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

SMART IDEA CLOCK

GROUP 18

ACHMAD ZAIDAN LAZUARDY	**2206059793**
BERES BAKTI P. S.	**2206817585**
DARMAWAN HANIF	**2206829175**
DIMAS DERMAWAN	**2206059654**

PREFACE

Di era modern ini, perangkat Internet of Things (IoT) telah menjadi bagian penting dari kehidupan sehari-hari. Salah satu aplikasi IoT yang banyak dikembangkan adalah perangkat rumah pintar (smart home). Perangkat ini bertujuan untuk memberikan kenyamanan, efisiensi energi, dan kemudahan dalam mengelola aktivitas rumah tangga.

Proyek Smart Idea Clock adalah pengembangan inovatif dari jam alarm konvensional menjadi perangkat IoT multifungsi. Selain berfungsi sebagai jam alarm, perangkat ini dilengkapi dengan fitur-fitur yang dapat memberikan informasi pada lingkungan secara real-time, seperti suhu, kelembaban, dan tingkat polusi udara. Selain itu, perangkat ini juga dapat mengontrol beberapa peralatan rumah tangga, seperti kipas angin dan lampu tidur melalui sistem yang berbasis web.

Dengan menggunakan teknologi ESP32 sebagai kendali dan jaringan mesh untuk konektivitas antar perangkat, Smart Idea Clock dirancang untuk memenuhi kebutuhan pengguna akan alat yang modern, efisien, dan mudah digunakan. Proyek ini diharapkan dapat menjadi solusi inovatif dalam mendukung pengelolaan aktivitas rumah tangga berbasis teknologi.

Depok, December 9, 2024

Group 18

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Dalam kehidupan modern, kebutuhan akan perangkat yang dapat memberikan kemudahan, kenyamanan, dan efisiensi menjadi semakin penting, terutama dalam lingkungan rumah tangga. Jam alarm konvensional, meskipun fungsional, memiliki keterbatasan dalam memberikan manfaat di luar fungsi dasar pengingat waktu. Di sisi lain, perangkat IoT telah membuka peluang untuk mengintegrasikan berbagai fungsi ke dalam satu perangkat yang terhubung.

Namun, terdapat beberapa tantangan yang dihadapi dalam mengembangkan perangkat seperti ini:

1. Kurangnya Informasi Lingkungan yang Komprehensif

   Perangkat jam alarm konvensional umumnya tidak mampu memberikan informasi lingkungan seperti suhu, kelembaban, atau tingkat polusi udara. Informasi ini penting untuk membantu pengguna membuat keputusan yang lebih baik terkait kenyamanan dan kesehatan.

2. Minimnya Integrasi dengan Perangkat Rumah Tangga Lain

   Perangkat rumah tangga seperti lampu tidur dan kipas angin sering kali memerlukan pengaturan manual, yang dapat menjadi tidak efisien. Dibutuhkan sistem yang memungkinkan kontrol perangkat ini secara terpusat dan otomatis.

3. Kesulitan dalam Pengaturan Alarm dan Data Real-Time

   Sistem pengaturan alarm yang ada seringkali terbatas dalam fleksibilitas dan tidak terintegrasi dengan data real-time. Hal ini mengurangi efektivitas perangkat dalam memberikan pengalaman pengguna yang optimal.

4. Keterbatasan Jaringan dan Interoperabilitas Perangkat

   Dalam sistem IoT, memastikan konektivitas yang andal antara berbagai perangkat sering kali menjadi tantangan. Sistem yang mudah diterapkan dan memungkinkan komunikasi antar-node secara efektif sangat diperlukan.

Maka dari itu, Smart Idea Clock ada bertujuan untuk menjawab permasalahan ini dengan mengembangkan perangkat IoT yang menggabungkan fungsi jam, alarm, dan sistem pengelolaan data lingkungan. Proyek ini akan memberikan solusi inovatif yang tidak hanya fungsional tetapi juga terintegrasi dengan baik untuk mendukung kebutuhan rumah pintar.

1.2 PROPOSED SOLUTION

Untuk mengatasi masalah yang telah diidentifikasi, proyek Smart Clock Idea menawarkan solusi perangkat IoT yang dirancang untuk memberikan kemudahan, kenyamanan, dan informasi lingkungan secara real-time. Solusi ini mengintegrasikan teknologi IoT, sensor lingkungan, dan kontrol perangkat rumah tangga dengan pendekatan berikut:

Penggunaan ESP32 sebagai Pusat Kendali dan Node Pengirim Data

- ESP1 digunakan sebagai pusat kendali utama yang mengelola fungsi perangkat, termasuk pengumpulan dan pemrosesan data dari sensor, pengendalian perangkat rumah tangga, dan koneksi ke dashboard web.
- ESP2 bertugas untuk membaca data dari sensor lingkungan dan mengirimkan data tersebut ke ESP1 menggunakan protokol komunikasi HTTP atau MQTT.

Integrasi Sensor untuk Informasi Lingkungan

- **DHT11** digunakan untuk membaca suhu dan kelembaban ruangan.
- **MQ135** digunakan untuk mendeteksi kualitas udara di ruangan, termasuk konsentrasi gas tertentu.
- Data dari kedua sensor ini ditampilkan secara real-time pada dashboard web, memberikan informasi yang berguna kepada pengguna tentang kondisi lingkungan.

Dashboard Web untuk Pengelolaan Data dan Alarm

- Dashboard berbasis web memberikan antarmuka yang user-friendly untuk memantau data sensor secara real-time dan mengatur alarm.

- Fitur alarm memungkinkan pengguna untuk mengatur waktu dan memicu buzzer sesuai kebutuhan, menjadikannya lebih fleksibel dibandingkan alarm konvensional.

Efisiensi Energi dan Skalabilitas

- Perangkat dirancang dengan mempertimbangkan efisiensi energi untuk penggunaan jangka panjang.
- Sistem berbasis jaringan lokal memungkinkan perangkat untuk diintegrasikan atau ditingkatkan dengan fitur tambahan di masa depan.

Dengan solusi ini, **Smart Clock Idea** tidak hanya berfungsi sebagai jam alarm tetapi juga menjadi pusat informasi lingkungan yang cerdas. Solusi ini dirancang untuk memberikan pengalaman pengguna yang efisien dan relevan dengan kebutuhan rumah modern.

1.3 ACCEPTANCE CRITERIA

Kriteria penerimaan untuk proyek ini adalah sebagai berikut:

1. Dapat menampilkan jam dan tanggal melalui dashboard web.
2. Dapat membaca dan menampilkan suhu serta kelembaban ruangan.
3. Dapat membaca kualitas udara menggunakan sensor MQ135.
4. Memiliki fitur alarm yang dapat diatur dan memicu buzzer pada waktu tertentu.

1.4 ROLES AND RESPONSIBILITIES

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

Roles	Responsibilities	Person
Main Program	Membuat main program untuk kedua ESP32, menyatukan semua program yang telah dibuat semua anggota agar project berjalan	Dimas Dermawan
Sensor Program	Merancang Program untuk setiap sensor digunakan agar berjalan sesuai kerja yang kita inginkan dari setiap sensor	Darmawan Hanif / Beres Bakti P. S.
Web Socket	Membuat sebuah Web Socket yang dapat menunjukkan data sensor secara realtime dan untuk mengatur sensor atau alarm secara manual	Dimas Dermawan
Display	Membuat program untuk display untuk menampilkan jam dan informasi lainnya seperti kualitas udara	Achmad Zaidan Lazuardy
Data transfer	Merancang pengiriman data dari ESP32 yang ke-2 ke ESP32 yang pertama yang mengolah semua data yang diterima	Beres Bakti P. S.

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Activity	Start	End	Description
Discuss ideas	24/11/ 2024	24/11/ 2024	Diskusi dan penentuan ide untuk proyek, termasuk konsep awal perangkat.
Software Development	08/12/2024	09/12/2024	Memulai pengembangan program untuk masing-masing komponen perangkat lunak.
Integration and Testing of Hardware and Software	09/12/2024	10/12/2024	Merangkai perangkat keras dan perangkat lunak, serta melakukan pengujian awal
Final and Testing	110/12/2024	10/12/2024	Merakit sistem final, menguji, dan memastikan sudah sesuai dengan acceptance criteria.

CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Komponen yang dibutuhkan:

- DHT11
- ESP32
- Buzzer
- MQ135
- Breadboard
- Kabel jumper

Perangkat Smart Idea Clock dibuat untuk mengintegrasikan beberapa komponen utama untuk membuat perangkat IoT berfungsi secara maksimal dan multifungsi. Setiap komponen memiliki perannya masing-masing yang memiliki fiturnya sendiri, seperti pengumpulan data lingkungan, pengendalian perangkat rumah tangga, dan komunikasi antar-node.

- ESP32 (2 unit)
    - Root Node: Mengelola data sensor, menampilkan informasi pada LCD, dan berkomunikasi dengan slave node.
    - Slave Node: Mengontrol perangkat tambahan, seperti lampu tidur atau kipas angin.
- Sensor Lingkungan:

    DHT11: Mengukur suhu dan kelembaban ruangan.

- Output Aktuator:

    Buzzer: Memberikan notifikasi alarm.

2.2 SOFTWARE DEVELOPMENT

Pengembangan software pada proyek Smart Clock Idea bertujuan untuk mengintegrasikan berbagai fungsi perangkat IoT, seperti pengumpulan data dari sensor, kontrol perangkat rumah tangga, komunikasi antar node ESP32, serta interface pengguna melalui dashboard web. Program untuk seluruh integrasi perangkat IoT ini dirancang menggunakan bahasa pemrograman C++ dengan platform Arduino IDE, beserta library tambahan untuk mendukung fungsi-fungsi tertentu, seperti pengelolaan sensor, komunikasi antar-ESP32, dan tampilan data pada dashboard.

- **Modul Sensor dan Data**

  Modul ini bertanggung jawab untuk membaca data dari sensor DHT11 dan MQ135. Sensor MQ135 mendeteksi kualitas udara, sementara DHT11 mengukur suhu dan kelembaban ruangan. Data yang diperoleh dari kedua sensor ini kemudian dikirimkan ke root node (ESP1) untuk diproses dan ditampilkan secara real-time pada dashboard web.
- **Modul Jaringan Komunikasi**

  Untuk komunikasi antara ESP32, kami menggunakan protokol HTTP atau MQTT, menggantikan jaringan mesh. ESP2 berfungsi untuk mengirimkan data sensor ke ESP1 dengan tingkat keberhasilan 100%. ESP1 bertindak sebagai pengolah utama data yang diterima, kemudian menampilkan data tersebut pada dashboard web secara real-time.
- **Modul Alarm dan Kontrol Perangkat**

  Modul ini memungkinkan pengguna untuk mengatur alarm melalui dashboard web, yang kemudian akan mengaktifkan buzzer sesuai dengan waktu yang telah ditentukan.
- **Modul Dashboard Web**

  Dashboard web menyediakan antarmuka yang memungkinkan pengguna untuk memantau data lingkungan secara real-time, mengatur alarm. Data sensor dari MQ135 dan DHT11 ditampilkan pada dashboard, memberikan informasi tentang kualitas udara, suhu, dan kelembaban secara langsung. Tampilan dashboard dirancang agar user-friendly, sehingga pengguna dapat dengan mudah mengakses dan mengelola perangkat.

2.3 HARDWARE AND SOFTWARE INTEGRATION

- ESP1 Code :

```
#include <Arduino.h>
#include <esp_now.h>
#include <WiFi.h>
#include <WiFiManager.h>
#include <WebSocketsServer.h>
#include <ESPAsyncWebServer.h>
#include <ESPmDNS.h>
#include <ArduinoJson.h>

#define MAX_ALARM 5
#define PIN_POLLUTION 17

/* ===== Constant Definitions ==== */
const String COUNTRY_CODES[13] = {
  "cn",  // China
  "hk",  // Hong Kong
  "id",  // Indonesia
  "jp",  // Japan
  "kr",  // Korea
  "my",  // Malaysia
  "ph",  // Philippines
  "ps",  // Palestinian Territory
  "sa",  // Saudi Arabia
  "sg",  // Singapore
  "th",  // Thailand
  "tw",  // Taiwan
  "vn",  // Vietnam
```

```
};


typedef struct SensorData {
  float T;
  float H;
} SensorData;


typedef struct AlarmItem {
  int8_t index;
  char label[100];
  char time[10];
} AlarmItem;
/* ===== Constant Definitions ==== */



/* ===== Function Definitions ==== */
// FreeRTOS
void taskUpdateTime(void* parameters);
void taskAirPollutionSensor(void* parameters);

// ESP-NOW
void OnDataRecv(const esp_now_recv_info_t* info, const uint8_t*
incomingData, int len);

// Web Server
void webSocketEvent(uint8_t num, WStype_t type, uint8_t* payload,
size_t length);

void webGetAlarm();
void webAddAlarm();
```

```
void webDeleteAlarm();

void  parseNewAlarm(AsyncWebServerRequest*  req,  uint8_t*  data,
size_t len, size_t index, size_t total);

void parseDeleteAlarm(AsyncWebServerRequest* req, uint8_t* data,
size_t len, size_t index, size_t total);

void webDashboard();


void startWebServer();

/* ===== Function Definitions ==== */



/* ===== Variable Declarations ==== */

// FreeRTOS TaskHandle

TaskHandle_t taskHandleUpdateTime;

TaskHandle_t taskHandleAirPollutionSensor;


String currentCC = COUNTRY_CODES[2];  // Indonesia

const char ntpServer[] = "id.pool.ntp.org";


String currentTime;

AlarmItem alarmLists[MAX_ALARM];

int alarmIndex = 0;


SensorData sensorData;


WebSocketsServer webSocketTime = WebSocketsServer(81);

WebSocketsServer webSocketDht = WebSocketsServer(82);

WebSocketsServer webSocketPollution = WebSocketsServer(83);

AsyncWebServer server(80);

/* ===== Variable Declarations ==== */
```

```
/* ===== Setup ==== */
void setup() {
  /* Starting Serial Monitor */
  Serial.begin(115200);
  delay(1000);

  for (int i = 0; i < MAX_ALARM; i++) {
    AlarmItem* tmp = (AlarmItem*)malloc(sizeof(AlarmItem));

    tmp->index = -1;
    stpcpy(tmp->label, "");
    stpcpy(tmp->time, "");

    alarmLists[i] = *tmp;

    free(tmp);
  }

  /* ====== Wifi Setup ====== */
  WiFi.mode(WIFI_STA);               // Stationary mode
   WiFi.setChannel(WiFi.channel());   // Channel is important for
ESP-NOW

  WiFiManager wifiManager;  // Using WiFiManager for dynamic WiFi
  wifiManager
    .autoConnect("SmartClock ESP Main");  // WiFi AP with name
  /* ====== Wifi Setup ====== */
```

```
  /* ====== DNS Setup ====== */
  // Akses ke web lewat http://smartclock18.local
  // Kalo gabisa akses, pastikan menggunakan DNS
  // Cloudflare 1.1.1.1 pada jaringan
  if (!MDNS.begin("smartclock18")) {
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(500);
    }
  }
            Serial.println("Access      the      dashboard      via:
http://smartclock18.local");
  /* ====== DNS Setup ====== */



  /* Config NTP Server for clock */
  configTime(25200, 0, ntpServer);
   xTaskCreate(taskUpdateTime, "Task Update Time", 2048, NULL, 1,
&taskHandleUpdateTime);


  /* Config MQ-2 */
  pinMode(PIN_POLLUTION, INPUT);
  analogSetAttenuation(ADC_11db);
  Serial.println("Warming up Sensor");
   xTaskCreate(taskAirPollutionSensor, "Task Update Time", 2048,
NULL, 1, &taskHandleAirPollutionSensor);


  /* ====== ESP-NOW Setup ====== */
  if (esp_now_init() != ESP_OK) {
```

```
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  // Register callback function to receive ESP-NOW data
  esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  Serial.println("ESP NOW Initialized!");
  /* ====== ESP-NOW Setup ====== */



  /* ====== WebSocket Setup ====== */
  webSocketTime.begin();
  webSocketDht.begin();
  webSocketPollution.begin();


  startWebServer();
}


void loop() {
  webSocketTime.loop();
  webSocketDht.loop();
  webSocketPollution.loop();
}


void taskUpdateTime(void* parameters) {
  while (1) {
    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) {
      Serial.println("Failed to obtain time");
      return;
```

```
    }

    char buffer[50];
    strftime(buffer, sizeof(buffer), "%H:%M:%S", &timeinfo);

    currentTime = String(buffer);

    webSocketTime.broadcastTXT(currentTime);

    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }

  vTaskDelete(NULL);
}

void taskAirPollutionSensor(void* parameters) {
  while (1) {
    int gasValue = analogRead(PIN_POLLUTION);

    String value = std::to_string(gasValue).c_str();

    webSocketPollution.broadcastTXT(value);
    vTaskDelay(300 / portTICK_PERIOD_MS);
  }

  vTaskDelete(NULL);
}

void webDashboard() {
```

```
  server.on("/", HTTP_GET, [](AsyncWebServerRequest* request) {
    request->send_P(200, "text/html", R"rawliteral(
      <!DOCTYPE html>
      <html lang="id">

      <head>
        <meta charset="UTF-8">
              <meta  name="viewport"  content="width=device-width,
initial-scale=1.0">

        <title>SmartClock</title>

                                                                <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500
;600;700&display=swap" rel="stylesheet">
                                                                <link
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/c
ss/all.min.css" rel="stylesheet">

        <style>
          :root {
            --color-lightest: #D3F1DF;
            --color-light: #85A98F;
            --color-medium: #5A6C57;
            --color-dark: #525B44;
            --font-family: 'Inter', sans-serif;
          }

          * {
            margin: 0;
            padding: 0;
```

```
  box-sizing: border-box;
  font-family: var(--font-family);
  /* border: black solid 1px; */
}

*::selection {
  background-color: var(--color-dark);
  color: var(--color-lightest);
}

body {
  background-color: var(--color-lightest);
  color: var(--color-dark);
  line-height: 1.6;
}

.button {
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 0.5rem;
  font-weight: 500;
  cursor: pointer;
  transition: all 0.3s ease;
  border: var(--color-dark) solid 2px;
}

.button-outline {
  background-color: var(--color-lightest);
  color: var(--color-dark);
```

```
    border: var(--color-dark) solid 2px;
  }

  .button-outline:hover {
    background-color: var(--color-medium);
    color: white;
  }

  .modal {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
  }

  .modal-content {
    background-color: white;
    padding: 2rem;
    border-radius: 1rem;
    width: 90%;
    max-width: 500px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
```

```
.modal-title {
  font-size: 1.5rem;
  font-weight: 600;
  margin-bottom: 1.5rem;
  color: var(--color-dark);
}

.form-group {
  margin-bottom: 1rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  color: var(--color-medium);
}

.form-control {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid var(--color-light);
  border-radius: 0.5rem;
  font-size: 1rem;
}

.modal-footer {
  display: flex;
  justify-content: flex-end;
```

```
            gap: 1rem;
            margin-top: 1.5rem;
        }

        @media (max-width: 768px) {
            .grid {
                grid-template-columns: 1fr;
            }
        }
    </style>
</head>

<body>
    <style id="navbar-style">
        .navbar {
            width: 100%;
            background-color: var(--color-light);
            padding: 1rem 2rem;
            border-bottom: var(--color-dark) solid 2px;
        }

        .navbar-content {
            max-width: 1200px;
            margin: 0 auto;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
```

```
        .navbar-logo {
          font-size: 2rem;
          font-weight: 700;
          letter-spacing: -3.5px;
          color: white;
          background-color: var(--color-medium);
          padding: 2px 10px;
          border-radius: 4px;
        }

        .navbar-buttons {
          display: flex;
          gap: 1rem;
        }

        .navbar-button {
          padding: 0.5rem 1rem;
          border-radius: 0.5rem;
          font-weight: bold;
          font-size: 1rem;
          cursor: pointer;
          transition: all 0.3s ease;
        }
      </style>
      <nav class="navbar">
        <div class="navbar-content">
          <div class="navbar-logo">SmartClock</div>
          <div class="navbar-buttons">
```

```
                <!-- <button class="navbar-button button-outline"
onclick="modalOpen('timezone-modal')">Timezone</button> -->
                    <button class="navbar-button button-outline"
onclick="modalOpen('alarm-modal')">New Alarm</button>
            </div>
          </div>
        </nav>

        <style id="container style">
          .container {
            width: 100%;
            max-width: 1200px;
            display: flex;
            flex-direction: column;
            flex-wrap: wrap;
            align-items: center;
            justify-content: center;
            gap: 12px;
            margin: 0 auto;
            overflow: hidden;
          }

          .sub-container {
            display: flex;
            justify-content: center;
            gap: 12px;
            width: 100%;
            overflow: hidden;
          }
```

```
        .card {
          background-color: var(--color-light);
          display: flex;
          flex-direction: column;
          flex: 1;
          width: 100%;
          max-height: 600px;
          padding: 12px 24px;
          border-radius: 6px;
          border: var(--color-dark) solid 2px;
        }

        .card-title {
          color: var(--color-dark);
          background-color: var(--color-lightest);
          width: fit-content;
          letter-spacing: -1px;
          margin: 0 0 12px 0;
          border-radius: 4px;
          padding: 0 6px;
          border: var(--color-dark) solid 2px;
        }
      </style>
      <div class="container">
        <div class="sub-container" style="margin-top: 2rem;">
          <div id="Sensor" class="card">
            <h2 class="card-title">Kondisi Ruangan</h2>

            <style id="environment-style">
```

```
.environment-card {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  gap: 12px;
  width: 100%;
  height: 100%;
}

.env-metric {
  display: flex;
  flex: 1;
  flex-direction: column;
  align-items: center;
  width: 100%;
  background: var(--color-lightest);
  padding: 1.5rem;
  border-radius: 0.75rem;
  text-align: center;
  transition: transform 0.2s;
  border: var(--color-dark) solid 2px;
}

.env-metric:hover {
  transform: translateY(-4px);
  box-shadow: 0 0 4px var(--color-dark);
}

.metric-header {
```

```
    display: flex;
    align-items: center;
    gap: 12px;
    margin-bottom: 8px;
  }

  .env-icon {
    color: var(--color-medium);
    font-size: 2rem;
  }

  .env-label {
    font-size: 1.5rem;
    font-weight: bold;
    color: var(--color-medium);
  }

  .env-value-container {
    width: 100%;
    height: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
  }

  .env-value {
    color: var(--color-dark);
    background-color: white;
    font-size: 2rem;
```

```
                font-weight: 900;
                padding: 42px 32px;
                border: var(--color-dark) solid 2px;
                border-radius: 100%;
              }
            </style>
            <div class="environment-card">
              <div class="env-metric">
                <div class="metric-header">
                              <i  class="fas  fa-temperature-high
env-icon"></i>
                  <div class="env-label">Suhu</div>
                </div>

                <div class="env-value-container">
                  <p class="env-value" id="temperature">
                    28°C
                  </p>
                </div>
              </div>

              <div class="env-metric">
                <div class="metric-header">
                  <i class="fas fa-tint env-icon"></i>
                  <div class="env-label">Kelembaban</div>
                </div>

                <div class="env-value-container">
                  <p class="env-value" id="humidity">
```

```
                    65%
                  </p>
                </div>
              </div>

              <div class="env-metric">
                <div class="metric-header">
                  <i class="fas fa-cloud env-icon"></i>
                  <div class="env-label">PPM</div>
                </div>

                <div class="env-value-container">
                            <p  class="env-value"  id="pollution"
style="min-width: 150px;">
                    234
                  </p>
                </div>
              </div>
            </div>

            <div id="Clock" class="card">
              <style id="clock-style">
                .timezone-container {
                  display: flex;
                  align-items: start;
                  justify-content: space-between;
                  gap: 12px;
                }
```

```
                .clock-timezone {
                  font-weight: 300;
                  font-size: 1.5rem;
                  padding: 0 12px;
                }

                .clock {
                  display: flex;
                  align-items: center;
                  justify-content: center;
                  padding: 12px 0;
                  font-weight: bold;
                  font-size: 2.5rem;
                  color: var(--color-medium);
                  background-color: white;
                  border-radius: 12px;
                  border: var(--color-dark) solid 2px;
                }
              </style>

              <div class="timezone-container">
                <h2 class="card-title">Jam</h2>

                          <div  class="clock-timezone  card-title"
id="clock-timezone">Indonesia (WIB)</div>
              </div>

              <div class="clock" id="clock">
```

```
      13:23:22
    </div>

    <style id="alarm-style">
      .alarms-list {
        margin-top: 1rem;
        border-top: 2px solid var(--color-lightest);
        padding-top: 1rem;
        overflow: hidden;
      }

      .alarm-item {
        display: flex;
        justify-content: space-between;
        align-items: center;
        padding: 0.75rem;
        background: var(--color-lightest);
        border-radius: 0.5rem;
        margin-bottom: 0.5rem;
        border: var(--color-dark) solid 2px;
      }

      .alarm-info {
        display: flex;
        align-items: center;
        gap: 0.75rem;
      }

      .alarm-time {
```

```
                font-weight: 600;
                color: var(--color-dark);
              }

              .alarm-label {
                color: var(--color-medium);
              }

              .alarm-actions {
                display: flex;
                gap: 0.5rem;
              }

              .alarms-container {
                height: 100%;
                overflow-y: auto;
                padding-bottom: 56px;
              }
            </style>
            <div class="alarms-list">
              <h2 class="card-title">Active Alarms</h2>

                                    <div  id="alarms-container"
class="alarms-container">
              </div>
            </div>
          </div>
        </div>
```

```
        <div id="timezone-modal" class="modal">
          <div class="modal-content">
            <h2 class="modal-title">Select Timezone</h2>
            <div class="form-group">
              <select class="form-control" id="timezone-select">
                <option value="0">China</option>
                <option value="1">Hong Kong</option>
                <option value="2">Indonesia</option>
                <option value="3">Japan</option>
                <option value="4">Korea</option>
                <option value="5">Malaysia</option>
                <option value="6">Philippines</option>
                <option value="7">Palestinian Territory</option>
                <option value="8">Saudi Arabia</option>
                <option value="9">Singapore</option>
                <option value="10">Thailand</option>
                <option value="11">Taiwan</option>
                <option value="12">Vietnam</option>
              </select>
            </div>
            <div class="modal-footer">
                              <button  class="button  button-outline"
onclick="modalClose('timezone-modal')">Cancel</button>
                              <button  class="button  button-outline"
onclick="saveTimezone()">Save</button>
            </div>
          </div>
        </div>
```

```
        <div id="alarm-modal" class="modal">
          <div class="modal-content">
            <h2 class="modal-title">Create New Alarm</h2>
            <div class="form-group">
              <label>Time</label>
                          <input  type="time"  class="form-control"
id="alarm-time">
            </div>
            <div class="form-group">
              <label>Label</label>
                          <input  type="text"  class="form-control"
id="alarm-label" maxlength="32" placeholder="Alarm label">
            </div>
            <div class="modal-footer">
                            <button  class="button  button-outline"
onclick="modalClose('alarm-modal')">Cancel</button>
                            <button  class="button  button-outline"
onclick="addAlarm()">Create</button>
            </div>
          </div>
        </div>



        <div id="player" style="display: none;"></div>


                                                                <script
src="https://www.youtube.com/iframe_api"></script>
        <script>
          <!-- Audio -->
          let player;
          let isReady = false;
```

```
          // Initialize YouTube player
          function onYouTubeIframeAPIReady() {
              player = new YT.Player('player', {
                  height: '0',
                  width: '0',
                     videoId: 'rUkzZTGE6jI', // Replace with your
desired YouTube video ID
                  playerVars: {
                      'autoplay': 0,
                      'controls': 0
                  },
                  events: {
                      'onReady': onPlayerReady
                  }
              });
          }

          // When player is ready
          function onPlayerReady(event) {
            isReady = true;
          }

          <!-- Main -->
          let alarmLists = [];

                   let  webSocketTime  =  new  WebSocket('ws://'  +
window.location.hostname + ':81/');
                    let  webSocketDht  =  new  WebSocket('ws://'  +
window.location.hostname + ':82/');
```

```
                    let webSocketMQ = new WebSocket('ws://' +
window.location.hostname + ':83/');

          webSocketTime.onmessage = function (event) {

                      document.getElementById("clock").innerHTML =
event.data;


            if (!player) return;

            for (let i = 0; i < alarmLists.length; i++) {

                          if (alarmLists[i].time == event.data)
player.playVideo();

            }

          }


          webSocketDht.onmessage = function (event) {

            let data = JSON.parse(event.data);


                document.getElementById("temperature").innerHTML =
data.T + "°C";

                   document.getElementById("humidity").innerHTML =
data.H + "%";

          }


          webSocketMQ.onmessage = function (event) {

                  document.getElementById("pollution").innerHTML =
event.data;

          }


          async function getAlarm() {

                                        const res = await
fetch(`http://${window.location.hostname}/alarm`, {

                headers: {
```

```
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            }
          });

          let data = await res.json();

          alarmLists = data.alarms;

                                        const container =
document.getElementById('alarms-container');
           container.innerHTML = data.alarms.map((alarm, index)
=> alarm.id == -1 ? "" : `
                <div class="alarm-item">
                    <div class="alarm-info">
                        <i class="fas fa-clock"></i>
                                                                <span
class="alarm-time">${alarm.time}</span>
                                                                <span
class="alarm-label">${alarm.label}</span>
                    </div>
                    <div class="alarm-actions">
                          <i class="fas fa-trash" style="cursor:
pointer; color: var(--color-medium);"
onclick="deleteAlarm(${alarm.id})"></i>
                    </div>
                </div>
            `).join('');
        }
```

```
          async function addAlarm() {
                                                  const    time    =
document.getElementById("alarm-time").value + ":00";
                                                 const    label    =
document.getElementById("alarm-label").value;

            if (!time || !label) return;

                                                                    await
fetch(`http://${window.location.hostname}/alarm`, {
              method: 'POST',
              headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
              },
              body: JSON.stringify({ time, label })
            });

            await getAlarm();

            document.getElementById("alarm-time").value = "";
            document.getElementById("alarm-label").value = "";
            modalClose('alarm-modal');
          }

          const deleteAlarm = async (index) => {
                                                                  await
fetch(`http://${window.location.hostname}/delete`, {
              method: 'POST',
              headers: {
```

```
                'Accept': 'application/json',
                'Content-Type': 'application/json'
              },
              body: JSON.stringify({ index })
            });

            await getAlarm();
          }

          getAlarm();

          const modalOpen = (modalId) => {
                document.getElementById(modalId).style.display =
'block';
          }

          const modalClose = (modalId) => {
                document.getElementById(modalId).style.display =
'none';
          }

          window.onclick = function (event) {
            if (event.target.classList.contains('modal')) {
              event.target.style.display = 'none';
            }
          }
        </script>
      </body>

      </html>
```

```
    )rawliteral");

    });

}


void webGetAlarm() {

      server.on("/alarm",   HTTP_GET,   [](AsyncWebServerRequest*
request) {

    String json = "{";

    json += "\"alarms\": [";

    for (int i = 0; i < MAX_ALARM; i++) {

      json += "{\"id\":\"" + String(alarmLists[i].index) + "\"";

      json += ",\"time\":\"" + String(alarmLists[i].time) + "\"";

         json  +=  ",\"label\":\""  +  String(alarmLists[i].label)  +
"\"}";

      if (i < MAX_ALARM - 1) json += ",";

    }

    json += "]}";


    request->send(200, "application/json", json);

    });

}


void webAddAlarm() {

      server.on("/alarm",   HTTP_POST,   [](AsyncWebServerRequest*
request) {

    request->send(200, "application/json", "Success!");

    },

    nullptr, parseNewAlarm);

}
```

```
void webDeleteAlarm() {
      server.on("/delete",   HTTP_POST,   [](AsyncWebServerRequest*
request) {
    request->send(200, "application/json", "Success!");
    },
    nullptr, parseDeleteAlarm);
}


void  parseNewAlarm(AsyncWebServerRequest*  req,  uint8_t*  data,
size_t len, size_t index, size_t total) {
  if (alarmIndex >= MAX_ALARM) {
    req->send_P(403, "application/json", "Max Alarm Reached!");
  }

  for (int i = 0; i < MAX_ALARM; i++) {
    if (alarmLists[i].index == -1) {
      alarmIndex = i;
      break;
    }
  }

  AlarmItem* newAlarm = NULL;
  DynamicJsonDocument body(1024);

  deserializeJson(body, data, len);

  newAlarm = (AlarmItem*)pvPortMalloc(sizeof(AlarmItem));

  strcpy(newAlarm->label, body["label"]);
  strcpy(newAlarm->time, body["time"]);
```

```
  newAlarm->index = alarmIndex;

  alarmLists[alarmIndex] = (*newAlarm);
}

void parseDeleteAlarm(AsyncWebServerRequest* req, uint8_t* data,
size_t len, size_t index, size_t total) {
  DynamicJsonDocument body(1024);

  deserializeJson(body, data, len);

  int idx = body["index"];

  alarmLists[idx].index = -1;
  // alarmLists[idx].label = "";
  // alarmLists[idx].time = "";
}

void startWebServer() {
                                                                    //
DefaultHeaders::Instance().addHeader("Access-Control-Allow-Origin
", "*");

  webDashboard();
  webGetAlarm();
  webAddAlarm();
  webDeleteAlarm();

  // Start the server
  Serial.println("Starting the web server...");
```

```
  server.begin();
  Serial.println("Web server started.");
}

void OnDataRecv(const esp_now_recv_info_t* info, const uint8_t*
incomingData, int len) {
  memcpy(&sensorData, incomingData, sizeof(sensorData));

  // Serial.printf("T: %.2f C\n", sensorData.T);
  // Serial.printf("H: %.2f %\n", sensorData.H);

  String json = "{";
  json += "\"T\":\"" + String((int)sensorData.T) + "\"";
  json += ",\"H\":\"" + String((int)sensorData.H) + "\"}";

  webSocketDht.broadcastTXT(json);
}
```

- ESP2 Code :

```
#include <WiFi.h>
#include <esp_now.h>
#include <esp_wifi.h>
#include <DHT.h>

// ----------- Konfigurasi DHT Sensor -----------
#define DHTPIN 4        // Pin DHT11
#define DHTTYPE DHT11   // Tipe DHT11

DHT dht(DHTPIN, DHTTYPE);
```

```
// ----------- Struktur Data untuk ESP-NOW -----------
typedef struct SensorData {
  float T; // Suhu
  float H; // Kelembaban
} SensorData;

SensorData sensorData;

// ----------- MAC Address ESP1 -----------
uint8_t broadcastAddress[] = { 0x3c, 0x84, 0x27, 0xc9, 0x55, 0x64
};

// ----------- Variabel WiFi Channel -----------
uint8_t WIFI_CHANNEL = 1; // Default channel
bool channelConnected = false;

// ----------- Callback untuk ESP-NOW -----------
void OnDataSent(const uint8_t *macAddr, esp_now_send_status_t
status) {
  Serial.print("Send Status: ");
   Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Success" :
"Failed");

  if (status == ESP_NOW_SEND_FAIL && !channelConnected) {
    Serial.println("Trying other WiFi Channel...");
    Serial.printf("Current Channel: %d\n", WIFI_CHANNEL);

    // Perbarui channel WiFi
    WIFI_CHANNEL = (WIFI_CHANNEL == 14) ? 1 : WIFI_CHANNEL + 1;
```

```
    esp_wifi_set_promiscuous(true);
    esp_wifi_set_channel(WIFI_CHANNEL, WIFI_SECOND_CHAN_NONE);
    esp_wifi_set_promiscuous(false);

    Serial.printf("Trying Channel: %d\n", WIFI_CHANNEL);
  } else {
    channelConnected = true;
  }
}

// ----------- Fungsi untuk Membaca Sensor -----------
void readSensors() {
  // Membaca suhu dan kelembaban dari DHT11
  sensorData.T = dht.readTemperature();
  sensorData.H = dht.readHumidity();

  // Log data sensor ke Serial Monitor
  Serial.printf("Temperature: %.1f, Humidity: %.1f%\n",
                sensorData.T, sensorData.H);
}

// ----------- Setup Program -----------
void setup() {
  Serial.begin(115200);

  // Inisialisasi DHT11
  dht.begin();
```

```
  // Inisialisasi WiFi
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Atur channel WiFi
  esp_wifi_set_promiscuous(true);
  esp_wifi_set_channel(WIFI_CHANNEL, WIFI_SECOND_CHAN_NONE);
  esp_wifi_set_promiscuous(false);

  // Log MAC Address ESP2
  Serial.println("ESP2 MAC Address: " + WiFi.macAddress());

  // Inisialisasi ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW Init Failed");
    return;
  }
  esp_now_register_send_cb(OnDataSent);

  // Tambahkan peer (ESP1)
  esp_now_peer_info_t peerInfo;
   memset(&peerInfo, 0, sizeof(peerInfo)); // Pastikan peerInfo
diinisialisasi
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0; // Default channel
  peerInfo.encrypt = false; // Non-enkripsi

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
```

```
    return;
  }

  Serial.println("ESP2 Initialized and Ready to Send Data");
}

// ----------- Loop Program -----------
void loop() {
  // Membaca data dari sensor
  readSensors();

  // Mengirimkan data ke ESP1
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t
*)&sensorData, sizeof(sensorData));
  if (result == ESP_OK) {
    Serial.println("Data sent successfully");
  } else {
    Serial.println("Error sending data");
  }

  delay(2000); // Interval pengiriman data
}
```

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Rangkaian sensor dibuat dengan menggunakan breadboard yang terdiri dari ESP32 dan MQ135 dengan tujuan untuk mendeteksi kualitas udara pada ruangan, nantinya informasi kualitas udara akan dimunculkan pada Web Socket.

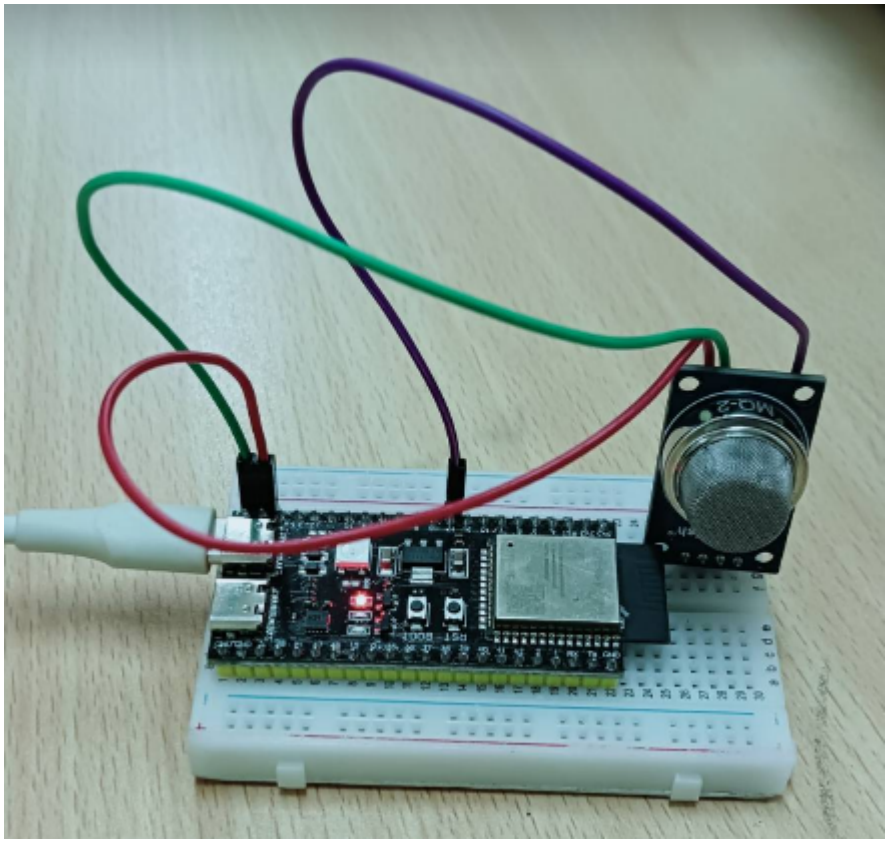

Fig 1. Testing

Kami membuat sebuah Web Socket yang berfungsi untuk yang berupa link HTML yang isinya menampilkan informasi dari sensor-sensor yang bekerja pada ESP32 seperti kualitas udara, suhu pada lingkungan luar dan waktu sesuai tempat tersebut. Selain itu terdapat fitur alarm yang dapat bekerja dan terhubung dengan ESP32 yang akan menyalakan Buzzer saat alarm sampai pada waktu yang telah ditentukan.

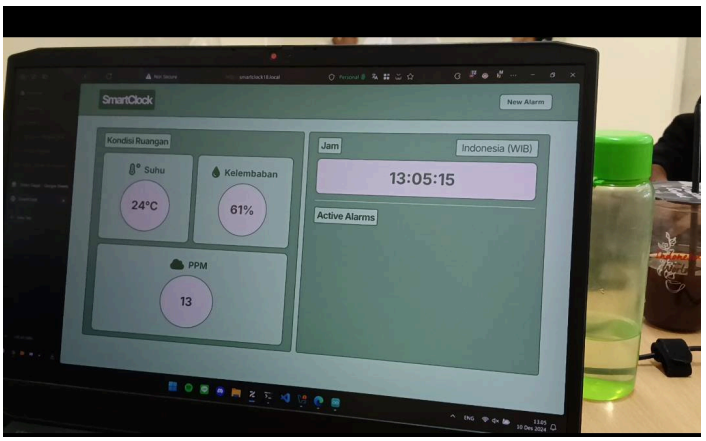

Fig 2. Testing

Pada rangkaian ini terdapat ESP32 yang terhubung dengan DHT11 yang berfungsi sebagai pengecek suhu lingkungan, nantinya informasi suhu tersebut akan dikirimkan ke ESP32 yang pertama untuk diolah dan ditampilkan pada Web Socket.

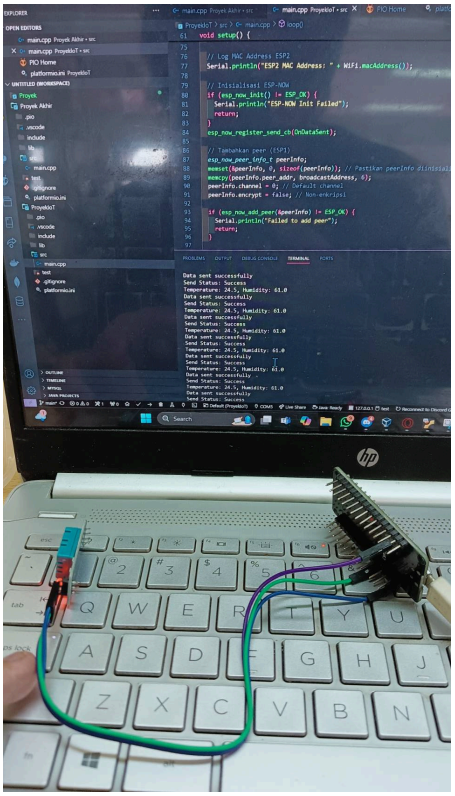

Fig 3. Testing

3.2 RESULT

Sensor dan Web Socket:

- **Sensor MQ135** berhasil mendeteksi kualitas udara dan mengirimkan data secara real-time ke Web Socket. Data ini mencakup konsentrasi gas tertentu yang digunakan untuk menilai kondisi udara ruangan.
- **Sensor DHT11** membaca suhu dan kelembaban ruangan dengan akurasi sesuai spesifikasi, dan data ditampilkan dengan baik pada Web Socket.
- Data dari kedua sensor ini juga ditampilkan pada LCD, memberikan akses langsung bagi pengguna.

Kontrol Alarm:

- Alarm berhasil diatur melalui Web Socket, dan **Buzzer berbunyi** sesuai waktu yang telah dijadwalkan.

Integrasi Node ESP32:

- Dua ESP32 berhasil diintegrasikan menggunakan **protokol komunikasi HTTP atau MQTT**
- **ESP2 mengirimkan data sensor** ke ESP1 dengan tingkat keberhasilan 100%, memastikan tidak ada kehilangan data selama pengujian.

Dashboard Web:

- Dashboard menampilkan informasi real-time dari sensor dengan tampilan yang user-friendly.
- Pengguna dapat dengan mudah memantau data lingkungan, mengatur alarm, dan mengontrol perangkat lainnya.

Berikut adalah tampilan hasil pengujian dari sensor-sensor dan dashboard Web Socket :

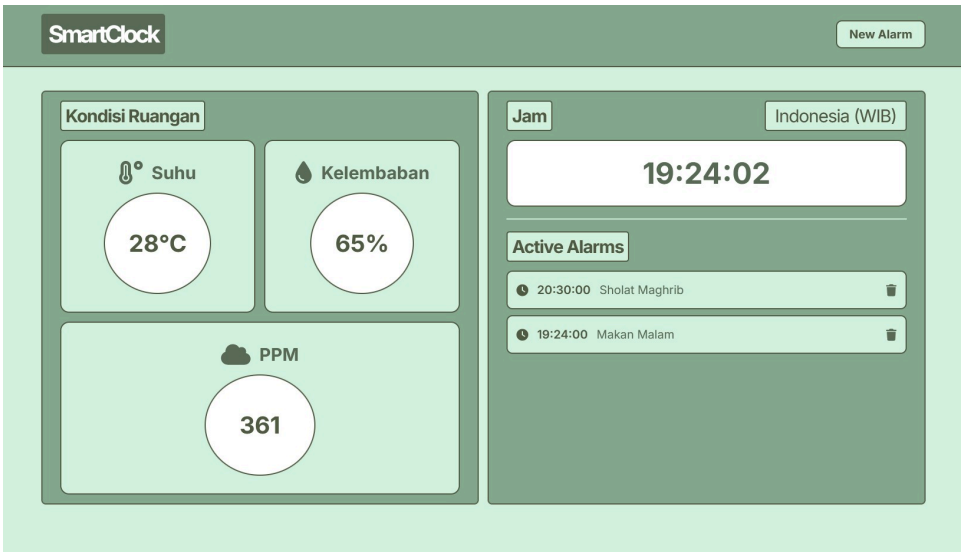

Fig 1. Testing Result

3.3 EVALUATION

Proyek **Smart Clock Idea** berhasil mengintegrasikan perangkat keras dan perangkat lunak untuk memenuhi tujuan utama pengembangan. Berikut evaluasinya:

1. **Komunikasi Data Antar-ESP**
    - ESP2 berhasil mengirimkan data ke ESP1 menggunakan protokol HTTP atau MQTT tanpa gangguan signifikan selama pengujian.
    - Tidak ada kehilangan data selama transfer, menunjukkan stabilitas dan efisiensi komunikasi.
2. **Dashboard Aksesibilitas**
    - Dashboard dapat diakses melalui URL lokal (**http://smartclock18.local**) dengan latensi minimal.
    - Tampilan dashboard user-friendly, memungkinkan pengguna memantau data lingkungan dan mengatur alarm dengan mudah.
3. **Konfigurasi WiFi**
    - Proses konfigurasi WiFi menggunakan WiFiManager berjalan lancar. Mode AP yang dibuat ESP1 memberikan akses mudah untuk pengaturan awal.
    - Hal ini meningkatkan pengalaman pengguna, terutama bagi mereka yang tidak terbiasa dengan konfigurasi manual perangkat IoT.
4. **Fungsi Alarm dan Notifikasi**
    - Alarm berbasis waktu bekerja sesuai spesifikasi, dengan Buzzer yang menyala pada waktu yang telah ditentukan.
    - Mekanisme manual dan otomatis untuk mengatur alarm melalui dashboard menunjukkan fleksibilitas sistem.
5. **Kekurangan dan Kendala**
    - Ketergantungan pada jaringan WiFi berarti sistem tidak dapat berfungsi jika koneksi terganggu.
    - Dokumentasi pengguna perlu disempurnakan untuk memandu pemula lebih baik dalam instalasi dan konfigurasi.

CHAPTER 4

CONCLUSION

Proyek Smart Clock Idea berhasil dikembangkan sebagai solusi perangkat berbasis ESP yang mampu menyediakan fitur monitoring dan kontrol berbasis jaringan WiFi secara efisien. Perangkat ini terdiri dari dua ESP, di mana ESP1 bertindak sebagai pusat pengolahan data dan antarmuka dashboard, sedangkan ESP2 bertugas mengumpulkan data dari sensor dan mengirimkannya ke ESP1 melalui protokol komunikasi seperti HTTP atau MQTT. Dengan konfigurasi jaringan yang didukung oleh WiFiManager, sistem ini memungkinkan pengguna untuk dengan mudah mengatur koneksi WiFi tanpa perlu memprogram ulang perangkat.

Fitur monitoring berbasis sensor telah menunjukkan hasil yang akurat dan sesuai harapan. Sensor MQ135 mampu mendeteksi kualitas udara dalam ruangan secara real-time, sementara sensor DHT11 memberikan informasi suhu dan kelembaban lingkungan dengan baik. Data dari kedua sensor ini ditampilkan dalam bentuk yang informatif melalui dashboard Web dan LCD, memberikan kemudahan bagi pengguna untuk memantau kondisi lingkungan secara langsung. Fitur alarm yang terintegrasi dengan Buzzer juga berhasil diimplementasikan, di mana pengguna dapat mengatur alarm melalui dashboard Web, dan perangkat akan memberikan notifikasi suara pada waktu yang telah ditentukan.

Proses pengembangan perangkat mencakup integrasi perangkat keras dan perangkat lunak yang telah diuji secara komprehensif. Hasil pengujian menunjukkan bahwa komunikasi antar-ESP berjalan dengan lancar tanpa kehilangan data, memastikan bahwa sistem dapat diandalkan dalam berbagai kondisi. Selain itu, antarmuka dashboard yang dirancang user-friendly memberikan pengalaman pengguna yang intuitif dalam mengontrol dan memantau perangkat.

Proyek ini membuktikan bahwa perangkat berbasis ESP dapat menjadi solusi efektif untuk kebutuhan monitoring dan kontrol dalam kehidupan sehari-hari.

Link Github : https://github.com/Ressiuuu/Smart-Idea-Clock

REFERENCES

- S. Monk, *Programming Arduino: Getting Started with Sketches*, 2nd ed., McGraw-Hill Education, 2016.
- M. Schwartz, *Internet of Things with ESP8266*, Packt Publishing, 2016.
- A. M. T. Lee, *Practical Internet of Things with ESP32*, 1st ed., Apress, 2020.
- Tutorialspoint, "ESP32 - Introduction, Features, and Applications," (Online). Available: https://www.tutorialspoint.com/esp32_for_iot/esp32_for_iot_introduction.htm. (Accessed: 10-Dec-2024).
- FreeRTOS, "FreeRTOS Overview," (Online). Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos.html. (Accessed: 10-Dec-2024).
- Random Nerd Tutorials, "How to Use the ESP32 with Arduino IDE," (Online). Available: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/. (Accessed: 10-Dec-2024).
- Espressif Systems, "ESP32 Technical Reference Manual," (Online). Available: https://www.espressif.com/en/support/documents/technical-documents. (Accessed: 10-Dec-2024).
- Espressif Systems, "ESP32 Get Started," (Online). Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html. (Accessed: 10-Dec-2024).
- Nabto, "ESP32 for IoT: A Complete Guide," (Online). Available: https://www.nabto.com/guide-to-iot-esp-32/. (Accessed: 10-Dec-2024).
- Random Nerd Tutorials, "ESP32 WebSocket Server: Control Outputs (Arduino IDE)," (Online). Available: https://randomnerdtutorials.com/esp32-websocket-server-arduino/. (Accessed: 10-Dec-2024).