

EL4236 PERANCANGAN PERANGKAT LUNAK JARINGAN

LAPORAN TUGAS BESAR

“Aplikasi Monitor Temperatur serta Kontrol Pendingin Ruangan”

Kelompok E

Samuel Benyamin Niman (13220050)

Muhammad Hanif Hibatullah (13220051)

Bayu Aji Nugroho (13221601)

A. Latar Belakang

Suhu dalam suatu ruangan memiliki dampak yang sangat signifikan dalam berbagai konteks kehidupan sehari-hari. Dari ruang server hingga rumah tangga, bahkan fasilitas industri, menjaga suhu dalam kisaran yang tepat adalah kunci untuk menjaga kinerja optimal peralatan, kenyamanan manusia, dan keselamatan operasional.

Pemantauan suhu adalah langkah kritis dalam upaya menjaga kestabilan dan keandalan sistem. Misalnya, dalam lingkungan ruang server, perubahan suhu yang tiba-tiba dan tidak terduga dapat menjadi sinyal awal akan terjadinya masalah serius. Sebuah kenaikan tiba-tiba dalam suhu server dapat menandakan risiko overheating yang bisa menyebabkan kerusakan perangkat keras, kegagalan sistem, bahkan kehilangan data yang mahal.

Namun, dampak dari suhu yang tidak terkendali bukanlah terbatas hanya pada ruang server. Dalam konteks rumah tangga, suhu yang ekstrem dapat mengganggu kenyamanan penghuni dan bahkan berpotensi menyebabkan risiko kesehatan, terutama pada anak-anak, lansia, dan individu dengan kondisi kesehatan yang rentan. Sementara dalam lingkungan industri, fluktuasi suhu yang tidak terkontrol dapat merusak peralatan, memperlambat proses produksi, dan bahkan mengancam keselamatan pekerja.

Oleh karena itu, penting untuk memantau suhu secara berkala dan mengambil tindakan proaktif ketika perubahan signifikan terjadi. Dengan menggunakan sistem pemantauan suhu, kita dapat memperoleh data secara real-time tentang perubahan suhu dalam lingkungan tertentu. Salah satu manfaat utama dari pemantauan suhu yang terus-menerus adalah kemampuan untuk mengambil tindakan preventif sebelum masalah berkembang menjadi lebih serius. Misalnya, jika pemantauan suhu menunjukkan peningkatan bertahap dalam suhu ruangan di sekitar peralatan kritis, langkah-langkah dapat diambil untuk mendinginkan sistem sebelum mencapai titik yang mengkhawatirkan. Ini dapat meliputi pengaturan suhu pendingin, pengaturan ulang peralatan, atau bahkan penggantian komponen yang rentan terhadap panas.

Dengan demikian, pemantauan suhu yang efektif tidak hanya membantu mencegah kerusakan peralatan dan kegagalan sistem, tetapi juga meningkatkan efisiensi operasional, meningkatkan kenyamanan, dan menjaga keselamatan dalam berbagai konteks kehidupan sehari-hari. Dalam dunia yang semakin terhubung dan terotomatisasi, investasi dalam sistem pemantauan suhu yang canggih tidak hanya menjadi kebutuhan, tetapi juga merupakan langkah yang bijaksana untuk memastikan kelangsungan dan keberhasilan operasional.

B. Deskripsi Umum Sistem

Secara umum, deskripsi sistem adalah “Sistem yang dapat melakukan monitoring suhu ruangan secara berkala dimana terdapat dua klien dengan klien pertama sebagai pihak yang mengakses data suhu dan klien kedua sebagai pihak yang mengirimkan data suhu”. Sistem memiliki *typical message* antar server dan client yang ditunjukkan pada Tabel 1 dan 2.

Tabel 1 Typical Message antara Client 1 (Pengguna yang Akses) dengan Server

Client 1 (Request)	Server (Response)
Login (username, password)	Berhasil/Gagal
dataTemperature	dataTemperature
controlUP	currentACTemperature
controlDOWN	currentACTemperature
getTemperature	lastTemperature
requestLog	logFile

Tabel 2 Typical Message antara Client 2 (Sistem Pembaca Suhu) dengan Server

Client 2 (POST/GET)	Server (POST/GET)
POST: temperature	GET: lastTemperature
GET: “generate temperature”	POST: “generate temperature”

C. Spesifikasi Teknis

Berdasarkan deskripsi umum sistem tersebut, dibuat spesifikasi teknis dari sistem dimana untuk server menggunakan *hardware* Komputer dengan basis bahasa pemrograman Python, untuk client menggunakan Raspberry Pi (untuk Client 1) berbasis bahasa Python dan NodeMCU (untuk Client 2) berbasis bahasa C++ (Arduino). Dikarenakan pada tugas ini kedua client perlu memiliki protokol yang berbeda, maka digunakan protokol HTTP (untuk Client 1) dan Socket (untuk Client 2). Secara garis besar spesifikasi teknis dari sistem adalah sebagai berikut.

- Server (Komputer) – Muhammad Hanif Hibatullah
 - Backend : Python dan Flask (untuk HTTP)
 - Penyimpanan Data : Logfile (TXT dan CSV)
 - Mekanisme timestamp secara otomatis menyesuaikan waktu server/komputer
- Client 1 (Raspberry Pi) – Samuel Benyamin Niman
 - Programming Language : Python dengan modul tkinter untuk GUI

- Protocol : HTTP
- Penyimpanan Data Raspi : Logfile (TXT dan CSV)
- Client 2 (NodeMCU) – Bayu Aji Nugroho
 - Programming Language : C++ (Arduino)
 - Protocol : Socket
 - Sensor temperatur : DHT22

Penyimpanan data yang dimaksudkan disini didefinisikan pada Tabel 3.

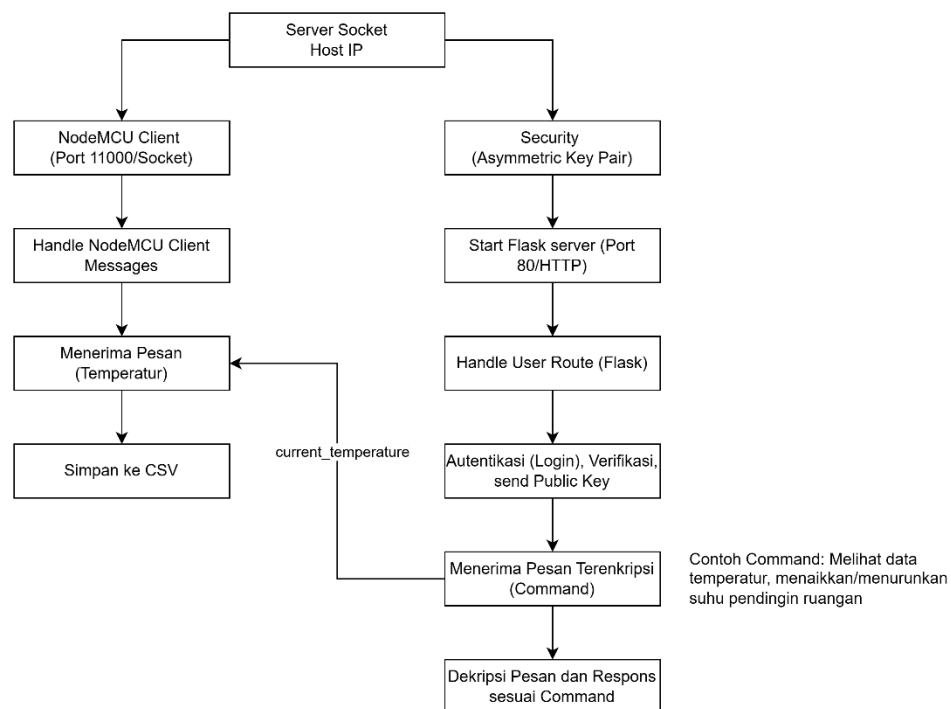
Tabel 3 Data yang disimpan pada Server

Jenis File	Nama Data	Keterangan
csv	user.csv	Data yang berisi username, password untuk autentikasi login pengguna. Berupa dua kolom ['username'] dan ['password'] dimana keduanya string
csv	data_{tanggal}.csv	Data yang berisi temperatur pada tanggal tersebut. Berupa dua kolom ['timestamp] dan ['temperatur'] dimana keduanya string
txt	ac_temperature.txt	Data temperatur AC
txt	log_{user}.txt	Data yang berisi log file user tersebut

D. Perancangan Sistem

• Rancangan Sistem di Server

Sistem di Server terdiri dari beberapa Block Diagram fungsi-fungsi di antaranya



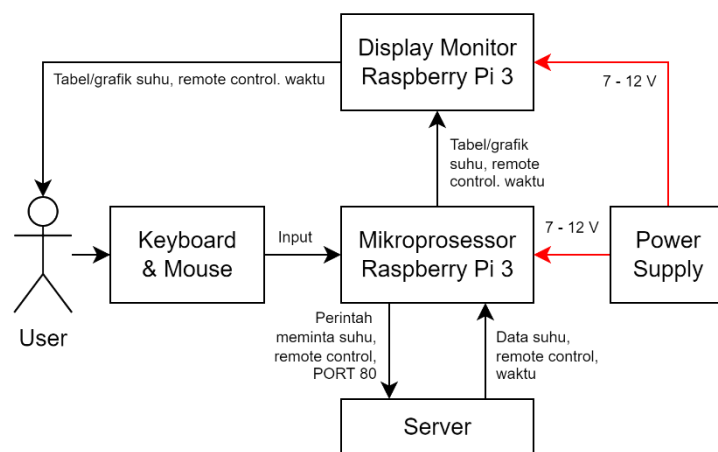
Gambar 1 Blok Diagram Rancangan sistem di Server

Seperti yang dijelaskan pada spesifikasi teknis, untuk Client 1 menggunakan protokol HTTP dengan penambahan sistem *security* berupa *asymmetric key*. Penambahan *security* ini dilakukan dengan *library* ‘*cryptography*’ dengan terlebih dahulu *generate private key* kemudian menghasilkan *pair* untuk *public key*nya (yang nanti dikirimkan ke Client 1). Selain itu, seperti yang diketahui digunakan Flask untuk HTTP dimana Flask menggunakan pendekatan Route untuk setiap kali server dan client berkomunikasi (Request dan Response) dimana setiap Route memiliki fungsinya masing-masing. Sistem yang dibuat juga memiliki webserver yang dapat menampilkan HTML masing-masing fungsi tetapi untuk *security* belum diterapkan (hanya menerapkan sistem session).

Sedangkan untuk Client 2, menggunakan protokol socket dimana diterapkan pada Port 11000 (dapat diatur) dimana metodenya adalah server akan selalu menerima data dari Client 2 tetapi jika Client 1 meminta data (*on demand*) maka server akan mengirimkan data berupa “*generate_temperature*” ke Client 2 sehingga Client 2 dapat merespons langsung dengan temperatur saat ini dan datanya diterima oleh server. Perlu diketahui bahwa data yang diterima server berupa string sehingga file data temperatur (csv) keduanya string (timestamp dan data temperatur).

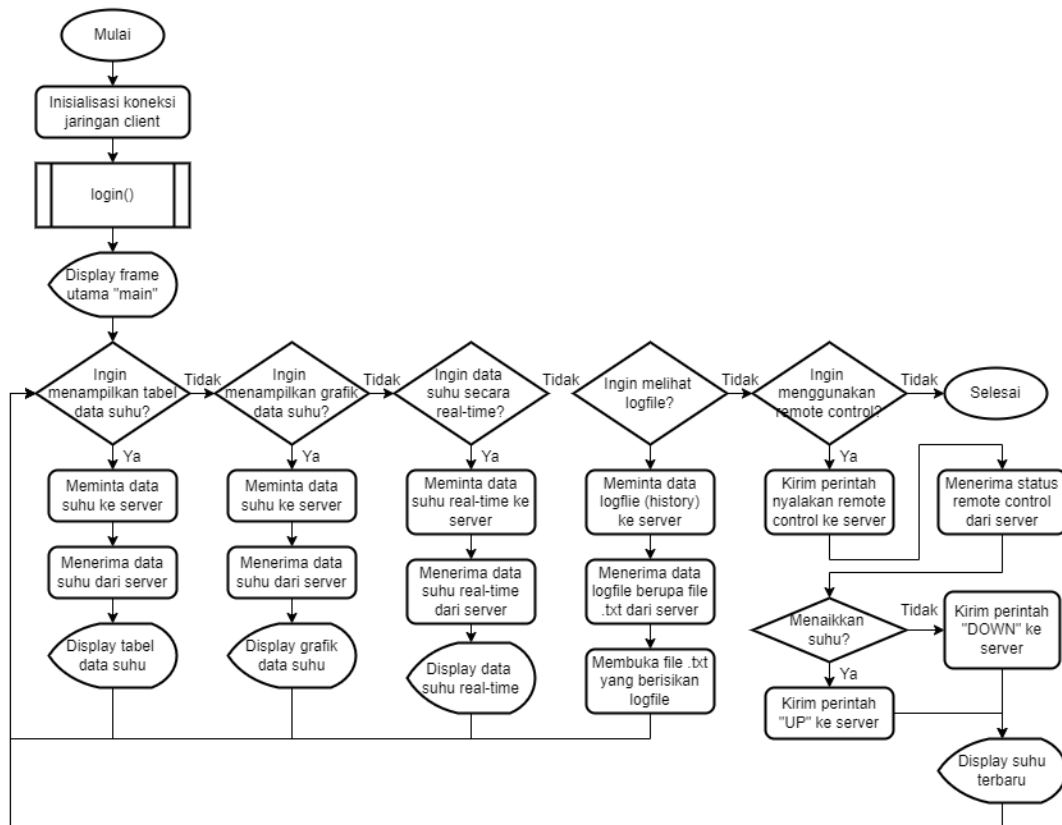
- **Rancangan Sistem di Client 1 – Raspberry Pi**

Client 1 yang berupa Raspberry Pi karena bertujuan memberikan perintah ke server serta menerima data hasil dari server, maka block diagram program pada client 1 adalah sebagai berikut:



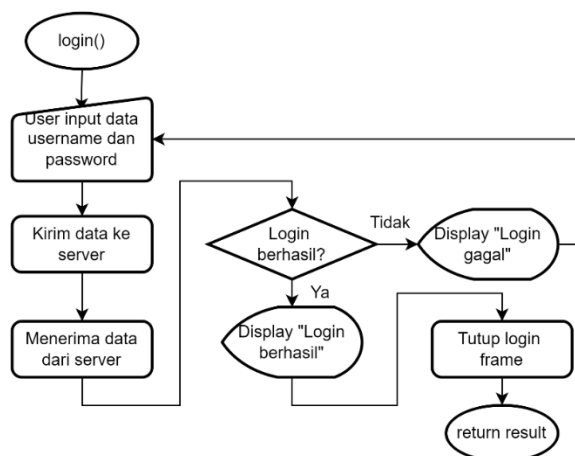
Gambar 2 Diagram Blok Sistem Client 1

Pada diagram blok client 1 (gambar 2), Raspberry Pi 3 akan terkoneksi dengan laptop menggunakan remote desktop connection, dengan kondisi bahwa koneksi antara Raspberry Pi 3, laptop, dan server terhubung dalam satu jaringan yang sama. Flowchart penggunaan aplikasi pada client 1 adalah sebagai berikut:



Gambar 3 User Flowchart Proses penggunaan Client 1

Setelah melakukan koneksi jaringan client, pengguna harus melakukan proses login (Gambar 4) yang meminta pengguna untuk memasukkan username serta password. Setelah client menerima data dari pengguna dan melakukan verifikasi, maka client 1 akan menerima hasil apakah login berhasil atau gagal. Jika login berhasil maka pengguna akan ke menu utama (Gambar 3) dimana terdapat fungsi mulai dari menampilkan tabel data suhu, menampilkan grafik data suhu, menampilkan data suhu secara real-time, meminta logfile (history) dari server, sehingga mengontrol suhu (remote control).

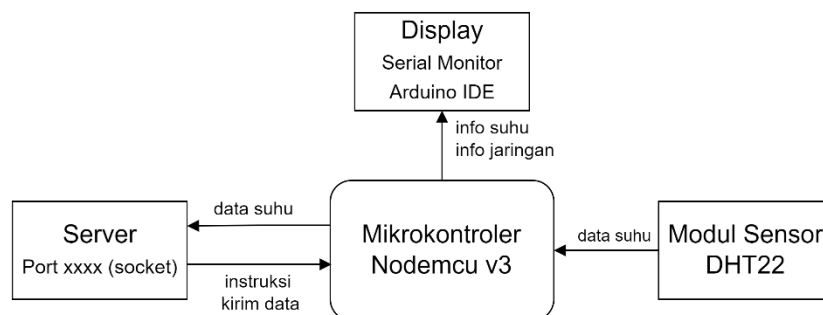


Gambar 4 Fungsi login pada Client 1

- **Rancangan Sistem di Client 2 – NodeMCU**

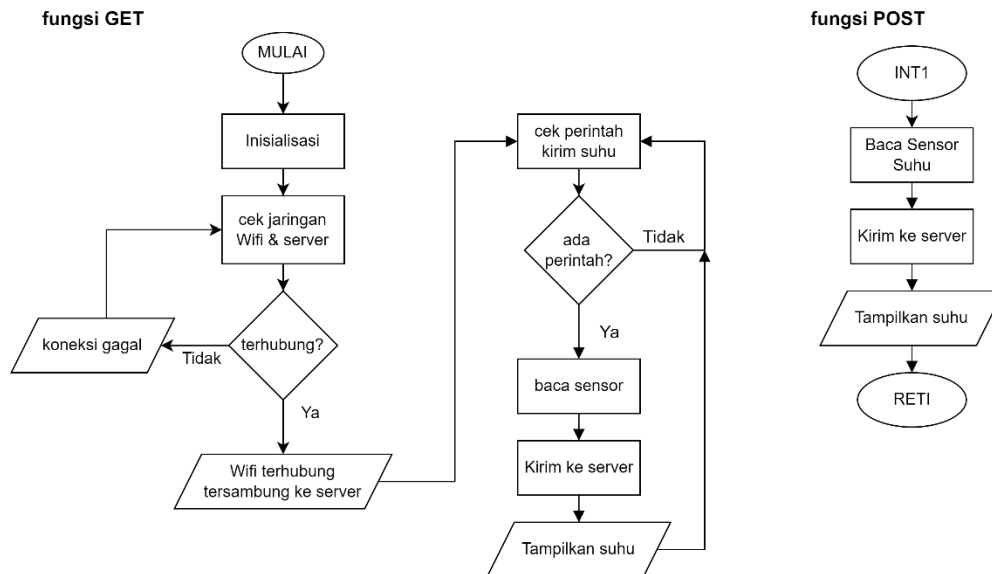
Dalam perancangannya, Client 2 menggunakan NodeMCU V3 Lolin sebagai mikrokontroler, Modul DHT22 sebagai modul sensor suhu, dan serial monitor Arduino IDE sebagai penampil data suhu dan informasi jaringan. NodeMCU V3 dipilih sebagai mikrokontroler pemroses data karena pada board NodeMCU V3 telah terintegrasi dengan modul wifi ESP8266 sehingga perangkat dapat berkomunikasi langsung dengan server menggunakan jaringan nirkabel. Pemilihan DHT22 sebagai modul sensor suhu didasarkan pada kemudahan pembacaan nilai suhu pada saat pembuatan kode program dan pada pengembangannya DHT22 juga dapat digunakan sebagai sensor kelembaban udara. Kemudian pemilihan serial monitor Arduino IDE karena penampil dalam sistem ini tidak digunakan secara terus menerus melainkan hanya pada saat tertentu ketika dibutuhkan pengecekan dan pemantauan data suhu dan informasi jaringan.

Mikrokontroler Nodemcu V3 pada client 2 berfungsi sebagai pemroses data suhu yang diambil dari modul sensor suhu DHT22. Data suhu selanjutnya dikirimkan ke server melalui komunikasi socket dan hasil pengiriman data suhu tersebut ditampilkan pada serial monitor Arduino IDE di komputer. Berikut ini merupakan diagram aliran data yang terjadi pada sistem Client 2.



Gambar 5 Diagram Aliran data Client 2

Dalam berkomunikasi dengan server, Client 2 dapat melakukan dua fungsi yaitu fungsi GET dan POST. Fungsi GET dilakukan ketika Client 2 menerima pesan perintah dari server untuk mengirimkan data suhu secara realtime (On Demand). Sedangkan fungsi POST dilakukan ketika Client 2 mengirimkan data suhu secara berkala dalam waktu 30 menit (Reguler). Untuk memenuhi dua fungsi sistem tersebut, program Client 2 dirancang dengan alur program seperti gambar berikut.



Gambar 6 Alur Program Client 2

Sesuai dengan gambar rancangan alur program di atas, sistem dapat melakukan dua fungsi GET dan POST ketika sistem sudah dapat berkomunikasi dengan server. Oleh karena itu sebelum melakukan 2 fungsi tersebut, sistem Client 2 harus tersambung terlebih dahulu pada jaringan wifi dan jaringan server.

E. Implementasi

• Implementasi Server

Server diimplementasikan dengan bahasa Python dan module Flask yang secara garis besar dibagi menjadi dua bagian (Handle NodeMCU) dan (Handle Route untuk HTTP/Client).

Pada implementasi server, terlebih dahulu diimport seluruh *module* yang digunakan di antaranya Flask, functools, csv, datetime, threading, socket, os, time, json, cryptography dan base64. Untuk inisialisasi awal berikut kodenya:

```

from flask import Flask, request, jsonify, render_template, session, redirect, url_for
from functools import wraps
import csv
from datetime import datetime
import threading
import socket
import os
import time
import json
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
import base64

```

```

## Inisialisasi app Flask dengan
app = Flask(__name__)
...
# Menjalankan Server (NodeMCU threading dan Flask di main)
if __name__ == '__main__':
    try:
        arduino_server_thread = threading.Thread(target=start_arduino_server)
        arduino_server_thread.daemon = True
        arduino_server_thread.start()
        app.run(host="0.0.0.0", port=80, debug=False)
    except KeyboardInterrupt:
        print("Shutting Down Server...")

```

Kemudian, dilakukan inisialisasi *private* dan *public key* untuk *security* berikut kodenya:

```

# RSA key untuk asymmetric key (bagian private key)
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
)
# Public key pair
public_key = private_key.public_key()

# Serialize the public key for sending to the client
public_key_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)

```

Dengan fungsi dekripsi pada server sebagai berikut (menggunakan algoritma *hashing* SHA256).

```

# Decrypt message
def decrypt_message(encrypted_message):
    decrypted_message = private_key.decrypt(
        base64.b64decode(encrypted_message)
        ,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return decrypted_message.decode('utf-8')

```

Untuk handle NodeMCU (namanya masih Arduino karena sebelumnya diasumsikan menggunakan Arduino), dilakukan dengan protokol *socket* dan fungsi yang membaca data yang dikirimkan oleh Arduino setiap saat.

```

# Fungsi untuk menangani permintaan klien Arduino (NodeMCU)
def handle_arduino_client(c, addr):

```



```

global arduino_c
arduino_c = c
try:
    while True:
        data = c.recv(1024).decode()
        if not data:
            print(f"Klien NodeMCU {addr} terputus")
            break
        else:
            print(f"Data diterima dari klien NodeMCU {addr}: {data}")
            timestamp = datetime.now().strftime('%d/%m/%Y %H:%M:%S')
            save_temperature_data(timestamp, data)
            temperature_data.append([timestamp, data]) # Tambahkan data suhu baru
except ConnectionResetError:
    print(f"Klien NodeMCU {addr} terputus")
finally:
    c.close()

# Mulai server untuk klien Arduino
def start_arduino_server():
    HOST = "0.0.0.0"
    PORT_ARDUINO = 11000
    s_arduino = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s_arduino.bind((HOST, PORT_ARDUINO))
    s_arduino.listen()
    print("Server NodeMCU dimulai di port", PORT_ARDUINO)

    while True:
        c_arduino, addr_arduino = s_arduino.accept()
        print(f"Klien NodeMCU {addr_arduino} terhubung")
        threading.Thread(target=handle_arduino_client, args=(c_arduino,
addr_arduino)).start()

```

Sedangkan untuk Client 1 (HTTP) berupa Raspberry Pi dilakukan dengan beberapa routing Flask seperti contoh untuk routing login berikut.

```

# Routing dari App
@app.route('/login', methods=['POST'])
def login():
    data = request.json
    username = data.get("username")
    password = data.get("password")
    ip = request.remote_addr

    if authenticate(username, password):
        session['username'] = username # Store username in session
        session['logged_in'] = True # Mark the user as logged in
        log_filename = f"log_{username}.txt"
        if not os.path.exists(log_filename):
            with open(log_filename, 'w') as log_file:

```

```

        log_file.write(f"Log File Client {username}:\n")
        client_log_files[username] = log_filename
        log_client_activity(username, f"{ip} - - [{datetime.now().strftime('%d/%b/%Y
%H:%M:%S')}]" \ "POST /login HTTP/1.1\" 200 -")
        return jsonify({"message": "Login berhasil", "public_key":
public_key_pem.decode('utf-8')})
    else:
        return jsonify({"error": "Username atau password salah"}), 401

```

Dapat dilihat bahwa terdapat fungsi yang terlebih dahulu mengambil data dari Client 1 kemudian mengirimkan json yang berisi message “Login Berhasil” dan public key yang sudah diserialisasi ke Client 1. Contoh lain adalah fungsi untuk menampilkan temperatur saat ini (*on demand*) dimana kodenya adalah sebagai berikut.

```

@app.route('/current_temperature', methods=['POST'])
def get_current_temperature():
    encrypted_data = request.data
    data = json.loads(decrypt_message(encrypted_data))
    username = data.get("username")
    password = data.get("password")
    ip = request.remote_addr

    if not authenticate(username, password):
        return jsonify({"error": "Kredensial tidak valid"}), 401

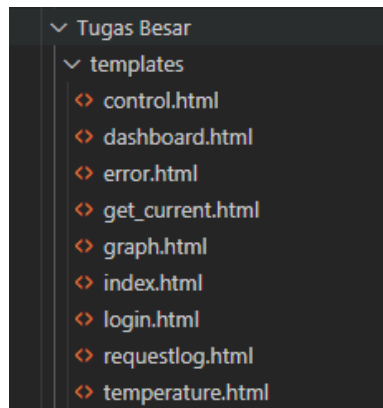
    global arduino_c

    try:
        arduino_c.send("generate_temperature.".encode())
        time.sleep(1)
        temperature = temperature_data[-1][1]
        log_client_activity(username, f"{ip} - - [{datetime.now().strftime('%d/%b/%Y
%H:%M:%S')}]" \ "POST /current_temperature HTTP/1.1\" 200 -")
        return jsonify({"current_temperature": temperature}), 200

    except Exception as e:
        return jsonify({"error": f"Gagal berkomunikasi dengan klien NodeMCU: {str(e)}"}),
500

```

Dapat dilihat bahwa fungsi ini terlebih dahulu melakukan verifikasi data yang sudah didekripsi kemudian server akan mengirimkan data (pesan) ke client 2 NodeMCU berupa “generate_temperature” (ada tambahan 1 karakter ‘.’ untuk penyesuaian dengan kode NodeMCU). Kemudian selanjutnya menunggu 1 detik untuk mengantisipasi pembacaan data yang salah dan lama waktu proses pertukaran data dari server ke client 2 kembali ke server lagi. Seluruh fungsi dilakukan routing yang sesuai seperti contoh login dan get_current_temperature di atas yang kode lengkapnya dapat dilihat di file “Server.py”. Selain itu, dilakukan juga implementasi untuk pengembangan webserver dengan menggunakan *template* html yang berada di folder yang sama dengan server.



Gambar 7 Template html untuk webserver

- **Implementasi Client 1 (Raspberry Pi)**

Client 1 diimplementasikan dengan bahasa pemrograman Python serta GUI menggunakan framework Tkinter. Protocol antara client dengan server menggunakan HTTP. Berikut penjelasan potongan kode program:

Program dimulai dengan melakukan import library yang dibutuhkan.

- requests: Digunakan untuk melakukan permintaan HTTP ke server.
- tkinter: Digunakan untuk membuat antarmuka grafis (GUI).
- messagebox dan ttk dari tkinter: Untuk kotak pesan dan widget khusus.
- matplotlib.pyplot: Digunakan untuk membuat grafik suhu.
- json: Untuk mengkonversi data ke dalam format JSON.
- cryptography: Untuk enkripsi dan dekripsi data menggunakan kunci publik/privat.
- base64: Untuk encoding data biner ke string ASCII.

```
import requests
import tkinter as tk
from tkinter import messagebox, ttk
import matplotlib.pyplot as plt
import json
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding
import base64
```

Kemudian melakukan koneksi dengan protokol HTTP.

```
SERVER_URL = "http://192.168.226.169:80"

username = None
password = None
public_key = None
```

Selanjutnya terdapat fungsi untuk enkripsi pesan. Fungsi `encrypt_message` akan mengenkripsi pesan menggunakan kunci publik server dengan padding OAEP dan algoritma SHA-256.

Selanjutnya terdapat berbagai fungsi yaitu

- `show_temperature_data`: Mengirim permintaan untuk data suhu dan menampilkannya menggunakan `show_data_window`.
- `show_data_window`: Menampilkan data suhu dalam tabel menggunakan Treeview.
- `show_temperature_graph`: Menampilkan grafik suhu menggunakan matplotlib.
- `open_remote_control`: Menampilkan kontrol jarak jauh untuk mengubah suhu AC dan memperbarui label suhu saat ini.
- `get_current_temperature`: Mengambil dan menampilkan suhu saat ini dari server.
- `request_log_file`: Meminta file log dari server dan menyimpannya secara lokal.

```
def show_temperature_data():
    message = encrypt_message(json.dumps({"username": username,
    "password": password}))
    response = requests.post(f"{SERVER_URL}/temperature",
    data=message)
    data = response.json()
    show_data_window(data)

def show_data_window(data):
    data_window = tk.Toplevel(window, bg='#add8e6')
    data_window.title("Data Suhu")

    table = ttk.Treeview(data_window)
    table["columns"] = ("timestamp", "temperature")
    table.column("#0", width=0, stretch=tk.NO)
    table.column("timestamp", anchor=tk.CENTER, width=150)
    table.column("temperature", anchor=tk.CENTER, width=100)

    table.heading("#0", text="", anchor=tk.CENTER)
    table.heading("timestamp", text="Waktu", anchor=tk.CENTER)
    table.heading("temperature", text="Suhu", anchor=tk.CENTER)

    for timestamp, temperature in data:
        table.insert("", tk.END, values=(timestamp, temperature))

    table.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

def show_temperature_graph():
    message = encrypt_message(json.dumps({"username": username,
    "password": password}))
    response = requests.post(f"{SERVER_URL}/temperature",
    data=message)
    data = response.json()
    timestamps = [d[0][10:] for d in data[-20:]]
    temperatures = [float(re.match(r'^\d+(\.\d+)?',
    d[1]).group()) for d in data[-20:]]
    plt.plot(timestamps, temperatures, marker='o')
    plt.xlabel('Waktu')
    plt.ylabel('Suhu')
    plt.title('Data Suhu Terbaru')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

def open_remote_control():
```

```

def send_command(command):
    message = encrypt_message(json.dumps({"username": username, "password": password}))
    response = requests.post(f"{SERVER_URL}/control/{command}", data=message)
    current_temp = response.json()["current_temperature"]
    label_current_temp.config(text=f"Suhu AC Saat Ini: {current_temp} °C")

    remote_control_window = tk.Toplevel(window, bg='#add8e6')
    remote_control_window.title("Kontrol Jarak Jauh")

    message = encrypt_message(json.dumps({"username": username, "password": password}))
    response = requests.post(f"{SERVER_URL}/control/current", data=message)
    current_temp = response.json()["current_temperature"]

    label_current_temp = tk.Label(remote_control_window, text=f"Suhu AC Saat Ini: {current_temp} °C", bg='#add8e6')
    button_up = tk.Button(remote_control_window, text="NAIK", command=lambda: send_command("UP"))
    button_down = tk.Button(remote_control_window, text="TURUN", command=lambda: send_command("DOWN"))

    label_current_temp.pack(pady=10)
    button_up.pack(pady=10)
    button_down.pack(pady=10)

def get_current_temperature():
    message = encrypt_message(json.dumps({"username": username, "password": password}))
    response = requests.post(f"{SERVER_URL}/current_temperature", data=message)
    if response.status_code == 200:
        current_temp = response.json()["current_temperature"]
        messagebox.showinfo("Suhu Saat Ini", f"Suhu saat ini adalah {current_temp} °C")
    else:
        messagebox.showerror("Error", "Gagal mengambil suhu saat ini")

def request_log_file():
    message = encrypt_message(json.dumps({"username": username, "password": password}))
    response = requests.post(f"{SERVER_URL}/request_logfile", data=message)
    if response.status_code == 200:
        with open(f"log_{username}.txt", "wb") as log_file:
            log_file.write(response.content)
        messagebox.showinfo("Request Log File", "Log file berhasil disimpan di direktori yang sama dengan aplikasi.")
    else:
        messagebox.showerror("Request Log File", "Gagal meminta log file.")

```

Selain itu juga terdapat fungsi login dan logout. Untuk fungsi login, terdapat beberapa langkah yang dilakukan yaitu

1. Ambil input pengguna:
 - username dan password diambil dari entry_username dan entry_password.

2. Kirim permintaan login:
 - Mengirim permintaan POST ke server dengan endpoint /login, membawa username dan password dalam format JSON.
3. Tanggapan server:
 - response menerima tanggapan dari server.
4. Proses tanggapan:
 - Mencoba mengurai tanggapan JSON (response.json()).
 - Jika status kode 200 (OK):
 - Mendapatkan kunci publik (public_key) dari server dan menyimpannya dalam format PEM.
 - Mengubah frame UI dari frame login (logged_in_frame) ke frame utama (main_frame).
 - Menampilkan pesan berhasil.
 - Jika status kode bukan 200:
 - Menampilkan pesan error sesuai dengan tanggapan server atau pesan error standar.
 - Menangkap kesalahan parsing JSON dan menampilkan pesan error.

```
def login():
    global username, password
    global public_key
    username = entry_username.get()
    password = entry_password.get()
    response = requests.post(f"{SERVER_URL}/login",
    json={"username": username, "password": password})
    try:
        response_json = response.json()
        if response.status_code == 200:
            public_key_pem =
response.json()["public_key"].encode('utf-8')
            public_key =
serialization.load_pem_public_key(public_key_pem)
            messagebox.showinfo("Login", "Login berhasil!")
            logged_in_frame.pack_forget()
            main_frame.pack()
        else:
            error_message = response_json.get("error", "Error
tidak dikenal")
            if response.status_code == 401 and error_message ==
"Invalid username or password":
                error_message = "Nama pengguna atau kata sandi
tidak valid"
            messagebox.showerror("Login Gagal", error_message)
    except json.decoder.JSONDecodeError:
        messagebox.showerror("Login Gagal", "Server error. Silakan
coba lagi nanti.")
```

Untuk fungsi logout, langkah-langkah yang dilakukan adalah mengirim permintaan logout terenkripsi ke server, dan jika berhasil, mengatur ulang status login dan menampilkan UI login.

```
def logout():
    global username, password
```

```

        message = encrypt_message(json.dumps({"username": username,
"password": password}))
        response = requests.post(f"{SERVER_URL}/logout",
data=message)
        if response.status_code == 200:
            messagebox.showinfo("Logout", "Logout berhasil!")
            username = None
            password = None
            main_frame.pack_forget()
            logged_in_frame.pack()
        else:
            messagebox.showerror("Logout", "Gagal",
response.json().get("error", "Error tidak dikenal"))

```

Kemudian terdapat pengaturan user interface dimana Pengaturan antarmuka grafis menggunakan tkinter. Terdapat `logged_in_frame`: Frame untuk form login serta `main_frame`: Frame untuk menu utama setelah login. Pada awalnya hanya menampilkan form login (`logged_in_frame.pack()`) dan menyembunyikan menu utama (`main_frame.pack_forget()`). Setelah login berhasil, frame login disembunyikan dan frame utama ditampilkan. Tampilan UI diatur untuk berwarna kebiruan, serta tombol-tombol diatur secara vertikal dengan ukuran yang sama besar.

```

window = tk.Tk()
window.title("Temperature Control App")
window.geometry("400x300")
window.configure(bg='#add8e6')

logged_in_frame = tk.Frame(window, bg='#add8e6')
main_frame = tk.Frame(window, bg='#add8e6')

label_username = tk.Label(logged_in_frame, text="Username",
bg='#add8e6')
label_password = tk.Label(logged_in_frame, text="Password",
bg='#add8e6')
entry_username = tk.Entry(logged_in_frame)
entry_password = tk.Entry(logged_in_frame, show="*")
button_login = tk.Button(logged_in_frame, text="Login",
command=login)
label_username.grid(row=0, column=0, padx=5, pady=5)
label_password.grid(row=1, column=0, padx=5, pady=5)
entry_username.grid(row=0, column=1, padx=5, pady=5)
entry_password.grid(row=1, column=1, padx=5, pady=5)
button_login.grid(row=2, columnspan=2, padx=5, pady=5)

logged_in_frame.pack()

button_show_data = tk.Button(main_frame, text="Tampilkan Data
Suhu", command=show_temperature_data)
button_show_graph = tk.Button(main_frame, text="Tampilkan
Grafik", command=show_temperature_graph)
button_open_remote = tk.Button(main_frame, text="Kontrol Suhu
AC", command=open_remote_control)
button_get_current_temp = tk.Button(main_frame, text="Tampilkan
Suhu Saat Ini", command=get_current_temperature)
button_request_logfile = tk.Button(main_frame, text="Request Log
File", command=request_log_file)
button_logout = tk.Button(main_frame, text="Logout",
command=logout)

```

```

buttons = [button_show_data, button_show_graph,
button_open_remote, button_get_current_temp,
button_request_logfile, button_logout]

for i, button in enumerate(buttons):
    button.grid(row=i, column=0, padx=5, pady=5, sticky='ew')

main_frame.pack_forget()
window.mainloop()

```

- **Implementasi Client 2 (NodeMCU)**

Rancangan program pada sistem mikrokontroler Nodemcu V3 diimplementasikan dengan kode program yang ditulis menggunakan bahasa C++ pada aplikasi pengembang program Arduino IDE 2.3.2. Kode program yang ditulis, didasarkan pada diagram alir program yang telah dibuat di awal pada rancangan sistem. Pada rancangan sistem, program dibuat dengan sistem yang memiliki dua fungsi yaitu GET dan POST. Pada fungsi GET, sistem menerima pesan berupa perintah pengiriman data suhu realtime dari server. Fungsi GET diimplementasikan dengan menggunakan program looping pada kode program. Selanjutnya pada fungsi POST, sistem mengirimkan data suhu secara berkala dalam rentang waktu 30 menit. Fungsi POST diimplementasikan dengan menggunakan program interupsi timer pada Nodemcu. Berikut merupakan cuplikan dari kode program dari pembuatan masing-masing fungsi tersebut:

Fungsi GET :

```

void loop() {
    // Baca dan simpan sensor
    temp = dht.readTemperature();

    // Menerima perintah permintaan suhu saat ini (realtime)
    String currentLine = ""; //variabel perintah
    while (client.connected()) {
        if (client.available()) {
            char c = client.read(); //baca karakter huruf dari serial
            if (currentLine == "generate_temperature") {
                Serial.print("server meminta data suhu --> ");
                sendDataToServer(temp, true);
                currentLine = "";
                break;
            }
            if (c != '\r') {
                currentLine += c;
            }
        }
    }
    delay(100);
}

```

Pada cuplikan kode program diatas dapat dijelaskan bahwa selama Nodemcu terhubung dengan server, maka Nodemcu selalu melakukan pembacaan sensor suhu dari sensor DHT serta pengecekan penerimaan perintah dari server melalui data serial

berupa pesan karakter “generate_temperature” secara berulang dengan periode waktu 100 ms. Ketika Nodemcu menerima pesan “generate_temperature” maka Nodemcu akan mengirimkan data suhu permintaan server secara realtime dan menampilkan informasi data yang dikirim melalui serial monitor.

Fungsi Interupsi Timer :

Void Setup :

```
// Setup timer dengan interval 1 detik (1000ms)
ticker.attach(1, onTimer);
```

Kode di atas merupakan cuplikan kode untuk mengaktifkan interupsi timer pada Nodemcu dengan interval 1 detik.

Void Timer :

```
// interupsi timer & data sensor di kirim sesuai timer
void ICACHE_RAM_ATTR onTimer() {
    detik++;
    if (detik >= 60) { //jumlah detik dalam 1 menit
        detik = 0;
        menit++;
    }
    if (menit >= 30){ //jumlah menit
        menit = 0;
        // Kirim data ke server melalui socket
        sendDataToServer(temp, false); //kirim data suhu berkala 30
    }
    menit
}
```

Cuplikan kode program di atas merupakan penghitung waktu berdasarkan interval interupsi timer (1 detik) untuk menghasilkan waktu berkala dengan periode 30 menit.

Void sendDataTo Server :

```
// void baca suhu dan kirim data ke server
void sendDataToServer(float temp, bool onDemand) {
    if (client.connected()) {
        String data = String(temp);
        if(onDemand){
            client.print(data + " (onDemand)"); //Kirim data berdasar
            permintaan
        } else {
            client.print(data); //Kirim data berdasar waktu berkala
        }
        Serial.print("Data sent: " + data); //tampilan di serial monitor
        Serial.println(" Celcius");
    }
}
```

Cuplikan kode di atas merupakan kode untuk mengirimkan data suhu ke server. Dalam pengirimannya, identitas data suhu dibedakan berdasarkan jenisnya. Jika atas dasar permintaan server maka data suhu dikirim dengan tambahan identitas pesan “onDemand”. Sedangkan jika atas dasar pengiriman suhu secara berkala (reguler interupsi) maka data yang dikirimkan hanyalah data suhu saja tanpa ada tambahan identitas.

F. Hasil Ujicoba

Uji coba Server:

Uji coba server dilakukan dengan menghubungkan Client dummy (untuk NodeMCU) dan menjalankan kode Client 1 di komputer yang sama dengan server. Berikut merupakan hasil uji coba server. Terlebih dahulu, diperlihatkan seluruh isi file data temperature, user, dan ac temperature.

```
Perancangan Perangkat Lunak Jaringan > Tugas Besar > data_060624.csv
1 |

Perancangan Perangkat Lunak Jaringan > Tugas Besar > user.csv
1 admin,admin
2 user1,user1
3 user2,user2

Perancangan Perangkat Lunak Jaringan > Tugas Besar > ac_temperature.txt
1 24
```

Gambar 8 Isi File data temperature, user dan ac temperature di awal

```
C:\WINDOWS\system32\cmd. X + v
D:\Perancangan Perangkat Lunak Jaringan\Tugas Besar>python Server.py
Server NodeMCU dimulai di port 11000
* Serving Flask app 'Server'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production d
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running on http://192.168.1.10:80
Press CTRL+C to quit
```

Gambar 9 Menjalankan Server

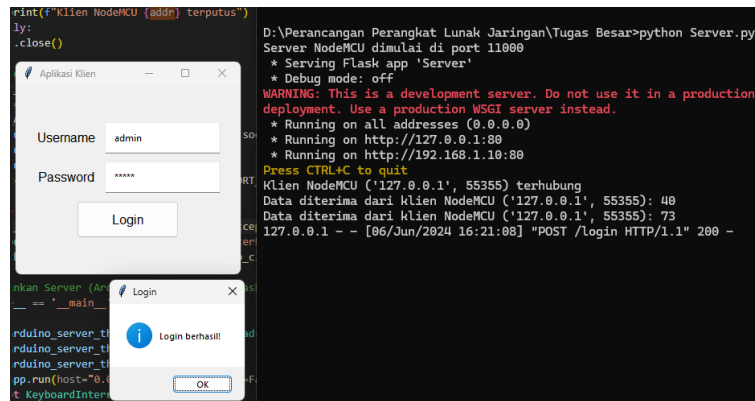
Client dummy berfungsi untuk mengirimkan data nilai temperature random setiap 1 menit (untuk mempermudah pengamatan) dan dapat menerima data “generate_temperature” kemudian mengirimkan nilai temperature random ke server.

```
D:\Perancangan Perangkat Lunak Jaringan\Tugas Besar>python Server.py
Server NodeMCU dimulai di port 11000
* Serving Flask app 'Server'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production dep
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running on http://192.168.1.10:80
Press CTRL+C to quit
Klien NodeMCU ('127.0.0.1', 55355) terhubung

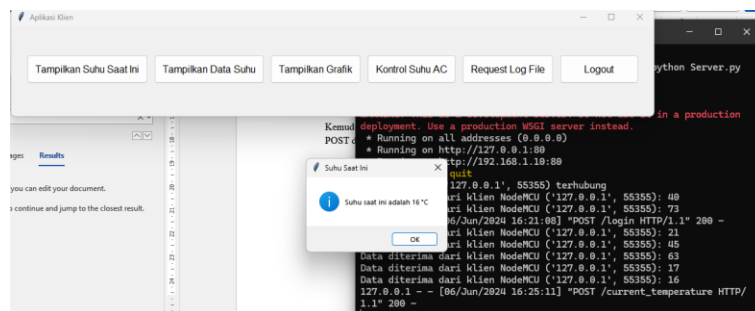
PS D:\Perancangan Perangkat Lunak Jaringan\Tugas Besar> python .\Client.py
```

Gambar 10 Menjalankan Client dummy NodeMCU sebagai pengirim data suhu

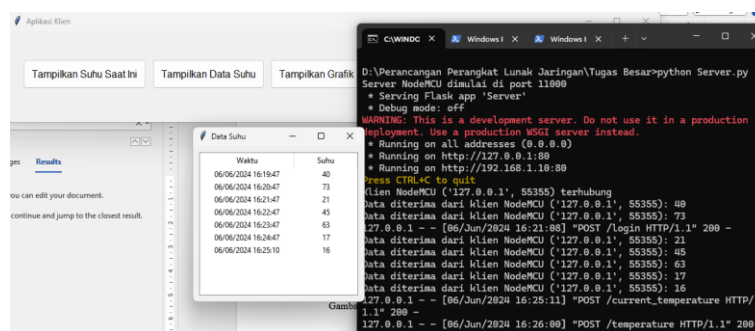
Kemudian, dijalankan Client 1 (Raspberry Pi) dan dicoba login untuk melihat apakah POST dari Client 1 berhasil diterima oleh server dan hasilnya dapat dilihat Gambar 11.



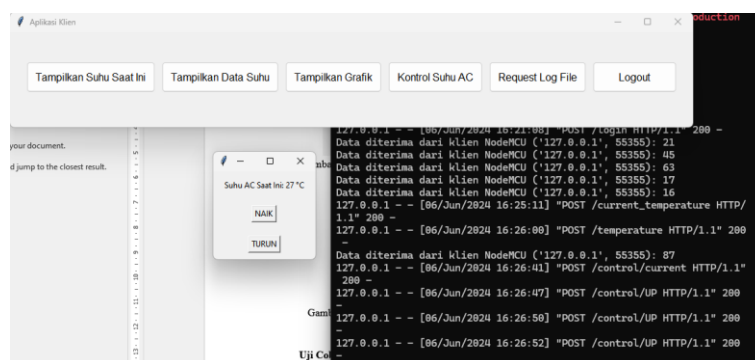
Gambar 11 Login pada Client 1 untuk melihat response pada server



Gambar 12 Tampilkan suhu saat ini pada Client 1 untuk melihat response pada server



Gambar 13 Tampilkan data suhu pada Client 1 untuk melihat response pada server



Gambar 14 Kontrol Suhu Naik tiga kali pada Client 1 untuk melihat response pada server

Dengan data pada server yang berubah seperti berikut.

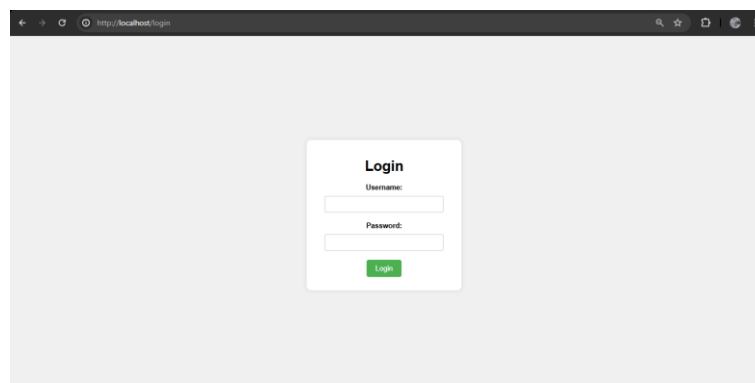
```
Perancangan Perangkat Lunak Jaringan > Tugas Besar > data_060624.csv
1 06/06/2024 16:19:47,40
2 06/06/2024 16:20:47,73
3 06/06/2024 16:21:47,21
4 06/06/2024 16:22:47,45
5 06/06/2024 16:23:47,63
6 06/06/2024 16:24:47,17
7 06/06/2024 16:25:10,16 (onDemand)
8 06/06/2024 16:25:47,87
9 06/06/2024 16:26:47,50
10

Perancangan Perangkat Lunak Jaringan > Tugas Besar > ac_temperature.txt
1 27

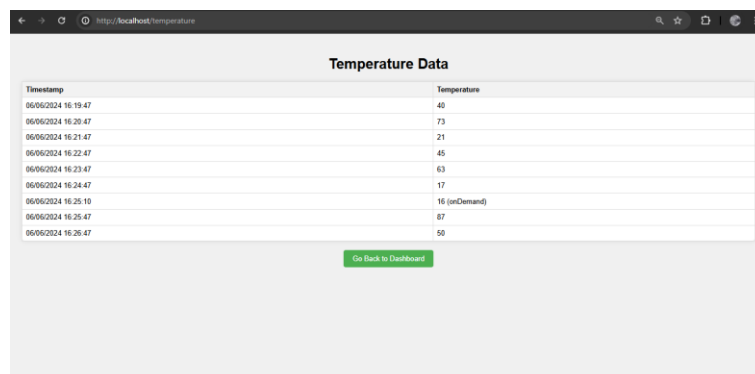
Perancangan Perangkat Lunak Jaringan > Tugas Besar > log_admin.txt
1 Log File Client admin:
2 127.0.0.1 - - [06/Jun/2024 16:21:08] "POST /login HTTP/1.1" 200 -
3 127.0.0.1 - - [06/Jun/2024 16:25:11] "POST /current_temperature HTTP/1.1" 200 -
4 127.0.0.1 - - [06/Jun/2024 16:26:00] "POST /temperature HTTP/1.1" 200 -
5 127.0.0.1 - - [06/Jun/2024 16:26:41] "POST /control/current HTTP/1.1" 200 -
6 127.0.0.1 - - [06/Jun/2024 16:26:47] "POST /control/UP HTTP/1.1" 200 -
7 127.0.0.1 - - [06/Jun/2024 16:26:50] "POST /control/UP HTTP/1.1" 200 -
8 127.0.0.1 - - [06/Jun/2024 16:26:52] "POST /control/UP HTTP/1.1" 200 -
9
```

Gambar 15 Isi File data temperature, ac temperature dan log_admin di akhir

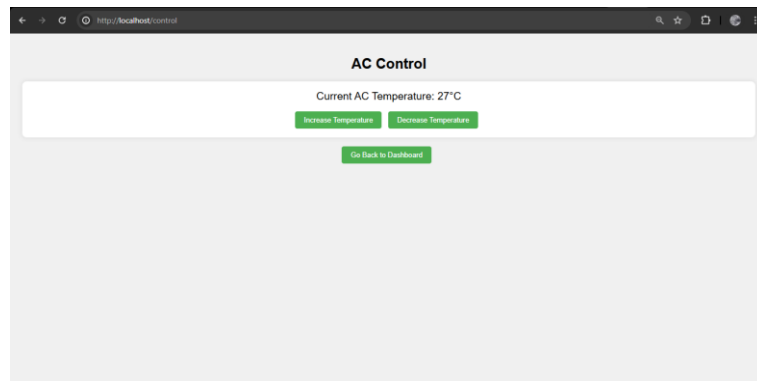
Dapat dilihat bahwa secara fungsional, server sudah dapat berfungsi dengan baik untuk keseluruhan fungsi. Untuk tampilan web server, dapat dilihat pada gambar berikut.



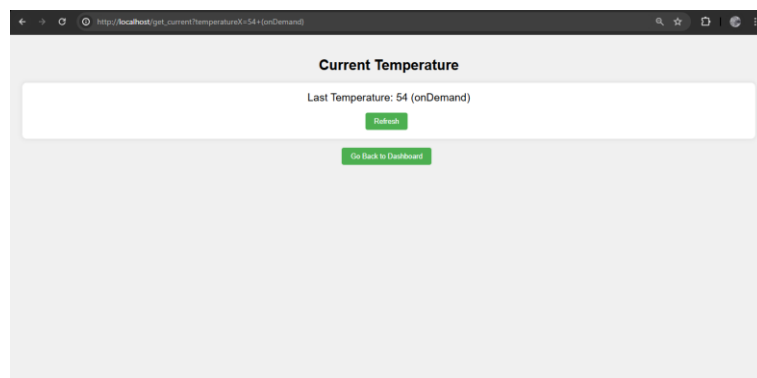
Gambar 16 Tampilan login webserver



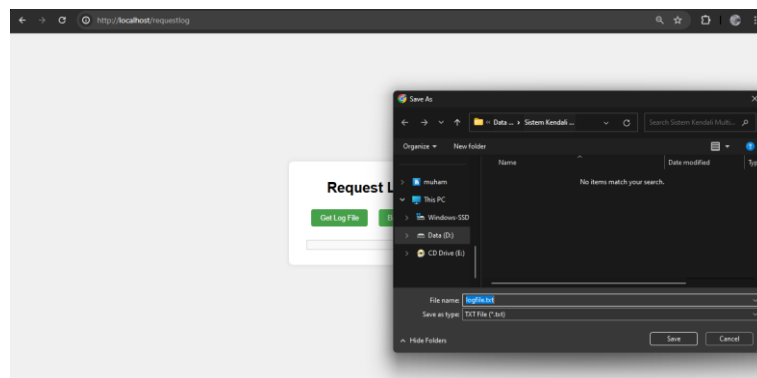
Gambar 17 Tampilan data temperatur webserver (data dari Client Dummy)



Gambar 18 Tampilan Kontrol Temperatur AC webserver



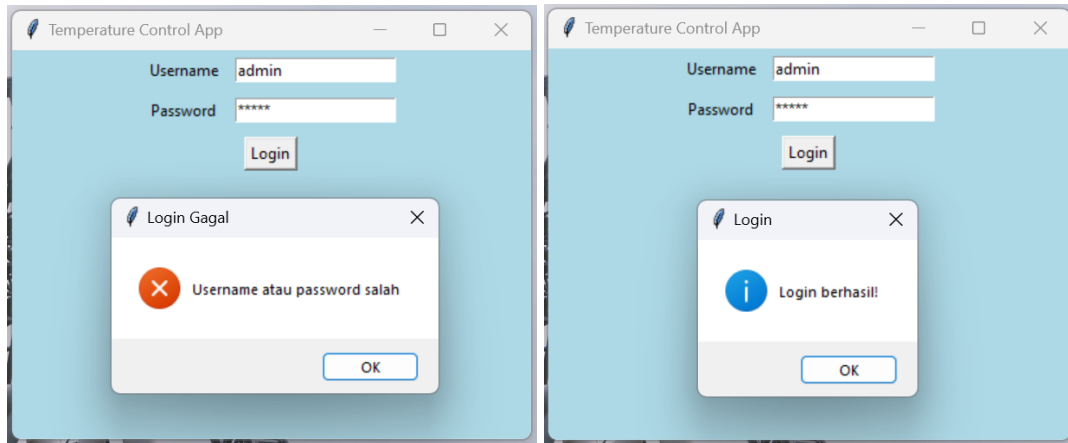
Gambar 19 Tampilan data temperature (onDemand) webserver (data dari Client Dummy)



Gambar 20 Tampilan request log file webserver

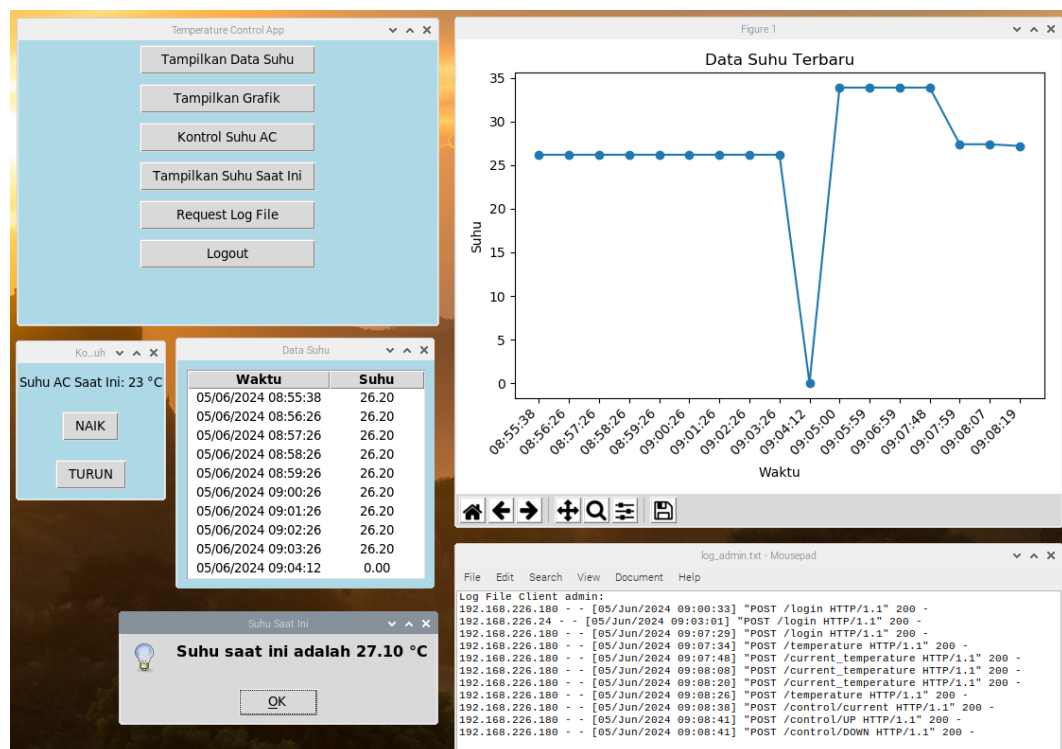
Uji Coba Client 1:

Untuk uji coba tampilan pada Client 1, pertama-tama dilakukan koneksi antara Raspberry Pi 3 ke laptop dengan menggunakan remote desktop connection. Setelah koneksi berhasil, maka dibuka program client 1 yang terdapat pada Raspberry Pi 3. Tampilan awal Client 1 adalah sebagai berikut:



Gambar 21 Tampilan Login Client 1 (Kiri: Login Gagal ; Kanan: Login Berhasil)

Berdasarkan gambar 21 terlihat bahwa mekanisme login berhasil diimplementasi. Terlihat bahwa jika terdapat masukan username dan/atau password yang salah, maka akan muncul keterangan login gagal, sedangkan jika masukan username dan password benar, maka login berhasil dan pengguna otomatis masuk ke menu utama.

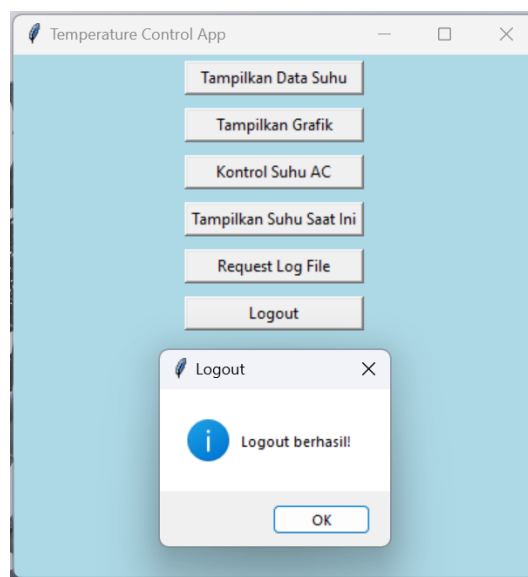


Gambar 22 Tampilan Menu Utama beserta Data Hasil Fungsi

Berdasarkan gambar 22 terlihat menu utama pada client 1 (window pojok kiri atas). Terdapat 5 fungsi yang dapat dilakukan yaitu

1. Menampilkan data suhu (window data suhu). Pada window ini terlihat tabel data suhu ruangan beserta timestamp, yang dikirim secara berkala (pada pengujian ini dilakukan pengiriman data setiap 1 menit) oleh Client 2 kepada server, yang kemudian datanya ditampilkan pada Client 1.

2. Menampilkan grafik suhu (window Figure 1). Pada window ini, data yang terdapat pada tabel data suhu diubah kedalam bentuk grafik agar memudahkan observasi naik turunnya suhu ruangan.
3. Kontrol suhu AC (window kontrol suhu). Pada window ini tertera suhu AC saat ini, dan suhu tersebut dapat dinaikkan atau diturunkan per 1°C .
4. Menampilkan suhu saat ini (window suhu saat ini). Pada window ini tertera data suhu secara real-time.
5. Request logfile (window log_admin.txt). Pada window ini, dibuka file log_admin.txt yang isinya merupakan history aktivitas yang dilakukan user Client 1, dalam konteks ini adalah user admin. Terlihat perintah-perintah yang dilakukan mulai dari login, meminta data suhu, suhu saat ini, control up, control down, dsb.

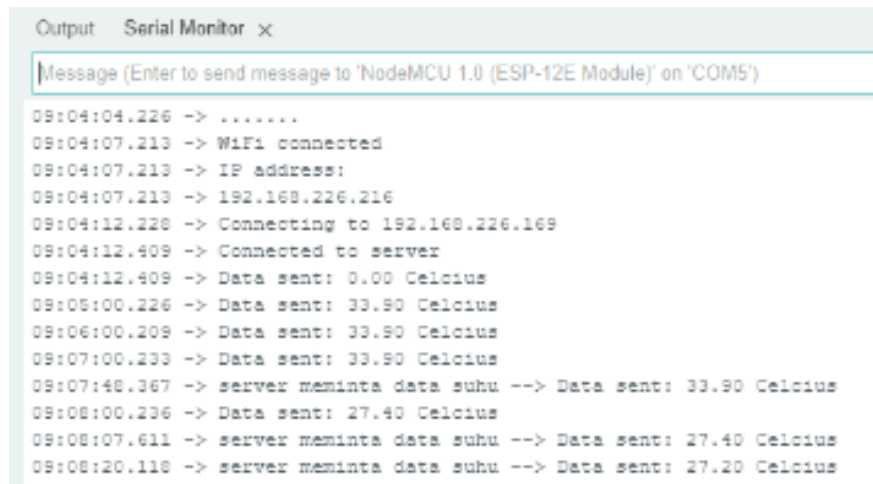


Gambar 23 Tampilan Logout

Berdasarkan gambar 23 terlihat bahwa proses logout berhasil, kemudian pengguna akan kembali ke laman login.

Uji Coba Client 2:

Berikut merupakan hasil uji coba tampilan dari serial monitor Arduino IDE pada komputer yang mampu memberikan gambaran bahwa sistem telah bekerja secara baik dengan menampilkan informasi data suhu yang dikirimkan ke server sesuai dengan permintaan server dan secara berkala (1 menit untuk mempermudah pengamatan):



```
Output Serial Monitor x
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')
09:04:04.226 -> .....
09:04:07.213 -> WiFi connected
09:04:07.213 -> IP address:
09:04:07.213 -> 192.168.226.216
09:04:12.228 -> Connecting to 192.168.226.169
09:04:12.409 -> Connected to server
09:04:12.409 -> Data sent: 0.00 Celcius
09:06:00.226 -> Data sent: 33.90 Celcius
09:06:00.209 -> Data sent: 33.90 Celcius
09:07:00.233 -> Data sent: 33.90 Celcius
09:07:48.367 -> server maminta data suhu --> Data sent: 33.90 Celcius
09:08:00.236 -> Data sent: 27.40 Celcius
09:08:07.611 -> server maminta data suhu --> Data sent: 27.40 Celcius
09:08:20.118 -> server maminta data suhu --> Data sent: 27.20 Celcius
```

Gambar 24 Hasil pengujian Client NodeMCU terhubung ke server

Pada tampilan gambar 24 terlihat di awal serial monitor ketika sistem nodemcu dihidupkan, maka sistem akan melakukan percobaan sambungan ke wifi dan server. Setelah sistem tersambung, maka akan mendapat pemberitahuan alamat IP dan telah tersambung ke server. Kemudian

G. Kesimpulan

Berdasarkan tahap-tahap yang telah dilakukan dimulai dari perancangan sistem, implementasi, dan ujicoba dapat disimpulkan bahwa sistem aplikasi pemantau suhu dan pengatur pendingin ruangan dapat bekerja dengan baik sesuai dengan fungsinya. Client 1 (RaspberryPi) dapat bekerja sebagai antarmuka pengguna dengan mampu menampilkan informasi data suhu serta mampu memberikan masukan pada sistem untuk menaikkan dan menurunkan suhu melalui kontrol pendingin ruangan. Client 2 dapat bekerja sebagai pemantau suhu ruangan serta mampu mengirimkan data suhu ruangan secara berkala atau atas dasar permintaan pengguna. Server dapat mengatur lalu lintas data serta memberikan layanan permintaan dan pemberian data antar Client.