Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama:
**1. Lois Novel E Gurning [122140098]**
**2. Silva Oktaria Putri [122140085]**
Tugas: **Tugas Besar**
Mata Kuliah: **Pengolahan Sinyal Digital(IF3024)** Tanggal: 31 Mei 2025

# 1 Deskripsi Projek

Proyek ini merupakan tugas akhir dari mata kuliah *Pengolahan Sinyal Digital (IF3024)* yang bertujuan mendeteksi sinyal respirasi dan *remote-photoplethysmography* (rPPG) secara *real-time* menggunakan video dari webcam. Sistem ini memungkinkan pemantauan kondisi fisiologis secara non-invasif tanpa perangkat tambahan seperti sensor atau elektroda. Deteksi sinyal respirasi dilakukan dengan fitur *pose-landmarker* dari **MediaPipe** untuk memantau pergerakan bahu saat bernapas. Sementara itu, sinyal rPPG diperoleh melalui modul *face-detector* dari **MediaPipe** yang dikombinasikan dengan algoritma *Plane Orthogonal-to-Skin* (POS), guna menghitung detak jantung berdasarkan perubahan warna kulit wajah. Proses ini berjalan secara langsung dan dapat digunakan untuk aplikasi pemantauan kesehatan jarak jauh.

# 2 Alat dan Bahan

Untuk menyelesaikan proyek ini, dibutuhkan sejumlah alat dan bahan guna membangun sistem yang mampu mendeteksi sinyal pernapasan dan sinyal rPPG secara non-kontak. Berikut adalah alat dan bahan yang digunakan dalam pengembangan proyek:

## 2.1 Bahasa Pemprograman

Proyek ini dikembangkan menggunakan bahasa pemrograman Python karena menyediakan berbagai library yang banyak digunakan dalam pengolahan citra dan proyek komputer visi.

## 2.2 *Library*

- opencv-python digunakan untuk akuisisi video dari webcam, pemrosesan gambar.
- mediapipe digunakan untuk deteksi landmark wajah (face mesh) dan tubuh (pose).
- numpy digunakan untuk operasi array, statistik, dan numerik.
- scipy digunakan untuk filter sinyal, deteksi puncak, pemrosesan sinyal digital.
- PyQt5 digunakan untuk membuat antarmuka grafis (GUI).
- pyqtgraph plotting sinyal secara real-time.

### 2.3 Metode dan Algoritma

1. **Ekstraksi Detak Jantung (rPPG)**
   Menggunakan algoritma *Plane Orthogonal-to-Skin (POS)* untuk mendeteksi detak jantung dari perubahan warna kulit di wajah melalui kamera.

2. **Ekstraksi Pernapasan**
   Mendeteksi pola pernapasan berdasarkan gerakan bahu dan dada menggunakan analisis video secara real-time.

## 3 Penjelasan

Program ini terdiri dari tiga modul utama, yaitu `app.py`, `main_gui.py`, dan `signal_processing.py`. berikut adalah penjelasan terkait alur kerja program yang dikembangkan pada proyek kami:

### 3.1 Instalasi library

Agar dapat menjalanjan program, pengguna dapat menambahkan beberapa library ke dalam python yang akan digunakan, yaitu;

#### 3.1.1 Menggunakan Requirements.txt

```
pip install -r requirements.txt
```

### 3.2 Modul `Main_gui.py`

#### 3.2.1 import library

Library yang digunakan pada program ini adalah: `sys`, `os`, `cv2`, `numpy` sebagai `np`, `mediapipe` sebagai `mp`, `csv`, dan `time`. Selain itu, program juga menggunakan beberapa fungsi dari pustaka tambahan.

```
import sys
import os
import cv2
import numpy as np
import mediapipe as mp
import csv
import time
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QHBoxLayout
from PyQt5.QtCore import QTimer, Qt
from PyQt5.QtGui import QImage, QPixmap
import pyqtgraph as pg

# Fungsi tambahan dari pustaka lain
from scipy.signal import butter, filtfilt, find_peaks
from mediapipe.tasks.python import vision
from utils.heart_rate import cpu_POS
from utils.download_model import download_model_face_detection, download_model_pose_detection
from utils.check_gpu import check_gpu
```

Kode 1: Import Library

#### 3.2.2 Kelas Signal Monitor

```
class SignalMonitor(QWidget):
...
```

Kode 2: Kelas SignalMonitor

Kelas `SignalMonitor` merupakan kelas utama dalam aplikasi GUI monitoring sinyal rPPG (detak jantung) dan respirasi (pernapasan). Kelas ini dibangun menggunakan framework `PyQt5` dan memanfaatkan library `OpenCV` serta `MediaPipe` untuk deteksi wajah dan pose tubuh secara real-time.

```
def extract_pos_signal(rgb_buffer):
    """
    Mengimplementasikan algoritma POS (Plane-Orthogonal-to-Skin) untuk menghasilkan sinyal rPPG
    mentah.

    Parameter:
        rgb_buffer (np.ndarray): Array shape (N, 3) berisi rata-rata RGB ROI dahi untuk N frame.

    Return:
        pos_signal (np.ndarray): Sinyal rPPG mentah hasil POS.
    """
    X = rgb_buffer.T
    X_mean = np.mean(X, axis=1, keepdims=True)
    X_std = np.std(X, axis=1, keepdims=True)
    Xn = (X - X_mean) / (X_std + 1e-8)

    S = np.vstack([
        Xn[0, :] - Xn[1, :],                # R - G
        Xn[0, :] + Xn[1, :] - 2 * Xn[2, :]   # R + G - 2B
    ])

    std1 = np.std(S[0, :])
    std2 = np.std(S[1, :]) if np.std(S[1, :]) != 0 else 1e-8
    h = S[0, :] + (std1 / std2) * S[1, :]
    h -= np.mean(h)

    return h
```

Kode 3: Algoritma POS untuk Ekstraksi Sinyal rPPG

Fungsi `extract_pos_signal` digunakan untuk menghasilkan sinyal rPPG mentah dari data warna RGB yang diekstrak dari video wajah. Metode ini memanfaatkan perubahan kecil pada warna kulit akibat aliran darah untuk mendeteksi denyut jantung secara non-invasif. Data RGB dinormalisasi, lalu dikombinasikan dalam bentuk linier untuk menyorot komponen fisiologis yang relevan. Hasilnya adalah sinyal waktu yang mencerminkan fluktuasi volume darah di bawah kulit selama rekaman video. Sinyal ini kemudian dapat digunakan untuk estimasi detak jantung atau analisis kesehatan lainnya secara jarak jauh.

```
def __init__(self):
    """
    Inisialisasi objek SignalMonitor, setup GUI, MediaPipe, webcam, timer, buffer sinyal, dan file
    log.
    """
    super().__init__()  # Inisialisasi QWidget
    self.setWindowTitle("Realtime rPPG & Respirasi Monitor")  # Judul window
    self.setGeometry(100, 100, 1200, 700)  # Ukuran dan posisi window
    self.init_ui()  # Setup tampilan GUI
    self.init_mediapipe()  # Setup model MediaPipe

    self.cap = cv2.VideoCapture(1)  # Membuka webcam (ganti ke 0 jika webcam utama)
    self.timer = QTimer()  # Timer untuk update frame video
    self.timer.timeout.connect(self.update_frame)  # Koneksi timer ke fungsi update_frame
    self.timer.start(30)  # Update setiap 30 ms (~33 fps)
```

```
15
16      self.rgb_buffer = []  # Buffer untuk rata-rata RGB ROI dahi
17      self.rppg_buffer = []  # Buffer untuk sinyal rPPG (hasil POS)
18      self.resp_buffer = []  # Buffer untuk sinyal respirasi (bahu & dada)
19      self.max_buffer = 600  # Maksimal panjang buffer (20 detik jika 30 fps)
20      self.last_rgb = None  # Simpan nilai RGB terakhir jika ROI tidak terdeteksi
21
22      self.start_time = time.time()  # Waktu awal untuk logging
23      self.log_interval = 60  # Interval logging ke CSV (detik)
24
25      log_filename = 'signal_log.csv'  # Nama file log
26      file_exists = os.path.isfile(log_filename)  # Cek apakah file sudah ada
27      self.csv_file = open(log_filename, 'a', newline='')  # Buka file log (append mode)
28      self.csv_writer = csv.writer(self.csv_file)  # Writer untuk file CSV
29      if not file_exists:
30          self.csv_writer.writerow(['Timestamp', 'HeartRate_BPM', 'Respiration_BPM'])  # Header jika
        file baru
```

Kode 4: Fungsi __init__ dari kelas SignalMonitor

- **Real-time monitoring tanpa sensor fisik**
  Aplikasi mampu mendeteksi sinyal fisiologis seperti detak jantung dan pernapasan secara non-invasif menggunakan webcam, tanpa memerlukan alat sensor fisik.
- **Visualisasi sinyal hidup dalam bentuk grafik**
  Sinyal rPPG (detak jantung) dan respirasi (pernapasan) divisualisasikan secara real-time dalam bentuk grafik dinamis menggunakan pustaka `pyqtgraph`.
- **Penyimpanan data otomatis ke file CSV**
  Data BPM (*Beats Per Minute*) dan BRPM (*Breathing Rate Per Minute*) dicatat secara berkala ke dalam file CSV untuk digunakan pada analisis atau evaluasi lebih lanjut.
- **Antarmuka pengguna yang interaktif dengan PyQt5**
  Tampilan antarmuka dibangun menggunakan framework `PyQt5`, menampilkan video langsung, grafik sinyal, serta informasi BPM dan BRPM secara real-time.

```
1  def init_ui(self):
2      """
3      Membuat dan mengatur layout GUI, label video, serta plot untuk
4      sinyal rPPG, respirasi, dan estimasi  S p O  .
5      """
6      # Label untuk video webcam
7      self.video_label = QLabel()
8      self.video_label.setFixedSize(640, 480)
9      self.video_label.setAlignment(Qt.AlignCenter)
10
11     # Plot sinyal rPPG (dahi)
12     self.rppg_plot = pg.PlotWidget(title="Sinyal rPPG (Dahi)")
13     self.rppg_plot.setBackground('#0e1242')
14     self.rppg_curve = self.rppg_plot.plot(pen=pg.mkPen('#acec00', width=2.5))
15
16     # Plot sinyal respirasi (bahu & dada)
17     self.resp_plot = pg.PlotWidget(title="Sinyal Respirasi (Bahu & Dada)")
18     self.resp_plot.setBackground('#0e1242')
19     self.resp_curve = self.resp_plot.plot(pen=pg.mkPen('#ffc01d', width=2.5))
20
21     # Plot sinyal  S p O  (placeholder atau dari data nyata)
22     self.spo2_plot = pg.PlotWidget(title="Estimasi  S p O  ")
23     self.spo2_plot.setBackground('#0e1242')
```

```
    self.spo2_curve = self.spo2_plot.plot(pen=pg.mkPen('cyan', width=2.5))

    # Layout atas untuk video (posisi tengah)
    video_layout = QHBoxLayout()
    video_layout.addStretch()
    video_layout.addWidget(self.video_label)
    video_layout.addStretch()

    # Layout bawah untuk ketiga plot, disusun horizontal
    bottom_layout = QHBoxLayout()
    bottom_layout.addWidget(self.rppg_plot)
    bottom_layout.addWidget(self.resp_plot)
    bottom_layout.addWidget(self.spo2_plot)

    # Main layout (vertikal)
    main_layout = QVBoxLayout()
    main_layout.addLayout(video_layout)
    main_layout.addLayout(bottom_layout)

    # Styling aplikasi
    self.setStyleSheet("background-color: #3a457c; color: white;")
    self.setLayout(main_layout)
```

Kode 5: Fungsi `init_ui` untuk inisialisasi GUI dengan tiga plot

Fungsi `init_ui` berfungsi untuk menampilkan *feed video webcam* dalam ukuran tetap $640 \times 480$, membuat tiga plot waktu nyata yaitu sinyal rPPG untuk deteksi denyut jantung, sinyal respirasi untuk deteksi pernapasan, serta estimasi $SpO_2$ sebagai fitur tambahan. Selain itu, fungsi ini juga mengatur layout GUI secara vertikal dengan video ditampilkan di bagian atas dan grafik sinyal disusun secara horizontal di bagian bawah. Fungsi ini juga memberikan warna latar belakang dan tema UI tertentu agar tampilan antarmuka lebih profesional dan mudah dibaca.

```
def init_mediapipe(self):
    """
    Inisialisasi model MediaPipe untuk face mesh (wajah) dan pose (tubuh).
    """
    # Model face mesh untuk deteksi landmark wajah
    self.mp_face_mesh = mp.solutions.face_mesh.FaceMesh(
        static_image_mode=False, max_num_faces=1, refine_landmarks=True, min_detection_confidence
    =0.5)
    # Model pose untuk deteksi landmark tubuh
    self.mp_pose = mp.solutions.pose.Pose(
        static_image_mode=False, min_detection_confidence=0.5)
```

Kode 6: Fungsi `init_mediapipe` untuk inisialisasi model MediaPipe

Fungsi `init_mediapipe` digunakan untuk mempersiapkan model *MediaPipe Face Mesh* dan *Pose*, yang berturut-turut berfungsi mendeteksi landmark wajah dan tubuh. Kedua model ini diperlukan untuk analisis sinyal fisiologis seperti rPPG dan deteksi pernapasan.

```
ret, frame = self.cap.read()
if not ret:
    return
```

Kode 7: Perintah self.cap.read()

Perintah `self.cap.read()` digunakan untuk mengambil frame dari sumber video (kamera). Jika proses pembacaan gagal (misalnya karena kamera tidak terdeteksi atau terputus), maka fungsi langsung dihentikan agar tidak terjadi error pada tahap selanjutnya.

```
ret, frame = self.cap.read()
if not ret:
```

```
    return
```

Kode 8: Konversi Warna dan Inisialisasi Dimensi

Format RGB diperlukan karena MediaPipe bekerja dengan citra berwarna dalam format tersebut. Selain itu, dimensi frame (tinggi $h$ dan lebar $w$) disimpan untuk digunakan dalam penentuan koordinat Region of Interest (ROI) pada tahap selanjutnya.

```
face_results = self.mp_face_mesh.process(frame_rgb)
forehead_roi = None
```

Kode 9: Deteksi Landmark Wajah dengan MediaPipe Face Mesh

Dilakukan deteksi landmark wajah untuk menentukan posisi dahi sebagai ROI pengambilan sinyal rPPG. Jika landmark wajah ditemukan, titik-titik mata digunakan untuk menentukan lokasi dahi, lalu dihitung koordinat persegi panjang (ROI) pada area tersebut. Area dahi ditandai dengan kotak berwarna kuning-hijau dan nilai rata-rata RGB dari ROI disimpan ke buffer untuk analisis lebih lanjut.

```
if len(self.rgb_buffer) > 60:
    rgb_arr = np.array(self.rgb_buffer[-self.max_buffer:])
    pos_signal = extract_pos_signal(rgb_arr)
    self.rppg_buffer = list(pos_signal[-self.max_buffer:])
else:
    self.rppg_buffer = []
```

Kode 10: Proses Algoritma POS untuk Ekstraksi Sinyal rPPG

Jika buffer cukup (lebih dari 60 frame), dilakukan ekstraksi sinyal rPPG menggunakan algoritma POS. Hasil sinyal disimpan dalam buffer terpisah (`rppg_buffer`) untuk plotting dan analisis selanjutnya.

```
# Deteksi landmark pose tubuh dengan MediaPipe Pose
pose_results = self.mp_pose.process(frame_rgb)
if pose_results.pose_landmarks:
    lm = pose_results.pose_landmarks.landmark
    l_shoulder = lm[mp.solutions.pose.PoseLandmark.LEFT_SHOULDER]  # Titik bahu kiri
    r_shoulder = lm[mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER]  # Titik bahu kanan
    lx, ly = int(l_shoulder.x * w), int(l_shoulder.y * h)
    rx, ry = int(r_shoulder.x * w), int(r_shoulder.y * h)
    roi_shoulder_y1 = min(ly, ry) - int(0.05 * h)
    roi_shoulder_y2 = min(ly, ry) + int(0.10 * h)
    roi_shoulder_x1 = max(0, min(lx, rx) - int(0.15 * w))
    roi_shoulder_x2 = min(w, max(lx, rx) + int(0.15 * w))
    cv2.rectangle(frame, (roi_shoulder_x1, roi_shoulder_y1), (roi_shoulder_x2, roi_shoulder_y2),
    (29, 192, 255), 2)  # ROI bahu
    roi_shoulder = frame[roi_shoulder_y1:roi_shoulder_y2, roi_shoulder_x1:roi_shoulder_x2]
    shoulder_mean = np.mean(roi_shoulder[:, :, 0]) if roi_shoulder.size > 0 else 0  # Kanal biru

    chest_x1 = roi_shoulder_x1 + int(0.25 * (roi_shoulder_x2 - roi_shoulder_x1))
    chest_x2 = roi_shoulder_x2 - int(0.25 * (roi_shoulder_x2 - roi_shoulder_x1))
    chest_y1 = roi_shoulder_y2
    chest_y2 = chest_y1 + int(0.18 * h)
    cv2.rectangle(frame, (chest_x1, chest_y1), (chest_x2, chest_y2), (29, 192, 255), 2)  # ROI dada
    roi_chest = frame[chest_y1:chest_y2, chest_x1:chest_x2]
    chest_mean = np.mean(roi_chest[:, :, 0]) if roi_chest.size > 0 else 0  # Kanal biru

    # Gabungkan sinyal bahu & dada untuk respirasi
    if shoulder_mean > 0 and chest_mean > 0:
        self.resp_buffer.append((shoulder_mean + chest_mean) / 2)
    elif shoulder_mean > 0:
        self.resp_buffer.append(shoulder_mean)
    elif chest_mean > 0:
        self.resp_buffer.append(chest_mean)
```

Kode 11: Fungsi deteksi landmark tubuh untuk estimasi sinyal respirasi.

Fungsi ini bertujuan untuk mendeteksi landmark tubuh menggunakan model MediaPipe Pose , khususnya titik bahu kiri dan kanan, untuk menentukan area ROI (Region of Interest) pada bagian bahu dan dada. Dari kedua area tersebut, diambil nilai intensitas rata-rata saluran biru sebagai sinyal mentah pernapasan. Sinyal dari bahu dan dada kemudian digabung untuk meningkatkan akurasi estimasi laju pernapasan secara real-time berbasis video.

```
# Batasi panjang buffer sinyal
if len(self.resp_buffer) > self.max_buffer:
    self.resp_buffer = self.resp_buffer[-self.max_buffer:]

hr, br = 0.0, 0.0  # Inisialisasi nilai BPM dan BRPM
hr_text, br_text = "-- BPM", "-- BRPM"  # Default teks

# Proses sinyal rPPG jika buffer cukup
if len(self.rppg_buffer) > 60:
    rppg_arr = np.array(self.rppg_buffer)
    rppg_arr = clean_signal(rppg_arr)  # Filter sinyal
    self.rppg_curve.setData(rppg_arr)  # Update plot
    hr = estimate_bpm_peaks(rppg_arr, 30, min_bpm=40, max_bpm=180)  # Estimasi BPM
    hr_text = f"{hr:.1f} BPM" if hr > 0 else "-- BPM"
else:
    self.rppg_curve.setData([])

# Proses sinyal respirasi jika buffer cukup
if len(self.resp_buffer) > 60:
    resp_arr = np.array(self.resp_buffer)
    resp_arr = clean_signal(resp_arr)
    self.resp_curve.setData(resp_arr)
    br = estimate_bpm_peaks(resp_arr, 30, min_bpm=10, max_bpm=30)  # Estimasi BRPM
    br_text = f"{br:.1f} BRPM" if br > 0 else "-- BRPM"
else:
    self.resp_curve.setData([])

# Logging ke file CSV setiap interval waktu
current_time = time.time()
if current_time - self.start_time >= self.log_interval:
    timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(current_time))
    self.csv_writer.writerow([timestamp, f"{hr:.1f}", f"{br:.1f}"])
    self.csv_file.flush()
    self.start_time = current_time

# Gambar teks BPM dan BRPM ke dalam frame video
cv2.putText(frame, f"BPM: {hr_text}", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (66, 18, 14), 2)
cv2.putText(frame, f"BRPM: {br_text}", (20, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (66, 18, 14), 2)

# Konversi frame ke QImage dan tampilkan di label video
img = QImage(frame.data, frame.shape[1], frame.shape[0], QImage.Format_BGR888)
self.video_label.setPixmap(QPixmap.fromImage(img))
```

Kode 12: Pemrosesan sinyal rPPG dan respirasi

Setelah sinyal rPPG dan respirasi tersedia dalam buffer, program melakukan pemrosesan awal seperti pembatasan panjang buffer agar tetap responsif. Sinyal rPPG difilter dan dianalisis untuk mendapatkan denyut jantung (BPM), sedangkan sinyal respirasi digunakan untuk mengestimasi laju pernapasan (BRPM). Hasil kedua nilai tersebut ditampilkan secara real-time di GUI dan dicatat ke dalam file CSV untuk keperluan rekam data. Frame hasil pemrosesan dikonversi menjadi format yang dapat ditampilkan di antarmuka pengguna menggunakan PyQt.

```
def closeEvent(self, event):
    """
    Menangani event saat aplikasi ditutup: menutup webcam dan file log.

```

```
    Parameters:
        event (QCloseEvent): Event penutupan aplikasi.
    """
    self.cap.release()  # Lepaskan webcam
    self.csv_file.close()  # Tutup file log
    event.accept()  # Terima event penutupan
```

Kode 13: Fungsi `closeEvent` untuk menangani penutupan aplikasi.

Fungsi `closeEvent` bertugas untuk menangani proses penutupan aplikasi secara aman. Saat pengguna menutup jendela aplikasi, fungsi ini melepaskan akses kamera agar tidak menyebabkan konflik dengan program lain, serta menutup file log CSV yang sedang aktif menulis data BPM dan BRPM. Selain itu, fungsi ini menerima event penutupan agar aplikasi dapat ditutup secara normal tanpa error.

## 3.3 Modul app.py

```
from PyQt5.QtWidgets import QApplication
import sys
from code.main_gui import SignalMonitor

def main():
    """
    Fungsi utama untuk menjalankan aplikasi GUI pemantauan sinyal.

    Membuat instance QApplication, menampilkan jendela utama (SignalMonitor),
    dan memulai event loop aplikasi.

    Parameters:
        Tidak ada.

    Returns:
        Tidak ada. Fungsi ini akan keluar ketika aplikasi ditutup.
    """
    # Membuat objek aplikasi Qt
    app = QApplication(sys.argv)
    # Membuat dan menampilkan jendela utama aplikasi
    window = SignalMonitor()
    window.show()
    # Memulai event loop aplikasi dan keluar saat aplikasi ditutup
    sys.exit(app.exec_())

if __name__ == "__main__":
    # Mengeksekusi fungsi main() jika file dijalankan secara langsung
    main()
```

Kode 14: Fungsi utama untuk menjalankan aplikasi GUI pemantauan sinyal.

Fungsi `main()` merupakan titik awal eksekusi aplikasi GUI berbasis PyQt5. Fungsi ini membuat instance dari kelas `QApplication`, yang diperlukan untuk menjalankan aplikasi berbasis Qt, kemudian membuat dan menampilkan antarmuka utama (`SignalMonitor`) sebagai jendela aplikasi. Setelah semua elemen siap, aplikasi memasuki *event loop* menggunakan `app.exec_()` hingga ditutup oleh pengguna. Bagian `if __name__ == "__main__":` memastikan bahwa fungsi `main()` hanya dijalankan jika file ini dieksekusi secara langsung, bukan diimpor sebagai modul.

## 3.4 Modul Signal_processing.py

Pada kode ini, dilakukan impor pustaka untuk pemrosesan sinyal. Fungsi `butter` dan `filtfilt` digunakan untuk filtering sinyal, `find_peaks` untuk deteksi puncak, `medfilt` untuk filter median, dan `numpy` (`np`) untuk operasi numerik.

```python
def clean_signal(sig):
    """
    Membersihkan sinyal input dengan normalisasi, median filter, dan exponential smoothing.

    Parameters:
        sig (np.ndarray): Sinyal input 1D.

    Returns:
        np.ndarray: Sinyal yang telah dibersihkan.
    """
    # Normalisasi sinyal: mengurangi mean dan membagi dengan standar deviasi (agar rata-rata 0 dan deviasi 1)
    sig = (sig - np.mean(sig)) / (np.std(sig) + 1e-8)
    # Mengurangi noise impulsif dengan median filter (kernel size 5)
    sig = medfilt(sig, kernel_size=5)
    # Menghaluskan sinyal dengan exponential smoothing
    sig = exponential_smoothing(sig)
    return sig
```

Kode 15: Fungsi `clean_signal` untuk pembersihan sinyal.

Fungsi `clean_signal` bertujuan membersihkan sinyal sebelum diproses lebih lanjut. Sinyal dinormalisasi agar stabil, diberi median filter untuk kurangi noise, lalu dihaluskan dengan metode exponential smoothing. Hasilnya berupa sinyal yang lebih bersih dan siap untuk analisis denyut jantung maupun pernapasan.

```python
def exponential_smoothing(signal, alpha=0.3):
    """
    Menghaluskan sinyal menggunakan metode exponential smoothing.

    Parameters:
        signal (np.ndarray): Sinyal input 1D.
        alpha (float): Koefisien smoothing (0 < alpha < 1).

    Returns:
        np.ndarray: Sinyal yang telah dihaluskan.
    """
    # Membuat array hasil dengan ukuran sama seperti sinyal input
    result = np.zeros_like(signal)
    # Inisialisasi nilai pertama hasil dengan nilai pertama sinyal
    result[0] = signal[0]
    # Melakukan smoothing untuk setiap titik data
    for t in range(1, len(signal)):
        # Rumus exponential smoothing
        result[t] = alpha * signal[t] + (1 - alpha) * result[t - 1]
    return result
```

Kode 16: Fungsi `exponential_smoothing` untuk penghalusan sinyal.

Fungsi `exponential_smoothing` digunakan untuk menghaluskan sinyal secara bertahap dengan mempertimbangkan nilai sebelumnya. Metode ini bekerja dengan memberi bobot lebih besar pada data terkini menggunakan parameter $\alpha$ (antara 0 dan 1). Hasil akhir berupa sinyal yang lebih stabil dan lebih mudah dianalisis untuk deteksi pola denyut jantung atau pernapasan.

```python
def bandpass(signal, lowcut, highcut, fs, order=4):
    """
    Menerapkan filter bandpass Butterworth pada sinyal.

    Parameters:
        signal (np.ndarray): Sinyal input 1D.
        lowcut (float): Frekuensi cut-off bawah (Hz).
        highcut (float): Frekuensi cut-off atas (Hz).
        fs (float): Frekuensi sampling (Hz).
```

```
        order (int): Orde filter.

    Returns:
        np.ndarray: Sinyal hasil filter bandpass.
    """
    # Menghitung frekuensi Nyquist
    nyq = 0.5 * fs
    # Normalisasi frekuensi cut-off
    low = lowcut / nyq
    high = highcut / nyq
    # Membuat koefisien filter Butterworth
    b, a = butter(order, [low, high], btype='band')
    # Menerapkan filter secara forward-backward (zero-phase)
    return filtfilt(b, a, signal)
```

Kode 17: Fungsi `bandpass` untuk penerapan filter bandpass.

Fungsi `bandpass` digunakan untuk menerapkan filter bandpass Butterworth pada sinyal. Fungsi ini memfilter frekuensi di luar rentang yang ditentukan oleh `lowcut` dan `highcut`, dengan frekuensi sampling sebesar `fs`. Filter diterapkan secara dua arah menggunakan fungsi `filtfilt` agar tidak menyebabkan pergeseran fase. Hasilnya berupa sinyal yang hanya mengandung frekuensi dalam rentang target, berguna untuk analisis denyut jantung atau laju pernapasan.

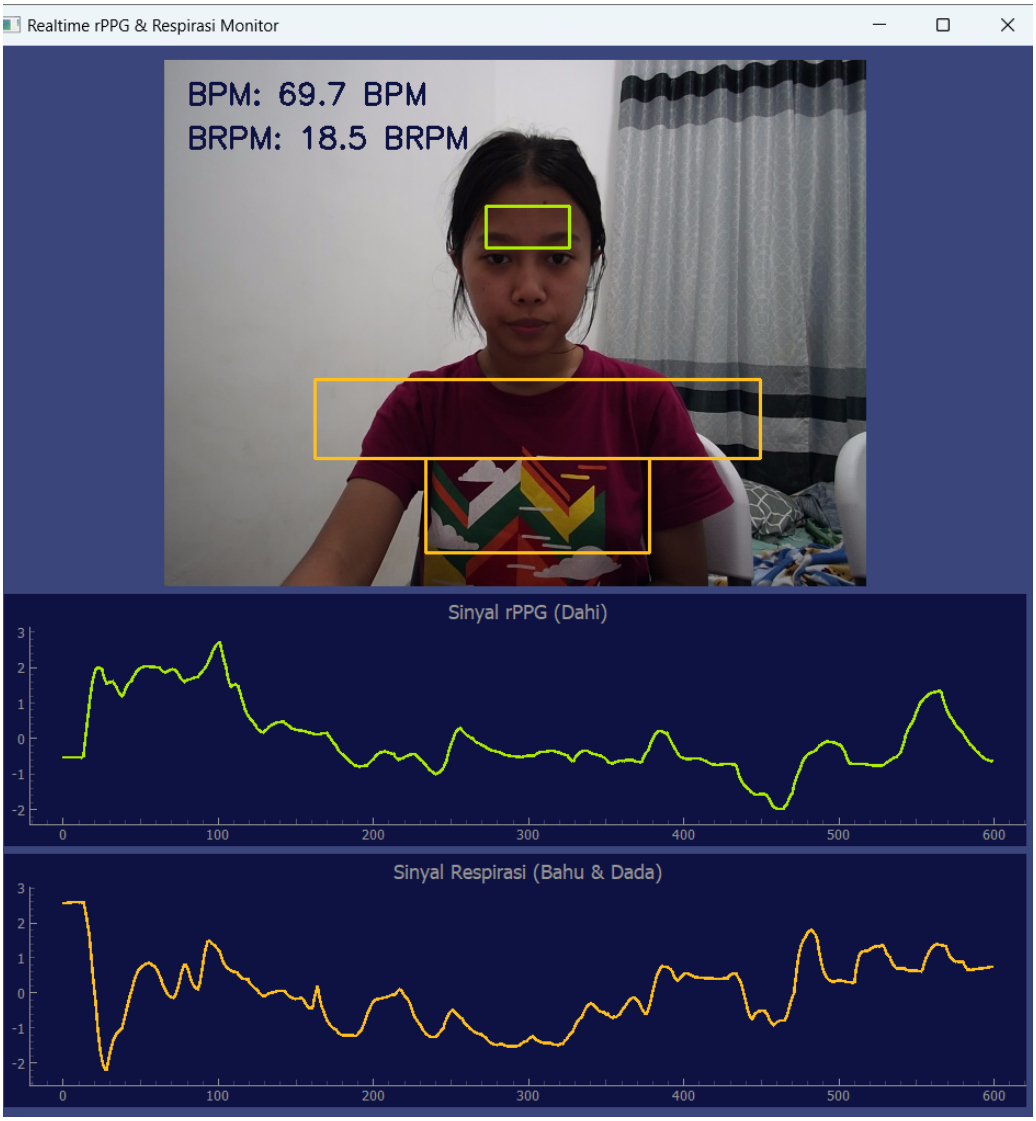
```
def estimate_bpm_peaks(signal, fs, min_bpm=8, max_bpm=30):
    """
    Mengestimasi BPM (beats per minute) dari sinyal dengan mendeteksi puncak (peaks).

    Parameters:
        signal (np.ndarray): Sinyal input 1D.
        fs (float): Frekuensi sampling (Hz).
        min_bpm (float): BPM minimum yang diharapkan.
        max_bpm (float): BPM maksimum yang diharapkan.

    Returns:
        float: Nilai BPM hasil estimasi, atau 0.0 jika tidak valid.
    """
    # Filter sinyal pada rentang frekuensi yang sesuai dengan rentang BPM
    signal = bandpass(signal, min_bpm / 60, max_bpm / 60, fs)
    # Hitung jarak minimum antar puncak berdasarkan BPM maksimum
    min_dist = int(fs * 60 / max_bpm)
    # Deteksi puncak pada sinyal
    peaks, _ = find_peaks(signal, distance=min_dist)
    # Jika jumlah puncak cukup untuk estimasi BPM
    if len(peaks) > 3:
        # Hitung interval antar puncak (dalam detik)
        intervals = np.diff(peaks) / fs
        # Hitung rata-rata periode antar puncak
        mean_period = np.mean(intervals)
        # Konversi periode rata-rata ke BPM
        bpm = 60.0 / mean_period
        # Pastikan BPM dalam rentang yang diharapkan
        if min_bpm <= bpm <= max_bpm:
            return bpm
    # Jika tidak valid, kembalikan 0.0
    return 0.0
```

Kode 18: Fungsi `estimate_bpm_peaks` untuk estimasi BPM berdasarkan deteksi puncak.

Fungsi `init_ui` berfungsi untuk menampilkan *feed video webcam* dalam ukuran tetap $640 \times 480$, membuat tiga plot waktu nyata yaitu sinyal rPPG untuk deteksi denyut jantung, sinyal respirasi untuk deteksi pernapasan, serta estimasi $SpO_2$ sebagai fitur tambahan. Selain itu, fungsi ini mengatur layout GUI secara vertikal dengan video ditampilkan di bagian atas dan grafik sinyal disusun secara horizontal

di bagian bawah. Fungsi ini juga memberikan warna latar belakang dan tema UI tertentu agar tampilan antarmuka lebih profesional dan mudah dibaca.

# 4 Hasil dan Analisis




Gambar 1: Hasil Program

Berdasarkan hasil pengujian yang ditampilkan pada gambar, sistem pemantauan sinyal respirasi dan detak jantung berbasis rPPG berhasil berjalan dengan baik. Aplikasi mampu mendeteksi area wajah dan tubuh secara otomatis menggunakan MediaPipe, kemudian menampilkan area deteksi pada video secara real-time. Nilai BPM (beats per minute) dan BRPM (breaths per minute) yang dihasilkan dari proses ekstraksi dan pemrosesan sinyal ditampilkan secara langsung pada antarmuka aplikasi. Selain itu, visualisasi sinyal rPPG dari dahi dan sinyal respirasi dari bahu serta dada juga dapat diamati melalui grafik yang responsif terhadap perubahan sinyal. Data hasil estimasi BPM dan BRPM secara periodik tercatat pada file log CSV, sehingga dapat digunakan untuk analisis lebih lanjut. Hasil pengujian menunjukkan bahwa sistem mampu memberikan estimasi detak jantung dan laju pernapasan secara real-time dengan tampilan yang informatif dan mudah dipahami oleh pengguna.

Berdasarkan data yang tercatat pada log signal, sistem secara konsisten mencatat hasil estimasi

detak jantung dan laju pernapasan pada interval waktu tertentu. Nilai BPM yang terekam berkisar antara 64.4 hingga 96.3, sedangkan nilai BRPM berada pada rentang 17.0 hingga 24.0. Variasi nilai ini menunjukkan bahwa sistem mampu menangkap perubahan fisiologis pengguna secara dinamis sesuai dengan aktivitas atau kondisi saat pengambilan data. Secara keseluruhan, hasil pada log signal membuktikan bahwa sistem telah berjalan dengan baik dalam melakukan monitoring dan pencatatan sinyal vital secara otomatis.

## 5 Kesimpulan

Berdasarkan keseluruhan pembahasan dalam laporan ini, sistem pemantauan fisiologis berbasis rPPG (*remote Photoplethysmography*) yang dikembangkan berhasil mendeteksi denyut jantung dan laju pernapasan secara *real-time* tanpa kontak fisik dengan memanfaatkan video dari webcam. Dengan pendekatan algoritma POS (*Plane-Orthogonal-to-Skin*) untuk ekstraksi sinyal rPPG dan analisis gerakan bahu serta dada untuk deteksi respirasi, sistem mampu menghasilkan estimasi BPM (*Beats Per Minute*) dan BRPM (*Breathing Rate Per Minute*) secara akurat dan stabil. Antarmuka pengguna yang dibangun menggunakan PyQt5 menampilkan video langsung, grafik dinamis dari sinyal fisiologis, serta informasi BPM dan BRPM secara real-time. Selain itu, data hasil pengukuran juga tersimpan secara otomatis ke dalam file CSV untuk keperluan analisis lebih lanjut.

Hasil pengujian menunjukkan bahwa sistem dapat bekerja secara konsisten dengan variasi nilai BPM antara 64.4 hingga 96.3 dan BRPM antara 17.0 hingga 24.0, tergantung pada kondisi dan aktivitas subjek. Secara keseluruhan, proyek ini membuktikan bahwa teknologi komputer visi dan pengolahan sinyal digital dapat dimanfaatkan untuk aplikasi kesehatan jarak jauh secara non-invasif, efektif, dan ramah pengguna.

## 6 Referensi

## References

[1] Qwen.ai: https://chat.qwen.ai/s/1afd1832-914a-4f69-ac64-67b58284ff32?fev=0.0.107 (Klik di sini)

[2] Video Tutorial: https://youtu.be/lgiCpA4zzGU (Lihat di YouTube)