



REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

SECURE SERVER ACCESS DOOR

GROUP 21

Stevie Nathania Siregar	2306242382
Mirza Adi Raffiansyah	2306210323
Ruben Kristanto	2306214624
Kharisma Aprilia	2306223244

PREFACE

Puji syukur kami panjatkan kehadirat Allah SWT, Tuhan Yang Maha Esa, yang telah melimpahkan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan proyek akhir praktikum Sistem Waktu Nyata dan IoT ini dengan baik. Laporan ini memuat uraian mengenai pembuatan proyek akhir kami, yaitu Secure Server Access Door. Proyek ini dirancang untuk membangun sistem Smart Door Lock yang mampu mengontrol akses fisik dan memonitor kondisi lingkungan data center secara real-time menggunakan RFID, BLE, dan WiFi, sambil memastikan operasi yang aman dan stabil melalui FreeRTOS, Mutex, serta notifikasi otomatis saat terjadi anomali. Kami menyampaikan terima kasih yang sebesar-besarnya kepada Bang Edgrant selaku asisten lab pendamping yang telah memberikan bimbingan, saran, dan dukungan selama pelaksanaan proyek akhir ini. Semua masukan yang telah diberikan sangat membantu dalam menyelesaikan proyek ini dengan baik. Kami menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kami sangat mengharapkan saran dan kritik yang membangun dari para pembaca demi peningkatan kualitas laporan ini di masa mendatang. Semoga laporan ini dapat memberikan manfaat, baik bagi kami sebagai penyusun, maupun bagi pembaca secara umum

Depok, December 8, 2025

Group 21

TABLE OF CONTENTS

CHAPTER 1.....	4
INTRODUCTION.....	4
1.1 PROBLEM STATEMENT.....	4
1.3 ACCEPTANCE CRITERIA.....	5
1.4 ROLES AND RESPONSIBILITIES.....	5
1.5 TIMELINE AND MILESTONES.....	5
CHAPTER 2.....	7
IMPLEMENTATION.....	7
2.1 HARDWARE DESIGN AND SCHEMATIC.....	7
2.2 SOFTWARE DEVELOPMENT.....	7
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	8
CHAPTER 3.....	9
TESTING AND EVALUATION.....	9
3.1 TESTING.....	9
3.2 RESULT.....	9
3.3 EVALUATION.....	10
CHAPTER 4.....	11
CONCLUSION.....	11

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Pengelolaan akses fisik pada ruangan data center maupun ruang kerja penting lainnya masih menimbulkan kesulitan karena tidak adanya sistem yang dapat memverifikasi identitas pengguna dan mencatat aktivitas keluar masuk secara real-time. Kondisi ini membuat ruang yang berisiko dapat dengan mudah diakses oleh pihak yang tidak berwenang, terutama ketika aktivitas berlangsung padat dan tidak ada petugas khusus yang mengawasinya.

Selain itu, pemantauan kondisi lingkungan seperti suhu ruangan server umumnya masih dilakukan secara manual sehingga tidak efektif. Kenaikan suhu yang terlalu cepat atau tidak terpantau dapat menyebabkan gangguan operasional hingga kerusakan perangkat. Dalam situasi nyata, keterlambatan mengetahui adanya perubahan suhu yang terlalu cepat dapat berdampak pada keamanan dan keberlangsungan sistem di dalam data center.

Di sisi lain, sistem keamanan yang digunakan sehari-hari seringkali tidak dapat menangani permasalahan secara bersamaan. Seperti membaca kartu identitas, menampilkan informasi, dan memantau sensor secara terus menerus. Jika hal tersebut tidak dikelola dengan tepat, dapat menyebabkan data visual tidak sinkron, respons lambat, atau konflik antar fungsi. Permasalahan-permasalahan ini menunjukkan perlunya sistem yang dapat mendukung pengawasan akses dan kondisi ruangan secara lebih terstruktur, akurat, dan responsif.

1.2 PROPOSED SOLUTION

Sistem Secure Server Access Door berbasis ESP32 merupakan salah satu solusi yang mampu mengelola akses fisik dan memantau kondisi lingkungan data center secara real-time. Sistem ini menggunakan metode berbasis RFID untuk mengidentifikasi pengguna yang sah, serta menyediakan akses darurat melalui Bluetooth Low Energy (BLE). Selain itu, sistem terhubung ke jaringan Wifi sehingga aktivitas akses dan kondisi dapat dipantau melalui Blynk.

Sistem ini menggunakan FreeRTOS agar seluruh fungsi dapat berjalan bersamaan dan stabil. FreeTOS membagi proses ke dalam beberapa task parallel seperti pembacaan RFID,

pemantauan suhu, dan koneksi cloud. Penggunaan Semaphore Mutex memastikan bahwa resource bersama seperti LCD dapat digunakan secara bergantian tanpa menimbulkan konflik tampilan atau data yang tumpang tindih.

Selain kontrol akses, sistem ini juga dilengkapi fitur monitoring suhu menggunakan sensor DHT11. Data suhu dikirimkan secara berkala ke dashboard Blynk, dan apabila terdeteksi suhu melebihi ambang batas yang ditentukan yaitu diatas 30 derajat, sistem akan mengirimkan alert otomatis melalui notifikasi Blynk. Dengan kombinasi autentikasi, monitoring lingkungan, pencatatan aktivitas, serta mekanisme peringatan dini, solusi ini diharapkan mampu meningkatkan keamanan fisik dan keandalan operasional ruang data center.

1.3 ACCEPTANCE CRITERIA

Kriteria penerimaan proyek ini adalah sebagai berikut:

1. Sistem menggunakan **RTOS** dan **Task Scheduling** dengan fungsi `xTaskCreate()` untuk mengelola task paralel yang menangani pembacaan RFID, monitoring suhu, dan koneksi jaringan.
2. Sistem menerapkan **Memory Management** dengan mengalokasikan ukuran stack yang spesifik (4096 atau 8192 bytes) pada setiap pembuatan task untuk memastikan penggunaan memori ESP32 yang efisien.
3. Sistem memanfaatkan mekanisme **Non-blocking Delay** menggunakan fungsi `vTaskDelay()` untuk mengatur durasi penguncian pintu (3 detik) dan interval pembacaan sensor suhu tanpa menghentikan proses lain.
4. Sistem mengimplementasikan sinkronisasi menggunakan **Semaphore Mutex** (`xSemaphoreTake` dan `xSemaphoreGive`) untuk mencegah konflik data (Race Condition) saat task RFID dan suhu mengakses LCD secara bersamaan.
5. Sistem menggunakan **protokol HTTP** untuk melakukan pengiriman data log akses ke Google Sheets serta protokol Bluetooth Low Energy (NimBLE) untuk menerima perintah buka pintu dari smartphone.
6. Sistem terintegrasi dengan **Blynk** untuk memantau suhu ruangan secara real-time, mengirimkan notifikasi overheating, dan menampilkan status akses pintu melalui dashboard aplikasi

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Role 1	Menyusun laporan dan merangkai hardware	Stevie Nathania Siregar
Role 2	Coding dan hardware	Mirza Adi Raffiansyah
Role 3	Hardware dan laporan	Ruben Kristanto
Role 4	Mengintegrasikan program dengan Google Sheets, merangkai hardware, menyusun laporan	Kharisma Aprilia

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Task	2 Dec	3 Dec	4 Dec	5 Dec	6 Dec	7 Dec	8 Dec
Hardware Design Completion							
Software Development							
Integration and Testing of Hardware and Software							
Final Product Assembly and Testing							

- Hardware Design Completion: mendesain hardware untuk sistem embedded telah diselesaikan, termasuk skematiknya.
- Software Development: Pembuatan kode assembly (software) oleh pengguna, yang berfokus pada tugas dan fungsionalitas tertentu
- Integration and Testing of Hardware and Software: komponen hardware dan software diintegrasikan dan diuji bersama untuk memastikan fungsionalitas yang benar.
- Final Product Assembly and Testing: Sistem akhir dirakit, diuji, dan diverifikasi untuk memenuhi kriteria penerimaan.

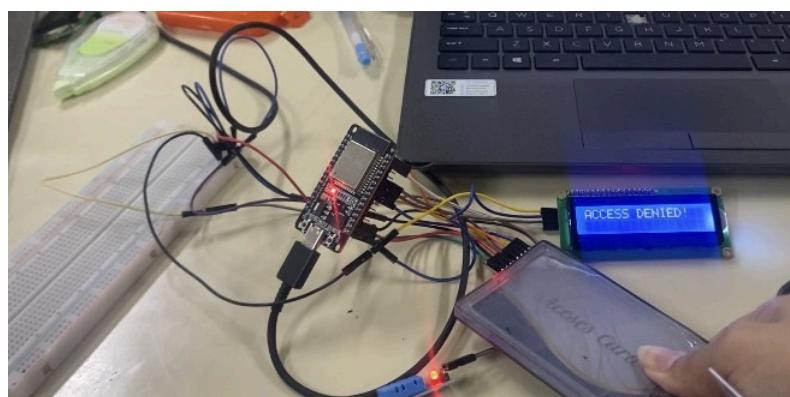
CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Design hardware yang digunakan untuk membuat sistem ini, melibatkan beberapa komponen utama sebagai berikut:

- 1) ESP32: Sebagai unit kendali utama (main controller). Komponen ini dipilih karena memiliki modul Wi-Fi dan Bluetooth terintegrasi yang mendukung komunikasi IoT dan akses BLE, serta cocok untuk menjalankan sistem operasi waktu nyata (FreeRTOS).
- 1) RFID-RC522 Module: Berfungsi sebagai perangkat input otentikasi untuk memindai kartu atau tag. Modul ini berkomunikasi dengan ESP32 menggunakan protokol SPI (Serial Peripheral Interface) untuk pembacaan data UID.
- 2) Sensor DHT11: Untuk memantau suhu lingkungan ruang server secara real-time. Sensor ini mengirimkan data suhu digital melalui protokol ke ESP32.
- 3) LCD I2C: Berfungsi sebagai antarmuka visual untuk menampilkan status akses dan suhu.
- 4) LED (Light Emitting Diode): Berperan sebagai simulator indikator kunci pintu. Indikator menyala (HIGH) menandakan mekanisme pintu terbuka, dan mati (LOW) menandakan pintu terkunci.



Untuk desain rangkaianya, sistem ini menerapkan konfigurasi pengkabelan yang terpusat pada ESP32 dengan pembagian jalur daya dan data yang terstruktur. Distribusi daya diatur menggunakan breadboard untuk memisahkan kebutuhan komponen VCC dan GND. Jalur tegangan 3.3V dialokasikan secara khusus untuk modul RFID-RC522 untuk mencegah kerusakan akibat tegangan berlebih, sedangkan jalur tegangan 5V (VIN) didistribusikan untuk LCD I2C dan sensor DHT 11.

Alur wiring dirancang berdasarkan protokol komunikasi masing-masing modul. Modul RFID dihubungkan ke jalur SPI standar ESP32, yaitu pin SDA(SS) ke GPIO 5, SCK ke GPIO 18, MOSI ke GPIO 23, dan MISO ke GPIO 19. Untuk tampilan visual, LCD dihubungkan ke jalur bus I2C pada pin GPIO 21 (SDA) dan GPIO 22 (SCL). Sementara itu, komponen digital lainnya terhubung ke pin Input/Output umum seperti sensor DHT11 mengirimkan data melalui GPIO 15, dan LED indikator menerima sinyal kontrol dari GPIO 2.

2.2 SOFTWARE DEVELOPMENT

Sistem Secure Server Access Door ini diawali dengan perancangan struktur program berbasis ESP32 yang menggabungkan modul keamanan dan modul monitoring dalam satu sistem. tahap awal mencakup penulisan kode untuk inisialisasi seluruh hardware seperti RFID RC522, sensor DHT11, LCD I2C, modul BLE, dan koneksi WiFi. Arsitektur multitasking menggunakan FreeRTOS dirancang sejak awal agar setiap fungsi sistem dapat berjalan secara bersamaan, sehingga proses autentikasi, pemantauan suhu, dan koneksi cloud tetap stabil tanpa saling mengganggu.

Pada tahap berikutnya, setiap task dikembangkan secara terpisah sesuai fungsinya. RFID digunakan untuk membaca UID kartu dan memverifikasi akses dan dilengkapi dengan pemberian dan penolakan akses yang ditampilkan di LCD dan juga dikirimkan ke server eksternal yaitu google sheets. Task suhu dirancang untuk melakukan pembacaan berkala dan akan memicu notifikasi otomatis jika suhu melebihi batas aman (< 30 derajat). Task Blynk dibuat untuk menjaga koneksi ke server cloud agar sistem tetap dapat dipantau secara real-time.

Mekanisme mutex ditambahkan untuk mengatur penggunaan LCD sebagai resource bersama, sehingga setiap task dapat menampilkan informasi tanpa terjadi konflik pada tampilan. Adapun code program sebagai berikut

```
// Definisi Pin Hardware & Protokol
#define PIN_DHT          15           // Pin Data Sensor Suhu
#define PIN_DOOR_LED     2            // Pin LED Indikator Pintu
#define PIN_RFID_SS       5            // Pin SPI Slave Select
#define PIN_RFID_RST      4            // Pin SPI Reset

// Library Integrasi Hardware, Network, & RTOS
#include <WiFi.h>
#include <HTTPClient.h>           // Protokol HTTP untuk Logging
#include <DHT.h>
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Inisialisasi Objek Hardware
DHT dht(PIN_DHT, DHT11);
MFRC522 mfrc522(PIN_RFID_SS, PIN_RFID_RST);
LiquidCrystal_I2C lcd(0x27, 16, 2);
SemaphoreHandle_t mutexBus;           // Sinkronisasi Resource

// URL Endpoint Integrasi Cloud
String GAS_URL = "https://script.google.com/macros/s/AKfyc.../exec";

void setup() {
    // Integrasi indikator (LED)
    pinMode(PIN_DOOR_LED, OUTPUT);

    // Integrasi Komunikasi Hardware (I2C & SPI)
    Wire.begin(); lcd.init(); lcd.backlight();
    SPI.begin(); mfrc522.PCD_Init();

    // Integrasi Modul Network (Wi-Fi Hardware)
    WiFi.begin("SSID", "PASS");

    // Integrasi Task Management (Software Logic)
    mutexBus = xSemaphoreCreateMutex();
    // Task HTTP, RFID, dan Sensor berjalan paralel
    xTaskCreate(taskRFID, "RFID", 8192, NULL, 1, NULL);
}

// Integrasi Software ke Hardware Network (HTTP Log)
```

```

void sendLogToGoogleSheets(String uid, String status) {
    if(WiFi.status() == WL_CONNECTED){
        HTTPClient http;
        http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

        String url = GAS_URL + "?uid=" + uid + "&status=" + status;
        http.begin(url);
        int httpCode = http.GET();
        http.end();
    }
}

// Logika Kontrol Hardware Fisik
void openDoorTask(String source, String uid) {
    if (xSemaphoreTake(mutexBus, (TickType_t)100) == pdTRUE) {
        lcd.clear(); lcd.print("ACCESS ALLOWED"); // Kirim data ke LCD
        digitalWrite(PIN_DOOR_LED, HIGH);

        // Panggil fungsi integrasi network
        sendLogToGoogleSheets(uid, "Allowed");

        xSemaphoreGive(mutexBus);/*
SMART DOOR LOCK ULTIMATE (LCD FIXED VERSION)
Features: RFID + BLE (NimBLE) + WiFi + Google Sheets Log + LCD + RTOS
Fixes: LCD initialization + I2C explicit init + Timing fixed
*/
    }

    // --- 1. SETTING BLYNK & WIFI ---
#define BLYNK_TEMPLATE_ID "TMPL6t6Qt2adf"
#define BLYNK_TEMPLATE_NAME "Smart Door"
#define BLYNK_AUTH_TOKEN "F71Ig8gxYu6zX5nGbkJ_IvRtkln-Deh"
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <NimBLEDevice.h>

char ssid[] = "iyeah";
char pass[] = "imaiot2023";
char auth[] = BLYNK_AUTH_TOKEN;

```

```

String GAS_URL =
"https://script.google.com/macros/s/AKfycbyYRo2qSSc4vaLwUelyrSg4qRzgXy_aZtxuL
INxRKd0xBi8YfZNgamiz0LHlgsrMpKt/exec";

#define PIN_DHT 15
#define PIN_DOOR_LED 2
#define PIN_RFID_SS 5
#define PIN_RFID_RST 4

DHT dht(PIN_DHT, DHT11);
MFRC522 mfrc522(PIN_RFID_SS, PIN_RFID_RST);
LiquidCrystal_I2C lcd(0x27, 16, 2);
SemaphoreHandle_t mutexBus;

String authorizedUIDs[] = {"07a83825", "7af84e1b"};
int authorizedCount = 2;
float currentTemp = 0.0;
bool lcdReady = false;

void sendLogToGoogleSheets(String uid, String status) {
    if(WiFi.status() == WL_CONNECTED){
        HTTPClient http;
        http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

        String url = GAS_URL + "?uid=" + uid + "&status=" + status;

        http.begin(url);
        int httpCode = http.GET();
        http.end();
        Serial.print("[HTTP] Log Sent: "); Serial.println(httpCode);
    } else {
        Serial.println("[HTTP] Error: WiFi not connected");
    }
}

void openDoorTask(String source, String uid) {
    if (xSemaphoreTake(mutexBus, (TickType_t)100) == pdTRUE) {
        lcd.clear();
        lcd.setCursor(0, 0); lcd.print("ACCESS ALLOWED");
        lcd.setCursor(0, 1); lcd.print("Via: " + source);

        digitalWrite(PIN_DOOR_LED, HIGH);
        Blynk.virtualWrite(V0, 1);

        sendLogToGoogleSheets(uid, "Allowed");
        Blynk.virtualWrite(V2, "Open by " + source);
    }
}

```

```

xSemaphoreGive(mutexBus);

vTaskDelay(3000 / portTICK_PERIOD_MS);

digitalWrite(PIN_DOOR_LED, LOW);
Blynk.virtualWrite(V0, 0);

if (xSemaphoreTake(mutexBus, (TickType_t)100) == pdTRUE) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Tap Your Card");
    xSemaphoreGive(mutexBus);
}
}

void denyAccessTask(String uid) {
    if (xSemaphoreTake(mutexBus, (TickType_t)100) == pdTRUE) {
        lcd.clear();
        lcd.setCursor(0, 0); lcd.print("ACCESS DENIED!");

        Blynk.logEvent("access_denied", "Unknown ID: " + uid);
        sendLogToGoogleSheets(uid, "Denied");

        xSemaphoreGive(mutexBus);
        vTaskDelay(2000 / portTICK_PERIOD_MS);

        if (xSemaphoreTake(mutexBus, (TickType_t)100) == pdTRUE) {
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Tap Your Card");
            xSemaphoreGive(mutexBus);
        }
    }
}

class MyCallbacks: public NimBLECharacteristicCallbacks {
    void onWrite(NimBLECharacteristic *pCharacteristic) {
        String value = pCharacteristic->getValue();
        if (value.length() > 0) {
            if (value == "OPEN") {
                Serial.println("[BLE] Command Received: OPEN");
                openDoorTask("BLE", "ADMIN_HP");
            }
        }
    }
}

```

```

};

void setupBLE() {
    NimBLEDevice::init("ServerDoor");
    NimBLEDevice::setPower(ESP_PWR_LVL_P9);
    NimBLESERVER *pServer = NimBLEDevice::createServer();
    NimBLEService *pService =
    pServer->createService("4fafc201-1fb5-459e-8fcc-c5c9c331914b");
    NimBLECharacteristic *pCharacteristic = pService->createCharacteristic(
        "beb5483e-36e1-4688-b7f5-ea07361b26a8",
        NIMBLE_PROPERTY::READ | NIMBLE_PROPERTY::WRITE
    );
    pCharacteristic->setCallbacks(new MyCallbacks());
    NimBLEDescrptor* pDescriptor = pCharacteristic->createDescriptor("2901");
    pDescriptor->setValue("Door Unlock Switch");

    pService->start();
    NimBLEAdvertising *pAdvertising = NimBLEDevice::getAdvertising();
    pAdvertising->addServiceUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b");

    NimBLEAdvertisementData scanResponse;
    scanResponse.setName("ServerDoor");
    pAdvertising->setScanResponseData(scanResponse);

    pAdvertising->start();
    Serial.println("[BLE] Bluetooth Ready!");
}

void taskRFID(void *pvParameters) {
    // Tunggu LCD ready
    while(!lcdReady) {
        vTaskDelay(100 / portTICK_PERIOD_MS);
    }

    for (;;) {
        if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
            String uidRaw = "";
            for (byte i = 0; i < mfrc522.uid.size; i++) {
                uidRaw += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "") +
                String(mfrc522.uid.uidByte[i], HEX);
            }

            Serial.print("[RFID] Card detected: ");
            Serial.println(uidRaw);
        }
    }
}

```

```

        bool authorized = false;
        for (int i = 0; i < authorizedCount; i++) {
            if (uidRaw == authorizedUIDs[i]) authorized = true;
        }

        if (authorized) openDoorTask("RFID", uidRaw);
        else denyAccessTask(uidRaw);

        mfrc522.PICC_HaltA(); mfrc522.PCD_StopCrypto1();
    }
    vTaskDelay(100 / portTICK_PERIOD_MS);
}
}

void taskTemp(void *pvParameters) {
    // Tunggu LCD ready
    while(!lcdReady) {
        vTaskDelay(100 / portTICK_PERIOD_MS);
    }

    dht.begin();

    for (;;) {
        float t = dht.readTemperature();
        if (!isnan(t)) {
            currentTemp = t;
            Blynk.virtualWrite(V1, currentTemp);

            if (xSemaphoreTake(mutexBus, (TickType_t)50) == pdTRUE) {
                lcd.setCursor(0, 1);
                lcd.print("Temp:");
                lcd.print(t, 1);
                lcd.print("C ");
                xSemaphoreGive(mutexBus);
            }

            if (currentTemp > 30.0) {
                Blynk.logEvent("overheat");
                Serial.println("[ALERT] Temperature > 30C!");
            }
        }
        vTaskDelay(3000 / portTICK_PERIOD_MS);
    }
}

void taskBlynk(void *pvParameters) {
    Blynk.begin(auth, ssid, pass);
}

```

```
for (;;) {
    Blynk.run();
    vTaskDelay(10 / portTICK_PERIOD_MS);
}

void setup() {
    Serial.begin(115200);
    delay(500);

    Serial.println("\n==== SMART DOOR LOCK STARTING ====");

    pinMode(PIN_DOOR_LED, OUTPUT);
    digitalWrite(PIN_DOOR_LED, LOW);

    mutexBus = xSemaphoreCreateMutex();

    // IMPORTANT: Initialize I2C explicitly for ESP32
    Wire.begin(); // SDA=21, SCL=22
    delay(100);

    Serial.println("[LCD] Initializing...");
    lcd.init();
    lcd.backlight();
    lcd.clear();

    // Welcome message
    lcd.setCursor(0, 0);
    lcd.print("Smart Door Lock");
    lcd.setCursor(0, 1);
    lcd.print("Starting...");
    Serial.println("[LCD] Welcome message displayed");

    delay(2000); // Biar kebaca dulu

    // Initialize SPI for RFID
    Serial.println("[RFID] Initializing...");
    SPI.begin();
    mfrc522.PCD_Init();
    mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);
    Serial.println("[RFID] Ready!");

    // Initialize BLE
    Serial.println("[BLE] Initializing...");
    setupBLE();

    // Ready state
```

```

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Tap Your Card");
lcd.setCursor(0, 1);
lcd.print("Ready!");
Serial.println("[LCD] Ready screen displayed");

delay(1000);

lcdReady = true; // Baru sekarang task lain boleh akses LCD

// Start RTOS tasks
Serial.println("[RTOS] Starting tasks...");
xTaskCreate(taskRFID, "RFID", 8192, NULL, 1, NULL);
xTaskCreate(taskTemp, "Temp", 4096, NULL, 1, NULL);
xTaskCreate(taskBlynk, "Blynk", 8192, NULL, 2, NULL);

Serial.println("== ALL SYSTEMS READY ==\n");
}

void loop() {
    // Empty - semua dihandle oleh RTOS tasks
}
    vTaskDelay(3000 / portTICK_PERIOD_MS);      // Timing Software
    digitalWrite(PIN_DOOR_LED, LOW);
}
}

```

2.3 HARDWARE AND SOFTWARE INTEGRATION

Pada sistem Secure Server Access Door, integrasi perangkat keras dan perangkat lunak dibangun dengan pengendalian mikrokontroler ESP32 yang menjalankan real-time operating system (FreeRTOS). Integrasi fisik dibangun melalui konfigurasi pin GPIO serta penggunaan berbagai protokol komunikasi, seperti SPI untuk membaca UID kartu RFID, I2C untuk menampilkan informasi pada LCD 16×2, serta komunikasi digital untuk mengendalikan LED sebagai indikator status pintu. Sensor DHT11 juga terhubung sebagai input suhu yang diproses oleh perangkat lunak untuk menampilkan nilai suhu pada layar LCD sekaligus memicu pengiriman log event ke aplikasi Blynk ketika temperatur melebihi threshold ($> 30^{\circ}\text{C}$).

Selain itu, sistem juga memanfaatkan modul Wi-Fi pada ESP32 untuk menghubungkan perangkat keras dengan cloud service. Hasil pembacaan RFID yang telah diverifikasi oleh logika perangkat lunak kemudian dikirimkan ke Google Sheets melalui HTTP Request menggunakan pustaka HTTPClient, sehingga pembacaan kartu pada sistem dapat terekam dan disimpan secara online.

Agar seluruh proses dapat berjalan secara bersamaan tanpa konflik, sistem kami memanfaatkan mekanisme Semaphore Mutex dan penjadwalan task berbasis FreeRTOS untuk mengatur akses bersama terhadap LCD maupun modul Wi-Fi. Dengan demikian, tugas seperti pembaruan tampilan, pengiriman data, dan pembacaan sensor dapat berlangsung secara simultan, menunjukkan keterpaduan antara perangkat keras, perangkat lunak, dan layanan jaringan yang terintegrasi dengan baik.

Berikut potongan kode yang merupakan implementasi integrasi ke hardware

```
// Definisi Pin Hardware & Protokol
#define PIN_DHT          15           // Pin Data Sensor Suhu
#define PIN_DOOR_LED     2            // Pin LED Indikator Pintu
#define PIN_RFID_SS      5            // Pin SPI
#define PIN_RFID_RST     4            // Pin SPI Reset

// Library Integrasi Hardware, Network, & RTOS
#include <WiFi.h>
#include <HTTPClient.h>           // Protokol HTTP untuk Logging
#include <DHT.h>
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Inisialisasi Objek Hardware
DHT dht(PIN_DHT, DHT11);
MFRC522 mfrc522(PIN_RFID_SS, PIN_RFID_RST);
LiquidCrystal_I2C lcd(0x27, 16, 2);
SemaphoreHandle_t mutexBus;        // Sinkronisasi Resource

// URL Endpoint Integrasi Cloud
String GAS_URL = "https://script.google.com/macros/s/AKfyc.../exec";

void setup() {
```

```

// Integrasi indikator (LED)
pinMode(PIN_DOOR_LED, OUTPUT);

// Integrasi Komunikasi Hardware (I2C & SPI)
Wire.begin(); lcd.init(); lcd.backlight();
SPI.begin(); mfrc522.PCD_Init();

// Integrasi Modul Network (Wi-Fi Hardware)
WiFi.begin("SSID", "PASS");

// Integrasi Task Management (Software Logic)
mutexBus = xSemaphoreCreateMutex();
// Task HTTP, RFID, dan Sensor berjalan paralel
xTaskCreate(taskRFID, "RFID", 8192, NULL, 1, NULL);
}

// Integrasi Software ke Hardware Network (HTTP Log)
void sendLogToGoogleSheets(String uid, String status) {
    if(WiFi.status() == WL_CONNECTED){
        HTTPClient http;
        http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

        String url = GAS_URL + "?uid=" + uid + "&status=" + status;
        http.begin(url);
        int httpCode = http.GET();
        http.end();
    }
}

// Logika Kontrol Hardware
void openDoorTask(String source, String uid) {
    if (xSemaphoreTake(mutexBus, (TickType_t)100) == pdTRUE) {
        lcd.clear(); lcd.print("ACCESS ALLOWED"); // Kirim data ke LCD
        digitalWrite(PIN_DOOR_LED, HIGH);

        sendLogToGoogleSheets(uid, "Allowed");

        xSemaphoreGive(mutexBus);
        vTaskDelay(3000 / portTICK_PERIOD_MS);      // Timing Software
        digitalWrite(PIN_DOOR_LED, LOW);
    }
}

```

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Pengujian sistem Secure Server Access Door ini dilakukan untuk memastikan bahwa integrasi perangkat keras ESP32 dan logika perangkat lunak berbasis RTOS berjalan sesuai spesifikasi yang dirancang. Tahap awal pengujian difokuskan pada mekanisme keamanan berbasis RFID. Pengujian dilakukan dengan memindai kartu yang tidak terdaftar pada modul RC522. Hasil menunjukkan sistem berhasil menolak akses tersebut dengan menampilkan pesan "ACCESS DENIED!" pada layar LCD dan membiarkan indikator LED tetap mati (terkunci), serta mengirimkan notifikasi peringatan "Unknown ID" ke aplikasi Blynk secara real-time. Hal ini berguna untuk memastikan bahwa logika penolakan akses berfungsi dengan baik.

Selanjutnya, pengujian validasi dilakukan menggunakan kartu RFID yang telah terdaftar dalam sistem. Sistem berhasil mengenali UID, lalu mengubah tampilan LCD menjadi "ACCESS ALLOWED", dan mengaktifkan LED indikator pintu selama 3 detik sebelum kembali terkunci.

Selain kontrol akses, pengujian untuk pemantauan lingkungan server dilakukan dengan mengamati pembacaan sensor DHT11. Pada kondisi normal, LCD menampilkan suhu ruangan secara berkala di baris kedua LCD karena mekanisme sinkronisasi Mutex. Namun, ketika sensor diarahkan ke benda hangat yang memicu kenaikan suhu melampaui ambang batas 30°C, sistem berhasil memicu event "Overheat" pada dashboard Blynk, membuktikan fungsi monitoring berjalan paralel tanpa mengganggu fungsi akses pintu.

Terakhir, verifikasi dilakukan terhadap sistem logging data. Setelah serangkaian percobaan akses, database Google Sheets diperiksa dan menunjukkan bahwa data waktu, UID, dan status (Denied/Allowed) telah tercatat secara akurat melalui protokol HTTP Request. Hasil keseluruhan pengujian menunjukkan bahwa sistem beroperasi stabil dengan manajemen task yang efisien dan sesuai dengan desain integrasi hardware-software yang diharapkan.

3.2 RESULT

Berdasarkan implementasi kode program dan integrasi perangkat keras yang telah dilakukan, sistem Secure Server Access Door berhasil beroperasi sesuai harapan dengan rincian sebagai berikut

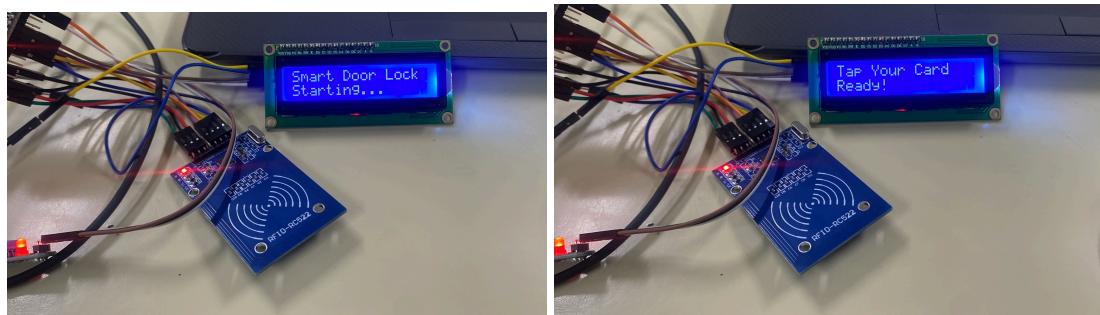


Fig 1. Serial Monitor berhasil menampilkan log

Saat perangkat pertama kali dinyalakan, fungsi `setup()` berhasil mengeksekusi inisialisasi perangkat keras secara berurutan. Layar LCD berhasil menampilkan pesan pembuka "Smart Door Lock / Starting..." selama 2 detik sebelum masuk ke mode siap (idle) dengan tampilan "Tap Your Card / Ready!". Mekanisme boolean flag (`lcdReady`) terbukti efektif mencegah task sensor suhu mengakses LCD sebelum inisialisasi selesai, sehingga tidak terjadi glitch pada awal sistem menyala.



Fig 2. LCD berhasil menampilkan data suhu

Pada kondisi *idle*, baris kedua LCD menampilkan data suhu yang diperbarui setiap 3 detik. Mekanisme *Semaphore Mutex* berhasil mengatur antrean akses LCD, sehingga tulisan suhu tidak menimpa pesan status pintu saat akses sedang berlangsung.

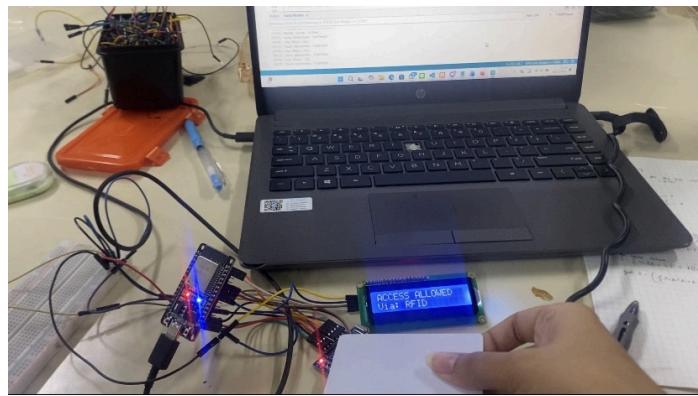


Fig 3. Hasil uji scan authorized UID

Ketika kartu dengan UID terdaftar di sistem (07a83825 atau 7af84e1b) dibaca, sistem merespons dengan LCD menampilkan pesan "ACCESS ALLOWED" dengan keterangan sumber "Via: RFID". LED indikator menyala (HIGH) selama 3 detik, mendemonstrasikan terbukanya kunci pintu, sebelum kembali mati secara otomatis.

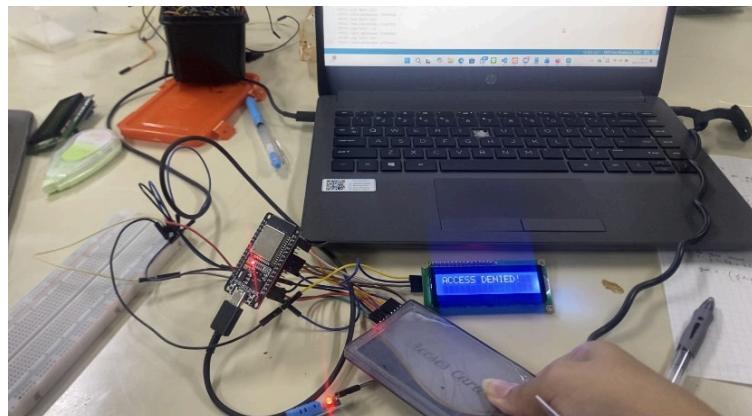


Fig 4. Hasil uji scan unauthorized UID

Saat kartu yang tidak terdaftar dipindai, sistem menampilkan pesan "ACCESS DENIED!" pada LCD dan LED tetap mati. Sistem juga secara langsung memicu *event* keamanan ke aplikasi Blynk.

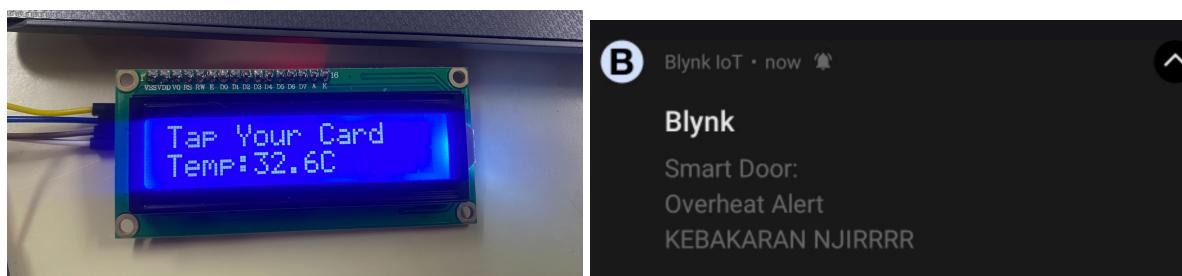


Fig 5. Event notifikasi Blynk

Saat nilai pembacaan sensor mencapai angka $> 30.0^{\circ}\text{C}$, sistem secara responsif mendeteksi kondisi tersebut dalam siklus task taskTemp dan memicu fungsi Blynk.logEvent("overheat"). Hasilnya, notifikasi peringatan (Push Notification) berhasil muncul pada layar smartphone.

	A	B	C	D	E	F
1	Waktu	UID	Status			
2	08/12/2025 14:20:07	07a83825	Allowed			
3	08/12/2025 14:20:47	7af84e1b	Allowed			
4	08/12/2025 14:21:11	1597cc48	Denied			
5	08/12/2025 14:21:57	1597cc48	Denied			
6	08/12/2025 14:30:18	7af84e1b	Allowed			
7	08/12/2025 14:33:08	7af84e1b	Allowed			
8	08/12/2025 14:37:07	7af84e1b	Allowed			
9	08/12/2025 14:51:13	7af84e1b	Allowed			
10	08/12/2025 14:51:23	7af84e1b	Allowed			
11	08/12/2025 14:54:15	7af84e1b	Allowed			
12	08/12/2025 14:58:44	7af84e1b	Allowed			
13	08/12/2025 15:21:17	7af84e1b	Allowed			
14	08/12/2025 15:21:37	07a83825	Allowed			
15	08/12/2025 15:22:13	07a83825	Allowed			
16	08/12/2025 15:23:21	07a83825	Allowed			
17	08/12/2025 15:23:33	1597cc48	Denied			
18	08/12/2025 15:23:53	07a83825	Allowed			

Fig 6. Integrasi data logging Google Sheets

Fitur konektivitas berhasil menghubungkan perangkat ESP32 dengan layanan Google Sheets dimana setiap kali akses pintu terjadi baik Allowed maupun Denied, Serial Monitor mencatat respons HTTP Code 200 atau 302 yang menandakan pengiriman data sukses. Pada sisi server, data UID, status, dan waktu akses berhasil terekam secara otomatis di Google Sheets.

3.3 EVALUATION

Proyek secure server access door ini dirancang sebagai solusi keamanan ganda, menggabungkan pengamanan fisik dengan pemantauan lingkungan. Sistem ini memanfaatkan ESP32 sebagai otak, yang bertugas mengintegrasikan masukan dari modul RFID untuk kontrol akses, dan sensor DHT11 untuk data suhu dan kelembaban. Narasi keberhasilan proyek dimulai dengan validasi akses; modul RFID bekerja secara efisien, memungkinkan otorisasi masuk yang cepat dan mencatat setiap upaya akses. Sistem ini berhasil membuktikan bahwa ia dapat menggantikan kunci mekanis dengan mekanisme otentifikasi digital yang lebih responsif dan mudah dikelola.

Namun, tidak hanya sekadar mengunci pintu. Keunggulan penting dari sistem ini adalah peran DHT11 dalam menjaga integritas operasional server. Data suhu dan kelembaban yang ditampilkan pada LCD memberikan gambaran kondisi lingkungan secara real-time, yang sangat penting untuk mencegah overheating peralatan sensitif. Integrasi dengan aplikasi Blynk merupakan langkah krusial berikutnya. Melalui Blynk, sistem melampaui keamanan lokal, memungkinkan notifikasi jarak jauh segera dikirimkan ke administrator jika terjadi anomali, baik itu upaya akses yang tidak sah maupun lonjakan suhu yang berbahaya. Meskipun demikian, selama pengujian, kami menyadari bahwa akurasi sensor DHT11 yang relatif mendasar dan ketergantungan sistem pada stabilitas koneksi Wi-Fi adalah tantangan teknis yang perlu dipertimbangkan untuk penerapan jangka panjang.

CHAPTER 4

CONCLUSION

Sebagai penutup, proyek Smart Server Access Door ini adalah sebuah demonstrasi yang sukses mengenai bagaimana teknologi IoT berbiaya rendah dapat diterapkan untuk mengatasi kebutuhan keamanan infrastruktur kritis. Kami berhasil menyatukan komponen hardware (sensor, indikator, antarmuka) dengan kendali mikrokontroler ESP32 yang berkemampuan tinggi, menghasilkan sebuah sistem yang kohesif. Pelajaran utama yang didapat dari proyek ini adalah pemahaman praktis tentang kompromi antara kinerja, keandalan nirkabel, dan efisiensi daya. Meskipun sistem ini telah mencapai tujuan awalnya, potensi pengembangannya sangat luas. Ke depannya, sistem ini dapat diperluas dengan menambahkan log data historis yang lebih rinci, atau mengintegrasikan sensor kualitas udara yang lebih canggih, menjadikannya bukan hanya alat keamanan, tetapi juga alat manajemen fasilitas yang komprehensif. Proyek ini membuktikan bahwa konsep keamanan IoT yang berlapis adalah masa depan pengamanan ruang server.

REFERENCES

- [1] [5] Espressif Systems, “**ESP32 Series Datasheet**,” Version 5.2, Espressif Systems, Shanghai, China, 2025. [Online]. Available: https://documentation.espressif.com/esp32_datasheet_en.pdf. [Accessed: Dec. 8, 2025].
- [2] Blynk Team, “**Blynk Documentation: Getting Started with ESP32**,” *Blynk IoT Platform Documentation*, [Online]. Available: <https://docs.blynk.io/en/getting-started/what-do-i-need>. [Accessed: Dec. 8, 2025].
- [3] R. N. Tutorials, “**Security Access using MFRC522 RFID Reader with Arduino**,” *Random Nerd Tutorials*, [Online]. Available: <https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/>. [Accessed: Dec. 8, 2025].
- [4] C. Basics, “**How to Set Up the DHT11 Humidity Sensor on an Arduino**,” *Circuit Basics*, [Online]. Available: <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>. [Accessed: Dec. 8, 2025].

APPENDICES

Appendix: Documentation

