Bilag 5 - PC-Koden

Main

```
import tkinter as tk
from tkinter import messagebox, ttk
import sqlite3
import threading
import serial
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg \
import FigureCanvasTkAgg

#Der oprettes en sensoer klasse, med henholdvis af tråde
class Sensor(threading.Thread):
 def __init__(self, app):
     super().__init__()
     self.serial_port = serial.Serial('COM5', 38400, timeout=1)
     self.ekg_data = []
     self.queue1 = []
     self.queue2 = []
     self.app = app

##Der bliver defineret en metode "run" i en klasse,heri læser metoden en
vedvarende data fra en serielport
# herefter bliver der plottet tal ind i en liste kaldet "ekg_data"
# der bliver blandt andet også tilføjet tal til to køer "quee1" og "quee2"
og der blider opdateret en plot og herefter bliver der gemt EKG dataen i en
database

 def run(self):
     while True:
         data = self.serial_port.readline().strip().decode()
         if data:
             value = int(data)
             self.ekg_data.append(value)
             if len(self.ekg_data) > 250:
                 self.ekg_data.pop(0)
                 self.queue1.append(value)
                 self.queue2.append(value)
                 self.app.update_plot()
                 self.save_to_database()

#Der bliver her defineret en metode "save_to_database" i en klasse. Her
skal metoden sikrer at der bliver gemt EKG-data i en SQLite database

 def save_to_database(self):
     conn = sqlite3.connect('ekg_tal.db')
     c = conn.cursor()
     c.execute("CREATE TABLE IF NOT EXISTS ekg_data (value INTEGER)")
     c.execute("DELETE FROM ekg_data")
     c.executemany("INSERT INTO ekg_data VALUES (?)", [(value,) for value in
self.ekg_data])
```

```
        conn.commit()
        conn.close()

#Der bliver defineret en klasse "loginPage", hvor der her er en underklasse
af tk.frame. klassen er en log in side, som indeholder GUI-elementer.
class LoginPage(tk.Frame):
 def __init__(self, parent, on_login):
        super().__init__(parent)
        self.parent = parent
        self.on_login = on_login

        self.username_label = tk.Label(self, text="Brugernavn:", font=("Times
new roman", 15))
        self.username_label.pack(pady=5)
        self.username_entry = tk.Entry(self, fg="black")
        self.username_entry.pack(pady=10)

        self.password_label = tk.Label(self, text="Adgangskode:", font=("Times
new roman", 15))
        self.password_label.pack(pady=5)
        self.password_entry = tk.Entry(self, fg="black", show="*")
        self.password_entry.pack(pady=10)

        self.login_button = tk.Button(self, text="Login", bg="light blue",
command=self.login)
        self.login_button.pack(pady=5)

#Der bliver her defineret en metode "login" i en klasse.Der bliver her
godkendt brugernavn og adgangskode, som brugeren selv har indtastet på
login-side
 def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        if username == "AIRS" and password == "1234":
            self.on_login()
        else:
            messagebox.showerror("Fejl", "Ugyldigt Brugernavn eller
Adgangskode.")

class HomePage(tk.Frame):
 def __init__(self, parent, cursor, sensor):
        super().__init__(parent)
        self.parent = parent
        self.cursor = cursor
        self.sensor = sensor

        self.home_button = tk.Button(self, text="Patient data", bg="light blue",
command=self.show_patient_data)
        self.home_button.pack(pady=10)

        self.data_button = tk.Button(self, text="EKGData", bg="light blue",
command=self.show_ekg_data)
        self.data_button.pack(pady=10)
```

```
    def show_patient_data(self):

        patient_data_window = tk.Toplevel(self.parent)
        patient_data_window.title("Patientdata")
        patient_data_window.geometry("400x300")

        tree_view = ttk.Treeview(patient_data_window)
        tree_view.pack(fill="both", expand=True)

        tree_view["columns"] = ("CPR", "Navn", "Køn", "Diagnose")
        tree_view.column("#0", width=0, stretch=tk.NO)
        tree_view.column("CPR", width=100)
        tree_view.column("Navn", width=100)
        tree_view.column("Køn", width=100)
        tree_view.column("Diagnose", width=100)

        tree_view.heading("CPR", text="CPR")
        tree_view.heading("Navn", text="Navn")
        tree_view.heading("Køn", text="Køn")
        tree_view.heading("Diagnose", text="Diagnose")

        self.cursor.execute("SELECT * FROM patients")
        rows = self.cursor.fetchall()

        for row in rows:
            tree_view.insert("", tk.END, text="", values=row)

    def show_ekg_data(self):

        ekg_data_window = tk.Toplevel(self.parent)
        ekg_data_window.title("EKG Data")
        ekg_data_window.geometry("600x400")

        fig, ax = plt.subplots()
        canvas = FigureCanvasTkAgg(fig, master=ekg_data_window)
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

        ax.set_xlabel("Tid")
        ax.set_ylabel("EKG-værdi")

        # Start sensoren for at opdatere og vise EKG-data
        self.sensor.start()

        def update_plot():
            ax.clear()
            ax.plot(self.sensor.queue2)
            canvas.draw()
            ekg_data_window.after(100, update_plot)

        update_plot()

class MainApplication(tk.Tk):
```

```
 def __init__(self):
     super().__init__()
     self.title("Patient-Portalen.dk")
     self.geometry("500x500")

     self.connection = sqlite3.connect("patients.db")
     self.cursor = self.connection.cursor()

     self.create_table()

     self.sensor = Sensor(self)
     self.login_page = LoginPage(self, self.show_home_page)
     self.home_page = HomePage(self, self.cursor, self.sensor)

     self.show_login_page()

 def create_table(self):

     create_table_query = """
         CREATE TABLE IF NOT EXISTS patients (
             cpr TEXT PRIMARY KEY,
             navn TEXT,
             køn TEXT,
             diagnose TEXT
         )
     """
     self.cursor.execute(create_table_query)

 def indtast_patient(self):
     cpr = input("Indtast CPR: ")
     navn = input("Indtast navn: ")
     køn = input("Indtast køn: ")
     diagnose = input("Indtast diagnose: ")

     insert_query = """
         INSERT INTO patients (cpr, navn, køn, diagnose)
         VALUES (?, ?, ?, ?)
     """
     data = (cpr, navn, køn, diagnose)
     self.cursor.execute(insert_query, data)
     self.connection.commit()

 def show_login_page(self):
     self.home_page.pack_forget()
     self.login_page.pack()

 def show_home_page(self):
     self.login_page.pack_forget()
     self.home_page.pack()

if _name_ == "_main_":
 app = MainApplication()
 app.mainloop()
```

Graf

```
import threading
import serial
import sqlite3
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk

class Sensor(threading.Thread):
  def __init__(self, app):
      super().__init__()
      self.serial_port = serial.Serial('COM5', 38400, timeout=1)
      self.ekg_data = []
      self.queue1 = []
      self.queue2 = []
      self.app = app

  def run(self):
      while True:
          data = self.serial_port.readline().strip().decode()
          if data:
              value = int(data)
              self.ekg_data.append(value)
              if len(self.ekg_data) > 250:
                  self.ekg_data.pop(0)
                  self.queue1.append(value)
                  self.queue2.append(value)
                  self.app.update_plot()
                  self.save_to_database()

  def save_to_database(self):
      conn = sqlite3.connect('ekg_tal.db')
      c = conn.cursor()
      c.execute("CREATE TABLE IF NOT EXISTS ekg_data (value INTEGER)")
      c.execute("DELETE FROM ekg_data")
      c.executemany("INSERT INTO ekg_data VALUES (?)", [(value,) for value in
self.ekg_data])
      conn.commit()
      conn.close()

class Application(tk.Tk):
  def __init__(self):
      super().__init__()
      self.title("EKG Plot")
      self.fig, self.ax = plt.subplots()
      self.canvas = FigureCanvasTkAgg(self.fig, master=self)
      self.canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
      self.sensor = Sensor(self)

  def start_sensor(self):
      self.sensor.start()
```

```
  def update_plot(self):
      self.ax.clear()
      self.ax.plot(self.sensor.queue2)
      self.canvas.draw()

if name == 'main':
  sensor_app = Application()
  sensor_app.start_sensor()
  sensor_app.after(100, sensor_app.update_plot)
  sensor_app.mainloop()
```

SQL

```python
import sqlite3

# Opret forbindelse til databasen
conn = sqlite3.connect('patients.db')
cursor = conn.cursor()

# Opret tabel, hvis den ikke allerede findes
create_table_query = """
 CREATE TABLE IF NOT EXISTS patients (
     cpr TEXT PRIMARY KEY,
     navn TEXT,
     køn TEXT,
     diagnose TEXT
 )
"""
cursor.execute(create_table_query)


# Funktion til at indtaste patientoplysninger
def indtast_patient():
 cpr = input("Indtast CPR: ")
 navn = input("Indtast navn: ")
 køn = input("Indtast køn: ")
 diagnose = input("Indtast diagnose: ")

 # Indsæt data i tabellen
 insert_query = """
     INSERT INTO patients (cpr, navn, køn, diagnose)
     VALUES (?, ?, ?, ?)
 """
 data = (cpr, navn, køn, diagnose)
 cursor.execute(insert_query, data)
 conn.commit()  # Udfør commit-operationen for hver patient


# Antal patienter, du ønsker at tilføje
antal_patienter = int(input("Indtast antal patienter: "))

# Indtast oplysninger for hver patient
for _ in range(antal_patienter):
 indtast_patient()

# Luk forbindelsen til databasen
cursor.close()
conn.close()
```