

Internet of Things (IoT)

IFA 2025





CoAP

outline

- IoT
- Cloud
- Blynk
- Firebase
- BoT Telegram
- MQTT Protocol



WHAT IS IOT ?

- Internet of Things (IoT) adalah jaringan objek fisik — perangkat, kendaraan, bangunan, dan barang-barang lainnya yang disisipi dengan elektronik, perangkat lunak, sensor, dan koneksi jaringan — yang memungkinkan objek-objek ini mengumpulkan dan bertukar data.



NAMA LAIN (ALIAS) DARI IOT

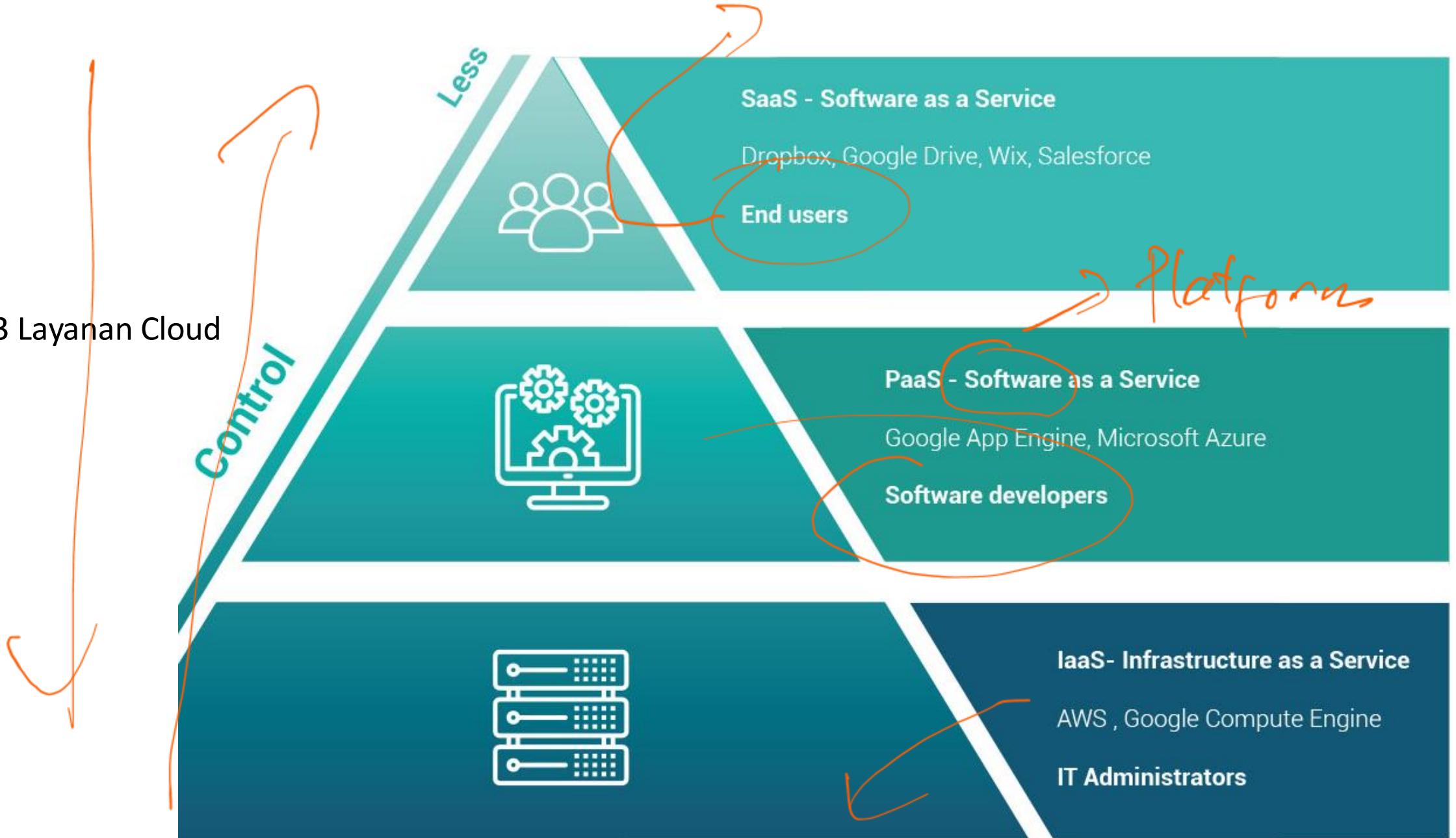
- M2M (Machine to Machine)
 - “Internet of Everything” (Cisco Systems) *manusia, data, proxy*
 - “World Size Web” (Bruce Schneier)
 - “Skynet” (Terminator movie)
-

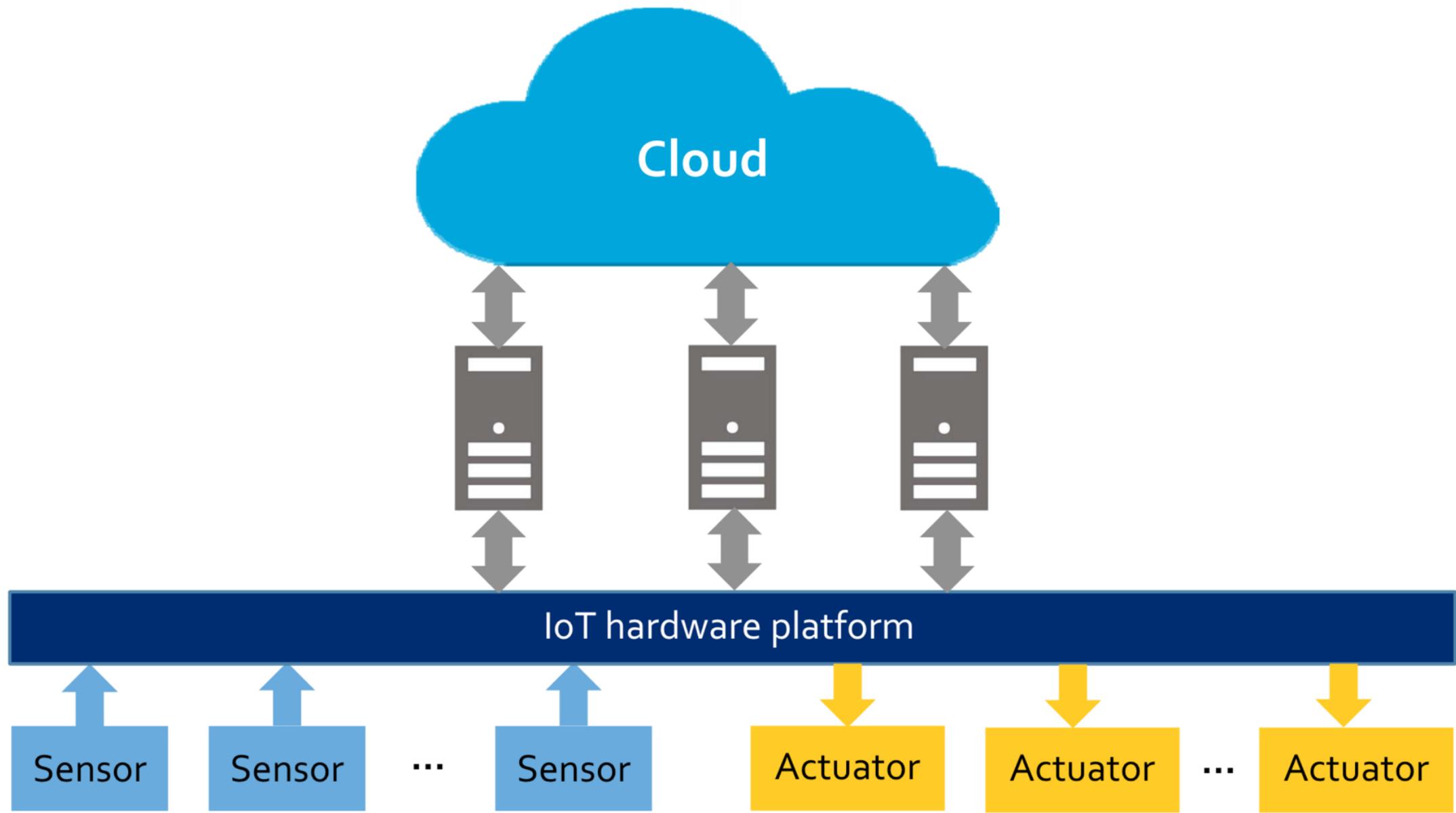
CLOUD COMPUTING

- Cloud Computing adalah istilah umum yang digunakan untuk menggambarkan komputasi berbasis jaringan yang terjadi melalui Internet
 - Menggunakan Internet untuk komunikasi dan transportasi serta penyediaan perangkat keras, perangkat lunak, dan layanan jaringan kepada klien

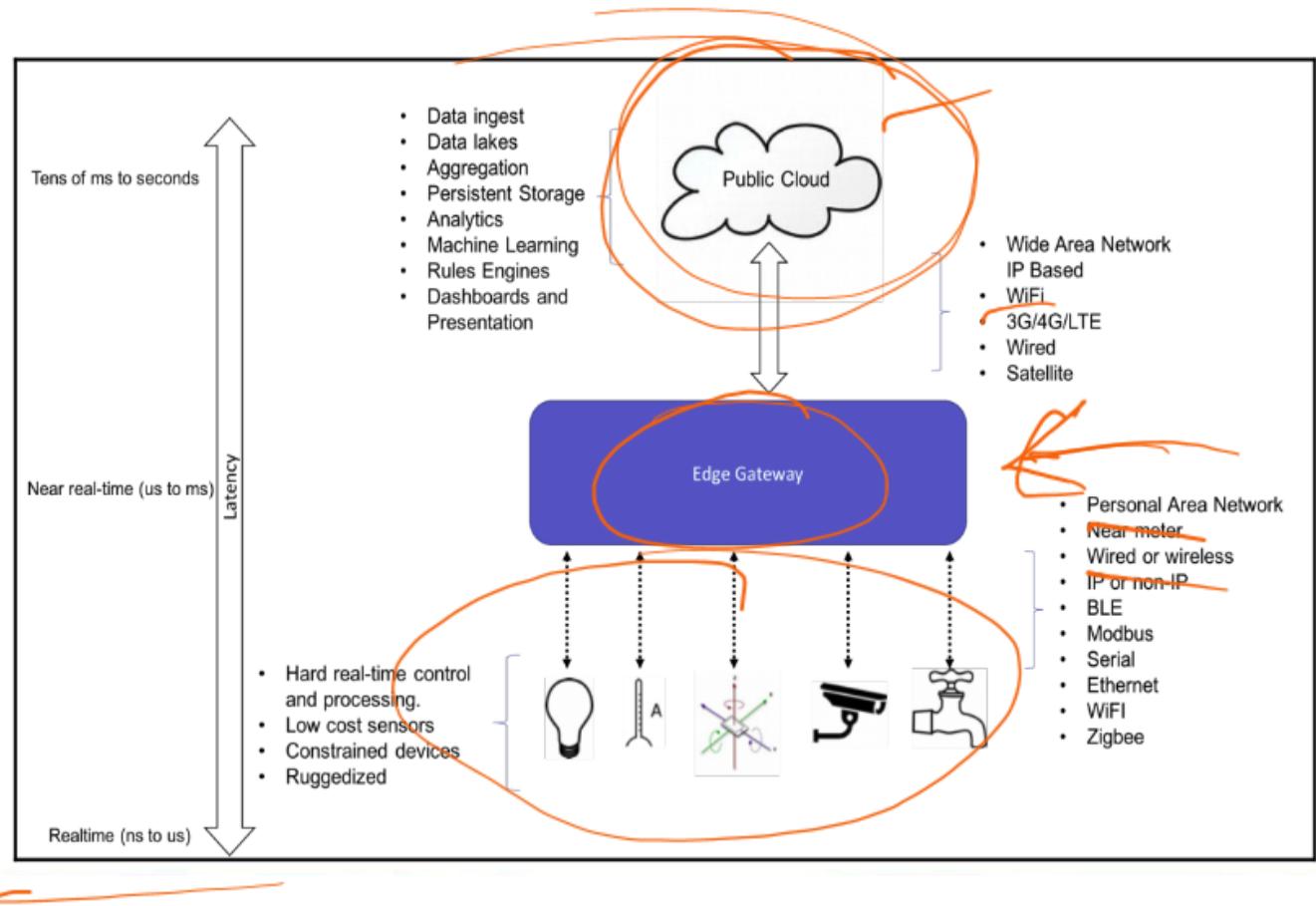


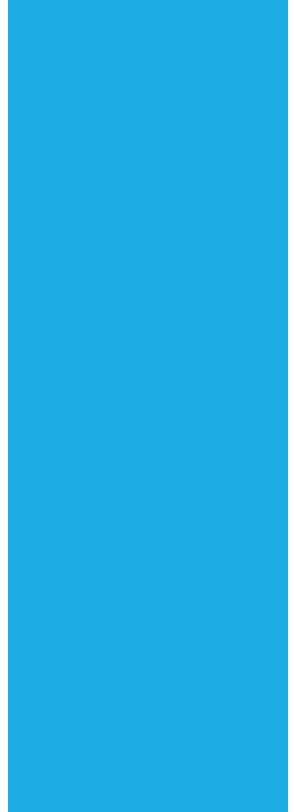
- 3 Layanan Cloud





- Batasan Arsitektur cloud





↳ scal - time .

MQTT – THE
PROTOCOL OF IOT

MQTT

- MQTT adalah singkatan dari Message Queuing Telemetry Transport. Ini adalah protokol komunikasi yang digunakan dalam komunikasi mesin ke mesin (M2M) dan Internet of Things (IoT).



Karakteristik MQTT

1. Ringan dan Efisien

2. Model Pub/Sub

3. QoS (Quality of Service) yang Dapat Disesuaikan

4. Koneksi yang Ringan

5. Keamanan (SSL/TLS), Verifikasi

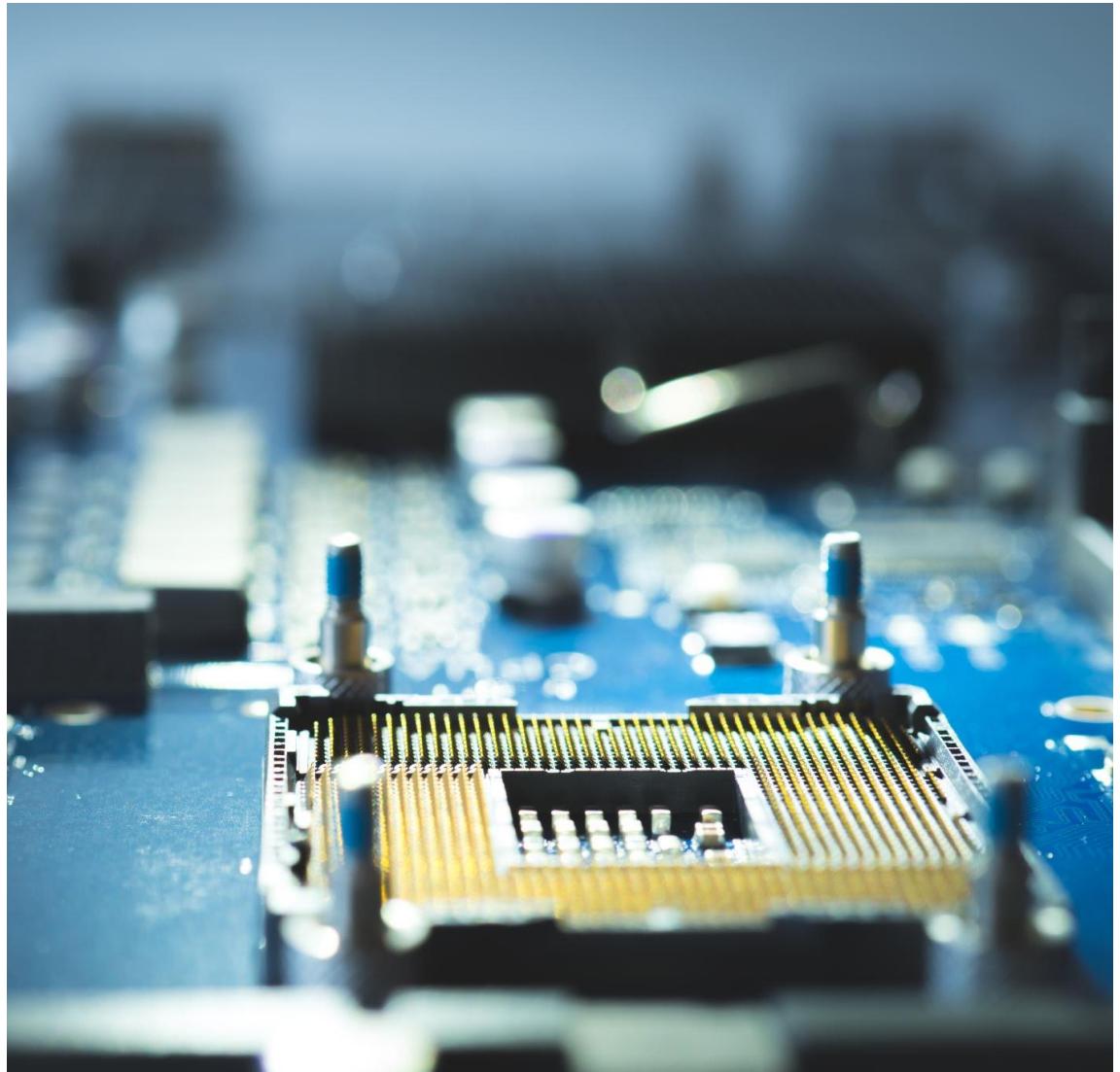
6. Skalabilitas → Broker server, pub.

7. Dukungan berbagai platform

QoS level 0, level 1, level 2.

→ 2 byte.

Publisher (pengirim)
Subscriber (penerima),
Broker



Broker

MQTT Broker memiliki suatu alamat yang dapat diakses oleh Publisher dan Subscriber.



Broker = Server WA.

Publisher = Orang yg
mengirim pesan

Subscriber = anggota WA b.



Sensor / temperatur / kanartidur
sensor / temperatur / relay tamu,

Elemen Data yang dikirimkan oleh MQTT

1. Topik (Topic)

2. Payload *teks* →

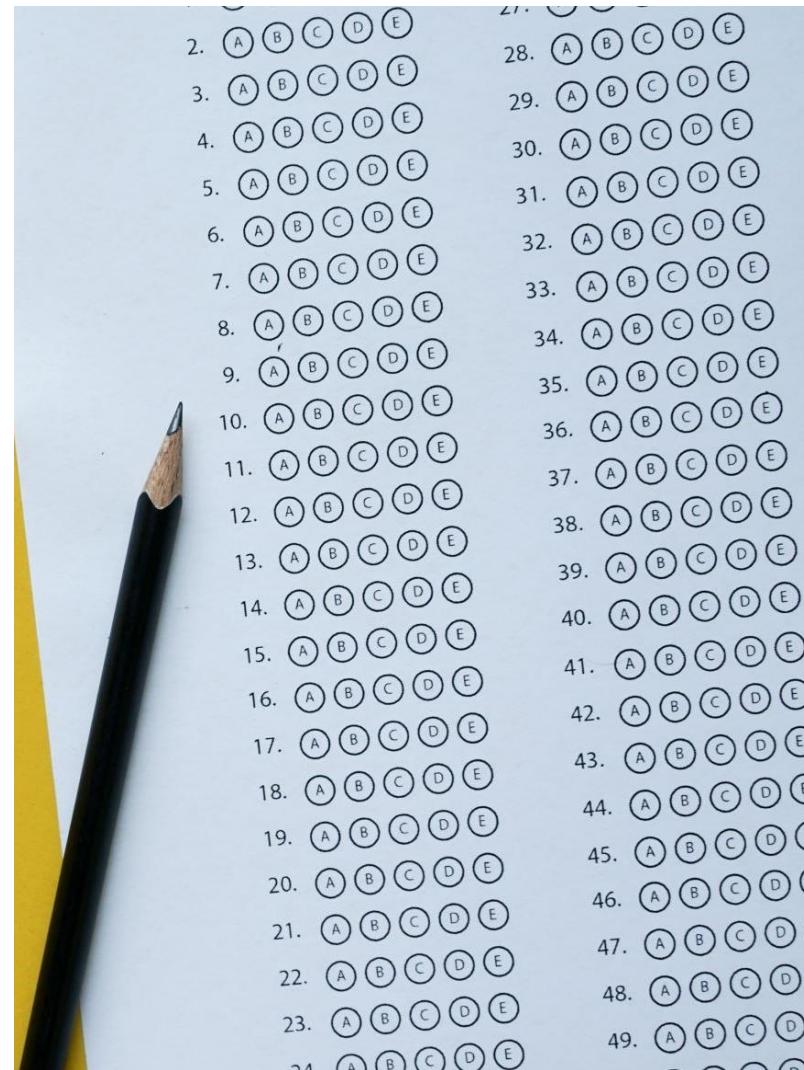
3. Pesan ID (Message ID) → *level QoS*,

4. Kualitas Layanan (Quality of Service/QoS):

5. Informasi Tambahan

temperatur = 30

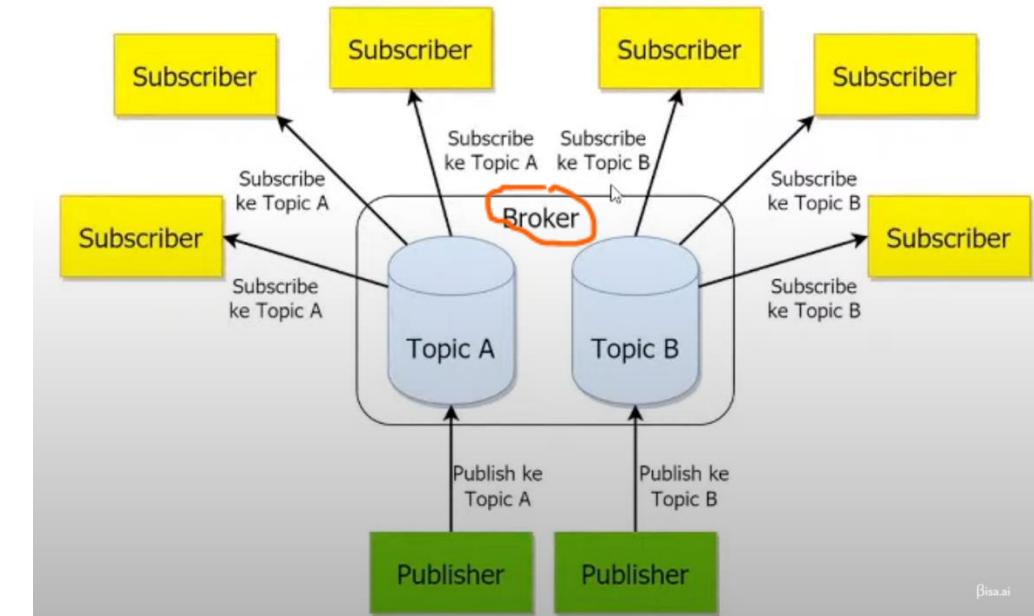
JSON { "temp": 24 }.



Komponen Utama MQTT

- MQTT terdiri dari tiga komponen utama yang bekerja bersama untuk memungkinkan komunikasi antara perangkat:

 - Publisher (Penerbit):** Publisher adalah perangkat atau entitas yang mengirimkan pesan ke broker MQTT.
 - Broker:** Broker adalah perantara pusat dalam arsitektur MQTT. Semua pesan yang diterima oleh broker dari publisher akan disalurkan ke subscriber yang tertarik pada topik yang sama dengan pesan yang diterbitkan.
 - Subscriber (Pelanggan):** Subscriber adalah perangkat atau entitas yang berlangganan (subscribe) ke broker untuk menerima pesan-pesan dari topik tertentu yang menarik minatnya.



Sensor Schule

Publisher



Arduino Uno



ESP32

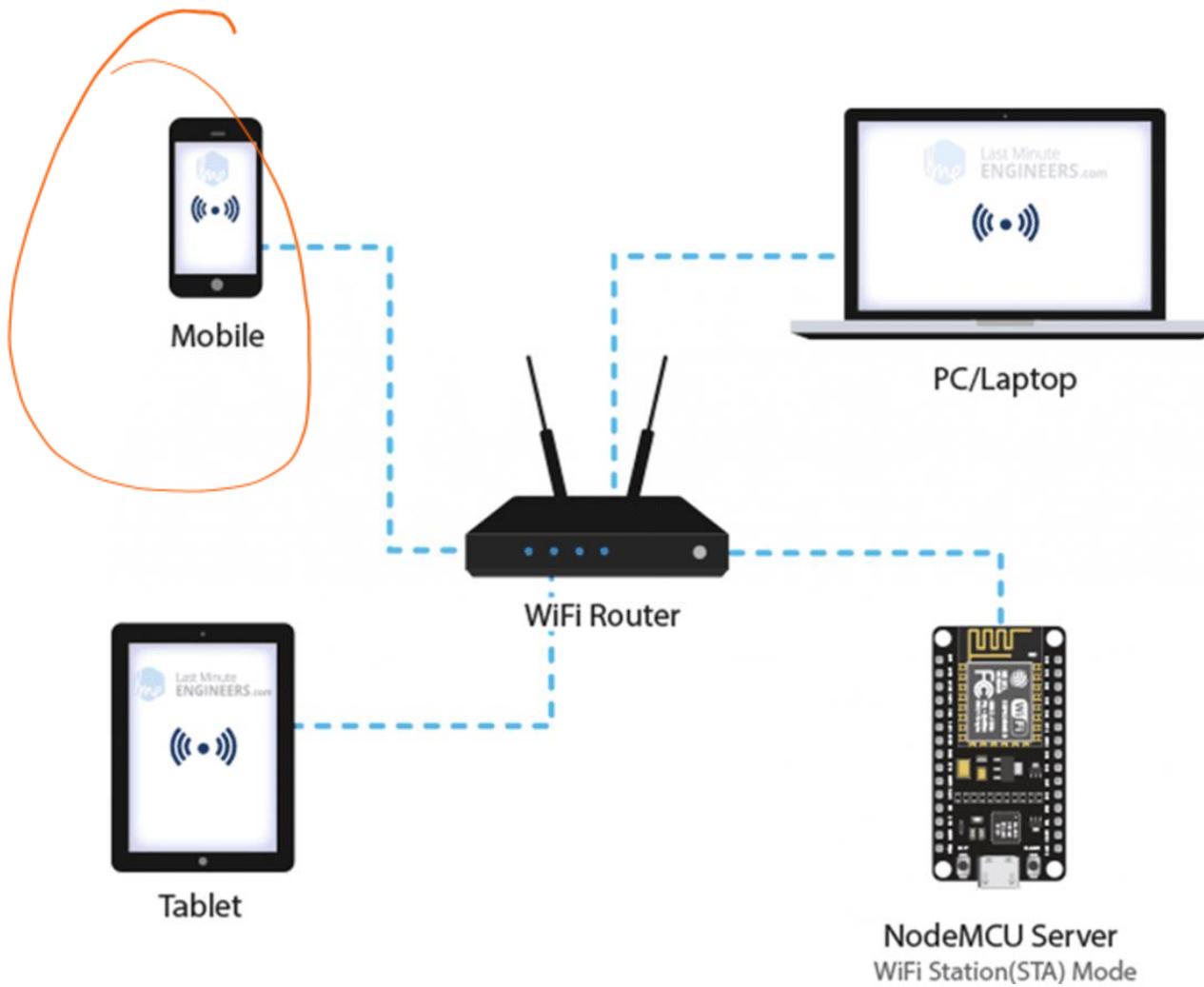


Raspberry Pi



Android

Subscriber



An aerial photograph of a large cargo ship sailing on the ocean. The ship's deck is filled with numerous shipping containers stacked in organized rows. The water around the ship is slightly choppy, creating white foam at the bow. The sky above is clear and blue.

3 Level QoS (Quality of Services) Pada MQTT

→ tercepat
ke handar,

1. QoS Level 0 (At most once):

1. Disebut juga "fire and forget".
2. Pesan dikirimkan sekali tanpa konfirmasi atau jaminan pengiriman.
3. Broker akan meneruskan pesan hanya sekali, tanpa memperhatikan apakah pesan tersebut sampai ke subscriber atau tidak.

2. QoS Level 1 (At least once):

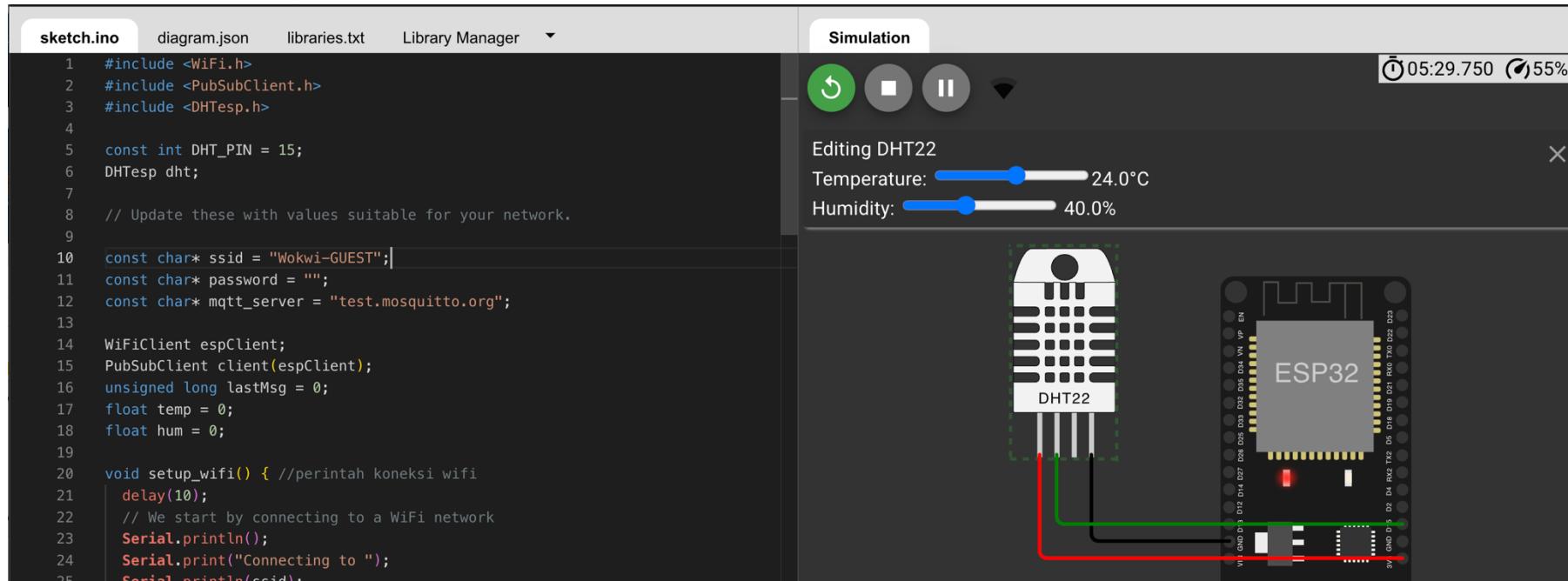
1. Memastikan setidaknya satu pengiriman pesan ke subscriber.
2. Pesan dikirimkan oleh publisher dan broker akan mengonfirmasi penerimaan pesan.
3. Jika pesan tidak terkonfirmasi oleh subscriber, broker akan mengirimkan kembali pesan tersebut.

3. QoS Level 2 (Exactly once):

1. Memberikan jaminan bahwa setiap pesan akan diterima oleh subscriber tepat satu kali.
2. Prosesnya lebih kompleks dan memakan lebih banyak sumber daya karena melibatkan tiga langkah pertukaran pesan antara publisher, broker, dan subscriber.

Project DHT 22

<https://wokwi.com/projects/425058095440888833>



The screenshot shows the Wokwi development environment. On the left, the code editor displays the `sketch.ino` file:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <DHTesp.h>
4
5 const int DHT_PIN = 15;
6 DHTesp dht;
7
8 // Update these with values suitable for your network.
9
10 const char* ssid = "Wokwi-GUEST";
11 const char* password = "";
12 const char* mqtt_server = "test.mosquitto.org";
13
14 WiFiClient espClient;
15 PubSubClient client(espClient);
16 unsigned long lastMsg = 0;
17 float temp = 0;
18 float hum = 0;
19
20 void setup_wifi() { //perintah koneksi wifi
21   delay(10);
22   // We start by connecting to a WiFi network
23   Serial.println();
24   Serial.print("Connecting to ");
25   Serial.println(ssid);
26
27   WiFi.mode(WIFI_STA); //setting wifi chip sebagai station/client
28   WiFi.begin(ssid, password); //koneksi ke jaringan wifi
29
30   while (WiFi.status() != WL_CONNECTED) { //perintah tunggu esp32 sampai terkoneksi
31     delay(500);
32     Serial.print(".");
33   }
34
35   randomSeed(micros());
```

The right side of the interface shows the **Simulation** tab. It features a digital circuit diagram with an ESP32 microcontroller and a DHT22 sensor. The simulation window displays the current sensor readings:

Temperature: 24.0°C
Humidity: 40.0%

Below the simulation window, the serial monitor output shows the following data:

Humidity: 40.0
Temperature: 24.00
Humidity: 40.0
Temperature: 24.00
Humidity: 40.0
Temperature: 24.00
Humidity: 40.0

MQTT Explorer

The screenshot shows the MQTT Explorer application interface. On the left, there's a sidebar titled "Connections" with a yellow "+" button. It lists two connections: "mqtt.eclipse.org" (selected) and "test.mosquitto.org". The main panel is titled "MQTT Connection" and shows the URL "mqtt://test.mosquitto.org:1883/". It includes fields for "Name" (set to "test.mosquitto.org"), "Protocol" (set to "mqtt://"), "Host" (set to "test.mosquitto.org"), "Port" (set to "1883"), "Validate certificate" (disabled), and "Encryption (tls)" (disabled). At the bottom are buttons for "DELETE" (with a trash icon), "ADVANCED" (with a gear icon), "SAVE" (yellow button with a save icon), and "CONNECT" (dark blue button with a power icon).

MQTT Connection mqtt://test.mosquitto.org:1883/

Name: test.mosquitto.org Validate certificate: Encryption (tls):

Protocol: mqtt:// Host: test.mosquitto.org Port: 1883

Username: Password:

DELETE ADVANCED CONNECT

Pilih : advanced untuk subscription

Pilih : connect

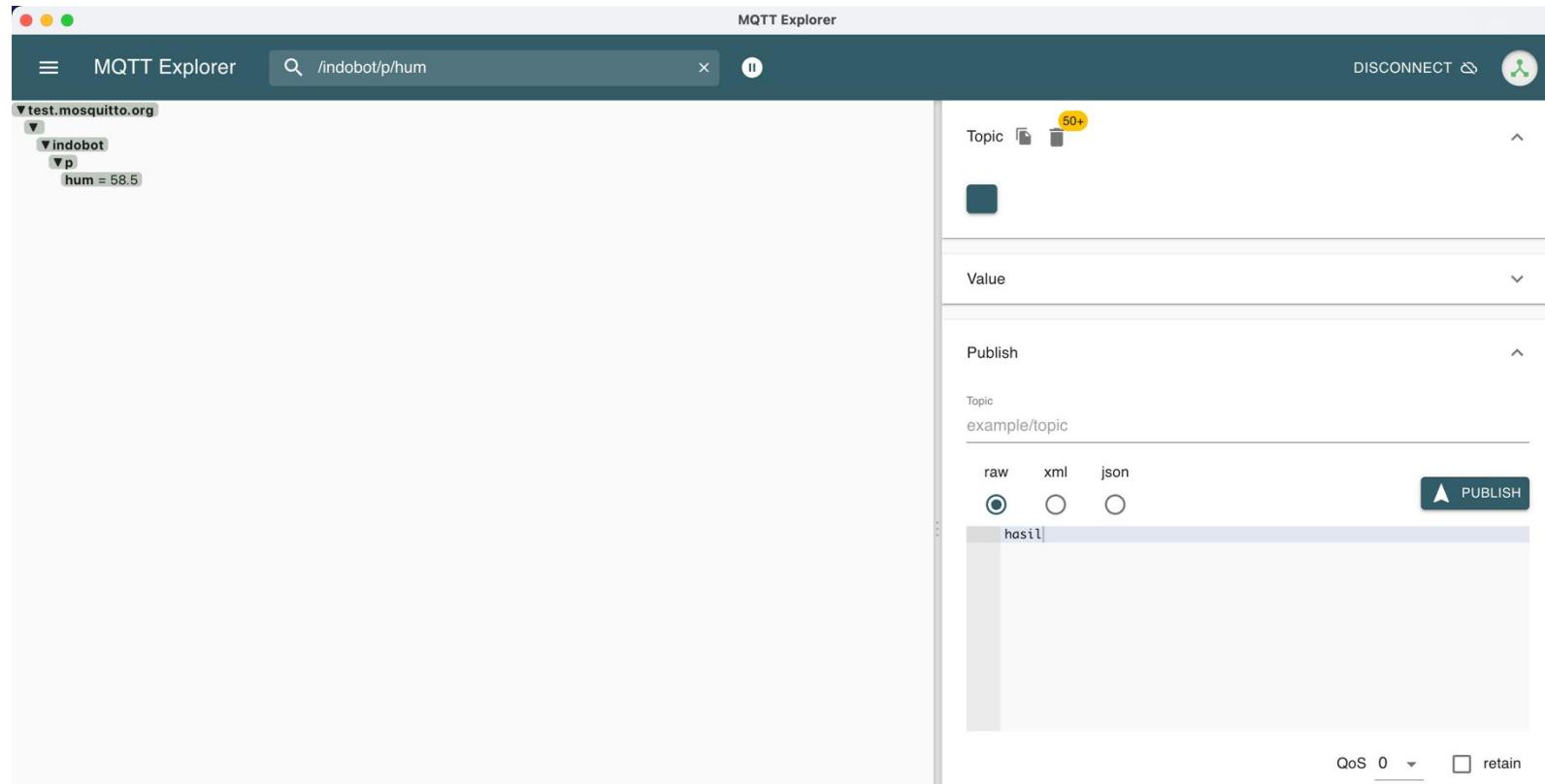
The screenshot shows the MQTT Connection configuration screen in MQTT Explorer. On the left sidebar, there's a 'Connections' section with two entries: 'mqtt.eclipse.org' and 'test.mosquitto.org'. The 'test.mosquitto.org' entry is selected and highlighted with a grey background. The main area is titled 'MQTT Connection' and shows the URL 'mqtt://test.mosquitto.org:1883/'. Below this, the 'Subscription' field contains the topic '/indobot/p/temp'. To the right of the subscription field is a yellow button with a plus sign and the text '+ ADD'. Below the subscription field, there's a list of three topics: '#', '\$SYS/#', and '/indobot/p/temp'. At the bottom of the screen, there's an 'MQTT Client ID' field containing 'mqtt-explorer-9ce2df0f', a 'CERTIFICATES' button with a lock icon, and a 'BACK' button with a left arrow icon.

Tampilan advanced dari menu sebelumnya
Subscription ke
/indobot/p/temp atau /indobot/p/hum, selanjutnya klik button add

DELETE

ADVANCED

MQTT Explorer



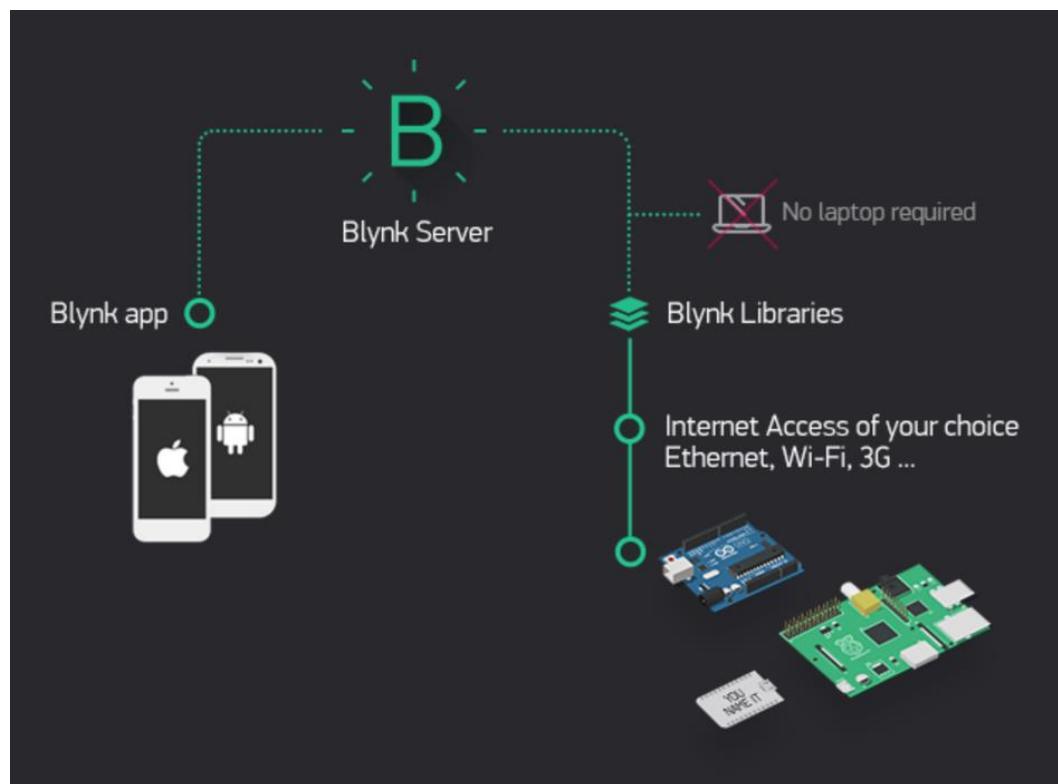
Masukkan /indobot/p/temp untuk melihat suhu

Masukkan /indobot/p/hum untuk melihat kelembaban

Masukkan /indobot/p/mqtt untuk melihat pesan lain dari ESP32

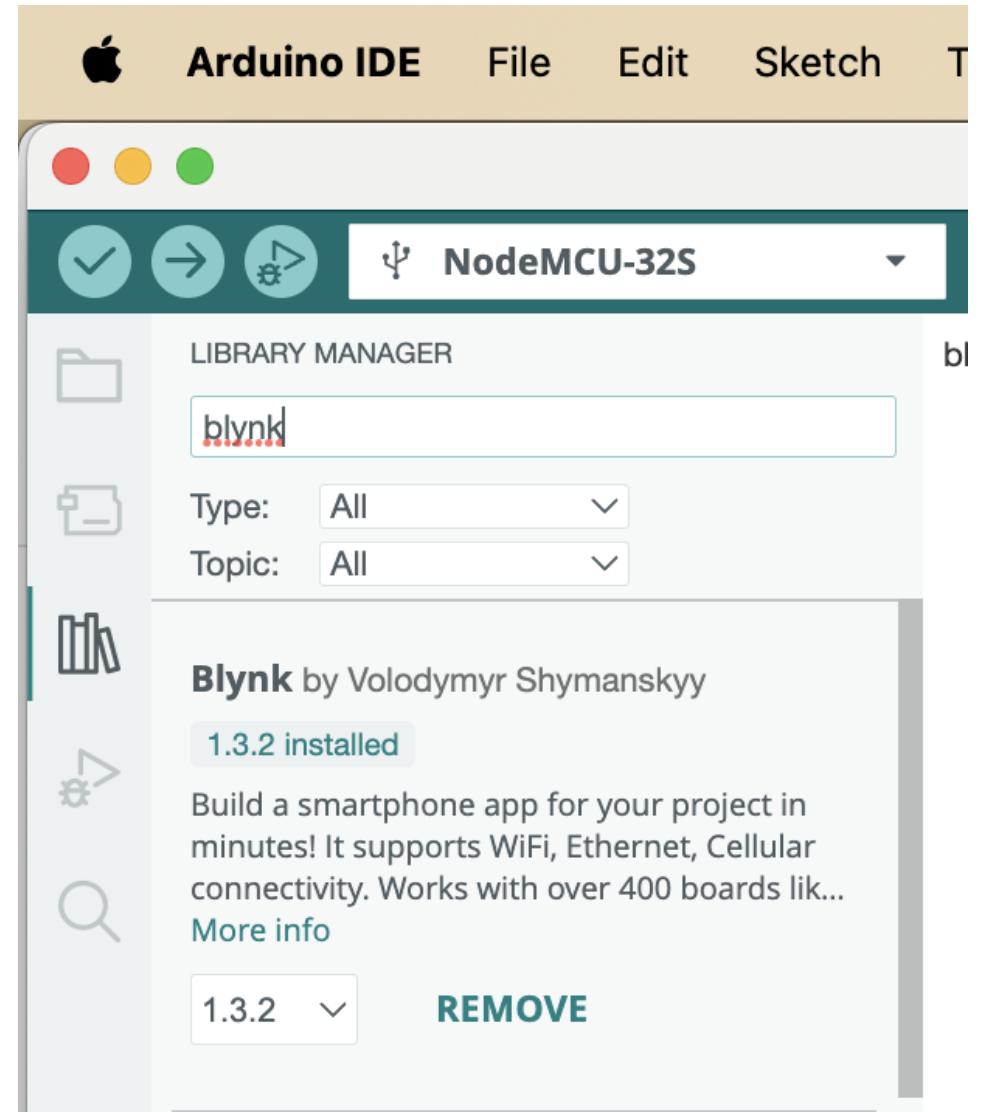
Konsep

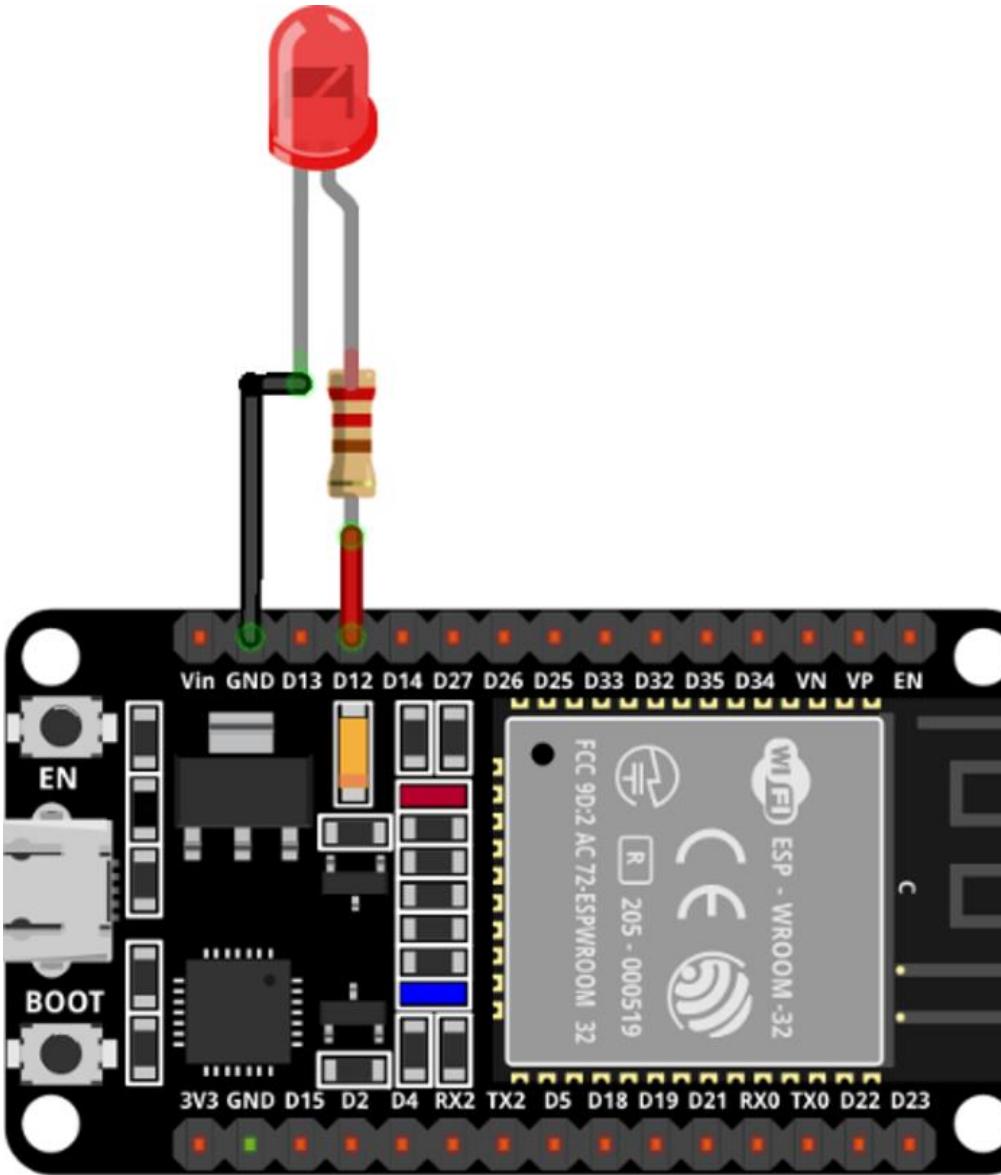
- Blynk adalah platform yang menyediakan alat dan infrastruktur untuk membuat aplikasi mobile yang mengontrol dan memantau perangkat IoT.
- Blynk juga mendukung berbagai perangkat IoT populer, mempermudah pengguna untuk mengembangkan solusi IoT yang fleksibel, intuitif, dan terhubung.



Blynk

- Buka Arduino IDE dan lakukan instalasi library **blink** dari **Volodymr**





Skema Rangkaian LED dan ESP32

- Lakukan koneksi pada rangkaian yang telah ditetapkan

Kode Program

- Setelah selesai terhubung, maka upload kode program berikut ke board ESP32
- Blynk template ID, name, auth token
- Library
- Koneksi WIFI

```
#define BLYNK_TEMPLATE_ID "TMPL6W-NhriSs"
#define BLYNK_TEMPLATE_NAME "LED Blink"
#define BLYNK_AUTH_TOKEN ""

#define BLYNK_PRINT Serial /* include Blynk Serial
#include <WiFi.h> /*ESP32 WiFi Library*/
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

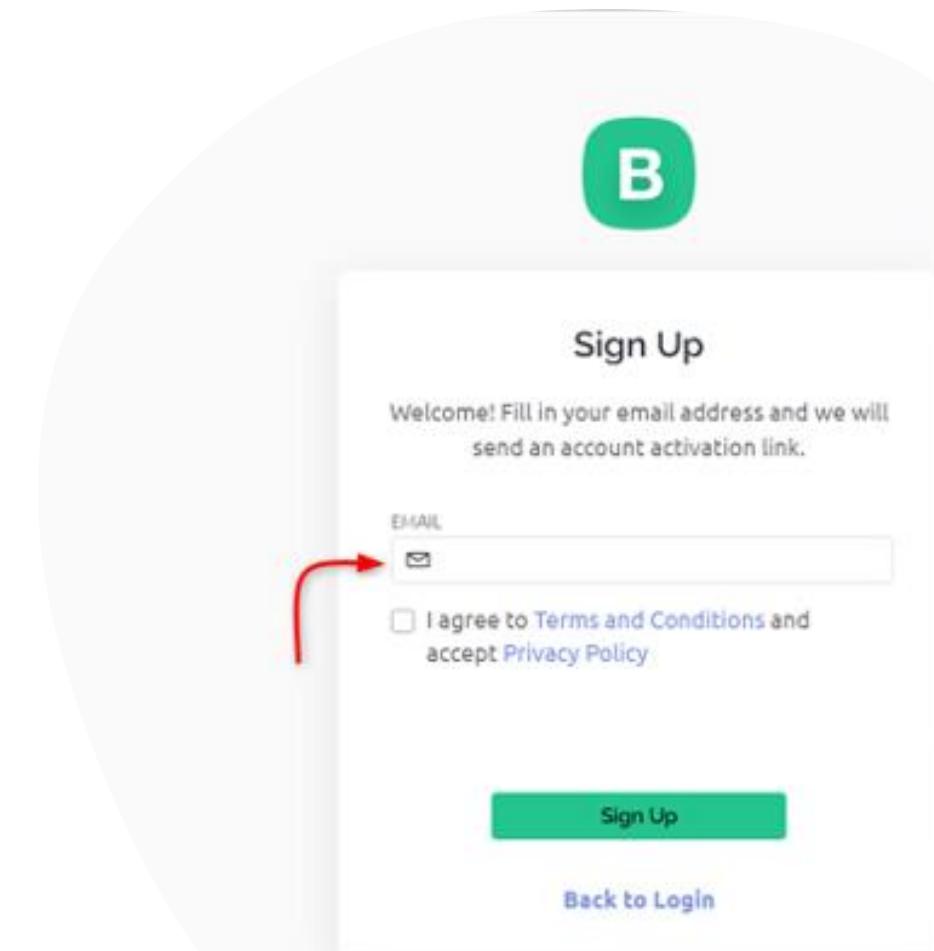
char auth[] = BLYNK_AUTH_TOKEN;
//Enter your WIFI SSID and password
char ssid[] = "          ";
char pass[] = "          ";

void setup(){
    Serial.begin(9600); /*Baud rate for serial communication
    Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
}

void loop(){
    Blynk.run();
}
```

Tahap 1

- Buka link blynk.cloud. Lakukan signup atau login dengan akun yang sudah didaftarkan.



The image shows the Blynk app's main dashboard for an organization named "My organization - 8991RP". On the left, there is a sidebar with various icons and labels: a magnifying glass for search, a building for devices, a grid for locations, a megaphone for users, and a gear for settings. The main area displays organization details: "My organization - 8991RP" with a building icon, "DEVICES" section with "My Devices" (0) and "All" (0), "LOCATIONS" section with "My locations" (0) and "All" (0), and "USERS" section with "My organization members" (1), "All" (1), and "With no devices" (1). To the right, a large text box says "All of your devices will be here." Below it, a message says "You can activate new devices by using your app for iOS or Android" with download links for "Download for iOS" and "Download for Android". A red arrow points from the "New Device" button in the download section to the "New Device" button in the main content area.

B

My organization - 8991RP

DEVICES

My Devices 0

All 0

LOCATIONS

My locations 0

All 0

USERS

My organization members 1

All 1

With no devices 1

All of your devices will be here.

You can activate new devices by using your app for iOS or Android

Download for iOS

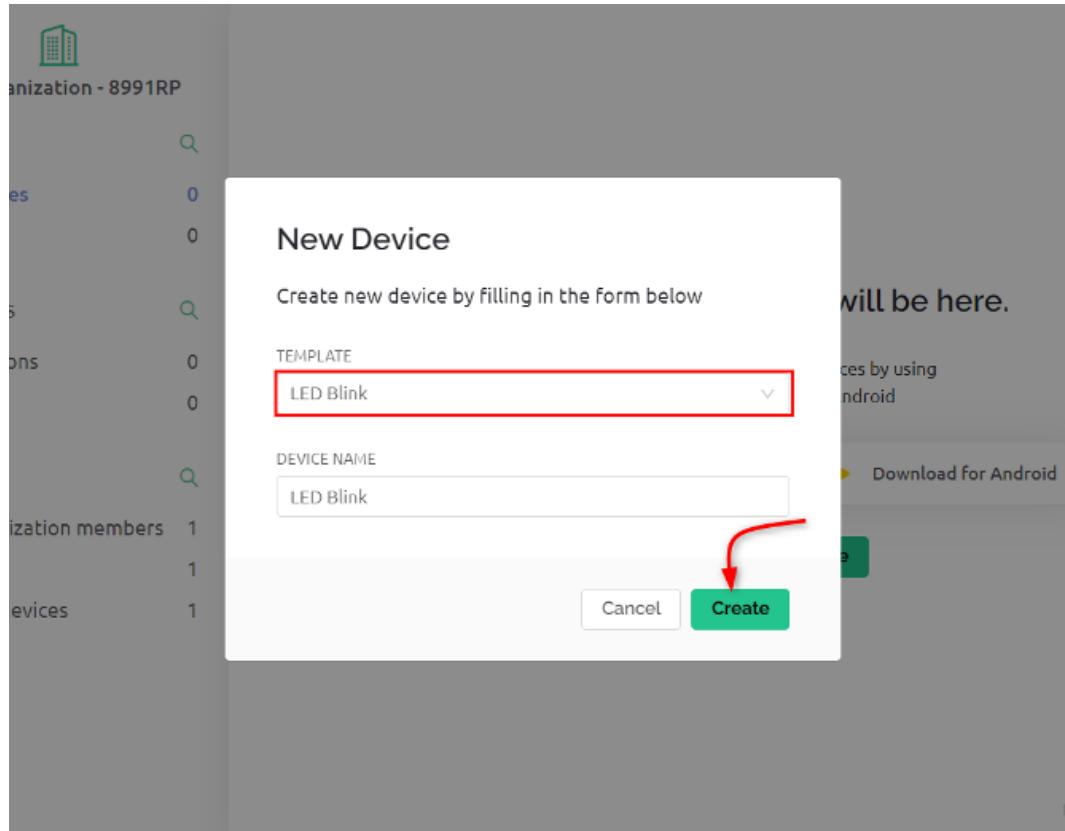
Download for Android

+ New Device

Tahap 2:

setelah login ke Blynk, buka perangkat baru (New Device) .

Tahap 3:



- **Tahap 3:** Namakan perangkat yang nanti nya akan dibuatkan GUI nya untuk control LED. Disini dinamakan LED Blink.

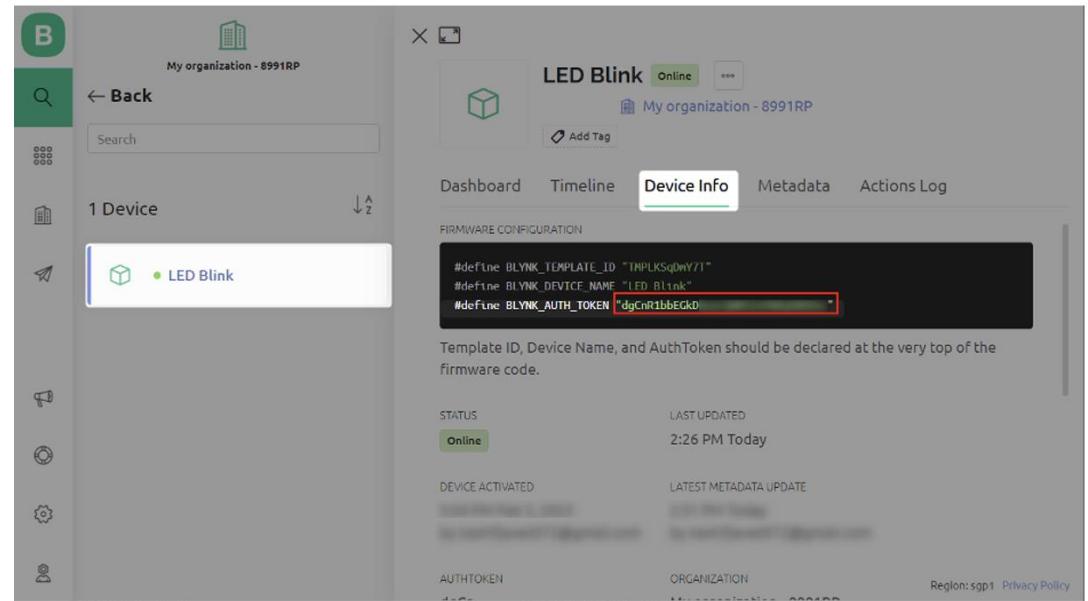
Tahap 4:

- Perangkat baru dengan nama LED Blynk sudah berhasil dibuat.

The screenshot shows the Blynk dashboard interface. On the left, there is a sidebar with various icons and sections: 'DEVICES' (My Devices: 1, All: 1), 'LOCATIONS' (My locations: 0, All: 0), and 'USERS' (My organization members: 1, All: 1, With no devices: 0). The main area is titled 'My Devices' and displays a table with one row. The row contains a device icon, the name 'LED Blink', and a status indicator 'Online'. A red box highlights the 'LED Blink' entry in the table. At the top right of the main area, there is a green button labeled '+ New Device'.

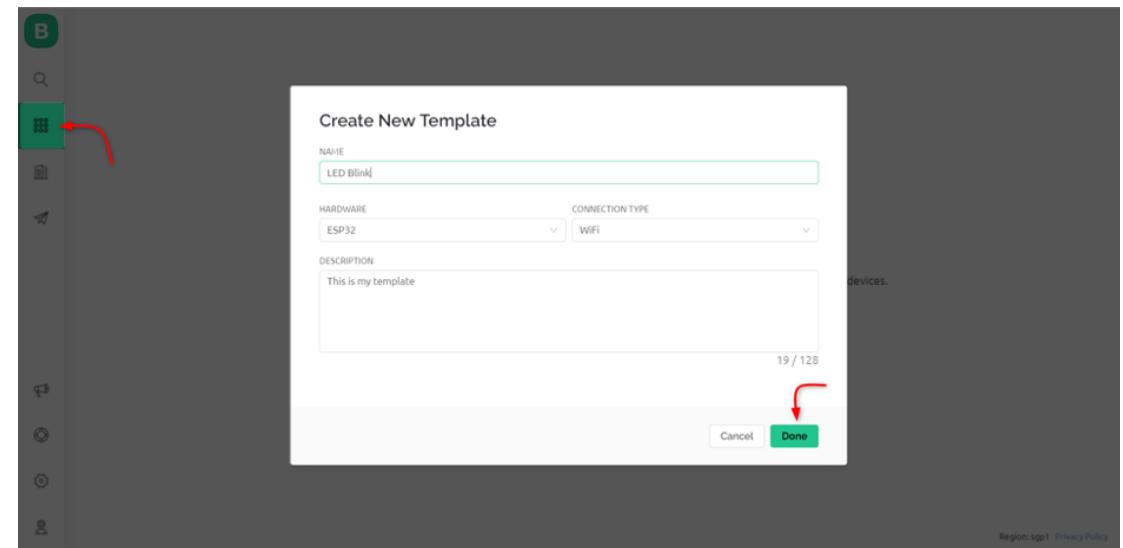
Tahap 5

- Pada bagian device info, maka didapatkan informasi terkait authentication token yang akan digunakan pada kode program di Arduino IDE



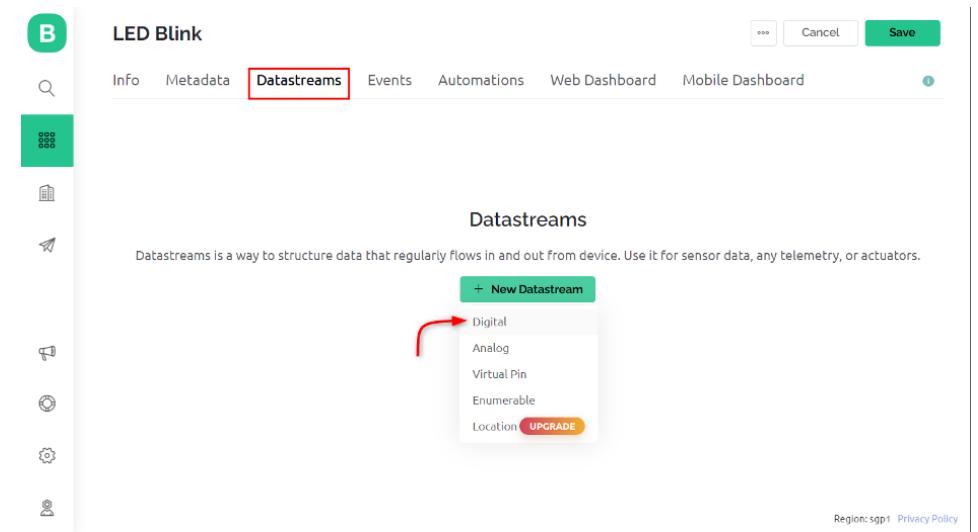
Tahap 6

- Selanjutnya buka template baru. Pada tahap ini dapat dipilih jenis hardware dan tipe koneksi. Jika sudah selesai diisikan maka klik **Done** untuk menyimpan konfigurasi.



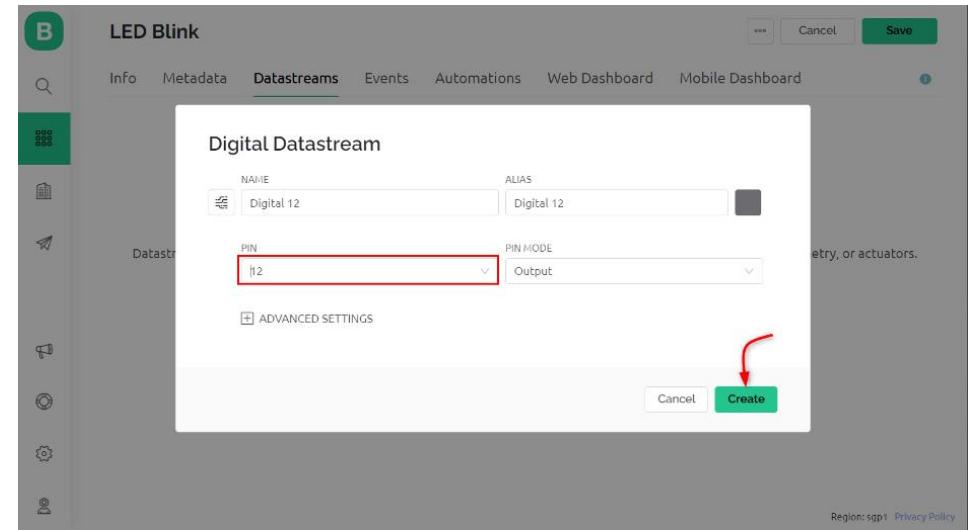
Tahap 7

- Setelah template baru dibuat, kita dapat menambahkan *datastream* di proyek kita. Dengan menggunakan *datastream* ini, maka dimungkinkan untuk dapat mengontrol pin ESP32 mana pun. Karena kita perlu mengontrol LED maka kita akan menggunakan pin digital untuk aliran data:



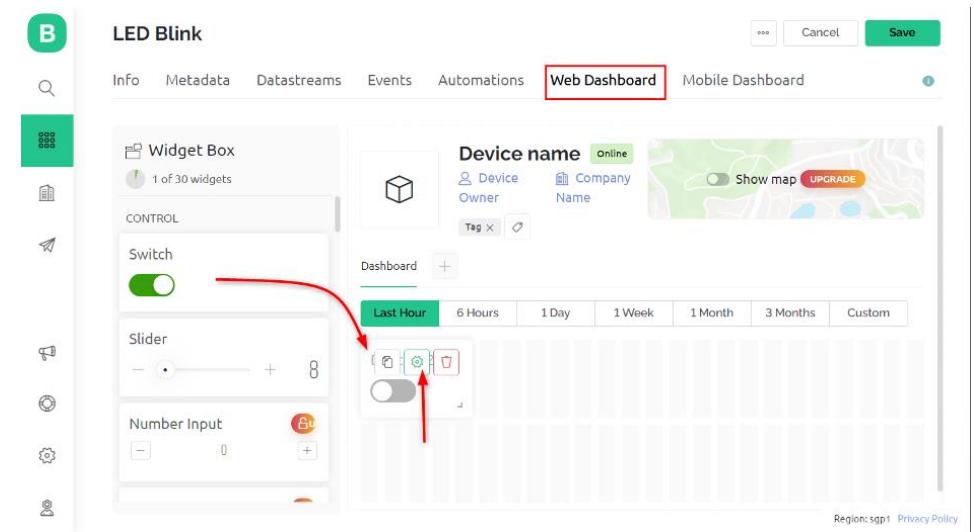
Tahap 8

- Sekarang kita pilih pin dimana LED akan terhubung. Disini kita menggunakan pin D12 pada ESP32 dan kita konfigurasi terkait pin mode nya adalah output.



Tahap 9

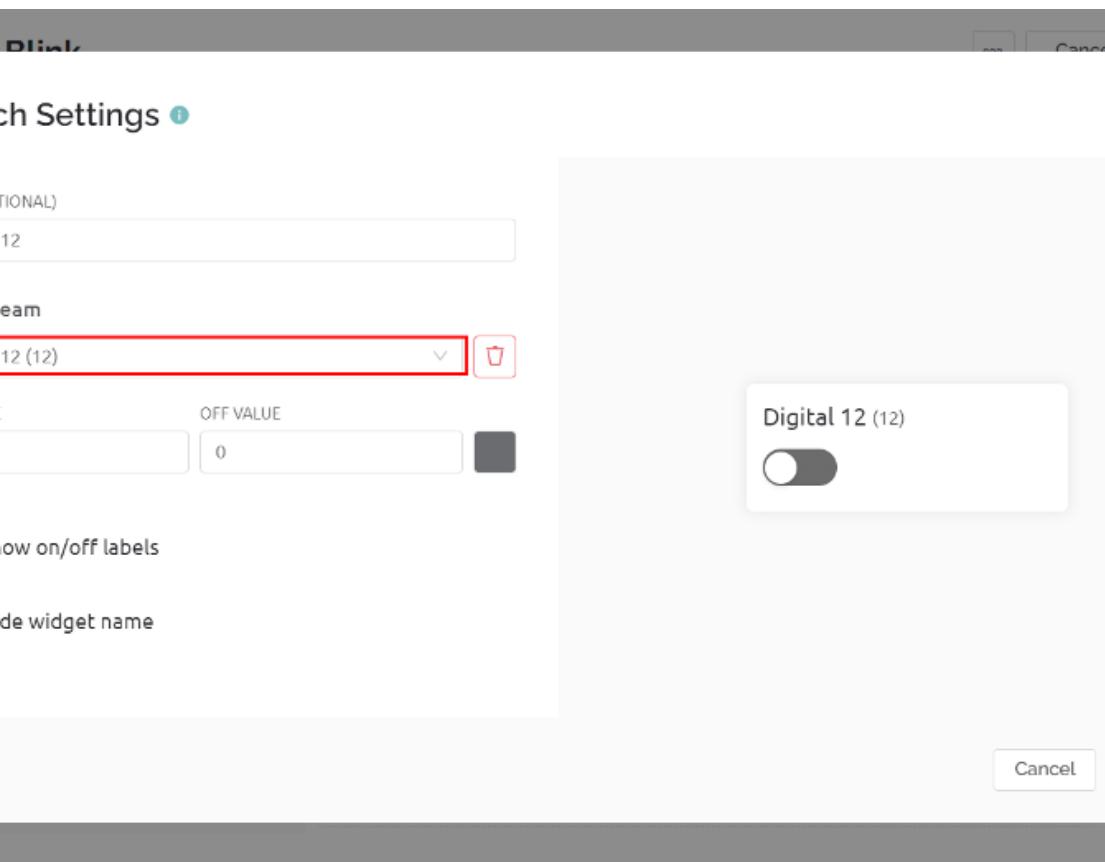
- Untuk mendesain dasbord, kita dapat membuka menu Dasboard Web. Lakukan *drag and drop* switch baru kedalam datastream:





Tahap 10

- Setelah menambahkan tombol baru sekarang pilih opsi pengaturan. Disini lakukan definisi terhadap sumber DataStream sebagai pin Digital 12 dan tetapkan nilai ON ke 1 dan nilai OFF ke 0:





Tahap 11

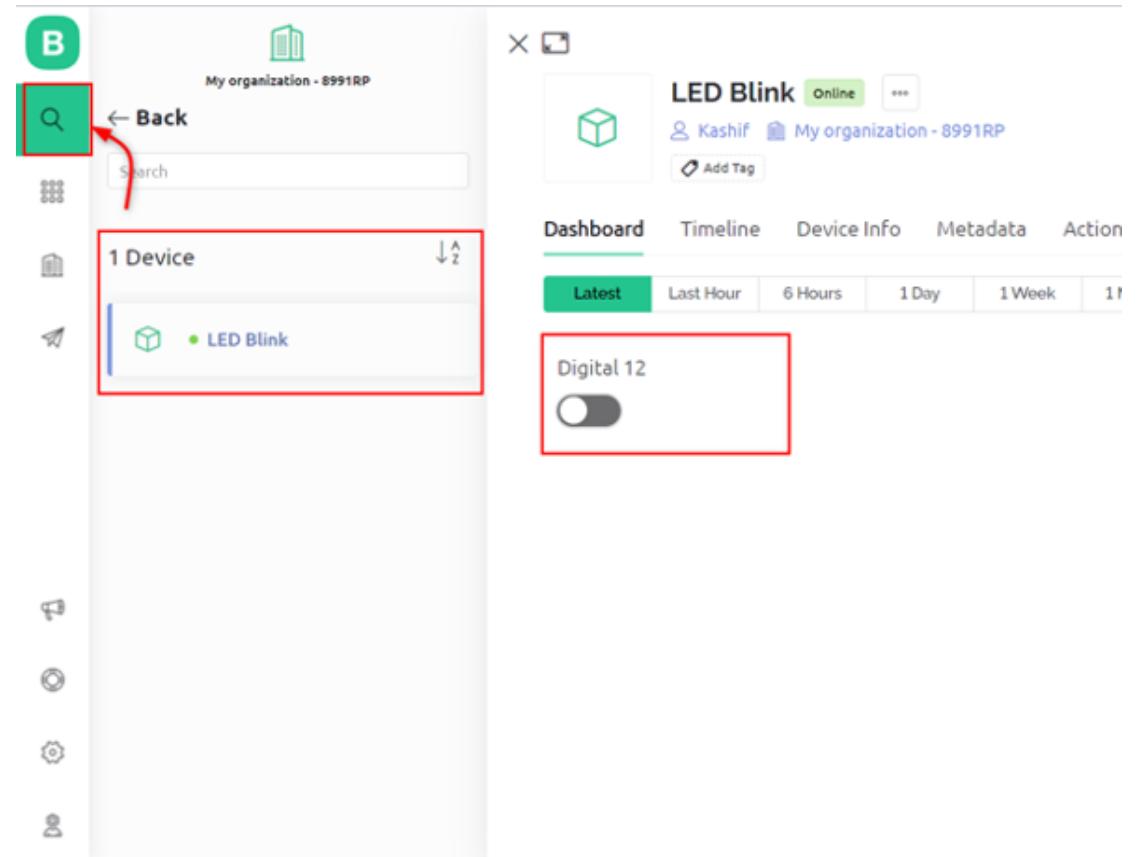
- setelah menambahkan button baru selanjutnya klik **Save** untuk menyimpan konfigurasi. Dengan menggunakan metode ini, kita dapat menambahkan switch apa pun yang sesuai dengan pin ESP32 tertentu:

The screenshot shows the Node-RED Web Dashboard interface. At the top, there are tabs for Metadata, Datastreams, Events, Automations, Web Dashboard (which is currently selected), and Mobile Dashboard. Below the tabs, there's a sidebar titled "get Box" containing "30 widgets". On the main dashboard, there's a section for "Device name" with fields for Device Owner and Company Name, both set to "Name". A "Show map" button with an "UPGRADE" link is also present. The main workspace displays a digital input labeled "Digital 12 (12)" with a toggle switch. The bottom of the screen shows a timeline with options for Last Hour, 6 Hours, 1 Day, 1 Week, 1 Month, 3 Months, and Custom.



Tahap 12

- Sekarang kita dapat control LED menggunakan Blynk, buka dashboard. Disini kita dapat melihat **switch** untuk mengontrol LED yang terhubung pada pin D12.



sketch.ino diagram.json libraries.txt Library Manager ▾

```
1 #define BLYNK_TEMPLATE_ID "TMPL6nT1uVlx5"
2 #define BLYNK_TEMPLATE_NAME "LCD ESP"
3 #define BLYNK_AUTH_TOKEN "0xFjhR1BkWEf8wHFrtBKC4s3dx2EM7CI"
4 #define BLYNK_PRINT Serial
5
6 #include <WiFi.h>
7 #include <WiFiClient.h>
8 #include <BlynkSimpleEsp32.h>
9
10 char auth[] = BLYNK_AUTH_TOKEN;
11 char ssid[] = "Wokwi-GUEST"; //nama hotspot yang digunakan
12 char pass[] = ""; //password hotspot yang digunakan
13
14 BLYNK_WRITE(V0)
15 {
16     int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable
17     if (pinValue == 1)
18     {
19         digitalWrite(4, HIGH);
20     }
21     else {
22         digitalWrite(4, LOW);
23     }
24 }
25 void setup()
26 {
27     Serial.begin(115200);
28     pinMode(4, OUTPUT);
29 }
```

Simulation

02:24.956 100%

The simulation shows an ESP32 development board. A digital output pin (D4) is connected through a green wire to the anode of a red LED. The cathode of the LED is connected through a brown and orange resistor to ground. The board's pins are labeled with their respective names.

[78524] Connecting to blynk.cloud:80
[88529] Connecting to blynk.cloud:8080

Contoh

<https://wokwi.com/projects/425388840359754753>

Success (Serial Monitor)

```
[1383] Connecting to Wokwi-GUEST  
[7717] Connected to WiFi  
[7717] IP: 10.10.0.2  
[7718]
```

/ _) / / _ _____ / / _
/ _ / / / / / _ \ / ' _ /
/ _____ / _ \ / , / _ / / / _ \ / _ \\\n/ _ / v1.3.2 on ESP32

#StandWithUkraine <https://bit.ly/swua>

```
[7728] Connecting to blynk.cloud:80
[8659] Redirecting to sgp1.blynk.cloud:80
[8661] Connecting to sgp1.blynk.cloud:80
[9517] Ready (ping: 196ms).
```

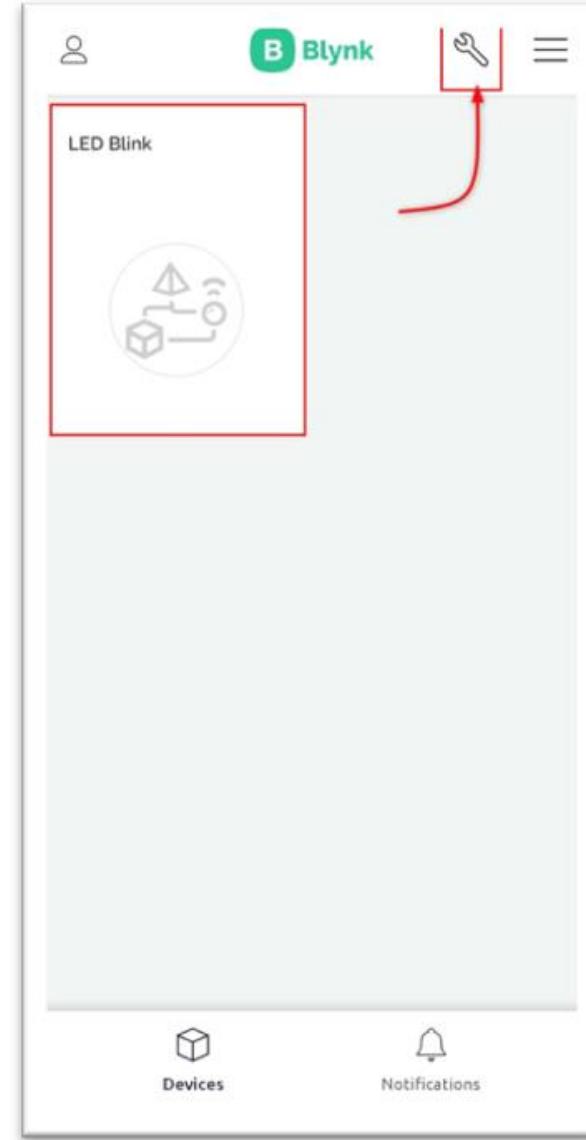
Blynk Mobile Apps

- Blynk menyediakan platform intuitif untuk membuat aplikasi mobile yang mengontrol perangkat IoT.
- Melalui antarmuka pengguna yang disesuaikan, pengguna dapat membuat tombol, slider, dan tampilan data untuk berinteraksi dengan perangkat IoT mereka.

The screenshot shows a web browser displaying the Blynk Documentation website at docs.blynk.io/en/downloads/blynk-apps-for-ios-and-android. The page title is "Blynk Mobile Apps". On the left, there is a sidebar with navigation links such as Introduction, GETTING STARTED, Supported Hardware, Quickstart, Device Activation Methods, Template Quick Setup, Send Data From Hardware To Blynk, Control Devices (GPIOs and beyond), Events, Notifications (Alerts), Sign Up / Sign In, GENERAL CONCEPTS, Developer Mode, Device, Device Template, and Users. On the right, there is a large image of a smartphone screen showing the Blynk app interface. The app displays various IoT devices and their status, including a desk lamp (24°C), garage door, kids room (22°C), DHT 12, and backyard lights. Below the phone image, there is a call-to-action button with the text "Control your hardware from anywhere in the world! With Blynk App" and download links for the App Store and Google Play. At the bottom of the page, there is a footer with the text "Powered By GitBook".

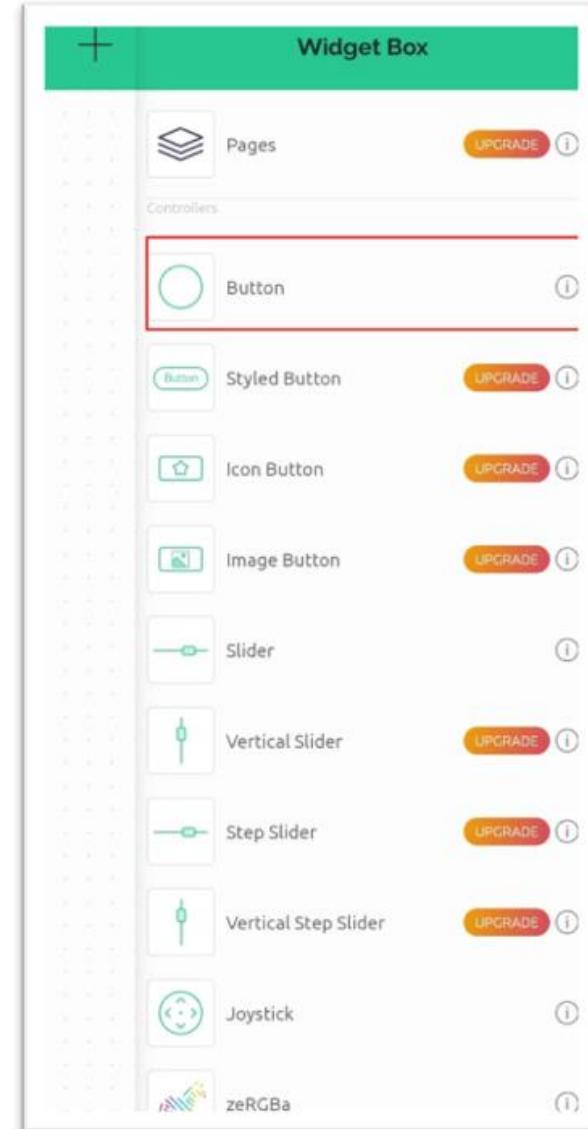
Mobile Development

- Pengembangan via web dashboard dan via mobile application adalah sama, yang perlu diperhatikan bahwasanya Blynk Web dan Mobile Application dibuka dengan menggunakan akun yang sama.
- Jika Anda masuk dengan akun yang sama, Anda akan melihat proyek LED Blink di dalam aplikasi Blynk IoT. Buka mode pengembang menggunakan ikon pengaturan di sudut kanan atas:



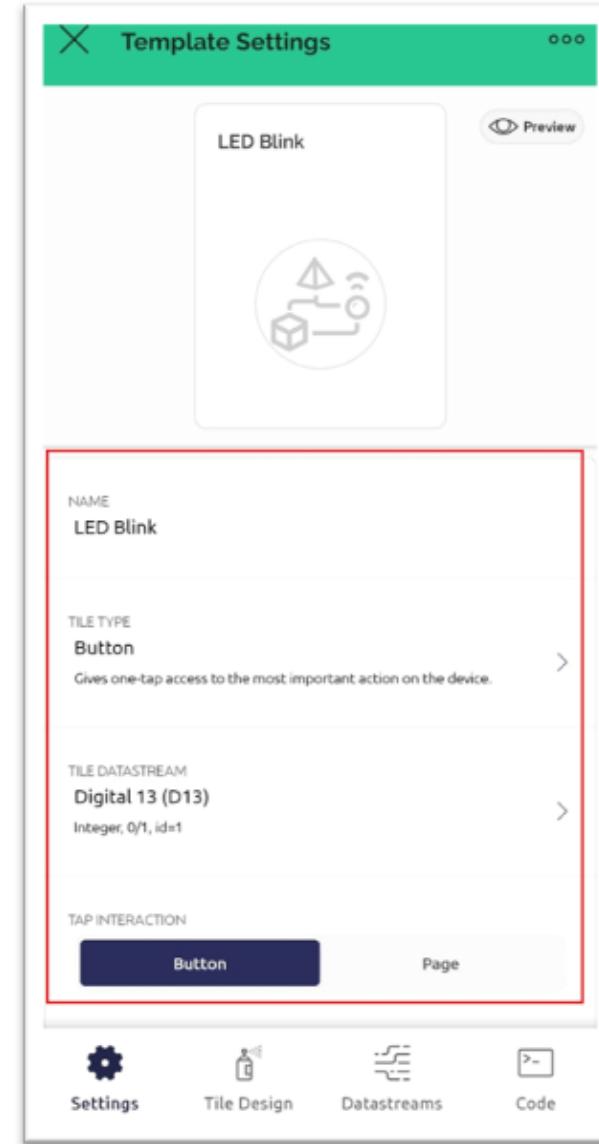
Tahap 1

- Di sini kita dapat membuat tombol baru untuk setiap pin di ESP32 atau menambahkan yang baru:



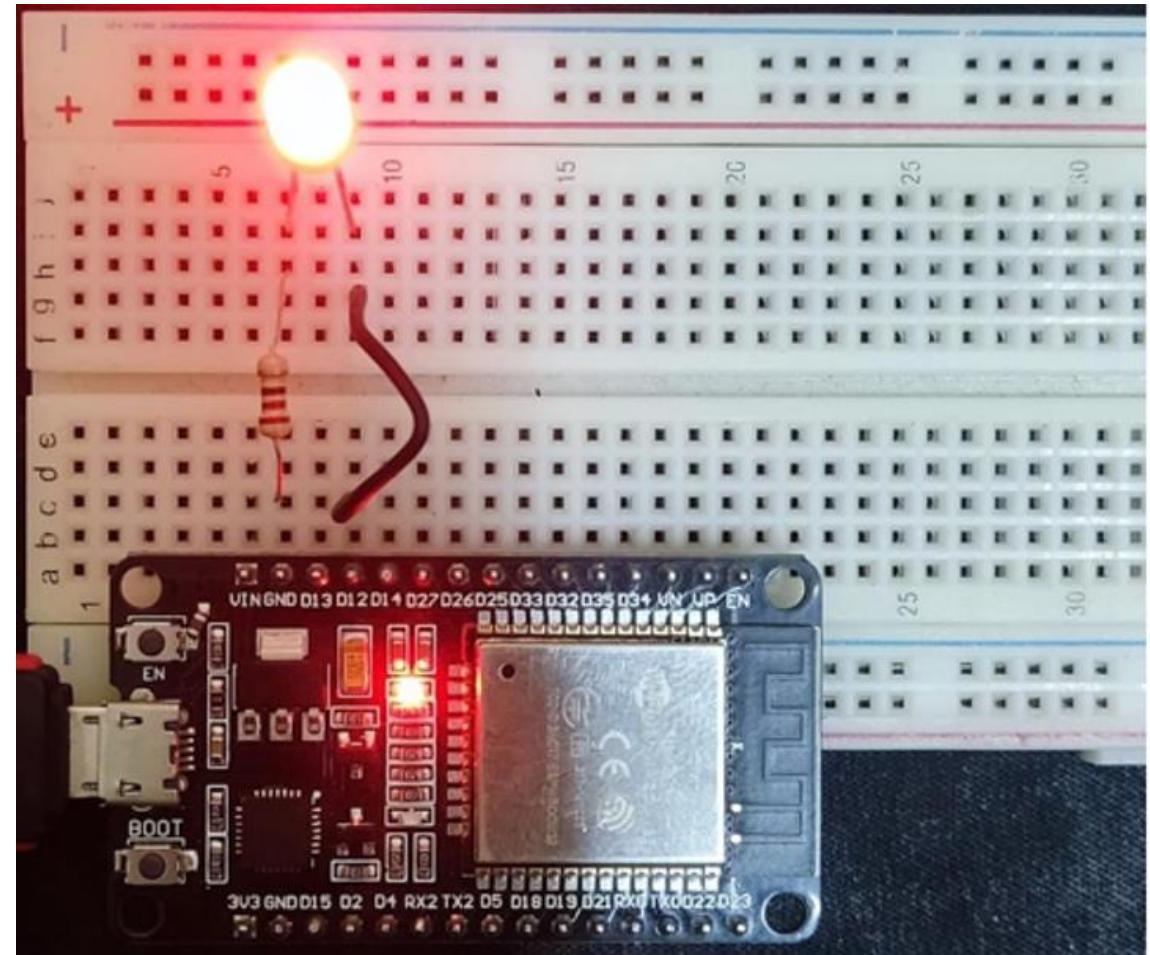
Tahap 2

- Kita juga dapat menyesuaikan pengaturan di dalam template seperti nomor pin atau mengganti mode kerja atau mengatur DataStream baru untuk pin:



Tahap 3

Setelah semua pengaturan selesai, kita dapat melihat LED menyala terhubung ke pin D4.



BoT Telegram



Karakteristik BoT Telegram

- 1. Komunikasi dua arah**
- 2. Perintah dan Tanggapan**
- 3. Notifikasi dan Pemberitahuan**



Library Telegram BoT

- Persiapan Awal
- 1. Library Universal Telegram Bot
- 2. Library Arduino JSON

LIBRARY MANAGER

UniversalTelegramBot

Type: All

Topic: All

UniversalTelegramBot by Brian Lough
1.3.0 installed

Arduino Telegram Bot library for multiple different architectures. A Universal Telegram library for arduino devices.

[More info](#)

1.3.0 [REMOVE](#)

LIBRARY MANAGER

arduinojson

Type: All

Topic: All

ArduinoJson by Benoit Blanchon <blog.benoitblanchon.fr>
6.21.3 installed

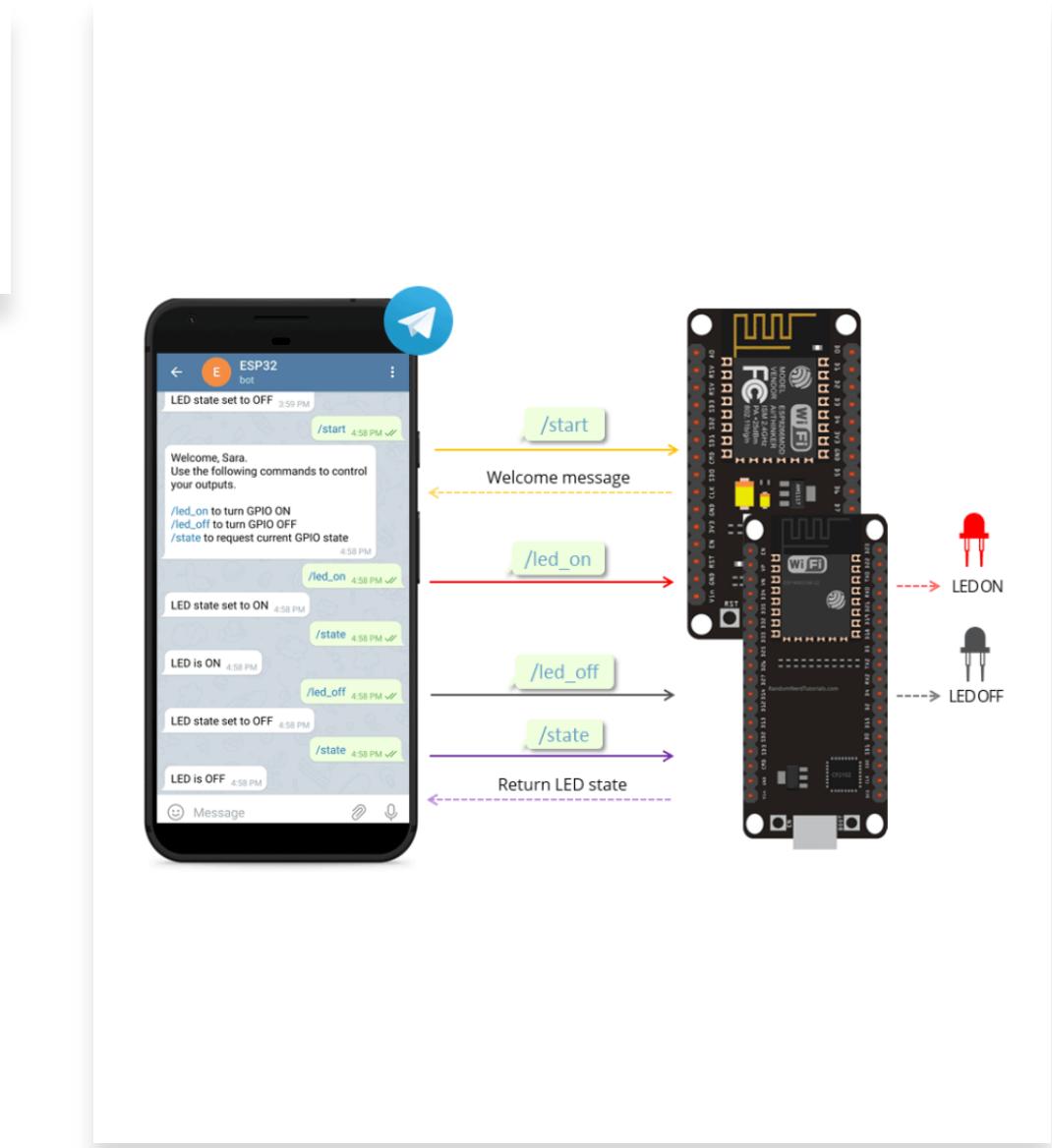
A simple and efficient JSON library for embedded C++. ArduinoJson supports ✓ serialization, ✓ deserialization, ✓...

[More info](#)

6.21.3 [REMOVE](#)

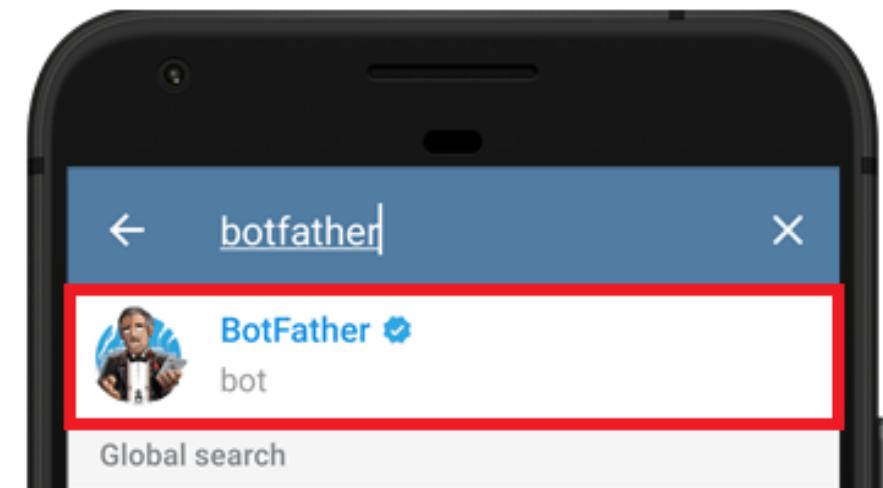
Skema

- Projek ini untuk memberikan command atau instruksi dari bot telegram kepada ESP32 untuk menghidupkan atau menyalakan lampu dan memberikan kondisi/state dari lampu



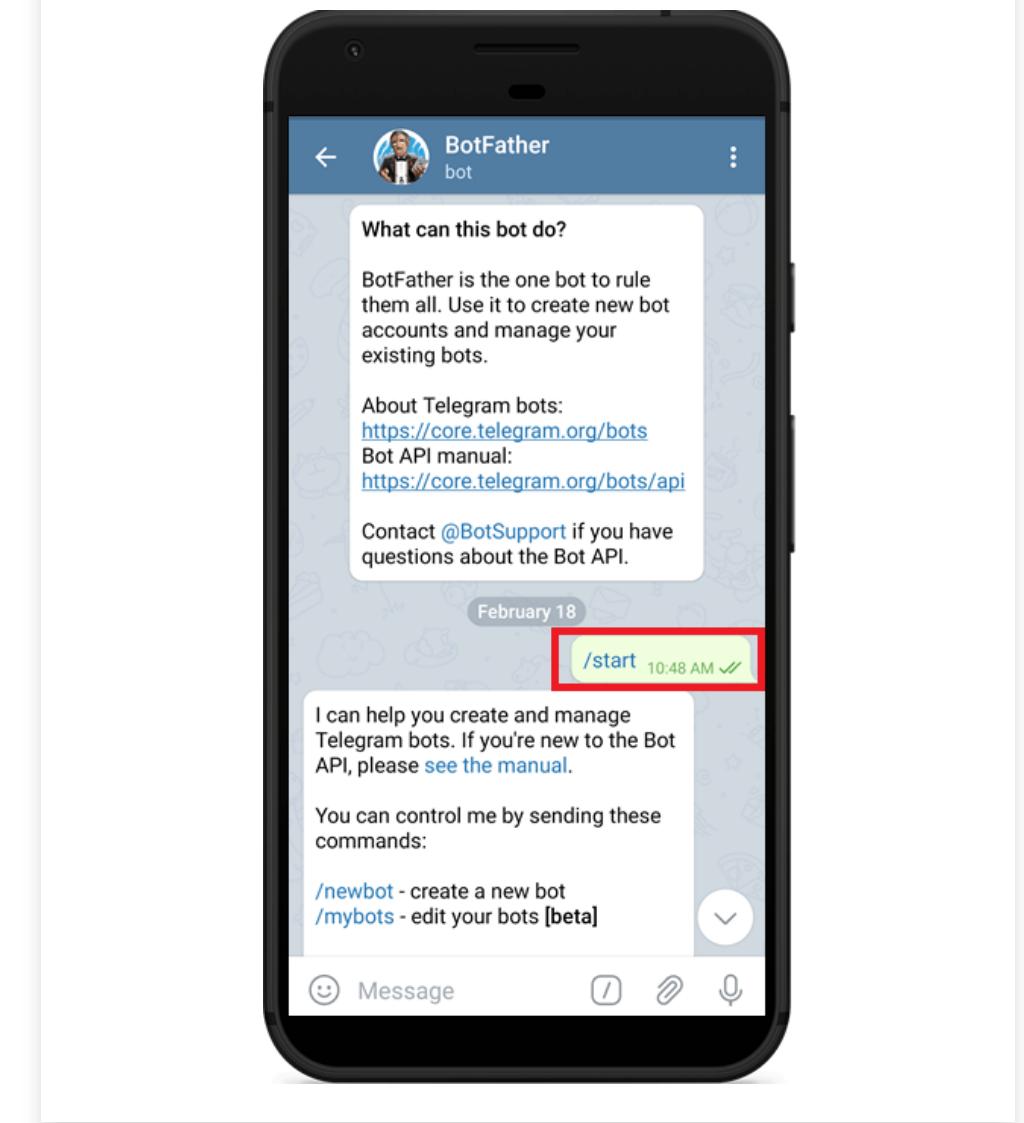
Tahap 1

Buka Telegram dan ikuti langkah-langkah berikut untuk membuat Bot Telegram. Pertama, cari "botfather" dan klik BotFather seperti yang ditunjukkan di bawah ini. Atau buka tautan ini t.me/botfather di smartphone Anda.



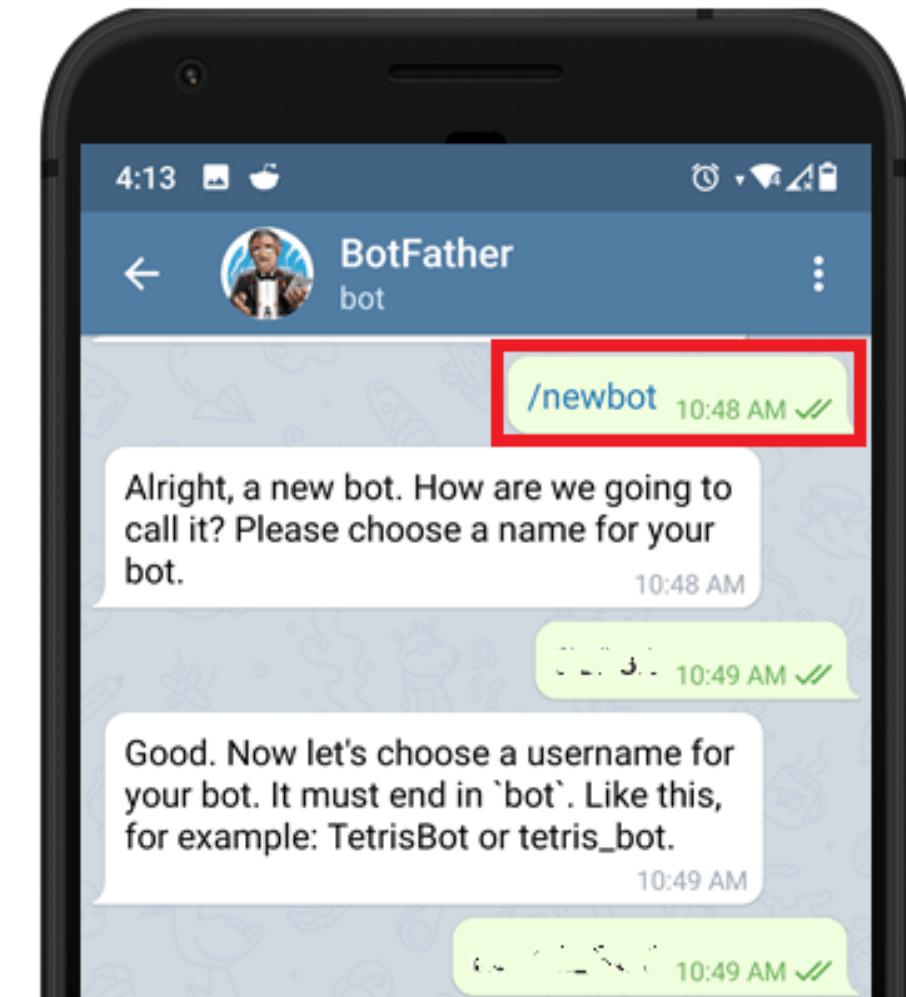
Tahap 2

Setelah diketikaan maka selanjutnya Ketika /start untuk memulai bot.



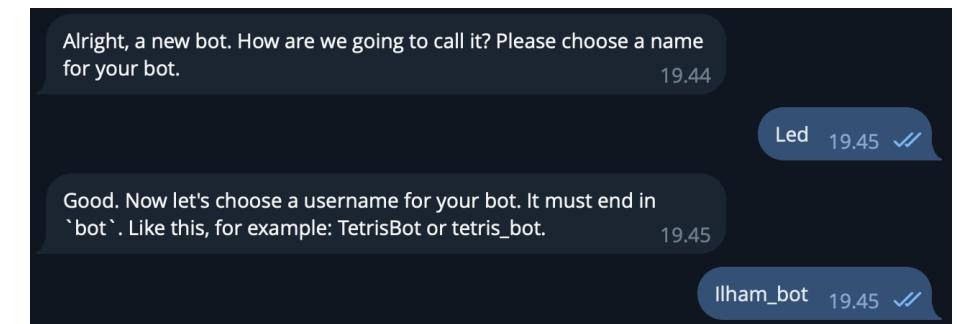
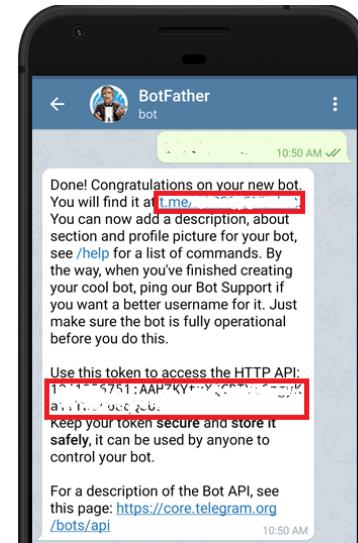
Tahap 3

- Ketik /newbot dan ikuti instruksi untuk membuat bot Anda. Beri nama dan nama pengguna. Tuliskan nama dan *username*



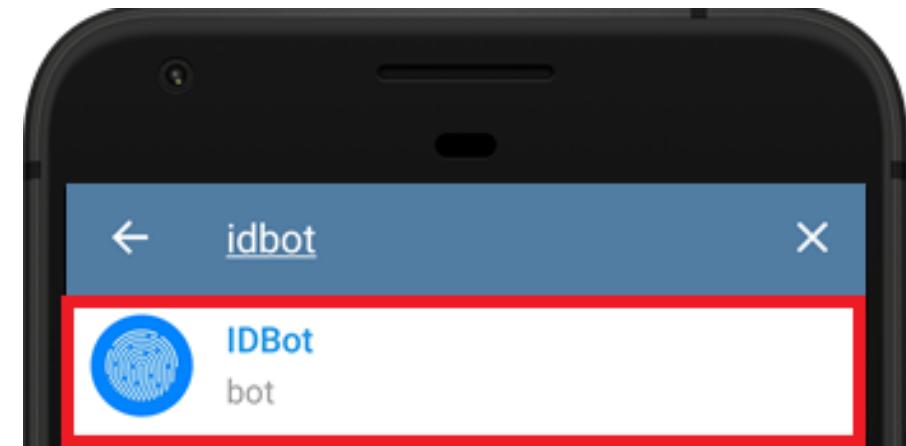
Tahap 4

Jika bot telah berhasil dibuat, maka kita akan menerima pesan berisi tautan untuk mengakses bot dan token bot. Simpan token bot karena nantinya akan diperlukan agar ESP32/ESP8266 dapat berinteraksi dengan bot.



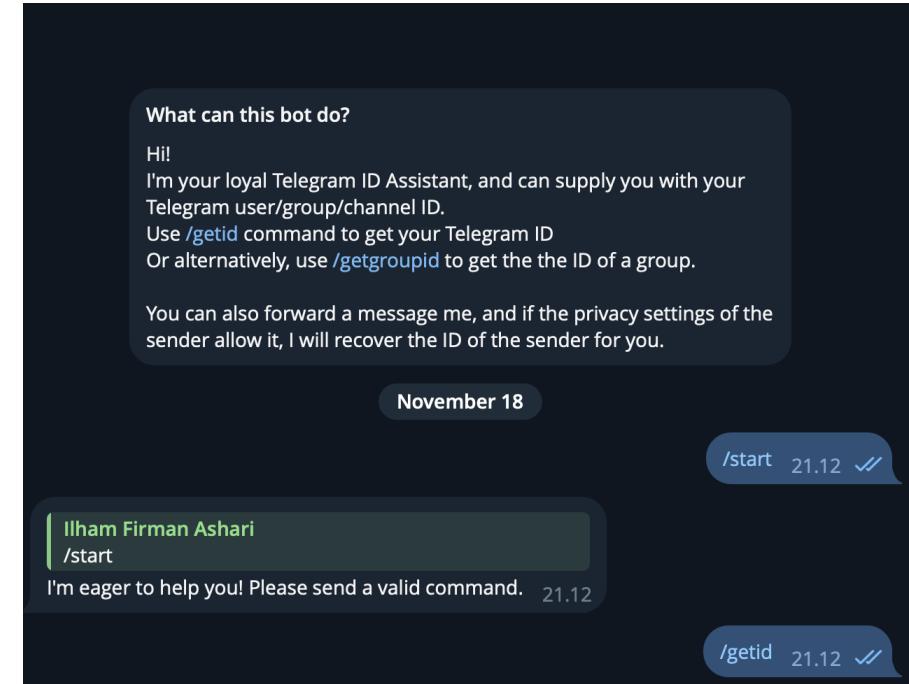
Tahap 5

Di akun Telegram Anda, cari "IDBot" atau buka tautan ini t.me/botfather di smartphone Anda.



ID BOTChat

- Klik /start
- Dan klik /getid untuk mendapatkan id



Kode Program

- Inisialisasi variable dan libraray
- BOT Token dan Chat ID

```
bot_tele_f.ino
1  #ifdef ESP32
2  | #include <WiFi.h>
3  #else
4  | #include <ESP8266WiFi.h>
5  #endif
6  #include <WiFiClientSecure.h>
7  #include <UniversalTelegramBot.h>
8  #include <ArduinoJson.h>
9  const char* ssid = "IFA-Homes";
10 const char* password = "berkah143";
11 // Initialize Telegram BOT
12 #define BOTtoken "6746225633:AAEjEHtIls2t1_1RwdotFKqXcvJqyRCcVFg"
13 #define CHAT_ID "171936607"
14 #ifdef ESP8266
15 | X509List cert(TELEGRAM_CERTIFICATE_ROOT);
16 #endif
17 WiFiClientSecure client;
18 UniversalTelegramBot bot(BOTtoken, client);
19 // Checks for new messages every 1 second.
20 int botRequestDelay = 1000;
21 unsigned long lastTimeBotRan;
22 const int ledPin = 2;
23 bool ledState = LOW;
24 |
```

Kode Program 2

- Fungsi: HandleNewMessage (atur informasi di bot)
- atur text bot untuk:
 - -led_on
 - -led_off
 - -state

```
bot_tele_f.ino
25 void handleNewMessages(int numNewMessages) {
26   Serial.println("handleNewMessages");
27   Serial.println(String(numNewMessages));
28
29   for (int i=0; i<numNewMessages; i++) {
30     // Chat id of the requester
31     String chat_id = String(bot.messages[i].chat_id);
32     if (chat_id != CHAT_ID){
33       bot.sendMessage(chat_id, "Unauthorized user", "");
34       continue;
35     }
36     // Print the received message
37     String text = bot.messages[i].text;
38     Serial.println(text);
39     String from_name = bot.messages[i].from_name;
40     if (text == "/start") {
41       String welcome = "Welcome, " + from_name + ".\n";
42       welcome += "Use the following commands to control your outputs.\n\n";
43       welcome += "/led_on to turn GPIO ON\n";
44       welcome += "/led_off to turn GPIO OFF \n";
45       welcome += "/state to request current GPIO state \n";
46       bot.sendMessage(chat_id, welcome, "");
47     }
48
49     if (text == "/led_on") {
50       bot.sendMessage(chat_id, "LED state set to ON", "");
51       ledState = HIGH;
52       digitalWrite(ledPin, ledState);
53     }
54
55     if (text == "/led_off") {
56       bot.sendMessage(chat_id, "LED state set to OFF", "");
57       ledState = LOW;
58       digitalWrite(ledPin, ledState);
59     }
60
61     if (text == "/state") {
62       if (digitalRead(ledPin)){
63         bot.sendMessage(chat_id, "LED is ON", "");
64       } else{
65         bot.sendMessage(chat_id, "LED is OFF", "");
66       }
67     }
68   }
69 }
70 }
```

Setup

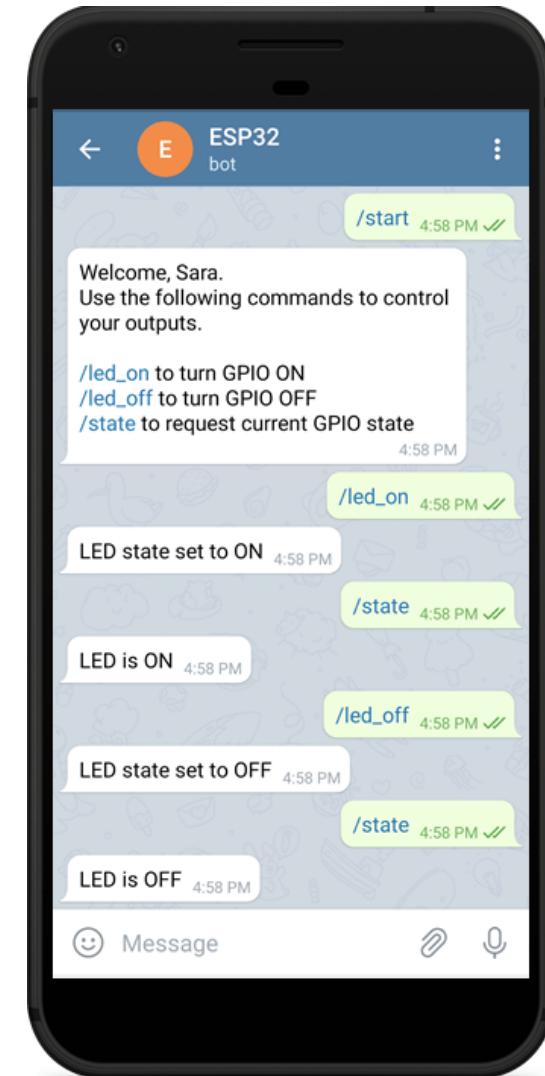
- Koneksi ke WIFI
- Dan void loop

```
72 void setup() {
73   Serial.begin(115200);
74
75 #ifdef ESP8266
76   configTime(0, 0, "pool.ntp.org");
77   client.setTrustAnchors(&cert);
78#endif
79
80 pinMode(ledPin, OUTPUT);
81 digitalWrite(ledPin, ledState);
82
83 // Connect to Wi-Fi
84 WiFi.mode(WIFI_STA);
85 WiFi.begin(ssid, password);
86 #ifdef ESP32
87   | client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
88#endif
89 while (WiFi.status() != WL_CONNECTED) {
90   delay(1000);
91   Serial.println("Connecting to WiFi..");
92 }
93 // Print ESP32 Local IP Address
94 Serial.println(WiFi.localIP());
95 }
96
97 void loop() {
98   if (millis() > lastTimeBotRan + botRequestDelay)  {
99     int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
100
101     while(numNewMessages) {
102       Serial.println("got response");
103       handleNewMessages(numNewMessages);
104       numNewMessages = bot.getUpdates(bot.last_message_received + 1);
105     }
106     lastTimeBotRan = millis();
107   }
108 }
```

Running

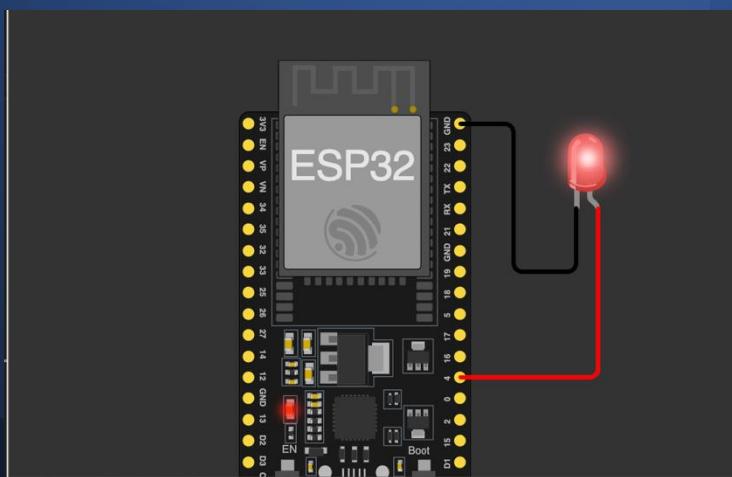
Buka akun Telegram Anda dan buka percakapan dengan bot Anda. Kirim perintah-perintah berikut dan lihat bot merespons:

- 1./start menampilkan pesan selamat datang beserta perintah-perintah yang valid.
- 2./led_on menyalaikan lampu LED.
- 3./led_off mematikan lampu LED.
- 4./state meminta status terkini dari LED.



<https://wokwi.com/projects/425439004471361537>

Contoh



entry 0x400805dc

Connecting to WiFi...

....

WiFi connected!

10.10.0.2

Pesan diterima: /start

Pesan diterima: /on

The screenshot shows the Wokwi IDE interface. On the left is the code editor with the file 'sketch.ino' containing C++ code for an ESP32. The code handles incoming messages from a chat bot, responding with welcome messages, turning the LED on or off, and reporting its state. On the right, the 'Simulation' tab shows a 3D model of the ESP32 board with a red LED glowing at pin D4. Below the simulation is a terminal window displaying the WiFi connection status and received messages. The top right corner shows the time as 01:48.480 and battery level at 98%.

```
50 void handle() {
51     for (int i = 0; i < 10; i++) {
52         if (chat_id != CHAT_ID) {
53             continue;
54         }
55
56         Serial.println("Pesan diterima: " + text);
57
58         if (text == "/start") {
59             String welcome = "Selamat datang!\n";
60             welcome += "Gunakan perintah berikut:\n";
61             welcome += "/on - Menyalakan LED\n";
62             welcome += "/off - Mematikan LED\n";
63             welcome += "/status - Melihat status LED\n";
64             bot.sendMessage(chat_id, welcome, "");
65         } else if (text == "/on") {
66             digitalWrite(ledPin, HIGH);
67             ledState = HIGH;
68             bot.sendMessage(chat_id, "LED Menyala", "");
69         } else if (text == "/off") {
70             digitalWrite(ledPin, LOW);
71             ledState = LOW;
72             bot.sendMessage(chat_id, "LED Mati", "");
73         } else if (text == "/status") {
74             if (ledState) {
75                 bot.sendMessage(chat_id, "LED dalam keadaan ON", "");
76             } else {
77                 bot.sendMessage(chat_id, "LED dalam keadaan OFF", "");
78             }
79         }
80     }
81 }
```

WiFi connected!
10.10.0.2
Pesan diterima: /led_on
Pesan diterima: /state
Pesan diterima: /led_on
Pesan diterima: /start
Pesan diterima: /on

Contoh 2

<https://wokwi.com/projects/425439061445741569>

The screenshot shows the Wokwi simulation environment. On the left, the code editor displays the `sketch.ino` file with C++ code for an ESP32 connected to a PIR sensor and a buzzer, and interfacing with a Telegram bot via WiFi. On the right, the simulation window shows the physical circuit setup with the ESP32, PIR sensor, and buzzer. The simulation log at the bottom shows the connection to the 'Wokwi-GUEST' WiFi network and a message indicating motion detection.

```
1 #include <WiFi.h>
2 #include <WiFiClientSecure.h>
3 #include <UniversalTelegramBot.h>
4 #include <ArduinoJson.h>
5
6 // Pengaturan WiFi SSID dan Password
7 const char* ssid = "Wokwi-GUEST"; //SSID Wifi yang digunakan
8 const char* password = ""; //Password Wifi tersebut
9
10 // Pengaturan API Bot Telegram dan Chat ID
11 const char* botToken = "7792039437:AAGTRoxSnRQGke_q-VVvNiCVFPjyx4RLCJM"; //API Bot
12 const char* chatID = "7433715043"; //Kepada siapakah bot tersebut mengirimkan pesan
13
14 // Declare Pin untuk PIR Sensor dan Buzzer
15 const int pirPin = 12; // Pin GPIO pada ESP32
16 const int buzzerPin = 2; // Pin GPIO pada ESP32
17
18 // Inisialisasi WiFiClientSecure untuk enkripsi SSL/TLS
19 WiFiClientSecure secured_client;
20
21 // Inisialisasi objek bot Telegram dengan WiFiClientSecure
22 UniversalTelegramBot bot(botToken, secured_client);
23
24 void setup() {
25     Serial.begin(115200);
26
27     // Inisialisasi PIR Sensor dan Buzzer
28     pinMode(pirPin, INPUT);
29     pinMode(buzzerPin, OUTPUT);
30     digitalWrite(buzzerPin, LOW);
31
32     // Koneksi ke WiFi
33     WiFi.begin(ssid, password);
34     while (WiFi.status() != WL_CONNECTED) {
35         delay(250);
36         Serial.print(".");
37     }
38 }
```

Simulation Log:

```
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
.....
WiFi terhubung
Wokwi-GUEST
Ada Gerakan.
```

Firebase



A photograph of a rocket launching from a launch pad. The rocket is white with blue stripes and is angled upwards, leaving a bright orange and yellow plume of fire and smoke against a dark blue sky. The foreground is dark, suggesting a night launch or a low-light environment.

Sejarah

- Firebase adalah suatu layanan dari Google untuk memberikan kemudahan bahkan mempermudah para developer aplikasi dalam mengembangkan aplikasinya.
- Firebase alias BaaS (*Backend as a Service*)
- Singkat cerita mengenai sejarah dari Firebase didirikan pertama kali pada tahun 2011 oleh Andrew Lee dan James Tamplin

Karakteristik Firebase Realtime Database

1. Database Real-Time
2. Struktur JSON
3. Synchronization
4. API RESTful
5. Security Rules
6. Dukungan untuk Aplikasi Android, iOS, dan Web
7. Skalabilitas



Tahapan Instalasi

- Tahapan yang dilakukan antara lain:
1. Pergi ke website <https://firebase.google.com> dan silahkan untuk melakukan sign ini menggunakan akun google
 2. Selanjutnya click Get Started dan Add Project untuk membuat project Baru
 3. Click *Get Started*, and then *Add project* to create a new project.
 4. Berikan nama pada project yang diingikan, misalkan : ESP32 Firebase Demo

Tahap 1

- Tuliskan nama project

×

Create a project (Step 1 of 3)

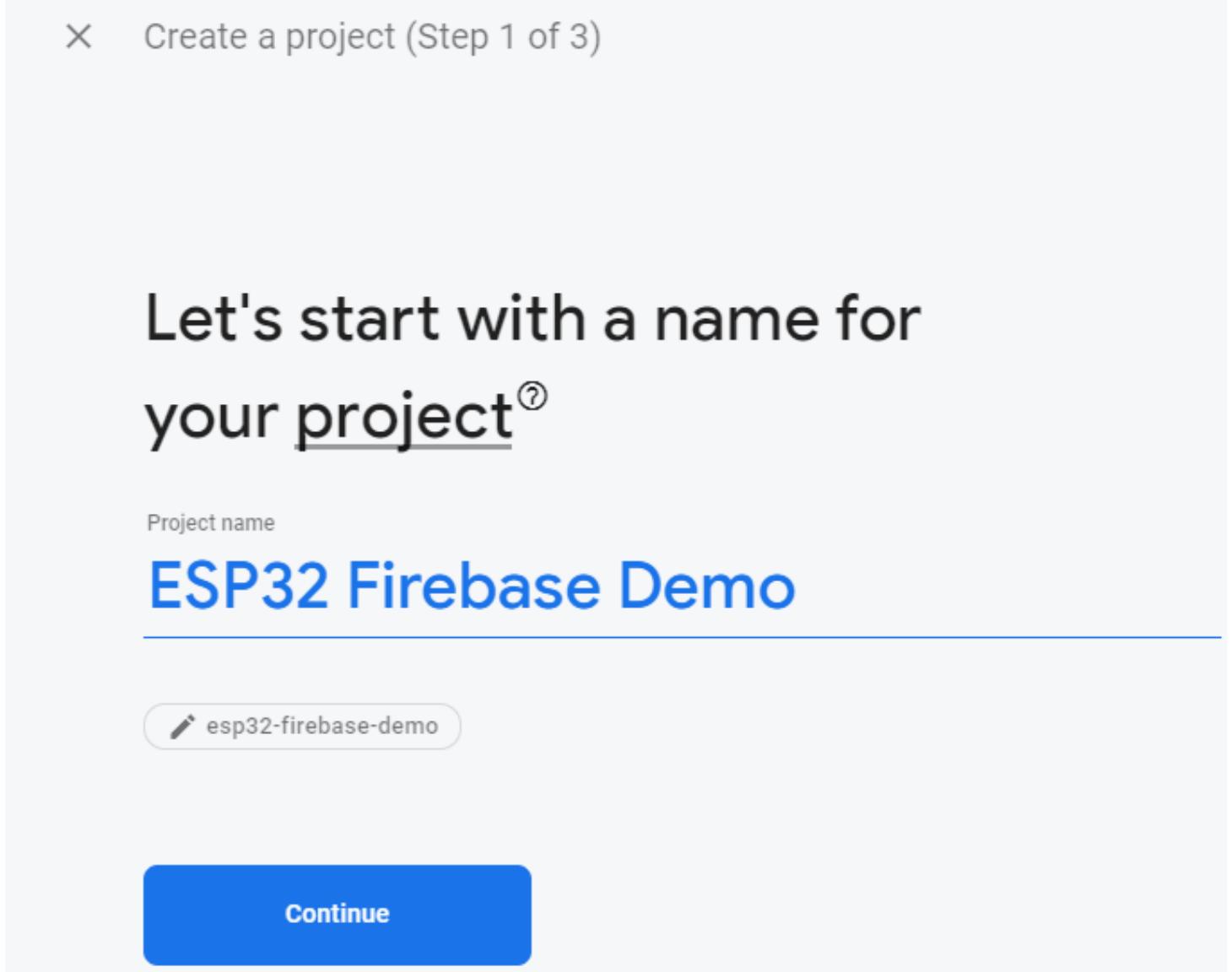
Let's start with a name for your project®

Project name

ESP32 Firebase Demo

 esp32-firebase-demo

Continue



Tahap 2

- Lakukan disable pada pilihan *Enable Google Analytics* dan selanjutnya click Create Project.

×

Create a project (Step 2 of 2)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.

Google Analytics enables:

- ×
- A/B testing ⓘ
- ×
- Crash-free users ⓘ
- User segmentation & targeting across ⓘ
- Firebase products
- Event-based Cloud Functions triggers ⓘ
- Free unlimited reporting ⓘ
- Predicting user behavior ⓘ

Enable Google Analytics for this project

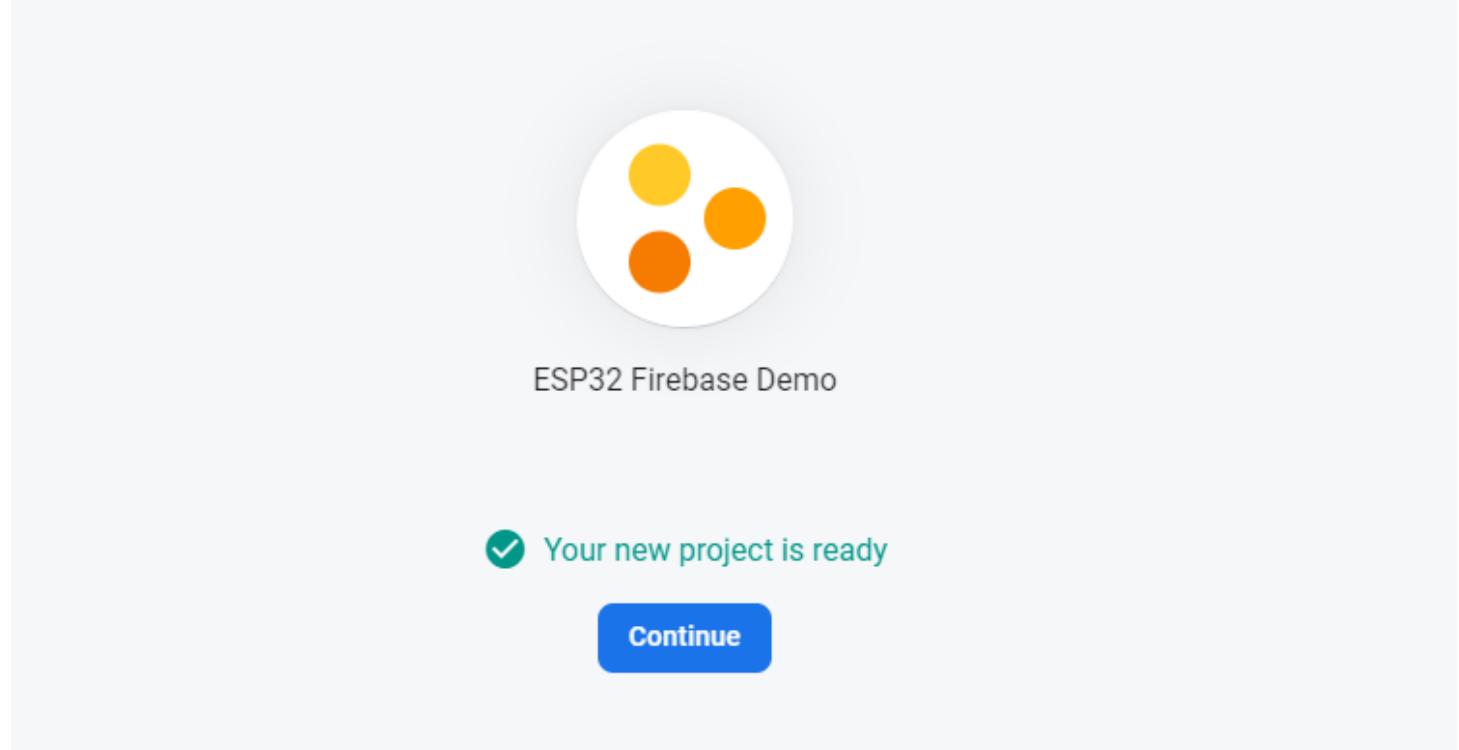
Recommended

Previous

Create project

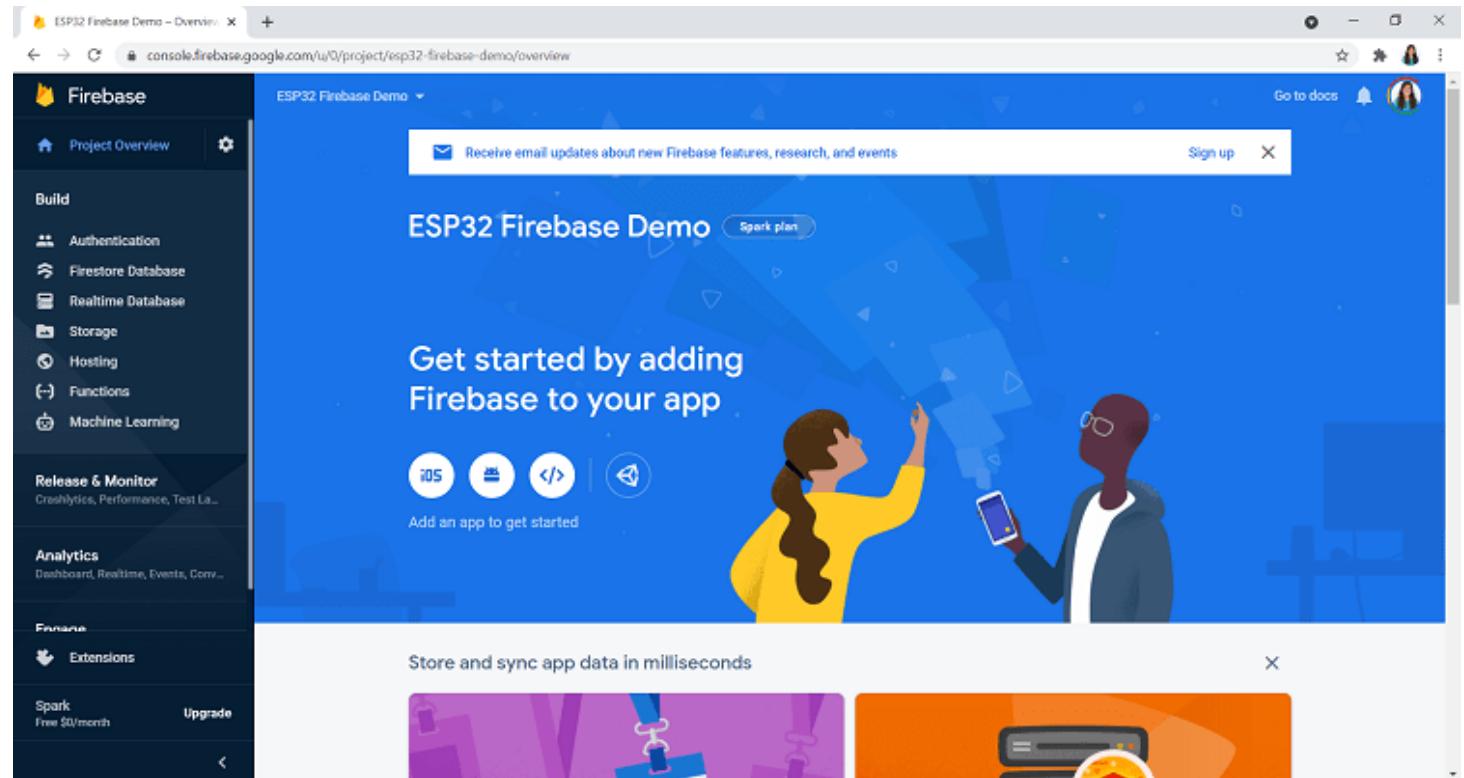
Tahap 3

- Diperlukan beberapa detik untuk menyiapkan proyek Anda. Lalu, klik continue jika sudah siap.



Tahap 4

- Selanjutnya akan diarahkan ke halaman console project yang akan dikembangkan

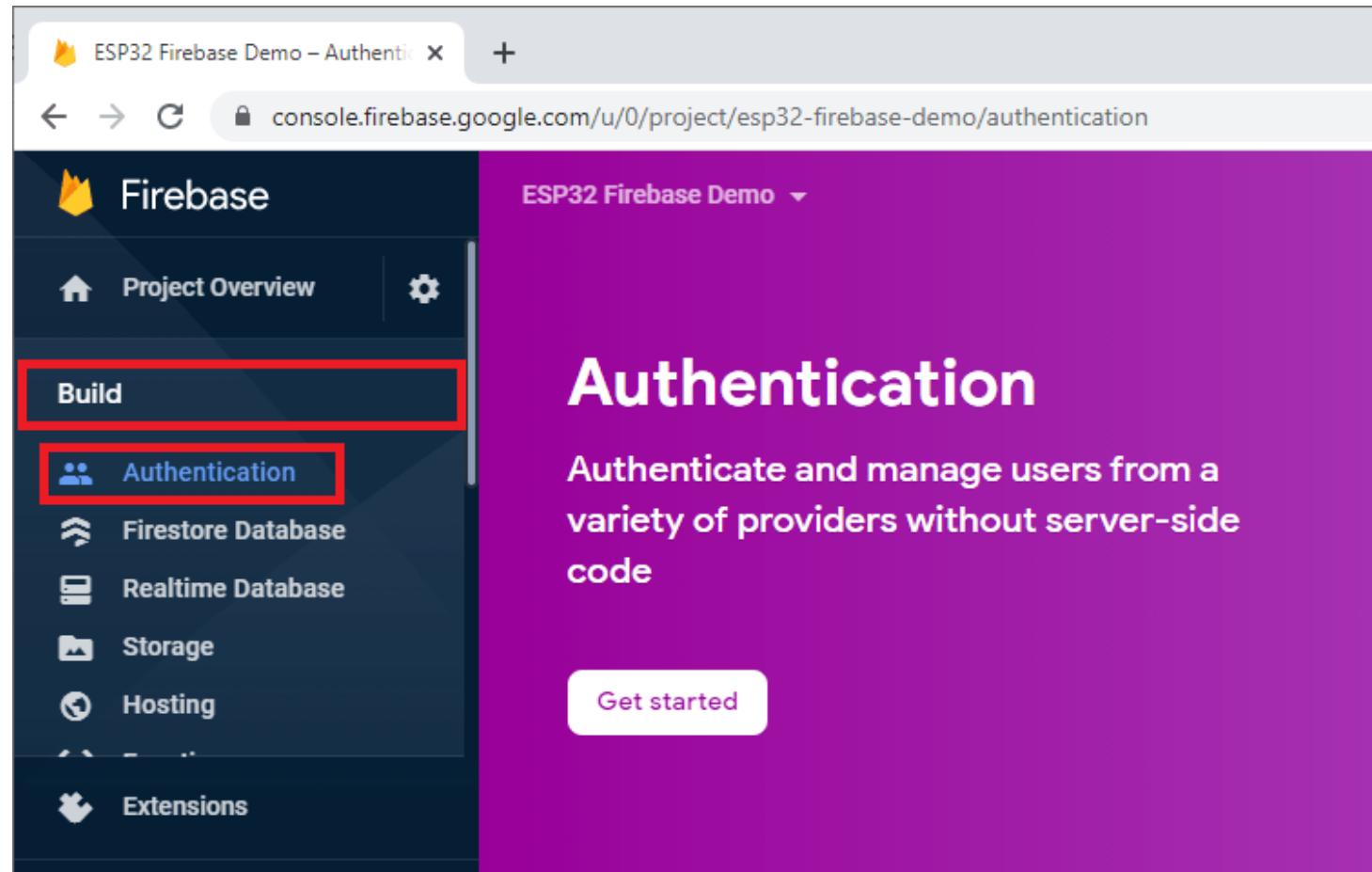


Identitas pengguna menjadi aspek krusial bagi sebagian besar aplikasi, yang melibatkan proses login untuk mengidentifikasi pengguna, seperti pada ESP32. Dengan informasi identitas pengguna yang terkelola, aplikasi dapat menyimpan data pengguna dengan aman di cloud dan menyajikan pengalaman terpersonalisasi yang konsisten di berbagai perangkat pengguna.

Pada menu sidebar sebelah kiri, klik *Authentication* dan *Get Started*

Tahap 1

- Pada menu sidebar sebelah kiri, klik *Authentication* dan *Get Started*



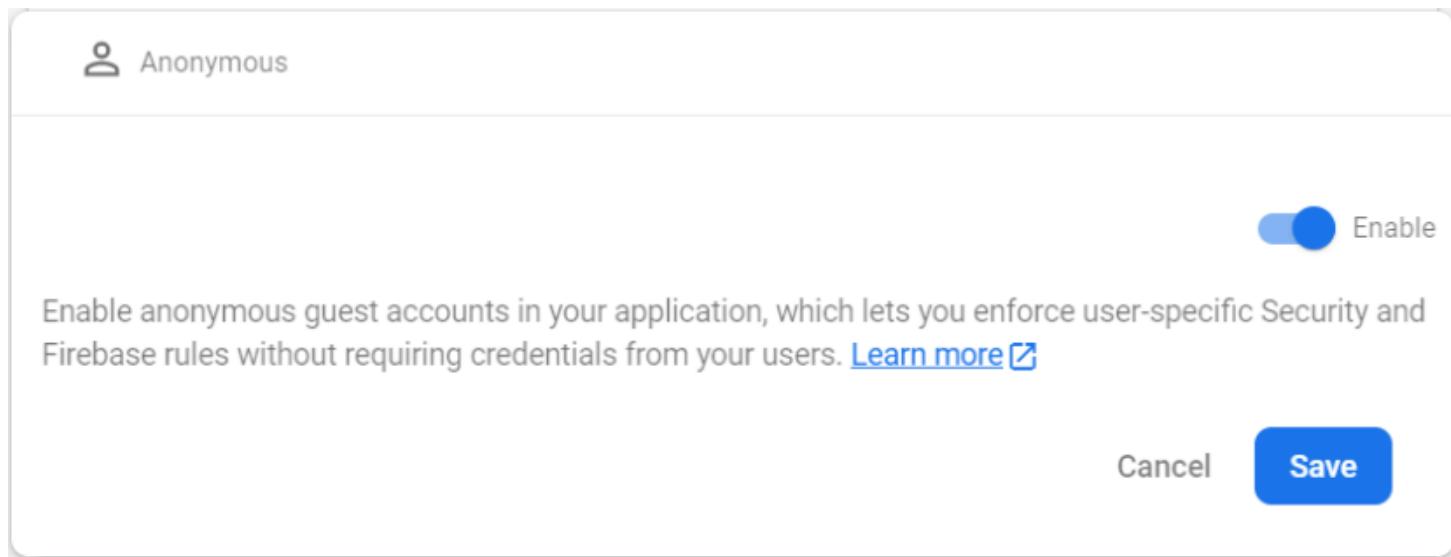
Tahap 2

- Terdapat banyak metode otentikasi yang dapat dipilih misalkan **email atau password, akun facebook, akun google, akun twitter, anonymous, dll.**

Sign-in providers		
Provider	Status	
Email/Password	Disabled	
Phone	Disabled	
Google	Disabled	
Play Games	Disabled	
Game Center	Disabled	
Facebook	Disabled	
Twitter	Disabled	
Github	Disabled	
Yahoo	Disabled	
Microsoft	Disabled	
Apple	Disabled	
Anonymous	Disabled	

Tahap 3

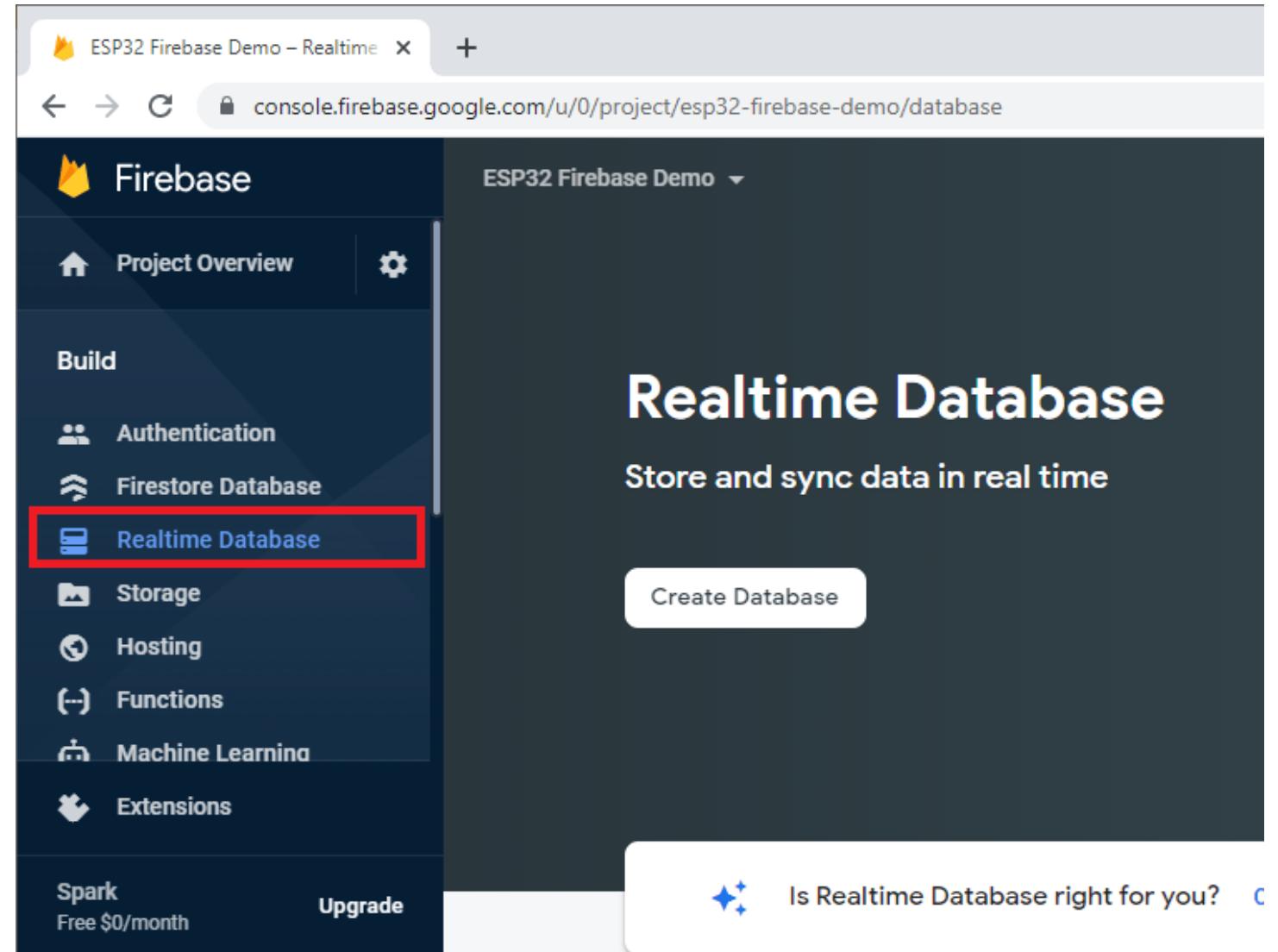
- Untuk tujuan pengujian, kita dapat memilih pengguna Anonim (membutuhkan otentikasi tanpa meminta pengguna untuk masuk terlebih dahulu dengan membuat akun anonim sementara). Aktifkan opsi tersebut dan klik Simpan.



Tahap 4

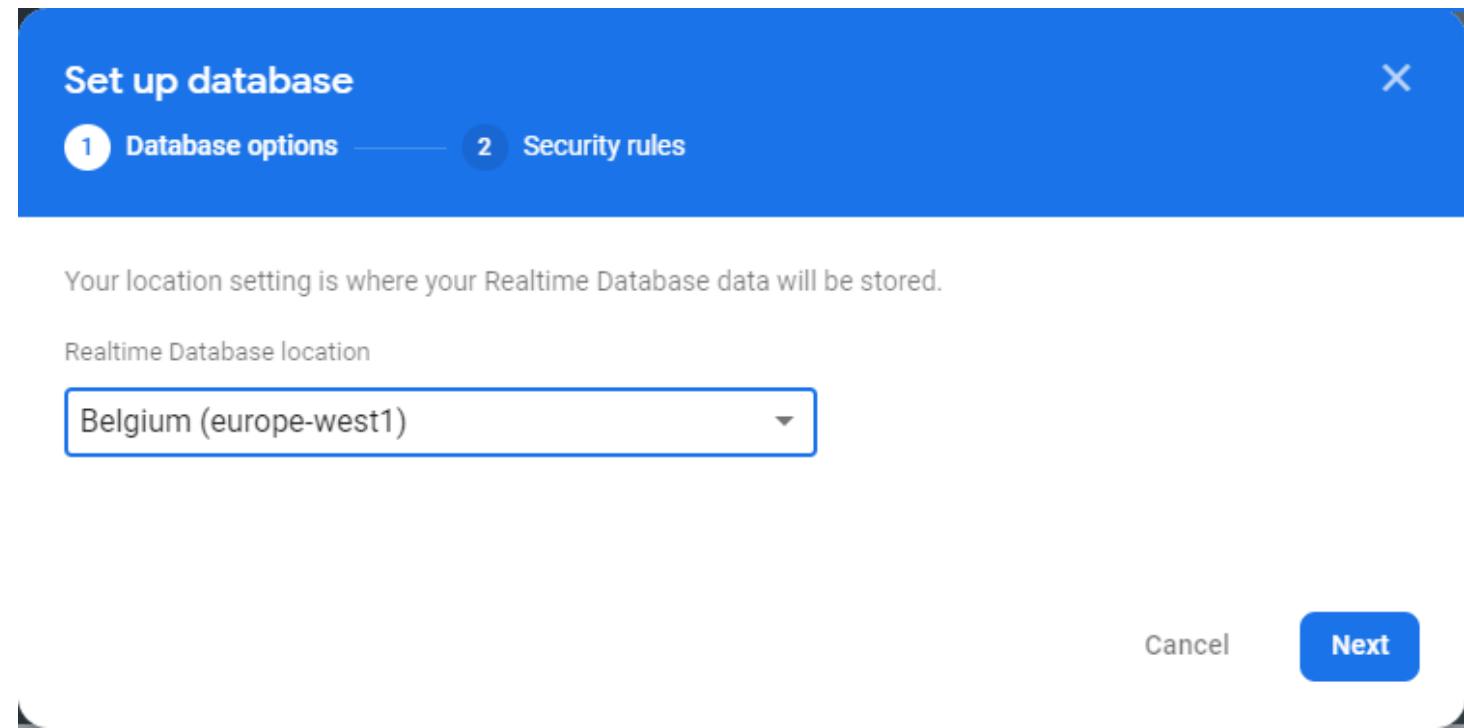
Pada tahap ini akan dibahas cara untuk membuat realtime database. Tahapan yang dilakukan antara lain:

- Pada sidebar sebelah kiri klik Realtime Database lalu klik Create Database.



Tahap 5

- Kemudian pilih lokasi realtime database sesuai domisili terdekat.



Tahap 6

- Atur mode keamanan untuk database. Untuk tujuan pengujian, silahkan pilih *Start in test mode*.

Set up database

1 Database options —— 2 Security rules

Once you have defined your data structure you will have to write rules to secure your data.
[Learn more](#)

Start in **locked mode**
Your data is private by default. Client read/write access will only be granted as specified by your security rules.

Start in **test mode**
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

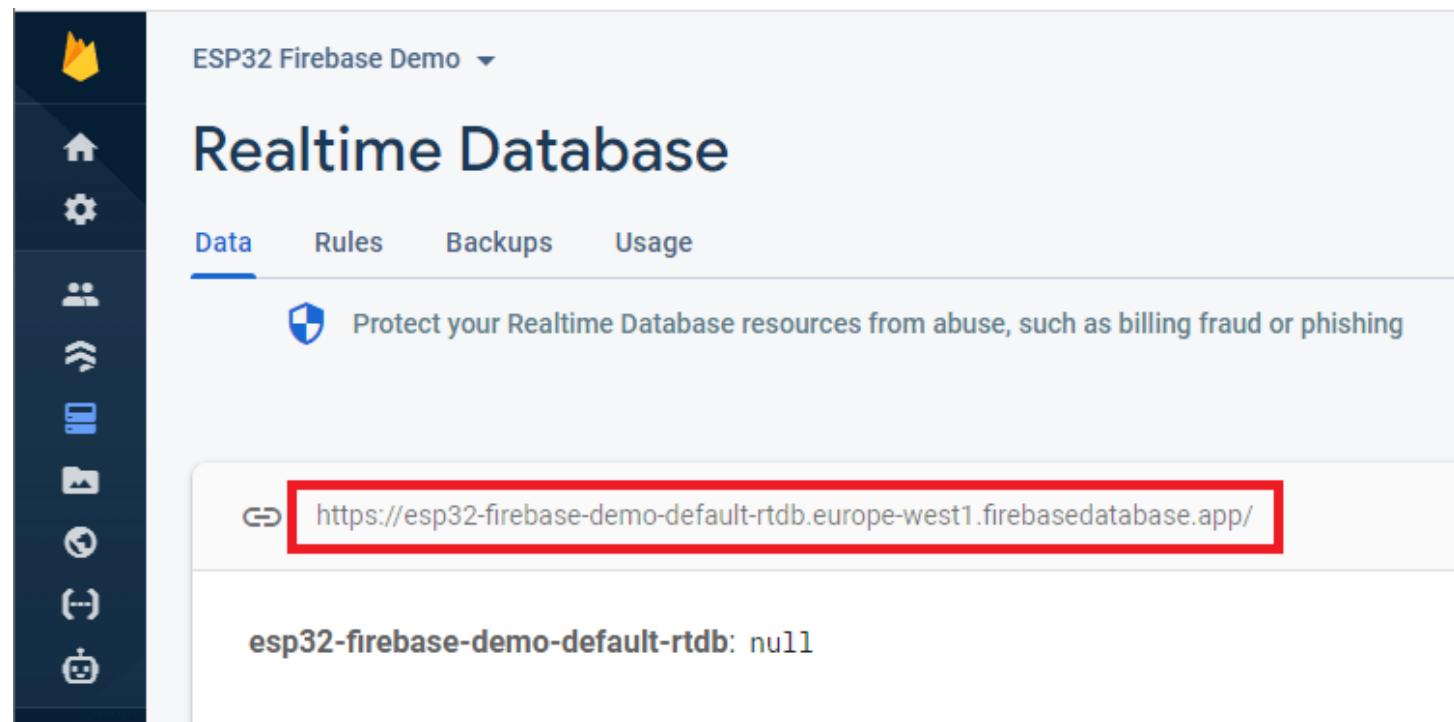
```
{  
  "rules": {  
    ".read": "now < 1630537200000", // 2021-9-2  
    ".write": "now < 1630537200000", // 2021-9-2  
  }  
}
```

! The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Cancel **Enable**

Tahap 7

- Database telah terbuat. Kita membutuhkan untuk copy dan save URL database (nanti akan dibutuhkan untuk kode program).



The screenshot shows the Firebase Realtime Database console for a project named "ESP32 Firebase Demo". The left sidebar has a dark theme with icons for Home, Settings, Groups, Analytics, Storage, Functions, and Cloud Firestore. The main area is titled "Realtime Database" and shows the "Data" tab selected. A blue shield icon with a checkmark is followed by the text "Protect your Realtime Database resources from abuse, such as billing fraud or phishing". Below this, there is a copy icon next to the database URL: <https://esp32-firebase-demo-default-rtdb.firebaseio.com/>. This URL is highlighted with a red rectangular box. At the bottom, the text "esp32-firebase-demo-default-rtdb: null" is displayed.

The screenshot shows the Firebase console interface. On the left, there's a sidebar with icons for Project Overview, Build, and Authentication. The main area has a dark header with the Firebase logo and the project name "ESP32 Firebase Demo". A dropdown menu is open over the "Project settings" link, which is highlighted with a red box. The menu items include "Project settings" (selected), "Users and permissions", "Usage and billing", and "Phishing". To the right of the menu, the "Database" section is visible, featuring tabs for "Backups", "Usage", and a large warning message about abuse.

Firebase

ESP32 Firebase Demo

Project settings

Users and permissions

Usage and billing

Phishing

Database

Backups Usage

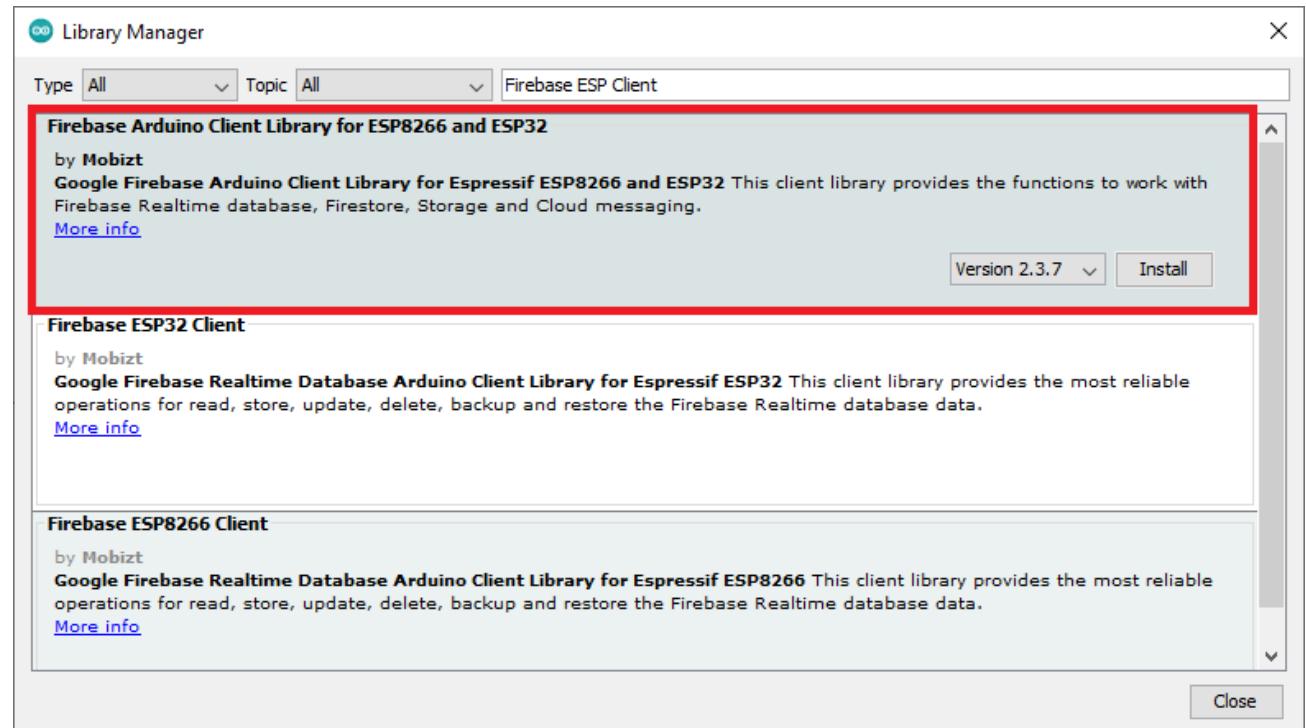
Prevent Database resources from abuse, such as billing fra

The screenshot shows the Firebase Project settings interface for the project "ESP32 Firebase Demo". The left sidebar lists various services: Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning, and a partially visible Analytics & Monitor section. The main content area is titled "Project settings" and has a "General" tab selected. Below the tabs, there's a "Your project" summary section. The "Web API Key" field is highlighted with a red border and contains the value: APE3yEPlfJ9cWhVQGzmtLjOuLsgjQZAcCGuyk.

Setting	Value
Project name	ESP32 Firebase Demo
Project ID	esp32-firebase-demo
Project number	112982951680
Default GCP resource location	Not yet selected
Web API Key	APE3yEPlfJ9cWhVQGzmtLjOuLsgjQZAcCGuyk

Library (Firebase)

- Jika Anda menggunakan Arduino IDE, ikuti langkah selanjutnya untuk menginstal perpustakaan.
 1. Buka Sketch > Include Library > Manage Libraries
 2. Cari Firebase ESP Client dan instal Firebase Arduino Client Library untuk ESP8266 dan ESP32 oleh Mobitz.



Contoh

<https://wokwi.com/projects/425440393242213377>

WOKWI SAVE SHARE PERCOBAAN FIREBASE PERT7 (IFA) Docs

sketch.ino diagram.json libraries.txt Library Manager

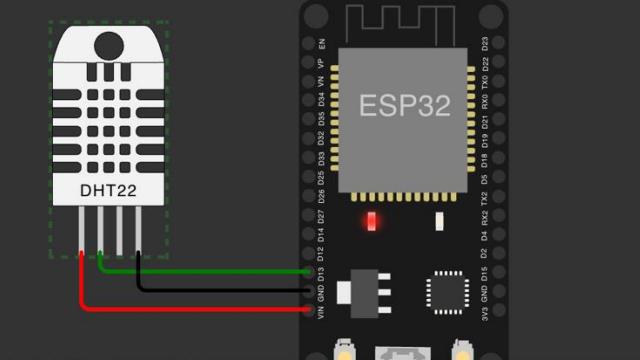
```
1 #include <WiFi.h> // untuk mengaktifkan wifi pada ESP32, agar memungk
2 #include <DHT.h> // untuk membaca data dari sensor DHT yang terpasang
3 #include <FirebaseESP32.h> //Dengan menggunakan FirebaseESP32, ESP32 d
4 #include <HTTPClient.h> // memungkinkan ESP32 untuk membuat permintaan
5
6 const char* ssid = "Wokwi-GUEST"; //nama jaringan wifi SSID yang akan
7 const char* password = ""; //kata sandi untuk mengakses jaringan WiFi
8
9 //Ini mendefinisikan sebuah konstanta makro dengan nama FIREBASE_HOST,
10 #define FIREBASE_HOST "https://dht-22-7ca44-default-rtdb.firebaseio.com"
11 //Ini mendefinisikan konstanta makro dengan nama FIREBASE_AUTH, yang n
12 #define FIREBASE_AUTH "AIzaSyC2d8-WEySkU0-zr_uA98h3B0_SakJVDi4"
13
14 #define DHTPIN 13 //sensor DHT22 dihubungkan ke pin 13 pada mikrokontrol
15 #define DHTTYPE DHT22 // untuk men definisikan jenis sensor DHT yang d
16
17 #define led 2 //digunakan untuk menentukan pin yang terhubung ke LED p
18 FirebaseData fbdo; //Ini mendeklarasikan objek dari kelas FirebaseData
19
20
21 void wifiConnection(){ //Memulai proses koneksi WiFi dengan menggunakan
22 WiFi.begin(ssid, password); // Memulai proses koneksi WiFi dengan me
23 while (WiFi.status() != WL_CONNECTED){ //Sebuah loop while yang akan
24   Serial.print("."); //untuk mengirim data ke Serial Monitor
25   delay(500); // perangkat menunggu selama 500 milidetik (setengah
26 }
27 Serial.println("Wifi Connected"); //Mencetak pesan bahwa perangkat t
28 Serial.println(WiFi.localIP()); //Mencetak alamat IP lokal perangkat
29 }
30
31 DHT dht(DHTPIN, DHTTYPE); //dapat digunakan untuk membaca nilai suhu c
32
33 void firebase(void) { //Membuat fungsi bernama firebase yang bertanggu
34   Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //Memulai koneksi ke F
35   Firebase.reconnectWiFi(true); //untuk memastikan bahwa koneksi WiFi
36 }
```

Simulation 00:26.461 59%

Editing DHT22

Temperatur: 46.7°C

Humidity: 69.0%



load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
.....Wifi Connected
10.10.0.2

[Project Overview](#)

Project shortcuts

[Authentication](#)[Firestore Database](#)[Realtime Database](#) ⓘ

What's new

[Genkit](#) NEW[Vertex AI](#) NEW

Product categories

Build

Run

Analytics

AI

All products

Related development tools

Realtime Database

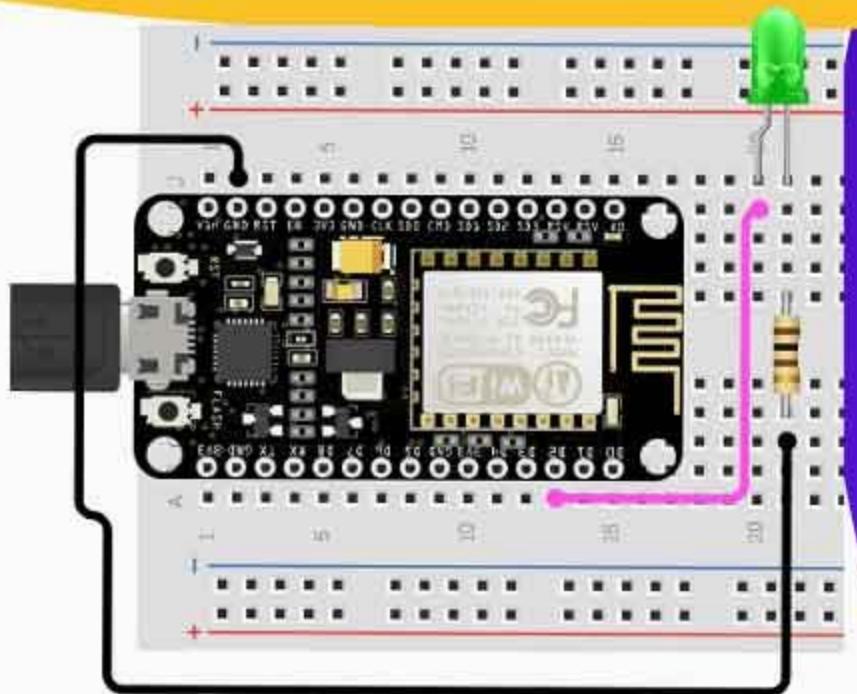
[Need help with Realtime Database? Ask Gemini](#)[Data](#)[Rules](#)[Backups](#)[Usage](#)[Extensions](#)

Protect your Realtime Database resources from abuse, such as billing fraud or phishing

[Configure App Check](#)<https://dht-22-7ca44-default-rtbd.firebaseio.west1.firebaseio.app><https://dht-22-7ca44-default-rtbd.firebaseio.west1.firebaseio.app/>

SUHU: 46.7

IoT Based LED Control using Google Firebase & ESP8266



ESP8266



Firebase

Firebase Code

- Deklarasi library dan API Key serta URL Firebase

```
firebase_led_2.ino
1  #include <Arduino.h>
2  #if defined(ESP32)
3  | #include <WiFi.h>
4  #elif defined(ESP8266)
5  | #include <ESP8266WiFi.h>
6  #endif
7  #include <Firebase_ESP_Client.h>
8
9  //Provide the token generation process info.
10 #include "addons/TokenHelper.h"
11 //Provide the RTDB payload printing info and other helper functions.
12 #include "addons/RTDBHelper.h"
13
14 // Insert your network credentials
15 #define WIFI_SSID "IFA-Homes"
16 #define WIFI_PASSWORD "berkah143"
17
18 // Insert Firebase project API Key
19 #define API_KEY "████████████████████████████████"
20
21 // Insert RTDB URLdefine the RTDB URL */
22 #define DATABASE_URL "https://ledcontrol-7557c-default-rtdb.firebaseio.com/"
```

Firebase Code

2

Fungsi untuk setup

Koneksi dan Konfigurasi Firebase

```
firebase_led_2.ino
24 //Define Firebase Data object
25 FirebaseData fpdo;
26
27 FirebaseAuth auth;
28 FirebaseConfig config;
29
30 //some importent variables
31 String sValue, sValue2;
32 bool signupOK = false;
33 int led1 = 2; //PIN D4 NODEMCU ESP8266 AMICA
34 void setup() {
35   Serial.begin(115200);
36
37   pinMode(led1,OUTPUT);
38   // pinMode(led2,OUTPUT);
39   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
40   Serial.print("Connecting to Wi-Fi");
41   while (WiFi.status() != WL_CONNECTED) {
42     Serial.print(".");
43     delay(300);
44   }
45   Serial.println();
46   Serial.print("Connected with IP: ");
47   Serial.println(WiFi.localIP());
48   Serial.println();
49
50 /* Assign the api key (required) */
51 config.api_key = API_KEY;
52
53 /* Assign the RTDB URL (required) */
54 config.database_url = DATABASE_URL;
55
56 /* Sign up */
57 if (Firebase.signUp(&config, &auth, "", "")) {
58   Serial.println("ok");
59   signupOK = true;
60 }
61 else {
62   Serial.printf("%s\n", config.signer.signupError.message.c_str());
63 }
64
65 /* Assign the callback function for the long running token generation task */
66 config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h
67
68 Firebase.begin(&config, &auth);
69 Firebase.reconnectWiFi(true);
70 }
```

Firebase code

3

Void Loop

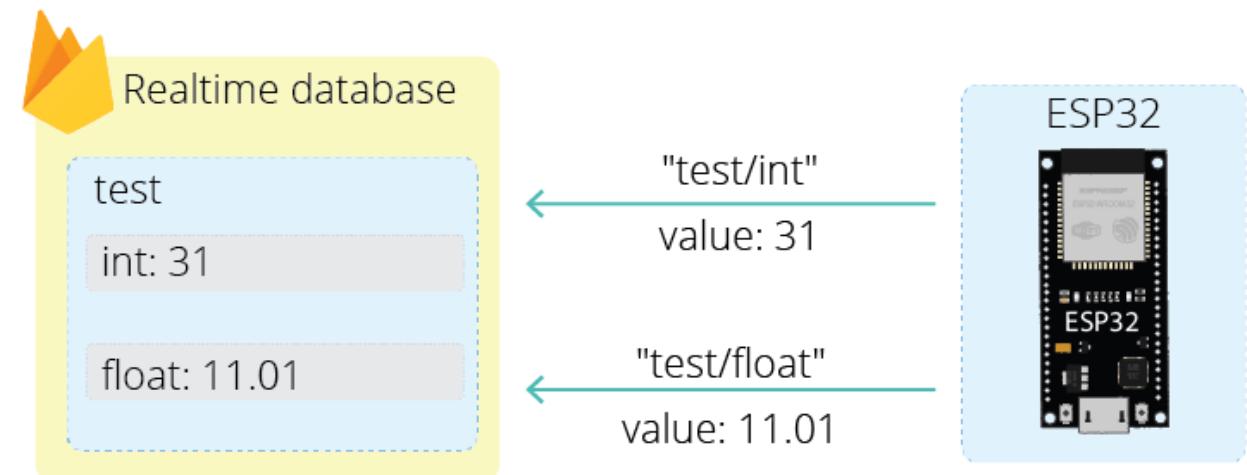
Fungsi untuk menyalakan lampu.

Menerima masukan dalam bentuk string
menggunakan **Fungsi GetString**

```
72 void loop() {  
73     if (Firebase.ready() && signupOK ) {  
74         if (Firebase.RTDB.getString(&fbdo, "/L1")) {  
75             if (fbdo.dataType() == "string") {  
76                 sValue = fbdo.stringData();  
77                 int a = sValue.toInt();  
78                 Serial.println(a);  
79                 if (a == 1){  
80                     digitalWrite(led1,HIGH);  
81                 }else{  
82                     digitalWrite(led1,LOW);  
83                 }  
84             }  
85         }  
86     else {  
87         Serial.println(fbdo.errorReason());  
88     }  
89 }  
90 }  
91 }
```

Simpan Data

- Simpan Data dengan Tipe Data : Int dan Float
- Hasil akan disimpan secara acak setiap 15 detik



Kode Program Write

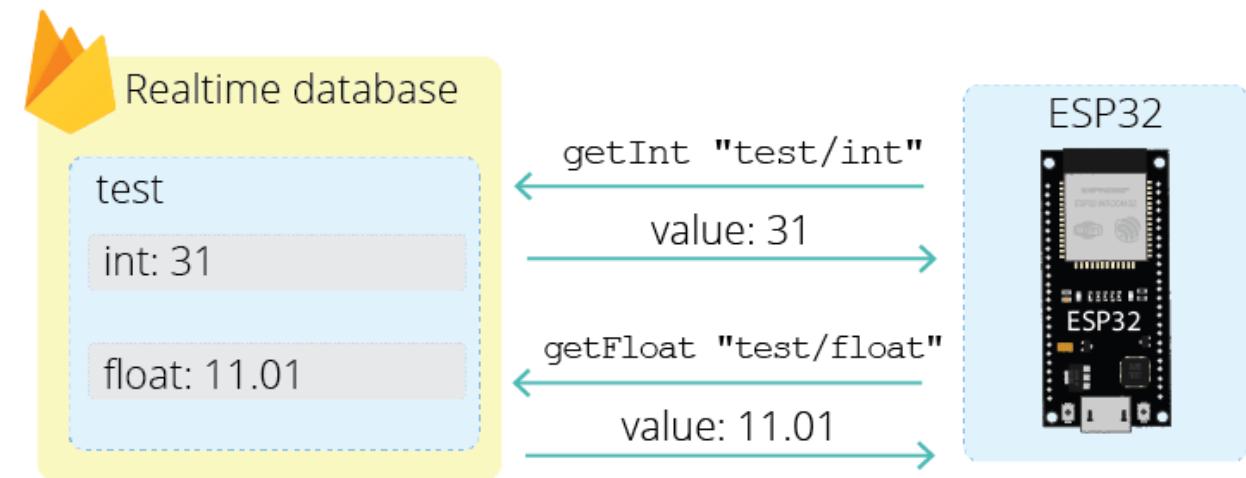
- Fungsi Loop untuk simpan data

SetInt

```
55 void loop(){
56   if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0)){
57     sendDataPrevMillis = millis();
58     // Write an Int number on the database path test/int
59     if (Firebase.RTDB.setInt(&fbdo, "test/int", count)){
60       Serial.println("PASSED");
61       Serial.println("PATH: " + fbdo.dataPath());
62       Serial.println("TYPE: " + fbdo.dataType());
63     }
64     else {
65       Serial.println("FAILED");
66       Serial.println("REASON: " + fbdo.errorReason());
67     }
68     count++;
69
70     // Write an Float number on the database path test/float
71     if (Firebase.RTDB.setFloat(&fbdo, "test/float", 0.01 + random(0,100))){
72       Serial.println("PASSED");
73       Serial.println("PATH: " + fbdo.dataPath());
74       Serial.println("TYPE: " + fbdo.dataType());
75     }
76     else {
77       Serial.println("FAILED");
78       Serial.println("REASON: " + fbdo.errorReason());
79     }
80   }
81 }
```

Baca Data

- Baca Data dengan Tipe Data : Int dan Float
- Data akan dibaca dari database setiap 15 detik



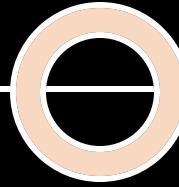
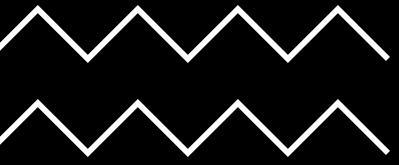
Kode Program

Read

- Fungsi Loop untuk baca data

GetInt

```
57 void loop() {  
58     if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0)) {  
59         sendDataPrevMillis = millis();  
60         if (Firebase.RTDB.getInt(&fbdo, "/test/int")) {  
61             if (fbdo.dataType() == "int") {  
62                 intValue = fbdo.intData();  
63                 Serial.println(intValue);  
64             }  
65             else {  
66                 Serial.println(fbdo.errorReason());  
67             }  
68         }  
69         if (Firebase.RTDB.getFloat(&fbdo, "/test/float")) {  
70             if (fbdo.dataType() == "float") {  
71                 floatValue = fbdo.floatData();  
72                 Serial.println(floatValue);  
73             }  
74         }  
75         else {  
76             Serial.println(fbdo.errorReason());  
77         }  
78     }  
79 }  
80 }
```



Terimakasih

