LAMPIRAN

LAMPIRAN 1

Lampiran 1 Data Pengujian Nomor Benang

100% Rayon 30/2 S Twist			
NO	Panjang (yard)	Berat (g)	Nomor
1	120	4,258	15,22
2	120	4,571	14,18
3	120	4,264	15,20
4	120	4,287	15,12
5	120	4,509	14,37
6	120	4,336	14,95
7	120	4,307	15,05
8	120	4,323	14,99
9	120	4,280	15,14
10	120	4,291	15,10
11	120	4,293	15,10
12	120	4,277	15,15
13	120	4,271	15,17
14	120	4,282	15,13
15	120	4,240	15,28
16	120	4,376	14,81
17	120	4,276	15,15
18	120	4,282	15,13
19	120	4,315	15,02
20	120	4,170	15,54
21	120	4,322	14,99
22	120	4,337	14,94
23	120	4,217	15,37
24	120	4,147	15,63
25	120	4,296	15,09
Rata-Rata			15,07189
Min			14,17633
Max			15,62575
SD			0,03254
CV			0,215899

Lampiran 1 Lanjutan Data Pengujian Nomor Benang

100% Rayon 30/2 Z Twist			
NO	Panjang (yard)	Berat (g)	Nomor
1	120	4,267	15,18631
2	120	4,286	15,11793
3	120	4,187	15,47537
4	120	4,498	14,4064
5	120	4,340	14,93088
6	120	4,295	15,08766
7	120	4,170	15,53957
8	120	4,260	15,21198
9	120	4,282	15,13312
10	120	4,527	14,31412
11	120	4,266	15,18952
12	120	4,276	15,15293
13	120	4,333	14,95431
14	120	4,307	15,04423
15	120	4,352	14,89142
16	120	4,239	15,28807
17	120	4,364	14,84774
18	120	4,290	15,10595
19	120	4,461	14,52589
20	120	4,264	15,19878
21	120	4,330	14,96432
22	120	4,292	15,0975
23	120	4,239	15,28771
24	120	4,315	15,01773
25	120	4,349	14,89963
Rata-Rata			15,03476
Min			14,31412
Max			15,53957
SD			0,03183
CV			0,211709

Lampiran 1 Lanjutan Data Pengujian Nomor Benang

100% Rayon Bright 30 Z Twist			
NO	Panjang (yard)	Berat (g)	Nomor
1	120	2,155	30,06821
2	120	2,147	30,17743
3	120	2,164	29,94316
4	120	2,164	29,94178
5	120	2,135	30,34987
6	120	2,172	29,837
7	120	2,173	29,82601
8	120	2,153	30,10313
9	120	2,148	30,17462
10	120	2,162	29,97086
11	120	2,157	30,03615
12	120	2,161	29,98612
13	120	2,142	30,25775
14	120	2,177	29,77257
15	120	2,148	30,169
16	120	2,149	30,15356
17	120	2,143	30,24081
18	120	2,152	30,10873
19	120	2,138	30,30445
20	120	2,100	30,85714
21	120	2,150	30,13813
22	120	2,152	30,11432
23	120	2,145	30,21683
24	120	2,152	30,11852
25	120	2,157	30,04312
Rata-Rata			30,11637
Min			29,77257
Max			30,85714
SD			0,03183
CV			0,10569

Lampiran 1 Lanjutan Data Pengujian Metode *Image Processing*

100% Rayon 30/2 Z Twist				
NO	Sudut	Tan	Diameter(mm)	TPI
1	62,25	1,901	0,3327	46,21
2	61,23	1,821	0,3557	41,42
3	52,97	1,326	0,2991	35,85
4	61,54	1,845	0,2957	50,47
5	56,51	1,511	0,3125	39,12
6	62,6	1,929	0,3159	49,40
7	56,19	1,493	0,3024	39,94
8	61,41	1,835	0,3260	45,53
9	59,07	1,669	0,3024	44,64
10	59,3	1,684	0,3058	44,55
11	60,3	1,753	0,3529	40,19
12	66,65	2,316	0,3361	55,75
13	55,77	1,470	0,2688	44,23
14	57,49	1,569	0,3260	38,93
15	56,11	1,489	0,3327	36,20
Rata-Rata				43
Min				36
Max				56
SD				1,477
CV				3,395709601

Lampiran 1 (Lanjutan) Data Pengujian Metode Image Processing

100% Rayon 30/2 S Twist				
NO	Sudut	Tan	Diameter(mm)	TPI
1	58,1	1,607	0,3058	42,50
2	59,37	1,689	0,3495	39,09
3	54,33	1,393	0,2957	38,11
4	66,01	2,247	0,3058	59,44
5	49,3	1,163	0,3260	28,85
6	51,8	1,271	0,3697	27,81
7	57,5	1,570	0,3325	38,19
8	53,3	1,342	0,3293	32,96
9	59,89	1,724	0,3529	39,53
10	66,29	2,277	0,3327	55,36
11	62,62	1,931	0,3125	49,98
12	70,53	2,829	0,3562	64,24
13	53,12	1,333	0,3192	33,78
14	62,32	1,906	0,3697	41,71
15	63,78	2,030	0,3293	49,88
Rata-Rata				43
Min				28
Max				64
SD				2,912
CV				6,809983671

Lampiran 1 (Lanjutan) Data Pengujian Metode Image Processing

100% Rayon 30 Z Twist				
NO	Sudut	Tan	Diameter(mm)	TPI
1	32,28	0,632	0,2163	23,62
2	33,17	0,654	0,2621	20,17
3	35,76	0,720	0,2218	26,26
4	28,73	0,548	0,2122	20,90
5	34,93	0,698	0,2145	26,34
6	35,55	0,715	0,2621	22,06
7	32,99	0,649	0,2419	21,71
8	31,49	0,613	0,1982	25,00
9	33,66	0,666	0,2151	25,04
10	28,61	0,545	0,2221	19,87
11	30,18	0,582	0,2117	22,22
12	33,16	0,653	0,2352	22,47
13	30,35	0,586	0,2251	21,04
14	36,26	0,733	0,2580	23,00
15	27,33	0,517	0,2117	19,75
Rata-Rata				23
Min				20
Max				26
SD				0,589
CV				2,602762846

Lampiran 1 (Lanjutan) Data Pengujian Metode Konvensional

100% Rayon 30/2 S Twist		
NO	Putaran	TPI
1	844	42,88
2	797	40,49
3	798	40,54
4	834	42,37
5	866	43,99
6	807	41,00
7	764	38,81
8	799	40,59
9	772	39,22
10	787	39,98
11	791	40,18
12	797	40,49
13	787	39,98
14	839	42,62
15	795	40,39
Rata-Rata		40,90
Min		38,81
Max		43,99
SD		0,3834
CV		0,937388714

Lampiran 1 (Lanjutan) Data Pengujian Metode Konvensional

100% Rayon 30/2 Z Twist		
NO	Putaran	TPI
1	859	43,6
2	855	43,4
3	861	43,7
4	839	42,6
5	876	44,5
6	789	40,1
7	819	41,6
8	837	42,5
9	844	42,9
10	830	42,2
11	768	39,0
12	833	42,3
13	837	42,5
14	867	44,0
15	849	43,1
Rata-Rata		43
Min		39
Max		45
SD		0,3883
CV		0,912642542

Lampiran 1 (Lanjutan) Data Pengujian Metode Konvensional

100% Rayon 30 Z Twist		
NO	Putaran	TPI
1	443	22,50
2	445	22,61
3	438	22,25
4	447	22,71
5	473	24,03
6	472	23,98
7	393	19,96
8	398	20,22
9	419	21,29
10	444	22,56
11	453	23,01
12	492	24,99
13	497	25,25
14	480	24,38
15	476	24,18
Rata-Rata		22,93
Min		19,96
Max		25,25
SD		0,425
CV		1,853646603

Lampiran 1 Data Pengujian Sudut

30/2 S	
No	Sudut
1	54,81
2	55,99
3	54,26
4	54,58
5	55,71
6	51,91
7	53,29
8	54,64
9	54,37
10	56,5
11	57,24
12	54,93
13	52,91
14	55,71
15	56,36
Rata-Rata	54,88067
Min	51,91
Max	57,24
SD	0,3848
CV	0,701158

Lampiran 1 (Lanjutan) Data Pengujian Sudut

30/2 Z	
No	Sudut
1	52,3
2	51,91
3	52,74
4	52,96
5	53,35
6	52,69
7	53,79
8	50,18
9	52,32
10	53
11	52
12	51,71
13	51,96
14	51,43
15	52,39
Rata-Rata	52,31533333
Min	50,18
Max	53,79
SD	0,232
CV	0,443464631

Lampiran 1 (Lanjutan) Data Pengujian Sudut

30 Z	
No	Sudut
1	29,61
2	30,3
3	29,94
4	31,73
5	29,43
6	31,2
7	30,18
8	30,95
9	31,19
10	31,08
11	30,85
12	30,92
13	30,57
14	30,11
15	31,29
Rata-Rata	30,62333333
Min	29,43
Max	31,73
SD	0,1787
CV	0,583541961

Lampiran 1 (Lanjutan) Uji Normalitas Metode Konvensional

Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Ne1	Statistic	df	Sig.	Statistic	df	Sig.
T Konv	30/2 S Twist	.253	15	.011	.908	15	.127
	30/2 Z Twist	.195	15	.129	.902	15	.103
	30 Z Twist	.146	15	.200*	.945	15	.452

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Lampiran 1 (Lanjutan) Uji Normalitas Metode *Image Processing*

Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Ne1	Statistic	df	Sig.	Statistic	df	Sig.
T Image	30/2 S Twist	.194	15	.134	.944	15	.428
	30/2 Z Twist	.139	15	.200*	.946	15	.467
	30 Z Twist	.211	15	.072	.900	15	.096

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Lampiran 1 (Lanjutan) Uji Homogenitas

Tests of Homogeneity of Variances

		Levene Statistic	df1	df2	Sig.
TPI	Based on Mean	2.658	1	88	.107
	Based on Median	3.935	1	88	.050
	Based on Median and with adjusted df	3.935	1	86.887	.050
	Based on trimmed mean	3.112	1	88	.081

Lampiran 1 (Lanjutan) Data Pengolahan Statistik Uji t Independen

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Antihan	Equal variances assumed	2.307	.132	.426	88	.671	.95556	2.24558	-3.50707	5.41818
	Equal variances not assumed			.426	82.212	.672	.95556	2.24558	-3.51145	5.42257

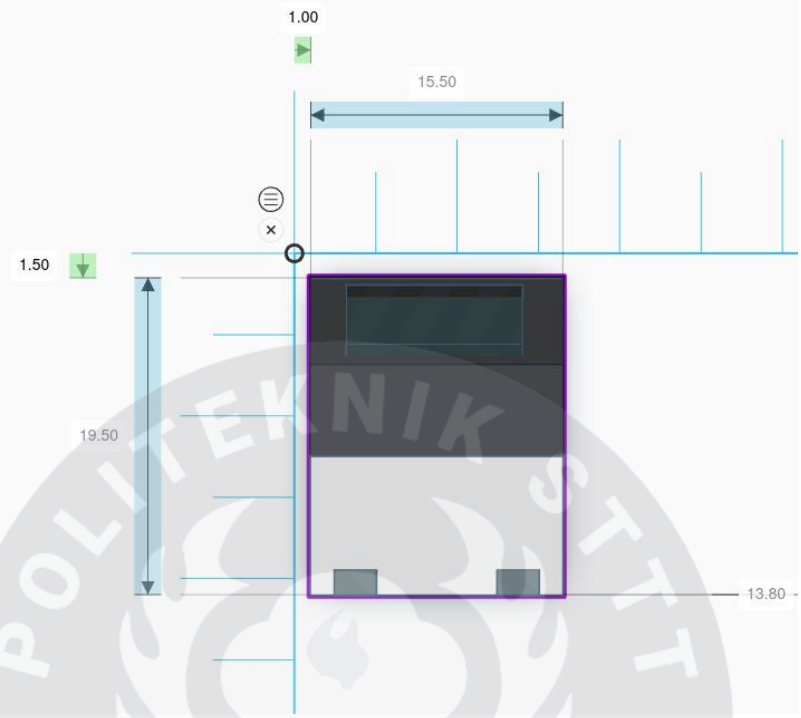
Lampiran 1 (Lanjutan) Pengolahan Statistik Uji t Independen Nilai Sudut

Independent Samples Test

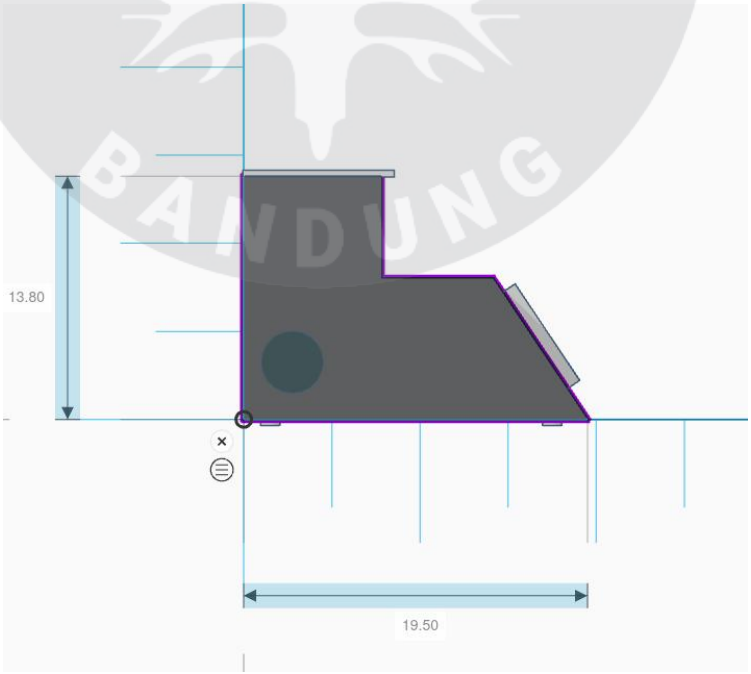
		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Sudut	Equal variances assumed	2.903	.092	-1.662	88	.100	-4.32911	2.60552	-9.50703	.84881
	Equal variances not assumed			-1.662	84.600	.100	-4.32911	2.60552	-9.50993	.85171

LAMPIRAN 2

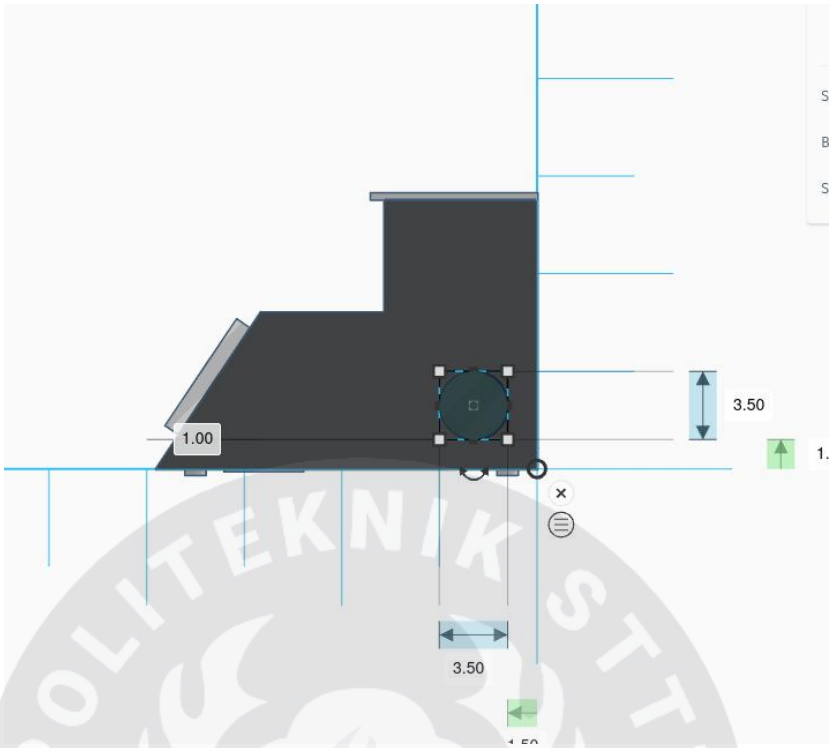
Lampiran 2 Desain Alat Tampak Atas dalam satuan (cm)



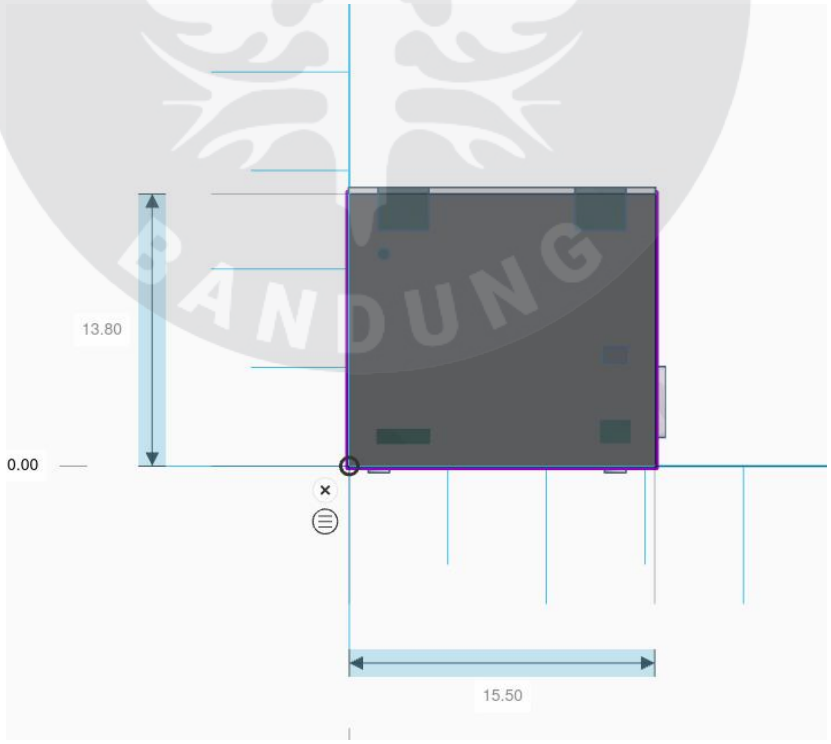
Lampiran 2 (lanjutan) Desain Alat Tampak Samping dalam satuan (cm)



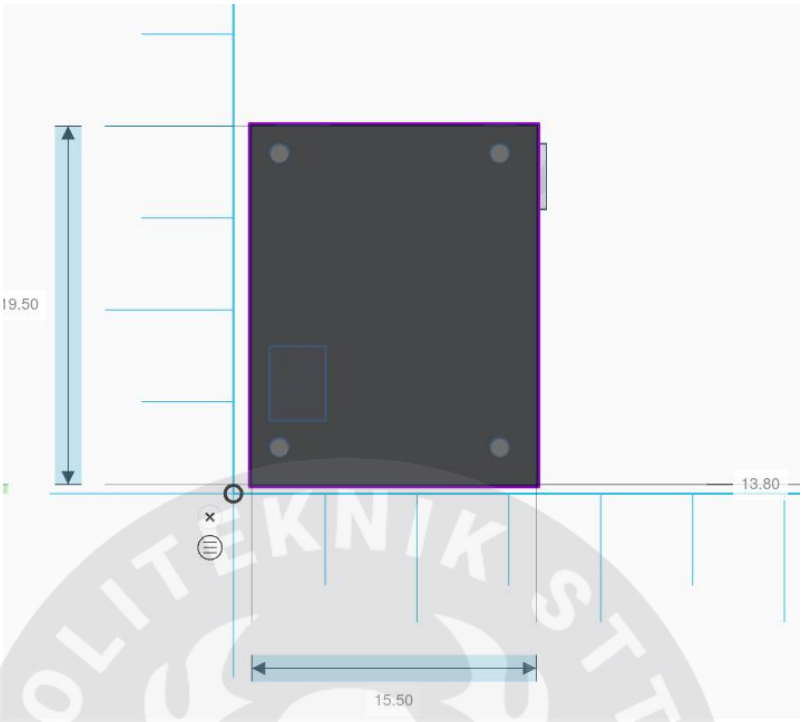
Lampiran 2 (lanjutan) Desain Alat Tampak Samping dalam satuan (cm)



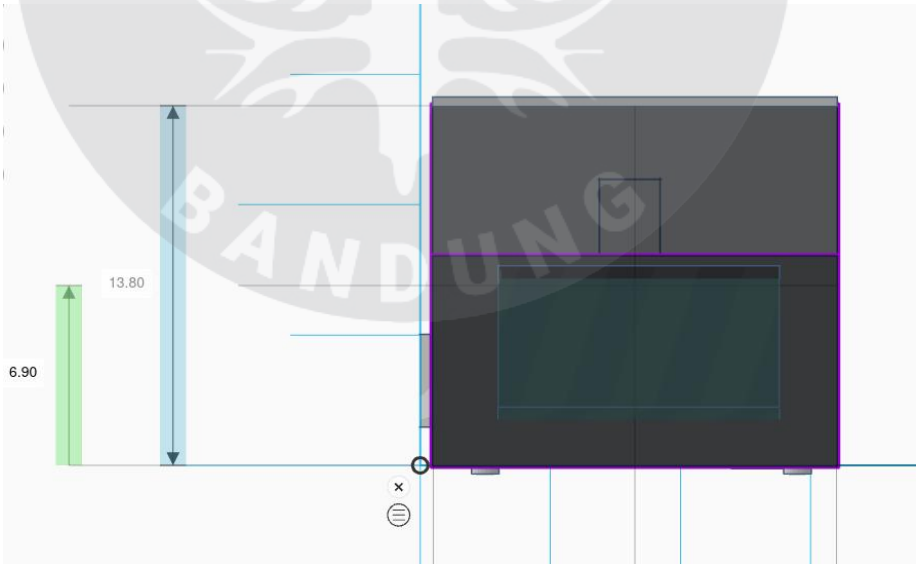
Lampiran 2 (lanjutan) Desain Alat Tampak Belakang dalam satuan (cm)



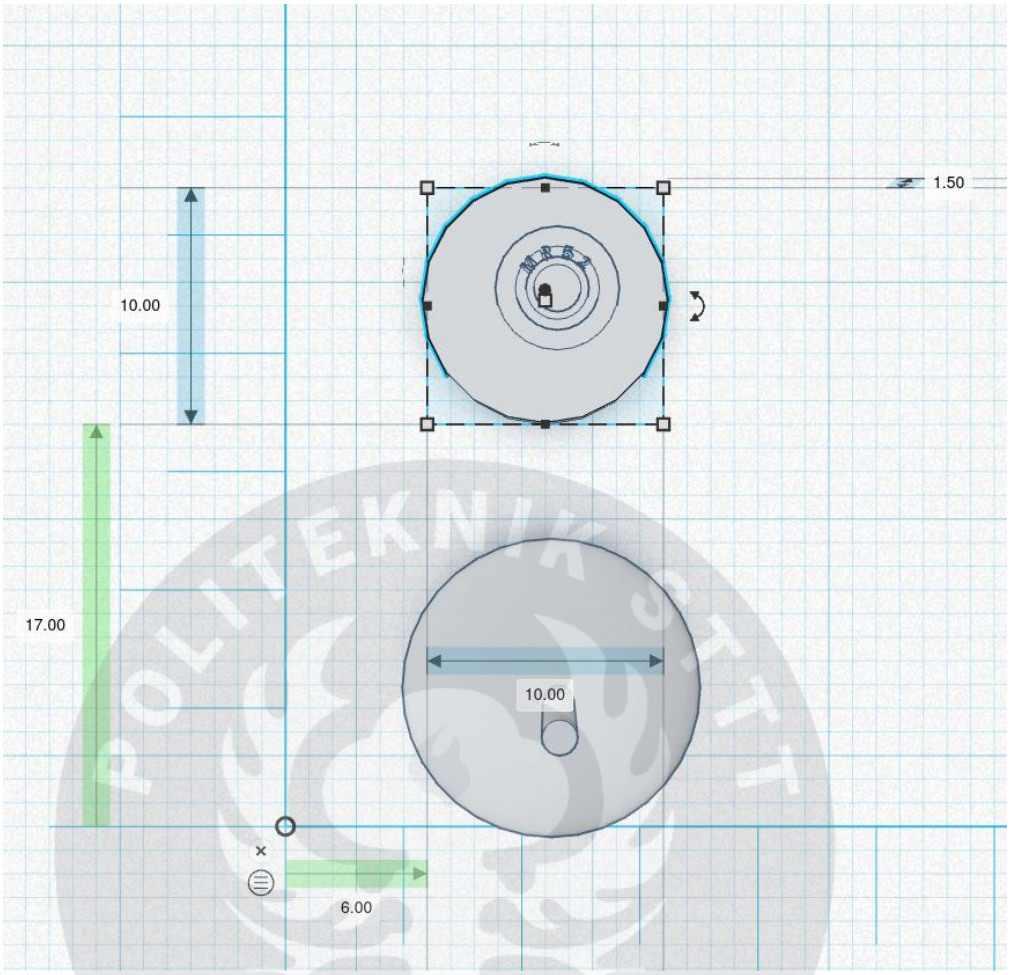
Lampiran 2 (lanjutan) Desain Alat Tampak Bawah dalam satuan (cm)



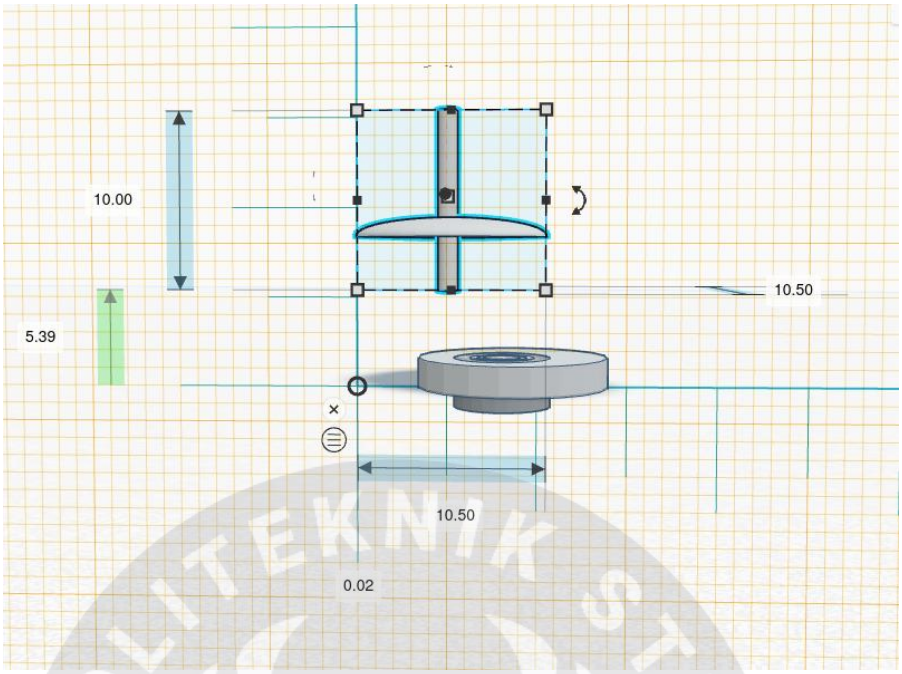
Lampiran 2 (lanjutan) Desain Alat Tampak Depan dalam satuan (cm)



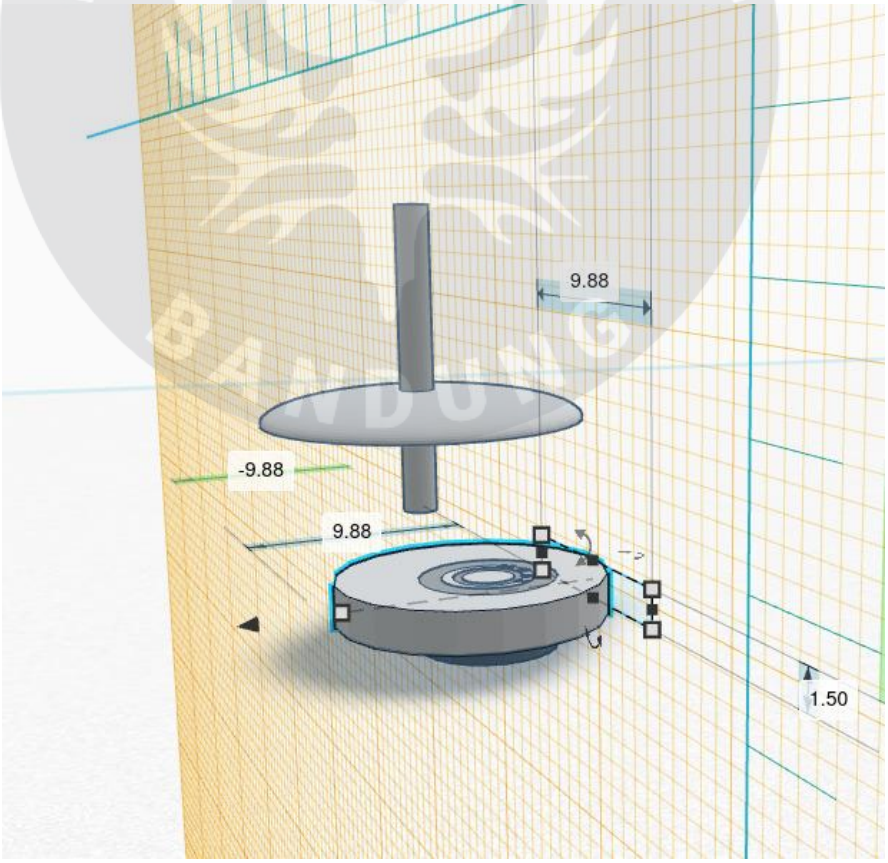
Lampiran 2 (lanjutan) Desain *Cone Holder* Tampak Atas dalam satuan (cm)



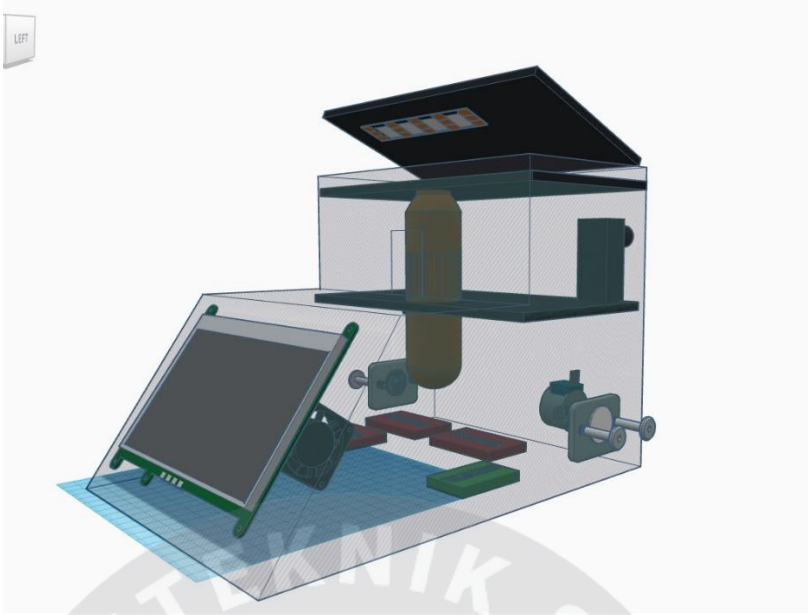
Lampiran 2 (lanjutan) Desain *Cone Holder* Tampak Depan dalam satuan (cm)



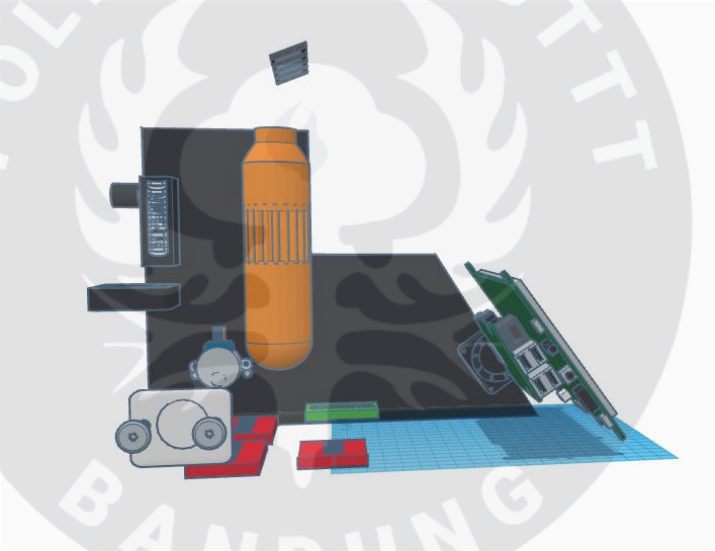
Lampiran 2 (lanjutan) Desain *Cone Holder* Tampak samping dalam satuan (cm)



Lampiran 2 (lanjutan) Penempatan komponen tampak isometrik



Lampiran 2 (lanjutan) Penempatan komponen tampak samping



Lampiran 2 (lanjutan) Penempatan komponen

LAMPIRAN 3

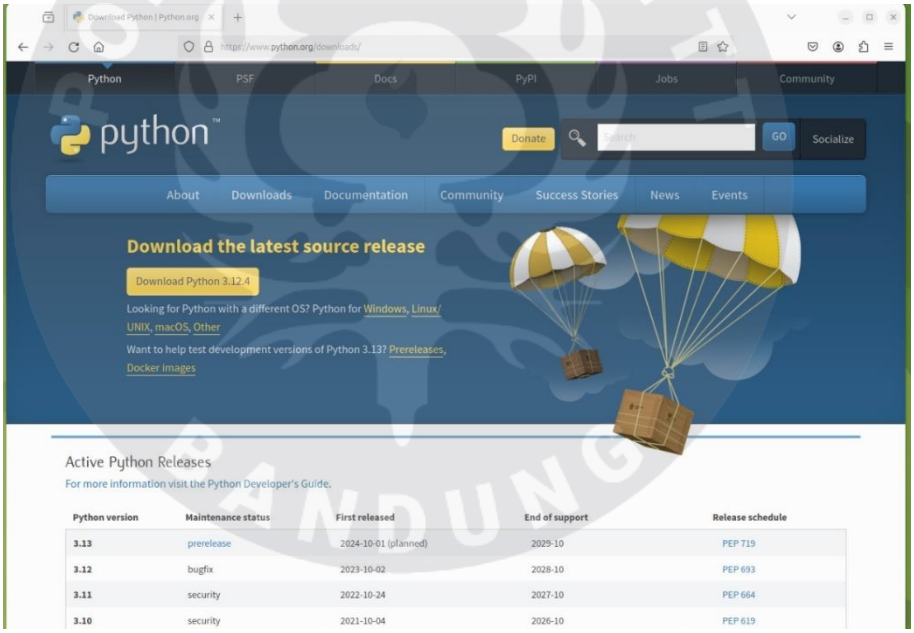
Lampiran 3 Spesifikasi *Server*

Processor	Core i5 8600 6 core/6 thread
Memory	16 Gigabyte
Penyimpanan	SSD 128 Gigabyte
Kartu Grafis	NVIDIA GTX 1060 6 GB
Sistem Operasi	Ubuntu Server 24.04 LTS

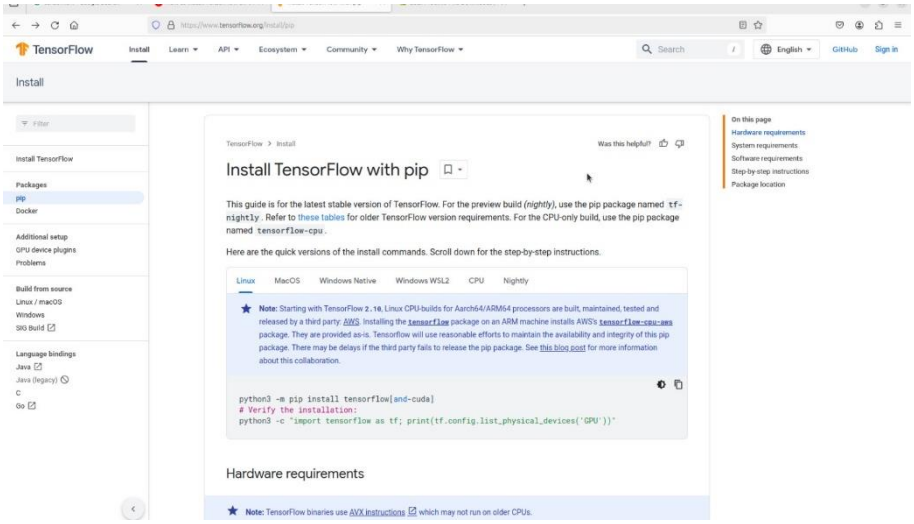
Lampiran 3 Spesifikasi *Raspberry PI*

Model	4B
Processor	Cortex-A72 4 Core
Memory	2 Gigabyte
Penyimpanan	Micro SD 16 Gigabyte
Sistem Operasi	Raspbian OS

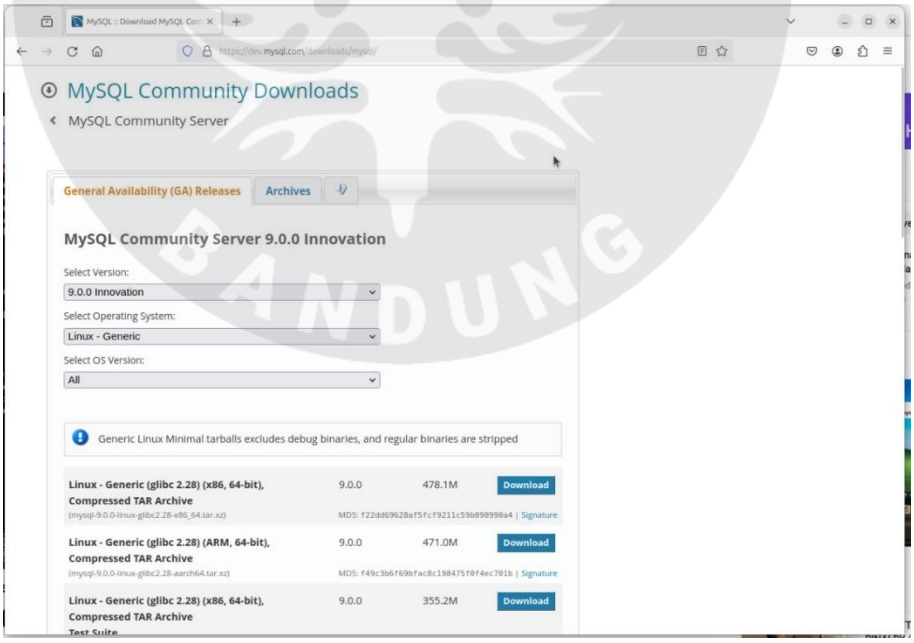
Lampiran 3 Instalasi *Python*



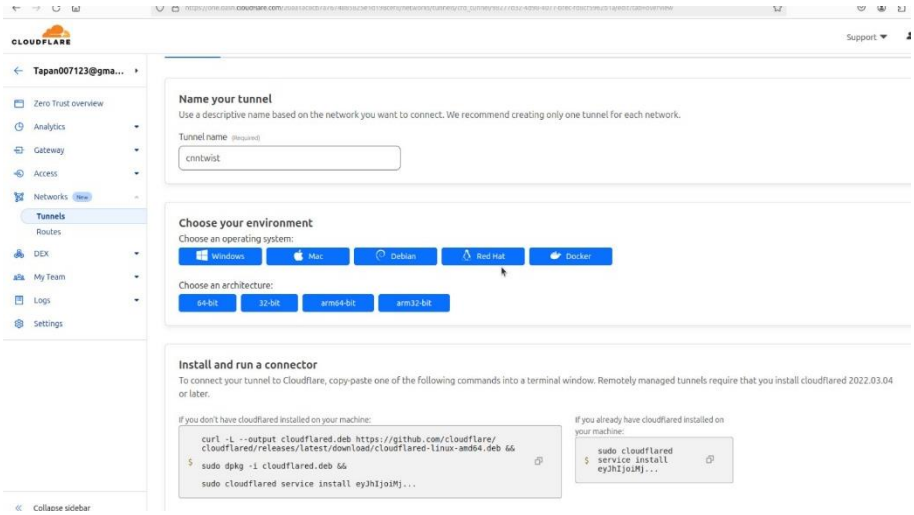
Lampiran 3 Instalasi *Tensorflow*



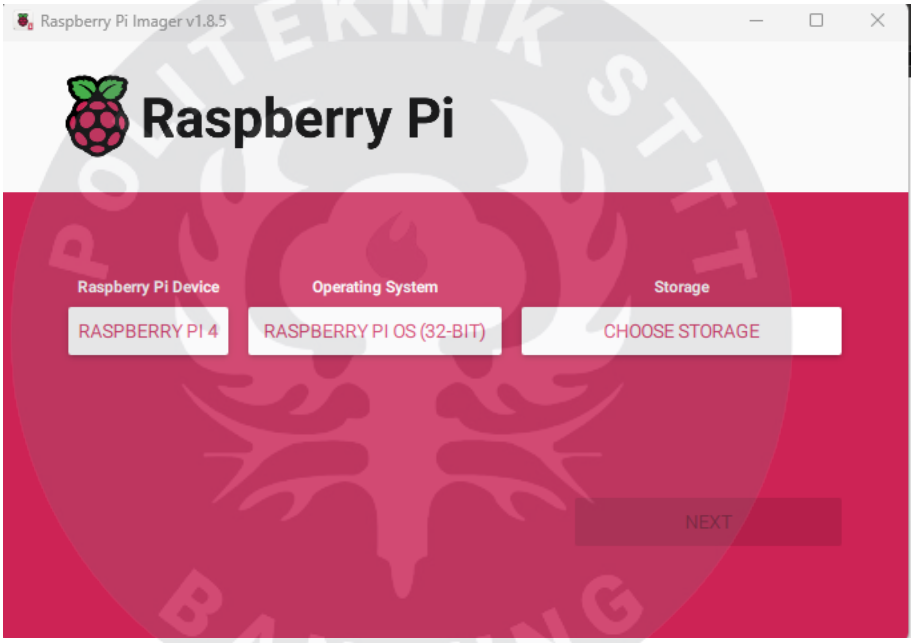
Lampiran 3 Instalasi *Mysql*



Lampiran 3 Instalasi *Cloudflare*



Lampiran 3 Instalasi Sistem Operasi *Raspberry Pi*



Lampiran 3 Kode Sisi *Server* – Training Model *CNN*

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set seed for reproducibility
tf.random.set_seed(42)

# Define constants
IMAGE_SIZE = (128, 128)
BATCH_SIZE = 16  # Reduce batch size
EPOCHS = 20

# Define data directories
train_dir = '/home/thafhan/Documents/CNN FINAL/FOTO BENANG/'
validation_dir = '/home/thafhan/Documents/CNN FINAL/VALIDASI/'

# Data preprocessing
train_datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

validation_datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
validation_generator = validation_datagen.flow_from_directory(
    train_dir,  # Use the same directory for validation, but different subset
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)

# Calculate steps_per_epoch and validation_steps
steps_per_epoch = train_generator.samples // BATCH_SIZE
validation_steps = validation_generator.samples // BATCH_SIZE
```

```
# Define the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train the model with manual looping to handle generator exhaustion
history = {'accuracy': [], 'val_accuracy': [], 'loss': [], 'val_loss': []}

for epoch in range(EPOCHS):
    print(f"Epoch {epoch+1}/{EPOCHS}")

    # Reset the generators
    train_generator.reset()
    validation_generator.reset()

    # Fit the model
    hist = model.fit(
        train_generator,
        steps_per_epoch=steps_per_epoch,
        validation_data=validation_generator,
        validation_steps=validation_steps,
        epochs=1,
        verbose=1
    )

    # Collect accuracy and loss metrics
    history['accuracy'].append(hist.history['accuracy'][0])
    history['val_accuracy'].append(hist.history['val_accuracy'][0])
    history['loss'].append(hist.history['loss'][0])
    history['val_loss'].append(hist.history['val_loss'][0])
```

```
# Save the model
model.save('/home/thafhan/Documents/CNN FINAL/model1.h5')

# Plot training history
plt.figure(figsize=(12, 4))

# Plot accuracy
plt.subplot(1, 2, 1)
plt.plot(history['accuracy'], label='Training Accuracy')
plt.plot(history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history['loss'], label='Training Loss')
plt.plot(history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')

plt.show()
```

Lampiran 3 (lanjutan) Kode Sisi *Server* – Klasifikasi dan Perhitungan

```
from flask import Flask, render_template, request, jsonify, session,
redirect, url_for
import tensorflow as tf
import cv2
import numpy as np
import base64
from datetime import datetime
import mysql.connector
import os
import paho.mqtt.client as mqtt
import time
from werkzeug.security import generate_password_hash,
check_password_hash

app = Flask(__name__)
app.secret_key = 'your_secret_key'
```

```
# MQTT settings
MQTT_BROKER =
'd46d543ca6ac43458a2eb15f66343a70.s1.eu.hivemq.cloud'
MQTT_PORT = 8883
MQTT_USERNAME = 'thafhan'
MQTT_PASSWORD = '$Realgn11'
MQTT_TOPIC = 'stepper/control'
MQTT_CLIENT_ID = 'flask_server_client'

# Load the pre-trained model
model = tf.keras.models.load_model('/home/thafhan/Documents/CNN
FINAL/model1.h5')
model.summary()

# Database connection
db_config = {
    'user': 'thafhan',
    'password': '#Realgn11',
    'host': 'localhost',
    'database': 'cnn_twist'
}

def get_db_connection():
    return mysql.connector.connect(**db_config)

def authenticate_user(username, password):
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "SELECT password FROM users WHERE username=%s"
            cursor.execute(sql, (username,))
            result = cursor.fetchone()
            if result and check_password_hash(result[0], password):
                return True
    finally:
        connection.close()
    return False

def register_user (username, password):
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "SELECT id FROM users WHERE username=%s"
            cursor.execute(sql, (username,))
            result = cursor.fetchone()
```

```
            if result:
                return "Username already exists"
            hashed_password = generate_password_hash(password)
            sql = "INSERT INTO users (username, password) VALUES (%s,
%s)"
            cursor.execute(sql, (username, hashed_password))
        connection.commit()
        return "User registered successfully"
    finally:
        connection.close()

def get_user_id(username):
    connection = get_db_connection()
    user_id = None
    try:
        with connection.cursor() as cursor:
            sql = "SELECT id FROM users WHERE username=%s"
            cursor.execute(sql, (username,))
            result = cursor.fetchone()
            if result:
                user_id = result[0]
    finally:
        connection.close()
    return user_id

def save_test_data(user_id, thread_number, batch, lot):
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = """
            INSERT INTO tests (user_id, thread_number, batch, lot,
timestamp)
            VALUES (%s, %s, %s, %s, %s)
            """
            cursor.execute(sql, (user_id, thread_number, batch, lot,
datetime.utcnow()))
        connection.commit()
    finally:
        connection.close()

def save_test_details(test_id, tipe_benang, arah_antihan,
diameter_benang, tpm, tpi):
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = """
```

```
        INSERT INTO test_details (test_id, tipe_benang, arah_antihan,
diameter_benang, tpm, tpi)
        VALUES (%s, %s, %s, %s, %s, %s)
        """
        cursor.execute(sql, (test_id, tipe_benang, arah_antihan,
diameter_benang, tpm, tpi))
      connection.commit()
   finally:
      connection.close()

def classify_image(image):
   input_size = (128, 128)
   resize = tf.image.resize(image, input_size)
   yhat = model.predict(np.expand_dims(resize / 255, 0))
   class_label = 'Z Twist' if yhat > 0.5 else 'S Twist'
   return class_label

def save_image(image, class_label, test_id):
   # Ensure the directory for the class exists
   save_dir = f"/home/thafhan/Documents/CNN
FINAL/FOTO/{class_label}"
   os.makedirs(save_dir, exist_ok=True)

   # Generate timestamp
   timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

   # Save the image with a unique filename based on test_id and
timestamp
   filename = f"{class_label}_test_{test_id}_{timestamp}.jpg"
   filepath = os.path.join(save_dir, filename)

   # Save the image
   cv2.imwrite(filepath, cv2.cvtColor(image, cv2.COLOR_RGB2BGR))
   return filename

def save_to_db(test_id, classification, timestamp):
   conn = mysql.connector.connect(**db_config)
   cursor = conn.cursor()
   cursor.execute('INSERT INTO twist_classifications (test_id,
classification, timestamp) VALUES (%s, %s, %s)', (test_id, classification,
timestamp))
   conn.commit()
   cursor.close()
   conn.close()

def capture_image(data_url):
```

```
    header, encoded = data_url.split(",", 1)
    data = base64.b64decode(encoded)
    npimg = np.frombuffer(data, np.uint8)
    image = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
    rgb_frame = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return rgb_frame

@app.route('/')
def index():
    return redirect(url_for('login'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if authenticate_user(username, password):
            session['username'] = username
            return redirect(url_for('dashboard'))
        else:
            return render_template('login.html', error="Invalid credentials")
    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        message = register_user(username, password)
        return render_template('register.html', message=message)
    return render_template('register.html')

@app.route('/dashboard')
def dashboard():
    if 'username' not in session:
        return redirect(url_for('login'))
    return render_template('dashboard.html',
username=session['username'])

@app.route('/form', methods=['GET', 'POST'])
def form():
    if 'username' not in session:
        return redirect(url_for('login'))
    if request.method == 'POST':
        thread_number = request.form['thread_number']
        batch = request.form['batch']
```

```
        lot = request.form['lot']
        user_id = get_user_id(session['username'])
        save_test_data(user_id, thread_number, batch, lot)
        return redirect(url_for('test_options'))
    return render_template('form.html')

@app.route('/test_options')
def test_options():
    if 'username' not in session:
        return redirect(url_for('login'))
    return render_template('test_options.html')

@app.route('/logout')
def logout():
    session.pop('username', None)
    return redirect(url_for('login'))

@app.route('/test_yarn_type')
def test_yarn_type():
    if 'username' not in session:
        return redirect(url_for('login'))
    return render_template('arah_twist.html')

@app.route('/classify', methods=['POST'])
def classify_frame():
    data_url = request.json['image']
    image = capture_image(data_url)
    class_label = classify_image(image)
    return jsonify({'label': class_label})

@app.route('/start_test', methods=['POST'])
def start_test():
    try:
        num_tests = int(request.json['num_tests'])

        client = mqtt.Client(client_id=MQTT_CLIENT_ID)
        client.username_pw_set(MQTT_USERNAME, MQTT_PASSWORD)
        client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
        client.connect(MQTT_BROKER, MQTT_PORT, 60)
        client.loop_start()

        classifications = []

        print(f"Number of tests: {num_tests}")

        for i in range(1, num_tests + 1):
```

```
            print(f"Sending FORWARD command for test {i}")
            client.publish(MQTT_TOPIC, 'FORWARD')
            time.sleep(10)  # Allow the machine to run for 10 seconds

            print(f"Sending STOP command for test {i}")
            client.publish(MQTT_TOPIC, 'STOP')
            time.sleep(5)  # Wait for the machine to stop

            # Polling for image from client
            response = request.json.get(f'image_{i}')
            if response:
                image = capture_image(response)
                classification = classify_image(image)
                filename = save_image(image, classification, i)
                timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                classifications.append(classification)
                save_to_db(i, classification, timestamp)

        client.loop_stop()
        client.disconnect()
        return jsonify({'result': 'Tests completed successfully', 'classifications':
classifications})

    except Exception as e:
        print(f"Error in start_test: {e}")
        return jsonify({'error': str(e)}), 500

@app.route('/results')
def results():
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM twist_classifications')
    data = cursor.fetchall()
    cursor.close()
    conn.close()
    return render_template('results.html', data=data)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Lampiran 3 (lanjutan) Kode Sisi *Server* Tampilan *(Front End) - Login*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login</title>
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        <form method="post">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" class="form-control" id="username"
name="username" required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" class="form-control" id="password"
name="password" required>
            </div>
            {% if error %}
                <div class="alert alert-danger">{{ error }}</div>
            {% endif %}
            <button type="submit" class="btn btn-primary">Login</button>
        </form>
        <br>
        <a href="{{ url_for('register') }}">Register</a>
    </div>
</body>
</html>
```

Lampiran 3 (lanjutan) Kode Sisi *Server* Tampilan *(Front End) – Register*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Register</title>
```

```
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <h2>Register</h2>
        <form method="post">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" class="form-control" id="username"
name="username" required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" class="form-control" id="password"
name="password" required>
            </div>
            {% if message %}
                <div class="alert alert-info">{{ message }}</div>
            {% endif %}
            <button type="submit" class="btn btn-primary">Register</button>
        </form>
        <br>
        <a href="{{ url_for('login') }}">Login</a>
    </div>
</body>
</html>
```

Lampiran 3 (lanjutan) Kode Sisi *Server* Tampilan *(Front End)* – *Dashboard*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Dashboard</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.mi
n.css">
</head>
<body>
    <div class="container">
        <h1 class="my-4">Dashboard - {{ username }}</h1>
```

```
      <div class="btn-group-vertical">
        <a href="{{ url_for('form') }}" class="btn btn-primary btn-lg btn-
block">Mulai Test</a>
        <a href="{{ url_for('results') }}" class="btn btn-primary btn-lg btn-
block">Data Saya</a>
        <a href="{{ url_for('logout') }}" class="btn btn-danger btn-lg btn-
block">Logout</a>
      </div>
   </div>
</body>
</html>
```

Lampiran 3 (lanjutan) Kode Sisi *Server* Tampilan *(Front End) – Form*

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Form Pengujian</title>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.mi
n.css">
</head>
<body>
   <div class="container">
      <h1 class="my-4">Form Pengujian</h1>
      <form method="POST">
        <div class="form-group">
          <label for="thread_number">Nomor Benang</label>
          <input type="text" class="form-control" id="thread_number"
name="thread_number" required>
        </div>
        <div class="form-group">
          <label for="batch">Batch</label>
          <input type="text" class="form-control" id="batch" name="batch"
required>
        </div>
        <div class="form-group">
          <label for="lot">Lot</label>
          <input type="text" class="form-control" id="lot" name="lot"
required>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
      </form>
```

```
    </div>
</body>
</html>
```

Lampiran 3 (lanjutan) Kode Sisi *Server* Tampilan *(Front End) – Laman Uji*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Twist Direction Classifier</title>
    <!-- Bootstrap CSS -->
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.mi
n.css" rel="stylesheet">
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>
    <!-- Bootstrap JS (optional for certain components) -->
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.j
s"></script>
    <style>
        body {
            text-align: center; /* Center align the content */
            font-size: 16px; /* Base font size */
        }
        h1 {
            font-size: 28px; /* Font size for h1 (judul) */
        }
        video, canvas {
            display: block;
            margin: 0 auto; /* Center the video and canvas */
            padding: 0; /* Remove any padding */
            width: 100%; /* Full width */
            height: auto; /* Maintain aspect ratio */
        }
        .container {
            max-width: 100%; /* Ensure container width is full width */
            padding: 0; /* Remove any padding */
        }
        #result {
```

```
            font-size: 18px; /* Font size for result */
            margin-top: 20px; /* Top margin for result */
            padding: 10px; /* Padding for result */
            border: 2px solid #007bff; /* Border color for result */
            border-radius: 5px; /* Border radius for result */
            background-color: #f0f0f0; /* Background color for result */
            color: #007bff; /* Text color for result */
            font-weight: bold; /* Bold font weight for result */
        }
        #result p {
            margin-bottom: 0; /* Remove bottom margin for paragraph inside
result */
        }
    </style>
</head>
<body class="container-fluid">
    <div class="text-center mt-3">
        <h1 style="margin-bottom: 10px;">Twist Direction Classifier</h1>

        <div class="row mt-3">
            <div class="col-md-8 mx-auto">
                <video id="video" autoplay></video>
                <canvas id="canvas" style="display:none;"></canvas>
            </div>
        </div>

        <div class="row mt-3 justify-content-center">
            <div class="col-md-4">
                <form id="test-form" class="form-inline">
                    <div class="form-group flex-fill mr-2">
                        <label class="sr-only" for="num_tests">Number of
Tests</label>
                        <input type="number" id="num_tests" name="num_tests"
min="1" max="100" required class="form-control w-100"
placeholder="Number of Tests">
                    </div>
                    <button type="submit" class="btn btn-primary flex-fill">Start
Test</button>
                </form>
            </div>
        </div>

        <div class="row mt-3">
            <div class="col-md-6">
                <button id="capture" class="btn btn-success btn-block">Classify
Image</button>
```

```
            </div>
            <div class="col-md-6">
                <button id="refresh" class="btn btn-secondary btn-
block">Refresh</button>
            </div>
        </div>

        <div id="result" class="mt-3"></div>

        <!-- Modal -->
        <div class="modal fade" id="testCompleteModal" tabindex="-1"
role="dialog" aria-labelledby="testCompleteModalLabel" aria-
hidden="true">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title" id="testCompleteModalLabel">Test
Completed</h5>
                        <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <p>Test completed successfully!</p>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <script>
        const video = document.getElementById('video');
        const canvas = document.getElementById('canvas');
        const context = canvas.getContext('2d');
        const captureButton = document.getElementById('capture');
        const refreshButton = document.getElementById('refresh');
        const resultDiv = document.getElementById('result');
        const testForm = document.getElementById('test-form');

        // Get access to the webcam
```

```
        if (navigator.mediaDevices &&
navigator.mediaDevices.getUserMedia) {
            navigator.mediaDevices.getUserMedia({ video: { width: 1280,
height: 960 } }).then(function(stream) {
                video.srcObject = stream;
                video.play();
            });
        }

        captureButton.onclick = async () => {
            context.drawImage(video, 0, 0, canvas.width, canvas.height);
            const dataUrl = canvas.toDataURL('image/jpeg');
            const response = await fetch('/classify', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ image: dataUrl })
            });
            const result = await response.json();
            const classification = result.label === 'Z_Twist' ? 'Z Twist' : 'S
Twist';
            alert(classification);
            resultDiv.innerHTML = '<p>Hasil Klasifikasi:</p><p style="font-
size: 24px;">' + result.label + '</p>';
        };

        refreshButton.onclick = () => {
            location.reload(); // Reload the page
        };

        testForm.onsubmit = async (e) => {
            e.preventDefault();
            const num_tests = document.getElementById('num_tests').value;
            const images = {};
            for (let i = 1; i <= num_tests; i++) {
                context.drawImage(video, 0, 0, canvas.width, canvas.height);
                images[`image_${i}`] = canvas.toDataURL('image/jpeg');
            }
            const response = await fetch('/start_test', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ num_tests, ...images })
            });
            const result = await response.json();
            resultDiv.innerHTML = '<p>Hasil Klasifikasi:</p><p style="font-
size: 24px;">' + result.result + '</p>';
```

```
            // Show test completion modal
            $('#testCompleteModal').modal('show');
        };
    </script>
</body>
</html>
```

Lampiran 3 (lanjutan) Kode Sisi *Raspberry PI*

```
import time
import paho.mqtt.client as paho
from paho import mqtt
import RPi.GPIO as GPIO
import logging

# Pengaturan MQTT
MQTT_BROKER =
'd46d543ca6ac43458a2eb15f66343a70.s1.eu.hivemq.cloud'
MQTT_PORT = 8883
MQTT_USERNAME = 'thafhan'
MQTT_PASSWORD = '$Realgn11'
MQTT_TOPIC = 'stepper/control'
MQTT_CLIENT_ID = 'raspberry_pi_client'

# Setup logging
logging.basicConfig(level=logging.DEBUG)

# Pin GPIO motor stepper
IN1 = 19
IN2 = 26
IN3 = 20
IN4 = 21

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)

# Mendefinisikan urutan langkah motor stepper
urutan_langkah = [
    [1, 0, 0, 0],
    [1, 1, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
```

```
    [0, 0, 1, 0],
    [0, 0, 1, 1],
    [0, 0, 0, 1],
    [1, 0, 0, 1]
]

# Fungsi untuk mengatur pin motor
def setel_pin_motor(langkah):
    GPIO.output(IN1, langkah[0])
    GPIO.output(IN2, langkah[1])
    GPIO.output(IN3, langkah[2])
    GPIO.output(IN4, langkah[3])

# Fungsi untuk menjalankan motor
def jalankan_motor(kecepatan):
    jumlah_langkah = len(urutan_langkah)
    for i in range(jumlah_langkah):
        setel_pin_motor(urutan_langkah[i])
        time.sleep(kecepatan)

def on_connect(client, userdata, flags, rc, properties=None):
    if rc == 0:
        logging.info("Terhubung ke broker")
        client.subscribe(MQTT_TOPIC)
    else:
        logging.error(f"Gagal terhubung dengan kode {rc}")

def on_message(client, userdata, msg):
    perintah = msg.payload.decode()
    logging.info(f"Menerima perintah: {perintah}")
    if perintah == 'MAJU':
        waktu_mulai = time.time()
        while time.time() - waktu_mulai < 10:
            jalankan_motor(0.001)  # Sesuaikan kecepatan sesuai kebutuhan
    elif perintah == 'BERHENTI':
        logging.info("Berhenti")
        setel_pin_motor([0, 0, 0, 0])

def on_subscribe(client, userdata, mid, granted_qos, properties=None):
    logging.info(f"Berlangganan: {mid} {granted_qos}")

def on_publish(client, userdata, mid, properties=None):
    logging.info(f"mid: {mid}")

# Inisialisasi Klien MQTT
```

```
client = paho.Client(client_id=MQTT_CLIENT_ID, userdata=None,
protocol=paho.MQTTv311)
client.on_connect = on_connect
client.on_message = on_message
client.on_subscribe = on_subscribe
client.on_publish = on_publish
# Aktifkan TLS untuk koneksi aman
client.tls_set()

# Mengatur nama pengguna dan kata sandi
client.username_pw_set(MQTT_USERNAME, MQTT_PASSWORD)

# Menghubungkan ke HiveMQ Cloud pada port 8883 (default untuk
MQTT)
logging.info("Menghubungkan ke broker")
client.connect(MQTT_BROKER, MQTT_PORT)
logging.info("Terhubung ke broker")

# Loop selamanya
client.loop_forever()
```