Datablad Tacx Trainer

In order to be able to emulate input from a Tacx Trainer, the upvoted python library [PyCycling](https://pypi.org/project/pycycling/) is used. This library is used as an easy python SDK for connecting to and interpreting the data from the Tacx trainer in order to omit interacting directly with the overly complicated BLE protocol. As this Tacx trainer as an input module is only used for demonstrative purposes in this project, we deemed it unnecessary to spend a lot of time integrating directly into the BLE protocol.

The Pycycling TacxTrainerControl (TTC) is a provider which lets you interact with the Tacx trainer. This includes configuring the settings on the trainer (out of scope for this project), as well as receiving data from the trainer. The TTC provider takes a BleakClient, which is a python client which handles the connection to the BLE client. It is important to note that the BleakClient is only handling the connection and reconnection to the BLE protocol, hence all the data read from BLE is passed rawly to the TTC provider to be interpreted and serialized.

As we have no access to a Trax Trainer, we will inject a compatible custom BleakClient, to simulate the receival of data from a BLE device (Tacx trainer). The CustomBleakClient (CBC) should be 100% compatible with the TTC, as we want the setup to be totally interchangeable with a real Tacx Trainer. To achieve this, the CBC mush support the all the functionalities from the BleakClient, which are used by the TTC, as well as delivering the data payload in a supported format.

The features to support are:

* The use of with (\_\_aenter\_\_ & \_\_aexit\_\_) to simulate the connection of the BLE.
* The method is\_connected
* The method write\_gatt\_char (to simulate writing back to the BLE)
* The method start\_notify (starts the listener from the BLE. This takes a callback from the TTC, to which payloads are passed)

## The heavy lifting is the data output:

(could be investigated further)

In order to know the expected format for the data, the TCC has been analyzed for the purpose of reverse engineering the data flow. Especially methods like \_fec\_nitification\_handler and \_general\_fe\_data\_page\_hander has been analyzed in depth, doing the reverse engineering process. When the data is received from a proper BleakClient, the data was of the type byte array. However, in python everything is treated as an object, hence we were able to create a custom class with the exact same behavior as the byte array received from the trainer.  
Et billede, der indeholder skærmbillede, tekst, software, Multimediesoftware

Automatisk genereret beskrivelse

For an in-debt description of the data, refer to the screenshot.