

## Topik 6

# Pengenalan Macam-macam Koneksi Protokol IoT seperti HTTP, MQTT, CoAP dan Praktikum Platform Blynk IoT V2

Digitalent Scholarship Professional Academy

**Isi dan elemen dari dokumen ini memiliki hak kekayaan intelektual yang dilindungi oleh undang-undang**

**Dilarang menggunakan, merubah, memperbanyak, dan mendistribusikan dokumen ini untuk tujuan komersil**

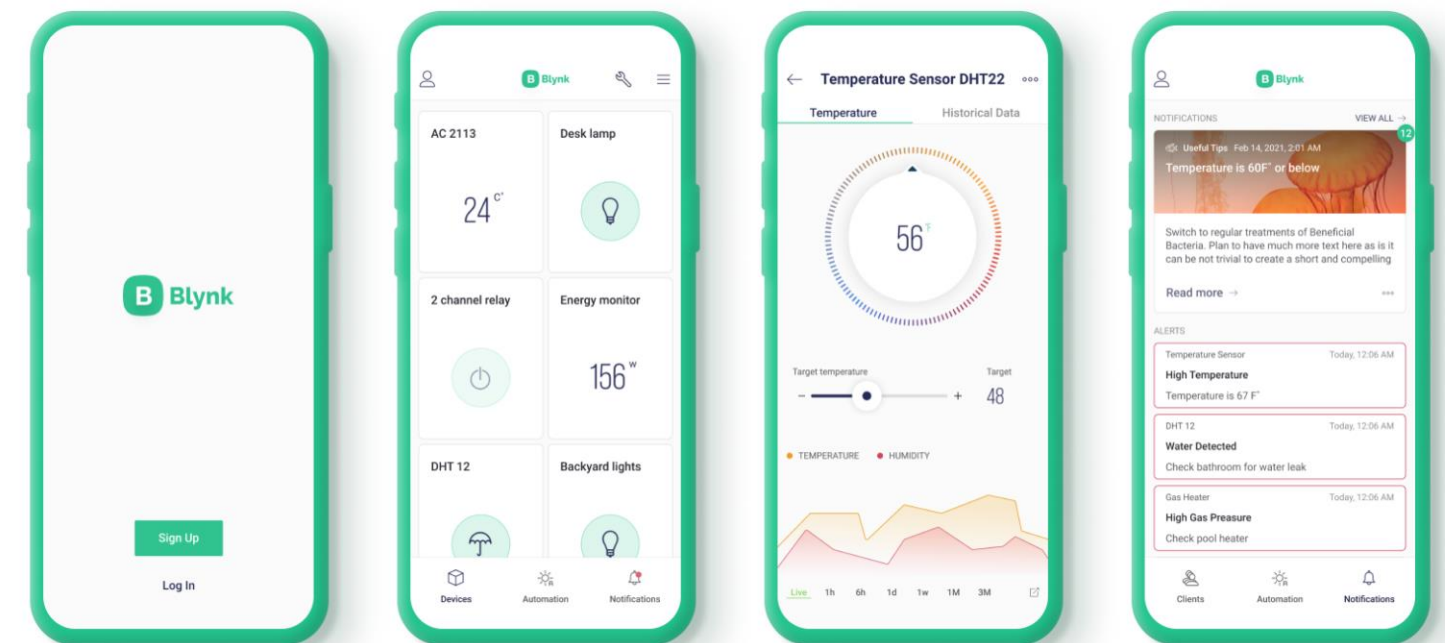
# Outline

- Protokol IoT
- HTTP
  - Pengertian HTTP
  - Cara Kerja HTTP
  - Praktikum Request dan Response HTTP
- MQTT
  - Pengenalan MQTT
  - Broker
  - Publish dan Subscribe
  - Topic
  - Cara Kerja MQTT
  - Mosquitto
  - Praktik MQTT dengan Mosquitto
- CoAP
  - Pengenalan CoAP
  - Cara Kerja CoAP
  - Praktikum Request dan Response CoAP
- Platform IoT : Blynk V2
  - Pengenalan Blynk V2
  - Praktik IoT Menggunakan Blynk V2



# Blynk IoT

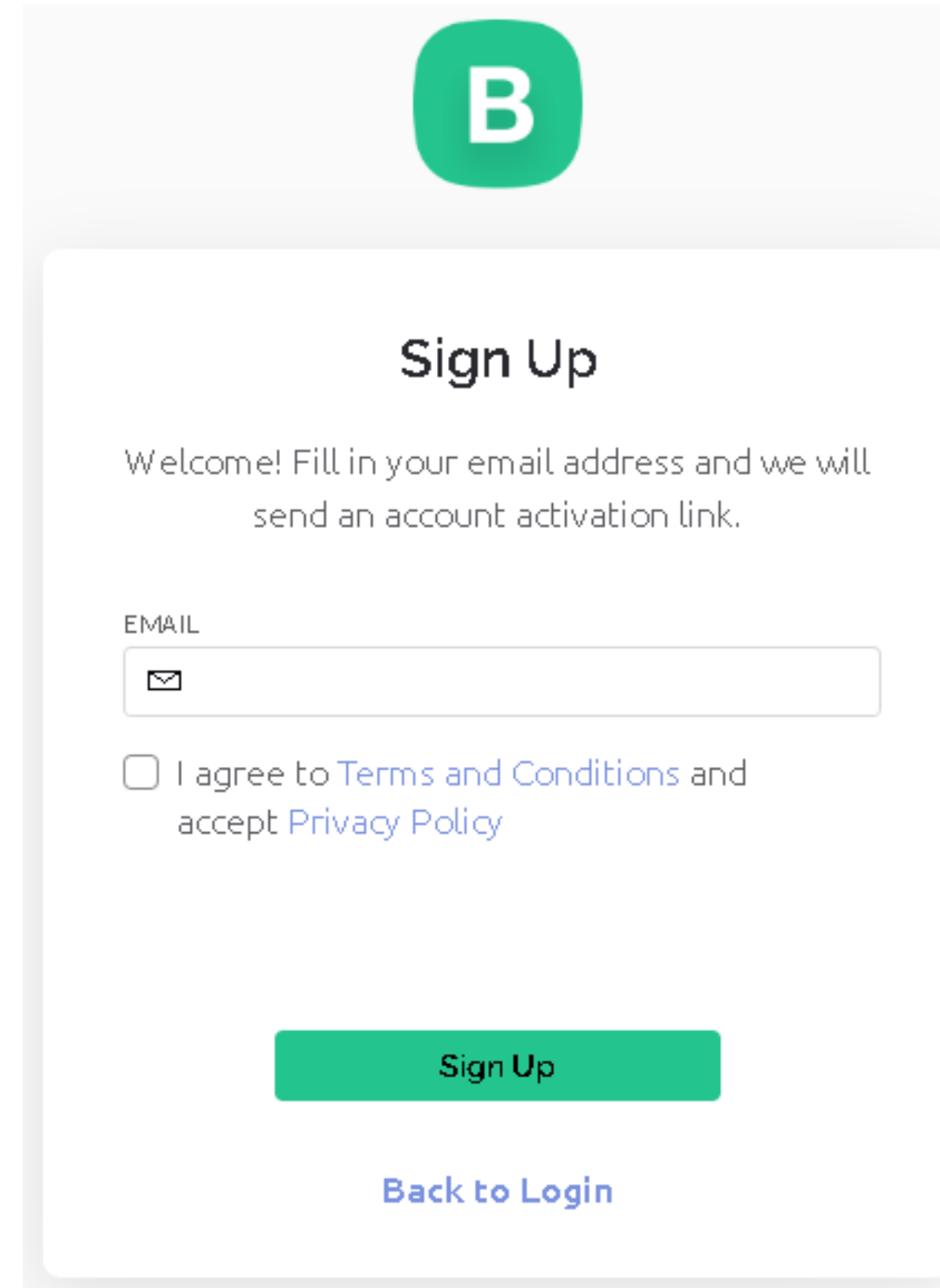
Blynk IoT merupakan platform IoT berbasis Cloud Website dan mobile apps. Dengan Blynk kita dapat membuat dashboard project IoT tanpa coding membuat website. Blynk dapat dikoneksikan dengan berbagai hardwares seperti ESP, Arduino, Raspberry Pi dan lain sebagainya Blynk dapat diunduh secara gratis tetapi terdapat beberapa fitur yang akan dibatasi dalam versi gratis.



# Membuat Akun Blynk IoT

## Konfigurasi Blynk

- Buka Website Blynk di <https://blynk.io>
- Klik Start Free untuk mendaftar
- Masukkan email
- Buka email untuk konfirmasi.
- Login menggunakan akun yang sudah dibuat.

A screenshot of the Blynk IoT website's sign-up page. At the top, there is a green rounded square logo with a white letter 'B'. Below the logo, the heading 'Sign Up' is centered in a bold, black font. Underneath the heading, a welcome message reads: 'Welcome! Fill in your email address and we will send an account activation link.' Below this message is a text input field with the placeholder text 'EMAIL' and a small envelope icon on the left. Under the input field, there is a checkbox followed by the text 'I agree to Terms and Conditions and accept Privacy Policy'. At the bottom of the form, there is a green button with the text 'Sign Up' and a blue link that says 'Back to Login'.

# HTTP

## Request dan Response

Sistem HTTP Response merupakan sebuah aktivitas dimana server memberikan jawaban terhadap request yang dikirim oleh client sebelumnya. Respon yang diberikan oleh server setelah mengirimkan data yang diinginkan adalah mengirimkan status line atau response header .Berikut ini adalah beberapa status header:

- 1xx, artinya request dari client diterima server, dan dilanjutkan dengan memberikan tindakan selanjutnya dari client
- 2xx, artinya server berhasil merespon permintaan client
- 3xx, artinya adanya redirection (pengalihan permintaan client)
- 4xx, artinya error pada client. Sehingga server tidak dapat membaca dan memproses permintaan client (kesalahan sintak HTML, atau sintak yang ditulis tidak full)
- 5xx, artinya terdapat kesalahan internal, biasanya berasal dari kesalahan pada database server.



# HTTP

## Pengenalan HTTP

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol jaringan lapisan aplikasi yang digunakan untuk sistem informasi terdistribusi, kolaboratif, dan menggunakan hipermedia.

HTTP adalah dasar dari World Wide Web (WWW) yang digunakan untuk memuat halaman web menggunakan tautan hypertext.

Saat ini HTTP sudah banyak berkembang sehingga memiliki banyak kegunaannya itu tidak hanya untuk mengambil dokumen hypertext, tetapi juga gambar dan video atau untuk mengirim konten ke server, misalnya hasil formulir HTML.



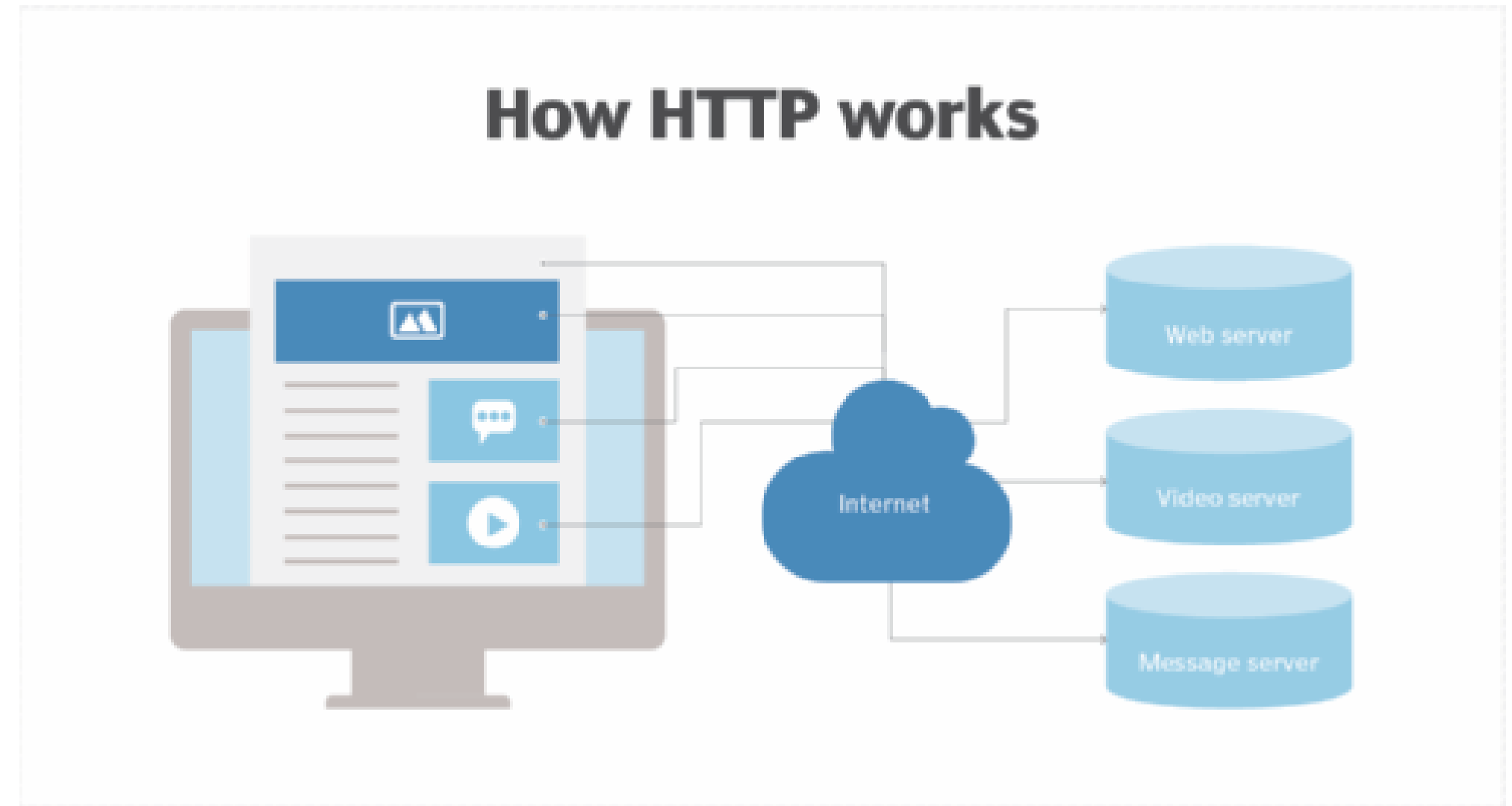
# HTTP

## Cara Kerja HTTP

HTTP membantu user dalam mengakses berbagai web resources seperti halaman website, dokumen, video, dll.

Pada web server. User menggunakan browser agar dapat mengakses berbagai macam resource didalam web server dengan menggunakan URL.

Protokol HTTP membantu koneksi yang membantu user untuk web server dengan URL tersebut.





# HTTP

## Cara Kerja HTTP

Contoh penggunaan http adalah saat kita mengunjungi <https://indobot.co.id/blog/monitoring-suhu-air-tambak-udang-berbasis-iot/>

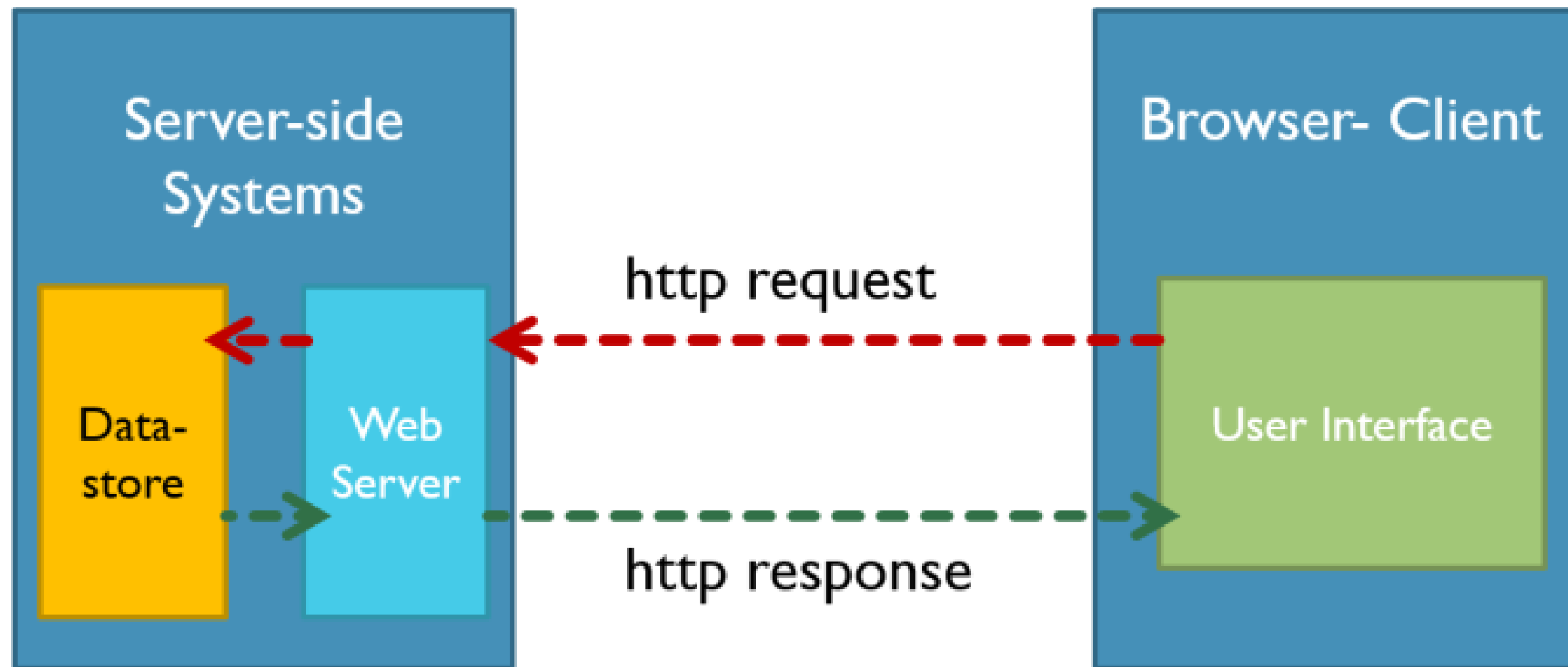
Protokol	Alamat Web Server	Lokasi Source didalam Web Server
https://	Indobot.co.id	monitoring-suhu-air-tambak-udang-berbasis-iot/

Melalui URL tersebut, protokol HTTP bertugas untuk mengirimkan permintaan (request) dan memberikan respon (response) dari request URL yang diberikan. Contohnya seperti menghapus (deleted), memperbarui (update), atau mengganti (replace) file yang ada didalam server.

# HTTP

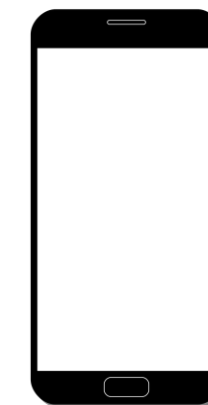
## Request dan Response

HTTP request merupakan aktivitas dimana client meminta data kepada server melalui alamat URL tertentu. Biasanya proses request dilakukan oleh client melalui web browser seperti Chrome, Firefox, Safari, dll.



# Koneksi HTTP dengan Wokwi

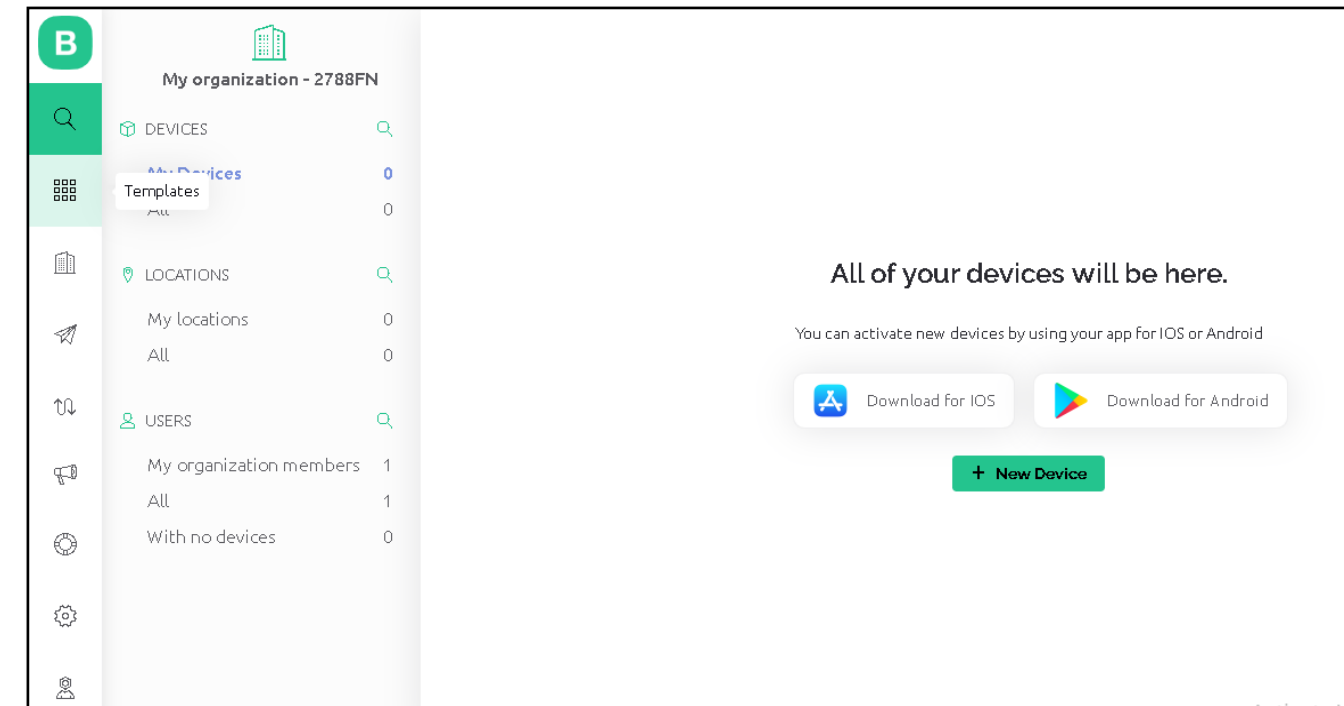
Pada Internet of Things, HTTP dapat digunakan sebagai salah satu protokol pengiriman data ke server maupun pengambilan data ke server. Kita akan mempraktikkan request data ke server dari ESP32 menggunakan simulator Wokwi.



# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

## Konfigurasi Blynk

- Klik Menu Templates.
- Klik Add New Templates untuk membuat Template Baru.
- Isikan Nama Template, Hardware adalah ESP32, Connection yaitu WiFi, masukkan deskripsi (opsional).
- Klik Done.



### Create New Template

NAME  
DHT22

HARDWARE  
ESP32

CONNECTION TYPE  
WiFi

DESCRIPTION  
Praktikum protokol IoT dengan ESP32 Wokwi

41 / 128

Cancel Done

# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

## Konfigurasi Blynk

- Perhatikan pada Firmware Configuration, terdapat Template ID yang nanti akan kita gunakan pada kode program.
- Masuk ke Menu Datastreams.
- Klik Add New Datastreams.
- Pilih Virtual Pin
- Masukkan nama data, Pin V0, Units adalah Celsius, Max yaitu 100.
- Klik Create.
- Klik Save (pojok kanan atas).
- Datastreams berhasil dibuat.

### DHT22

Info Metadata Datastreams Events Automations Web Dashboard Mobile Dashboard

DHT22

HARDWARE

ESP32

CONNECTION TYPE

WiFi

DESCRIPTION

Praktikum protokol IoT dengan ESP32 Wokwi

41 / 128

TEMPLATE ID

TMPL5EYssrpf

MANUFACTURER


My organization 2788FN

OFFLINE IGNORE PERIOD

00 hrs 00 mins 00 secs

HOTSPOT PREFIX

Hotspot Prefix



Add image

Upload from computer or drag-n-drop  
.png or .jpg, minimum width 500px

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPL5EYssrpf"
#define BLYNK_DEVICE_NAME "DHT22"
```


Template ID and Device Name should be included at the top of  
your main Firmware

# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

## Konfigurasi Blynk

- Perhatikan pada Firmware Configuration, terdapat Template ID yang nanti akan kita gunakan pada kode program.
- Masuk ke Menu Datastreams.
- Klik Add New Datastreams.
- Pilih Virtual Pin
- Masukkan nama data, Pin V0, Units adalah Celsius, Max yaitu 100.
- Klik Create.
- Klik Save (pojok kanan atas).
- Datastreams berhasil dibuat.

### Virtual Pin Datastream



NAME

Suhu

ALIAS

Suhu

PIN

V0

DATA TYPE

Integer

UNITS

Celsius, °C

MIN

0

MAX

100

DEFAULT VALUE

0

ADVANCED SETTINGS

Cancel

Create

### Datastreams

Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.

+ New Datastream

Digital

Analog

Virtual Pin

Enumerable

Location

UPGRADE

# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

## Konfigurasi Blynk

- Masuk ke menu Search.
- Klik Add New Devices untuk menambahkan devices baru.
- Pilih From Templates.
- Pilih Template DHT22.
- Masukkan nama device.
- Klik Create.

## New Device

Create new device by filling in the form below

TEMPLATE

DHT22

DEVICE NAME

DHT22

Cancel

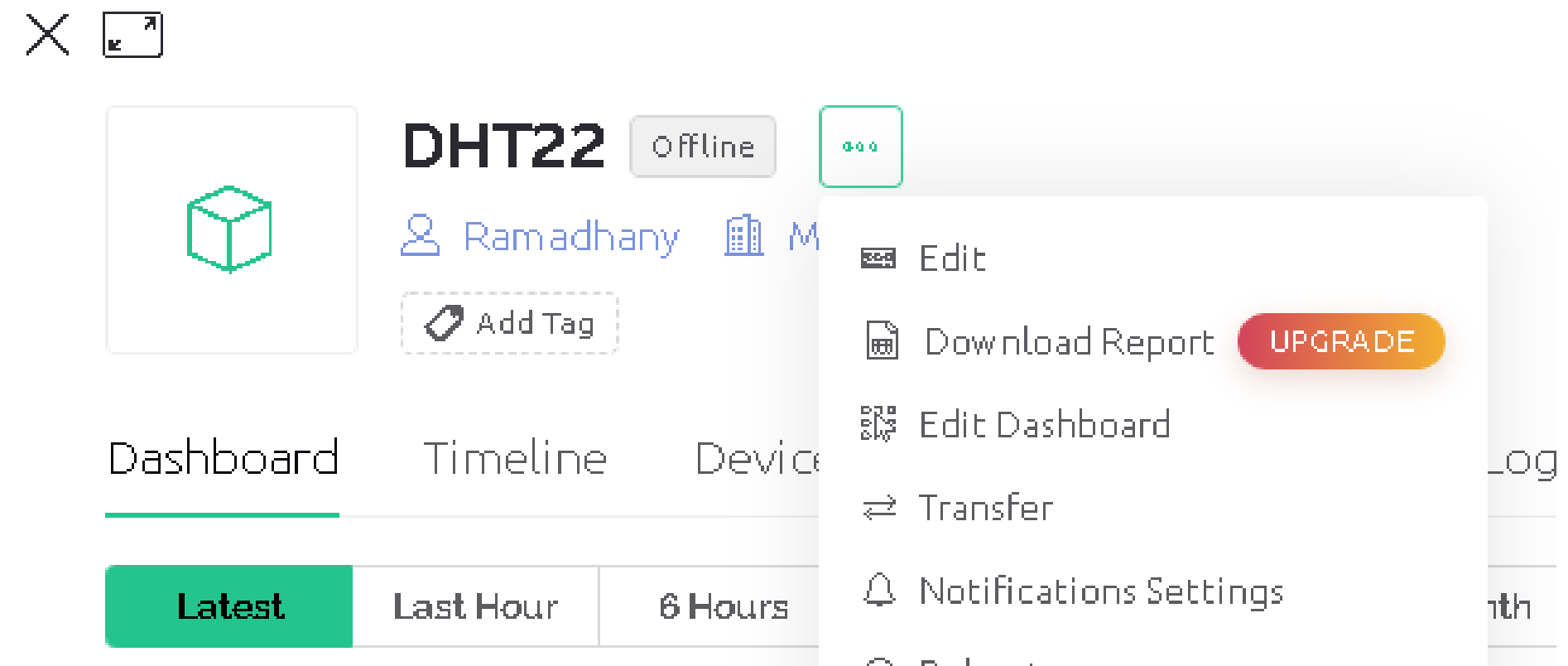
Create



# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

## Konfigurasi Blynk

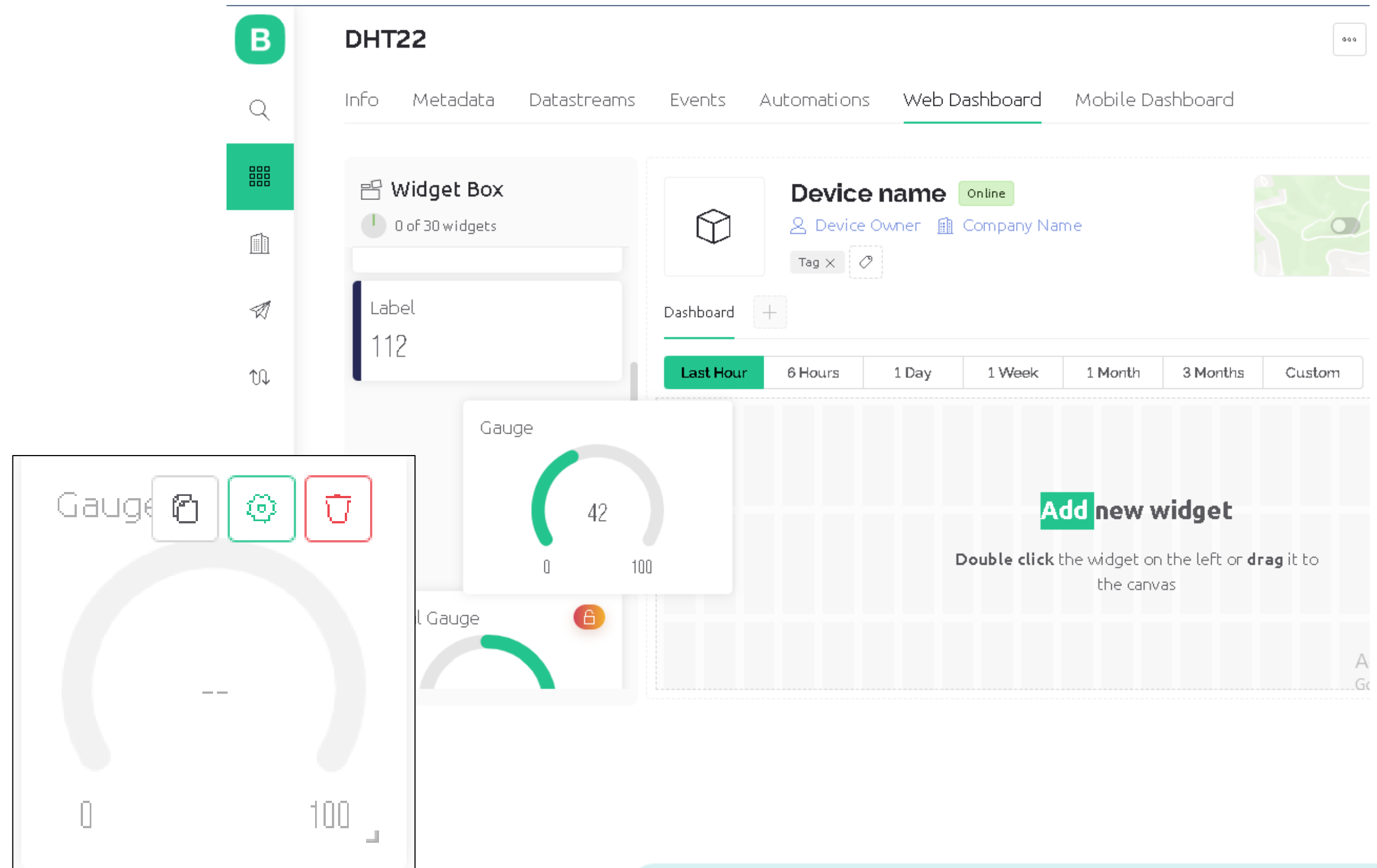
- Membuat dashboard dengan klik disamping nama devices.
- Klik Edit Dashboard.



# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

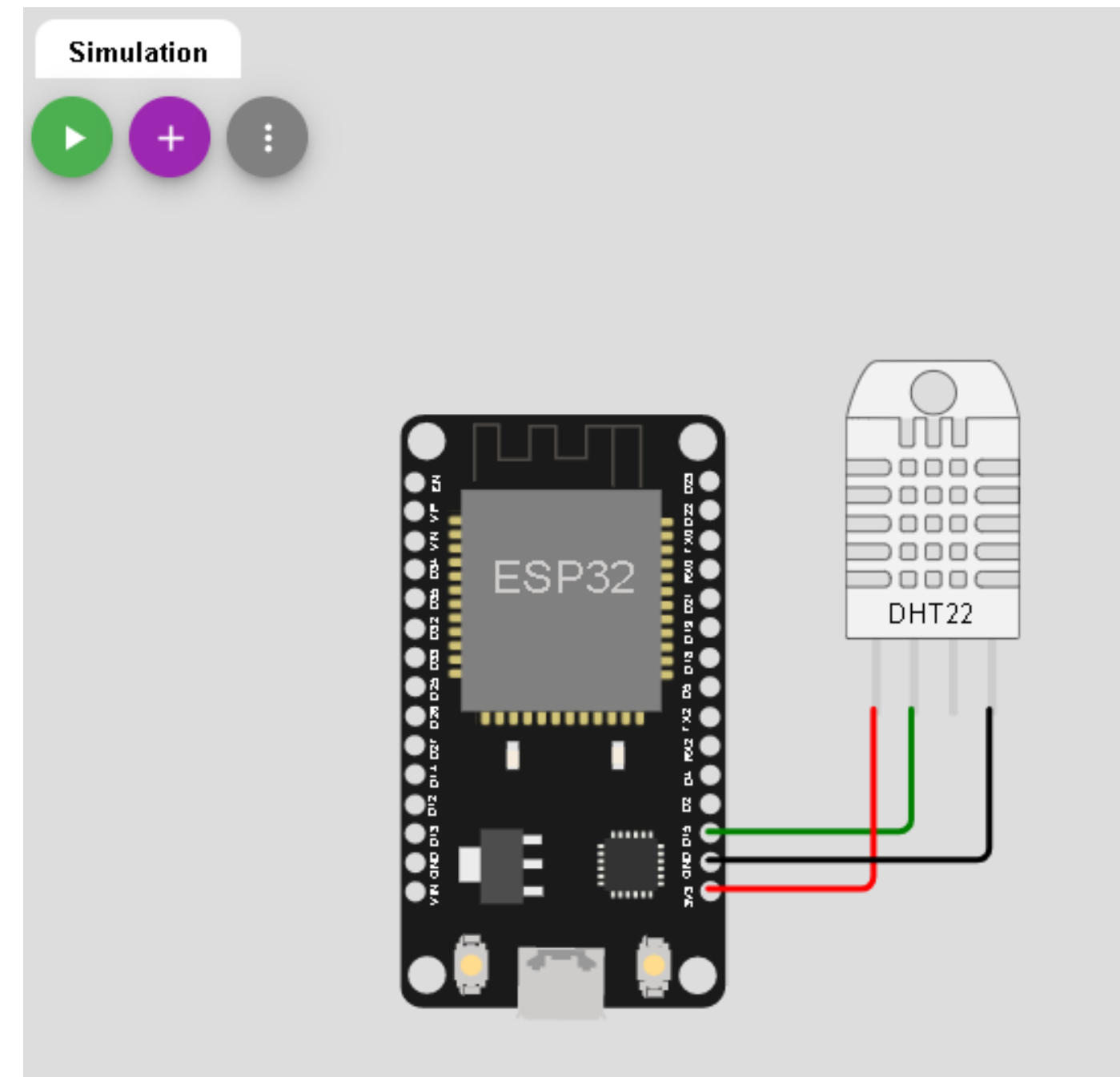
# KonfigurasiBlynk

- Pilih Widget yang diinginkan, misal Gauge dan drag ketengah.
- Klik Setting pada Gauge yang ditambahkan.
- Pilih Data stream yang sudah dibuat yaitu Suhu.
- Klik Save And Apply (pojokkananatas).



# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

- Buka Wokwi
- Klik add device (+), pilih DHT22
- Hubungkan DHT22 dengan ESP32 sesuai berikut :
  - VCC >> 3V3
  - GND >> GND
  - SDA >> D15



# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

- Buatlah program seperti gambar disamping
- Program akan menampilkan data pengukuran kelembaban dan temperatur dan juga status koneksi Blynk pada serial monitor.
- Pada bagian ssid dan pass, diisi dengan nama WiFi dan password yang Anda gunakan
- Untuk auth diisi dengan auth token dari website blynk sebelumnya.

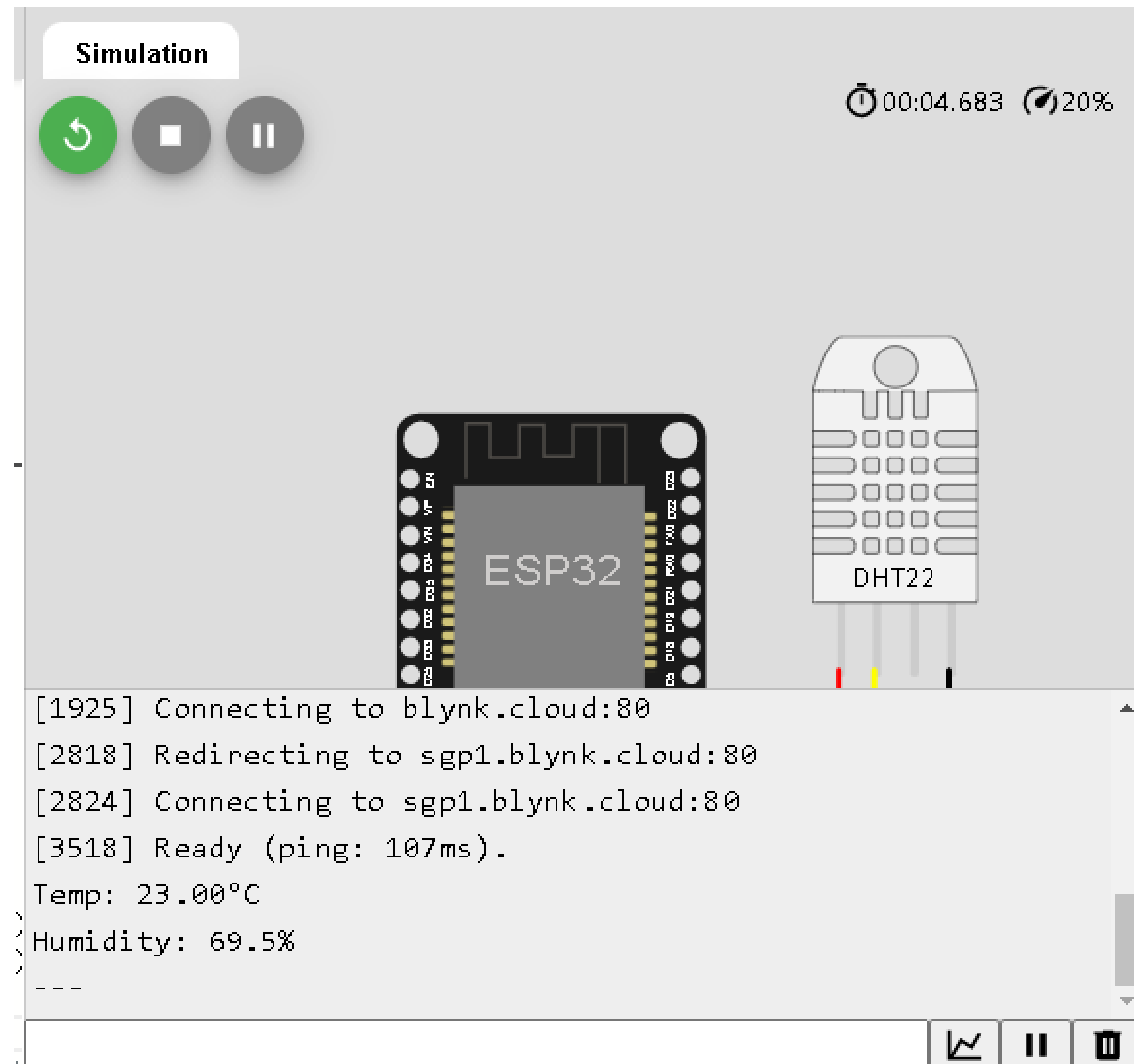
Coding : [Download](#)

```
blynk.ino • diagram.json libraries.txt Library Manager
1 // Define Blynk pin and DHT22 pin
2
3
4 #define BLYNK_AUTH_TOKEN "nBs9gfFUTbxedF2Q5Eav43UqUZZZlgzG"
5 #define LED 2
6
7 #include <WiFi.h>
8 #include <WiFiClient.h>
9 #include <BlynkSimpleEsp32.h>
10 #include "DHTesp.h"
11
12 const int DHT_PIN = 15;
13 DHTesp dhtSensor;
14
15 char auth[] = BLYNK_AUTH_TOKEN;
16 char ssid[] = "Wokwi-GUEST";
17 char pass[] = "";
18
19 BlynkTimer timer;
20
21
22
23 void sendSensor()
24 {
25   TempAndHumidity data = dhtSensor.getTempAndHumidity();
26   Serial.println("Temp: " + String(data.temperature, 2) + "°C");
27   Serial.println("Humidity: " + String(data.humidity, 1) + "%");
28   Serial.println("---");
29   Blynk.virtualWrite(V0, data.temperature);
```

# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

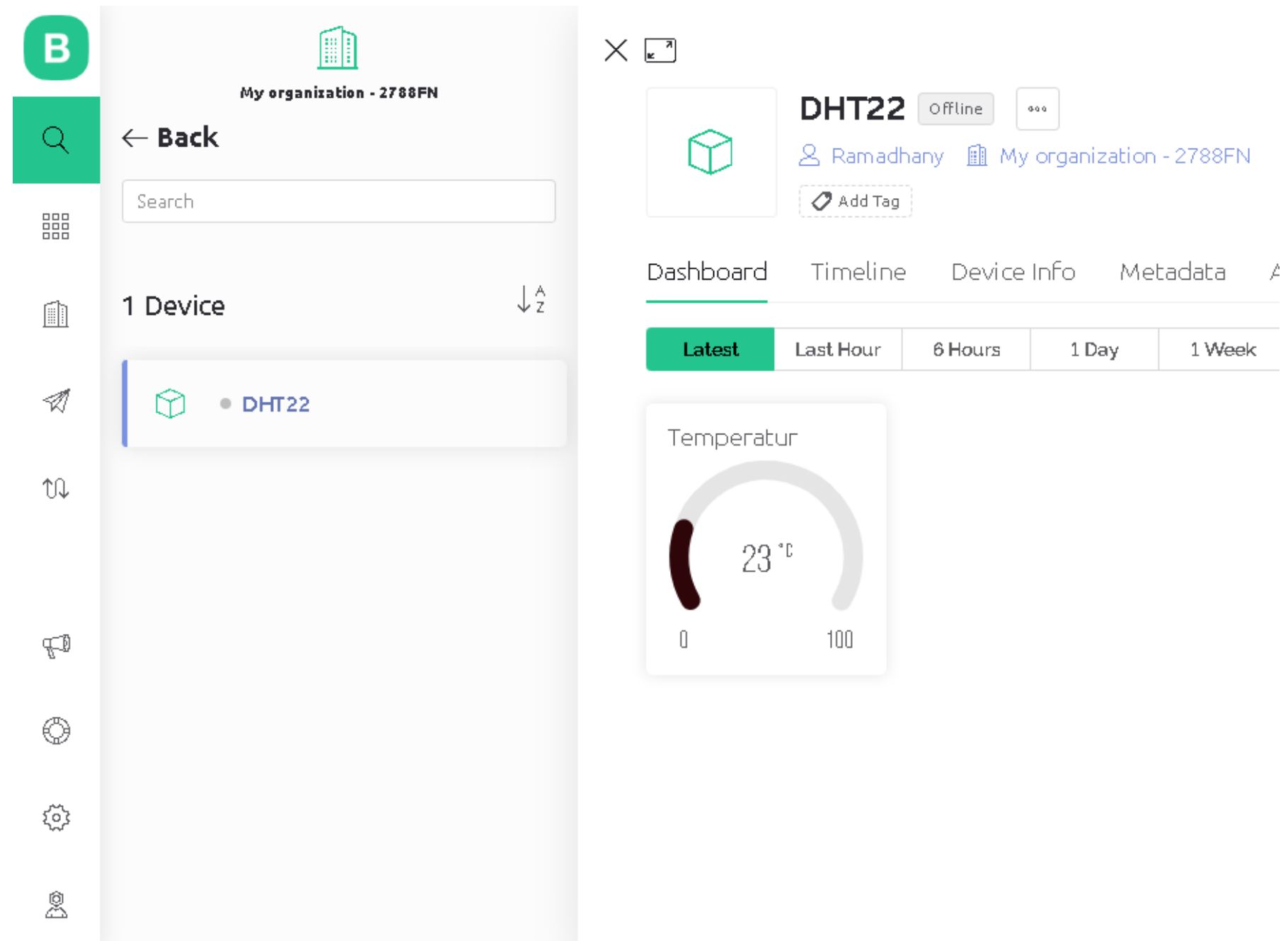
## Memulai Simulasi

- Klik Start Simulation untuk memulai simulasi.
- Tunggu hingga terdapat keterangan Wifi terkoneksi pada Serial Monitor.
- Nilai pembacaan suhu dan kelembaban akan muncul.



# Praktik Monitoring Suhu dan Kelembaban dengan DHT22 Blynk IoT V2

- Memulai Simulasi
- Buka Dashboard pada Blynk.
- Nilai pembacaan suhu akan terbaca pada Gauge yang sudah ditambahkan.
- Ubah nilai suhu DHT22 pada Wokwi, maka nilai pada Gauge juga akan berubah.

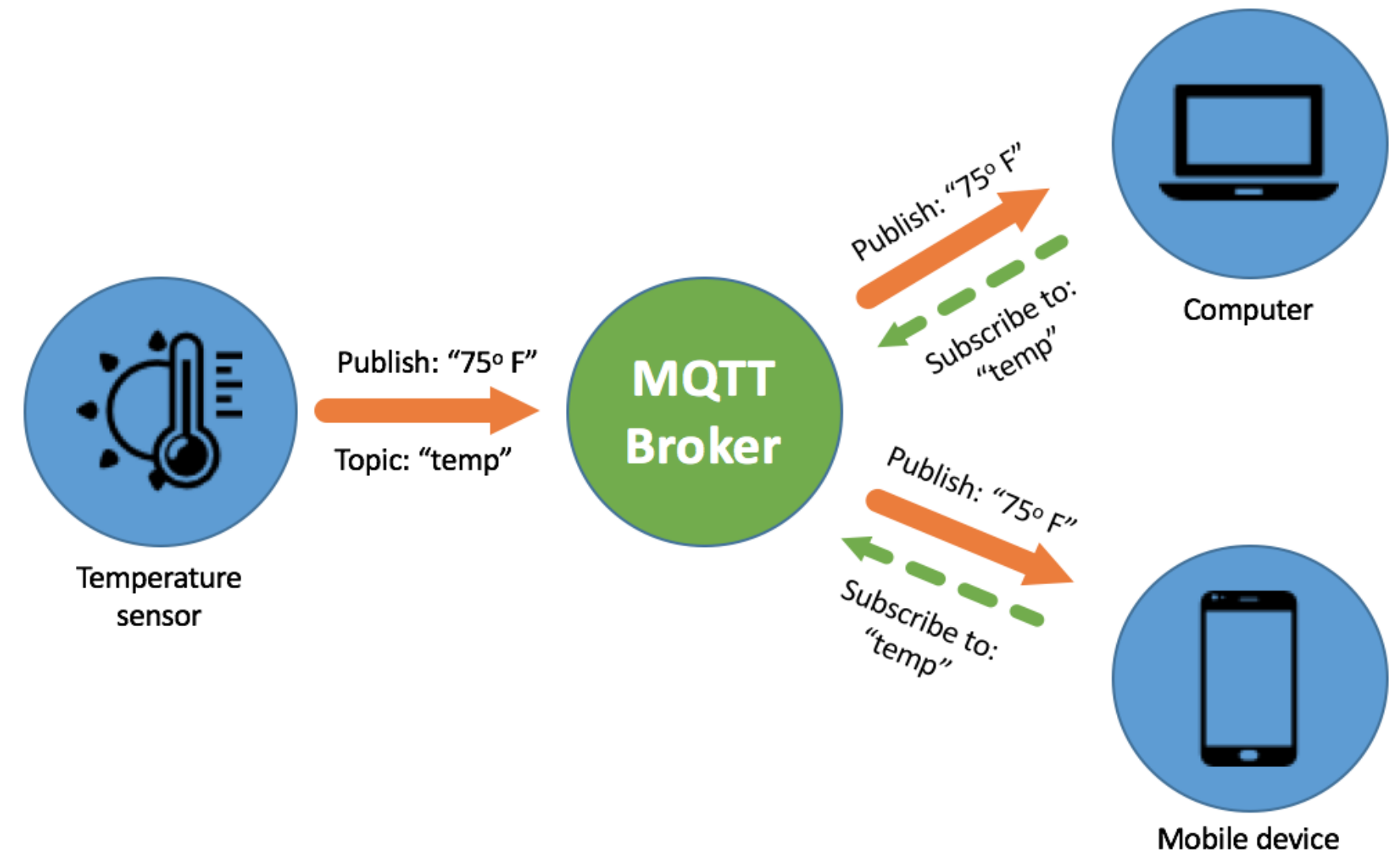


# MQTT

## Pengenalan MQTT

MQTT (Message Queuing Telemetry Transport) adalah protokol komunikasi pengiriman pesan yang dibentuk dengan TCP/IP berdasarkan model messaging publish-subscribe dan dirancang khusus untuk machine to machine yang tidak memiliki alamat khusus.

Publisher mengirim pesan, subscriber menerima pesan yang dibutuhkan, dan broker akan menyampaikan pesan yang diminta, dari pengirim ke penerima. Publisher dan subscriber adalah klien MQTT yang hanya berkomunikasi dengan broker MQTT. Contoh publisher dan subscriber adalah mikrokontroler, komputer, smartphone, dll.





# MQTT

## Broker, Publish, Subscribe dan Topic

### 1. Broker

Broker pada MQTT berfungsi untuk mengorganisir data publish dan subscribe dari berbagai device. Broker bisa diibaratkan sebagai server data yang memiliki alamat IP khusus. Beberapa contoh dari Broker yang ada seperti Mosquitto, HiveMQ, dan Mosca.



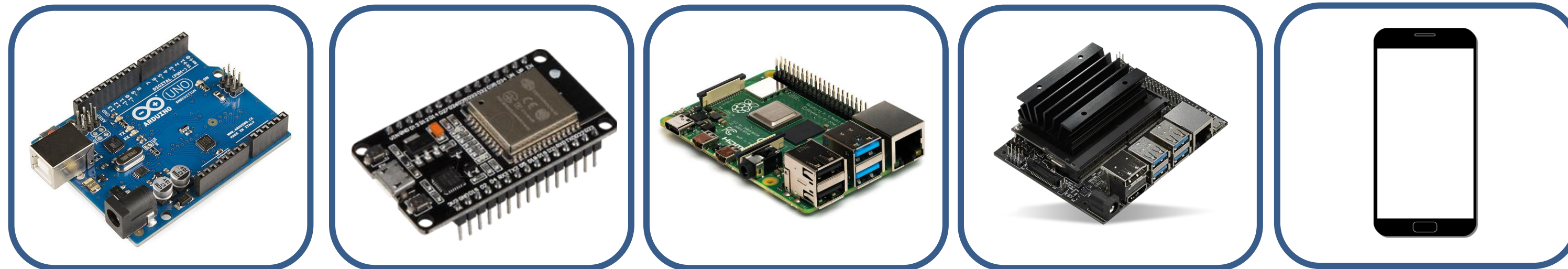
# MQTT

## Broker, Publish, Subscribe dan Topic

### 2. Publish

Publish merupakan cara suatu perangkat untuk mengirimkan datanya ke subscribers. Pengirim data tersebut dinamakan publisher. Biasanya pada publisher adalah sebuah device yang terhubung dengan sensor tertentu.

Contoh publisher adalah Arduino, ESP32, RaspberryPi, Android, dll.



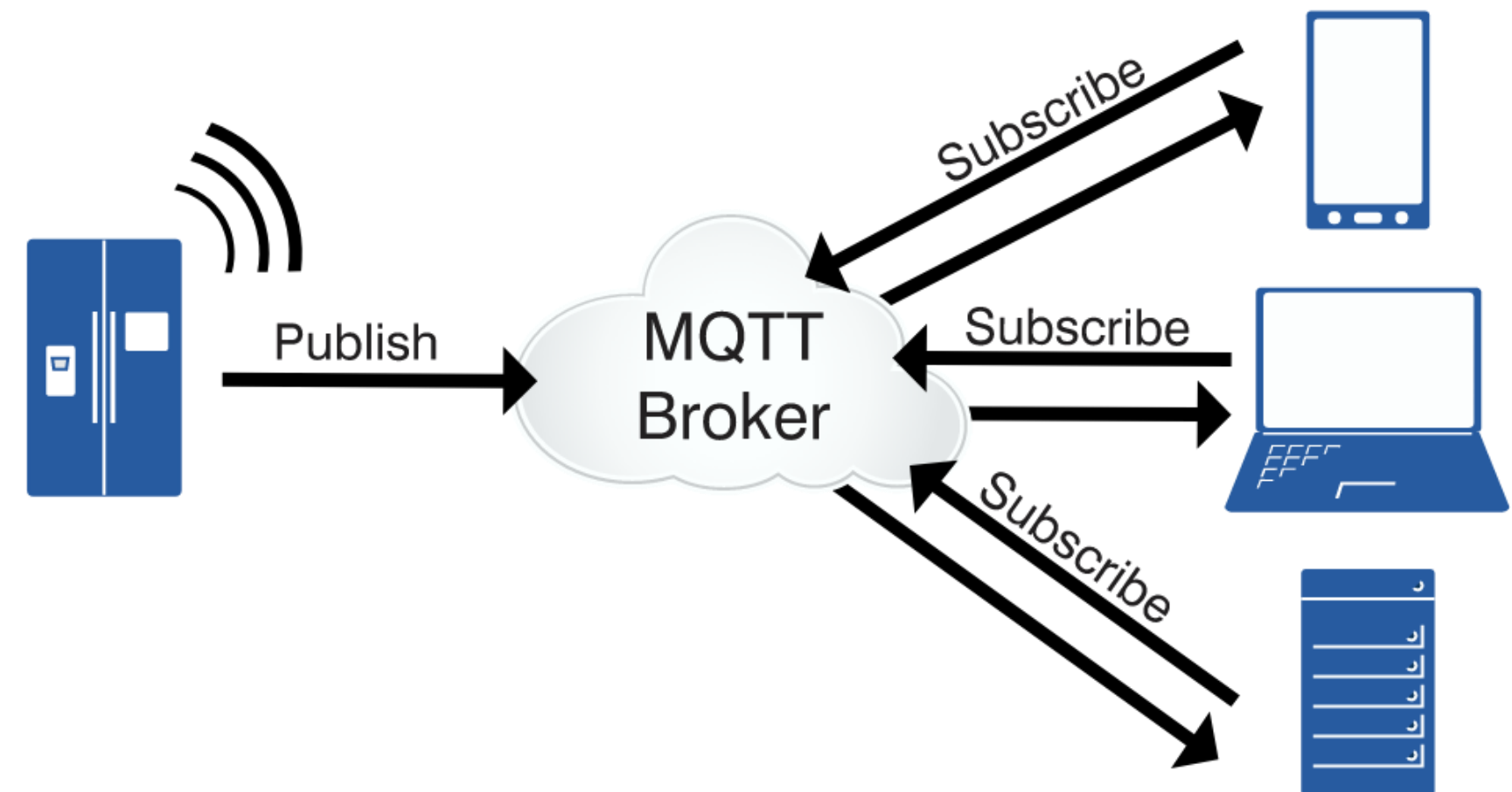
# MQTT

## Broker, Publish, Subscribe dan Topic

### 3. Subscribe

Subscribe merupakan cara suatu device untuk menerima berbagai macam data dari publisher melalui broker. Penerima data disebut subscriber.

Subscriber dapat berupa aplikasi monitoring sensor dan sebagainya, subscriber ini yang nantinya akan meminta data dari publisher melalui broker.



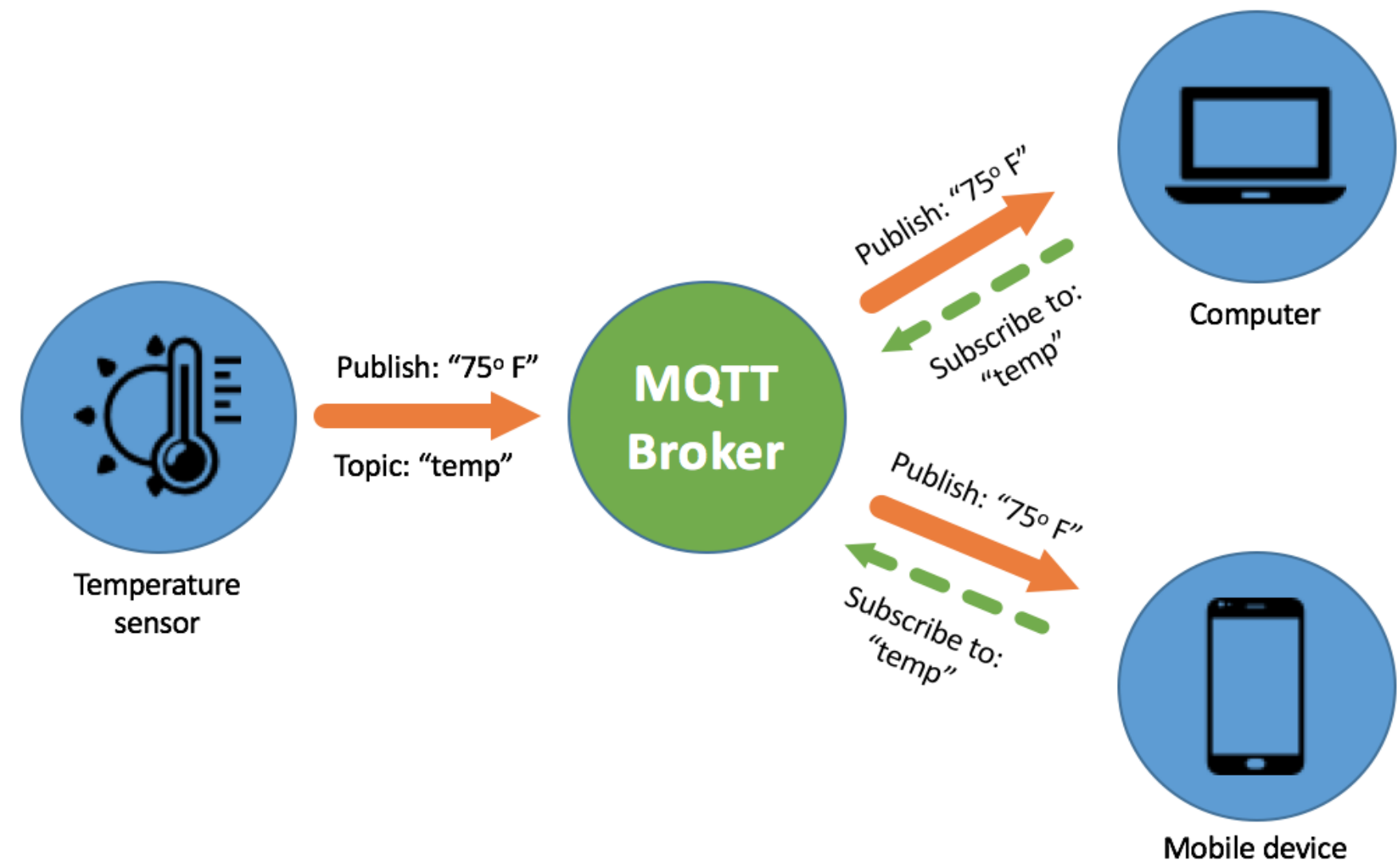
# MQTT

## Broker, Publish, Subscribe dan Topic

### 4. Topic

Topic seperti halnya pengelompokan data disuatu kategori tertentu yang dibuat oleh user. Pada sistem kerja MQTT protokol ini, topic bersifat wajib untuk membedakan data satu dengan yang lainnya.

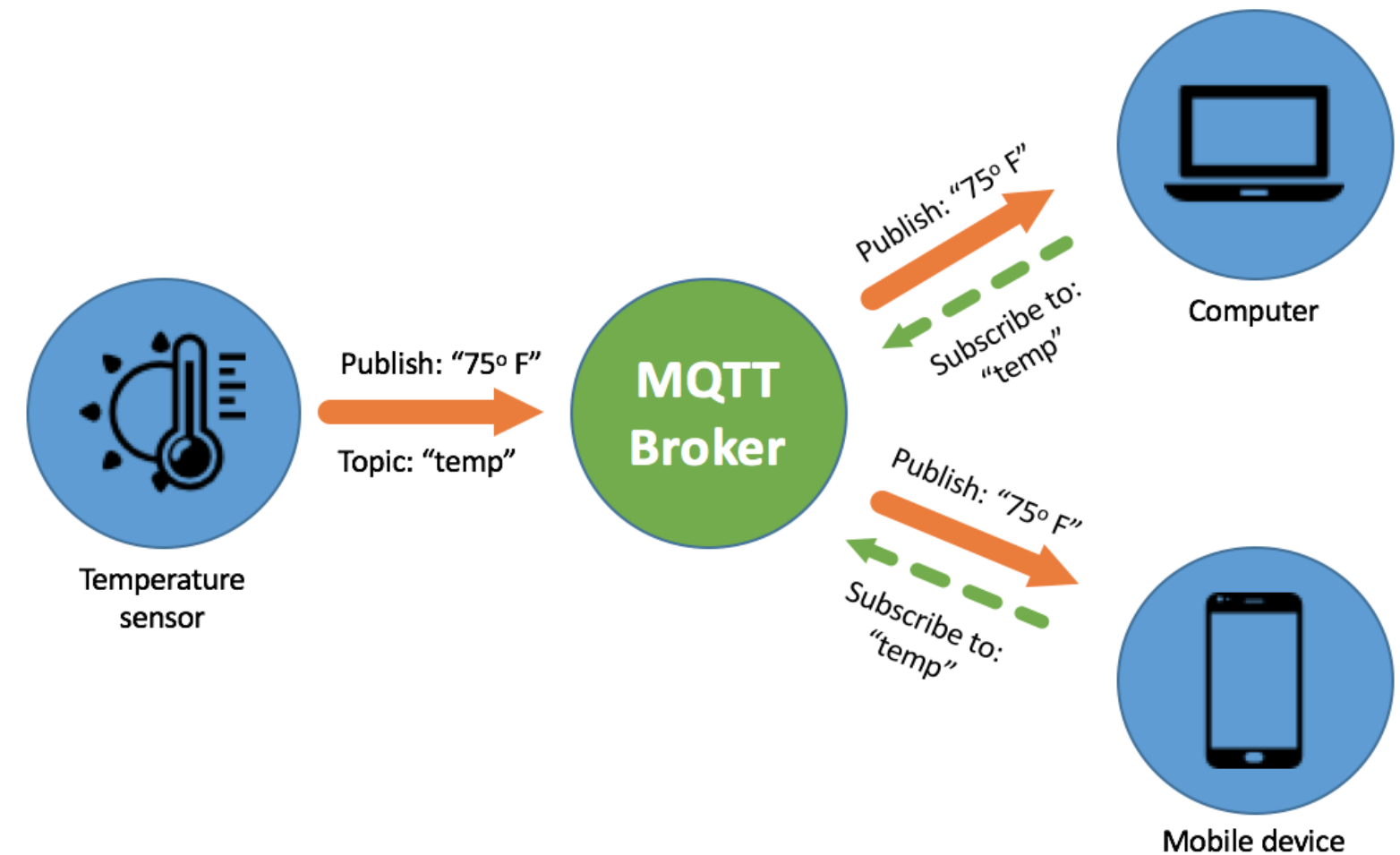
Pada setiap transaksi data antara Publisher dan Subscriber harus memiliki suatu topic tertentu. Contoh penggunaan topic: "temperature", "humidity", dll.



# MQTT

## Cara Kerja MQTT

MQTT sejak awal memang dirancang untuk komunikasi Machine-to-Machine. MQTT mengirim data sebagai byte array, yang mana ini menjadi kelebihan protokol MQTT karena data yang dikirimkan sangat kecil. Pada saat Protokol MQTT di uji dengan jaringan 3G, Protokol MQTT 93 kali lebih cepat dari pada HTTP, karena paket headernya hanya 2 byte. Publisher mengirimkan data/messages dengan topic tertentu kepada broker. Pengiriman data dapat melalui WiFi, GSM, LTE, dll. Subscriber kemudian mengambil data/messages yang diperlukan dengan topik tertentu di broker.



# MQTT

## Mosquitto

Eclipse Mosquitto™ adalah Message broker yang mengimplementasikan protokol MQTT versi 3.1 dan 3.1.1. Broker Mosquitto cukup handal terutama dalam sistem internet of things karena sudah digunakan oleh banyak user.

Mosquitto telah mendukung berbagai sistem operasi mulai dari MacOS, Microsoft Windows, dan berbagai varian distro Linux.





# MQTT

## Keamanan Mosquitto

Mosquitto mendukung keamanan dengan mengenkripsi port yang digunakan dalam transmisi data.

Beberapa port yang dapat digunakan yaitu:

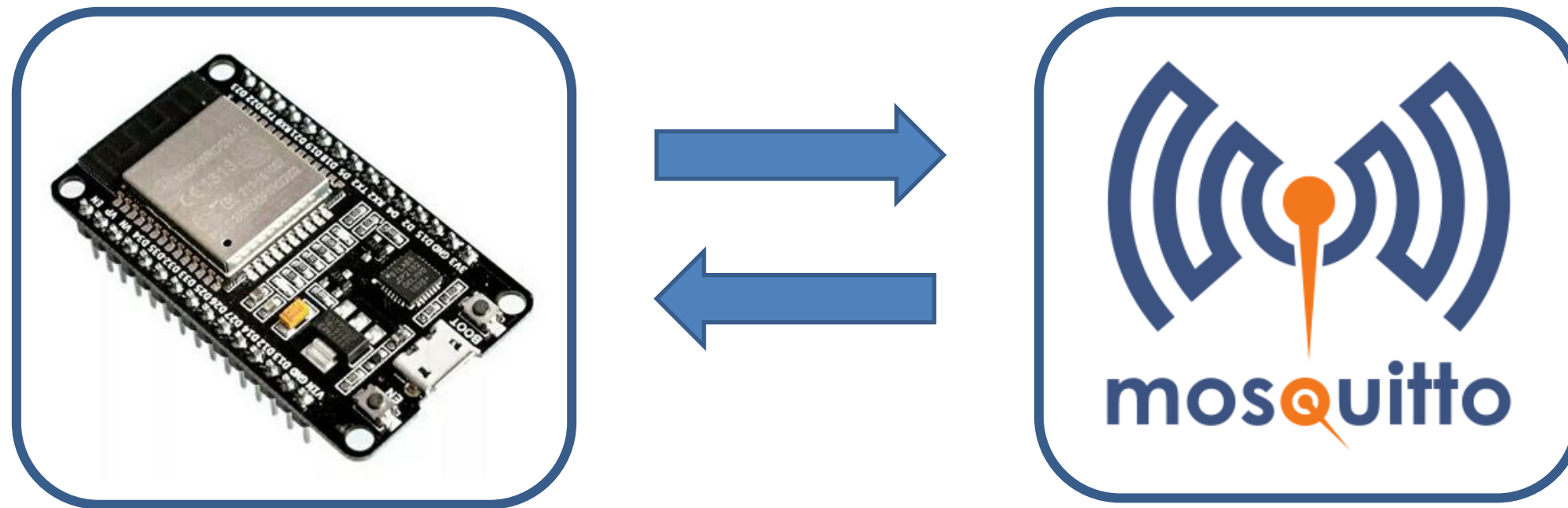
- 1883: MQTT, unencrypted, unauthenticated
- 1884: MQTT, unencrypted, authenticated
- 8883: MQTT, encrypted, unauthenticated
- 8884: MQTT, encrypted, client certificate required
- 8885: MQTT, encrypted, authenticated
- 8886: MQTT, encrypted, unauthenticated
- 8887: MQTT, encrypted, server certificate deliberately expired
- 8080: MQTT over Web Sockets, unencrypted, unauthenticated
- 8081: MQTT over Web Sockets, encrypted, unauthenticated
- 8090: MQTT over Web Sockets, unencrypted, authenticated
- 8091: MQTT over Web Sockets, encrypted, authenticated



# MQTT

## Praktik MQTT dengan Mosquitto

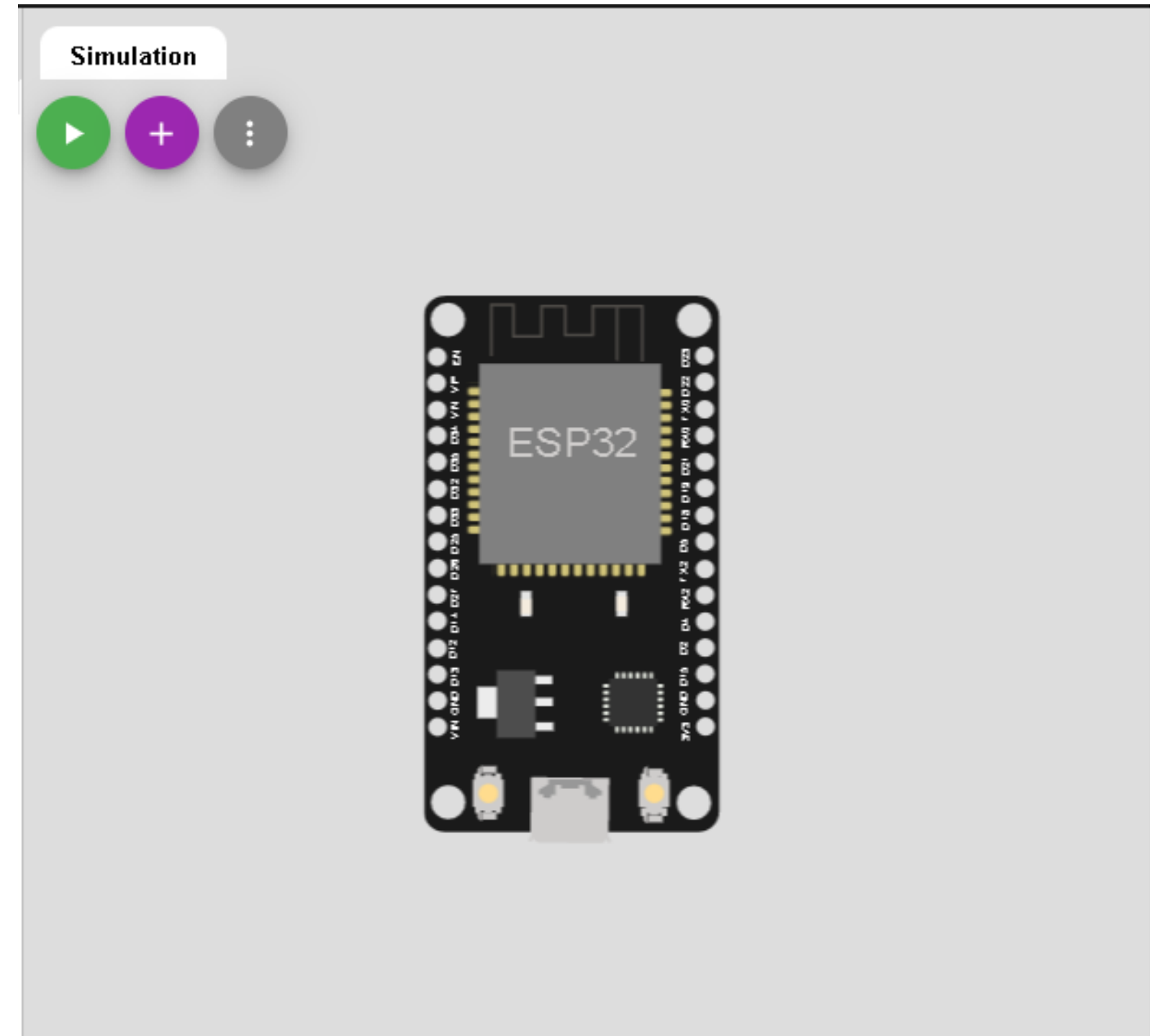
Praktik pengiriman data dari ESP32 ke Mosquitto, kemudian ESP32 tersebut mengambil data dari Mosquitto yang baru saja dikirim.



# MQTT

## Praktik MQTT dengan Mosquitto

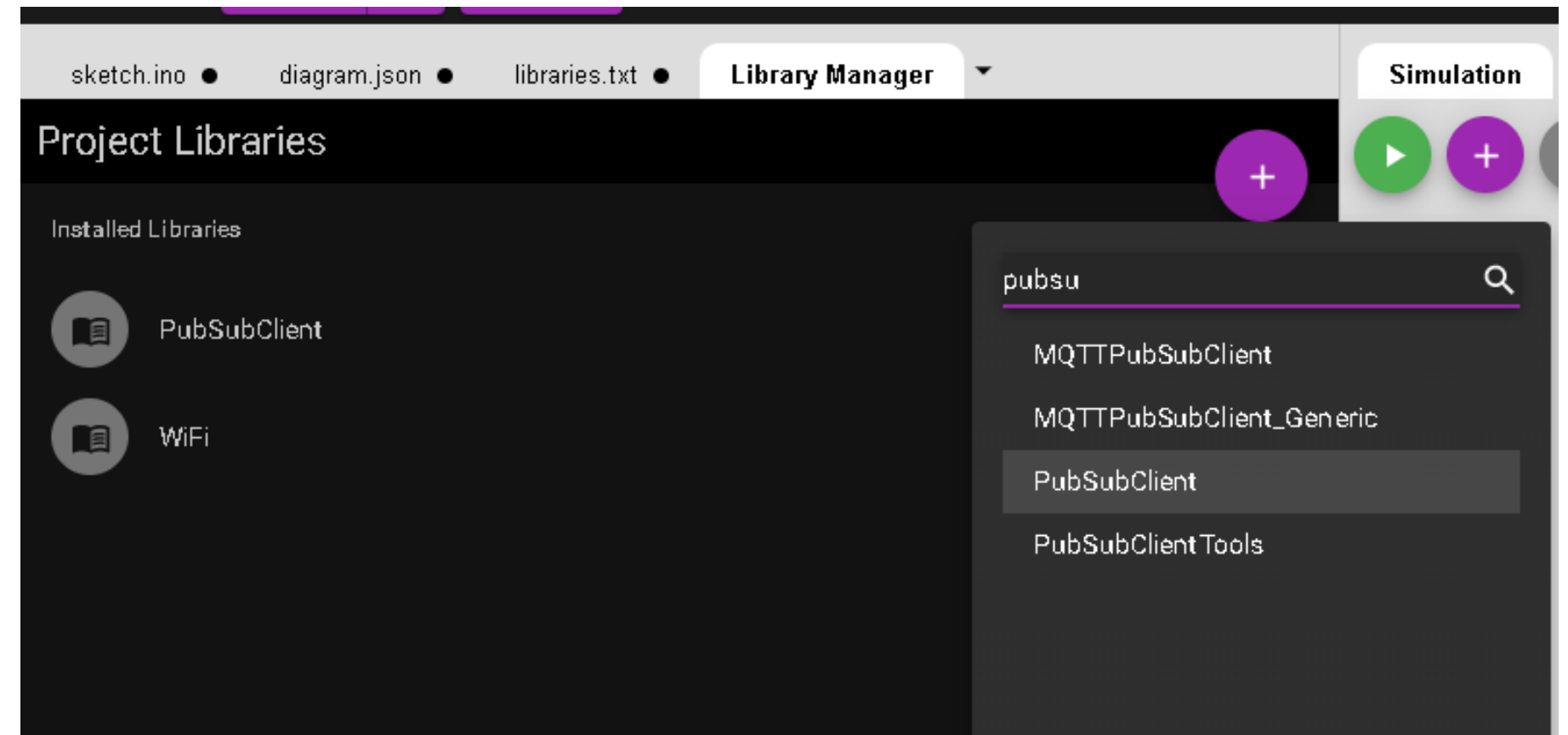
- Buka Wokwi dengan masuk d i<https://wokwi.com>
- Buat projek baru dengan klik Add New Projects
- Pilih ESP32



# MQTT

## Praktik MQTT dengan Mosquitto

- Pada library manager klik add library (+)
- Tambahkan library WiFi dan PubSubClient



# MQTT

## Praktik MQTT dengan Mosquitto

- Buatlah kode program seperti gambar disamping.
- Kode program tersebut digunakan untuk mengirimkan data/messages ke broker Mosquitto sekaligus menerima data yang telah dikirimkan ke broker dengan topic/indobot/mqtt
- Data yang dikirim atau diterima ditampilkan diserial monitor.

Coding : [Download](#)

```
sketch.ino • diagram.json • libraries.txt • Library Manager ▼
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3
4  // Update these with values suitable for your network.
5
6  const char* ssid = "Wokwi-GUEST";
7  const char* password = "";
8  const char* mqtt_server = "test.mosquitto.org";
9
10 WiFiClient espClient;
11 PubSubClient client(espClient);
12 unsigned long lastMsg = 0;
13 #define MSG_BUFFER_SIZE (50)
14 char msg[MSG_BUFFER_SIZE];
15 int value = 0;
16
17 void setup_wifi() {
18
19     delay(10);
20     // We start by connecting to a WiFi network
21     Serial.println();
22     Serial.print("Connecting to ");
23     Serial.println(ssid);
24
25     WiFi.mode(WIFI_STA);
26     WiFi.begin(ssid, password);
27
28     while (WiFi.status() != WL_CONNECTED) {
29         delay(500);
30         Serial.print(".");
31     }
32
33     // ...
34 }
```

# MQTT

## Praktik MQTT dengan Mosquitto

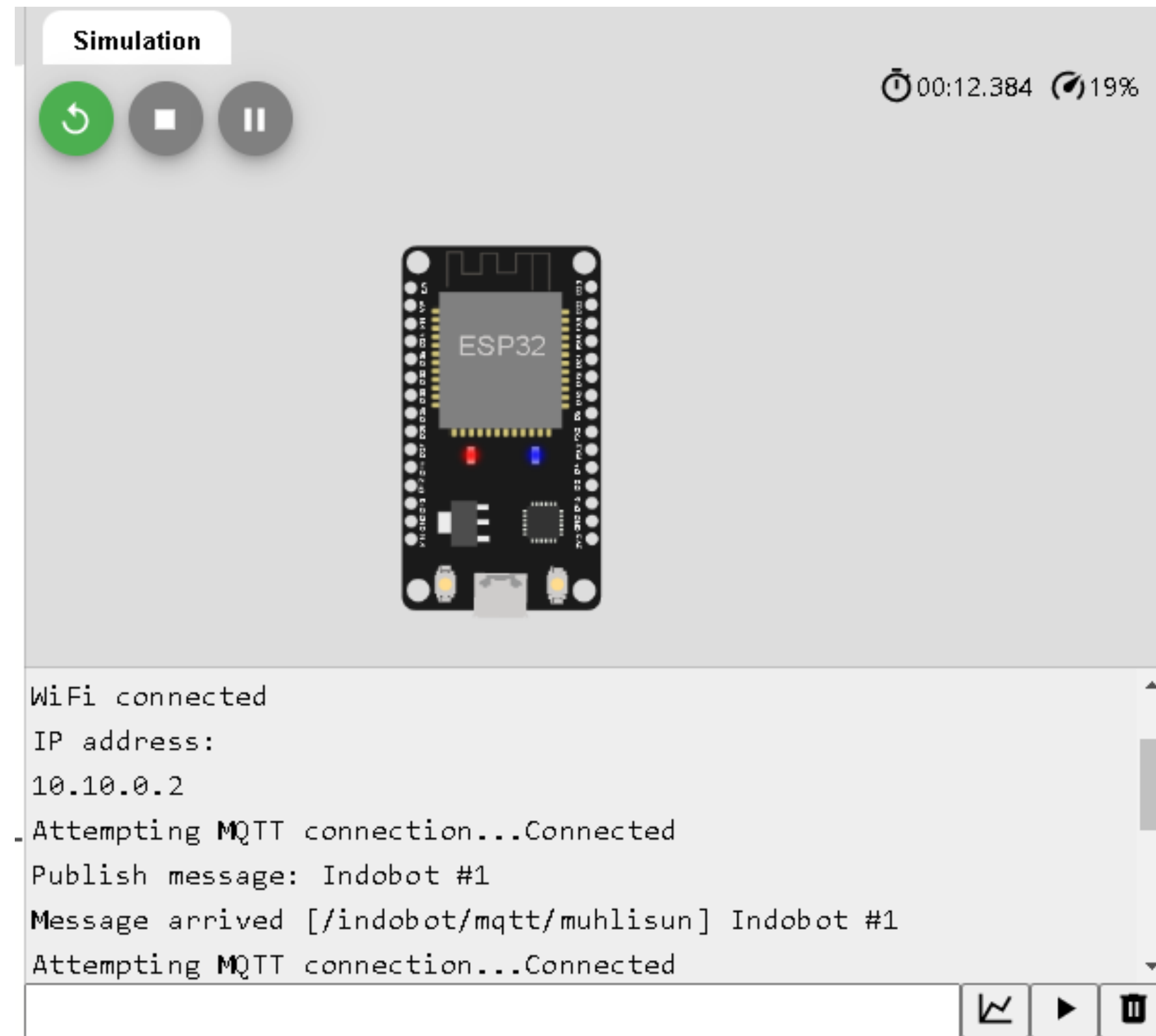
- Pada bagian yang ditandai garis merah, ssid adalah nama jaringan WiFi yang digunakan dan password adalah password yang WiFi yang digunakan dan mqtt\_server diisi "test.mosquitto.org". Karena kita menggunakan simulasi wokwi, ssid dapat diisi seperti gambar disamping

```
sketch.ino • diagram.json • libraries.txt • Library Manager
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3
4  // Update these with values suitable for your network.
5
6  const char* ssid = "wokwi-GUEST";
7  const char* password = "";
8  const char* mqtt_server = "test.mosquitto.org";
9
10 WiFiClient espClient;
11 PubSubClient client(espClient);
12 unsigned long lastMsg = 0;
13 #define MSG_BUFFER_SIZE (50)
14 char msg[MSG_BUFFER_SIZE];
15 int value = 0;
16
17 void setup_wifi() {
18
19   delay(10);
20   // We start by connecting to a WiFi network
21   Serial.println();
22   Serial.print("Connecting to ");
23   Serial.println(ssid);
24
25   WiFi.mode(WIFI_STA);
26   WiFi.begin(ssid, password);
27
28   while (WiFi.status() != WL_CONNECTED) {
29     delay(500);
30     Serial.print(".");
31   }
32
33   ...
34 }
```

# MQTT

## Praktik MQTT dengan Mosquitto

- Jalankan simulasi dengan klik Start Simulation.
- Tunggu sebentar hingga wifi terkoneksi.
- ESP32 akan mengirim data sesuai dengan program ke broker Mosquitto.
- ESP32 juga menerima data dari Mosquitto sesuai dengan topic.
- Klik Stop Simulation untuk menghentikan simulasi.



# MQTT

## Praktik MQTT Komunikasi antar ESP32

Praktik pengiriman data dari ESP32 ke Mosquitto, kemudian ESP32 yang lain mengambil data dari Mosquitto.

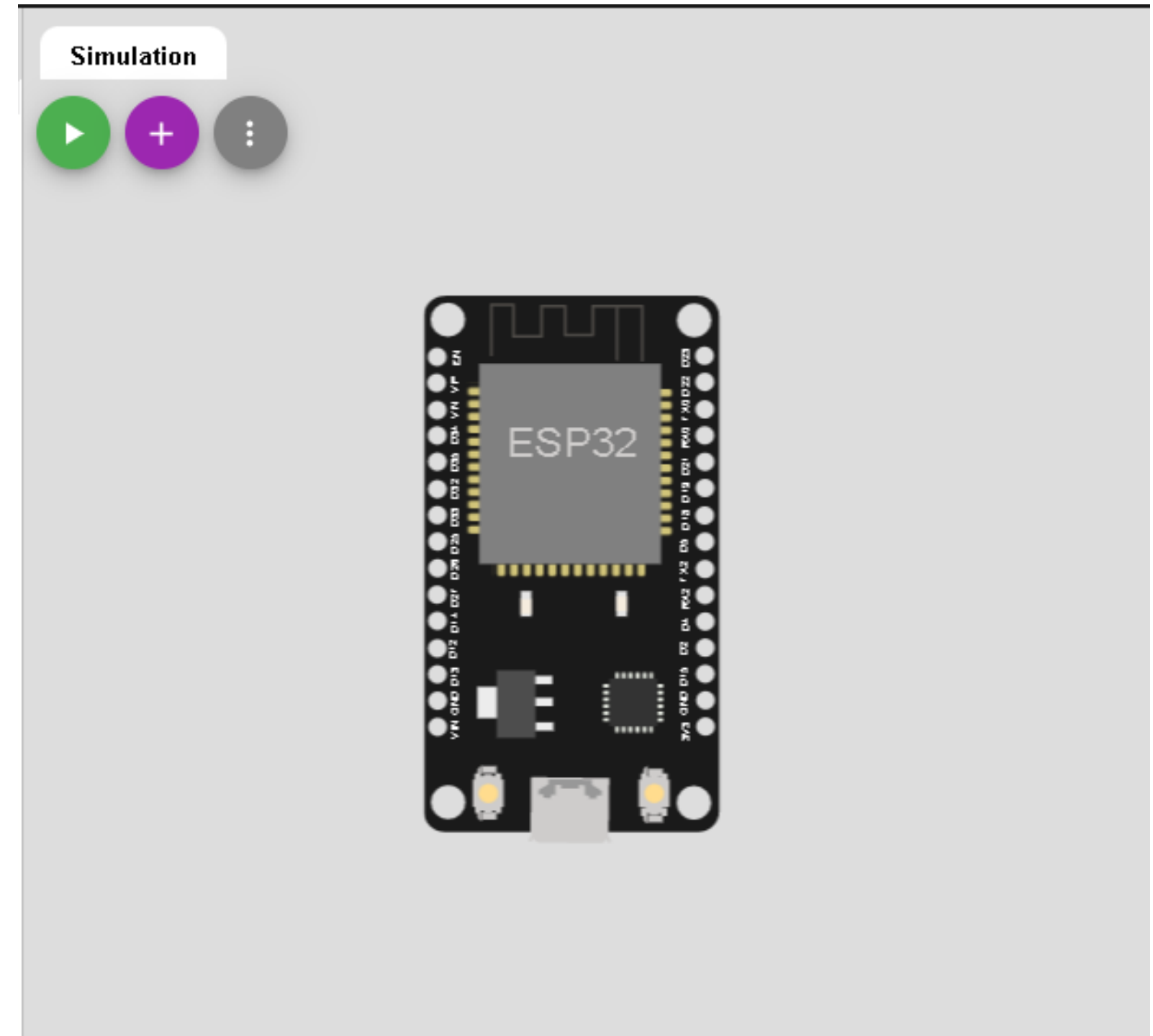




# MQTT

## Praktik MQTT Komunikasi antar ESP32

- Buka Wokwi dengan masuk di <https://wokwi.com>
- Buat proyek baru dengan klik Add New Projects
- Pilih ESP32 yang digunakan sebagai publisher.
- Buka jendela baru dan buka Wokwi lagi lalu tambahkan ESP32 yang digunakan sebagai subscriber.
- Sehingga total terdapat 2tab Wokwi.



# MQTT

## Praktik MQTT Komunikasi antar ESP32

- Buatlah kode program seperti gambar disamping pada ESP32 yang berfungsi sebagai publisher.
- Kode program tersebut digunakan untuk mengirimkan data/messages ke broker Mosquitto sekaligus menerima data yang telah dikirimkan ke broker dengan topic/indobot/mqtt.

Coding : [Download](#)

```
publisher.ino  diagram.json  libraries.txt  Library Manager
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3
4  // Update these with values suitable for your network.
5
6  const char* ssid = "Wokwi-GUEST";
7  const char* password = "";
8  const char* mqtt_server = "test.mosquitto.org";
9
10 WiFiClient espClient;
11 PubSubClient client(espClient);
12 unsigned long lastMsg = 0;
13 #define MSG_BUFFER_SIZE (50)
14 char msg[MSG_BUFFER_SIZE];
15 int value = 0;
16
17 void setup_wifi() {
18
19     delay(10);
20     // We start by connecting to a WiFi network
21     Serial.println();
22     Serial.print("Connecting to ");
23     Serial.println(ssid);
24
25     WiFi.mode(WIFI_STA);
26     WiFi.begin(ssid, password);
27
28     while (WiFi.status() != WL_CONNECTED) {
29         delay(500);
30         Serial.print(".");
31     }
32
33     ...
34 }
```

# MQTT

## Praktik MQTT Komunikasi antar ESP32

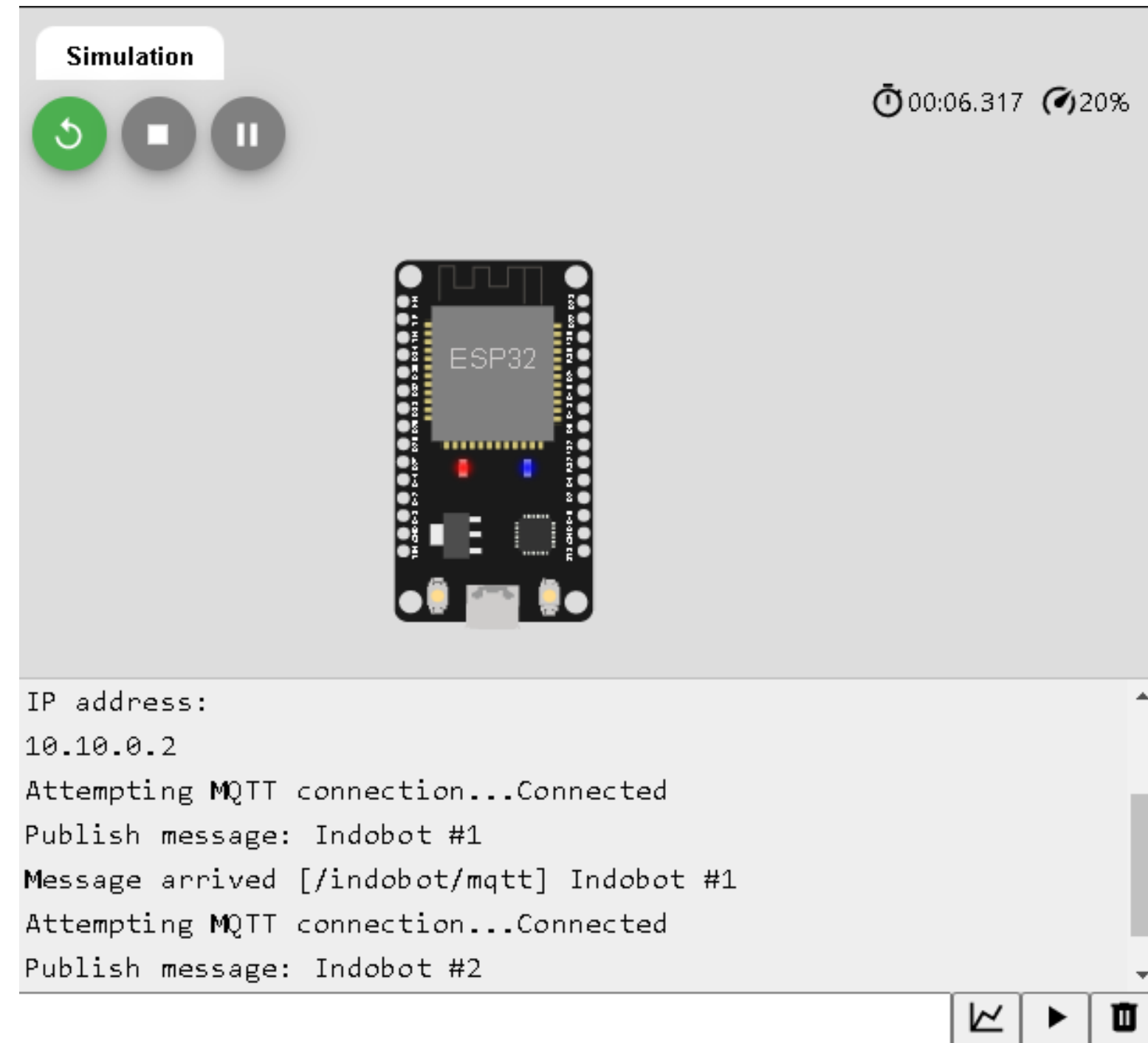
- Buatlah kode program seperti gambar disamping untuk ESP32 yang berfungsi sebagai subscriber.
- Kode program tersebut digunakan untuk menerima data yang telah dikirimkan oleh publisher ke broker dengan topic/indobot/mqtt.

```
subscriber.ino  diagram.json  libraries.txt  Library Manager  ▼
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3
4  // Update these with values suitable for your network.
5
6  const char* ssid = "Wokwi-GUEST";
7  const char* password = "";
8  const char* mqtt_server = "test.mosquitto.org";
9
10 WiFiClient espClient;
11 PubSubClient client(espClient);
12 unsigned long lastMsg = 0;
13 #define MSG_BUFFER_SIZE (50)
14 char msg[MSG_BUFFER_SIZE];
15 int value = 0;
16
17 void setup_wifi() {
18
19     delay(10);
20     // We start by connecting to a WiFi network
21     Serial.println();
22     Serial.print("Connecting to ");
23     Serial.println(ssid);
24
25     WiFi.mode(WIFI_STA);
26     WiFi.begin(ssid, password);
27
28     while (WiFi.status() != WL_CONNECTED) {
29         delay(500);
30         Serial.print(".");
31     }
32
33     ...
34 }
```

# MQTT

## Praktik MQTT Komunikasi antar ESP32

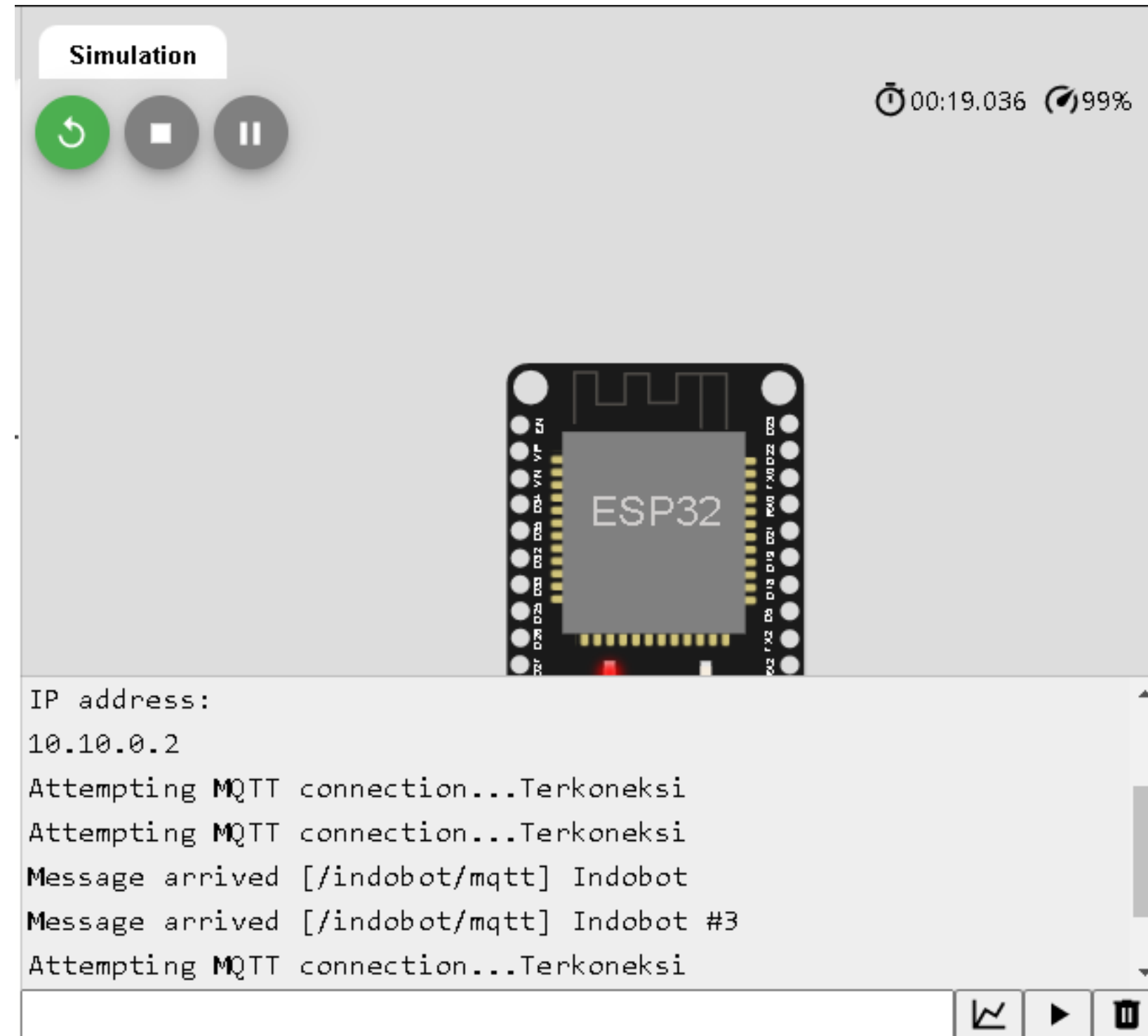
- Buka tab ESP32 publisher, lalu jalankan simulasi.
- Buka tab ESP32 subscriber, lalu jalankan simulasi.
- Buka tab ESP32 publisher.
- Tunggu sebentar hingga Wifi terkoneksi dan mulai mengirimkan data.
- Saat ESP32 sudah mengirimkan data, buka tab ESP32 subscriber dan pastikan data sudah diterima sesuai dengan yang dikirimkan oleh ESP32 publisher.



# MQTT

## Praktik MQTT Komunikasi antar ESP32

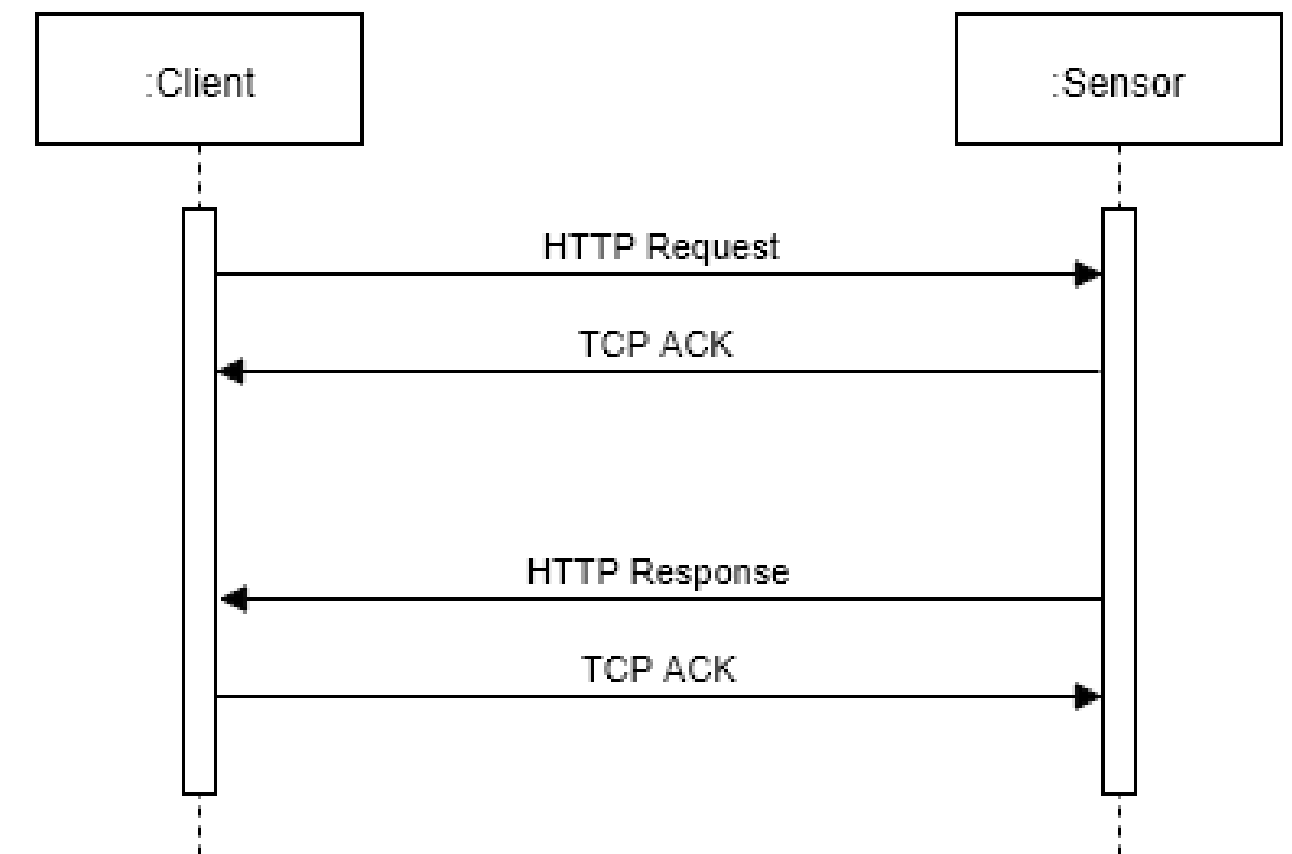
- Buka tab ESP32 subscriber, akan muncul data yang dikirimkan oleh ESP32 Publisher. Jika sudah muncul, berarti ESP32 Subscriber berhasil menerima data.
- Data ditampilkan di serial monitor.
- Klik Stop Simulation untuk menghentikan simulasi.



# CoAP

## Pengenalan CoAP

- CoAP merupakan suatu protokol yang didesain untuk IoT.
- CoAP merupakan singkatan dari Constrained Application Protocol yang dibuat oleh Internet Engineering Task Force (IETF)
- Hampir sama dengan HTTP
- RESTful (Representational State Transfer): Client/server structur,stateless:
  - Get: Membaca informasi
  - PUT: Memperbaharui informasi
  - POST : Membuat informasi baru
  - DELETE : menghapus informasi
- CoAP ditujukan pada perangkat yang memiliki keterbatasan dalam energi, RAM, bandwidth data, memori, hingga catu daya.





# CoAP

## Pengenalan CoAP

CoAP adalah alternatif protokol yang bisa digunakan disamping HTTP yang lebih rakus terhadap Resource. CoAP sendiri, merupakan standar dari Internet Engineering Task Force (IETF) dan ditetapkan pada RFC7252 <https://tools.ietf.org/html/rfc7252>.

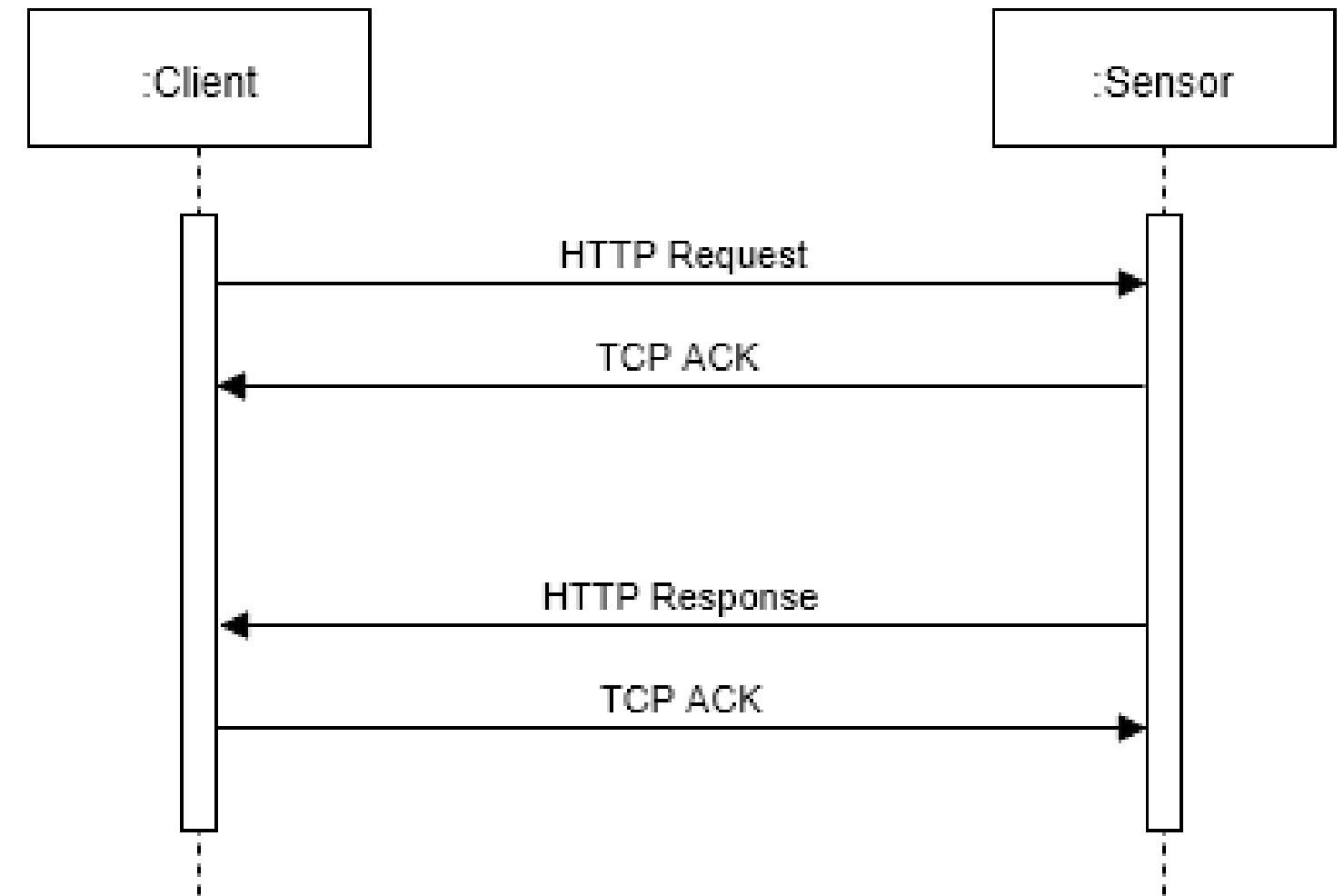
CoAP melakukan optimasi secara drastis, ia mengubah dua hal. Pertama, CoAP berada diatas UDP bukan TCP. Di UDP, paket -paket IP disebut sebagai datagram, tidak mengenal *acknowledge* atau urutan sebagaimana TCP. Hal ini berarti datagram bisa hilang atau diterima dengan urutan yang berbeda dari yang seharusnya saat dia dikirim. Pertimbangan semacam ini yang harus dipikirkan oleh implementasi CoAP.

Application layer	HTTP, CoAP, EBHTTP, LTP, SNMP, IPfix, DNS, NTP, SSH, DLMS, COSEM, DNP, MODBUS
Network/Communication layer	IPv6/IPv4, RPL, TCP/UDP, uIP, SLIP, 6LoWPAN,
PHY/MAC layer	IEEE 802.11 Series, 802.15 Series, 802.3, 802.16, WirelessHART, Z-WAVE, UWB, IrDA, PLC, LonWorks, KNX

Application (CoAP, XML)
Security ( <b>DTLS</b> )
Transport (UDP)
Network (IPv6)
PHY/MAC (IEEE 802.15.4)

## Tipe Pesan

- **Confirmable (CON):** Pesan yang berisi request dan memerlukan Acknowledgment
- **Non Confirmable(NON):** Pesan yang digunakan berulang secara teratur tanpa memerlukan acknowledgement
- **Acknowledgement(ACK):** Pesan yang berisi response
- **Reset(RST):** Pesan yang digunakan ketika pesan CON tidak diterima dengan benar atau terdapat konteks yang hilang





# CoAP

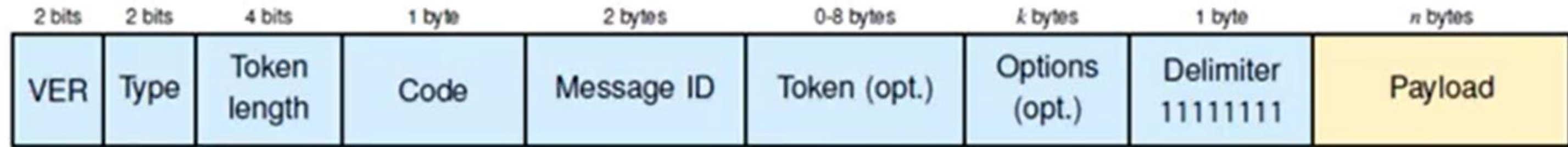
## CoAP Frame Structure

CoAP memiliki 2 bits core version, kemudian 2 bits pesan yang dikirimkan, 4 bits dari panjang tokenya, namun token length akan bernilai 0 apabila kita tidak menggunakannya, namun token akan sangat berguna untuk mengidentifikasi komunikasi request response. Kemudian akan mengirimkan code, pesan, token, dan seterusnya

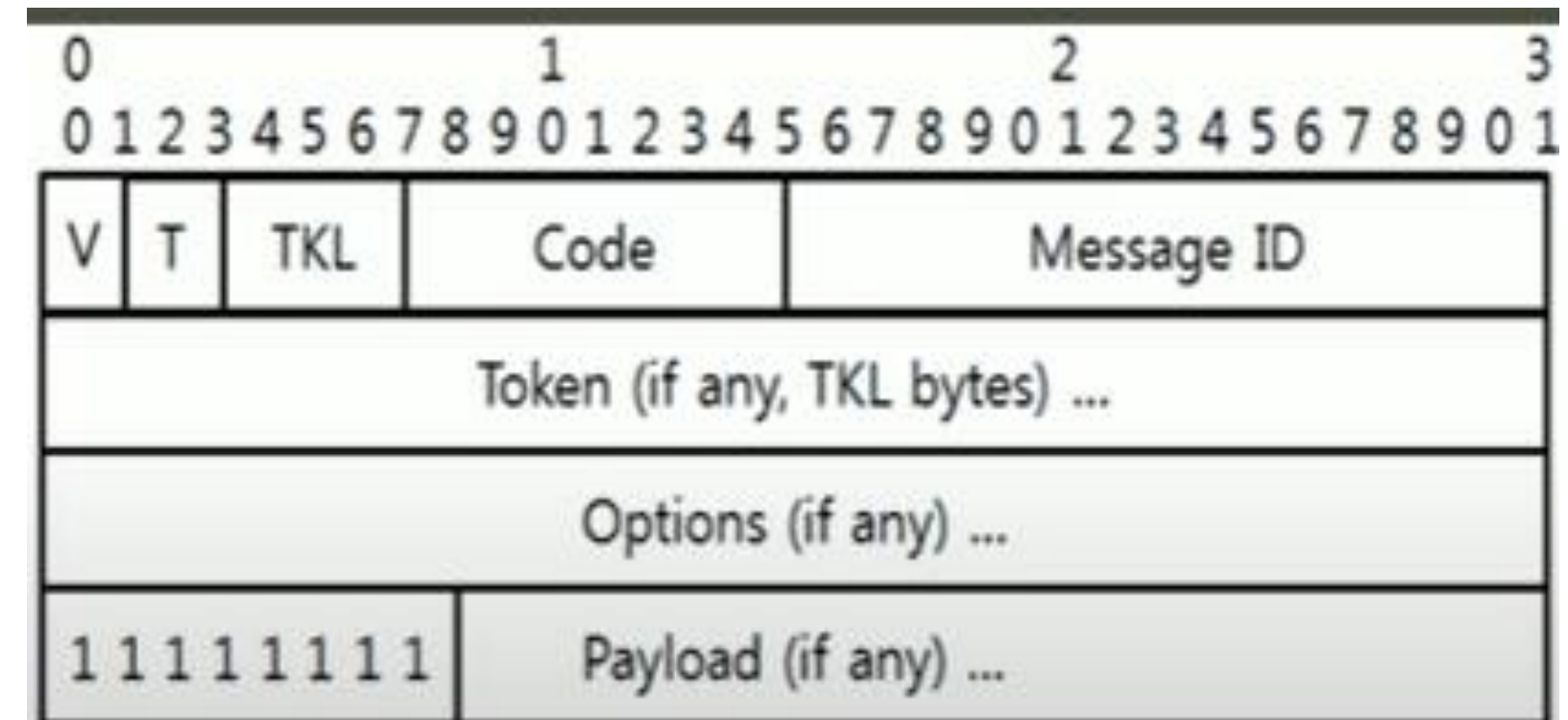


# CoAP

## CoAP Frame Structure



- **Ver** : 2 bit unsigned integer. Menyebutkan versi CoAP
- **Type**: 2 bit unsigned integer. Mengindikasikan pesan (confirmable ()), non confirmable (1), ACK(2), or Reset(3)
- **Token Length**: 4 bit unsigned integer, mengindikasikan panjang dari tokennya
- **Code**: response dari suatu kodenya
- **Message ID**: Mengidentifikasi dari setiap pesan yang dikirimkan

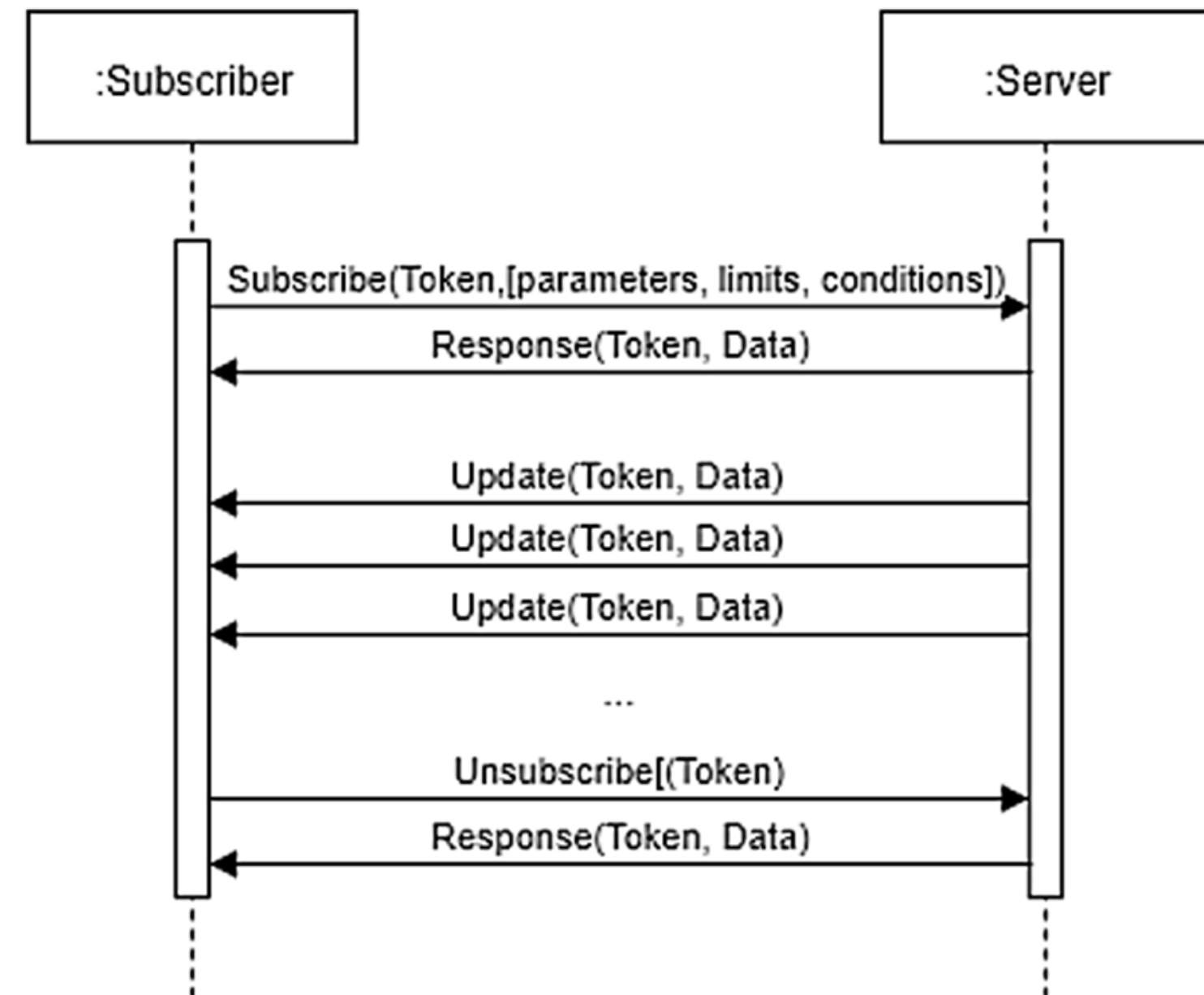


# CoAP

## Distribusi Data yang Efisien

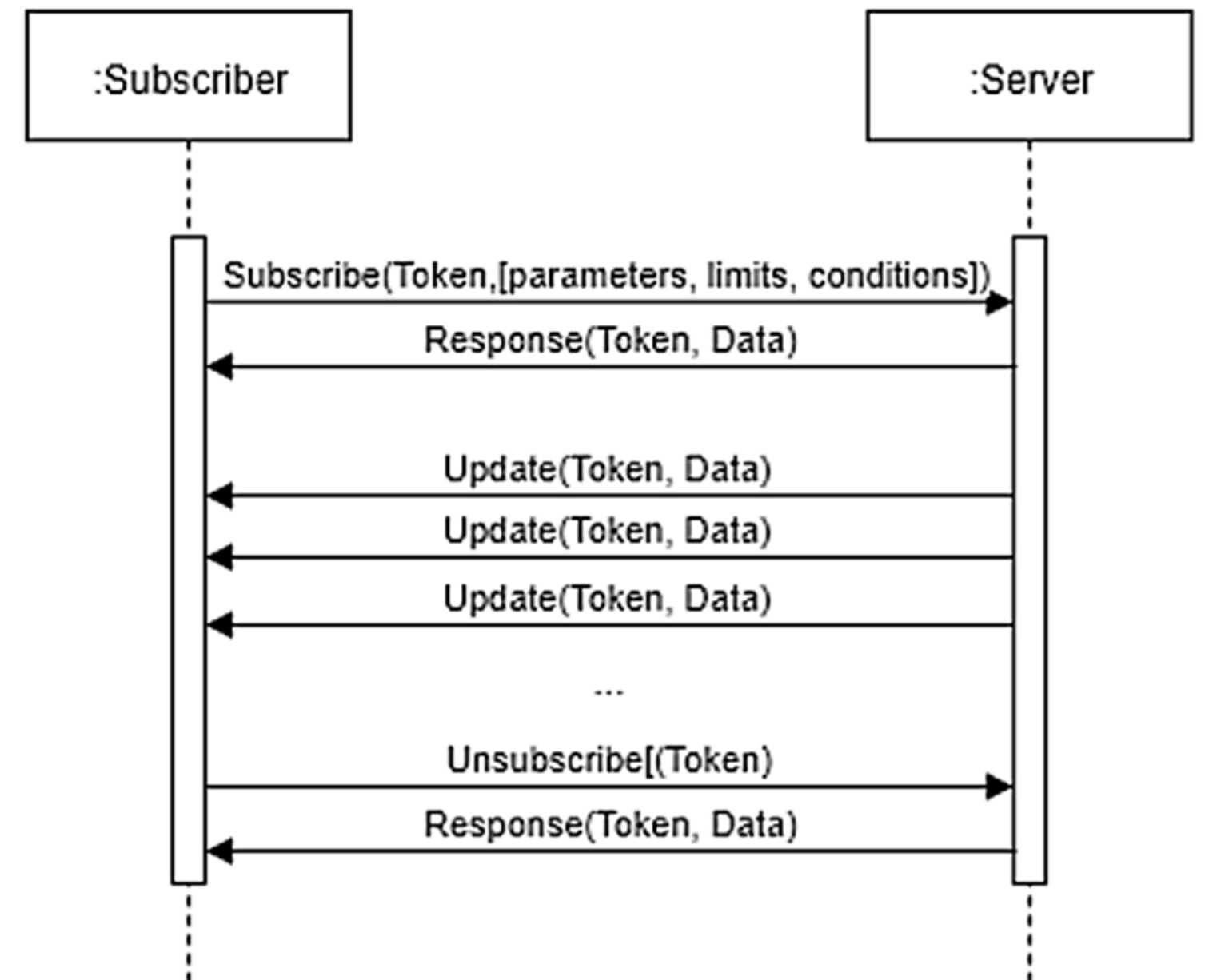
CoAP juga memperkenalkan pola komunikasi baru yang disebut sebagai **Multicast** dan **Event / Observe**. Karena CoAP berdasarkan pada UDP dan UDP mendukung multicasting, menggunakan Internet Group Management Protocol atau IGMP, pesan-pesan CoAP bisa juga menggunakan multicast, setidaknya ketika enkripsi tidak digunakan.

Ini bisa dilakukan dengan mengirim pesan CoAP ke alamat multicast yang telah ditentukan sebelumnya



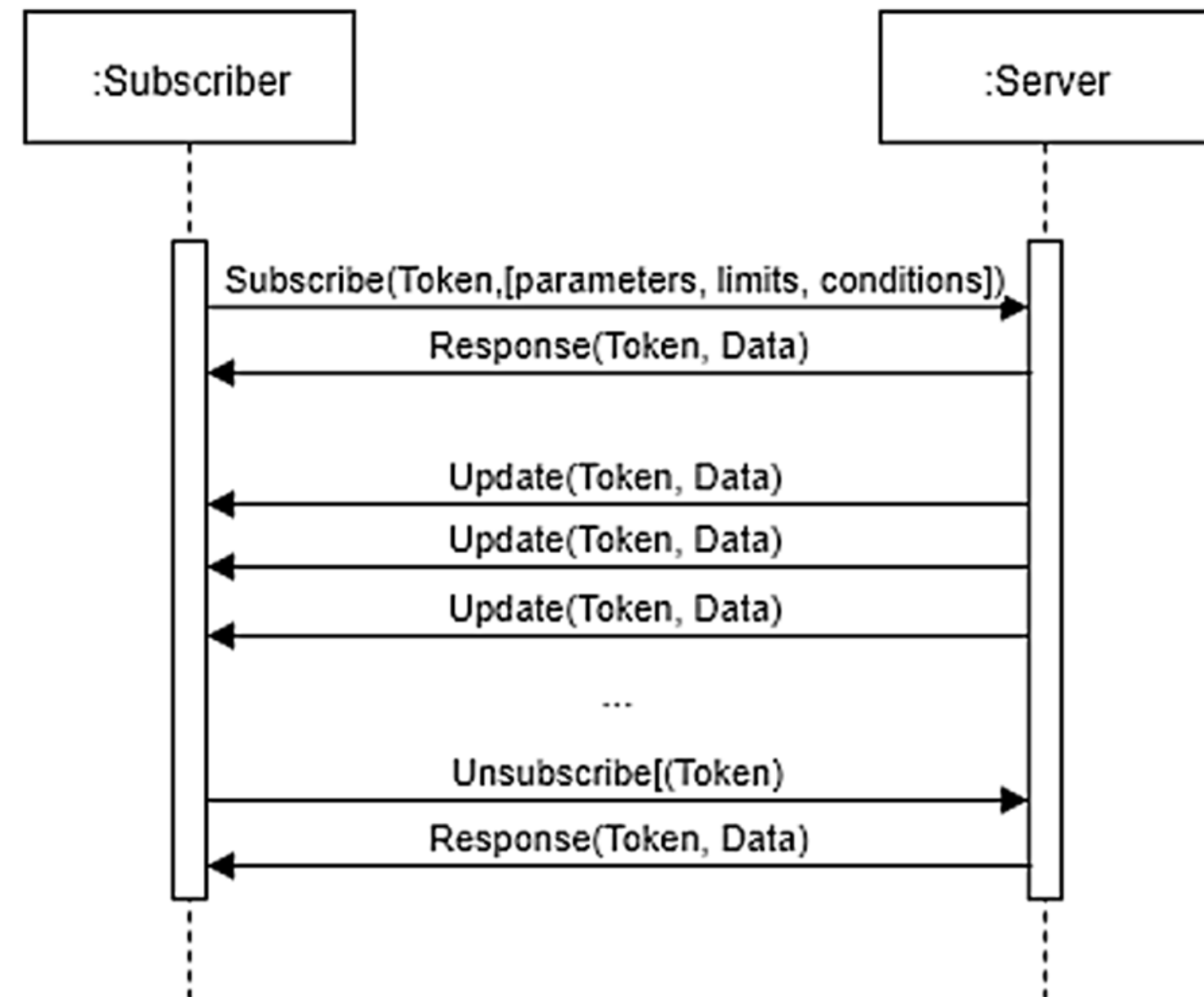
## Observe Pattern

- **Pattern observe (event subscription)** berfungsi untuk sensor yang melaporkan data secara rutin dan aman, optimisasi yang terbesar, dengan menggunakan patern ini.
- **Subscription** merupakan pihak yang melakukan request (requestor) atau subscriber hanya perlu mengirim permintaan sekali.
- Sama seperti request pada biasanya, perangkat akan mengembalikan respon yang berisi data yang ingin dikembalikan. Perbedaannya terletak pada resource yang berubah. Ketika hal ini terjadi perangkat akan otomatis mengirim respon baru ke subscriber tanpa harus menunggu request yang sesuai.



## Observe Pattern

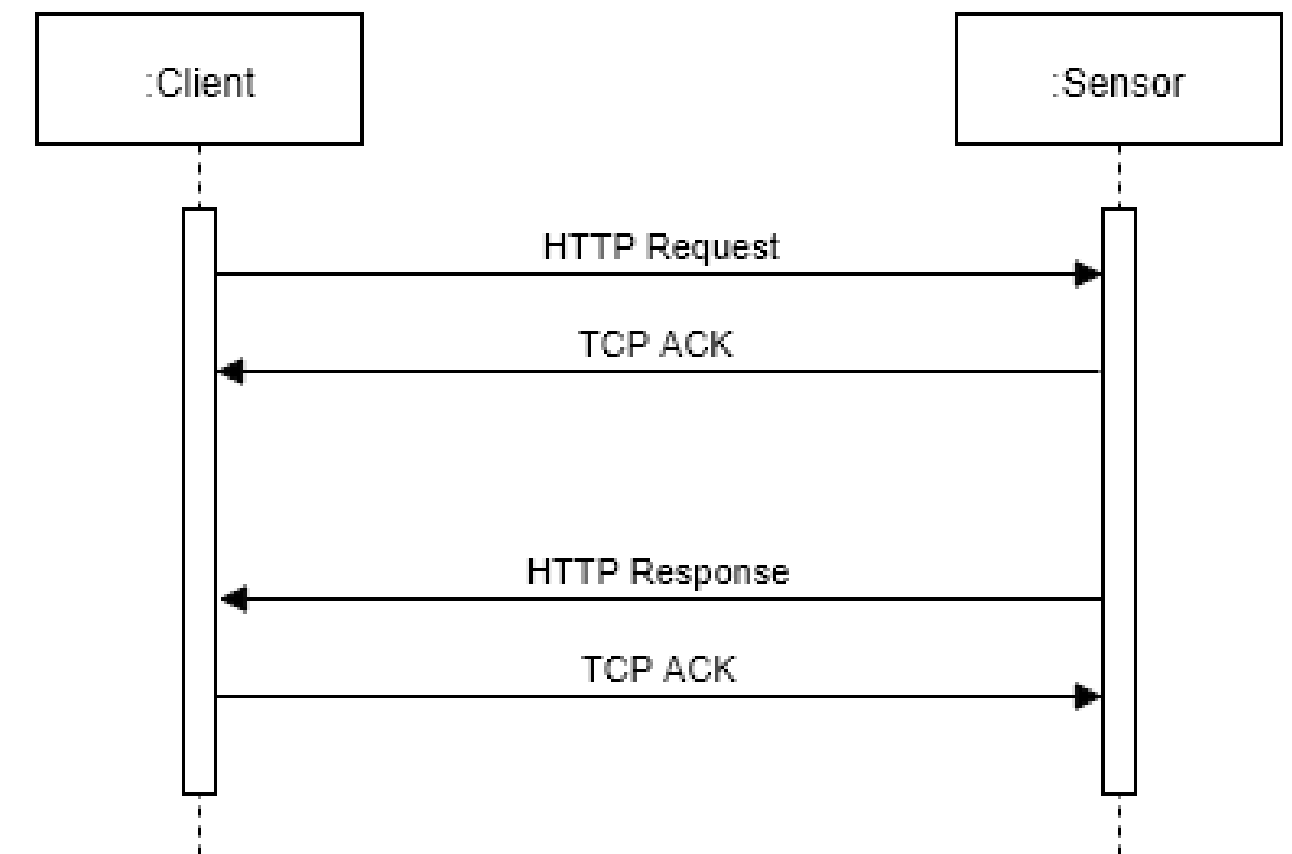
- Pada situasi yang normal, hal ini akan mengurangi jumlah pesan yang dikirim dengan faktor (pendekatan) dua.
- Dari empat pesan, dapat dikurangi menjadi satu bergantung pada arah komunikasi.
- Jika data dikirim dari perangkat ke penerima, maka jumlah pesan yang dikirim dapat dipangkas hingga 4 pada kasus normal tanpa acknowledge.
- Jika data dikumpulkan oleh client, jumlah pesan yang diterima bergantung pada seberapa sering pengumpulan dilakukan



# CoAP

## Kelebihan CoAP

- Pengoperasannya tidak membutuhkan daya yang besar
- CoAP digunakan untuk perangkat yang dengan energi rendah seperti sensor mikrokontroler, switch dan komponen lainnya.
- Protokol komunikasi yang lebih ringan daripada protokol HTTP.
- CoAP memiliki fitur yang menyerupai HTTP yang menyediakan model interaksi request/response antara aplikasi dan endpoint.





**Sekian Materi**

# **Pengenalan Macam-macam Koneksi Protokol IoT seperti HTTP, MQTT, CoAP dan Praktikum Platform Blynk IoT V2**

**Digitalent Scholarship Professional Academy**