Lampiran 2 Source Code

Arduino Uno (sensor warna)

```
const int s0Pin = 2;
const int s1Pin = 3;
const int s2Pin = 4;
const int s3Pin = 5;
const int outPin = 6;
const int ledPin = 7; // Optional, if you are using an
external LED

void setup() {
  pinMode(s0Pin, OUTPUT);
  pinMode(s1Pin, OUTPUT);
  pinMode(s2Pin, OUTPUT);
  pinMode(s3Pin, OUTPUT);
  pinMode(outPin, INPUT);
  pinMode(ledPin, OUTPUT); // Optional

  // Set the frequency scaling to 20%
  digitalWrite(s0Pin, HIGH);
  digitalWrite(s1Pin, LOW);

  Serial.begin(9600); // Start serial communication at
9600 baud
}

void loop() {
  // Turn on the LED for better reading (optional)
  digitalWrite(ledPin, HIGH);

  // Read red color
  digitalWrite(s2Pin, LOW);
  digitalWrite(s3Pin, LOW);
  int redValue = pulseIn(outPin, LOW);

  // Read green color
  digitalWrite(s2Pin, HIGH);
  digitalWrite(s3Pin, HIGH);
  int greenValue = pulseIn(outPin, LOW);
  // Read blue color
  digitalWrite(s2Pin, LOW);
  digitalWrite(s3Pin, HIGH);
  int blueValue = pulseIn(outPin, LOW);

  // Turn off the LED (optional)
  digitalWrite(ledPin, LOW);

  // Print color values to the Serial Monitor
  Serial.print("Red: ");
  Serial.print(redValue);
  Serial.print("\tGreen: ");
  Serial.print(greenValue);
  Serial.print("\tBlue: ");
  Serial.println(blueValue);

  delay(1000); // Wait for 1 second before next reading
}
```

Arduino Uno (sensor bau)

```
const int mq9AnalogPin = A0;  // Pin analog yang
terhubung ke sensor MQ-9
const int mq9DigitalPin = 2;  // Pin digital yang
terhubung ke sensor MQ-9 (opsional)

void setup() {
  Serial.begin(9600);         // Memulai komunikasi
serial dengan baud rate 9600
  pinMode(mq9DigitalPin, INPUT);  // Mengatur pin
digital sebagai input
}

void loop() {
  int analogValue = analogRead(mq9AnalogPin);  //
Membaca nilai analog dari sensor MQ-9
```

```
  int digitalValue = digitalRead(mq9DigitalPin);  //
Membaca nilai digital dari sensor MQ-9

  // Konversi nilai analog ke tegangan
  float voltage = analogValue * (5.0 / 1023.0);

  // Menampilkan nilai analog, tegangan, dan nilai
digital di Serial Monitor
  Serial.print("Analog Value: ");
  Serial.print(analogValue);
  Serial.print("\t Voltage: ");
  Serial.print(voltage);
  Serial.print("V\t Digital Value: ");
  Serial.println(digitalValue);

  delay(1000);  // Menunggu 1 detik sebelum membaca
nilai sensor lagi
}
```

Google Collab

- **Fungsi Mendapatkan Nilai Minimun dan Maximum RGB**

```
# @title Default title text

import os

import cv2

import pandas as pd



from google.colab import drive



drive.mount('/content/drive')
```

```python
# Function to get the average RGB values from an image using OpenCV
def get_average_rgb(image_path):
    image = cv2.imread(image_path)
    avg_color_per_row = image.mean(axis=0)
    avg_color = avg_color_per_row.mean(axis=0)
    return int(avg_color[2]), int(avg_color[1]), int(avg_color[0])  # Convert BGR to RGB

# Process a folder to get the average RGB values of all images
def process_folder(folder_path):
    rgb_values = []
    for file_name in os.listdir(folder_path):
        file_path = os.path.join(folder_path, file_name)
        if os.path.isfile(file_path) and file_name.lower().endswith(('.png', '.jpg', '.jpeg')):
            avg_color = get_average_rgb(file_path)
            rgb_values.append({'r': avg_color[0], 'g': avg_color[1], 'b': avg_color[2]})
            # rgb_values.append(file_name)
    return rgb_values

# Define paths to the image folders
belum_matang_path = '/content/drive/MyDrive/KlasifikasiMatang/BelumMatang'
setengah_matang_path = '/content/drive/MyDrive/KlasifikasiMatang/SetengahMatang'
```

```
matang_path = '/content/drive/MyDrive/KlasifikasiMatang/Matang'


# Get RGB values for each category
data_belum_matang = process_folder(belum_matang_path)
data_setengah_matang = process_folder(setengah_matang_path)
data_matang = process_folder(matang_path)


# Create DataFrames for each category
df_belum_matang = pd.DataFrame(data_belum_matang)
df_setengah_matang = pd.DataFrame(data_setengah_matang)
df_matang = pd.DataFrame(data_matang)


# Calculate min and max for each DataFrame
min_belum_matang = df_belum_matang.min()
max_belum_matang = df_belum_matang.max()


min_setengah_matang = df_setengah_matang.min()
max_setengah_matang = df_setengah_matang.max()


min_matang = df_matang.min()
max_matang = df_matang.max()




# df = pd.DataFrame(data)
# Cetak DataFrame dengan pemisah tab
print(df_belum_matang.to_csv(sep='\t', index=False))
print(df_setengah_matang.to_csv(sep='\t', index=False))
print(df_matang.to_csv(sep='\t', index=False))

print("df_belum_matang: ", df_belum_matang)
print("df_setengah_matang: ", df_setengah_matang)
print("df_matang: ", df_matang)
```

- **Fungsi Memberhship Function RGB**

```
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz

# Define the range of x values
x_rgb = np.arange(0, 256, 1)

# Define the membership functions for Red, Green, and Blue
mean_belum_matang = df_belum_matang.mean().tolist()
mean_setengah_matang = df_setengah_matang.mean().tolist()
mean_matang = df_matang.mean().tolist()
```

```
# Define the membership functions for Red, Green, and Blue

red_belum_matang = fuzz.trapmf(x_rgb, [0,0,
min_belum_matang['r'], max_belum_matang['r']])

red_setengah_matang = fuzz.trimf(x_rgb,
[min_setengah_matang['r'], max_belum_matang['r'],
max_setengah_matang['r']])

red_matang = fuzz.trapmf(x_rgb, [min_matang['r'],
max_matang['r'], 255, 255])





print(max_setengah_matang['b'], max_matang['b'])

green_belum_matang = fuzz.trapmf(x_rgb, [0, 0,
min_belum_matang['g'], max_belum_matang['g']])

green_setengah_matang = fuzz.trimf(x_rgb,
[min_setengah_matang['g'], max_belum_matang['g'],
max_setengah_matang['g']])

green_matang = fuzz.trapmf(x_rgb, [min_matang['g'],
max_matang['g'],  255, 255])


blue_belum_matang = fuzz.trapmf(x_rgb, [0, 0,
min_belum_matang['b'], max_belum_matang['b']])

blue_setengah_matang = fuzz.trimf(x_rgb,
[min_setengah_matang['b'],min_belum_matang['b'],
max_setengah_matang['b'],])

blue_matang = fuzz.trapmf(x_rgb, [min_matang['b'],
max_matang['b'], 255, 255])
```

```python
# Plot membership functions for Red
plt.figure(figsize=(10, 5))
plt.plot(x_rgb, red_belum_matang, 'r', label='Red - Belum Matang')
plt.plot(x_rgb, red_setengah_matang, 'g', label='Red - Setengah Matang')
plt.plot(x_rgb, red_matang, 'b', label='Red - Matang')
plt.title('Red Membership Functions')
plt.xlabel('Color Intensity')
plt.ylabel('Membership')
plt.legend()
plt.grid(True)

plt.show()

# Plot membership functions for Green
plt.figure(figsize=(10, 5))
plt.plot(x_rgb, green_belum_matang, 'r', label='Green - Belum Matang')
plt.plot(x_rgb, green_setengah_matang, 'g', label='Green - Setengah Matang')
plt.plot(x_rgb, green_matang, 'b', label='Green - Matang')
plt.title('Green Membership Functions')
plt.xlabel('Color Intensity')
plt.ylabel('Membership')
plt.legend()
plt.grid(True)


plt.show()


# Plot membership functions for Blue

plt.figure(figsize=(10, 5))

plt.plot(x_rgb, blue_belum_matang, 'r', label='Blue - Belum Matang')

plt.plot(x_rgb, blue_setengah_matang, 'g', label='Blue - Setengah Matang')

plt.plot(x_rgb, blue_matang, 'b', label='Blue - Matang')

plt.title('Blue Membership Functions')

plt.xlabel('Color Intensity')

plt.ylabel('Membership')

plt.legend()

plt.grid(True)


plt.show()
```

- **Fungsi Pengujian RGB**

```python
def determine_maturity_red(rgb):


    maturity_level = ''


    r, g, b = rgb['r'], rgb['g'], rgb['b']

    if r >= min_belum_matang['r'] and r <= max_belum_matang['r']:

        maturity_level = 'Belum Matang'


    elif r >= min_setengah_matang['r'] and r <= max_setengah_matang['r']:

        maturity_level = 'Setengah Matang'


    elif r >= min_matang['r'] and r <= max_matang['r']:

        maturity_level = 'Matang'


    return maturity_level




belum_matang_path_test = r'/content/drive/MyDrive/KlasifikasiMatang/Testing/Belum Matang'

setengah_matang_path_test = r'/content/drive/MyDrive/KlasifikasiMatang/Testing/Setengah Matang'

matang_path_test = r'/content/drive/MyDrive/KlasifikasiMatang/Testing/Matang'
```

```
data_belum_matang_test = process_folder(belum_matang_path_test)
data_setengah_matang_test = process_folder(setengah_matang_path_test)
data_matang_test = process_folder(matang_path_test)


df_belum_matang_test = pd.DataFrame(data_belum_matang_test, columns=['r', 'g', 'b'])
df_setengah_matang_test = pd.DataFrame(data_setengah_matang_test, columns=['r', 'g', 'b'])
df_matang_test = pd.DataFrame(data_matang_test, columns=['r', 'g', 'b'])


# Example usage
df_belum_matang_test['kondisi'] = 'Belum Matang'
df_belum_matang_test['predicting'] = df_belum_matang_test.apply(determine_maturity_red, axis=1)
df_setengah_matang_test['kondisi'] = 'Setengah Matang'
df_setengah_matang_test['predicting'] = df_setengah_matang_test.apply(determine_maturity_red, axis=1)
df_matang_test['kondisi'] = 'Matang'
df_matang_test['predicting'] = df_matang_test.apply(determine_maturity_red, axis=1)




output_folder = '/content/drive/MyDrive/KlasifikasiMatang/testing'
```

```
show_color_patches(df_belum_matang_test, 'Belum Matang
Testing', output_folder)

show_color_patches(df_setengah_matang_test, 'Setengah Matang
Testing', output_folder)

show_color_patches(df_matang_test, 'Matang Testing',
output_folder)


print(df_belum_matang_test)

print(df_setengah_matang_test)

print(df_matang_test)


df_combined = pd.concat([df_belum_matang_test,
df_setengah_matang_test, df_matang_test])


# Calculate accuracy

accuracy = (df_combined['kondisi'] ==
df_combined['predicting']).mean()

print(f'Accuracy: {accuracy * 100:.2f}%')


def determine_maturity_green(rgb):


    maturity_level = 'Belum Matang'


    r, g, b = rgb['r'], rgb['g'], rgb['b']


    if g >= min_belum_matang['g'] and g <=
max_belum_matang['g']:
        maturity_level = 'Belum Matang'

    elif g >= min_setengah_matang['g'] and g <=
max_setengah_matang['g']:
        maturity_level = 'Setengah Matang'

    elif g >= min_matang['g'] and g <= max_matang['g']:
        maturity_level = 'Matang'

    return maturity_level


belum_matang_path_test =
r'/content/drive/MyDrive/KlasifikasiMatang/Testing/Belum
Matang'
setengah_matang_path_test =
r'/content/drive/MyDrive/KlasifikasiMatang/Testing/Setengah
Matang'
matang_path_test =
r'/content/drive/MyDrive/KlasifikasiMatang/Testing/Matang'

data_belum_matang_test = process_folder(belum_matang_path_test)
data_setengah_matang_test =
process_folder(setengah_matang_path_test)
data_matang_test = process_folder(matang_path_test)

df_belum_matang_test = pd.DataFrame(data_belum_matang_test,
columns=['r', 'g', 'b'])
```

```
df_setengah_matang_test = pd.DataFrame(data_setengah_matang_test, columns=['r', 'g', 'b'])

df_matang_test = pd.DataFrame(data_matang_test, columns=['r', 'g', 'b'])


# Example usage

df_belum_matang_test['kondisi'] = 'Belum Matang'

df_belum_matang_test['predicting'] = df_belum_matang_test.apply(determine_maturity_green, axis=1)

df_setengah_matang_test['kondisi'] = 'Setengah Matang'

df_setengah_matang_test['predicting'] = df_setengah_matang_test.apply(determine_maturity_green, axis=1)

df_matang_test['kondisi'] = 'Matang'

df_matang_test['predicting'] = df_matang_test.apply(determine_maturity_green, axis=1)




output_folder = '/content/drive/MyDrive/KlasifikasiMatang/testing'

show_color_patches(df_belum_matang_test, 'Belum Matang Testing', output_folder)

show_color_patches(df_setengah_matang_test, 'Setengah Matang Testing', output_folder)

show_color_patches(df_matang_test, 'Matang Testing', output_folder)


print(df_belum_matang_test)

print(df_setengah_matang_test)
print(df_matang_test)


df_combined = pd.concat([df_belum_matang_test,
df_setengah_matang_test, df_matang_test])


# Calculate accuracy
accuracy = (df_combined['kondisi'] ==
df_combined['predicting']).mean()
print(f'Accuracy: {accuracy * 100:.2f}%')





def determine_maturity_blue(rgb):


    maturity_level = 'Belum Matang'


    r, g, b = rgb['r'], rgb['g'], rgb['b']


    if b >= min_belum_matang['b'] and b <=
max_belum_matang['b']:
        maturity_level = 'Belum Matang'


    elif b >= min_setengah_matang['b'] and b <=
max_setengah_matang['b']:
        maturity_level = 'Setengah Matang'
    elif b >= min_matang['b'] and b <= max_matang['b']:

        maturity_level = 'Matang'


    return maturity_level




# Example usage

df_belum_matang_test['kondisi'] = 'Belum Matang'

df_belum_matang_test['predicting'] =
df_belum_matang_test.apply(determine_maturity_blue, axis=1)

df_setengah_matang_test['kondisi'] = 'Setengah Matang'

df_setengah_matang_test['predicting'] =
df_setengah_matang_test.apply(determine_maturity_blue, axis=1)

df_matang_test['kondisi'] = 'Matang'

df_matang_test['predicting'] =
df_matang_test.apply(determine_maturity_blue, axis=1)



output_folder =
'/content/drive/MyDrive/KlasifikasiMatang/testing'

show_color_patches(df_belum_matang_test, 'Belum Matang
Testing', output_folder)

show_color_patches(df_setengah_matang_test, 'Setengah Matang
Testing', output_folder)

show_color_patches(df_matang_test, 'Matang Testing',
output_folder)
```

```
print(df_belum_matang_test)

print(df_setengah_matang_test)

print(df_matang_test)


df_combined = pd.concat([df_belum_matang_test,
df_setengah_matang_test, df_matang_test])


# Calculate accuracy

accuracy = (df_combined['kondisi'] ==
df_combined['predicting']).mean()

print(f'Accuracy: {accuracy * 100:.2f}%')
```

- **Fungsi Membership Bau**

```
data = {

    'nilai': [(202, 203, 204), (179, 180, 181), (173, 174, 175),
(165, 166, 167),

              (158, 159, 160), (142, 144, 145), (135, 136, 137),
(131, 132, 133),

              (132, 133, 134), (148, 149, 150)],

    'tingkat_kebauan': ['belum_matang', 'belum_matang',
'belum_matang', 'belum_matang',

                        'setengah_matang', 'setengah_matang',
'setengah_matang', 'setengah_matang',

                        'matang', 'matang']
}



# Membuat DataFrame

df_kebauan = pd.DataFrame(data)



# Pecah data menjadi tiga kategori

df_belum_matang_bau = df_kebauan[df_kebauan['tingkat_kebauan']
== 'belum_matang']

df_setengah_matang_bau =
df_kebauan[df_kebauan['tingkat_kebauan'] == 'setengah_matang']

df_matang_bau = df_kebauan[df_kebauan['tingkat_kebauan'] ==
'matang']





min_belum_matang_bau =
df_belum_matang_bau['nilai'].apply(lambda x: min(x)).min()

max_belum_matang_bau =
df_belum_matang_bau['nilai'].apply(lambda x: max(x)).max()



min_setengah_matang_bau =
df_setengah_matang_bau['nilai'].apply(lambda x: min(x)).min()

max_setengah_matang_bau =
df_setengah_matang_bau['nilai'].apply(lambda x: max(x)).max()



min_matang_bau = df_matang_bau['nilai'].apply(lambda x:
min(x)).min()

max_matang_bau = df_matang_bau['nilai'].apply(lambda x:
max(x)).max()
```

```
print(min_belum_matang_bau, max_belum_matang_bau,
min_setengah_matang_bau, max_setengah_matang_bau,
min_matang_bau, max_matang_bau )

# Define the range of x values for the membership functions

x = np.arange(0, 256, 1)


print(min_setengah_matang_bau, max_setengah_matang_bau,
max_matang_bau)

belum_matang = fuzz.trapmf(x, [0, 0, min_belum_matang_bau ,
max_belum_matang_bau])

setengah_matang = fuzz.trimf(x, [min_setengah_matang_bau,
max_matang_bau, max_setengah_matang_bau])

matang = fuzz.trapmf(x, [min_matang_bau, max_matang_bau, 255,
255])


# Plotting

plt.figure(figsize=(8, 6))


plt.plot(x, belum_matang, 'r', label='Belum Matang')

plt.plot(x, setengah_matang, 'g', label='Setengah Matang')

plt.plot(x, matang, 'b', label='Matang')


plt.title('Fungsi Keanggotaan for Tingkat Kebauan')

plt.xlabel('Tingkat Kebauan')

plt.ylabel('Membership')

plt.legend()
plt.grid(True)
plt.show()
```

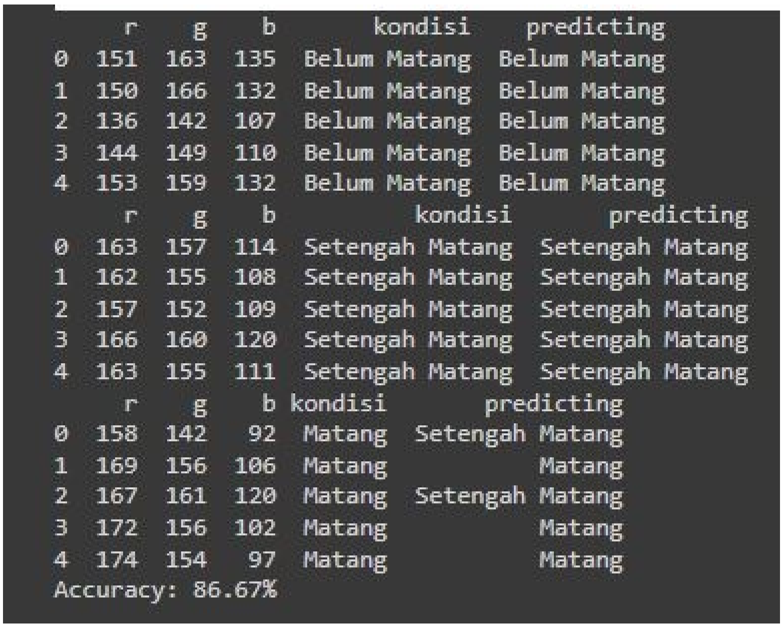
- **Pengujian Membership Function Bau**

```
def determine_maturity_bau(rgb):
    maturity_level = ''

    r, g, b = rgb['nilai']
    if b >= min_belum_matang_bau and b <= max_belum_matang_bau:
        maturity_level = 'belum_matang'

    elif b >= min_setengah_matang_bau and b <=
max_setengah_matang_bau:
        maturity_level = 'setengah_matang'

    elif b >= min_matang_bau and b <= max_belum_matang_bau:
        maturity_level = 'matang'

    return maturity_level
```

```
# Menambahkan kolom 'tingkat_kematangan' ke DataFrame

df_kebauan['predicting'] =
df_kebauan.apply(determine_maturity_bau, axis=1)


# Membagi DataFrame berdasarkan tingkat kematangan

df_belum_matang_bau = df_kebauan[df_kebauan['predicting'] ==
'belum_matang']

df_setengah_matang_bau = df_kebauan[df_kebauan['predicting'] ==
'setengah_matang']

df_matang_bau = df_kebauan[df_kebauan['predicting'] == 'matang']


# Output

print(df_belum_matang_bau)

print(df_setengah_matang_bau)

print(df_matang_bau)


df_combined = pd.concat([df_belum_matang_bau,
df_setengah_matang_bau, df_matang_bau])


# Calculate accuracy

accuracy = (df_combined['tingkat_kebauan'] ==
df_combined['predicting']).mean()

print(f'Accuracy: {accuracy * 100:.2f}%')
```

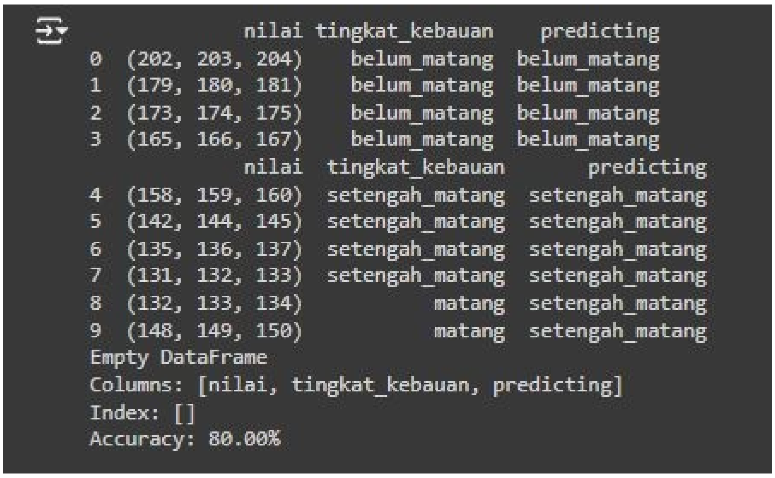
Lampiran 3 Hasil Akurasi

- **Warna**



```
     r    g    b       kondisi    predicting
0  151  163  135  Belum Matang  Belum Matang
1  150  166  132  Belum Matang  Belum Matang
2  136  142  107  Belum Matang  Belum Matang
3  144  149  110  Belum Matang  Belum Matang
4  153  159  132  Belum Matang  Belum Matang
     r    g    b          kondisi       predicting
0  163  157  114  Setengah Matang  Setengah Matang
1  162  155  108  Setengah Matang  Setengah Matang
2  157  152  109  Setengah Matang  Setengah Matang
3  166  160  120  Setengah Matang  Setengah Matang
4  163  155  111  Setengah Matang  Setengah Matang
     r    g    b kondisi       predicting
0  158  142   92  Matang  Setengah Matang
1  169  156  106  Matang           Matang
2  167  161  120  Matang  Setengah Matang
3  172  156  102  Matang           Matang
4  174  154   97  Matang           Matang
Accuracy: 86.67%
```

- **Bau**



```
             nilai tingkat_kebauan    predicting
0  (202, 203, 204)    belum_matang  belum_matang
1  (179, 180, 181)    belum_matang  belum_matang
2  (173, 174, 175)    belum_matang  belum_matang
3  (165, 166, 167)    belum_matang  belum_matang
             nilai  tingkat_kebauan       predicting
4  (158, 159, 160)  setengah_matang  setengah_matang
5  (142, 144, 145)  setengah_matang  setengah_matang
6  (135, 136, 137)  setengah_matang  setengah_matang
7  (131, 132, 133)  setengah_matang  setengah_matang
8  (132, 133, 134)           matang  setengah_matang
9  (148, 149, 150)           matang  setengah_matang
Empty DataFrame
Columns: [nilai, tingkat_kebauan, predicting]
Index: []
Accuracy: 80.00%
```