LAMPIRAN

Surat Pengantar

UNIVERSITAS 17 AGUSTUS 1945 (UNTAG) SURABAYA
FAKULTAS TEKNIK

Kampus : Jl. Semolowaru No. 45 Surabaya 60118 Telp. +62 31 5931800 (hunting) Fax, +62 31 5927817

- Program Studi Teknik Industri
- Program Studi Teknik Mesin
- Program Studi Teknik Sipil
- Program Studi Arsitektur
- Program Studi Teknik Elektro
- Program Studi Teknik Informatika
- Program Studi Sistem dan Teknologi Informasi
- Program Studi Teknik Robotika & Kecerdasan Buatan
- Program Studi Magister Teknik Sipil
- Program Studi Magister Arsitektur
- Program Studi Program Profesi Insinyur

Homepage : ft.untag-sby.ac.id Email : uknik@untag-sby.ac.id

Nomor : **1827/K/FT/Akd/XI/2024** **Surabaya, 11 November 2024**
Lampiran : -
Perihal : **Penelitian Tugas Akhir**

Kepada Yth : **PT. PLN (PERSERO) UPT Surabaya**
Jl. Ketintang Baru No. 9 Ketintang Kec. Gayungan Surabaya

Dengan hormat,
Sebagai salah satu persyaratan untuk menyelesaikan studi pada program Strata 1, maka mahasiswa/mahasiswi diwajibkan untuk melakukan **Penelitian Tugas Akhir** sebagai penerapan teori dan praktek yang diperoleh selama masa studinya.
Sehubungan dengan hal tersebut, maka dengan ini kami mohon Bapak/Ibu berkenan untuk memberikan ijin kepada mahasiswa/mahasiswi sebagai berikut :

No	Nama	NBI	EMAIL	No.HP
1.	Riyan Agus S	1452100044	riyanagus649@gmail.com	082337907370
2.	Ifan Maulana	1452100045		
3.	Aswanda H	1452100054		

Program Studi Teknik Elektro
Guna melaksanakan **Penelitian Tugas Akhir** di :

"PT. PLN (PERSERO) UPT Surabaya"

yang akan dimulai pada : **Semester Gasal 2024/2025**
Demikian permohonan kami, atas perkenannya disampaikan terima kasih.

Dekan,

Dr. Ir. Sajiyo, M.Kes., IPU., ASEAN Eng.
NPP : 20410.90.0197

Surat Balasan Dari PLN

PLN

UIT JBM
UPT SURABAYA

Nomor : 0621/STH.01.04/F34050000/2024
Lampiran : 1 Lembar
Sifat : Segera - Biasa
Hal : Persetujuan Melaksanakan Penelitian

24 Desember 2024

Kepada

Yth. Universitas 17 Agustus 1945
(UNTAG) Surabaya
Fakultas Teknik

Menunjuk Surat Saudara Nomor; 1827/K/FT/Akd/XI/2024 tanggal 11 Nopember 2024 perihal Izin Penelitian Tugas Akhir , dengan ini kami sampaikan Ijin kepada mahasiswa/i sebagai berikut :

No.	Nama	NIM
1	Ifan Maulana	1452100045
2	Riyan Agus Setiawan	1452100044
3	Aswanda Harja	1452100054

Untuk melakukan pengambilan data baik melalui survey maupun pengukuran di **PT PLN (Persero) UPT Surabaya ULTG Surabaya Utara Gardu Induk Surabaya Selatan** mulai tanggal **06 Januari s/d 03 Februari 2025** dengan persyaratan sebagai berikut :

1. Mahasiswa mengisi dan menanda tangani surat pernyataan 1 (satu) lembar bermaterai Rp. 10.000,-
2. Mahasiswa yang bersangkutan agar mematuhi peraturan / ketentuan yang berlaku di PT PLN (Persero) sehingga faktor-faktor kerahasiaan harus benar-benar diutamakan. Semua biaya perjalanan, penginapan, makan dan lain sebagainya tidak menjadi tanggungan PT PLN (Persero) UPT Surabaya.
3. Mahasiswa wajib mentaati protokol kesehatan dan 5M selama melaksanakan PKL / penelitian.
4. Mahasiswa sanggup tidak membocorkan hal-hal yang bersifat rahasia perusahaan PT PLN (Persero) UIT JBM UPT Surabaya, dan bahan yang diperoleh dalam Training / Praktek Kerja / Riset, dan tidak mempergunakan untuk hal-hal yang dapat merugikan PT PLN (Persero) UIT JBM UPT Surabaya
5. Untuk informasi lebih lanjut dapat menghubungi PT PLN (Persero) UPT Surabaya Cq. Bidang ADM & UMUM.

Demikian kami sampaikan, atas perhatian dan kerjasamanya kami ucapkan terimakasih.

MANAGER UNIT PELAKSANA
TRANSMISI SURABAYA,

GUNAWAN WIJAYA

Tembusan:

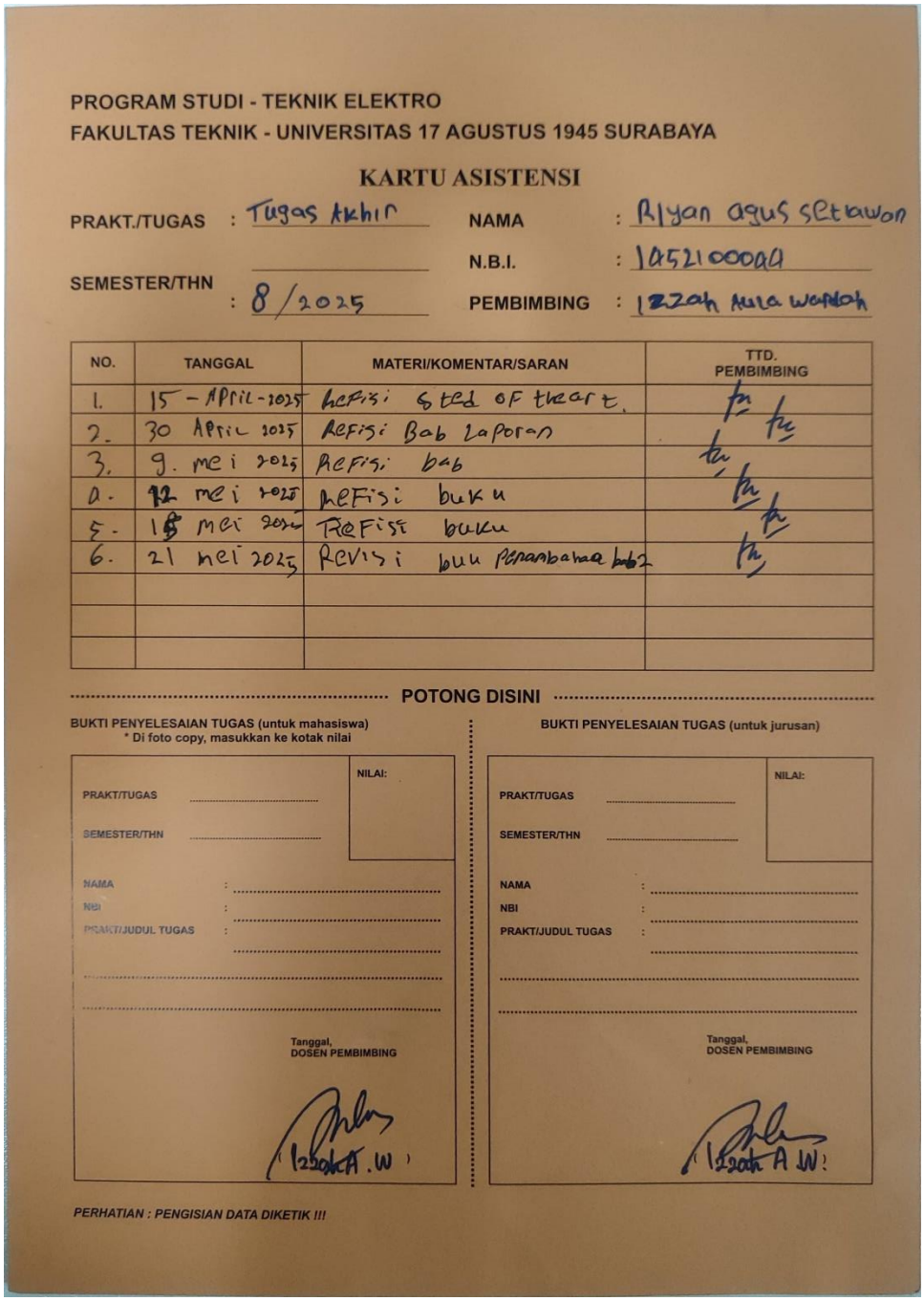
1. MUL ULTG SURABAYA UTARA ULTG SURABAYA UTARA PLN

Jl. Kalimantang Baru No 9, Surabaya 60231
T (031)8285555, 8285506 F (031)8284400 W www.pln.co.id

Paraf 9 1 dan 2

Kartu Asistensi

Dosen Pembimbing 1



PROGRAM STUDI - TEKNIK ELEKTRO
FAKULTAS TEKNIK - UNIVERSITAS 17 AGUSTUS 1945 SURABAYA

KARTU ASISTENSI

PRAKT./TUGAS : Tugas Akhir
NAMA : Riyan agus setiawan
SEMESTER/THN : 8/2025
N.B.I. : 1452100044
PEMBIMBING : Izzah Aula Wardah

NO.	TANGGAL	MATERI/KOMENTAR/SARAN	TTD. PEMBIMBING
1.	15 - April - 2025	Refisi & ttd OF theart.	
2.	30 April 2025	Refisi Bab Laporan	
3.	9. mei 2025	Refisi bab	
4.	12 mei 2025	Refisi buku	
5.	19 mei 2025	Refisi buku	
6.	21 mei 2025	Revisi buku Penambahan bab2	

POTONG DISINI

BUKTI PENYELESAIAN TUGAS (untuk mahasiswa)
* Di foto copy, masukkan ke kotak nilai

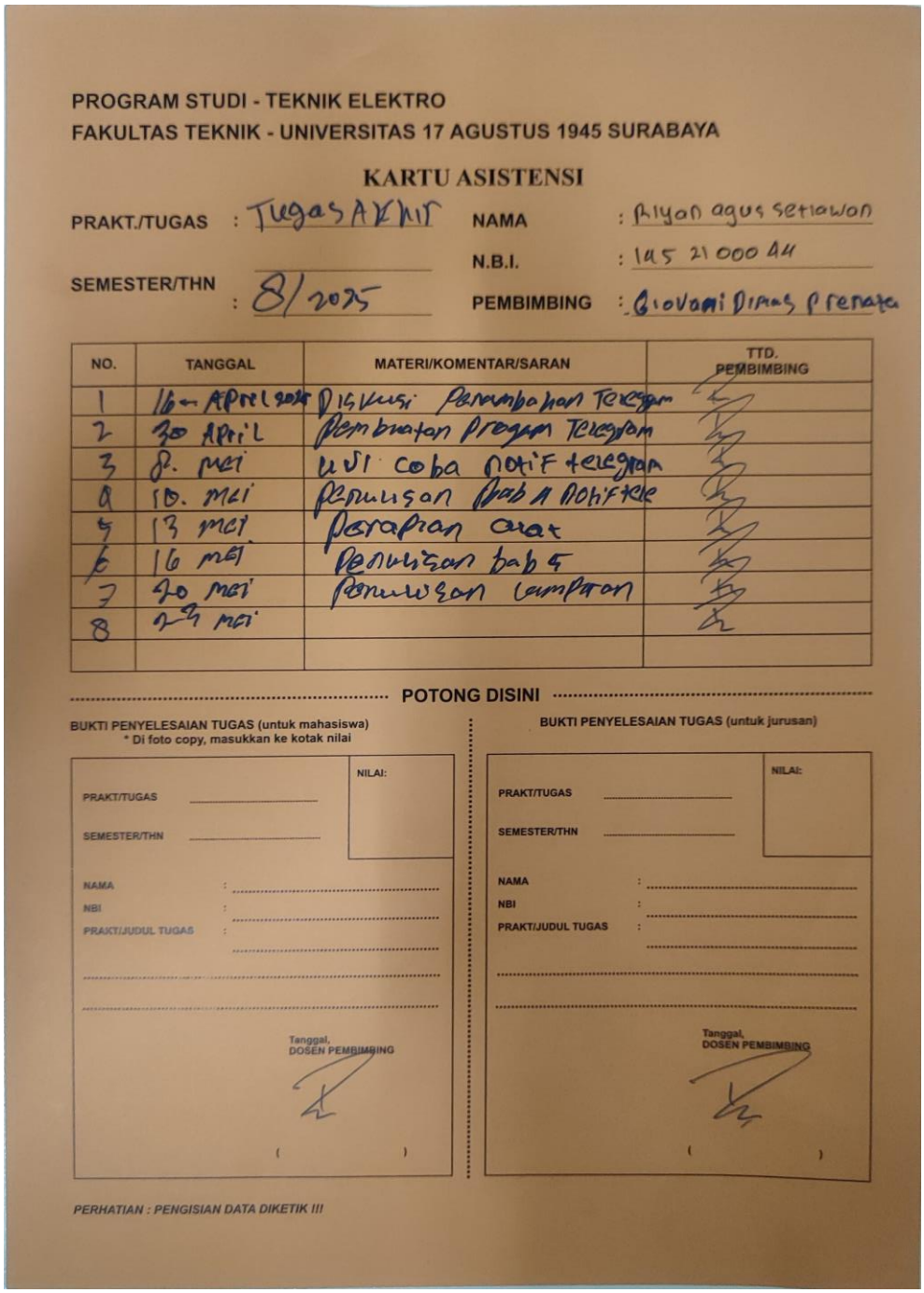
PRAKT/TUGAS .....
SEMESTER/THN .....
NILAI:
NAMA : .....
NBI : .....
PRAKT/JUDUL TUGAS : .....
Tanggal,
DOSEN PEMBIMBING
(Izzah A.W)

BUKTI PENYELESAIAN TUGAS (untuk jurusan)

PRAKT/TUGAS .....
SEMESTER/THN .....
NILAI:
NAMA : .....
NBI : .....
PRAKT/JUDUL TUGAS : .....
Tanggal,
DOSEN PEMBIMBING
(Izzah A.W)

PERHATIAN : PENGISIAN DATA DIKETIK !!!

Dosen Pembimbing 2



PROGRAM STUDI - TEKNIK ELEKTRO
FAKULTAS TEKNIK - UNIVERSITAS 17 AGUSTUS 1945 SURABAYA

KARTU ASISTENSI

PRAKT./TUGAS : Tugas AKHIR
NAMA : Riyan agus setiawan
N.B.I. : 145 21 000 44
SEMESTER/THN : 8/2025
PEMBIMBING : Giovani Dimas Prenata

NO.	TANGGAL	MATERI/KOMENTAR/SARAN	TTD. PEMBIMBING
1	16 April 2025	Diskusi Penambahan Telegram	
2	30 April	Pembuatan Program Telegram	
3	8. Mei	Uji coba notif telegram	
4	10. Mei	Penulisan Bab 4 notif tele	
5	13 Mei	Perapian alat	
6	16 Mei	Penulisan bab 5	
7	20 Mei	Penulisan kesimpulan	
8	24 Mei		

POTONG DISINI

BUKTI PENYELESAIAN TUGAS (untuk mahasiswa)
* Di foto copy, masukkan ke kotak nilai

PRAKT/TUGAS
SEMESTER/THN
NILAI:
NAMA :
NBI :
PRAKT/JUDUL TUGAS :
Tanggal,
DOSEN PEMBIMBING
( )

BUKTI PENYELESAIAN TUGAS (untuk jurusan)

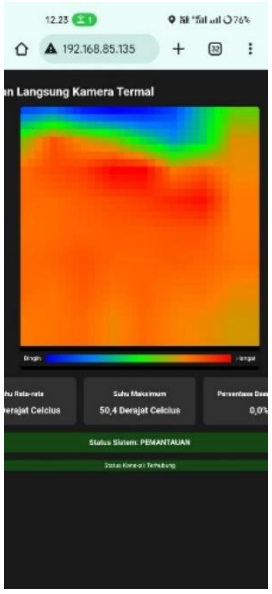
PRAKT/TUGAS
SEMESTER/THN
NILAI:
NAMA :
NBI :
PRAKT/JUDUL TUGAS :
Tanggal,
DOSEN PEMBIMBING
( )

PERHATIAN : PENGISIAN DATA DIKETIK !!!

Pengambilan Data Menggunakan Alat Thermal AMG8833

Foto dokumentasi pengambilan data

Foto alat yang digunakan pengambilan data thermal



Gambar screenshot pada hp yang terhubung di alat AMG8833 untuk pengambilan data thermal

Foto Dokumentasi Implementasi Alat Kamera Termal
Di Bay Trafo 3 150/20 KV 60 MVA

Tabel Data Hasil Training Singgle Perceptron

Iterasi ke-	Data latih ke-	Output	Error	Bobot w[0]	Bobot w[1]	Bobot w[2]
0				-1,07E+13	0.169078	0.169078
1	1	138.354	-1	-185.518	-797.537	-0.0420078
	2	-627.021	0			
	3	-811.096	0			
	4	-317.521	1	647.129	-593.147	0.0579922
	5	-599.907	0			
	6	-579.6	0			
	7	575.706	0			
	8	-574.416	0			
	9	-591.923	0			
	10	-613.473	0			
2	1	-362.949	0			
	2	-179.242	0			
	3	-603.142	0			
	4	417.656	0			
	5	-599.907	0			
	6	-579.6	0			
	7	-575.706	0			
	8	-574.416	0			
	9	-591.923	0			
	10	-613.473	0			
3	1	-362.949	0			
	2	-179.242	0			
	3	-603.142	0			
	4	417.656	0			
	5	-599.907	0			
	6	-579.6	0			
	7	-575.706	0			
	8	-574.416	0			
	9	-591.923	0			
	10	-613.473	0			
4	1	-362.949	0			
	2	-179.242	0			

	3	-603.142	0			
	4	417.656	0			
	5	-599.907	0			
	6	-579.6	0			
	7	-575.706	0			
	8	-574.416	0			
	9	-591.923	0			
	10	-613.473	0			
5	1	-362.949	0			
	2	-179.242	0			
	3	-603.142	0			
	4	417.656	0			
	5	-599.907	0			
	6	-579.6	0			
	7	-575.706	0			
	8	-574.416	0			
	9	-591.923	0			
	10	-613.473	0			
6	1	-362.949	0			
	2	-179.242	0			
	3	-603.142	0			
	4	417.656	0			
	5	-599.907	0			
	6	-579.6	0			
	7	-575.706	0			
	8	-574.416	0			
	9	-591.923	0			
	10	-613.473	0			

Program uji AI menentukan presentase dan klasifikasi single perceptron menggunakan software visual studio

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <cstdint>
#include <string>

using namespace std;
// Struktur untuk menyimpan header file BMP
#pragma pack(push, 1)
struct BMPHeader
{
   uint16_t fileType;                  // File type, harus 'BM'
   uint32_t fileSize;                  // Ukuran file
   uint16_t reserved1;                 // Reserved
   uint16_t reserved2;                 // Reserved
   uint32_t offsetData;          // Offset ke data piksel
};
struct BMPInfoHeader
{
   uint32_t size;                      // Ukuran header info
   int32_t width;                            // Lebar gambar
   int32_t height;                           // Tinggi gambar
   uint16_t planes;                    // Jumlah plane (harus 1)
   uint16_t bitCount;                  // Bit per piksel (24 untuk RGB)
   uint32_t compression;               // Kompresi (0 = none)
   uint32_t sizeImage;                 // Ukuran data gambar
   int32_t xPixelsPerMeter;      // Resolusi horizontal
   int32_t yPixelsPerMeter;      // Resolusi vertikal
   uint32_t colorsUsed;          // Jumlah warna digunakan
   uint32_t colorsImportant;     // Warna penting
};
#pragma pack(pop)
// Fungsi untuk membaca file BMP
bool readBMP(const string& filename, vector<uint8_t>& pixelData, int& width,
int& height)
{
   ifstream file(filename, ios::binary);
   if (!file)
   {
```

```
        cerr << "Error: Tidak dapat membuka file!" << endl;
        return false;
    }
    BMPHeader fileHeader;
    BMPInfoHeader infoHeader;
    // Baca header file BMP
    file.read(reinterpret_cast<char*>(&fileHeader), sizeof(fileHeader));
    file.read(reinterpret_cast<char*>(&infoHeader), sizeof(infoHeader));
    // Periksa apakah file benar-benar BMP 24-bit
    if (fileHeader.fileType != 0x4D42 || infoHeader.bitCount != 24)
    {
        cerr << "Error: File bukan BMP 24-bit!" << endl;
        return false;
    }
    width = infoHeader.width;
    height = infoHeader.height;
    // Pindah ke data piksel
    file.seekg(fileHeader.offsetData, ios::beg);
    // Hitung jumlah piksel
    size_t dataSize = infoHeader.sizeImage;
    if (dataSize == 0)
    {
        dataSize = width * height * 3; // RGB = 3 byte per piksel
    }
    pixelData.resize(dataSize);
    file.read(reinterpret_cast<char*>(pixelData.data()), dataSize);
    return true;
}
void calculateColorPercentages(const vector<uint8_t>& pixelData, int
totalPixelCount, double& redPercentage, double& otherPercentage)
{
    int redPixelCount = 0;
    int otherPixelCount = 0;
    for (size_t i = 0; i < pixelData.size(); i += 3) {
        uint8_t blue = pixelData[i];
        uint8_t red = pixelData[i + 2];

        if (red > 200 && green < 100 && blue < 100) {
            redPixelCount++;
        }
        else {
            otherPixelCount++;
        }
    }
```

```
    redPercentage = (double)redPixelCount / totalPixelCount * 100.0;
    otherPercentage = (double)otherPixelCount / totalPixelCount * 100.0;
}
float d_rand(void)
{
    return(((float)(rand() % 32767) / (32767.0 - 0.5) * 2.0));
}
int main()
{
    string filename;
    int i, j, out, ERR;
    float x[3][10] = { {0,0,0,0,0,0,0,0,0,0},
                       {0,0,0,0,0,0,0,0,0,0},
                       {1,1,1,1,1,1,1,1,1,1} };
    int T[10] = { 0,0,0,0,0,0,0,0,0,0 };
    float w[3], O, LR = 0.1, init = 0.15;

    //inisialisasi bobot
    for (i = 0; i < 3; i++)
    {
        w[i] = init * d_rand();
        cout << "bobot awal w[" << i << "] = " << w[1] << endl;
    }
    // ekstraksi fitur data trainning
    for (int a = 1; a <= 10; a++)
    {
        cout << "Masukkan nama file BMP data trainning ke-" << a << ": ";
        cin >> filename;
        vector<uint8_t> pixelData;
        int width, height;

        if (!readBMP(filename, pixelData, width, height))
        {
            return -1;
        }
        double redPercentage, otherPercentage;
        int totalPixelCount = width * height;

        calculateColorPercentages(pixelData, totalPixelCount, redPercentage,
otherPercentage);

        cout << "Nama file: " << filename << endl;
        cout << "Persentase warna merah: " << redPercentage << "%" << endl;
        x[0][a - 1] = redPercentage;
```

```
        cout << "Persentase warna selain merah: " << otherPercentage << "%" <<
endl;
        x[1][a - 1] = otherPercentage;
        if (redPercentage >= 50)
        {
            T[a - 1] = 1;
        }
    }
    //training single perceptron
    for (i = 1; i <= 6; i++)
    {
        cout << endl << "__________training ke - " << i << endl;
        for (j = 0; j < 10; j++)
        {
            O = x[0][j] * w[0] + x[1][j] * w[1] + x[2][j] * w[2];
            cout << "Nilai Treshold = " << T[j] << endl;
            if (O > 0.0)
                out = 1;
            else
                out = 0;
            ERR = T[j] - out;
            cout << "out = " << out << endl;
            cout << "Error = " << ERR << endl;
            if (ERR != 0)
            {
                w[0] = w[0] + LR * x[0][j] * ERR;
                cout << "___Update bobot" << endl;
                cout << "___bobot w[0] = " << w[0] << endl;
        }
        cout << i << ":" << ERR << endl;
    }
    cut << "Masukkan nama file BMP data uji: ";
        cin >> filename;
        vector<uint8_t> pixelData;
        int width, height;

        if (!readBMP(filename, pixelData, width, height))
        {
            return -1;
        }
        double redPercentage, otherPercentage;
        int totalPixelCount = width * height;
        calculateColorPercentages(pixelData, totalPixelCount, redPercentage,
otherPercentage);
```

```
        cout << "Nama file: " << filename << endl;
        cout << "Persentase warna merah: " << redPercentage << "%" << endl;
        cout << "Persentase warna selain merah: " << otherPercentage << "%" <<
endl;
        O = redPercentage * w[0] + otherPercentage * w[1] + x[2][0] * w[2];
        cout << O << endl;
        if (O > 0.0)
            cout << "Output = Bahaya" << endl;
        else
            cout << "Output = Aman" << endl;
    }
    return 0;
}
```

Program alat Keselurahan yang sudah sesuai di Arduino ide

1. **Program utama alat**

```
. #include <Wire.h>
. #include <Adafruit_AMG88xx.h>
. #include <SD.h>
. #include "Arduino.h"
. #include <WiFiS3.h>
. #include <ArduinoJson.h>
. #include <Adafruit_GFX.h>
. #include <Adafruit_SSD1306.h>
.
0.// Definisi OLED
1.#define SCREEN_WIDTH 128
2.#define SCREEN_HEIGHT 64
3.#define OLED_RESET    -1
4.#define DEBUG_LOOP 1
5.Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
  OLED_RESET);
6.
7.// WiFi settings
8.const char* ssid = "OPPO Reno10";
9.const char* password = "123456789";
0.
```

```
1.// Definisi pin untuk kartu SD
2.const int SD_CS = SS;
3.const int SD_MOSI = MOSI;
4.const int SD_MISO = MISO;
5.const int SD_SCK = SCK;
6.
7.// Pin untuk tombol kalibrasi
8.const int CAL_UP_PIN = 2;
9.const int CAL_DOWN_PIN = 3;
0.const int CAL_SAVE_PIN = 4;
1.
2.// Variabel kalibrasi
3.float tempOffset = 0.0;
4.const float CAL_STEP = 0.5;
5.
6.// Kompensasi jarak dan ambang batas suhu
7.const float DISTANCE_FACTOR = 1.2;
8.const float TEMP_THRESHOLD = 30.0;
9.const float TEMP_HYSTERESIS = 2.0;
0.
1.// Konstanta untuk klasifikasi 50/50
2.const float DANGER_THRESHOLD = 50.0;  // 50% threshold for
  danger classification
3.
4.// Struktur data training
5.struct TrainingData {
6.    float redPercentage;   // Persentase area panas
7.    float otherPercentage; // Persentase area lain
8.    int label;             // 0 = normal, 1 = bahaya
9.};
0.
1.// Konstanta untuk manajemen data training
2.const int MAX_TRAINING_SAMPLES = 50;
3.TrainingData trainingDataset[MAX_TRAINING_SAMPLES];
4.int currentTrainingDataCount = 0;
5.bool isTrainingDataLoaded = false;
6.
7.// Variabel bobot global untuk Single Perceptron
8.float w[3];
```

```
9.
0.// Deklarasi fungsi eksternal untuk fitur Telegram
1.extern void setupTelegram();
2.extern void checkAndSendNotification(float avgTemp, double
  redPercentage, bool isDanger);
3.extern void setupTelegramCommands();
4.extern void handleTelegramMessages(float pixels[], float
  avgTemp, double redPercentage, bool isDanger);
5.extern bool monitoringActive;
6.
7.// HTML untuk web interface (dalam PROGMEM untuk menghemat
  memori)
8.const char WEBPAGE[] PROGMEM = R"rawliteral(
9.
0.        // Get DOM elements
1.        const canvas =
  document.getElementById('thermalCanvas');
2.        const ctx = canvas.getContext('2d');
3.        const pixelGrid =
  document.getElementById('pixelGrid');
4.        const connectionStatusEl =
  document.getElementById('connectionStatus');
5.        const percentageBar =
  document.getElementById('percentageBar');
6.        const percentageText =
  document.getElementById('percentageText');
7.        const ipAddressEl =
  document.getElementById('ipAddress');
8.
9.        // Set up canvas
0.        canvas.width = 256; // Larger internal resolution for
  better quality
1.        canvas.height = 256;
2.
3.        // Initialize pixel grid (alternative rendering
  method)
4.        function initPixelGrid() {
5.            pixelGrid.innerHTML = '';
6.            for (let i = 0; i < 64; i++) {
```

```
                const pixel = document.createElement('div');
                pixel.className = 'pixel';
                pixel.id = `pixel-${i}`;
                pixelGrid.appendChild(pixel);
            }
        }

        // Initialize both rendering methods
        initPixelGrid();

        // Keep track of connection state
        let isConnected = true;
        let retryCount = 0;
             const maxRetries = 3;

             // Set the IP address in the header
             function updateIPAddress() {
                 // Get the current URL's hostname and port
                 const host = window.location.hostname;
                 const port = window.location.port;

                 // Update the IP display
                 if (port) {
                     ipAddressEl.textContent = `IP:
  ${host}:${port}`;
                 } else {
                     ipAddressEl.textContent = `IP: ${host}`;
                 }
             }

             // Call immediately to set IP
             updateIPAddress();

             function interpolateColor(value) {
                 const colors = [
                     {temp: 0, color: [0, 0, 255]},      //
  Blue (coolest)
                     {temp: 0.25, color: [0, 128, 255]}, //
  Light blue
```

```
                    {temp: 0.5, color: [0, 255, 0]},    //
  Green
                    {temp: 0.75, color: [255, 128, 0]}, //
  Orange
                    {temp: 1, color: [255, 0, 0]}       //
  Red (warmest)
                ];

                let startColor, endColor;
                let startTemp, endTemp;

                for (let i = 0; i < colors.length - 1; i++) {
                    if (value >= colors[i].temp && value <=
  colors[i + 1].temp) {
                        startColor = colors[i].color;
                        endColor = colors[i + 1].color;
                        startTemp = colors[i].temp;
                        endTemp = colors[i + 1].temp;
                        break;
                    }
                }

                if (!startColor) {
                    if (value <= 0) return
  `rgb(${colors[0].color.join(',')})`;
                    if (value >= 1) return
  `rgb(${colors[colors.length - 1].color.join(',')})`;
                }

                const ratio = (value - startTemp) / (endTemp
  - startTemp);
                const r = Math.round(startColor[0] +
  (endColor[0] - startColor[0]) * ratio);
                  const g = Math.round(startColor[1] +
  (endColor[1] - startColor[1]) * ratio);
                const b = Math.round(startColor[2] +
  (endColor[2] - startColor[2]) * ratio);

                return `rgb(${r},${g},${b})`;
```

```
52.             }
53.
54.             // Enhanced rendering with bilinear interpolation
55.             function updateThermalImage(pixels) {
56.                 if (!pixels || pixels.length !== 64) {
57.                     console.error('Invalid pixel data
  received');
58.                     return;
59.                 }
60.
61.                 const minTemp = Math.min(...pixels);
62.                 const maxTemp = Math.max(...pixels);
63.                 const range = maxTemp - minTemp;
64.
65.                 if (isNaN(minTemp) || isNaN(maxTemp) ||
  isNaN(range)) {
66.                     console.error('Invalid temperature
  range');
67.                     return;
68.                 }
69.
70.                 document.getElementById('maxTemp').textConten
  t = `${maxTemp.toFixed(1)}°C`;
71.
72.                 // Clear canvas
73.                 ctx.clearRect(0, 0, canvas.width,
  canvas.height);
74.
75.                 // Method 1: Draw to canvas with
  interpolation for smoother image
76.                 const pixelsPerUnit = canvas.width / 8;
77.
78.                 // Draw using bilinear interpolation for
  smoother image
79.                 for (let y = 0; y < canvas.height; y++) {
80.                     for (let x = 0; x < canvas.width; x++) {
81.                         // Map pixel coordinates to grid
  coordinates
82.                         const gx = x / pixelsPerUnit;
```

```
                        const gy = y / pixelsPerUnit;

                        // Get the four nearest grid points
                        const x0 = Math.floor(gx);
                        const y0 = Math.floor(gy);
                        const x1 = Math.min(x0 + 1, 7);
                        const y1 = Math.min(y0 + 1, 7);

                        // Calculate interpolation factors
                        const sx = gx - x0;
                        const sy = gy - y0;

                        // Get the four pixel values
                        const p00 = pixels[y0 * 8 + x0];
                        const p10 = pixels[y0 * 8 + x1];
                        const p01 = pixels[y1 * 8 + x0];
                        const p11 = pixels[y1 * 8 + x1];

                        // Bilinear interpolation formula
                        const p = (1 - sx) * (1 - sy) * p00 +
                                  sx * (1 - sy) * p10 +
                                  (1 - sx) * sy * p01 +
                                  sx * sy * p11;

                        // Normalize temperature for color
  mapping
                        const normalizedTemp = range === 0 ?
  0 : (p - minTemp) / range;

                        // Set pixel color
                        ctx.fillStyle =
  interpolateColor(normalizedTemp);
                        ctx.fillRect(x, y, 1, 1);
                    }
                }

                // Method 2: Update the grid (simpler but
  with visible grid cells)
                for (let i = 0; i < 64; i++) {
```

```
                    const pixelElement =
  document.getElementById(`pixel-${i}`);
                    if (pixelElement) {
                        const normalizedTemp = range === 0 ?
  0 : (pixels[i] - minTemp) / range;
                        pixelElement.style.backgroundColor =
  interpolateColor(normalizedTemp);
                    }
                }
            }

            function updateStats(avgTemp, hotPercentage,
  isDanger) {
                document.getElementById('avgTemp').textConten
  t = `${avgTemp.toFixed(1)}°C`;
                document.getElementById('hotPercentage').text
  Content = `${hotPercentage.toFixed(1)}%`;

                // Update percentage visualization
                percentageBar.style.width =
  `${hotPercentage}%`;
                percentageText.textContent =
  `${hotPercentage.toFixed(1)}%`;

                // Set color based on danger threshold (50%)
                const dangerThreshold = 50;
                if (hotPercentage >= dangerThreshold) {
                    percentageBar.style.backgroundColor =
  'rgba(255, 0, 0, 0.7)';
                } else if (hotPercentage >= dangerThreshold *
  0.75) {
                    percentageBar.style.backgroundColor =
  'rgba(255, 165, 0, 0.7)';
                } else if (hotPercentage >= dangerThreshold *
  0.5) {
                    percentageBar.style.backgroundColor =
  'rgba(255, 255, 0, 0.7)';
                } else {
```

```
                    percentageBar.style.backgroundColor =
  'rgba(0, 255, 0, 0.7)';
                }

                const statusIndicator =
  document.getElementById('statusIndicator');
                if (isDanger) {
                    statusIndicator.className = 'status-
  indicator danger';
                    statusIndicator.textContent = 'DANGER:
  High Temperature Detected!';
                } else {
                    statusIndicator.className = 'status-
  indicator safe';
                    statusIndicator.textContent = 'System
  Status: NORMAL';
                }
            }

            function updateConnectionStatus(connected) {
                isConnected = connected;
                connectionStatusEl.className = connected ?
  'connected' : 'disconnected';
                connectionStatusEl.textContent = `Connection
  Status: ${connected ? 'Connected' : 'Disconnected -
  Retrying...'}`;
            }

            async function pollThermalData() {
                try {
                    const response = await fetch('/thermal-
  data', {
                        method: 'GET',
                        cache: 'no-store',
                        headers: {
                            'Cache-Control': 'no-cache'
                        }
                    });

```

```
                    if (!response.ok) {
                        throw new Error(`HTTP error! status:
  ${response.status}`);
                    }

                    const data = await response.json();

                    if (!isConnected) {
                        updateConnectionStatus(true);
                        retryCount = 0;
                    }

                    if (data && data.pixels &&
  Array.isArray(data.pixels)) {
                        updateThermalImage(data.pixels);
                        updateStats(data.averageTemp,
  data.hotPercentage, data.isDanger);
                    } else {
                        console.error('Unexpected data
  format:', data);
                    }
                } catch (error) {
                    console.error('Error fetching thermal
  data:', error);

                    if (isConnected) {
                        updateConnectionStatus(false);
                    }

                    retryCount++;
                    if (retryCount > maxRetries) {
                        // Implement exponential backoff
                        setTimeout(pollThermalData, 5000); //
  Wait 5 seconds before retrying again
                        return;
                    }
                }

                // Schedule next poll
```

```
                setTimeout(pollThermalData, 500);
            }

            // Toggle between rendering methods (Optional feature)
            let useCanvas = true;
            function toggleRenderingMethod() {
                useCanvas = !useCanvas;
                if (useCanvas) {
                    canvas.style.display = 'block';
                    pixelGrid.style.display = 'none';
                } else {
                    canvas.style.display = 'none';
                    pixelGrid.style.display = 'grid';
                }
            }

            // Default to canvas rendering for better quality
            canvas.style.display = 'block';
            pixelGrid.style.display = 'none';

            // Start polling immediately
            pollThermalData();
        </script>
    </body>
    </html>
    )rawliteral";

    // Deklarasi fungsi
    String getFormattedTime();
    void logStatus(bool isDanger, float avgTemp, double redPercentage);
    void updateOLEDDisplay(float avgTemp, double redPercentage, bool isDanger);
    float d_rand(void);
    void trainPerceptron();
    // Inisialisasi komponen utama
    Adafruit_AMG88xx amg;
    WiFiServer server(80);
```

```
41.
42.     // Fungsi untuk memuat data training dari CSV
43.     bool loadTrainingDataFromCSV() {
44.         // Cek keberadaan file training
45.         if (!SD.exists("training.csv")) {
46.             Serial.println("File training.csv tidak
  ditemukan!");
47.             return false;
48.         }
49.
50.         // Buka file CSV
51.         File trainingFile = SD.open("training.csv",
  FILE_READ);
52.         if (!trainingFile) {
53.             Serial.println("Gagal membuka file
  training.csv");
54.             return false;
55.         }
56.
57.         // Reset counter data training
58.         currentTrainingDataCount = 0;
59.
60.         // Lewati baris header jika ada
61.
62.             // Parsing data
63.             int firstComma = line.indexOf(',');
64.             int secondComma = line.lastIndexOf(',');
65.
66.             if (firstComma != -1 && secondComma != -1 &&
  firstComma != secondComma) {
67.                 float redPercentage = line.substring(0,
  firstComma).toFloat();
68.                 float otherPercentage =
  line.substring(firstComma + 1, secondComma).toFloat();
69.                 int label = line.substring(secondComma +
  1).toInt();
70.
71.                 // Simpan ke dataset
```

```
72.                 trainingDataset[currentTrainingDataCount].red
  Percentage = redPercentage;
73.                 trainingDataset[currentTrainingDataCount].oth
  erPercentage = otherPercentage;
74.                 trainingDataset[currentTrainingDataCount].lab
  el = label;
75.
76.                 currentTrainingDataCount++;
77.             }
78.         }
79.
80.         trainingFile.close();
81.
82.         // Tampilkan informasi yang dimuat
83.         Serial.print("Berhasil memuat ");
84.         Serial.print(currentTrainingDataCount);
85.         Serial.println(" sampel training");
86.         return currentTrainingDataCount > 0;
87.     }
88.
89.     // Format timestamp untuk logging
90.     String getFormattedTime() {
91.         return String(millis()/1000);
92.     }
93.
94.     // Fungsi untuk logging
95.     void logStatus(bool isDanger, float avgTemp, double
  redPercentage) {
96.         String timestamp = getFormattedTime();
97.         String status = isDanger ? "Bahaya" : "Normal";
98.         String logEntry = timestamp + "," +
  String(redPercentage, 1) + "," + status;
99.
00.         if (isDanger) {
01.             File logFile = SD.open("alerts.txt", FILE_WRITE);
02.             if (logFile) {
03.                 logFile.println(logEntry);
04.                 logFile.close();
05.                 Serial.println("Alert logged: " + logEntry);
```

```
            }
        } else {
            File logFile = SD.open("normal.txt", FILE_WRITE);
            if (logFile) {
                logFile.println(logEntry);
                logFile.close();
                Serial.println("Normal status logged: " +
  logEntry);
            }
        }
    }

    // Update tampilan OLED
    void updateOLEDDisplay(float avgTemp, double
  redPercentage, bool isDanger) {
        display.clearDisplay();

        // Tampilkan header
        display.setTextSize(1);
            display.setCursor(0, 0);
        display.println("Thermal Monitor");
        display.drawLine(0, 10, display.width(), 10,
  SSD1306_WHITE);

        // Tampilkan persentase area panas
        display.setCursor(0, 26);
        display.print(redPercentage, 1);
        display.println("%");

        // Tampilkan status klasifikasi
        display.setCursor(0, 38);
        display.print("Status: ");

        // Tampilkan threshold yang digunakan
        display.setCursor(72, 38);
        display.print("(T:");
        display.print(DANGER_THRESHOLD, 0);

        // Gunakan font lebih besar untuk status
```

```
42.         display.setCursor(0, 48);
43.
44.         if (isDanger) {
45.             display.println("BAHAYA!");
46.         } else {
47.             display.println("NORMAL");
48.         }
49.
50.         display.display();
51.     }
52.
53.     // Fungsi untuk menghasilkan bilangan acak
54.     float d_rand(void) {
55.         return (((float)(rand() % 32767) / (32767.0 - 0.5) *
56.         // 1. Hitung statistik dasar
57.         float minTemp = 100.0;
58.         float maxTemp = -100.0;
59.         float sumTemp = 0.0;
60.
61.         for (int i = 0; i < totalPixels; i++) {
62.             // Sesuaikan suhu dengan faktor jarak dan offset
  kalibrasi
63.
64.             sumTemp += adjustedTemp;
65.             minTemp = min(minTemp, adjustedTemp);
66.             maxTemp = max(maxTemp, adjustedTemp);
67.         }
68.
69.         // Hitung suhu rata-rata
70.         float avgTemp = sumTemp / totalPixels;
71.
72.         // 2. Hitung deviasi standar
73.         float sumSquaredDiff = 0.0;
74.         for (int i = 0; i < totalPixels; i++) {
75.             float adjustedTemp = pixels[i] * DISTANCE_FACTOR
  + tempOffset;
76.             float diff = adjustedTemp - avgTemp;
77.             sumSquaredDiff += (diff * diff);
78.         }
```

```
79.         float stdDev = sqrt(sumSquaredDiff / totalPixels);
80.
81.         // 3. Tentukan threshold untuk area merah (panas)
82.         // Menggunakan rata-rata + faktor standar deviasi
83.         float redThreshold = avgTemp + (1.5 * stdDev);
84.
85.         // 4. Hitung pixel merah (panas)
86.         int redPixelCount = 0;
87.         for (int i = 0; i < totalPixels; i++) {
88.             float adjustedTemp = pixels[i] * DISTANCE_FACTOR
89.             // 5. Hitung presentase
90.         redPercentage = (double)redPixelCount / totalPixels *
  100.0;
91.         otherPercentage = 100.0 - redPercentage;
92.
93.         // 6. Batasi rentang presentase
94.         redPercentage = constrain(redPercentage, 0.0, 100.0);
95.         otherPercentage = constrain(otherPercentage, 0.0,
  100.0);
96.
97.         // 7. Debug output
98.         Serial.println("\n===== Analisis Persentase Thermal
  =====");
99.         Serial.print("Suhu Min: "); Serial.print(minTemp);
  Serial.println("°C");
00.         Serial.print("Suhu Rata-rata: ");
01.         Serial.print("Deviasi Standar: ");
  Serial.print(stdDev); Serial.println("°C");
02.         Serial.print("Threshold Merah: ");
  Serial.print(redThreshold); Serial.println("°C");
03.
04.         Serial.print("Pixel Merah: ");
  Serial.print(redPixelCount);
05.         Serial.print(" dari total ");
  Serial.print(totalPixels); Serial.println(" pixel");
06.
07.         Serial.print("Presentase Merah: ");
08.         Serial.print(redPercentage);
09.         Serial.println("%");
```

```
10.     }
11.
12.     // Hitung suhu rata-rata
13.     float calculateAverageTemp(float pixels[]) {
14.         float sum = 0;
15.         int validReadings = 0;
16.
17.         for (int sample = 0; sample < 3; sample++) {
18.             for (int i = 0; i < AMG88xx_PIXEL_ARRAY_SIZE;
  i++) {
19.
20.                 // Filter out unreasonable values
21.                 if (temp > -20 && temp < 80) {
22.                     sum += temp;
23.                     validReadings++;
24.                 }
25.             }
26.             delay(10);
27.         }
28.
29.         return validReadings > 0 ? (sum / validReadings) +
  tempOffset : 0;
30.     }
31.
32.     // Tampilkan suhu
33.     void displayTemperature(float temperature) {
34.         Serial.print("Suhu: ");
35.           Serial.println(" °C");
36.     }
37.
38.     // Tampilkan peringatan bahaya
39.     void displayDangerWarning(float temperature) {
40.         Serial.println("PERINGATAN: SUHU BERBAHAYA!");
41.         Serial.print("Suhu: ");
42.         Serial.print(temperature);
43.         Serial.println(" °C");
44.     }
45.         }
```

```
46.    oid sendThermalData(WiFiClient client, float pixels[],
  float avgTemp, double redPercentage, bool isDanger) {
47.        // Make sure the JsonDocument is large enough
48.
49.        JsonArray pixelArray =
  doc.createNestedArray("pixels");
50.            }
51.
52.        doc["averageTemp"] = avgTemp;
53.        doc["hotPercentage"] = redPercentage;
54.        doc["isDanger"] = isDanger;
55.        doc["maxTemp"] = maxTemp;
56.
57.    void handleWebRequest(WiFiClient client, float pixels[],
  float avgTemp, double redPercentage, bool isDanger) {
58.        String currentLine = "";
59.        String requestLine = "";
60.
61.
62.        while (client.connected() && (millis() - startTime <
  1000)) {
63.            if (client.available()) {
64.                char c = client.read();
65.
66.                if (c == '\n') {
67.                    if (currentLine.length() == 0) {
68.                        // End of HTTP headers
69.                        if (requestLine.indexOf("GET
  /thermal-data") >= 0) {
70.    / Proses training perceptron
71.    void trainPerceptron() {
72.        Serial.println("\n===== Proses Pelatihan Perceptron
  =====");
73.        display.clearDisplay();
74.        display.setCursor(0, 0);
75.        display.println("Proses Pelatihan");
76.        display.println("Perceptron");
77.        display.display();
78.
```

```
79.         // Siapkan data untuk training
80.         float x[3][MAX_TRAINING_SAMPLES];
81.         int T[MAX_TRAINING_SAMPLES];
82.
83.         // Konversi data training ke format yang dibutuhkan
84.         for (int j = 0; j < currentTrainingDataCount; j++) {
85.             x[0][j] = trainingDataset[j].redPercentage;
86.             x[1][j] = trainingDataset[j].otherPercentage;
87.             x[2][j] = 1.0;  // Bias
88.         }
89.         // Proses training
90.         for (int iter = 1; iter <= 6; iter++) {
91.             Serial.print("\nTraining Iteration - ");
92.             Serial.println(iter);
93.
94.             for (int j = 0; j < currentTrainingDataCount;
  j++) {

95.                 // Tentukan output biner
96.                 int out = (O > 0.0) ? 1 : 0;
97.
98.                 // Hitung error
99.                 int ERR = T[j] - out;
00.
01.                 // Update bobot jika ada error
02.                 if (ERR != 0) {
03.                     w[0] += LR * x[0][j] * ERR;
04.                     w[1] += LR * x[1][j] * ERR;

05.                     Serial.println("--- Memperbarui Bobot ---
  ");
06.                     Serial.print("New w[0]: ");
07.                     Serial.println(w[0]);
08.                     Serial.print("New w[1]: ");
09.                     Serial.println(w[1]);
10.                 }
11.             }
12.         }
13.
```

```
        // Simpan bobot akhir ke file
        File weightFile = SD.open("weights.txt", FILE_WRITE);
        if (weightFile) {
            for (int i = 0; i < 3; i++) {
                weightFile.print(w[i]);
                weightFile.print(",");
            }
            weightFile.println();
            weightFile.close();
            Serial.println("\nBobot akhir berhasil disimpan
  ke kartu SD");
        }

        Serial.println("\n===== Pelatihan Selesai =====");

        // Tampilkan pesan penyelesaian pada OLED
        display.clearDisplay();
        display.setCursor(0, 0);
        display.println("Pelatihan Selesai!");
        display.println("Sistem siap untuk");
        display.println("klasifikasi realtime");
        display.display();
        delay(2000);
    }

    // Setup utama
    void setup() {
        Serial.begin(115200);
        while (!Serial) {
           pinMode(CAL_UP_PIN, INPUT_PULLUP);
        pinMode(CAL_DOWN_PIN, INPUT_PULLUP);
        pinMode(CAL_SAVE_PIN, INPUT_PULLUP);

        Serial.println("\n===== Inisialisasi Kartu SD
  =====");
        Serial.print("Initializing SD card...");
        display.setCursor(0, 24);
        display.println("SD card init...");
        display.display();
```

```
51.
52.    // Muat kalibrasi sebelumnya
53.        loadCalibration();
54.
55.        Serial.println("\n===== Inisialisasi Sensor Termal
  =====");
56.        display.setCursor(0, 40);
57.        display.println("AMG8833 init...");
58.        display.display();
59.
60.        if (!amg.begin()) {
61.            Serial.println("Tidak dapat menemukan sensor
  AMG8833 yang valid!");
62.            display.println("AMG8833 failed!");
63.            display.display();
64.            while (1);
65.        }
66.        Serial.println("Sensor AMG8833 berhasil
  diinisialisasi!");
67.        display.println("AMG8833 OK!");
68.        display.display();
69.
70.        // Koneksi WiFi
71.        Serial.println("\n===== Koneksi Wi-Fi =====");
72.        display.setCursor(0, 56);
73.        display.println("WiFi connecting...");
74.        display.display();
75.
76.        WiFi.begin(ssid, password);
77.
78.        int attempts = 0;
79.        while (WiFi.status() != WL_CONNECTED && attempts <
  20) {
80.            delay(500);
81.            Serial.print(".");
82.            attempts++;
83.        }
84.
85.        if (WiFi.status() != WL_CONNECTED) {
```

```
            Serial.println("\nGagal terhubung ke WiFi!");
            display.clearDisplay();
            display.setCursor(0, 0);
            display.println("WiFi failed!");
            display.display();
        } else {
            Serial.println("\nBerhasil terhubung ke WiFi!");
            Serial.print("IP address: ");
            Serial.println(WiFi.localIP());
            server.begin();

            display.clearDisplay();
            display.setCursor(0, 0);
            display.println("WiFi connected!");
            display.println("IP: ");
            // Inisialisasi perintah Telegram (untuk
  notifikasi manual)
            setupTelegramCommands();
        }

        // Proses setup training
        bool trainingReady = loadTrainingDataFromCSV();

        if (trainingReady) {
            // Jika data training tersedia, lakukan training
            trainPerceptron();
        } else {

            Serial.println("\nSiapkan file training.csv di
  kartu SD");
            Serial.println("Format:
  RedPercentage,OtherPercentage,Label");
        }
    }

    // Loop utama
    void loop() {
     // Debug: indikator loop berjalan
        static unsigned long lastLoopDebug = 0;
```

```
22.         if (DEBUG_LOOP && millis() - lastLoopDebug > 5000) {
23.             Serial.println(">>> LOOP IS RUNNING <<<");
24.             lastLoopDebug = millis();
25.         }
26.         // Kalibrasi suhu
27.         if (digitalRead(CAL_UP_PIN) == LOW) {
28.             tempOffset += CAL_STEP;
29.             Serial.println("\n===== Penyesuaian Kalibrasi
  =====");
30.             Serial.print("Offset kalibrasi ditingkatkan
  menjadi: ");
31.              delay(250);
32.         }
33.
34.         if (digitalRead(CAL_DOWN_PIN) == LOW) {
35.             tempOffset -= CAL_STEP;
36.             Serial.println("\n===== Penyesuaian Kalibrasi
  =====");
37.             Serial.print("Offset kalibrasi menurun menjadi:
  ");
38.             Serial.print(tempOffset);
39.             Serial.println("°C");
40.
41.             display.clearDisplay();
42.             display.setTextSize(1);
43.             display.setCursor(0, 0);
44.             display.println("Kalibrasi");
45.             display.print("Offset: ");
46.             display.print(tempOffset, 1);
47.             display.println(" C");
48.             display.display();
49.
50.             delay(250);
51.         }
52.         // Baca piksel sensor termal
53.         float pixels[AMG88xx_PIXEL_ARRAY_SIZE];
54.         amg.readPixels(pixels);
55.
56.         // Hitung suhu rata-rata
```

```
        float avgTemp = calculateAverageTemp(pixels);
        Serial.println("\n===== Pembacaan Sensor Termal
  =====");
        Serial.print("Suhu Rata-Rata: ");
        Serial.print(avgTemp);
        Serial.println(" °C");

        // Hitung persentase termal
        double redPercentage,

            // Periksa pesan Telegram masuk dan tanggapi
  perintah
            handleTelegramMessages(pixels, avgTemp,
  redPercentage, isDanger);

            // Debug: Konfirmasi setelah
  handleTelegramMessages
            if (DEBUG_LOOP) {
                Serial.println("=== TELEGRAM HANDLER FINISHED

            // Kirim notifikasi otomatis hanya jika
  pemantauan aktif
            if (monitoringActive) {
                if (DEBUG_LOOP) {
                    Serial.println("=== CHECKING
  NOTIFICATIONS ===");
                }
                checkAndSendNotification(avgTemp,
  redPercentage, isDanger);
            }
        } else {
            if (DEBUG_LOOP) {
                Serial.println("!!! WiFi DISCONNECTED !!!");
            }
        }

        delay(100);
    }
```

2. Program untuk konfigurasi notifikasi ke Telegram

```
#include <WiFiS3.h>
#include <ArduinoJson.h>

// Pengaturan Telegram Bot
const char* BOT_TOKEN = "7926501747:AAGntWz0stgT1bBuBUz-MKA2hGZCmZOuvk4";  // Token Anda
const char* CHAT_ID = "1589405327";     // Chat ID Anda

// Variabel untuk kontrol monitoring
bool monitoringActive = true;
long lastUpdateID = 0;
bool botInitialized = false;

// Variabel untuk melacak pengiriman notifikasi
unsigned long lastNotificationTime = 0;
unsigned long lastCheckTime = 0;
const unsigned long NOTIFICATION_INTERVAL = 36000;  // 1 jam dalam milidetik
const unsigned long CHECK_INTERVAL = 3000;  // Cek pesan setiap 3 detik

// Variabel untuk data dari program utama
extern float tempOffset;
extern const float DANGER_THRESHOLD;

// Helper function untuk encode URL
String urlEncode(String str) {
  String encoded = "";
  char c;
  char code0;
  char code1;

  for (int i = 0; i < str.length(); i++) {
    c = str.charAt(i);
    if (c == ' ') {
      encoded += '+';
    } else if (isalnum(c) || c == '-' || c == '_' || c == '.' || c == '~') {
      encoded += c;
```

```
22.         } else {
23.           encoded += '%';
24.           code1 = (c & 0xf) + '0';
25.           if ((c & 0xf) > 9) {
26.             code1 = (c & 0xf) - 10 + 'A';
27.           }
28.           c = (c >> 4) & 0xf;
29.           code0 = c + '0';
30.           if (c > 9) {
31.             code0 = c - 10 + 'A';
32.           }
33.           encoded += code0;
34.           encoded += code1;
35.         }
36.         yield();
37.       }
38.       return encoded;
39.     }
40.
41.     // Fungsi untuk mengirim pesan Telegram menggunakan
  HTTPS
42.     bool sendTelegramMessage(String message) {
43.       Serial.println("\n[Telegram] === SENDING MESSAGE START
  ===");
44.       Serial.println("[Telegram] Raw message: " + message);
45.       // Baca response
46.       String response = "";
47.       unsigned long timeout = millis();
48.       boolean headersDone = false;
49.
50.       while (client.connected() && millis() - timeout <
  10000) {
51.         if (client.available()) {
52.           String line = client.readStringUntil('\n');
53.           line.trim();
54.
55.           if (!headersDone) {
56.             if (line.startsWith("HTTP/1.1")) {
57.               Serial.println("[Telegram] " + line);
```

```
            }
            if (line.length() == 0) {
              headersDone = true;
            }
          } else {
            response += line;
          }
        }
      }
      client.stop();

      Serial.println("[Telegram] Response: " + response);

      // Cek response
      if (response.indexOf("\"ok\":true") > 0) {
        Serial.println("[Telegram] === SENDING MESSAGE
  SUCCESS ===");
        return true;
      } else {
        Serial.println("[Telegram] === SENDING MESSAGE
  FAILED ===");
        return false;
      }
    }
    // Cek pesan baru dari Telegram dengan HTTPS
    void checkTelegramMessages() {
      if (millis() - lastCheckTime < CHECK_INTERVAL) {
        return;
      }
      lastCheckTime = millis();

      Serial.println("\n[Telegram] === CHECKING MESSAGES
  START ===");
      Serial.println("[Telegram] Last Update ID: " +
  String(lastUpdateID));

      WiFiSSLClient client;  // Gunakan SSL client untuk
  HTTPS
```

```
92.
93.       Serial.print("[Telegram] Connecting to HTTPS port
  443... ");
94.       if (!client.connect("api.telegram.org", 443)) {
95.         Serial.println("FAILED");
96.         return;
97.       }
98.       Serial.println("SUCCESS");
99.
00.       String url = "/bot" + String(BOT_TOKEN) +
  "/getUpdates?offset=" + String(lastUpdateID + 1) +
  "&limit=1&timeout=0";
01.       // Debug URL (tanpa token penuh)
02.       String debugUrl = url;
03.       debugUrl.replace(BOT_TOKEN, "XXXXX");
04.       Serial.println("[Telegram] URL: " + debugUrl);
05.
06.       // Kirim request HTTPS
07.       client.println("GET " + url + " HTTP/1.1");
08.       client.println("Host: api.telegram.org");
09.       client.println("Connection: close");
10.       client.println();
11.
12.       // Baca response
13.       String response = "";
14.       unsigned long timeout = millis();
15.       boolean headersDone = false;
16.
17.       while (client.connected() && millis() - timeout <
  10000) {
18.         if (client.available()) {
19.           String line = client.readStringUntil('\n');
20.           line.trim();
21.
22.     atusMessage += "⏱ Uptime: " + getFormattedDateTime() +
  "\n";
23.         statusMessage += "🔧 Offset Kalibrasi: " +
  String(tempOffset, 1) + "°C\n";
```

```
24.         statusMessage += "⚠ Ambang Bahaya: " +
  String(DANGER_THRESHOLD, 0) + "%";
25.
26.         sendTelegramMessage(statusMessage);
27.       }
28.       else if (command == "/monitor_on") {
29.         monitoringActive = true;
30.         sendTelegramMessage("✅ Monitoring telah
  DIAKTIFKAN");
31.       }
32.       else if (command == "/monitor_off") {
33.
34.       char buffer[50];
35.       sprintf(buffer, "%lu hari, %02lu:%02lu:%02lu",
36.               days, hours % 24, minutes % 60, seconds % 60);
37.
38.       return String(buffer);
39.     }
40.
41.     // Setup Telegram
42.     void setupTelegram() {
43.       Serial.println("\n===== Inisialisasi Notifikasi
  Telegram =====");
44.       Serial.println("[Telegram] Sending initialization
  message...");
45.
46.       // Tunggu sebentar untuk memastikan WiFi sudah stabil
47.       delay(1000);
48.
49.       bool result = sendTelegramMessage("🚀 Sistem
  Pemantauan Termal telah dimulai!\n\nSistem siap menerima
  perintah.");
50.
51.       if (result) {
52.         Serial.println("[Telegram] Initialization
  successful!");
53.       } else {
54.         Serial.println("[Telegram] Initialization failed!
  Will retry...");
```

```
55.         // Retry sekali lagi setelah 5 detik
56.         delay(5000);
57.         result = sendTelegramMessage("🚀 Sistem Pemantauan
  Termal telah dimulai!\n\nSistem siap menerima perintah.");
58.
59.         if (result) {
60.           Serial.println("[Telegram] Initialization
  successful on retry!");
61.         } else {
62.           Serial.println("[Telegram] Initialization failed
  after retry!");
63.         }
64.       }
65.     }
66.
67.     void setupTelegramCommands() {
68.       Serial.println("===== Inisialisasi Perintah Telegram
  =====");
69.
70.       if (!botInitialized) {
71.         String initMessage = "🌡️ SISTEM PEMANTAUAN TERMAL
  SIAP!\n\n";
72.         initMessage += "ℹ️ Ketik /start untuk melihat menu
  utama\n\n";
73.         initMessage += "📊 Status monitoring: " +
  String(monitoringActive ? "AKTIF ✅" : "NONAKTIF ❌");
74.
75.         Serial.println("[Telegram] Sending startup
  message...");
76.         bool result = sendTelegramMessage(initMessage);
77.
78.         if (result) {
79.           botInitialized = true;
80.           Serial.println("[Telegram] Startup message
  sent!");
81.         } else {
82.           Serial.println("[Telegram] Failed to send startup
  message!");
83.         }
```

```
84.       }
85.     }
86.
87.     void handleTelegramMessages(float pixels[], float
  avgTemp, double redPercentage, bool isDanger) {
88.       checkTelegramMessages();
89.
90.       // Kirim data thermal jika diminta
91.       static bool thermalRequested = false;
92.
93.       if (lastUpdateID > 0 && !thermalRequested) {
94.         // Cek apakah perintah /thermal diminta
95.         // Ini akan ditangani oleh checkTelegramMessages()
96.       }
97.     }
98.
99.     void sendHourlyNotification(float avgTemp, double
  redPercentage, bool isDanger) {
000.      String message = "---STATUS SISTEM TERMAL---\n\n";
001.      message += "⏰ Waktu: " + getFormattedDateTime() +
  "\n\n";
002.      message += "🌡️ Suhu Rata-rata: " + String(avgTemp, 1) +
  "°C\n";
003.      message += "🔥 Persentase Panas: " +
  String(redPercentage, 1) + "%\n";
004.      //message += "⚠️ Ambang Bahaya: " +
  String(DANGER_THRESHOLD, 0) + "%\n\n";
005.
006.      if (isDanger) {
007.        message += "🚨 STATUS: BAHAYA!\n";
008.        message += "⚠️ Area panas telah melebihi ambang
  batas yang ditentukan.";
009.      } else {
010.        message += "✅ STATUS: NORMAL\n";
011.        message += "✔️ Semua parameter dalam batas normal.";
012.      }
013.     //message += "\n\n🔧 Offset Kalibrasi: " +
  String(tempOffset, 1) + "°C";
014.
```

```
015.      sendTelegramMessage(message);
016.    }
017.
018.    void checkAndSendNotification(float avgTemp, double
  redPercentage, bool isDanger) {
019.      unsigned long currentTime = millis();
020.
021.      // Kirim notifikasi per jam
022.      if (currentTime - lastNotificationTime >=
  NOTIFICATION_INTERVAL) {
023.        sendHourlyNotification(avgTemp, redPercentage,
  isDanger);
024.        lastNotificationTime = currentTime;
025.      }
026.
027.      // Kirim alert segera jika masuk status bahaya
028.      static bool lastDangerState = false;
029.      if (isDanger && !lastDangerState) {
030.        String alertMessage = "🚨⚠️ PERINGATAN SEGERA:
  STATUS BAHAYA TERDETEKSI! ⚠️🚨\n\n";
031.        alertMessage += "🌡️ Suhu Rata-rata: " +
  String(avgTemp, 1) + "°C\n";
032.        alertMessage += "🔥 Area Panas: " +
  String(redPercentage, 1) + "%\n";
033.        alertMessage += "📊 Melebihi ambang batas " +
  String(DANGER_THRESHOLD, 0) + "%\n";
034.        alertMessage += "⏰ Waktu: " +
  getFormattedDateTime() + "\n\n";
035.        alertMessage += "⚡ TINDAKAN DIPERLUKAN!";
036.
037.        sendTelegramMessage(alertMessage);
038.      }
039.
040.      lastDangerState = isDanger;
041.    }
042.
043.    // Fungsi untuk mengirim data thermal saat diminta
044.    void sendThermalData(float pixels[], float avgTemp,
  double redPercentage, bool isDanger) {
```

```
045.      String message = "🌡️ DATA THERMAL TERKINI\n\n";
046.      message += "⏰ Waktu: " + getFormattedDateTime() +
  "\n\n";
047.      message += "📊 Statistik:\n";
048.      message += "• Suhu Rata-rata: " + String(avgTemp, 1) +
  "°C\n";
049.      message += "• Area Panas: " + String(redPercentage, 1)
  + "%\n";
050.      message += "• Ambang Bahaya: " +
  String(DANGER_THRESHOLD, 0) + "%\n\n";
051.
052.      if (isDanger) {
053.        message += "🚨 Status: BAHAYA\n";
054.      } else {
055.        message += "✅ Status: NORMAL\n";
056.      }
057.
058.      message += "\n🔧 Offset: " + String(tempOffset, 1) +
  "°C";
059.
060.      sendTelegramMessage(message);
061.    }
062.    // Reset function untuk memulai dari awal jika
  diperlukan
063.    void resetTelegramBot() {
064.      lastUpdateID = 0;
065.      botInitialized = false;
066.      Serial.println("[Telegram] Bot state reset!");
067.    }
```