UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria
Informatica e dell'Automazione

SVILUPPO DI UN SISTEMA PER LA REGISTRAZIONE E CATALOGAZIONE DI SEGNALI INERZIALI ACQUISITI DA SENSORI BLUETOOTH

DEVELOPMENT OF A SYSTEM FOR RECORDING AND CATALOGING INERTIAL SIGNALS ACQUIRED BY BLUETOOTH SENSORS

Relatore: Chiar.mo
Prof. Crippa Paolo

Candidato:
Prifti Eris

A.A. 2019/2020

1. INTRODUZIONE

Il Bluetooth è uno standard tecnico che viene usato per la trasmissione di dati tramite reti personali senza fili. In altre parole, due device dotati entrambi di Bluetooth possono scambiarsi file e informazioni senza bisogno che ci sia un collegamento fisico tra i due.

Il Bluetooth permette a due dispositivi di comunicare tramite una frequenza radio sicura e a corto raggio. Solitamente il raggio di azione è di qualche decina di metri (ma esistono anche Bluetooth con un raggio molto più ampio). Una volta attivato il Bluetooth è in grado di rilevare un altro dispositivo che si trovi nel raggio della frequenza e che supporti a sua volta questo tipo di tecnologia.

In questo caso specifico per la ricezione dei dati si andrà ad usare un dispositivo Bluetooth Low-Energy che viene utilizzato principalmente per:

- Il trasferimento di piccole quantità di dati tra dispositivi vicini.
- L'interazione con sensori di prossimità come Google Beacon per offrire agli utenti un'esperienza personalizzata in base alla loro posizione corrente

Da questo dispositivo Bluetooth si riceveranno dei dati che attraverso dei programmi verranno convertiti in file in modo da poterli salvare e successivamente catalogare.

Per fare in modo di catalogare i dati in modo informatico si utilizzano due strumenti principali, il primo è un server, il quale sarà Postgresql mentre il secondo è Python, un linguaggio di programmazione.

2. BLUETOOTH LOW-ENERGY

Commercializzato anche come Bluetooth Smart, il Bluetooth Low Energy (o BLE) è stato introdotto nelle specifiche Bluetooth 4.0 come alternativa al Bluetooth Classico.

Come il suo predecessore, la tecnologia BLE utilizza una tecnologia wireless basata su una frequenza radio, nella banda libera dei 2,4 GHz al fine di connettere tra loro dispositivi vicini.

La principale differenza con il Bluetooth Classico, come si può dedurre facilmente dal nome, è il ridotto consumo energetico. Infatti, per il BLE, il bit rate è di 1 Mbit/s (con un'opzione di 2 Mbit/s nel Bluetooth 5) e la potenza di trasmissione massima è di 10 mW (100 mW nel Bluetooth 5). Questo significa che la potenza utilizzata è meno della metà rispetto a prima.

Il Bluetooth Low Energy è stato presentato per la prima volta nel 2004, a seguito di un progetto di ricerca di Nokia. Il primo smartphone dotato di Bluetooth 4.0 è stato l'iPhone 4S, nell'ottobre 2011, a cui ne sono seguiti molti altri. Oggi, praticamente tutti i nuovi smartphone sono dotati di Bluetooth 4.0 o superiori.

Il BLE è una delle principali tecnologie che rendono possibile l'Internet of Things (IoT). Ad esempio, moltissimi dispositivi connessi a Internet utilizzati per l'assistenza sanitaria personale, il fitness, lo sport, l'intrattenimento e la localizzazione ora utilizzano il Bluetooth LE per comunicare con smartphone e tablet, inclusi iPhone, telefoni Android, Windows e BlackBerry. Si stima che nel 2018 i dispositivi dotati di Bluetooth abbiano raggiunto la quota di 4 miliardi.

La durata della batteria di questi dispositivi dipende da vari fattori: dall'hardware, dalla distanza di trasmissione e dal ciclo di lavoro. In generale, la durata è stimata in un range da 1 a 40 mesi. Ad esempio, un iBeacon – un dispositivo che emette impulsi Bluetooth periodici per consentire la localizzazione – potrebbe facilmente funzionare per un paio di anni prima che la batteria si scarichi.

Il BLE è interessante per l'elettronica di consumo e per i produttori di macchine connesse a Internet a causa del basso costo, della lunga durata della batteria e della facilità di implementazione. Dai termometri ai cardiofrequenzimetri, dagli smartwatch ai sensori di prossimità, il Bluetooth Low Energy facilita la trasmissione di dati wireless a corto raggio tra dispositivi, alimentata da una semplice una batteria per orologi.

3. DISPOSITIVO BLUETOOTH

Per lo svolgimento del progetto si va ad utilizzare un sensore wireless per l'acquisizione in tempo reale di segnali bioelettrici come l'elettromiografia (EMG) e l'elettrocardiografia (ECG), accoppiato con un sensore inerziale, per fornire un flusso completo di dati adatto per il rilevamento dell'attività umana, l'analisi del movimento e la tecnologia di assistenza infermieristica di persone con disabilità fisiche o cognitive.

Il sensore è in grado di acquisire fino a tre canali bioelettrici indipendenti (sei elettrodi), ciascuno con 24 bit di risoluzione e una frequenza di campionamento fino a 3,2 kHz, e dispone di una piattaforma inerziale 6-DoF che misura l'accelerazione lineare e la velocità angolare. Il collegamento wireless Bluetooth a basso consumo è stato scelto perché consente un facile interfacciamento con molti dispositivi di elettronica di consumo, come smartphone o tablet.

3.1 DESCRIZIONE DEL SISTEMA E HARDWARE

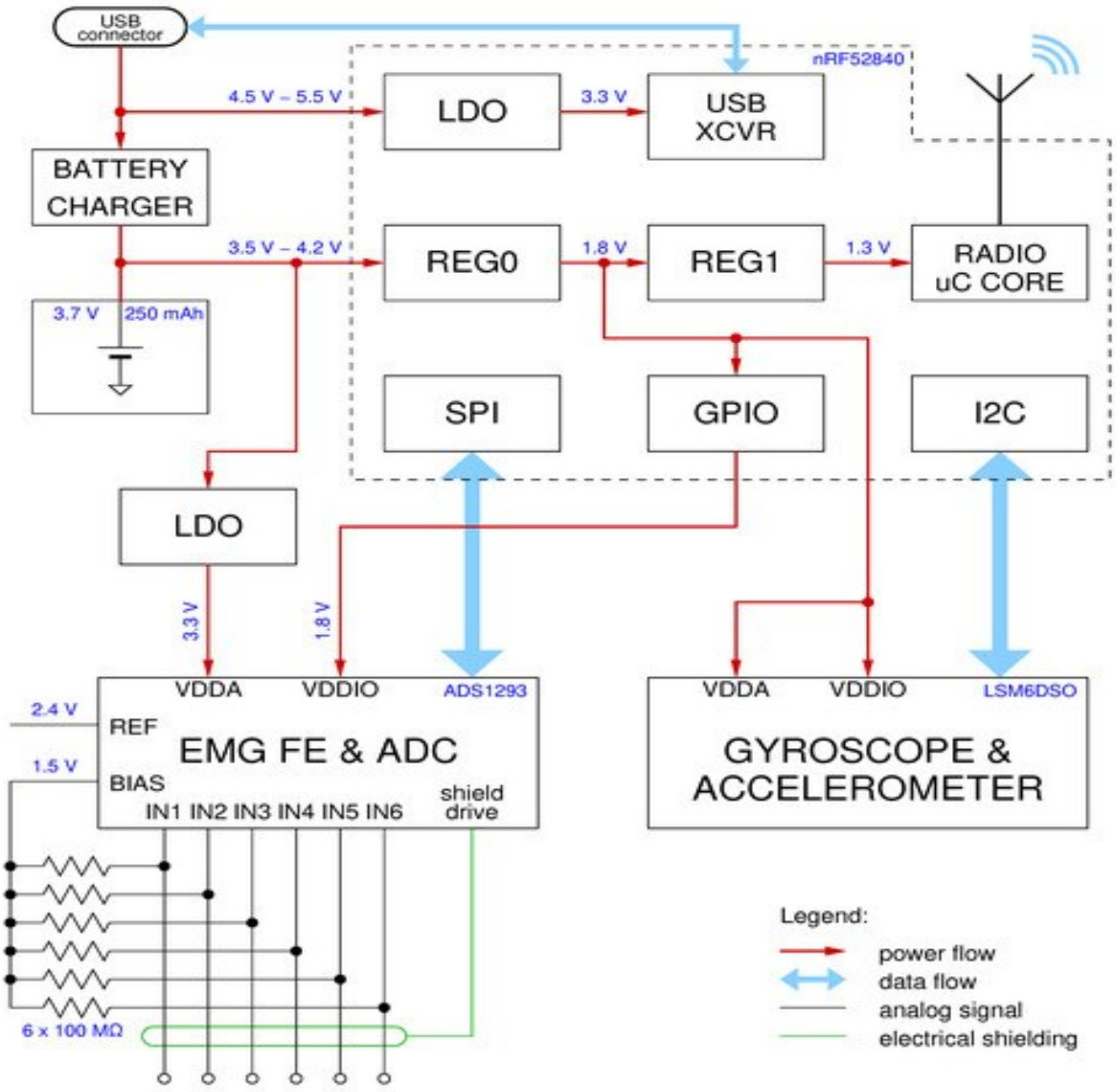
Poiché si prevede che il sistema venga utilizzato per periodi di tempo prolungati, il suo design è stato guidato dall'esigenza di ridurre il più possibile il consumo energetico, in modo da massimizzare la durata della batteria. Quindi, particolare attenzione sarà data al funzionamento a bassa potenza dell'intero sensore, che è ottenuto da un'attenta selezione di componenti e caratteristiche hardware, combinata con una strategia di gestione energetica su misura sviluppata in software. Nelle due sottosezioni seguenti viene fornita una panoramica dei componenti hardware selezionati per raggiungere i nostri obiettivi, insieme a una descrizione delle modalità operative del dispositivo.

3.2 CIRCUITO DI IMPLEMENTAZIONE

Il modulo sensore wireless è composto dai seguenti componenti principali:

- Microcontrollore con ricetrasmettitore wireless integrato. Utilizzando il versatile System-on-Chip (SoC) nRF52840 di Nordic Semiconductor, programmato per utilizzare lo stack di protocollo radio BLE, offre la connettività wireless necessaria insieme alla connettività cablata (USB) per la ricarica della batteria, accoppiamento semplificato e sicuro.
- Front-end analogico biopotenziale. Basato sul front-end (FE) altamente integrato ADS1293 di Texas Instruments, che include filtri digitali configurabili, amplificatori di strumentazione e ADC, potenziati con una rete di polarizzazione esterna adatta per l'acquisizione di segnali EMG ed ECG tramite elettrodi accoppiati in CC o CA.
- Piattaforma inerziale. Realizzato con l'accelerometro digitale e il giroscopio LSM6DSO a bassissima potenza di ST Microelectronics, è dotato di un accelerometro sempre attivo utilizzato per riattivare il sistema quando viene rilevato un movimento e viene utilizzato per lo streaming di dati di movimento a 6 assi quando il sistema è operativo.
- Alimentazione elettrica. Consiste in un caricabatterie ai polimeri di litio e un gruppo di regolatori lineari e di commutazione per fornire alimentazione ai diversi sottosistemi solo quando effettivamente necessario e progettati per massimizzare l'efficienza energetica

Uno schema a blocchi del sistema con il percorso di alimentazione evidenziato si vede sotto in figura:



Schema a blocchi. A Multi-Channel Electromyography, Electrocardiography and Inertial Wireless Sensor Module Using Bluetooth Low-Energy. MDPI. Web. Ottobre 2020. < https://www.mdpi.com/2079-9292/9/6/934/htm>

Schema a blocchi del sensore evidenzia i vari sottosistemi di alimentazione sfruttati per minimizzare i consumi energetici e le interfacce dati. L'ADS1293 acquisisce i segnali analogici e li trasmette al SoC tramite un'interfaccia

periferica seriale (SPI) quando abilitato dalla linea GPIO (general-purpose input/ output). L'LSM6DSO integra la piattaforma inerziale e trasmette i suoi dati al microcontrollore (uC) attraverso un'interfaccia a circuito inter-integrato (I2C).

Qui sotto si vede come è stato realizzato fisicamente il dispositivo:

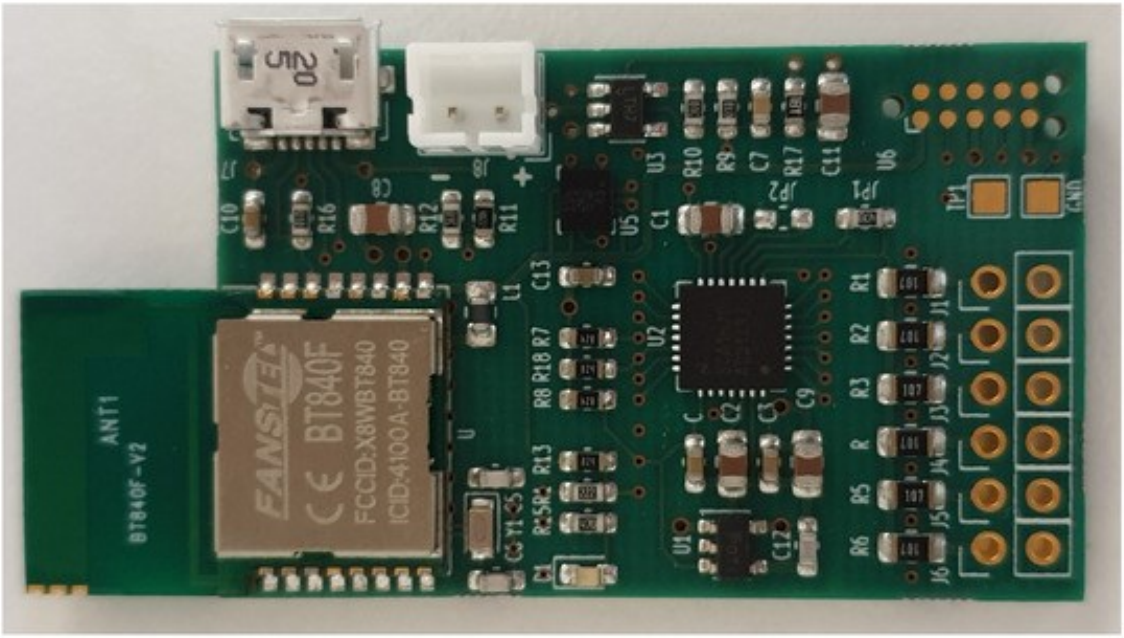


Figura del sensore prototipo. A Multi-Channel Electromyography, Electrocardiography and Inertial Wireless Sensor Module Using Bluetooth Low-Energy. MDPI. Web. Ottobre 2020. < https://www.mdpi.com/2079-9292/9/6/934/htm>

Fotografia del prototipo del sensore costruito a scopo di test e sperimentazione. Il chip di rilevamento EMG / ECG è U2, il sensore inerziale è U5.

3.1 MODALITÀ OPERATIVA DEL SISTEMA

In qualsiasi momento il sistema può essere in una delle cinque diverse modalità operative (OFF, IDLE, ADVERTISING, CONNECTED, STREAMING), ognuna delle quali è caratterizzata da diversi requisiti di alimentazione:

- OFF: questo è lo stato di alimentazione più basso, con il sistema completamente disabilitato, pensato per essere utilizzato per la conservazione a lungo termine, con un consumo di corrente paragonabile o inferiore al tasso di autoscarica della batteria. Viene inserito su richiesta dell'host o quando viene rilevata una condizione di batteria scarica. In questo stato, la CPU e l'accelerometro sono programmati nello stato di spegnimento per il minor consumo di corrente. Il front-end EMG non è affatto alimentato.

- IDLE: viene inserito dopo un periodo di inattività, ovvero dopo una lunga attesa senza che nessuna centrale BLE richieda una connessione. In questo stato l'accelerometro è mantenuto attivo in uno stato di alimentazione ultra-bassa e una frequenza di campionamento bassa (12,5 Hz). Quando rileva una variazione dell'accelerazione misurata su uno qualsiasi dei suoi tre assi resetta la CPU e sposta il sistema nello stato attesa. Il front-end EMG non è alimentato affatto come nello stato OFF.

- ADVERTISING: questo stato viene inserito dopo una riattivazione da IDLE, un ripristino da OFF o dopo che l'host ha disconnesso il collegamento BLE. In questo stato l'accelerometro viene mantenuto come in IDLE per evitare che l'attesa termini se c'è movimento, così come il front-end EMG, ancora non alimentato. La CPU è temporizzata da un LFXO esterno per ridurre il consumo energetico rispetto all'oscillatore RC interno.

- CONNECTED: inserito quando una centrale BLE stabilisce una connessione. In questo stato gli host possono controllare la maggior parte delle funzioni del sensore. Per impostazione predefinita, solo il monitoraggio della batteria è abilitato, l'LSM6DSO è spento, mentre l'ADS1293, che ha un tempo di avvio relativamente lungo a causa delle resistenze di polarizzazione di alto valore, è acceso e tenuto in stand-by in

attesa del host per richiedere lo streaming di dati. La CPU è ancora cronometrata dall'LFXO.

- STREAMING: inserito quando l'host abilita le notifiche per un servizio specifico. In questo stato, i dati richiesti vengono abilitati alla velocità dati prescritta e il flusso di dati non compresso viene inviato tramite il collegamento BLE. Per una marcatura temporale accurata dei dati, l'oscillatore a cristallo ad alta frequenza (HFXO) viene mantenuto attivo in ogni momento e non solo durante l'attività radio come avviene nei due stati precedenti. Ciò si traduce in un consumo di corrente leggermente superiore, ma è necessario garantire una corretta sincronizzazione dei dati.

4. POSTGRESQL

PostgreSQL ha alle proprie spalle oltre 30 anni di sviluppo. Il sistema di gestione del database relazionale a oggetti (ORDBMS) ha le proprie radici nel progetto POSTGRES dell'Università della California a Berkeley. È stato avviato nel 1986 sotto la guida di Michael Stonebraker e sponsorizzato dalla Defence Advanced Research Project Agency (DARPA) e dalla National Science Foundation (NSF). Nel 1994 gli studenti Andrew Yu e Jolly Chen estesero il codice di base con un interpreter SQL e pubblicarono questa nuova versione dal 30 al 50 percento più veloce con il nome di Postgres95 come soluzione open source (con una licenza propria che assomiglia alle licenze BSD e MIT). Due anni dopo l'applicazione per database, con la versione 6.0, ha preso il nome di PostgreSQL, utilizzato ancora oggi.

Il progetto POSTGRES ha svolto un prezioso lavoro e ha sviluppato numerosi concetti che sono stati introdotti successivamente in altri sistemi di database (principalmente commerciali). Pertanto, PostgreSQL non si distingue solo come database conforme a SQL, ma anche attraverso le seguenti moderne feature:

- Possibilità di domande complesse
- Chiavi esterne (foreign keys) per il collegamento di dati da due tabelle
- Trigger che vengono attivati automaticamente in ingresso e controllano, confermano, modificano, eliminano o in alternativa inseriscono i dati di riferimento
- Visualizzazioni aggiornabili

- Concetto di transazione completo
- Multiversion Concurrency Control (MVCC) per eseguire in modo efficiente l'accesso simultaneo al database

Inoltre, gli utenti possono modificare ed estendere notevolmente PostgreSQL grazie alla licenza gratuita, aggiungendo ad esempio nuovi tipi di dati, funzioni, operatori, metodi di indicizzazione o linguaggi procedurali (linguaggi di programmazione per la scrittura di funzioni e trigger).

Con questo server principalmente si creeranno tre relazioni: la prima relazione è il soggetto dove si andrà ad inserire i suoi dati. La seconda relazione sarà misure. La terza relazione si chiamerà salvafile dove verrà salvato il file binario.

5. PYTHON

Python è un linguaggio di programmazione ad alto livello ideato e rilasciato pubblicamente dal suo creatore Guido van Rossum nei primi anni Novanta.

Viene anche definito un linguaggio multi-paradigma in grado di supportare:

- la programmazione ad oggetti, in cui sono proprio gli oggetti ad essere l'elemento organizzativo principale;
- la programmazione procedurale, in cui sono le funzioni ad essere l'elemento organizzativo principale.

Per capire concretamente che cosa è Python e perché sono in tanti a sceglierlo basta analizzare tutti i vantaggi che contempla il suo utilizzo.

1. Questo programmatore di linguaggio non è a pagamento, ed inoltre nonostante è completamente gratuito viene costantemente aggiornato;
2. È ricco di librerie, per l'esattezza contiene più di 200 moduli per poter svolgere con facilità compiti diversi;
3. Contempla una memoria gestita automaticamente, grazie al "garbage collection" che gestisce il rilascio automatico della memoria;
4. È adatto a diverse piattaforme e nello specifico il linguaggio può essere eseguito su Windows, Macintosh, Dos e Linux ma anche sulle piattaforme mobile come IOS e Android;
5. È di semplice uso, soprattutto perché è facilmente intuibile, costruito in modo chiaro e di semplice interpretazione.

Con il programma Python si andrà a creare un'interfaccia, la quale permetterà di fare tutte le operazioni che servono per caricare i dati nel server e scaricarli successivamente. Come programma si utilizzerà PyCharm che a sua volta userà Python come interprete di programmazione.

5.1 PYCHARM

PyCharm è un IDE (*Integrated Development Environment*) pensato per lo sviluppo di applicazioni in Python. Si tratta di un progetto pensato per i professionisti e sviluppato dalla JetBrain.

Quello che rende interessante e appetibile PyCharm è l'insieme degli strumenti di cui è dotato, infatti i Pycharm Development Tool permettono di testare i propri script tramite una comoda Python console integrata. Grazie a tale feature lo sviluppo di un'applicazione risulta essere notevolmente più fluido, Pycharm consente infatti di focalizzarsi sul proprio progetto senza dover passare da un programma all'altro.

Questo vale per tutte le varie fasi dello sviluppo, dopo la prima scrittura del codice è possibile affidarsi al debugging tool incluso in PyCharm per verificare la presenza di bug e imperfezioni nel codice. È inoltre possibile realizzare delle Macro per automatizzare vari task, una killer feature per tutti coloro che si ritrovano ad eseguire dei task estremamente ripetitivi durante le fasi di sviluppo.

PyCharm è multipiattaforma, infatti è disponibile per Linux, Windows e MacOS. Viene distribuito in due versioni: la *Community Edition*, rilasciata sotto licenza Apache 2.0, e la *Professional Edition*, che dispone invece di una licenza proprietaria con cui sbloccare determinate feature "premium" nonché il supporto dedicato per le aziende e i privati che decidono di acquistare questa variante.

Ecco in breve le feature principali di PyCharm:

- Coding assistance/analysis, con code completion, syntax ed error highlighting, linter integration e sistema per il quick fixing.
- Project/code navigation: vista rapida della struttura dei file e quick jumping tra i file, classi e methodi.
- Python refactoring: rename, gli extract method, gestione di variabili e costanti, pull up e push down.
- Supporto nativo per i web frameworks Django, web2py e Flask.
- Python debugger e unit testing, con code coverage line-by-line.
- Supporto per il Google App Engine.
- Controllo di versione integrato con UI per la gestione unificata di Mercurial, Git, Subversion, Perforce e CVS.

Installare Pycharm è molto semplice, si può usare l'installer per il proprio sistema oppure cercare nei registri delle distribuzioni che lo includono.

Inoltre, è stato molto utile quando si dovevano installare dei pacchetti che non erano presenti, infatti si installano tutti dall'applicazione mentre si programma e non c'è bisogno di andare come in Python nella shell e scrivere del codice per installarli.

6. SVOLGIMENTO PROGETTO

6.1 INTRODUZIONE

Si è partiti dal fatto che serviva fare il catalogo di alcuni dati, quindi per prima cosa si è considerato cosa serviva per fare ciò e subito è venuto alla luce che serviva un server, e tra i tanti server open source disponibili si è deciso a PostgreSql per le sue caratteristiche.

Dopo che è stato scelto il server si dovevano scegliere le relazioni che servivano, infatti non era possibile mettere tutto in un'unica relazione, dato che si avrebbe avuto un gran disordine e quindi è stato pensato di utilizzare diverse entità.

Successivamente dopo aver deciso tutto questo si doveva scegliere il linguaggio di programmazione e la scelta è ricaduta su Python che ultimamente si è diffuso in maniera esponenziale dato che adesso è uno dei tre linguaggi di programmazione più utilizzati al mondo. Dopo aver scelto il linguaggio è stato scelto di usare un IDE che è un software progettato per la realizzazione di applicazioni che aggrega strumenti di sviluppo comuni in un'unica interfaccia utente grafica. E per questo la scelta è ricaduta su Pycharm ed è stato usato come interprete Python in modo che se ci fosse stata la necessità di importare un pacchetto che al momento dell'installazione di PyCharm non era presente, sarebbe stato importato facilmente.

Tutto è partito con la decodifica dei dati ricevuti dal dispositivo Bluetooth, i quali vengono convertiti in un file testo che contiene le informazioni delle misure e in un file binario che contiene tutte le informazioni. L'obbiettivo principale è salvare questi dati in modo chiaro e rinvenirli in modo semplice.

Quindi quello che è stato fatto è con l'iniziare prendendo il file testo e spezzettarlo nelle diverse parti e poi da queste parti si sono presi solo i dati che servivano, e successivamente salvare completamente il file binario e scaricarlo sul computer.

6.2 CREAZIONE RELAZIONE POSTGRESQL

6.2.1 CONNESSIONE AL SERVER

Si è partito all'inizio con l'importare il pacchetto pyscopg2 in modo che si possa connettere al server utilizzando il seguente codice:

```
import psycopg2


DB_NAME = "postgres"
DB_USER = "postgres"
DB_PASS = "python"
DB_HOST = "localhost"
DB_PORT = "5432"

conn = psycopg2.connect(database=DB_NAME, user=DB_USER,
                        password=DB_PASS, host=DB_HOST, port=DB_PORT)

print(" connessione con successo")

cur = conn.cursor()
```

Si devono passare le seguenti informazioni per connetersi al database:

- Nome Server
- Nome Utente del Server
- La Password del Server
- Host
- Numero di Porta

6.2.2 CREAZIONE RELAZIONE SOGGETTO

Dato che le misure che si vanno a fare riguardano un soggetto fisico, si deve avere la possibilità salvare le sue informazioni nel server in modo da averle permanentemente e si devono salvare le sue informazioni principali che sono:

- ID_soggetto, questa colonna identificherà univocamente una persona, ed un modo per identificarla potrebbe essere il suo codice fiscale
- Nome
- Cognome
- Anno di Nascita
- Luogo di Nascita
- Residenza

Di sotto viene riportato il codice della creazione della relazione Soggetto:

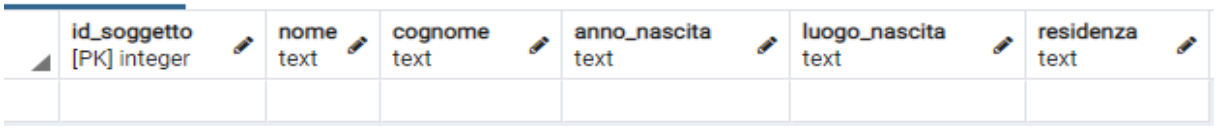
```
cur.execute("""

CREATE TABLE Soggetto
(
 ID_soggetto INT PRIMARY KEY NOT NULL,
 NOME TEXT NOT NULL,
 COGNOME TEXT NOT NULL,
 ANNO_NASCITA TEXT NOT NULL,
 LUOGO_NASCITA TEXT NOT NULL,
 RESIDENZA TEXT NOT NULL
 )
""")

conn.commit()
```

Attraverso il commando execute si va ad eseguire la query della creazione della tabella, mentre per eseguire la query, in modo che si crei realmente la tabella nel server, si utilizza conn.commit().

Di sotto si può vedere com'è graficamente la tabella nel Database:



6.2.3 CREAZIONE RELAZIONE MISURE

Successivamente dopo la creazione del soggetto si va a creare la relazione misure in modo da caricare le seguenti informazioni prese da un file di testo:

- ID_misure, sempre per identificare univocamente l'informazione della misura fatta
- ID_soggetto che sarà una chiave esterna per mettere in relazione la tabella soggetto con la tabella delle misure, in modo che quando si va a cercare una misura la si cerca attraverso l'ID del soggetto
- Mac_address
- La frequenza del campionamento del EMG
- Il canale 1 che ha come parametro l'elettrodo 1-2
- Il canale 2 che ha come parametro l'elettrodo 3-4
- Il canale 3 che ha come parametro l'elettrodo 5-6
- La frequenza del giroscopio
- Fondo scala lineare del Giroscopio
- Fondo scala angolare del Giroscopio
- La data quando si carica la misura nel server

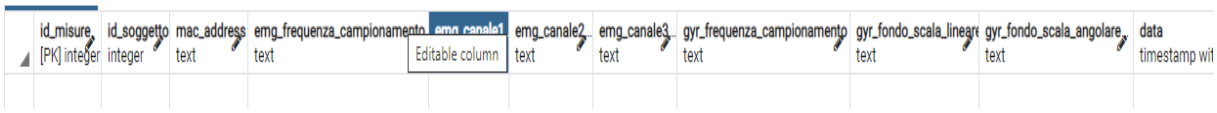
Sotto si riporta il codice della creazione della tabella Misure:

```
cur.execute("""

CREATE TABLE Misure
(
 ID_misure INT PRIMARY KEY NOT NULL,
 ID_soggetto INT NOT NULL,
 mac_address TEXT,
 emg_frequenza_campionamento TEXT,
 emg_canale1 TEXT,
 emg_canale2 TEXT,
 emg_canale3 TEXT,
 gyr_frequenza_campionamento TEXT,
 gyr_fondo_scala_lineare TEXT,
 gyr_fondo_scala_angolare TEXT,
 data TIMESTAMP,
 FOREIGN KEY (ID_soggetto) REFERENCES soggetto(ID_soggetto)
 )
""")

conn.commit()
```

Qui sotto invece, so ha come si vede la tabella nel server:



6.2.4 CREAZIONE RELAZIONE SALVAFILE

Oltre alla tabella delle misure ci serve un'altra tabella dove si andrà a caricare il file binario, questa tabella avrà le seguenti caratteristiche:

- ID_salvafile, che lo identifica univocamente da ogni riga della stessa tabella presente nel database
- ID_misure in modo che si possano collegare la relazione misure con salvafile
- File dove si andrà a mettere il file binario
- Data dove stamperà sulla tabella la data e l'ora in cui è stato immesso il file binario

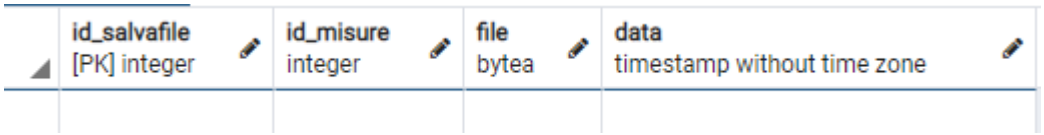
Codice della creazione della tabella Salvafile:

```
cur.execute("""

CREATE TABLE Salvafile
(
 id_salvafile INT PRIMARY KEY,
 id_misure INT NOT NULL,
 file BYTEA,
 data TIMESTAMP,
 FOREIGN KEY(id_misure) REFERENCES misure(id_misure)
)
""")

conn.commit()
```

Di sotto si riporta come la tabella si vede graficamente nel server:

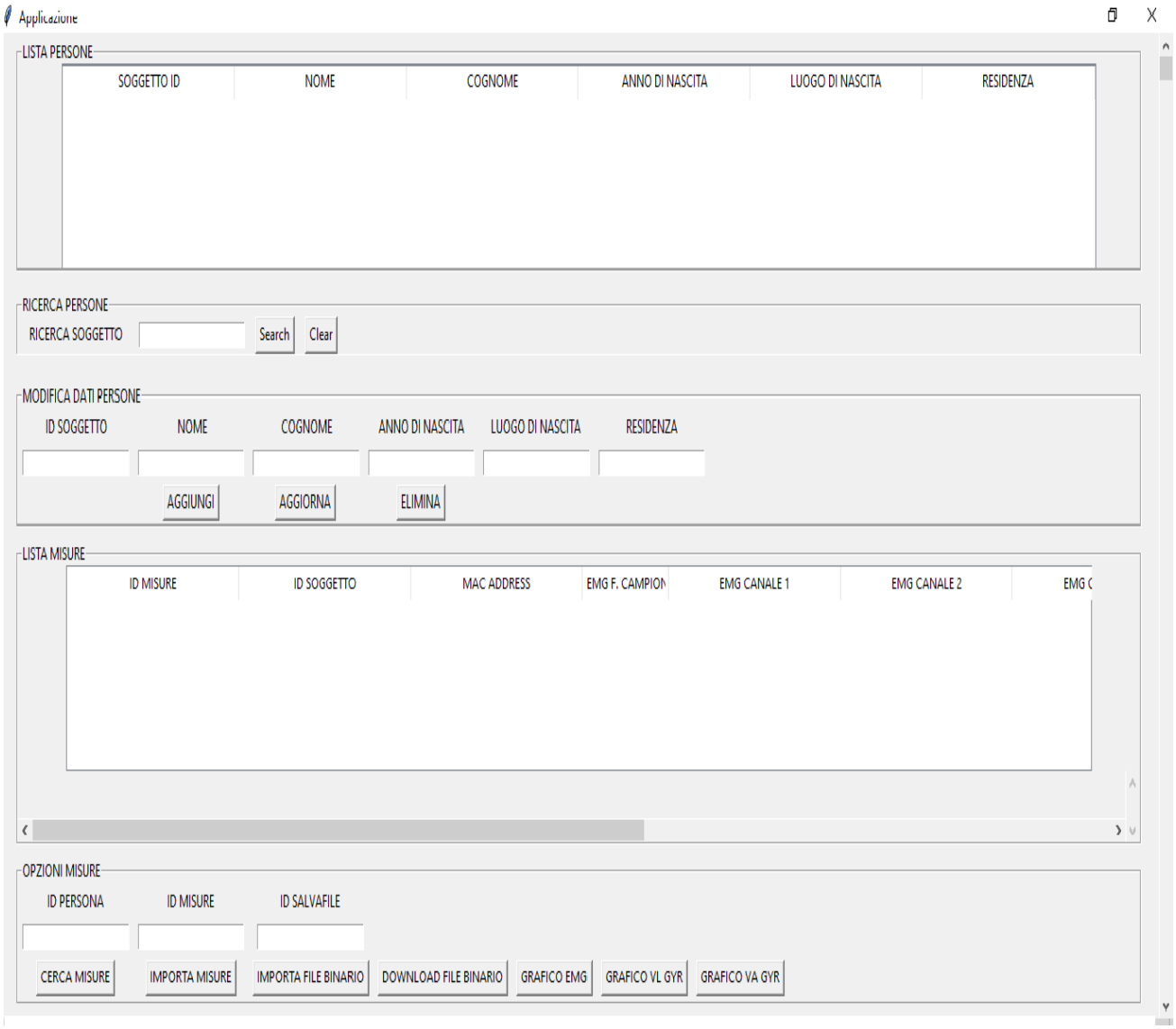


6.3 CREAZIONE INTERFACCIA GRAFICA

Si deve costruire un'interfaccia grafica per fare tutte le operazioni che servono per questo è stato usato come GUI (Graphic User Interface) tkinter, una libreria che è stato importato nella nostra applicazione.

L'interfaccia è stata divisa in cinque frame, su ciascuno di questi frame si vanno a fare delle operazioni attraverso dei tasti che si devono premere con il mouse o con invio.

Qui sotto viene riportata un'immagine di com'è l'interfaccia che vedrà un utente:



Come si vede dall'immagine l'interfaccia è divisa in cinque parti:

- Nel primo frame si ha una finestra che mostra le informazioni delle persone
- Nel secondo frame, attraverso i pulsanti è possibile fare la ricerca di una persona
- Il terzo frame va ad effettuare tutte le modifiche su una specifica persona
- Nel quarto frame si ha sempre una finestra che mostra le informazioni delle misure
- Nel quinto frame si vanno a compiere delle operazioni con il file testo e con il file binario ed inoltre si hanno dei pulsanti che mostrano come viene graficamente una serie di dati prese sempre da un file di testo

Per fare determinate operazioni sono state importare le seguenti librerie:

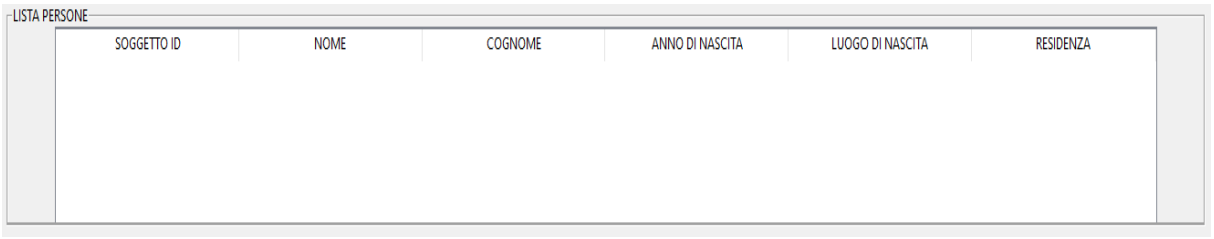
```
from tkinter import *
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox, filedialog
import psycopg2
import os
import matplotlib.pyplot as pt
import numpy as np
```

6.3.1 PRIMO FRAME: *LISTA PERSONE*

Questo frame mostra graficamente come si vedono le informazioni delle varie persone a schermo. Infatti, mostra le informazioni che una persona comunemente ha che sono:

- ID_persona
- Nome
- Cognome
- Anno di nascita
- Luogo di nascita
- Residenza

Il risultato graficamente viene visto cosi:



Di sotto viene riportato il codice che è stato utilizzato per la sua costruzione:

```
trv = ttk.Treeview(wrapper1, columns=(1, 2, 3, 4, 5, 6), show="headings",
height="6")
trv.pack()

trv.heading(1, text="SOGGETTO ID")
trv.heading(2, text="NOME")
trv.heading(3, text="COGNOME")
trv.heading(4, text="ANNO DI NASCITA")
trv.heading(5, text="LUOGO DI NASCITA")
trv.heading(6, text="RESIDENZA")
```

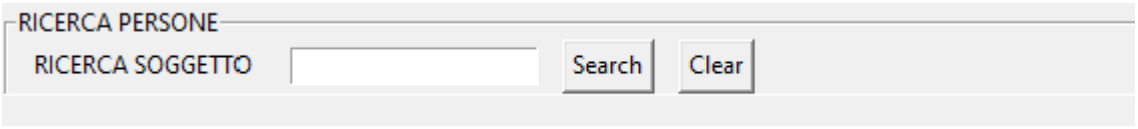
Praticamente per la sua creazione è stato usato un Treeview e gli viene passato dove creare questa lista nell'interfaccia tkinter e il numero di colonne, inoltre successivamente ad ogni colonna viene assegnato un nome con il comando trv.heading.

6.3.2 SECONDO FRAME: *RICERCA PERSONE*

In questo frame si vanno ad effettuare due semplici operazioni:

- La prima è la ricerca della persona interessata, e questo avviene mediante l'inserimento del cognome o nome nell'apposita casella
- La seconda sarebbe Clear, cioè una volta aver trovato la persona vado a ripulire la lista in modo che possa cercare un'altra persona nel caso mi serva

Di sotto un'immagine del frame numero due:



Il codice per la ricerca è il seguente:

```
def ricerca():
    q2 = q.get()
    query ="SELECT id_soggetto,nome, cognome, anno_nascita, luogo_nascita,
residenza FROM  soggetto WHERE nome LIKE '%"+q2+"%' OR cognome LIKE
'%"+q2+"%'"
    cur.execute(query)
    rows = cur.fetchall()
    update(rows)
```

Così facendo si utilizza la query per fare la ricerca; questa è una funzione che poi viene richiamata nel pulsante Search. Con cur.fetchall() si vanno a riempire le righe della lista delle persone e con update(rows) si aggiornano le righe della lista in modo che si possa vedere a schermo la persona desiderata.

Codice per la pulizia della lista:

```
def clear():
    query=" SELECT id_soggetto,nome,cognome, anno_nascita, luogo_nascita,
residenza FROM soggetto "
    cur.execute(query)
    rows=cur.fetchall()
    update(rows)
```

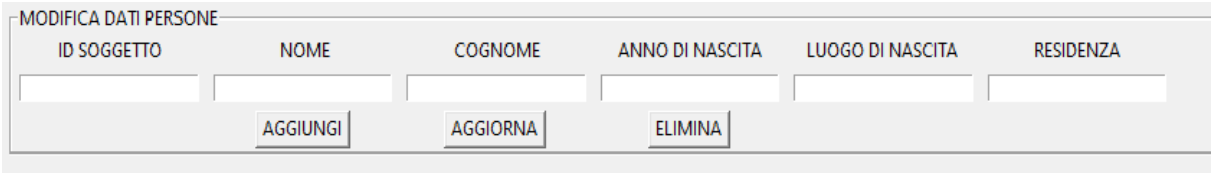
Praticamente fa vedere tutte le informazioni delle persone.

6.3.3 TERZO FRAME: *MODIFICA DATI PERSONE*

Questo frame mostra tutte le possibili operazioni che posso essere eseguite su una persona, le quali sono le seguenti:

- Aggiungere una persona, infatti se una persona non è presente nel database si deve aggiungere
- Aggiornare una persona, questo pulsante serve principalmente per aggiornare la residenza della persona perché secondo i dati che sono presenti solo essa può cambiare
- Eliminare una persona dal sistema

Il frame è fatto nel seguente modo:



Praticamente per aggiungere una persona si deve inserire l'ID della persona, il nome, cognome, anno di nascita, luogo di nascita e la residenza. Per aggiornare la persona lo si fa attraverso l'ID del soggetto in modo da modificargli la residenza e lo stesso vale per l'eliminazione di una persona che lo si fa sempre attraverso l'ID della persona.

Sotto si può vedere il codice per aggiungere una persona:

```
def aggiungi_persona():
    persona_id = t1.get()
    nome = t2.get()
    cognome = t3.get()
    anno_nascita = t4.get()
    luogo_nascita = t5.get()
    residenza = t6.get()
    try:
       query = "INSERT INTO soggetto(id_soggetto, nome, cognome,
anno_nascita, luogo_nascita, residenza) VALUES(%s, %s, %s, %s, %s, %s)"
       cur.execute(query, (persona_id, nome, cognome, anno_nascita,
luogo_nascita, residenza))
       conn.commit()
       messagebox.showinfo("Salva persona", "Persona salvata con
successo!")
    except psycopg2.Error as error:
        messagebox.showinfo("Errore", "INSERIRE ID PERSONA O ID PERSONA
GIA' PRESENTE!").format(error)
        conn.rollback()
    clear()
```

La prima parte della funzione serve per prendere i dati dalle caselle, dopo di che si ha una seconda parte dove si va a fare l'inserimento della persona attraverso la query e se l'inserimento è stato eseguito correttamente compare un messaggio che la persona è stata salvata attraverso l'istruzione messagebox.showinfo. Inoltre, se si sbaglia l'immissione dell'id della persona attraverso il commando try ed except posso ritornare indietro e infatti sbagliando l'id appare sullo schermo un messaggio di errore, e attraverso l'istruzione conn.rollback() ritorno indietro.

Sotto si ha il codice per l'aggiornamento della persona:

```
def aggiorna_persona():
       persona_id = t1.get()
       nome = t2.get()
       cognome = t3.get()
       annonascita = t4.get()
       luogonascita = t5.get()
       residenz = t6.get()
       try:
           if messagebox.askyesno("Conferma aggiornamento", "Sei sicuro che
vuoi aggiornare?"):
               query = "UPDATE soggetto SET nome = %s, cognome = %s,
```

```
anno_nascita=%s, luogo_nascita=%s, residenza=%s  WHERE id_soggetto = %s"
            cur.execute(query, (nome, cognome, annonascita, luogonascita,
residenz, persona_id))
            conn.commit()
            clear()
          else:
           return True
      except psycopg2.Error as error:
          messagebox.showinfo("Errore", "AGGIUNGI ID PER
AGGIORNARE!").format(error)
          conn.rollback()
```

Il codice è molto simile al codice per inserire una persona solo che qui si va a fare l'aggiornamento. La cosa che cambia a parte la query è che si ha un if attraverso il quale si chiede se si è sicuri di voler aggiornare quella determinata persona.

In basso il codice per l'eliminazione di una persona:

```
def elimina_persona():
    persona_id = t1.get()
    try:
        if messagebox.askyesno("Conferma cancellazione?", "Sei sicuro che
vuoi cancellare questa persona?"):
            query = "DELETE from soggetto WHERE id_soggetto=" + persona_id
            cur.execute(query)
            conn.commit()
            clear()
        else:
            return True
    except psycopg2.Error as error:
        messagebox.showinfo("Errore", "DIGITA ID PERSONA PER
ELIMINARE!").format(error)
        conn.rollback()
```

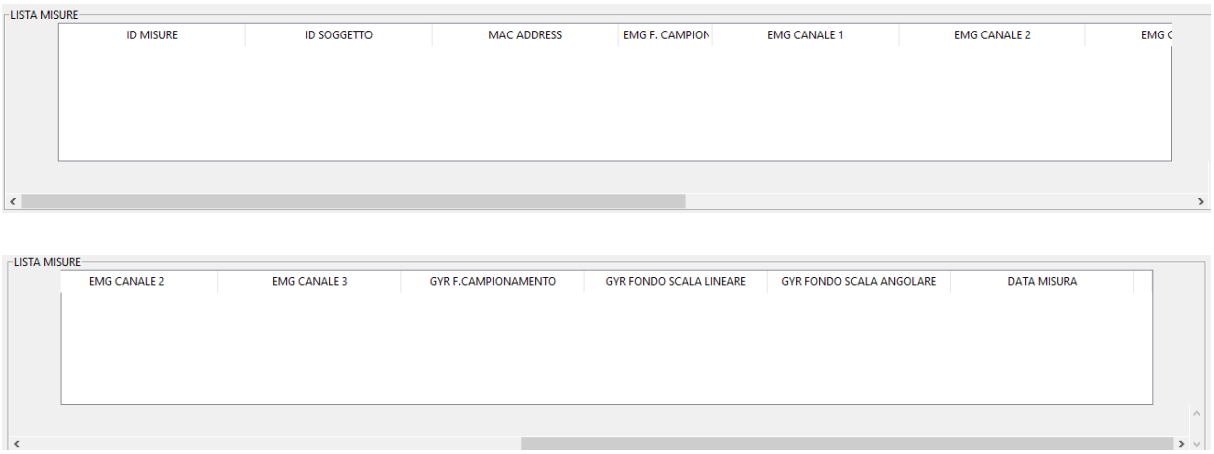
Pure qui prima di andare a fare la query dell'eliminazione della persona desiderata si va a chiedere se si è sicuri di volerla fare attraverso l'if.

6.3.4 QUARTO FRAME: *LISTA MISURE*

In questo frame si ha una lista che contiene tutte le caratteristiche delle misure che una persona ha fatto, cioè praticamente vengono mostrate le seguenti informazioni dal database:

- ID della misura effettuata in modo da identificarla univocamente
- ID della persona che andrà a fare la misura
- Il mac address
- La frequenza di campionamento dell'EMG
- Canale 1 con i rispettivi elettrodi
- Canale 2 con i suoi elettrodi
- Canale 3 sempre con i suoi elettrodi
- La frequenza di campionamento del giroscopio
- Il fondo scala lineare del giroscopio
- Il fondo scala angolare del giroscopio
- La data insieme all'ora di quando si inseriscono i dati nel server

Sotto si può vedere come è fatto il frame numero 4:



Si è dovuto aggiungere una barra di scorrimento orizzontale in modo da vedere tutti i dati.

Sotto si ha il codice utilizzato per la creazione della Treview usata nel frame numero 4:

```
trv2 = ttk.Treeview(wrapper4, columns=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12), show="headings", height="6")
trv2.pack()


trv2.heading(1, text="ID MISURE")
trv2.heading(2, text="ID SOGGETTO")
trv2.heading(3, text="MAC ADDRESS")
trv2.heading(4, text="EMG F. CAMPIONAMENTO")
trv2.heading(5, text="EMG CANALE 1")
trv2.heading(6, text="EMG CANALE 2")
trv2.heading(7, text="EMG CANALE 3")
trv2.heading(8, text="GYR F.CAMPIONAMENTO")
trv2.heading(9, text="GYR FONDO SCALA LINEARE")
trv2.heading(10, text="GYR FONDO SCALA ANGOLARE")
trv2.heading(11, text="DATA MISURA")
trv2.column('#0', width=50, minwidth=100)
trv2.column('#1', width=90, minwidth=200)
trv2.column('#2', width=90, minwidth=200)
trv2.column('#3', width=90, minwidth=200)
trv2.column('#4', width=90, minwidth=100)
trv2.column('#5', width=90, minwidth=200)
trv2.column('#6', width=90, minwidth=200)
trv2.column('#7', width=90, minwidth=200)
trv2.column('#8', width=90, minwidth=200)
trv2.column('#9', width=90, minwidth=200)
trv2.column('#10', width=90, minwidth=200)
trv2.column('#11', width=90, minwidth=200)
```

Di seguito si ha il codice per creare una barra orizzontale:

```
xscrollbar = ttk.Scrollbar(wrapper4, orient="horizontal",
command=trv2.xview)
xscrollbar.pack(side=BOTTOM, fill="x")
trv2.configure(xscrollcommand=xscrollbar.set)
```

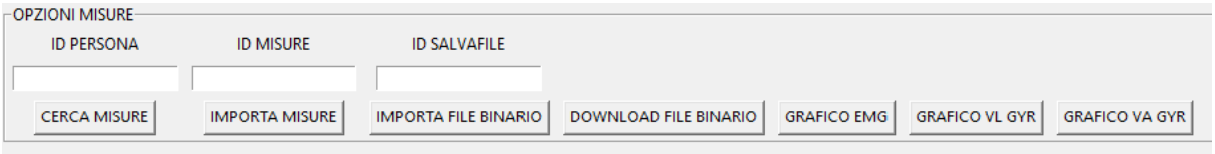
Con questo codice dico che voglio la barra orizzontale nella parte inferiore del Treview del frame numero 4.

6.3.5 QUINTO FRAME: *OPZIONI MISURE*

In questa parte dell'interfaccia si hanno tutte le operazioni più importanti, le quali sono:

- Cercare una misura attraverso l'id della persona
- Importare le informazioni dal file di testo alla tabella misure che si trova nel nostro server
- Importare completamente il file binario nel server e precisamente nella tabella salvafile
- Scaricare il file binario dal server dandogli anche un nome alternativo se si vuole
- Si ha anche la possibilità di vedere graficamente come vengono le misure fatte e quindi si avrà un grafico per l'EMG, un grafico per il fondo scala lineare del giroscopio e l'ultimo grafico sarà utilizzato per il fondo scala angolare del giroscopio

Sotto si può vedere come viene il frame numero 5 graficamente:



Se nell'importare un file di testo o importare un file binario o esportarlo si sbaglia a digitare un'informazione è possibile ritornare indietro con una sequenza di codici.

Sotto si ha il codice per cercare una misura:

```
def search1():
    ricerca = q1.get()
    query = "SELECT
id_misure,id_soggetto,mac_address,emg_frequenza_campionamento,emg_canale1,e
mg_canale2,emg_canale3,gyr_frequenza_campionamento,gyr_fondo_scala_lineare,
```

```
gyr_fondo_scala_angolare,data FROM misure WHERE id_soggetto="+ricerca+""
    cur.execute(query)
    rows = cur.fetchall()
    update1(rows)
```

È molto semplice ed intuitivo infatti si fa a vare una query e si cerca la misura attraverso l'ID della persona.

Di sotto si può vedere il codice per importare un file di testo:

```
def save():
    id_persona = q1.get()
    id_misure = t7.get()
    list = []
    fn = filedialog.askopenfilename(title="Seleziona file Testo",
filetypes=(("Text file", "*.txt"), ("all files", "*.*")))
    with open(fn,"r") as f:
        for line in f:
            list.append(line)
    mac = list[0].split()[2]
    fre = list[1].split()[2]
    ch1 = list[1].split()[4].split(':')[1]
    ch2 = list[1].split()[5].split(':')[1]
    ch3 = list[1].split()[6].split(':')[1]
    fgyr = list[2].split()[2]
    va = list[2].split()[4]
    vl = list[2].split()[6]
    try:
       query = "INSERT INTO
misure(id_misure,id_soggetto,mac_address,emg_frequenza_campionamento,emg_ca
nale1,emg_canale2,emg_canale3,gyr_frequenza_campionamento,gyr_fondo_scala_l
ineare,gyr_fondo_scala_angolare,data)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,NOW()) "
       cur.execute(query, (id_misure, id_persona, mac, fre,
ch1,ch2,ch3,fgyr, va, vl,))
       conn.commit()
       messagebox.showinfo("Salva Misure","Le misure sono state salvate
corretamente!")
    except psycopg2.Error as error:
        messagebox.showinfo("Errore", "AGGIUNGI ID PERSONA E
MISURE!").format(error)
        conn.rollback()
```

Praticamente viene creata una lista in modo da passare le informazioni del file di testo nella lista, dopo aver creato la lista si cerca il file di testo nel computer e lo si apre attraverso open() passandogli all'interno il file di testo e naturalmente

anche il modo di come aprirlo , cioè se è in lettura o scrittura. Dopo aver fatto ciò attraverso list.append(line) si salvano gli elementi del file di testo nella lista.

Successivamente essendo che il file di testo è strutturato nello stesso modo vengono divise le righe nella lista e si prende attraverso split() l'informazione che serve di ogni riga e il gioco è fatto, infatti dopo ciò si va ad eseguire la query di inserimento dei dati della misura fatta e vengono inseriti a mano attraverso una casella l'ID della persona che ha fatto la misura e naturalmente l'ID misura.

Per quanto riguardo l'importazione di un file binario il codice cambia di poco, infatti:

```
def salvafile():
    id_salvafile=t8.get()
    id_misure=t7.get()
    fn = filedialog.askopenfilename(title="select file", filetypes=(("Bytea
file","*.bt"),("allfiles","*.*")))

    with open(fn,"rb") as f:
        data=f.read()# this is binary data
    try:
        sql="INSERT INTO salvafile(id_salvafile,id_misure,file,data)
values(%s,%s,%s,NOW())"
        cur.execute(sql,(id_salvafile,id_misure,data,))
        conn.commit()
        messagebox.showinfo("Success", "Il tuo file è stato inserito
corretamente nel database!")

    except psycopg2.Error as error:
        messagebox.showinfo("Errore", "AGGIUNGI ID MISURE!").format(error)
        conn.rollback()
```

In questo caso si va sempre ad aprire il file con open e dopo di che si inserisce completamente il file binario passandogli a mano attraverso le apposite caselle, l'ID del file e l'ID della misura a cui sarà collegato.

Mentre per quanto riguarda il download del file binario si va a fare un'operazione un po' diversa:

```
def importafile():

    id_salvafile =t8.get()
    try:
       fn = filedialog.asksaveasfilename(initialdir=os.getcwd(), title =
"download", filetypes=(("Bytea file", "*.bt"), ("all files", "*.*")) )
       sql = "SELECT file FROM salvafile WHERE id_salvafile =
"+id_salvafile+""
       cur.execute(sql)
       impor = cur.fetchall()
       for i in impor:
          data = i[0]
       with open(fn, "wb") as f:
          f.write(data)
       f.close()
       messagebox.showinfo("success", "file scaricato corettamente")
    except psycopg2.Error as error:
        messagebox.showinfo("Errore", "IMPOSSIBILE SCARICARE
FILE!").format(error)
        conn.rollback()
```

Attraverso l'ID del file si prende il file binario interessato dal database e si va a salvare il file con un nome attraverso filedialog.asksaveasfilename, dopo di che si apre il file binario con open() e lo si va ad aprire in modalità scrittura.

Sotto si mostra com'è fatto il codice per fare il grafico, grazie a numpy vengono salvate tutte le 720000 righe del file di testo senza che il programma vada in crash:

```
def graficoemg():
    fn = filedialog.askopenfilename(title="Seleziona file Testo",
                                    filetypes=(("Text file", "*.txt"),
("all files", "*.*")))
    data = np.loadtxt(fn)
    x = data[:, 0]
    y = data[:, 1]
    pt.plot(x, y)
    pt.show()
```

Grazie a numpy come libreria e grazie succesivamente a loadtext() si riesce a salvare il file su data e successivamente a dividerlo in colonne.

Codice per il grafico della velocità lineare del giroscopio:

```
def graficovlgyr():
    fn = filedialog.askopenfilename(title="Seleziona file Testo",
                                    filetypes=(("Text file", "*.txt"),
("all files", "*.*")))
    data = np.loadtxt(fn)
    x = data[:, 0]
    y = data[:, 2]
    y2 = data[:, 3]
    y3 = data[:, 4]
    pt.plot(x, y, label="asse x", color="red")
    pt.plot(x, y2, label="asse y", color="green")
    pt.plot(x, y3, label="asse z", color="black")
    pt.legend()
    pt.show(
```

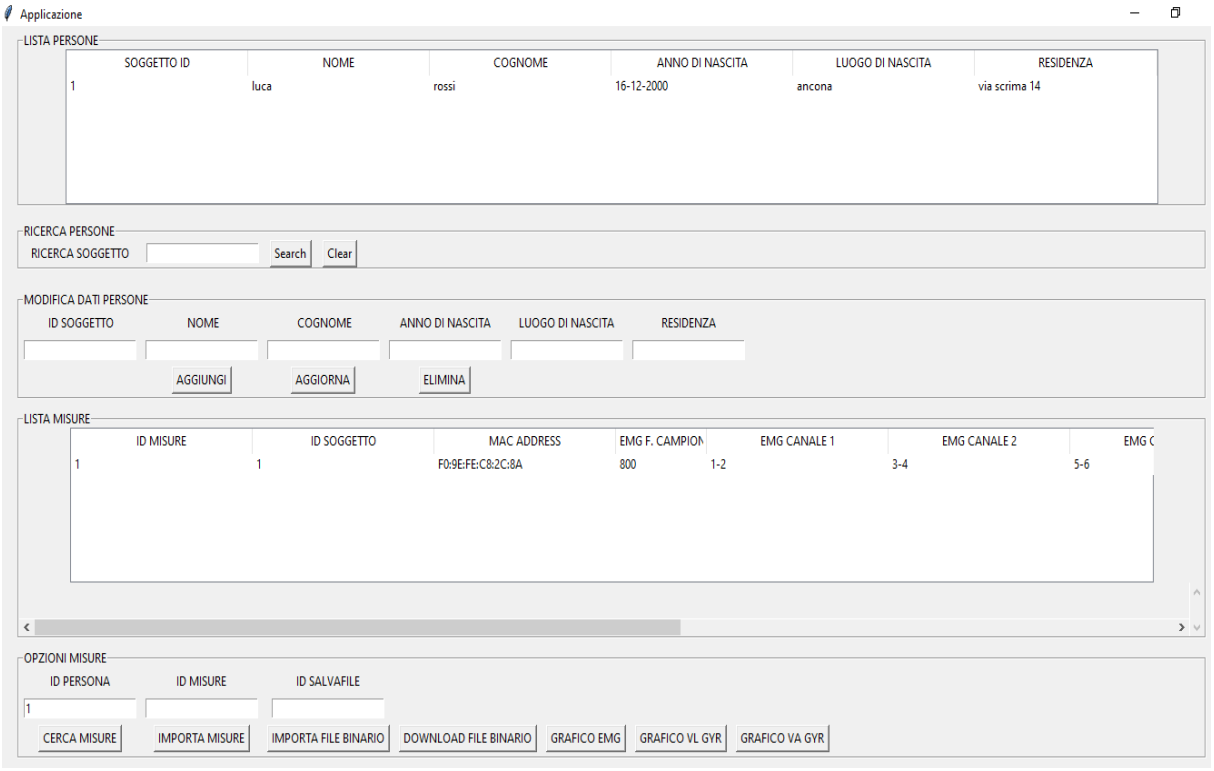
Codice per il grafico della velocità angolare del giroscopio:

```
def graficovagyr():
    fn = filedialog.askopenfilename(title="Seleziona file Testo",
                                    filetypes=(("Text file", "*.txt"),
("all files", "*.*")))
    data = np.loadtxt(fn)
    x = data[:, 0]
    y = data[:, 5]
    y2 = data[:, 6]
    y3 = data[:, 7]
    pt.plot(x, y, label="asse x", color="red")
    pt.plot(x, y2, label="asse y", color="green")
    pt.plot(x, y3, label="asse z", color="black")
    pt.legend()
    pt.show()
```

7. SPERIMENTAZIONE

Si riporta sotto una delle tante prove effettuate.

Sotto si può vedere com'è l'interfaccia quando si fa la ricerca di una persona e la ricerca di una misura:



Qui sotto si vede come le informazioni immesse vengono salvate nelle relazioni nel server.

Relazione soggetto:

	id_soggetto [PK] integer	nome text	cognome text	anno_nascita text	luogo_nascita text	residenza text
1	1	luca	rossi	16-12-2000	ancona	via scrima 14

Relazione misure:

id_misure [PK] integer	id_soggetto integer	mac_address text	emg_frequenza_campionamento text	emg_canale1 text	emg_canale2 text	emg_canale3 text	gyr_frequenza_campionamento text	gyr_fondo_scala_lineare text	gyr_fondo_scala_angolare text	data timestamp
1	1	F0:9E:FE:C8:2C:8A	800	1-2	3-4	5-6	104.1667	500	8	2020-10-11

Relazione salvafile:

	id_salvafile [PK] integer	id_misure integer	file bytea	data timestamp without time zone
1	1	1	[binary data]	2020-10-11 20:31:16.555536
2	2	1	[binary data]	2020-10-11 20:31:56.799498

Adesso si possono vedere come vengono graficamente i dati presi dal file di testo riguardante il giroscopio e l'EMG.

Grafico EMG:

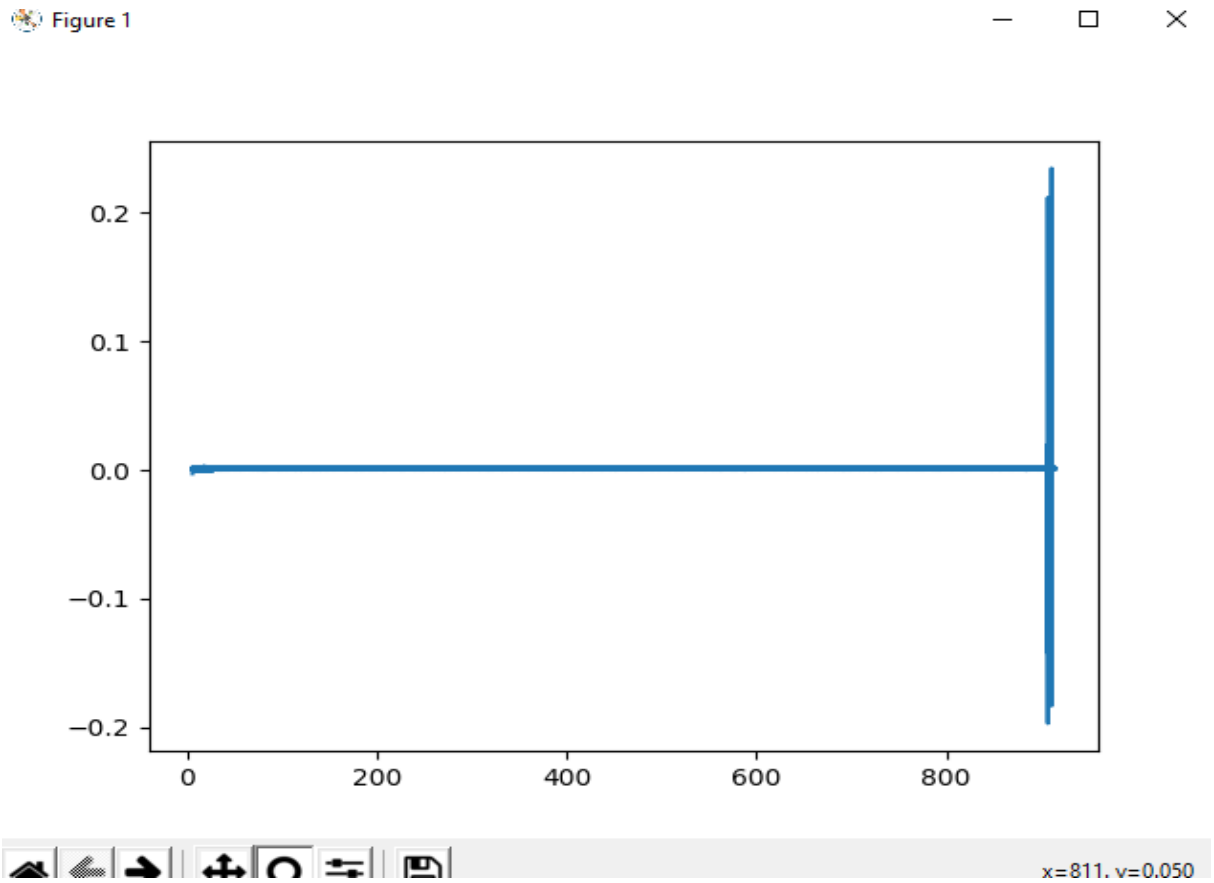


Figura EMG ingrandendo la parte finale:

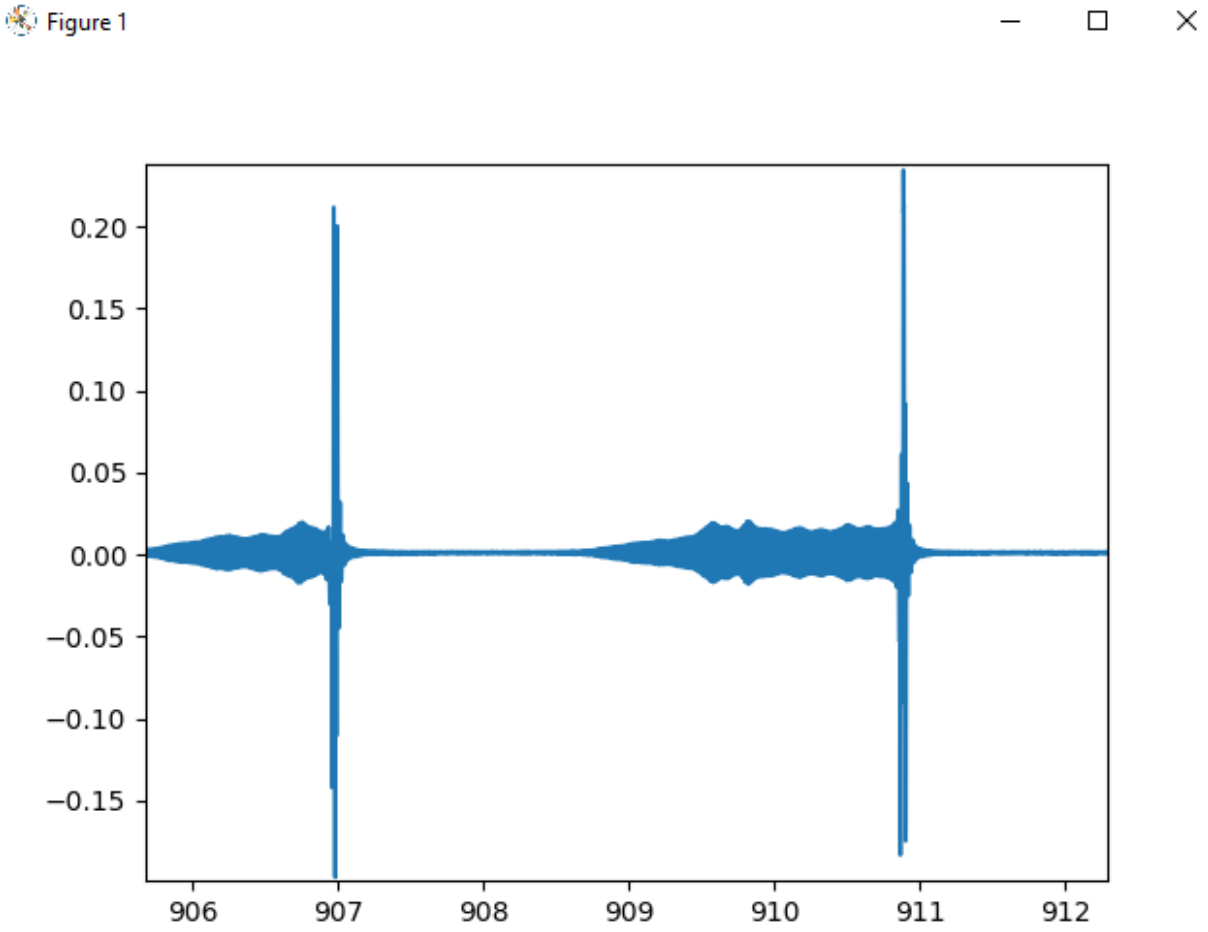


Grafico fondo scala lineare giroscopio:

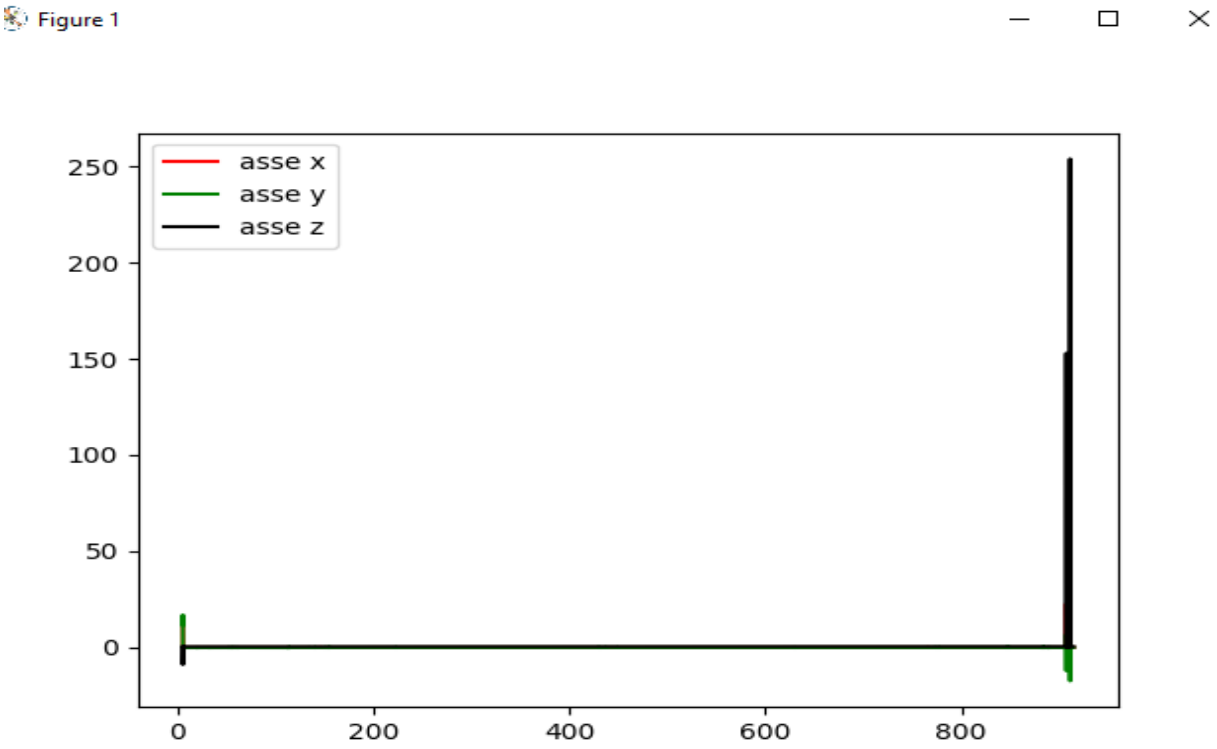


Figura fondo scala lineare giroscopio ingrandendo la parte finale:

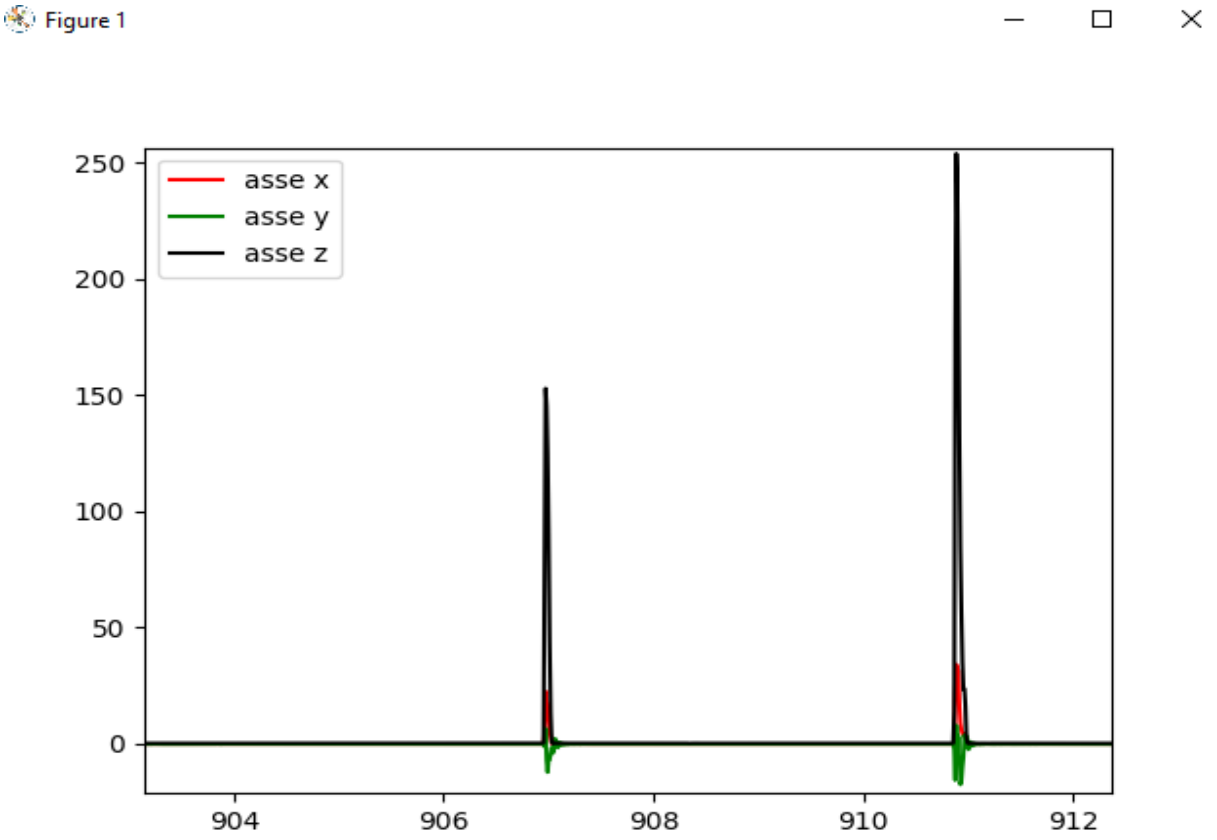
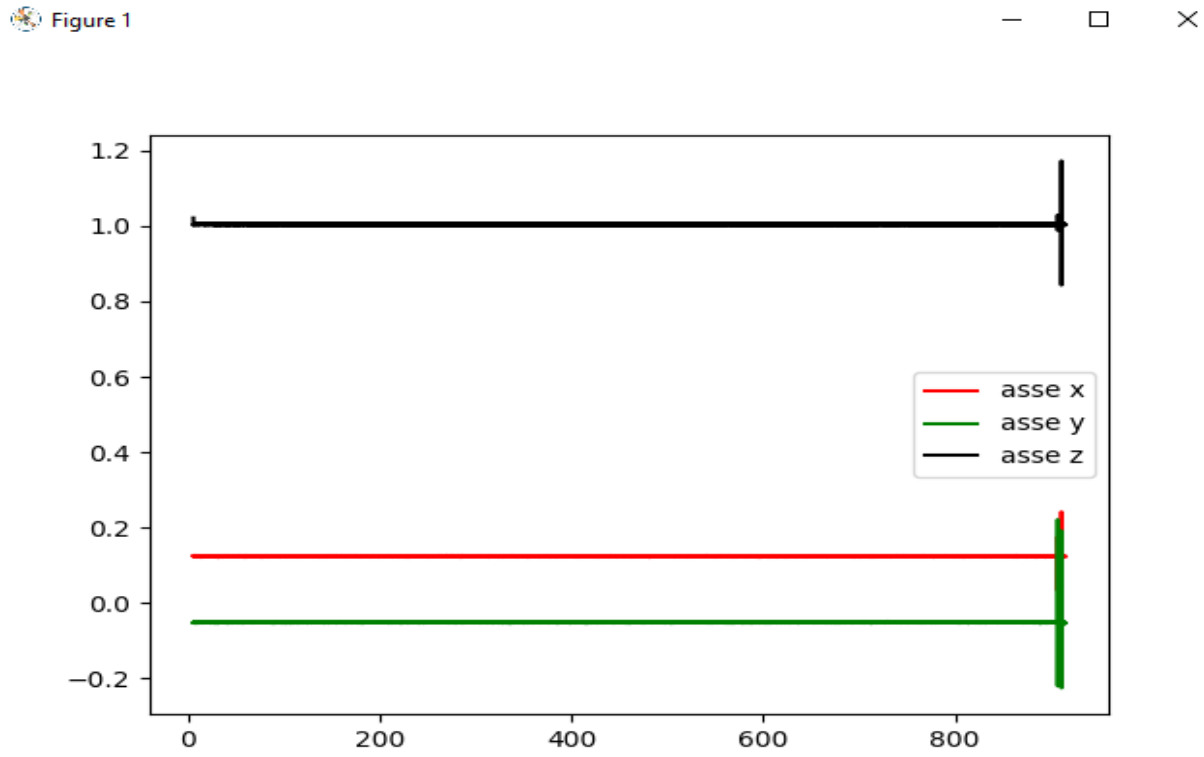
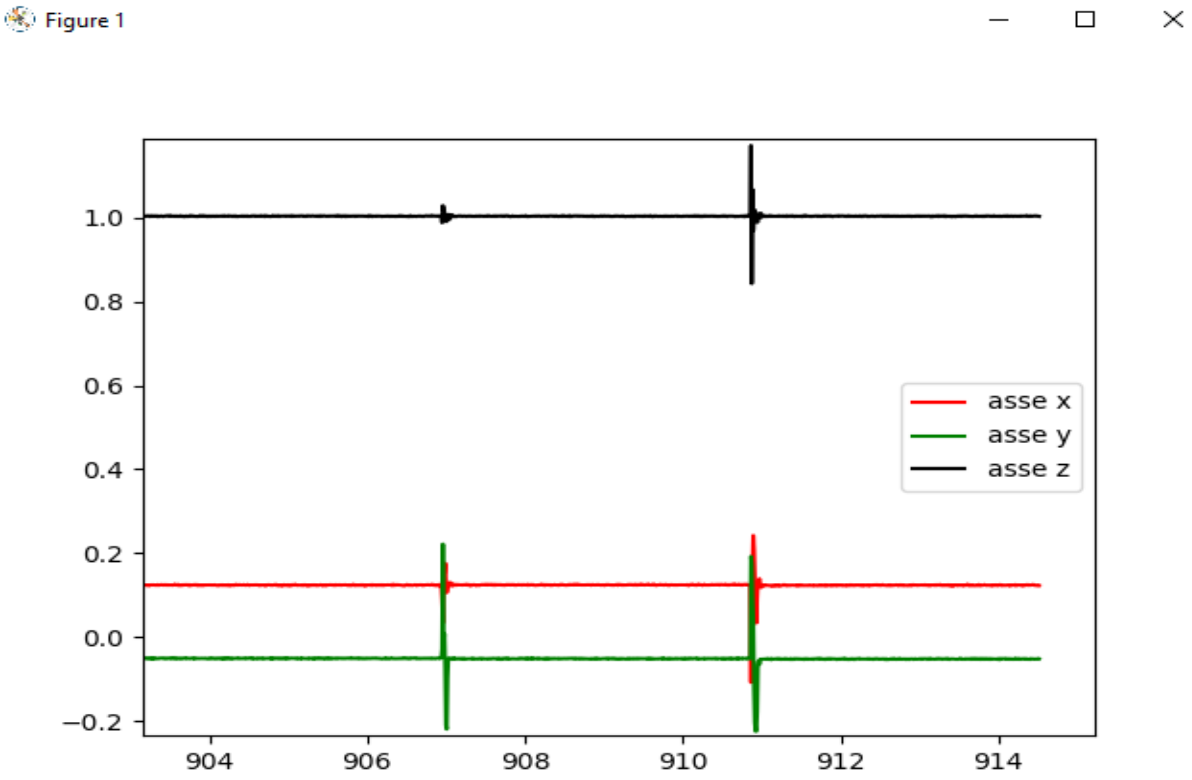


Grafico fondo scala angolare giroscopio:



Ingrandimento dell'ultima parte del grafico riportato sopra:



8. CONCLUSIONI E SVILUPPI FUTURI

È stato creato un programma che esegue le operazioni di immissione dei dati di una persona manualmente attraverso l'interfaccia, ed inoltre prende dei dati di misure da un file di testo li carica nel server in modo che si possano salvare queste informazioni in maniere permanente. Oltre, a questo l'interfaccia salva sempre sul server un file binario che contiene tutti i dati delle misure ed altre informazioni di nostro interesse ed inoltre grazie a questa interfaccia è possibile vedere l'andamento grafico dei dati presi sempre da un file di testo dell'EMG e del giroscopio.

Si può concludere che il programma funziona abbastanza bene infatti dopo varie prove fatte il programma risponde molto bene e qualora si presentassero dei problemi di inserimento riguardo a qualche ID sbagliato sull'interfaccia compare una finestra di errore che ti dice il possibile inserimento sbagliato.

Il database è stato scelto in modo ottimale infatti risponde in modo eccellente alle nostre richieste ed inoltre i dati vengono salvati correttamente.

Oltre a ciò, se si volesse sviluppare ulteriormente il progetto si potrebbe aggiungere un'altra relazione manualmente nel database. Dopo che è stata aggiunta la relazione nel programma prima dell'interfaccia che abbiamo adesso si potrebbe creare un'interfaccia di accesso per l'utente e attraverso quella si avrebbe anche una protezione in più per i dati delle persone infatti creando delle credenziali garantiamo la sicurezza per le persone in modo che i loro dati personali rimangono protetti.

9. Bibliografia

[1] G. Biagetti, P. Crippa, L. Falaschetti e C. Turchetti, «A Multi-Channel Electromyography, Electrocardiography and Inertial Wireless Sensor Module Using Bluetooth Low-Energy,» 4 giugno 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/6/934/htm. [Consultato il giorno 2020].

[2] RFID.it, «Cos'è il Bluetooth Low Energy,» [Online]. Available: https://rfid.it/cos-e-bluetooth-low-energy/.

[3] C. D. Ferrara, «PyCharm: Python IDE per i professionisti,» 19 marzo 2018. [Online]. Available: https://www.html.it/19/03/2018/pycharm-python-ide-per-i-professionisti/.

[4] T. P. G. D. Group, «PostgreSQL 7.4.30 Documentation,» [Online]. Available: https://www.postgresql.org/docs/7.4/index.html.

[5] T. P. G. D. Group, «Storing Binary Data,» [Online]. Available: https://www.postgresql.org/docs/7.4/jdbc-binary-data.html.

[6] V. Ndlovu, «Working With Files in Python,» [Online]. Available: https://realpython.com/working-with-files-in-python/.

[7] K. Lioy, Python: La guida per imparare a programmare, Independently published, 2019.

[8] M. Buttu, Programmare con Python: Guida completa, Edizioni LSWR, 2014.

[9] C. S. Horstmann e R. D. Necaise, Concetti di informatica e fondamenti di Python, Apogeo Education, 2019.

[10] K. A. Lambert, Programmazione in Python, Maggioli Editore, 2018.

[11] PYnative, «Python PostgreSQL Tutorial Using Psycopg2,» [Online]. Available: https://pynative.com/python-postgresql-tutorial/.

[12] T. Point, «PostgreSQL Tutorial,» [Online]. Available: https://www.tutorialspoint.com/postgresql/index.htm.

10. Ringraziamenti

Questo momento tanto atteso è arrivato, ed è ora di fare i ringraziamenti.

In primi vorrei ringraziare i miei genitori che mi hanno cresciuto e che mi hanno aiutato in modo loro nonostante le difficoltà che ci sono state durante questi anni; un grande ringraziamento va a mio padre, che è immigrato in Italia per una vita migliore quando io sono nato e con tante difficoltà ha portato me e mia madre qui dove stiamo adesso, per quello fa tutti i giorni per noi; a mia madre che in tenera età mi ha cresciuto da sola e poi quando ci siamo trasferiti in Italia c'è sempre stata in ogni difficoltà della mia vita. A loro dedico questo traguardo che sarà uno dei tanti (si spera) che raggiungerò nella vita.

Ringrazio la mia ragazza Erinda che in ogni momento di questa triennale c'è stata nei bei momenti e in quelli brutti e mi ha sempre cercato di aiutare e mi ha supportato ad andare avanti ed a raggiungere questo obiettivo il prima possibile.

A mio cugino Alessio che è sempre stato presente nei momenti di difficoltà.

Vorrei ringraziare inoltre i miei amici, chi conosciuto prima chi dopo di esserci stati in questo periodo e di aver passato grazie a loro dei momenti stupendi e delle serate memorabili e di aver fatto in modo che questo periodo di università non fosse così pesante : Aldo, Bruno, Disi, Romario, Marjo, Altin, Claudio, Denaldo, Gazi, Danoci, Ronaldo, Daniel, Xhonni, Tony, Mateo, Tosi, Poci, Giacinto.

E un grazie anche alle persone che non ho nominato ma che in un modo o nell'altro ci sono stati in questo lungo percorso.