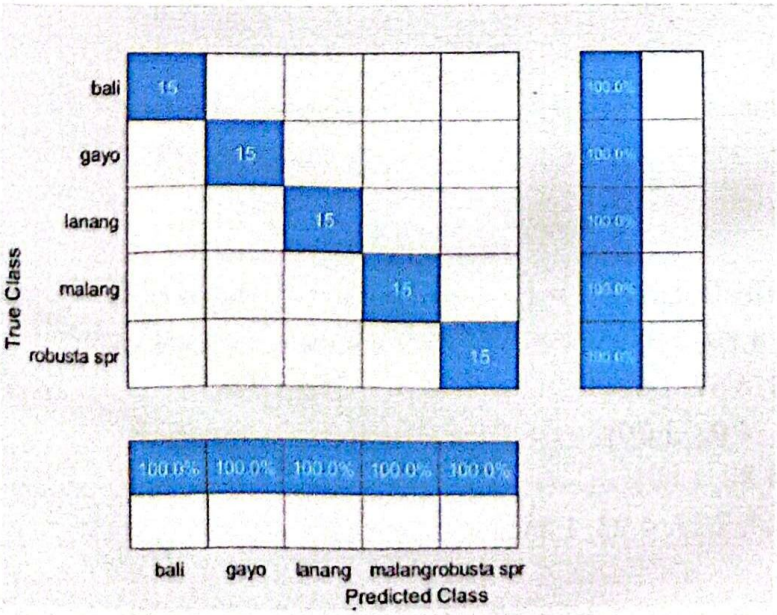
LAMPIRAN

LAMPIRAN

Lampiran 1

Voltage MQ7	Voltage MQ135	Voltage MQ4
0.35	0.59	2.03
0.35	0.59	2.03
0.35	0.59	2.03
0.34	0.59	2.01
0.34	0.59	2
0.34	0.59	1.99
0.34	0.61	1.99
0.34	0.61	1.98
0.34	0.61	1.97
0.34	0.62	1.96
0.34	0.62	1.96
0.33	0.61	1.95
0.33	0.61	1.94
0.33	0.61	1.93
0.33	0.61	1.92
0.33	0.61	1.92
0.33	0.62	1.91
0.33	0.62	1.91
0.31	0.62	1.9
0.32	0.62	1.89
0.31	0.62	1.89
0.31	0.62	1.88
0.32	0.61	1.88
0.31	0.62	1.88
0.31	0.62	1.87
0.31	0.61	1.87



Lampiran 2

Coding Arduino Mega 2560

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin sensor gas
#define MQ7_PIN A0
#define MQ135_PIN A1
#define MQ4_PIN A2

// Inisialisasi LCD 20x4
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Konstanta untuk kalibrasi sensor
const float RL_VALUE = 5.0;   // Tahanan beban
const float R0_MQ7 = 10.0;    // Nilai R0 untuk MQ7
const float R0_MQ135 = 10.0;  // Nilai R0 untuk MQ135
const float R0_MQ4 = 10.0;    // Nilai R0 untuk MQ4

// Struktur data untuk sampel kopi
struct CoffeeSample {
 String name;
 float mq7_value;
 float mq135_value;
 float mq4_value;
};

CoffeeSample coffeeDatabase[] = {
 {"Gayo", 10.38, 4.89, 2.07},
 {"Malang", 7.29, 4.67, 1.62},
 {"Lanang", 9.31, 4.92, 1.70},
 {"Bali", 10.07, 4.89, 1.69},
 {"Robusta Super", 7.27, 4.92, 1.53}
};
```

```
void setup() {
 // Inisialisasi komunikasi serial
 Serial.begin(9600);

 // Inisialisasi LCD
 lcd.init();
 lcd.backlight();
 lcd.setCursor(0, 0);
 lcd.print("TEKNIK ELEKTRO UNTAG");
 lcd.setCursor(4, 1);
 lcd.print("KLASIFIKASI");
 lcd.setCursor(8, 2);
 lcd.print("AROMA");
 lcd.setCursor(5, 3);
 lcd.print("JENIS KOPI");
 delay(3000);
 lcd.clear();
 lcd.setCursor(0, 0); // Baris pertama, kolom pertama
 lcd.print("Tunggu Sebentar");
 lcd.setCursor(0, 1); // Baris kedua, kolom pertama
 lcd.print("Klasifikasi");
 lcd.setCursor(0, 2); // Baris ketiga, kolom pertama
 lcd.print("Dalam Proses...");
 delay(5000);
 lcd.clear();

 // Kalibrasi sensor
 calibrateSensors();
}

void loop() {
 // Baca nilai sensor
 float mq7_reading = readMQ7Sensor();
 float mq135_reading = readMQ135Sensor();
 float mq4_reading = readMQ4Sensor();


 float voltagemq7 = analogRead(MQ7_PIN) * (5.0 / 1023.0);
```

```
  float voltagemq135 = analogRead(MQ135_PIN) * (5.0 / 1023.0);
  float voltagemq4 = analogRead(MQ4_PIN) * (5.0 / 1023.0);


  // Menampilkan nilai di Serial Monitor
  Serial.print("DATA,");
  Serial.print(mq7_reading); Serial.print(",");
  Serial.print(mq135_reading); Serial.print(",");
  Serial.print(mq4_reading); Serial.print(",");
  Serial.print(voltagemq7); Serial.print(",");
  Serial.print(voltagemq135); Serial.print(",");
  Serial.println(voltagemq4);

  // Lakukan klasifikasi LDA
  String coffeeName = performLDAClassification(mq7_reading,
mq135_reading, mq4_reading);

  // Tampilkan hasil di LCD
  displayResultOnLCD(coffeeName);

  delay(5000); // Tunggu 5 detik sebelum pengukuran berikutnya
 }

float readMQ7Sensor() {
  int sensorValue = analogRead(MQ7_PIN);
  float voltage = sensorValue * (5.0 / 1023.0);
  float rs = ((5.0 - voltage) / voltage) * RL_VALUE;
  float ratio = rs / R0_MQ7;
  return ratio;
 }

float readMQ135Sensor() {
  int sensorValue = analogRead(MQ135_PIN);
  float voltage = sensorValue * (5.0 / 1023.0);
  float rs = ((5.0 - voltage) / voltage) * RL_VALUE;
  float ratio = rs / R0_MQ135;
  return ratio;
 }
```

```
float readMQ4Sensor() {
 int sensorValue = analogRead(MQ4_PIN);
 float voltage = sensorValue * (5.0 / 1023.0);
 float rs = ((5.0 - voltage) / voltage) * RL_VALUE;
 float ratio = rs / R0_MQ4;
 return ratio;
}

void calibrateSensors() {
 // Proses kalibrasi sensor
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Kalibrasi Sensor...");

 // Implementasi kalibrasi sensor
 // Anda perlu menyesuaikan dengan prosedur kalibrasi aktual
 delay(2000);
}

String performLDAClassification(float mq7, float mq135, float mq4) {
 // Implementasi sederhana algoritma LDA
 float minDistance = 1000.0;
 String predictedCoffee = "Tidak Dikenal";

 for (int i = 0; i < 5; i++) {
  float distance = calculateEuclideanDistance(
   mq7, mq135, mq4,
   coffeeDatabase[i].mq7_value,
   coffeeDatabase[i].mq135_value,
   coffeeDatabase[i].mq4_value
  );

  if (distance < minDistance) {
   minDistance = distance;
   predictedCoffee = coffeeDatabase[i].name;
  }
 }
```

```
  return predictedCoffee;
 }

 float calculateEuclideanDistance(float x1, float y1, float z1, float x2, float y2,
float z2) {
  return sqrt(
   pow(x1 - x2, 2) +
   pow(y1 - y2, 2) +
   pow(z1 - z2, 2)
  );
 }

 void displayResultOnLCD(String coffeeName) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hasil Klasifikasi:");
  lcd.setCursor(0, 1);
  lcd.print(coffeeName);
 }
```

Coding Aplikasi Matlab

```
% Program Klasifikasi 5 Jenis Kopi menggunakan 3 Sensor Gas dan LDA
clear all;
close all;
clc;

% Load data training dari Excel
% Format Excel training:
% Kolom 1-3: Data sensor (MQ7, MQ135, MQ4)
% Kolom 4: Label kelas (1=Gayo, 2=Bali, 3=Malang, 4=Robusta Super,
5=Lanang)
[filename, pathname] = uigetfile('*.xlsx', 'Pilih file Excel data
training');
if filename == 0
    disp('Tidak ada file yang dipilih');
    return;
end

try
    training_data = readtable(fullfile(pathname, filename));
    % Konversi table ke array
    X = table2array(training_data(:,1:3));  % Data sensor
    y = table2array(training_data(:,4));    % Label kelas
catch
    error('Error membaca file Excel. Pastikan format sesuai!');
end

% Load data testing dari Excel
% Format sama dengan data training
[filename, pathname] = uigetfile('*.xlsx', 'Pilih file Excel data
testing');
if filename == 0
    disp('Tidak ada file yang dipilih');
    return;
end

try
    testing_data = readtable(fullfile(pathname, filename));
    % Konversi table ke array
    X_test = table2array(testing_data(:,1:3));  % Data sensor
    y_test = table2array(testing_data(:,4));    % Label kelas
sebenarnya
catch
```

```
    error('Error membaca file Excel. Pastikan format sesuai!');
end

% Lakukan LDA pada data training
[W, transformed_training] = perform_lda(X, y);

% Transform data testing
transformed_testing = X_test * W;

% Visualisasi hasil training dan testing
figure;
% Plot data training
scatter(transformed_training(y==1,1), transformed_training(y==1,2),
'r', 'filled');
hold on;
scatter(transformed_training(y==2,1), transformed_training(y==2,2),
'b', 'filled');
scatter(transformed_training(y==3,1), transformed_training(y==3,2),
'g', 'filled');
scatter(transformed_training(y==4,1), transformed_testing(y==4,2),
'y', 'filled');
scatter(transformed_training(y==5,1), transformed_training(y==5,2),
'k', 'filled');

% Plot data testing dengan marker berbeda
scatter(transformed_testing(y_test==1,1),
transformed_testing(y_test==1,2), 'rd');
scatter(transformed_testing(y_test==2,1),
transformed_testing(y_test==2,2), 'bd');
scatter(transformed_testing(y_test==3,1),
transformed_testing(y_test==3,2), 'gd');
scatter(transformed_training(y_test==4,1),
transformed_testing(y_test==4,2), 'yd');
scatter(transformed_training(y_test==5,1),
transformed_training(y_test==5,2), 'kd');

legend('Training Gayo', 'Training Bali', 'Training Malang',
'Training Robusta Super', 'Training Lanang',...
      'Testing Gayo', 'Testing Bali', 'Testing Malang', 'Training
Robusta Super', 'Training Lanang');
xlabel('First Discriminant');
ylabel('Second Discriminant');
title('LDA Classification of Coffee Types');
grid on;

% Evaluasi akurasi pada data testing
num_correct = 0;
```

```
predictions = zeros(size(y_test));

for i = 1:length(y_test)
    predictions(i) = classify_sample(X_test(i,:), W, X, y);
    if predictions(i) == y_test(i)
        num_correct = num_correct + 1;
    end
end

accuracy = num_correct / length(y_test) * 100;

% Tampilkan confusion matrix
conf_matrix = confusionmat(y_test, predictions);
figure;
confusionchart(conf_matrix, {'Gayo', 'Bali', 'Malang', 'Robusta
Super', 'Lanang'});
title('Confusion Matrix');

% Tampilkan hasil evaluasi
fprintf('\nHasil Evaluasi:\n');
fprintf('Jumlah data testing: %d\n', length(y_test));
fprintf('Jumlah prediksi benar: %d\n', num_correct);
fprintf('Akurasi: %.2f%%\n', accuracy);

% Simpan hasil prediksi ke Excel
results_table = table(X_test(:,1), X_test(:,2), X_test(:,3), y_test,
predictions, ...
    'VariableNames', {'Sensor1', 'Sensor2', 'Sensor3',
'Actual_Class', 'Predicted_Class'});
writetable(results_table, 'hasil_klasifikasi.xlsx');
fprintf('\nHasil klasifikasi telah disimpan dalam file
"hasil_klasifikasi.xlsx"\n');

% ===== Definisi Fungsi =====
function [W, transformed_data] = perform_lda(X, y)
    % Hitung mean total
    mean_total = mean(X);

    % Inisialisasi matriks within-class scatter dan between-class
scatter
    [n, d] = size(X);
    Sw = zeros(d);
    Sb = zeros(d);

    % Hitung matriks scatter untuk setiap kelas
    classes = unique(y);
    class_means = zeros(length(classes), d);
```

```
    for i = 1:length(classes)
        class_data = X(y == classes(i), :);
        class_mean = mean(class_data);
        class_means(i,:) = class_mean;

        % Within-class scatter
        centered_data = class_data - class_mean;
        Sw = Sw + (centered_data' * centered_data);

        % Between-class scatter
        class_diff = (class_mean - mean_total)';
        Sb = Sb + n * (class_diff * class_diff');
    end

    % Hitung eigenvector dan eigenvalue
    [W, D] = eig(inv(Sw) * Sb);

    % Sort eigenvalues in descending order
    [~, ind] = sort(diag(D), 'descend');
    W = W(:, ind);

    % Ambil 2 komponen pertama untuk visualisasi
    W = W(:, 1:2);

    % Transform data
    transformed_data = X * W;
end

function predicted_class = classify_sample(sample, W, X, y)
    % Transform sample
    transformed_sample = sample * W;

    % Transform training data
    transformed_training = X * W;

    % Hitung jarak ke semua titik training
    distances = pdist2(transformed_sample, transformed_training);

    % Ambil k tetangga terdekat (k=5)
    k = 5;
    [~, indices] = sort(distances);
    nearest_neighbors = indices(1:k);
    % Voting mayoritas
    neighbor_classes = y(nearest_neighbors);
    predicted_class = mode(neighbor_classes);
end
```