Universidade de Aveiro

Departamento de Eletrónica, Telecomunicações e Informática

2019

Francisco Henriques Martins

Eyempact: a low-cost system for integrated eye tracking and physiology analysis

Eyempact: um sistema de baixo custo para análise do olhar e fisiologia

Universidade de Aveiro
2019

Departamento de Eletrónica, Telecomunicações e Informática

Francisco Henriques Martins

Eyempact: a low-cost system for integrated eye tracking and physiology analysis

Eyempact: um sistema de baixo custo para análise do olhar e fisiologia

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor José Maria Amaral Fernandes, Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e da Doutora Susana Manuela Martinho dos Santos Baía Brás, Investigadora do IEETA, Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**Prof. Doutor Augusto Marques Ferreira da Silva** Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
vogais / examiners committee	**Prof. Doutor Sergi Bermúdez i Badia** Professor Associado da Universidade da Madeira
	Prof. Doutor José Maria Amaral Fernandes Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho às minhas avós e avô (in memoriam) e aos meus pais.

agradecimentos/ acknowledgments

Ao meu orientador, Professor Doutor José Maria Amaral Fernandes, à minha coorientadora Doutora Susana Manuela Martinho dos Santos Baía Brás e ao Professor Doutor Ilídio Fernando de Castro Oliveira, por todo o apoio, disponibilidade e compreensão ao longo deste percurso, pelos bons momentos de discussão de ideias e também pelos momentos de descompressão, o meu muito obrigado!

À Professora Doutora Sandra Cristina de Oliveira Soares, à Joana Filipa Domingues Cerqueira e ao André Francisco Pinelas da Silva pela colaboração no caso de estudo apresentado, assim como a todos os participantes, sem os quais o estudo não teria sido possível.

Aos poucos, mas bons amigos que me acompanharam nos bons e nos maus momentos ao longo de todo o curso, pelas palavras de motivação e apoio em geral.

À minha família, fonte de compreensão e apoio durante todos estes anos, sem a qual este trabalho nunca teria sido possível.

palavras-chave Seguimento do olhar; Sistema baixo custo; Atenção visual; Monitorização fisiológica; Avaliação psicofisiológica

resumo Um sinal que é habitualmente usado para estudar a atenção humana é o olhar. Para além deste, sinais bioelétricos também são usados neste contexto. O ramo de pesquisa que estuda a atenção humana, assim como comportamentos e emoções, utilizando estes sinais é chamado de psicofisiologia. Estudos de psicofisiologia utilizam vídeos e imagens para despertar emoções, sentimentos e comportamentos. No entanto, os participantes nem sempre prestam atenção aos estímulos, algo que pode ser justificado pelos próprios estímulos. Este comportamento de evitamento, se não for de todo pretendido (de acordo com o desenho da experiência), pode ter um impacto negativo na avaliação psicofisiológica e comportamental realizada *a posteriori.* Deste modo, é essencial compreender a atenção efetiva do participante durante toda a experiência, e ter especial cuidado durante o processo de idealização e desenho da mesma. O objetivo deste trabalho é fornecer um conjunto de ferramentas, o sistema Eyempact, para a avaliação do impacto de estímulos visuais (imagens estáticas ou vídeos) em participantes utilizando o seguimento do olhar (capturado pelo *Tobii eye tracker*). Outras variáveis fisiológicas, ECG, EDA, EMG, são também monitorizadas (utilizando o sensor *BITalino*). Eyempact é capaz de (1) adquirir simultaneamente toda esta informação, e (2) fornecer um ambiente de revisão dos dados (*offline*) que auxilia a análise deste tipo de dados. O sistema foi utilizado com sucesso num caso de estudo, o ambiente de revisão foi útil para avaliar visualmente sessões específicas de alguns participantes e para exportar métricas e medidas para a análise de dados apresentada neste documento.

keywords Eye tracking; Low-cost system; Visual attention; Physiological monitoring; Psychophysiology assessment

abstract The usual signal to evaluate human attention is eye tracking. Beyond this, bioelectrical signals are also used in this context. The field of research that studies human attention, as well as human behavior and emotions, using these signals is called psychophysiology. Psychophysiological studies usually make use of movies or images to trigger emotions, feelings and behaviors. However, the participants do not always pay the expected attention to the stimulus, which may be justified by the stimulus itself. This avoidance behavior, if not intended at all, can have a negative influence on the psychophysiological and behavioral evaluation *a posteriori*. Therefore, it is important to understand the effective attention of the participant in the entire experience and have special caution in idealization and designing of said experience. The objective of this work is to provide a set of *software* tools, the Eyempact system, to evaluate the impact of visual stimuli (static images or movies) on participants, using eye tracking (Tobii eye tracker). Other physiological variables are also monitored (ECG, EDA, EMG using BITalino board). Eyempact is capable of (1) synchronously acquire all this information, and (2) provide a comprehensive offline review environment to support domain specific analysis, namely by psychophysiology features identification. The system was successfully utilized in a case study, the review environment was very useful to visually evaluate specific participant sessions and to export data metrics and measurements for the data analysis presented in this document.

I. Contents

I. Contents ..... **i**

II. List of figures ..... **iii**

III. List of tables ..... **v**

IV. List of abbreviations and acronyms ..... **vii**

1 Introduction **1**

1.1 Objectives 2

1.2 Dissertation structure 3

2 State of the art **5**

2.1 Eye tracking technology 5

2.1.1 Eye trackers 5

2.1.2 Eye tracking measurements and metrics 6

2.1.3 Applications 7

2.2 Integrated systems (all-in-one) 8

3 The Eyempact system **11**

3.1 Target use cases 11

3.2 System architecture 12

3.3 Review environment user interface 14

3.3.1 Data visualizations 16

4 Where's Wally? – Proof of concept application **21**

5 Eyempact implementation **23**

5.1 Data sources 24

5.1.1 Tobii eye tracker 24

5.1.2 BITalino board 25

5.2 Data types and formats 26

5.3 Data acquisition 28

5.3.1 Measurements and metrics 29

5.3.2 Control Panel user interface 30

5.3.3 Configuration files 32
5.4 Review Environment options 34
5.4.1 Interface: OpenCV vs PyQt 34
5.4.2 Widgets and menus 35
6 Study on tobacco packages health warnings 39
6.1 Experiment protocol and details 40
6.2 Setup 41
6.3 Hypothesis 42
6.4 Information and collected data 43
6.5 Analysis results and discussion 43
6.5.1 Physiology results 45
7 Conclusions and discussion 49
7.1 Future Work 50
References 52
Appendix A – Eyempact 55
Appendix B – Health warnings in tobacco packages study 57
Appendix C – Public events 64

II. List of figures

Figure 1 – Eyempact use case diagram. 11
Figure 2 – High level architecture of Eyempact. 13
Figure 3 – Initial screen and file selection. 14
Figure 4 – File explorer panel (left) and tab widget view with gaze representation (right). 15
Figure 5 – *Review Environment* window overview. Settings on the left, metrics on the right. 15
Figure 6 – Eyempact scan path visualization. 16
Figure 7 – Examples of two scan path representations of the same data file. Dots only representation on the left, segments only representati\on on the right. 17
Figure 8 – Scan path with background image and numerated data points. 17
Figure 9 – Fixations file representation. 18
Figure 10 – Bioelectrical signals plot from data acquired by Eyempact acquisition module. 18
Figure 11 – Gaze heatmap visualization overlay. 19
Figure 12 – *Where's Wally* application menu. User is opting to start the game on level zero. 21
Figure 13 – User found Wally (bottom centre of the image). 22
Figure 14 – Eyempact architecture and technologies (detailed view). 23
Figure 15 – Tobii EyeX (top) and Tobii 4C. 24
Figure 16 – BITalino board. 25
Figure 17 – BITalino board kit and VJ box. 25
Figure 18 – Metrics panel view. More details on the panels in Appendix A. 29
Figure 19 – User selected AOI is shown as a transparent rectangle. 30
Figure 20 – Control panel application main user interface. 31
Figure 21 – Configuration dialogs. Eye tracking (left) and BITalino (right) configurations. 31
Figure 22 – Example of configuration file for the *Eyes Data Gatherer* program. 32
Figure 23 – Example of configuration file for the *BITalino Data Gatherer* program. 33
Figure 24 – OpenCV *Review Environment* program menu interface. 34
Figure 25 – Widget used for traversing gaze and fixation files. 35
Figure 26 – This is the widget that allows the replay of gaze and fixation files. 36
Figure 27 – Generate menu. Possible heatmap and video generation operations shown. 37
Figure 28 – Example images used in the experiment, one of each category. Neutral (top left), positive (top right), negative (bottom left) and smoker package example images (bottom right). 39
Figure 29 – Initial form and SAM scale print screens from the Unity application. 40
Figure 30 – Material disposition in the experiment setup. 41

Figure 31 – Example of electrodes placement used in the experiment. (1) ECG, (2) EDA, (3) EMG. Description and decision making on the chosen electrode placement in Appendix B. 42
Figure 32 – Participant mean arousal and valence values for each image in the experiment. 43
Figure 33 – Affective states represented in space by arousal and valence values; figure copied from [20]. 44
Figure 34 – Participant ten had higher HR for the positive and tobacco image groups. 45
Figure 35 – Higher EMG entropy for smokers versus non-smokers in all the image groups. 46
Figure 36 – Slightly higher number of points over the images in the smoker group for all the image groups. 46
Figure 37 – Participant twenty-seven fixations over the AOI in the different groups of images (exact area of the image). 47
Figure 38 – Image on background is for visual purposes only, represented data is from all the tobacco package image group from the participant twenty-seven. 48
Figure 39 – Settings panel (left and center image) and measurements / metrics panel (right). ..56
Figure 40 – First screen shown to the user (left) has a hidden button on the upper left corner to configure the experiment parameters (right). 59
Figure 41 – Initial formulary presented to the participant. 59
Figure 42 – Simple explanation of on the image classification scale (SAM). Further oral explanations were given in the section. 59
Figure 43 – Text presented for five seconds to gather a baseline (left). Example of image placement (right). 60
Figure 44 – SAM scale screen. 60
Figure 45 – Fagerström dependency test shown to smoker and ex-smoker participants. 60
Figure 46 – Poster displayed at Students@DETI event. 64
Figure 47 – *Where's Wally* live demo at Xperimenta event. 64

III. List of tables

Table 1 – Comparison between different tier eye trackers. 6
Table 2 – *Control Panel* program main use cases. 12
Table 3 – *Review Environment* program main use cases. 12
Table 4 – Data file types and descriptions. 26
Table 5 – Gaze data file format. 26
Table 6 – Fixations data file format. 26
Table 7 – Status notifications data file format. 27
Table 8 – Eyes coordinates data file format. 27
Table 9 – Physiological signals data file format. 27
Table 10 – *Eyes Data Gatherer* program configuration descriptions. 33
Table 11 – *BITalino Data Gatherer* program configuration descriptions. 34
Table 12 –Table comprising all the sociodemographic data gathered. 61

IV. List of abbreviations and acronyms

AOI – Area of interest

API – Application programming interface

BT – Bluetooth

ECG – Electrocardiogram / electrocardiography

EDA – Electrodermal activity

EEG – Electroencephalogram

EMG – Electromyogram

GUI – Graphical user interface

HCI – Human-computer interaction

HDMI – High-Definition Multimedia Interface

HR – Heart rate

HW – Hardware

OS – Operating system

PC – Personal computer

SW – Software

TS – Timestamp

TTFF – Time to first fixation

UX – User experience

VBB – Virtual bounding box

VR – Virtual reality

VJ – VitalJacket

1 Introduction

Nowadays, we live in a world where our devices compete for attention, more and more we divide it between multiple screens and multiple technologies (Smartphones, smart TV's, personal computers (PC's), E-readers, etc…). This way, there is an increasing interest in understanding how the human attention works and changes when presented with different contexts and tasks [1].

One good example to illustrate this, is the investment of major technological players like Microsoft. In order to improve their marketing, Microsoft tried to understand the human attention by using the electroencephalogram (EEG) signal, studying which areas of the brain "light" up in pre-determined scenarios. The main conclusion was that the average human attention span decreased from twelve seconds to eight seconds in a period of just thirteen years (2000-2013). Having a decreased attention span certainly has its downsides, but, as it is observed in the study "…consumers are becoming better at doing more with less via shorter bursts of high attention and more efficient encoding to memory" [2]. This may support the idea that our brains are evolving to deal with the ever-increasing stimuli we are being exposed to every day and encoding the useful information more efficiently.

In this context, the EEG, the electrocardiogram (ECG) and other bioelectrical signals can be good options to study how our attention and other physiological mechanisms change over time [3], however, they have the drawback of requiring invasive setups, namely via the use of electrodes. Besides limiting our freedom of movement, this does not allow realistic conditions assessment namely in relation to comfort. Technology is slowly catching up, wearable solutions like smartwatches and smart bands that allow e.g. measuring our heart rate (HR), are already available.

Although wearables are an alternative, and are considered good enough for some research scenarios, these are not ideal solutions, their accuracy is still considered an issue [4]. Also, they still do not beat the value of raw physiological data, since we can extract more information from it [4]. From a technical perspective, not all wearables provide flexible and open application programming interfaces (API's) for development and integration with third-party systems. There is however a bio-signal that does not compromise our comfort (using standalone eye trackers), and in some solutions (head mounted eye trackers and more recently virtual reality (VR) eye tracking) our freedom of movement as well, and that is our gaze.

In this line of thought, non-invasive solutions to follow the user's attention using eye tracking seem attractive. Although eye tracking is not a new technology, it has been a niche research field, probably in part due to the high costs of acquiring an eye tracking system. Most eye tracking solutions remain expensive, but new off-the-shelf solutions used in gaming seem to be a good alternative.

Eye tracking has started to be popularized, and it is now being used as means of interaction with your PC and in gaming to enhance your games experiences. This happened mostly due to the Tobii[1] company that is now providing affordable eye tracking devices for the everyday consumers, bringing eye tracking to the mass market.

Hardware (HW) is getting inexpensive and thus more affordable to more researchers and enthusiasts, on the other side the cost of software (SW) is almost debatable. The companies that are selling eye tracking systems nowadays, either sell the HW separately from the SW, bundle both with considerably higher prices for the SW addition or use the "Contact with us" approach to bundle or deal with the SW side of the question. Although companies like Tobii and Gazepoint[2] are providing open source API's for some or all their devices, open source SW in the eye tracking area is still in its infancy. Thereby, it is still required some computer science/programming knowledge or background for a researcher to get value out of the open source eye tracking SW out there. The present work will explore this lack of SW accessibility in eye tracking.

1.1 Objectives

The main objective of this dissertation was to develop a portable and affordable system that measures the impact of the visualization of images or videos on a person's behaviour (evaluated by an eye tracking system) and physiology (through ECG, electrodermal activity (EDA) and electromyogram (EMG) signal monitoring). Ultimately, it is intended to infer alterations on the person emotional state when presented to a particular stimuli.

The developed system needs to be able to do:

- **Data acquisition** – from an eye tracker or eye tracking system and physiological signal board or similar;
- **Data visualization and analysis** – from the different data sources coming from the acquisition module;
- Support a feedback loop that allows using the own system outputs to adapt/change the user experience (UX) in a study or program in real time.

The data visualization and analysis operations should be transversal to the operating system (OS), that is, the SW should run in any main OS: Windows, Mac OS and Linux. Unfortunately, the data acquisition side of the question might be tied to the HW support.

---

[1] Tobii AB is a high-tech company that develops and sells eye tracking products (Official website: https://www.tobii.com/).

[2] Gazepoint is a company that "provides affordable solutions for eye tracking, neuromarketing and biometric research" (Official website: https://www.gazept.com/).

1.2 Dissertation structure

This dissertation is composed by the following chapters:

Chapter 1 introduces the reader to the context, motivation and main objectives of this dissertation that will be explored throughout the document and presents the dissertation structure.

Chapter 2 describes the state of the art either in terms of technology (sensors mostly), SW developed with similar objectives of the present work, research and applications in this area of study.

In **chapter 3**, the architecture of the developed system is presented, as well as the modules which constitute it. Each piece of developed SW is investigated from a perspective of their main functionality.

Chapter 4 goes into more detail on the implementation aspect of the system. Some implementation decisions are also discussed in this chapter.

Chapter 5 shows a proof of concept program, that demonstrates the real time eye tracking capabilities as means of interaction with a PC.

Chapter 6 presents a study that made use of all the system's components. Presents some hypothesis and results from the analysis of the system exported data.

This work discussion, conclusions and possible future work is presented in **chapter 7**.

2 State of the art

A usual solution to evaluate someone's attention is the use of an eye tracking system. Eye tracking is the process of measuring the point of focus of one's gaze[3]. From the end of the nineteenth century, when eye tracking was started to be studied through naked eye observations [5], to today, technology has evolved immensely. There are now available almost an overwhelming number of different eye trackers (devices that allow eye tracking measurements), eye tracking solutions and technologies for the most varied uses.

2.1 Eye tracking technology

The use of eye tracking systems has been growing in the last decades, and it has become an invaluable resource in multiple research areas [5]. Nowadays it is also becoming an accessible tool for human-computer interaction (HCI) [6] for the most varied types of users and user interfaces.

2.1.1 Eye trackers

Over time, multiple types of eye tracking devices and technologies have been developed. There are eye trackers that use special contact lenses with an embedded mirror, eye trackers that use two pairs of electrodes on the skin around the eyes, that measure the electrooculogram, and the most common type of eye trackers, that use infrared or near-infrared light to measure the corneal reflections and obtain a person's gaze and other eye related features. Nowadays we have standalone eye trackers, designed to connect to a machine via USB, or integrated eye trackers, already embedded on the machine's display (on a laptop or in a dedicated display); these eye trackers use the last methodology (corneal reflection) to measure one's gaze. Also, the use of lenses is starting to reappear with the emergence of VR headsets supporting eye tracking.

There is a huge variety of HW available in the market, finding the eye tracker that suits your study can be a difficult process, it depends on what you want to study and how you want to do it. In the table below, standalone eye trackers from different price ranges are presented with some of the most looked after characteristics. As it is possible to observe, the presented examples have minimal differences, but depending on the study requirements those can be significant. It is of note that, this is not a significantly representative sample, and it is recommended that a more in-depth comparison of the different types of eye trackers and eye tracker characteristics are considered when acquiring an eye tracking system.

[3] *Simple definition from Wikipedia: https://en.wikipedia.org/wiki/Eye_tracking.*

Eye tracker grade Characteristics	Low-end		Middle-end		High-end
Manufacturer	Gazepoint	Tobii	Gazepoint	Tobii	Tobii
Model	GP3	4C	GP3 HD	X2-30	X2-60
Data rate	60 Hz	90 Hz	150 Hz	30 Hz	60 Hz
Accuracy	0.5-1º	N/A	0.5-1º	0.4º	0.4º
Measures pupil size	Yes	No	Yes	Yes	Yes
Operating distance	50-80 cm	50-95 cm	50-80 cm	40-90 cm	40-90 cm

Table 1 – Comparison between different tier eye trackers.[4]

2.1.2 Eye tracking measurements and metrics

When talking about eye tracking there are two terms that will always come up, gaze and fixations, they are the base units for eye tracking analysis. Gaze is a set of points that represent the coordinates in a screen that a person looked at. Fixations are groups of gaze points that are very close together, in space and usually in time; this is, a series of points that were captured in a small region of the stimulus in a short amount of time. Another term that is usually referenced is saccades, they are the eye movements between fixations, although in this work we do not use them. Data on user presence is usually gathered too. In conclusion, what can usually be gathered in terms of raw or lightly processed data:

- Gaze;
- Fixations;
- Saccades;
- Eye coordinate positions;
- Pupil diameter;
- Blinks;
- User presence.

Using these measurements, a wide range of possible metrics can be computed; fixation-based metrics, gaze-based metrics, metrics based on saccades, etc [7]. Fixation-based metrics are very common, some of the most utilized are number of fixations, number of fixations in an area of interest (AOI), fixation duration, time to first fixation on target (on AOI), fixation density; and from these simple metrics many others can be derived [8]. There are also measurements that use the AOI, like revisits (how many times the gaze left and re-entered the AOI), time spent in each AOI, etc.

[4] *This table information was gathered from iMotions cheat sheet (information sources confirmed). For a more comprehensive table please consult: https://imotions.com/blog/eye-tracking-hardware-walkthrough.*

2.1.3 Applications

Eye tracking is utilized in the most varied areas, from psychology [9] (and respective sub-areas, e.g. psycholinguistics and psychophysiology), to marketing [10], packaging [11], medicine [12], the aforementioned HCI and UX [6], and other investigations on human behavior and attention, where understanding gaze patterns can be a great benefit. HCI/UX and psychophysiology research are two relevant areas of study for the present dissertation, this way the next paragraphs are dedicated to present in more detail the current uses of eye tracking in these research areas.

Eye tracking in HCI/UX research

HCI researchers have been using eye tracking to improve usability, either on web pages or graphical user interfaces (GUI), in traditional PC displays and mobile displays like smartphones and tablets. By providing data that enables understanding the attention patterns, eye tracking helps the researchers designing and optimizing user interactions therefore creating better user experiences and more usable applications [6]. Eye tracking has also been adopted as means of interaction in computer gaming [13], web navigation and tracking[5], OS navigation[6] and other GUI's in general.

Eye tracking in Psychophysiology research

The use of the participant's gaze in a psychology study already qualifies it as a psychophysiology study, because one's gaze is a signal, thus being part of our physiology. Eye tracking is usually utilized as means of studying a person emotional response when presented with a specific stimuli; for example, in this study [14] the gaze allows the evaluation of participant reactions to stuttering. Another use is the study of emotional response to imagery using the pupil diameter [15], which is proved to be related to the participants state of arousal. Although eye tracking is quite common in psychology research, the combination of the gaze with other biosignals (specifically the bioelectrical ones) is scarce. One good example, is the use of eye movements, HR and EDA in the study of eye to eye contact [16].

[5] *Possible using xLabs browser plugins: https://xlabsgaze.com/user-manual.*

[6] *Get started with eye control in windows 10: https://support.microsoft.com/en-us/help/4043921/windows-10-get-started-eye-control.*

2.2 Integrated systems (all-in-one)

There are only a few systems that gather physiological sensors data acquisition and representation under the same SW platform for human behaviour study. These systems are usually high cost, therefore mostly directed at big research units, and commercial companies, capable of supporting such costs. In this section, we will present some of these systems that can gather data from a big pool of sensors and represent it in organized manner in a timeline, in conjunction with data metrics, and relevant information for the future analysis.

Tobii Pro Lab and Pro Studio

Tobii Pro Lab is a "complete solution for researching human behaviour"[7] using eye tracking, supported by most of their Tobii Pro eye trackers (some were discontinued), and bioelectrical signals, supported by the third-party sensor *Shimmer3*. The Pro Lab is constituted by three modules: the *Designer*, where the user can create their experiment, inputting *stimuli* and defining the user interactions; the *Recorder*, responsible for the eye trackers configuration and calibration, as well as recording all the configured data; and the *Analyser* that allows the user to visualize, replay and analyse the recorded data. Tobii recommends a standalone machine for the research purposes, and the system specifications are mid to high end. From the product description sheet, we assess that the Pro Lab does provide a comprehensive set of eye tracking recording options as well as metrics; although in the metrics side, it could have a more extensive list. Tobii Pro Studio is another piece of SW that seemingly is capable of everything that the Pro Lab is. From the Tobii main online pages for these products and respective product description documents it is possible to see similarities in both systems, and seem to have the same purposes, to provide a full workflow for research studies. The main disadvantage of Tobii SW is probably the lack of options for third-party sensors integration. From the eye tracking perspective, Tobii does provide the full range of eye tracking HW for the various possible situations; from the bioelectrical signals perspective, only one sensor is currently available, which can be limitative for a research study that requires the acquisition of said signals, with specific requirements (higher sample rate e.g.).

[7] ***https://www.tobiipro.com/product-listing/tobii-pro-lab*** *- Tobii Pro Lab web page.*

Biopac Systems

Biopac is a well-known and proven system in the physiology research field. Biopac is "cited in over 34,500 peer-reviewed" scientific documents, and therefore trusted by the world's premier laboratories and leading researchers[8]. Biopac has the most comprehensive HW support out the presented systems in this section, providing multiple systems, to acquire the most varied physiological signals and measurements. Their main advantage being, the flexibility of having a large number of system options, may as well be their main disadvantage, it can be overwhelming for a small research group, that is starting to do psychophysiological research to have such a big range of HW and SW options; it can also be a tough decision to choose certain system in detriment of other with similar capabilities. As far as eye tracking goes, Biopac has a partnership with our following entry, iMotions that provides the SW platform for Biopac eye tracking solutions.

iMotions

It is not a simple task to differentiate these systems, as their main offer is the same, in simple terms, an integrated system for the study of human behaviour (in general). iMotions has the integration of over fifty biosensor devices[9] from the biggest names in the different areas; it supports all the major eye tracking devices (including Tobii eye trackers) and has a partnership with Biopac which has talked above, provides a great set of bioelectrical acquisition HW. iMotions supports all the capabilities referred on the systems above, and more. It supports, a really important feature, that is "real-time / live view". This enables the researchers to have a live representation of the data being acquired during the whole experiment, allowing the researchers to tweak or change configurations in the last minute, if that reveals do be necessary. iMotion also has a great "library" of articles and documentation for all the signals that it acquires, their writing is simple and concise, providing the absolute beginners with a great source of information.

[8] **https://www.biopac.com/** - *Biopac website.*

[9] **https://imotions.com/** - *iMotions website.*

3 The Eyempact system

In this chapter, Eyempact is presented as a proof of concept system that combines both attention and physiology monitoring into a low-cost solution for research purposes. This combination, although technically viable seems to be an unexplored niche. Within the existing state of the art, we conclude that there are only a few systems trying to accomplish this, although they are high-cost and targeted at big research units. By aiming at a low-cost and small size setup, Eyempact can be easily adapted and deployed for different scenarios, from simple eye tracking demonstrations to challenging psychophysiological research protocols. This concept was motivated out of need, as attention monitoring plays an important role in research protocols in which IEETA research unit (hosting this work) has been collaborating.

3.1 Target use cases

The main use cases set for this system are those related to psychophysiology research. The system main function is to monitor the user changes in attention and other physiological signatures (supported by the acquisition module) and ensure that the information is time tagged for future use.

Eyempact provides two subsystems, *Control Panel* and *Review Environment*, with which an actor can interact with. We consider two main usage environments (Figure 1):

1. The same researcher is responsible to setup, run and analyse the data originated from a study and gather results, using both modules;

2. One actor (psychology researcher as an example) is responsible to setup and run a study, using the *Control Panel*, and another researcher is responsible for the data analysis and results, using the *Review Environment*.

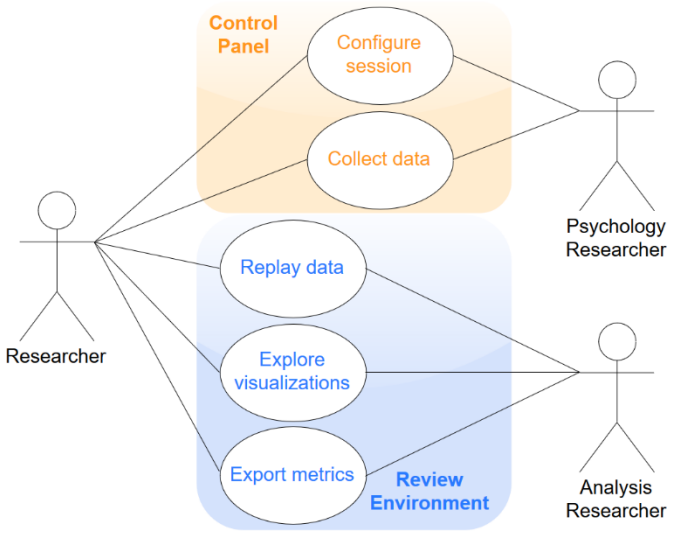

Figure 1 – Eyempact use case diagram.

Use case	Description
Configure session	The researcher must be able to select the data variables relevant for the study and configure the sensors accordingly.
Collect data	After configuration, the researcher starts collecting data, for a given period. The data should be stored locally and have the possibility of upload to a cloud service.

Table 2 – *Control Panel* program main use cases.

Use case	Description
Play data	The researcher must be able to play any data variable selected for the study, after the realization of the same. The researcher should be able to generate video replay visualizations.
Explore visualizations	It must be possible to explore data files alongside their visualizations.
Export metrics	Information on the data and calculated metrics should be able to be exported in a straightforward manner, for posterior data analysis.

Table 3 – *Review Environment* program main use cases.

In terms of non-functional requirements, Eyempact should be usable in the main operating system (Windows, Linux and Mac operating systems), when possible. This might be limited by sensors support to these operating systems, in the case of the *Control Panel* (responsible for the acquisition programs).

3.2 System architecture

The proposed Eyempact system includes the following modules (**Figure 2**):

- *Control panel*: Data acquisition and storage;
- *Review Environment*: Visualizations and analysis;
- Visual stimulus.

The acquisition module is responsible to store all the incoming information from the supported sensors. It also provides a GUI (*Control Panel*) enabling the user to manage what and where the data is to be stored for posterior analysis or use on the *Review Environment* – Eyempact own visualization and analysis tool (also supported by a GUI).

The three modules work independently. Each one of them represents a program that does not necessarily need the other. The *Review Environment* module uses, for its core operations, the output resultant from the usage of the *Control Panel* and output from the stimulus program. However, these outputs could also be resultant from other third-party programs if they were to output in the same data formats as Eyempact.

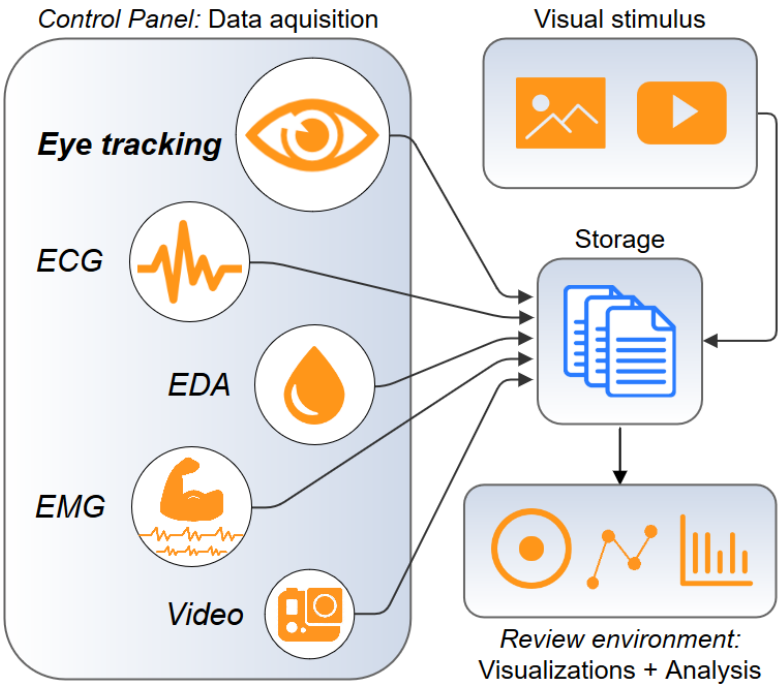

Figure 2 – High level architecture of Eyempact.

Control panel: Data acquisition and storage

This module is responsible for the start and stop of eye tracking, physiology and video data acquisition. It can start the acquisition of physiological signals (ECG, EDA, EMG), eye tracking data (Gaze, fixations, …) and video (*GoPro*), simultaneously. Still in this module, the user can configure, before the start of each acquisition, what is to be recorded, data rate (for the physiological signals), the names and locations of the data files, etc. At any moment, the user can upload the locally saved files to the cloud (*Dropbox*) as a backup option.

Review Environment: Visualizations and analysis

The *Review Environment* is the program where the user will review the acquired data. As the name suggests, in this program you can replay the previously acquired data. As soon as the data file is selected by the user and read, a raw visualization will be presented; then the data can be simply replayed or traversed. It is always possible to review e.g. gaze data points combined with the visual stimulus. There are also available, some metrics and information on the data that can be exported for posterior use and further analysis.

Visual stimulus

Presents visual stimuli (image or video) and saves events (e.g. image transitions, start of video, answers to custom made questionnaires). Along the dissertation multiple applications were developed for the visual stimulus presentation. Since this is not the focus of this work, a general application for this purpose was not developed.

3.3 Review environment user interface

The first thing presented to the user is simply two “Open files” buttons, used to select from the Windows file explorer window which gaze or fixations file(s) to open (left button) or physiological signal file(s) (right button).

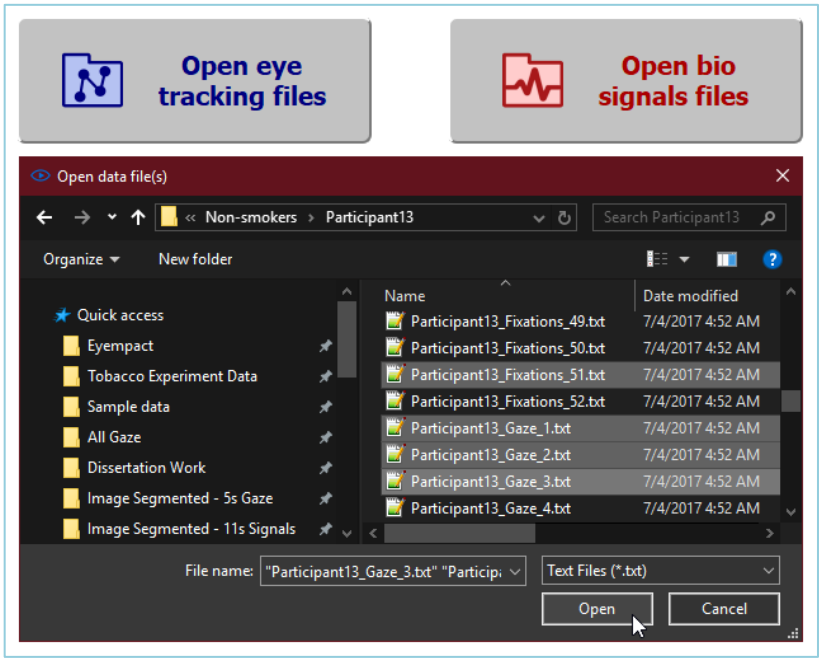

Figure 3 – Initial screen and file selection.

Each one of the selected files will be represented in a tab named after the respective file names. You can navigate through the opened files as you would in any modern browser; alternatively, you can select which file you want to be drawn in the explorer panel. You can continue to open new files through the toolbar buttons or menu options as in any other program. Upon closing all tabs, the main menu is revealed again.

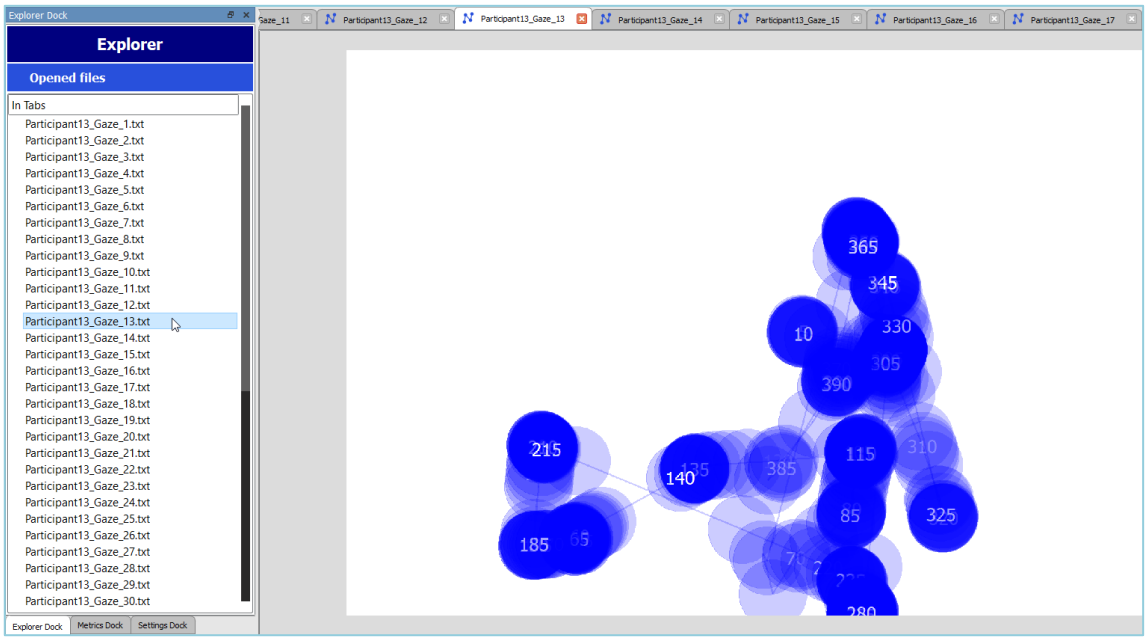

Figure 4 – File explorer panel (left) and tab widget view with gaze representation (right).

After drawing, you have available the panels that enable you to edit the visualizations, create new visualizations (heatmaps), review and export measures and metrics. All interface widgets are intuitive and simple to use. Also, the panels are movable, so you can configure the positions that suit your pattern of utilization the best. More detailed screenshots of this interface are presented in appendix A.

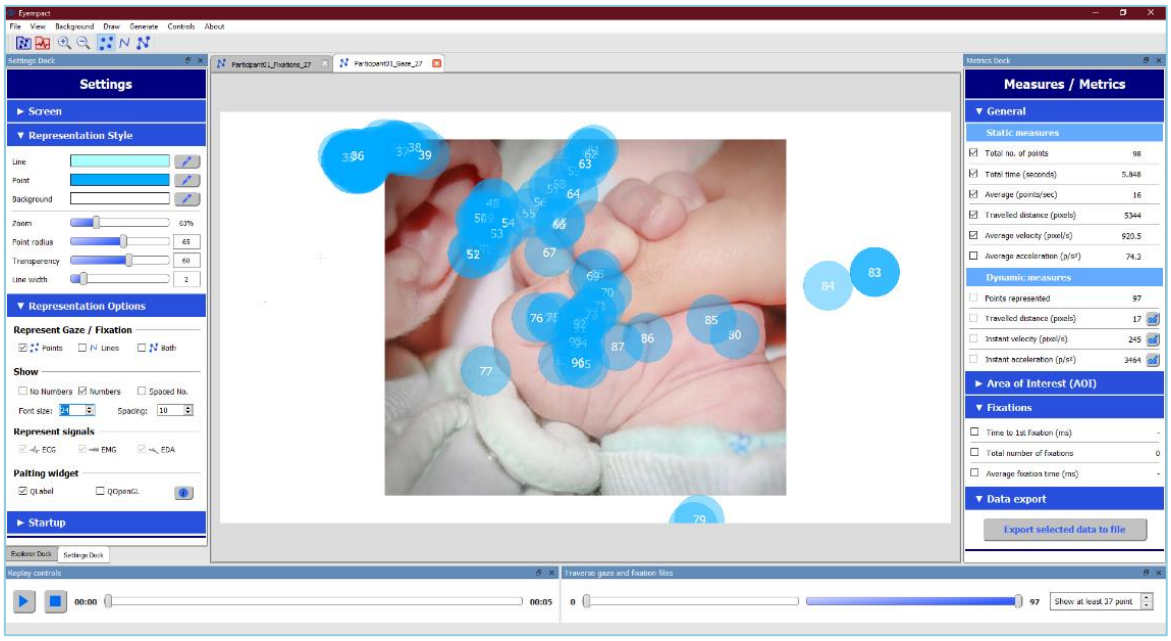

Figure 5 – *Review Environment* window overview. Settings on the left, metrics on the right.

3.3.1 Data visualizations

The available visualizations in Eyempact are divided in three categories: exploratory, heatmaps and video replays. In the following sub-sections, we will present and explain the capabilities of each visualization.

Exploratory visualizations

The most basic visualization is the gaze representation (**Figure 6**), it draws the data points as circles and unites them with a straight line segment. Consecutive data points are connected by the said straight line, points are drawn from older to newer records. This is traditionally known in eye tracking as a "scan path" visualization.

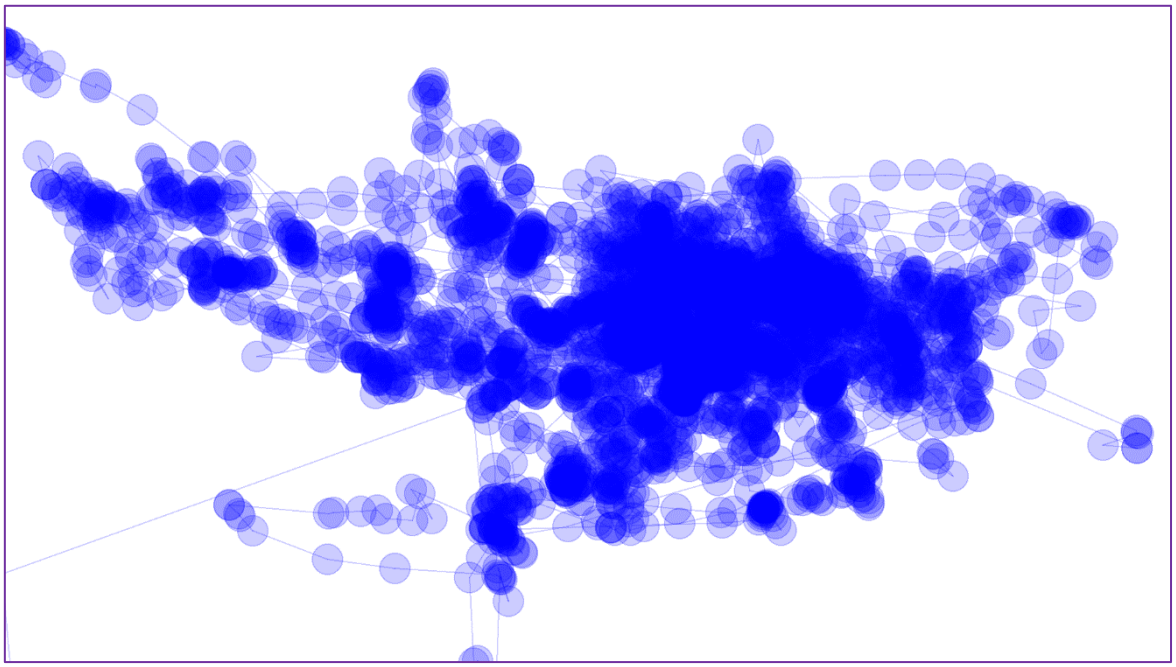

Figure 6 – Eyempact scan path visualization.

The figure above is only an example, the user can modify the aesthetics to better fit their goals or image use. For instance, it is possible to only represent the points (**Figure 7** – left) for a clearer vision of where the points are concentrated, or represent only the segments that connect the points (**Figure 7** – right). To achieve a visualization of the users' personal preference, or even a visualization that suits better the dataset size (e.g.), you can change the circle radius, line thickness, overall transparency and the colour of the points, lines and background.

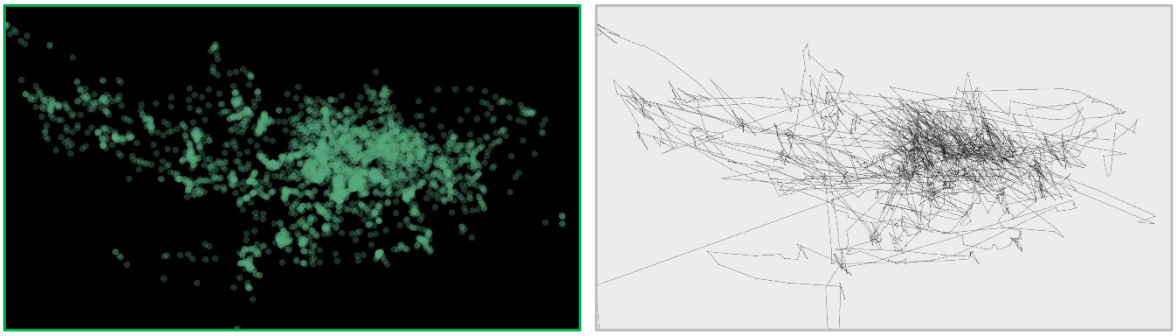

Figure 7 – Examples of two scan path representations of the same data file. Dots only representation on the left, segments only representation on the right.

By themselves, the scan path visualizations presented above gives little information. We can identify which were the major areas of focus, and the areas in which the user spent more time, by adjusting the transparency but that is about it. The context in which the gaze was recorded, the stimuli, is extremely important; for this it is possible to add a background image instead of having a solid colour background. It is also important to know the actual path of the gaze. To achieve this, we can add numbers to the gaze points (**Figure 8** – below). If the quantity of numbers in the visualization is too overwhelming, you can adjust them to be drawn in any interval of your choice (e.g. 100 to 100, 20 to 20, 13 to 13, etc…).

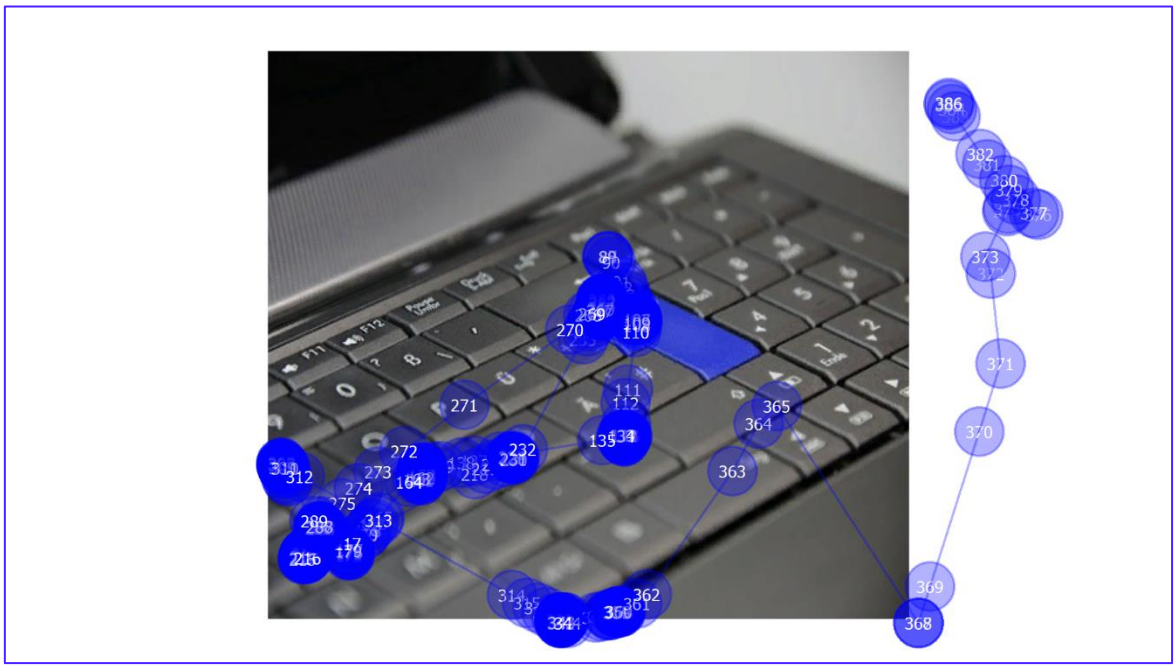

Figure 8 – Scan path with background image and numerated data points.

For the fixations file visualization, each fixation is represented by a single circle, in which the centre is the centroid of all points included in the fixation. The circle is progressively bigger as the fixation duration is higher until a certain defined maximum size. The defined maximum size is the same as the maximum gaze radius circle (around hundred and twenty pixels).

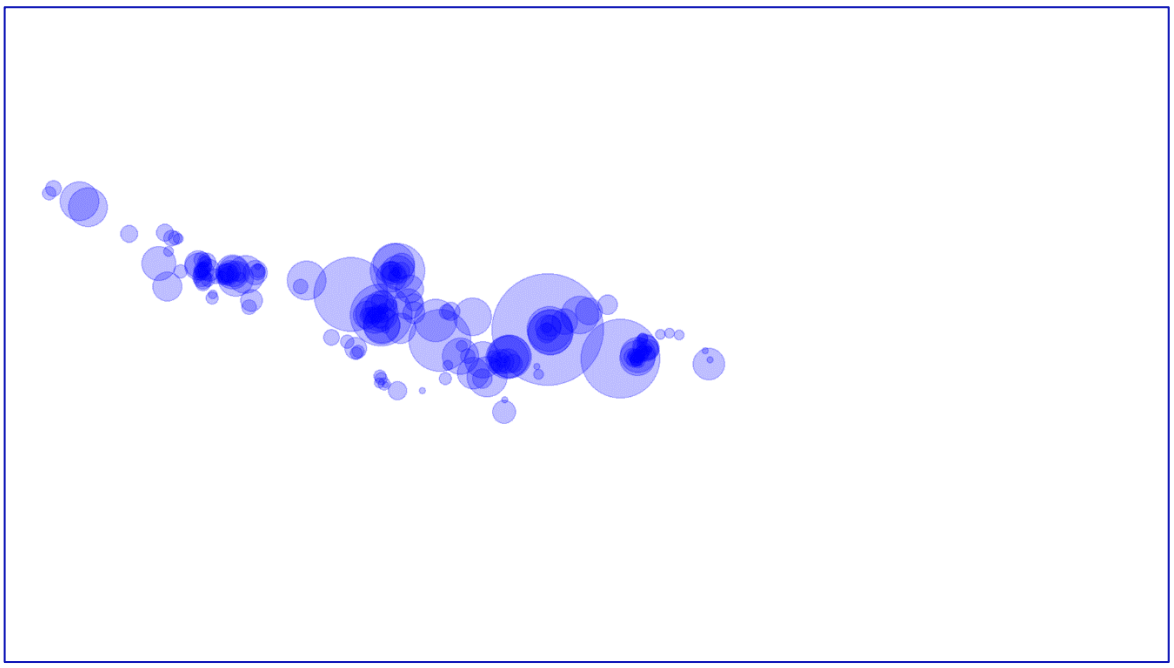

Figure 9 – Fixations file representation.

For the bioelectrical signals file, the library *matplotlib* was used, a custom widget class was created in order to use *matplotlib* graphs embedded in the *Review Environment* interface. Similarly to the gaze and fixation visualizations options presented before, the user can choose which signals to represent. Other customizations like, colour changes, different styles of lines, zoom in and zoom out (e.g.) are also available from the navigator tool provided by *matplotlib*.

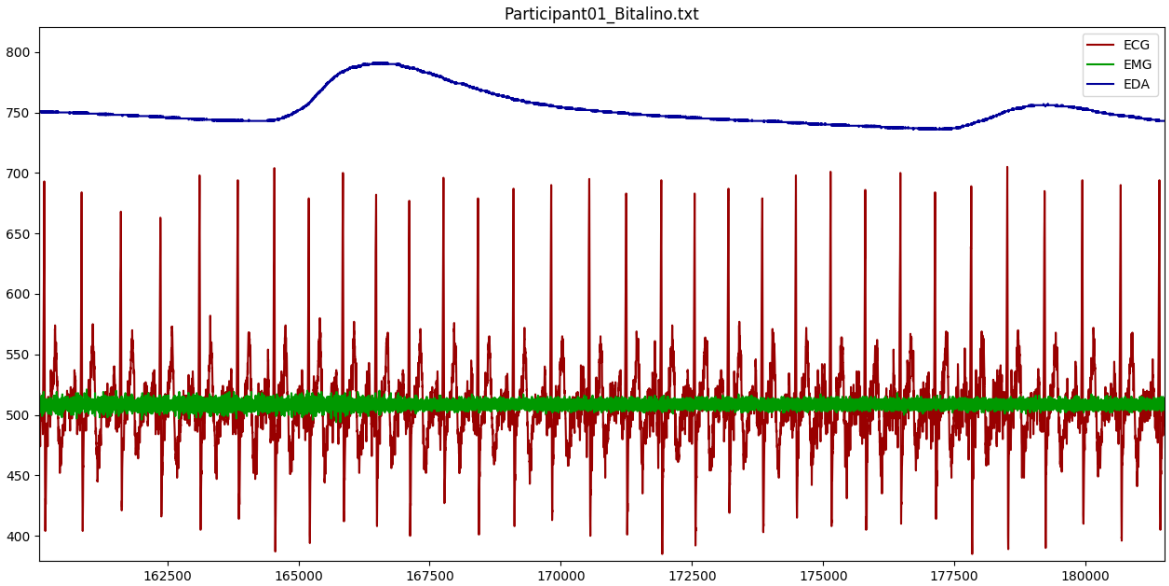

Figure 10 – Bioelectrical signals plot from data acquired by Eyempact acquisition module.

We call the above representations “Exploratory” because they were implemented in a way that the user can explore them. The user can zoom in and zoom out in each visualization, through the created widgets in the case of gaze and fixations, and through the *matplotlib* integrated navigator tool for the physiological signals. This functionality is more useful to watch in detail the signal plots than the scan paths; e.g. **Figure 10** was zoomed in multiple times so that it was possible to identify EDA variations and ECG peaks. For the scan path visualizations, zooming in and zooming out has little interest, even more if you are working with big datasets where you will end up with a cluttered image full of data points; setting a lower transparency can help in these cases but it is often not enough if you are looking for details along the time. This way, we have created simple widgets that help the user traverse the gaze and fixations visualizations.

Heatmaps and video replays

The system also provides the visualization of attention points using heatmaps. Alongside the scan path visualizations, these are also a very popular gaze representation. Analogously to the previously presented visualizations, heatmaps have some customizations available too. The user can set different data point sizes, transparency and choose from a set of different colorization styles. All the visualizations can be seen in video like replays; at the moment only the bioelectrical signals cannot be actually processed and saved as a video file.

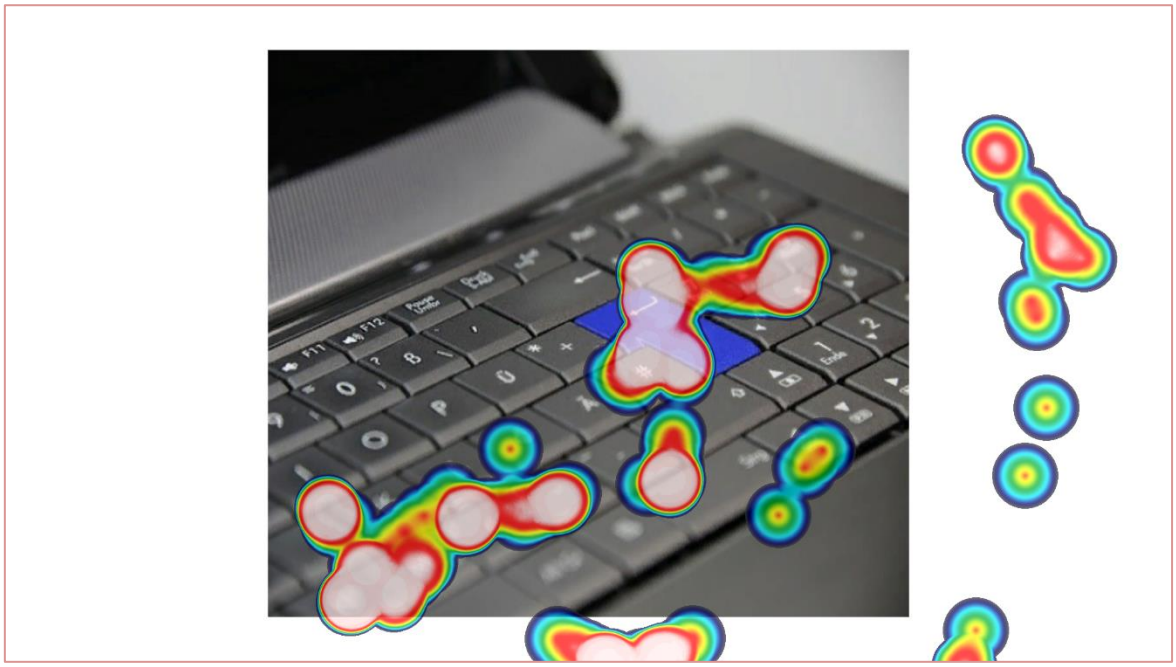

Figure 11 – Gaze heatmap visualization overlay.

4 Where's Wally? – Proof of concept application

We implemented a game named "Look at Wally" inspired by "Where's Wally" children books aimed to assess and show what can be done with the technology through the Tobii Unity API. Our concerns were mainly if it was possible to record the position of one's gaze in a screen and, in positive answer, if there were any constraints in the collection process.

The objective of the game was to, as the name suggests, look at Wally's face to clear each level. The entire interaction with the application was gaze based, i.e., from the moment the application is run the user only needs its gaze to control it.

The application is composed by four levels with randomly selected Wally images, and has a time limit to clear each one. Level zero introduces Wally to those who might not recognize the character; the user still needs to look at its face to clear the level. The remaining levels present an increasing difficulty and slightly higher timers, to give the user extra time to raster scan the image (if that was the selected method to find Wally).

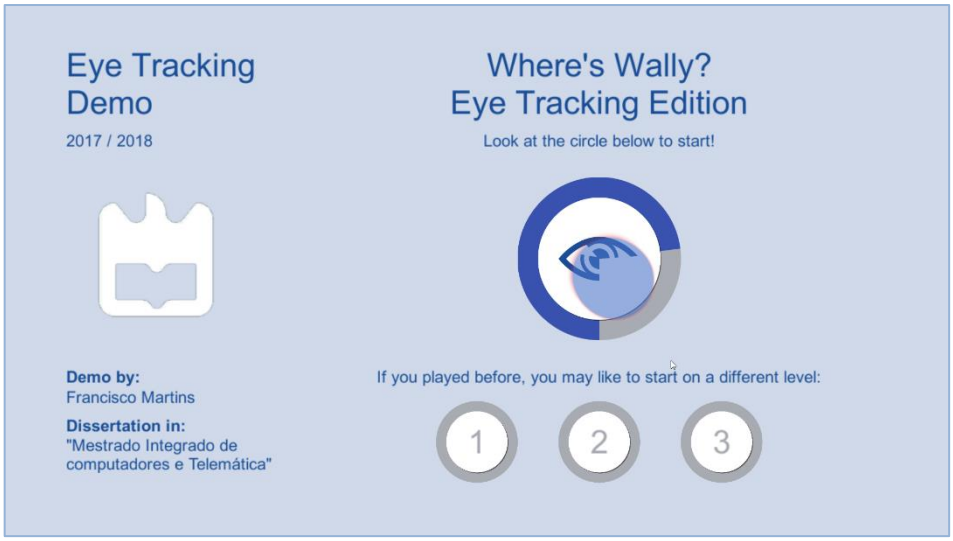

Figure 12 – *Where's Wally* application menu. User is opting to start the game on level zero.

On the main menu (picture above), the user is presented with four UI elements, to start the game (level zero) or select one of the other levels. All interactions with the application require the user to fixate its gaze for a couple of seconds, either fixating on the UI or Wally's face (when they find it). This way we make sure of the users' intentions: start the game, retry (in case the timer runs out) or return to the main menu (on game completion). The fixation was accompanied with a radial progress bar as feedback, also represented in the image above. This feedback was delayed for a couple hundred milliseconds while playing, so that the solutions would not be revealed if the user simply raster scanned the image quickly, this way we guaranteed that the user really found Wally.

This application was used and showcased in three events at the University of Aveiro: Students@deti[10], UA Open Campus[11] and Xperimenta[12]. In general, the participants had no problems in understanding and utilizing the application. Some participants required eye tracker calibration, what is expected, principally for people with glasses/lenses and taller or shorter than average. To show that the technology really works, and that the computer "knows" where the user is looking at, a bubble overlay was shown, pointing out where the user was looking at (this is a built-in tool of the Tobii eye trackers). Although, the application works without this feature by default, as this may be nauseating for some people (more even if the eye tracker is not calibrated for the person).

Figure 13 – User found Wally (bottom centre of the image).

From the "Look at Wally" game it was possible to test the collection process and confirm the basic steps needed for both setup and data collection using Tobii HW and API SDK. It was also concluded that eye tracking is a viable solution for computer interaction, even using a low-end eye tracking device.

[10] ***http://studentsandteachersdeti.web.ua.pt** - students@deti is an open day at Department of Electronics, Telecommunications and Informatics where students showcase their projects developed over the year.*

[11] ***http://opencampus.web.ua.pt** - Specially thought for high school students, UA Open Campus is a 3-day event where they can attend to speeches and workshops all around the University of Aveiro campus..*

[12] ***http://xperimenta.web.ua.pt** - Xperimenta was an open event for students, teachers and public in general to "Xperiment" and get to know projects developed in the University of Aveiro.*

5 Eyempact implementation

The Eyempact system, besides the implementation of the three modules, implied several decisions ranging from which data sources and what technologies to use, to the actual specifics of the system, e.g. definition of data formats and acquisition configurations, as well as which visualizations options and metrics should be supported. Some decisions can be already observable in the following figure.

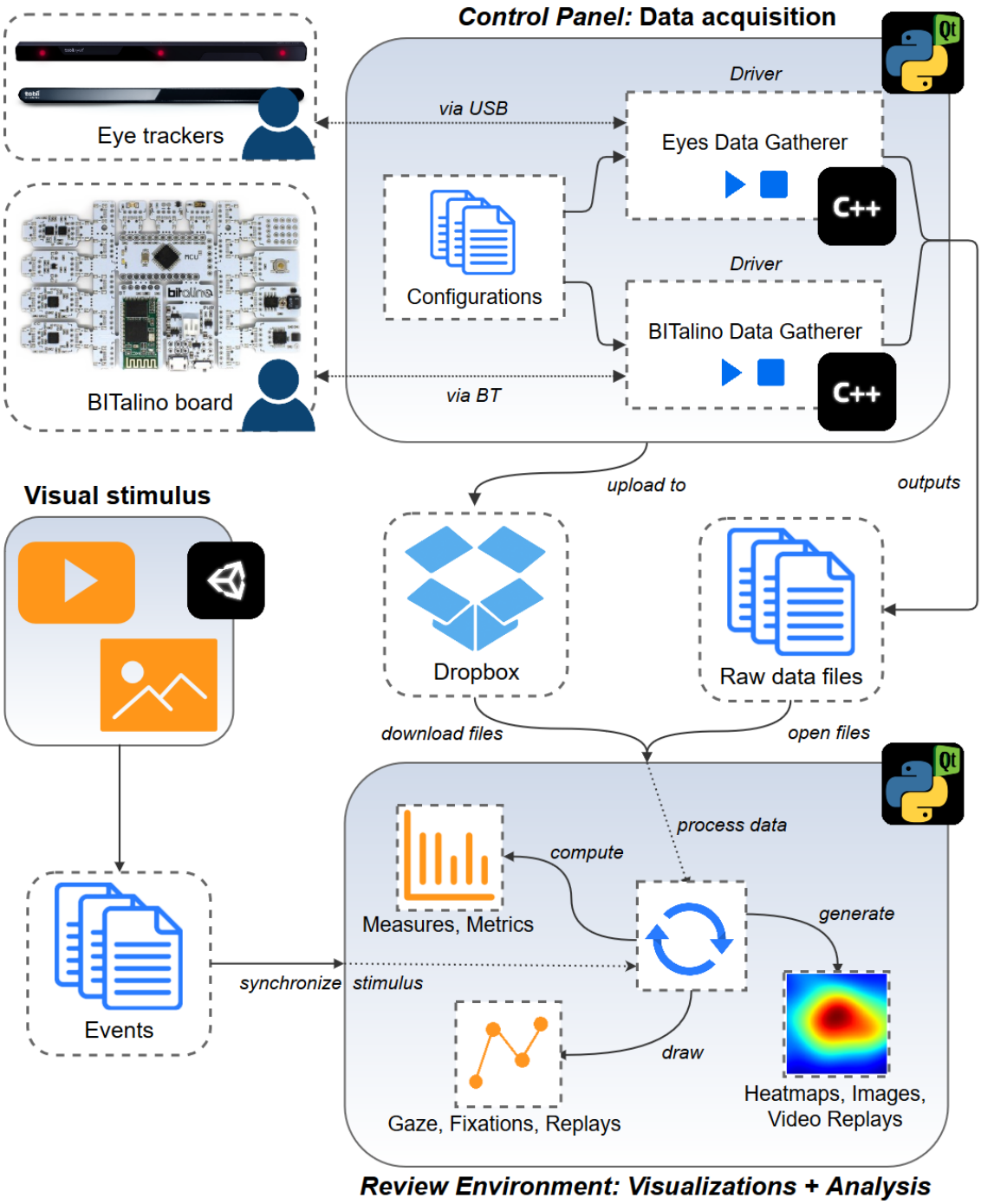

Figure 14 – Eyempact architecture and technologies (detailed view).

5.1 Data sources

Eyempact data sources comprise eye tracking and bioelectrical signal sources. For eye tracking, Eyempact has integrated the use of two eye trackers of the Tobii company, specifically eye trackers of the gaming line: EyeX and 4C models (**Figure 15**). As for the bioelectrical signals, Eyempact uses the BITalino (r)evolution Board Kit bluetooth (BT).

5.1.1 Tobii eye tracker

The first available sensor was Tobii EyeX, later on Tobii 4C was released and utilized to accomplish this dissertation goals. Both eye trackers are low cost when comparing to professional ones from the same company and others; actually, they are the most affordable eye trackers on the market.

Figure 15 – Tobii EyeX (top) and Tobii 4C.

Source: https://tinyurl.com/ycc8mz5m | https://tinyurl.com/ybl7954a

The EyeX version specifications provides a sampling frequency of 60Hz, on the other hand the 4C has a sampling rate of 90Hz, this meaning that we have available approximately sixty and ninety gaze points to work with at any given second, respectively for each eye tracker. For comparison purposes, at the moment of writing Tobii only has three standalone eye tracking devices, besides the before mentioned EyeX and 4C. These three devices are from their professional line and have 30Hz/60Hz, 60Hz and 120Hz data rates (Tobii models Pro X2, Pro Nano, Pro X3-120 respectively).

Considering the HW, Tobii's gaming eye trackers are comparable to professional ones. However, they have two drawbacks when it comes to SW development for these devices. There is a compatibility problem; they are only compatible with Windows OS (versions 7, 8.1 and 10). Besides that, the biggest drawback is that not all Tobii's API's are available for the gaming line. We have only available C/C++ and .NET API's, and for game development we have available API's for the two biggest game engines, Unity and Unreal engine. Tobii has support for many more programming languages and advanced API's dedicated for analytical purposes, but they are only available for the professional line.

5.1.2 BITalino board

The physiological data was collected using the BITalino board (**Figure 16**). As stated by the BITalino website: "BITalino is a low-cost toolkit to learn and prototype applications using body signals."[13]; which accomplishes the goal to develop a low-cost system for behaviour and physiological monitoring. The BITalino board version utilized can acquire ECG, EDA and EMG signals at a rate of 1000Hz and has a resolution of ten bits (0-1023).

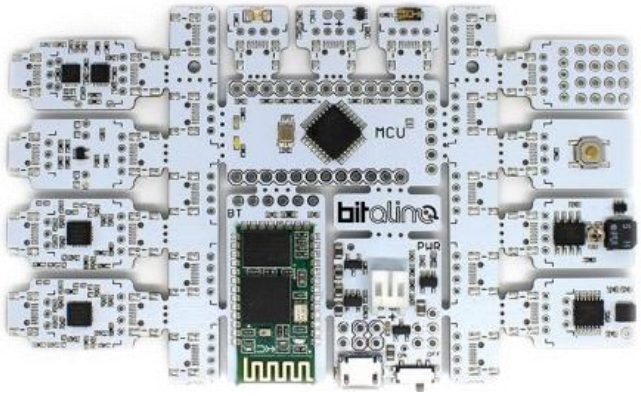

Figure 16 – BITalino board.

Source: **https://tinyurl.com/yapze8k4**

Another device available in our laboratory, and considered to be integrated, was VitalJacket (VJ). VJ is a medical certified wearable device[14], capable of acquiring ECG. This device is less intrusive than BITalino, since it can be placed in the users' pocket e.g., after the electrodes application, allowing movement flexibility. However, on traditional eye tracking studies, making use of a standalone eye tracker, the user is expected to stay relatively in the same position, around the eye tracking reachability, so it is not a problem to leave the BITalino board standing still in the table. Besides that, the sample rate of the VJ (500Hz) is half of the BITalino and has less resolution, seven bits (0-127).

Figure 17 – BITalino board kit and VJ box.

Source: **https://tinyurl.com/y7alc3a4** | **https://tinyurl.com/yb2s8ct9**

[13] ***http://bitalino.com/en/*** *- BITalino official website.*

[14] ***http://www.vitaljacket.com/*** *- VitalJacket official website.*

5.2 Data types and formats

Other concern present while integrating the sensors was to define data storage formats to ensure that the acquired data was still usable regardless of using or not Eyempact to process and review it. For this purpose, it was decided to encode all data in textual formats which are described in more detail in this section. By default, in each full acquisition comprising both eye tracking and physiology data, we end up with five data files presented in the table below.

Data file	Description
Gaze	Stores the points on the screen where the participant looked at.
Fixations	Stores small groups of points (fixations), from a small area on which the user fixated its gaze. The start and end of a fixation are marked.
Status notifications	If the user looks away from the screen or gets out of the eye trackers bounding box a status notification is stored.
Eye coordinates	Stores the eyes 3D positions in relation to the centre of the screen and in a virtual bounding box (VBB) in front of the screen with one unit volume.
Physiology	All the physiological signals are saved in a single data file. By default, the signals that are saved are EMG, ECG and EDA (saved in this order).

Table 4 – Data file types and descriptions.

In the first line of each data file we have the participant name and date. Every file has a local timestamp for every line of data (current time with format HHMMSSFFF). The data files are structured in the following way:

Local timestamp (TS)	X screen coordinate	Y screen coordinate	Eye tracker TS

Table 5 – Gaze data file format.

Local TS	Fixation status	X screen coord.	Y screen coord.	Eye tracker TS

Table 6 – Fixations data file format.

The gaze and fixations files are the most important data files. As seen in the previous chapter, Eyempact uses these files to draw various compelling representations, to extract basic information (measurements) and calculate eye tracking metrics.

Local TS	Status code

Table 7 – Status notifications data file format.

The status notification file allows Eyempact to replay the gaze and fixation files data with high precision. This is, if the user looks away for a second, there is no gaze and fixation data in that second, so the replay should not draw any point and an indication that the user looked away should be presented, this indication is only possible with this data file.

Local TS	Eye	X mm	Y mm	Z mm	X VBB	Y VBB	Z VBB

Table 8 – Eyes coordinates data file format.

The captured eyes coordinates enable us to create visualizations on the users' movements. We can obviously create a 3D visualization, but we can also create 2D visualizations. For example, we could represent each eye with a circle or oval shape, increase its' size as the user gets closer to the screen and decrease the size if the user gets away from the screen. In combination with the status notifications file and head tracking (future work), we can create high fidelity replays of the user' movements.

The eyes coordinate data can also be used to get some information like e.g. average distance from the screen; and even derive a metric that tells how much the user moved in a period of time. This is certainly an invaluable data file; however, this was not very explored in the present work. When Tobii 4C was release head tracking was introduced, although this feature was not explored either.

Local TS	Sample No.	Signal 1	Signal 2	...	Signal N

Table 9 – Physiological signals data file format.

Finally, Eyempact also uses the physiological signals files to represent the acquired physiological data. Eyempact can also replay this type of data. As for the most well-known physiological signal (ECG), Eyempact can calculate the HR. Analogously to the eye coordinates data file, there is much to be done regarding these signals' information extraction.

5.3 Data acquisition

To gather data from the eye trackers (Tobii EyeX and Tobii 4C) and the BITalino board, two separate standalone applications were developed. The eye tracking data gatherer application was simply called "Eyes Data Gatherer" and the BITalino application, "BITalino Data Gatherer".

For the eye tracking program, C/C++ and Unity API's[15] were used and tested. The Unity API includes a "Scripting reference", that serves as a walkthrough through all the API capabilities, with C# example scripts. However, the Unity API was not really built to gather and save eye tracking data streams, as it was already mentioned, it is really directed to game developing and real time use of the data streams. This way, it was decided that the C/C++ API[16] would be used for the data gathering, that is a low-level API and enables us to easily store the data we want.

Some sample examples were provided with the API as Visual Studio projects, making use of what was provided, the end application was developed as a console application in a Visual Studio console project (the IDE version used was 2015, update 3).

Eyes Data Gatherer (EDG) connects to the eye tracker if it is connected to the PC through USB and "listens/subscribes" the eye tracker data streams and events. EDG gathers and saves in files the faze points, fixations, eyes coordinates and gaze status using the Eyempact formats.

For the physiological signals acquisition, the BITalino C++ API was used, Unity API[17] was also explored but only as means of real-time visualization (plot signals). The implementation of the *BITalino Data Gatherer* (BDG) program was straightforward, since the provided example in the official BITalino website was already functional, it was only needed to implement a way to save the data into files and all the settings behaviour.

Eyempact also has integrated the possibility support video recording using a *GoPro*[18] action camera by using an unofficial wireless API[19] to remotely start, stop and download video after an Eyempact session. Video in this context can provide an independent source for protocol validation and results interpretation. Although out of the scope of our work, the ability of having video may provide another source of data to explore namely emotions extraction based on video and audio processing.

---

[15] ***https://developer.tobii.com/consumer-eye-trackers** - SDK's and doc. can be found here.*

[16] *Tobii has been updating its developer websites, API's and documentation, very frequently. Therefore, the exact C/C++ API used in this work is not available anymore. The new substitute for the low-level API can be found in the link above and it is named "Tobii stream engine".*

[17] ***http://bitalino.com/en/development/apis** - BITalino API's can be found here.*

[18] ***https://gopro.com** - GoPro is a well-known device in the action cam department, this is their official site.*

[19] ***https://github.com/KonradIT/goprowifihack** - The utilized API is hosted in this github project.*

5.3.1 Measurements and metrics

In Eyempact, we present general information and various metrics about the captured data, at any given timestamp, which are presented in this section. In the *Review Environment*, the user has the ability to traverse the files and explore each data point. Information available at each timestamp:

- Number of points represented;
- Travelled distance up to the represented point, instant velocity and acceleration (in pixels, pixels/s, pixels/s^2) – these can also be plotted; they can be a good indication of the user's attention along the time.

Some measures referent to the whole data file. This is information that is not modified by traversing the file:

- Total number of points and acquisition time (in seconds);
- Average gaze points captured (points/second) – this can also be good to evaluate attention, or it can be an indication of a bad calibration e.g. if the number is too low;
- Total travelled distance;
- Average velocity and acceleration.

An AOI can be easily set by the user by dragging the mouse in the visualization space or manually inputting the desired coordinates. For the AOI metrics we focused on the fixation ones. It is available to the user the following information and metrics:

- Time to first fixation (TTFF);
- Number of fixations in the AOI;
- Number of fixations out of the AOI;
- Time spent in AOI.

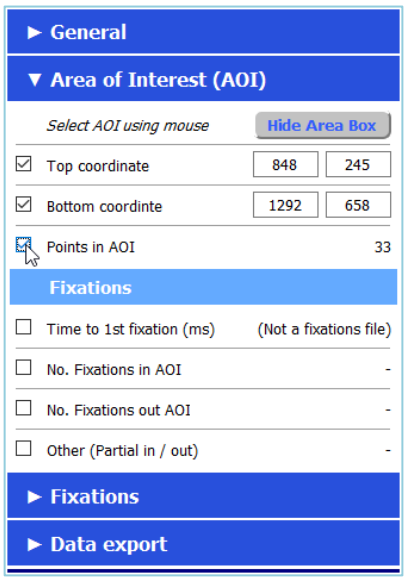

Figure 18 – Metrics panel view. More details on the panels in Appendix A.

Finally, some dedicated metrics for the fixations:

- TTFF (it can be or not in the AOI);
- Total number of fixations;
- Average fixation times (ms).

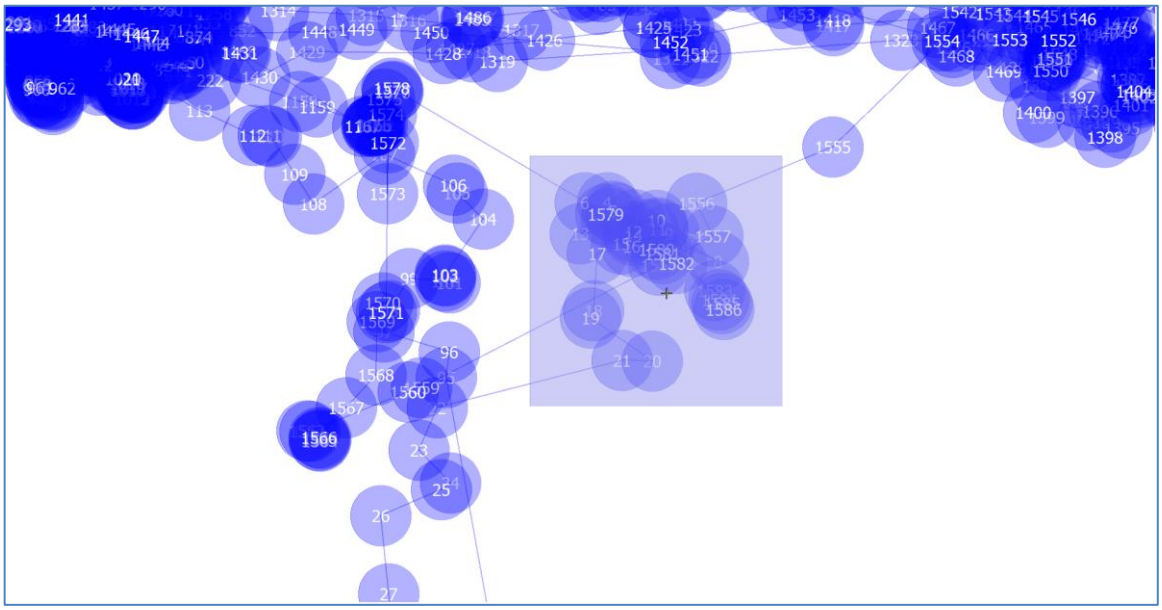

Figure 19 – User selected AOI is shown as a transparent rectangle.

5.3.2 Control Panel user interface

As we will see in the following section, simple configuration files were used to configure the acquisition process. However, editing a configuration file manually is a very rudimentary way of changing program settings and more importantly it is very prone to errors. This way, a simple interface was built for this effect and to start and stop our acquisition programs.

In the main window, we can start, stop and configure each program individually as we can see in the **Figure 20**. The participant name is a shared configuration, for the acquisition programs, so it appears on the main interface. The user also has the option to start and stop all programs simultaneously. Below these actions, we have a simple indication in form of icons, of which program(s) are running and which are not. Further below, we have a togglable log section, which entries are simply the actions taken in the interface and connection logs (when a connection with the eye tracker / BITalino was made or interrupted).

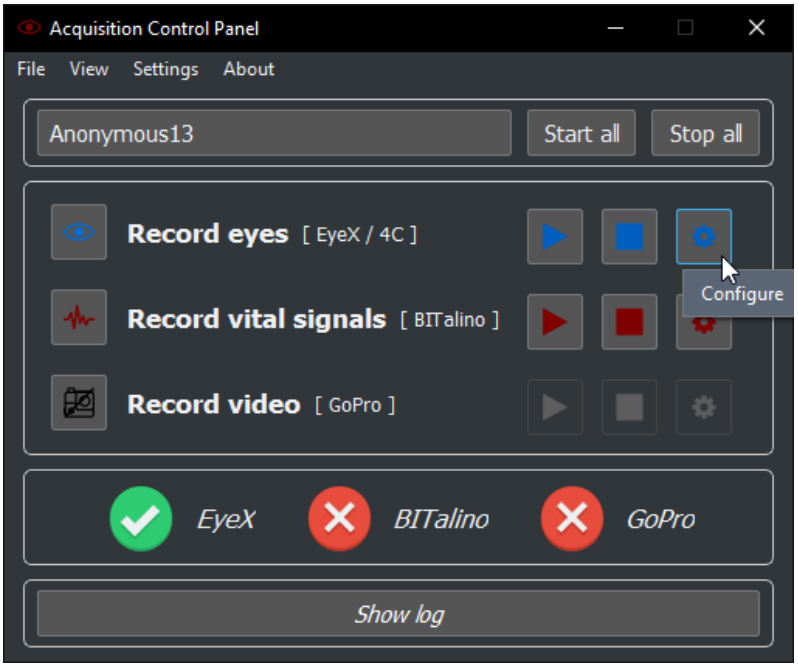

Figure 20 – Control panel application main user interface.

The configuration dialogs, where the user can edit the acquisition programs behaviour, are presented in the **Figure 21.** Eye tracking program configurations on the left, BITalino program configurations on the right.

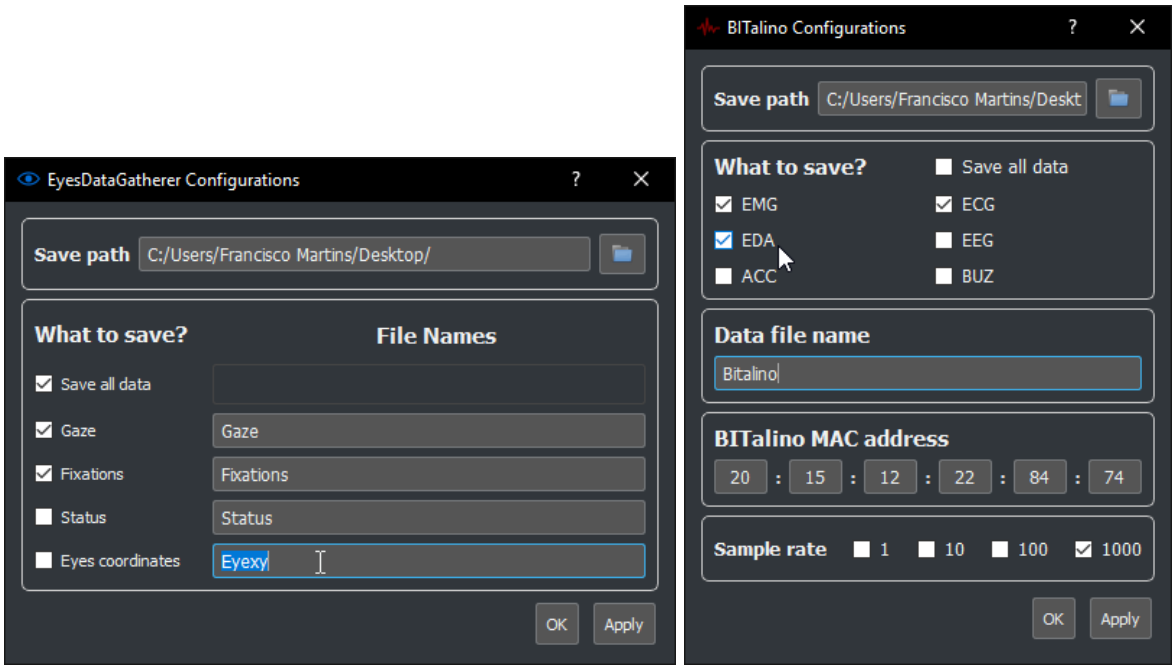

Figure 21 – Configuration dialogs. Eye tracking (left) and BITalino (right) configurations.

This simple interface can be easily updated and/or extended in the future, for example to start a key logger program, if further information on the user's interactions are needed.

5.3.3 Configuration files

The configuration of a data acquisition session in Eyempact is supported by configuration files. These files are text-based and can be manually edited, although it is highly inadvisable to do so. For that purpose, we provide the *Control Panel* GUI, previously presented, that directly edits the configuration files and performs validation on the user input.

For each of the sensors we provide a configuration file definition and respective configuration context menu in the *Control* Panel. An example of the EDG configuration file, with all the editable settings is presented below.

```
EyesDataGatherer configuration file
-------------------------------------------------------------------------
 1) Participant name         >Participant007
 2) Write path               >C:\Users\Francisco Martins\Desktop\
-------------------------------------------------------------------------
 3) Save all data            >1
 4) Save gaze                >0
 5) Save fixations           >0
 6) S. status notifications  >0
 7) Save eyes coordinates    >0
-------------------------------------------------------------------------
 8) Default gaze file name   >Gaze
 9) Def. fixations file name >Fixations
10) Def. status notif. file  >Status
11) Def. eyes coords name    >Eyexy
-------------------------------------------------------------------------
12) Use global file          >0
13) Show terminal console    >1
14) Ask for participant name >0
-------------------------------------------------------------------------
```

Figure 22 – Example of configuration file for the *Eyes Data Gatherer* program.

The configuration files were developed early on the dissertation work, at the moment of their development, it was not planned to build GUI based tool for the typical user, and that is the reason why we chose to use readable text in our configuration files. These are very simple, and only comprise about a dozen configurations each, so we did not feel compelled in converting them into a more modern format like JSON or YAML, leaving them in a way that can still be realistically manually editable by anyone.

A brief description of each configuration available for the EDG is presented in the following table.

Setting(s)	Description
Participant name (1)	The participant name precedes the default names in the file name (it can be left empty if not needed).
Write path (2)	Path to which all the generated files are to be saved.
Save all data (3)	If enabled, all data streams are saved into files, if not, the following configurations tell which data is to be saved.
Save gaze, fixations, status, coordinates (4-7)	If "Save all data" configuration is disabled, these configurations specify which data streams are saved.
Default file names (8-11)	These configurations are used to define the name of each data file. The final name format will be: "ParticipantName_DefaulfFileName.txt".
Use global file (12)	Enabling this configuration enables data to be saved in the default file names (without the participant name); this is useful if you want to make some tests.
Show console (13)	Show / hide console with log messages.
Ask for participant name (14)	If console is showing, it is possible to enable this setting to ask for the participant name on console.

Table 10 – *Eyes Data Gatherer* program configuration descriptions.

Analogously to the EDG program, a simple text configuration file (**Figure 23**) was created to configure the acquisition performed with Bitalino using the BDG program.

```
>>> BITalinoDataGatherer configuration file <<<
--------------------------------------------------------------------------
 1) Participant name  >Participant007
 2) Write path        >C:\Users\Francisco Martins\Desktop\
 3) Default data file >Bitalino
--------------------------------------------------------------------------
 4) BITalino MAC      >20:15:12:22:84:74
 5) Sample rate       >1000
--------------------------------------------------------------------------
 6) Save all          >0
 7) Save EMG          >0
 8) Save ECG          >0
 9) Save EDA          >0
10) Save EEG          >1
11) Save ACC          >0
12) Save BUZ          >0
--------------------------------------------------------------------------
```

Figure 23 – Example of configuration file for the *BITalino Data Gatherer* program.

A brief description of each configuration available on the BTG is presented in the table below:

Setting(s)	Description
Part. name, write path (1-2)	Same as EDG.
Default file name (3)	All data is saved to the same file name, e.g., using **Figure 23** configurations, the data file would be named: "Bitalino.txt".
BITalino MAC (4)	The BITalino board connects to the PC via BT, in this configuration the equipment MAC address should be inserted or updated.
Sample rate (5)	Sample rate at which the board should transmit in Hz.
Save all, EMG, ECG, EDA, EEG, ACC, BUZ (6-12)	Defines what is going to be saved; although BITalino supports all these signals, only EMG, ECG and EDA were acquired and tested.

Table 11 – *BITalino Data Gatherer* program configuration descriptions.

5.4 Review Environment options

In this sub-section we will present some of the main decisions made while developing Eyempact *Replayer* program as well as present some implementation details on topics previously presented.

5.4.1 Interface: OpenCV vs PyQt

At start, a simple OpenCV program was developed. Making use of configuration files and a terminal interface (**Figure 24**), it was simple to develop an application capable of representing the gaze points and fixations present in a file, as well as to replay and save videos of the recorded data.

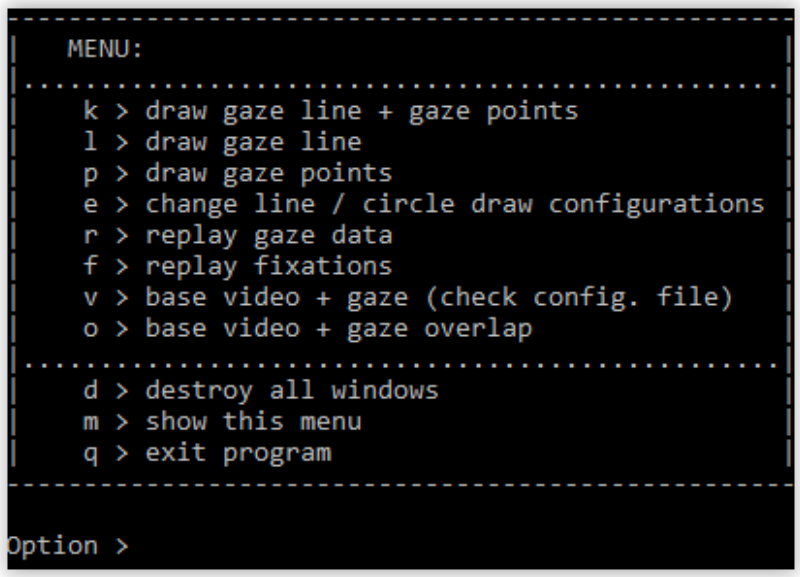

Figure 24 – OpenCV *Review Environment* program menu interface.

However, OpenCV has a very limited support for interface widgets, this way, adding functionalities to this program meant adding more entries on the console menu and/or more configuration files. This would lead to a very unpractical application, and a not so straightforward to use application in terms of usability for the typical user.

To solve this problem, it was decided that going forwards we would use an interface framework. After deciding Python as the main project language, we had two options to choose from: PySide[20] and PyQt[21]. These packages bring to Python all the functionality of the well-known Qt application framework (that uses C++ as its main language). It was decided that PyQt5 was the right package to use, at the moment it supported the latest version of Qt (now both projects do) and it has in our opinion a simpler syntax for the "Signals & Slots"[22] mechanisms, needed for any Qt project. With PyQt, we were able to develop a rich and intuitive GUI by making using of the big selection of widget options already available and developing new widgets when needed.

5.4.2 Widgets and menus

After choosing which technologies to use, decisions relative to what functionalities and how they should work had to be made. In the following paragraphs, it is presented in detail some of these functionalities and inherent decisions, made along the way.

Traverse scan path visualizations

When representing big gaze files, files in the thousands of points, you will end up with a "blurred" canvas, which may not be very interesting, as the main attention points may not be identifiable even with high transparency. To solve this problem, we implemented a way to traverse the points in the file. Using two slider widgets it is possible to choose which range of points you want to represent, by setting e.g. one thousand in the first slider and one thousand one hundred in the second slider (**Figure 25**). In this case, only the one thousandth point and the hundred following points in the file are represented.

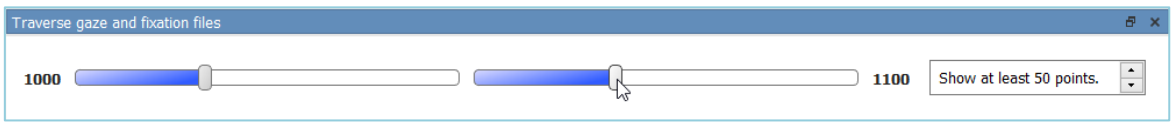

Figure 25 – Widget used for traversing gaze and fixation files.

[20] ***https://pypi.org/project/PySide2*** *- PySide2 is the most updated package of the PySide project.*

[21] ***https://pypi.org/project/PyQt5*** *- PyQt5 currently is the most up-to-date package of the PyQt project.*

[22] ***https://doc.qt.io/qt-5/signalsandslots.html*** *- Signals & Slots documentation page.*

You can also set a limit of points to be represented at any time. In the example image above, we have the sliders limited to a fifty points difference; as you move the first slider, the second one will be updating its' values maintaining a distance of fifty points at each moment, allowing the user to traverse the whole file.

Replay

The widget above gives the user the freedom to represent every interval of time possible and traverse the file at any pace. If the user intends to review the participants eye movements in a more realistic way, it is possible to use the replay widget (**Figure 26**) to start and stop the replay of the participants gaze and fixations files. For a precise replay, we still suggest the generation of a video replay. This replay operation averages the interval of times between the gaze points and fixations, and simply sets a timer to draw every average period. This way, situations where the user looked away for an interval of time for example, will not be accurately presented in this operation.

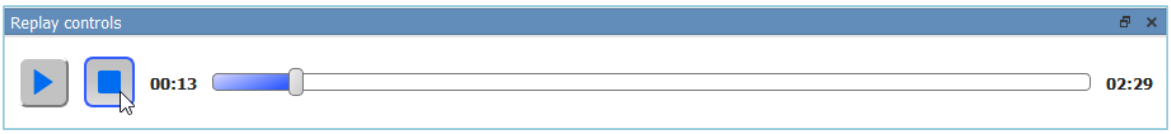

Figure 26 – This is the widget that allows the replay of gaze and fixation files.

The replay widget also uses the point limit established in the traverse widget to know the interval of points to draw at each timer iteration.

GitHub heatmap vs. Matplotlib heatmap

For the heatmap generation functionalities, several solutions were tried and experimented with, from Python packages to existing GitHub project solution. In the end, it was decided to keep two of the implementations: heatmaps using the *Matplotlib* package and an old GitHub project[23] for the core algorithm of the heatmap colorization. At first, we used the *Matplotlib* solution, because for a single image heatmap, it is way more performant than the counterpart. However, this method soon revealed some limitations: the gained performance would be lost with the implementation of the heatmap in the main interface canvas (so, this heatmap is shown in its own window); from this first point, it was also concluded that there was no good way of adding the stimuli in the background; and last but not least, this operation is very high level and thus it was not possible to tamper with the algorithm to improve performance for the heatmap video replay that we had planned to do beforehand.

[23] ***https://github.com/jjquy/heatmap*** *- GitHub project used for heatmap colorization.*

The utilized GitHub project was not up utilized as is. Firstly, it was converted from Python 2 to Python 3; secondly it was built for a very specific purpose, so the code irrelevant for our purposes was removed; lastly, it was not optimised, it had useless blocks of code. All in all, from this project we utilized the relevant code, mainly the colorization algorithm and colour schemes. After this code clean-up, the algorithm was improved to be more performant for video generation; in simple terms, instead of colorizing each heatmap frame every time, only the different points in the range are colorized (the new data points, and the ones being removed).

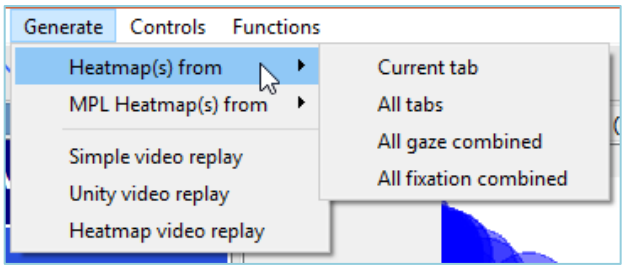

Figure 27 – Generate menu. Possible heatmap and video generation operations shown.

In Eyempact's *Review Environment* it is possible to simply generate a heatmap on the current shown tab (a new tab with the heatmap is created); generate a heatmap tab for every open tab; combine the gaze of all opened tabs into a single heatmap (useful to create group heatmaps) and combine all the fixation tabs in a heatmap.

6 Study on tobacco packages health warnings

In order to evaluate our solution, we adopted a psychophysiology experiment protocol in a real case study scenario. The case study idea was proposed by a psychology student in the context of several existing collaboration with DEP – Department of Education and Psychology of the University of Aveiro.

The main objective was to assess the psychophysiology emotional impact of images use to dissuade smoking in tobacco packages in three different groups: smokers, ex-smokers and non-smokers. The initial assumption was that the use of the current tobacco package images could have a lighter reaction from the smoker group, since they would be familiarized with the set of images. This way, it was given to us access and consent to use new and uncirculated tobacco pack images in our experiment.

The role of Eyempact was to monitor the experiments and allow combining both the physiological response and the image observation monitoring i.e. eye tracking while observing the tobacco package images and other unrelated images (neutral, positive, negative) to test the assumption on the impact of tobacco packaging images in the different groups emotional responses.

Figure 28 – Example images used in the experiment, one of each category. Neutral (top left), positive (top right), negative (bottom left) and smoker package example images (bottom right).

6.1 Experiment protocol and details

This experimental protocol included the visualization and evaluation of fifty-two images. The evaluation of the images consisted in the self-evaluation (of each participant) in the scales of valence and arousal. The valence/arousal scale used was the SAM (self-assessment manikin) scale [17].

The images were divided in four distinct groups: neutral, positive, negative and tobacco packs, with equal number of images (thirteen). It was decided that the selected images would be shown randomly and that it would only be possible to show up to two consecutive images of the same group. Before the displaying of each image there should be a five second slot with no image presentation, to gather a baseline for the bioelectrical signals. The randomly selected image was displayed for six seconds. After this period, the self-assessment scaled would appear for filling (**Figure 29**– right).

A Unity application was developed to comply with the experiment conditions defined beforehand and as a tool to automatically save the required information. At the beginning of the experiment the participants were presented with a small questionnaire (**Figure 29**– left). In this questionnaire we ask in what group the participant fits, smoker, ex-smoker or non-smoker. In the case of the participant being a smoker or ex-smoker, an extra questionnaire would appear at the end of the experiment (after the displaying of all images). The participants were asked to answer the Fagerström nicotine dependence test [18], to evaluate their degree of dependence.

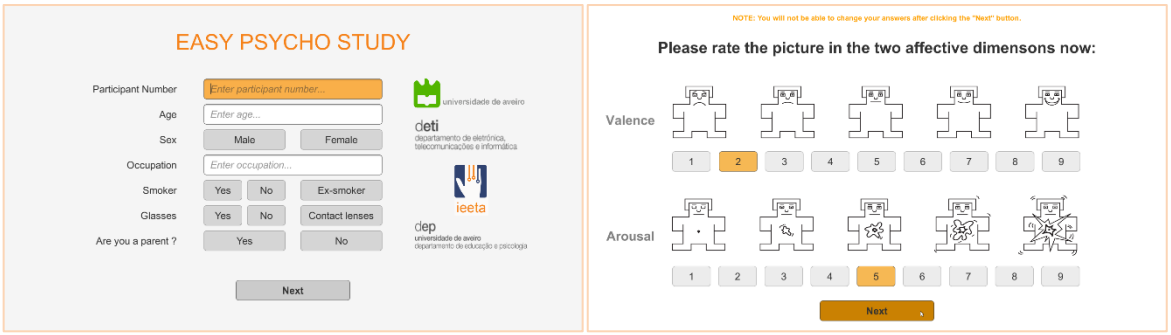

Figure 29 – Initial form and SAM scale print screens from the Unity application.

Summing up the experiment runs with the following steps for each participant:

- Initial participant information form
- For each one of the fifty-two randomly selected images:
  - five second waiting slot;
  - six second image display slot;
  - unlimited time to evaluate the image;
- Dependency questionnaire (for smokers and ex-smokers).

The application saves the following information for each participant:

- Initial questionnaire answers;
- Image identifier of the displayed image and respective timestamp;
- Image evaluation results;
- Smoker dependency questionnaire answers.

This image display timestamps were saved so that we could synchronize the images with the gathered data from the other programs (Eyex and Bitalino Data Gatherer) afterwards.

6.2 Setup

The setup was constituted by a laptop (MSI GL62 6QF laptop – for specifications reference), a secondary monitor, Tobii 4C, BITalino, two 3-lead accessory cables (for ECG and EMG), one 2-lead accessory cable (for EDA), eight electrodes, keyboard and mouse and an high-definition multimedia interface (HDMI) cable.

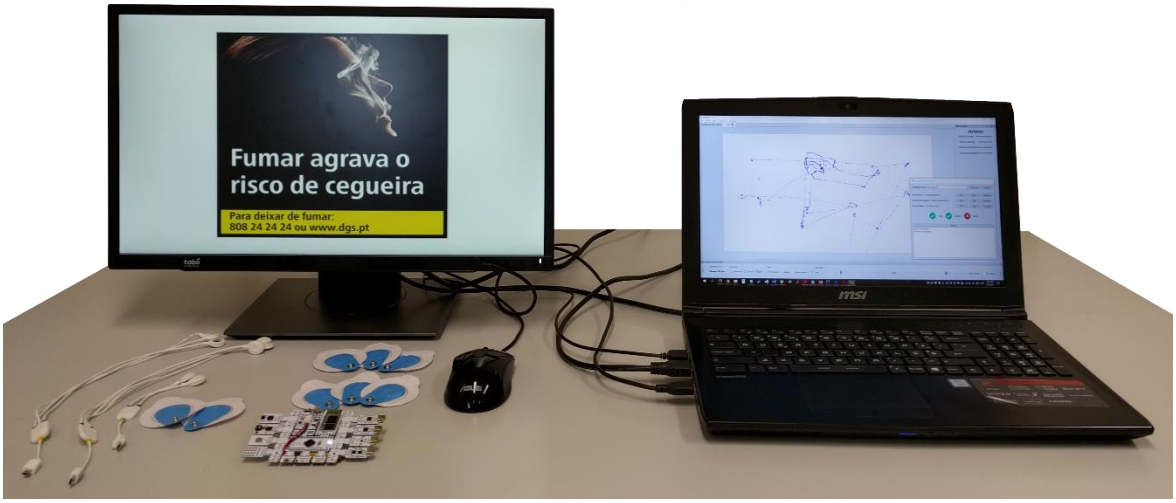

Figure 30 – Material disposition in the experiment setup.

After mounting the setup as in **Figure 30**, we would invite the first participant of the session to sit comfortably in front of the secondary monitor, so we could start applying all eight electrodes (**Figure 31**). After that, we would start recording, eye tracking and physiological signals simultaneously, using the Control Panel and run the Unity application. The participant would then fill in the initial form and read the SAM scale information, we gave further information or clarifications about the SAM scale or the experiment in general if requested by the participant.

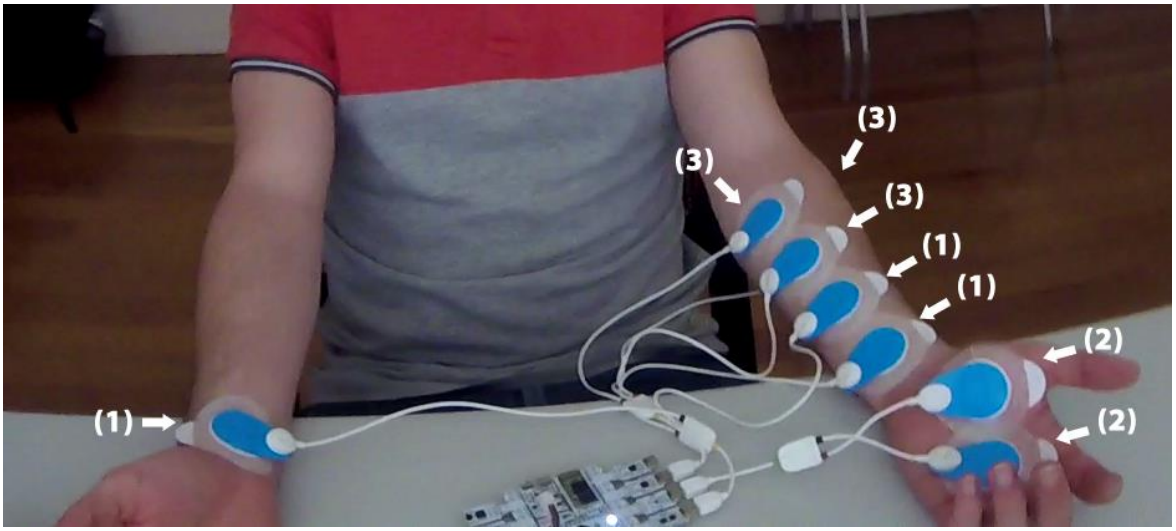

Figure 31 – Example of electrodes placement used in the experiment. (1) ECG, (2) EDA, (3) EMG. Description and decision making on the chosen electrode placement in Appendix B.

To minimize the interferences in the signal data (transmitted by the BITalino) we asked every participant to leave their electronics on a table far away from the setup. When actually running the experiment, the laptop was further away from the secondary monitor than showed above, the cables would be already connected to the BITalino board, and a keyboard was temporarily available for the participant to fill the initial questionnaire.

6.3 Hypothesis

In the beginning of the experiment idealization some hypotheses were defined. Some of the main hypotheses are presented below, and in front which data source should be used in the analysis to evaluate and answer them.

1) Health warning pictures in tobacco packages provoke the participants to look less at the screen than the other groups of images (Eye-tracking data);
2) Smokers try to avoid the tobacco images more than non-smokers or ex-smokers (Eye-tracking data);
3) Smokers present more pronounced physiological activation than the other groups when looking at the tobacco pack images (Physiological signals);
4) Valence and arousal self-evaluation are in accordance with the image avoidance (Eye-tracking data);
5) Valence and arousal self-evaluation are in accordance with the physiological activation (Participant evaluations and physiological signals);
6) There is a correlation between the tobacco image avoidance and the physiological activation (Eye-tracking data and physiological signals 1. and 2. comparison).

6.4 Information and collected data

In this experiment we had a total of forty-seven participants involved, twenty-six female and twenty-one male participants. The average age was twenty-three years old and the majority of the participants were students (thirty-nine). We had twenty smoker, twenty-two non-smoker and five ex-smoker participants.

Also, noteworthy that, twenty-three participants had their vision corrected (either by glasses or contact lenses), five of the participants were parents. Out of the twenty-seven Fagerström dependency test questionnaires answered, thirteen scored low dependency and no participant scored has having high dependency.

A table comprising the complete sociodemographic information is present in Appendix B.

We collected all eye tracking data types and the ECG, EDA and EMG physiological signals, as well as the Unity data referred above. When running the experiment, we recorded the data directly into three directories: one for eye tracker, one for the BITalino and one for the Unity application data.

Due to some unpredictable problems, and a couple of human errors, some data recordings were not completed with the presentation of the complete fifty-two image set. Therefore, only thirty-seven participant have a complete dataset.

6.5 Analysis results and discussion

The following image shows the distribution of the participant average arousal and valence self-evaluations for each of the images presented in the experiment. Valence quantifies how pleasant the stimuli is, in its turn arousal evaluation should reflect how strong of an emotion a stimulus unleashes in a person.

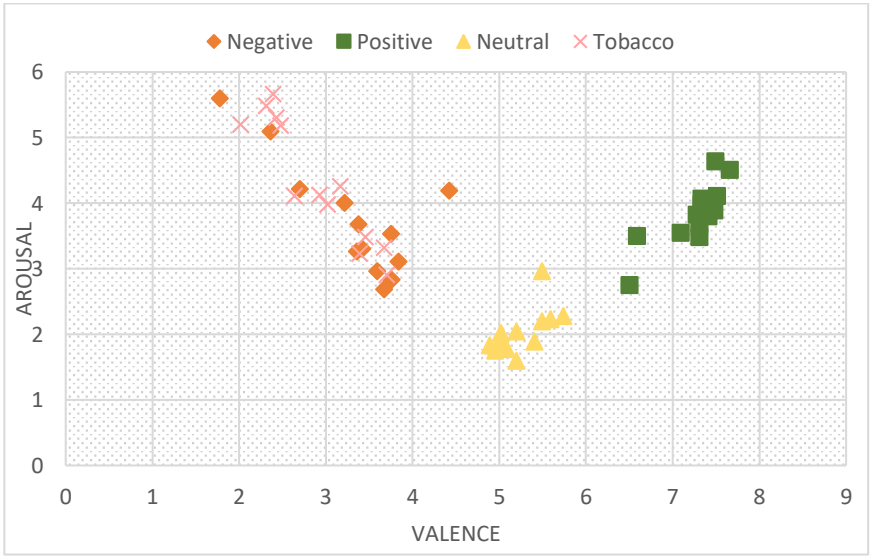

Figure 32 – Participant mean arousal and valence values for each image in the experiment.

These results are in accordance with our expectations, low values of valence are an indication of negative feelings, represented by the images in the "Negative" and "Tobacco" groups, high values of valence indicate positive feeling, represented by the images in the "Positive" group. In the **Figure 33**, it is possible to see a set of example emotions and their relation in the arousal-valence space, from those it is possible to associate our negative and tobacco package images to emotional tension and depression, and our positive images, to happiness, emotional content and relaxation. The remainder group, the neutral images, didn't trigger any emotional response, showing very low values of arousal. In general, all the arousal evaluations were on the low side, although the relation between the different groups are normal. Further confirming our good results in this area of affective evaluation, using the SAM scale, this study [19] published in the "Frontiers of neuroscience", with one hundred and twenty-two participants, and four groups of stimuli (including negative, positive and neutral stimulus), reached an extremely similar image in terms of distribution of average values in the arousal-valence space.

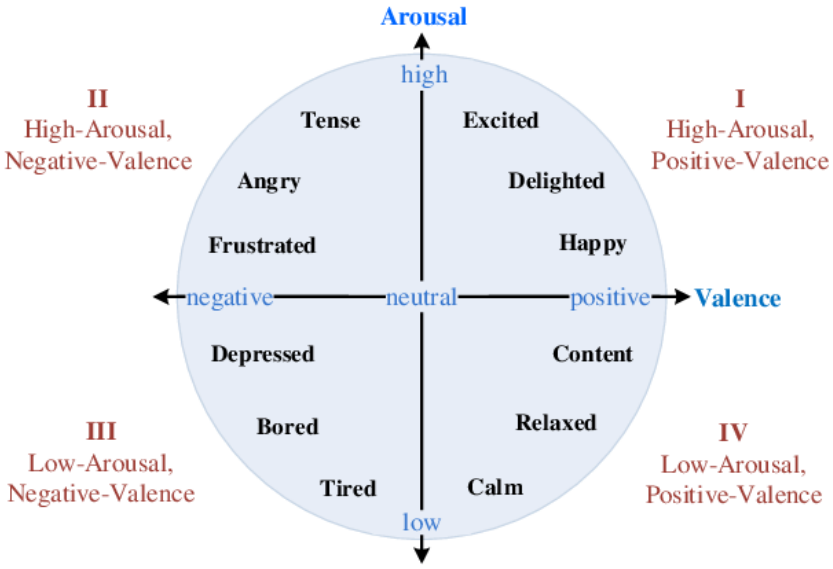

Figure 33 – Affective states represented in space by arousal and valence values; figure copied from [20].

With the conclusion of the experiment, the physiology dataset was reorganized to ease the analysis process. To start, the segmentation of the useful data was performed by "one-time use" Python script; it was defined as useful data for our analysis, eye tracking gaze on the display image slot and bioelectrical signals data on the baseline and display image slots. In conclusion, for each of the gaze, fixation and bioelectrical signals variables fifty-two data files per participant were generated, disregarding, data from the start and end of the experiment as well as the evaluation moment.

Afterwards, we proceeded to generate the final data files for analysis. All eye tracking metrics and measurements were exported from the *Review Environment*. Again, using a Python script, all this exported data was compiled, in conjunction with metadata, in a single eye tracking data file. For the bioelectrical signals, it was performed feature extraction via the Python library *neurokit*[24], analogously compiled in a single data file (extracted feature and metadata). The format of these files can be consulted in Appendix B.

6.5.1 Physiology results

Initially, many of the variables were studied using the boxplot resource, either grouping those variables by type of image or by type of participant and image group. In general, it was not observable significant differences between the different groups of images, apart from punctual examples, and small differences were observable between the groups of smokers and non-smokers.

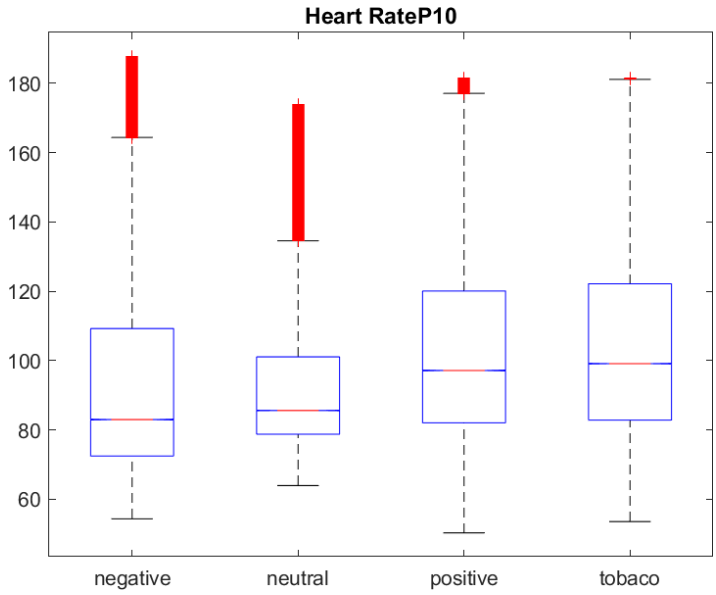

Figure 34 – Participant ten had higher HR for the positive and tobacco image groups.

The lack of differences between groups of images, could be explained by the low display time of each image combined with the randomness of the images, this leaves the participant's body with very little time to trigger emotions and to change between emotional states (represented by the different groups) if the participant does not take time to carefully and calmly evaluate each of the images. In this case, participant ten was a smoker and the tobacco package images did seem to unleash a higher physiological activation.

[24] ***https://github.com/neuropsychology/NeuroKit.py*** *- Neurokit github page.*

The small differences reported above were observable e.g. in the EMG entropy and number of gaze data points over the image area. Both examples are presented in the figures below.

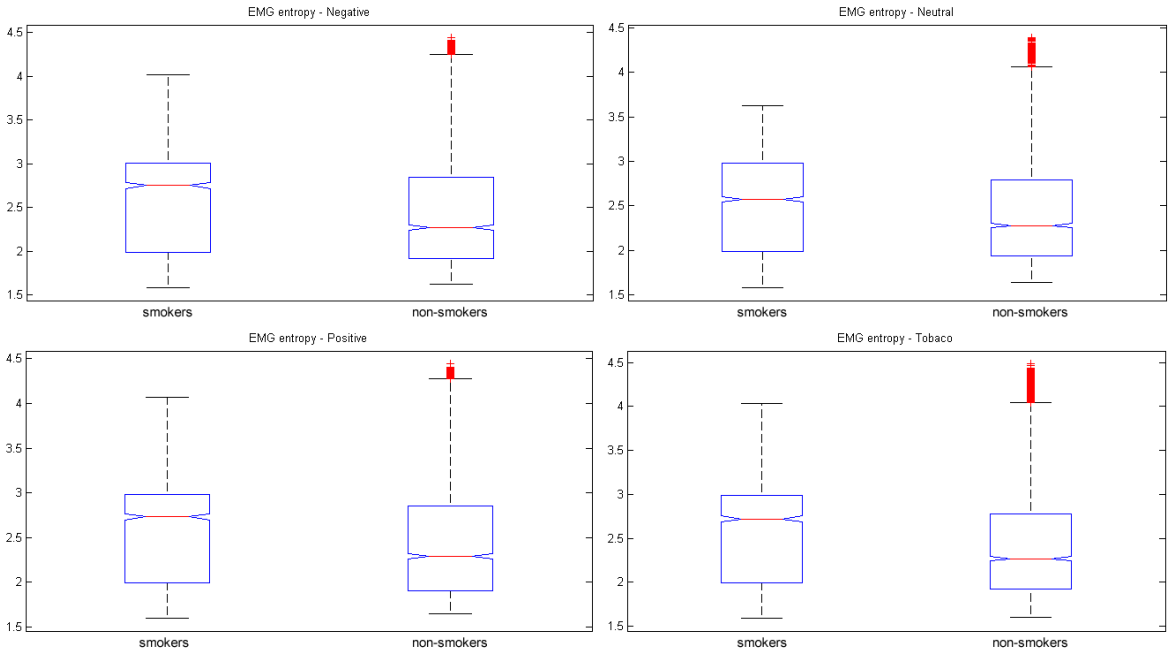

Figure 35 – Higher EMG entropy for smokers versus non-smokers in all the image groups.

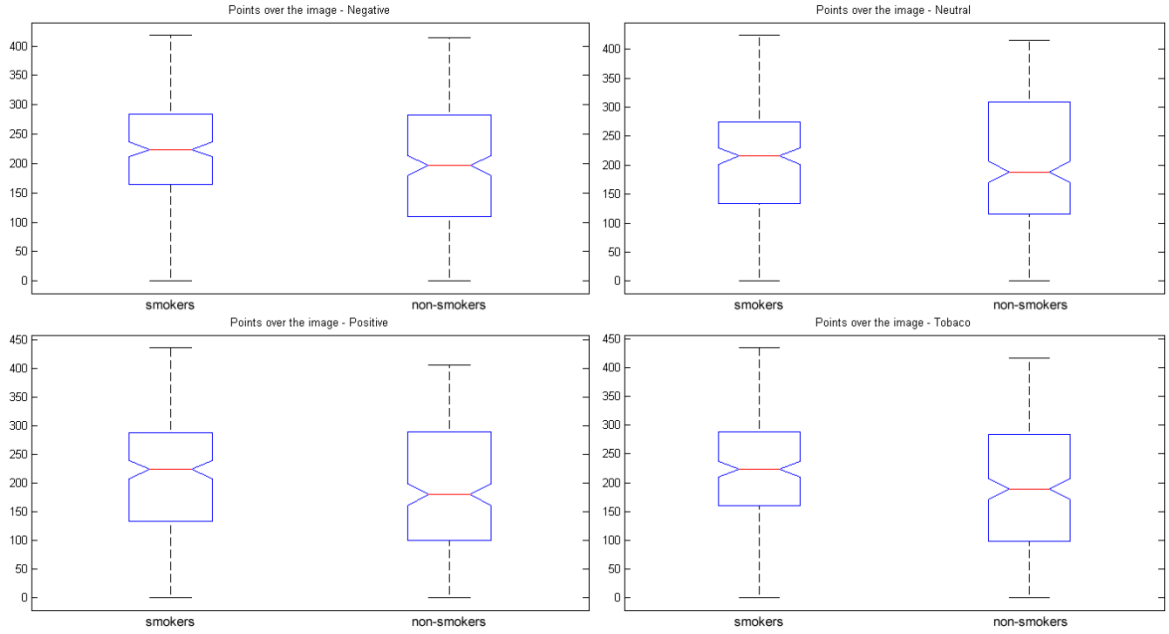

Figure 36 – Slightly higher number of points over the images in the smoker group for all the image groups.

An interesting case was observed by an ex-smoker participant, which had a higher number of fixations over the images of the tobacco pack images than any other groups. On average, three fixations per negative image, four fixations per positive or neutral image and six fixations per tobacco pack image. After observing all the heatmaps related to this participant tobacco package images it was clear that the participant had a tendency to look at the lower half of the image; generating the heatmap in the *Review* Environment, combining all the tobacco package images data this tendency could be confirmed. The participant was deliberately avoiding the images per se. At a first look, he might have considered them as accustomed too (besides the fact that they were in fact uncirculated images) and he spent the time actually reading the text messages. It is also noteworthy, that this participant was the only ex-smoker that scored a moderate dependency (out of the total of five, from the twenty-seven questionnaires).

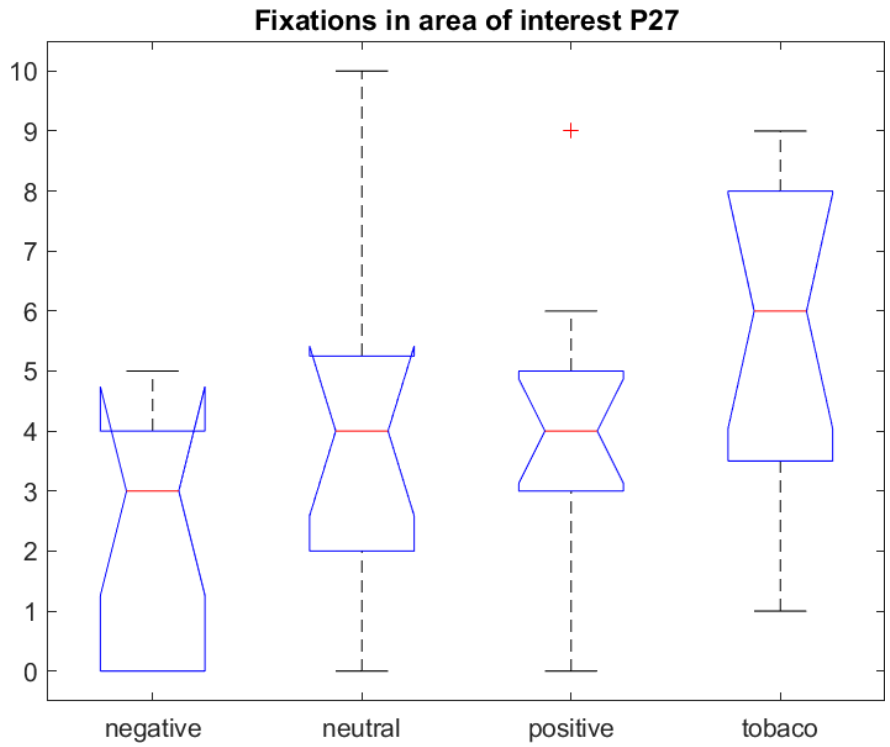

Figure 37 – Participant twenty-seven fixations over the AOI in the different groups of images (exact area of the image).

Figure 38 – Image on background is for visual purposes only, represented data is from all the tobacco package image group from the participant twenty-seven.

After this initial analysis through boxplots a cluster analysis was performed for the eye tracking fixation data using:

1. Type of image, separating tobacco package images from the other groups;
2. Type of participant, dividing the participants in smoker group and others group, evaluating only the tobacco package images.

From this analysis (k-means with two clusters), using the different combinations of variables, we verified a hit percentage of approximately forty-eight, meaning we could not observe a clear separation between the groups, as the partitions are really close to a coin-flip.

In conclusion, although it is possible to observe a clear distinction between image groups from the arousal and valence self-evaluations, when the physiology is considered, that distinction is not observable apart from punctual cases.

At the start of the experiment, we presented to the participants the equipment, and told them that their eyes would be tracked, something that they could assume by the process of the eyes calibrations. Anyways, we consider that the fact of explaining the equipment might add a negative impact on the experiments results, since the participants possibly have felt the necessity of looking at the screen at all times, and this way the question of image avoidance could not be clearly differentiated.

7 Conclusions and discussion

In this dissertation, we proposed to build Eyempact, a portable and affordable system for eye tracking and bioelectric physiology monitoring purposes, all in all this objective was met. Eyempact allows one to easily record their gaze and other physiology data, do a posterior review and export measurements and metrics for analytical purposes.

Eyempact was built with the intent of being accessible to everyone (with or without computer science/programming background), and to a certain degree, the presented GUI's accomplishes this objective. There was an attempt on freezing[25] the applications, using multiple well-known Python freeze programs like *pyInstaller*[26] and *cx_Freeze*[27] (between others); several problems were encountered, as it is to be expected with the use of multiple packages, some probably not in a stable version. Although we do not have out-of-the-box executables to run our system, Eyempact is accessible for anyone with basic computer skills, the installation process is quite simple and available in appendix A.

The system was deployed and utilized in our case study, proving its robustness and usability. In this study, were involved directly two dissertation students besides me, one from psychology and other from Computers and Telematics Engineering, both found it straightforward to use the *Control Panel*. Also, in the study pilot were involved two researchers, which provided informal feedback, which was very rewarding. Overall, we have over thirteen hours of continuous acquisition, with data from the forty-seven participants of the study, and several more hours from testing and the study pilot, realized during multiple sessions.

The Tobii eye trackers, revealed at times problems with accuracy, requiring calibration when dealing with different kinds of people (such as taller or shorter, with glasses, with contact lenses), what may even happen with professional eye trackers. Despite these problems, revealed mainly during the live presentations of "Where's Wally" game, and despite the Tobii eye trackers used not being dedicated for research purposes, they served well for our intentions of proof of concept and to our case study.

[25] ***In this context, the meaning of freezing is to create application executables and/or installers.***

[26] ***https://www.pyinstaller.org** - Official site.*

[27] ***https://anthony-tuininga.github.io/cx_Freeze** - Official site.*

There is a lack of open source SW solutions to support eye tracking visualizations and metrics. The few existing examples consist of simple scripts and code snippets that are, mostly, meant to be integrated in bigger projects with wider scopes, but not as an out-of-the-box solution to support eye tracking analysis per se. For this, to the best of knowledge, we did not find any SW for addressing the typical researcher concerns – acquire, quantify and represent eye tracking for later analysis. Eyempact provides the basics for eye tracking investigations, allowing researchers with lower budgets to have at hand a simple visualization tool alongside some of the most used metrics and measurements used in the area. Obviously, Eyempact cannot compete with the SW capabilities of big technologic players like Tobii (with almost two decades of existence) or research solutions like iMotions (also in the works for more than a decade), however we do think that our solution is unique in its own way and can be useful for basic research objectives.

7.1 Future Work

The SW is open to improvements or inclusion of new features. Suggestions for the maintenance and evolution of the current solution are presented below, accompanied by their purpose and a small argument on why they should be implemented, and which are more relevant. The importance of implementation was based on the following topics:

1. **Scalability of the system:** does the change allow more people to use the system?
2. **Importance for research:** does the change increase the value of the system for researchers?
3. **Usability improvement:** does the change significantly improve the system usability?
4. **Difficulty of implementation:** how difficult is the change to implement either in logistic or development terms?

Starting with the acquisition part of the system, adding system support for more eye trackers and other physiological sensors and cameras could be very beneficial. This could be achieved by implementing new acquisition programs (or updating the current one, adding support for more Tobii eye tracker) for other HW devices, and updating the control panel interface to support them. Although this feature allows a great scalability of the system's usage, and thus it is of great importance for researchers (reaching a higher population), it is logistically hard to implement, before anything new equipment needs to be acquired. As far as eye trackers go, we only have available to us the aforementioned Tobii eye trackers. Physiological sensor support for VJ and Mi Band could be added. Also, considering the system's "affordability tag", as far as we know eye trackers as affordable as Tobii's do not exist.

In terms of new features for the *Review Environment* the possibilities are almost endless, only a few that seemed more relevant are presented here. Firstly, we would implement the possibility of a combined visualization of multiple participant data sources, where the gaze or fixations of two or more participants would be shown at the same time (with different colour schemes). This type of visualization would allow a direct comparison of multiple participants, and maybe a preliminary group classification in a study, which could be further investigated afterwards, confirming or updating the group formations with the data analysis. Taking into consideration the current state of the solution, this would not be very difficult to implement.

Another great extension to the current set of visualizations would be the combined visualization of different data sources. For example, representing gaze and HR synchronized. This would allow researchers to observe how multiple data sources change over time (in a single visualization). In terms of development, this has considerably higher difficulty than the previous point. Firstly, it needs to be decided, which combination of data sources would be allowed. Secondly, how they are going to be drawn, in the same widget/canvas or not. And after this, implement it in a way that does not compromise performance.

To conclude the *Review Environment* features: addition of 3D eyes position replays and head tracking visualizations and metrics. Mainly, it would be important (and straightforward) to implement metrics like the distance of the participant to the screen along time, as well as measure quantity of movement. These could also be presented as graphs.

Regarding the improvements to already existent functionalities it would be imperative to:

1. Add terminal-based operations to generate images, video and data exportation**.** Eyempact's GUI is useful to quickly check individual participants data, although for big experiments, with a large number of participants, the generation of all the images/videos and data exports can be quite time consuming; allowing all these kind of operations through the terminal would be very useful for the automation of this kind of work;
2. Use multithreading for better performance on computationally intensive task, diminishing the time the user must wait for outputs. Currently, Eyempact's approach to computationally intensive tasks is to perform them in their own thread (in order to avoid freezing the GUI). So, the program is not taking the full potential of the machine. This could be improved with the use of multi-processing; this was explored in some cases.
3. Improve system memory management and add related settings. Now, Eyempact only clears from memory the data dedicated to a tab, when that tab is closed. Eyempact does not impose any memory limit, so it is possible to open as many data files as the user wants, until the system memory is completely filled up. It should be implemented a way, a setting maybe, to at least limit Eyempact memory usage to a percentage of the system available memory.

References

[1] U. Rashid, M. A. Nacenta, and A. Quigley, “Factors influencing visual attention switch in multi-display user interfaces,” in *Proceedings of the 2012 International Symposium on Pervasive Displays - PerDis ’12*, 2012, pp. 1–6.

[2] Microsoft, “Attention spans,” *Consum. insights*, 2015.

[3] N.-H. Liu, C.-Y. Chiang, and H.-C. Chu, “Recognizing the Degree of Human Attention Using EEG Signals from Mobile Sensors,” *Sensors*, vol. 13, no. 8, pp. 10273–10286, Aug. 2013.

[4] A. Kamišalić, I. Fister, M. Turkanović, and S. Karakatič, “Sensors and Functionalities of Non-Invasive Wrist-Wearable Devices: A Review,” *Sensors*, vol. 18, no. 6, p. 1714, May 2018.

[5] D. C. Richardson and M. J. Spivey, “Eye-Tracking : Characteristics and Methods Eye-Tracking,” *Biomedical Engineering*, vol. 2, no. 932839148. pp. 1–32, 2004.

[6] R. J. K. Jacob and K. S. Karn, “Eye Tracking in Human-Computer Interaction and Usability Research,” in *The Mind’s Eye*, Elsevier, 2003, pp. 573–605.

[7] M. Borys and M. Plechawska-Wójcik, “Eye-tracking metrics in perception and visual attention research,” 2017, pp. 11-23.

[8] Z. Sharafi, T. Shaffer, B. Sharif, and Y.-G. Gueheneuc, “Eye-Tracking Metrics in Software Engineering,” in *2015 Asia-Pacific Software Engineering Conference (APSEC)*, 2015, pp. 96–103.

[9] M. L. Mele and S. Federici, “Gaze and eye-tracking solutions for psychological research,” *Cogn. Process.*, vol. 13, no. S1, pp. 261–265, Aug. 2012.

[10] M. Wedel and R. Pieters, “A Review of Eye-Tracking Research in Marketing,” in *Review of Marketing Research*, 2008, pp. 123–147.

[11] R. Rebollar, I. Lidón, J. Martín, and M. Puebla, “The identification of viewing patterns of chocolate snack packages using eye-tracking techniques,” *Food Qual. Prefer.*, vol. 39, pp. 251–258, Jan. 2015.

[12] E. M. Kok and H. Jarodzka, “Before your very eyes: the value and limitations of eye tracking in medical education,” *Med. Educ.*, vol. 51, no. 1, pp. 114–122, Jan. 2017.

[13] P. Isokoski, M. Joos, O. Spakov, and B. Martin, “Gaze controlled games,” *Univers. Access Inf. Soc.*, vol. 8, no. 4, pp. 323–337, Nov. 2009.

[14] J. Zhang and J. Kalinowski, “Culture and listeners’ gaze responses to stuttering,” *Int. J. Lang. Commun. Disord.*, vol. 47, no. 4, pp. 388–397, Jul. 2012.

[15] R. R. Henderson, M. M. Bradley, and P. J. Lang, “Emotional imagery and pupil diameter,” *Psychophysiology*, vol. 55, no. 6, p. e13050, Jun. 2018.

[16] M. J. Wieser, P. Pauli, G. W. Alpers, and A. Mühlberger, “Is eye to eye contact really threatening and avoided in social anxiety?—An eye-tracking and psychophysiology study,” *J. Anxiety Disord.*, vol. 23, no. 1, pp. 93–103, Jan. 2009.

[17] M. M. Bradley and P. J. Lang, “Measuring emotion: The self-assessment manikin and the semantic differential,” *J. Behav. Ther. Exp. Psychiatry*, vol.25, no. 1, pp. 49-59, 1994.

[18] L. Ferreira, C. Quintal, I. Lopes, and N. Taveira, “Teste de dependência à nicotina: validação linguística e psicométrica do teste de Fagerström,” *Rev. Port. Saúde Pública*, 2009.

[19] N. Watanabe and M. Yamamoto, “Neural mechanisms of social dominance,” *Front. Neurosci.*, vol. 9, Jun. 2015.

[20] L.-C. Yu *et al.*, “Building Chinese Affective Resources in Valence-Arousal Dimensions,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 540–545.

Appendix A – Eyempact

Installation process

1. Install supported Python version 3.6.5 - 3.6.8 (versions 3.7.X should also work for the core functionalities, although some of the packages used are still not up to date for Python 3.7); if this is the first time installing python, you just need to visit *https://www.python.org/* and download the executable, follow the steps as any other windows installation process;
2. Open windows console terminal; search “terminal” in the Windows menu and open it;
3. For each of the programs (*Control Panel* and *Review Environment*), follow the steps:
    a. Navigate on the terminal to the directory of the program using the command “*cd <path>*”, where path should be replaced by the respective directory path;
    b. Run “*pip install -r requirements.txt*” to install all the dependencies;
4. Now, Eyempact programs are ready to use. Just run the command:
    a. *python ControlPanel.py* to run the *Control Panel* program;
    b. *python ReviewEnvironment.py* to run the *Review Environment* program.

Configuration file notes

ECF – *Eyes Data Gatherer* Configuration File

BCF – *Bitalino Data Gatherer* Configuration File

Some important notes regarding the manual editing of the configuration files:

- All configurations must be edited ahead of the ">" character;
- Configurations 3-7 and 12-14 (in ECF) and 6-12 (in BCF) can only assume 2 values, the characters "0" and "1", meaning the option is enabled or disabled;
- In ECF, configuration 2 must be a valid path in your system;
- In BCF, configuration 4 must be a valid MAC address; configuration 5, can only assume “1”, “10”, “100” and “1000” values;
- All the other configurations are "string" based and must only include valid characters for a regular Windows file name (invalid characters will be removed).

Review environment screenshots

In the following screenshot it is possible to view in detail the *Settings* and the *Metrics* panels present in the *Review Environment*. These panels have available various sections, that we call drawers, which the user can open and close at will to show the most important operations at each moment.

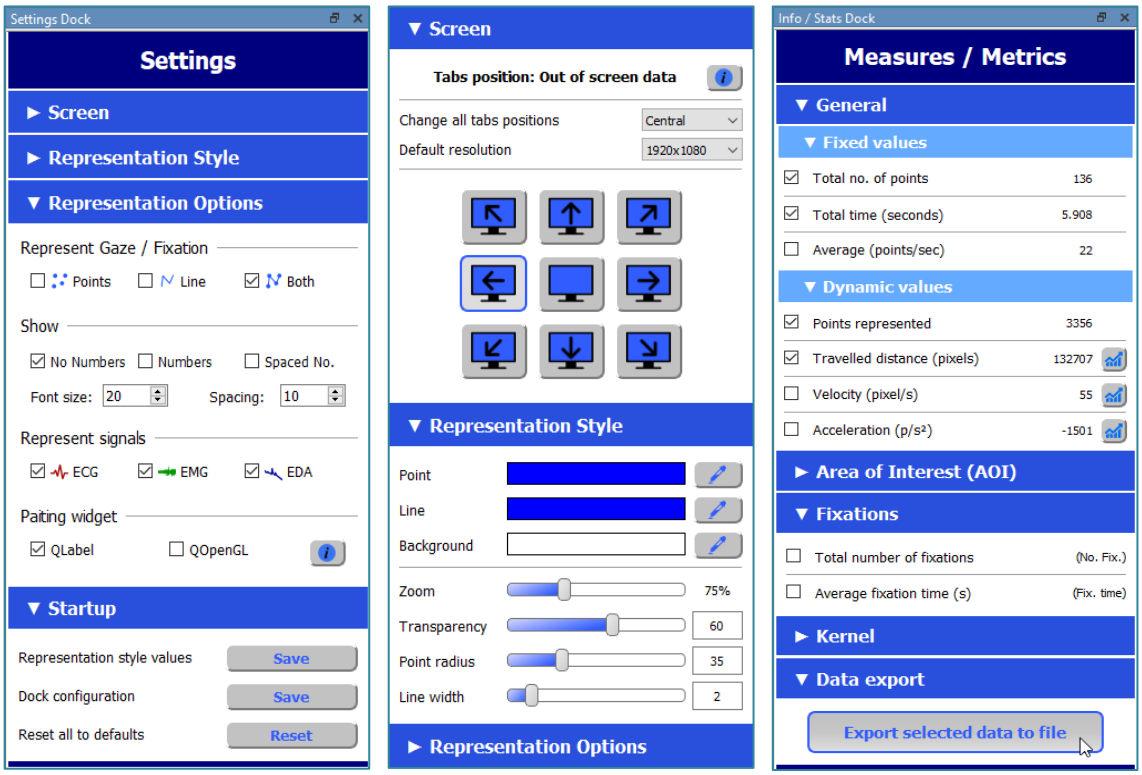

Figure 39 – Settings panel (left and center image) and measurements / metrics panel (right).

On the left we can see the *Representation Options* and the *Startup* drawers open. On the first, the user chooses what to represent in the visualization. In the second, the user can choose to save the settings on the *Representation Options* and *Representation Style* (open in the center image) for future program uses or reset the settings to their defaults.

On the right, the user can observe and select measures and metrics to export. The user simply "ticks" the relevant checkboxes and "clicks" on the export button to save a file with the selected data. Some of the metrics can be plotted, as we can see in this image "traveled distance", "velocity" and "acceleration" both have a button to their right for that effect.

Appendix B – Health warnings in tobacco packages study

Setup material list

Complete material list, gatherer for each session:

- Eye tracker - Tobii 4C
- BITalino board and 8 electrodes per participant
- *GoPro*+ and tripod
- Thermal camera + Raspberry combo + Power cable (Micro B)
- Secondary monitor (1080p 23")
- Laptop and respective power cable + HDMI cable + Cabled mouse and Keyboard
- Lighter (to serve as Thermal camera sync point)
- Paper towels (to clean electrode gel after removing)

Setup preparation

For each session of experiments the setup preparation followed the following steps:

1. Connect laptop to secondary monitor and extend desktop via HDMI;
2. Place eye tracker on the secondary monitor and connect it to the laptop via USB;
3. Set up eye tracker to the secondary monitor;
4. Turn on *GoPro* wi-fi in "*GoPro* App" mode;
5. Connect laptop to *GoPro* wi-fi network;
6. Connect all the cables to BITalino, turn on BITalino and connect it to the laptop via BT;
7. Run "Control panel" application and verify all the configurations.
8. Setup thermal camera.

Experimental Protocol

1. Run Unity application;
2. Ask the participant to seat comfortably in front of the secondary monitor and handout the informed consent;

3. Ask participant to remove watch and hand out any electronic devices before the experiment (electronic devices were proved to deteriorate the physiological signals quality in the pilot experiment);
4. Calibrate eye tracker to participant's eyes;
5. Start/Play Unity application;
6. Ask participant to fill in participant information form and read information on the experiment;
7. Step aside the keyboard since it will not be needed anymore. Only mouse will be needed for the image evaluation;
8. Clarify and answer any question from the participant;
9. Place all the electrodes in the participant arms;
10. Press "Start all" button in "Control panel";
11. Check if data is being collected (Eye tracking and BITalino files) and if *GoPro* is recording;
12. At this point the participant can click on the "Next" button to start visualizing the images;
13. When participant finishes, press "Stop all" button;
14. Turn off BITalino and remove the electrodes from the participant;
15. Prepare setup for next participant.

Electrode disposition

Description and reasoning behind the electrode placement shown in:

- In general, all the electrode positions were chosen in order to be less intrusive as possible;
- ECG positive electrode was placed on the right forearm, negative and ground electrodes on the right forearm (close to the hand) – we decided against e.g. a chest placement mainly for the reason stated above;
- EMG upper forearm housed the positive and negative electrodes, the ground was placed in the elbow – decision against other placements was due to two reasons, the lead size was very short greatly limiting our options, and the electrodes available to us were to big thus making it impractical for a facial placement for example;
- EDA electrodes in the palm of the hand – this is a good location for capturing EDA; although the two-lead cord was even smaller (also limiting our options).

Unity Application Print Screens

The following screenshots show the complete workflow (in a sequential way) of the Unity application used in the presented case study.

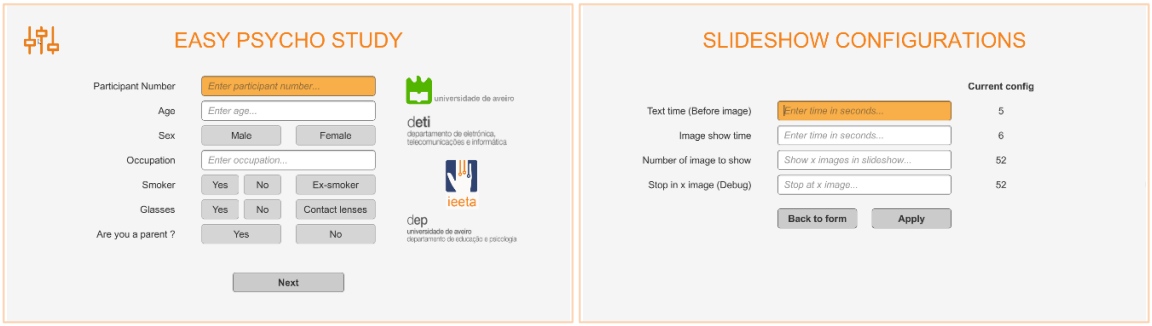

Figure 40 – First screen shown to the user (left) has a hidden button on the upper left corner to configure the experiment parameters (right).

Figure 41 – Initial formulary presented to the participant.

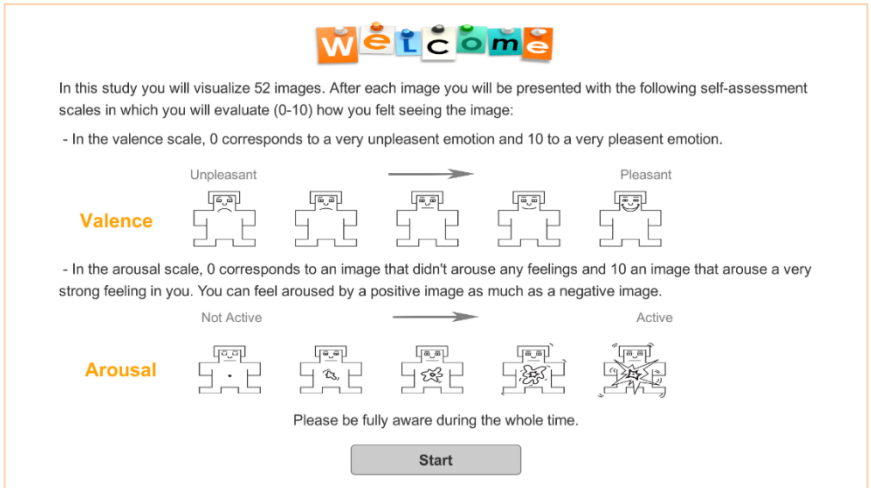

Figure 42 – Simple explanation of on the image classification scale (SAM). Further oral explanations were given in the section.

Figure 43 – Text presented for five seconds to gather a baseline (left). Example of image placement (right).

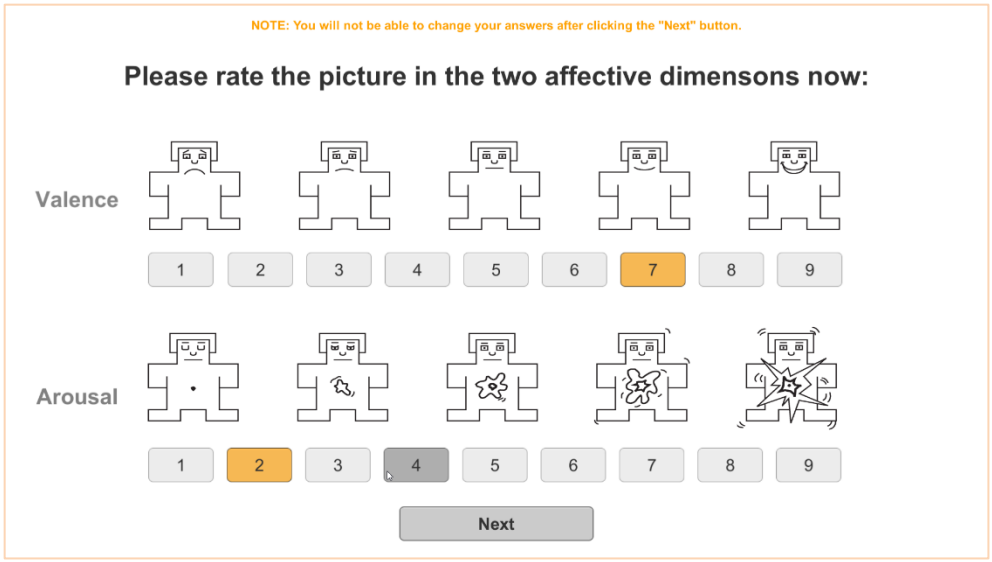

Figure 44 – SAM scale screen.

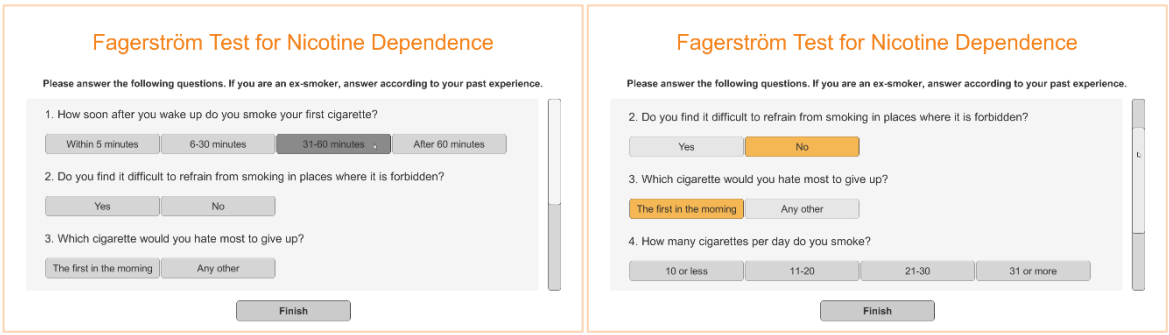

Figure 45 – Fagerström dependency test shown to smoker and ex-smoker participants.

Statistics for sociodemographic attributes

CHARACTERISTICS	FREQUENCY	PERCENTAGE (N=47)
AGE (Mean ± SD)	≈ 23.8 ± 4.3	
≤ 21	15	≈ 31.0
22-25	22	≈ 46.8
≥ 26	10	≈ 21.3
GENDER		
Female	26	≈ 55.3
Male	21	≈ 44.7
OCCUPATION		
Student	39	≈ 83.0
Researcher	2	≈ 4.3
Other	6	≈ 12.8
CORRECTED VISION		
Glasses	19	≈ 40.4
Contact lenses	4	≈ 8.5
No correction	24	≈50.1
PARENTS		
Number of parents	5	≈ 10.6
GROUP		
Smokers	20	≈ 42.6
Non-smokers	22	≈ 46.8
Ex-smokers	5	≈ 10.6
TOBACCO DEPENDENCE		**N = 22+5**
Smokers low dependency	10	≈ 37.0
Smokers low-moderate dependency	5	≈ 18.5
Smokers moderate dependency	3	≈ 11.1
Ex-smokers low dependency	3	≈ 11.1
Ex-smokers low-moderate depend.	1	≈ 3.7
Ex-smokers moderate dependency	1	≈ 3.7
Smokers / Ex-smokers high depend.	0	= 0
Missing questionnaire answers	4	≈ 14.8

Table 12 –Table comprising all the sociodemographic data gathered.

Case study by the numbers

- Original data[28] size: **84.9 GB**;
- **1.44 GB** Text-based dataset[29];
- ≈ **5 GB** of text-based data all together[30];
- **2000+** images generated; heatmaps, boxplots and others;
- ≈ **13 hours** of continuous data acquisition;
- **540K+** Gaze points**.**

Eye tracking complete data file format

Headers order

id, cat_id, img, pos_img, cat_img, seg_apr, local_ts, <gaze_cols> , <fixation_cols>

id participant identifier

cat_id participant category identifier

0 = non-smoker

1 = smoker

2 = ex-smoker

img image identifier

pos_img image display order position

cat_img image category

0 = negative

1 = neutral

2 = positive

3 = tobacco

seg_apr millisecond in which image was presented

local_ts local timestamp

<gaze_cols> columns referent to gaze data

no_gpts	number of gaze points captured
g_time	gaze file capture time
g_dist	travelled distance by gaze (in pixels)
img_pts	number of gaze points over the image

[28] ***This includes the GoPro videos.***

[29] ***All original eye tracking and physiological data files.***

[30] ***Original data + Re-organized data + Segmented data + Eyempact exported data.***

<fixat_cols> columns referent to fixation data

no_fpts	number of fixation points captured
f_time	fixations file capture time
img_pts	number of fixations points over the image
f_aoi	number of fixation over the image (over AOI)
f_oaoi	number of fixations out of the image (out of AOI)
f_paoi	number of fixations partially in and out of the image (partial AOI hits)

Bioelectrical signals complete data file format

Headers order:

id, img, cat_img, section, seg_apr, local_ts, samp_id, emg, ecg, eda, time

id participant identifier

img image identifier

cat_img image category

0 = negative

1 = neutral

2 = positive

3 = tobacco

section "image display" was divided in three sections

-1 = period of time not relevant for analysis

1 = sample in the baseline period (five seconds before image display)

2 = sample in the period of display (six seconds of image display)

3 = sample in the SAM image evaluation (undefined period of time)

ms_apr elapsed time from the start of image display in milliseconds

local_ts timestamp

samp_id sample id sent by BITalino board (for missing sample checksum purposes)

emg EMG sample

ecg ECG sample

eda EDA sample

elapsed_time elapsed time from the acquisitin start in hh:mm:ss:fff format

ECG_Filtered, ECG_R_Peaks, Heart_Rate, ECG_Signal_Quality, ECG_RR_Interval, ECG_HRV_HF, ECG_HRV_LF, ECG_HRV_ULF, ECG_HRV_VHF, ECG_HRV_VLF, EMG_Pulse_Onsets, EMG_Filtered, EMG_Envelope, EMG_Activation, EDA_Filtered, EDA_Phasic, EDA_Tonic, SCR_Onsets, SCR_Peaks >> extracted from neurokit, please consult their documentation for details

Appendix C – Public events

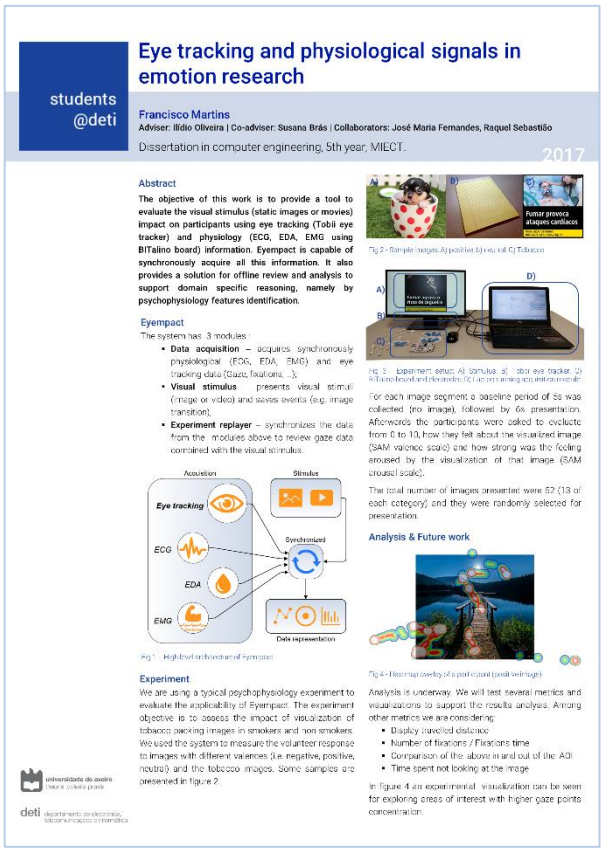

Figure 46 – Poster displayed at Students@DETI event.

Figure 47 – *Where's Wally* live demo at Xperimenta event.