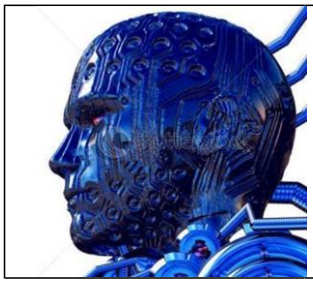


*TRS

Tecnologia, Redes e Sociedade

e-planning | networks | e-learning | e-government

Relatório Interno TRS 03/2020

Título

Aquisição de dados para monitorizar a qualidade do sono

Autor(es)

Francisco Domingues, UFP

Luís Borges Gouveia, UFP

Mês, Ano

Maio, 2020

Local de presença Web *http://tecnologiaredesesociedade.wordpress.com*

Repositório de trabalho científico *trs *http://bdigital.ufp.pt/handle/10284/3787*

Universidade Fernando Pessoa

Praça 9 de Abril, 349

4249-004 Porto, Portugal

Tabela de Conteúdos

Aquisição de dados para monitorizar a qualidade do sono

Francisco Domingues, Luís Borges Gouveia

Resumo

Este documento tem por objetivo dar reporte da atividade realizada no contexto do estágio curricular no âmbito do Mestrado em Computação Móvel, da Faculdade de Ciência e Tecnologia, da Universidade Fernando Pessoa. O estágio decorreu nas instalações da Universidade Fernando Pessoa localizadas na rua Delfim Maia, Porto, no polo principal da universidade, gabinete 19.

O trabalho realizado está associado com o uso crescente de equipamentos tecnológicos e os impactos que estes causam na capacidade de dormir – sono. Existem tecnologias de informação e comunicação que, nos dias de hoje, nos permitem realizar estudos que aprofundem o conhecimento sobre a fisiologia humana e sobre a qualidade de sono. Em especial, qual o impacto das TIC no estudo do sono e dos distúrbios do sono e os seus possíveis efeitos. Assim, este trabalho estuda dispositivos de captura de dados fisiológicos e forma de o realizar, avaliando opções disponíveis com dispositivos de baixo custo.

Palavras-Chave: sinais fisiológicos, dispositivos de aquisição de sinal, equipamentos eletrónicos; monitorização do sono.

1. Introdução

Este relatório apresenta um estudo sobre estratégias de aquisição de dados com o objetivo posterior de estudar os efeitos que o uso intensivo de dispositivos eletrónicos pode causar na qualidade do sono e os distúrbios associados. Propõe uma análise sobre potenciais estratégias para monitorizar os sinais associados com o sono e o que está envolvido de forma a mitigar os custos e dificuldades de aquisição de dados associados ao problema. O uso crescente de equipamentos tecnológicos causa problemas na qualidade do sono, mas também proporciona formas de o monitorizar e melhor compreender como se pode mitigar o problema.

Deste modo, foi explorada a utilização de dispositivos que permitam a monitorização e o cruzamento de dados biométricos, com a finalidade de obter a informação acerca dos contextos e circunstâncias associadas com as perturbações de sono e que soluções se podem aplicar para mitigar estes efeitos.

2. Objetivos gerais do trabalho

Investigar sobre uma uma base de trabalho para o uso e exploração de dispositivos para monitorização dos diferentes sinais gerados pelo corpo (fisiológicos), e que servem de dados para as análises de padrões durante o sono. Testar e propor uma abordagem tecnológica para o efeito.

2.1 Objetivos específicos

São objetivos específicos do trabalho a realizar os seguintes:

- Descrever uma seleção dos diferentes dispositivos de monitorização do sono, existentes;
- Fazer uma descrição detalhada do OpenBCI CYTON;
- Fazer uma descrição detalhada da FITBIT Band AltaHR;
- Realizar testes de operação dos dispositivos;
- Obter (capturar) dados dos dispositivos;
- Apresentar e organizar os dados recolhidos

3. Atividades desenvolvidas

O projeto teve a duração de 200 horas de trabalho com início a 17 de setembro e término a 3 de novembro de 2018. O presente relatório descreve o trbalho realizado nesse momento, sendo o seu conteúdo libertado a maio de 2020, no contexto de presente relatório interno.

Durante a realização do estágio foi realizada uma revisão dos dispositivos de monitorização existentes no mercado, com a finalidade de conhecer a relação de eficiência versus custo, assim como realizado um levantamento das características de cada uma das soluções identificadas e dos tipos de dados que se podem obter.

Entre estes dispositivos, são os monitores de dados biométricos (cerebral), os mais comuns, são os dispositivos de Eletroencefalografia (EEG). Entre a multiplicidades deste

dispositivos existentes no mercado, encontra-se o Cyton de OpenBCI, que foi o dispositivo selecionado para este trabalho.

Os dispositivos de EEG constituem uma técnica muito usada na área da saúde, que hoje está a ter muita relevância nas áreas da interação humano-computador, por meio do desenvolvimento da neurociência e da capacidade de controlar dispositivos ou envolventes (de ambiente) através do reconhecimento de padrões do pensamento ou movimento dos músculos à volta do sensor, através da captura e medida de ondas geradas pelo cérebro.

Este aumento na procura e uso destes dispositivos tem vindo a criar um novo mercado no desenvolvimento de dispositivos EEG portáteis e que fornecem uma solução de qualidade adequada para as necessidades específicas dos utilizadores e de aplicações como a que se pretende, de monitorização do sono. A sua diversidade, quer nos dados que são capazes de adquirir, quer no número de sensores, permite a escolha de uma grande variedade de dispositivos, com diferentes caraterísticas e preços.

Existem dispositivos com funcionalidades e usos limitados a algumas áreas específicas por causa do número de sensores como é o caso do NeuroSky. Este é um dispositivo que se encontra entre os da gama mais económica por possuir um único sensor na parte superior da frente e um sensor de referência na zona da orelha. Este dispositivo está pensado para fazer uma monitorização dos estados de meditação que o utilizador pode atingir, assim como, alguns exercícios para melhorar estes estados da mente e conseguir desenvolver ou melhorar estas capacidades (meditação).

A novidade deste desenvolvimento é o software universal de medição, coleta de dados, processamento de dados e visualização, que pode ser a base de várias outras pesquisas. O desenvolvimento destes programas permite que os utilizadores investiguem como os sinais das ondas cerebrais – medidos pelo sistema de EEG – se alternam no tempo e como eles dependem das mudanças na atividade cerebral.

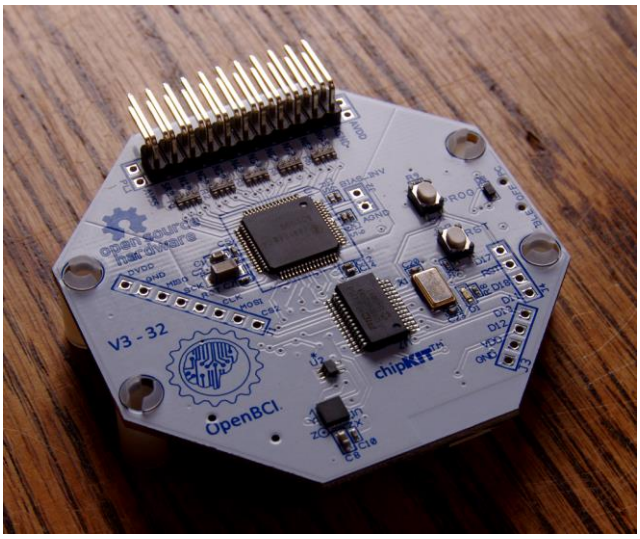
Numa gama mais avançada encontra-se a companhia Emotiv, que tem diferentes dispositivos. É reconhecida pela sua flexibilidade e facilidade de uso: o Emotiv Epoc, qual é o seu dispositivo intermédio, com 14 sensores e duas referências, é um dispositivo BCI (*Brain Computer Interaction*) sem fio e com custo médio. Possui ainda um menor tempo de

preparação para o seu uso e possibilita a realização de uma pesquisa escalável e contextual do cérebro humano, permitindo o desenvolvimento de aplicações. O EPOC Emotiv não tem muito código aberto e tem algumas limitações na programação do Matlab (programa complementar para a apresentação e visualização dos resultados). No entanto, como forma de superar essas dificuldades, para o processamento de sinal *off-line*, o Emotiv TestBench (software fornecido pela Emotiv) é usado para gravar e converter os dados do EEG para o formato que o Matlab pode processar.

O Matlab é um programa de tratamento de dados, de processamento matemático e de visualização (*https://www.mathworks.com/products/matlab.html*). Existe uma alternativa compatível de código aberto e de software livre, designada por Octave (*https://www.gnu.org/software/octave/*). Ambas estas peças de software permitem o estudo dos padrões, com uma rápida configuração. Existe no entanto, uma dificuldade na conexão direta, pelo qual o BCI2000 é empregue para conetar o Emotiv. Este permite o seu uso no controlo de dispositivos externos, como dispositivos de rádio controlo ou algumas ações através do telemóvel. O dispositivo também reconhece o movimento dos olhos ou dos músculos faciais, os quais podem ser dados importantes para o reconhecimento dos estados emocionais (bem como os níveis ou fases do sono).

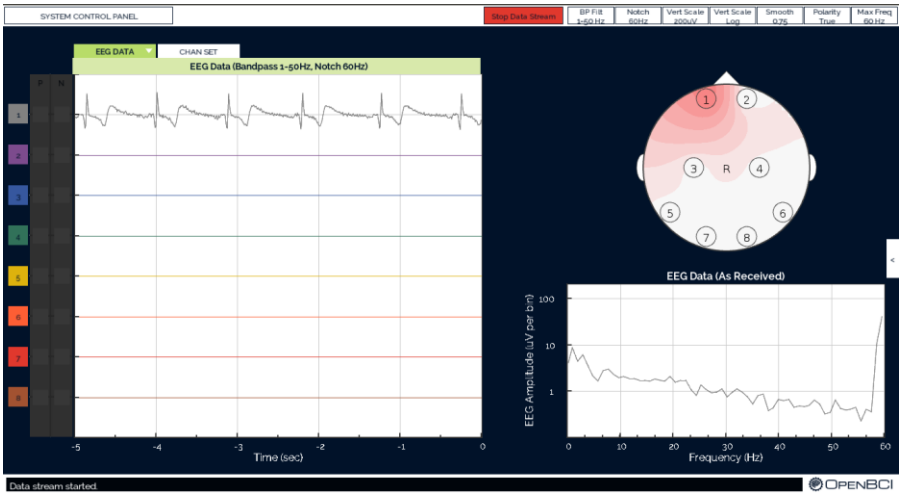
Dentro desta gama de dispositivos como um projeto livre de desenvolvimento, temos o OpenBCI, que também possui vários produtos, entre os que se encontra o Cyton, que é um dispositivo com 8 biossensores e duas referências, que permitem obter dados de EEG, para conhecer os dados da atividade cerebral, Eletromiografia (EMG) de modo a recolher informação sobre a atividade muscular, além do batimento cardíaco, movimento dos olhos, entre outros dados biométricos.

A figura seguinte ilustra o hardware associado com o OpenBCI (a página oficial do projeto é *https://openbci.com/* e as imagens foram retiradas de *https://en.wikipedia.org/wiki/OpenBCI*). Este dispositivo é utilizado para medir a atividade elétrica produzida pelo cérebro.



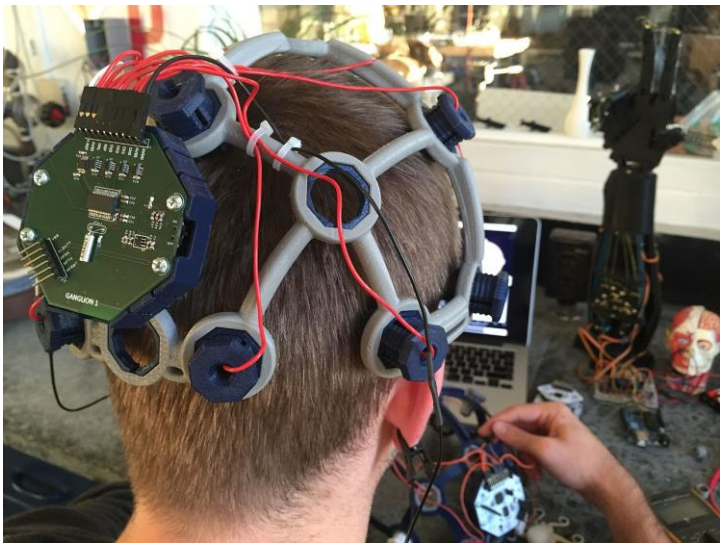
Uma foto do OpenBCI (hardware)

A figura seguinte mostra o painel de controlo dos resultados obtidos pelo OpenBCI, utilizando o interface associado com o produto.

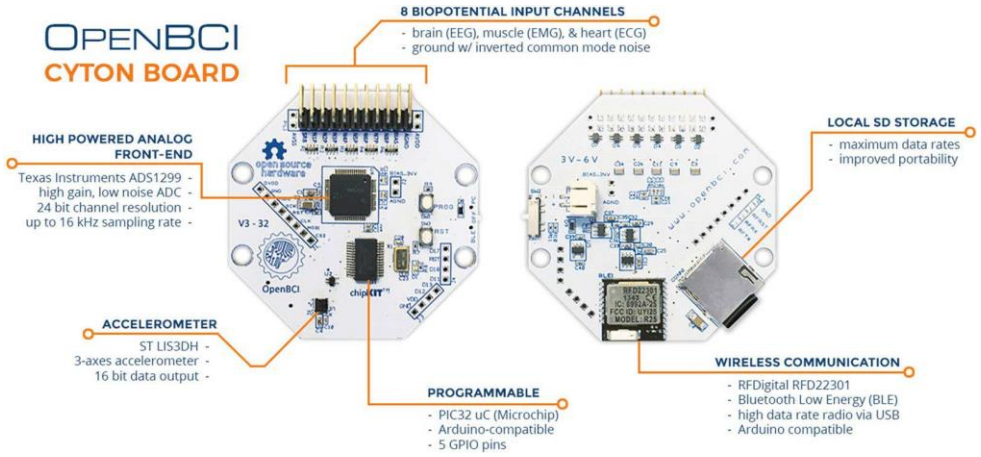


OpenBCI (software)

Adicionalmente, os sensores de aquisição da atividade cerebral do OpenBCI são exemplificados pelo dispositivo que integra a placa com os suportes e sensores a aplicar na cabeça do utilizar a monitorizar, conforme ilustrado na figura seguinte (os suportes podem ser impressos numa impressora 3D, existindo os projetos para o efeito):



Por último, é apresentado o esquima que resume o hardware associado com o OpenBCI cyton board, o hardware de aquisição de dados com base no Arduino, conforme esquema seguinte:



A atividade motora é considerada mais significativa em frequências na gama de 76 a 100 Hz (Miller, et al., 2010), enquanto o pensamento ativo e a resolução de problemas são melhor capturados em gamas de 13 a 30 Hz (West Pomeranian University of Technology, 2014). Medindo apenas as amplitudes nessas bandas de frequência, os dados podem ser adaptados ao esquema de controlo específico. Sabendo que o Cyton e capaz de

obter múltiplas frequências, é possivel explorar este tipo de dispositivo para se perceber o estado de sono.

Outros dispositivos utilizados na monitorização da atividade corporal são as bandas do pulso, as quais funcionam com um sensor PPG (*Photoplethysmography*). O sensor é usado para medir o batimento cardíaco pela medição do volume de sangue que passa através dos vasos. Funciona pela iluminação da superfície da pele. Diferentes quantidades dessa luz são absorvidas pelo sangue e pelos tecidos circundantes. A luz não absorvida é refletida para o detetor. Com emissão na cor verde ou vermelha, possibilita a medição do batimento no pulso.

Para efeitos do estágio, foi escolhido como dispositivo do pulso a pulseira Fitbit Alta HR (*https://www.fitbit.com/*), por possuir no seu software a função de monitoramento do sono, que deteta automaticamente o momento em que o utilizador acorda ou dorme. Este sistema é capaz de reconhecer a profundidade do sono, a luz do ambiente e outros detalhes associados.

A figura seguinte apresenta a pulseira utilizada, cujas caraterísticas técnicas podem ser consultadas em *https://help.fitbit.com/?p=alta_hr*.

Sendo fácil de aceder a seus dados, existem facilidades para relacionar os seus dados, com os obtidos pelo Cyton de OpenBCI.

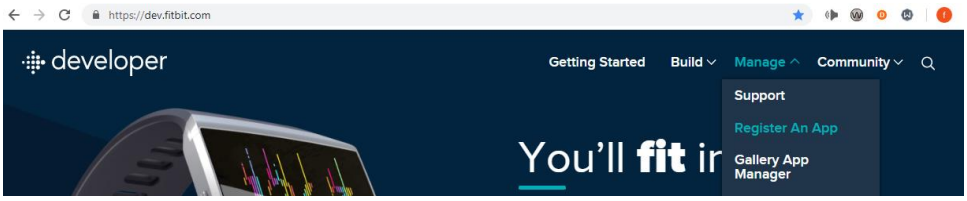
A tabela seguinte lista as funcionalidades em aquisição de sinal, que os dois dispositivos utilizados possuem:

	OpenBCI	Fitbit Band
EEG	✓	✗
Rastreamento do sono	✗	✓
Alarme inteligente	✗	✓
Suporte de sono calmo	✗	✗
Melhoria do sono	✗	✓
Rastreamento de saúde	✗	✓
Batimento cardíaco	✓	✓
Temperatura corporal	✓	✗

Para a obtenção destes dados se tem de fazer a recolha dos dados de ambos dispositivos, em separado, para serem armazenaos de modo a permitir o seu uso ao mesmo tempo.

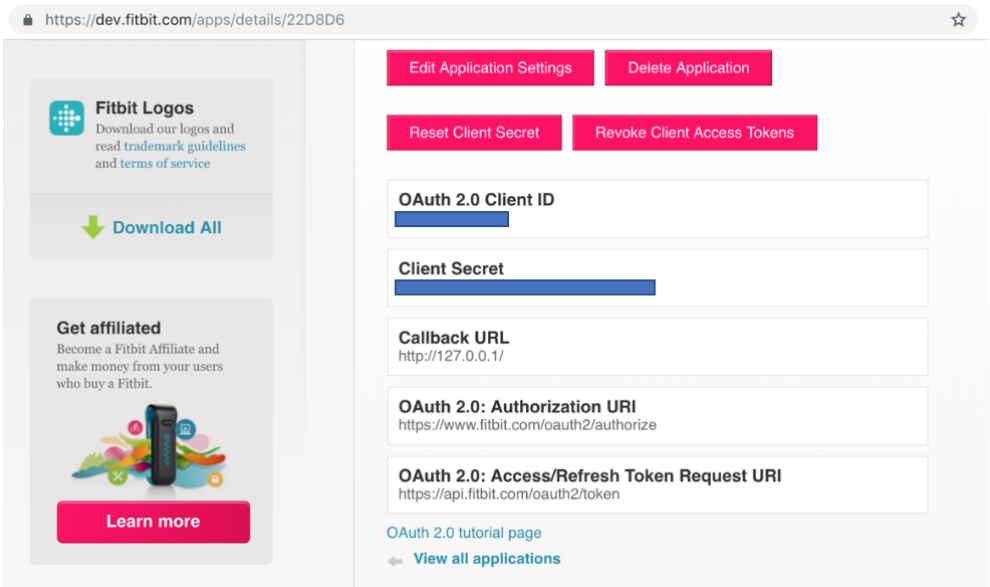
Foi preciso desenvolver um script em Python que permite obter os dados que se vão armazenar numa mesma matriz, em um mesmo intervalo de tempo, para acesso e análise posterior.

Para realizar este script e obter os dados da Fitbit Band, foi preciso aceder ao portal Web de desenvolvimento da marca, *http://dev.fitbit.com*, com as credenciais de utilizador de Fitbit que também permite fazer autenticação com a conta do Google, seleccionando a área "*Register An App*".



Ao entrar no serviço de desenvolvimento, é preciso carregar no separador de registar uma aplicação e preencher os dados da aplicação a criar. Assim, é possível obter do servidor de Fitbit os dados de autorização, como a identificação (ID) do cliente, segredo do

cliente, Access e chave de refrescamento. Em conjunto, estes elementos vão permitir ao script obter os dados que estão a ser recolhidos em tempo real – captura de sinal.



Para a obtenção dos dados é importante utilizar a biblioteca Fitbit para Python que pode ser descarregada do github https://github.com/orcasgit/python-fitbit. Esta biblioteca possui o script Gather_keys_oauth2.py (Anexo A), que combinado no prompt com o Cliente ID, e o Secret Client, é a forma através da qual se obtém o refresh token, e o Access Token.

Para a realização deste procedimento, foi desenvolvido um outro script que utiliza o anterior, de modo a automatizar o procedimento. Estes scripts permitem obter os dados biométricos, armazenados na conta do Fitbit e convertê-los em formato csv.

No script também é preciso definir quais são os dados biométricos que vão ser descarregados, de modo a conhecer a atividade diária do objeto de estudo (a unidade carbono…) , através dos dados obtidos. Um exemplo desta atividade é ilustrado abaixo:

```
37 #get heart rate data / should be today
38 fitbit_stats2 = auth2_client.intraday_time_series('activities/heart', base_date=today2, detail_level='1sec')
39 stats2 = fitbit_stats2
```

```
75 #get sleep data / should be today
76 fitbit_stats3 = auth2_client.sleep(date='today')
77 stime_list = []
78 sval_list = []
```

Para este estudo buscou-se armazenar apenas os dados do batimento cardíaco diário e os estados ou fases do sonho registados pela pulseira. Esta informação vai ser útil devido ao potencial de cruzamento dos dados, oferecendo assim uma visão clara de, por exemplo, efeitos da luminosidade no ciclo circadiano de uma determinada pessoa e como estes podem ser mitigados.

4. Considerações finais

Com base nos trabalhos realizados, as informações obtidas pela banda de FitBit AltaHR, podem ser processadas e usadas em várias áreas de pesquisa. Por exemplo, pesquisa médica, permitindo criar aplicações ou controles em forma de avisos para os utilizadores de aplicações multimédia, ajudando a melhorar hábitos diurnos, assim como torna possível, em outras áreas, a criação de criar jogos com o intuito de desenvolver novas capacidades no utilizador através da indução de atividades, permitindo monitorizar a reação do individuo através das suas reações.

Um programa com a utilização destes dispositivos pode ser facilmente desenvolvido, sendo também possível adicionar novas funções devido à documentação existente para a criação de novas aplicações, com base no uso de diferentes dados biométricos. A API existente para o desenvolvimento com utilização dos dispositivos da FitBit, permitem processamento individuais.

Por outro lado, o sistema EEG torna possível a exploração de aplicações que podem incluir a interface do controlo com outros dispositivos. Esta tecnologia exige um conhecimento prévio do individuo para ser capaz de reconhecer os padrões das ondas cerebrais produzidas pelo qual é necessário desenvolver exercícios que permitam induzir virtualmente o utilizador, para que estas aplicações de controle de dispositivos externos, também possam ser utilizadas. Por exemplo, no controle de velocidade de um robô móvel.

É assim possível afirmar que o BCI pode ser considerado como a base para futuros desenvolvimentos de aplicações. Para além disto, tendo em conta o conhecimento de alguns destes padrões, assim como a intensidade das ondas/áreas onde estas se produzam também pode constituir a base de conhecimento necessária para melhorar outras condições do individuo em áreas como aprendizagem, ou de regularização das etapas do sonho – abrindo a oportunidade de desenvolver um treinador de sono, para a melhoria de qualidade do sono.

No âmbito do estágio e das aplicações associadas com os distúrbios do sono, foi ainda realizada a 16 de Novembro de 2018, uma apresentação deste trabalho no VI Congresso de Educação Especial, realizado na Santa Casa da Misericórdia do Porto, com o título Treinar o Sono. É Possível? (Domingues e Gouveia, 2018).

Referências

Domingues, F. e Gouveia, L. (2018). Treinar o sono. É possível? VI Congresso Ibérico Educação Especial. Educação e Inclusão na Lusofonia. 16 de novembro. Misericórdia do Porto. Porto. [Em linha] Disponível em *https://bdigital.ufp.pt/handle/10284/7041* [Consultado a 25/11/2018]

Emotiv Inc. (n.d.). Emotiv EPOC. [Em linha] Disponível em *https://www.emotiv.com/epoc/* [Consultado a 5/11/2018]

Johnson, L. (2018). Fitbit Alta HR review. [Em linha] Disponível em *https://www.techradar.com/reviews/fitbit-alta-hr* [Consultado a 5/11/2018]

Miller, K. J., Schalk, G., Fetz, E. E., den Nijs, M., Ojemann, J. G., & Rao, R. P. N. (2010). Cortical activity during motor execution, motor imagery, and imagery-based online feedback. Proceedings of the National Academy of Sciences of the United States of America, 107(9), 4430–5. *https://doi.org/10.1073/pnas.0913697107*

NeuroSky, I. (n.d.). Neurosky. [Em linha] Disponível em *https://store.neurosky.com/pages/mindwave* [Consultado a 5/11/2018]

Polley, P., & Gallati, W. G. (2018). Non-Invasive Neural Controller NON-INVASIVE NEURAL CONTROLLER, (April).

Stephen H. (2018). Code: fitbit.py. [Em linha] Disponível em *https://github.com/stephenjhsu/fitbit/blob/master/fitbit.py* [Consultado a 5/11/2018]

Team, OpenBCI. (n.d.). Documentation. [Em linha] Disponível em *http://docs.openbci.com/Hardware/02-Cyton* [Consultado a 6/11/2018]

Anexo A – Script Gather_keys_oauth2.py

```
#!/usr/bin/env python
import cherrypy
import os
import sys
import threading
import traceback
import webbrowser

from base64 import b64encode
from fitbit.api import Fitbit
from oauthlib.oauth2.rfc6749.errors import MismatchingStateError,
MissingTokenError


class OAuth2Server:
   def __init__(self, client_id, client_secret,
                redirect_uri='http://127.0.0.1:8080/'):
       """ Initialize the FitbitOauth2Client """
       self.success_html = """
           <h1>You are now authorized to access the Fitbit API!</h1>
           <br/><h3>You can close this window</h3>"""
       self.failure_html = """
           <h1>ERROR: %s</h1><br/><h3>You can close this window</h3>%s"""

       self.fitbit = Fitbit(
           client_id,
           client_secret,
           redirect_uri=redirect_uri,
           timeout=10,
       )

   def browser_authorize(self):
       """
       Open a browser to the authorization url and spool up a CherryPy
       server to accept the response
       """
       url, _ = self.fitbit.client.authorize_token_url()
       # Open the web browser in a new thread for command-line browser
support
       threading.Timer(1, webbrowser.open, args=(url,)).start()
       cherrypy.quickstart(self)

   @cherrypy.expose
   def index(self, state, code=None, error=None):
       """
       Receive a Fitbit response containing a verification code. Use the
code
       to fetch the access_token.
       """
       error = None
       if code:
           try:
```

```
                self.fitbit.client.fetch_access_token(code)
            except MissingTokenError:
                error = self._fmt_failure(
                   'Missing access token parameter.</br>Please check that'
                   'you are using the correct client_secret')
            except MismatchingStateError:
                error = self._fmt_failure('CSRF Warning! Mismatching
state')
        else:
            error = self._fmt_failure('Unknown error while
authenticating')
        # Use a thread to shutdown cherrypy so we can return HTML first
        self._shutdown_cherrypy()
        return error if error else self.success_html

    def _fmt_failure(self, message):
        tb = traceback.format_tb(sys.exc_info()[2])
        tb_html = '<pre>%s</pre>' % ('\n'.join(tb)) if tb else ''
        return self.failure_html % (message, tb_html)

    def _shutdown_cherrypy(self):
        """ Shutdown cherrypy in one second, if it's running """
        if cherrypy.engine.state == cherrypy.engine.states.STARTED:
            threading.Timer(1, cherrypy.engine.exit).start()


if __name__ == '__main__':

    if not (len(sys.argv) == 3):
        print("Arguments: client_id and client_secret")
        sys.exit(1)

    server = OAuth2Server(*sys.argv[1:])
    server.browser_authorize()

    profile = server.fitbit.user_profile_get()
    print('You are authorized to access data for the user: {}'.format(
        profile['user']['fullName']))

    print('TOKEN\n=====\n')
    for key, value in server.fitbit.client.session.token.items():
        print('{} = {}'.format(key, value))
```

Anexo B – Script SlpData.py

```
# -*- coding: utf-8 -*-
"""
@author: Utilizador
"""

import fitbit
import gather_keys_oauth2 as Oauth2
import numpy as np
import pandas as pd
import datetime

"""ClientIdNumber is a string that contains the client id number provided
by oauth2 FitBit API
CLIENT_ID = <ClientIdNumber>
CLIENT_SECRET = <ClientSecretNumber>
"""

"""for obtaining Access-token and Refresh-token"""

server = Oauth2.OAuth2Server(CLIENT_ID, CLIENT_SECRET)
server.browser_authorize()

ACCESS_TOKEN = str(server.fitbit.client.session.token['access_token'])
REFRESH_TOKEN = str(server.fitbit.client.session.token['refresh_token'])


"""Authorization"""
auth2_client = fitbit.Fitbit(CLIENT_ID, CLIENT_SECRET, oauth2=True,
access_token=ACCESS_TOKEN, refresh_token=REFRESH_TOKEN)

yesterday = str((datetime.datetime.now() -
datetime.timedelta(days=5)).strftime ("%Y%m%d"))
yesterday2 = str((datetime.datetime.now() -
datetime.timedelta(days=5)).strftime ("%Y-%m-%d"))
today = str(datetime.datetime.now().strftime ("%Y%m%d"))
today2 = str(datetime.datetime.now().strftime ("%Y-%m-%d"))


#get heart rate data / should be today
fitbit_stats2 = auth2_client.intraday_time_series('activities/heart',
base_date=today2, detail_level='1sec')
stats2 = fitbit_stats2

time_list = []
val_list = []

for i in stats2['activities-heart-intraday']['dataset']:
    val_list.append(i['value'])
    time_list.append(i['time'])

heartdf = pd.DataFrame({'Heart Rate':val_list,'Time':time_list})
```

```
heartdf.to_csv('/Users/Utilizador/Desktop/SLEEPTRAINNER/HealthData/Heart/
heart'+today+'.csv', columns=['Time','Heart Rate'], header=True, index =
False)

#get heart summary
hsummarydf = pd.DataFrame({'Date':stats2["activities-
heart"][0]['dateTime'],
                    'HR max':stats2["activities-
heart"][0]['value']['heartRateZones'][0]['max'],
                    'HR min':stats2["activities-
heart"][0]['value']['heartRateZones'][0]['min']},index=[0])
hsummarydf.to_csv('/Users/Utilizador/Desktop/SLEEPTRAINNER/HealthData/hea
rtsummary.csv', header=False, index=False, mode = 'a')


#get sleep summary
fitbit_stats2 = auth2_client.sleep(date='today')['sleep'][0]
ssummarydf = pd.DataFrame({'Date':fitbit_stats2['dateOfSleep'],
                           'MainSleep':fitbit_stats2['isMainSleep'],
                           'Efficiency':fitbit_stats2['efficiency'],
                           'Duration':fitbit_stats2['duration'],
                           'Minutes Asleep':fitbit_stats2['minutesAsleep'],
                           'Minutes Awake':fitbit_stats2['minutesAwake'],
                           'Awakenings':fitbit_stats2['awakeCount'],
                           'Restless Count':fitbit_stats2['restlessCount'],
                           'Restless
Duration':fitbit_stats2['restlessDuration'],
                           'Time in Bed':fitbit_stats2['timeInBed']
                          },index=[0])
ssummarydf.to_csv('/Users/Utilizador/Desktop/SLEEPTRAINNER/HealthData/sle
epsummary.csv', header=False, index=False, mode = 'a')


#get sleep data / should be today
fitbit_stats3 = auth2_client.sleep(date='today')
stime_list = []
sval_list = []

for i in fitbit_stats3['sleep'][0]['minuteData']:
    stime_list.append(i['dateTime'])
    sval_list.append(i['value'])
sleepdf = pd.DataFrame({'State':sval_list,
                        'Time':stime_list})
sleepdf['Interpreted'] = sleepdf['State'].map({'2':'Awake','3':'Very
Awake','1':'Asleep'})
sleepdf.to_csv('/Users/Utilizador/Desktop/SLEEPTRAINNER/HealthData/Sleep/
sleep'+today+'.csv', columns = ['Time','State','Interpreted'],
header=True, index = False)
```

Anexo C – Dados Obtidos do HeartRate

heart20181105csv - NumPy array

	0	1
0	Time	Heart Rate
1	00:00:01	71
2	00:00:06	69
3	00:00:11	70
4	00:00:21	76
5	00:00:26	75
6	00:00:31	74
7	00:00:36	75
8	00:00:51	77
9	00:01:01	76
10	00:01:06	75
11	00:01:16	74

Anexo D – Dados obtidos do SleepStages

sleep20181105csv - NumPy array

	0	1	2
0	Time	State	Interpreted
1	01:14:00	1	Asleep
2	01:15:00	1	Asleep
3	01:16:00	2	Awake
4	01:17:00	1	Asleep
5	01:18:00	1	Asleep
6	01:19:00	1	Asleep
7	01:20:00	1	Asleep
8	01:21:00	1	Asleep
9	01:22:00	1	Asleep
10	01:23:00	1	Asleep
11	01:24:00	1	Asleep
12	01:25:00	1	Asleep
13	01:26:00	1	Asleep
14	01:27:00	1	Asleep
15	01:28:00	1	Asleep