Creating an Accelerometer Data Stream With Polar Device

Plotting a live Accelerometer data-stream with Python using Polar Fitness Tracker

Pareeknikhil
Jan 3 · 7 min read

Introduction

With a plethora of Biofeedback apps emerging in the health and fitness space, wearable biosensing devices are becoming pivotal in determining the “Active” state of the users. The active state of the user could be defined as the mood; between happy-sad-neutral or stress state; between relaxed-stressed-neutral, or even health state; healthy-risky, depending upon the scope of the application being developed. But a common physiological indicator that pins all the aforementioned use-cases is *cardiovascular data*. And this article aims to give you a brief summary on filtering raw

cardiovascular data from BCI/Fitness tracking devices such as *Polar H10* and *OpenBCI-Cyton* into noise-free data.

Most of the Biofeedback systems rely on two key physiological measurements as indicators of the active state of the user: Cardiovascular activity and Breathing capacity. These signals could further be split into specific metrics such as *Heart Rate Variability* (HRV), *RR interval*, *Tidal Volume*, etc. but we would stick with processing raw cardiovascular activity here (For a detailed read on indicators — here). And *Electrocardiograph (ECG)* is the savior here in providing the most accurate electrical activity of the heart in real-time as compared to techniques such as *Photoplethysmography (PPG)*. Essentially ECG signals convey information about the structure and function of the heart and further, we are specifically interested in capturing a biomedical signal — the *QRS complex* of the heart which is useful in diagnosing cardiac arrhythmias, conduction abnormalities, ventricular hypertrophy, myocardial infarction, electrolyte derangements, and other disease states.

One-stop API for all Polar devices: Polar SDK

> Polar mobile SDK enables you to read and interpret live data from Polar heart rate sensors, including ECG data, acceleration data and heart rate broadcast. — Polar SDK official documentation

The SDK will work with 3 devices that Polar has developed so far out of which only two have the capability to stream accelerometer data. The following three elucidate the features of each of these devices in descending order of cost and capabilities:

1. POLAR H10 — Heart rate sensor

The first one is Polar H10 which is globally recognized for its accuracy and high quality. If you are looking for an accurate heart rate/ accelerometer sensor that could be worn on the chest, this is the one !! Also for the purpose of demonstration, all the visualization and the code that is used in this tutorial have been developed on Polar H10. Following are its capabilities as listed on the official PolarH10 page.

- Electrocardiography & RR intervals
- Heart rate broadcast — Heart Beat

- Acceleration
- Sensor memory read
- Device ID

2. POLAR OH1 — Optical heart rate sensor

The second one is the Optical Heart Rate Sensor, OH1. The advantage of the optical technology is its ability to be worn on the arm as well as on the temple along with the chest, therefore one could use the device while swimming as well. The code and visualization showcased in this tutorial would work very well with the Polar OH1 as well. Following are its capabilities as listed on the official Polar OH1 page.

- Photoplethysmogram (PPG)
- Heart rate broadcast — Heart Beat
- Acceleration
- Sensor memory read
- Device ID

3. POLAR H9 — Heart rate sensor

Similar to Polar H10, this model is a chest strap Heart Rate Sensor but without the Accelerometer or ECG data broadcasting capabilities and therefore is not eligible for the purpose of live Accelerometer visualization in this tutorial. Following are its capabilities as listed on the official Polar H9 page.

- RR intervals
- Heart rate broadcast — Heart Beat
- Device ID

Data Format Description for Polar SDK API — Data Acquisition

The data-format and types are different for different devices produced by Polar and we would be referencing the same from the official Polar documentation for the same

in this tutorial:

1. **Polar H10** heart rate sensor available data types(From version 3.0.35 onwards):

> Heart rate as beats per minute.
>
> RR Interval in ms and 1/1024 format.
>
> Electrocardiography (ECG) data in μV.
>
> The default epoch for the timestamp is 1.1.2000.
>
> Accelerometer data with sample rates of 25Hz, 50Hz, 100Hz, and 200Hz and range of 2G, 4G, and 8G (2G is sufficient if you would like to pick up respiration data from the chest).
>
> Axis specific acceleration data in mG.
>
> Recording supports RR, HR with one-second sample time, or HR with five-second sample time.
>
> List, read, and remove for stored internal recording (sensor supports only one recording at the time). — Source: Official Polar Github repo

2. **Polar OH1** Optical heart rate sensor available data types(From version 2.0.8 onwards):

> The **Polar OH1** is a rechargeable device that measures the user's heart rate with LED technology.
>
> Heart rate as beats per minute.
>
> Photoplethysmography (PPG) values.
>
> PP interval (milliseconds) representing cardiac pulse-to-pulse interval extracted from the PPG signal.
>
> Accelerometer data with a sample rate of 50Hz and a range of 8G.
>
> Axis specific acceleration data in mG.

> List, read, and remove stored exercise. Recording of exercise requires that the sensor is registered to Polar Flow account. — Source: Official Polar Github repo

Let's get coding !!

A Python-based application for visualizing Acceleration with Polar-H10 tracker in real-time is available here. The communication with the Polar device is via a Bluetooth service protocol, knows as **GATT (Generic Attribute Profile)**. It defines the way that two Bluetooth Low Energy devices transfer data back and forth using concepts called *Services and Characteristics*.

Furthermore, along with the *GATT protocol*, a special protocol — **Accelerometer GATT Service Protocol** that the device manufacturer follows to allow users to obtain specific responses such as Acceleration, battery level, Manufacturing ID, etc. via an API would be used in this tutorial. For each of these features such as Acceleration, battery level, etc. we will use predefined UUID (Universal Unique Identifier) mapping and write it on the device to obtain real-time data.

Let's get started ❤

We will use the **Bleak** library for the Bluetooth Low Energy (BLE) connection. And also A**syncio** for the asynchronous programming of the application.

```
import asyncio
import math
import os
import signal
import sys
import time
import pandas as pd
from bleak import BleakClient
from bleak.uuids
import uuid16_dict
import matplotlib.pyplot as plt
import matplotlib
```

We will further define the UUID's for all the features we want to extract from the Polar device (Source: Polar Tech doc — refer to this if you would like to change the

sampling frequency or measurement range):

```
""" Predefined UUID (Universal Unique Identifier) mapping are based
on Heart Rate GATT service Protocol that most Fitness/Heart Rate
device manufacturer follow (Polar H10 in this case) to obtain a
specific response input from the device acting as an API """

uuid16_dict = {v: k for k, v in uuid16_dict.items()}


## This is the device MAC ID, please update with your device ID
ADDRESS = "D4:52:48:88:EA:04"

## UUID for model number ##
MODEL_NBR_UUID = "0000{0:x}-0000-1000-8000-00805f9b34fb".format(
         uuid16_dict.get("Model Number String"))

## UUID for manufacturer name ##
MANUFACTURER_NAME_UUID = "0000{0:x}-0000-1000-8000-
00805f9b34fb".format(uuid16_dict.get("Manufacturer Name String"))

## UUID for battery level ##

BATTERY_LEVEL_UUID = "0000{0:x}-0000-1000-8000-00805f9b34fb".format(
       uuid16_dict.get("Battery Level"))

## UUID for connection establsihment with device ##
PMD_SERVICE = "FB005C80-02E7-F387-1CAD-8ACD2D8DF0C8"

## UUID for Request of stream settings ##
PMD_CONTROL = "FB005C81-02E7-F387-1CAD-8ACD2D8DF0C8"

## UUID for Request of start stream ##
PMD_DATA = "FB005C82-02E7-F387-1CAD-8ACD2D8DF0C8"

## UUID for Request of ACC Stream ##
ACC_WRITE = bytearray(

[

   0x02,

   0x02,

   0x00,

   0x01,

   0xC8,

   0x00,
```

```
    0x01,

    0x01,

    0x10,

    0x00,

    0x02,

    0x01,

    0x08,

    0x00,

  ]

)

## For Plolar H10  sampling frequency ##
ACC_SAMPLING_FREQ = 200

## Session resources

acc_session_data = []
acc_session_time = []
```

The data from the Polar device is received as a stream and in hexadecimal bytes and we would require to decode and build a streaming service for the same in decimal bytes. The following functions do the same:

```
def data_conv(sender, data):
     if data[0] == 0x02:
        timestamp = convert_to_unsigned_long(data, 1, 8)
        frame_type = data[9]
        resolution = (frame_type + 1) * 8
        step = math.ceil(resolution / 8.0)
        samples = data[10:]
        offset = 0

        while offset < len(samples):
           x = convert_array_to_signed_int(samples, offset, step)
           offset += step
           y = convert_array_to_signed_int(samples, offset, step)
           offset += step
           z = convert_array_to_signed_int(samples, offset, step)
           offset += step
```

```
acc_session_data.extend([[x, y, z]])
acc_session_time.extend([timestamp])
```

Now comes the last bit to write an async method that would write and assemble all necessary UUID's on the device and open the Accelerometer stream from Polar. This is a fairly simple task, but a lengthy one, and it would be convenient for the readers here to direct you to my Github repo for a complete working application. On running the **main.py**, one could replicate the plot shown below.

Fig — 2: Working ACC stream with Polar-H10

Want to learn more?

Building a live data-stream and filtering signals for Polar devices are vital in building real-time biofeedback applications and are fairly simple tasks with a little time and sufficient documentation. Luckily we have both !!

- How to create a real-time data-stream with Polar devices — ECG
- How to create a real-time data-stream with OpenBCI devices — ECG
- Getting the beat right !! — Processing ECG data with band-pass filters

References

1. Polar documentation, Polar docs, Github

2. N.Pareek, Getting the beat right !! (2021), Towards Data Science

3. Mohamed Elgendi,Carlo Menon, Assessing Anxiety Disorders Using Wearable Devices: Challenges and Future Directions (2019) ,Brain Sciences

4. Inma Mohino-Herranz, Roberto Gil-Pita, Manuel Rosa-Zurera, Fernando Seoane, Activity Recognition Using Wearable Physiological Measurements: Selection of Features from a Comprehensive Literature Study (2019) ,NCBI

5. National Centre for Adaptive Neurotechnologies (NCAN), Contributions:OpenBCI Module

6. Adafruit, Introductin to Bluetooth Low Energy

7. N.Pareek, Creating an ECG Data Stream with Polar device (2020), Towards Data Science

Ecg Respiratory Heart Rate Monitor Polar Fitness Tracker Datastream