



# Internet of Things

ELITIA Laboratory  
2024/2025



---

## **TIM PENYUSUN MODUL PRAKTIKUM**

**Electronics and Industrial Automation Laboratory 2024/2025**  
P303-P304, Gedung Deli, Telkom University

Kepala Laboratorium	:	Dr. Junartho Halomoan
Pembina Laboratorium	:	Istiqomah, S.T. M.Sc
Koordinator Asisten	:	Syifana Wulan Suci 1102213032
Divisi Praktikum	:	Alif Rizky Dimandani 1102213058
		Ghufron Andriansyah 1102213040
		Naufal Rayhan 1102213096
		Nurahma Tillah 1102210283
		Zaidan Fitra Baihaqi 1102210105
		Garry Nelson 1102223258

## **ASISTEN PRAKTIKUM LAB. ELITIA 2024/2025**

<b>NIM</b>	<b>Nama</b>	<b>Jabatan</b>	<b>No Hp</b>
1102213032	Syifana Wulan Suci	Koordinator Asisten	081383311306
1102213068	Bintang Mulya Prima	Wakil Koordinator Asisten	081377769082
1102213064	Wina Salsabila Lestari	Sekretaris I	08115414121
1102223133	Ni Kadek Cindy Sriastiti	Sekretaris II	0895394594943
1102210249	Shania Nurul Falah	Bendahara I	081280385831
1102220192	Daniel Parulian	Bendahara II	082297216942
1102213039	Akhmad Nabil Al Mu'ammor	Koordinator Bidang Discipline Committee	089677262980
1102213020	Ikhsan Habib	Koordinator Bidang Logistic	082288130403
1102210124	Muhammad Endy Rafif	Koordinator Bidang Media and Publication	081267231499
1102213058	Alif Rizky Dimandani	Koordinator Bidang Practice and Education	085791356223
1102213061	Mayco Ikhsan Hanafi	Koordinator Bidang Public Relation	08985478186
1102210135	Abdillah Nur Isnaini	Koordinator Bidang Research and Competition	085254648494
1102213042	Ade Surya Saputra	Anggota Bidang Discipline Committee	081293889515
1102213153	Ashel Izzah Ulfiyah	Anggota Bidang Discipline Committee	085862737530
1102213015	Dhion Razin Pavito	Anggota Bidang Discipline Committee	081226170666
1102213070	Florensa Anugrah Thosuly	Anggota Bidang Discipline Committee	082339443189
1102213024	Giovanni Salim	Anggota Bidang Discipline Committee	085781196247

1102213238	Bijak Agung Rizki	Anggota Bidang Logistic	081388209134
1102213317	Fallerina Ribka Angela	Anggota Bidang Logistic	082138161039
1102213209	Jeremy Pieter Hutahaean	Anggota Bidang Logistic	081220332856
1102213050	Hanif Prasetyo Herlambang	Anggota Bidang Media and Publication	082223753727
1102213133	Muhammad Adrian Maulana Nst	Anggota Bidang Media and Publication	08116720024
1102223258	Garry Nelson	Anggota Bidang Practice and Education	081261469886
1102213040	Ghufron Andriansyah	Anggota Bidang Practice and Education	081231717659
1102213096	Naufal Rayhan Ali Rahman	Anggota Bidang Practice and Education	081229280741
1102210283	Nurrahma Tillah Agustin	Anggota Bidang Practice and Education	083123277686
1102210105	Zaidan Fitra Baihaqi	Anggota Bidang Practice and Education	081280041558
1102210057	Dermawan Setiananda	Anggota Bidang Public Relation	085645215078
1102210291	Fadillah Syofyan	Anggota Bidang Public Relation	081276002552
1102213115	Malsa Zainuffitri	Anggota Bidang Public Relation	082136310090
1102213051	Muhammad Hafizhuda	Anggota Bidang Public Relation	082387400366
1102213172	Farhan Naufal	Anggota Bidang Research and Competition	087827575920
1102213167	Julia Monica Berliana	Anggota Bidang Research and Competition	085384283115
1102223024	Justin Fairman Tan	Anggota Bidang Research and Competition	081277991505
1102223051	Teguh Patriananda	Anggota Bidang Research and Competition	087885538877

## **LEMBAR REVISI**

Yang bertanda tangan di bawah ini:

Nama : Dr. Junartho Halomoan

NIP : 10820038

Jabatan : Kepala Lab Electronics and Industrial Automation (ELITIA)

Dengan ini menyatakan pelaksanaan Revisi Modul Pemrograman Perangkat IoT untuk Prodi S1 Teknik Elektro, telah dilaksanakan dengan penjelasan sebagai berikut :

No	Modul	Keterangan Revisi	Tanggal Revisi Terakhir
1	Modul 1	Menambahkan informasi tentang <i>Peer to Peer</i> (P2P).	Senin, 5 Agustus 2024
2	Modul 2	Menambahkan gambar skema komunikasi ESP32 agar dapat terkoneksi dengan Firebase.	Senin, 5 Agustus 2024
3	Modul 3	Merapihkan format penulisan.	Senin, 5 Agustus 2024
4	Modul 4	Merapihkan format penulisan.	Senin, 5 Agustus 2024

## **LEMBAR PERNYATAAN**

Yang bertanda tangan di bawah ini:

Nama : Dr. Junartha Halomoan

NIP : 10820038

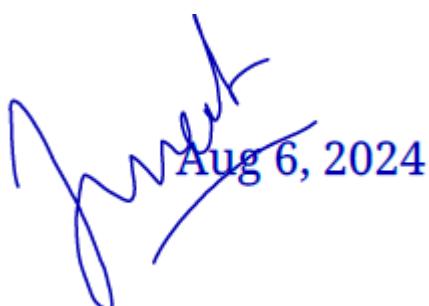
Jabatan : Kepala Lab Electronics and Industrial Automation (ELITIA)

Menerangkan dengan sesungguhnya bahwa modul praktikum ini telah di-review dan akan digunakan untuk pelaksanaan praktikum di Semester Ganjil Tahun Akademik 2024/2025 di Laboratorium Electronics and Industrial Automation Fakultas Teknik Elektro Universitas Telkom.

Bandung, 6 Agustus 2024

Kepala Laboratorium Automation  
Systems and Intelligent Control (ASIC)

Koordinator Asisten Laboratorium  
Electronics and Industrial Automation  
(ELITIA)



Aug 6, 2024



Dr. Junartha Halomoan  
NIP 10820038

Syifana Wulan Suci  
NIM 110213032

## **VISI DAN MISI**

### **FAKULTAS TEKNIK ELEKTRO**

#### **VISI:**

Menjadi fakultas berstandar internasional yang berperan aktif dalam pengembangan pendidikan, riset, dan entrepreneurship di bidang teknik elektro dan teknik fisika, berbasis teknologi informasi.

#### **MISI:**

1. Menyelenggarakan sistem pendidikan yang berstandar internasional di bidang teknik elektro dan teknik fisika berbasis teknologi informasi.
2. Menyelenggarakan, menyebarluaskan, dan memanfaatkan hasil-hasil riset berstandar internasional di bidang teknik elektro dan fisika.
3. Menyelenggarakan program entrepreneurship berbasis teknologi bidang teknik elektro dan teknik fisika di kalangan sivitas akademika untuk mendukung pembangunan ekonomi nasional.
4. Mengembangkan jejaring dengan perguruan tinggi dan industri terkemuka dalam dan luar negeri dalam rangka kerjasama pendidikan, riset, dan entrepreneurship.
5. Mengembangkan sumberdaya untuk mencapai keunggulan dalam pendidikan, riset, dan entrepreneurship.

## **VISI DAN MISI**

### **PROGRAM STUDI S1 TEKNIK ELEKTRO**

#### **VISI:**

Menjadi program studi berstandar internasional yang berperan aktif dalam pengembangan pendidikan, riset, dan kewirausahaan serta menghasilkan sarjana di bidang Sistem Elektronika, Sistem Kendali, atau Sistem Tertanam.

#### **MISI:**

1. Menyelenggarakan pendidikan berstandar internasional untuk menghasilkan lulusan yang menguasai ilmu pengetahuan dan teknologi Sistem Elektronika, Sistem Kendali atau Sistem Tertanam.
2. Mengembangkan dan menyebarluaskan ilmu pengetahuan dan teknologi Sistem Elektronika, Sistem Kendali, dan Sistem Tertanam yang diakui secara internasional dengan melibatkan mahasiswa secara aktif.
3. Memanfaatkan ilmu pengetahuan dan teknologi Sistem Elektronika, Sistem Kendali, dan Sistem Tertanam untuk pengembangan kewirausahaan di kalangan sivitas akademika dalam rangka mendukung pembangunan ekonomi nasional.
4. Mengembangkan jejaring dengan perguruan tinggi dan industri terkemuka dalam dan luar negeri dalam rangka kerja sama pendidikan, riset, dan kewirausahaan dalam bidang Sistem Elektronika, Sistem Kendali, atau Sistem Tertanam.
5. Mengembangkan sumber daya untuk mencapai keunggulan dalam pendidikan, riset, da kewirausahaan dalam bidang Sistem Elektronika, Sistem Kendali, atau Sistem Tertanam.

# **ATURAN PRAKTIKUM LABORATORIUM**

## **FAKULTAS TEKNIK ELEKTRO**

### **TELKOM UNIVERSITY**

Untuk memastikan bahwa setiap sesi praktikum berlangsung dengan lancar, ada beberapa aturan dan pedoman yang perlu diperhatikan oleh setiap praktikan. Kepatuhan terhadap aturan ini tidak hanya akan membantu menciptakan lingkungan yang kondusif dan aman, tetapi juga akan memastikan bahwa setiap praktikan dapat memaksimalkan kegiatan praktikum. Berikut adalah beberapa hal yang harus diperhatikan dan dipatuhi oleh setiap praktikan.

#### **Praktikan yang Tidak Diizinkan Masuk**

Praktikan tidak akan diizinkan masuk atau mengikuti praktikum jika :

- Terlambat maksimal 20 menit dari dimulainya suatu *shift*.
  - Shift 1 : 06.30 – 09.00 WIB
  - Shift 2 : 09.30 – 12.00 WIB
  - Shift 3 : 12.30 – 15.00 WIB
  - Shift 4 : 15.30 – 18.00 WIB
- Tidak terdaftar di seelabs.
- Selain itu, alasan seperti pakaian tetap diperbolehkan mengikuti praktikum dengan sanksi yang telah diberikan sesuai ketetapan lab bersangkutan.

#### **Pengajuan Praktikum Susulan**

Apabila tidak mengikuti praktikum, maka praktikan dapat mengajukan praktikum susulan dengan ketetapan yang dibuat oleh laboran, yaitu :

- Susulan dilakukan setelah praktikum selesai di waktu yang telah ditetapkan oleh laboran.

- Hanya praktikan yang disetujui oleh laboran yang diperbolehkan susulan (Sakit dengan bukti surat sakit dokter, rawat inap, kematian keluarga inti, dan dispensasi dari BK).
- Pengajuan susulan dilakukan praktikan melalui OA Seelabs (@jit0659i) dengan melampirkan bukti.

---

## PANDUAN UMUM KESELAMATAN DAN PENGGUNAAN PERALATAN LABORATORIUM

### Keselamatan

Untuk mewujudkan praktikum yang aman diperlukan partisipasi seluruh praktikan dan asisten pada kegiatan praktikum yang bersangkutan. Dengan demikian, kepatuhan setiap praktikan terhadap uraian panduan pada bagian ini akan sangat membantu mewujudkan praktikum yang aman.

### Bahaya Listrik

Perhatikan dan pelajari tempat-tempat sumber listrik (*stop-kontak* dan *circuit breaker*) serta cara menyala-matikannya. Jika melihat ada kerusakan yang dapat menimbulkan bahaya, segera lapor pada asisten.

- Hindari daerah atau benda yang berpotensi menimbulkan bahaya listrik (sengatan listrik/strum) secara tidak disengaja, misalnya kabel jala-jala yang terkelupas, dll.
- Tidak melakukan sesuatu yang dapat menimbulkan bahaya listrik pada diri sendiri atau orang lain.
- Keringkan bagian tubuh yang basah karena, misalnya, keringat atau sisa air wudhu.
- Selalu waspada terhadap bahaya listrik pada setiap aktivitas praktikum.

Kecelakaan akibat bahaya listrik yang sering terjadi adalah tersengat arus listrik. Berikut ini adalah hal-hal yang harus dilakukan oleh praktikan jika hal tersebut terjadi :

- Jangan panik;
- Matikan semua peralatan elektronik dan sumber listrik di meja masing-masing dan di meja orang yang tersengat arus listrik;

- Bantu orang yang tersengat arus listrik untuk melepaskan diri dari sumber listrik; dan
- Beritahukan dan minta bantuan pada asisten, praktikan lain, serta orang di sekitar tentang terjadinya kecelakaan akibat bahaya listrik.

## Bahaya Api atau Panas Berlebih

Dilarang membawa benda-benda yang mudah terbakar (korek api, gas, dll.) ke dalam ruangan praktikum bila tidak diisyaratkan dalam modul praktikum.

- Jangan melakukan sesuatu yang dapat menimbulkan api, percikan api atau panas yang berlebihan.
- Jangan melakukan sesuatu yang dapat menimbulkan bahaya api atau panas berlebih pada diri sendiri dan orang lain.
- Selalu waspada terhadap bahaya api atau panas berlebih pada setiap aktivitas praktikum.

Berikut ini adalah hal-hal yang harus diikuti praktikan jika menghadapi bahaya api atau panas berlebih :

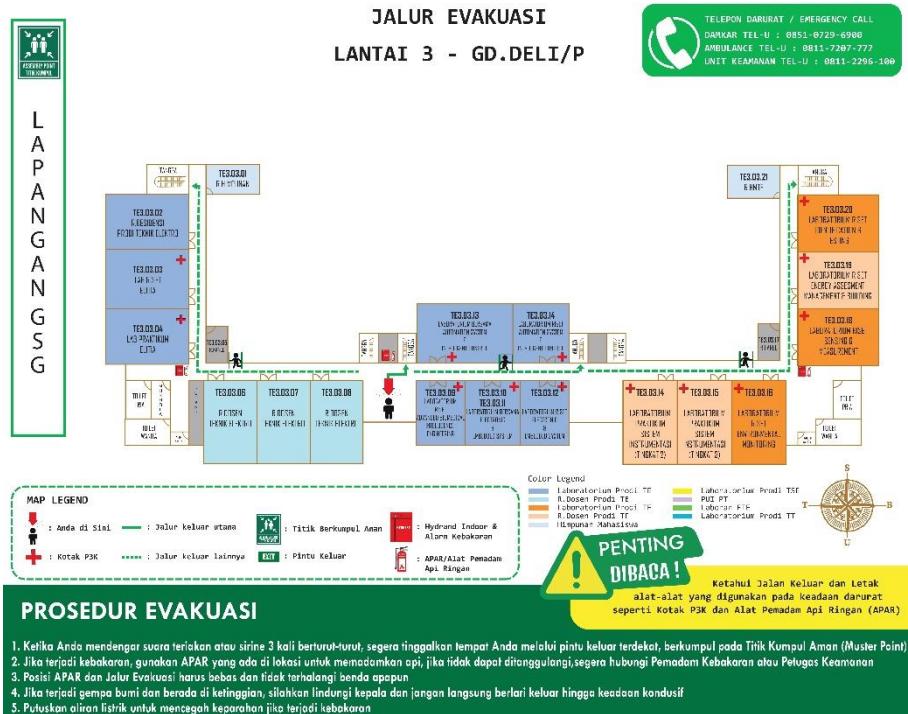
- Jangan panik;
- Beritahukan dan minta bantuan pada asisten, praktikan lain, serta orang di sekitar tentang terjadinya bahaya api atau panas berlebih;
- Matikan semua peralatan elektronik dan sumber listrik ruangan praktikum;
- Menjauh dari ruang praktikum.

## Bahaya Lain

Untuk menghindari terjadinya hal-hal yang tidak diinginkan selama pelaksanaan praktikum, perhatikan juga hal-hal berikut :

- Jangan membawa benda tajam (pisau, gunting, dan sejenisnya) ke ruangan praktikum bila tidak diperlukan untuk pelaksanaan percobaan.
- Jangan memakai perhiasan dari logam, misalnya cincin, kalung, gelang, dll.
- Hindari daerah, benda, atau logam yang memiliki bagian tajam dan dapat melukai.
- Hindari melakukan sesuatu yang dapat menimbulkan luka pada diri sendiri atau orang lain, seperti bermain-main saat praktikum berlangsung.

## DENAHA EVAKUASI LANTAI 3 - GEDUNG DELI/P



## **DAFTAR ISI**

<b>TIM PENYUSUN MODUL PRAKTIKUM</b>	i
<b>ASISTEN PRAKTIKUM LAB. ELITIA 2024/2025</b>	ii
<b>LEMBAR REVISI</b>	iv
<b>LEMBAR PERNYATAAN</b>	v
<b>VISI DAN MISI FAKULTAS TEKNIK ELEKTRO</b>	vi
<b>VISI DAN MISI PROGRAM STUDI S1 TEKNIK ELEKTRO</b>	vii
<b>ATURAN PRAKTIKUM LABORATORIUM FAKULTAS TEKNIK ELEKTRO TELKOM UNIVERSITY</b>	viii
<b>PANDUAN UMUM KESELAMATAN DAN PENGGUNAAN PERALATAN LABORATORIUM</b>	x
<b>DENAH EVAKUASI LANTAI 3 - GEDUNG DELI/P</b>	xiii
<b>DAFTAR ISI</b>	xiv
<i>Introduction to Internet of Things (IoT)</i>	1
A.    Apa itu <i>Internet of Things (IoT)</i> ? .....	1
B. <i>Architecture Layer</i> pada IoT .....	1
a.    Physical Layer/Perception Layer .....	2
b.    Network Layer .....	2
c.    Application Layer.....	6
<b>MODUL 1 KONEKTIVITAS BLUETOOTH</b>	7
1.1    Bluetooth.....	8
1.2    Jenis-Jenis Bluetooth .....	9
1.2.1        Bluetooth 1.0 .....	9
1.2.2        Bluetooth 2.0 .....	10

1.2.3	Bluetooth 3.0 .....	11
1.2.4	Bluetooth 4.0 (Bluetooth Low Energy) .....	11
1.2.5	Bluetooth 5.0 .....	12
1.3	Alur Komunikasi Bluetooth.....	13
1.3.1	Protokol Bluetooth.....	13
1.4	Implementasi <i>peer-to-peer</i> Bluetooth Menggunakan ESP32 .....	16
1.4.1	Master .....	16
1.4.2	Slave.....	17
<b>MODUL 2 HTTP DENGAN FIREBASE DAN KODULAR</b>		<b>20</b>
2. 1	<i>Hypertext Transfer Protocol</i> (HTTP).....	21
2. 1. 1	CRUD .....	22
2. 1. 2	Metode Komunikasi Protokol HTTP.....	23
2. 2	Firebase.....	25
2. 2. 1	Fitur Firebase.....	25
2. 2. 2	Firebase Realtime Database (RTDB) .....	28
2. 2. 3	Implementasi Firebase RTDB dengan ESP32 dan Kodular .....	39
2. 2. 4	Pembuatan Aplikasi .....	52
<b>MODUL 3 MQTT DENGAN MICROPYTHON MENGGUNAKAN RASPBERRY PICO W DAN ADAFRUIT IO MQTT BROKER</b>		<b>55</b>
3. 1	<i>Message Queuing Telemetry Transport</i> (MQTT).....	56
3. 2	Kelebihan MQTT .....	56
3. 3	Alur Komunikasi MQTT .....	56
3. 3. 1	Publish-Subscribe .....	57
3. 3. 2	Topics.....	57

3. 3. 3	Broker .....	57
3. 4	Adafruit IO.....	58
3. 5	MicroPython.....	58
3. 6	Raspberry Pi Pico W .....	60
3. 7	Implementasi Protokol MQTT menggunakan Raspberry Pi Pico W dan Adafruit IO MQTT Broker.....	61
3. 7. 1	Setup Adafruit IO MQTT Broker .....	61
3. 7. 2	Setup Raspberry Pi Pico W untuk MQTT .....	64
3. 7. 3	<i>Source code</i> implementasi protokol MQTT menggunakan Micropython .....	65
<b>MODUL 4 KOMUNIKASI LONG RANGE (LoRa)</b>		<b>71</b>
4.1	LoRa .....	72
4.2	Tipe <i>Class LoRa</i> .....	73
4.3	Alur Komunikasi LoRa.....	74
4.4	Implementasi <i>Peer-to-Peer</i> Menggunakan Cosmic LoRa Aurora V2 .....	75
4.4.1	Cosmic LoRa Aurora V2 .....	75
4.4.2	LoRa Library by Sandeep Mistry .....	76
4.4.3	Source Code LoRa peer-to-peer .....	76
<b>LAMPIRAN</b>		<b>82</b>
<b>LAMPIRAN PINOUT DIAGRAM MIKROKONTROLER</b>		<b>83</b>
	ESP32 .....	83
	Raspberry Pi Pico W.....	83
	Cosmic LoRa Aurora V2 .....	84
<b>LAMPIRAN MODUL 2</b>		<b>85</b>

Lampiran 1. <i>Firebase with ESP32 Source Code</i> .....	85
<b>LAMPIRAN MODUL 3</b>	<b>89</b>
Lampiran 1. <i>Source code</i> untuk publish MQTT ke Adafruit IO .....	89
Lampiran 2. <i>Source code</i> untuk subscribe MQTT ke Adafruit IO .....	90
<b>LAMPIRAN MODUL 4</b>	<b>93</b>
Lampiran 1. <i>Source code transmitter LoRa</i> .....	93
Lampiran 2. <i>Source code Receiver LoRa</i> .....	94

## ***Introduction to Internet of Things (IoT)***

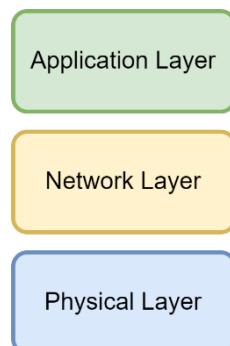
### **A. Apa itu *Internet of Things (IoT)*?**

*Internet of Things (IoT)* merupakan suatu terobosan baru dalam dunia internet dan servis. IoT sendiri tidak memiliki definisi khusus, namun memiliki konsep bahwa segala sesuatu dapat berkomunikasi satu sama lain kapanpun dan di manapun karena adanya perantara komunikasi seperti internet. Dengan maksud tersebut, maka kita dapat membuat suatu sistem IoT selama antar perangkat-perangkat, perangkat-pengguna, pengguna-pengguna dapat berkomunikasi satu sama lain dan tentunya dengan protokol komunikasi tertentu.

Beraktivitas dalam sehari-hari dapat lebih mudah dengan adanya sistem IoT. Ketika kita bangun tidur, IoT dapat membantu kita dengan membuat secangkir kopi secara otomatis hanya dengan aplikasi dari android, ketika akan beraktivitas di luar rumah IoT dapat membantu memastikan keamanan rumah dan tidak takut dengan peliharaan di rumah yang akan kelaparan karena kita dapat mengontrolnya dari luar. Ketika kembali ke rumah dan akan tidur lampu bisa secara otomatis dimatikan.

### **B. Architecture Layer pada IoT**

Ada beberapa lapisan arsitektur dalam sistem IoT atau biasa disebut dengan *IoT Layer*. *IoT Layer* ini terdiri dari 3 *layer*, yaitu *Physical*, *Network*, dan *Application* seperti yang ditunjukkan pada **Gambar 1**.



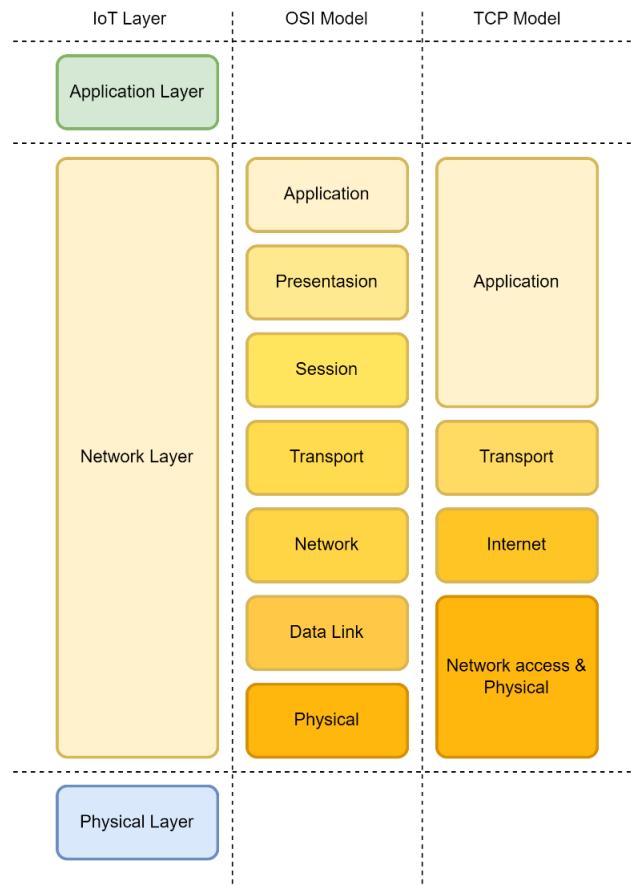
**Gambar 1** *Architecture layer* pada IoT

### a. Physical Layer/Perception Layer

*Physical Layer* merupakan *layer* paling dasar dalam sistem IoT. Sistem ini biasanya berisi sensor untuk proses akuisisi data atau actuator untuk melakukan suatu aksi maupun sistem kontrol. Sebagai contoh, pada sistem *smart house* ada sensor yang berguna untuk mengawasi suhu, keamanan rumah, dan sebagainya. Sedangkan untuk aktuator digunakan untuk mengunci pintu, menyalakan rumah, mengontrol suhu AC dan lainnya.

### b. Network Layer

Dalam sistem IoT, *Network Layer* merupakan kunci penting yang dapat menjadikan suatu sistem disebut “IoT”. Layer ini merupakan proses komunikasi data atau transfer data dari *physical layer* menuju *application layer*. Dalam domain *network layer*, ada model komunikasi yang umum digunakan, yaitu model OSI layer dan model TCP Layer yang terdapat pada **Gambar 2**.



**Gambar 2** Perbandingan layer pada IoT, OSI, dan TCP

### i. OSI Model

#### a) Physical

*Physical layer* adalah *layer* paling bawah pada model OSI. *Layer* ini menentukan sifat fisik dari media yang digunakan untuk mengirimkan data antar perangkat. Contohnya, tingkat tegangan, jarak transmisi maksimum, konektor fisik, dan sebagainya. Pada layer ini, bit digital diubah menjadi sinyal listrik untuk koneksi kabel dan menjadi sinyal radio untuk transmisi nirkabel.

#### b) Data link

*Layer* ini memfasilitasi komunikasi dan transfer data antar *node* (contohnya, PC ke switch, switch ke router, dan router ke router). Alamat fisik

(MAC Address) ditambahkan ke data pada lapisan ini, termasuk alamat MAC sumber dan tujuan.

c) **Network**

*Network layer* menghubungkan host akhir pada jaringan yang berbeda (yaitu, di luar LAN Anda). Lapisan ini menangani alamat logis (*Logical address*) menggunakan alamat IP.

d) **Transport**

Untuk memastikan bahwa tidak ada data yang hilang, *transport layer* digunakan untuk penanganan kesalahan dan pengecekan urutan. Lapisan ini juga menyediakan komunikasi dari host ke host, juga dikenal sebagai komunikasi end-to-end.

e) **Session**

*Layer* ini mengontrol komunikasi dari host ke host (*session*). Ini membuat dan menghapus koneksi antara aplikasi lokal (seperti browser web Anda) dan aplikasi remote (misalnya, YouTube).

f) **Presentation**

*Layer* ini memformat data agar dapat dimengerti oleh aplikasi penerima. *Layer* ini juga dapat **mengenkripsi data saat dikirim** dan **mendekripsi saat diterima**, memastikan bahwa hanya penerima yang dimaksud yang dapat membacanya.

g) **Application**

*Layer* ini adalah terdekat dengan *end-user*. *Layer* ini merupakan tempat aplikasi dan pengguna berkomunikasi. Untuk komunikasi antara browser web dan server web, *application-specific protocol* seperti HTTP (Hyper Text Transfer Protocol) digunakan pada lapisan ini.

ii. **TCP Model**

Model TCP/IP adalah model jaringan yang praktis dan banyak diimplementasikan, menjadi dasar bagi internet modern. Dinamai setelah dua protokol paling terkenal, yaitu Transmission Control Protocol (TCP) dan Internet

Protocol (IP). Berbeda dengan model OSI yang memiliki tujuh lapisan, model TCP/IP menyederhanakan arsitektur menjadi empat lapisan, menjadikannya lebih cocok untuk implementasi dan pemecahan masalah di dunia nyata

a) **Network access & physical atau Link (Ethernet, Wifi, Bluetooth, MAC, VLAN, dsb.)**

*Link layer* terletak di dasar model TCP/IP, berfungsi sebagai antarmuka antara perangkat keras jaringan dan tumpukan protokol yang lain. Menggabungkan fungsi-fungsi dari *Physical layer* dan Data Link OSI, *layer* ini menangani koneksi fisik antara perangkat, menentukan karakteristik perangkat keras seperti kabel dan konektor. Selain itu, *Link layer* mengelola pembingkaian data dan pengalaman pada tingkat jaringan lokal. Lapisan ini sangat penting untuk membentuk saluran komunikasi yang handal dalam jaringan langsung, menggunakan protokol seperti Ethernet dan Wi-Fi untuk memfasilitasi pertukaran data yang lancar antara perangkat yang terhubung.

b) **Internet (IP, NAT, ARP, dsb.)**

Di atas Lapisan Link terletak Lapisan Internet, bertanggung jawab atas pengalaman logis, routing, dan fragmentasi data saat melintasi berbagai jaringan. Protokol Internet (IP) memainkan peran sentral dalam lapisan ini, memberikan alamat unik kepada perangkat dan memungkinkan pengarahan efisien paket data di seluruh jaringan yang terhubung. Lapisan Internet memastikan bahwa data mencapai tujuan yang dimaksud dengan mengelola skema pengalaman dan memfasilitasi komunikasi antara perangkat pada jaringan yang berbeda. Pada intinya, ini membentuk tulang punggung dari interkoneksi global yang mendefinisikan internet modern.

c) **Transport (TCP, UDP, RTP, SCTP, DCCP)**

Lapisan Transport mengurus komunikasi end-to-end, memastikan pengiriman data yang handal antar perangkat. Pada lapisan ini, dua protokol yang prominent digunakan. Transmission Control Protocol (TCP) menawarkan pendekatan berorientasi koneksi, menjamin transmisi data yang teratur dan terverifikasi kesalahan. Di sisi lain, User Datagram Protocol (UDP) menyediakan

alternatif yang lebih cepat namun tanpa koneksi, cocok untuk aplikasi di mana komunikasi real-time dan kecepatan lebih diutamakan daripada kehandalan. Lapisan Transport bertindak sebagai perantara penting antara lapisan-lapisan di atasnya dan jaringan yang mendasarinya, mengelola mekanisme pengendalian aliran dan pemulihan kesalahan yang diperlukan untuk transmisi data yang kuat.

d) **Application (HTTP, DNS, SSH, FTP, SSL, dsb.)**

Di bagian puncak model TCP/IP terdapat Lapisan Aplikasi, berfungsi sebagai antarmuka antara jaringan dan aplikasi pengguna akhir. Lapisan ini mencakup berbagai protokol, masing-masing disesuaikan untuk layanan dan aplikasi tertentu. Contohnya adalah Hypertext Transfer Protocol (HTTP) untuk penjelajahan web, File Transfer Protocol (FTP) untuk pertukaran file, dan Simple Mail Transfer Protocol (SMTP) untuk komunikasi email. Lapisan Aplikasi bertanggung jawab untuk menyediakan layanan jaringan secara langsung kepada pengguna, memungkinkan mereka berinteraksi dan menggunakan berbagai aplikasi yang mendefinisikan pengalaman internet modern. Pada intinya, lapisan ini melengkapi model TCP/IP dengan memfasilitasi komunikasi tanpa hambatan antara aplikasi pengguna dan infrastruktur jaringan yang mendasarinya.

c. **Application Layer**

*Application layer* di sistem IoT memiliki fungsi untuk menyediakan layanan kepada para pengguna seperti membaca, menganalisis, mengontrol data pada lapisan *physical*. Layanan tersebut dapat diberikan kepada pengguna dalam bentuk *graphical user interface* (GUI) seperti aplikasi mobile maupun web, atau dengan menggunakan tampilan lainnya yang dapat memudahkan pengguna dalam mendapatkan layanan-layanan tersebut

# MODUL 1

## KONEKTIVITAS BLUETOOTH

---

### TUJUAN

- 1) Mengetahui teknologi Bluetooth.
- 2) Memahami proses komunikasi antar perangkat menggunakan Bluetooth.
- 3) Mengimplementasikan Bluetooth untuk IoT.

### ***REQUIREMENT:***

- 1) Arduino IDE/PlatformIO,
- 2) ESP32/ESP8266
- 3) Sensor (opsional),
- 4) Aktuator (Opsional).

Konektivitas bluetooth merupakan salah satu contoh dari *Peer to Peer* (P2P). Dimana *Peer to Peer* (P2P) adalah arsitektur jaringan terdesentralisasi di mana peserta, yang disebut peer, berinteraksi langsung satu sama lain tanpa memerlukan otoritas atau server pusat. Jika melihat pengertian diatas dapat didapati bahwasannya selain bluetooth bisa juga menggunakan Wifi untuk menjadi sarana P2P. Dimana memiliki dua buah perangkat yang saling terhubung dimana salah satu disebut *transmitter* dan satu lagi disebut *receiver*.

### 1.1 Bluetooth

Bluetooth adalah sebuah media komunikasi yang dapat digunakan untuk menghubungkan sebuah perangkat komunikasi dengan perangkat komunikasi lainnya tanpa menggunakan kabel. Bluetooth biasa digunakan pada handphone, laptop, komputer dan lain-lain.



Bluetooth ditemukan oleh Jacobus Cornelis Haartsen pada tahun 1994 dan dikembangkan oleh para tim insinyur di perusahaan Ericsson, yaitu perusahaan telekomunikasi asal Swedia. Nama Bluetooth diambil dari nama seorang raja dari Denmark yang terkenal karena sudah menyatukan suku-suku di Denmark dan Norway yaitu Harald "Blåtand" Gormsson atau Harald Bluetooth.



Bluetooth memiliki fungsi untuk mengirim dan menerima data, bisa menghubungkan perangkat elektronik contohnya seperti headset, speaker, dan keyboard tanpa kabel, dan *tethering* internet.

Selain itu, Bluetooth juga beroprasi pada pita frekuensi 2.4 GHz (antara 2.402 GHz sampai 2.480 GHz) dengan menggunakan sebuah *frequency hopping tranceiver* yang mampu menyediakan layanan komunikasi data dan suara secara real-time antara host-host Bluetooth dengan jarak jangkauan layanan yang terbatas.

## 1.2 Jenis-Jenis Bluetooth

### 1.2.1 Bluetooth 1.0

Pada tahun 1999, versi pertama dari Bluetooth ini mengalami banyak masalah dan produsen mengalami kesulitan untuk menciptakan sebuah produk yang bisa saling berhubungan antara satu sama lain dengan benar. Versi 1.0 dan versi perbaikannya 1.0B bisa dibilang mengalami kegagalan.

Lalu pada tahun 2001 Bluetooth 1.1 pun diciptakan dengan perbaikan dari versi pendahulunya. Selain itu, Bluetooth 1.1 juga telah memiliki standar yang dinamakan IEEE 802.15.1-2002.



Pada tahun 2003, Bluetooth 1.2 diciptakan. Bluetooth versi ini memiliki kecepatan hingga 721 Kbit per detik. Versi ini juga dapat menemukan koneksi Bluetooth lain lebih cepat. Namun, versi ini tidak bisa digunakan dengan perangkat yang menggunakan Bluetooth 1.1



### 1.2.2 Bluetooth 2.0

Pada tahun 2004, Bluetooth 2.0 rilis dan tidak kompatibel dengan Bluetooth 1.2 karena pada Bluetooth 2.0 menggunakan perkembangan baru yang bernama Enhanced Data Rate(EDR) yang memiliki fungsi untuk mempercepat transfer data dan menghemat konsumsi daya. Bluetooth 2.0 memiliki kecepatan transfer hingga 2.1 Mbps



Tiga tahun selanjutnya, Bluetooth mengeluarkan versi terbarunya yaitu Bluetooth 2.1. Bluetooth memiliki teknologi baru, yaitu *Secure Simple Pairing* (SSP) untuk membuat pairing lebih cepat dan aman. Bluetooth juga memiliki teknologi baru yang lainnya yaitu *Extended Inquiry Response* (EIR), yang membantu pengguna untuk

mengetahui informasi perangkat yang akan dihubungkan sebelum terkoneksi dan konsumsi daya lebih hemat.



### 1.2.3 Bluetooth 3.0

Pada tahun 2009, Bluetooth mengeluarkan versi Bluetooth 3.0 + High Speed (HS) menggunakan teknologi link 802.11. Bluetooth ini memiliki kecepatan hingga menyentuh 24 Mbps. Pada versi ini link Bluetooth hanya digunakan untuk pairing dan pembentukan jalur akses data saja. Untuk pengiriman dan penerimaan data, menggunakan link wireless 802.11. Berkat fitur baru yaitu Alternate MAC/PHY(AMP) yang memberikan dukungan untuk link 802.11 untuk transfer data yang lebih cepat.

### 1.2.4 Bluetooth 4.0 (Bluetooth Low Energy)

Pada tahun 2010, Bluetooth mengeluarkan versi Bluetooth 4.0. Bluetooth ini memiliki konsumsi daya yang lebih rendah dari sebelumnya sehingga dikenal dengan Bluetooth Low Energy. Versi ini ditenagai oleh baterai sel koin yang membuat konsumsi daya menjadi perubahan yang signifikan. Pada versi ini juga, memungkinkan prangkat untuk “highly integrated and compact”, yaitu kemampuan mencari atau membaca perangkat lain lebih mudah dan cepat. Transfer data pun memiliki sistem keamanan lebih baik. Bluetooth ini memiliki kecepatan 1 Mbps dan jangkauan area yang luas hingga 100 meter.



Pada tahun 2013, Bluetooth memiliki versi Bluetooth 4.1. Versi ini dapat mengelola daya lebih baik dari versi sebelumnya. Versi ini dapat secara otomatis menghidupkan dan mematikan koneksi Bluetooth sesuai dengan power plan. Mentransfer data menjadi lebih efisien dan mampu berdampingan lebih baik dengan frekuensi LTE dibandingkan versi sebelumnya yang sering kali terganggu.

Selang satu tahun, Bluetooth kembali mengeluarkan versi terbarunya, yaitu Bluetooth 4.2. Pada versi ini, Bluetooth meningkatkan kecepatan dan keamanan lebih tinggi. Versi ini memiliki kecepatan unduh 2,6 kali lebih cepat dibandingkan versi sebelumnya.

### 1.2.5 Bluetooth 5.0

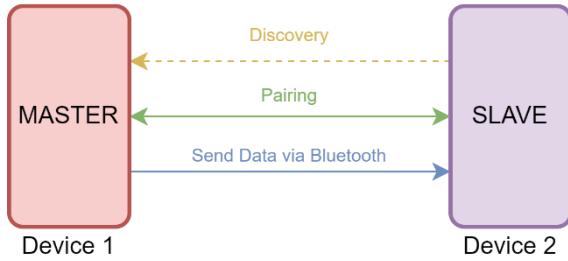
Pada tahun 2016, Bluetooth mengeluarkan versi 5.0 nya. Pada versi ini, Bluetooth dirancang khusus untuk menyediakan komunikasi yang aman dan tanpa gangguan. Pada versi ini pula tersedia fitur-fitur yang mendukung teknologi Internet of Things atau IoT.

Tiga fitur utama yang diperkenalkan adalah :

- 2x kecepatannya
- 4x jangkauannya
- 8x kapasitas iklan

Pada versi ini Bluetooth memiliki 2 mode kecepatan yaitu pada kecepatan standar yaitu 1 Mbps dan mode 2 Mbps. Tentu dengan mode 2 Mbps pengiriman data akan menjadi lebih cepat, akan tetapi jangkauannya akan dikurangi dibandingkan dengan mode standar.

### 1.3 Alur Komunikasi Bluetooth



Langkah-langkah Bluetooth bekerja seperti yang ditunjukkan pada gambar di atas, penjelasannya adalah sebagai berikut:

- **Discovery**

Perangkat Bluetooth akan melakukan proses pencarian discovery untuk perangkat-perangkat lain yang berada pada jangkauan. Pada tahap ini, **perangkat akan mengirimkan sinyal Inquiry untuk mencari perangkat yang tersedia.**

- **Pairing**

Setelah menemukan perangkat yang dituju, proses pairing akan dimulai. **Pairing adalah proses saling mengenal dan membangun koneksi aman antara perangkat-perangkat yang terhubung.**

- **Koneksi**

Setelah berhasil dipasangkan, perangkat akan membangun koneksi Bluetooth yang aktif. Ini melibatkan pembentukan saluran komunikasi antara perangkat pengirim dan penerima. Dalam koneksi Bluetooth salah satu perangkat akan berperan sebagai **Master (Pemancar)** dan yang lainnya sebagai **Slave (Penerima)**

#### 1.3.1 Protokol Bluetooth

Bluetooth *Special Interest Group* (SIG) telah mengembangkan spesifikasi Bluetooth yang berisi tentang protokol yang akan digunakan dalam teknologi Bluetooth ini. Protokol dasar Bluetooth adalah Bluetooth Radio, Baseband dan *Link Manager*

*Protocol* (LMP) yang disebut protokol inti. Sedangkan protokol yang ada di atasnya adalah protokol-protokol terapan yang dapat diadaptasikan pada arsitektur protocol Bluetooth dan telah dikembangkan oleh organisasi lain seperti ETSI. Radio, baseband dan LMP ekivalen dengan lapis fisik dan data link pada lapis protokol OSI.

Stack protokol Bluetooth dapat dibagi ke dalam empat layer sesuai dengan tujuannya. Berikut protokol-protokol dalam layer-layer di dalam stack protokol Bluetooth yang tertera pada Tabel dibawah ini :

**Tabel 1.1** Layer Protokol dan stack Bluetooth

Layer Protokol	Protokol di Stack
<i>Bluetooth Core Protocols</i>	Baseband, LMP, L2CAP, SDP
<i>Cable Replacement Protocols</i>	RFCOMM
<i>Telephony Control Protocols</i>	TCS Binary, AT-commands
<i>Adopted Protocols</i>	PPP, UDP/TCP/IP, OBEX, WAP, vCard, vCal, IrMC, WAE

Selain itu juga terdapat *Host Controller Interface* (HCI) yang melakukan perintah-perintah yang menghubungkan control baseband, pengaturan hubungan yang dibangun, dan akses ke hardware dan pengaturan register.

- ***Baseband***

Lapis yang memungkinkan hubungan RF terjadi antara beberapa unit Bluetooth membentuk piconet. Sistem RF dari Bluetooth ini menggunakan frekuensi-hopping-spread spectrum yang mengirimkan data dalam bentuk paket pada time slot dan frekuensi yang telah ditentukan, lapis ini melakukan prosedur pemeriksaan dan paging untuk sinkronisasi transmisi frekuensi hopping dan clock dari perangkat Bluetooth yang berbeda.

- ***Link Manager Protocol (LMP)***

*Link manager protocol* adalah perespon, menset dan menghubungkan kanal antara perangkat keras. Protokol ini terdapat meningkatkan performa keamanan seperti membentuk autentifikasi, pertukaran, dan verifikasi dan kunci enkripsi dan negosiasi ukuran paket baseband.

- ***Logical Link Control and Adaptation Protocol (L2CAP)***

Paket L2CAP membawa muatan yang penting yang dibawa ke layer protokol yang lebih tinggi.

- ***Service Discovery Protocol (SDP)***

Protokol ini digunakan untuk memberikan informasi device, pelayanan diperbolehkan untuk mengakses device yang berfungsi.

- ***Cable Replacement Protocol (RFCOMM)***

RFCOMM adalah emulasi jalur serial.

- ***Telephony Control Protocol***

The Telephony Control - Binary (TCS Binary) and Telephony Control – AT Commands digunakan untuk menyusun percakapan dan data antara device dan mengontrol mobile phone dan modem.

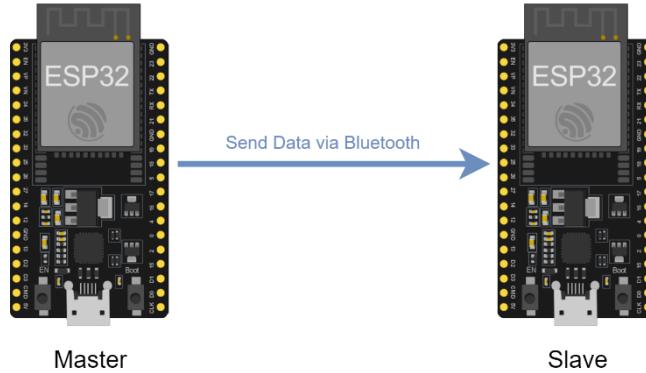
- ***Adopted Protocols***

Bluetooth juga mensupport protokol PPP, TCP/UDP/IP, OBEX dan WAP untuk memaksimalkan interoperabilitasnya.

- ***Radio Frequency (RF)***

Adalah lapis terendah dari spesifikasi Bluetooth. Unit RF merupakan sebuah transceiver yang memfasilitasi hubungan wireless antar perangkat Bluetooth yang beroperasi pada International Scientific and Medical (ISM) band dengan frekuensi 2,4 GHz. ISM band bekerja dengan frequencyhopping, dan pembagiannya dibuat dalam 79 hop dengan spasi 1 MHz.

## 1.4 Implementasi *peer-to-peer* Bluetooth Menggunakan ESP32



Dalam implementasi *peer-to-peer* Bluetooth ini, master akan mengirimkan data ke slave menggunakan mikrokontroler ESP32.

### 1.4.1 Master

*Master* merupakan bagian yang akan mencari koneksi kepada *Slave* dan akan mengirimkan data kepada mereka.

Inisialisasi *Source code* untuk *Master* pada implementasi *peer-to-peer* ESP32

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
String Name = "ESP32-BT-Slave";
bool connected;
int Value = 0;
```

Dalam menginisialisasi P2P ESP32, kita menggunakan library BluetoothSerial.h agar bisa mengakses kode-kode yang mendukung fungsi Bluetooth. Setelah itu, siapkan variable-variabel seperti variable untuk mengakses fungsi Bluetooth seperti variable SerialBT yang dapat mengakses BluetoothSerial, variable String Name yang digunakan untuk mencari nama *device* yang dituju yaitu ESP32-BT-Slave, bool untuk konektifitas dan integer Value untuk digunakan sebagai penyimpan nilai yang akan dikirimkan ke *Slave*.

Setup pada *Source code Master* pada implementasi *peer-to-peer* ESP32

```
void setup() {
    Serial.begin(115200);
    SerialBT.begin("ESP32Master", true);
    Serial.println("The device started in master mode, make sure remote BT
device is on!");
```

```
connected = SerialBT.connect(Name);

if(connected) {
    Serial.println("Connected Succesfully!");
}
else {
    while(!SerialBT.connected(10000)) {
        Serial.println("Failed to connect. Make sure remote device is available
and in range, then restart app.");
    }
}
SerialBT.connect();
}
```

Pada **setup()**, kita menggunakan **SerialBT.begin("ESP32Master", true)** untuk menentukan nama yang akan terlihat oleh prangkat lain. Untuk **connected = SerialBT.connect(Name)** digunakan sebagai koneksi kepada Bluetooth yang dituju dan indikator untuk terkoneksi atau tidak. Dan dilanjutkan dengan menghubungkan dengan Bluetooth pada **SerialBT.connect();**

#### *Loop pada Source code Master pada implementasi peer-to-peer ESP32*

```
void loop() {
    if(Value<=10){
        SerialBT.println(Value);
        Serial.println(Value);
        Value++;
        delay(1000);
    }
    else {
        Value = Value - 10;
    }
}
```

Pada tahap perulangan, kita akan mencoba mengirimkan kepada *Slave* nilai dari variabel *Value* hingga bernilai 10. Kita menggunakan **SerialBT.println(Value)** yang dimana itu digunakan untuk mengirimkan nilai kepada *Slave* berdasarkan nilai dari variable *Value*.

#### 1.4.2 Slave

Slave merupakan bagian yang akan menerima komunikasi dari Master.

**Inisialisasi Source code untuk Slave pada implementasi peer-to-peer ESP32**

```
#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to enable it
#endif

BluetoothSerial SerialBT;
int Value=0;
String ValueSlave;
int ValueSlaveInt;
String Name = "ESP32-BT-Slave";
```

Pada proses inisialisasi pada *Slave*, kita kembali menggunakan library **BluetoothSerial.h**. Untuk `#if !defined(CONFIG_BT_ENABLED)` untuk memeriksa apakah Bluetooth telah memungkinkan untuk terhubung atau tidak. Lalu inisialkan kembali BluetoothSerial pada variable **SerialBT**. Dan siapkan variable variable yang akan digunakan seperti `int Value`, `String ValueSlave` yang akan digunakan sebagai data yang diterima berupa string, dan `int ValueSlaveInt` yang akan digunakan untuk menyimpan nilai int yang telah diubah dari string ke int. Lalu, `String Name = "ESP32-BT-Slave"` yang merupakan inisial untuk nama perangkat kita.

**Setup pada Source code Slave pada implementasi peer-to-peer ESP32**

```
void setup() {
    Serial.begin(115200);
    SerialBT.begin(Name); //Bluetooth device name
    Serial.println("The device started, now you can pair it with bluetooth!");
}
```

Pada `setup`, kita akan mencoba menyambungkan koneksi dengan **SerialBT.begin(Name)** dan menentukan nama yang akan terlihat oleh perangkat lain berdasarkan nilai dari variabel Name.

**Loop pada Source code Slave pada implementasi peer-to-peer ESP32**

```
void loop() {
    if(SerialBT.available()){
        ValueSlave=SerialBT.readStringUntil('\n');
        ValueSlaveInt=ValueSlave.toInt();
        Serial.print("Data Dari Master : ");
        Serial.println(ValueSlaveInt);
    }
}
```

Pada perulangan. Kita mencoba untuk memeriksa apakah Bluetooth sudah terhubung dengan perangkat master atau tidak dengan **if(SerialBT.available())**. Lalu variabel ValueSlave akan membaca data string hingga akhir kalimat sebelum baris baru. dari Master menggunakan **SerialBT.readStringUntil('\n')**. Lalu, kita akan mengubah nilai string yang dibaca dari master menjadi int yang akan disimpan ke variabel ValueSlaveInt menggunakan **ValueSlaveInt=ValueSlave.toInt()** lalu akan ditampilkan.

```
Data Dari Master : 1  
Data Dari Master : 2  
Data Dari Master : 3  
Data Dari Master : 4  
Data Dari Master : 5  
Data Dari Master : 6  
Data Dari Master : 7  
Data Dari Master : 8  
Data Dari Master : 9  
Data Dari Master : 10  
Data Dari Master : 1
```

Gambar di atas merupakan tampilan Serial Monitor pada *Slave* jika menerima data yang dikirimkan dari *Master* berhasil.

# MODUL 2

## HTTP DENGAN FIREBASE DAN KODULAR

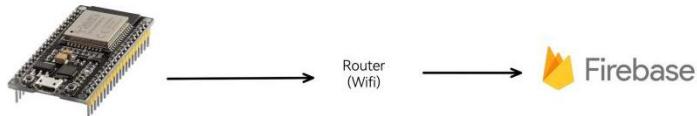
---

### TUJUAN

- 1) Mengetahui protokol HTTP.
- 2) Memahami komunikasi protokol HTTP.
- 3) Memahami penggunaan firebase dalam konteks IoT.
- 4) Mengetahui *application layer* menggunakan Kodular.

### ***REQUIREMENT:***

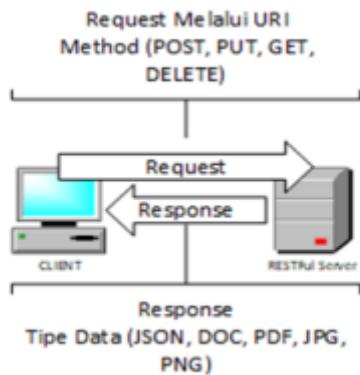
- 1) Arduino IDE/PlatformIO,
- 2) ESP32/ESP8266,
- 3) Firebase ESP Client Library by Mobitz,
- 4) Firebase console di Web.
- 5) Aplikasi Kodular di Android.
- 6) Sensor (opsional).
- 7) Aktuator (opsional).



Pada dasarnya ESP32 tidak bisa langsung berkomunikasi dengan Firebase. ESP32 harus melalui Router atau Wifi untuk bisa berkomunikasi dengan Firebase, tanpa adanya Router ESP32 tidak dapat berkomunikasi secara langsung dengan Firebase. Fungsi Router adalah sebagai gateway atau penghubung yang nantinya Router dengan Firebase akan berkomunikasi dengan menggunakan HTTP yang akan di dalami dalam modul ini.

## 2. 1 *Hypertext Transfer Protocol (HTTP)*

HTTP (Hypertext Transfer Protocol) adalah sebuah protokol yang memiliki metode komunikasi data dengan cara REQUEST (meminta) dan RESPONSE (menjawab) antara client dan server. Client mengirim permintaan (REQUEST) kepada server, lalu server akan merespon (RESPONSE) dan mengirim data yang diminta oleh client dan menampilkannya. Protokol ini berguna untuk mentransfer informasi seperti dokumen, file, gambar, dan video antar komputer. Protokol ini digunakan untuk akses antara client dan server pada jaringan komputer, khususnya internet, atau dapat diartikan juga sebagai sebuah protokol pada internet yang digunakan sebagai metode dalam mentransfer halaman *World Wide Web* (www). Sehingga dalam menerima data dari server umumnya client menggunakan web browser.



### 2. 1. 1 CRUD

CRUD adalah singkatan dari *Create*, *Read*, *Update* dan *Delete*. Keempat hal tersebut merupakan perintah dengan peran yang esensial dalam sebuah aplikasi database yang sifatnya relational. Beberapa contoh DBMS (*Database Management System*) yang menggunakan CRUD sebagai fungsi utamanya yaitu MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, dan lainnya.

Fungsi-fungsi dalam CRUD adalah sebagai berikut.

- **CREATE** : untuk membuat entri atau record baru dalam sebuah tabel di dalam database. Di dalam aplikasi berbasis SQL, fungsi create seringkali disebut dengan insert
- **READ** : untuk membaca entri data yang sudah terdaftar dalam sebuah database. Namun, fungsi ini tidak akan memberikan akses untuk mengubah data yang telah tersimpan di dalam database.
- **UPDATE** : untuk memperbarui entri data ketika informasi di dalamnya memerlukan perubahan. Dengan fungsi ini, kamu dapat mengubah detail dari sebuah entri di database.
- **DELETE** : untuk menghapus entri data yang sudah tidak diperlukan dalam sebuah database. Ketika menggunakan fungsi ini, kamu akan mengakses detail terkait sebuah entri dan kemudian memberikan perintah kepada sistem untuk menghilangkannya dari database.

Terdapat beberapa variasi lain dari CRUD yang bisa diterapkan dan terletak pada sistem HTTP. Beberapa variasi dari CRUD adalah sebagai berikut:

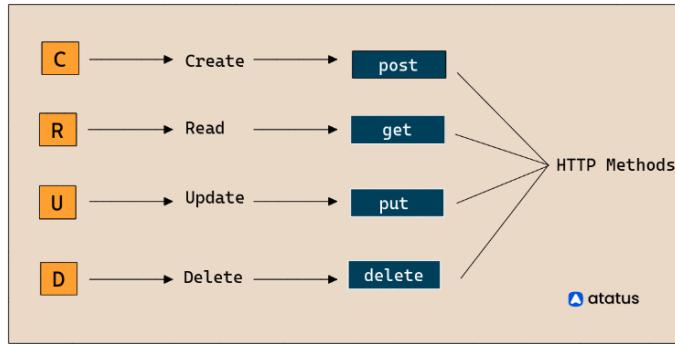
- BREAD (Browse, Read, Edit, Add, Delete)
- CRAP (Create, Replicate, Append, dan Process)
- CRUDL (Create, Read, Update, Delete, Lis(t))
- DAVE (Delete, Add View, Edit)

### 2. 1. 2 Metode Komunikasi Protokol HTTP

Pada sisi HTTP, metode komunikasi yang digunakan menggunakan prinsip CRUD dengan sedikit perbedaan pada istilah yang digunakan:

- ketika anda ingin membaca suatu data (READ), maka metode yang digunakan adalah GET.
- Jika ingin membuat sebuah record baru atau (CREATE), maka kamu akan menggunakan metode POST.
- Jika ingin memperbarui data atau (UPDATE), kamu harus menggunakan metode PUT atau PATCH.

- Jika ingin menghapus data pada sebuah record kamu bisa menggunakan (DELETE).



Selain dari metode yang telah disebutkan sebelumnya, ada beberapa metode komunikasi yang dapat digunakan pada protokol HTTP seperti yang ditunjukkan pada tabel berikut.

Method HTTP	Definisi	Penjelasan
GET	Mengambil Sumber daya	Digunakan untuk mengambil data tertentu dari server
POST	Menyimpan atau mengirim data	Digunakan untuk mengirim data ke server untuk disimpan atau diolah
PUT	Memperbarui sumber daya	Digunakan untuk memperbarui data yang sudah ada pada server
DELETE	Menghapus sumber daya	Digunakan untuk menghapus data tertentu dari server
PATCH	Mengubah bagian dari sumber daya	Digunakan untuk memperbarui atau mengubah bagian dari data yang sudah ada pada server
HEAD	Mendapatkan informasi header	Digunakan untuk meminta informasi header dari data yang diminta melalui method GET
OPTIONS	Mendapatkan informasi dari server	Digunakan untuk meminta informasi tentang metode-metode yang didukung oleh server untuk data tertentu
CONNECT	Membuat koneksi ke server	Digunakan untuk membuat koneksi dengan server melalui proxy server
TRACE	Mencatat aktivitas jaringan	Digunakan untuk meminta server untuk mengirimkan kembali permintaan yang telah diterima

## 2. 2 Firebase



Firebase adalah suatu layanan dari Google untuk memberikan kemudahan bahkan mempermudah para developer aplikasi dalam mengembangkan aplikasinya. Firebase alias BaaS (Backend as a Service) merupakan solusi yang ditawarkan oleh Google untuk mempercepat pekerjaan developer. Dengan menggunakan Firebase, apps developer bisa fokus dalam mengembangkan aplikasi tanpa memberikan effort yang besar untuk urusan backend dan Firebase ini juga menggunakan protokol HTTP JSON.

Firebase didirikan pertama kali pada tahun 2011 oleh Andrew Lee dan James Tamplin. Produk Firebase yang pertama kali adalah Realtime Database. Realtime Database digunakan developer untuk menyimpan data dan synchronize ke banyak user. Kemudian ia berkembang sebagai layanan pengembang aplikasi. Pada bulan Oktober 2014, perusahaan tersebut diakuisisi oleh Google.

Mengenai segi layanan, dulu Firebase memberikan service trial (percobaan), namun saat ini kamu bisa memanfaatkan dan menggunakan layanan Firebase secara free (gratis). Tentu saja dengan adanya batasan-batasan tertentu.

### 2. 2. 1 Fitur Firebase

Terdapat beberapa fitur pada Firebase: Google Analytics, Cloud Messaging and Notifications, Authentication, Cloud Firestore, Realtime Database, and Hosting.

Berikut adalah penjelasan mengenai fitur-fitur yang terapat di firebase.

#### 1) Analytics

Fitur Analytics adalah salah satu fitur pada Firebase yang digunakan sebagai koleksi data dan reporting untuk aplikasi Android maupun iOS. Koleksi data pun bervariasi. Sebagai contoh, kamu dapat membuat suatu laporan atau report untuk pengguna aplikasi di negara Indonesia saja, atau mungkin negara

lain seperti Singapura. Kamu juga bisa melihat bagian mana saja dari aplikasi yang paling sering digunakan oleh user.

Fitur ini mempunyai kelebihan yang memungkinkan kita untuk bisa membuat segmentasi user berdasarkan user attribute. User attribute adalah suatu parameter yang bisa kita gunakan sebagai filter yang bertujuan untuk reporting dan notifikasi. Contohnya pada aplikasi online shop. Dengan user attribute, kamu bisa tahu jumlah user yang membeli handphone merk ‘O’ atau bahkan bisa mencari tahu jam berapa transaksi yang dilakukan user sering terjadi.

## 2) Cloud Messaging and Notification

FCM (Firebase Cloud Messaging) yaitu menyediakan koneksi yang handal dan tentunya hemat baterai antar server maupun antar perangkat, sehingga kamu dapat mengirim dan menerima pesan serta notifikasi di Android, iOS, dan web tanpa perlu biaya.

Untuk menargetkan pesan lanjutan, kamu bisa targetkan pesan dengan mudah menggunakan segment yang telah ditentukan sebelumnya yakni menggunakan demografi dan behavior/perilaku. Anda dapat menargetkan pesan ke perangkat yang telah berlangganan pada topik tertentu. Selain itu, Anda bisa juga menargetkan hanya ke satu perangkat untuk mendapatkan informasi data yang terperinci. Biasanya ini dilakukan untuk proses pengujian.

Pesan notifikasi ini terintegrasi sepenuhnya dengan *Google Analytics for Firebase*, sehingga kamu memiliki akses pada interaksi dan tracking konversi secara detail. Anda dapat memantau suatu efektivitas dari satu dashboard tanpa perlu coding atau membuat program sendiri.

## 3) Authentication

Firebase Authentication adalah salah satu layanan back-end, fitur Android dan iOS, SDK yang mudah digunakan, dan tampilan interfaces yang siap pakai untuk mengautentikasi pengguna ke aplikasi yang kamu buat. Firebase Authentication mendukung autentikasi menggunakan nomor telepon, sandi, penyedia identitas gabungan populer seperti seperti Google, Facebook, dan sebagainya.

Firebase Authentication terintegrasi dengan fitur layanan Firebase lainnya. Sistem ini memanfaatkan berbagai jenis standar industri, seperti OAuth 2.0 dan OpenID Connect, yang memudahkan integrasi dengan backend khusus buatanmu.

Fitur ini juga dapat memudahkan pengguna untuk login ke aplikasi dengan menggunakan fitur Firebase UI (tampilan interfaces), sebagai alternatif full drop-in authentication.

#### 4) Cloud Firestore

Cloud Firestore merupakan database NoSQL yang dihosting di cloud dan dapat diakses melalui SDK real oleh aplikasi iOS, Android dan web. Seperti halnya Firebase Realtime Database, Cloud Firestore membuat datamu tetap terkoneksi di aplikasi user melalui listener realtime dan menawarkan layanan secara offline untuk aplikasi seluler dan web. Dengan begitu, kamu dapat membuat aplikasi yang *powerfull*, responsif, dan mampu bekerja tanpa bergantung pada latensi koneksi internet.

#### 5) Realtime Database

Firebase Realtime Database adalah database yang di-host melalui cloud. Data disimpan dan dieksekusi dalam bentuk JSON dan disinkronkan secara realtime ke setiap user yang terkoneksi. Hal ini berfungsi memudahkan kamu dalam mengelola suatu database dengan skala yang cukup besar. Ketika kamu membuat aplikasi lintas-platform/multiplatform menggunakan SDK Android, iOS, dan juga JS (JavaScript), semua pengguna akan berbagi sebuah instance Realtime Database dan menerima *update-an* data secara serentak dan otomatis.

Kemampuan lain dari Firebase Realtime Database adalah tetap responsif bahkan saat offline karena SDK Firebase Realtime Database menyimpan data langsung ke disk device atau memori lokal. Setelah perangkat terhubung kembali dengan internet, perangkat pengguna (*user*) akan menerima setiap perubahan yang terjadi.

#### 6) Hosting

Selanjutnya ada Firebase Hosting, suatu layanan hosting konten web. Hanya dengan satu instruksi, kamu dapat mengimplementasikan aplikasi web

serta menyajikan konten statis maupun dinamis ke CDN (jaringan penayangan konten) global dengan cepat. Kegunaan dari Firebase Hosting itu sendiri yaitu mampu menayangkan konten melalui koneksi yang begitu aman, mengirimkan konten secara cepat, dan mendukung semua jenis konten untuk di hosting, mulai dari file HTML dan CSS hingga API atau layanan mikro Express.js.

### 2. 2. 2 Firebase Realtime Database (RTDB)

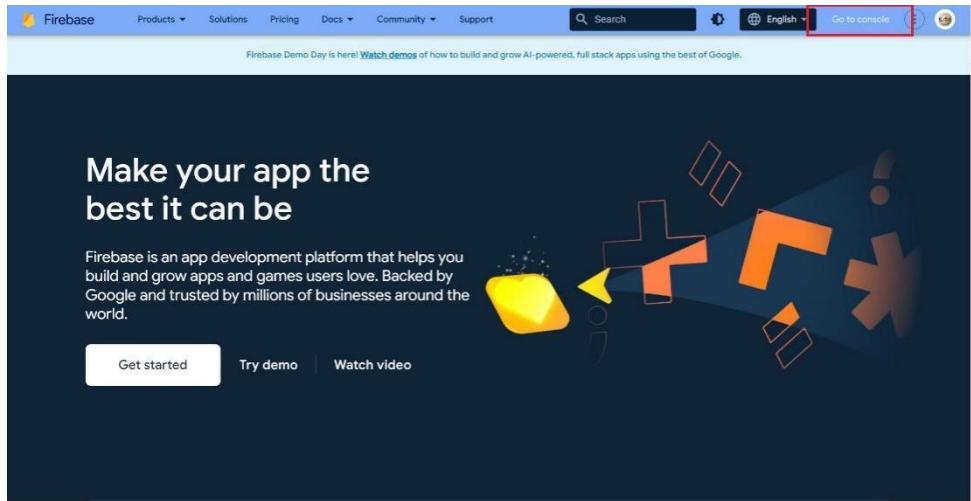
Firebase Realtime Database adalah sebuah Cloud-Hosted database yang dapat menyimpan dan melakukan sinkronisasi data secara realtime untuk setiap client yang terhubung. Setiap kali pengguna memperbarui data, itu akan menyimpannya pada cloud dan sekaligus memberitahu ke semua client yang terhubung dan secara otomatis menerima pembaruan dengan data terbaru.

#### 2.2.2.1 Setup Firebase Realtime Database

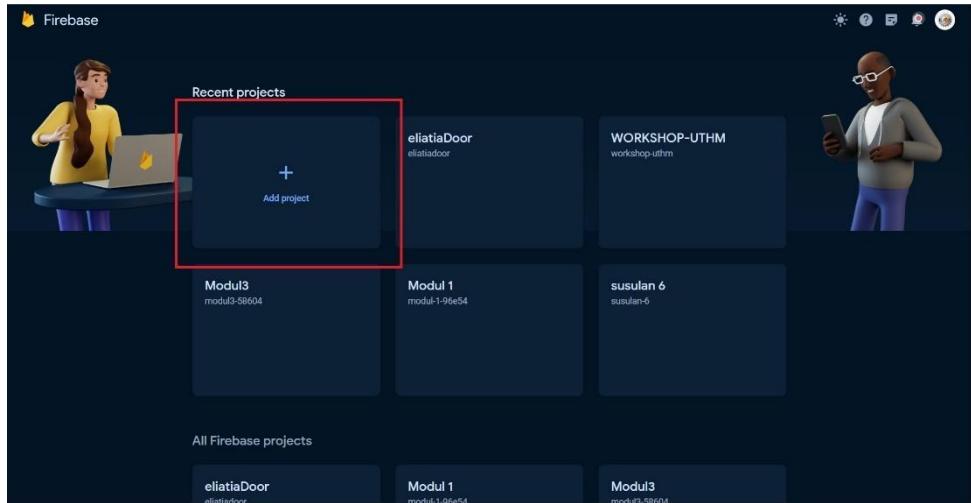
(Jelaskan cara setup Firebase RTDB mulai dari akun sampai dapat digunakan beserta penjelasannya )

Setup awal untuk Firebase Realtime Database ini dilakukan dengan mengakses website dari firebasenya (<https://firebase.google.com/>) dengan tahap-tahap sebagai berikut:

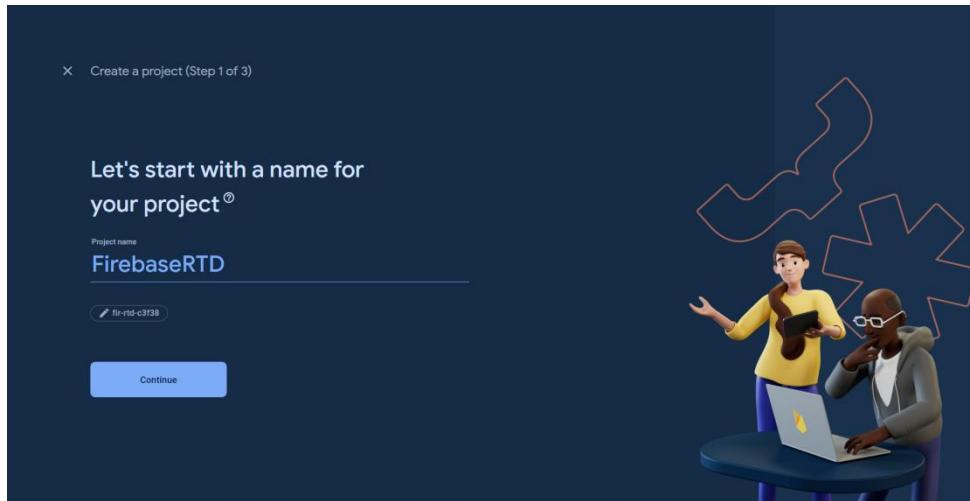
1. Buka laman dari firebase <https://firebase.google.com/> lalu tekan **Go to console**.



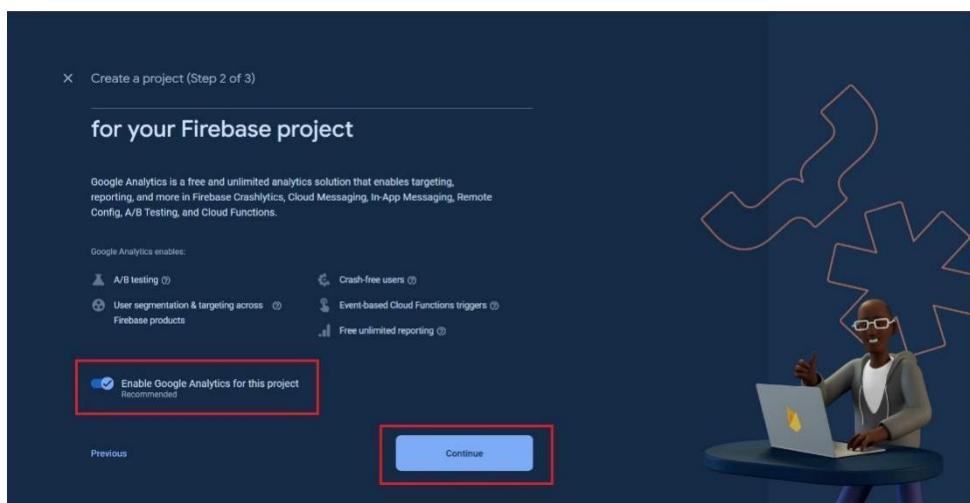
2. Ini adalah halaman dimana dapat menambahkan project baru maupun melihat project yang lama. Langkah selanjutnya tekan bagian **Add project**.



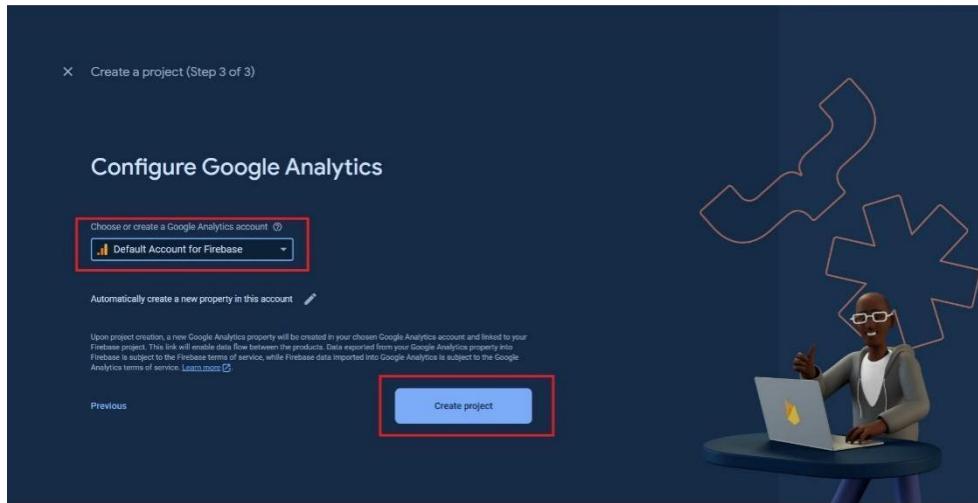
3. Selanjutnya isi nama projek yang akan dibuat di kolom **Project name**.



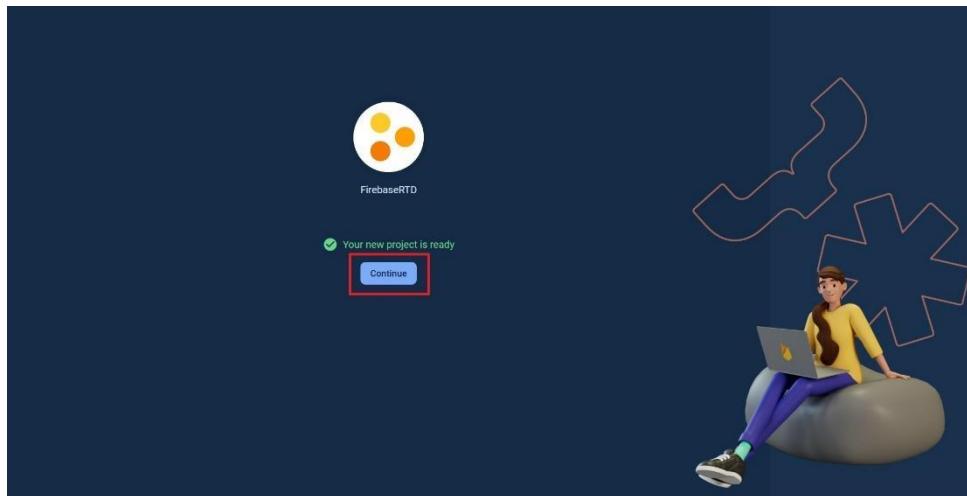
4. Centang bagian **Enable Google Analytics for this project**, lalu tekan **Continue**.



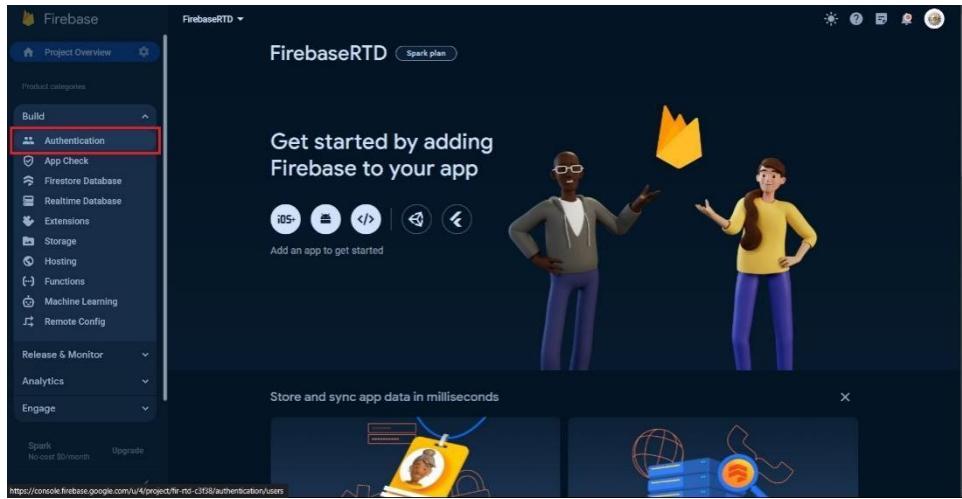
5. Untuk bagian **Choose or create a Google Analytics account** pilih **Default Account for Firebase**, lalu tekan **Create project**.



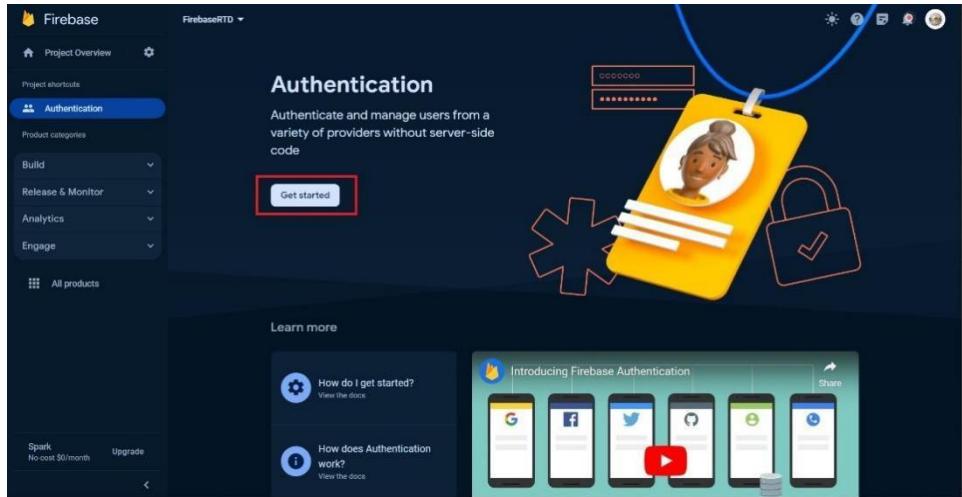
6. Pada halaman ini projek di firebase yang ingin dibuat sudah jadi. Tekan **Continue** untuk lanjut ke halaman projectnya.



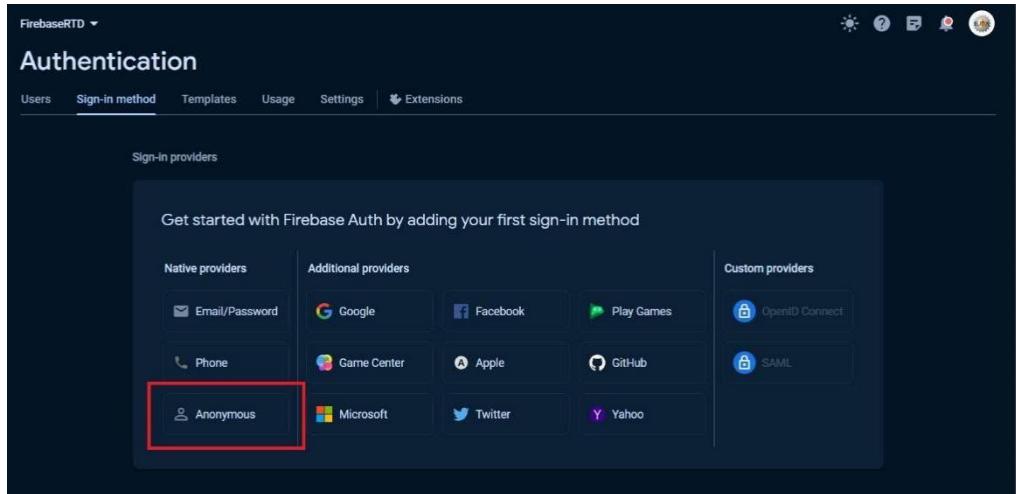
7. Pada halaman awal dari project ini, tekan bagian **Build** lalu pilih bagian **Authentication**.



8. Selanjutnya kita melakukan pendaftaran kepada user-user yang dapat menggunakan project firebase ini dengan tekan **Get Started**.



9. Pilih bagian **Sign-in method**. Teken bagian **Anonymous** untuk menambah akun user tanpa memberikan email pendaftarnya. Selanjutnya, centang bagian **Anonymous**, lalu tekan **Save**. Jika bagian Anonymous sudah **Enabled**, maka project ini sudah bisa diakses oleh akun Anonymous.



FirebaseRTD ▾

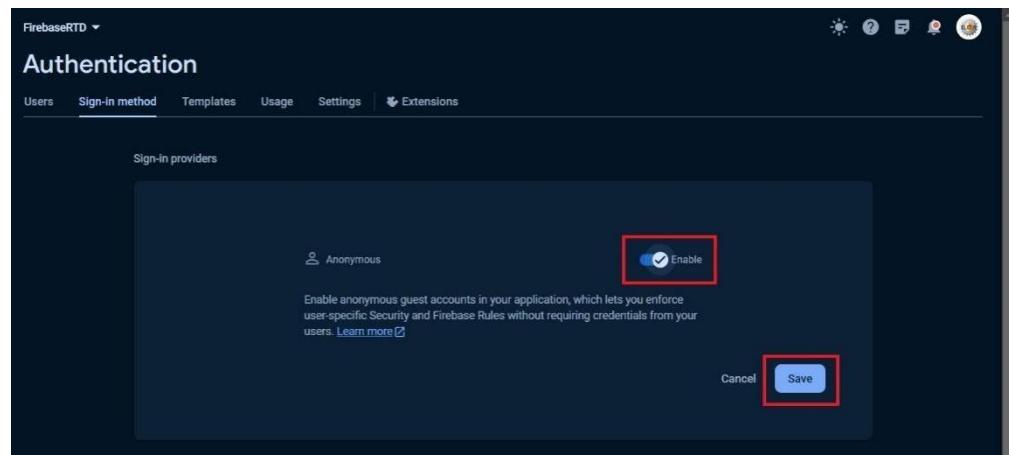
## Authentication

Users Sign-in method Templates Usage Settings Extensions

Sign-in providers

Get started with Firebase Auth by adding your first sign-in method

Native providers	Additional providers	Custom providers
Email/Password	Google Facebook Play Games	OpenID Connect
Phone	Game Center Apple GitHub	SAML
<b>Anonymous</b>	Microsoft Twitter Yahoo	



FirebaseRTD ▾

## Authentication

Users Sign-in method Templates Usage Settings Extensions

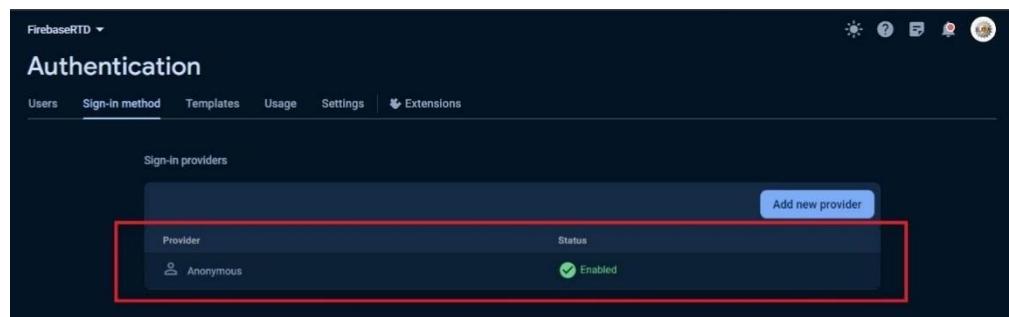
Sign-in providers

Anonymous

Enable

Enable anonymous guest accounts in your application, which lets you enforce user-specific Security and Firebase Rules without requiring credentials from your users. [Learn more](#)

Cancel Save



FirebaseRTD ▾

## Authentication

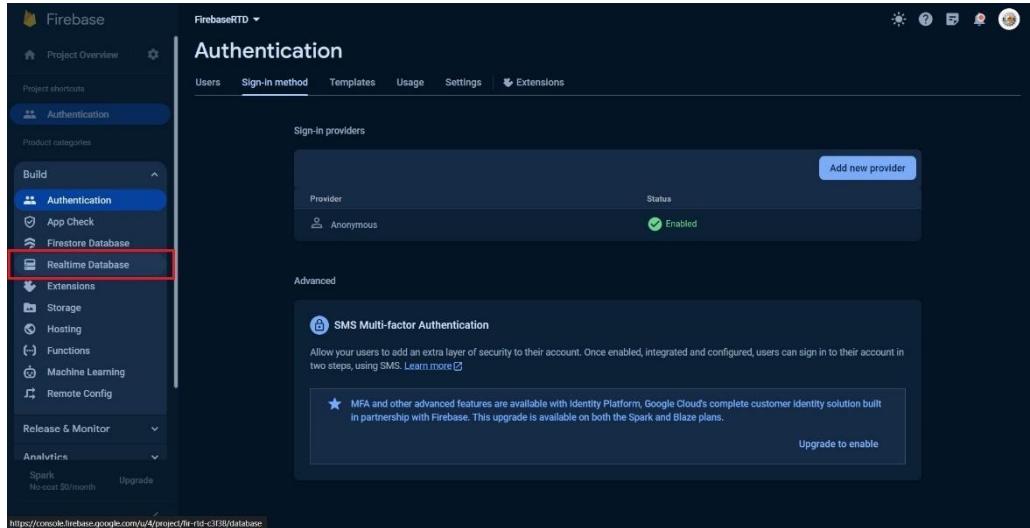
Users Sign-in method Templates Usage Settings Extensions

Sign-in providers

Provider	Status
Anonymous	Enabled

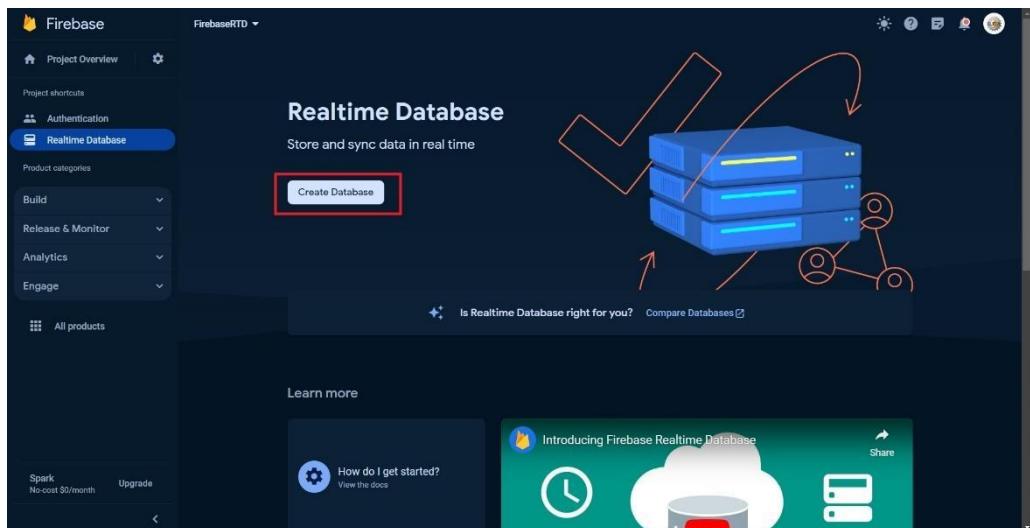
Add new provider

10. Selanjutnya pilih bagian **Build** lalu tekan **Realtime Database**.



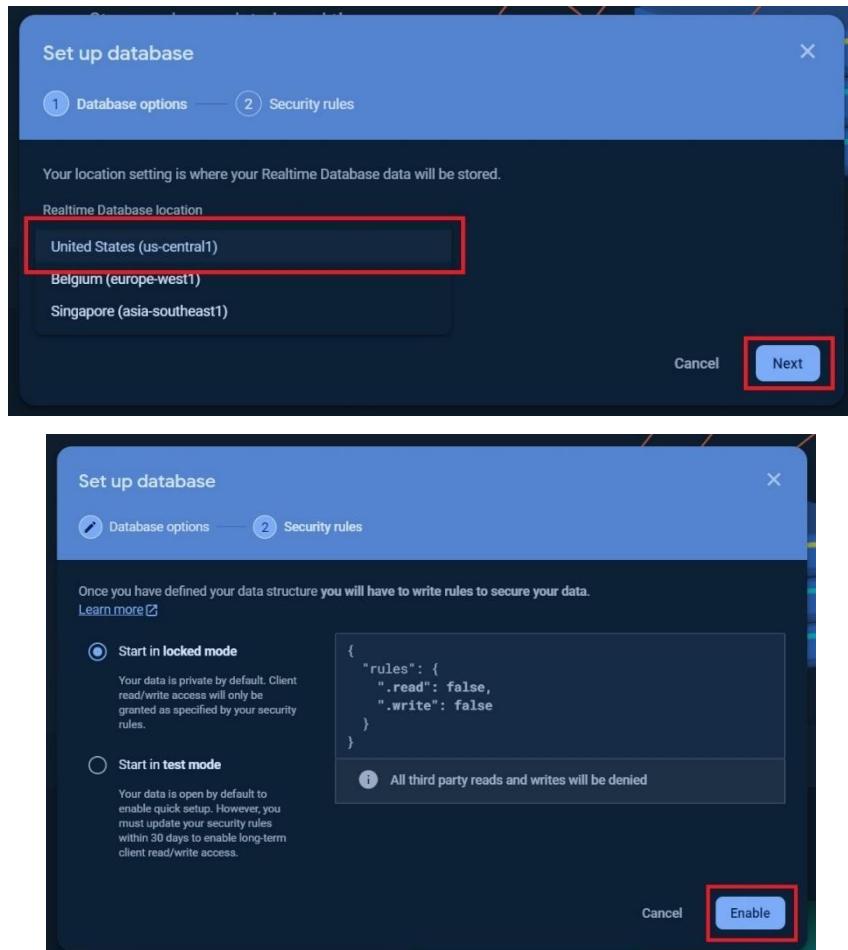
The screenshot shows the Firebase console's Project Overview page. On the left, there is a sidebar with categories like Project shortcuts, Authentication (selected), Build (selected), Analytics, and Engage. Under Build, Realtime Database is highlighted with a red box. The main content area is titled "Authentication" and shows "Sign-in method" as the active tab. It displays a table of sign-in providers with "Anonymous" listed and "Enabled". Below this is a section for "SMS Multi-factor Authentication" with a "Upgrade to enable" button.

11. Dalam halaman Realtime Database, tekan **Create Database** untuk membuat Databasenya.

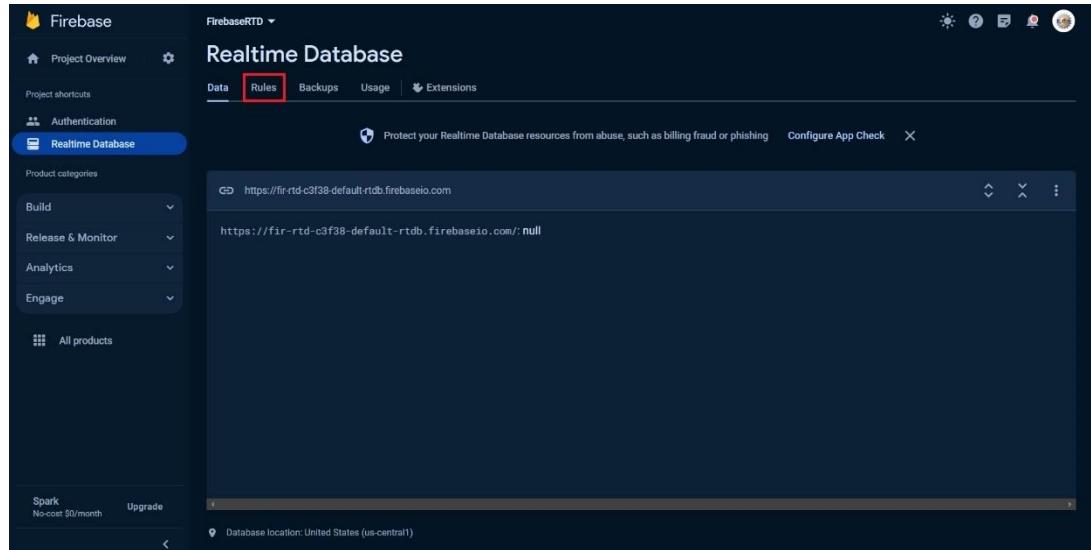


The screenshot shows the Firebase console's Realtime Database page. The sidebar on the left has "Realtime Database" selected. The main content area is titled "Realtime Database" and features a large blue server icon with a checkmark and arrows indicating data flow. A red box highlights the "Create Database" button. Below the button is a section titled "Is Realtime Database right for you?" with a "Compare Databases" link. At the bottom, there are links for "Learn more", "How do I get started?", and a "Share" button.

12. Di bagian **Database options**, pilih **Realtime Database location** di **United States (us-central1)**. Bagian **Security rules**, pilih **Start in locked mode** lalu tekan **Enable**.



13. Ini adalah halaman utama dari Realtime Database, selanjutnya tekan bagian **Rules**, lalu ganti rules di **read** dan **write** dari “**false**” ke “**true**”. Lalu tekan **Publish**.



The screenshot shows the Firebase Realtime Database Rules tab. The URL `https://fir-rtd-c3f38-default-rtdb.firebaseio.com/.json` is highlighted with a red box. Below the URL, the default security rules are displayed:

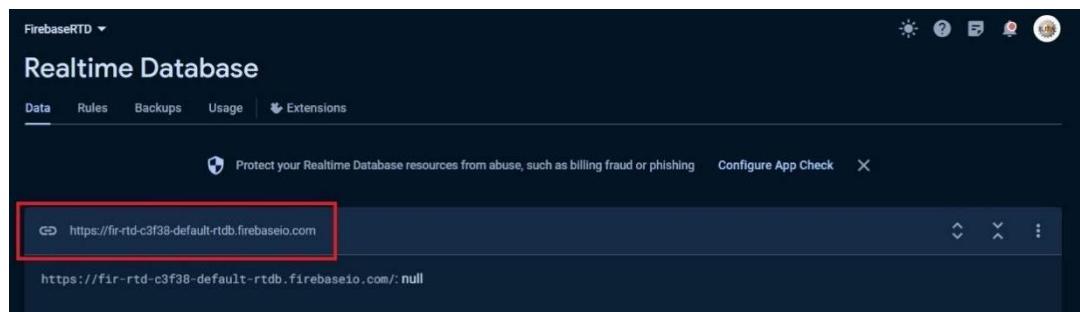
```

1  {
2    "rules": {
3      ".read": true,
4      ".write": true
5    }
6  }

```

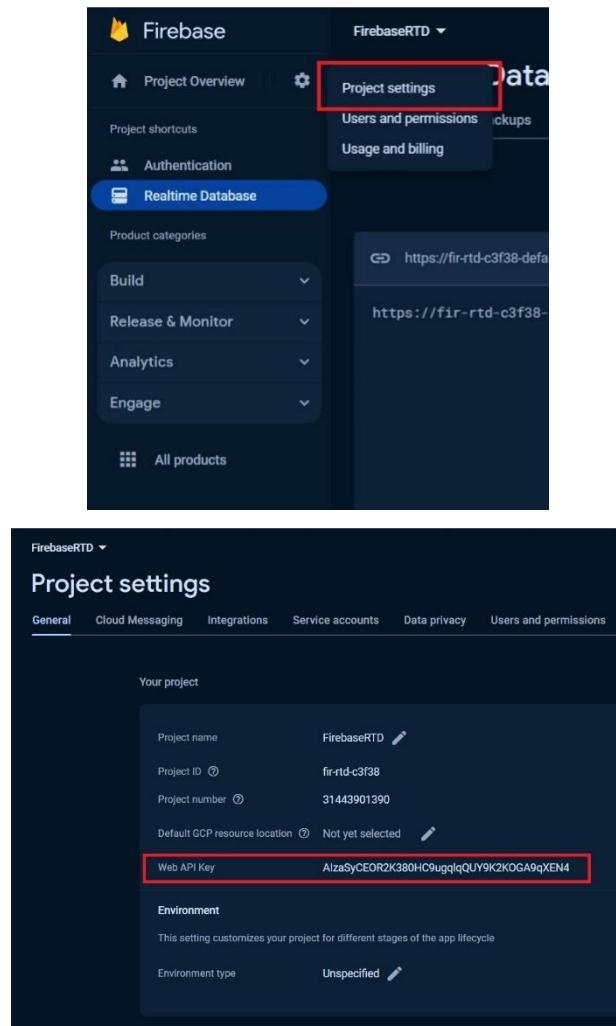
A red box also highlights the **Publish** button at the top of the rules editor.

14. Untuk melihat URL dari project Realtime Database, bisa dilihat di kotak merah di bagian **Data**.



The screenshot shows the Firebase Realtime Database Data tab. The URL `https://fir-rtd-c3f38-default-rtdb.firebaseio.com/.json` is highlighted with a red box. Below the URL, the message `https://fir-rtd-c3f38-default-rtdb.firebaseio.com/.json: null` is displayed.

15. Untuk melihat API dari project ini, dengan membuka di bagian **Project settings** lalu di bagian **Web API Key**.

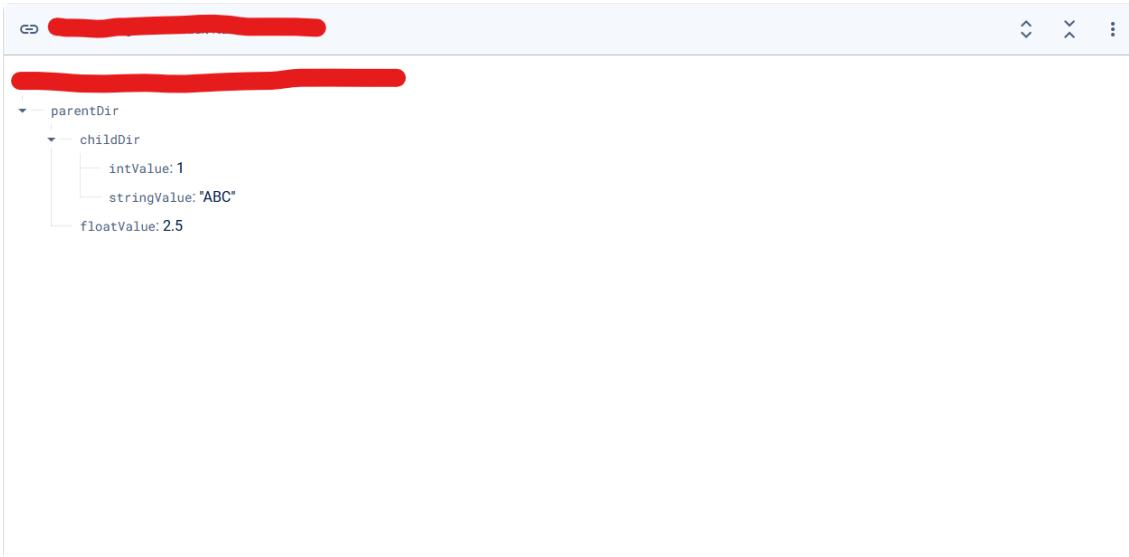


The image consists of two screenshots from the Firebase console.

The top screenshot shows the "Project Overview" page. The "Realtime Database" tab is selected. On the right, there is a "Project settings" button, which is highlighted with a red box. Below it are other tabs: "Users and permissions", "Usage and billing", and "Data".

The bottom screenshot shows the "Project settings" page under the "General" tab. It displays project details: "Project name: FirebaseRTD", "Project ID: fir-rtd-c3f38", "Project number: 31443901390", and "Default GCP resource location: Not yet selected". A "Web API Key" field is present, containing the value "AlzaSyCEO...XEN4", which is also highlighted with a red box.

### 2.2.2.2 Directory Penyimpanan Firebase RTDB



Directory pada database menggunakan JSON data. Data di dalam JSON ditulis dengan pasangan nama dan nilai, seperti pada gambar di atas terdapat directory dengan nama intValue dengan nilai 1. Pada database tersebut terdapat directory yang bernama parentDir dan childDir, kedua directory tersebut digunakan untuk menggambarkan hierarki yang ada pada Firebase RTDB. Dengan hirarki tersebut, maka kita dapat menentukan directory yang ingin digunakan, sebagai contoh:

**“Kita ingin menyimpan nilai berupa integer sebesar 34 ke directory stringValue”**

Maka directory tersebut adalah:

**/parentDir/childDir/intValue/34**

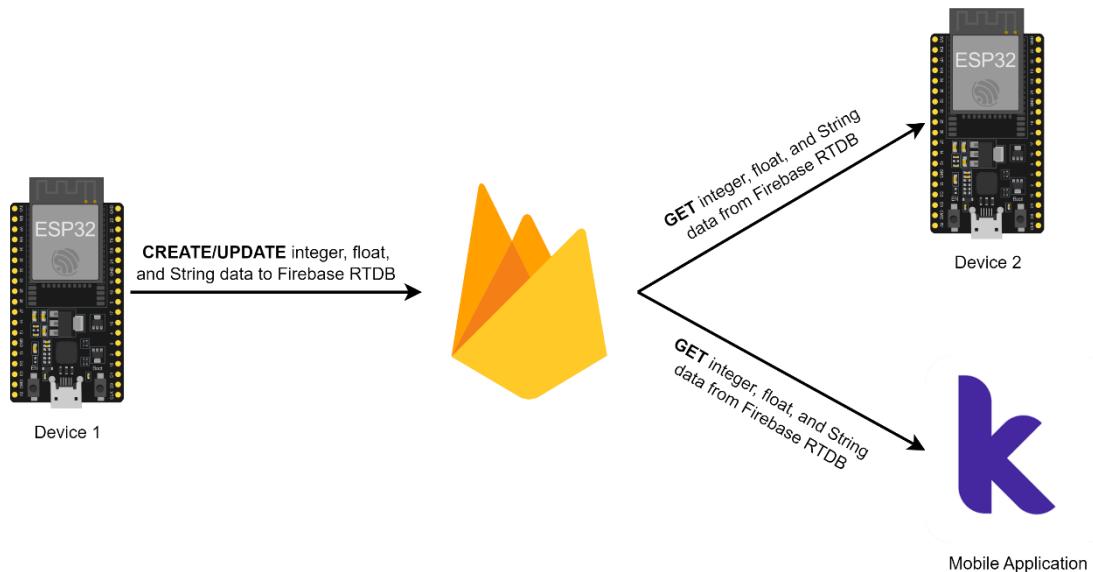
Dari contoh tersebut, maka kita dapat melihat hierarki yang ada pada database tersebut, dimulai dengan "/" yang biasa disebut dengan **root**, kemudian parentDir, childDir, intValue, dan diakhiri dengan nilai yang akan kita simpan.

Contoh selanjutnya adalah jika ingin menyimpan suatu nilai 23 ke directory floatValue, maka directory floatValue adalah:

**/parentDir/floatValue/23**

Untuk **directory floatValue** tidak diketik setelah **childDir**, hal tersebut dikarenakan floatValue memiliki hirarki setara dengan childDir, atau hirarki dari floatValue terletak setelah parentDir.

### 2. 2. 3 Implementasi Firebase RTDB dengan ESP32 dan Kodular



Dalam implementasi ini, ESP32 device 1 akan melakukan CREATE/UPDATE ke Firebase RTDB, dan ESP32 Device 2 serta kodular akan GET data dari Firebase RTDB.

#### 2.2.3.1 ESP32

ESP32 adalah Mikrokontroler System on Chip (SoC) berbiaya rendah dari Espressif Systems, yang juga sebagai pengembang dari SoC ESP8266 yang terkenal dengan NodeMCU. ESP32 adalah penerus SoC ESP8266 dengan menggunakan Mikroprosesor Xtensa LX6 32-bit Tensilica dengan Wi-Fi dan Bluetooth yang terintegrasi. Fitur Wi-Fi ini digunakan untuk megakses internet pada ESP32. Sehingga, Firebase RTDB bisa diakses dengan metode HTTP dari ESP32.

#### 2.2.3.2 Firebase Arduino Client Library for ESP32 and ESP8266 by Mobitz

Library ini menyediakan Firebase *Realtime Database*, Firebase *Cloud Messaging*, Firestore *Cloud Database*, Firebase *Storage*, Google *Cloud Storage*, dan *Cloud Functions* untuk Firebase serta mendukung sebagian besar perangkat Arduino kecuali perangkat AVR yang memiliki sumber daya dan batas *compiler*.

*Library* ini dapat bekerja dengan jaringan *on-chip/on-board* (WiFi/Ethernet) dan modul jaringan eksternal melalui *library* klien. Fitur-fiturnya dapat dikonfigurasi untuk menambah dan mengurangi beberapa fitur yang tidak digunakan. Tersedia versi ESP8266 dan Raspberry Pi Pico dan ESP32 yang hanya menyediakan Firebase *Realtime Database* dan fungsi Firebase *Cloud Messaging*.

Dokumentasi/fungsi yang biasa digunakan dari untuk proses CRUD adalah sebagai berikut.

#### C. firebaseData

`FirebaseData` biasanya digunakan sebagai nama variabel yang mewakili objek data atau struktur dalam aplikasi Firebase. Ini mungkin digunakan untuk menyimpan data yang diambil dari Firebase Realtime Database, Firestore, atau layanan Firebase lainnya.

#### D. firebaseAuth

`FirebaseAuth` atau `auth` umumnya mewakili objek otentikasi di Firebase. Ini digunakan untuk mengelola otentikasi pengguna, termasuk pendaftaran, masuk, keluar, pengaturan ulang kata sandi, dan fungsionalitas terkait otentikasi lainnya.

#### E. firebaseConfig

`FirebaseConfig` atau `config` biasanya merujuk pada objek konfigurasi untuk Firebase. Objek ini mencakup pengaturan dan kredensial yang diperlukan untuk menginisialisasi SDK Firebase dalam aplikasi Anda.

**Untuk dokumentasi lengkap dapat dilihat pada link:**

<https://github.com/mobizt/Firebase-ESP-Client>

#### 2.2.3.3 Source Code Implementasi Firebase Menggunakan ESP32

##### 1) Library, Variables, dan Objects yang Dibutuhkan.

Hal yang pertama kali dilakukan adalah mendeklarasikan *library* yang akan digunakan. Ada dua bagian *library* yang dibutuhkan:

**Pertama** adalah *library* untuk menghubungkan perangkat mikrokontroler ke Wifi. Pada kodingan ini akan digunakan *library* adalah `<Wifi.h>` untuk ESP32.

**Inisialisasi library wifi yang akan digunakan**

```
#include <WiFi.h>
```

**Kedua** adalah library untuk Firebase. Ada banyak library untuk firebase yang dapat digunakan, seperti yang dikembangkan oleh mobitz, Suwatchai K., Rupak Podar, dan lain sebagainya., akan tetapi *library* yang paling banyak digunakan dan stabil adalah **Firebase Arduino Client for ESP8266 and ESP32 by mobitz**, oleh karena itu kita akan menggunakan library tersebut.

Pada *library* tersebut, kita dapat menggunakan 3 *library*:

<b>&lt;Firebase_ESP_Client.h&gt;</b>	Provides the fundamental function for firebase like FirebaseData, FirebaseAuth, FirebaseConfig, CRUD (Create, Read, Update, Delete), etc.
<b>&lt;Addons/TokenHelper.h&gt;</b>	Provide the token generation process info.
<b>&lt;addons/RTDBHelper.h&gt;</b>	Provide the RTDB payload printing info and other helper function.

**Inisialisasi library untuk firebase**

```
// FIREBASE LIBRARY
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h" // Provide the token generation process info.
#include "addons/RTDBHelper.h" // Provide the RTDB payload printing info
and other helper functions.
```

Berikutnya adalah mendeklarasikan kebutuhan untuk konfigurasi Firebase.

**Inisialisasi kebutuhan untuk setup Firebase**

```
// FIREBASE SETUP
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
bool signupOK = false;
```

Selanjutnya, deklarasikan konstanta berupa String untuk SSID dan Password untuk Wifi.

**Inisialisasi SSID dan Password Wifi**

```
/* 1. Define the WiFi credentials */
#define WIFI_SSID "<YOUR_SSID_HERE>"
#define WIFI_PASSWORD "<YOUR_WIFI_PASSWORD_HERE>"
```

Lalu, deklarasikan konstanta yang dibutuhkan oleh firebase, salah satunya adalah **API\_KEY** dan **DATABASE\_URL** yang telah didapatkan sebelumnya.

**\*Jika anda menggunakan akun email (bukan anonymous), silahkan hilangkan comment pada nomor 4.**

#### Inisialisasi API, URL, dan Akun (opsional)

```
/* 2. Define the API Key */  
#define API_KEY "<PUT_YOUR_FIREBASE_WEB_API_HERE>"  
/* 3. Define the RTDB URL */  
#define DATABASE_URL "<YOUR_FIREBASE_URL_HERE>"  
/* 4. ACTIVATE IT for authenticated account: Define the user Email and  
password that already registerd or added in your project */  
// #define USER_EMAIL "<YOUR EMAIL HERE>"  
// #define USER_PASSWORD "<YOUR PASSWORD HERE>"
```

#### 2) **setup()**

Setelah mendeklarasikan *library*, variabel, dan konstanta yang dibutuhkan, maka selanjutnya adalah kode bagian **setup()**.

Berikut langkah yang dijalankan untuk terkoneksi ke jaringan Wifi.

1. Dimulai dari **Wifi.begin(WIFI\_SSID, WIFI\_PASSWORD)** untuk memulai Wifi dan menghubungkannya ke SSID dan Password yang telah ditentukan.
2. Pada bagian while loop akan memeriksa status koneksi sudah terkoneksi atau belum. **Wifi.status()** akan mengembalikan nilai **WL\_CONNECTED** jika sudah terkoneksi. \*Anda bisa melihat dokumentasi **Wifi.status()** dari <https://reference.arduino.cc/reference/en/libraries/wifi/wifi.status/>.
3. **Wifi.localIP()** memberikan informasi Alamat IP pada jaringan wifi yang terhubung.

#### Menyalakan Wifi

```
/* WIFI START */  
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
Serial.print("Connecting to Wi-Fi");  
  
while (WiFi.status() != WL_CONNECTED)  
{  
    Serial.print(".");  
    delay(500);
```

```

    }
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    /* WIFI END */
}

```

Setelah terhubung dengan Wifi, selanjutnya adalah konfigurasi Firebase:

1. **config.api\_key** akan memasukkan RTDB API Key yang telah ditentukan sebelumnya.

#### Menentukan API\_KEY Fierbase RTDB

```

/* FIREBASE START */
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

config.api_key = API_KEY; // Assign RTDB API Key

```

2. Selanjutnya adalah autentikasi. Ada dua cara autentikasi yang akan dijelaskan, yaitu menggunakan akun *anonymous* dan autentikasi email. Untuk akun *anonymous*, kita dapat membuat akun baru menggunakan **Firebase.signUp(&config, &auth, "", "")**. Tanda kutip "" di 2 parameter terakhir adalah email dan password, dikarenakan akun yang digunakan adalah *anonymous*, maka dapat dikosongkan dengan diisi tanda kutip ("").

#### Menentukan autentikasi akun

```

/*For anonymous account: Sign up */
if (Firebase.signUp(&config, &auth, "", ""))
{
    Serial.println("Firebase success");
    digitalWrite(LED_BUILTIN, LOW);
    signupOK = true;
}
else
{
    String firebaseErrorMessage =
config.signer.signupError.message.c_str()
;
}

```

**Firebase.signUp(&config, &auth, "", "")** will signed up an account and assign it at &config and &auth data. You can comment it if you want to use email auth.

```

        Serial.printf("%s\n",
firebaseErrorMessage);
}

/* ACTIVATE IT For authenticated account:
Assign the user sign in credentials */
// auth.user.email = USER_EMAIL;
// auth.user.password = USER_PASSWORD;
    
```

We can assign the email and password to &auth using **auth.user.email** and **auth.user.password** directly if we use **an email authentication**.

3. Setelah autentikasi dimasukkan, **config.database.url** akan memasukkan nilai URL, kemudian fungsi *callback* **config.token\_status\_callback**, dan jika sudah dimasukkan ke dalam konfigurasi semua, terakhir adalah memulai firebase menggunakan fungsi **firebase.begin(&config, &auth)**.

#### Memulai Firebase

```

config.database_url = DATABASE_URL; // Assign rtbd url
config.token_status_callback = tokenStatusCallback; // Set callback

Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
/* FIREBASE END */
    
```

### 3) Loop() and Firebase ESP CRUD

`setup()` sudah selesai, sekarang saatnya masuk ke bagian utama IoT menggunakan HTTP protokol : **the CRUD** atau **CREATE, READ, UPDATE, DELETE**, namun kode di bawah hanya menggunakan metode CREATE/POST dan READ/GET.

- **CREATE / POST**

Kamu dapat CREATE nilai menggunakan tipe data tertentu ke destinasi di firebase menggunakan:

```

Firebase.RTDB.set<DATA_TYPE>(<FirebaseData Object>, <Database
Directory String>, <Value>)
    
    
```

Jika ingin CREATE/POST **nilai integer sebesar 15** ke dalam *directory* **int\_data** dengan **fbdo** sebagai **FirebaseData**, maka kode dapat diketik menjadi:

```

Firebase.RTDB.setInt(&fbdo, "int_data", 15)
    
    
```

Di bawah ini adalah fungsi yang dibuat untuk melakukan CREATE dengan tipe data integer, float, dan string beserta *error check*-nya.

### 1. Integer

#### Fungsi CREATE/POST untuk INT

```
void firebaseSetInt(String databaseDirectory, int value)
{
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setInt(&fbdo, databaseDirectory, value))
    {
        Serial.print("PASSED: ");
        Serial.println(value);
    }
    else
    {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo.errorReason());
    }
}
```

### 2. Float

#### Fungsi CREATE/POST untuk Float

```
void firebaseSetFloat(String databaseDirectory, float value)
{
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setFloat(&fbdo, databaseDirectory, value))
    {
        Serial.print("PASSED: ");
        Serial.println(value);
    }
    else
    {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo.errorReason());
    }
}
```

### 3. String

#### Fungsi CREATE/POST untuk String

```
void firebaseSetString(String databaseDirectory, String value)
{
    // Write a string on the database path
    if (Firebase.RTDB.setString(&fbdo, databaseDirectory, value))
    {
        Serial.print("PASSED: ");
        Serial.println(value);
    }
    else
    {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo.errorReason());
    }
}
```

- **READ / GET**

Untuk menggunakan metode READ/GET, maka fungsi yang digunakan adalah dengan mengganti kata set pada **setInt** menjadi **getInt** dengan format:

```
Firebase.RTDB.get<DATA_TYPE>(<FirebaseData Object>, <Database
Directory String>)
```

Pada **LAMPIRAN MODUL 2 Lampiran 1.** terdapat implementasi kode dan fungsi untuk melakukan CREATE/POST dan READ/GET pada bagian loop().

#### 2.2.3.4 Kodular



**Kodular** adalah salah suatu aplikasi atau *tools IDE open source* seperti MIT App Inventor. Kodular ini memiliki fitur-fitur widget yang paling banyak dari *tools IDE* sejenisnya. Situs Kodular ini tidak hanya bisa membuat aplikasi Android saja, tapi juga bisa mengunggah hasil pembuatan aplikasi tersebut ke dalam Kodular Store dan/atau bisa

membuat ekstensi sendiri untuk menjadikan widget yang belum ada dari bawaan. Pada sebelum ada perubahan nama Kodular, situs ini diberi nama Makeroid.

Kodular didirikan pada 6 Juli 2017 oleh Conor Shipp, Diego Barreiro, Mika, Pavitra Golchha, Sander Jochems, Sivagiri Visakan dan Vishwas Adiga. Beta publik pertama tersedia pada bulan berikutnya. Pada saat itu, fitur Kodular hampir sama dengan MIT App Inventor, dan antarmukanya tidak banyak berubah.

Saat ini, Situs Kodular ini terus berkembang dalam pembuatan aplikasi tools untuk memudahkan si developer dalam membuat aplikasi Android tanpa coding (ketik program). Untuk pembuatan aplikasi Android, hanya mengandalkan drag and drop saja dan menyusun puzzle blok program agar program aplikasi tersebut bisa berjalan dengan baik. Pengenalan Kodular

## 1. Start Up

Website Kodular dapat diakses pada laman berikut ini : <https://www.kodular.io/>

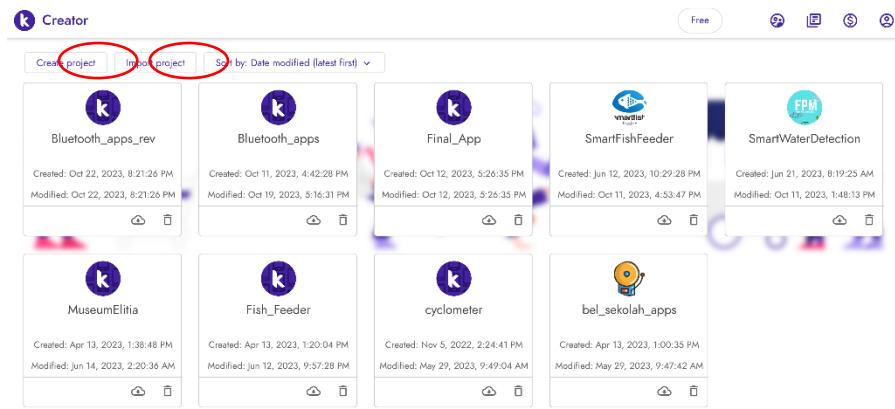


Setelah anda masuk pada laman website kodular, maka anda akan disuguhkan dengan tampilan seperti pada gambar di atas. Untuk memulai membuat aplikasi anda dapat mengikuti langkah-langkah dibawah ini.

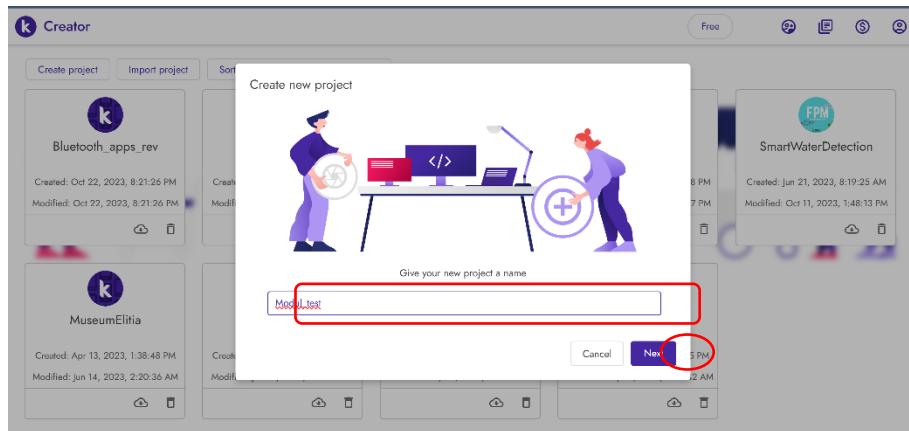
- 1) Tekan tombol **CREATE APPS!** yang ada pada laman tersebut.



- 2) Setelah itu, anda akan disuguhkan dengan tampilan seperti gambar di bawah. Anda dapat melihat seluruh projek aplikasi yang sudah pernah anda buat. Untuk membuat projek baru anda dapat memilih tombol **Create project** dan untuk memasukan projek lain kedalam kodular anda dapat memilih tombol **Import Project**.

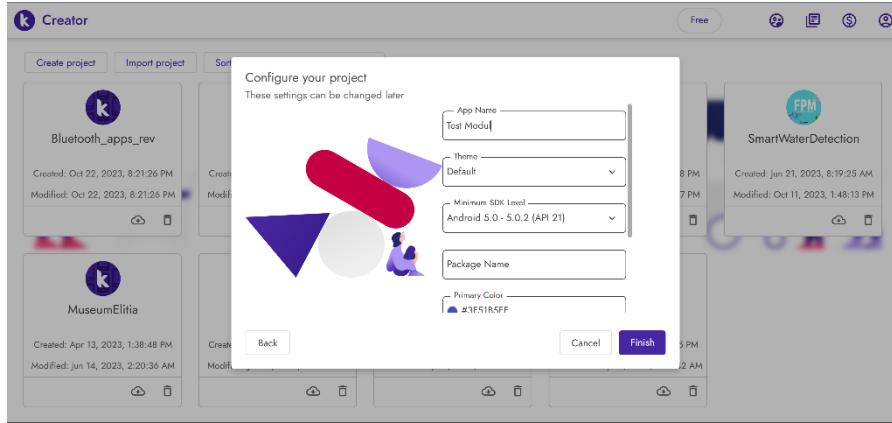


- 3) Tuliskan nama projek anda, kemudian tekan tombol **Next**.

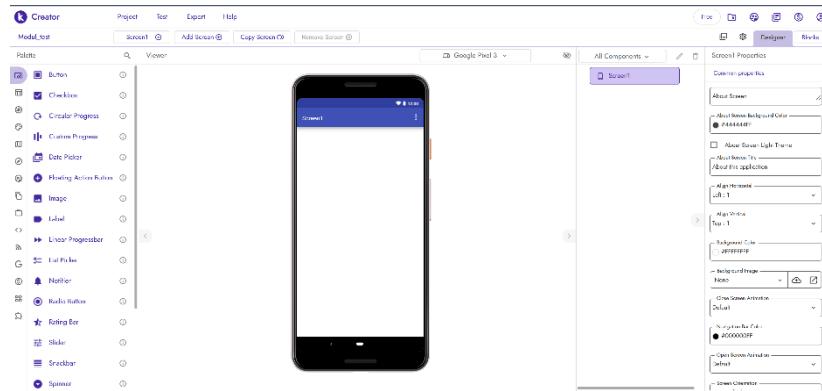


- 4) Kemudian anda dapat memberi nama aplikasi, tema (bisa diatur *default*), minimum SDK level (digunakan untuk mengatur aplikasi tersebut dapat di-

*download* di level android sesuai keinginan anda, namun **disarankan set pada level terendah**), *Package Name* (bisa dikosongkan), *Primary color*, dll. Kemudian jika selesai dapat menekan tombol “**Finish**”.



- 5) Setelah itu, kalian akan ditujukan pada workspace halaman **designer**. Pada workspace kodular terdapat 2 halaman workspace yaitu halaman designer dan juga blocks.

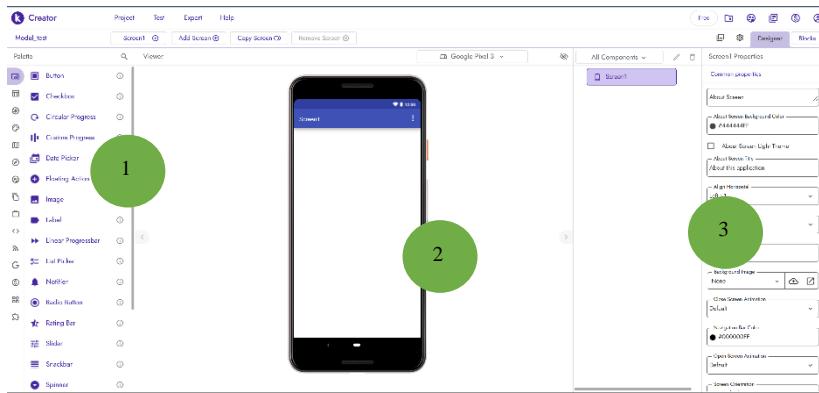


## 2. Halaman Designer

Pada halaman designer digunakan untuk membuat *user interface* dari aplikasi yang akan anda buat, dan pada halaman ini juga digunakan untuk memasukan fitur atau *library* yang tersedia pada Kodular. Untuk penjelasan secara detail mengenai bagian pada halaman designer dapat melihat sesuai nomor dibawah ini:

- 1) **Palette:** pada bagian ini merupakan kumpul Toolbox yang dapat digunakan untuk menambahkan komponen dan fitur ke dalam aplikasi dengan sesuai kebutuhan.

- 2) **Screen Viewer:** Sebagai tampilan aplikasi yang sedang dibuat
- 3) **All component dan Properties:** pada bagian All component menampilkan list widget dan juga fitur yang digunakan. Properties untuk mengatur tampilan komponen seperti warna atau ukuran dan juga untuk mengatur fitur yang digunakan.



**Pada bagian palette** terdapat berbagai komponen dan fitur yang digunakan untuk mendesain aplikasi tersebut , berikut adalah komponen dan fitur tersebut yang akan digunakan pada modul ini :

### 1) User Interface

	<b>Image</b>	Memasukan Gambar dalam Aplikasi yang sedang dibuat
	<b>Label</b>	Menampilkan Teks dalam aplikasi yang sedang dibuat
	<b>Switch</b>	Menampilkan komponen yang memungkinkan user untuk memilih dua keadaan Aktif – Non Aktif

### 2) Google

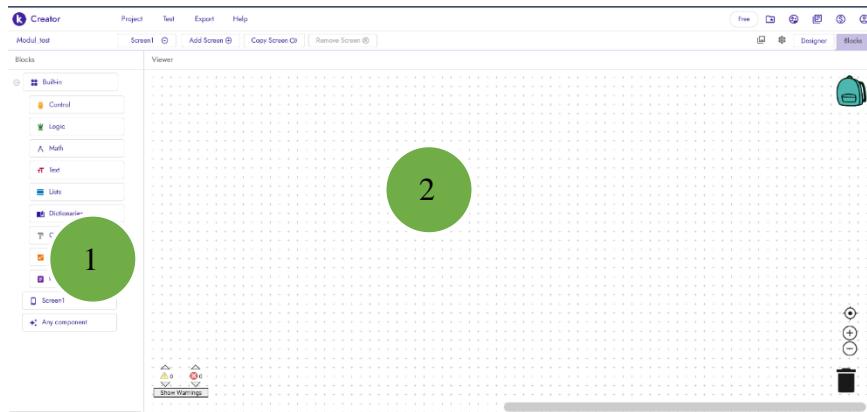
	<b>Firebase Database</b>	Memasukan fitur firebase database untuk dapat menyambungkan antara aplikasi ke firebase realtime data
---	--------------------------	---

Untuk penjelasan fitur dan komponen lainnya secara rinci dapat diakses di laman web <https://docs.kodular.io/components/>.

### 3. Halaman Blocks

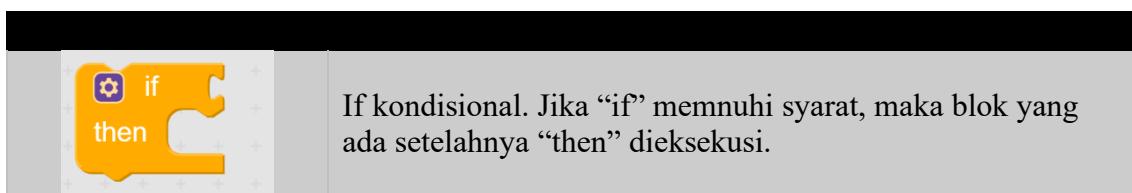
Pada halaman blocks ini dibagi menjadi 2 bagian :

- 1) **Blocks** : terdiri dari list codeblock
- 2) **Workspace** : tempat untuk drop codeblock

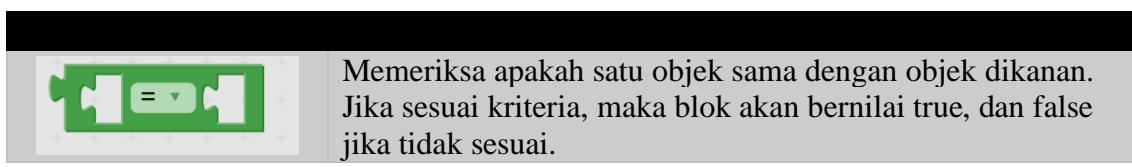


Pada blocks terdapat *codeblock* yang terdiri dari beberapa komponen seperti, Control, Logic, Math, Text, List, Colors, Variabel, Procedures dan juga codeblock sesuai dengan fitur dan komponen yang digunakan. Berikut ini penjelasan codeblocks yang digunakan pada modul ini:

#### 1) Control

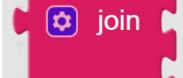


#### 2) Logic

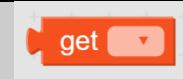


#### 3) Text



	Teks Kosong.
	Menggabungkan dua atau lebih teks.

#### 4) Variabel

	Mengambil variable global.
---	----------------------------

#### 5) Firebase\_Database

	Kondisi ketika fungsi Firebase Database mengalami perubahan data, maka akan mengeksekusi code tersebut.
--	---

#### 6) Label

	Untuk set komponen label yang digunakan untuk menuliskan nilai atau teks yang diinginkan.
---	---

Untuk penjelasan lebih rinci terkait codeblock yang ada di kodular dapat diakses dilaman berikut ini : <https://docs.kodular.io/components/>

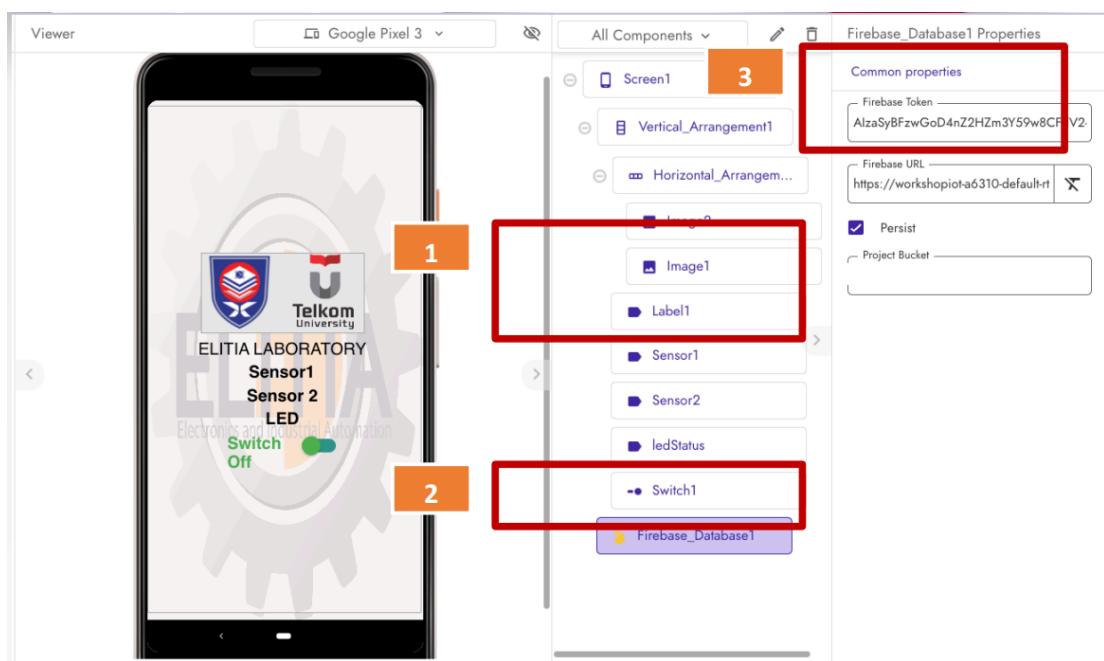
#### 2. 2. 4 Pembuatan Aplikasi

Aplikasi yang dibuat pada modul ini yaitu menghubungkan aplikasi dengan firebase untuk mendapatkan nilai dari sensor dan juga dapat melakukan control on/off untuk sensor tersebut.

##### 2.2.4.1 User Interface Aplikasi

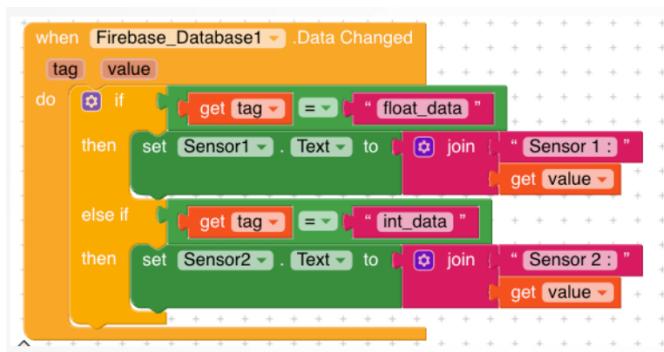
Pada aplikasi ini terdapat beberapa komponen dan fitur penting yang digunakan untuk membuat user interface pada aplikasi ini :

- 1) **Label:** Digunakan untuk mendapatkan informasi nilai data dari sensor,
- 2) **Switch:** Digunakan untuk mengkontrol LED pada posisi On / Off.
- 3) **Firebase\_Database:** Fitur yang digunakan untuk menyambungkan aplikasi ke firebase. Kemudian mengisi Firebase Token, firebase URL (didapat dibagian firebase) dan Project Bucket (Untuk bagian ini dikosongkan pada modul ini dikarenakan pada database di firebase yang digunakan untuk modul ini tidak di set nama project bucket nya).



#### 2.2.4.2 Codeblocks

Pada bagian *codeblocks* menggunakan fungsi `Firebase_Database1.Data_changed` yang digunakan untuk menampilkan data ketika data pada Firebase berubah atau *update* data.



#### 2.2.4.3 Hasil



# MODUL 3

## MQTT DENGAN MICROPYTHON MENGGUNAKAN RASPBERRY PICO W DAN ADAFRUIT IO MQTT BROKER

---

### TUJUAN

- 1) Mengetahui protokol MQTT.
- 2) Memahami proses komunikasi protokol MQTT dengan broker Adafruit IO.
- 3) Memahami IoT menggunakan python di mikrokontroler Raspberry Pi Pico W.

### *REQUIREMENT:*

- 1) Thonny MicroPython IDE,
- 2) Raspberry Pi Pico W,
- 3) Adafruit IO di Web,
- 4) Pemahaman dasar pemrograman Python.

### 3. 1 *Message Queuing Telemetry Transport (MQTT)*

*Message Queuing Telemetry Transport* (MQTT) merupakan protokol komunikasi Machine-to-Machine (M2M) yang menerapkan sistem komunikasi **Publish-Subscribe**. Protokol ini memiliki kelebihan yaitu proses transfer data yang ringan dan mudah sehingga sering digunakan pada perangkat IoT yang memiliki keterbatasan bandwith seperti perangkat *wearable*.

### 3. 2 **Kelebihan MQTT**

Protokol MQTT memiliki beberapa kelebihan yang menjadikannya sebagai standard dari perangkat IoT, beberapa diantaranya adalah:

#### 1) **Ringan dan efisien**

Penggunaan MQTT pada sistem IoT dapat diimplementasikan pada mikrokontroler kecil dikarenakan kecilnya data yang digunakan. Dengan kecilnya data tersebut, maka *bandwidth* dalam jaringan dapat dioptimalkan.

#### 2) **Scalable**

Skalabilitas pada MQTT dapat dilihat dari kecilnya daya yang dibutuhkan pada saat pengoperasiannya. Protokol ini juga memiliki fitur untuk dapat berkomunikasi dengan perangkat IoT dalam jumlah besar.

#### 3) **Reliable**

Sistem IoT memerlukan komunikasi dengan *latency* yang kecil sehingga komunikasi antar perangkat dapat berjalan dengan cepat, dengan MQTT maka kebutuhan tersebut dapat terpenuhi.

#### 4) **Well-supported**

MQTT memiliki banyak dukungan pada beberapa bahasa pemrograman, seperti python yang memiliki library khusus untuk implementasi protokol MQTT pada perangkat IoT. Oleh karena itu *developer* dapat mengembangkan perangkat IoT dengan optimal.

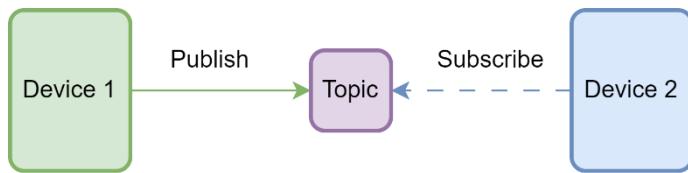
### 3. 3 **Alur Komunikasi MQTT**

Seperti yang telah disebutkan sebelumnya, komunikasi MQTT menerapkan sistem **Publish-Subscribe** untuk melakukan proses transfer data. Pertukaran data antar

perangkat yang dilakukan oleh MQTT merupakan **pesan/messages** yang berisi informasi seperti pembacaan sensor atau perintah.

### 3. 3. 1 Publish-Subscribe

Istilah **publish** dan **subscribe** di protokol MQTT ini digunakan oleh perangkat saat bertukar data. **Publish** berarti perangkat akan mengirimkan informasi ke Topic yang telah ditentukan yang kemudian akan diterima oleh perangkat yang telah **subscribe** Topic tersebut.



Seperti yang ditunjukkan pada **Gambar**, Device 1 melakukan proses publish ke topic dan device 2 men-subsrcibe topic untuk menerima seluruh data yang masuk ke topic tersebut. Saat Device 1 publish informasi ke Topic tertentu, device 2 akan menerima informasi secara langsung dari Topic tersebut.

### 3. 3. 2 Topics

Kita telah membahas tentang publish dan subscribe, namun **apa itu Topic?**

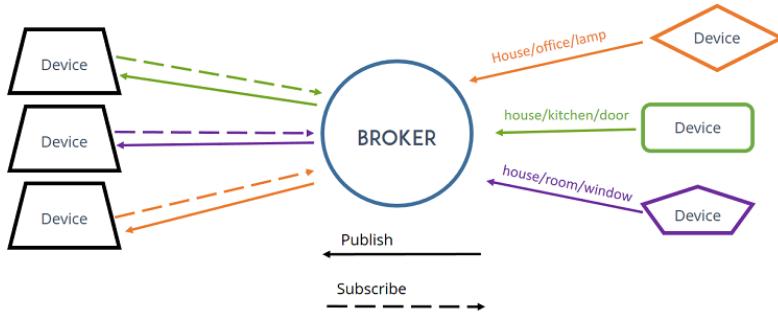
**Topic** dapat dianalogikan seperti channel seseorang yang ada di-youtube. Ketika penonton ingin mendapatkan informasi update terbaru dari channel tertentu, maka penonton akan subscribe ke channel tersebut sehingga setiap pemilik channel meng-update konten terbaru, penonton mendapatkan notifikasi dari channel yang di-subscribe. Maka di protokol MQTT, Topic memiliki fungsi yang sama seperti channel di youtube.

**Topic di MQTT direpresentasikan dalam bentuk String dan case sensitive** sehingga penulisan Topic harus diperhatikan huruf kapitalnya.

### 3. 3. 3 Broker

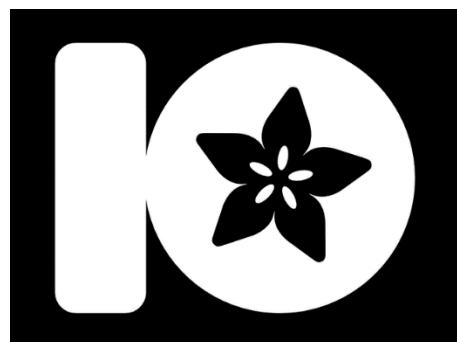
**Broker** merupakan hal yang sangat penting dalam mengatur alur informasi data di protokol MQTT. Broker berfungsi untuk menerima semua *messages*, menyaring, dan

mem-publish *messages* tersebut ke seluruh MQTT client yang telah subscribe topic pada broker tersebut.



Ada banyak platform MQTT broker yang dapat digunakan, mulai dari yang gratis seperti Adafruit IO, berbayar seperti HiveMQ, ataupun mengembangkan broker itu sendiri pada mikrokomputer seperti Raspberry Pi Zero W menggunakan software gratis seperti Mosquitto.

### 3. 4 Adafruit IO



Adafruit IO adalah platform yang memungkinkan pengguna untuk berinteraksi dengan data, terutama untuk sistem IoT. Salah satu layanan Adafruit IO adalah MQTT broker gratis yang dapat digunakan dengan batasan tertentu. Adafruit IO dapat diakses melalui link <https://io.adafruit.com/>.

### 3. 5 MicroPython

MicroPython adalah implementasi Python yang dioptimalkan untuk berjalan di mikrokontroler dan perangkat berdaya rendah. Diciptakan oleh Damien George, MicroPython memungkinkan pengembang untuk menggunakan bahasa pemrograman

Python pada perangkat keras yang memiliki sumber daya terbatas, seperti mikrokontroler, mikroprosesor, dan perangkat Internet of Things (IoT).

Berikut adalah beberapa fitur dan karakteristik utama dari MicroPython:

#### **1) Ringan dan Optimal**

MicroPython dirancang untuk berjalan pada perangkat keras yang memiliki sumber daya terbatas. Interpreter dan inti bahasa Python diperkecil untuk memastikan kinerja yang optimal pada mikrokontroler dengan kapasitas memori yang rendah.

#### **2) Dukungan untuk Mikrokontroler Populer**

MicroPython mendukung sejumlah mikrokontroler populer seperti ESP8266, ESP32, STM32, dan sebagainya. Ini memungkinkan pengembang untuk menggunakan bahasa Python pada berbagai perangkat keras yang umum digunakan dalam proyek mikrokontroler.

#### **3) Interaktif**

Salah satu fitur unik dari MicroPython adalah mode interaktifnya. Pengguna dapat berinteraksi langsung dengan perangkat keras menggunakan repl (read-eval-print loop) yang mirip dengan Python pada komputer desktop. Hal ini memudahkan pengembangan dan debugging.

#### **4) Dukungan Modul Standar Python**

Meskipun dibuat untuk mikrokontroler, MicroPython mencakup sejumlah modul standar Python yang dapat ditemui di implementasi Python lainnya. Meskipun modul tersebut dapat berbeda dalam ukuran atau implementasi khusus, banyak modul standar dapat digunakan di MicroPython tanpa modifikasi.

#### **5) Kemampuan I/O**

MicroPython menyediakan antarmuka untuk mengakses pin I/O dan perangkat keras lainnya pada mikrokontroler, memungkinkan pengembang untuk mengendalikan sensor, motor, dan perangkat keras lainnya.

#### 6) Dukungan WiFi dan Jaringan

Beberapa implementasi MicroPython, terutama pada mikrokontroler seperti ESP8266 dan ESP32, menyediakan dukungan untuk konektivitas WiFi dan jaringan, memungkinkan perangkat untuk terhubung dengan internet.

#### 7) Komunitas dan Ekosistem yang Berkembang

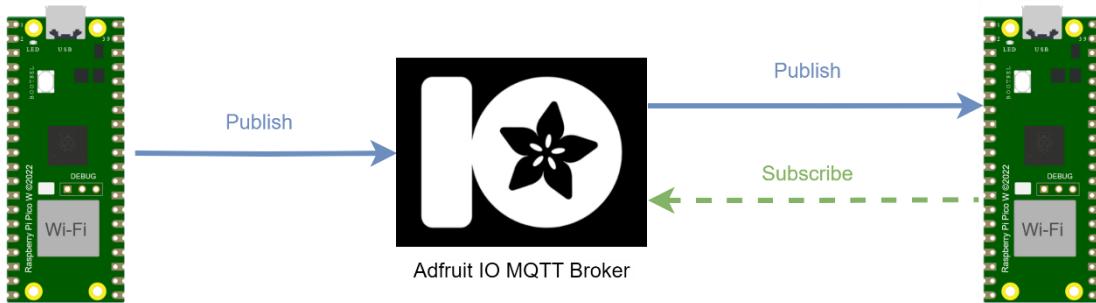
MicroPython memiliki komunitas pengembang yang aktif, dan banyak sumber daya dan tutorial yang tersedia secara online. Ini membuatnya lebih mudah bagi pengembang untuk memulai dengan MicroPython dan memecahkan masalah yang mereka hadapi.

MicroPython memberikan kemudahan pengembangan dan kekuatan bahasa Python pada mikrokontroler, membuatnya menjadi pilihan yang menarik untuk proyek-proyek IoT dan pengembangan perangkat keras berdaya rendah.

### 3. 6 Raspberry Pi Pico W

Raspberry Pi Pico W adalah mikrokontroler berkinerja tinggi dan berbiaya rendah dengan antarmuka nirkabel tunggal-band 2.4GHz dan **menggunakan MicroPython** sebagai bahasa pemrograman *default*-nya. Mikrokontroler ini memiliki prosesor Dual-core Arm Cortex M0+, 264kB SRAM on-chip, dan 2MB memori flash on-board. Selain itu, Pico W menambahkan antarmuka nirkabel on-board tunggal-band 2.4GHz (802.11n) dan Bluetooth 5.2. Antarmuka nirkabel ini terhubung melalui SPI ke mikrokontroler RP2040. Untuk diagram *pinout* dari Raspberry Pi Pico W dapat dilihat pada **LAMPIRAN PINOUT DIAGRAM MIKROKONTROLER**.

### 3. 7 Implementasi Protokol MQTT menggunakan Raspberry Pi Pico W dan Adafruit IO MQTT Broker



Implementasi protokol MQTT menggunakan Pico W dan Adafruit IO MQTT Broker ini memiliki alur:

1. Device 1 akan publish ke Topic 'A' MQTT Broker.
2. Device 2 akan Subscribe ke Topic 'A' MQTT Broker.
3. MQTT Broker akan Publish ke device 2 karena device 2 telah subscribe ke topic yang ditentukan.

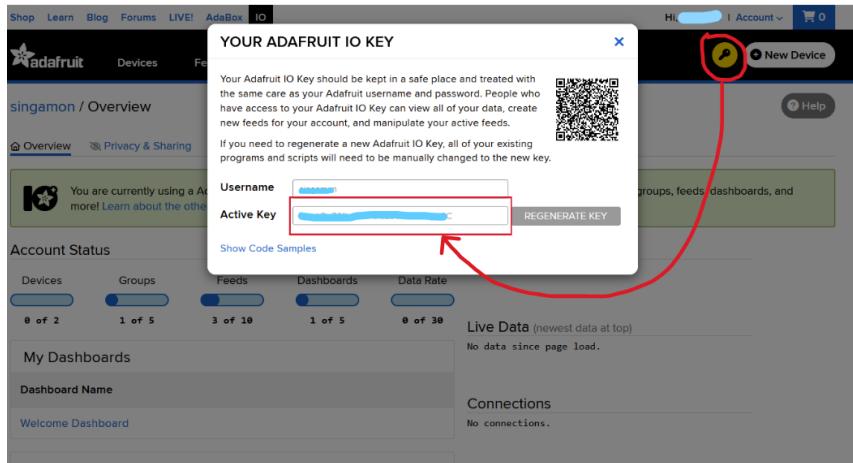
#### 3. 7. 1 Setup Adafruit IO MQTT Broker

##### 1. Sign Up Account

Hal yang pertama kali dilakukan untuk setup Adafruit IO MQTT Broker adalah dengan melakukan sign up account terlebih dahulu dengan mengakses link [https://accounts.adafruit.com/users/sign\\_up](https://accounts.adafruit.com/users/sign_up).

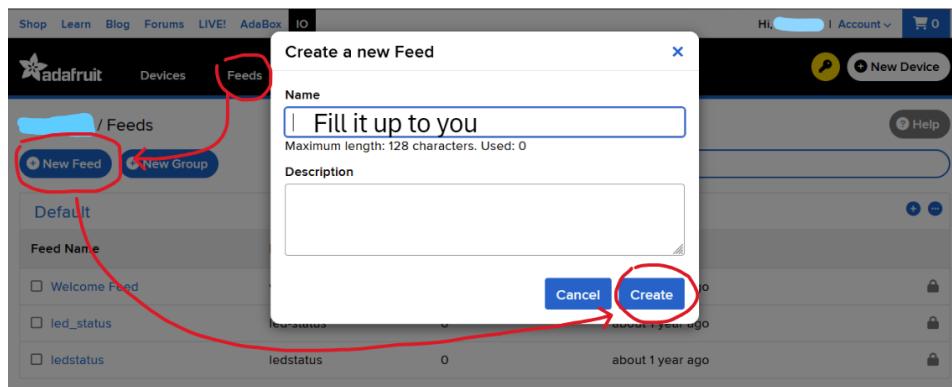
##### 2. Adafruit IO Key

Setelah berhasil sign up, maka selanjutnya adalah mengambil Adafruit IO key yang akan digunakan oleh mikrokontroler agar dapat mengakses Adafruit IO yang digunakan.



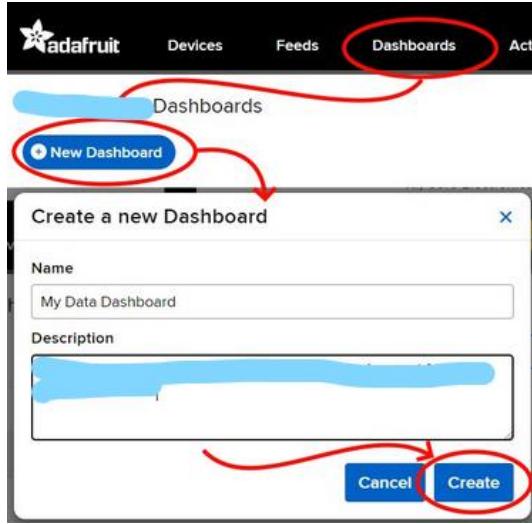
### 3. Membuat feed

**Feed** merupakan topic yang akan digunakan pada proses publish dan subscribe, sehingga kita harus membuat feed terlebih dahulu sebelum melakukan proses pertukaran data.

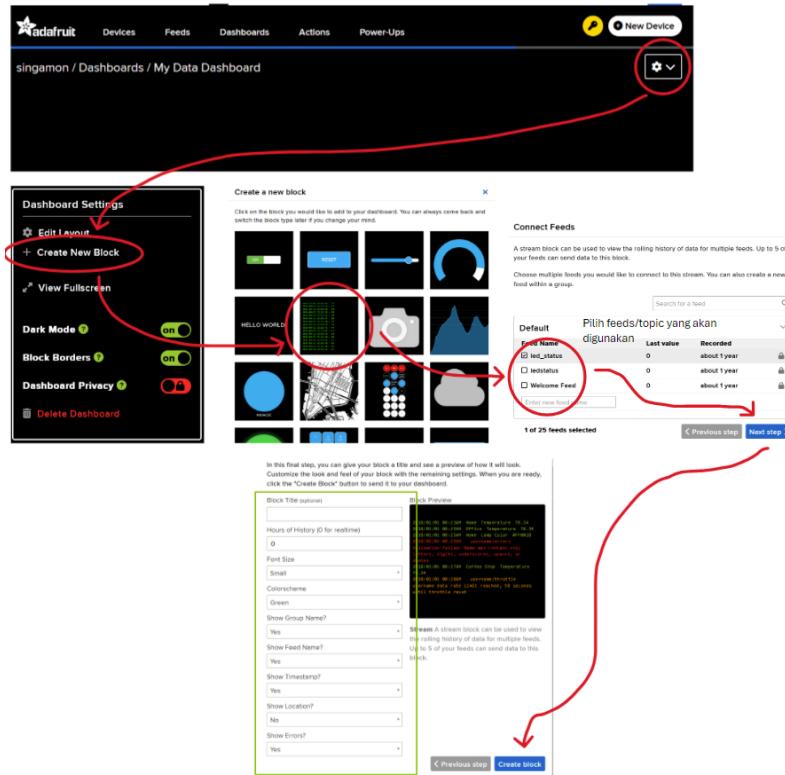


### 4. Membuat dashboard

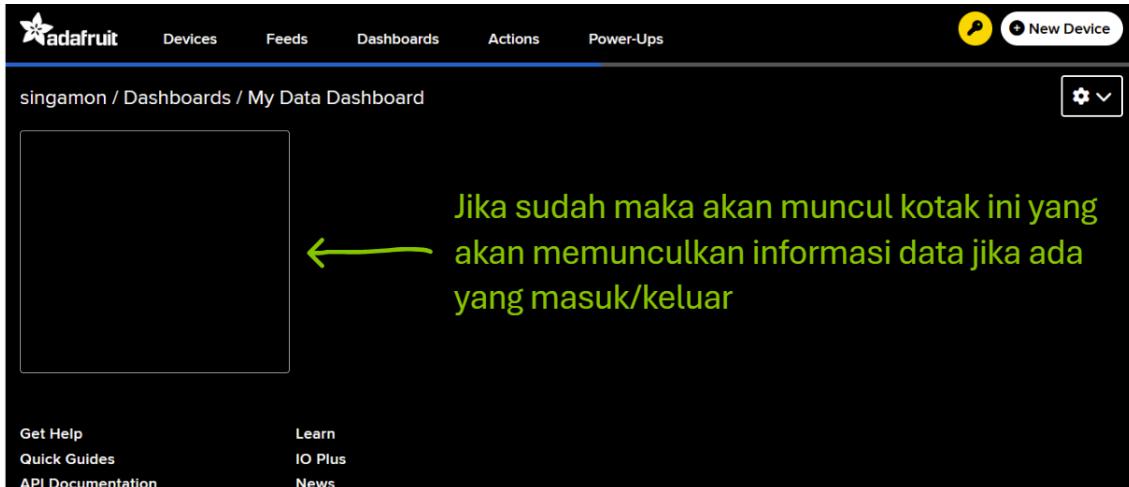
Dashboard digunakan sebagai UI dalam melakukan pengolahan data yang masuk maupun yang akan dipublish. Untuk membuat dashboard baru dapat dilakukan dengan cara berikut:



Setelah menambahkan dashboard, buka dashboard tersebut dan lakukan langkah berikut untuk menambahkan blok yang dapat melihat informasi yang mengalir pada topik yang dipilih.



Jika sudah melakukan hal di atas, maka tampilan dashboard akan seperti gambar di bawah, dan setup adafruit IO telah selesai.

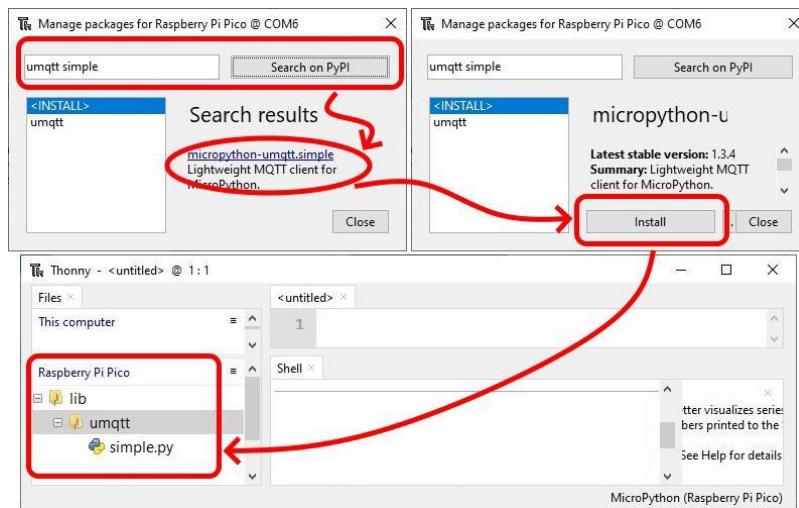


### 3. 7. 2 Setup Raspberry Pi Pico W untuk MQTT

Library yang digunakan untuk menggunakan protokol MQTT di Pico W adalah micropython-umqtt.simple.

Berikut adalah cara untuk menambahkan library pada Thonny Micropython:

Tools -> Manage Packages



### 3. 7. 3 *Source code* implementasi protokol MQTT menggunakan Micropython

#### 1.7.3.1 Publish

##### 1) Kebutuhan library dan variabel

Library yang dibutuhkan untuk menggunakan protokol mqtt pada micropython adalah library **umqtt.simple**.

<i>Source code import library</i>
import network import time from umqtt.simple import MQTTClient import utime

Setelah import library, maka langkah selanjutnya yang akan dilakukan adalah mendeklarasikan variabel yang akan digunakan. Variabel yang digunakan antara lain **variabel untuk terhubung ke jaringan Wifi**, **variabel untuk MQTT client**, dan **variabel yang berisi nilai yang akan di-publish** pada kodingan di bawah terdapat beberapa variabel yang dapat kalian isi sesuai dengan sistem kalian.

<i>Source code variabel yang dibutuhkan pada contoh kali ini</i>
# Fill in your WiFi network name (ssid) and password here: wifi_ssid = "<YOUR WIFI SSID>" wifi_password = "<YOUR WIFI PASSWORD>"  # Fill in your Adafruit IO Authentication and Feed MQTT Topic details mqtt_host = "io.adafruit.com" mqtt_username = "<YOUR ADAFRUIT IO USERNAME HERE>" # Your Adafruit IO username mqtt_password = "<YOUR Adafruit_IO_Key>" # Adafruit IO Key mqtt_publish_topic = "<YOUR mqtt_topic/feed_directory>" # The MQTT topic for your Adafruit IO Feed  # Enter a random ID for this MQTT Client # It needs to be globally unique across all of Adafruit IO. mqtt_client_id = "uus9asndajsndiqiwd10wid01n1kwd"  # Value that will be published counter = 0

## 2) Menghubungkan Pico W ke Wifi dan MQTT server

Setelah variabel dideklarasikan, maka langkah selanjutnya adalah dengan menghubungkan Pico W ke wifi dan MQTT server. Fungsi yang digunakan untuk terhubung ke wifi adalah:

```
wlan.connect(<String your_wifi_ssid >,<String your_wifi_password>)
```

dan fungsi untuk terhubung ke MQTT client adalah:

```
mqtt_client.connect()
```

*Source code untuk terhubung ke Wifi dan MQTT server*

```
# Connect to WiFi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(wifi_ssid, wifi_password)
while wlan.isconnected() == False:
    print('Waiting for connection...')
    time.sleep(1)
print("Connected to WiFi")

# Initialize our MQTTClient and connect to the MQTT server
mqtt_client = MQTTClient(
    client_id= mqtt_client_id,
    server= mqtt_host,
    user= mqtt_username,
    password= mqtt_password)

mqtt_client.connect()
```

## 3) Kodingan try

Pada kodingan di bawah ini, maka Pico W akan melakukan publish informasi dari variabel `count` menggunakan fungsi:

```
mqtt_client.publish(<Topic yang dituju>, str(<Nilai yang akan di-
publish>))
```

*Source code try (sama seperti loop pada arduino).*

```
try:
    while True:
        # Generate some dummy data that changes every loop
        counter += 3
```

```

# Get the current timestamp
current_time = utime.localtime()
timestamp = "{:04}-{:02}-{:02}"
{:02}:{:02}:{:02}" .format(
    current_time[0], current_time[1], current_time[2],
    current_time[3], current_time[4], current_time[5]
)

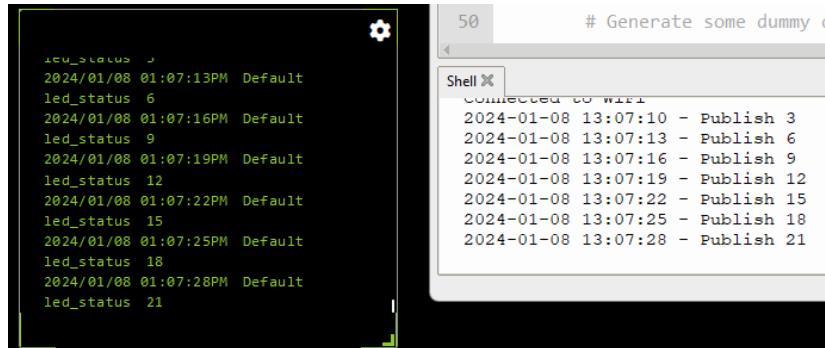
# Publish the data to the topic!
print(f'{timestamp} - Publish {counter}')
mqtt_client.publish(mqtt_publish_topic, str(counter))

# Delay a bit to avoid hitting the rate limit
time.sleep(3)
except Exception as e:
    print(f'Failed to publish message: {e}')
finally:
    mqtt_client.disconnect()

```

Pengiriman/publish informasi ke mqtt server tersebut akan berulang setiap 3 detik menggunakan fungsi **time.sleep(3)**.

#### 4) Hasil output sistem



##### 1.7.3.2 Subscribe

###### 1) Kebutuhan library dan variabel

###### Source code import library

```

import network
import time
from umqtt.simple import MQTTClient
import utime

```

*Source code variabel yang dibutuhkan pada contoh kali ini*

```
# Fill in your WiFi network name (ssid) and password here:  
wifi_ssid = "<YOUR WIFI SSID>"  
wifi_password = "<YOUR WIFI PASSWORD>"  
  
# Fill in your Adafruit IO Authentication and Feed MQTT Topic  
# details  
mqtt_host = "io.adafruit.com"  
mqtt_username = "<YOUR ADAFRUIT IO USERNAME HERE>" # Your  
Adafruit IO username  
mqtt_password = "<YOUR Adafruit_IO_Key>" # Adafruit IO Key  
mqtt_receive_topic = "<YOUR mqtt_topic/feed_directory>" # The  
MQTT topic for your Adafruit IO Feed  
  
# Enter a random ID for this MQTT Client  
# It needs to be globally unique across all of Adafruit IO.  
mqtt_client_id = "iij0dfq8f012nif2jnfijn"
```

## 2) Menghubungkan Pico W ke Wifi dan MQTT server

*Source code untuk terhubung ke Wifi dan MQTT server*

```
# Connect to WiFi  
wlan = network.WLAN(network.STA_IF)  
wlan.active(True)  
wlan.connect(wifi_ssid, wifi_password)  
while wlan.isconnected() == False:  
    print('Waiting for connection...')  
    time.sleep(1)  
print("Connected to WiFi")  
  
# Initialize our MQTTClient and connect to the MQTT server  
mqtt_client = MQTTClient(  
    client_id= mqtt_client_id,  
    server= mqtt_host,  
    user= mqtt_username,  
    password= mqtt_password)  
  
mqtt_client.connect()
```

## 3) Fungsi *callback* untuk

Perbedaan *source code* subscribe dan publish terletak pada fungsi callback yang akan dipanggil setiap ada informasi baru yang datang di Topic yang disubscribe

*Source code fungsi callback untuk subscribe mqtt agar dapat menampilkan pesan saat ada pesan baru di topic yang ditentukan*

```
# So that we can respond to messages on an MQTT topic, we need
# a callback
# function that will handle the messages.
def mqtt_subscription_callback(topic, message):
    # Get the current timestamp
    current_time = utime.localtime()
    timestamp = "{:04}-{:02}-{:02} {:02}:{:02}:{:02}".format(
        current_time[0], current_time[1], current_time[2],
        current_time[3], current_time[4], current_time[5]
    )

    print(f'{timestamp} - Topic {topic} received message
{message}') # Debug print out of what was received over MQTT

# Before connecting, tell the MQTT client to use the callback
mqtt_client.set_callback(mqtt_subscription_callback)
mqtt_client.connect()
```

Setelah membuat fungsi callback, selanjutnya adalah subscribe ke Topic yang dipilih menggunakan fungsi:

`mqtt_client.subscribe(<String Topic_yang_di-subscribe>)`

*Source code fungsi callback untuk subscribe mqtt agar dapat menampilkan pesan saat ada pesan baru di topic yang ditentukan*

```
# Once connected, subscribe to the MQTT topic
mqtt_client.subscribe(mqtt_receive_topic)
print("Connected and subscribed")
```

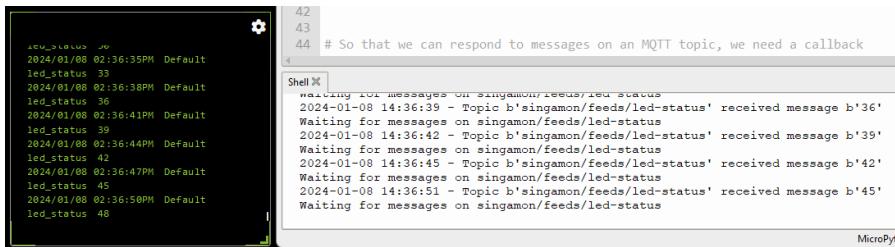
#### 4) Kodingan try

*Source code try (sama seperti loop pada arduino).*

```
try:
    while True:
        # Infinitely wait for messages on the topic.
        # Note wait_msg() is a blocking call, if you're doing
        multiple things
        # on the Pico you may want to look at putting this on
        another thread.
        print(f'Waiting for messages on {mqtt_receive_topic}')
        mqtt_client.wait_msg()
except Exception as e:
```

```
    print(f'Failed to wait for MQTT messages: {e}')
finally:
    mqtt_client.disconnect()
```

## 5) Hasil output sistem



The screenshot shows two terminal windows. The left window is a MicroPython REPL showing a loop that prints 'led\_status' values every second. The right window is a shell interface on a singamon broker, showing messages being received on the 'singamon/feed/led-status' topic.

MicroPython REPL Output:

```
led_status 36
2024/01/08 02:36:35PM Default
led_status 33
2024/01/08 02:36:38PM Default
led_status 36
2024/01/08 02:36:41PM Default
led_status 39
2024/01/08 02:36:44PM Default
led_status 42
2024/01/08 02:36:47PM Default
led_status 45
2024/01/08 02:36:50PM Default
led_status 48
```

Shell on singamon broker:

```
42
43
44 # So that we can respond to messages on an MQTT topic, we need a callback
4
Shell [x]
Waiting for messages on singamon/feed/led-status
2024-01-08 14:36:39 - Topic b'singamon/feed/led-status' received message b'36'
Waiting for messages on singamon/feed/led-status
2024-01-08 14:36:42 - Topic b'singamon/feed/led-status' received message b'39'
Waiting for messages on singamon/feed/led-status
2024-01-08 14:36:45 - Topic b'singamon/feed/led-status' received message b'42'
Waiting for messages on singamon/feed/led-status
2024-01-08 14:36:51 - Topic b'singamon/feed/led-status' received message b'45'
Waiting for messages on singamon/feed/led-status
```

# MODUL 4

## KOMUNIKASI LONG RANGE (LoRa)

---

### TUJUAN

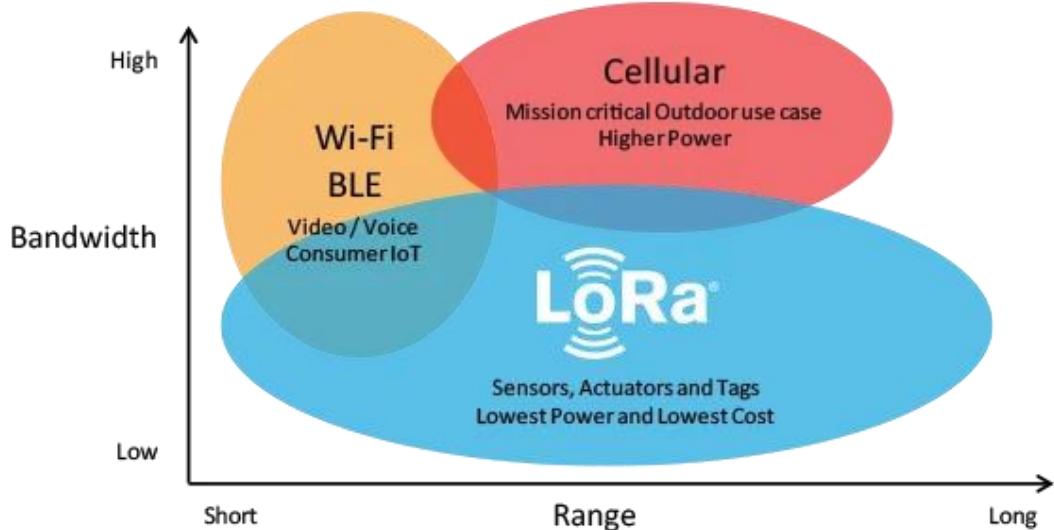
- 1) Mengetahui protokol *Long Range* (LoRa).
- 2) Memahami komunikasi protokol LoRa.

### ***REQUIREMENT:***

- 1) Arduino IDE/PlatformIO,
- 2) LoRa Module/Cosmic LoRa Aurora V2 ESP32 LoRa Module,
- 3) LoRa Library by Sandeep Mistry,
- 4) Sensor (opsional),
- 5) Aktuator (opsional).

#### 4.1 LoRa

LoRa (Long Range), adalah suatu protokol komunikasi nirkabel berdaya rendah dan memiliki jarak pengiriman yang jauh antar perangkat yang terhubung. LoRa beroperasi pada spektrum frekuensi sub-GHz, dan dikenal karena kemampuannya mengirimkan data dalam rentang yang lebih luas dengan mengonsumsi daya yang sangat sedikit, sehingga ideal untuk aplikasi yang memerlukan kecepatan data rendah dan masa pakai baterai yang lama. seperti pertanian pintar, kota pintar, pemantauan industri, dan penginderaan lingkungan.



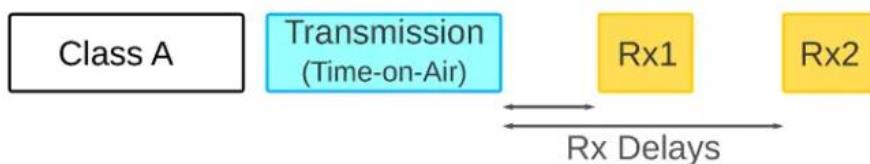
LoRa menggunakan teknik modulasi spread spectrum yang memanfaatkan Chirp Spread Spectrum (CSS), suatu teknik modulasi linier, ditambah dengan Forward Error Correction (FEC) untuk mencapai kemampuan jangka panjangnya dan menggunakan topologi star-of-stars, di mana kumpulan dari end-devices berkomunikasi dengan gateway pusat, memungkinkan penerapan jaringan IoT yang terukur dan hemat biaya. Sifat teknologi yang terbuka dan terstandarisasi, ditambah dengan jangkauan jangka panjang dan pengoperasian yang hemat energi, menjadikan LoRa pilihan populer untuk membangun solusi IoT berskala besar dan berbiaya rendah yang menghubungkan beragam perangkat, menyediakan data berharga untuk berbagai kebutuhan aplikasi.

## 4.2 Tipe Class LoRa

Class LoRa mengacu pada sistem klasifikasi untuk perangkat LoRaWAN, dan mengelompokkannya ke dalam kelas berbeda berdasarkan kemampuan dan fitur hemat dayanya, LoRa class mencakup Kelas A, Kelas B, dan Kelas C. Kelas ini menentukan cara perangkat berkomunikasi dan bagaimana perangkat menghemat daya.

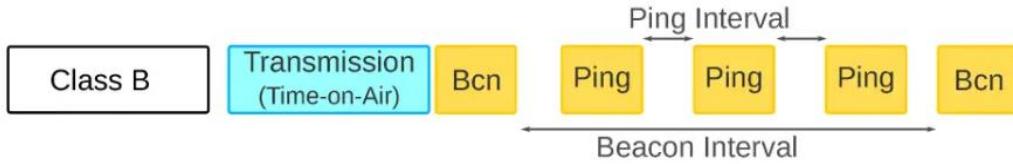
- **Class A**

LoRa class A terkenal dengan efisiensi dayanya yang tinggi. Perangkat ini memiliki memilih delay sedikit setelah mengirim data setelah setiap transmisi, menjadikannya ideal untuk sensor berteknologi baterai dan perangkat yang terutama berfokus pada pengiriman data. Meskipun dapat menghemat daya secara efektif, perangkat Kelas A mungkin tidak dapat menerima data sesering kelas lainnya, namun perangkat ini memberikan keseimbangan antara penghematan daya dan penerimaan data sesekali, sehingga cocok untuk berbagai aplikasi IoT.



- **Class B**

LoRa class B dirancang untuk menyediakan jendela penerimaan terjadwal yang disinkronkan dengan waktu jaringan, sehingga cocok untuk aplikasi dengan persyaratan komunikasi spesifik dan sensitif terhadap waktu. Perangkat ini menawarkan **peluang komunikasi yang lebih dapat diprediksi** dibandingkan perangkat Kelas A, karena perangkat ini menggabungkan fitur hemat daya sekaligus memungkinkan penerimaan data terjadwal dan terputus-putus, sehingga memungkinkan perangkat tersebut melayani aplikasi yang memerlukan pengiriman data tepat waktu secara efisien sambil tetap menghemat daya untuk masa pakai baterai yang lebih lama.



- **Class C**

Lora class C dikenal karena kemampuan penerimaannya yang hampir terus menerus dengan hanya interupsi singkat selama transmisi. Perangkat ini cocok untuk aplikasi yang memerlukan komunikasi dua arah hampir real-time, karena perangkat ini dapat menerima data hampir kapan saja. Namun, perangkat ini kurang hemat daya dibandingkan perangkat Kelas A dan Kelas B, sehingga lebih cocok untuk aplikasi dengan sumber daya berkelanjutan atau anggaran daya lebih tinggi yang mengutamakan komunikasi yang konsisten.



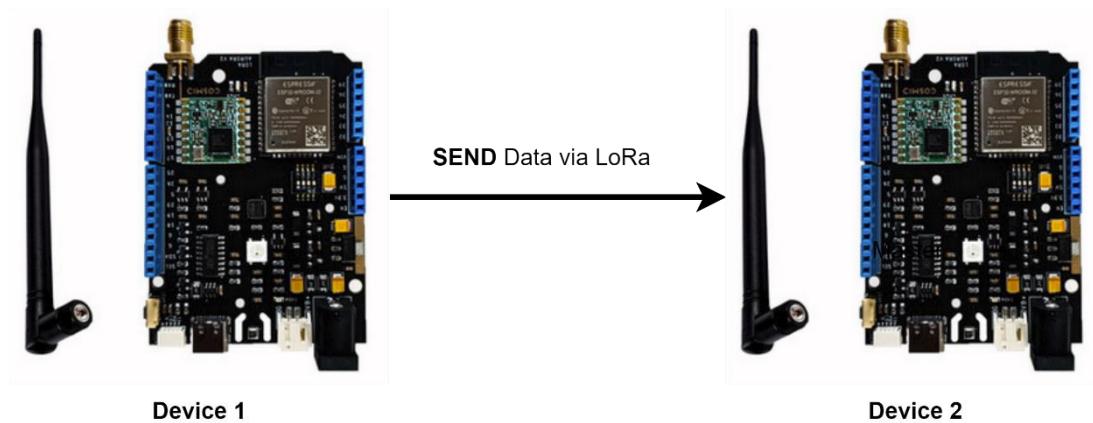
### 4.3 Alur Komunikasi LoRa

Dalam aliran komunikasi LoRa, perangkat akhir menggunakan pita frekuensi radio tertentu yang tidak berlisensi, seperti untuk **Asia adalah 433 MHz**, untuk **Amerika 902-928 MHz**, atau **Eropa 863-870 MHz** untuk mengirimkan data secara nirkabel. Pilihan frekuensi mempengaruhi jangkauan dan kinerja komunikasi. Perangkat akhir, seperti sensor IoT atau objek yang terhubung, menggunakan modulasi LoRa untuk mengubah data yang dikodekan menjadi sinyal chirp, yang ditransmisikan melalui gelombang udara. LoRa dapat mengirim data dengan menggunakan teknik modulasi yang memungkinkan transmisi informasi secara efisien dalam jarak jauh sekaligus menghemat daya.

LoRa menggunakan metode yang dikenal sebagai modulasi **Chirp Spread Spectrum (CSS)**, di mana data yang akan dikirim dikodekan menjadi sinyal chirp – sapuan frekuensi berkelanjutan yang frekuensinya bervariasi dari waktu ke waktu. Sinyal chirp ini sangat tahan terhadap gangguan dan dapat menempuh jarak jauh, bahkan di

lingkungan yang menantang dengan rintangan dan gangguan seperti pohon dan bangunan tinggi. Perangkat ini biasanya bertenaga baterai untuk masa pakai operasional yang lebih lama dan menggunakan teknik modulasi LoRa untuk mengirim data secara efisien dalam jarak jauh sambil mengonsumsi daya minimal. Data yang dikirimkan diterima oleh gateway LoRa, yang bertindak sebagai perantara dalam jaringan. Gateway ini mengumpulkan data dari beberapa perangkat akhir dan menyampaikannya ke server jaringan pusat, di mana data tersebut dapat diproses dan dialihkan ke tujuan atau aplikasi yang sesuai. Arsitektur gateway-ke-server ini memungkinkan agregasi dan pengelolaan data yang efisien dari berbagai perangkat akhir dalam jaringan LoRa.

#### 4.4 Implementasi *Peer-to-Peer* Menggunakan Cosmic LoRa Aurora V2



Implementasi *peer-to-peer* ini akan menggunakan protokol LoRa dengan alur seperti pada gambar di atas.

##### 4.4.1 Cosmic LoRa Aurora V2

Cosmic LoRa Aurora V2 merupakan *dev board* LoRa yang menggunakan *chip* LoRa(WAN) RFM95W yang dikeluarkan oleh Cosmic dan dapat dilihat pada dapat dilihat pada **LAMPIRAN PINOUT DIAGRAM MIKROKONTROLER Cosmic LoRa Aurora V2**. *Board* ini memakai ESP32 sebagai mikrokontrolernya sehingga selain memiliki LoRa, *board* ini juga sudah dapat menggunakan konektivitas Wifi. Selain itu, LoRa Aurora V2 ini memiliki 3 sumber tenaga yang dapat digunakan, yaitu USB, LiPo Battery, dan DC Jack dan sudah memiliki sistem *smart power switching* sehingga secara

otomatis dapat memilih sumber tenaga yang digunakan, seperti jika LiPo *Battery* sedang diisi, maka sistem dapat secara otomatis berganti ke USB atau DC Jack.

#### 4.4.2 LoRa Library by Sandeep Mistry

*Library* LoRa yang dikembangkan oleh Sandeep Mistry ini adalah *library* yang akan digunakan pada implementasi LoRa menggunakan Cosmic LoRa Aurora V2. *Library* ini merupakan *library* yang digunakan untuk mengirim dan menerima data menggunakan komunikasi LoRa dengan *header* **LoRa.h**.

#### 4.4.3 Source Code LoRa peer-to-peer

Pada *source code* LoRa *peer-to-peer* baik untuk *transmitter* dan *receiver* menggunakan dokumentasi contoh kode yang diberikan oleh Cosmic yang terdapat ada laman web <https://github.com/cosmic-id/cosmic-lora-aurora/blob/main/examples/aurora-v2-lora-p2p-receiver/aurora-v2-lora-p2p-receiver.ino>.

##### 4.4.3.1 Transmitter

###### 1) Inisialisasi

Inisialisasi *Source code* untuk *Transmitter* pada implementasi *peer-to-peer* menggunakan LoRa pada *board* Cosmic LoRa Aurora V2

```
*****
Modified from the examples of the Arduino LoRa library
More resources: https://github.com/cosmic-id/
*****
// Dalam receiver dan transmitter, menggunakan 2 library yaitu
SPI.h dan LoRa.h by Sandeep Mistry
#include <SPI.h>
#include <LoRa.h>

// Tentukan pin yang digunakan oleh modul receiver
#define LORA_AURORA_V2_NSS 15
#define LORA_AURORA_V2_RST 0
#define LORA_AURORA_V2_DIO0 27
#define LORA_AURORA_V2_EN 32

#define LORA_TX_POWER 20
#define LORA_SPREADING_FACTOR 12

int counter = 0;
```

Dalam menginisialisasi P2P LoRa, kita akan menggunakan library **SPI.h** dan **LoRa.h** oleh Sandeep Mistry. Setelah itu tentukan pin-pin LoRa mana yang digunakan di setiap **#define**. Pada kode di atas ada variable tambahan yang diinisialisasi, yaitu variable **counter** yang akan digunakan sebagai nilai yang akan dikirim ke *receiver*.

## 2) Setup

*Source code setup() untuk Transmitter pada implementasi peer-to-peer menggunakan LoRa pada board Cosmic LoRa Aurora V2*

```
void setup() {
    // Inisialisasi mode pin LoRa
    pinMode(LORA_AURORA_V2_EN, OUTPUT);
    // LoRa chip dinyalakan
    digitalWrite(LORA_AURORA_V2_EN, HIGH);

    // Inisialisasi Serial Monitor
    Serial.begin(115200);
    while (!Serial);
    Serial.println("LoRa Sender");

    // setup modul receiver LoRa
    LoRa.setPins(LORA_AURORA_V2_NSS, LORA_AURORA_V2_RST,
LORA_AURORA_V2_DIO0);

    // Ganti LoRa.begin(---E-) argument dengan frekuensi yang
cocok di tempat kamu
    // 433E6 for Asia
    // 866E6 for Europe
    // 915E6 for North America
    while (!LoRa.begin(920E6)) {
        Serial.println(".");
        delay(500);
    }

    LoRa.setTxPower(LORA_TX_POWER, PA_OUTPUT_PA_BOOST_PIN);
    LoRa.setSpreadingFactor(LORA_SPREADING_FACTOR);

    // Ganti sync word (0xF3) menyesuaikan transceiver
    // sync word memastikan LoRa tidak mendapatkan data dari LoRa
transceivers yang lain
    // memiliki range dari 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}
```

Pada bagian **setup()** untuk LoRa sebagai *receiver*, ada beberapa hal yang harus kita tentukan:

- a. **Deklarasi pinMode untuk pin Enable** (Berdasarkan kode, maka pin enable terdapat pada konstanta **LORA\_AURORA\_V2\_EN**) dan mengubah *state*-nya menjadi HIGH.
- b. **Setup pin untuk modul LoRa** menggunakan fungsi **LoRa.setPins(NSS\_PIN, RST\_PIN, DI00\_PIN)**.
- c. **Memulai modul LoRa** menggunakan fungsi **LoRa.begin(frequency)** dengan parameter **frequency** diisi dengan frekuensi daerah LoRa yang akan kita gunakan.
  - 433E6 untuk Asia
  - 866E6 untuk Eropa
  - 915E6 untuk Amerika Utara (NA)
- d. **Menentukan sync word atau alamat LoRa** yang disesuaikan antara pengirim (*transceivers*) dan penerima (*receiver*). Sync word memiliki range antara 0-0xFF dan ditentukan menggunakan fungsi **LoRa.setSyncWord(sync\_word)** dengan parameter **sync\_word** diisi dengan sync word yang akan digunakan.

### 3) Loop

*Source code loop() untuk Transceiver pada implementasi peer-to-peer menggunakan LoRa pada board Cosmic LoRa Aurora V2*

```
void loop() {
    Serial.print("Sending packet: ");
    Serial.println(counter);

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();

    counter++;

    delay(5000);
}
```

#### 4.4.3.2 Receiver

##### 1) Inisialisasi

Inisialisasi pada *Source code* untuk *Receiver* pada implementasi *peer-to-peer* menggunakan LoRa pada *board* Cosmic LoRa Aurora V2

```
*****  
Modified from the examples of the Arduino LoRa library  
More resources: https://github.com/cosmic-id/  
*****  
// Dalam receiver dan transmitter, menggunakan 2 library yaitu  
SPI.h dan LoRa.h by Sandeep Mistry  
#include <SPI.h>  
#include <LoRa.h>  
  
// Tentukan pin yang digunakan oleh modul receiver  
#define LORA_AURORA_V2_NSS 15  
#define LORA_AURORA_V2_RST 0  
#define LORA_AURORA_V2_DIO0 27  
#define LORA_AURORA_V2_EN 32  
  
#define LORA_TX_POWER 20  
#define LORA_SPREADING_FACTOR 12
```

Tidak ada perbedaan antara inisialisasi untuk *transmitter* dan *receiver*, seperti pin dan *library* yang digunakan. Namun, pada kode di atas tidak ada tambahan variable **counter** seperti pada *transmitter* yang digunakan sebagai data yang akan dikirim melalui LoRa.

##### 2) Setup

*Source code* `setup()` untuk *Receiver* pada implementasi *peer-to-peer* menggunakan LoRa pada *board* Cosmic LoRa Aurora V2

```
void setup() {  
    // Inisialisasi mode pin LoRa  
    pinMode(LORA_AURORA_V2_EN, OUTPUT);  
    // LoRa chip dinyalakan  
    digitalWrite(LORA_AURORA_V2_EN, HIGH);  
  
    // Inisialisasi Serial Monitor  
    Serial.begin(115200);  
    while (!Serial);  
    Serial.println("LoRa Receiver");  
  
    // setup modul receiver LoRa
```

```
LoRa.setPins(LORA_AURORA_V2_NSS, LORA_AURORA_V2_RST,  
LORA_AURORA_V2_DIO0);  
  
// Ganti LoRa.begin(---E-) argument dengan frekuensi yang  
cocok di tempat kamu  
// 433E6 for Asia  
// 866E6 for Europe  
// 915E6 for North America  
while (!LoRa.begin(920E6)) {  
    Serial.println(".");
    delay(500);
}  
  
LoRa.setTxPower(LORA_TX_POWER, PA_OUTPUT_PA_BOOST_PIN);  
LoRa.setSpreadingFactor(LORA_SPREADING_FACTOR);  
  
// Ganti sync word (0xF3) menyesuaikan transceiver  
// sync word memastikan LoRa tidak mendapatkan data dari  
LoRa transceivers yang lain  
// memiliki range dari 0-0xFF  
LoRa.setSyncWord(0xF3);  
Serial.println("LoRa Initializing OK!");  
}
```

Pada bagian **setup()** di sisi *receiver*, perlu diperhatikan pada bagian frekuensi dan *sync word* yang digunakan. Kedua hal tersebut harus memiliki nilai yang sama baik dari sisi pengirim (*transmitter*) dan penerima (*receiver*). Pada *source code* kali ini, frekuensi yang digunakan adalah 920E6 dan untuk *sync word* menggunakan nilai 0xF3.

### 3) Loop

<i>Source code loop() untuk Receiver pada implementasi peer-to-peer menggunakan LoRa pada board Cosmic LoRa Aurora V2</i>
<pre>void loop() {     // Mecoba untuk parse packet yang diterima     int packetSize = LoRa.parsePacket();     if (packetSize) {         // received a packet         Serial.print("Received packet '");          // Membaca packet yang diterima         while (LoRa.available()) {             String LoRaData = LoRa.readString();             Serial.print(LoRaData);         }     } }</pre>

```
// Menampilkan RSSI dari packet yang diterima
Serial.print("' with RSSI ");
Serial.println(LoRa.packetRssi());
}
}
```

Pada bagian **loop()**, *receiver* akan menerima data dari *transceiver* yang memiliki *sync word* yang sama. **LoRa.parsePacket()** akan memeriksa paket yang datang. Jika ada paket yang datang, maka paket akan dibaca menggunakan **LoRa.readString()** selama paket tersedia sesuai dengan kondisi **LoRa.available()**. Pada bagian terakhir, **LoRa.packetRSSI()** akan mendapatkan RSSI dari paket LoRa yang didapat dalam satuan dB.

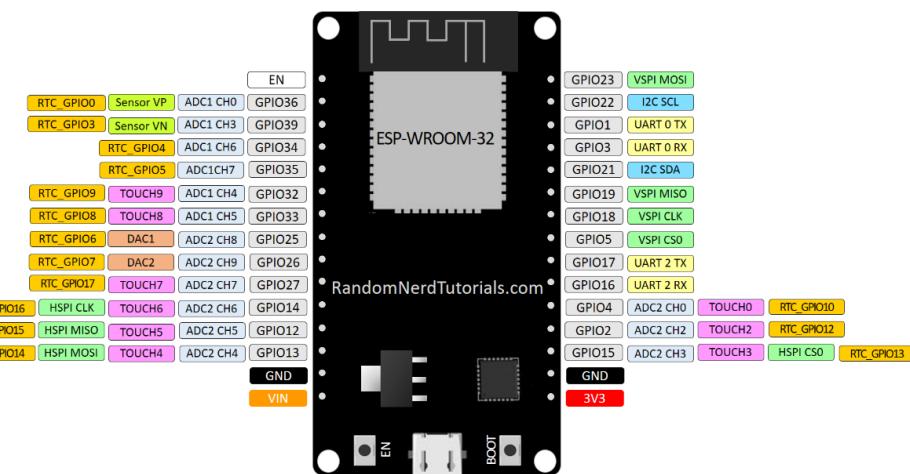
## LAMPIRAN

## LAMPIRAN PINOUT DIAGRAM MIKROKONTROLER

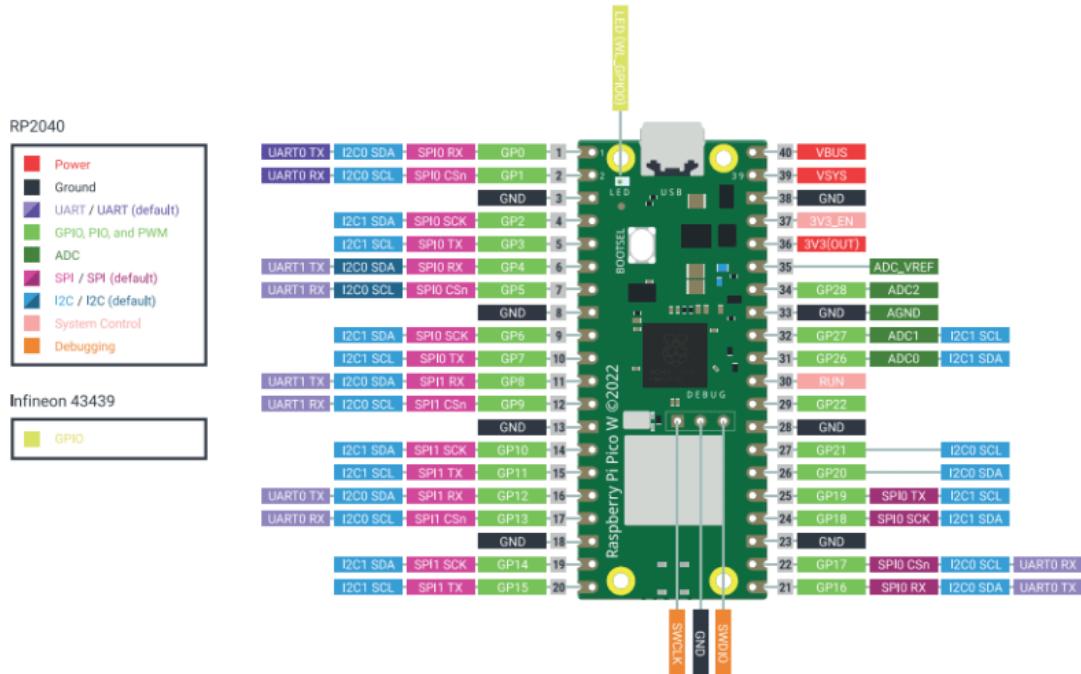
### ESP32

#### ESP32 DEVKIT V1 – DOIT

version with 30 GPIOs



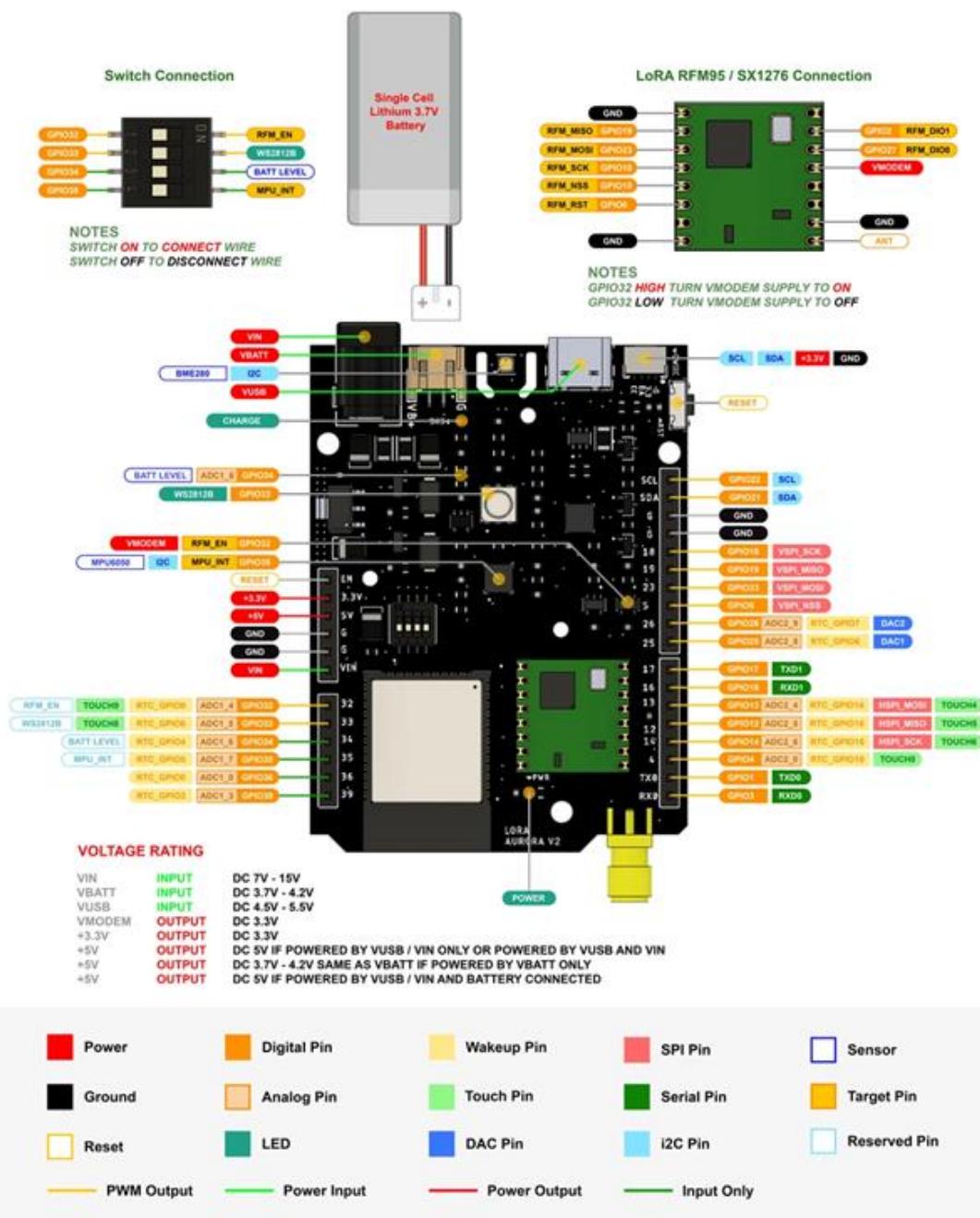
### Raspberry Pi Pico W



## Cosmic LoRa Aurora V2



### COSMIC LORA AURORA V2



## LAMPIRAN MODUL 2

### Lampiran 1. Firebase with ESP32 Source Code

#### Firebase with ESP32 Source Code - C Arduino

```
/*
Here we decide the wifi library:
-> We use "Wifi.h" if our device is ESP32.
-> We use "ESP8266WiFi.h" our device is ESP8266.

*/
#ifndef defined(ESP32) || defined(ARDUINO_RASPBERRY_PI_PICO_W)
#include <WiFi.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#endif

/* FIREBASE START */
// FIREBASE LIBRARY
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h" // Provide the token generation process
info.
#include "addons/RTDBHelper.h" // Provide the RTDB payload printing info
and other helper functions.

// FIREBASE SETUP
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
bool signupOK = false;

/* 1. Define the WiFi credentials */
#define WIFI_SSID "<YOUR_SSID>"
#define WIFI_PASSWORD "<YOUR_WIFI_PASSWORD>"
/* 2. Define the API Key */
#define API_KEY "<YOUR_FIREBASE_API>"
/* 3. Define the RTDB URL */
#define DATABASE_URL "<YOUR_RTDB_URL>"
/* 4. ACTIVATE IT for authenticated account: Define the user Email and
password that already registered or added in your project */
// #define USER_EMAIL "<YOUR EMAIL HERE>"
// #define USER_PASSWORD "<YOUR PASSWORD HERE>

// Function List
void firebaseSetInt(String, int);
void firebaseSetFloat(String, float);
void firebaseSetString(String databaseDirectory, String value);
String firebaseGetString(String databaseDirectory);
```

```
// Root directory
String device_root = "/";
/* FIREBASE END */

int int_value = 0;
float float_value = 0;
char char_value = 'a';
String statusLED;

void setup() {
    Serial.begin(115200);
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, HIGH);

    /* WIFI START */
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to Wi-Fi");

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

    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    /* WIFI END */

    /* FIREBASE START */
    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

    config.api_key = API_KEY; // Assign RTDB API Key

    /*For anonymous account: Sign up */
    if (Firebase.signUp(&config, &auth, "", "")) {
        Serial.println("Firebase success");
        digitalWrite(LED_BUILTIN, LOW);
        signupOK = true;
    } else {
        String firebaseErrorMessage = config.signer.signupError.message.c_str();

        Serial.printf("%s\n", firebaseErrorMessage);
    }
}
```

```
/* ACTIVATE IT For authenticated account: Assign the user sign in
credentials */
// auth.user.email = USER_EMAIL;
// auth.user.password = USER_PASSWORD;

config.database_url = DATABASE_URL; // Assign rtdb url
config.token_status_callback = tokenStatusCallback; // Set callback

Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
/* FIREBASE END */
}

void loop() {
    firebaseSetInt("int_data", int_value);
    firebaseSetFloat("float_data", float_value);
    String string_value = String(char_value);
    firebaseSetString("string_data", string_value);

    int_value += 1;
    float_value += 3;

    statusLED = firebaseGetString("LED");
    if (statusLED == "1") {
        digitalWrite(LED_BUILTIN, HIGH);
    } else {
        digitalWrite(LED_BUILTIN, LOW);
    }
}

String firebaseGetString(String databaseDirectory) {
    if (Firebase.RTDB.getString(&fbdo, databaseDirectory)) {
        if (fbdo.dataType() == "string") {
            String stringValue = fbdo.stringData();
            return stringValue;
        }
    } else {
        Serial.println(fbdo.errorReason());
    }
}

void firebaseSetFloat(String databaseDirectory, float value) {
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setFloat(&fbdo, databaseDirectory, value)) {
        Serial.print("PASSED: ");
        Serial.println(value);
    } else {

```

```
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo.errorReason());
}
}

void firebaseSetInt(String databaseDirectory, int value) {
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setInt(&fbdo, databaseDirectory, value)) {
        Serial.print("PASSED: ");
        Serial.println(value);
    } else {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo.errorReason());
    }
}

void firebaseSetString(String databaseDirectory, String value) {
    // Write a string on the database path
    if (Firebase.RTDB.setString(&fbdo, databaseDirectory, value)) {
        Serial.print("PASSED: ");
        Serial.println(value);
    } else {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo.errorReason());
    }
}
```

## LAMPIRAN MODUL 3

**Lampiran 1.** *Source code* untuk publish MQTT ke Adafruit IO

### ***Firebase with ESP32 Source Code - C Arduino***

```
"""
A simple example that connects to the Adafruit IO MQTT server
and publishes values that represent a sine wave
"""

import network
import time
from math import sin
from umqtt.simple import MQTTClient
import utime

# Fill in your WiFi network name (ssid) and password here:
wifi_ssid = "ELITLAB"
wifi_password = "l@b3l1tiA"

# Connect to WiFi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(wifi_ssid, wifi_password)
while wlan.isconnected() == False:
    print('Waiting for connection...')
    time.sleep(1)
print("Connected to WiFi")

# Fill in your Adafruit IO Authentication and Feed MQTT Topic details
mqtt_host = "io.adafruit.com"
mqtt_username = "singamon" # Your Adafruit IO username
mqtt_password = "aio_kDnZ01tnnKQ781aZ3CbsaYSQpkCC" # Adafruit IO Key
mqtt_publish_topic = "singamon/feeds/led-status" # The MQTT topic for your
Adafruit IO Feed

# Enter a random ID for this MQTT Client
# It needs to be globally unique across all of Adafruit IO.
mqtt_client_id = "uus9asndajsndiqiwd10wid01n1kwd"

# Initialize our MQTTClient and connect to the MQTT server
mqtt_client = MQTTClient(
    client_id= mqtt_client_id,
    server= mqtt_host,
    user= mqtt_username,
    password= mqtt_password)
```

```
mqtt_client.connect()

# Publish a data point to the Adafruit IO MQTT server every 3 seconds
# Note: Adafruit IO has rate limits in place, every 3 seconds is frequent
# enough to see data in realtime without exceeding the rate limit.
counter = 0
try:
    while True:
        # Generate some dummy data that changes every loop
        counter += 3

        # Get the current timestamp
        current_time = utime.localtime()
        timestamp = "{:04}-{:02}-{:02} {:02}:{:02}:{:02}".format(
            current_time[0], current_time[1], current_time[2],
            current_time[3], current_time[4], current_time[5]
        )

        # Publish the data to the topic!
        print(f'{timestamp} - Publish {counter}')
        mqtt_client.publish(mqtt_publish_topic, str(counter))

        # Delay a bit to avoid hitting the rate limit
        time.sleep(3)
except Exception as e:
    print(f'Failed to publish message: {e}')
finally:
    mqtt_client.disconnect()
```

## Lampiran 2. *Source code* untuk subscribe MQTT ke Adafruit IO

### *Firebase with ESP32 Source Code - C Arduino*

```
"""
A simple example that connects to the Adafruit IO MQTT server
and subscribes to a topic
"""

import time
import network
from machine import Pin
from umqtt.simple import MQTTClient
import utime

# Fill in your WiFi network name (ssid) and password here:
wifi_ssid = "ELITLAB"
wifi_password = "l@b3l1tiA"
```

```
# Connect to WiFi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(wifi_ssid, wifi_password)
while wlan.isconnected() == False:
    print('Waiting for connection...')
    time.sleep(1)
print("Connected to WiFi")

# Fill in your Adafruit IO Authentication and Feed MQTT Topic details
mqtt_host = "io.adafruit.com"
mqtt_username = "singamon" # Your Adafruit IO username
mqtt_password = "aio_kDnZ01tnnKQ78laZ3CbsaYSQpkCC" # Adafruit IO Key
mqtt_receive_topic = "singamon/feeds/led-status" # The MQTT topic for your Adafruit IO Feed

# Enter a random ID for this MQTT Client
# It needs to be globally unique across all of Adafruit IO.
mqtt_client_id = "iij0dfq8f012nif2jnfijn"

# Initialize our MQTTClient and connect to the MQTT server
mqtt_client = MQTTClient(
    client_id= mqtt_client_id,
    server= mqtt_host,
    user= mqtt_username,
    password= mqtt_password)

# So that we can respond to messages on an MQTT topic, we need a callback
# function that will handle the messages.
def mqtt_subscription_callback(topic, message):
    # Get the current timestamp
    current_time = utime.localtime()
    timestamp = "{:04}-{:02}-{:02} {:02}:{:02}:{:02}".format(
        current_time[0], current_time[1], current_time[2],
        current_time[3], current_time[4], current_time[5])
    )

    print(f'{timestamp} - Topic {topic} received message {message}') # Debug print out of what was received over MQTT

# Before connecting, tell the MQTT client to use the callback
mqtt_client.set_callback(mqtt_subscription_callback)
mqtt_client.connect()

# Once connected, subscribe to the MQTT topic
```

```
mqtt_client.subscribe(mqtt_receive_topic)
print("Connected and subscribed")

try:
    while True:
        # Infinitely wait for messages on the topic.
        # Note wait_msg() is a blocking call, if you're doing multiple things
        # on the Pico you may want to look at putting this on another thread.
        print(f'Waiting for messages on {mqtt_receive_topic}')
        mqtt_client.wait_msg()
except Exception as e:
    print(f'Failed to wait for MQTT messages: {e}')
finally:
    mqtt_client.disconnect()
```

## LAMPIRAN MODUL 4

### Lampiran 1. *Source code transmitter LoRa*

#### **Source code transmitter LoRa - C Arduino**

```
*****  
Modified from the examples of the Arduino LoRa library  
More resources: https://github.com/cosmic-id/  
*****/  
// Dalam receiver dan transmitter, menggunakan 2 library yaitu SPI.h dan  
LoRa.h by Sandeep Mistry  
#include <SPI.h>  
#include <LoRa.h>  
  
// Tentukan pin yang digunakan oleh modul receiver  
#define LORA_AURORA_V2_NSS 15  
#define LORA_AURORA_V2_RST 0  
#define LORA_AURORA_V2_DIO0 27  
#define LORA_AURORA_V2_EN 32  
  
#define LORA_TX_POWER 20  
#define LORA_SPREADING_FACTOR 12  
  
int counter = 0;  
  
void setup()  
{  
    // Inisialisasi mode pin LoRa  
    pinMode(LORA_AURORA_V2_EN, OUTPUT);  
    // LoRa chip dinyalakan  
    digitalWrite(LORA_AURORA_V2_EN, HIGH);  
  
    // Inisialisasi Serial Monitor  
    Serial.begin(115200);  
    while (!Serial)  
        ;  
    Serial.println("LoRa Sender");  
  
    // setup modul receiver LoRa  
    LoRa.setPins(LORA_AURORA_V2_NSS, LORA_AURORA_V2_RST,  
LORA_AURORA_V2_DIO0);  
  
    // Ganti LoRa.begin(---E-) argument dengan frekuensi yang cocok di tempat  
kamu  
    // 433E6 for Asia  
    // 866E6 for Europe  
    // 915E6 for North America
```

```
while (!LoRa.begin(920E6))
{
    Serial.println(".");
    delay(500);
}

LoRa.setTxPower(LORA_TX_POWER, PA_OUTPUT_PA_BOOST_PIN);
LoRa.setSpreadingFactor(LORA_SPREADING_FACTOR);

// Ganti sync word (0xF3) menyesuaikan transceiver
// sync word memastikan LoRa tidak mendapatkan data dari LoRa
transceivers yang lain
// memiliki range dari 0-0xFF
LoRa.setSyncWord(0xF3);
Serial.println("LoRa Initializing OK!");
}

void loop()
{
    Serial.print("Sending packet: ");
    Serial.println(counter);

    // Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();

    counter++;

    delay(5000);
}
```

## Lampiran 2. Source code Receiver LoRa

<b>Source code Receiver LoRa - C Arduino</b>
<pre>***** Modified from the examples of the Arduino LoRa library More resources: <a href="https://github.com/cosmic-id/">https://github.com/cosmic-id/</a> ***** // Dalam receiver dan transmitter, menggunakan 2 library yaitu SPI.h dan LoRa.h by Sandeep Mistry #include &lt;SPI.h&gt; #include &lt;LoRa.h&gt;  // Tentukan pin yang digunakan oleh modul receiver</pre>

```
#define LORA_AURORA_V2_NSS 15
#define LORA_AURORA_V2_RST 0
#define LORA_AURORA_V2_DIO0 27
#define LORA_AURORA_V2_EN 32

#define LORA_TX_POWER 20
#define LORA_SPREADING_FACTOR 12

void setup()
{
    // Inisialisasi mode pin LoRa
    pinMode(LORA_AURORA_V2_EN, OUTPUT);
    // LoRa chip dinyalakan
    digitalWrite(LORA_AURORA_V2_EN, HIGH);

    // Inisialisasi Serial Monitor
    Serial.begin(115200);
    while (!Serial)
        ;
    Serial.println("LoRa Receiver");

    // setup modul receiver LoRa
    LoRa.setPins(LORA_AURORA_V2_NSS, LORA_AURORA_V2_RST,
LORA_AURORA_V2_DIO0);

    // Ganti LoRa.begin(---E-) argument dengan frekuensi yang cocok di
tempat kamu
    // 433E6 for Asia
    // 866E6 for Europe
    // 915E6 for North America
    while (!LoRa.begin(920E6))
    {
        Serial.println(".");
        delay(500);
    }

    LoRa.setTxPower(LORA_TX_POWER, PA_OUTPUT_PA_BOOST_PIN);
    LoRa.setSpreadingFactor(LORA_SPREADING_FACTOR);

    // Ganti sync word (0xF3) menyesuaikan transceiver
    // sync word memastikan LoRa tidak mendapatkan data dari LoRa
transceivers yang lain
    // memiliki range dari 0-0xFF
    LoRa.setSyncWord(0xF3);
    Serial.println("LoRa Initializing OK!");
}
```

```
void loop()
{
    // Mencoba untuk parse packet yang diterima
    int packetSize = LoRa.parsePacket();
    if (packetSize)
    {
        // received a packet
        Serial.print("Received packet ''");

        // Membaca packet yang diterima
        while (LoRa.available())
        {
            String LoRaData = LoRa.readString();
            Serial.print(LoRaData);
        }

        // Menampilkan RSSI dari packet yang diterima
        Serial.print("'' with RSSI ");
        Serial.println(LoRa.packetRssi());
    }
}
```