Workshop Aplikasi Mikroprosesor & Antarmuka

PROGRAM STUDI
TEKNIK TELKOMUNIKASI

Akuwan Saleh, MT

PENILAIAN

- Laporan + Tugas + Presentasi = 60%
  - eval-1(Lap.1-5) = 20%
  - eval-2(Lap.6-10) = 20%
  - eval-3(Lap.11-15 & (PPT+ presentasi)) = 20%
- TPS = 40%
  - eval-4 (TPS = Tugas Proyek Semester)

REFERENSI

- Rui Santos & Sara Santos, “ESP32 Web Server With Arduino IDE.pdf: Step By Step Project Guide”, https://randomnerdtutorials.com/esp32-web-server-arduino-ide/, juni 2020.
- Julien Bayle, “C Programming for Arduino”, Packt Publishing Ltd, Birmingham, May 2013
- Fabian Winkler, “Arduino/Processing Communication Workshop”, Fall, 2013.
- Jack Purdum, “Beginning C for Arduino, Learn C Programming for the Arduino and Compatible Microcontrollers”, Apress, 2012.
- John-David Warren, Josh Adams, and Harald Molle, “Arduino Robotics”, Springer, New York, 2011.
- Casey Reas and Ben Fry, “Getting Started with Processing”, O’Reilly Media, Inc., June 2010.
- Joshua Noble, “Programming Interactivity: A Designer’s Guide to Processing, Arduino, and openFrameworks”, O’Reilly Media, Inc., July 2009.
- Peter Hoddie, Lizzie Prader, “IoT Development for ESP32 and ESP8266 with JavaScript; A Practical Guide to XS and Moddable SDK”, Apress, Menlo Park, CA,USA, 2020.

MATERI

PENDAHULUAN

1. KOMUNIKASI MIKROKONTROLER DENGAN SOFTWARE PROCESSING
2 . ANALOG INPUT DAN AUDIO PROCESSING
3. KONTROL MULTI LED MENGGUNAKAN ARDUINO DAN PROCESSING
4. PENCAMPUR WARNA VIRTUAL MENGGUNAKAN ARDUINO DAN PROCESSING
5. MONITORING SUHU DENGAN ARDUINO DAN PROCESSING
6. MONITORING INTENSITAS CAHAYA DENGAN ARDUINO DAN PROCESSING
7. KONTROL MOTOR DC MENGGUNAKAN ARDUINO DAN PROCESSING
8. APLIKASI SENSOR ULTRASONIC MENGGUNAKAN ARDUINO DAN PROCESSING

MATERI

9. KONTROL LAMPU AC 220 V BERBASIS ARDUINO DAN PROCESSING
10. MODUL WiFi ESP32 DENGAN ARDUINO IDE
11. KOMUNIKASI NIRKABEL MENGGUNAKAN MODUL RF 434 MHz DAN PROCESSING
12. ESP32 WEB SERVER UNTUK KONTROL LED DAN MENAMPILKAN GAMBAR
13. ANTARMUKA MODUL GPS DENGAN MIKROKONTROLER DAN PROCESSING
14. ESP32 WEB SERVER UNTUK PENGUKURAN SUHU DAN KELEMBABAN
15. KOMUNIKASI DATA BERBASIS BLUETOOTH DAN HP

14. ESP32 WEB SERVER UNTUK PENGUKURAN SUHU DAN KELEMBABAN

TUJUAN

- Membangun web server mandiri.
- Membuat program untuk mengukur teperatur dan kelembaban menggunakan sensor DHT22 (AM2302).
- Menampilkan nilai temperature dan kelembaban pada WEB

DASAR TEORI

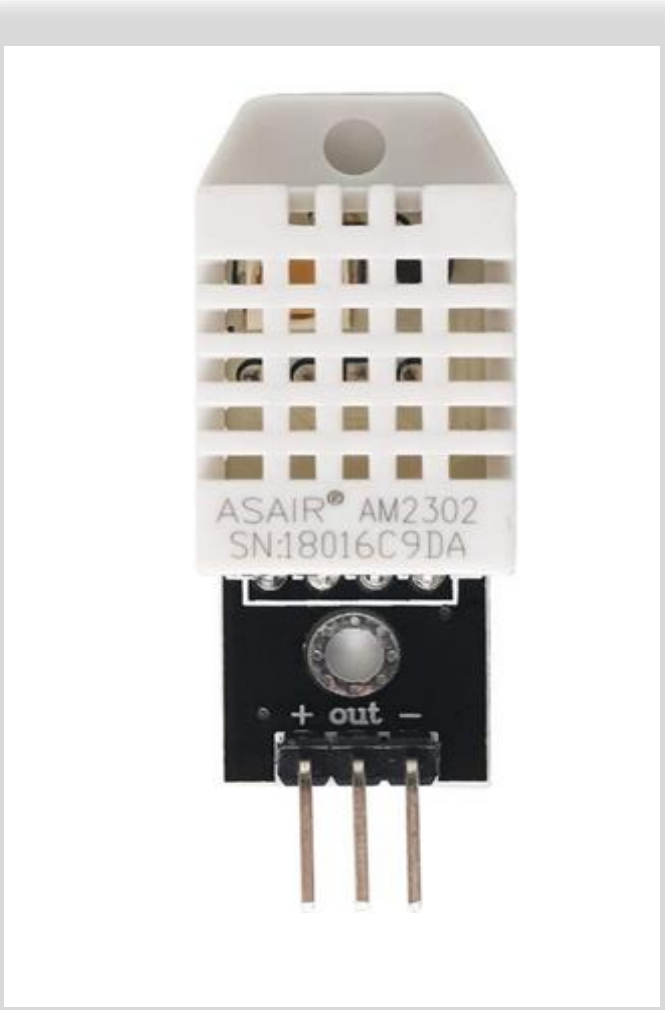
➢ Sensor DHT22 (AM2302)

- DHT-22 atau AM2302 adalah sensor suhu dan kelembaban,
- Sensor ini memiliki keluaran berupa sinyal digital dengan konversi dan perhitungan dilakukan oleh MCU 8-bit terpadu.
- Sensor ini memiliki kalibrasi akurat dengan kompensasi suhu ruang penyesuaian dengan nilai koefisien tersimpan dalam memori OTP terpadu.
- Sensor DHT22 memiliki rentang pengukuran suhu dan kelembaban yang luas,
- DHT22 mampu mentransmisikan sinyal keluaran melewati kabel hingga 20 meter sehingga sesuai untuk ditempatkan di mana saja

➢Sensor DHT22 (AM2302)

- Format keluaran sinyal: sinyal digital.
- Rentang pengukuran suhu: -40 ℃ - 80 ℃.
- Akurasi pengukuran: ± 0,5 ℃.
- Resolusi: 16 bit.
- Rentang pengukuran kelembaban: 0--100% RH.
- Akurasi: ± 2% RH.

Catatan: selama proses penggunaan, interval antara pembacaan pertama dan kedua harus lebih lama dari 2 detik.



➢ Komunikasi DHT22

- Komunikasi dan sinyal Data bus tunggal digunakan untuk komunikasi antara MCU dan DHT22, dengan waktu 5mS untuk satu kalikomunikasi. Data terdiri dari bagian integral dan desimal, berikut ini adalah rumus untuk data :

DATA = 16 bit data RH + 16 bit Data suhu + 8 bit check-sum

MCU telah menerima data 40 bit dari AM2302:

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16bit data RH **16bit data T** **8bit Jumlah cek**

Pengubahan16 bit data RH dari sistem biner ke sistem desimal,

0000 0010 1000 1100 → 652

Sistem biner → Sistem desimal

RH = 652/10 = 65,2% RH

Pengubahan16 bit data T dari sistem biner ke sistem desimal,

0000 0001 0101 1111 → 351

Sistem biner → Sistem desimal

T = 351/10 = 35,1 ℃

Sum = 0000 0010 + 1000 1100 + 0000 0001 + 0101 1111 = 1110 1110

Check-sum = 8 bit terakhir dari Sum = $1110\ 1110^{2}$

➢ Asynchronous Web Server

Membangun server web dengan menggunakan library **ESPAsyncWebServer** yang menyediakan cara mudah untuk membuat server web asinkron.

Keuntungan :

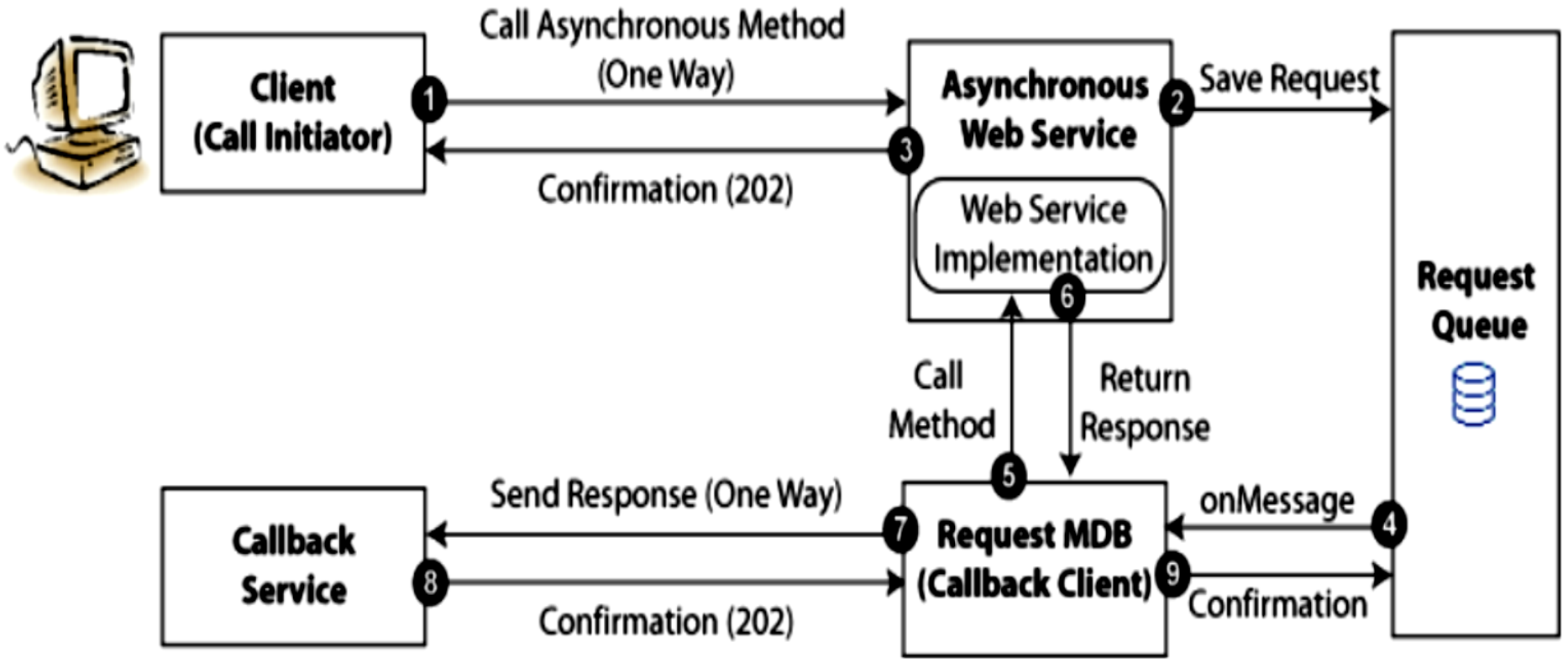
- Menangani lebih dari satu koneksi pada waktu yang sama
- Mesin pemroses template sederhana untuk menangani template
- Ketika mengirim respon, segera siap untuk menangani koneksi lain sementara server mengurus pengiriman respon di latar belakang
- Dengan memanggil layanan web secara asinkron, klien dapat melanjutkan pemrosesannya, tanpa interupsi, dan akan diberi tahu ketika respons asinkron dikembalikan.
- dll

Beberapa diagram alur pesan asinkron, memberikan perspektifdarisisi klien, dan menjelaskan bagaimana pesan asinkron berkorelasi:

- ✓ Memahami Alur Layanan Web Asinkron Menggunakan Antrian PermintaanTunggal (Single Request Queue)
- ✓ Memahami Alur Layanan Web Asinkron Menggunakan Permintaan dan Antrian Respon
- ✓ Memahami Perspektif Klien dari Panggilan Layanan Web Asinkron
- ✓ Memahami Bagaimana Pesan Asinkron Berkorelasi

Diagram berikut menggambarkan aliran layanan web asinkron menggunakan antrean permintaan tunggal. Dalam skenario ini, message-driven bean (MDB) yang terkait dengan antrian permintaan yang menangani permintaan dan pemrosesan respons.

Alur Layanan Web Asinkron Menggunakan Antrian Permintaan Tunggal (*Single Request Queue*)

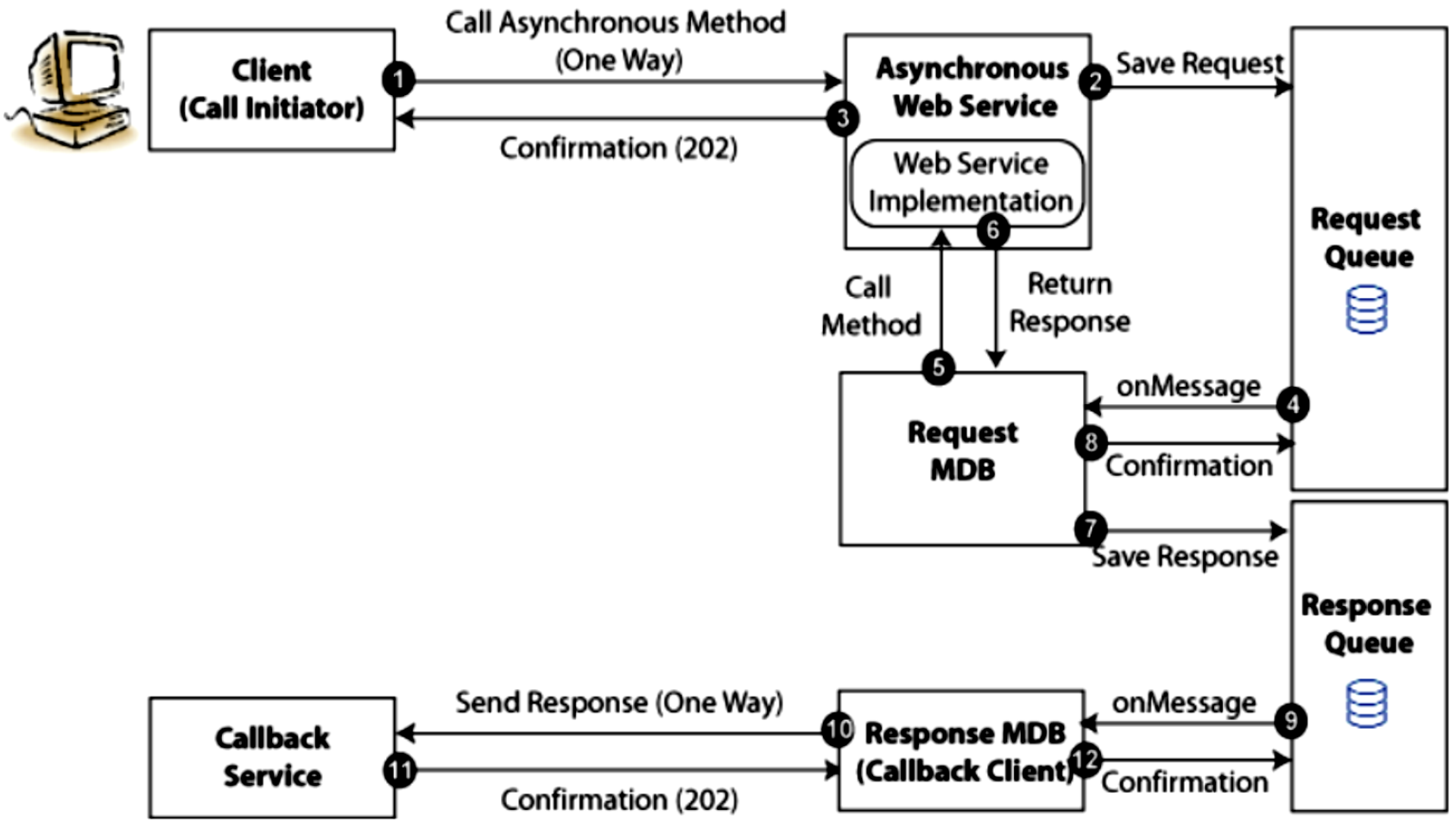


1. Klien memanggil metode asynchronous.
2. Layanan web asinkron menerima permintaan dan menyimpannya dalam antrian permintaan.
3. Layanan web asinkron mengirimkan konfirmasi tanda terima ke klien.
4. MDB listener pada antrian permintaan menerima pesan dan memulai pemrosesan permintaan.
5. Permintaan MDB memanggil metode yang diperlukan dalam implementasi layanan web.
6. Implementasi layanan web mengembalikan respons.
7. MDB permintaan, yang bertindak sebagai klien panggilan balik, mengembalikan respons ke layanan panggilan balik.
8. Layanan panggilan balik mengembalikan pesan konfirmasi tanda terima.
9. Permintaan MDB mengembalikan pesan konfirmasi ke antrian permintaan untuk menghentikan proses.

- Dalam skenario ini, jika ada masalah saat menyambung ke layanan panggilan balik (di Langkah 7), maka respons tidak akan dikirim.
- Jika permintaan tersebut dicoba lagi nanti, aliran akan dilanjutkan dari Langkah 4 dan implementasi layanan web akan dipanggil lagi (pada Langkah 5). Ini mungkin tidak diinginkan tergantung pada logika aplikasi atau pemrosesan kontrol transaksionalnya.
- Dalam skenario berikutnya, respons disimpan dalam antrean respons terpisah, menghilangkan kebutuhan untuk memanggil kembali implementasi layanan web jika layanan callback tidak tersedia pada awalnya.

Diagram berikut mengilustrasikan aliran panggilan metode asinkron menggunakan antrean permintaan tunggal. Dalam skenario ini, ada dua MDB, satu untuk menangani pemrosesan permintaan dan satu lagi untuk menangani pemrosesan respons.
Dengan memisahkan eksekusi logika bisnis dari respons kembali, skenario ini menyediakan pemulihan kesalahan yang ditingkatkan melalui model antrian tunggal yang dijelaskan dalam Memahami Alur Layanan Web Asinkron Menggunakan Antrian Permintaan Tunggal (*Single Request Queue*).

Alur Layanan Web Asinkron Menggunakan Permintaan dan Antrian Respon (*Request and Response Queue*)

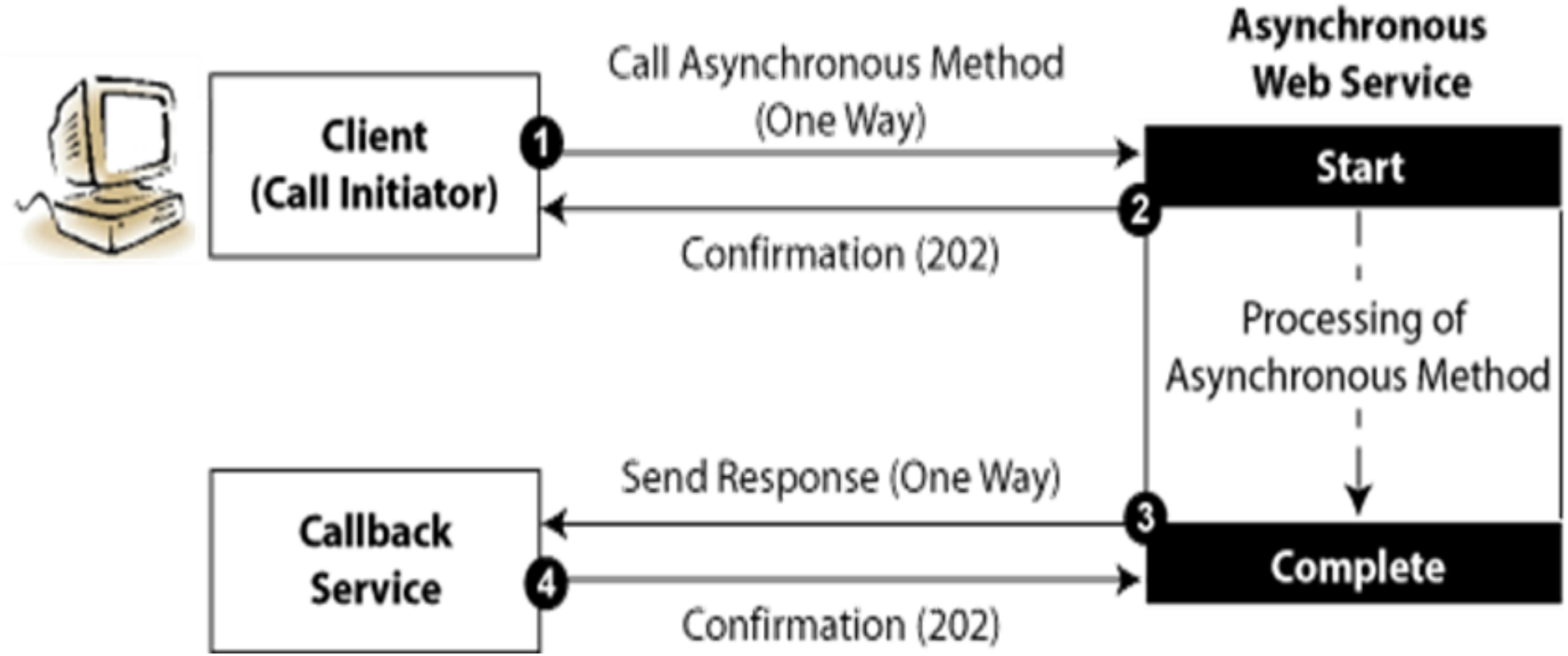


1. Klien memanggil metode asynchronous.
2. Layanan web asinkron menerima permintaan dan menyimpannya dalam antrian permintaan.
3. Layanan web asinkron mengirimkan konfirmasi tanda terima ke klien.
4. MDB listener pada antrian permintaan menerima pesan dan memulai pemrosesan permintaan.
5. Permintaan MDB memanggil metode yang diperlukan dalam implementasi layanan web.
6. Implementasi layanan web mengembalikan respons.
7. Permintaan MDB menyimpan respon ke antrian respon.
8. Permintaan MDB mengirimkan konfirmasi ke antrian permintaan untuk menghentikan proses.

9. Pemroses onMessage di antrean respons memulai pemrosesan respons.
10. Respons MDB, yang bertindak sebagai klien panggilan balik, mengembalikan respons ke layanan panggilan balik.
11. Layanan panggilan balik mengembalikan pesan konfirmasi tanda terima.
12. Respon MDB mengembalikan pesan konfirmasi ke antrian respon untuk menghentikan urutan.

Diagram berikut mengilustrasikan aliran, dari perspektif klien, panggilan metode asinkron yang terdiri dari dua pertukaran pesan satu arah.

Memahami Perspektif Klien dari Panggilan Layanan Web Asinkron



Ditunjukkan pada gambar, sebelum memulai panggilan asinkron, klien harus menerapkan layanan panggilan balik untuk mendengarkan respons dari layanan web asinkron.

Langkah-langkah berikut menjelaskan aliran pesan yang ditunjukkan pada gambar sebelumnya:

1. Klien memanggil metode asynchronous.
2. Layanan web asynchronous menerima permintaan tersebut, mengirimkan pesan konfirmasi ke klien yang memulai, dan mulai memproses permintaan tersebut.
3. Setelah pemrosesan permintaan selesai, layanan web asinkron bertindak sebagai klien untuk mengirim respons kembali ke layanan panggilan balik.
4. Layanan panggilan balik mengirimkan pesan konfirmasi ke layanan web asinkron.

Memahami Bagaimana Pesan Asinkron Berkorelasi

catatan:
Korelasi pesan ditangani secara otomatis oleh runtime. Bagian ini hanya untuk tujuan informasi.

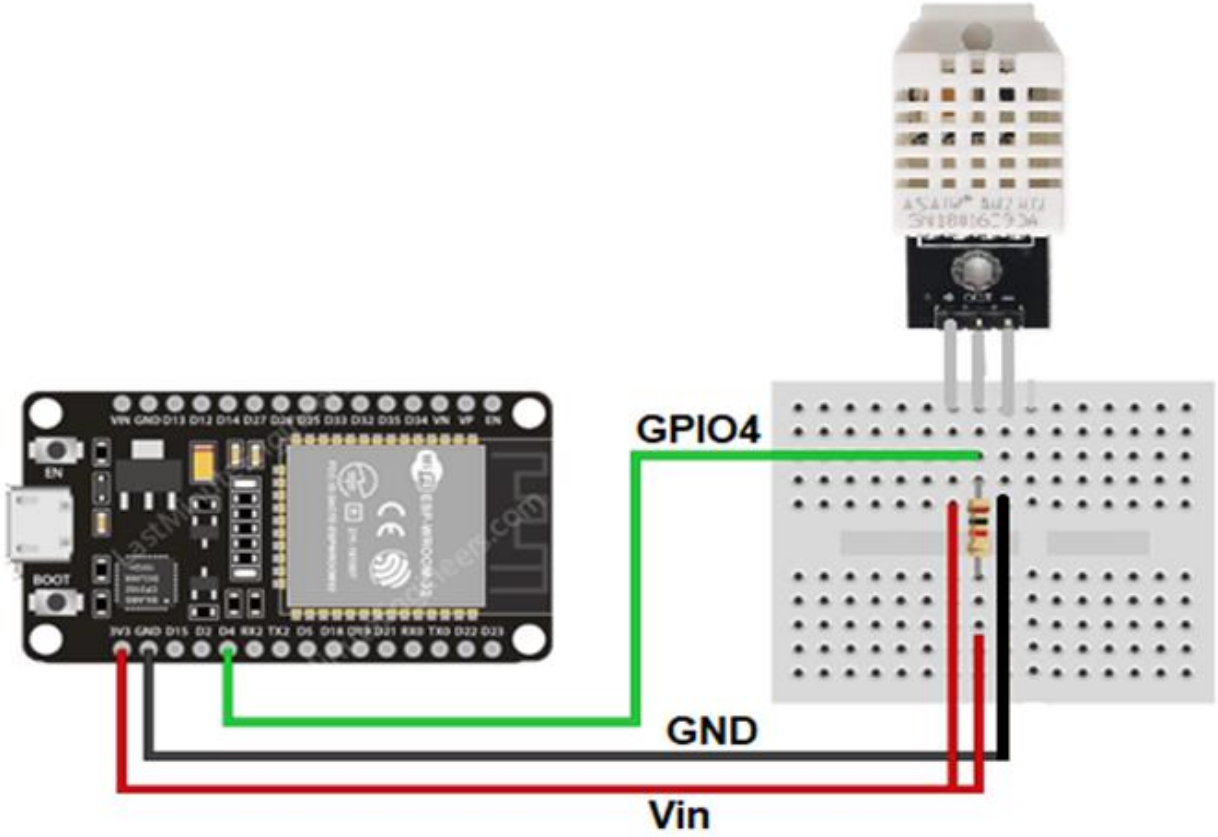
Saat layanan callback menerima respons, diperlukan cara untuk menghubungkan respons kembali ke permintaan asli. Ini dicapai dengan menggunakan WS-Addressing dan ditangani secara otomatis oleh runtime.Klien menetapkan dua bidang berikut di bagian WS-Addressing dari header SOAP:

- ReplyTo address— Alamat layanan panggilan balik.
- MessageID — ID unik yang mengidentifikasi permintaan. Misalnya, UUID.

Klien panggilan balik mengirimkan Message Id yang sesuai dengan permintaan awal di bidang relatesToId di header WS-Addressing. Jika data tambahan diperlukan oleh layanan panggilan balik untuk memproses respons, klien dapat melakukan salah satu tugas berikut:

- Klien dapat mengirim data sebagai parameter referensi di kolom ReplyTo. Layanan web asinkron mengembalikan semua parameter referensi beserta responsnya, sehingga layanan panggilan balik akan dapat mengakses informasi tersebut.
- Jika pengiriman data sebagai bagian dari pesan permintaan asinkron tidak praktis, maka klien dapat menyimpan MessageID dan data yang diperlukan ke penyimpanan data lokal.

Rangkaian



PROGRAM:

Arduino IDE

```
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include <Adafruit_Sensor.h>
#include <DHT.h>

const char* ssid     = “ ";
const char* password = “ ";
#define DHTPIN 4
#define DHTTYPE    DHT22     // DHT 22 (AM2302)

// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

String readDHTTemperature() {
float t = dht.readTemperature();
```

PROGRAM: *Lanjutan*

```
 if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");     return "--";
  }
  else {
    Serial.println(t);     return String(t);
  }
}
String readDHTHumidity() {
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  if (isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");     return "--";
  }
  else {
    Serial.println(h);     return String(h);
  }
}
```

PROGRAM: Lanjutan

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
  <style>
    html {
     font-family: Arial;      display: inline-block;
     margin: 0px auto;      text-align: center;
    }
    h2 { font-size: 3.0rem; }     p { font-size: 3.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;      vertical-align:middle;      padding-bottom: 15px;     }
  </style>
</head>
```

PROGRAM: Lanjutan

```
<body>
  <h2>ESP32 DHT Server</h2>
  <p>
    <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels">Temperature</span>
    <span id="temperature">%TEMPERATURE%</span>
    <sup class="units">&deg;C</sup>
  </p>
  <p>
    <i class="fas fa-tint" style="color:#00add6;"></i>
    <span class="dht-labels">Humidity</span>
    <span id="humidity">%HUMIDITY%</span>
    <sup class="units">%</sup>
  </p>
</body>
<script>
setInterval(function ( ) {
```

PROGRAM: Lanjutan

```
var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);   xhttp.send(); }, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();   xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;     }
  };
  xhttp.open("GET", "/humidity", true);   xhttp.send(); }, 10000 ) ;
</script>
</html>)rawliteral";
```

PROGRAM: Lanjutan

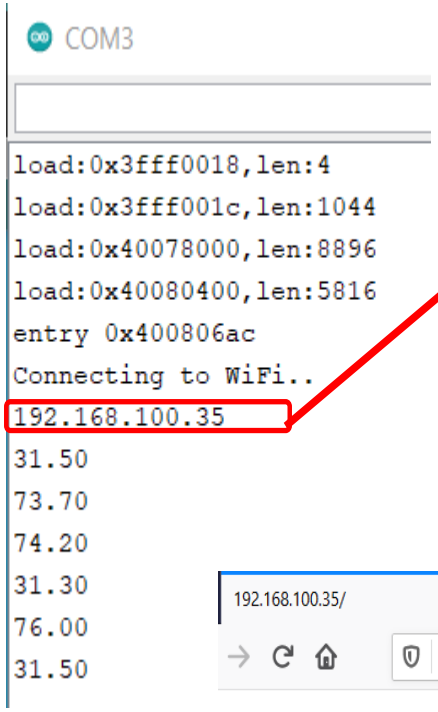
```
// Replaces placeholder with DHT values
String processor(const String& var){
  //Serial.println(var);
  if(var == "TEMPERATURE"){
    return readDHTTemperature();
  }
  else if(var == "HUMIDITY"){
    return readDHTHumidity();
  }
  return String();
}

void setup(){
  Serial.begin(115200);
  dht.begin();
   // Connect to Wi-Fi
  WiFi.begin(ssid, password);
```

PROGRAM: *Lanjutan*

```
while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.println("Connecting to WiFi..");
  }
Serial.println(WiFi.localIP());
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readDHTTemperature().c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readDHTHumidity().c_str());
  });
  // Start server
  server.begin();
}
```

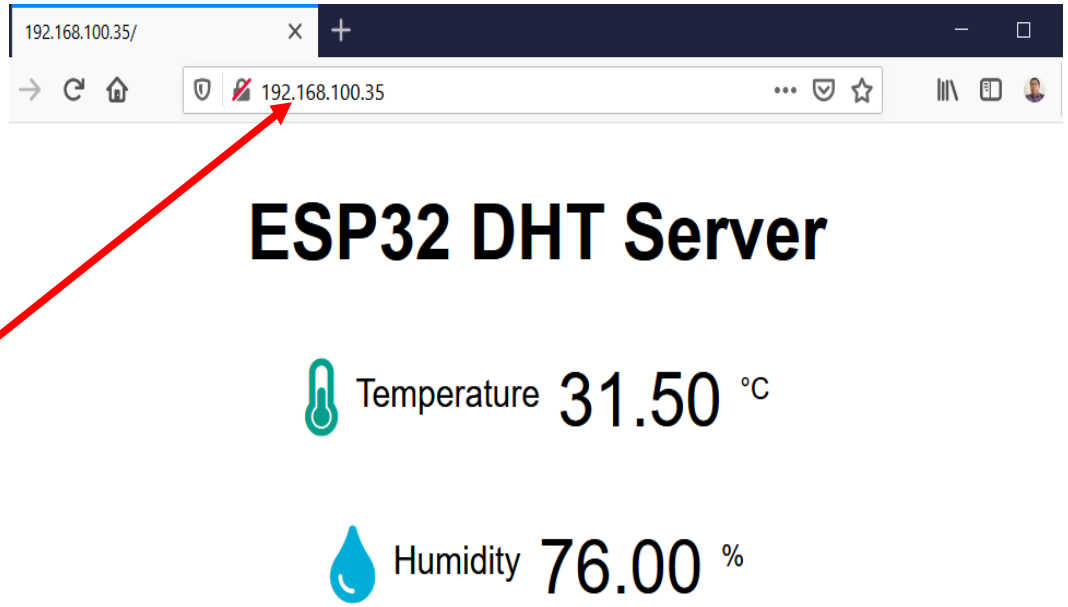
```
void loop(){

}
```

- Setelah program diupload, buka Serial Monitor dengan baud rate = 115200.



IP address

- Tekan tobol button “EN”.
- Buka browser, paste IP address ESP32, dan lihat halaman web.



Hasil :

- Amati/foto dan catat nilai temperatur dan kelembaban yg ada di web

Latihan :

1. Buatlah tampilan yang berbeda pada web dari contoh percobaan. Tampilkan data-data berikut:
   - Nilai suhu dalam derajat Celsius dan Fahreheit
   - Nilai Humidity dalam %