IMPLEMENTASI SISTEM MULTITASKING BERBASIS FREERTOS PADA ESP32 UNTUK MONITORING SENSOR DAN KENDALI JARAK JAUH VIA MQTT

Untuk memenuhi Ujian Akhir Semester pada mata kuliah Sistem Mikroprosessor

Dosen Pengampu :
Muhammad Ikhwan Fathulloh, S.Kom.

DISUSUN OLEH :

Nama : Tinton Despi Alkhifari
NIM : 23552011162
Kelas : TIF K 23A

TEKNIK INFORMATIKA

UNIVERSITAS TEKNOLOGI BANDUNG

BANDUNG

TAHUN 2026

ABSTRAK

Perkembangan sistem embedded menuntut perangkat untuk mampu menjalankan beberapa proses secara bersamaan, berkomunikasi melalui jaringan, serta merespons input secara real-time. Pada proyek ini dirancang dan diimplementasikan sebuah sistem embedded berbasis ESP32 yang memanfaatkan FreeRTOS sebagai sistem operasi real-time untuk mendukung multitasking. Sistem mampu membaca data dari sensor analog, mendeteksi input tombol menggunakan mekanisme interupsi tanpa polling, serta mengendalikan LED sebagai output.

Komunikasi data dilakukan melalui jaringan WiFi menggunakan protokol MQTT dengan broker public MQTT Explorer. Data hasil pembacaan sensor dikirimkan secara periodik ke broker MQTT, sementara perintah untuk mengontrol LED diterima secara real-time melalui topik yang telah ditentukan. Penggunaan FreeRTOS memungkinkan pembagian tugas secara terstruktur, sehingga setiap fungsi sistem dapat berjalan secara efisien dan responsif tanpa saling mengganggu.

Hasil pengujian menunjukkan bahwa sistem dapat beroperasi dengan stabil, mampu menangani komunikasi jaringan, interupsi, dan multitasking dengan baik. Implementasi ini menunjukkan penerapan konsep manajemen mikroprosesor, meliputi manajemen proses, komunikasi data, serta pengendalian perangkat keras, sehingga sistem yang dibangun memenuhi kriteria sebagai sebuah sistem embedded berbasis IoT yang efisien dan terintegrasi.

Kata Kunci: ESP32, FreeRTOS, Multitasking, Hardware Interrupt, MQTT, IoT.

BAB I : PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi sistem embedded dan Internet of Things (IoT) mendorong kebutuhan perangkat yang mampu bekerja secara efisien, responsif, dan terintegrasi dengan jaringan. Sistem embedded modern tidak hanya dituntut untuk mengendalikan perangkat keras, tetapi juga mampu menjalankan beberapa proses secara bersamaan, mengelola komunikasi data, serta merespons input secara real-time.

ESP32 merupakan salah satu mikrokontroler yang mendukung kebutuhan tersebut karena telah dilengkapi dengan prosesor dual-core, modul WiFi, serta dukungan sistem operasi real-time (FreeRTOS). Dengan adanya FreeRTOS, pengelolaan tugas (task), interupsi, serta komunikasi antar proses dapat dilakukan secara terstruktur dan efisien.

Pada proyek ini dirancang sebuah sistem embedded berbasis ESP32 yang memanfaatkan FreeRTOS untuk multitasking, mekanisme interupsi untuk input tombol, serta protokol MQTT sebagai media komunikasi data melalui jaringan WiFi. Sistem ini diharapkan mampu menjadi contoh implementasi konsep manajemen mikroprosesor yang mencakup manajemen proses, komunikasi data, dan pengendalian perangkat keras dalam satu sistem terintegrasi.

1.2 Tujuan

1. Merancang dan mengimplementasikan sebuah sistem embedded berbasis ESP32 yang mampu menerapkan konsep manajemen mikroprosesor secara terintegrasi. Sistem dikembangkan dengan memanfaatkan FreeRTOS untuk mendukung multitasking, sehingga beberapa proses dapat dijalankan secara bersamaan tanpa saling mengganggu.

2. Proyek ini bertujuan untuk mengimplementasikan mekanisme interupsi hardware sebagai metode penanganan input tombol secara efisien tanpa menggunakan polling. Sistem juga dirancang agar mampu melakukan komunikasi data melalui jaringan WiFi menggunakan protokol MQTT, baik untuk pengiriman data sensor maupun penerimaan perintah kendali secara real-time.
3. Mengendalikan perangkat output berupa LED sebagai simulasi aktuator, serta memastikan sistem dapat beroperasi secara stabil, responsif, dan terstruktur. Melalui proyek ini, diharapkan mahasiswa dapat memahami dan mengaplikasikan konsep dasar sistem embedded, komunikasi jaringan, serta pengelolaan proses pada mikroprosesor dalam satu sistem yang terpadu.

BAB II : LANDASAN TEORI

2.1 Diagram Blok Sistem

Sistem yang dirancang terdiri dari beberapa blok utama yang saling terhubung dan bekerja secara terkoordinasi, yaitu:

1. ESP32 (Unit Pemroses Utama)

   ESP32 berperan sebagai pusat pengendali sistem. Seluruh proses pengolahan data, manajemen task FreeRTOS, komunikasi WiFi, dan kontrol perangkat keras dijalankan pada mikrokontroler ini.
2. Sensor Analog (GPIO 34)

   Sensor analog digunakan sebagai sumber input data. Nilai sensor dibaca secara periodik oleh task khusus (TaskSensor) dan kemudian dikirimkan ke broker MQTT.

3. Tombol (GPIO 4)

   Tombol berfungsi sebagai input darurat yang diproses menggunakan mekanisme interupsi hardware. Ketika tombol ditekan, interupsi akan memicu semaphore untuk memberi sinyal ke task utama tanpa menggunakan metode polling.
4. LED (GPIO 2)

   LED digunakan sebagai output sistem yang dikendalikan melalui perintah MQTT. LED dapat dinyalakan atau dimatikan berdasarkan pesan yang diterima dari broker.
5. Jaringan WiFi

   ESP32 terhubung ke jaringan WiFi sebagai media komunikasi dengan broker MQTT.
6. Broker MQTT (HiveMQ)

   Broker MQTT berfungsi sebagai perantara komunikasi antara ESP32 dan pengguna. Data sensor dipublikasikan ke broker, sedangkan perintah kontrol LED dikirimkan dari pengguna ke ESP32 melalui topik tertentu.
7. MQTT Client (MQTT Explorer)

   Digunakan untuk memantau data sensor dan mengirimkan perintah kendali LED secara real-time.

BAB III : PERANCANGAN SISTEM

3.1 Gambaran Umum Sistem

Perancangan sistem pada proyek ini bertujuan untuk membangun sebuah sistem embedded berbasis ESP32 yang mampu menjalankan beberapa fungsi secara bersamaan, merespons input secara real-time, serta melakukan komunikasi data melalui jaringan.

3.2 Arsitektur Sistem

Arsitektur sistem dirancang menggunakan model berbasis task pada FreeRTOS. Setiap task memiliki fungsi spesifik dan berkomunikasi menggunakan mekanisme komunikasi antar-proses yang disediakan oleh FreeRTOS, seperti queue dan semaphore**.**

Secara umum, sistem terdiri dari:

1. Task pembacaan sensor analog
2. Task pembacaan sensor analog
3. Mekanisme interupsi untuk tombol eksternal
4. Output berupa LED sebagai actuator

3.3 Perancangan Perangkat Keras

3.3.1 ESP32

ESP32 digunakan sebagai mikrokontroler utama karena memiliki prosesor yang mendukung FreeRTOS, modul WiFi terintegrasi, serta jumlah GPIO yang mencukupi untuk menghubungkan sensor, tombol, dan LED.

3.3.2 Sensor Analog

Sensor analog dihubungkan ke GPIO 34 yang berfungsi sebagai pin input analog. Sensor ini digunakan untuk mensimulasikan pembacaan data analog yang kemudian diproses dan dikirimkan ke broker MQTT.

3.3.3 Tombol

Tombol eksternal dihubungkan ke GPIO 4 dan dikonfigurasikan sebagai input dengan mode pull-up internal. Tombol digunakan sebagai input darurat yang diproses menggunakan mekanisme interupsi hardware.

3.3.4 LED

LED dihubungkan ke GPIO 2 dan berfungsi sebagai output sistem. LED digunakan untuk menunjukkan hasil perintah yang diterima melalui MQTT, yaitu perintah untuk menyalakan dan mematikan LED.

3.4 Perancangan Perangkat Lunak

3.4.1 Sistem Operasi FreeRTOS

FreeRTOS digunakan untuk mengelola multitasking dalam sistem. Sistem dibagi menjadi dua task utama, yaitu TaskSensor dan TaskMQTT, yang dijalankan secara bersamaan dengan prioritas yang berbeda**.**

3.4.2 Task Pembacaan Sensor (TaskSensor)

TaskSensor bertugas membaca data dari sensor analog secara periodik setiap 5 detik. Data hasil pembacaan sensor kemudian dikirimkan ke task lain melalui mekanisme queue.

3.4.3 Task Manajemen MQTT (TaskMQTT)

TaskMQTT bertanggung jawab untuk:

1. Menjaga koneksi WiFi
2. Mengelola koneksi ke broker MQTT
3. Menerima data sensor dari queue
4. Menerima data sensor dari queue
5. Mengirimkan data sensor ke broker MQTT

3.5 Mekanisme Interupsi Tombol

Tombol eksternal dirancang menggunakan mekanisme interupsi hardware untuk menghindari penggunaan metode polling. Ketika tombol ditekan, interupsi akan memicu fungsi ISR (Interrupt Service Routine) yang kemudian memberikan sinyal ke task utama menggunakan semaphore. Dengan pendekatan ini, sistem dapat merespons input tombol secara cepat dan efisien tanpa membebani prosesor.

3.6 Komunikasi Antar-Task

Untuk menjaga integritas data dan sinkronisasi antar-task, sistem memanfaatkan fitur FreeRTOS berupa:

1. **Queue**, digunakan untuk mengirim data sensor dari TaskSensor ke TaskMQTT
2. **Semaphore**, digunakan untuk memberikan sinyal dari ISR tombol ke TaskMQTT

Penggunaan mekanisme ini memastikan komunikasi antar-task berjalan aman dan terstruktur.

3.7 Perancangan Komunikasi MQTT

Komunikasi data dilakukan menggunakan protokol MQTT dengan broker publik HiveMQ. Sistem menggunakan dua topik utama, yaitu:

1. Topik untuk mengirim data sensor dari ESP32 ke broker2.
2. Topik untuk menerima perintah kendali LED dari broker ke ESP32

Melalui mekanisme publish dan subscribe, sistem dapat melakukan komunikasi dua arah secara real-time.

3.8 Alur Kerja Sistem

Secara umum, alur kerja sistem adalah sebagai berikut:

1. ESP32 melakukan inisialisasi perangkat keras dan FreeRTOS
2. Sistem terhubung ke jaringan WiFi
3. ESP32 terkoneksi ke broker MQTT
4. ESP32 terkoneksi ke broker MQTT
5. TaskSensor membaca data sensor secara periodic
6. TaskMQTT mempublikasikan data ke broker MQTT
7. ESP32 menerima perintah dari broker MQTT
8. LED dikendalikan sesuai perintah yang diterima
9. Interupsi tombol diproses dan dikirim sebagai notifikasi ke broker

3.9 Ringkasan Perancangan Sistem

Perancangan sistem ini mengintegrasikan perangkat keras dan perangkat lunak secara terstruktur dengan memanfaatkan FreeRTOS, mekanisme interupsi, serta protokol MQTT. Pendekatan ini memungkinkan sistem untuk berjalan secara multitasking, responsif, dan stabil, sesuai dengan konsep manajemen mikroprosesor yang menjadi fokus pada proyek ini**.**

BAB IV : IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap penerapan hasil perancangan sistem ke dalam bentuk perangkat keras dan perangkat lunak. Implementasi dilakukan dengan menggunakan ESP32 sebagai mikrokontroler utama, Arduino IDE sebagai lingkungan pengembangan,

serta FreeRTOS sebagai sistem operasi real-time yang berjalan di atas ESP32.

Perangkat keras dihubungkan sesuai dengan perancangan yang telah dibuat, meliputi sensor analog pada GPIO 34, tombol eksternal pada GPIO 4, dan LED sebagai output pada GPIO 2. Seluruh komponen diintegrasikan pada breadboard untuk memudahkan pengujian.

Pada sisi perangkat lunak, program ditulis menggunakan bahasa C/C++ pada Arduino IDE. Program dibagi menjadi beberapa bagian utama, yaitu inisialisasi sistem, konfigurasi WiFi dan MQTT, definisi task FreeRTOS, mekanisme interupsi tombol, serta logika pengendalian LED.

4.2 Implementasi Perangkat Keras

Berikut adalah dokumentasi foto hasil perakitan rangkaian pada breadboard yang terhubung dengan modul WeMos D1 R32 (ESP32).

(Gambar 4.2.1 Hardware)

4.3 Implementasi Kode Program (Source Code)

Berikut adalah kode program lengkap yang telah diimplementasikan dan diunggah ke ESP32. Kode ini menggabungkan semua fungsi menggunakan struktur FreeRTOS :

```
// ====================================================================
// SKETCH FINAL: TUGAS MIKROPROSESOR ESP32 + FREERTOS
// Integrasi: Sensor Analog (D34) + Interupsi Tombol (D4) + WiFi/MQTT + Output LED (D2)
// ====================================================================

#include <WiFi.h>
#include <PubSubClient.h>

// --- 1. KONFIGURASI PIN HARDWARE ---
#define PIN_SENSOR_ANALOG 34   // Sensor palsu (pembagi tegangan)
#define PIN_TOMBOL_EKSTERNAL 4 // Tombol di breadboard (Active Low)
#define PIN_LED_ONBOARD 2      // LED biru bawaan

// --- 2. KONFIGURASI JARINGAN (WAJIB DIGANTI!) ---
// ============================================================
const char* ssid = "Homestay2";      // <--- SSID WIFI
const char* password = "tinton123";  // <--- PASSWORD WIFI
// ============================================================

// --- 3. KONFIGURASI MQTT ---
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;
const char* topik_kirim_data = "kampus/esp32/proyek_bapak/data";
const char* topik_terima_perintah = "UAS/esp32/Mikroprosessor/perintah";

// --- OBJEK GLOBAL & VARIABEL FREERTOS ---
WiFiClient espClient;
PubSubClient client(espClient);
```

```
// Handle untuk Task (Pekerja)
TaskHandle_t TaskSensorHandle = NULL;
TaskHandle_t TaskMQTTHandle = NULL;

// Queue (Antrean) untuk mengirim data sensor antar-task dengan aman
QueueHandle_t sensorQueue;
// Semaphore (Bendera) untuk sinyal interupsi tombol
SemaphoreHandle_t buttonSemaphore;

// ====================================================================
// BAGIAN INTERUPSI & CALLBACK (Respon Cepat)
// ====================================================================

// ISR: Fungsi kilat saat tombol ditekan
void IRAM_ATTR handleTombolISR() {
  // Beri sinyal (semaphore) ke task utama bahwa tombol ditekan.
  // "FromISR" artinya aman dipanggil dari interupsi.
  xSemaphoreGiveFromISR(buttonSemaphore, NULL);
}

// Callback MQTT: Saat ada perintah masuk dari browser
void callback(char* topic, byte* payload, unsigned int length) {
  String pesan = "";
  for (int i = 0; i < length; i++) pesan += (char)payload[i];
  Serial.printf("\n[PERINTAH MASUK] %s\n", pesan.c_str());

```

```
// ========================================================================
// FUNGSI PENDUKUNG KONEKSI
// ========================================================================
void setup_wifi() {
  delay(10);

  WiFi.mode(WIFI_STA);
  WiFi.disconnect(true);
  delay(100);

  Serial.print("\nMenghubungkan WiFi: ");
  Serial.println(ssid);

  Serial.print("MAC ESP32: ");
  Serial.println(WiFi.macAddress());

  WiFi.begin(ssid, password);

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

  Serial.println("\nWiFi Terhubung!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}
```

```
void reconnect_mqtt() {
  while (!client.connected()) {
    Serial.print("Konek MQTT...");
    String clientId = "ESP32-TugasAkhir-" + String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("Berhasil!");
      client.subscribe(topik_terima_perintah);
      client.publish(topik_kirim_data, "Sistem ESP32 FreeRTOS Online!");
    } else {
      Serial.print("Gagal, rc="); Serial.print(client.state()); delay(2000);
    }
  }
}

// ========================================================================
// DEFINISI TASKS FREERTOS (PEKERJA UTAMA)
// ========================================================================

// --- TASK 1: Pembaca Sensor (Jalan setiap 5 detik) ---
void TaskSensor(void *pvParameters) {
  int nilaiSensorRaw;
  for (;;) { // Loop tak berujung (pengganti loop() biasa)
    // 1. Baca Sensor
    nilaiSensorRaw = analogRead(PIN_SENSOR_ANALOG);

    // 2. Kirim nilai ke dalam Antrean (Queue) agar diambil Task MQTT
    // portMAX_DELAY artinya tunggu sampai antrean kosong jika penuh.
    xQueueSend(sensorQueue, &nilaiSensorRaw, portMAX_DELAY);
```

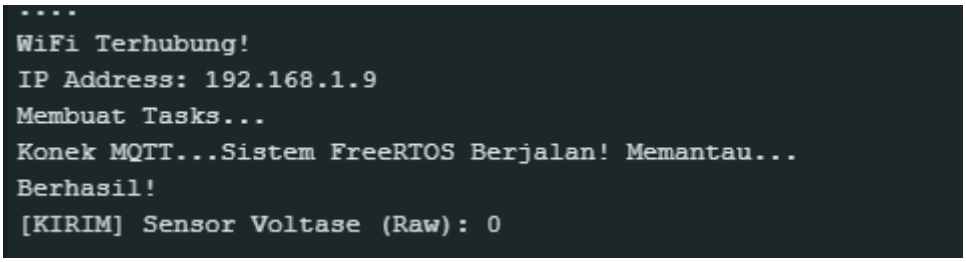
```
// --- TASK 2: Manajer MQTT & Logika Utama (Jalan secepat mungkin) ---
void TaskMQTT(void *pvParameters) {
  int dataSensorDiterima;

  // Setup awal khusus task ini
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);

  for (;;) {
    // Jaga koneksi
    if (WiFi.status() != WL_CONNECTED) setup_wifi();
    if (!client.connected()) reconnect_mqtt();
    client.loop(); // Penting untuk menerima pesan MQTT

    // A. Cek apakah ada data sensor baru di Antrean?
    // (Tunggu 0ms, kalau gak ada lanjut)
    if (xQueueReceive(sensorQueue, &dataSensorDiterima, 0) == pdTRUE) {
      String pesanData = "Sensor Voltase (Raw): " + String(dataSensorDiterima);
      Serial.println("[KIRIM] " + pesanData);
      client.publish(topik_kirim_data, pesanData.c_str());
    }
```

```
// SETUP UTAMA (Hanya jalan sekali saat boot)
// ================================================================
void setup() {
  Serial.begin(115200);
  Serial.println("\n--- MEMULAI SISTEM FREERTOS ESP32 ---");

  // 1. Setup Pin
  pinMode(PIN_SENSOR_ANALOG, INPUT);
  pinMode(PIN_TOMBOL_EKSTERNAL, INPUT_PULLUP); // Penting untuk tombol active low
  pinMode(PIN_LED_ONBOARD, OUTPUT);
  digitalWrite(PIN_LED_ONBOARD, LOW);

  // 2. Setup Koneksi Awal
  setup_wifi();

  // 3. Buat Sumber Daya RTOS
  // Antrean untuk menampung maksimal 10 data integer
  sensorQueue = xQueueCreate(10, sizeof(int));
  // Semaphore biner untuk sinyal tombol
  buttonSemaphore = xSemaphoreCreateBinary();

  // 4. Pasang Interupsi
  attachInterrupt(digitalPinToInterrupt(PIN_TOMBOL_EKSTERNAL), handleTombolISR, FALLING);

  // 5. Buat dan Jalankan Task (Pekerja)
  Serial.println("Membuat Tasks...");
  // xTaskCreate(FungsiTask, "NamaBebas", UkuranStack, Parameter, Prioritas, Handle);
  xTaskCreate(TaskSensor, "SensorRead", 2048, NULL, 1, &TaskSensorHandle);
  xTaskCreate(TaskMQTT,   "MQTTManager", 4096, NULL, 2, &TaskMQTTHandle); // Prioritas lebih tinggi
```

(Gambar 4.3.1 Source Code)

4.4 Pengujian Sistem

4.4.1 Skenario 1 : Monitoring Sensor Periodik



(Gambar 4.4.1 Membuat Task Terhubung Ke WIFI)

Berdasarkan tampilan Serial Monitor tersebut, dapat disimpulkan bahwa:

- ESP32 berhasil terhubung ke jaringan WiFi
- Sistem FreeRTOS berjalan dengan baik
- Koneksi MQTT berhasil dilakukan
- Task pembacaan sensor dan pengiriman data berjalan sesuai dengan perancangan

Dengan demikian, menunjukan bukti bahwa sistem telah beroperasi secara normal dan seluruh fungsi utama sistem berhasil diimplementasikan.

4.4.2 Skenario 2 : Pengujian Interupsi Tombol Darurat

(Gambar 4.4.2 Task Tombol Eksternal Alert di Broadbord)

Sistem merespon seketika dengan menampilkan pesan "ALERT" di Serial Monitor, mengirim pesan peringatan ke MQTT, dan mengedipkan LED onboard. Ini membuktikan hardware interrupt dan mekanisme semaphore berfungsi.

4.4.3 Skenario 3 : Kendali Jarak Jauh via MQTT

```
[PERINTAH MASUK] NYALA
-> Aksi: LED Dinyalakan.
[KIRIM] Sensor Voltase (Raw): 0
[KIRIM] Sensor Voltase (Raw): 0
[KIRIM] Sensor Voltase (Raw): 0
```

```
[PERINTAH MASUK] MATI
-> Aksi: LED Dimatikan.
[KIRIM] Sensor Voltase (Raw): 0
[KIRIM] Sensor Voltase (Raw): 0
[KIRIM] Sensor Voltase (Raw): 0
```

(Gambar 4.4.3 Mengirim pesan "NYALA" dan "MATI")

Sistem ESP32 yang dikembangkan ***telah berhasil memenuhi fungsi utama***, yaitu:

- Terhubung ke WiFi dan broker MQTT
- Menerima dan mengeksekusi perintah MQTT secara real-time
- Mengontrol LED sebagai aktuator
- Menjalankan pembacaan sensor secara periodik menggunakan FreeRTOS

BAB V : KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem, dapat disimpulkan bahwa sistem embedded berbasis ESP32 yang dikembangkan pada proyek ini telah berhasil diimplementasikan dengan baik. Sistem mampu menjalankan beberapa proses secara bersamaan menggunakan FreeRTOS, membaca data sensor analog, menangani input tombol melalui mekanisme interupsi, serta melakukan komunikasi data melalui protokol MQTT.

Implementasi sistem ini menunjukkan bahwa konsep manajemen mikroprosesor, seperti manajemen proses, komunikasi antar-proses, serta pengendalian perangkat keras, dapat diterapkan secara efektif pada ESP32. Sistem juga terbukti berjalan secara stabil dan responsif sesuai dengan tujuan yang telah ditetapkan.

5.2 Saran

Untuk pengembangan lebih lanjut, sistem ini dapat ditingkatkan dengan menambahkan penyimpanan data non-volatile menggunakan EEPROM atau Preferences untuk menyimpan konfigurasi sistem. Selain itu, pengendalian output dapat dikembangkan menggunakan teknik PWM yang lebih variatif, serta penambahan antarmuka berbasis web atau dashboard IoT untuk meningkatkan kemudahan monitoring dan kontrol sistem.

DAFTAR PUSTAKA

Espressif Systems. (2023). *ESP32 Technical Reference Manual*. Espressif Systems.

Espressif Systems. (2023). *ESP32 Arduino Core Documentation*. https://docs.espressif.com/projects/arduino-esp32

Barry, R. (2021). *Mastering the FreeRTOS™ Real Time Kernel*. Real Time Engineers Ltd.

FreeRTOS. (2023). *FreeRTOS Documentation*. https://www.freertos.org

HiveMQ. (2023). *MQTT Essentials Guide*. https://www.hivemq.com/mqtt-essentials

MQTT.org. (2023). *MQTT Version 3.1.1Specification*. https://mqtt.org

Arduino. (2023). *Arduino IDE Documentation*. https://www.arduino.cc

Monk, S. (2016). *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education.

Kurniawan, A. (2019). *Internet of Things dengan ESP32*. Jakarta: Elex Media Komputindo.