

# LAPORAN PROYEK KELOMPOK IoT's

## Inthernet Of Things

---



No	NIM	Nama	Prodi
1.	11323047	Agnes Elyestra Sidabutar	D3TI
2.	11323053	Natasya Dian Elena Siahaan	D3TI

**Institut Teknologi Del**  
**Fakultas Vokasi 2024/2025**

## 1. Pendahuluan

### 1.1. Latar Belakang

Proyek motion detector menggunakan kamera HP yang akan menyambung ke dalam database IoT memiliki beberapa latar belakang penting:

- Keamanan Rumah dan Properti:

Berbagai kasus pencopelan dan pelacakan telah membuat keamanan menjadi prioritas utama. Sistem deteksi gerakan dapat membantu mencegah masalah keamanan dengan mengidentifikasi adanya aktivitas tak diharapkan<sup>2</sup>.

- Teknologi Modern:

Teknologi modern seperti Internet of Things (IoT) memungkinkan integrasi perangkat-perangkat pintar untuk meningkatkan efisiensi dan keamanan. Integrasi kamera HP dengan database IoT memungkinkan pengguna untuk memantau lingkungan secara real-time<sup>12</sup>.

- Efektifitas dan Fleksibilitas:

Sistem deteksi gerakan dapat diprogram untuk mengirimkan notifikasi kepada pengguna melalui berbagai saluran komunikasi, seperti email, SMS, atau bahkan aplikasi IoT. Hal ini meningkatkan efektifitas respons terhadap situasi darurat

Proyek ini juga menerapkan penggunaan MQTT yaitu protokol komunikasi berbasis publish-subscribe yang ringan, dirancang untuk perangkat dengan sumber daya terbatas. Ini memungkinkan perangkat seperti kamera ponsel untuk mengirimkan data deteksi gerakan secara efisien tanpa membebani jaringan atau perangkat itu sendiri. Dengan menggunakan MQTT dalam proyek motion detector ini, sistem menjadi lebih efisien, responsif, dan mudah diintegrasikan dengan berbagai perangkat IoT lainnya. Protokol ini tidak hanya meningkatkan komunikasi antara kamera ponsel dan server tetapi juga memastikan bahwa data deteksi gerakan dapat dikelola dengan baik untuk analisis lebih lanjut, memberikan solusi keamanan yang lebih komprehensif bagi pengguna.

### 1.2. Tujuan Proyek

**Tujuan dari proyek motion detector ini adalah sebagai berikut:**

1. Deteksi Gerakan:

Membangun sistem yang mampu mendeteksi gerakan menggunakan kamera ponsel secara efektif. Sistem akan mengidentifikasi perubahan dalam frame video untuk menentukan apakah ada aktivitas yang terjadi.

2. Pencatatan Pergerakan ke Database:

Setiap kali gerakan terdeteksi, informasi tersebut akan dicatat dalam database IoT. Data yang dicatat dapat mencakup waktu deteksi, jenis gerakan (misalnya, manusia atau objek), dan lokasi deteksi.

### 3. Analisis Data Historis

Dengan menyimpan data deteksi gerakan dalam database, pengguna dapat melakukan analisis historis untuk memahami pola aktivitas di area tertentu. Ini bisa membantu dalam merencanakan tindakan keamanan lebih lanjut atau mengidentifikasi waktu-waktu rawan terjadinya pelanggaran.

#### 1.3.Komponen Utama

- **Webcam:** Digunakan untuk menangkap gambar dan video dari area yang dipantau. Webcam akan terhubung ke komputer server menggunakan kabel data.
- **Komputer Server:** Berfungsi sebagai pusat pengolahan data, di mana video dari webcam akan diproses dan disiarkan.
- **Aplikasi Web:** Dibangun menggunakan Python dengan framework web (seperti Flask atau Django) untuk menampilkan video streaming kepada pengguna melalui browser.
- **Database MySQL:** Digunakan untuk menyimpan data terkait aktivitas yang terdeteksi oleh webcam, seperti waktu deteksi gerakan dan status pemantauan.

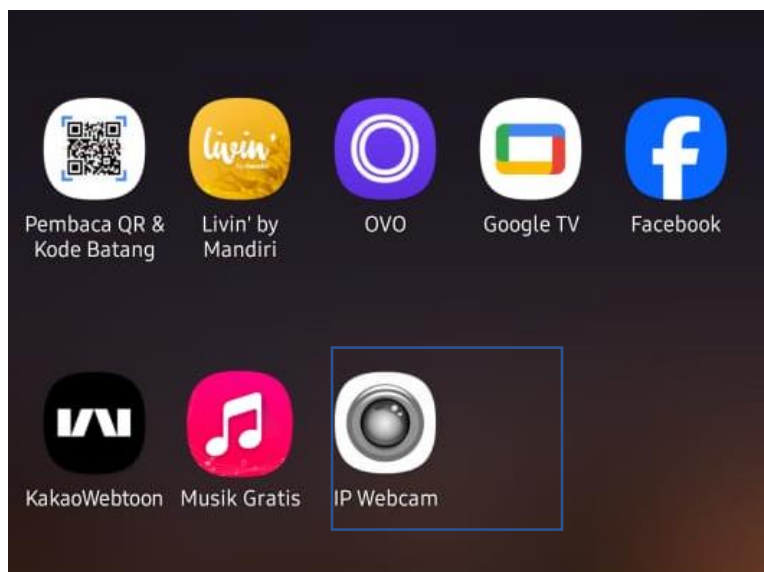
#### 1.4.Teknologi yang Digunakan

- **Python:** Bahasa pemrograman utama yang digunakan untuk mengembangkan aplikasi backend dan implementasi logika deteksi gerakan.
- **MySQL:** Database yang digunakan untuk menyimpan data terkait aktivitas pemantauan.
- **Visual Studio Code:** Editor kode yang digunakan untuk mengembangkan aplikasi, memudahkan pengelolaan proyek dengan fitur seperti debugging dan integrasi kontrol versi.
- **Kabel Data:** Menghubungkan webcam ke komputer server, memastikan koneksi stabil dan berkualitas tinggi.

## 2. Cara Kerja

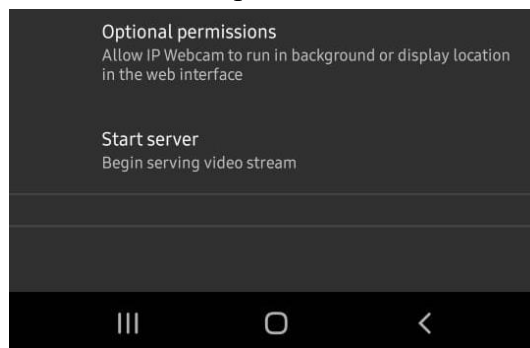
Untuk menjalankan motion detector ini ada beberapa Langkah yang harus di lakukan yaitu

### a. Install aplikasi IP Webcam



### b. Hubungkan hp yang sudah terinstall dengan aplikasi web cam menggunakan kabel data

### c. Kemudian buka aplikasi web cam tersebut dan klik start



### d. Buka code pyhton yang sudah ada

#### 1. Import Library

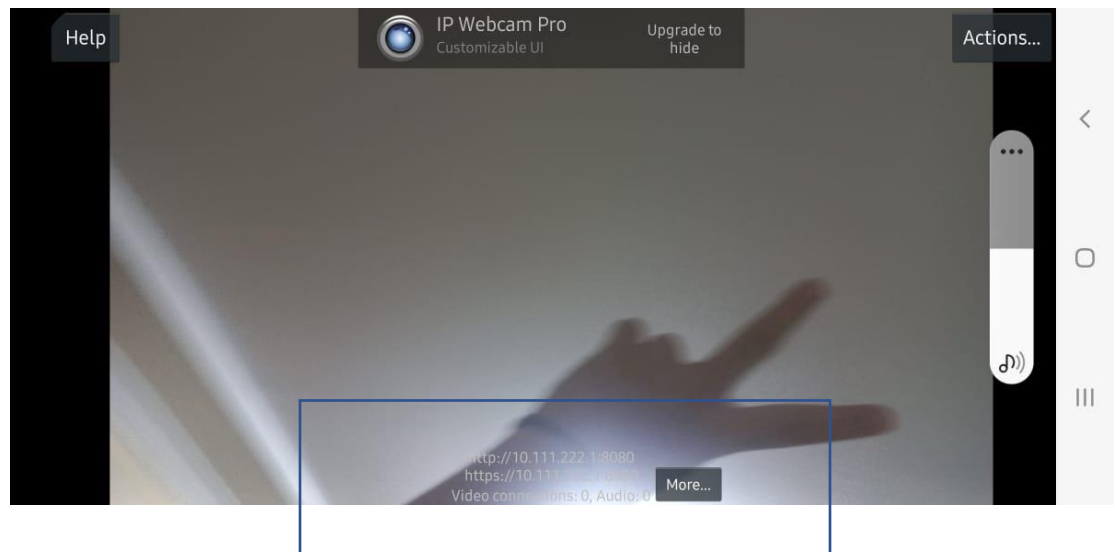
```
import cv2
import time
import paho.mqtt.client as mqtt
import mysql.connector
from datetime import datetime
```

- cv2: Digunakan untuk pemrosesan gambar dan video.
- time: Untuk mengatur waktu dan jeda.
- paho.mqtt.client: Digunakan untuk berkomunikasi dengan broker MQTT.
- mysql.connector: Untuk menghubungkan dan berinteraksi dengan database MySQL.
- datetime: Untuk mendapatkan waktu dan tanggal saat ini.

## 2. Konfigurasi URL Kamera dan MQTT

```
# URL kamera ponsel
url = "http://192.168.249.81:8080/video"
```

Sesuaikan url yang ada pada url ip web cam nya



```
# MQTT Broker
mqtt_broker = "broker.hivemq.com"
mqtt_topic = "home/motion"
```

- mqtt\_broker: Alamat broker MQTT yang digunakan untuk mengirim pesan.
- mqtt\_topic: Topik di mana pesan akan diterbitkan.

### 3. Fungsi untuk Menyimpan Data ke MySQL

```
conn = mysql.connector.connect(
    host="localhost",
    port=3307,
    user="root",
    password="",
    database="iot"
```

Fungsi ini bertanggung jawab untuk menyimpan pesan deteksi gerakan ke dalam tabel motion\_detectors di database MySQL. Jika terjadi kesalahan saat menyimpan, akan ditampilkan pesan kesalahan.

```
query = "INSERT INTO motion_detectors (date_data, message) VALUES (%s, %s)"
data = (datetime.now(), message)
cursor.execute(query, data)
conn.commit()
print("Data berhasil disimpan ke database.")
except mysql.connector.Error as e:
    print(f"Error: Tidak dapat menyimpan data ke MySQL: {e}")
finally:
    if conn.is_connected():
        cursor.close()
        conn.close()
```

- cursor = conn.cursor(): Membuat objek cursor yang digunakan untuk menjalankan perintah SQL di database yang terhubung (dalam hal ini, conn adalah objek koneksi yang sudah dibuat sebelumnya).
- query = "INSERT INTO motion\_detectors (date\_data, message) VALUES (%s, %s)": Mendefinisikan query SQL untuk memasukkan data baru ke dalam tabel motion\_detectors. Placeholder %s digunakan untuk parameter yang akan diisi.

- `data = (datetime.now(), message)`: Menyiapkan data yang akan dimasukkan ke dalam tabel. `datetime.now()` digunakan untuk mendapatkan waktu saat ini, dan `message` adalah variabel yang berisi pesan yang ingin disimpan.
- `except mysql.connector.Error as e`: Menangkap kesalahan yang mungkin terjadi saat berinteraksi dengan database. Jika terjadi kesalahan, pesan error akan dicetak ke konsol.
- `conn.commit()`: Menyimpan perubahan yang dilakukan pada database. Tanpa langkah ini, data yang dimasukkan tidak akan disimpan secara permanen.

#### 4. Setup MQTT

```
# MQTT setup
try:
    client = mqtt.Client()
    client.connect(mqtt_broker, 1883, 60)
    print("Berhasil terhubung ke MQTT Broker")
except Exception as e:
    print(f"Error: Tidak dapat terhubung ke broker MQTT: {e}")
    exit()
```

- `client = mqtt.Client()`: Membuat objek klien MQTT yang akan digunakan untuk berinteraksi dengan broker MQTT.
- `client.connect(mqtt_broker, 1883, 60)`: Mencoba untuk terhubung ke broker MQTT yang ditentukan oleh variabel `mqtt_broker` pada port 1883, dengan timeout 60 detik. Port 1883 adalah port default untuk MQTT.
- `print("Berhasil terhubung ke MQTT Broker")`: Jika koneksi berhasil, program akan mencetak pesan konfirmasi ke konsol.
- `except Exception as e`: Menangkap segala jenis pengecualian yang mungkin terjadi saat mencoba terhubung ke broker.
- `print(f"Error: Tidak dapat terhubung ke broker MQTT: {e}")`: Jika terjadi kesalahan, pesan error akan dicetak ke konsol, memberikan informasi tentang masalah yang terjadi.
- `exit()`: Menghentikan eksekusi program jika tidak dapat terhubung ke broker.

```
# Timer untuk membatasi pengiriman pesan MQTT
last_sent = time.time()
cooldown = 1 # Kirim pesan setiap 1 detik jika deteksi gerakan berulang
```

`last_sent = time.time()`: Variabel ini menyimpan waktu saat ini dalam detik sejak epoch (1 Januari 1970). Ini digunakan untuk melacak kapan pesan terakhir kali dikirim.

cooldown = 1: Variabel ini menetapkan waktu cooldown dalam detik. Dalam hal ini, cooldown diatur selama 1 detik, yang berarti pesan hanya akan dikirim sekali setiap detik jika terdeteksi adanya gerakan berulang

## 5. Loop Utama untuk Deteksi Gerakan

```
while cap.isOpened():
    diff = cv2.absdiff(frame1, frame2)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Loop ini terus berjalan selama stream video terbuka:

- Menghitung Perbedaan Frame: Menggunakan `cv2.absdiff()` untuk menghitung perbedaan antara dua frame, kemudian mengubahnya menjadi grayscale dan menerapkan Gaussian blur untuk mengurangi noise.
- Thresholding dan Dilasi: Menggunakan thresholding untuk menghasilkan gambar biner dari perbedaan, kemudian menerapkan dilasi untuk memperbesar area yang terdeteksi.
- Menemukan Kontur: Menggunakan `cv2.findContours()` untuk menemukan kontur dari area yang terdeteksi sebagai gerakan.
- Deteksi Gerakan: Jika area kontur lebih besar dari 1000 piksel, dianggap sebagai gerakan terdeteksi.

## 6. Mengirim Pesan dan Menyimpan Data

```
# Kirim pesan MQTT dan simpan ke database jika gerakan terdeteksi
if motion_detected and time.time() - last_sent > cooldown:
    try:
        client.publish(mqtt_topic, "Motion Detected")
        print("Motion Detected - Pesan dikirim ke MQTT")

        # Simpan pesan ke database
        save_to_mysql("Motion Detected")
        last_sent = time.time()
    except Exception as e:
        print(f"Error: Gagal mengirim pesan atau menyimpan data: {e}")
```

gerakan terdeteksi dan waktu sejak pengiriman pesan terakhir lebih besar dari cooldown (1 detik), maka:

- Mengirim pesan "Motion Detected" ke topik MQTT.
- Menyimpan pesan ke database menggunakan fungsi `save_to_mysql`.



## 7. Menampilkan Frame Video

```
# Tampilkan frame video
cv2.imshow("Motion Detector", frame1)
frame1 = frame2
ret, frame2 = cap.read()
```

Menampilkan frame video dengan kotak hijau di area yang terdeteksi gerakan.

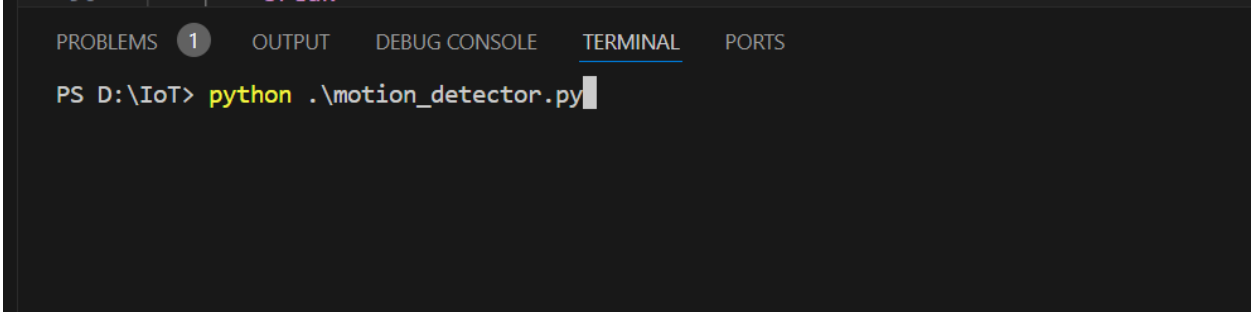
## 8. Penanganan Frame Baru dan Keluar dari Aplikasi python

```
# Periksa jika tidak ada frame baru
if not ret:
    print("Error: Tidak ada frame baru dari video stream.")
    break

# Keluar jika tombol 'Esc' ditekan
if cv2.waitKey(10) == 27:
    print("Keluar dari aplikasi.")
    break
```

Mengupdate frame untuk iterasi berikutnya. Jika tidak ada frame baru atau jika tombol 'Esc' ditekan, loop akan berhenti

e. Jalankan code Pythonnya dan ketikan pada terminalnya sebagai berikut



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\IoT> python .\motion_detector.py
```

Ketika dijalankan pada server akan muncul video dan frame dari pergerakan yang terjadi dan nantinya akan langsung tersimpan pada database, kita dapat melihat nya

f. Buka my Sql kemudian jalankan Query nya

Provides intelligent code completion for fast, accurate SQL : Reason #16 to upgrade

Query 1 x History +

```
1 SELECT * FROM motion_detectors
2
3 CREATE TABLE motion_detectors(
4     id SERIAL PRIMARY KEY,
5     date_data DATETIME,
6     message VARCHAR(255)
7 );
```

1 Result 2 Profiler 3 Messages 4 Table Data 5 Info

(Read Only)

id	date_data	message
1	2024-11-23 10:07:30	Motion Detected
2	2024-11-23 10:07:48	Motion Detected
3	2024-11-23 10:08:06	Motion Detected
4	2024-11-23 10:08:47	Motion Detected
5	2024-11-23 10:08:48	Motion Detected
6	2024-11-23 10:08:49	Motion Detected
7	2024-11-23 10:08:50	Motion Detected
8	2024-11-23 10:08:56	Motion Detected
9	2024-11-23 10:08:58	Motion Detected
10	2024-11-23 10:09:01	Motion Detected
11	2024-11-23 10:09:02	Motion Detected

Limit rows

Pada database tersebut telah tercatat pergerakan yang dilakukan setiap detik nya.