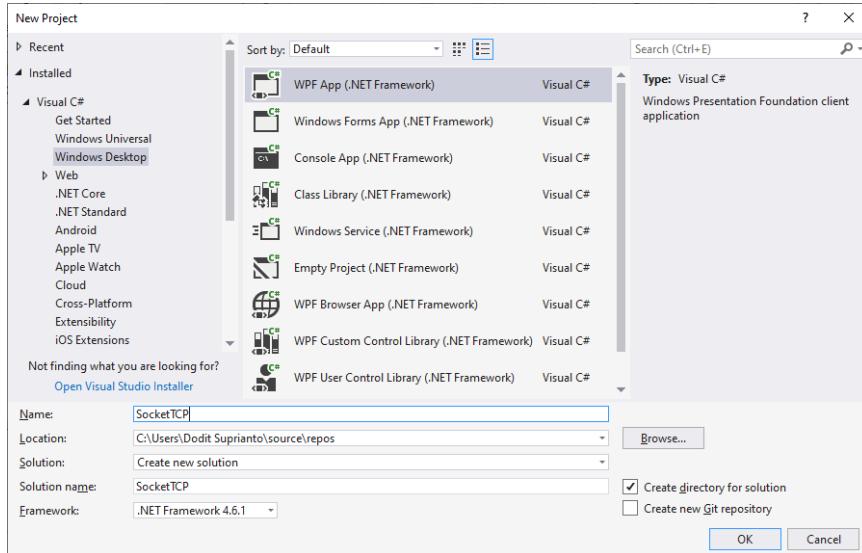
    for (int i = 0; i <= maxIndex && found <=
index; i++) {
        if (data.charAt(i) == separator || i ==
maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i + 1 :
i;
        }
    }
}
```

NodeMCU Sebagai Server Socket

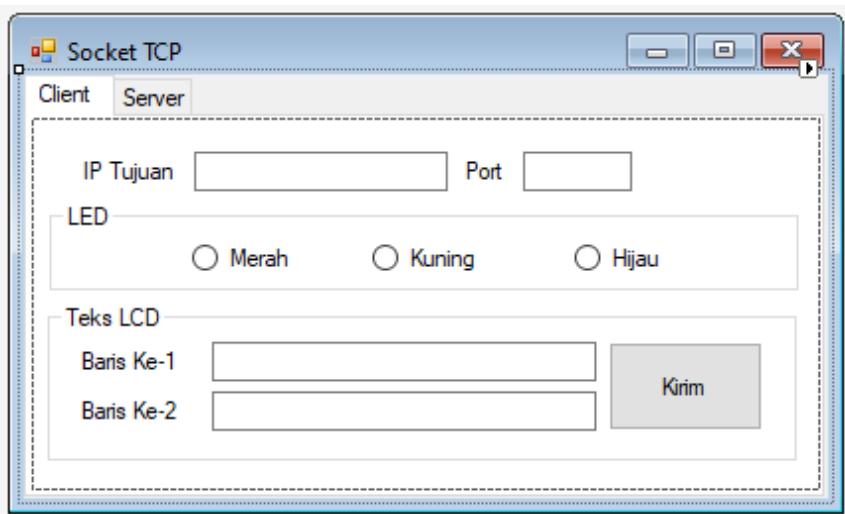
```
return found > index ?  
data.substring(strIndex[0], strIndex[1]) : "";  
}
```

Aplikasi C#.NET Dengan Visual Studio

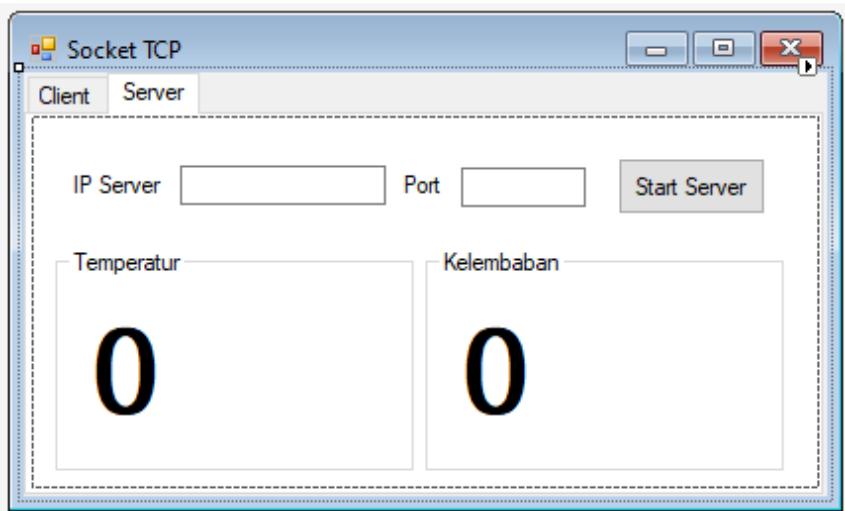
Buat Project C# baru bernama “SocketTCP” dari menu **File → New → Project**



Tambahkan beberapa komponen dari toolbox ke form seperti tampak di bawah



Tampilan Form Client (Socket Server)



Tampilan Form Server (Client Socket)

Berikut komponen-komponen, properties dan event yang dibutuhkan oleh Form.

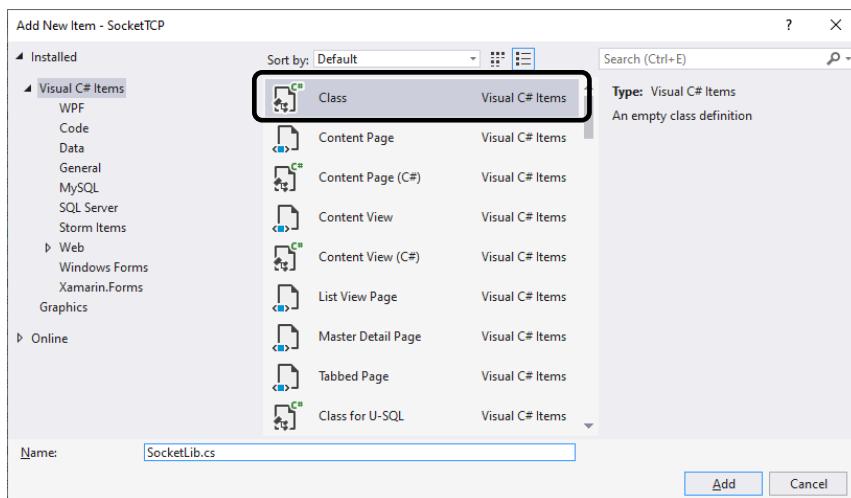
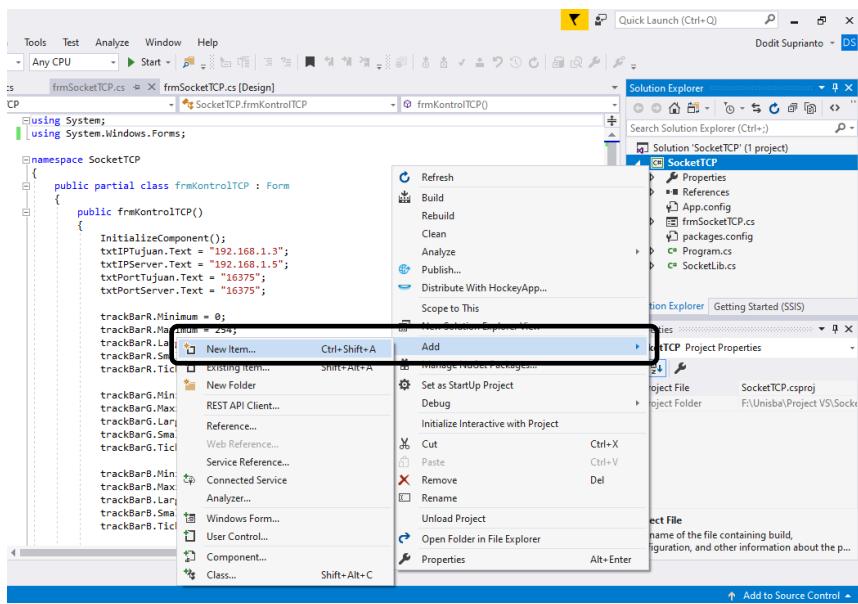
Komponen	Properties	Events
Form	Name: frmSocketTCP, Text: SocketTCP	

Komponen	Properties	Events
tabControl	Name: tabControl1	
tabPage	Name: tabPage1, Text: "Client" Name: tabPage2, Text: "Server"	
timer	Name: timer1, interval: 1000	Tick: timer1_Tick
Dalam lingkup "tabPage1" (Client)		
Label	Name: lblIPTujuan, Text: "IP Tujuan" Name: lblPortTujuan, Text: "Port"	
TextBox	Name: txtIPTujuan Name: txtPortTujuan	
GroupBox	Name: groupBoxLED, Text: "LED" Name: groupBoxRGB, Text: "Komposisi warna RGB" Name: groupBoxLCD, Text: "Teks LC"	
Dalam lingkup "groupBoxLED"		
RadioButton	Name: rdMerah, Text: "Merah" Name: rdKuning, Text: "Kuning" Name: rdHijau, Text: "Hijau"	Click: rdMerah_Click, rdKuning_Click, rdHijau_Click
Dalam lingkup "groupBoxLCD"		
Label	Name: lblBaris1, Text: "Baris Ke-1" Name: lblBaris2, Text: "Baris Ke-2"	
TextBox	Name: txtTeksDikirim1 Name: txtTeksDikirim2	
Button	Name: btnKirim, Text: "Kirim"	Click: btnKirim_Click
Dalam lingkup tabPage2 ("Server")		
Label	Name: lblIPServer, Text: "IP Server"	

Komponen	Properties	Events
	Name: lblPortServer, Text: "Port"	
TextBox	Name: txtIPServer Name: txtPortServer	
Button	Name: btnStartServer, Text: "Start Server"	Click: btnStartServer_Clic k
GroupBox	Name: groupBoxTemperatur , Text: "Temperatur" Name: groupBoxKelembaban , Text: "Kelembaban"	
Dalam lingkup “groupBoxTemperatur”		
Label	Name: lblTemp, Text: "0", Font Size: 45, Bold: True	
Dalam Lingkup “groupBoxKelembaban”		
Label	Name: lblHum, Text: "0", Font Size: 45, Bold: True	

Kode Program C#

Buat class baru bernama **SocketLib.cs** yang berisi rutin-rutin server socket dan client socket. Klik kanan pada projek **SocketTCP** → **Add** → **New Item**



class SocketLib.cs

```
using System;
using System.Text;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.Windows.Forms;
```

class SocketLib.cs

```
namespace SocketTCP
{
    public static class GlobalVariable
    {
        //public const string ipserver =
        "192.168.1.6"; //listening komputer
        //public const string iptujuan =
        "192.168.4.1";
        //public const int port = 16375;
        public static string ipserver;
        public static string iptujuan;
        public static int port;
        public static string pesanTerima;
    }

    class StateObject
    {
        public Socket workSocket = null;
        public const int BufferSize = 256;
        public byte[] buffer = new
byte[BufferSize];
        public StringBuilder sb = new
StringBuilder();
    }

    public static class SocketLib
    {
        public static ManualResetEvent
connectDone = new ManualResetEvent(false);
        public static ManualResetEvent sendDone
= new ManualResetEvent(false);
        public static ManualResetEvent
receiveDone = new ManualResetEvent(false);
        public static String response =
String.Empty;

        // ##########
        //          SOCKET CLIENT
        // ##########
    }
}
```

class SocketLib.cs

```
    public static void StartClient(string
pesan)
    {
        IPAddress ip =
IPAddress.Parse(GlobalVariable.iptujuan);
        IPEndPoint remoteEP = new
IPEndPoint(ip, GlobalVariable.port);
        Socket client = new
Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

        var result =
client.BeginConnect(remoteEP, new
 AsyncCallback(ConnectCallback), client);

result.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));
        System.Threading.Thread.Sleep(10);

        if (client.Connected)
        {
            Send(client, pesan);

System.Threading.Thread.Sleep(10);
            sendDone.WaitOne();

client.Shutdown(SocketShutdown.Both);
            client.Close();
        }
    }

    public static void
ConnectCallback(IAsyncResult ar)
{
    try
    {
        Socket client =
(Socket)ar.AsyncState;
        client.EndConnect(ar);
        connectDone.Set();
    }
}
```

class SocketLib.cs

```
        catch
        {
            //MessageBox.Show("Koneksi
Audio Kelas Terputus\n\rAudio Kelas Dalam
Kondisi Down", "Perangkat Client Down",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    public static void Send(Socket client,
String data)
{
    byte[] byteData =
Encoding.ASCII.GetBytes(data);
    client.BeginSend(byteData, 0,
byteData.Length, 0, new
 AsyncCallback(SendCallback), client);
}

private static void
SendCallback(IAsyncResult ar)
{
    try
    {
        Socket client =
(Socket)ar.AsyncState;
        int bytesSent =
client.EndSend(ar);
        sendDone.Set();
    }
    catch
    {

    }
}

public static void Receive(Socket
client)
{
    StateObject state = new
StateObject();
```

class SocketLib.cs

```
        state.workSocket = client;
        client.BeginReceive(state.buffer,
0, StateObject.BufferSize, 0, new
 AsyncCallback(ReceiveCallback), state);
    }

    private static void
ReceiveCallback(IAsyncResult ar)
{
    StateObject state =
(StateObject)ar.AsyncState;
    Socket client = state.workSocket;
    int bytesRead =
client.EndReceive(ar);

    if (bytesRead > 0)
    {

state.sb.Append(Encoding.ASCII.GetString(state.
buffer, 0, bytesRead));

client.BeginReceive(state.buffer, 0,
StateObject.BufferSize, 0, new
AsyncCallback(ReceiveCallback), state);
    }
    else
    {
        if (state.sb.Length > 1)
        {
            response =
state.sb.ToString();
        }
        receiveDone.Set();
    }
}

// ##########
//      SOCKET SERVER LISTENER
// #########
public static ManualResetEvent allDone
= new ManualResetEvent(false);
```

class SocketLib.cs

```
//Listening
public static void StartListening()
{
    IPAddress ipAddress =
IPAddress.Parse(GlobalVariable.ipserver);
    IPEndPoint localEndPoint = new
IPEndPoint(ipAddress, GlobalVariable.port);
    Socket listener = new
Socket(ipAddress.AddressFamily,
SocketType.Stream, ProtocolType.Tcp);
    try
    {
        listener.Bind(localEndPoint);
        listener.Listen(100);

        while (true)
        {
            allDone.Reset();
            //MessageBox.Show("Waiting
for a connection...");
            listener.BeginAccept(new
 AsyncCallback(AcceptCallback), listener);
            allDone.WaitOne();
        }
    }
    catch (Exception e)
    {
        MessageBox.Show(e.ToString());
    }
}

public static void
AcceptCallback(IAsyncResult ar)
{
    allDone.Set();

    // Get the socket that handles the
client request.
    Socket listener =
(Socket)ar.AsyncState;
```

class SocketLib.cs

```
    Socket handler =
listener.EndAccept(ar);

        // Create the state object.
        StateObject state = new
StateObject();
        state.workSocket = handler;
        handler.BeginReceive(state.buffer,
0, StateObject.BufferSize, 0, new
 AsyncCallback(ReadCallback), state);
    }

    public static void
ReadCallback(IAsyncResult ar)
{
    String content = String.Empty;

    StateObject state =
(StateObject)ar.AsyncState;
    Socket handler = state.workSocket;
    int bytesRead =
handler.EndReceive(ar);

    if (bytesRead > 0)
    {

state.sb.Append(Encoding.ASCII.GetString(state.
buffer, 0, bytesRead));

        content = state.sb.ToString();
        GlobalVariable.pesanTerima =
content;
    }
}
}
```

Kode Program frmSocketTCP.cs

```
using System;
using System.Windows.Forms;
```

Kode Program frmSocketTCP.cs

```
namespace SocketTCP
{
    public partial class frmKontrolTCP : Form
    {
        public frmKontrolTCP()
        {
            InitializeComponent();
            //alamat IP sesuai dengan kondisi
masing-masing
            txtIPTujuan.Text = "192.168.1.3";
            txtIPServer.Text = "192.168.1.5";
            txtPortTujuan.Text = "16375";
            txtPortServer.Text = "16375";

            trackBarR.Minimum = 0;
            trackBarR.Maximum = 254;
            trackBarR.LargeChange = 30;
            trackBarR.SmallChange = 20;
            trackBarR.TickFrequency = 30;

            trackBarG.Minimum = 0;
            trackBarG.Maximum = 254;
            trackBarG.LargeChange = 30;
            trackBarG.SmallChange = 20;
            trackBarG.TickFrequency = 30;

            trackBarB.Minimum = 0;
            trackBarB.Maximum = 254;
            trackBarB.LargeChange = 30;
            trackBarB.SmallChange = 20;
            trackBarB.TickFrequency = 30;

            txtTeksDikirim1.MaxLength = 16;
            txtTeksDikirim2.MaxLength = 16;
        }

        private void btnKirim_Click(object
sender, EventArgs e)
        {
```

Kode Program frmSocketTCP.cs

```
        if (txtPortTujuan.Text != "" &&
txtPortTujuan.Text != "")
{
    GlobalVariable.iptujuan =
txtIPTujuan.Text;
    GlobalVariable.port =
Convert.ToInt16(txtPortTujuan.Text);
    SocketLib.StartClient ("LCD#" +
txtTeksDikirim1.Text + "#" +
txtTeksDikirim2.Text + "#");
}
else
{
    MessageBox.Show ("Alamat IP dan
Port harus diisi!", "Kesalahan");
}
}

private void rdMerah_Click(object
sender, EventArgs e)
{
    if (txtPortTujuan.Text != "" &&
txtPortTujuan.Text != "")
{
    GlobalVariable.iptujuan =
txtIPTujuan.Text;
    GlobalVariable.port =
Convert.ToInt16(txtPortTujuan.Text);
    SocketLib.StartClient ("Merah");
}
else
{
    MessageBox.Show ("Alamat IP dan
Port harus diisi!", "Kesalahan");
}
}

private void rdHijau_Click(object
sender, EventArgs e)
{
```

Kode Program frmSocketTCP.cs

```
        if (txtPortTujuan.Text != "" &&
txtPortTujuan.Text != "")
{
    GlobalVariable.iptujuan =
txtIPTujuan.Text;
    GlobalVariable.port =
Convert.ToInt16(txtPortTujuan.Text);
    SocketLib.StartClient("Hijau");
}
else
{
    MessageBox.Show("Alamat IP dan
Port harus diisi!", "Kesalahan");
}
}

private void rdKuning_Click(object
sender, EventArgs e)
{
    if (txtPortTujuan.Text != "" &&
txtPortTujuan.Text != "")
    {
        GlobalVariable.iptujuan =
txtIPTujuan.Text;
        GlobalVariable.port =
Convert.ToInt16(txtPortTujuan.Text);

        SocketLib.StartClient("Kuning");
    }
    else
    {
        MessageBox.Show("Alamat IP dan
Port harus diisi!", "Kesalahan");
    }
}

private void
btnStartServer_Click(object sender, EventArgs
e)
{
```

Kode Program frmSocketTCP.cs

```
GlobalVariable.ipserver =
txtIPServer.Text;
GlobalVariable.port =
Convert.ToInt16(txtPortServer.Text);
System.Threading.Thread thread =
new
System.Threading.Thread(SocketLib.StartListening);
thread.Start();
timer1.Enabled = true;
}

private void timer1_Tick(object sender,
EventArgs e)
{
    if (GlobalVariable.pesanTerima != null)
    {
        string data =
GlobalVariable.pesanTerima;
        string[] words =
data.Split('#');
        lblTemp.Text = words[0] + "°C";
        lblHum.Text = words[1] + "H";
    }
}
}
```

Jalankan kode program di atas dengan menekan tombol F5. Kemudian coba radio button yang tersedia untuk menghidupkan dan mematikan LED secara jarak jauh. Coba juga untuk menuliskan pesan pada 2 text box yang ada kemudian klik tombol kirim, maka pada LCD NodeMCU akan muncul tulisan sesuai dengan apa yang dituliskan pada textbox.

4.3.2 NodeMCU Sebagai Socket Client

Bila dibandingkan dengan project sebelumnya maka pada projek kali ini adalah kebalikannya. Aplikasi pada laptop bertugas sebagai

Socket Server, sedangkan NodeMCU berposisi sebagai Client Socket.

Aplikasi yang dibangun mengacu pada topologi jaringan yang sama dengan sebelumnya.

Karena aplikasi disisi laptop (Visual Studio C#) telah dibuat form *user interface*, program user interface dan class library Socket-nya maka kita tidak perlu membahasnya ulang. Perubahan hanya terjadi pada program di sisi NodeMCU.

Rancangan dan pengkabelan NodeMCU juga tidak berubah, sehingga kita dapat membahasnya ulang.

Tujuan projek ini adalah membaca data temperatur dan kelembaban lingkungan yang berasal dari sensor DHT11 di sisi NodeMCU, kemudian mengirimkan data tersebut ke aplikasi C#.NET pada laptop secara berkala.

Dimana NodeMCU bertindak sebagai client socket dan laptop bertindak sebagai server socket.

Kode Program

NodeMCU Sebagai Client Socket

```
#include <ESP8266WiFi.h>
#include <SimpleDHT.h>
#include <LiquidCrystal_I2C.h>

#define pinDHT 10 // SD3 pin signal sensor DHT

#ifndef STASSID
#define STASSID "Fanny 2004"
#define STAPSK "fanny_200504"
#endif

const char* ssid      = STASSID;
const char* password = STAPSK;

//sesuaikan IP wireless pada laptop sebagai
Socket Server
//cek dengan command "IPCONFIG"
const char* host = "192.168.1.6";
const uint16 t port = 16375;
```

NodeMCU Sebagai Client Socket

```
//instan sensor dht11
SimpleDHT11 dht11(pinDHT);

//instan LCD I2C
LiquidCrystal_I2C lcd(0x27, 16, 2);

//membuat karakter derajat custom
//https://maxpromer.github.io/LCD-Character-
Creator/
byte derajat[] = {
    B01110,
    B10001,
    B10001,
    B10001,
    B01110,
    B00000,
    B00000,
    B00000
};

byte temperature = 0;
byte humidity = 0;

void setup() {
    Serial.begin(9600);

    //pin sensor DHT11
    pinMode(pinDHT, INPUT);

    KonfigurasiLCD();
    KoneksiAP();
}

void loop() {
    KelembabanSuhu();
    Serial.print("connecting to ");
    Serial.print(host);
    Serial.print(':');
    Serial.println(port);
```

NodeMCU Sebagai Client Socket

```
// Use WiFiClient class to create TCP
connections
WiFiClient client;
if (!client.connect(host, port)) {
    Serial.println("connection failed");
    delay(5000);
    return;
}

// kirim data temperature dan kelembaban ke
server
Serial.println("sending data to server");
if (client.connected()) {
    client.println(String((int)temperature) +
"#" + String((int)humidity));
}

// Close the connection
// sifatnya opsional untuk pengiriman sekali
//Serial.println();
//Serial.println("closing connection");
//client.stop();
}

void KelembabanSuhu() {
    int err = SimpleDHTErrSuccess;
    if ((err = dht11.read(&temperature,
&humidity, NULL)) != SimpleDHTErrSuccess)
    {
        Serial.print("Pembacaan DHT11 gagal,
err=");
        Serial.println(err); delay(1000);
        return;
    }

    Serial.print("Sample OK: ");
    Serial.print((int)temperature);
    Serial.print(" *C, ");
    Serial.print((int)humidity);
    Serial.println(" H");
```

NodeMCU Sebagai Client Socket

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temperatur: " +
String((int)temperature));
lcd.write(0);
lcd.print("C ");
lcd.setCursor(0, 1);
lcd.print("Kelembaban: " +
String((int)humidity) + "H");

//baca dan kirim setiap 1,5 detik
delay(1500);
}

void KoneksiAP() {
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);

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

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void KonfigurasiLCD() {
lcd.begin(); // Inisialisasi LCD
lcd.createChar(0, derajat);
lcd.backlight(); // Menghidupkan backlight
lcd.setCursor(0, 0);
lcd.print("Project");
lcd.setCursor(0, 1);
```

NodeMCU Sebagai Client Socket

```
lcd.print("Socket TCP");
delay(3000);
lcd.clear();
}
```

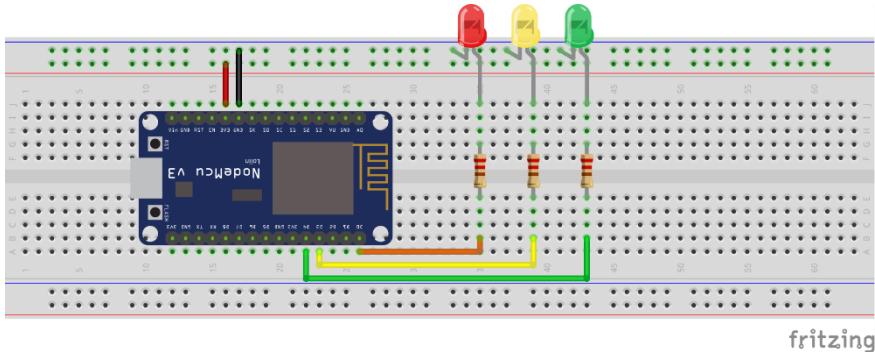
4.4 NodeMCU & Aplikasi Desktop Java

Pada bagian ini kita akan mengkomunikasikan antara aplikasi desktop Java Swing dengan NodeMCU. Jika pada aplikasi C# sebelumnya kita membuat dua sisi socket yaitu client-socket dan server-socket, pada Java ini kita hanya membuat socket client.

Tugas utama program yang akan dibuat adalah menghidupkan dan mematikan 3 LED yang berada di NodeMCU dari aplikasi desktop Java Swing secara jarak jauh. Kontrol LED pada Java Swing menggunakan komponen radio button.

4.4.1 Rancangan NodeMCU Sebagai Socket Server

Rancangan dan pengkabelan pada NodeMCU sebagai socket server adalah sebagai berikut



Kode Program

```
///////////
// Kode program belum terverifikasi //
// Created by Dodit Suprianto      //
// Email doditsuprianto@gmail.com   //
///////////
```

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

Kode Program

```
#define pinLedMerah 16      // D0 pin led merah
#define pinLedKuning 0       // D3 pin led kuning
#define pinLedHijau 2        // D4 pin led hijau

int port = 16375; // nomor port
WiFiServer server(port);

//Koneksi ke network Wifi AccessPoint
//Sesuaikan SSID Wifi masing-masing
const char *ssid = "Fanny 2004";

//Sesuaikan password WiFi masing-masing
const char *password = "fanny_200504";

void setup()
{
    Serial.begin(9600);

    //pin LED
    pinMode(pinLedMerah, OUTPUT);
    pinMode(pinLedKuning, OUTPUT);
    pinMode(pinLedHijau, OUTPUT);

    KoneksiAP(); // inisialisasi koneksi ke AP
}

void loop()
{
    WiFiClient client = server.available();
    String pesan = "";
    if (client) {
        if (client.connected())
        {
            Serial.println("Client Connected");
        }

        while (client.connected()) {
            while (client.available() > 0) {
                //Serial.write(client.read());
                //membaca data dari client yang
terhubung
            }
        }
    }
}
```

Kode Program

```
delay(10);
char c = client.read();
pesan += String(c);
}

Serial.println(pesan);

if (pesan == "MerahHidup") {
    digitalWrite(pinLedMerah, HIGH);
} else if (pesan == "MerahMati") {
    digitalWrite(pinLedMerah, LOW);
} else if (pesan == "KuningHidup") {
    digitalWrite(pinLedKuning, HIGH);
} else if (pesan == "KuningMati") {
    digitalWrite(pinLedKuning, LOW);
} else if (pesan == "HijauHidup") {
    digitalWrite(pinLedHijau, HIGH);
} else if (pesan == "HijauMati") {
    digitalWrite(pinLedHijau, LOW);
}
}

client.stop(); //Putuskan koneksi dengan
client
Serial.println("Client disconnected");
}
}

void KoneksiAP() {
// Konfigurasi WIFI
// Koneksikan NodeMCU ke AP
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);

// Menunggu koneksi dengan AP
Serial.println("Connecting to Wifi");
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    delay(500);
}
```

Kode Program

```
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);

Serial.print("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
Serial.print("Open Telnet and connect to
IP:");
Serial.print(WiFi.localIP());
Serial.print(" on port ");
Serial.println(port);
}
```

4.4.2 Antarmuka Aplikasi Desktop Java Swing

Untuk menuliskan kode program Java kita gunakan aplikasi IDE intellij IDEA versi community yang bisa di-download di <https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code=IIC>.

Pastikan pula bahwa laptop terinstall dengan JDK (*Java Development Kit Standard Edition*) yang bisa di-download di <https://www.oracle.com/java/technologies/javase-jdk15-downloads.html>.

www.oracle.com/java/technologies/javase-jdk15-downloads.html

Linux ARM 64 Compressed Archive	157.01 MB	jdk-15.0.1_linux-arm64_bin.tar.gz
Linux x64 Debian Package	154.79 MB	jdk-15.0.1_linux-x64_bin.deb
Linux x64 RPM Package	162.02 MB	jdk-15.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive	179.33 MB	jdk-15.0.1_linux-x64_bin.tar.gz
macOS Installer	175.94 MB	jdk-15.0.1_osx-x64_bin.dmg
macOS Compressed Archive	176.53 MB	jdk-15.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	159.69 MB	jdk-15.0.1_windows-x64_bin.exe
Windows x64 Compressed Archive	179.27 MB	jdk-15.0.1_windows-x64_bin.zip

Download JDK Standard Edition

www.jetbrains.com/idea/download/#section=windows

Download IntelliJ IDEA

Windows Mac Linux

Ultimate
For web and enterprise development

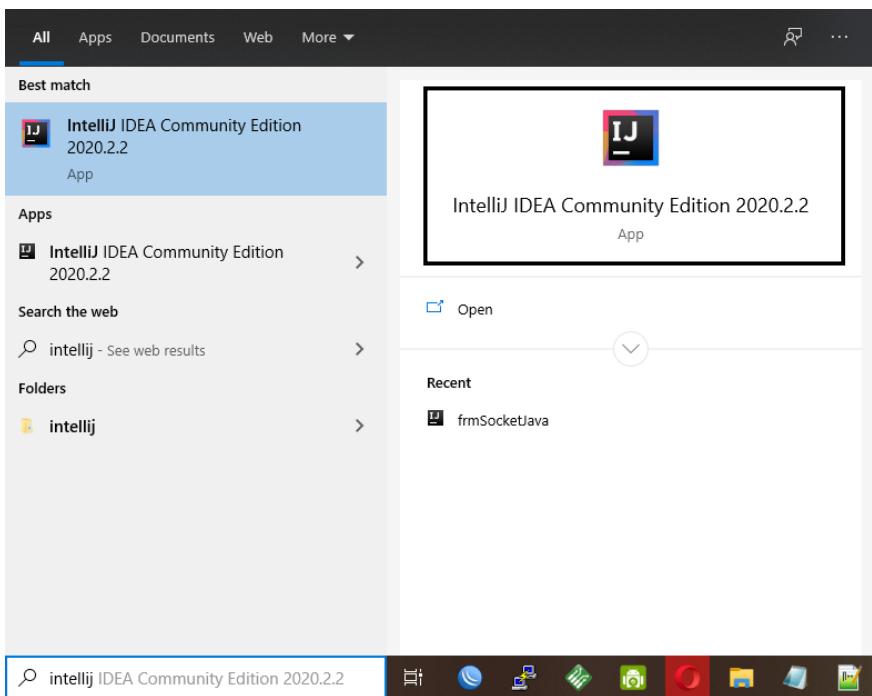
[Download .exe](#)

Community
For JVM and Android development

[Download .exe](#)

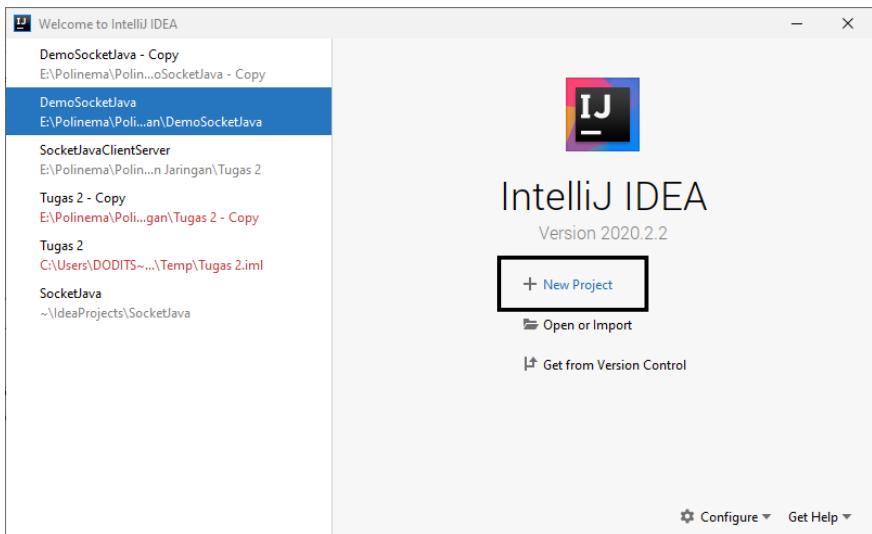
Free 30-day trial Free, open-source

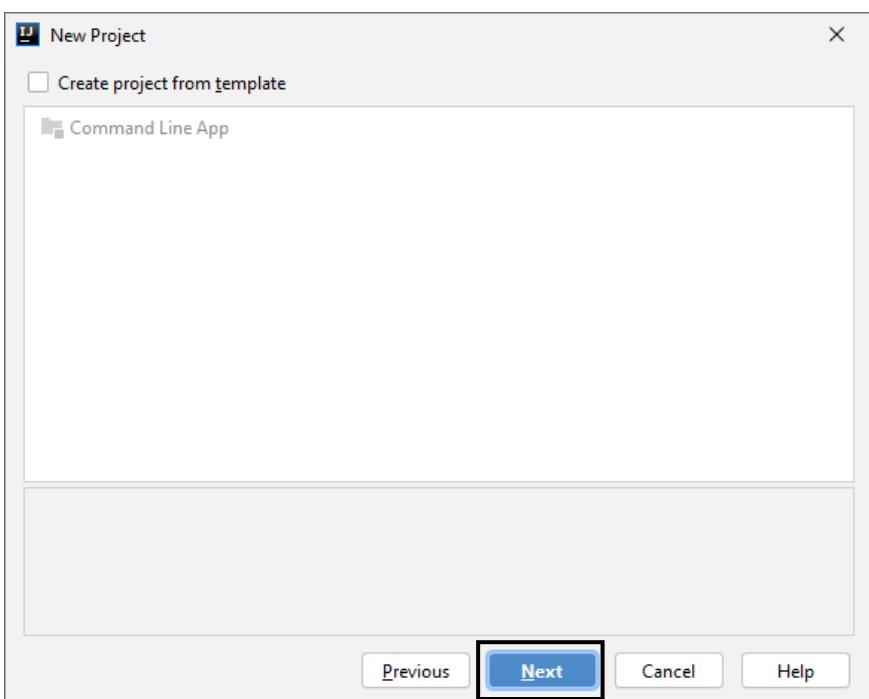
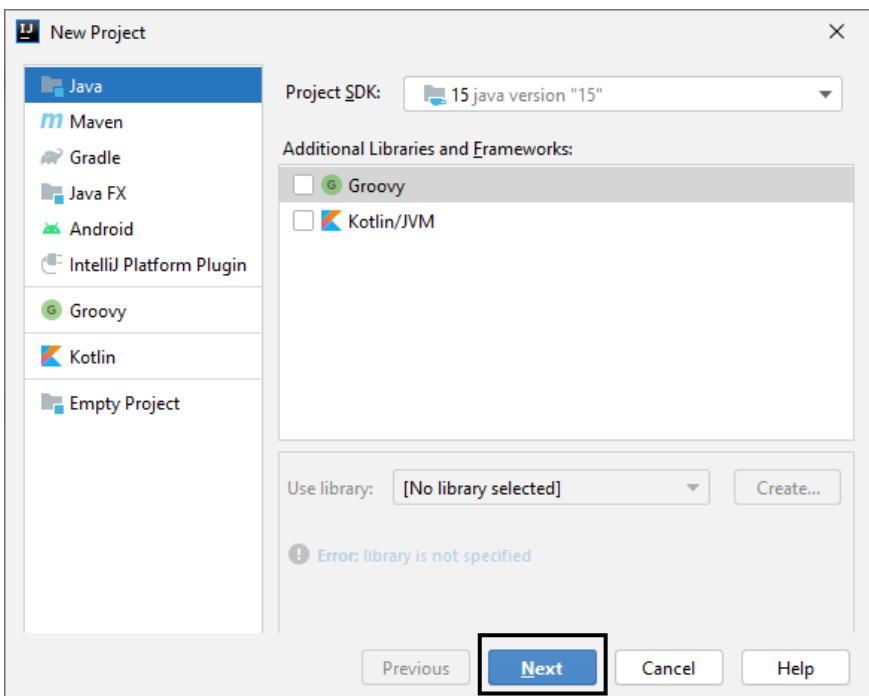
Download IDE IntelliJ IDEA

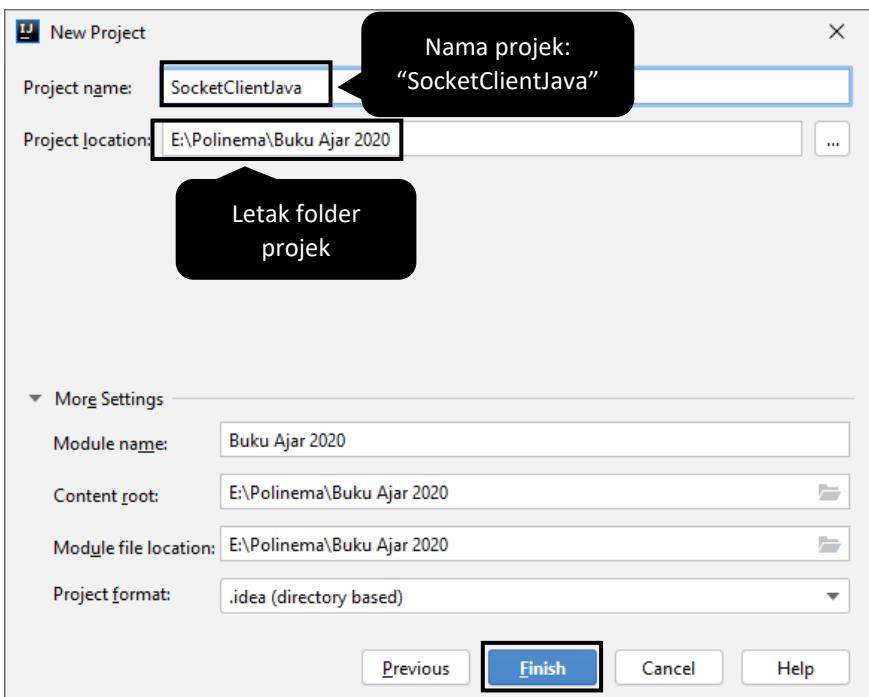


Aplikasi Intellij IDEA

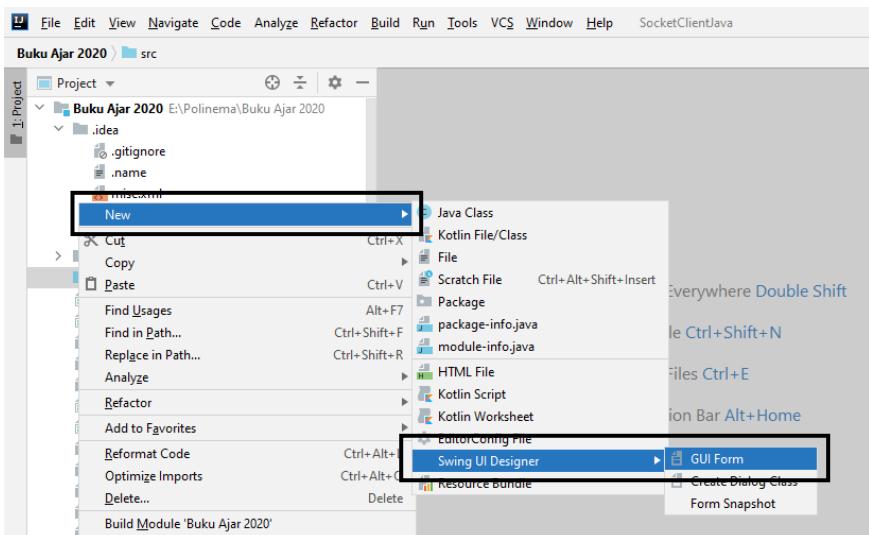
Buatlah projek bernama “**SocketClientJava**”. Jalankan aplikasi Intellij, kemudian klik menu File → New → New Project.



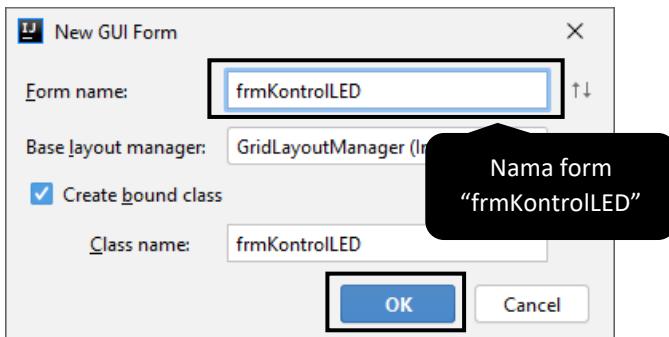




Buat form swing dari menu File → New → Swing UI Designer → GUI Form.

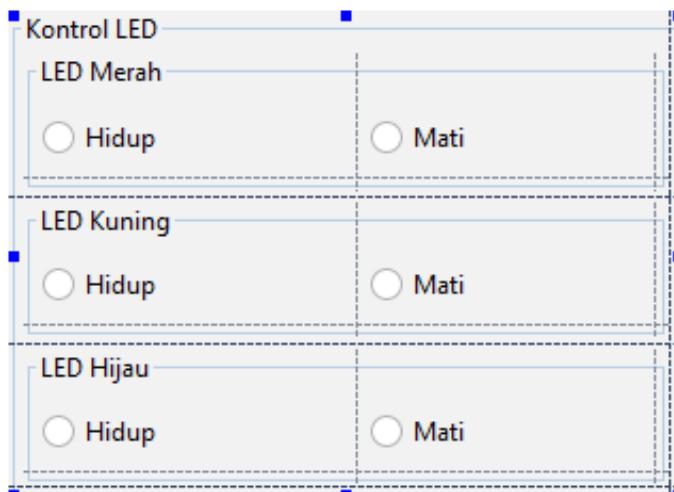


Beri nama form dengan “frmKontrolLED”

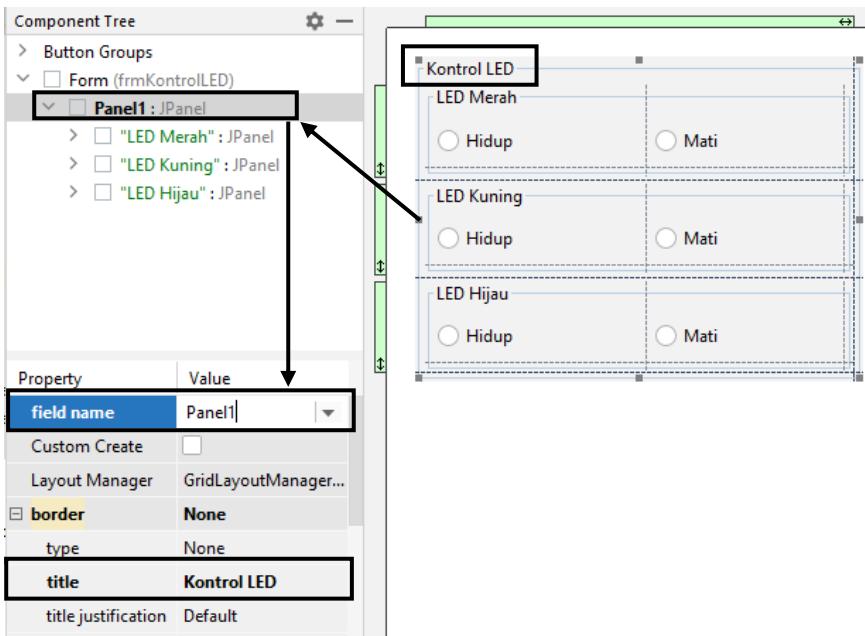


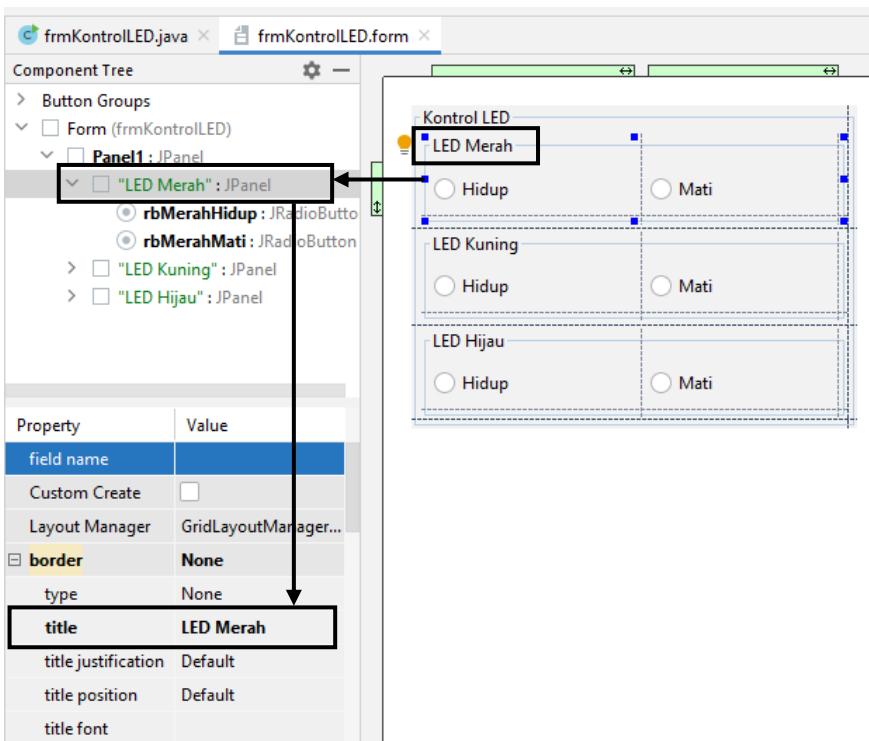
Secara garis besar desain formnya tampak seperti di bawah ini:

- Form terdiri dari satu panel induk dengan field name “Panel1” dan title “Kontrol LED”,
- Di dalam Panel1 terdapat tiga panel yang disusun relative top bottom. Antara lain panel title “LED Merah”, panel title “LED Kuning” dan panel title “LED Hijau”. Gunakan drag drop untuk menyusun ketiga panel agar rapi tersusun secara vertikal.
- Dalam ketiga panel (panel “LED Merah”, panel “LED Kuning” dan panel “LED Hijau”) terdapat dua radio button dengan title “Hidup” dan “Mati”.



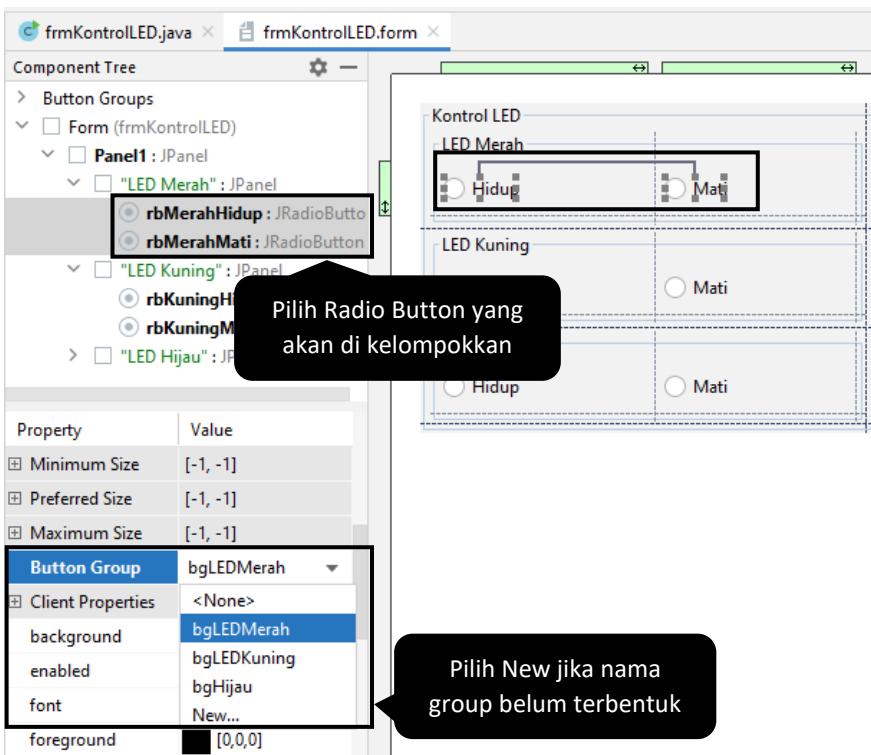
Agar lebih jelas bagaimana menambahkan komponen-komponen yang dibutuhkan, ikut langkah-langkah berikut ini



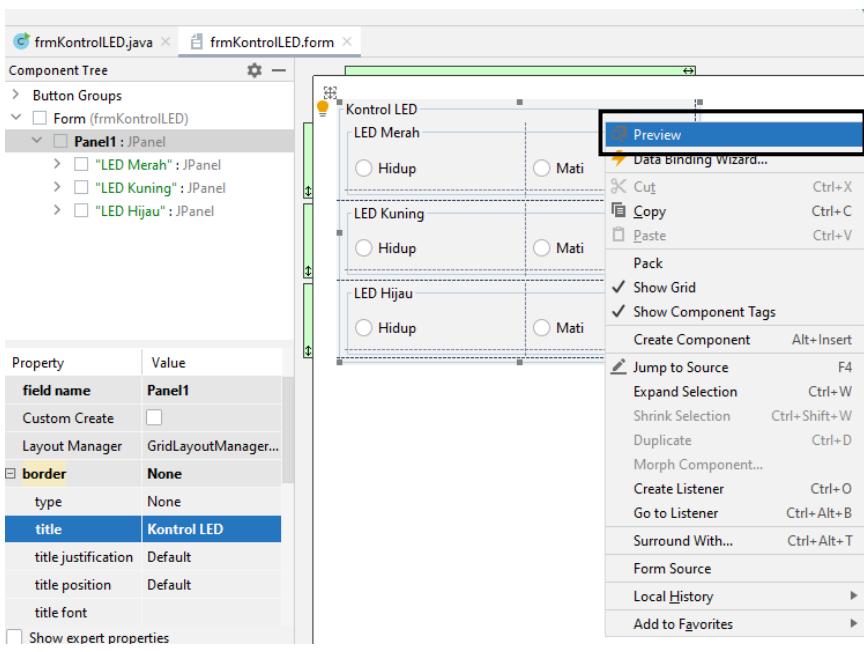


Pada kasus ini, setiap pasang radio button hidup dan radio button mati dikelompokkan dalam satu jenis panel LED (misalnya panel LED Merah saja) dan tidak boleh bercampur status radio button yang berada di panel LED lainnya (panel LED Kuning dan panel LED Hijau).

Oleh karena itu radio button yang berada dalam satu panel harus dikelompokkan terlebih dahulu menggunakan Button Group. Jika nama button group belum tersedia maka Anda dapat membuat nama button group baru, misalnya "bgLEDMerah", "bgLED" dan seterusnya.



Setelah membuat antarmuka form, selanjutnya klik kanan pada form dan pilih menu **Preview** untuk melihat hasil antarmuka sementara tampak di bawah ini.



4.4.3 Program Kontrol Socket Client Java Swing

Setelah desain antarmuka sistem kontrol LED terbentuk, langkah selanjutnya adalah memprogramnya. Di dalam program terdapat dua pokok bahasan, yaitu kode program untuk helper socket dan kode program untuk event radio button saat diklik.

Berikut kode helper socket server dan client socket, meskipun yang akan kita gunakan saat ini hanya client socket. Anda dapat menggunakan fungsi-fungsi pada helper socket server untuk kebutuhan lain.

Sedangkan fungsi/method java yang akan dijalankan pertama kali oleh sistem terdapat pada method void main.

Fungsi Helper Socket Java

```
import javax.swing.*;
import java.awt.*;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import
java.nio.channels.AsynchronousServerSocketChann
el;
import
java.nio.channels.AsynchronousSocketChannel;
import java.nio.channels.CompletionHandler;
import
java.util.concurrent.atomic.AtomicInteger;

public class frmKontrolLED {
    private JRadioButton rbMerahHidup;
    private JRadioButton rbMerahMati;
    private JRadioButton rbKuningHidup;
    private JRadioButton rbKuningMati;
    private JRadioButton rbHijauHidup;
    private JRadioButton rbHijauMati;
    private JPanel Panell;

    //main merupakan method yang akan
    dijalankan pertamakali
    //public static void main(String[] args)
    throws UnknownHostException, IOException {
        public static void main(String[] args) {
            // Membentuk instance JFrame dan
            mengatur property JFrame Form
            // JFrame merupakan wakil dari form,
            sebagai pondasi GUI di atasnya
            JFrame gui = new JFrame("Socket Client
Java Swing");
            gui.setContentPane(new
frmKontrolLED().Panell);
            gui.setPreferredSize(new Dimension(400,
300));
        }
    }
}
```

Fungsi Helper Socket Java

```
gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gui.pack();
        gui.setVisible(true);
    }

///////////////////////////////
// Socket Client Helper //
/////////////////////////////
public void EchoClient( String host, int
port, final String message, final AtomicInteger
messageWritten, final AtomicInteger
messageRead ) throws IOException {
    //create a socket channel
    AsynchronousSocketChannel sockChannel =
AsynchronousSocketChannel.open();

    //try to connect to the server side
    sockChannel.connect( new
InetSocketAddress(host, port), sockChannel, new
CompletionHandler<Void,
AsynchronousSocketChannel >() {
        @Override
        public void completed(Void result,
AsynchronousSocketChannel channel ) {
            //start to read message

startRead( channel,messageRead );

            //write an message to server
side
            startWrite( channel, message,
messageWritten );
        }

        @Override
        public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
            System.out.println( "fail to
connect to server");
        }
    }
}
```

Fungsi Helper Socket Java

```
        }

    });

    private void startRead( final
AsynchronousSocketChannel sockChannel, final
AtomicInteger messageRead ) {
    final ByteBuffer buf =
ByteBuffer.allocate(2048);

    sockChannel.read( buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel>() {

        @Override
        public void completed(Integer
result, AsynchronousSocketChannel channel) {
            //message is read from server
            messageRead.getAndIncrement();

            //print the message
            System.out.println( "Read
message:" + new String( buf.array()) );
        }

        @Override
        public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
            System.out.println( "fail to
read message from server");
        }
    });
}

private void startWrite( final
AsynchronousSocketChannel sockChannel, final
String message, final AtomicInteger
messageWritten ) {
    ByteBuffer buf =
ByteBuffer.allocate(2048);
```

Fungsi Helper Socket Java

```
buf.put(message.getBytes());
buf.flip();
messageWritten.getAndIncrement();
sockChannel.write(buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel >() {
    @Override
    public void completed(Integer
result, AsynchronousSocketChannel channel ) {
        //after message written
        //NOTHING TO DO
    }

    @Override
    public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
        System.out.println( "Fail to
write the message to server");
    }
});
}

///////////////
///
// Socket Server
//
// Gunakan helper ini fungsi di bawah
//
// Jika java bertindak sebagai socket
server  //

///////////////
///
    public void EchoServer( String bindAddr,
int bindPort ) throws IOException {
        InetSocketAddress sockAddr = new
InetSocketAddress(bindAddr, bindPort);
```

Fungsi Helper Socket Java

```
//create a socket channel and bind to
local bind address
AsynchronousServerSocketChannel
serverSock =
AsynchronousServerSocketChannel.open().bind(soc
kAddr);

//start to accept the connection from
client
serverSock.accept(serverSock, new
CompletionHandler<AsynchronousSocketChannel,Asy
nchronousServerSocketChannel >() {

    @Override
    public void
completed(AsynchronousSocketChannel
sockChannel, AsynchronousServerSocketChannel
serverSock ) {
        //a connection is accepted,
        start to accept next connection
        serverSock.accept( serverSock,
this );
        //start to read message from
the client
        startRead( sockChannel );

    }

    @Override
    public void failed(Throwable exc,
AsynchronousServerSocketChannel serverSock) {
        System.out.println( "fail to
accept a connection");
    }
);
}

private void
startRead( AsynchronousSocketChannel
sockChannel ) {
```

Fungsi Helper Socket Java

```
final ByteBuffer buf =
ByteBuffer.allocate(2048);

//read message from client
sockChannel.read( buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel >() {
    /**
     * some message is read from
client, this callback will be called
     */
    @Override
    public void completed(Integer
result, AsynchronousSocketChannel channel ) {
        buf.flip();

        // echo the message
        startWrite( channel, buf );

        //start to read next message
again
        startRead( channel );

        // menampilkan string pesan ke
textfield txtPesanDiterimaServer

//txtPesanDiterimaServer.setText(new
String( buf.array()));
    }

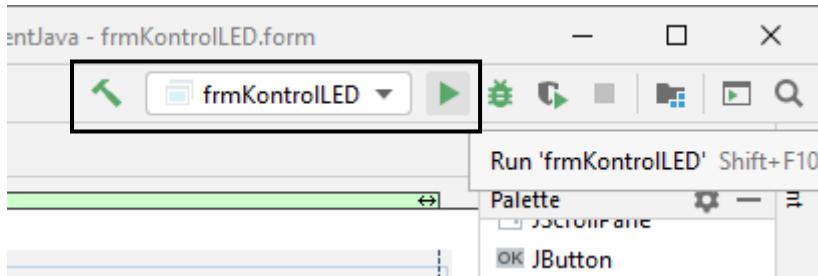
    @Override
    public void failed(Throwable exc,
AsynchronousSocketChannel channel ) {
        System.out.println( "fail to
read message from client");
    }
});
```

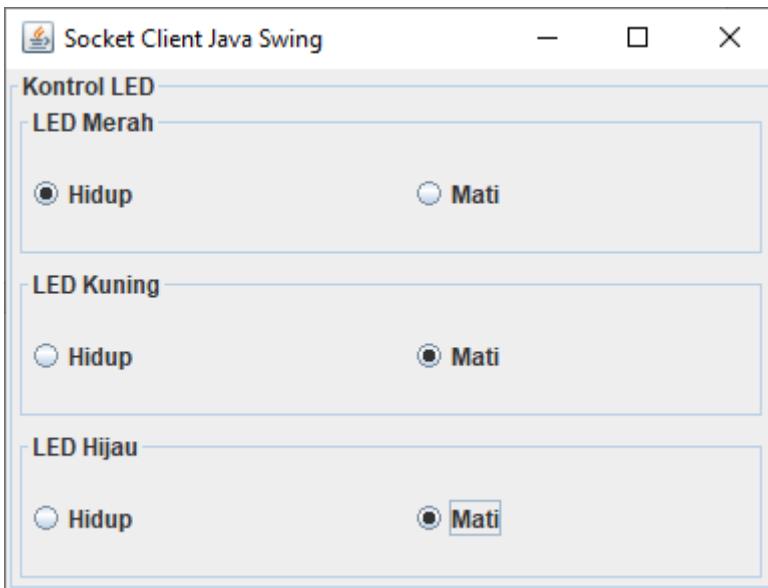
Fungsi Helper Socket Java

```
private void
startWrite( AsynchronousSocketChannel
sockChannel, final ByteBuffer buf) {
    sockChannel.write(buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel >() {
        @Override
        public void completed(Integer
result, AsynchronousSocketChannel channel) {
            //finish to write message to
client, nothing to do
        }

        @Override
        public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
            //fail to write message to
client
            System.out.println( "Fail to
write message to client");
        }
    });
}
```

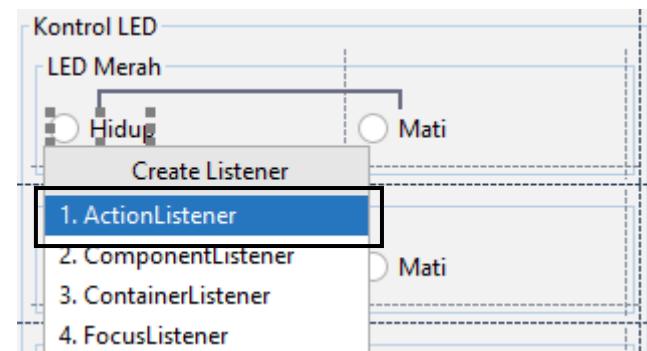
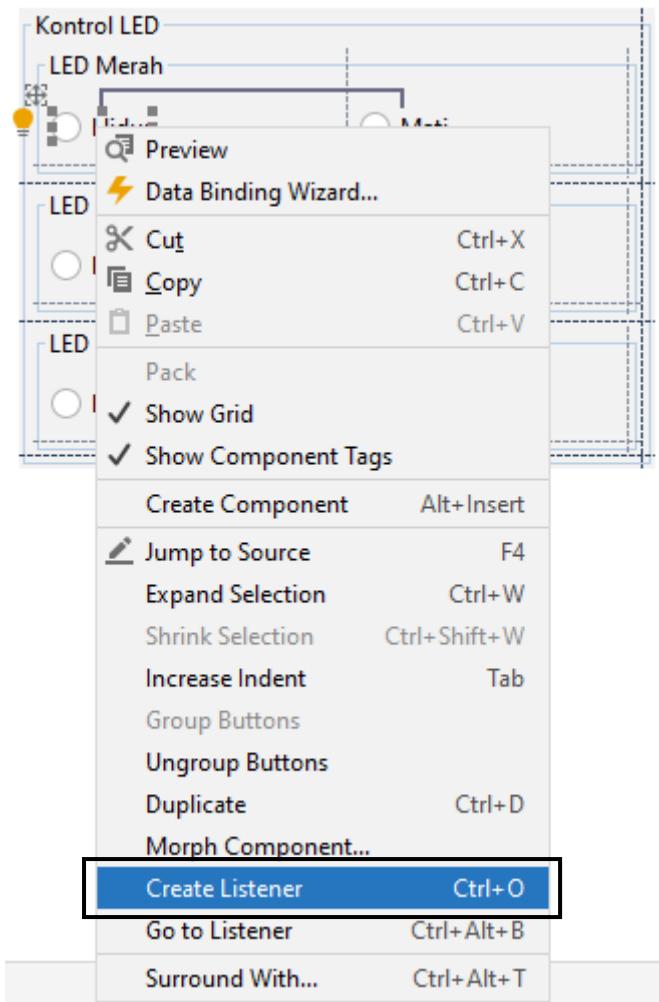
Sampai di sini jika aplikasi dijalankan akan tampak seperti berikut

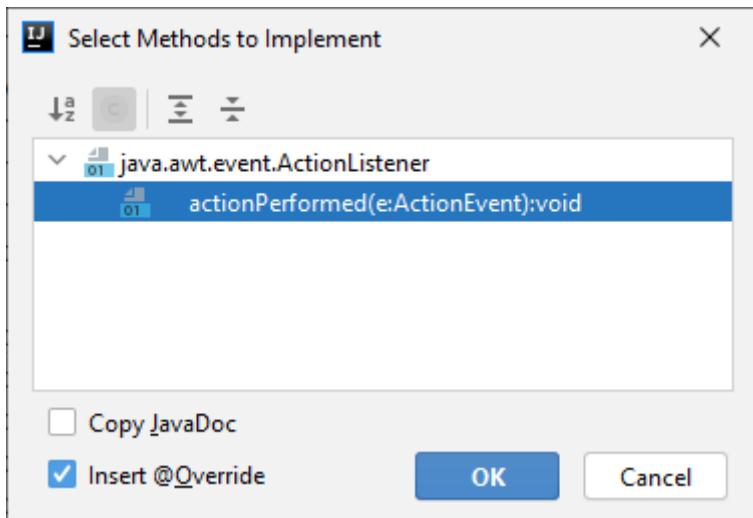




Tahap selanjutnya adalah membuat event pada setiap radio button. Ketika radio button di-klik akan menghidupkan dan mematikan LED di sisi NodeMCU secara jarak jauh.

Klik kanan pada salah satu radio button, kemudian pilih menu Create Listener → ActionListener.





Tambahkan addActionListener pada semua radio button dengan cara yang sama.

Berikut kode program action listener pada semua radio button

Method/function addActionListener radio button

```
rbMerahHidup.addActionListener(new
ActionListener() {
    @Override
    public void
actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null, "LED Merah
Hidup");
    try {
        AtomicInteger
messageWritten = new AtomicInteger(0);
        AtomicInteger messageRead =
new AtomicInteger(0);

        EchoClient(ipNodeMCU,
portNodeMCU, "MerahHidup", messageWritten,
messageRead);
    } catch (Exception ex) {
```

Method/function add ActionListener radio button

```
JOptionPane.showMessageDialog(null,
ex.getMessage());
}
}
})
);
rbMerahMati.addActionListener(new
ActionListener() {
    @Override
    public void
actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null, "LED Merah
Mati");
try {
    AtomicInteger
messageWritten = new AtomicInteger(0);
    AtomicInteger messageRead =
new AtomicInteger(0);

    EchoClient(ipNodeMCU,
portNodeMCU, "MerahMati", messageWritten,
messageRead);
} catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
}
}
});
rbKuningHidup.addActionListener(new
ActionListener() {
    @Override
    public void
actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null, "LED Kuning
Hidup");
try {
    AtomicInteger
messageWritten = new AtomicInteger(0);
```

Method/function addActionListener radio button

```
        AtomicInteger messageRead =
new AtomicInteger(0);

                EchoClient(ipNodeMCU,
portNodeMCU,      "KuningHidup",      messageWritten,
messageRead);
            } catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
}
}
});  
rbKuningMati.addActionListener(new
ActionListener() {
    @Override
    public void
actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(null, "LED Kuning
Mati");
try {
    AtomicInteger
messageWritten = new AtomicInteger(0);
    AtomicInteger messageRead =
new AtomicInteger(0);

    EchoClient(ipNodeMCU,
portNodeMCU,      "KuningMati",      messageWritten,
messageRead);
} catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
}
}
});
rbHijauHidup.addActionListener(new
ActionListener() {
    @Override
```

Method/function addActionListener radio button

```
        public void actionPerformed(ActionEvent e) {  
  
    JOptionPane.showMessageDialog(null, "LED Hijau Hidup");  
    try {  
        AtomicInteger messageWritten = new AtomicInteger(0);  
        AtomicInteger messageRead = new AtomicInteger(0);  
  
        EchoClient(ipNodeMCU,  
portNodeMCU, "HijauHidup", messageWritten,  
messageRead);  
    } catch (Exception ex) {  
  
    JOptionPane.showMessageDialog(null,  
ex.getMessage());  
    }  
}  
});  
rbHijauMati.addActionListener(new  
ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
  
    JOptionPane.showMessageDialog(null, "LED Hijau Mati");  
    try {  
        AtomicInteger messageWritten = new AtomicInteger(0);  
        AtomicInteger messageRead = new AtomicInteger(0);  
  
        EchoClient(ipNodeMCU,  
portNodeMCU, "HijauMati", messageWritten,  
messageRead);  
    } catch (Exception ex) {  
    }  
}});
```

Method/function addActionListener radio button

```
JOptionPane.showMessageDialog(null,  
ex.getMessage());  
}  
}  
});
```

Berikut adalah gabungan seluruh kode program:

Kode Program File frmKontrolLED.Java

```
//////////////////////////////  
// Kode program belum terverifikasi //  
// Created by Dodit Suprianto //  
// email: doditsuprianto@gmail.com //  
/////////////////////////////  
  
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.IOException;  
import java.net.InetSocketAddress;  
import java.nio.ByteBuffer;  
import  
java.nio.channels.AsynchronousServerSocketChann  
el;  
import  
java.nio.channels.AsynchronousSocketChannel;  
import java.nio.channels.CompletionHandler;  
import  
java.util.concurrent.atomic.AtomicInteger;  
  
public class frmKontrolLED {  
    private JRadioButton rbMerahHidup;  
    private JRadioButton rbMerahMati;  
    private JRadioButton rbKuningHidup;  
    private JRadioButton rbKuningMati;  
    private JRadioButton rbHijauHidup;  
    private JRadioButton rbHijauMati;  
    private JPanel Panel1;  
  
    // alamat IP tujuan silahkan disesuaikan
```

```

    // misalnya IP NodeMCU sebagai socket
    server adalah 192.168.1.6
    private final String ipNodeMCU =
"192.168.1.6";
    // nomor port NodeMCU
    private final int portNodeMCU = 16375;

    public frmKontrolLED() {
        rbMerahHidup.addActionListener(new
ActionListener() {
            @Override
            public void
actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null, "LED Merah
Hidup");
            try {
                AtomicInteger
messageWritten = new AtomicInteger(0);
                AtomicInteger messageRead =
new AtomicInteger(0);

                EchoClient(ipNodeMCU,
portNodeMCU, "MerahHidup", messageWritten,
messageRead);
            } catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
            }
        });
        rbMerahMati.addActionListener(new
ActionListener() {
            @Override
            public void
actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null, "LED Merah
Mati");
            try {

```

```

        AtomicInteger
messageWritten = new AtomicInteger(0);
        AtomicInteger messageRead =
new AtomicInteger(0);

        EchoClient(ipNodeMCU,
portNodeMCU, "MerahMati", messageWritten,
messageRead);
    } catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
}
}
});
rbKuningHidup.addActionListener(new
ActionListener() {
    @Override
    public void
actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(null, "LED Kuning
Hidup");
try {
    AtomicInteger
messageWritten = new AtomicInteger(0);
    AtomicInteger messageRead =
new AtomicInteger(0);

    EchoClient(ipNodeMCU,
portNodeMCU, "KuningHidup", messageWritten,
messageRead);
} catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
}
}
});
rbKuningMati.addActionListener(new
ActionListener() {
    @Override

```

```

        public void
actionPerformed(ActionEvent e) {

    JOptionPane.showMessageDialog(null, "LED Kuning
Mati");
        try {
            AtomicInteger
messageWritten = new AtomicInteger(0);
            AtomicInteger messageRead =
new AtomicInteger(0);

            EchoClient(ipNodeMCU,
portNodeMCU, "KuningMati", messageWritten,
messageRead);
        } catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
        }
    });
    rbHijauHidup.addActionListener(new
ActionListener() {
        @Override
        public void
actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(null, "LED Hijau
Hidup");
        try {
            AtomicInteger
messageWritten = new AtomicInteger(0);
            AtomicInteger messageRead =
new AtomicInteger(0);

            EchoClient(ipNodeMCU,
portNodeMCU, "HijauHidup", messageWritten,
messageRead);
        } catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
    }
}

```

```

        }
    });
    rbHijauMati.addActionListener(new
ActionListener() {
    @Override
    public void
actionPerformed(ActionEvent e) {
    JOptionPane.showMessageDialog(null, "LED Hijau
Mati");
    try {
        AtomicInteger
messageWritten = new AtomicInteger(0);
        AtomicInteger messageRead =
new AtomicInteger(0);

        EchoClient(ipNodeMCU,
portNodeMCU, "HijauMati", messageWritten,
messageRead);
    } catch (Exception ex) {

JOptionPane.showMessageDialog(null,
ex.getMessage());
    }
}
));
}

//main merupakan method yang akan
dijalankan pertamakali
public static void main(String[] args) {
    // Membentuk instance JFrame dan
mengatur property JFrame Form
    // JFrame merupakan wakil dari form,
sebagai pondasi GUI di atasnya
    JFrame gui = new JFrame("Socket Client
Java Swing");
    gui.setContentPane(new
frmKontrolLED().Panell);
    gui.setPreferredSize(new Dimension(400,
300));
}

```

```

gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gui.pack();
        gui.setVisible(true);
    }

///////////////
// Socket Client Helper //
/////////////
    public void EchoClient( String host, int
port, final String message, final AtomicInteger
messageWritten, final AtomicInteger
messageRead ) throws IOException {
        //create a socket channel
        AsynchronousSocketChannel sockChannel =
AsynchronousSocketChannel.open();

        //try to connect to the server side
        sockChannel.connect( new
InetSocketAddress(host, port), sockChannel, new
CompletionHandler<Void,
AsynchronousSocketChannel >() {
            @Override
            public void completed(Void result,
AsynchronousSocketChannel channel ) {
                //start to read message

startRead( channel,messageRead );

                //write an message to server
                side
                    startWrite( channel, message,
messageWritten );
            }

            @Override
            public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
                System.out.println( "fail to
connect to server");
            }

```

```
        });
    }

    private void startRead( final
AsynchronousSocketChannel sockChannel, final
AtomicInteger messageRead ) {
    final ByteBuffer buf =
ByteBuffer.allocate(2048);

    sockChannel.read( buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel>() {

        @Override
        public void completed(Integer
result, AsynchronousSocketChannel channel) {
            //message is read from server
            messageRead.getAndIncrement();

            //print the message
            System.out.println( "Read
message:" + new String( buf.array() ) );
        }

        @Override
        public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
            System.out.println( "fail to
read message from server");
        }
    });
}

private void startWrite( final
AsynchronousSocketChannel sockChannel, final
String message, final AtomicInteger
messageWritten ) {
    ByteBuffer buf =
ByteBuffer.allocate(2048);
    buf.put(message.getBytes());
    buf.flip();
```

```

        messageWritten.getAndIncrement();
        sockChannel.write(buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel >() {
    @Override
    public void completed(Integer
result, AsynchronousSocketChannel channel ) {
        //after message written
        //NOTHING TO DO
    }

    @Override
    public void failed(Throwable exc,
AsynchronousSocketChannel channel) {
        System.out.println( "Fail to
write the message to server");
    }
});
}

```

```

///////////
///
// Socket Server
//
// Gunakan helper ini fungsi di bawah
//
// Jika java bertindak sebagai socket server
//
///////////
///
    public void EchoServer( String bindAddr,
int bindPort ) throws IOException {
        InetSocketAddress sockAddr = new
InetSocketAddress(bindAddr, bindPort);

        //create a socket channel and bind to
local bind address
        AsynchronousServerSocketChannel
serverSock =

```

```
AsynchronousServerSocketChannel.open().bind(soc  
kAddr);  
  
        //start to accept the connection from  
client  
        serverSock.accept(serverSock, new  
CompletionHandler<AsynchronousSocketChannel,Asy  
nchronousServerSocketChannel >() {  
  
            @Override  
            public void  
completed(AsynchronousSocketChannel  
sockChannel, AsynchronousServerSocketChannel  
serverSock ) {  
                //a connection is accepted,  
start to accept next connection  
                serverSock.accept( serverSock,  
this );  
                //start to read message from  
the client  
                startRead( sockChannel );  
  
            }  
  
            @Override  
            public void failed(Throwable exc,  
AsynchronousServerSocketChannel serverSock) {  
                System.out.println( "fail to  
accept a connection");  
            }  
        } );  
    }  
  
    private void  
startRead( AsynchronousSocketChannel  
sockChannel ) {  
        final ByteBuffer buf =  
ByteBuffer.allocate(2048);  
  
        //read message from client
```

```

        sockChannel.read( buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel >() {
    /**
     * some message is read from
client, this callback will be called
     */
    @Override
    public void completed(Integer
result, AsynchronousSocketChannel channel ) {
        buf.flip();

        // echo the message
        startWrite( channel, buf );

        //start to read next message
again
        startRead( channel );

        // menampilkan string pesan ke
textfield txtPesanDiterimaServer

//txtPesanDiterimaServer.setText(new
String( buf.array()));
    }

    @Override
    public void failed(Throwable exc,
AsynchronousSocketChannel channel ) {
        System.out.println( "fail to
read message from client");
    }
});

}

private void
startWrite( AsynchronousSocketChannel
sockChannel, final ByteBuffer buf) {
    sockChannel.write(buf, sockChannel, new
CompletionHandler<Integer,
AsynchronousSocketChannel >() {
    @Override

```

```
        public void completed(Integer result, AsynchronousSocketChannel channel) {
            //finish to write message to client, nothing to do
        }

        @Override
        public void failed(Throwable exc, AsynchronousSocketChannel channel) {
            //fail to write message to client
            System.out.println( "Fail to write message to client");
        }
    );
}

}
```

BAB 5

Edge Computing Blynk

Jika sebelumnya kita telah mengembangkan aplikasi “smart device / things” microcontroller NodeMCU yang berjalan secara lokal, juga mengkomunikasikan microcontroller NodeMCU dengan Aplikasi secara jarak jauh menggunakan socket maka pada bagian ini kita akan mengimplementasikan *Internet of Things* dengan *Edge Computing*.

Terdapat banyak macam aplikasi Edge Computing yang dapat dimanfaatkan, misalnya tenant.io, Thinger.io, Thingspeak.io dan lain-lain. Namun pada buku ini kita akan menggunakan blynk.io.

Blynk telah mendukung berbagai jenis protokol komunikasi, salah satunya MQTT. Konsep *publiser*, *subscriber* dan *broker message* dibungkus menjadi *service API namespace* yang disediakan oleh blynk siap digunakan fungsionalitasnya pada microcontroller (salah satunya MCU berbasis chip ESP8266, NodeMCU termasuk di dalamnya).

Terdapat dua sisi blynk yang harus diinstal, yaitu sisi server yang berfungsi sebagai *back end* dan sisi client yang diinstall pada perangkat mobile (dapat diunduh dari playstore). Blynk memberi kemudahan dalam mengatur widget-widget untuk memonitor dan mengontrol smart device. Sebagai referensi silahkan kunjungi dokumentasi blynk di <https://docs.blynk.cc>

Layanan IoT Server Blynk versi cloud disediakan secara gratis. Keuntungan menggunakan layanan Blynk Cloud, antara lain:

- Kita akan mudah mengaksesnya dimanapun dan kapanpun karena IoT Server Blynk Cloud bersifat publik.
- Kita tidak perlu membangun infrastruktur Blynk yang dibangun karena mereka sudah membuatnya dengan keamanan tinggi tentunya.
- Berbiaya rendah karena layanannya bersifat gratis. Dengan catatan dibatasi fiturnya.

Kerugian menggunakan layanan Blynk Cloud, antara lain:

- Memiliki keterbatasan jumlah widget yang bisa dipasang.
- Memiliki keterbatasan jumlah smart device yang bisa dihubungkan.
- Kurang memiliki kebebasan dalam mengkonfigurasi server blynk karena posisi kita sebagai client. Ini berarti hanya tersedia satu akun user, yaitu kita saja.
- Untuk memanfaatkan banyak smart device dan widget harus membeli “Energy”, yaitu semacam deposit voucher yang akan dipotong sesuai pemakaian layanan.

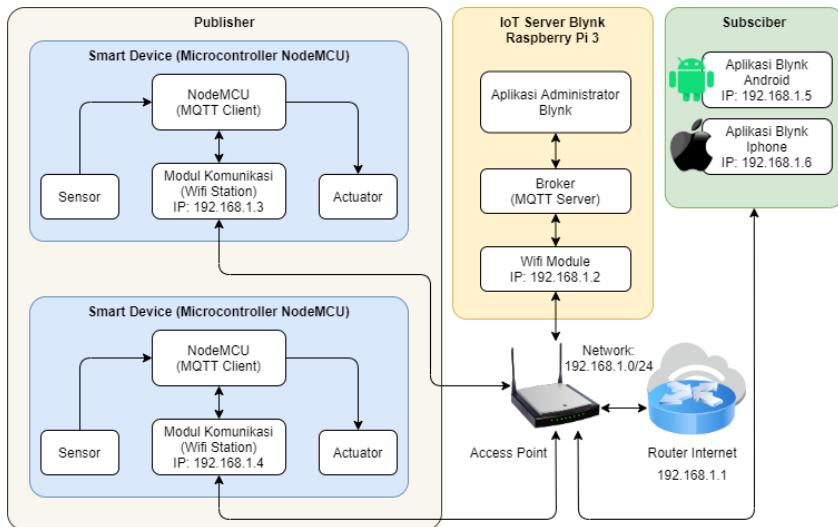
Untungnya Blynk berbaik hati karena menyediakan aplikasinya secara gratis, sehingga blynk dapat di-install pada infrastruktur yang kita miliki. Oleh karena itu dalam buku ini akan dipelajari bagaimana menginstall dan mengkonfigurasi blynk pada server lokal dengan Raspberry Pi 3.

Blynk mendukung banyak smart device, misalnya NodeMCU (ESP8266), Arduino, Raspberry Pi, ARM Mbed dan lain-lain. Silahkan kunjungi link <https://github.com/blynkkk/blynkkk.github.io/blob/master/SupportedHardware.md> untuk mengetahui lebih detil mengenai smart device yang didukung oleh blynk. Disarankan pula untuk membaca halaman dokumennya di <https://docs.blynk.cc> yang berisi cara instalasi dan konfigurasi blynk IoT Server.

5.1 Topologi IoT Server Jaringan Lokal

Arsitektur IoT Server jaringan lokal yang dibangun dapat dijelaskan sebagai berikut:

- *Publisher* berupa smart device dengan memanfaatkan microcontroller NodeMCU
- Smart Device melibatkan sensor dan actuator.
- Sebagai server IoT adalah Blynk, berisi broker MQTT (secara default tertanam dalam blynk) sekaligus aplikasi pengendali dan monitoring smart device yang terhubung dengannya.
- Smartphone Android/Iphone sebagai *Subscriber* yang meminta layanan *monitoring* dan *controlling* ke IoT Server Blynk.



Jika ditinjau dari sisi jaringan komputer maka NodeMCU, blynk, dan smartphone berposisi sebagai station yang terhubung dengan Access Point.

Diasumsikan alamat network-nya adalah 192.168.1.0/24. Paling ujung adalah router gateway menuju ke internet dengan IP 192.168.1.1. Dimana IP dapat dikonfigurasi secara statis atau DHCP.

5.2 Kebutuhan Hardware & Software IoT Server Blynk

Server IoT Blynk tergolong aplikasi ringan, sehingga blynk dapat diinstall pada Raspberry Pi 3 yang sumberdayanya rendah.

Adapun kebutuhan *hardware* dan *software* dalam projek ini antara lain:

- Komputer mini Raspberry Pi 3 Model B atau B+ (dikenal juga istilah SoC – system on chip)
- MicroSD minimal 8/16 GB, sebagai tempat penyimpanan sistem operasi linux dan aplikasi IoT Server Blynk. Sebaiknya gunakan MicroSD class 10 karena berpengaruh pada kecepatan proses.
- Adapter MicroSD ke USB atau MicroSD ke SD-Card, tergantung kebutuhan.

- Adaptor tegangan 5 Volt, arus minimal 2 Amper sebagai power supply Raspberry PI 3



Raspberry Pi 3 Model B



Adaptor 5 Volt



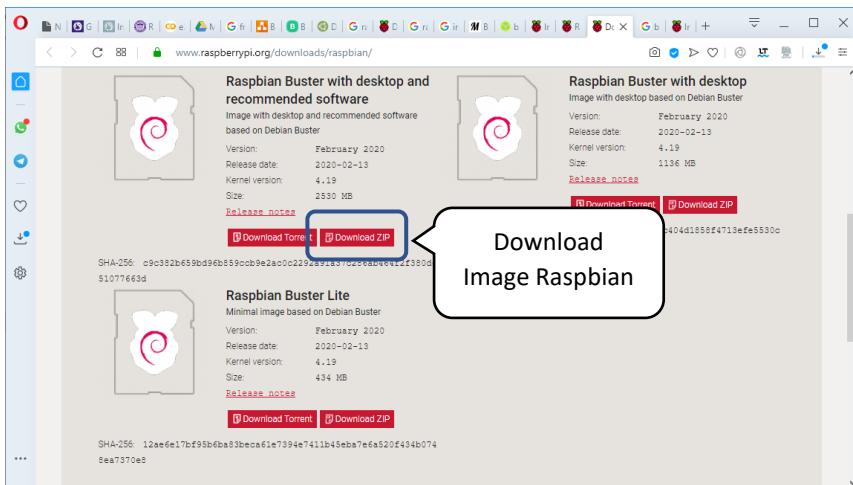
Keyboard



Adapter MicroSD

- File image Raspbian **2020-02-13-raspbian-buster-full.zip** yang akan diinstal pada Raspberry Pi 3 sebagai sistem operasi linux raspbian. Versi sistem operasi tidak harus sama. Anda dapat men-download-nya di
<https://www.raspberrypi.org/downloads/raspbian/>,
khususnya
https://downloads.raspberrypi.org/raspbian_full_latest.

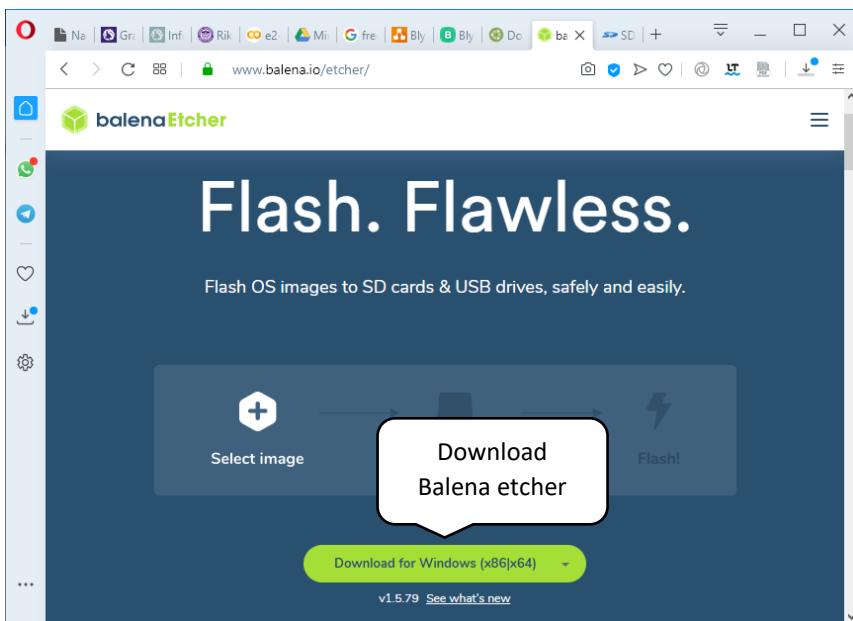
Anda harus mengekstrak terlebih dahulu file .ZIP tersebut menjadi file .IMG.



- **SDFormatter**, yaitu aplikasi untuk mem-format MicroSD sebelum dilakukan burning sistem operasi. File di-download di https://www.sdcard.org/downloads/formatter/eula_windows/.

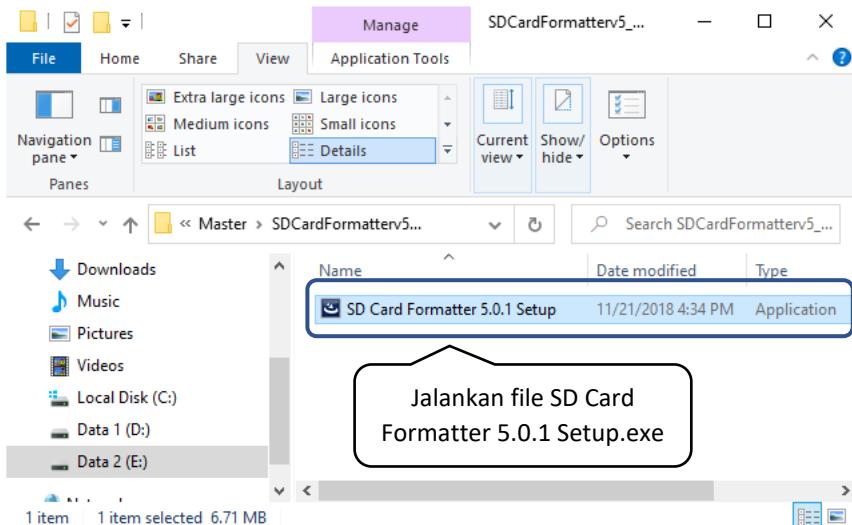


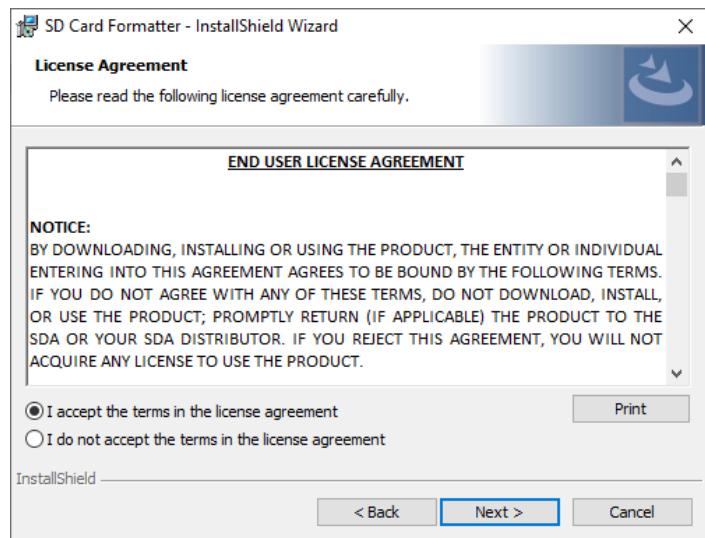
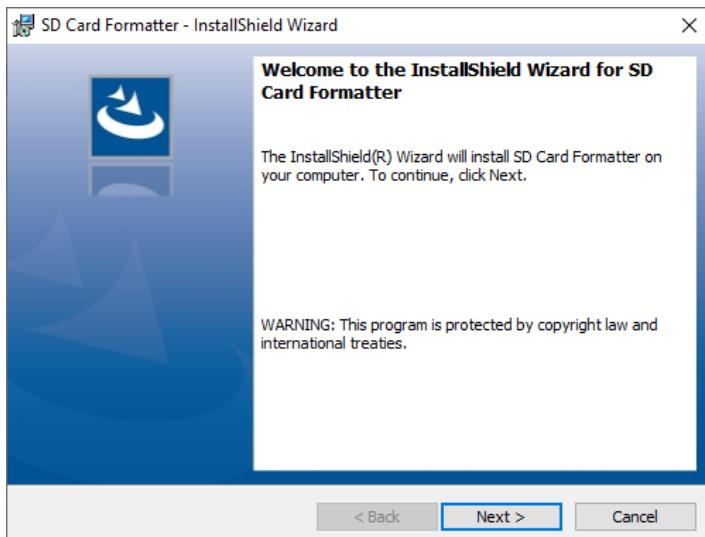
- **Balena Etcher**, yaitu aplikasi untuk mem-burning image Linux Raspbian ke dalam MicroSD. Etcher dapat di-download di <https://www.balena.io/etcher/>.

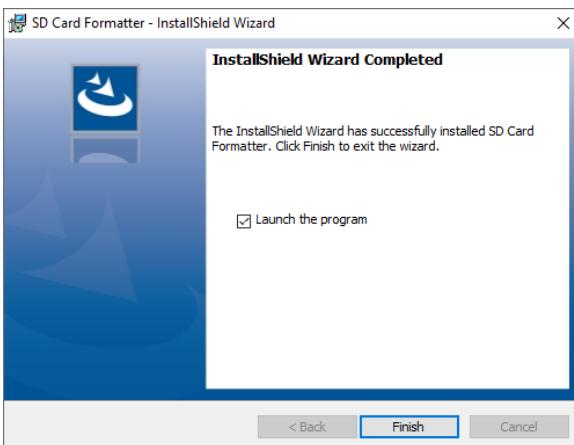
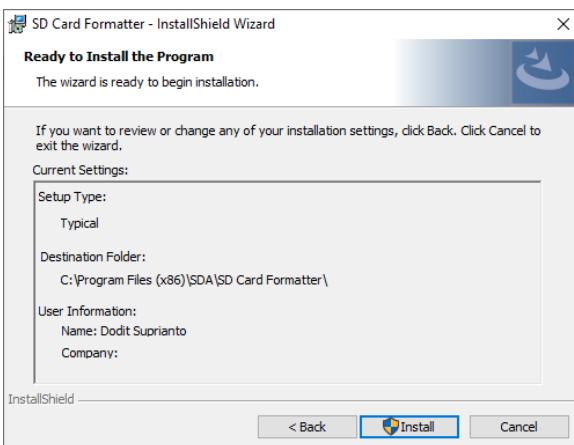
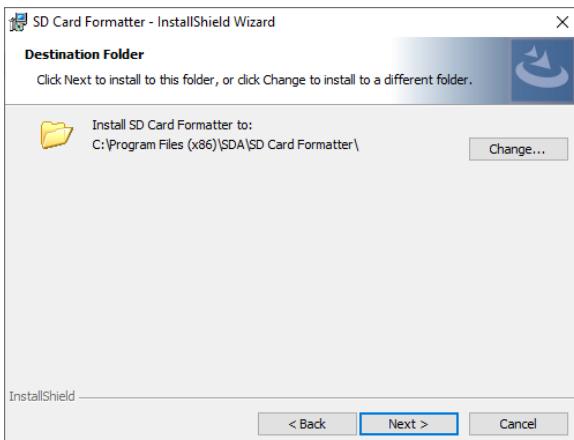


5.3 Software SDCard Formatter

Ekstrak file **SDCardFormatterv5_WinEN.zip** yang telah di-download. Jalankan file **SD Card Formatter 5.0.1 Setup.exe**. Kemudian ikuti tahapan instalasinya sampai selesai.

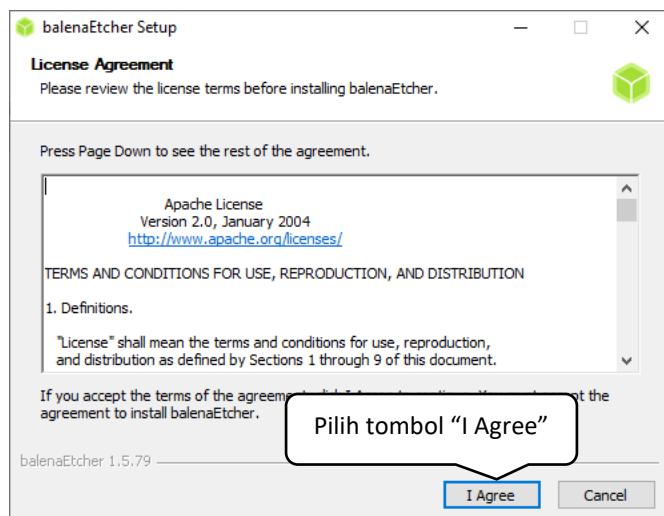
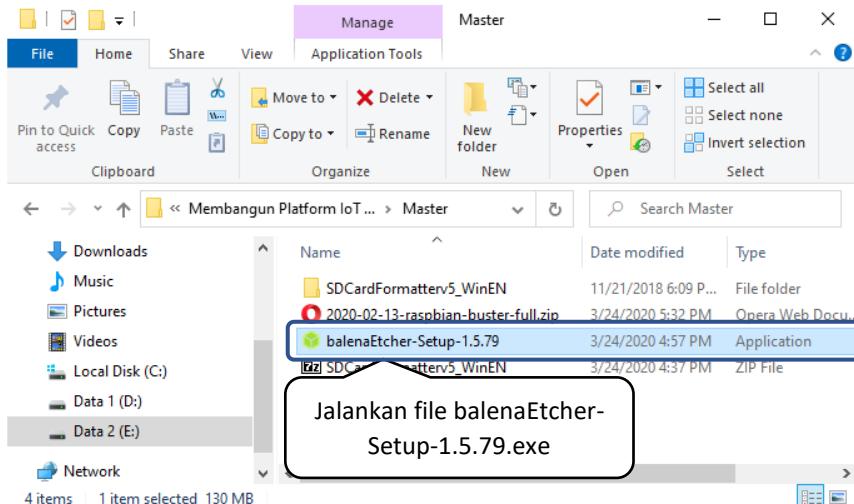


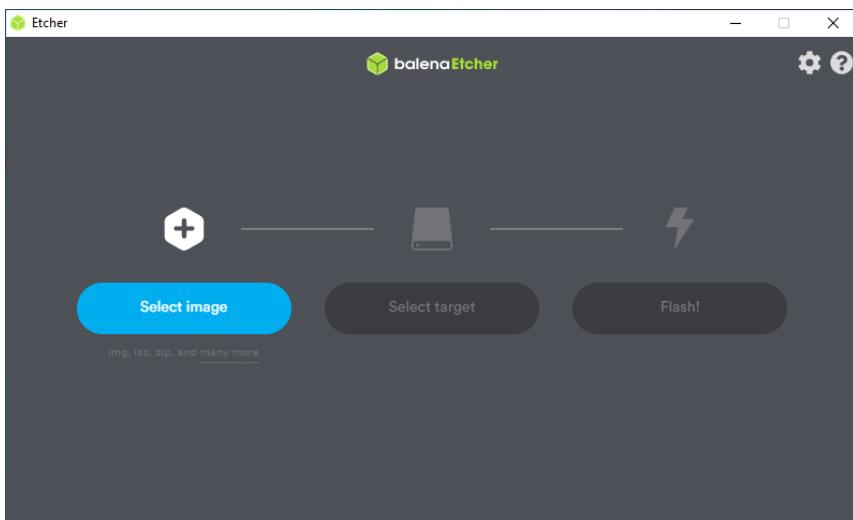




5.4 Software Balena Etcher

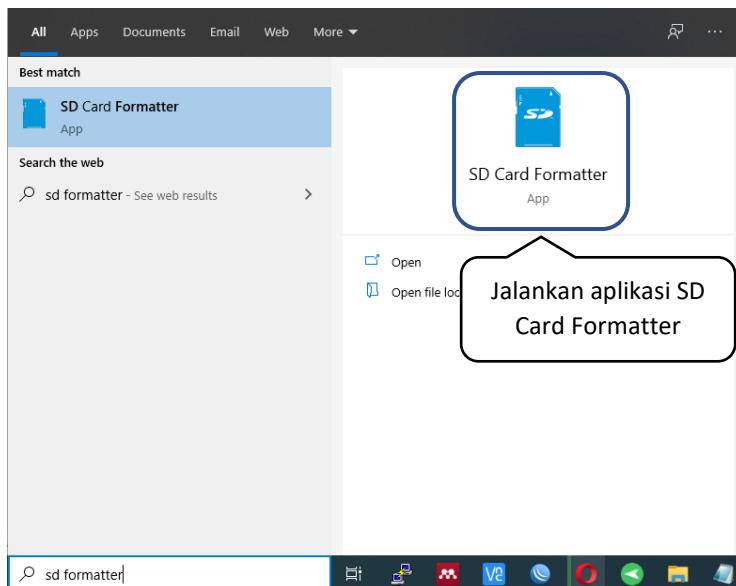
Jalankan file **balenaEtcher-Setup-1.5.79.exe** dari windows explorer. Nama file yang di-download mengikuti update versinya sehingga tidak harus sama dengan yang tertulis dalam tulisan buku.

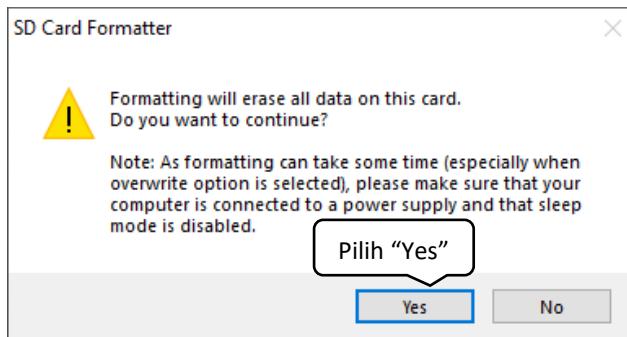
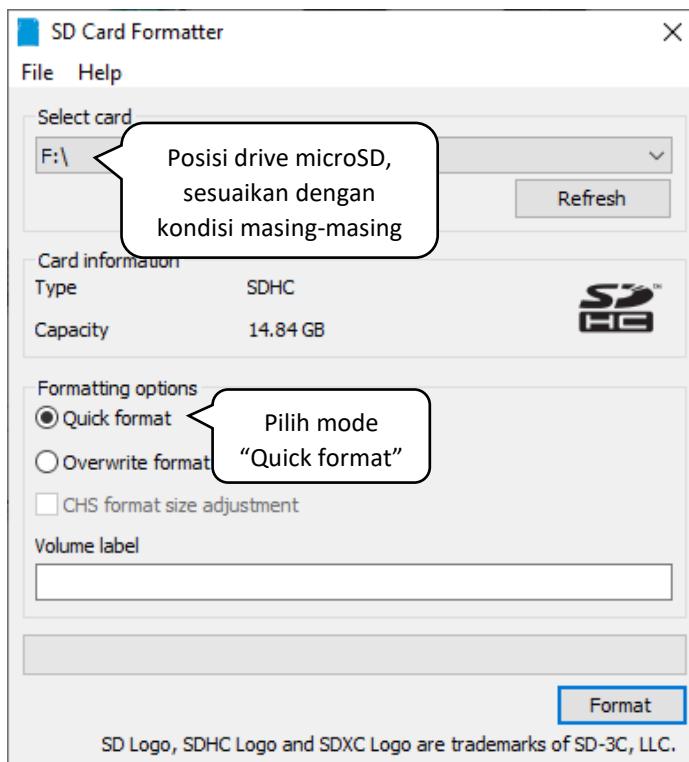


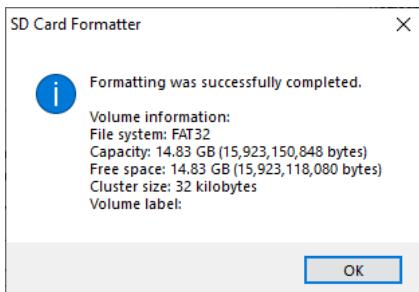


5.5 Format Memori microSD

Masukkan MicroSD + Adapter ke USB komputer atau SD Card Reader komputer, kemudian jalankan aplikasi **SD Card Formatter**. Sebelum memformat, pastikan lokasi drive MicroSD sudah benar agar tidak terjadi salah format. Pilih **Quick format** pada Formatting options.

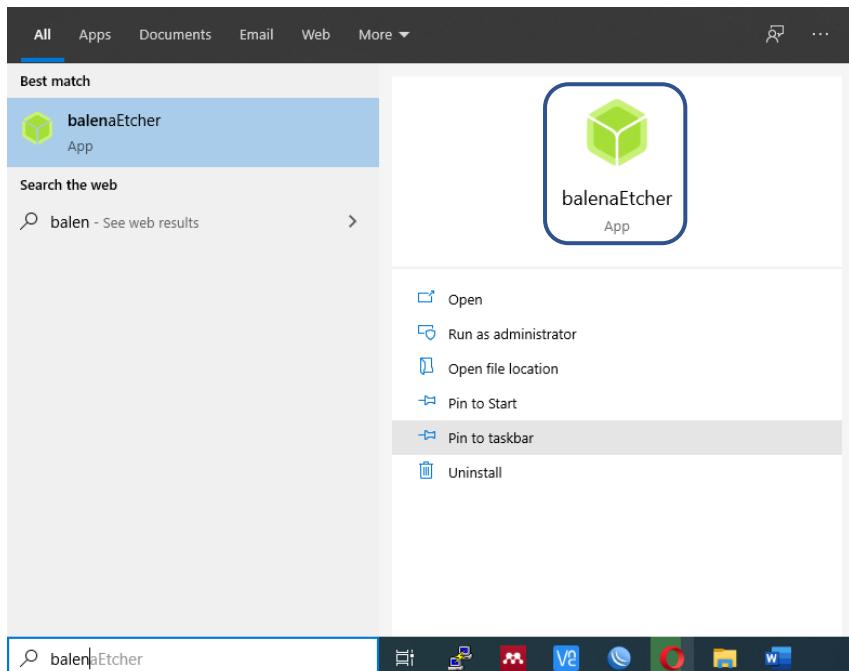


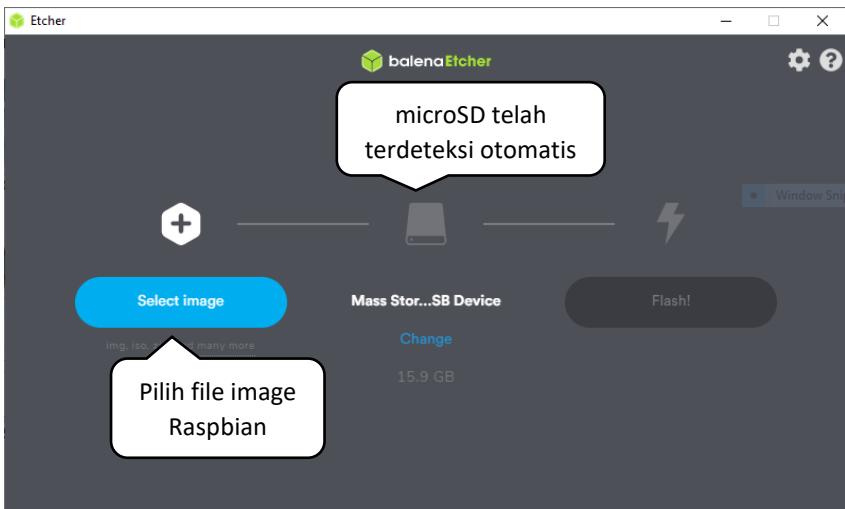




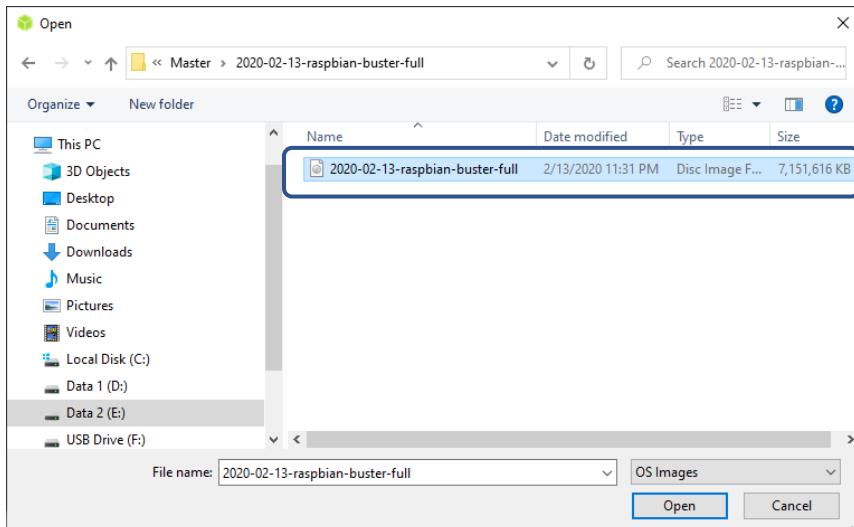
5.6 Burning Image Linux Raspbian Ke MicroSD

- Masukkan MicroSD + Adapter ke USB komputer atau SD Card Reader komputer.
- Jalankan aplikasi **balenaEtcher**. Di sini MicroSD akan terdeteksi secara otomatis oleh Balena Etcher tanpa perlu memilih tombol **Select Drive**.
- Pilih tombol Select Image, kemudian pilih file image **2020-02-13-raspbian-buster-full.img**.
- Pilih tombol Flash untuk mem-burning image Linux Raspbian ke dalam MicroSD.

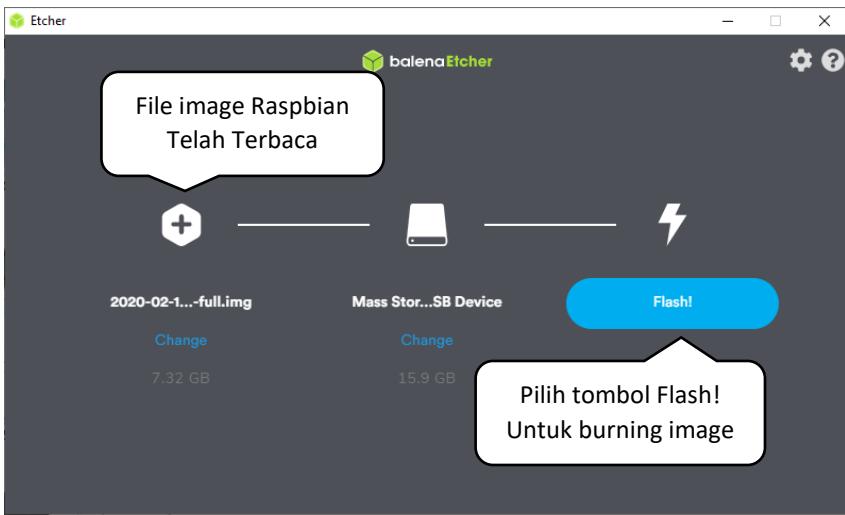




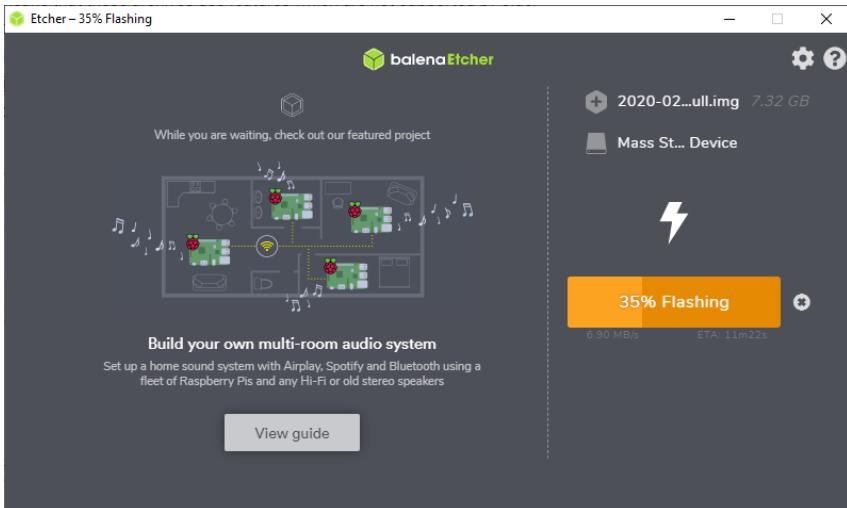
Pilih tombol **Select image**



File image Raspbian **2020-02-13-raspbian-buster-full.img**

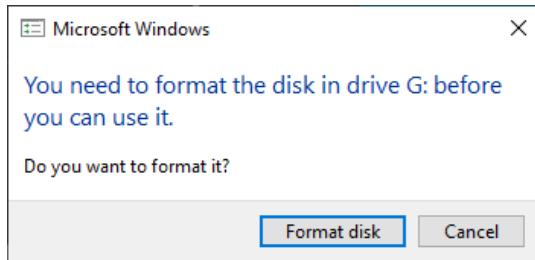


Pilih tombol **Flash** untuk burning image



Proses burning image

Jika proses burning Raspbian selesai, namun Anda diminta untuk mem-format drive microSD maka abaikan saja, jangan di-format!



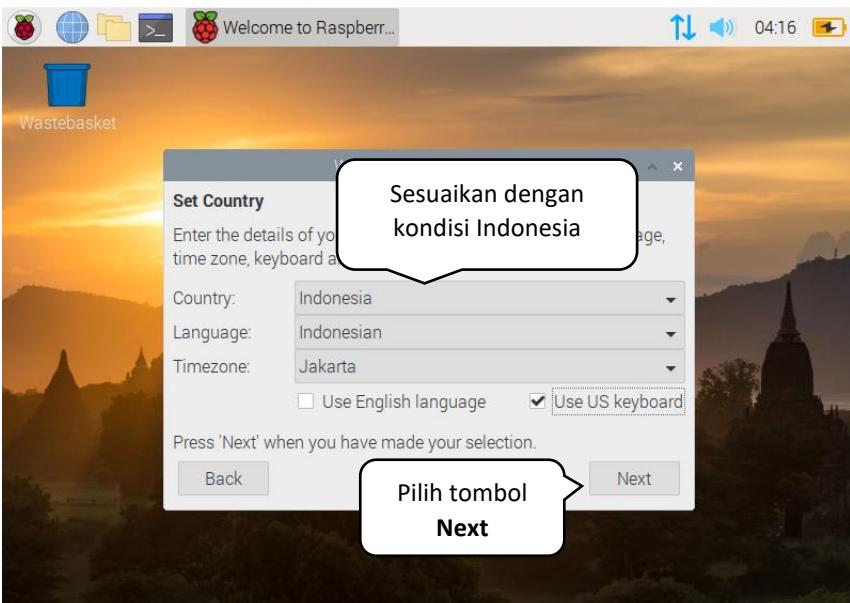
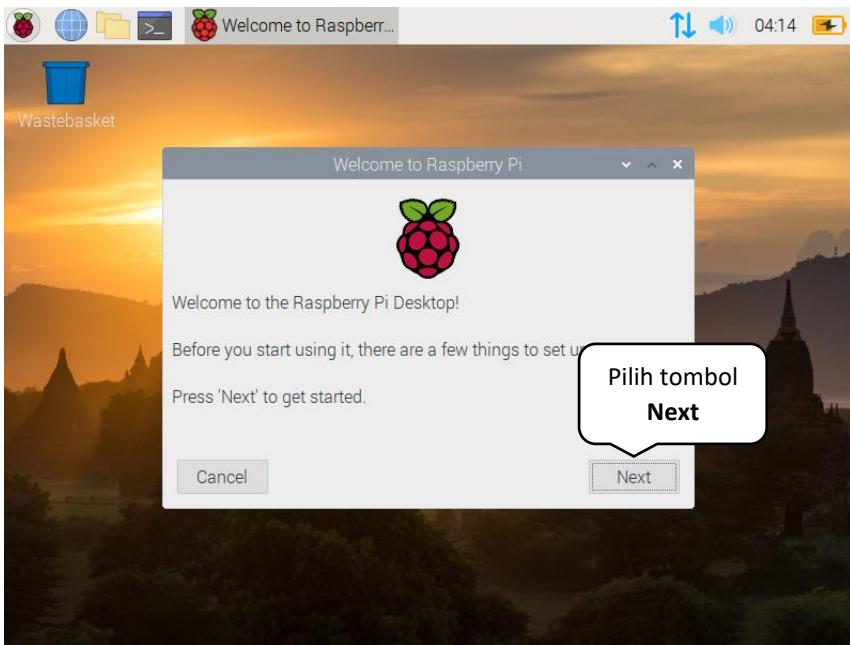
Abaikan, jangan diformat

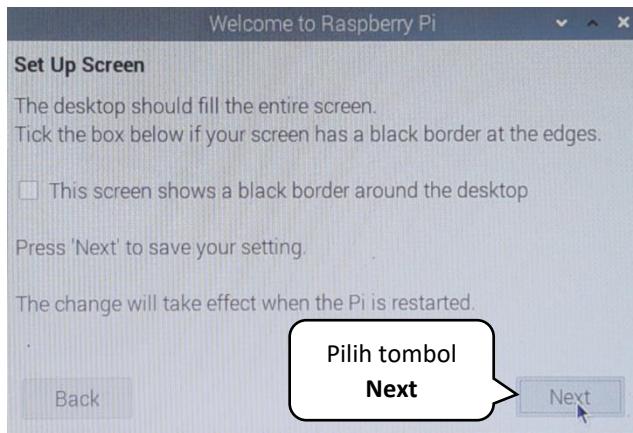
5.7 Instalasi Linux Raspbian Pada Raspberry Pi 3

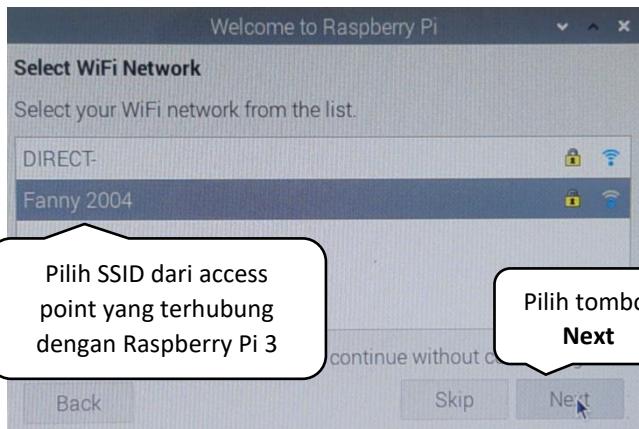
Berikut ini tahapan instalasi Linux Raspbian ke Raspberry Pi 3:

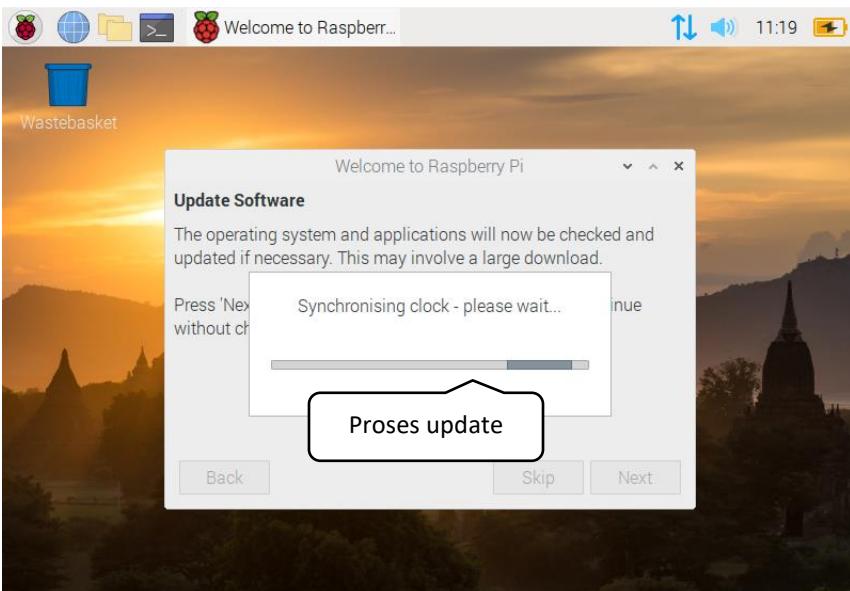
- Lepas MicroSD dari adapternya setelah diburning dengan Linux Raspbian.
- Pasang memori MicroSD ke slot MicroSD Raspberry Pi 3.
- Pasang keyboard USB dan mouse USB ke Raspberry Pi 3.
- Hubungkan jack VGA HDMI Raspberry Pi 3 ke monitor (gunakan adapter VGA jika diperlukan). Monitor hanya digunakan saat proses instalasi, setelah itu kita dapat mengkonfigurasi Raspbian dengan aplikasi remote Putty melalui jaringan komputer.
- Hubungkan Adaptor ke Raspberry Pi 3 untuk menghidupkannya.

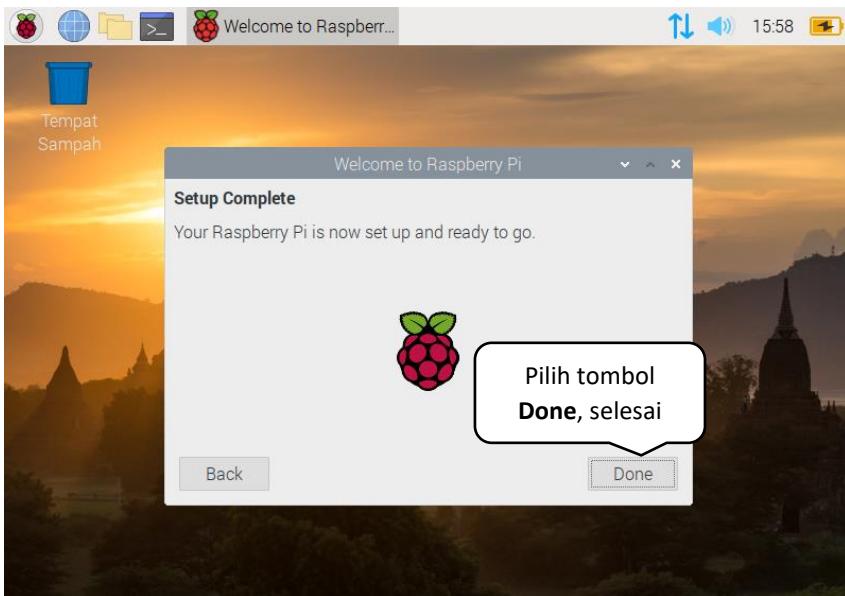
Ketika power supply dihidupkan maka Raspberry Pi 3 akan melakukan booting dan instalasi awal. Selanjutnya kuti langkah-langkahnya berikut ini:





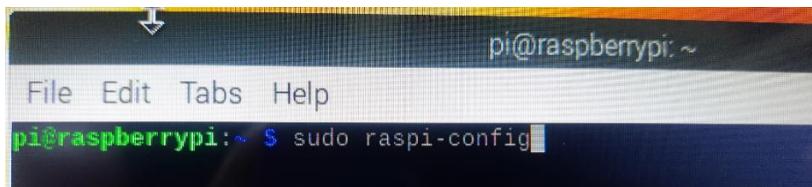




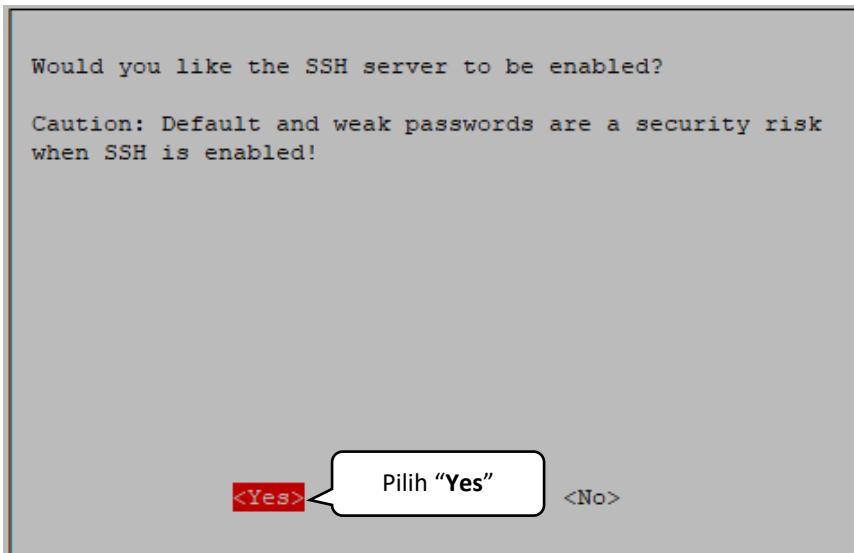
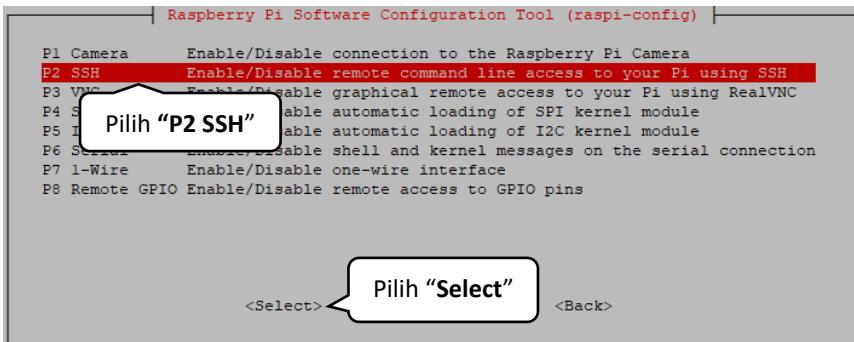
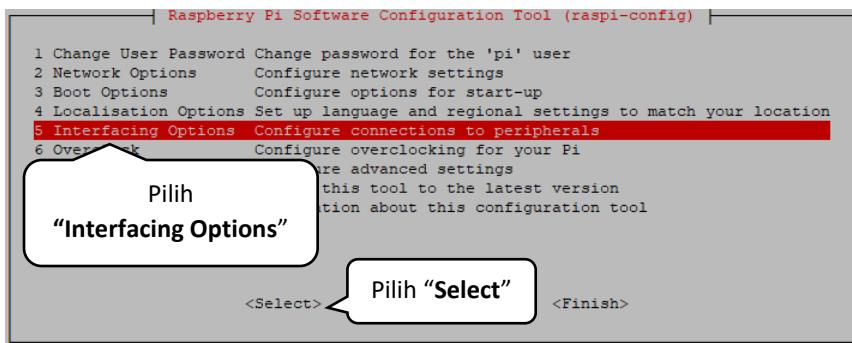


Untuk memudahkan proses instalasi Blynk, gunakan aplikasi remote SSH Putty. Tujuannya untuk mengendalikan Raspberry Pi 3 secara jarak jauh. Oleh karena itu aktifkan terlebih dahulu service SSH server pada Raspberry Pi 3 dengan cara buka aplikasi *console/terminal* dengan menggunakan perintah

`sudo raspi-config.`



Dari sini akan muncul windows daftar layanan, pilih **Interfacing Options**, kemudian pilih tombol **Select**.



```
The SSH server is enabled
```

Pilih “OK”
SSH Server aktif

<Ok>

Karena aplikasi remote membutuhkan IP tujuan untuk dikoneksikan, yaitu alamat IP Raspberry Pi 3, maka silahkan buka console Raspberry Pi 3, kemudian gunakan perintah ifconfig.

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      ether b8:27:eb:ff:b0:99  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10<host>
          loop  txqueuelen 1000  (Local Loopback)
          RX packets 0  bytes 0 (0.0 B)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 0  bytes 0 (0.0 B)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

wlan0: flags=1103<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.1.6  netmask 255.255.255.0  broadcast 192.168.1.255
          inet6 fe80::e27:ebff:feff:b099%wlan0  prefixlen 64  scopeid 0x20<link>
          ether b8:27:eb:ff:b0:99  txqueuelen 1000  (Ethernet)
          RX packets 0  bytes 0 (0.0 B)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 0  bytes 0 (0.0 B)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
          192.168.1.6 adalah IP dengan
          interface Wifi Raspberry saat ini
          collisions 0

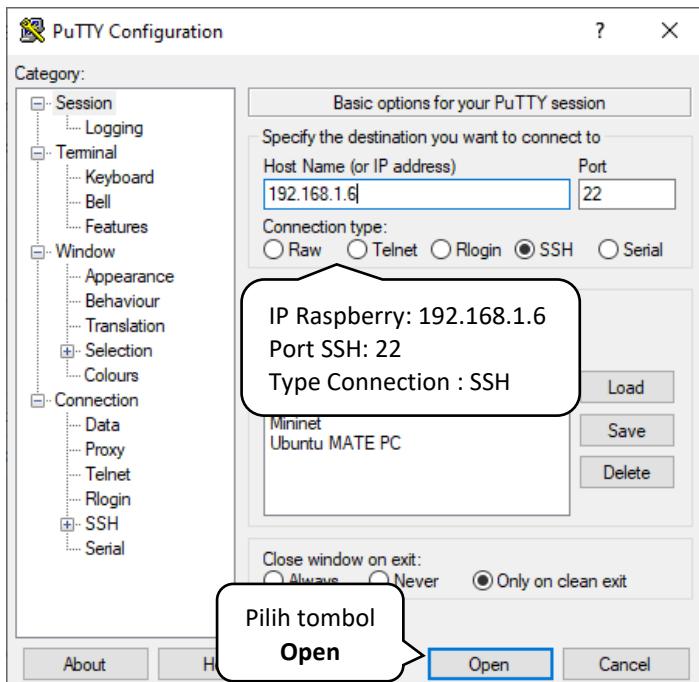
pi@raspberrypi:~ $
```

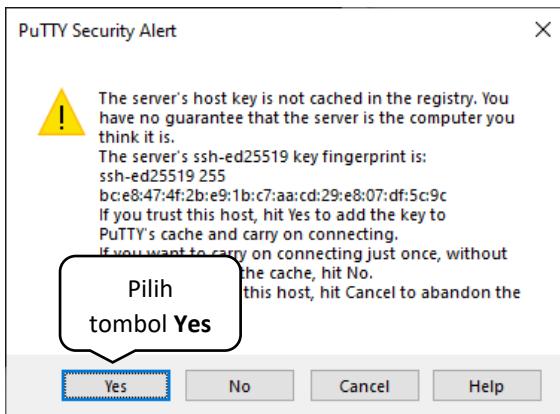
Diasumsikan Raspberry Pi 3 telah terhubung dengan *access point* sehingga Raspberry Pi 3 akan mendapatkan alamat IP dinamis dari router. Sebagai contoh alamat IP 192.168.1.6. Bisa jadi IP Anda akan berubah tergantung dari pemberian alamat IP oleh router Anda.

Untuk membuktikan apakah Raspberry Pi 3 dapat di-remote dari laptop, pertama download terlebih dahulu aplikasi putty di :

- <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- <https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.73-installer.msi> (64bit)
- <https://the.earth.li/~sgtatham/putty/latest/w32/putty-0.73-installer.msi> (32bit).

Selanjutnya instal putty dan buka aplikasinya.





Login default Raspberry Pi 3 adalah “pi”, sedangkan password-nya sama dengan password ketika Anda menginstal Raspberry Pi 3 pertamakali.

Dari sini sudah dapat me-remote Raspberry Pi 3 dari laptop.

```
pi@raspberrypi: ~
pi@raspberrypi: ~$ login as: pi
pi@192.168.1.6's password:
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

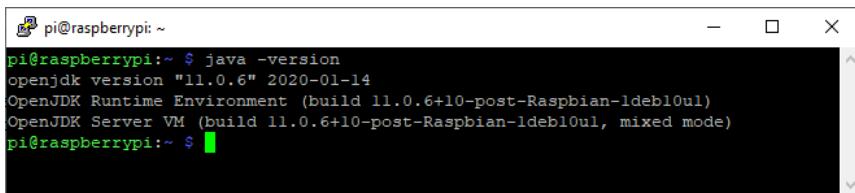
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 25 17:48:53 2020 from 192.168.1.4
pi@raspberrypi: ~ $
```

5.8 Instalasi Server IoT Blynk

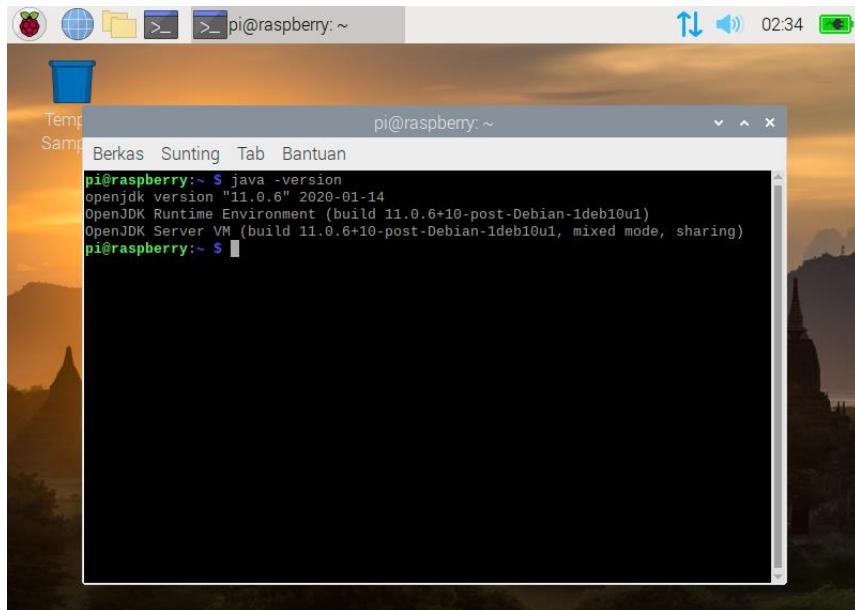
Langkah berikutnya adalah instalasi Server IoT Blynk. Anda dapat mengkonfigurasinya secara langsung melalui console/shell Raspberry Pi 3 atau melalui aplikasi putty secara remote.

Blynk membutuhkan java dalam pengoperasianya, sehingga pastikan Raspberry Pi 3 telah diinstall dengan java. Untuk mengetahuinya gunakan perintah `java -version`. Bila menggunakan aplikasi putty tampak seperti ini:



```
pi@raspberrypi: ~
pi@raspberrypi: ~ $ java -version
openjdk version "11.0.6" 2020-01-14
OpenJDK Runtime Environment (build 11.0.6+10-post-Raspbian-1deb10u1)
OpenJDK Server VM (build 11.0.6+10-post-Raspbian-1deb10u1, mixed mode)
pi@raspberrypi: ~ $
```

Bila menggunakan console Raspberry Pi 3 akan tampak seperti ini:



Bila muncul pesan seperti di bawah ini, berarti Raspberry Pi 3 telah terinstal java.

```
openjdk version "11.0.6" 2020-01-14
OpenJDK Runtime Environment (build 11.0.6+10-
post-Raspbian-1deb10u1)

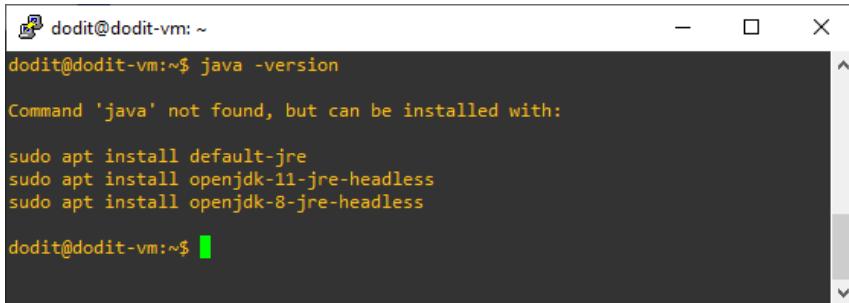
OpenJDK Server VM (build 11.0.6+10-post-Raspbian-
1deb10u1, mixed mode)
```

Namun bila tampak pesan seperti tampak di bawah ini, berarti java belum terinstal. Silahkan instal java terlebih dahulu dengan perintah

```
sudo apt install default-jre
```

atau

```
sudo apt-get install default-jdk.
```



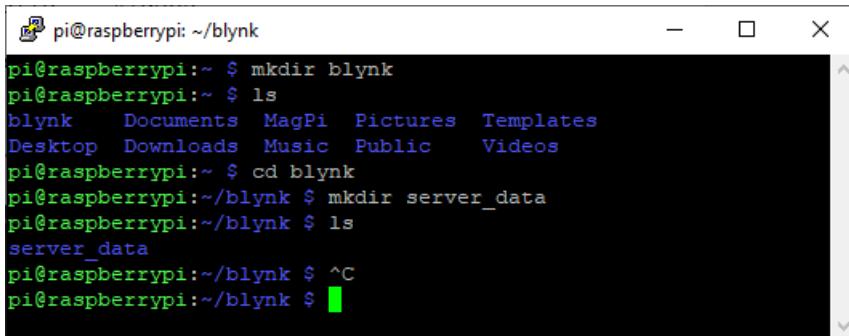
```
dodit@dodit-vm: ~
dodit@dodit-vm:~$ java -version
Command 'java' not found, but can be installed with:

sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless

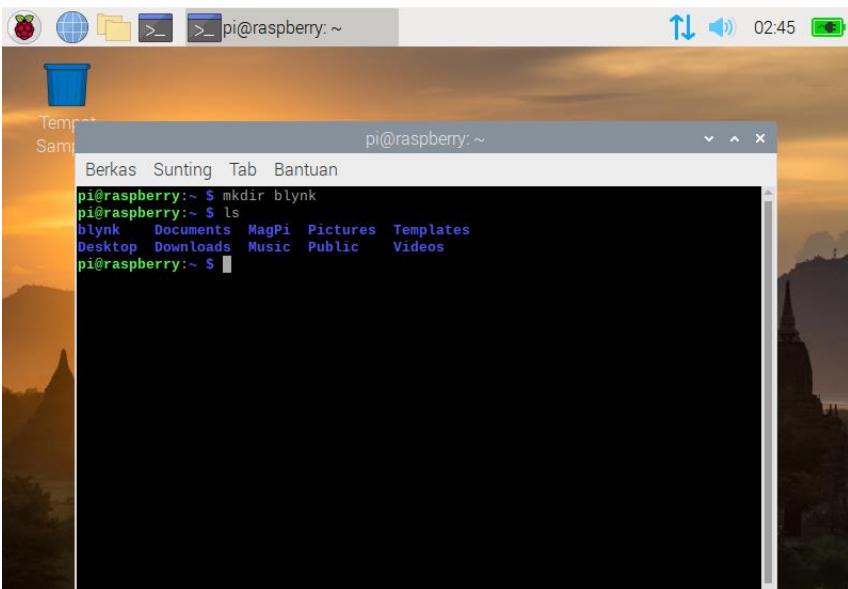
dodit@dodit-vm:~$
```

Selanjutnya adalah membuat folder dimana aplikasi Blynk akan diinstall. Gunakan perintah `mkdir blynk` yang berarti membuat folder bernama "blynk". Kemudian buat folder baru lagi bernama "server_data" di dalam folder blynk sebagai tempat penyimpanan data Blynk. Berikut perintah lengkapnya:

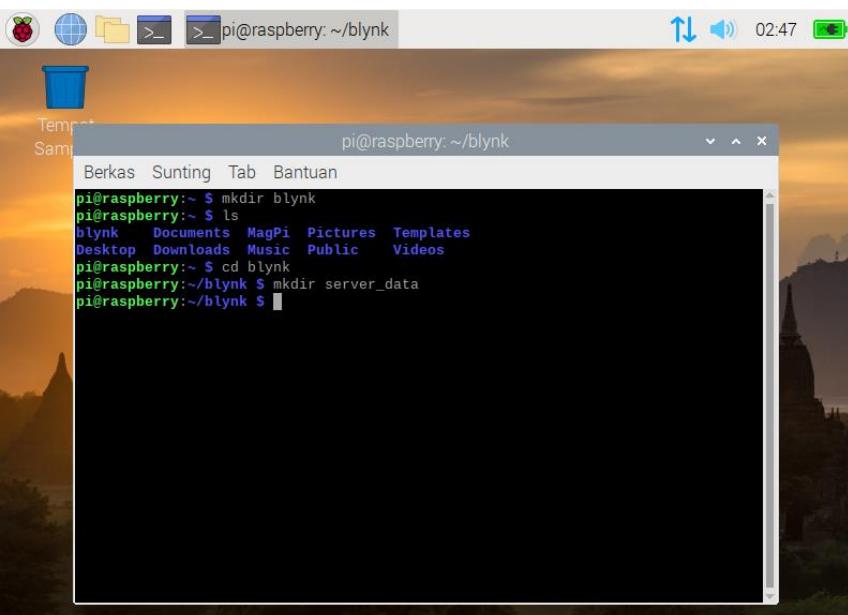
```
pi@raspberrypi:~ $ mkdir blynk
pi@raspberrypi:~ $ cd blynk
pi@raspberrypi:~/blynk $ mkdir server_data
```



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ mkdir blynk
pi@raspberrypi:~ $ ls
blynk  Documents  MagPi  Pictures  Templates
Desktop  Downloads  Music  Public    Videos
pi@raspberrypi:~ $ cd blynk
pi@raspberrypi:~/blynk $ mkdir server_data
pi@raspberrypi:~/blynk $ ls
server_data
pi@raspberrypi:~/blynk $ ^C
pi@raspberrypi:~/blynk $
```



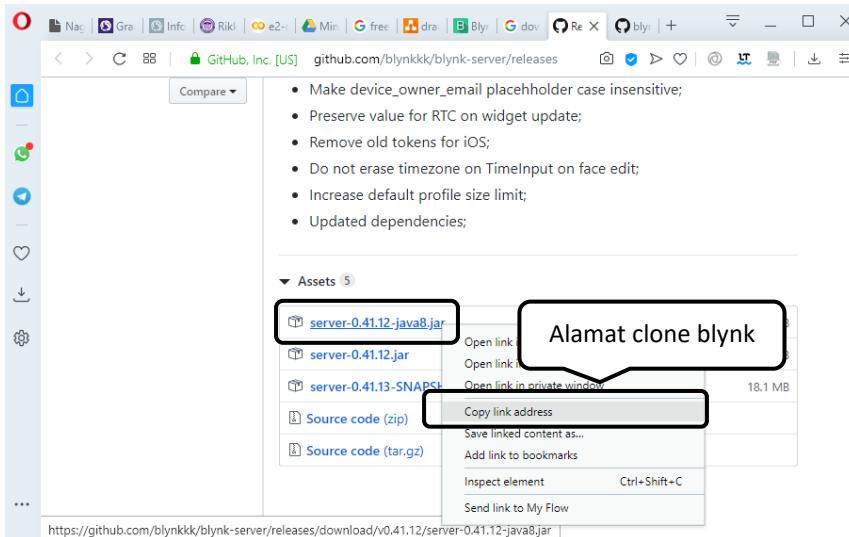
```
Berkas Sunting Tab Bantuan
pi@raspberry:~ $ mkdir blynk
pi@raspberry:~ $ ls
blynk  Documents  MagPi  Pictures  Templates
Desktop  Downloads  Music  Public  Videos
pi@raspberry:~ $
```



```
Berkas Sunting Tab Bantuan
pi@raspberry:~ $ mkdir blynk
pi@raspberry:~ $ ls
blynk  Documents  MagPi  Pictures  Templates
Desktop  Downloads  Music  Public  Videos
pi@raspberry:~ $ cd blynk
pi@raspberry:~/blynk $ mkdir server_data
pi@raspberry:~/blynk $
```

Masih berada di dalam folder “blynk”, kita akan meng-install aplikasi IoT Server Blynk dengan cara meng-*cloning* file Blynk dari repository github ke Raspberry Pi 3. Untuk mengetahui release

Blynk versi terbaru, silahkan kunjungi halaman ini <https://github.com/blynkkk/blynk-server/releases>.



Perintah untuk meng-clone aplikasi Server IoT blynk dari github adalah sebagai berikut:

```
 wget https://github.com/blynkkk/blynk-server/releases/download/v0.41.12/server-0.41.12-java8.jar
```

```
pi@raspberrypi: ~$ wget "https://github.com/blynkkk/blynk-server/releases/download/v0.41.12/server-0.41.12-java8.jar"
--2020-03-25 22:40:01-- https://github.com/blynkkk/blynk-server/releases/download/v0.41.12/server-0.41.12-java8.jar
Resolving github.com (github.com) ... 13.250.177.223
Connecting to github.com (github.com) |13.250.177.223|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/32753016/fbba7400-1812-11ea-997a-52fea0bb24fd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200325%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200325T154007Z&X-Amz-Expires=300&X-Amz-Signature=75dd27d2f4b32a15f4a4399da5c0de2cc3b407eec16be49c358a8b3203d0589b&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dserver-0.41.12-java8.jar&response-content-type=application%2Foctet-stream [following]
--2020-03-25 22:40:07-- https://github-production-release-asset-2e65be.s3.amazonaws.com/32753016/fbba7400-1812-11ea-997a-52fea0bb24fd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200325%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200325T154007Z&X-Amz-Expires=300&X-Amz-Signature=75dd27d2f4b32a15f4a4399da5c0de2cc3b407eec16be49c358a8b3203d0589b&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dserver-0.41.12-java8.jar&response-content-type=application%2Foctet-stream
```

```
pi@raspberrypi:~/blynk
4-HMAC-SHA256&X-Amz-Credential=AKIAIWNYAX4CSVEH53A%2F20200325%2Fsus-east-1%2F
s%2Faws4_request&X-Amz-Date=20200325T154007Z&X-Amz-Expires=300&X-Amz-Signature=
re=75dd27d2f4b32a15f4a4399da5c0de2cc3b407eec16be49c358a8b3203d0589b&X-Amz-Sig
nedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filen
ame%3Dserver-0.41.12-java8.jar&response-content-type=application%2Foctet-stre
am
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-pro
duction-release-asset-2e65be.s3.amazonaws.com) ... 52.216.30.68
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-
production-release-asset-2e65be.s3.amazonaws.com) |52.216.30.68|:443... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 18944356 (18M) [application/octet-stream]
Saving to: 'server-0.41.12-java8.jar'

server-0.41.12-java 100%[=====] 18.07M 225KB/s in 2m 13s

2020-03-25 22:42:24 (139 KB/s) - 'server-0.41.12-java8.jar' saved [18944356/1
8944356]

pi@raspberrypi:~/blynk $
```

```
Tempat
Sampah
pi@raspberrypi:~/blynk
pi@raspberrypi:~/blynk
Berkas Sunting Tab Bantuan
pi@raspberrypi:~/blynk
pi@raspberrypi:~/blynk $ wget https://github.com/blynkkk/blynk-server/releases/download/v0.41.12/server-0.41.12.jar
--2020-03-03 03:08:51-- https://github.com/blynkkk/blynk-server/releases/download/v0.
41.12/server-0.41.12.jar
Resolving github.com (github.com)... 52.74.223.119
Menghubungi github.com (github.com)|52.74.223.119|:443... terhubung.
Permintaan HTTP dikirimkan, menunggu balasan... 302 Found
Lokasi: https://github-production-release-asset-2e65be.s3.amazonaws.com/32753016/6cad5
c00-1812-11ea-937c-5009c1f647c97X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIA
IWNYAX4CSVEH53A%2F20200304%2Fsus-east-1%2Fs%2Faws4_request&X-Amz-Date=20200304T064042
Z&X-Amz-Expires=300&X-Amz-Signature=cdd2cdf8c03c452fb32ef4f699613414a46da947b643b904
7d0ebdbb9d114d&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attach
ment%3B%20filename%3Dserver-0.41.12.jar&response-content-type=application%2Foctet-stre
am [mengikuti]
--2020-03-03 03:08:52-- https://github-production-release-asset-2e65be.s3.amazonaws.c
om/32753016/6cad5c00-1812-11ea-937c-5009c1f647c97X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Am
z-Credential=AKIAIWNYAX4CSVEH53A%2F20200304%2Fsus-east-1%2Fs%2Faws4_request&X-Amz-Dat
e=20200304T064042Z&X-Amz-Expires=300&X-Amz-Signature=cdd2cdf8c03c452fb32ef4f69961341
4a46da947b643b9047d0ebdbb9d114d&X-Amz-SignedHeaders=host&actor_id=0&response-content-d
isposition=attachment%3B%20filename%3Dserver-0.41.12.jar&response-content-type=appli
cation%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-r
elease-asset-2e65be.s3.amazonaws.com)... 52.216.139.171
```

```
Berkas Sunting Tab Bantuan
Z&X-Amz-Expires=300&X-Amz-Signature=cdd2cdf8c03c452fbb32efd4f69961341a46da947b643b9047d0ebdbb9d114d&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dserver-0.41.12.jar&response-content-type=application%2Foctet-stream [mengikuti]
--2020-03-03 03:08:52-- https://github-production-release-asset-2e65be.s3.amazonaws.com/32753016/6cad5c00-1812-11ea-937c-5009c1f647c9?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNVJAX4CSVEH53A%2F20200304%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200304T064042Z&X-Amz-Expires=300&X-Amz-Signature=cdd2cdf8c03c452fbb32efd4f699613414a46da947b643b9047d0ebdbb9d114d&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dserver-0.41.12.jar&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.139.171
Menghubungi github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.139.171|:443... terhubung.
Permintaan HTTP dikirimkan, menunggu balasan... 200 OK
Besar: 18944433 (18M) [application/octet-stream]
Simpan ke: 'server-0.41.12.jar'

server-0.41.12.jar 100%[=====] 18,07M 486KB/s in 52s
2020-03-03 03:09:46 (352 KB/s) - 'server-0.41.12.jar' disimpan [18944433/18944433]

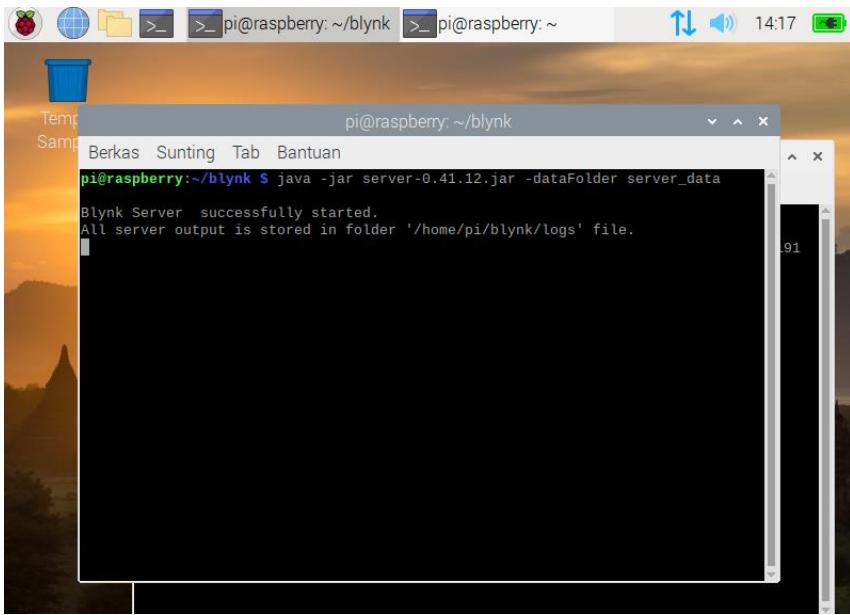
pi@raspberry:~/blynk $
```

Selanjutnya, untuk mengaktifkan Server IoT Blynk gunakan perintah:

```
java -jar server-0.41.12-java8.jar -dataFolder
server_data &
```

```
pi@raspberrypi:~ $ cd blynk
pi@raspberrypi:~/blynk $ ls
-dataFolder logs server-0.41.12-java8.jar server_data static
pi@raspberrypi:~/blynk $ java -jar server-0.41.12-java8.jar -dataFolder server_data &
[1] 878
pi@raspberrypi:~/blynk $
Blynk Server successfully started.
All server output is stored in folder '/home/pi/blynk/logs' file.
Your Admin url is https://127.0.1.1:9443/admin
Your Admin login email is admin@blynk.cc
Your Admin password is admin
pi@raspberrypi:~/blynk $
```

Jika muncul pesan seperti di atas maka Server IoT Blynk telah aktif dan siap digunakan.



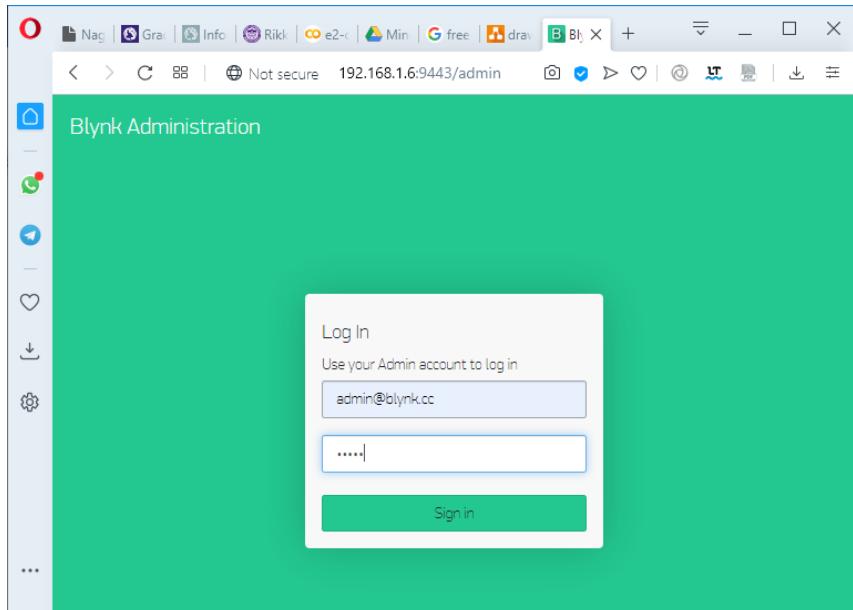
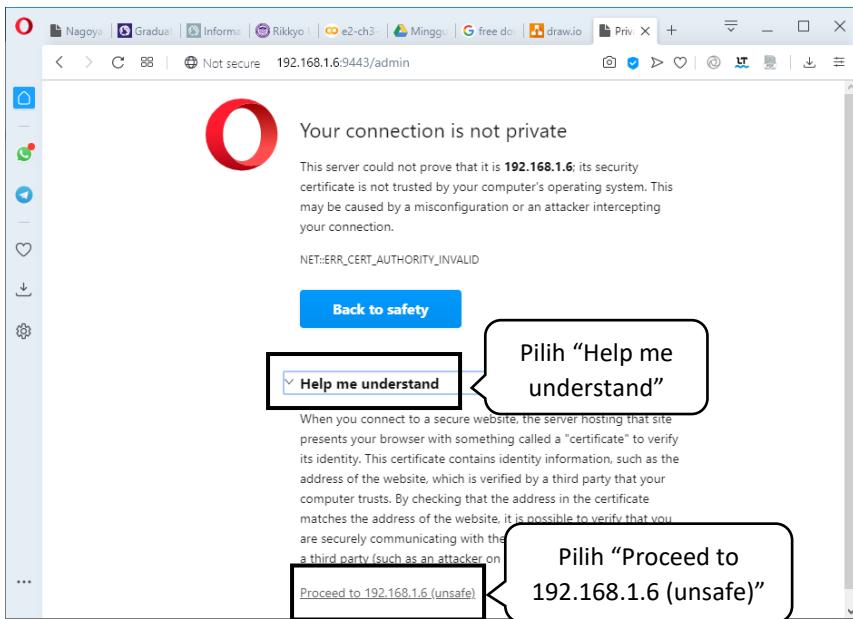
A screenshot of a terminal window titled "pi@raspberry: ~/" running on a Raspberry Pi. The window shows the command "java -jar server-0.41.12.jar -dataFolder server_data" being executed, followed by the output: "Blynk Server successfully started. All server output is stored in folder '/home/pi/blynk/logs' file." The terminal has a dark background with white text. The window title bar also displays the command and output.

Anda bisa menggunakan browser internet untuk mengadministrasi *user*, mengkonfigurasi, dan lain-lain melalui halaman Dashboard Blynk, baik secara lokal di Raspberry Pi 3 itu sendiri atau melalui komputer lain dalam satu jaringan.

Berikut ini cara untuk membuka halaman Dashboard Blynk secara remote. Dimana saya menggunakan browser internet Opera.

- Alamat URL secara remote:
<https://192.168.1.6:9443/admin>
- Alamat URL secara lokal:
<https://127.0.0.1:9443/admin>
- User admin Blynk default: **admin@blynk.cc**
- Password admin Blynk default: **admin**.

User dan password admin dapat diubah melalui Dashboard Blynk. Tetapi untuk saat ini kita biarkan menggunakan user dan password default.



Tampilan di bawah ini merupakan user yang telah terdaftar dalam Server IoT Blynk. User yang dimaksud adalah user yang memiliki terasosiasi dengan *smart devic*. Satu user bisa memiliki beberapa

smart device. Namun untuk saat ini belum ada user yang terdaftar kecuali Admin. Hal ini akan kita bahas dibagian selanjutnya.

The screenshot shows a web browser window titled "Blynk Administration". The URL is 192.168.1.6:9443/admin#/users/list. On the left, there is a sidebar with icons for Nagoya, Gradua, Inform, Rikkyo, e2-ch3, Minggi, free do, draw.io, and Blynk. The "Users" icon is highlighted with a black rectangle. The main content area is titled "Users list". It features a search bar with a magnifying glass icon and an "Export" button. A table displays one user entry:

<input type="checkbox"/>	Email	AppName	# Of Projects	LastModifiedTs
<input type="checkbox"/>	admin@blynk.cc	Blynk	0	2020-03-25 23:05:49

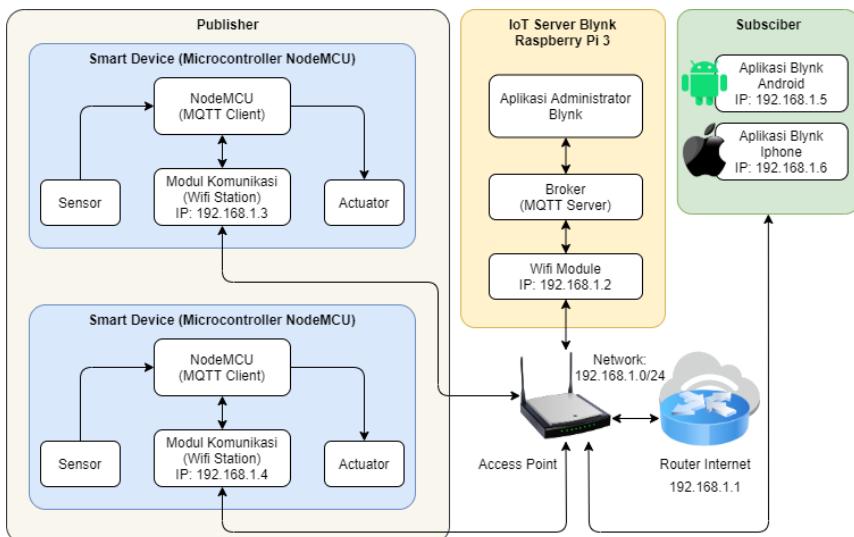
At the bottom of the table, it says "1 - 1 on 1".

BAB 6

IoT Server Blynk

Pada bagian sebelumnya kita telah membahas bagaimana memprogram microcontroller NodeMCU dan bagaimana membangun server IoT Blynk pada Raspberry Pi 3. Sekarang saatnya mengkomunikasikan antara NodeMCU sebagai smart device dan Blynk sebagai Server IoT.

Topologi jaringan proyek ini tetap sama dengan sebelumnya, yaitu menggunakan jaringan lokal yang terhubung dengan internet.



6.1 Blynk Client Android

Blynk menyediakan aplikasi klien yang berjalan pada perangkat *mobile* dengan platform Android dan IoS. Dengan aplikasi klien blynk, kita dapat memonitor dan mengendalikan "things" / smart device secara real-time secara jarak jauh.

Bila mengacu pada projek microcontroller NodeMCU sebelumnya, dimana ia melibatkan sensor DHT11, LDR, dan GPS maka dengan aplikasi klien Blynk dapat dilihat perubahan temperatur, kelembaban, intensitas cahaya lingkungan secara berkala, baik

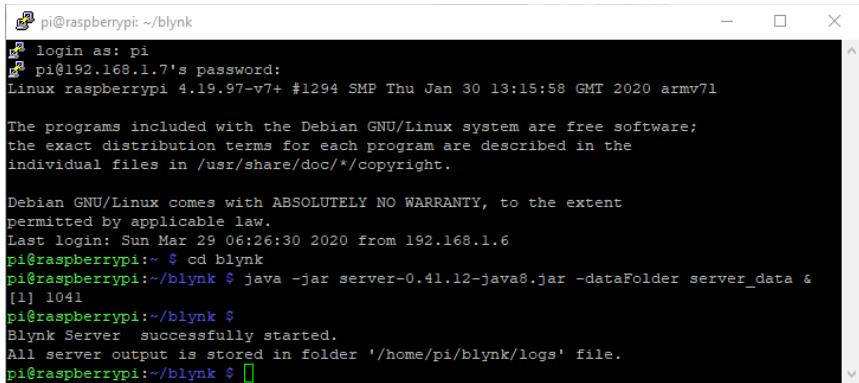
berupa angka atau grafis chart. Dapat diketahui pula lokasi koordinat smart device saat ini untuk ditampilkan pada peta digital.

Dengan aplikasi klien blynk, memungkinkan pula bagi kita untuk mengendalikan smart device yang telah dibangun. Misalnya menghidupkan sebagian atau semua LED dan menggerakan motor servo sekian derajat.

6.2 Instalasi Aplikasi Blynk Client Android

Sebelum meng-install blynk client android, pastikan terlebih dahulu Server IoT Blynk pada Raspberry Pi 3 telah aktif. Cara mengaktifkannya, dari dalam folder “blynk” gunakan perintah berikut:

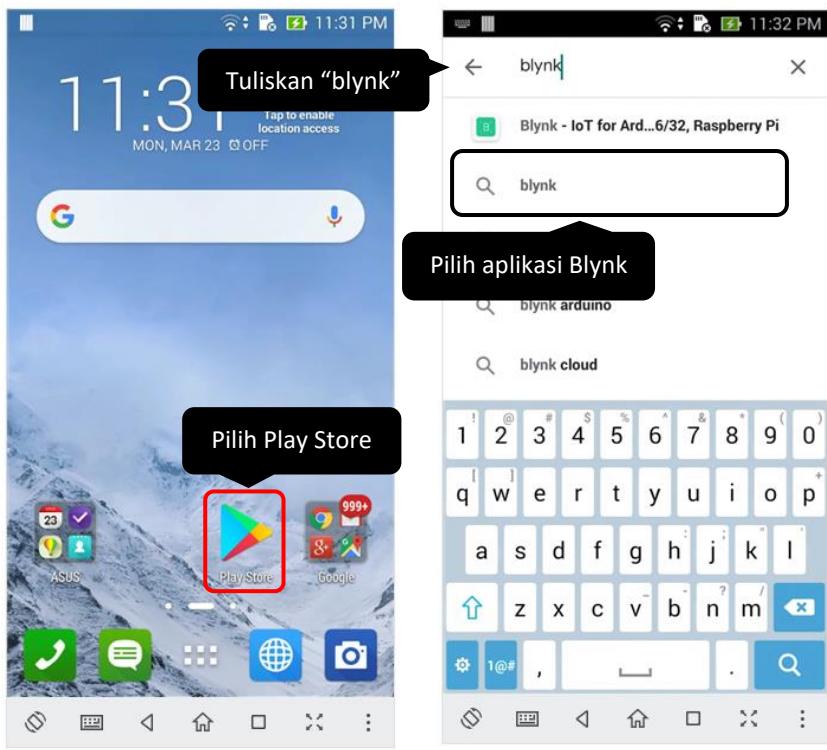
```
java -jar server-0.41.12-jar8.jar -dataFolder  
server_data &
```



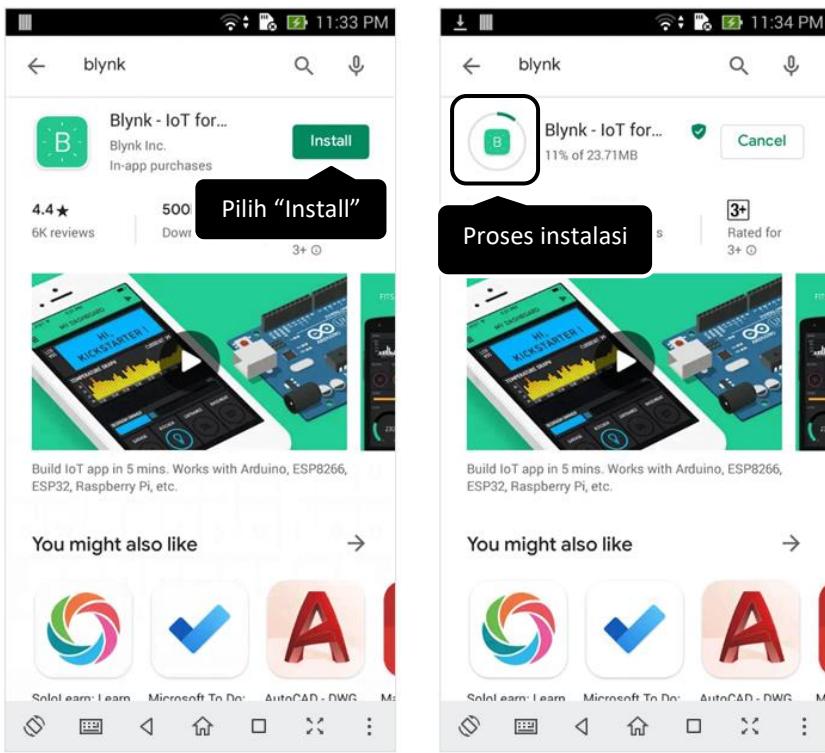
The screenshot shows a terminal window titled "pi@raspberrypi: ~ / blynk". It displays the following command-line session:

```
pi@raspberrypi: ~ / blynk  
pi@raspberrypi: ~ / blynk$ login as: pi  
pi@raspberrypi: ~ / blynk$ pi@192.168.1.7's password:  
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/* /copyright.  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sun Mar 29 06:26:30 2020 from 192.168.1.6  
pi@raspberrypi: ~ $ cd blynk  
pi@raspberrypi: ~ / blynk $ java -jar server-0.41.12-jar8.jar -dataFolder server_data &  
[1] 1041  
pi@raspberrypi: ~ / blynk $  
Blynk Server successfully started.  
All server output is stored in folder '/home/pi/blynk/logs' file.  
pi@raspberrypi: ~ / blynk $
```

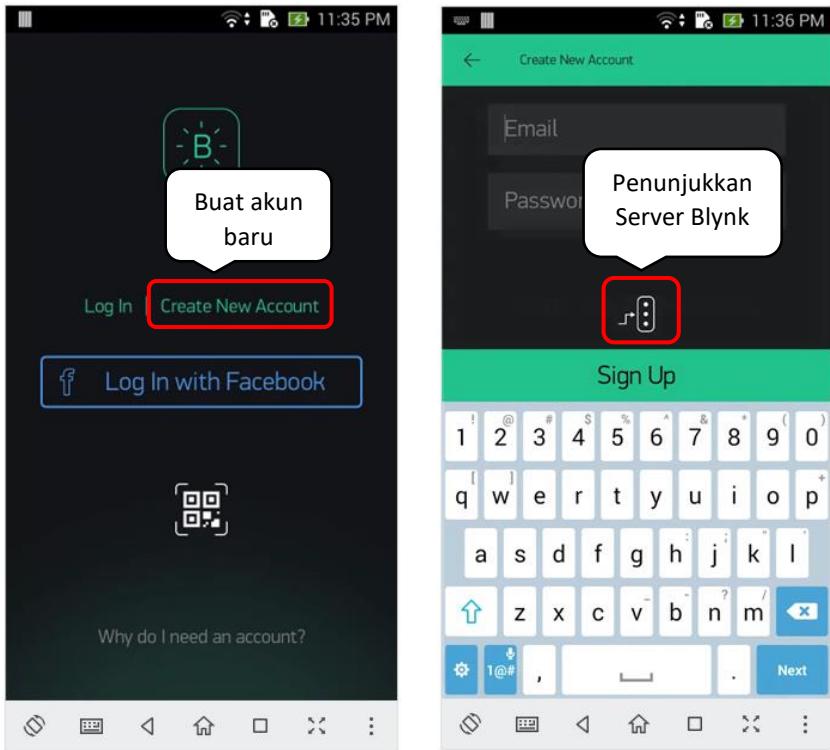
Selanjutnya, buka aplikasi Play Store Android dengan menuliskan “blynk” pada textbox pencarian.



Kemudian pilih tombol “Install”, tunggu beberapa saat sampai proses instalasi blynk selesai.

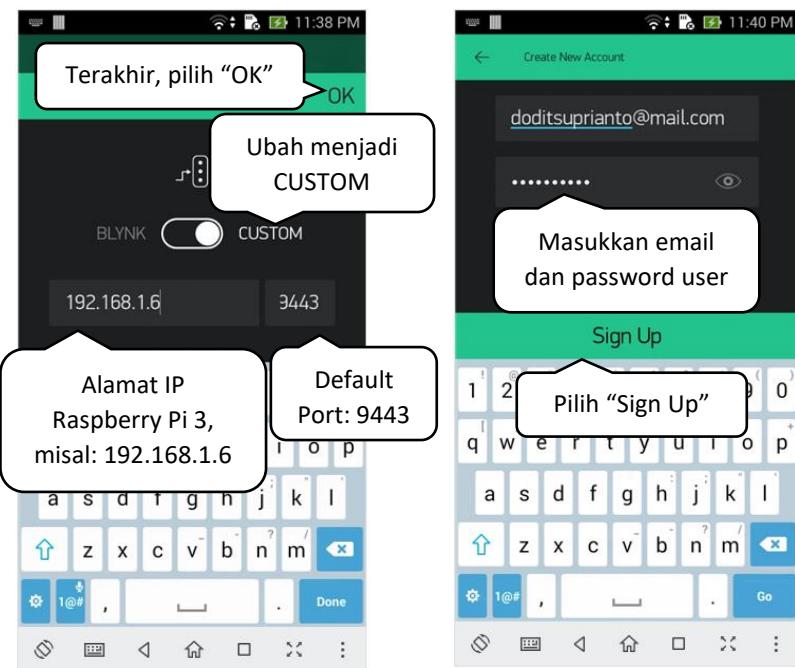


Karena kita akan membuat akun baru maka pilih "Create New Account". Di sini kita harus mengubah penunjukan server Blynk. Secara default, server yang ditunjuk adalah alamat Server IoT Blynk Cloud, oleh karena itu kita harus mengubahnya ke server Blynk Raspberry Pi 3 yang telah dibangun.



Pilih tombol selektor menjadi “CUSTOM”, tuliskan alamat IP Server IoT Blynk Raspberry Pi 3, misal alamat 192.168.1.6, dan Port defaultnya adalah 9443. Setelah itu pilih tombol “OK”.

Pada halaman berikutnya tulis email user dan passwordnya, kemudian klik tombol “Sign Up”.

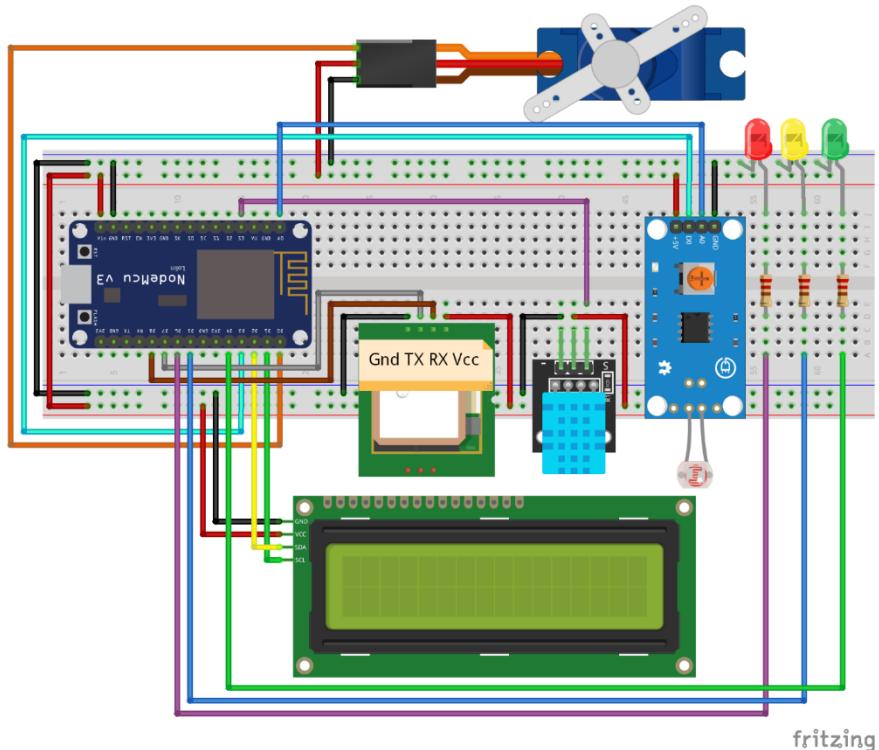


Setelah akun Blynk terbentuk maka data tersebut akan disimpan di dalam server Blynk. Untuk mengetahuinya silahkan buka dashboard blynk dari browser internet dengan alamat, misal <https://192.168.1.7:9443/admin/>. Dapat dilihat bahwa user list dengan akun email doditsuprianto@gmail.com telah berhasil ditambahkan.

Email	AppName	# Of Projects	LastModifiedTs
doditsuprianto@mail.com	Blynk	1	2020-03-29 18:06:06
admin@blynk.cc	Blynk	0	2020-03-25 23:05:49

6.3 Antarmuka Blynk Client Pada Android

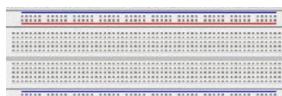
Adapun rancangan fisik dan pengkabelan Smart Device NodeMCU adalah sebagai berikut:



Kebutuhan bahan smart device:

Bahan	Jumlah	Nilai	Keterangan
LED 5MM	3 pcs		
Resistor 1/4 watt	5 pcs	220Ohm	
Sensor LDR	1 pcs		



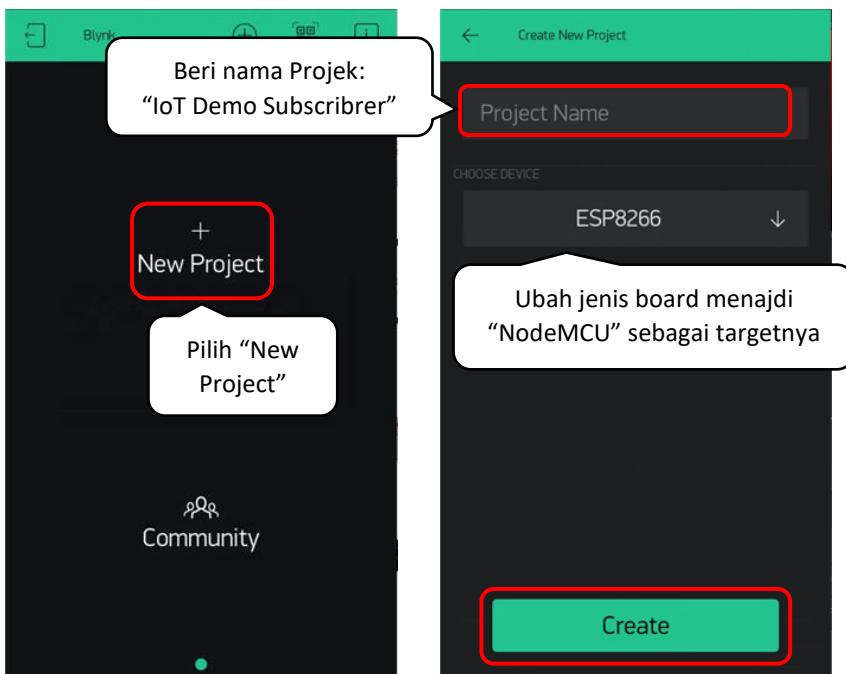
Bahan	Jumlah	Nilai	Keterangan
Servo SG-90	1 pcs		
Sensor DHT11	1 pcs		
LCD 1602 + Module I2C	1 pcs		
Sensor GPS Ublox NEO-6M	1 pcs		
NodeMCU Amica	1 pcs		
Project Board	1 pcs		

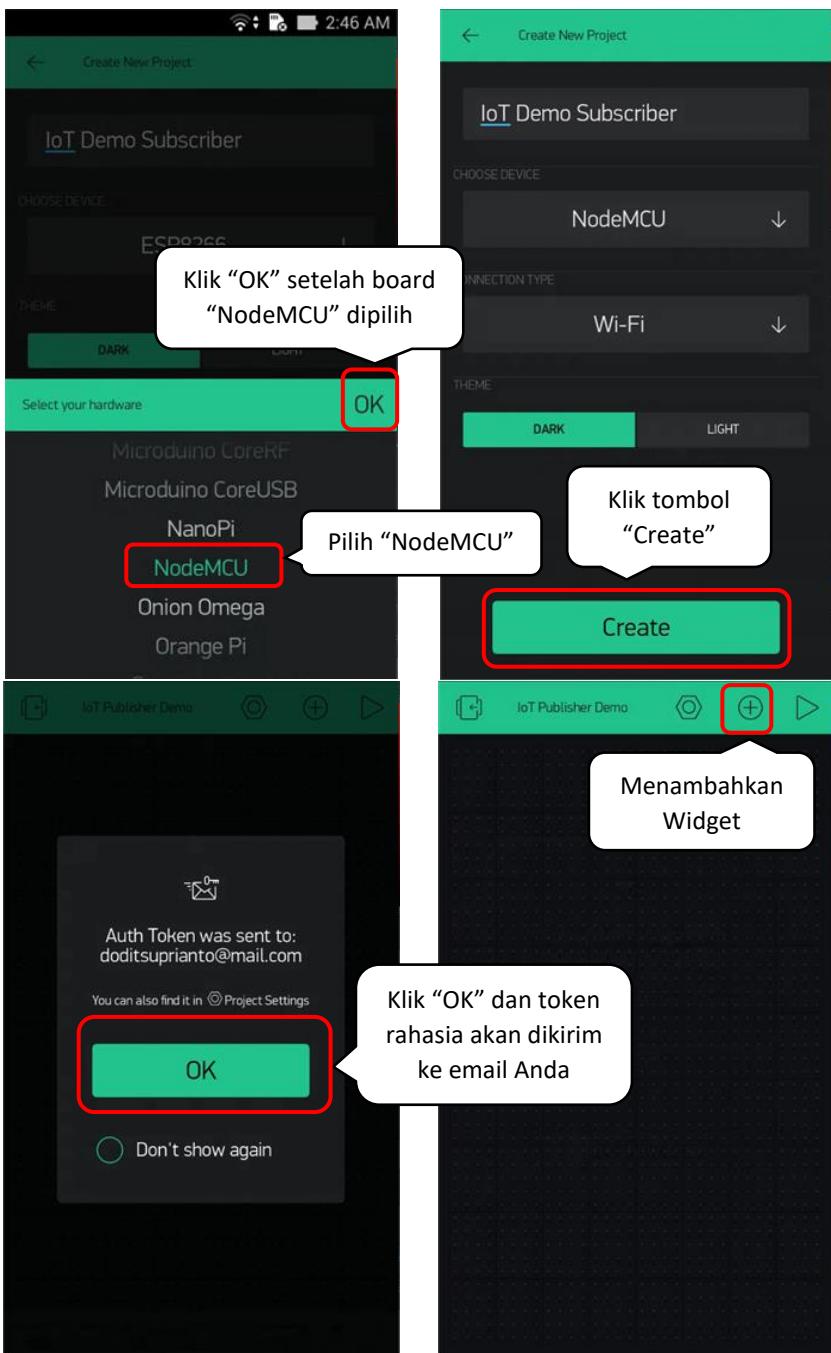
Pemetaan variabel antara pin input/output smart device NodeMCU dan aplikasi blynk client adalah sebagai berikut:

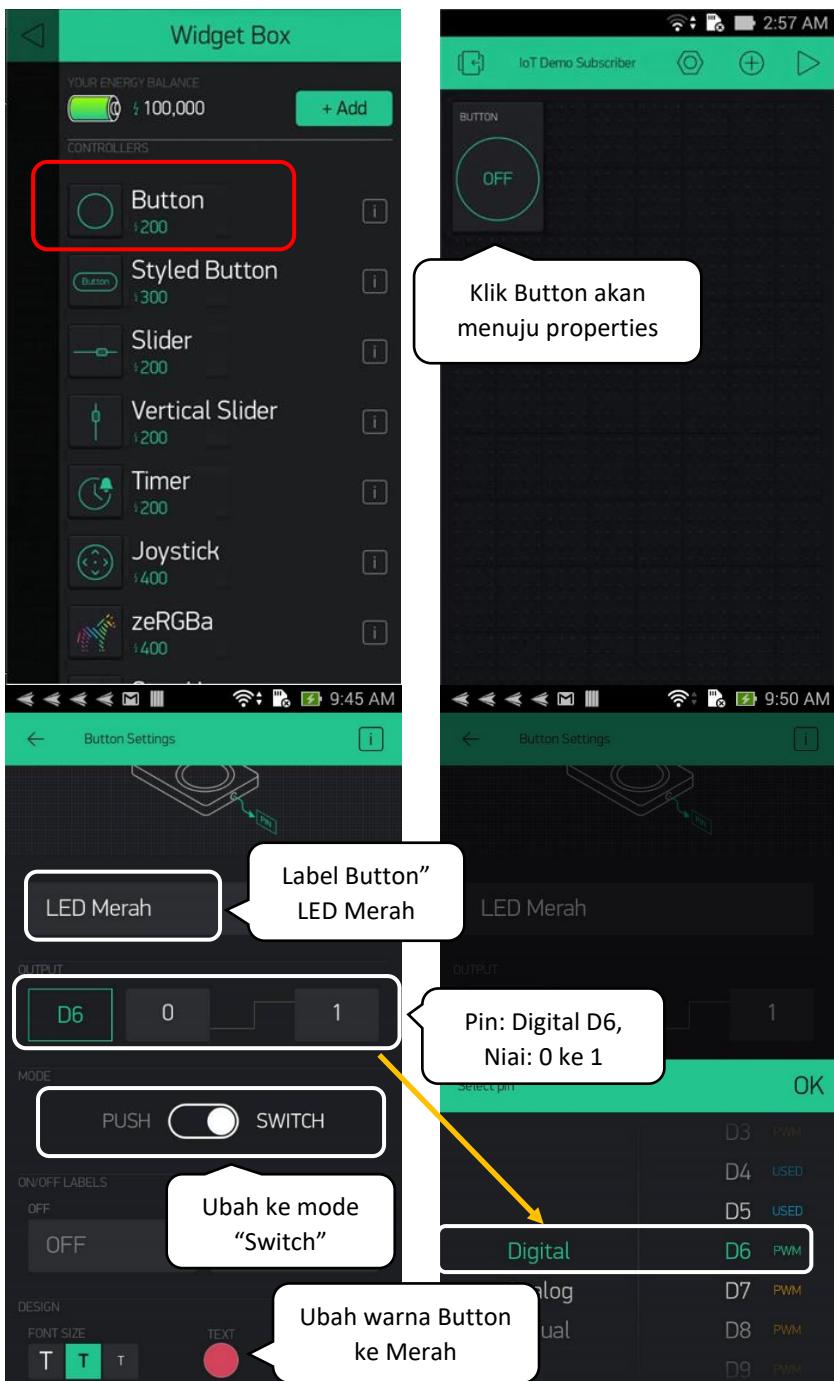
No	Jenis Komponen	Pin NodeMCU		Tipe Variabel Blynk	Nama Variable Blynk
1	LED Merah	GPIO2	→	Digital	D4
2	LED Kuning	GPIO14	→	Digital	D5
3	LED Hijau	GPIO12	→	Digital	D6
4	Servo SG-90	GPIO16	→	Virtual	V0

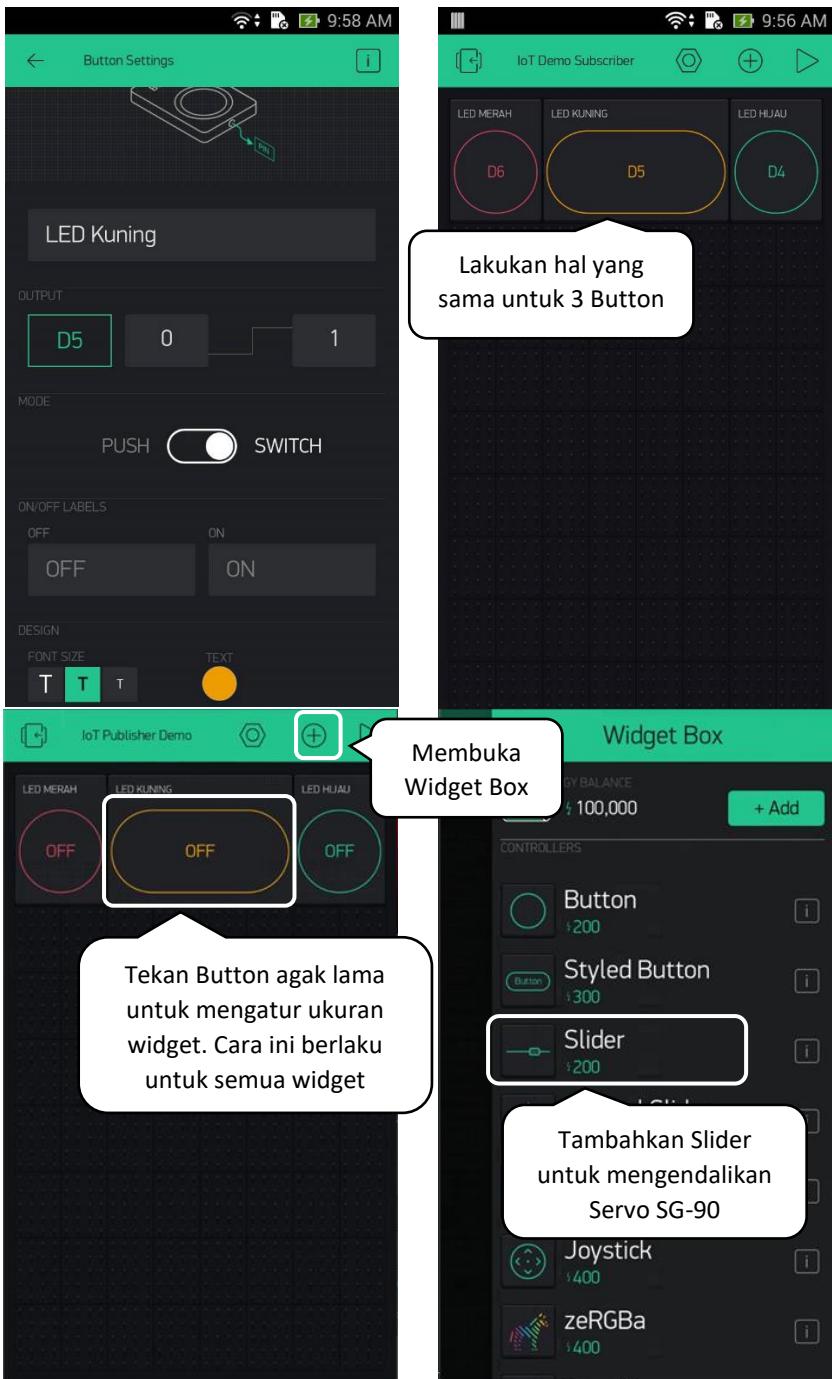
No	Jenis Komponen	Pin NodeMCU		Tipe Variabel Blynk	Nama Variable Blynk
5	DHT11: Temperatur	GPIO15	→	Virtual	V1
6	DHT11: Kelembaban	GPIO15	→	Virtual	V2
7	LDR	A0	→	Virtual	V3
8	GPS	GPIO15, GPIO13	→	Virtual	V6

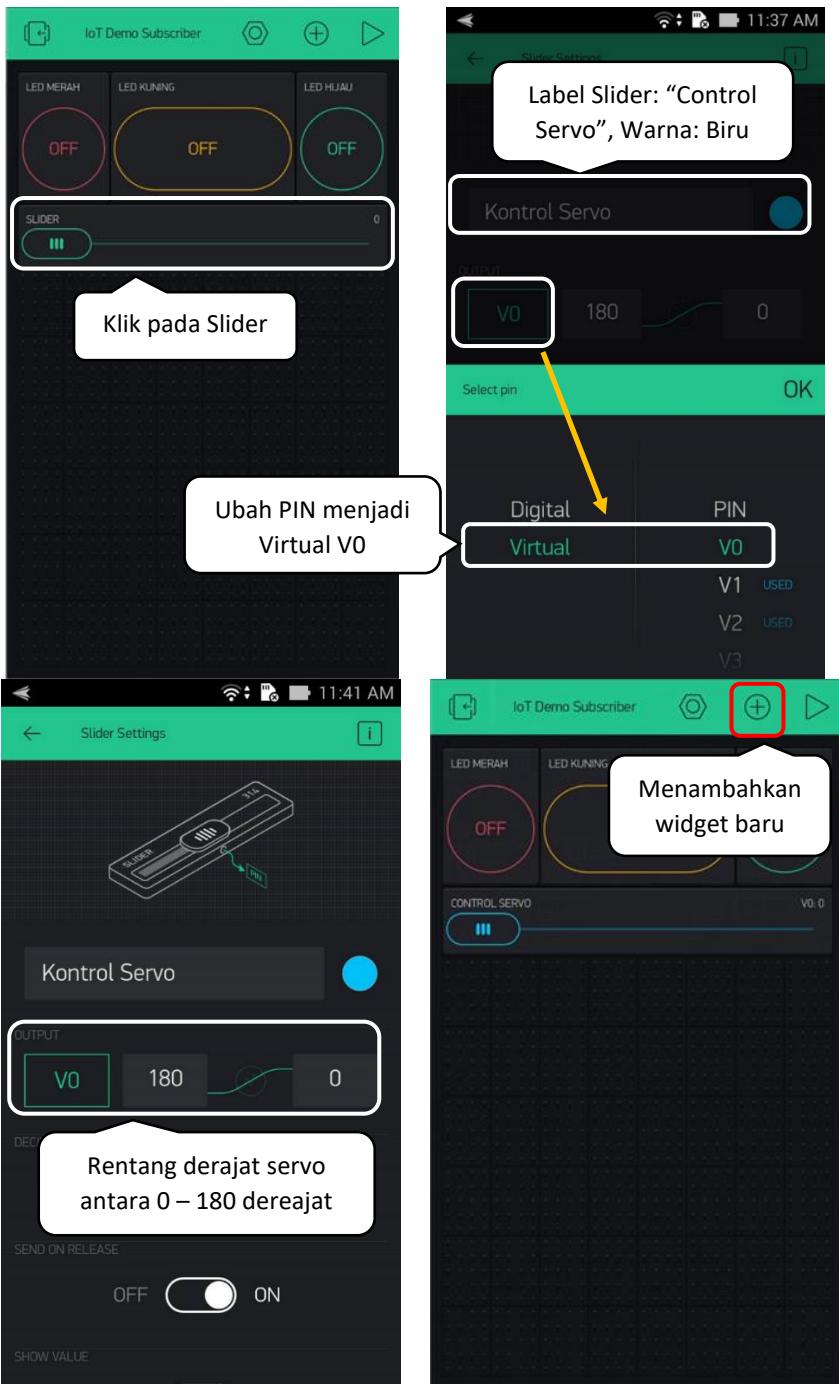
Berikut ini langkah-langkah merancang dan mengkonfigurasi antarmuka Blynk Client. Kita harus hati-hati saat menentukan alamat PIN antara Blynk Client dan NodeMCU. Alamat PIN antara rancangan fisik smart device microcontroller NodeMCU dan Blynk Client harus sesuai.











Widget Box

YOUR ENERGY BALANCE: 99,200 + Add

Value Display: 3.141 +200

Labeled Value: 25°C +400

LED: +100

Gauge: +300

Tambahkan widget Gauge

Terminal: +200

Video Streaming: +500

Gauge Settings

Label Gauge: "Temperatur"

Temperatur

V1 0 100

/pin/°C

Beri satuan derajat celcius

Beri warna merah

IoT Demo Subscriber

LED MERAH: OFF

LED KUNING: OFF

LED HIJAU: OFF

CONTROL SERVO: V0.0

Gauge: 1023

Klik Gauge untuk konfigurasi

IoT Demo Subscriber

LED MERAH: OFF

LED KUNING: OFF

LED HIJAU: OFF

V0.0

V1 KELEMBABAN

V2

Lakukan hal yang sama untuk Gauge Kelembaban. Nilai virtual adalah V2

The image consists of three screenshots from the Widget Box application, illustrating the configuration of a SuperChart component.

Screenshot 1: Widget Box - SuperChart Selection

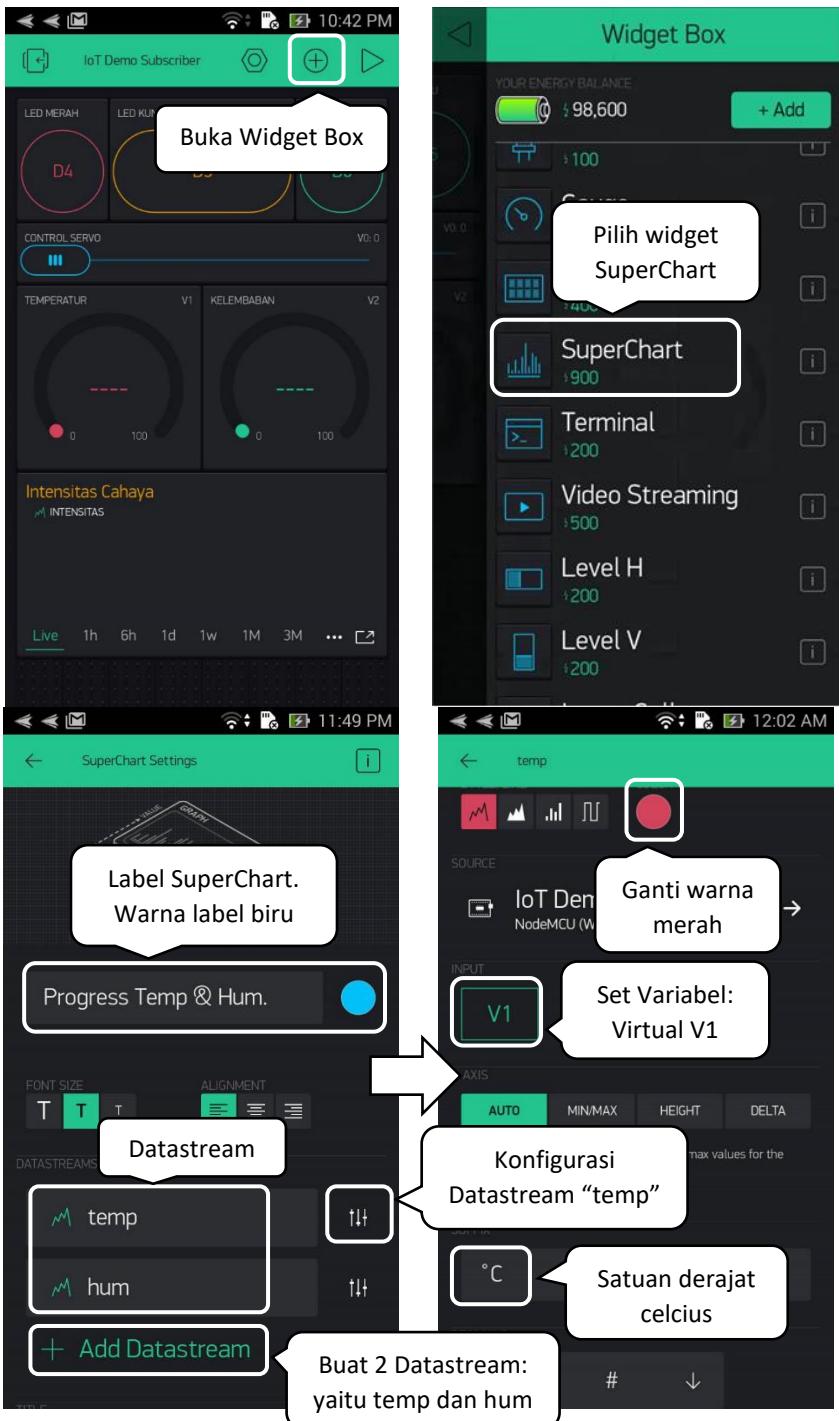
A list of widget options on a dark-themed screen. The "SuperChart" option is highlighted with a white border and a green callout bubble containing the text: "Label Superchart. Ubah warna jadi kuning" (Change the color of the Superchart label to yellow).

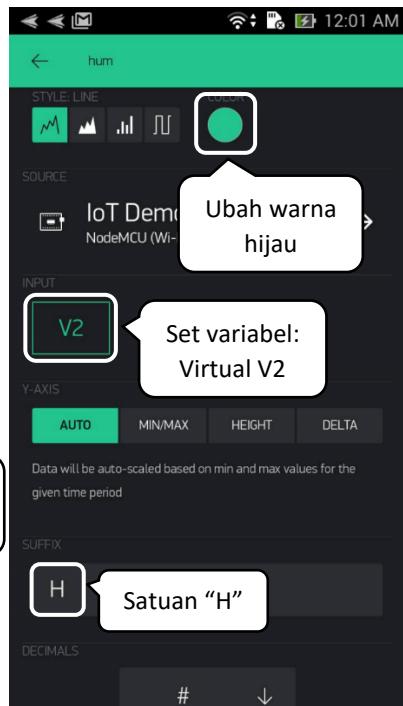
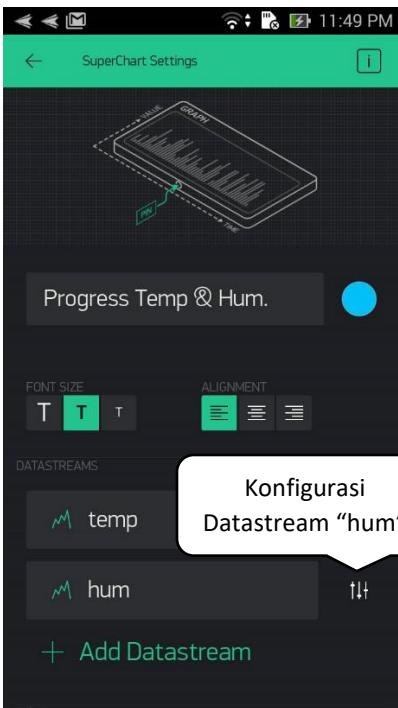
Screenshot 2: SuperChart Settings

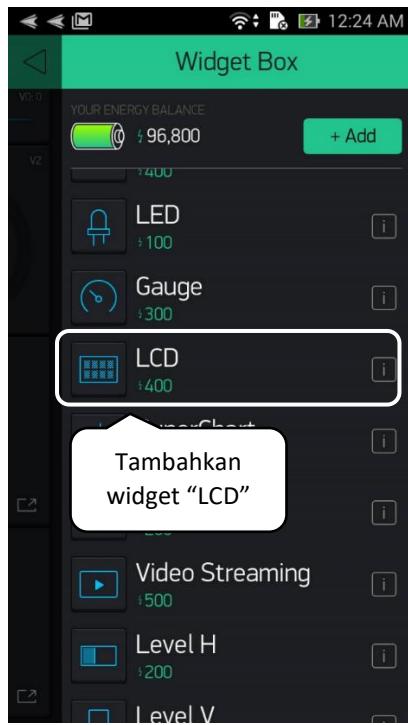
The "SuperChart" component is selected. A green callout bubble points to the "+ Add Datastream" button with the text: "Klik 'Add Datastream'" (Click 'Add Datastream').

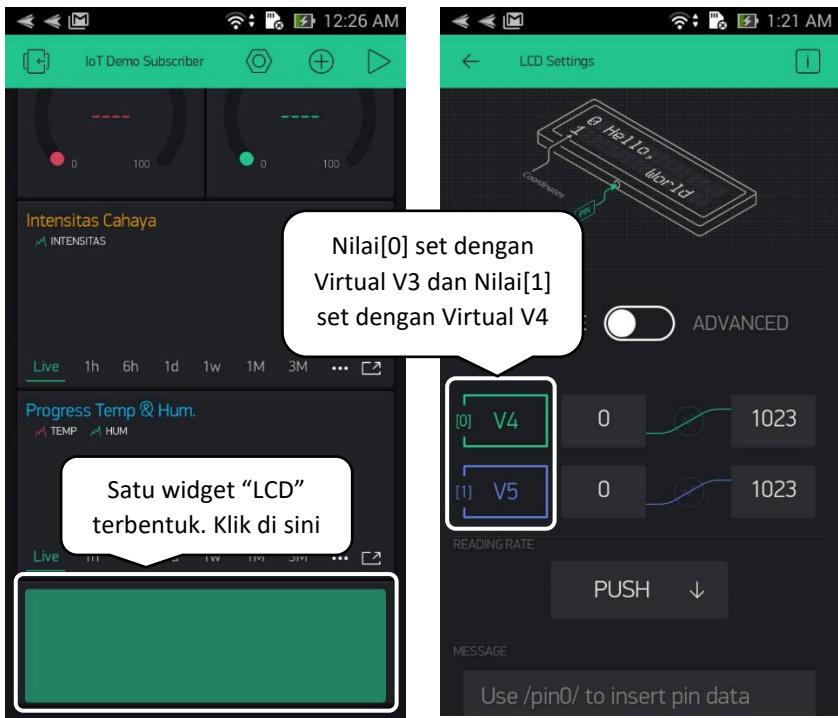
Screenshot 3: SuperChart Datastream Configuration

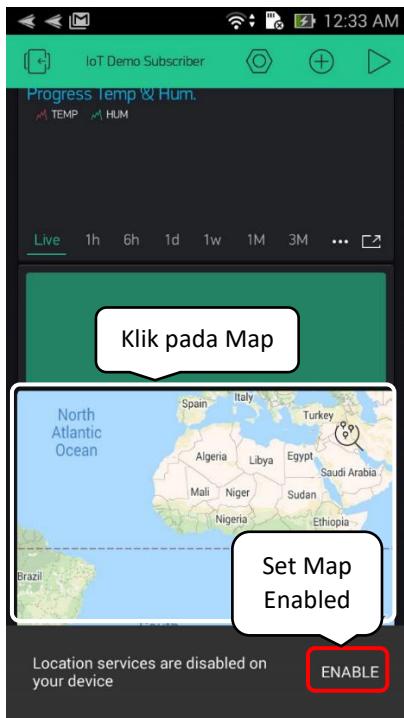
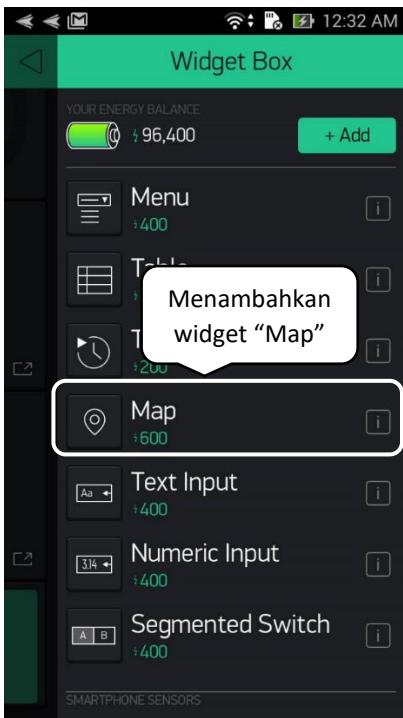
The "Intensitas" datastream is being configured. A green callout bubble points to the chart icon with the text: "Tipe Chart" (Chart Type). Another green callout bubble points to the "V3" input field with the text: "Variabel Virtual: V3". A third green callout bubble points to the "Intensitas" datastream entry with the text: "Atur Datastream Intensitas" (Configure Datastream Intensitas).

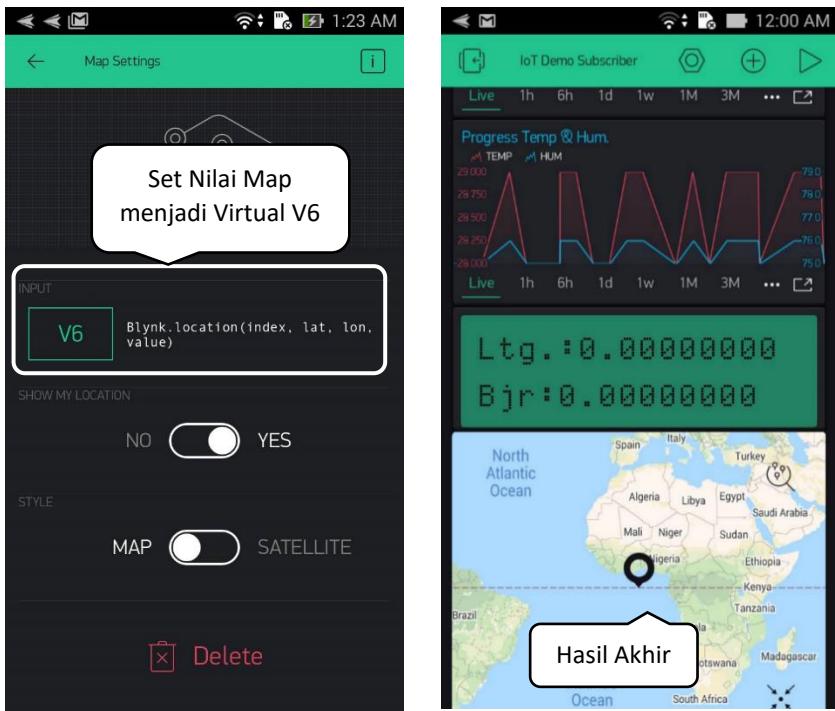












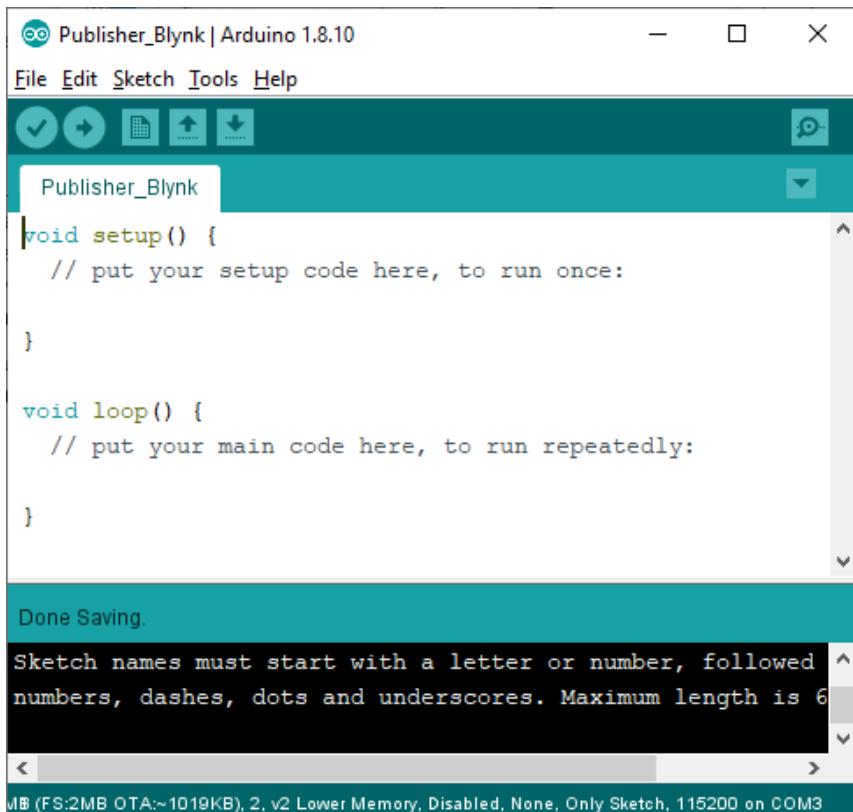
Sampai di sini kita berhasil merancang antarmuka aplikasi client Blynk serta menentukan variabel-nya.

Selanjutnya, kita akan memprogram microcontroller NodeMCU yang telah dibahas pada bagian selanjutnya.

6.4 Menghubungkan Smart Device NodeMCU Ke Blynk

Transport protocol yang digunakan pada Blynk pada dasarnya menggunakan MQTT. Namun blynk memberi kemudahan bagi pengembang dengan membungkus segala kerumitan komunikasi MQTT dalam sebuah library. Sehingga kita cukup meng-*include*-kan library tersebut ke dalam kode program Microcontroller NodeMCU, programer cukup menggunakan fungsi-fungsi yang tersedia sesuai aturan main yang ditetapkan oleh Blynk saat mengirim dan menerima data.

Untuk memprogram smart device yang terhubung dengan blynk, buatlah program baru melalui IDE Arduino dari menu **File → New** dan simpan dengan nama “**Publisher Blynk**”.



The screenshot shows the Arduino IDE interface with the title bar "Publisher_Blynk | Arduino 1.8.10". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, download, and other functions. The main code area contains the following code:

```
void setup() {
    // put your setup code here, to run once:

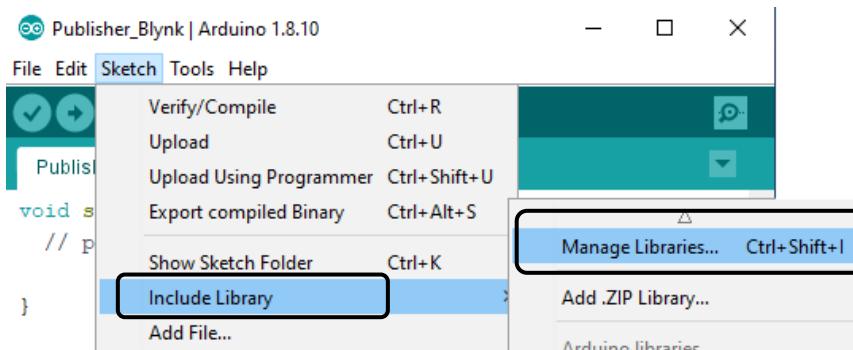
}

void loop() {
    // put your main code here, to run repeatedly:

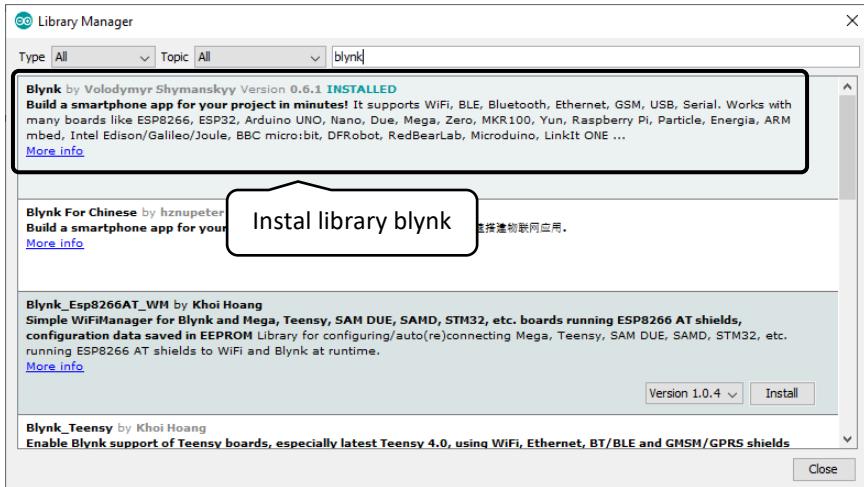
}
```

A message "Done Saving." is displayed in the status bar at the bottom. A warning message "Sketch names must start with a letter or number, followed by numbers, dashes, dots and underscores. Maximum length is 64" is shown in a black box. The status bar also indicates "FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3".

Tambahkan library blynk secara online dari menu **Sketch → Include Library → Managed libraries...**.



Tuliskan "Blynk" pada kolom pencarian library. Berdasarkan daftar library yang tampil, instal library **"Blynk by volodymyr..."**.



Selanjutnya tuliskan kode program berikut ini:

Kode Program

```
#define BLYNK_PRINT Serial

#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <SimpleTimer.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SimpleDHT.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

#define pinServo 13
#define TX_GPS 14
#define RX_GPS 12
#define pinLedMerah 16      //D0 pin led merah
#define pinLedKuning 0       //D3 pin led kuning
#define pinLedHijau 2        //D4 pin led hijau
#define pinDHT 10           // SD3 pin signal sensor DHT

Servo myservo;
int degServo = 0;
```

Kode Program

```
SimpleTimer timerDHT;

// Cek di email konfirmasi saat membuat akun
// Sesuaikan dengan token yang di-generate oleh
blynk
char auth[] = "NFYIBY4Pq-ttmL9HA0tX-
9GBq7c6X8o";
// SSID, silahkan disesuaikan
char ssid[] = "Fanny 2004";
// Password AP, silahkan disesuaikan
char pass[] = "fanny_200504";

SoftwareSerial UART_GPS(RX_GPS, TX_GPS);
TinyGPSPlus gps;
WidgetMap myMap(V4);

// Instan objek sensor dht11
SimpleDHT11 dht11(pinDHT);

// Instan objek LCD I2C
LiquidCrystal_I2C lcd(0x27, 16, 2);

//membuat karakter derajat custom
//https://maxpromer.github.io/LCD-Character-
Creator/
byte derajat[] = {
    B01110,
    B10001,
    B10001,
    B10001,
    B01110,
    B00000,
    B00000,
    B00000
};

void setup() {
    myservo.attach(pinServo);
    myservo.write(degServo);
```

Kode Program

```
Serial.begin(9600);
UART_GPS.begin(9600); //inisialisasi GPS

// Sesuaikan alamat IP Raspberry Pi 3
// dimana Blynk Server di-install
Blynk.begin(auth, ssid, pass, "192.168.1.4",
8080);

pinMode(pinLedMerah, OUTPUT);
pinMode(pinLedKuning, OUTPUT);
pinMode(pinLedHijau, OUTPUT);
pinMode(pinDHT, INPUT);

KonfigurasiLCD();

timerDHT.setInterval(2000, KelembabanSuhu);
timerGPS.setInterval(2000, Lokasi_gps);
}

void loop() {
    Blynk.run();
    timerDHT.run();
    smartDelay(1000);
    Lokasi_gps();
}

BLYNK_WRITE(V5)
{
    degServo = param.asInt();
    myservo.write(degServo);
}

void KonfigurasiLCD() {
    lcd.begin(); // Inisialisasi LCD
    lcd.createChar(0, derajat);
    lcd.backlight(); // Menghidupkan backlight
    lcd.setCursor(0, 0);
    lcd.print("Welcome Blynk");
    lcd.setCursor(0, 1);
    lcd.print("Hello World!");
}
```

Kode Program

```
delay(3000);
lcd.clear();
}

void KelembabanSuhu() {
    byte temperature;
    byte humidity;
    int err = SimpleDHTerrSuccess;

    if ((err = dht11.read(&temperature,
&humidity, NULL)) != SimpleDHTerrSuccess)
    {
        Serial.print("Pembacaan DHT11 gagal,
err=");
        Serial.println(err);
        delay(1000);
        return;
    }

    Serial.print("Sample OK: ");
    Serial.print((int)temperature);
    Serial.print(" *C, ");
    Serial.print((int)humidity);
    Serial.println(" H");
    Serial.println("Derajat Servo: " +
String(degServo));

    Blynk.virtualWrite(V0, temperature);
    Blynk.virtualWrite(V1, humidity);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Tem:" + String((int)temperature));
    lcd.write(0);
    lcd.print("C ");
    lcd.print("Hum:" + String((int)humidity) +
"H");
    lcd.setCursor(0, 1);
    lcd.print("Servo Deg:" + String(degServo));
    lcd.write(0);
}
```

Kode Program

```
void Lokasi_gps() {
    Serial.println("Lat: " +
String(gps.location.lat()) + " - Lng: " +
gps.location.lng());
    Blynk.virtualWrite(V2, "Lint.:" +
String(gps.location.lat(), 6));
    Blynk.virtualWrite(V3, "Bujur:" +
String(gps.location.lng(), 6));
    myMap.location(2, gps.location.lat(),
gps.location.lng(), "Lokasi Saya");
}

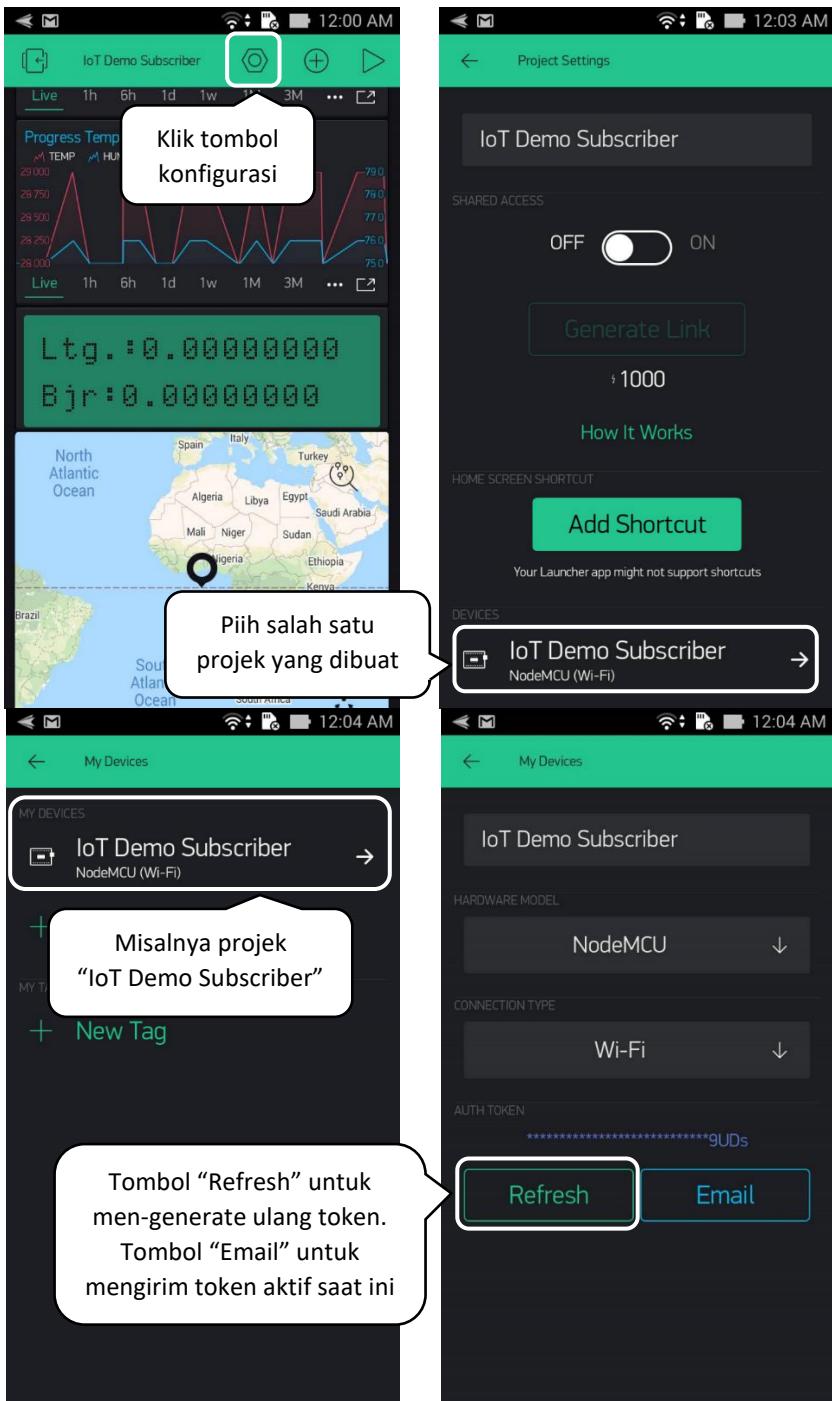
static void smartDelay(unsigned long ms)
{
    unsigned long start = millis();
    do
    {
        while (UART_GPS.available())
            gps.encode(UART_GPS.read());
    } while (millis() - start < ms);
}
```

Perlu diperhatikan bahwa token

`auth[] = "NFYIBY4Pq-ttmL9HA0tX-9GEBq7c6X8o"`

adalah kode otorisasi hak akses yang di-generate oleh Blynk. Setiap projek baru terbentuk, Blynk otomatis akan membangkitkan token masing-masing projek. Token akan dikirim ke email pemilik akun. Dari sini Anda dapat melakukan copy-paste ke program NodeMCU seperti tampak di atas.

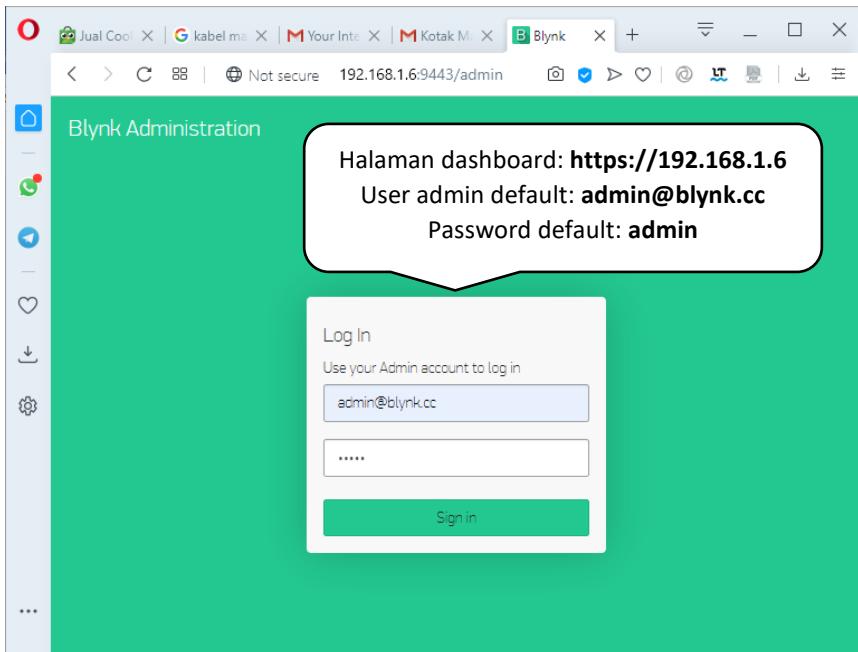
Jika karena suatu hal, Anda tidak menerima email konfirmasi yang berisi kode hash otorisasi tersebut maka Anda dapat mencarinya di aplikasi client Blynk Android. Berikut ini caranya:



Dari halaman tersebut Anda juga bisa melakukan penyegaran token autentikasi untuk memperoleh kode autentikasi baru.

Cara lain, Anda bisa meminta administrator Server IoT Blynk untuk menunjukkan kode autentikasi milik Anda dari halaman Dashboard Admin, khususnya pada bagian halaman user list.

Silahkan buka halaman berikut <https://192.168.1.6:9443/admin/> (Alamat IP tidak harus sama dengan tulisan dalam buku ini. Sesuaikan dengan alamat IP Raspberry Pi 3 Anda dimana Server IoT Blynk di-install).



The screenshot shows the Blynk Administration interface with the URL 192.168.1.6:9443/admin#/users/list. On the left sidebar, there are several icons: a house (Home), a person (Users), a chart (Stats), a gear (Config), and a downward arrow (Downloads). The main area is titled "Users list" and contains a table with two rows of user information:

Email	AppName	# Of Projects	LastModifiedTs
doditsuprianto@gmail.com	Blynk	1	2020-04-01 10:59:40
admin@blynk.cc	Blynk	0	2020-03-25 23:05:49

A callout bubble labeled "Pilih menu ‘Users’" points to the person icon in the sidebar. Another callout bubble labeled "Pilih user" points to the first row of the table.

The screenshot shows the Blynk Administration interface with the URL 192.168.1.6:9443/admin#/users/edit/doditsu. The left sidebar is identical to the previous screenshot. The main area is titled "edit" and shows a form for a device configuration:

value	<input type="text"/>
Devices	
Id	<input type="text" value="0"/>
Name	<input type="text" value="IoT Demo Subscriber"/>
BoardType	<input type="text" value="NodeMCU"/> <input type="button" value="x"/>
Token	-puC-EeMsky53wMJ81izzleMB_Cs9UDs
LastLoggedIn	2020-04-01 10:59:40
Connectivity	WI_FI

A callout bubble labeled "Token autentifikasi yang harus dicantumkan pada program NodeMCU" points to the "Token" field. Another callout bubble labeled "Add new widgets" points to the "value" input field.

Sekarang perhatikan hasil program pada Serial Monitor ketika proses writing/flash ke smart device NodeMCU selesai.

Publisher_Blynk | Arduino 1.8.10

File Edit Sketch

Proses writing program ke chipset NodeMCU

```
#define BLYNK_PRINT Serial

// memanggil library
#include <SimpleDHT.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

// library berkaitan dengan WIFI dan Blynk
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// membuat ketetapan pin input output
#define pinLEDMerah 12 // pin LED Merah
#define pinLEDKuning 14 // pin LED Kuning
#define pinLEDHijau 2 // pin LED Hijau
#define pinAnalogLDR A0 // pin Analog LDR
#define pinDigitalLDR 0 // pin Digital LDR
<
```

Uploading...

Compressed 298352 bytes to 215648...

Writing at 0x00000000... (7 %)

Serial ciphers (most compatible), 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

COM3

LDR tidak terkena cahaya
Intensitas Cahaya : 741
Lat: -7.96802282 - Lng: 112.67160034
Suhu: 27

Kelembaban: 82

LDR tidak terkena cahaya
Intensitas Cahaya : 735
Lat: -7.96802235 - Lng: 112.67160034
Suhu: 27

Kelembaban: 82

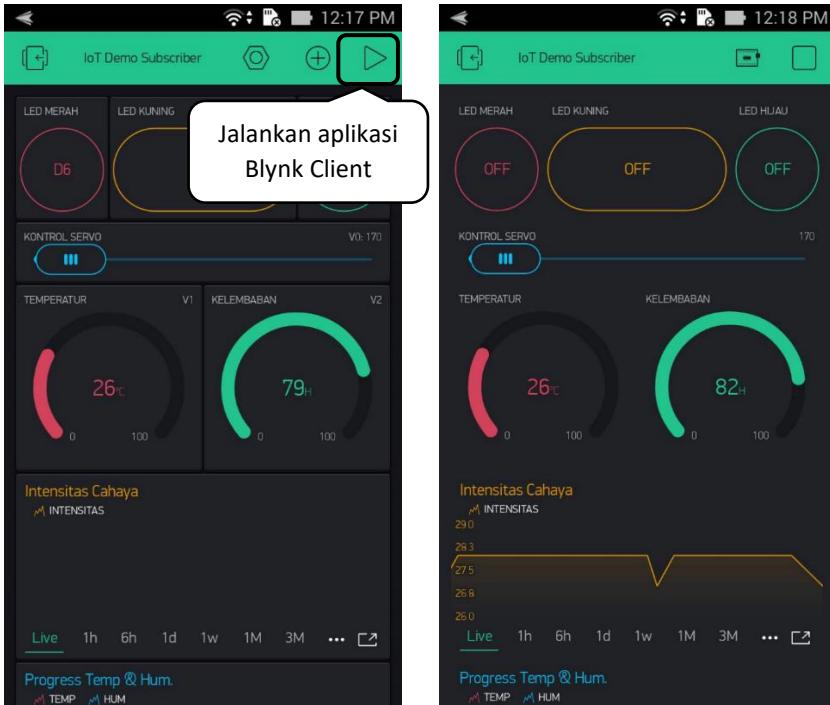
LDR tidak terkena cahaya
Intensitas Cahaya : 736
Lat: -7.96802282 - Lng: 112.67160034

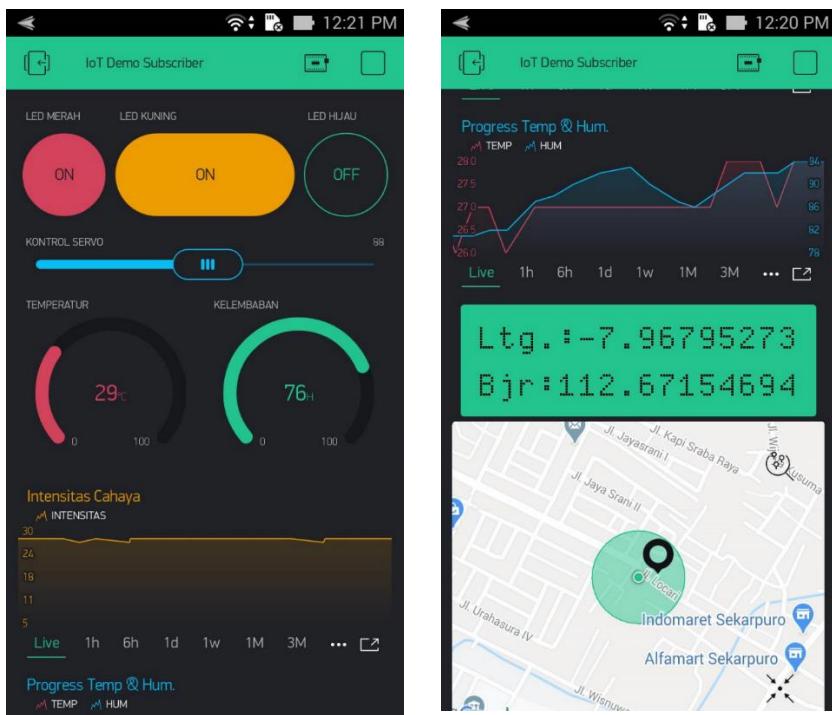
Autoscroll Show timestamp Both NL & CR 9600 baud Clear output

Hasil program secara lokal

Bila tidak ada kesalahan, selanjutnya lakukan pengujian Smart Device NodeMCU untuk dikomunikasikan dengan Server IoT Blynk.

Berikut ini beberapa tampilan aplikasi client Blynk pada platform Android yang telah kita bangun sebelumnya:

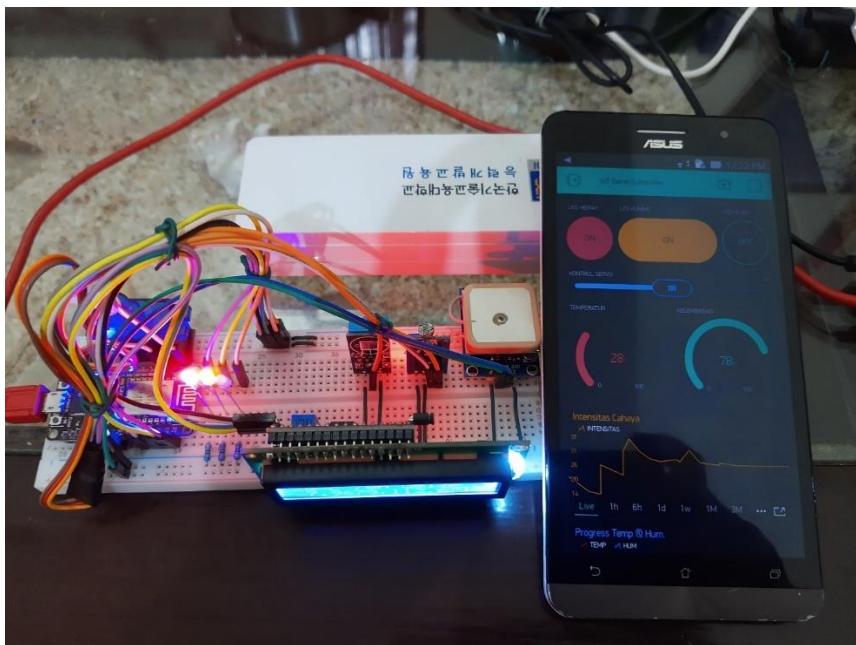


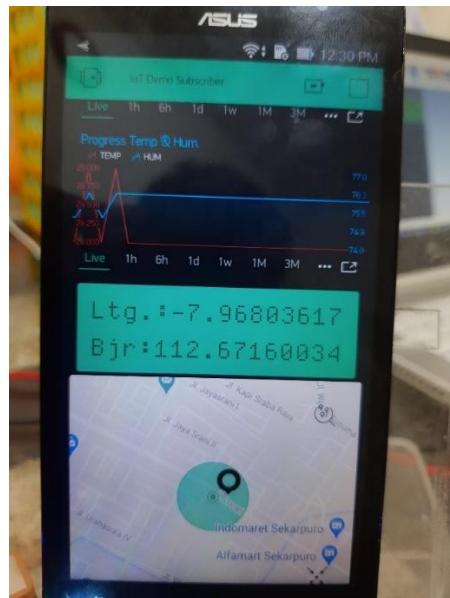
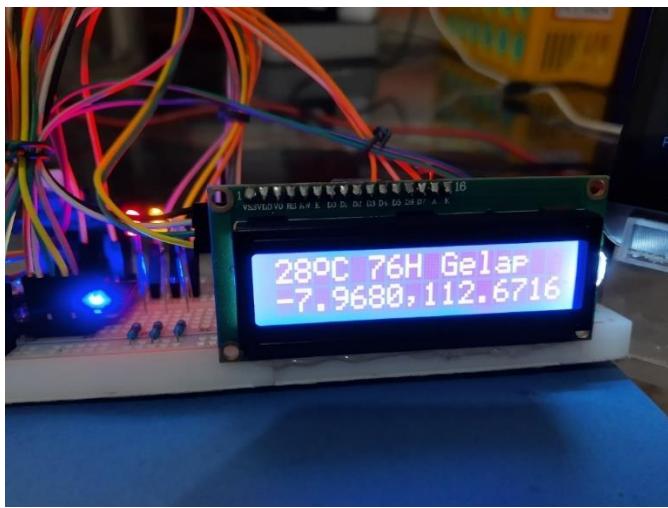


Jika di sisi smart device NodeMCU dan Blynk berhasil maka Anda dapat melakukan hal berikut:

- Menghidupkan dan mematikan 3 LED pada smart device NodeMCU menggunakan widget Button blynk.
- Memutar motor servo mulai 0° sampai 180° menggunakan widget slider.
- Memonitor suhu dan kelembaban menggunakan widget Gauge dan Superchart.
- Mengetahui koordinat lintang dan bujur dimana NodeMCU berada, berdasarkan data GPS yang dikirim ke widget LCD kemudian menampilkannya ke peta Blynk.

Semua dilakukan secara jarak jauh dan realtime menggunakan arsitektur IoT.





BAB 7

Server IoT Thingsboard

Serupa dengan dengan Blynk, Thingsboard adalah Aplikasi IoT sekaligus Server IoT yang dapat digunakan secara gratis dengan beberapa batasan tentunya. Untuk memperoleh layanan tambahan, Anda harus membayar sejumlah biaya yang akan dibebankan kepada pemilik akun. Smart Devices yang Anda bangun dapat dikomunikasikan ke Cloud IoT Thingsboard. Silahkan kunjungi websitenya di <https://thingsboard.io> untuk mencoba dan mengetahui lebih rinci tentang Cloud IoT Thingsboard.

Pada bagian ini kita akan membangun Server IoT Thingsboard versi komunitas yang gratis diinstal pada komputer lokal kita. Server IoT Thingsboard sedikit berbeda dengan Blynk, bila sebelumnya perangkat Android difungsikan sebagai aplikasi subscriber dan Raspberry Pi 3 sebagai server maka sekarang kita akan menggunakan Thingsboard sebagai Server IoT, yang akan diinstal pada komputer/laptop, sedangkan subscriber-nya menggunakan browser internet. Sistem operasi yang digunakan oleh Thingsboard adalah Linux Ubuntu MATE.

Arsitektur IoT dan topologi jaringan yang digunakan tetap mengacu pada arsitektur dan topologi sebelumnya, termasuk rancangan Smart Device NodeMCU yang digunakan. Perbedaanya mendasarnya adalah:

- Server IoT sebelumnya adalah Blynk, sekarang Thingsboard
- Perangkat Server sebelumnya adalah Raspberry Pi 3, sekarang gPC/Laptop
- Penyesuaian kode program microcontroller-nya, bagaimana cara mengirim dan menerima data dari Smart Device NodeMCU ke Blynk, sekarang ke Thingsboard dan sebaliknya.

Mengapa kita memilih komputer PC atau Laptop sebagai server Thingsboard? Karena sumberdaya (memori, processor dan lain-lain) yang dibutuhkan Thingsboard lebih besar dari Blynk. Meskipun Thingsboard masih memungkinkan untuk diinstal pada

Raspberry Pi 3, namun kinerjanya sangat lambat dan *slow respon*. Oleh karena itu, kita menerapkannya pada komputer PC/Laptop, sekaligus sebagai pengayaan untuk mencoba Server IoT pada platform berbeda.

7.1 Instalasi Sistem Operasi Linux Ubuntu MATE

Anda perlu menyiapkan komputer atau Laptop yang akan difungsikan sebagai server. Kita juga memerlukan file image sistem operasi Linux Ubuntu Mate yang dapat didownload di halaman berikut <https://ubuntu-mate.org/download/>.

Download

Choose your architecture

64-bit

Ideal for computers with:



- More than 3 GB of RAM.
- 64-bit capable Intel and AMD processors.
- UEFI PCs booting in CSM mode.
- Modern Intel-based Apple Macs

32-bit

Ideal for computers with:



- Less than 2 GB of RAM.
- Intel and AMD processors.
- Aging PCs with low-RAM resources.
- Older Intel-based Apple Macintosh systems.

Raspberry Pi (recommended)



For arch32 (ARMv7) computers, like:

- Raspberry Pi Model B 2
- Raspberry Pi Model B 3
- Raspberry Pi Model B 3+

Pilih salah satu antara
64 bit atau 32 bit

(experimental)

For arch64 (ARMv8) computers, like:

- Raspberry Pi Model B 3
- Raspberry Pi Model B 3+

Download

Which release would you like?

for a 64-bit system



18.04.4 LTS (Bionic)

Recommended for stability and mission critical systems.

Supported until April 2021

Pilih Ubuntu
MATE versi Bionic

Choose a different architecture



Setelah mendapatkan file image Ubuntu Mate “**ubuntu-mate-18.04.4-desktop-amd64.iso**” (nama file tidak harus sama, karena tergantung versi yang di-release), selanjutnya *burning* file image tersebut ke CD atau Flashdisk yang akan digunakan sebagai installer Ubuntu Mate ke komputer PC.

7.1.1 Burning Ubuntu Mate Ke CD

Download aplikasi burning CDBurningXP dari halaman <https://cdburnerxp.se/en/download>, atau gunakan aplikasi burning lainnya juga tidak masalah. Jalankan file **cdbxp_setup_4.5.8.7128_minimal.exe** hasil download sebelumnya, instal aplikasi terebut

 Setup - CDBurnerXP 4.5.8.7128

License Agreement

Please read the following important information before continuing.



Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

CDBurnerXP License

By using, copying or distributing CDBurnerXP and accompanying documentation (the program's object code and documentation are collectively referred to as „the software”), you indicate your acceptance of the following license terms:

1. Usage

I accept the agreement I do not accept the agreement

Print

Next > **Cancel**

 Setup - CDBurnerXP 4.5.8.7128

Select Destination Location

Where should CDBurnerXP be installed?

 Setup will install CDBurnerXP into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files (x86)\CDBurnerXP **Browse...**

At least 12.9 MB of free disk space is required.

< Back **Next >** **Cancel**

Setup - CDBurnerXP 4.5.8.7128

Select Components

Which components should be installed?



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Compact installation	
<input checked="" type="checkbox"/> Program Files	11.2 MB
<input type="checkbox"/> Languages	
<input type="checkbox"/> Arabic (Jordan)	0.2 MB
<input type="checkbox"/> Bulgarian (Bulgaria)	0.2 MB
<input type="checkbox"/> Czech (Czech Republic)	0.2 MB
<input type="checkbox"/> Croatian (Croatia)	0.2 MB
<input type="checkbox"/> Catalan (Catalonia)	0.2 MB
<input type="checkbox"/> Danish (Denmark)	0.2 MB
<input type="checkbox"/> Dutch (Netherlands)	0.2 MB

Current selection requires at least 12.9 MB of disk space.

< Back Next > Cancel

Setup - CDBurnerXP 4.5.8.7128

Select Additional Tasks

Which additional tasks should be performed?



Select the additional tasks you would like Setup to perform while installing CDBurnerXP, then click Next.

Create a desktop shortcut
 For all users
 For the current user only

Create a Quick Launch shortcut

Add to "Send to" menu

Other tasks:

Open Data (.dpx) and Audio (.axp) compilations with CDBurnerXP

Associate ISO (.iso) files with CDBurnerXP

< Back Install Cancel

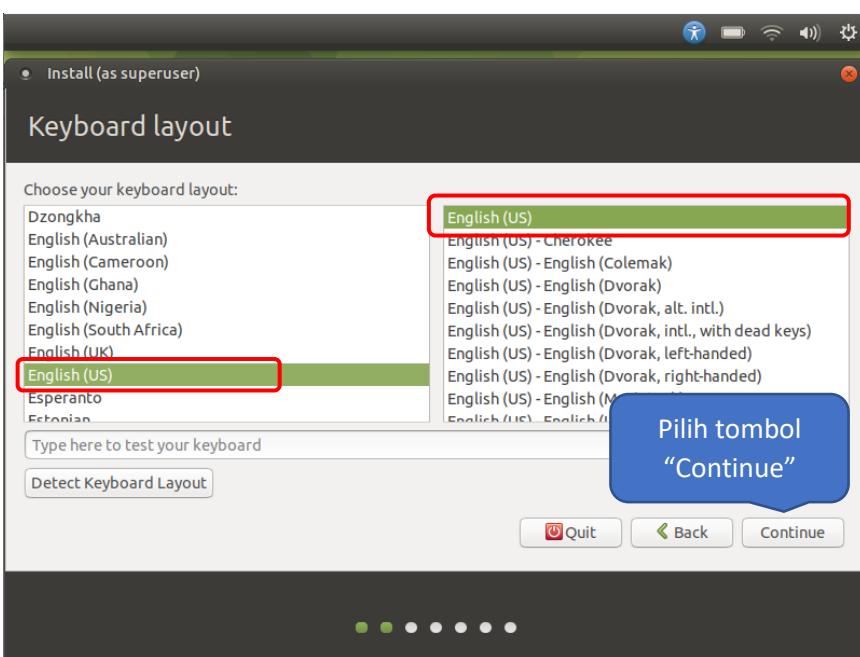
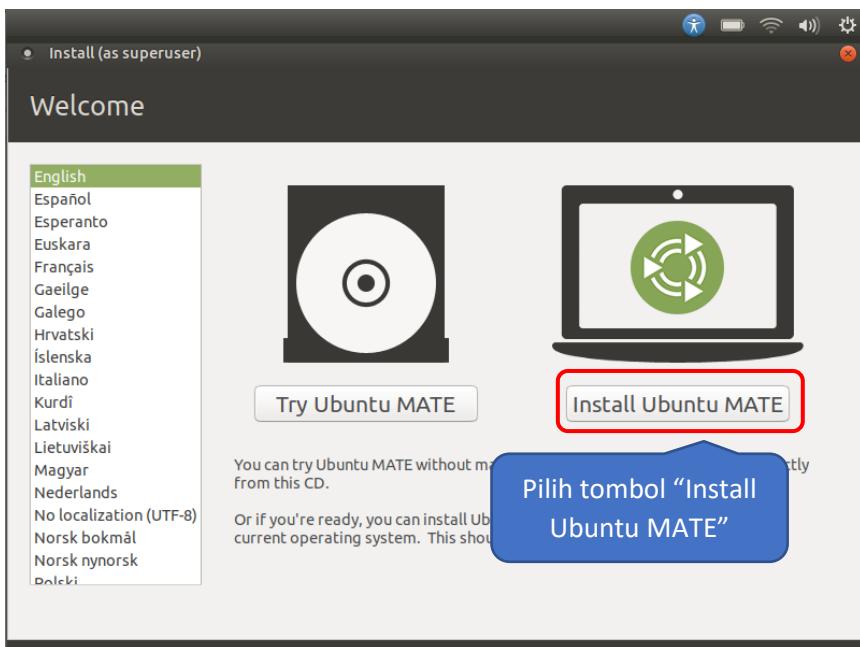


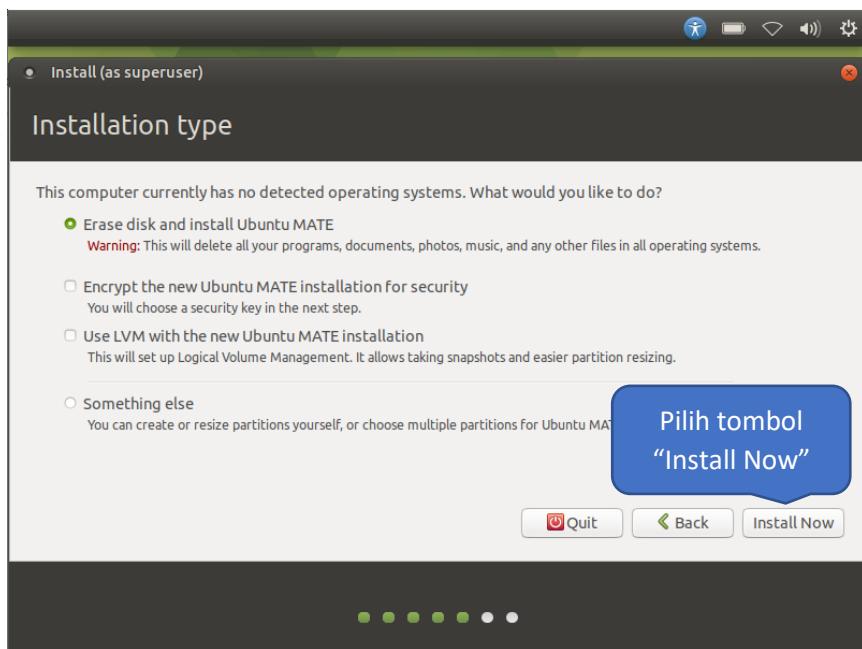
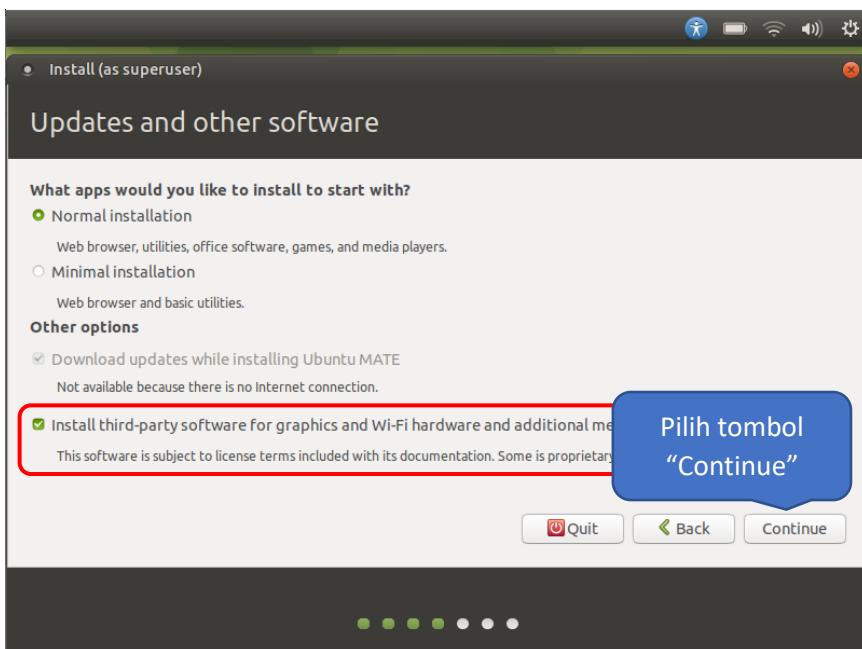
7.1.2 Burning Ubuntu Mate ke Flashdisk

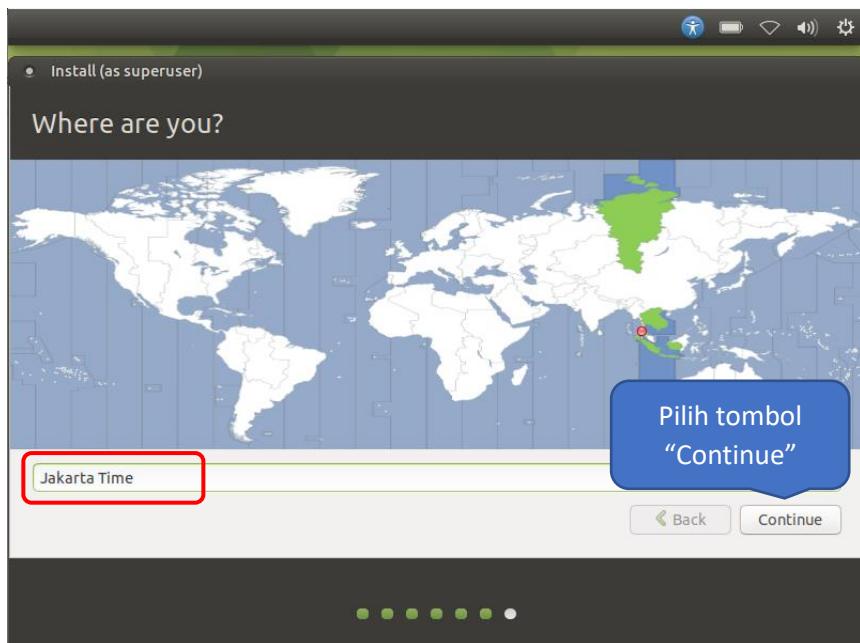
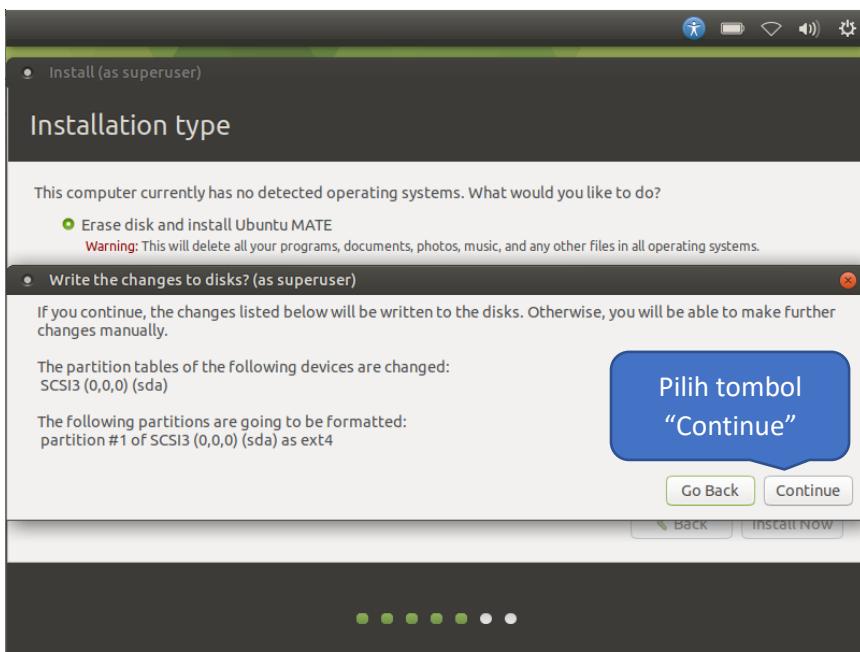
Belum selesai

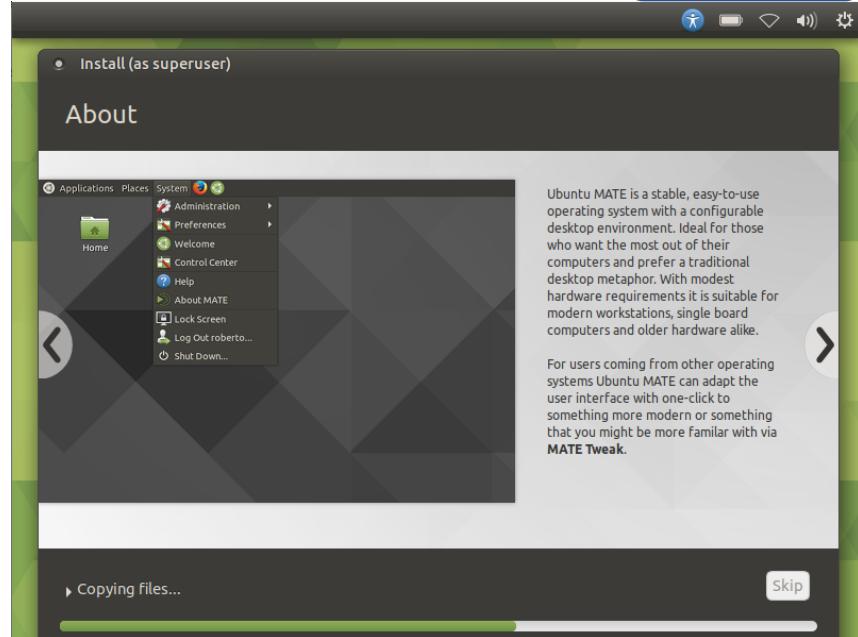
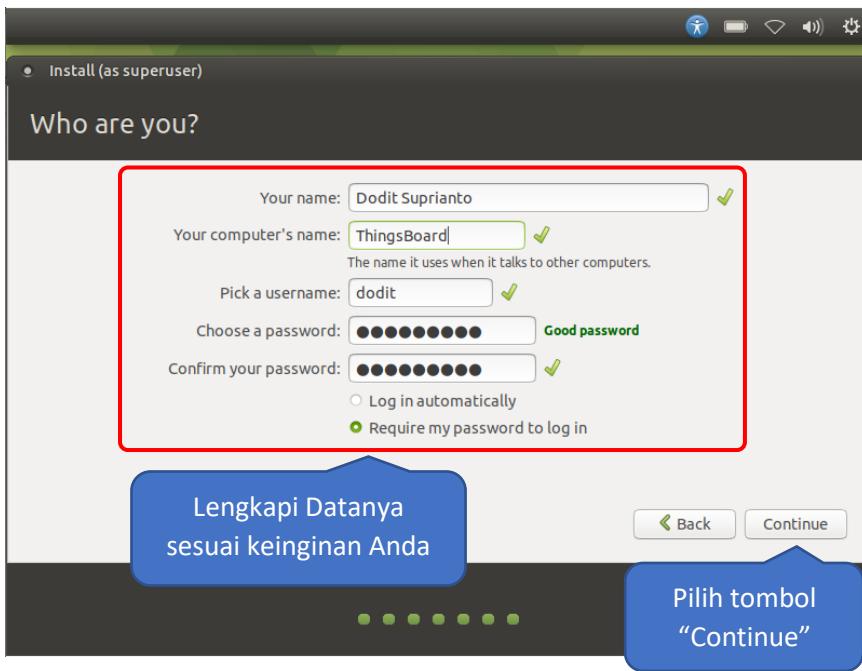
7.2 Instalasi Linux Ubuntuk Mate ke Komputer PC/Laptop

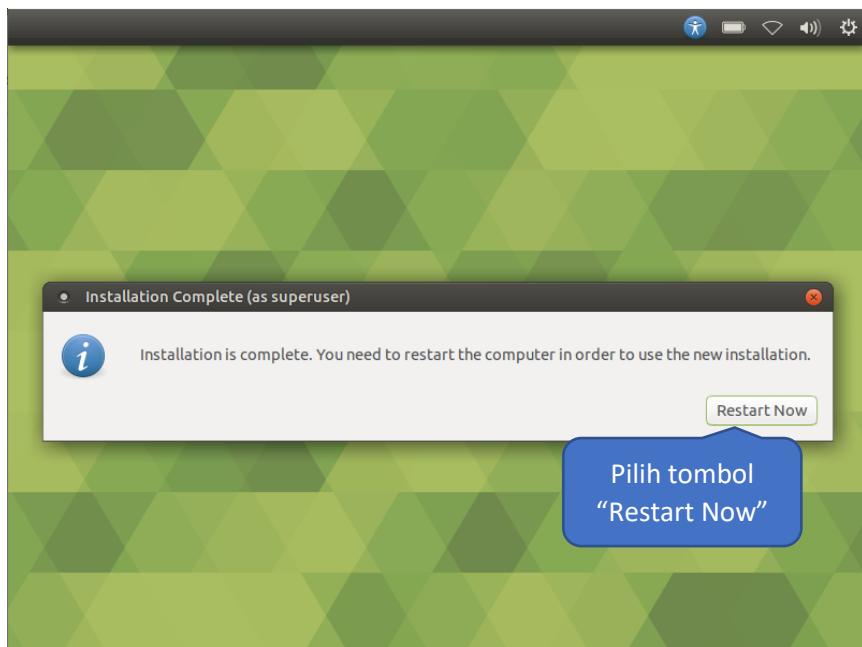
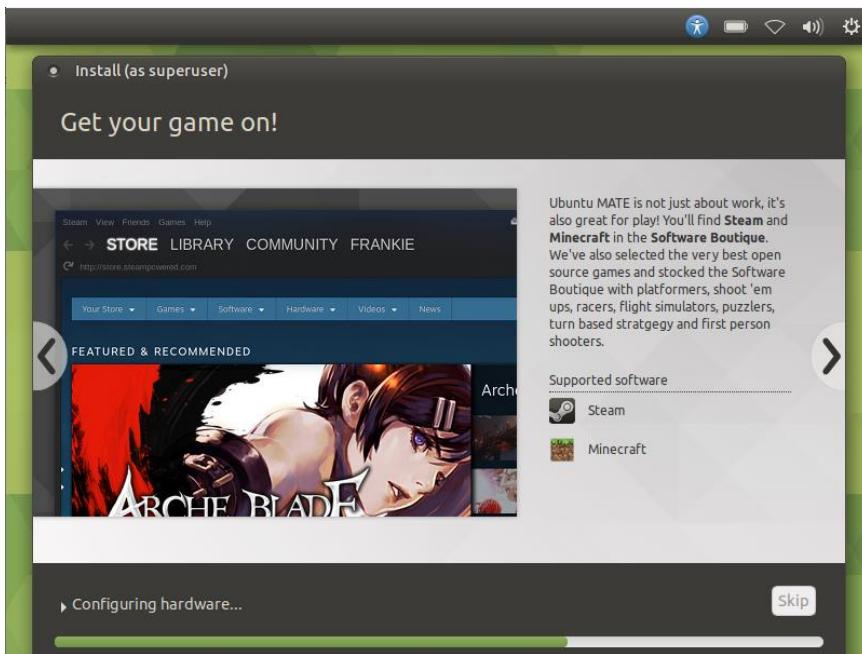
Masukkan keping CD ke dalam CD-ROM atau USB ke socket USB komputer PC/laptop yang berisi master Linux Ubuntu MATE hasil burning. Pastikan bahwa urutan pertama booting komputer adalah CD-ROM atau USB . Anda dapat mengurnya pada BIOS.









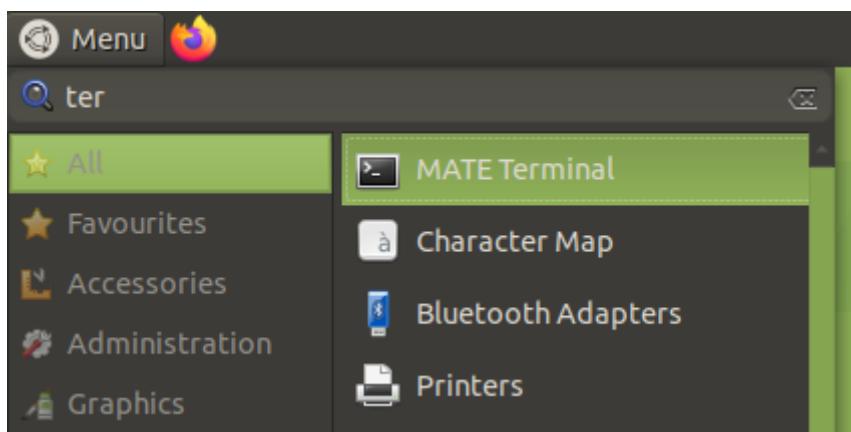


Jangan lupa untuk mengembalikan urutan booting pada BIOS kembali di set ke HDD. Sampai di sini Anda sudah menginstall Linux Ubuntu MATE dengan mudah.

7.3 Install IoT Server Thingsboard Pada Ubuntu Mate

Sebelum mulai menginstal Thingsboard, alangkah baiknya kita menginstall SSH server terlebih dahulu, sehingga kita bisa me-remote Ubuntu Mate saat proses instalasi Thingsboard, meskipun kita juga bisa menginstalnya secara langsung melalui “**MATE Terminal**”. Untuk menginstall SSH server, silahkan ikut langkah berikut ini:

Buka aplikasi MATE Terminal melalui “**Menu**”, kemudian tuliskan “**terminal**” pada kolom pencarian, kemudian klik menu “**MATE Terminal**” atau cara mudahnya menggunakan keyboard shortcut **Ctrl+Alt+T**.



Tuliskan script berikut:

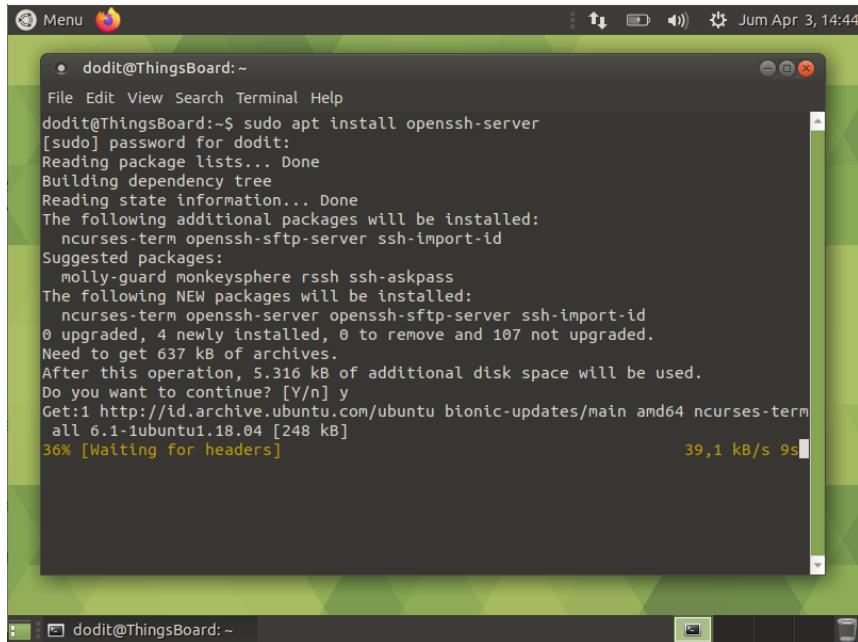
```
sudo apt update  
sudo apt upgrade  
sudo reboot
```

```
● dodit@ThingsBoard:~  
File Edit View Search Terminal Help  
dodit@ThingsBoard:~$ sudo apt update  
[sudo] password for dodit:  
Err:1 http://security.ubuntu.com/ubuntu bionic-security InRelease  
Temporary failure resolving 'security.ubuntu.com'  
Err:2 http://id.archive.ubuntu.com/ubuntu bionic InRelease  
Temporary failure resolving 'id.archive.ubuntu.com'  
Get:3 http://id.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]  
Get:4 http://id.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]  
Get:5 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages  
[1.012 kB]  
Get:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages  
[1.061 kB]  
Ign:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages  
Get:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages  
[1.061 kB]  
Ign:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages  
Ign:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages  
Err:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages  
Connection timed out [IP: 91.189.88.142 80]  
Fetched 1.175 kB in 2min 29s (7.872 B/s)  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
107 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
● dodit@ThingsBoard:~  
File Edit View Search Terminal Help  
dodit@ThingsBoard:~$ sudo apt upgrade  
[sudo] password for dodit:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Calculating upgrade... Done  
The following packages were automatically installed and are no longer required:  
  efibootmgr libappstream glib2 libfwup1  
Use 'sudo apt autoremove' to remove them.  
The following NEW packages will be installed:  
  fwupd-signed libxml2b1 linux-headers-5.3.0-45 linux-headers-5.3.0-45-generic  
  linux-image-5.3.0-45-generic linux-modules-5.3.0-45-generic  
  linux-modules-extra-5.3.0-45-generic  
The following packages will be upgraded:  
  apport apport-gtk bluez bluez-cups bluez-obexd bsutils cpp-7 dmidecode  
  fdisk firefox firefox-locale-en fwupd fwupdate fwupdate-signed gcc-7-base  
  gcc-8-base gir1.2-javascriptcoregtk-4.0 gir1.2-webkit2-4.0 grub-common  
  grub-pc grub-pc-bin grub2-common libarchive13 libasound2 libasound2-data  
  libblkid1 libbluetooth3 libcc1-0 libdrm-amdgpu1 libdrm-common libdrm-intel1  
  libdrm-nouveau2 libdrm-radeon1 libdrm2 libegl-mesa0 libegl1-mesa libexif12  
  libexiv2-14 libfdisk1 libfwupd2 libgbm1 libgcc1 libgd3 libgl-mesa-dri  
  libgl1-mesa0 libglapi-mesa libgl2b0-0 libgl2b2-0-bin libglib2.0-data  
  libglx-mesa0 libgomp1 libicu60 libjavascriptcoregtk-4.0-18 libllvm9  
  libmount1 libnss libnss-systemd libpam-systemd libqt5core5a libqt5dbus5  
Click here to hide all windows and show the desktop.
```

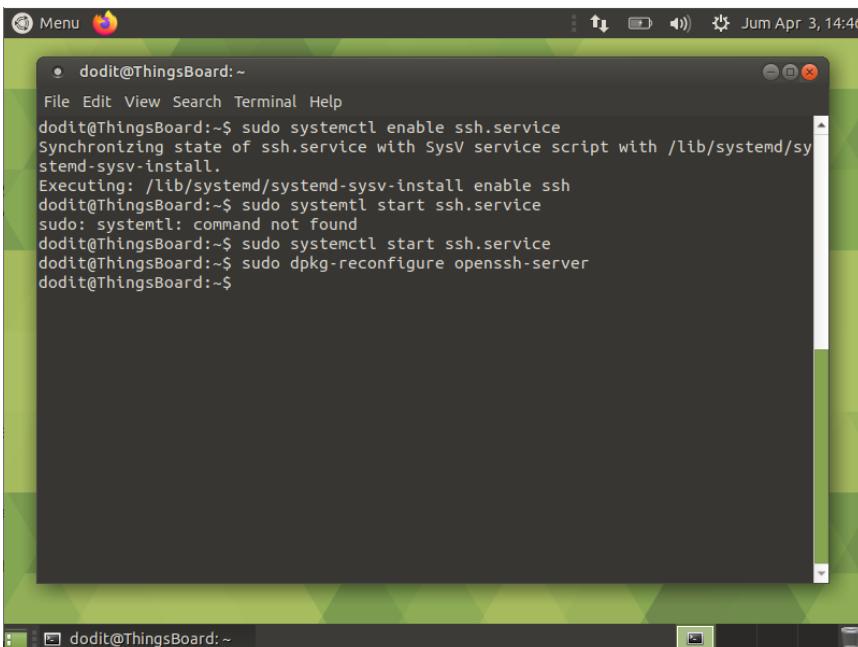
```
sudo apt install openssh-server  
sudo systemctl enable ssh.service
```

```
sudo systemctl start ssh.service  
sudo dpkg-reconfigure openssh-server
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window title is 'dodit@ThingsBoard:~'. The window contains the following text:

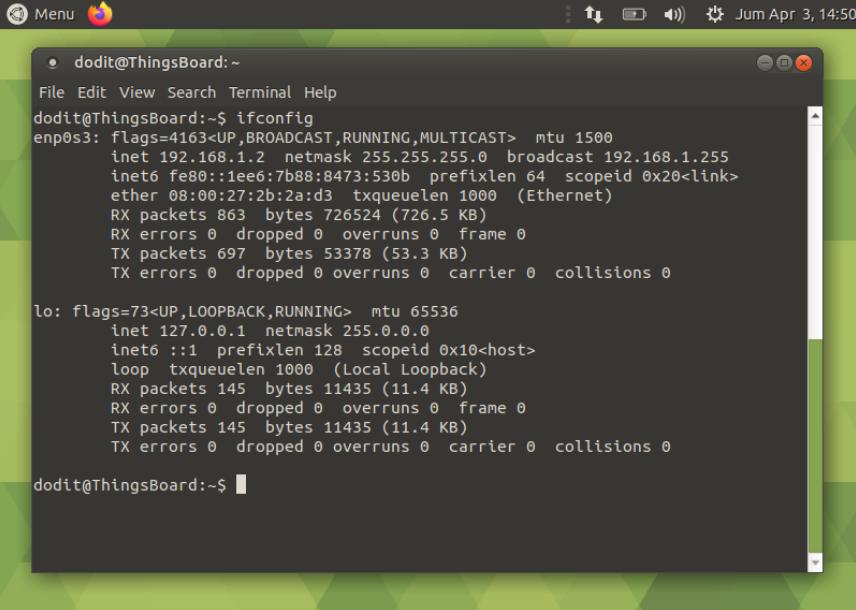
```
File Edit View Search Terminal Help  
dodit@ThingsBoard:~$ sudo apt install openssh-server  
[sudo] password for dodit:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  ncurses-term openssh-sftp-server ssh-import-id  
Suggested packages:  
  molly-guard monkeysphere rssh ssh-askpass  
The following NEW packages will be installed:  
  ncurses-term openssh-server openssh-sftp-server ssh-import-id  
0 upgraded, 4 newly installed, 0 to remove and 107 not upgraded.  
Need to get 637 kB of archives.  
After this operation, 5.316 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://id.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ncurses-term  
  all 6.1-1ubuntu1.18.04 [248 kB]  
36% [Waiting for headers] 39,1 kB/s 9s
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window title is 'dodit@ThingsBoard:~'. The window contains the following text:

```
File Edit View Search Terminal Help  
dodit@ThingsBoard:~$ sudo systemctl enable ssh.service  
Synchronizing state of ssh.service with SysV service script with /lib/systemd/sys  
temd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable ssh  
dodit@ThingsBoard:~$ sudo systemctl start ssh.service  
sudo: systemctl: command not found  
dodit@ThingsBoard:~$ sudo systemctl start ssh.service  
dodit@ThingsBoard:~$ sudo dpkg-reconfigure openssh-server  
dodit@ThingsBoard:~$
```

Cek terlebih dahulu alamat IP Ubuntu Mate dengan perintah ifconfig. Berdasarkan contoh di bawah ini, didapatkan alamat IP 192.168.1.2. Bisa jadi alamat IP yang Anda dapatkan pada komputer Anda akan berbeda. Karena posisi Ubuntu MATE pada jaringan adalah sebagai client/station DHCP maka bisa jadi alamat IP-nya suatu saat akan berubah. Untuk mengatasinya dijadikan IP statis, namun pada buku ini kita tidak membahasnya.



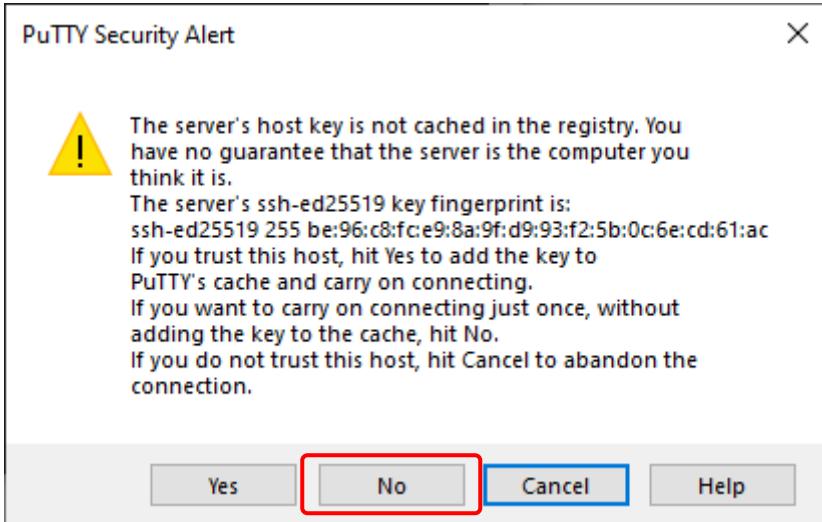
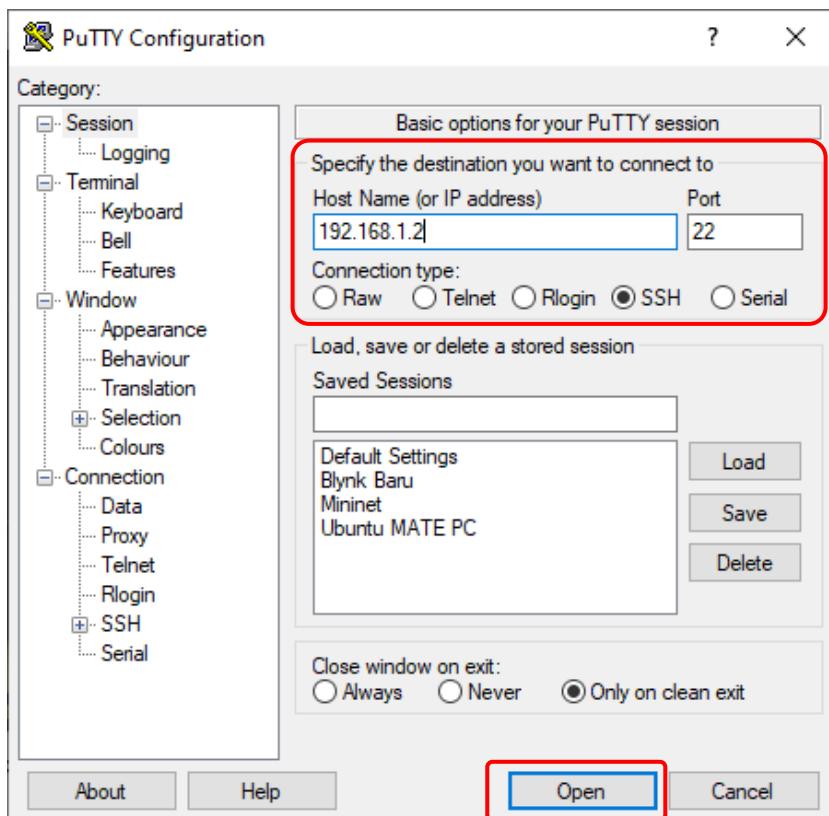
```
dodit@ThingsBoard:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0 broadcast 192.168.1.255
                inet6 fe80::1ee6:7bb8:8473:530b  prefixlen 64  scopeid 0x20<link>
                    ether 08:00:27:2b:2a:d3  txqueuelen 1000  (Ethernet)
                        RX packets 863  bytes 726524 (726.5 KB)
                        RX errors 0  dropped 0  overruns 0  frame 0
                        TX packets 697  bytes 53378 (53.3 KB)
                        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

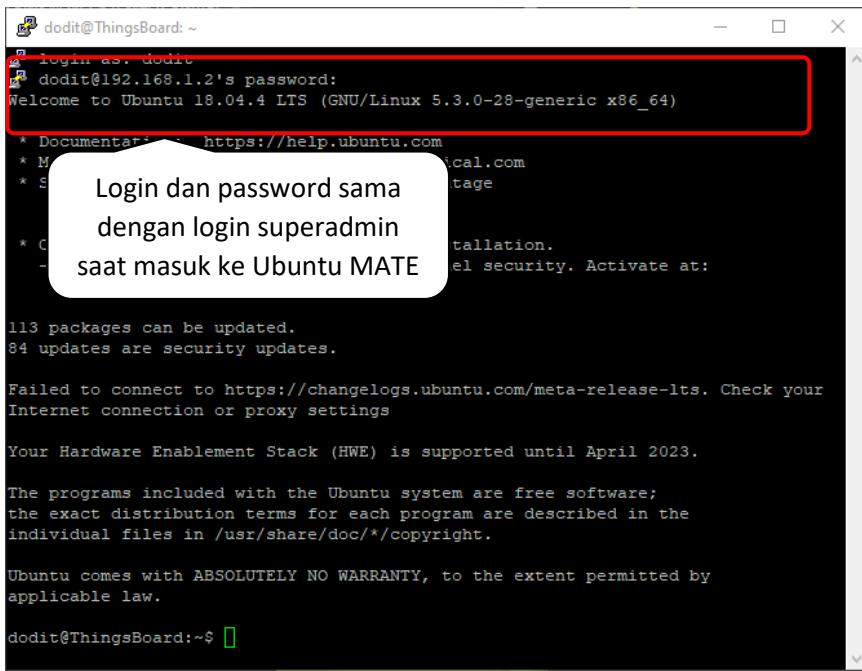
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
                inet6 ::1  prefixlen 128  scopeid 0x10<host>
                    loop  txqueuelen 1000  (Local Loopback)
                        RX packets 145  bytes 11435 (11.4 KB)
                        RX errors 0  dropped 0  overruns 0  frame 0
                        TX packets 145  bytes 11435 (11.4 KB)
                        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

dodit@ThingsBoard:~$
```

Sekarang kita uji apakah server SSH telah aktif dan siap digunakan dengan aplikasi remote Putty.

Buka laptop lain atau perangkat lain yang terhubung dengan jaringan, kemudian buka aplikasi Putty, isikan alamat IP (nama host) dengan 192.168.1.2, nomor port SSH adalah 22, jenis koneksi adalah “SSH”, dan kemudian pilih tombol “Open”.





```
dodit@ThingsBoard: ~
dodit@192.168.1.2's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Mailing lists:   https://lists.ubuntu.com/mailman/listinfo/ubuntu-devel
 * Support:        https://ubuntu.com/advantage
 * Bug tracking:   https://bugs.launchpad.net/ubuntu/+source/linux/+bug
 * Installation:  https://ubuntu.com/installation
 * Security:      https://ubuntu.com/security. Activate at:
113 packages can be updated.
84 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Your Hardware Enablement Stack (HWE) is supported until April 2023.

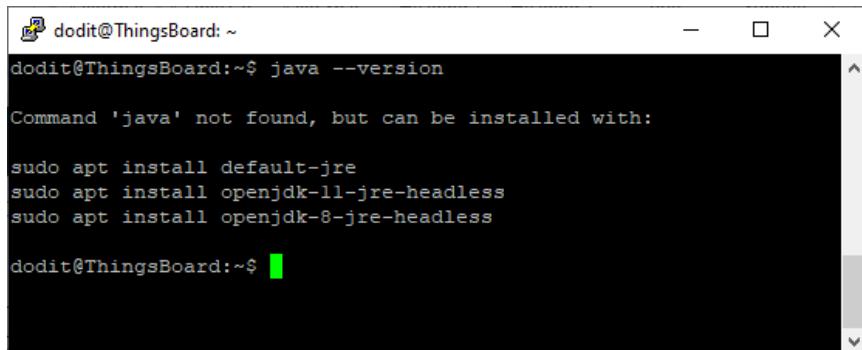
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

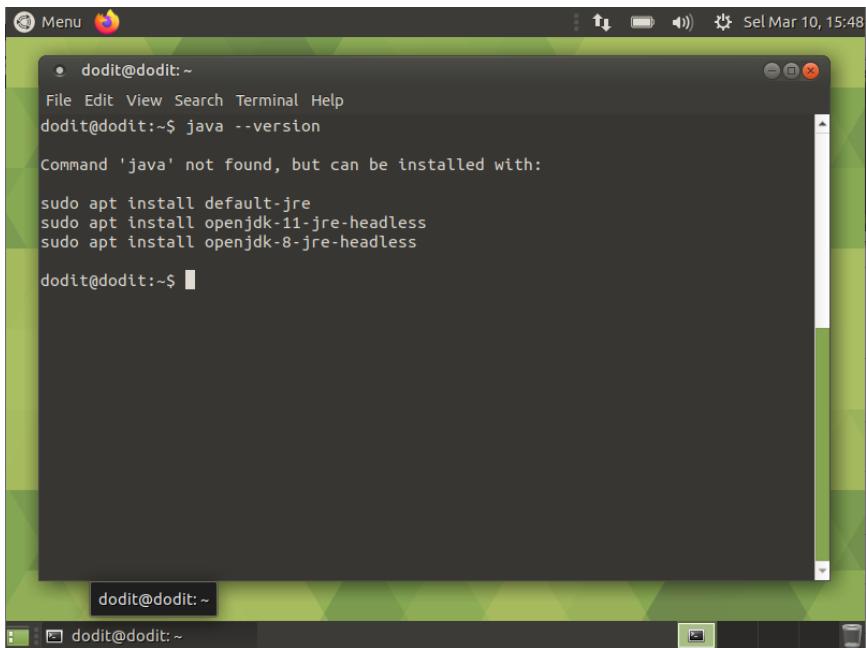
dodit@ThingsBoard:~$
```

Sampai sini kita sudah bisa me-remote Ubuntu MATE dari perangkat lain menggunakan aplikasi Putty.

Mari kita lanjutkan dengan menginstall aplikasi Thingsboard. Karena Thingsboard membutuhkan Java JDK maka kita perlu mengecek apakah Ubuntu MATE telah terinstall dengan Java JDK. Untuk mengetahuinya gunakan perintah `java --version`.



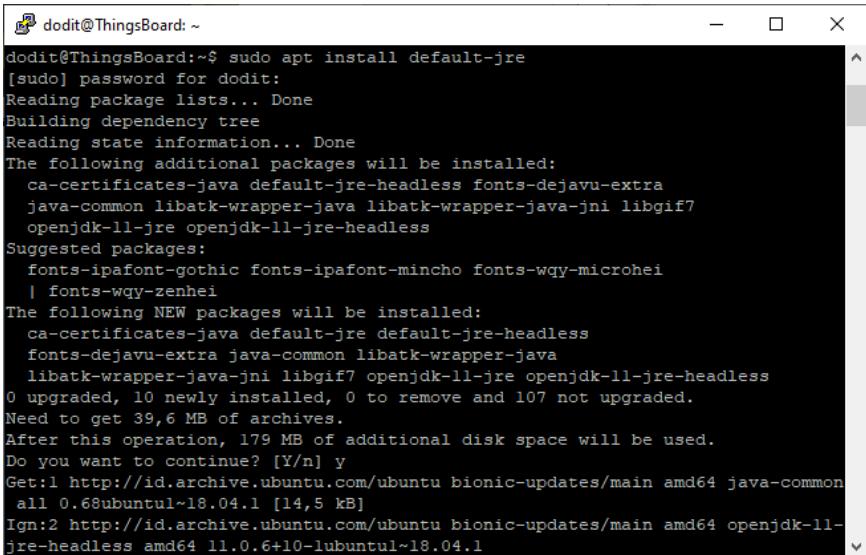
```
dodit@ThingsBoard: ~
dodit@ThingsBoard:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless
dodit@ThingsBoard:~$
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and displays the following text:

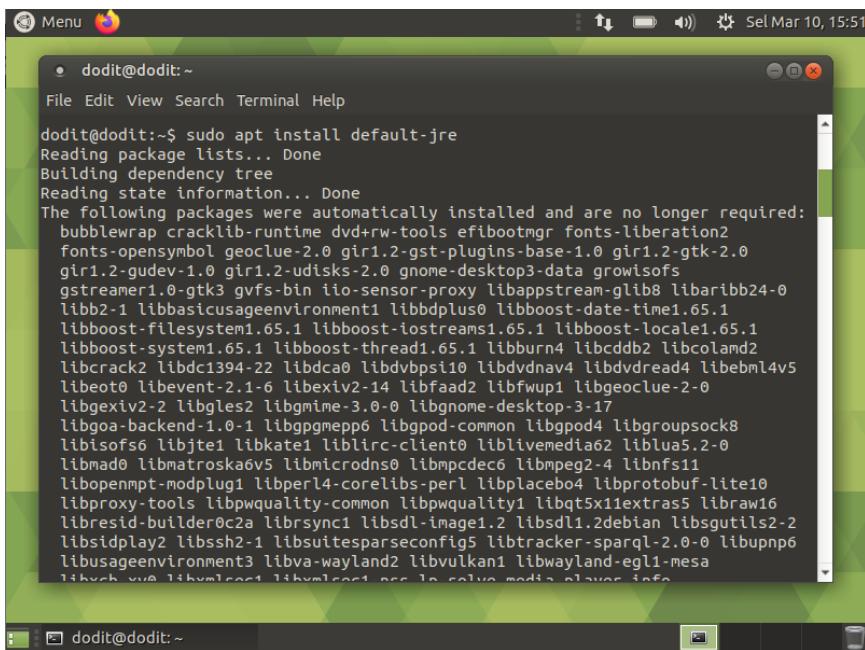
```
dodit@dodit: ~
File Edit View Search Terminal Help
dodit@dodit:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless
dodit@dodit:~$
```

Bila muncul pesan seperti di atas maka kita perlu menginstall java terlebih dahulu. Gunakan perintah: `sudo apt install default-jre`.



A screenshot of a terminal window titled "dodit@ThingsBoard: ~". The terminal displays the output of the command `sudo apt install default-jre`. The output shows the package manager reading lists, building dependency trees, and listing packages to be installed. It also shows suggested packages and new packages to be installed. The user is prompted to continue the operation.

```
dodit@ThingsBoard:~$ sudo apt install default-jre
[sudo] password for dodit:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jre-headless fonts-dejavu-extra
    java-common libatk-wrapper-java libatk-wrapper-java-jni libgif7
      openjdk-11-jre openjdk-11-jre-headless
Suggested packages:
  fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  | fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java default-jre default-jre-headless
    fonts-dejavu-extra java-common libatk-wrapper-java
      libatk-wrapper-java-jni libgif7 openjdk-11-jre openjdk-11-jre-headless
0 upgraded, 10 newly installed, 0 to remove and 107 not upgraded.
Need to get 39,6 MB of archives.
After this operation, 179 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://id.archive.ubuntu.com/ubuntu bionic-updates/main amd64 java-common all 0.68ubuntul~18.04.1 [14,5 kB]
Ign:2 http://id.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openjdk-11-jre-headless amd64 11.0.6+10-lubuntul~18.04.1
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and displays the following command and its output:

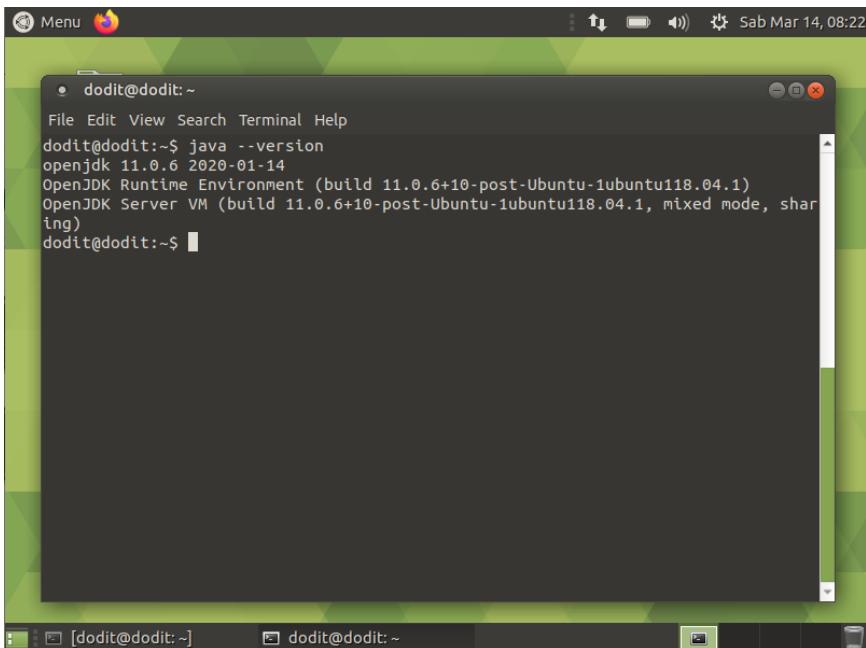
```
dodit@dodit:~$ sudo apt install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bubblewrap cracklib-runtime dvd+rw-tools efibootmgr fonts-liberation2
  fonts-opensymbol geoclue-2.0 gir1.2-gst-plugins-base-1.0 gir1.2-gtk-2.0
  gir1.2-gudev-1.0 gir1.2-udisks-2.0 gnome-desktop3-data growisofs
  gstreamer1.0-gtk3 gvfs-bin iio-sensor-proxy libappstream-glib8 libaribbb24-0
  libbb2-1 libbasicusageenvironment1 libbbplus0 libboost-date-time1.65.1
  libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1
  libboost-system1.65.1 libboost-thread1.65.1 libburn4 libcdrom2 libcolamd2
  libcrack2 libdc1394-22 libdca0 libdvbpsi0 libdvnav4 libdvread4 libebml4v5
  libeot0 libevent-2.1-6 libexiv2-14 libfaad2 libfwup1 libgeoclue-2-0
  libgexiv2-2 libgles2 libgmime-3.0-0 libgnome-desktop-3-17
  libgoa-backend-1.0-1 libgpmep0 libgpod-common libgroupsock4
  libisofs0 libjte1 liblirc-client0 liblivemedia62 libluas5.2-0
  libmad0 libmatroska6v5 libmicrodns0 libmpcdec6 libmpeg2-4 libnfs11
  libopenpmt-nodplug1 libperl4-corelibs-perl libplacebo4 libprotobuf-lite10
  libproxy-tools libpwquality-common libpwquality1 libqt5x11extras5 libraw16
  libresid-builder0c2a librsvg2-1 libsshd2-1 libsuitesparseconfigs libtracker-sparql-2.0-0 libupnp6
  libusageenvironment3 libva-wayland2 libvulkan1 libwayland-egl1-mesa
  libvpx0 libxml2-2.9.1 libxmlsec1-ocaml libxmlsec1-ocaml-media libxmlsec1-ocaml-info
```

Lanjutkan dengan perintah: `sudo apt install openjdk-8-jre-headless`.

```
dodit@dodit:~$ sudo apt install openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bubblewrap cracklib-runtime dvd+rw-tools efibootmgr fonts-liberation2
  fonts-opensymbol geoclue-2.0 gir1.2-gst-plugins-base-1.0 gir1.2-gtk-2.0
  gir1.2-gudev-1.0 gir1.2-udisks-2.0 gnome-desktop3-data growisofs
  gstreamer1.0-gtk3 gvfs-bin iio-sensor-proxy libappstream-glib8 libaribb24-0
  libbb2-1 libbasicusagenvironment1 libbdplus0 libboost-date-time1.65.1
  libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1
  libboost-system1.65.1 libboost-thread1.65.1 libburn4 libcdbs2 libcolamd2
  libcrack2 libdc1394-22 libdca0 libdvbpsi10 libdyndnav4 libdvread4 libebml4v5
  libeot0 libevent-2.1-6 libexiv2-14 libfaad2 libfwup1 libgeoclue-2-0
  libgexiv2-2 libgles2 libgimme-3.0-0 libgnome-desktop-3-17
  libgoa-backend-1.0-1 libgpgmepp6 libgpod-common libgpod4 libgroupsock8
  libisofs6 libjte1 libkate1 liblirc-client liblivemedia62 liblua5.2-0
  libmad0 libmatroska6v5 libmicrodns0 libmpcdec6 libmpeg2-4 libnfs11
  libopenmpt-modplug1 libperl4-corelibs-perl libplacebo4 libprotobuf-lite10
  libproxy-tools libpwquality-common libpwquality1 libqt5x11extras5 libraw16
  libresid-builder0c2a librsync1 libSDL-image1.2 libSDL1.2debian libsgutils2-2
  libsidplay2 libssh2-1 libsuiteparseconfig5 libtracker-sparql-2.0-0 libupnp6
  libusageenvironment3 libva-wayland2 libvulkan1 libwayland-egl1-mesa
  libxcb-xv0 libxmlsec1 libxmlsec1-nss lp-solve media-player-info
```

Cek sekali lagi apakah Java telah terinstall dengan perintah `java --version`.

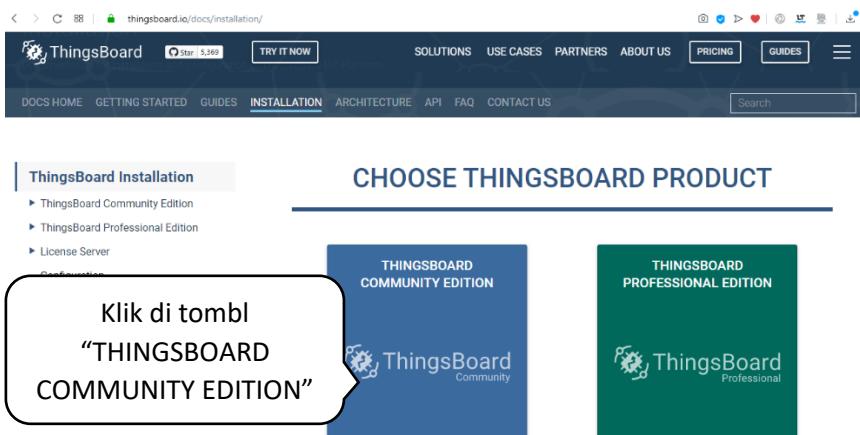
```
dodit@ThingsBoard:~$ java --version
openjdk 11.0.6 2020-01-14
OpenJDK Runtime Environment (build 11.0.6+10-post-Ubuntu-lubuntull8.04.1)
OpenJDK 64-Bit Server VM (build 11.0.6+10-post-Ubuntu-lubuntull8.04.1, mixed mode, sharing)
dodit@ThingsBoard:~$
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and displays the following text:

```
• dodit@dodit:~  
File Edit View Search Terminal Help  
dodit@dodit:~$ java --version  
openjdk 11.0.6 2020-01-14  
OpenJDK Runtime Environment (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1)  
OpenJDK Server VM (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode, sharing)  
dodit@dodit:~$
```

Untuk mengetahui langkah-langkah instalasi Thingsboard pada Ubuntu silahkan kunjungi halaman <https://thingsboard.io/docs/installation/>. Kemudian pilih tombol “THINGSBOARD COMMUNITY EDITION”.



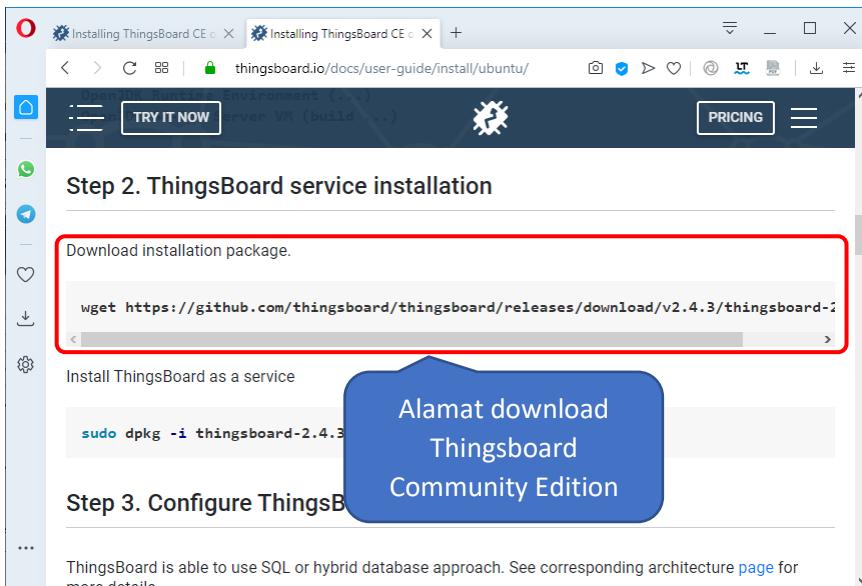
Dari sini kita akan menuju halaman <https://thingsboard.io/docs/user-guide/install/ubuntu/> yang

berisi alamat download Thingsboard Community dan bagaimana tahapan instalasinya.

The screenshot shows a web browser displaying the ThingsBoard documentation at thingsboard.io/docs/user-guide/install/ubuntu/. The page has a header with navigation links like TRY IT NOW, SOLUTIONS, USE CASES, PARTNERS, ABOUT US, PRICING, and GUIDES. The main content is titled "Installing ThingsBoard CE on Ubuntu Server". On the left, there's a sidebar with a tree view of installation options: "ThingsBoard Community Edition" expanded, showing "Installation options" which includes "Installing ThingsBoard CE on Ubuntu Server" (selected), "Installing ThingsBoard CE on CentOS/RHEL Server", "Installing ThingsBoard CE on Windows", "Installing ThingsBoard CE on Raspberry Pi 3", "Installing ThingsBoard CE using Docker (Windows)", "Installing ThingsBoard CE using Docker (Linux or Mac OS)", "Installing ThingsBoard CE on AWS", "Installing ThingsBoard CE on Azure", and "Installing ThingsBoard CE on". The right side contains sections for "Prerequisites" (with a list of steps from 1 to 6), "Step 1. Install Java 8 (OpenJDK)" (with a note about memory requirements), and "Step 2. ThingsBoard service installation".

Kita lanjutkan sekarang dengan menginstall Thingsboard dengan cara *cloning* dari github. Repository package Thingsboard diletakkan pada alamat berikut:

```
wget  
https://github.com/thingsboard/thingsboard/releases/download/v2.4.3/thingsboard-2.4.3.deb
```



Step 2. ThingsBoard service installation

Download installation package.

```
wget https://github.com/thingsboard/thingsboard/releases/download/v2.4.3/thingsboard-2.4.3.deb
```

Install ThingsBoard as a service

```
sudo dpkg -i thingsboard-2.4.3.deb
```

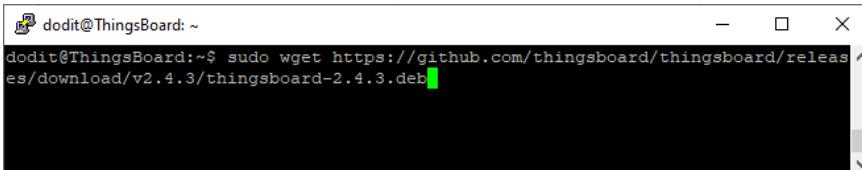
Step 3. Configure ThingsBoard

...
ThingsBoard is able to use SQL or hybrid database approach. See corresponding architecture [page](#) for more details.

Alamat download
Thingsboard
Community Edition

Sekarang masuk lagi ke aplikasi Terminal Ubuntu MATE dengan perintah:

```
sudo wget  
https://github.com/thingsboard/thingsboard/releases/download/v2.4.3/thingsboard-2.4.3.deb
```



```
mz-Date=20200403T090154Z&X-Amz-Expires=300&X-Amz-Signature=84b772lab0965ba085a3  
93f864fe2f5b40cd027ae6f534d3a2083d179fce270c&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dthingsboard-2.4.3.deb&response-content-type=application%2Foctet-stream [following]  
--2020-04-03 16:03:41-- https://github-production-release-asset-2e65be.s3.amazonaws.com/75277003/9ac8f600-322d-11ea-91a0-f7a175ef7b19?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200403%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200403T090154Z&X-Amz-Expires=300&X-Amz-Signature=84b772lab0965ba085a393f864fe2f5b40cd027ae6f534d3a2083d179fce270c&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dthingsboard-2.4.3.deb&response-content-type=application%2Foctet-stream  
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 54.231.50.83  
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com) |54.231.50.83|:443... connected  
.HTTP request sent, awaiting response... 200 OK  
Length: 116676468 (111M) [application/octet-stream]  
Saving to: 'thingsboard-2.4.3.deb'  
  
thingsboard-2.4.3.d 62%[=====>] 69,24M 220KB/s eta 2m 15s
```



```
File Edit View Search Terminal Help  
dodit@dodit:~$ wget https://github.com/thingsboard/thingsboard/releases/download/v2.4.3/thingsboard-2.4.3.deb  
--2020-03-14 07:04:50-- https://github.com/thingsboard/thingsboard/releases/download/v2.4.3/thingsboard-2.4.3.deb  
Resolving github.com (github.com)... 13.229.188.59  
Connecting to github.com (github.com)|13.229.188.59|:443... connected.  
HTTP request sent, awaiting response... 302 Found  
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/75277003/9ac8f600-322d-11ea-91a0-f7a175ef7b19?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200314%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200314T000452Z&X-Amz-Expires=3008X-Amz-Signature=a97ecfc51df0365da7a407c2604fb2eaaaa628e2ba1ddc1278184ef0839a1658X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dthingsboard-2.4.3.deb&response-content-type=application%2Foctet-stream [following]  
--2020-03-14 07:04:52-- https://github-production-release-asset-2e65be.s3.amazonaws.com/75277003/9ac8f600-322d-11ea-91a0-f7a175ef7b19?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200314%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200314T000452Z&X-Amz-Expires=3008X-Amz-Signature=a97ecfc51df0365da7a407c2604fb2eaaaa628e2ba1ddc1278184ef0839a1658X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dthingsboard-2.4.3.deb&response-content-type=application%2Foctet-stream  
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.80.232
```

dodit@dodit: ~

File Edit View Search Terminal Help

HTTP request sent, awaiting response... 302 Found

Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/75277003/9ac8f600-322d-11ea-91a0-f7a175ef7b19?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKTAIMNJJAX4CSVEH53A%2F20200314%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200314T00045Z&X-Amz-Signature=a97ecfc511df0365da7a407c2604fb2aaaaa628e2ba1ddc1278184ef0839a165&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dthingsboard-2.4.3.deb&response-content-type=application%2Foctet-stream [following]

--2020-03-14 07:04:52-- https://github-production-release-asset-2e65be.s3.amazonaws.com/75277003/9ac8f600-322d-11ea-91a0-f7a175ef7b19?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKTAIMNJJAX4CSVEH53A%2F20200314%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200314T00045Z&X-Amz-Signature=a97ecfc511df0365da7a407c2604fb2aaaaa628e2ba1ddc1278184ef0839a165&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dthingsboard-2.4.3.deb&response-content-type=application%2Foctet-stream

Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.80.232

Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.80.232|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 116676468 (111M) [application/octet-stream]

Saving to: 'thingsboard-2.4.3.deb'

thingsboard-2.4.3.d 82%[=====] 92.13M 486KB/s eta 26s

Cek apakah thingsboard sukses di download dengan perintah ls, sekaligus lakukan instalasi Thingsboard sebagai service dengan perintah

```
sudo dpkg -i thingsboard-2.4.3.deb
```

```
dodit@ThingsBoard: ~
```

```
dodit@ThingsBoard:~$ ls
```

```
Desktop Downloads Pictures snap thingsboard-2.4.3.deb
```

```
Documents Music Public Templates Videos
```

```
dodit@ThingsBoard:~$ sudo dpkg -i thingsboard-2.4.3.deb
```

```
[sudo] password for dodit:
```

```
Selecting previously unselected package thingsboard.
```

```
(Reading database ... 170440 files and directories currently installed.)
```

```
Preparing to unpack thingsboard-2.4.3.deb ...
```

```
Adding group `thingsboard' (GID 129) ...
```

```
Done.
```

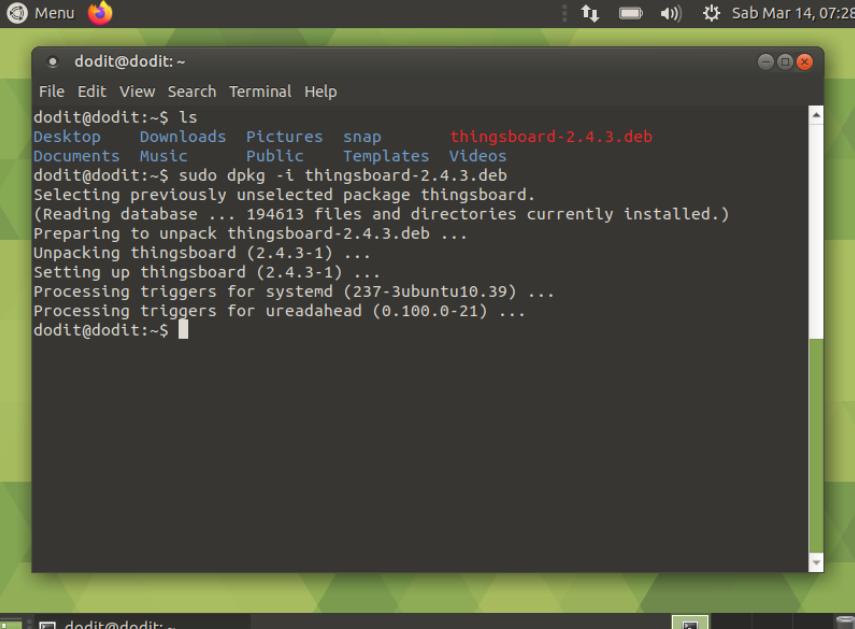
```
Unpacking thingsboard (2.4.3-1) ...
```

```
Setting up thingsboard (2.4.3-1) ...
```

```
Processing triggers for ureadahead (0.100.0-21) ...
```

```
Processing triggers for systemd (237-3ubuntu10.33) ...
```

```
dodit@ThingsBoard:~$
```

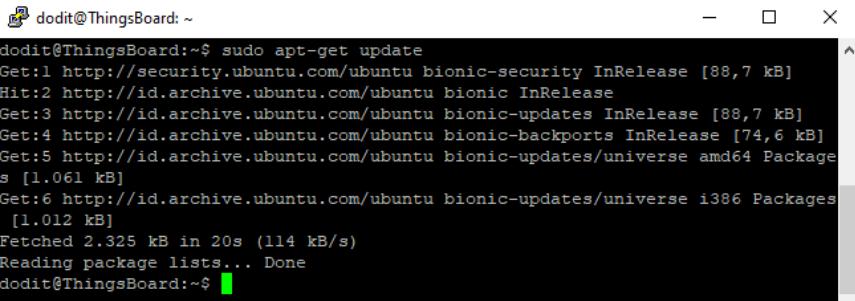


A screenshot of a Linux desktop environment. At the top, there's a green header bar with icons for file operations, network, battery, and system status. The date "Sab Mar 14, 07:28" is visible. Below the header is a terminal window titled "dodit@dodit: ~". The terminal shows the command "sudo dpkg -i thingsboard-2.4.3.deb" being run, followed by the output of the package installation process. The desktop background is a green and yellow abstract pattern.

```
dodit@dodit:~$ ls
Desktop Downloads Pictures snap      thingsboard-2.4.3.deb
Documents Music   Public   Templates Videos
dodit@dodit:~$ sudo dpkg -i thingsboard-2.4.3.deb
Selecting previously unselected package thingsboard.
(Reading database ... 194613 files and directories currently installed.)
Preparing to unpack thingsboard-2.4.3.deb ...
Unpacking thingsboard (2.4.3-1) ...
Setting up thingsboard (2.4.3-1) ...
Processing triggers for systemd (237-3ubuntu10.39) ...
Processing triggers for ureadahead (0.100.0-21) ...
dodit@dodit:~$
```

Lakukan update sekali lagi dengan perintah

```
sudo apt-get update
```



A screenshot of a terminal window titled "dodit@ThingsBoard: ~". It shows the command "sudo apt-get update" being run and its output. The output includes several "Get:" lines for different packages and their sizes, followed by "Fetched 2.325 kB in 20s (114 kB/s)", "Reading package lists... Done", and the prompt "dodit@ThingsBoard:~\$".

```
dodit@ThingsBoard:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Hit:2 http://id.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://id.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Get:4 http://id.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Get:5 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1.061 kB]
Get:6 http://id.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [1.012 kB]
Fetched 2.325 kB in 20s (114 kB/s)
Reading package lists... Done
dodit@ThingsBoard:~$
```

Instal database PosgreSQL dengan perintah

```
sudo apt-get install postgresql postgresql-contrib
```

dan sekaligus menjalankan service PosgreSQL dengan perintah

```
sudo service postgresql start
```

```
dodit@ThingsBoard:~$ sudo apt-get install postgresql postgresql-contrib
[sudo] password for dodit:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpq5 postgresql-10 postgresql-client-10 postgresql-client-common
  postgresql-common sysstat
Suggested packages:
  postgresql-doc locales-all postgresql-doc-10 libjson-perl isag
The following NEW packages will be installed:
  libpq5 postgresql postgresql-10 postgresql-client-10
  postgresql-client-common postgresql-common postgresql-contrib sysstat
0 upgraded, 8 newly installed, 0 to remove and 107 not upgraded.
Need to get 5.298 kB of archives.
After this operation, 20,9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://id.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libpq5 amd64
  10.12-0ubuntu0.18.04.1 [107 kB]
4% [Waiting for headers]
```

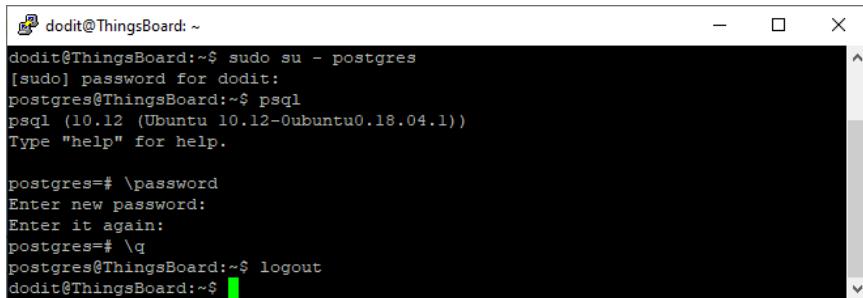
```
dodit@ThingsBoard:~$ sudo service postgresql start
dodit@ThingsBoard:~$
```

```
● dodit@dodit:~
File Edit View Search Terminal Help
dodit@dodit:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://ppa.launchpad.net/mosquitto-dev/mosquitto-ppa/ubuntu bionic InRelease
Hit:3 http://id.archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://id.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://id.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
dodit@dodit:~$ 
dodit@dodit:~$ sudo apt-get install postgresql postgresql-contrib
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bubblewrap cracklib-runtime dvd+rw-tools efibootmgr fonts-liberation2
  fonts-opensymbol geoclue-2.0 gir1.2-gst-plugins-base-1.0 gir1.2-gtk-2.0
  gir1.2-gudev-1.0 gir1.2-udisks-2.0 gnome-desktop3-data growlsofts
  gstreamer1.0-gtk3 gvfs-bin iio-sensor-proxy libappstream-glib8 libaribb24-0
  libbb2-1 libbasicusageenvironment1 libbdplus0 libboost-date-time1.65.1
  libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1
  libboost-system1.65.1 libboost-thread1.65.1 libburn4 libcddb2 libcolamd2
  libcrack2 libdc1394-22 libdca0 libdvbsip10 libdvdnav4 libdvdread4 libebml4v5
  libeot0 libevent-2.1-6 libexiv2-14 libfaad2 libfwup1 libgeoip2-2.0
  libgexiv2-2 libgles2 libqmime-3.0-0 libgnome-desktop-3-17
```

Setelah PostgreSQL diinstal, buatlah user dan password untuk user utama. Perintah di bawah ini untuk membuat password sebagai pengguna postgresql utama. Kemudian, tekan “Ctrl + D” untuk logout kembali ke konsol user utama.

```
sudo su - postgres
psql
\password
\q
sudo service postgresql start
```

Nama user
utama

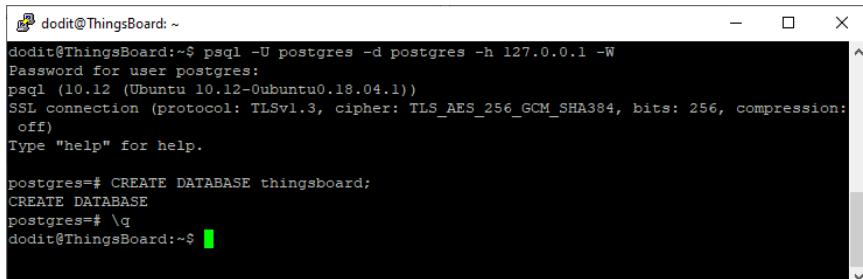


```
dodit@ThingsBoard:~$ sudo su - postgres
[sudo] password for dodit:
postgres@ThingsBoard:~$ psql
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
Type "help" for help.

postgres=# \password
Enter new password:
Enter it again:
postgres=# \q
postgres@ThingsBoard:~$ logout
dodit@ThingsBoard:~$
```

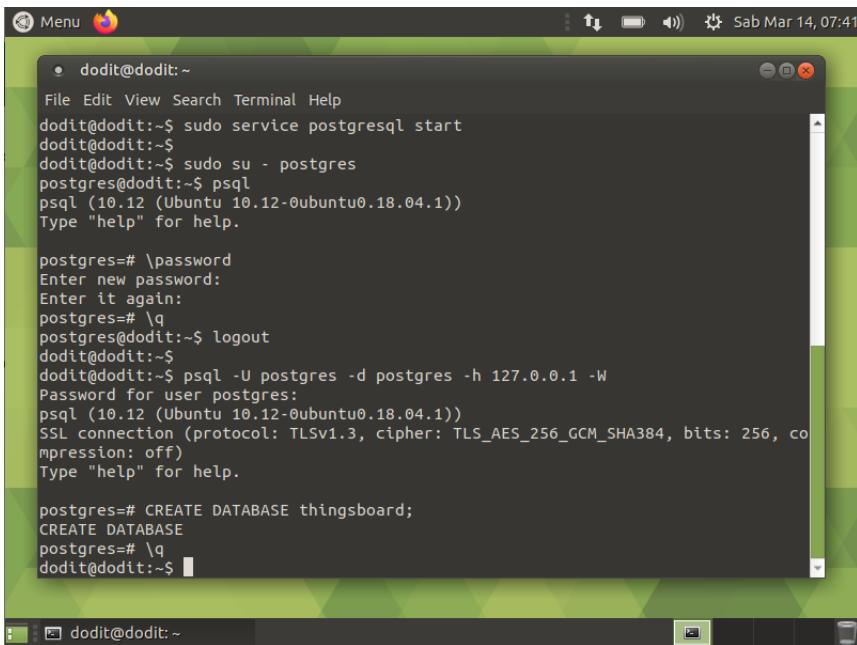
Hubungkan user utama tadi ke database dengan nama “thingsboard” yang akan kita buat dengan perintah berikut:

```
psql -U postgres -d postgres -h 127.0.0.1 -W
CREATE DATABASE thingsboard;
\q
exit
```



```
dodit@ThingsBoard:~$ psql -U postgres -d postgres -h 127.0.0.1 -W
Password for user postgres:
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# CREATE DATABASE thingsboard;
CREATE DATABASE
postgres=# \q
dodit@ThingsBoard:~$
```



A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "dodit@dodit: ~". The terminal content shows the following steps:

```
File Edit View Search Terminal Help
dodit@dodit:~$ sudo service postgresql start
dodit@dodit:~$ sudo su - postgres
postgres@dodit:~$ psql
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
Type "help" for help.

postgres=# \password
Enter new password:
Enter it again:
postgres=# \q
postgres@dodit:~$ logout
dodit@dodit:~$ psql -U postgres -d postgres -h 127.0.0.1 -W
Password for user postgres:
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# CREATE DATABASE thingsboard;
CREATE DATABASE
postgres=# \q
dodit@dodit:~$
```

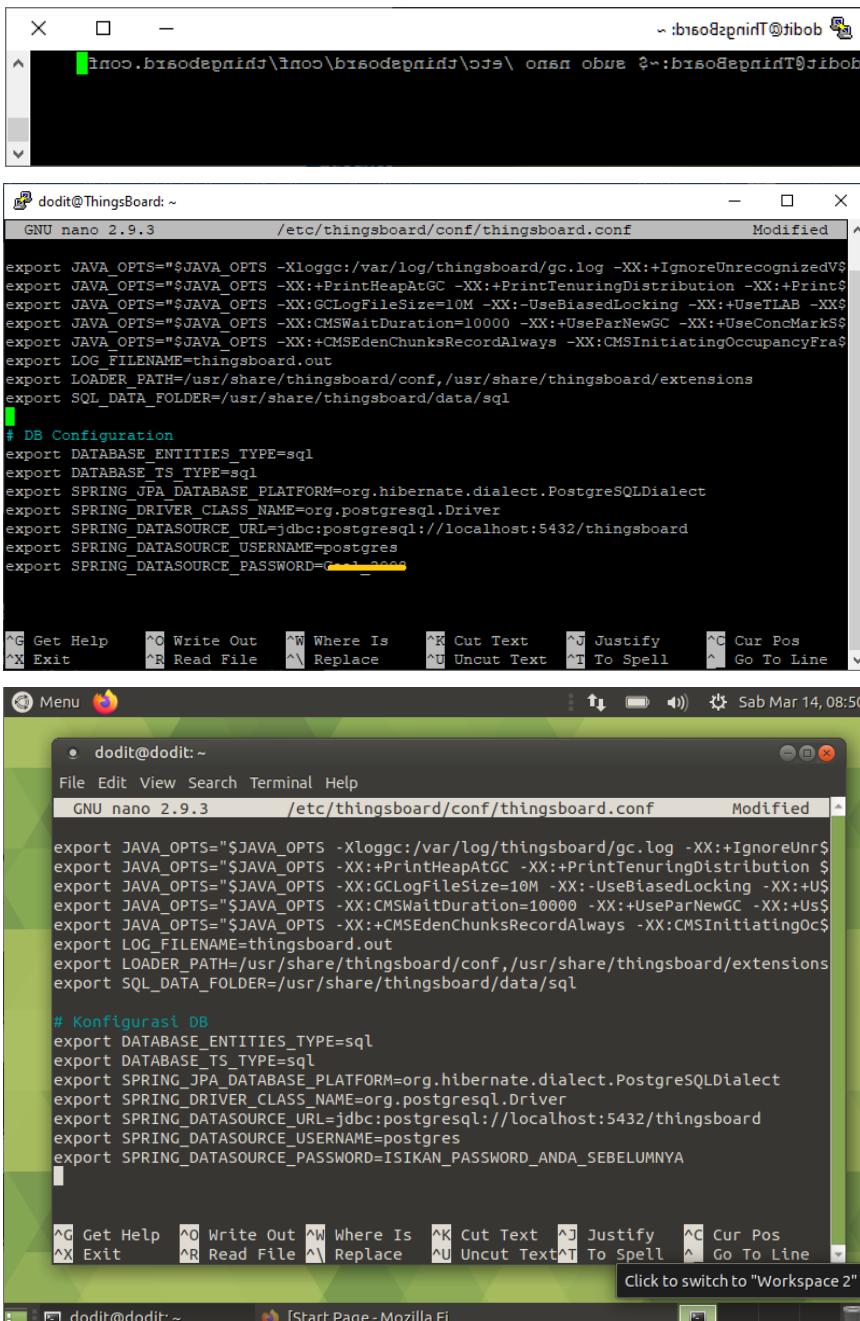
Lakukan editing terhadap file konfigurasi `thingsboard.conf` dengan perintah:

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

Kemudian tambahkan skrip berikut:

```
# DB Configuration
export DATABASE_ENTITIES_TYPE=mysql
export DATABASE_TS_TYPE=mysql
export
SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect
export
SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
export
SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=PASSWORD
```

Khusus "**PASSWORD**" silahkan diganti dengan password user utama database PosgresSQL yang telah kita buat sebelumnya.



```

dodit@ThingsBoard: ~
GNU nano 2.9.3 /etc/thingsboard/conf/thingsboard.conf Modified

export JAVA_OPTS="$JAVA_OPTS -Xloggc:/var/log/thingsboard/gc.log -XX:+IgnoreUnrecognizedVMOptions"
export JAVA_OPTS="$JAVA_OPTS -XX:+PrintHeapAtGC -XX:+PrintTenuringDistribution -XX:+PrintGCTimeStamps"
export JAVA_OPTS="$JAVA_OPTS -XX:GCTimeLimit=10M -XX:-UseBiasedLocking -XX:+UseTLAB -XX:+UseConcMarkSweepGC"
export JAVA_OPTS="$JAVA_OPTS -XX:CMSWaitDuration=10000 -XX:+UseParNewGC -XX:+UseConcMarkSweepGC"
export LOG_FILENAME=thingsboard.out
export LOADER_PATH=/usr/share/thingsboard/conf,/usr/share/thingsboard/extensions
export SQL_DATA_FOLDER=/usr/share/thingsboard/data/sql

# DB Configuration
export DATABASE_ENTITIES_TYPE=mysql
export DATABASE_TS_TYPE=mysql
export SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect
export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
export SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=ISIKAN_PASSWORD_ANDA_SEBELUMNYA

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^L Replace ^U Uncut Text ^T To Spell ^_ Go To Line

```

File Edit View Search Terminal Help

GNU nano 2.9.3 /etc/thingsboard/conf/thingsboard.conf Modified

```

export JAVA_OPTS="$JAVA_OPTS -Xloggc:/var/log/thingsboard/gc.log -XX:+IgnoreUnrecognizedVMOptions"
export JAVA_OPTS="$JAVA_OPTS -XX:+PrintHeapAtGC -XX:+PrintTenuringDistribution -XX:+PrintGCTimeStamps"
export JAVA_OPTS="$JAVA_OPTS -XX:GCTimeLimit=10M -XX:-UseBiasedLocking -XX:+UseTLAB -XX:+UseConcMarkSweepGC"
export JAVA_OPTS="$JAVA_OPTS -XX:CMSWaitDuration=10000 -XX:+UseParNewGC -XX:+UseConcMarkSweepGC"
export LOG_FILENAME=thingsboard.out
export LOADER_PATH=/usr/share/thingsboard/conf,/usr/share/thingsboard/extensions
export SQL_DATA_FOLDER=/usr/share/thingsboard/data/sql

# Konfigurasi DB
export DATABASE_ENTITIES_TYPE=mysql
export DATABASE_TS_TYPE=mysql
export SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect
export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
export SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=ISIKAN_PASSWORD_ANDA_SEBELUMNYA

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^L Replace ^U Uncut Text ^T To Spell ^_ Go To Line

```

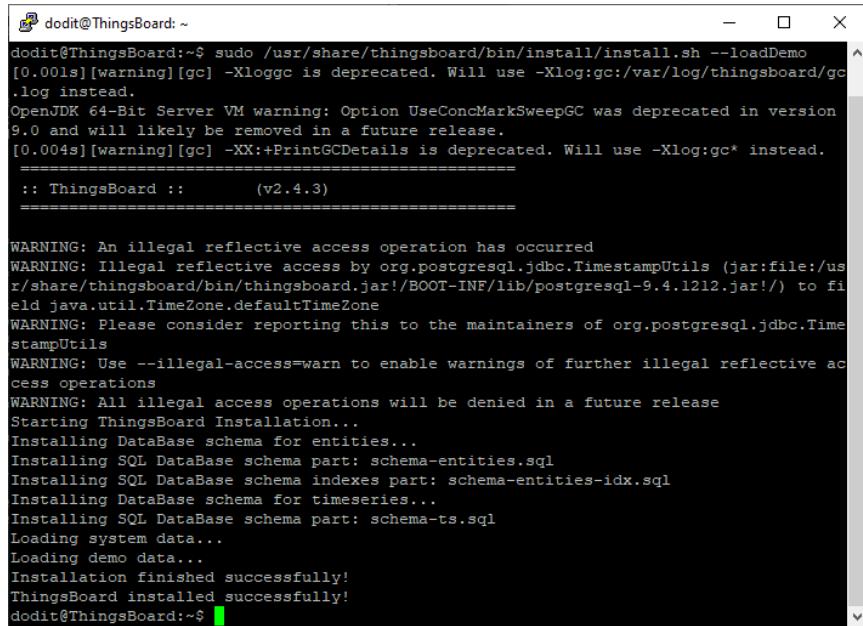
Click to switch to "Workspace 2"

Setelah ditambahkan skrip, kemudian gunakan keyboard Ctrl+X, Y dan Enter untuk keluar dari editor dan menyimpannya.

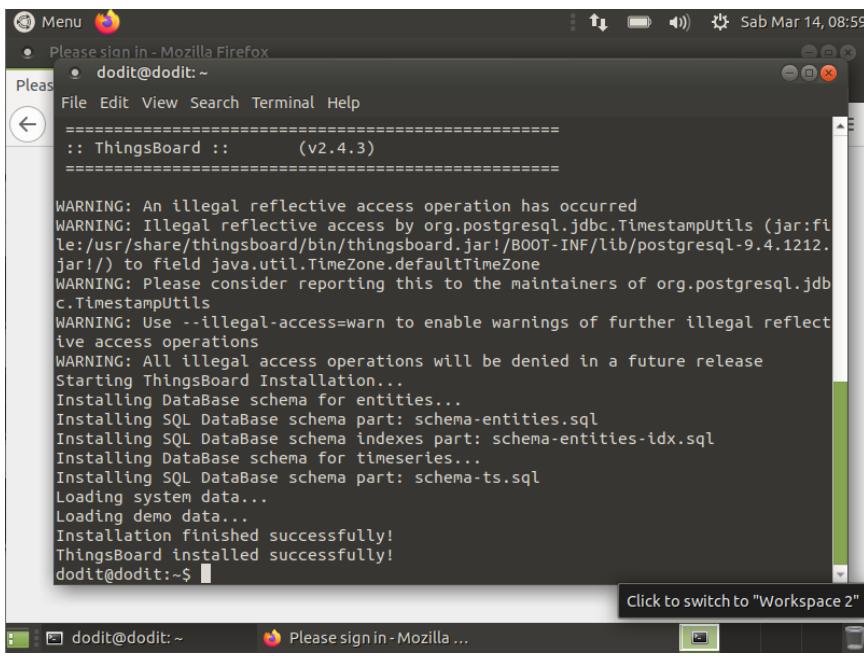
Setelah service ThingsBoard diinstal dan konfigurasi DB diperbarui, Anda dapat menjalankan skrip berikut:

```
# --loadDemo option will load demo data: users,  
devices, assets, rules, widgets.  
sudo  
/usr/share/thingsboard/bin/install/install.sh --  
loadDemo
```

Pilihan --loadDemo bertujuan untuk memuat data demo, misalnya user, devices, assets, rules dan widget.



```
dodit@ThingsBoard:~$ sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo  
[0.001s][warning][gc] -Xloggc is deprecated. Will use -Xlog:gc:/var/log/thingsboard/gc.log instead.  
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version  
9.0 and will likely be removed in a future release.  
[0.004s][warning][gc] -XX:+PrintGCDetails is deprecated. Will use -Xlog:gc* instead.  
=====  
:: ThingsBoard :: (v2.4.3)  
=====  
  
WARNING: An illegal reflective access operation has occurred  
WARNING: Illegal reflective access by org.postgresql.jdbc.TimestampUtils (jar:file:/usr/share/thingsboard/bin/thingsboard.jar!/BOOT-INF/lib/postgresql-9.4.1212.jar!/) to field java.util.TimeZone.defaultTimeZone  
WARNING: Please consider reporting this to the maintainers of org.postgresql.jdbc.TimestampUtils  
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations  
WARNING: All illegal access operations will be denied in a future release  
Starting ThingsBoard Installation...  
Installing DataBase schema for entities...  
Installing SQL DataBase schema part: schema-entities.sql  
Installing SQL DataBase schema indexes part: schema-entities-idx.sql  
Installing DataBase schema for timeseries...  
Installing SQL DataBase schema part: schema-ts.sql  
Loading system data...  
Loading demo data...  
Installation finished successfully!  
ThingsBoard installed successfully!  
dodit@ThingsBoard:~$
```

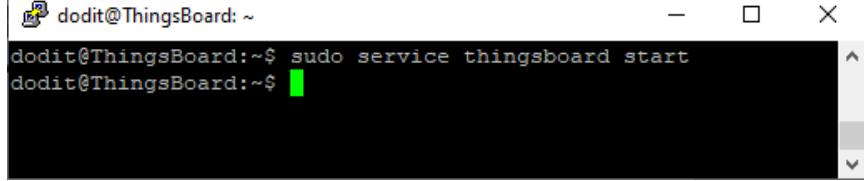


```
Menu Please sign in - Mozilla Firefox dodit@dodit:~ File Edit View Search Terminal Help ======: ThingsBoard :: (v2.4.3)=====WARNING: An illegal reflective access operation has occurredWARNING: Illegal reflective access by org.postgresql.jdbc.TimestampUtils (jar:file:/usr/share/thingsboard/bin/thingsboard.jar!/BOOT-INF/lib/postgresql-9.4.1212.jar!) to field java.util.TimeZone.defaultTimeZoneWARNING: Please consider reporting this to the maintainers of org.postgresql.jdbc.TimestampUtilsWARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operationsWARNING: All illegal access operations will be denied in a future releaseStarting ThingsBoard Installation...Installing DataBase schema for entities...Installing SQL DataBase schema part: schema-entities.sqlInstalling SQL DataBase schema indexes part: schema-entities-idx.sqlInstalling DataBase schema for timeseries...Installing SQL DataBase schema part: schema-ts.sqlLoading system data...Loading demo data...Installation finished successfully!ThingsBoard installed successfully!dodit@dodit:~
```

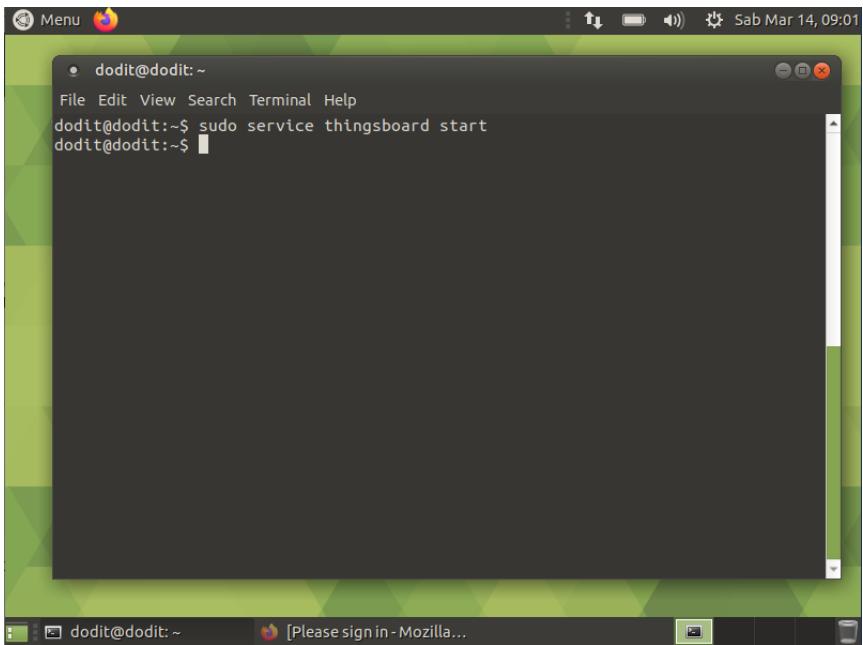
Click to switch to "Workspace 2"

Perintah berikut bertujuan agar Thingsboard mulai berkerja:

```
sudo service thingsboard start
```



```
dodit@ThingsBoard:~$ sudo service thingsboard start
dodit@ThingsBoard:~$
```

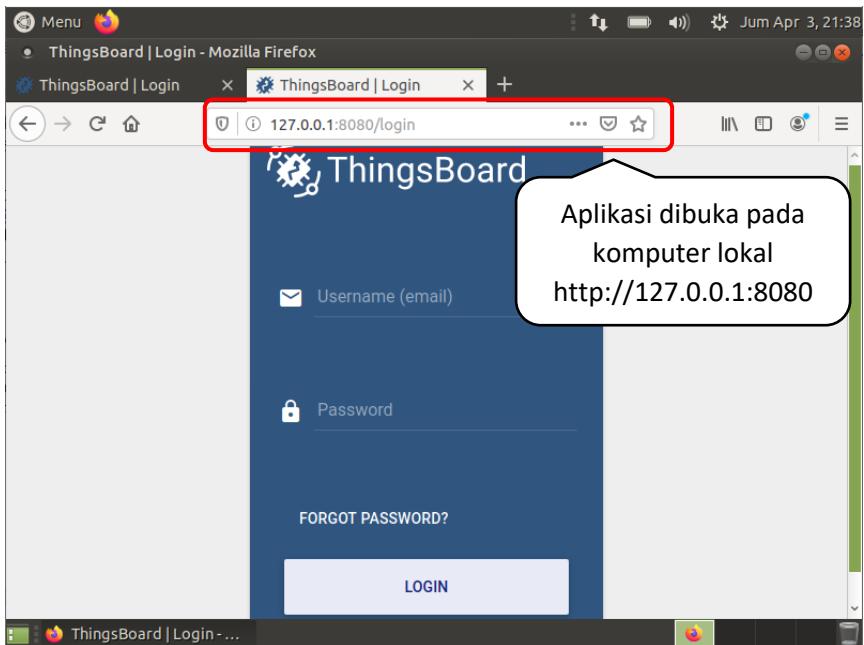
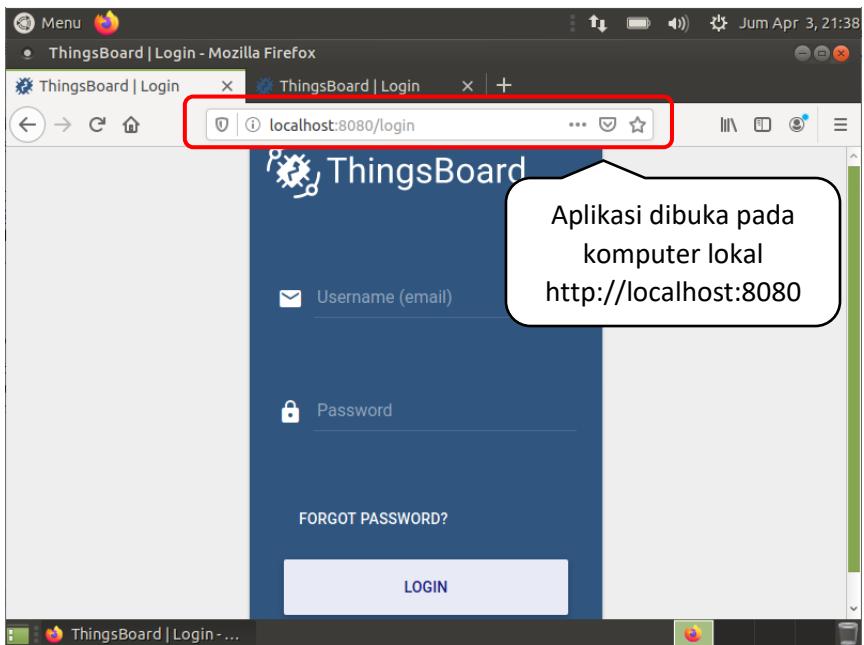


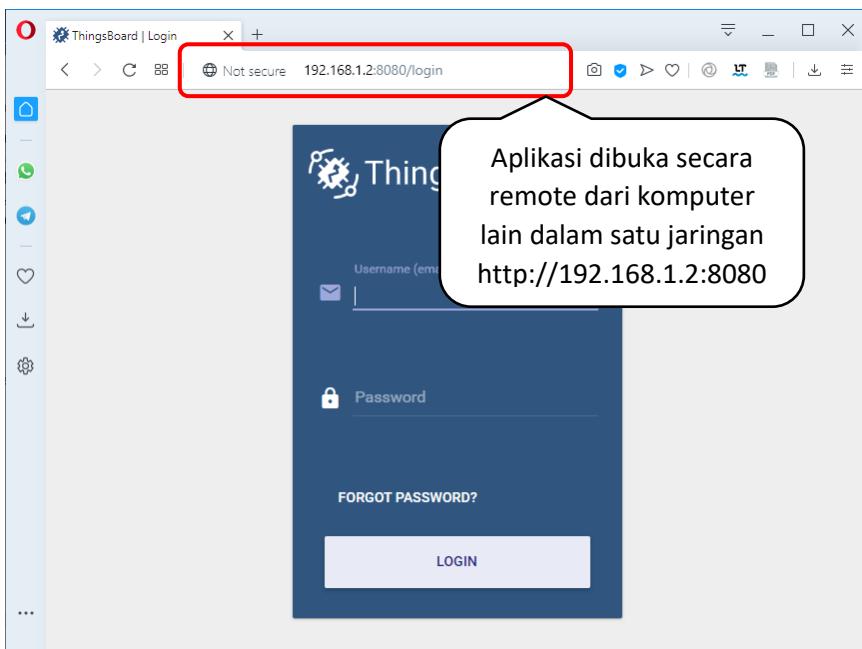
A screenshot of a Linux desktop environment. At the top, there's a green header bar with icons for system status and a date/time indicator ('Sab Mar 14, 09:01'). Below it is a terminal window titled 'dodit@dodit: ~'. The terminal shows the command 'sudo service thingsboard start' being run. In the background, a Firefox browser window is open with the URL '[Please sign in - Mozilla...]' visible in the address bar. The desktop has a green and grey geometric pattern.

```
dodit@dodit: ~
File Edit View Search Terminal Help
dodit@dodit:~$ sudo service thingsboard start
dodit@dodit:~$
```

Untuk mengoperasikan Dashboard Thingsboard, silahkan buka browser internet dengan alamat:

- Jika browser internet dijalankan secara remote maka gunakan IP dimana Thingsboard diinstall, misalnya <http://192.168.1.2:8080>
- Jika browser internet dijalankan pada komputer lokal maka bisa menggunakan alamat <http://127.0.0.1:8080>, <http://localhost:8080> atau <http://192.168.1.2:8080/>





Secara default user dan password ketika -loadDemo adalah sebagai berikut:

- **Administrator System:** sysadmin@thingsboard.org / sysadmin
- **Administrator Tenant:** tenant@thingsboard.org / tenant
- **Pengguna customer:** customer@thingsboard.org / customer

Akun Thingsboard dibagi menjadi tiga, posisi tertinggi adalah Administrator System yang bisa memiliki banyak Administrator Tenant dan Administrator bisa memiliki banyak pengguna pelanggan/kustomer. User tersebut bisa diganti setiap saat untuk menjaga keamanan.

Pembagian akun tersebut akan kita terapkan saat membangun sistem kontrol dan monitoring smart device NodeMCU dengan Thingsboard melalui aplikasi browser.

Contoh dashboard Thingsboard dengan akun **sysadmin**:

The image consists of two screenshots of the ThingsBoard web application interface.

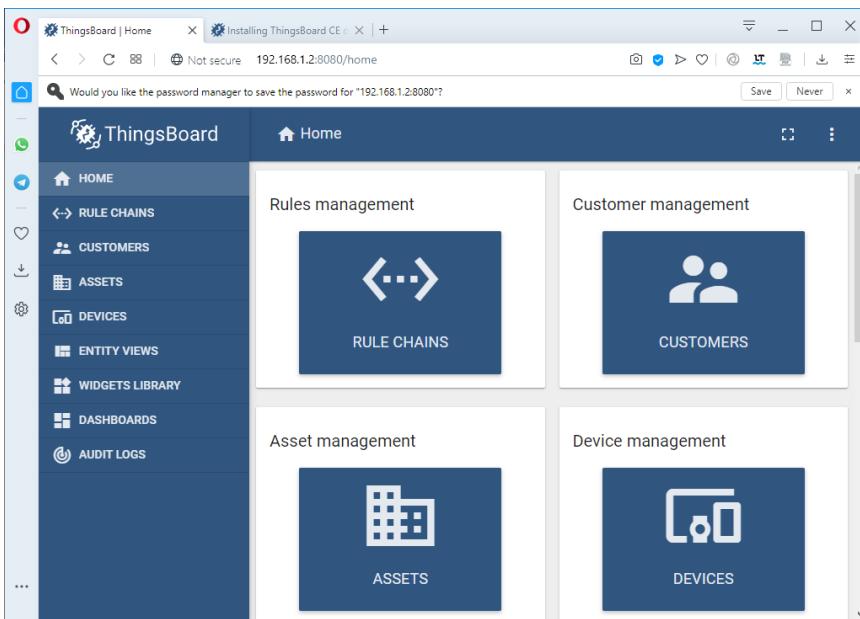
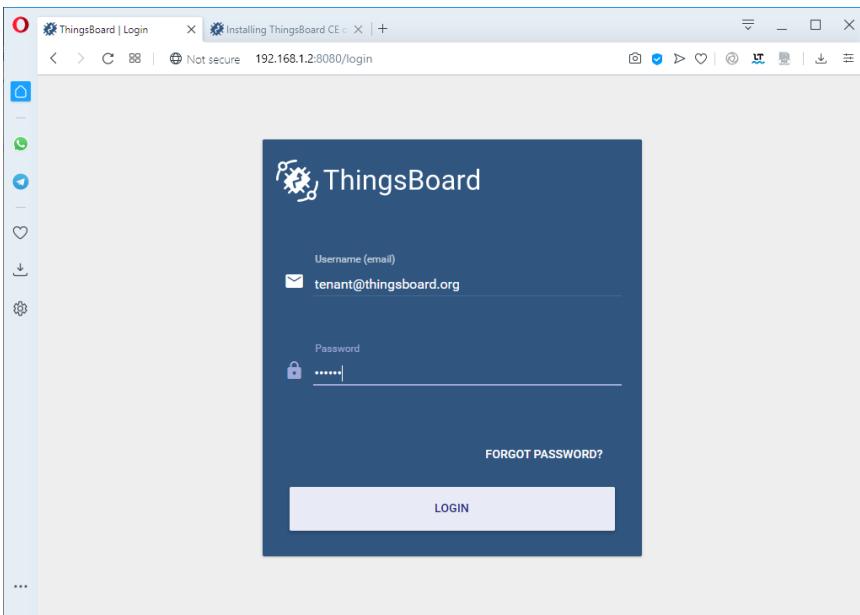
Top Screenshot (Login Screen):

- Header:** Shows the title "ThingsBoard | Login" and the URL "192.168.1.2:8080/login".
- Left Sidebar:** Contains icons for Home, Tenant Management, Widgets Library, and System Settings.
- Central Content:** A dark blue login form with:
 - A logo icon.
 - The text "ThingsBoard".
 - A "Username (email)" field containing "sysadmin@thingsboard.org".
 - A "Password" field with a lock icon.
 - A "FORGOT PASSWORD?" link.
 - A "LOGIN" button.

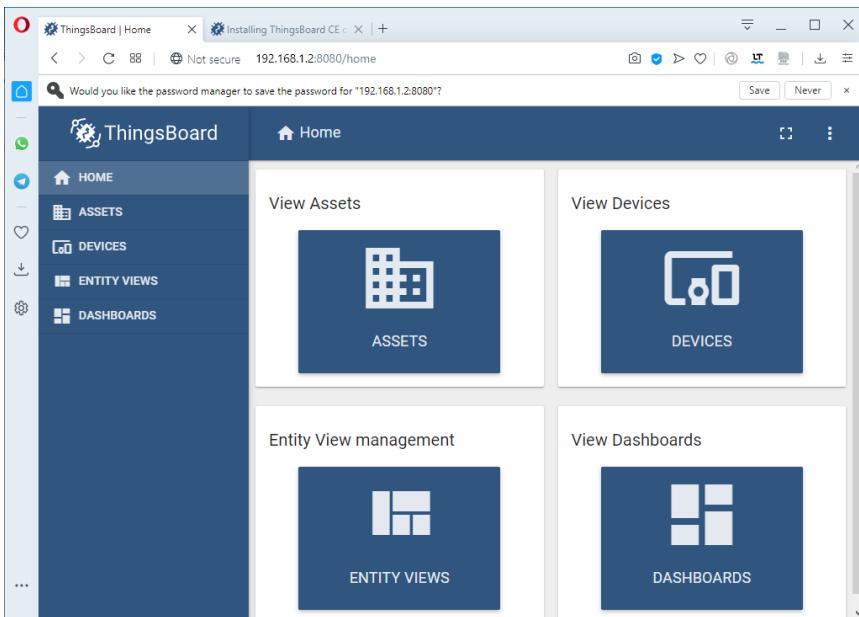
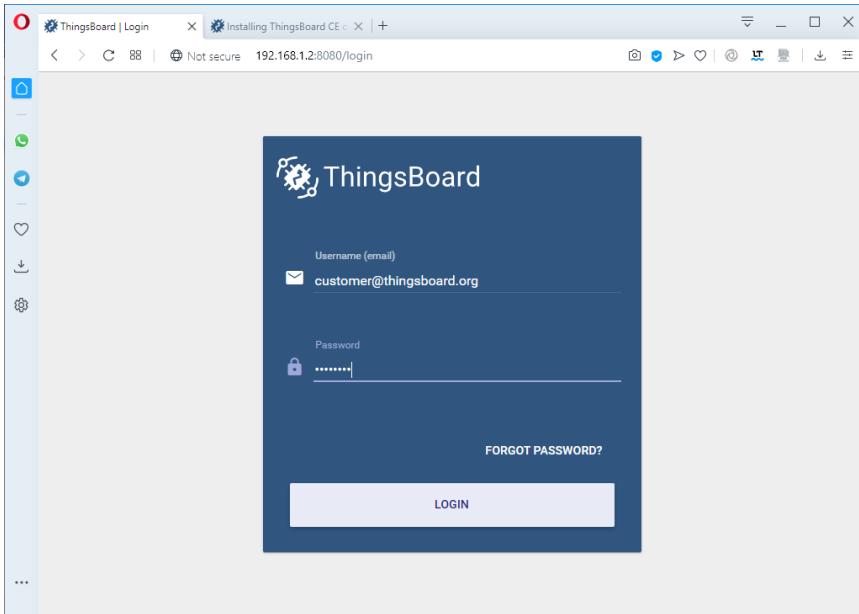
Bottom Screenshot (Home Dashboard):

- Header:** Shows the title "ThingsBoard | Home" and the URL "192.168.1.2:8080/home".
- Left Sidebar:** Contains icons for Home, Tenant Management, Widgets Library, and System Settings.
- Central Content:** A dashboard with three main sections:
 - Tenant management:** Includes a "TENANTS" button with a user icon.
 - Widget management:** Includes a "WIDGETS LIBRARY" button with a square icon.
 - System Settings:** Includes three buttons: "GENERAL" (gear icon), "MAIL SERVER" (envelope icon), and "SECURITY SETTINGS" (shield icon).

Contoh dashboard Thingsboard dengan akun **tenant**:



Contoh dashboard dengan akun **customer**:



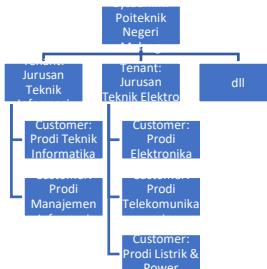
7.4 Coding Thingsboard Client Pada IoT Development

Setelah Anda menyiapkan IoT Server Blynk, sekarang kita akan mengembangkan aplikasi di sisi client atau dalam hal ini di sisi embedded system (IoT Development Board).

Di sini kita akan n

7.5 Manajemen Thingsboard

Secara struktur kepemilikan hak akses user pada Thingsboard dibagi tiga, yaitu Sysadmin, Tenant dan Customer. Sysadmin mempunyai hak akses tertinggi



7.5.1 Manajemen Thingsboard

Halaman Sysadmin memiliki menu yaitu Home, Tenants, Widget Library dan System Settings. Tidak banyak yang dapat dilakukan dari halaman Sysadmin. Halaman Home merupakan halaman ringkas dari layanan yang tersedi, halaman Tenants merupakan halaman untuk menambah dan mengatur tenant-tenant yang menjadi akun thingsboard, halaman library adalah halaman yang berisi informasi widget apa saja yang tersedia, Anda bisa pula menambahkan widget lain yang bersumber dari forum atau komunitas Thingsboard.

Halaman Home, pada halaman ini hanya berisi shortcut yang akan menuju ke halaman layanan lainnya.

The screenshot shows the Thingsboard Home page at the URL 192.168.1.4:8080/home. The interface is dark-themed. On the left is a sidebar with navigation links: HOME, TENANTS, WIDGETS LIBRARY, and SYSTEM SETTINGS. The main content area has four main sections: 'Tenant management' (with a 'TENANTS' button), 'Widget management' (with a 'WIDGETS LIBRARY' button), 'System Settings' (with three sub-options: 'GENERAL', 'MAIL SERVER', and 'SECURITY SETTINGS'). The browser's address bar shows the URL and indicates it is not secure.

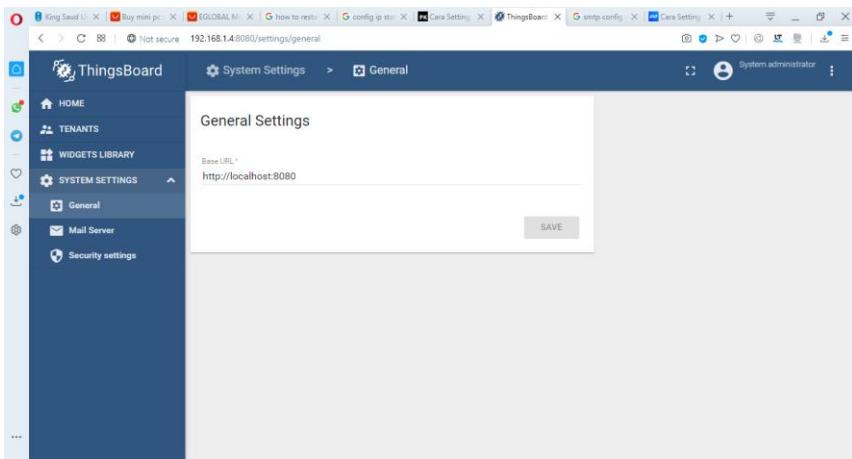
Halaman Tenants adalah halaman untuk menambah dan mengatur tenant, setiap tenant nantinya boleh menambahkan customer/pelanggan.

The screenshot shows the ThingsBoard web interface with the URL `192.168.1.4:8080/tenants`. The left sidebar has navigation links for HOME, TENANTS (which is selected), WIDGETS LIBRARY, and SYSTEM SETTINGS. The main content area is titled 'Tenants' and displays a single tenant entry: 'Tenant' with 'NO ADDRESS'. There are edit and delete icons below the entry. A red circular button with a '+' sign is located in the bottom right corner of the content area.

Halaman Widgets Library berisi informasi mengenai widget apa saja yang telah terinstall secara defualt, misalnya widget Chart, widget Analogue gauges, Cards dan lain sebagainya. Anda bisa menambahkan widget baru yang bersumber dari luar jika diperlukan.

The screenshot shows the ThingsBoard web interface with the URL `192.168.1.4:8080/widgets-bundles`. The left sidebar has navigation links for HOME, TENANTS, WIDGETS LIBRARY (which is selected), and SYSTEM SETTINGS. The main content area is titled 'Widgets Bundles' and lists various widget categories in a grid format. Each category has a checkbox, a name, and a 'SYSTEM' label. The categories are: Alarm widgets, Charts, Digital gauges, Analogue gauges, Control widgets, Entity admin widgets, Cards, Date, and Gateway widgets. Each category row has edit and delete icons. A red circular button with a '+' sign is located in the bottom right corner of the content area.

Halaman System Settings berisi tentang pengaturan mail server dan keamanan.



Langkah pertama adalah kita membuat tenant
Masih berlanjut

sdfsdf

Penutup

Buku ini ditujukan bagi siapapun dan kalangan manapun yang ingin belajar IoT pada tingkat dasar. Topik Internet of Things sangat luas sehingga tidak mungkin bagi kami untuk menuangkan tulisan tentang gagasan seputar IoT dalam satu buku saja.

Jika masih ada kesempatan, kami ingin mengembangkan tulisan IoT menuju ke tingkat lanjut. Bagaimana membangun sebuah arsitektur IoT “*create from scratch*” sehingga kita dapat mengkustomisasi sepenuhnya teknologi-teknologi IoT, terutama bagaimana pemanfaatan *message protocol* (MQTT, CoAP, dll), bagaimana membuat dashboard sistem yang melibatkan database, webserver, dan pemrograman web sebagai sarana monitoring IoT dan lain sebagainya.

Akhir kata semoga buku dapat bermanfaat bagi kita semua, membawa dampak positif terhadap pemahaman IoT bagi kalangan awam meskipun secara tampilan dan konten mungkin belum bisa memenuhi kebutuhan pembaca. Paling tidak dengan membaca buku ini bisa menjadi titik awal bagi pembaca untuk meningkatkan kemampuannya dalam mengerjakan projek-projek IoT dan belajar dari referensi lain yang lebih *advance*.

Referensi

- [1] Dodit Suprianto, Vipkas Al Hadid, Rini Agustina, D. W. W, *Microcontroller Arduino Untuk Pemula*, 1st ed. Jakarta: Jasakom, 2019.
- [2] D. Suprianto, R. N. Hasanah, and others, "Sistem Pengenalan Wajah Secara Real-Time dengan Adaboost, Eigenface PCA & MySQL," *J. EECIS*, vol. 7, no. 2, pp. 179–184, 2014.
- [3] D. Suprianto, D. W. Wibowo, A. Setiawan, and R. Agustina, "Integrated application design to facilitate effective and safe student pick-up process by utilizing microcontrollers, socket TCP, and client-server database," *J. Phys. Conf. Ser.*, vol. 1402, no. 7, 2019, doi: 10.1088/1742-6596/1402/7/077052.
- [4] D. Suprianto, *Membuat Aplikasi Desktop Menggunakan MySQL dan VB. Net*. Ciganjur Jagakarsa: Mediakita, 2010.
- [5] R. Agustina E Sutadji, "Analysis of implementation Augmented Reality (AR) introduction of temple and ancient objects based on android to increasing student learning outcomes," in *Annual Applied Science and Engineering Conference (AASEC 2018)*, 2018, no. IOP Publishing.
- [6] R. H. Yoga Perdana, N. Hidayati, A. W. Yulianto, V. Al Hadid Firdaus, N. N. Sari, and D. Suprianto, "Jig detection using scanning method base on internet of things for smart learning factory," *IEMTRONICS 2020 - Int. IOT, Electron. Mechatronics Conf. Proc.*, pp. 0-4, 2020, doi: 10.1109/IEMTRONICS51293.2020.9216392.
- [7] R. Agustina, D. Suprianto, and I. Muslimin, *Analisis Perancangan Pemesanan Makanan Menggunakan Smartphone Berbasis Android*, vol. 7, no. 02. 2017.
- [8] D. Suprianto and R. Agustina, "Pemrograman Aplikasi Android," *Yogyakarta: MediaKom*, 2012.

Biografi Penulis



Dodit Suprianto

Profesi sebagai pengajar di Politeknik Negeri Malang, jurusan Teknologi Informasi, konsultan IT dan penulis buku komputer. Minat penelitian pada bidang IoT dan Machine Learning.

Web: <http://doditsuprianto.blogspot.com/>
Email: doditsuprianto@gmail.com



Rini Agustina

Dosen pada program studi Sistem Informasi di Universitas Kanjuruhan Malang. Lulus S1 jurusan Sistem Informasi di STIKI Malang, S2 dan S3 di Jurusan Pendidikan Kejuruan di Universitas Negeri Malang. Minat penelitian pada bidang Multimedia, DSS, Social Humanities, Career Development dan Vocational Leadership.

Web: <https://rinimyhanny.wordpress.com/>
Email: riniagustina@unikama.ac.id



Tiffany Azhar Izzuddin

Lulus dari pondok pesantren dan sekarang berstatus sebagai mahasiswi, menyukai IoT, anime, drakor seperti layaknya remaja putri lainnya.