2014

Design of a drone's control with a neuronal headset

SEARCHING THE FUTURE TECHNOLOGY

ENRIQUE FERNÁNDEZ SERRA

UNIVERSITAT POLITECNICA DE CATALUNYA | C. Colom, 1 08222 Terrassa

Abstract

The project is about combining new engineering technologies with healthy investigation, using Brain Computer Interface (BCI) readings from a project external device. As a pilot project, a drone is going to be controlled by mind with a neuronal headset (*Emotiv EPOC+*) and a personal computer (PC).

The main goal is the investigation and development in neuroscience, building and walking through a bridge between engineering and medics' technology, pulling forward the biomedical's future and helping changing the world, improving the actual technology and creating a media impact.

Abbreviations

BCI: Brain Computer Interface

PC: Personal Computer

EEG: Electroencephalogram

UAV: Unmanned Aerial Vehicle

RPA: Remotely Piloted Aircraft

SDK: Software Development Kit

UI: User Interface

Gratefulness

This Project could not been done without help from different people. The help of all this people is appreciated.

First of all, thanks to the tutor, Ramón Pérez Magrané, who has fought for doing a good job and for the project interests. Also Bernardo Morcego Seix for letting the drone's battery charger.

Thanks to Richard Hebert for his support and for the opportunity provided when testing and using all the material, from the drone until the neuronal headset.

To Daniel González del río for lending the Android device.

Thanks to my family and Cristina Paola Santini Díaz for plenty support and personal motivation, for cheer me when there seemed no way out.

To my English teacher, Cristina Cano, who has helped with grammar corrections at the end of the project.

Thanks to Vita Entrepreneurship Centre, the centre where has been developed this and other projects in order to create new companies.

Finally, thanks to everybody who has contributed in one way or another in the development of this project.

Content

Summary

Goals and motivations

Investigation and technology development in a healthy context is the bridge between medicals and engineering. The main goal's project is to contribute that bridge building and trying to create a media impact and generate more investigation in neuronal studying.

The project began with the idea that, in a future, the computing processes are going to be carried out by user's brain and the information is going to be sent directly to the brain. So investigation in neuroscience is a step for the technology's future.

Neuroscience

To achieve that goal, diverse neurosensors headset are going to be compared and tested, so it is going to choose the right one. Also, a basic neuroscience learning has been acquired to read and understand brainwaves.

Starting point

This project tries to build a part of the medical-engineering bridge writing an application that uses neurosensors to control a drone's movement, using a computer (PC) to connect both headset and drone.

Programming

The project program has been done using Unity 3D, which is a graphic motor. The application has been written in C# and the investigation code has been written with Matlab.

The program connects the headset using the USB dongle provided and drone using the own drone's generated Wi-Fi.

Testing

The more axes are added, the more difficult is the application to control. So, even if the drone's movement is controlled by the user's brainwaves, not all the axes can be controlled. For that reason, the application uses only keyboard to control take off and land, despite that all the commands are able to be controlled by the keyboard.

Conclusions

On the one hand, the technology used has a long way to walk before it could be used in a real engineering project because of the precision the headsets have. On the other hand, the technology can be a chance for medical uses and investigation.

Problem description

Nowadays, there are people with healthy problems that are not able to use some parts of their bodies, as tetraplegia or some types of paraplegia. The project tries to be a pilot project orientated to more wide uses of Brain Computer Interface (BCI) technology, having a development of brain control.

Daily life common actions and activities could be a real problem for people with any mobile disability: shopping, walking or even taking a bath, the smallest movement can suppose difficulty and pain both physical and mental.

Moving around with a wheelchair is complicated for people that cannot move rightly the hands, and can be a bigger problem if the person cannot move any end part of the body. That kind of people needs third parties constant attention, provided from sanitary professionals or familiars.

The paralytic's and that third parties' life levels are another direct problem from disabilities, which are affected by a daily economic, time and social costs.

FIGURE 1: BRAIN CONTROLLED WHEELCHAIR

Extracted from: http://www.instructables.com/id/Brain-Controlled-Wheelchair/

Goals and motivations

This Project is about the study, investigation and development of the neurosensors in a technologic context. The pilot idea is going to be the control of a drone by these neurosensors, which read the thoughts, firstly in a plane and then, if it is viable, in a free 3D space. To delimit the drone movement, Bluetooth or Wi-Fi beacons can be included in the model to get a position indoor or outdoor.

Basically, the prototype will be made-up by two components. Firstly, the neurosensors headset, that will be the new or older model of the *Emotiv Company*. Lastly, the drone we are going to control, which is going to be an *AR Drone*.

As said before, the pilot idea is to change the actual program of the drone's control for Android to another which will get the input signal from the headset, choosing the right way of programming and the right devices. Also the headset will need to be calibrated to recognise the different kind of thoughts because of the variance between the user's brains.

The study of brain injuries and the investigation to help the medics' development comes in the wake of a close family experience, which boosted and drive the biomedical interesting, wondering what the next step is and trying to get there first.

Finally, the project began with the intention to support the idea of using the brain to process digital information and receive directly the outcomes, allowing the user to see and listen without using the eyes and ears, so deaf and blind people would be able to recover their lost senses (if it is not a brain injury). So this project has born to be the first step for the technology's future.

Economy and time estimation costs

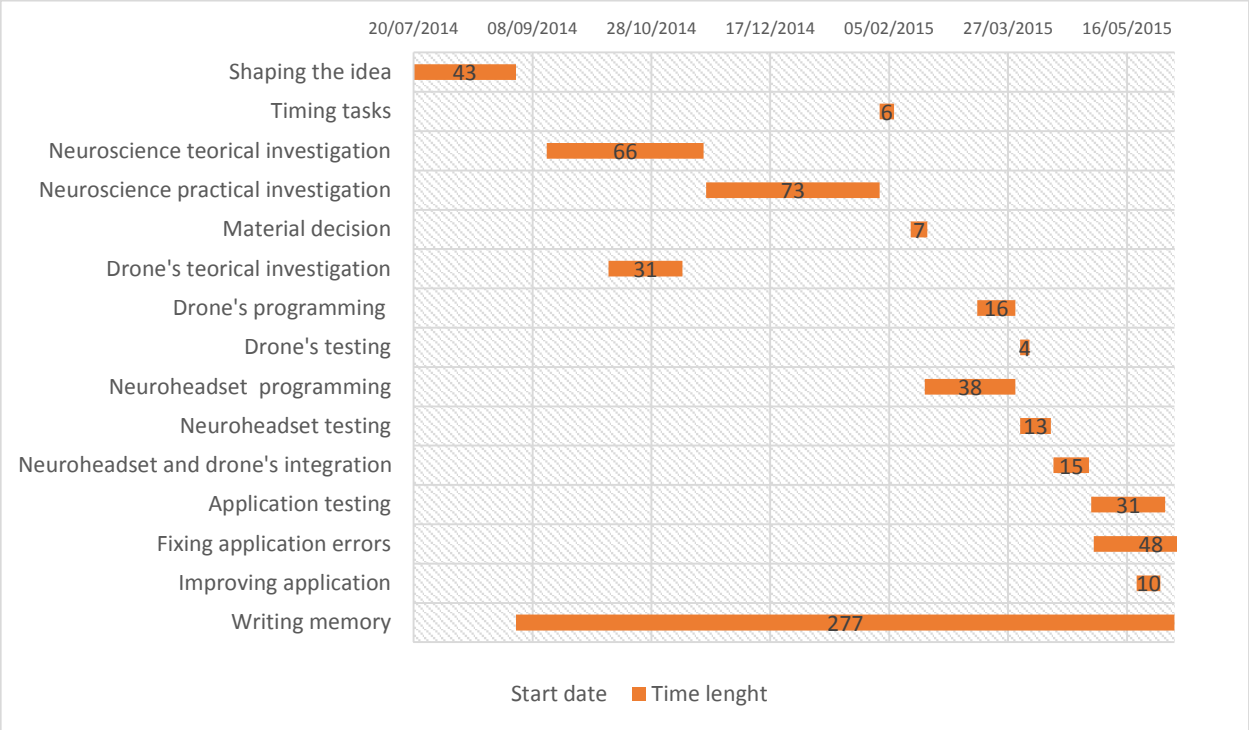


TABLE 1: GANTT TABLE

Budget estimatation approach	
Title	**Cost ($)**
Working time	40680
Neuronal headset	1000
Drone	350
Laptop	750
Total	**42780**

Working time estimation		
Time (hours)	**Cost ($)**	**Total**
2712	15	40680

TABLE 2: BUDGET ESTIMATION TABLE

An estimation in time and money has been done to get an approach of a fully project costs.

First, time is estimated from prioritize investigation and testing, because of the hard work that supposes learning how to use the devices and testing them to get an idea of which is going to be the best choice in this project.

Costs include from the team working time to material used. As it is no clear which is going to be the one chosen, it has been done a big estimation. The working time has been calculated from working days, supposing a 4 hours working per day without weekends. This is, of course, not realist at all in the working time supposed, but is an approach from working more hours one day and less another.

Neuroscience

Since the dawn of science and the beginning of medicine, humans have been looking for satisfy their curiosity and improve knowledge about ourselves. The most complicated barrier is the brain, the core of humans and all intelligent species, the communication centre of bodies and the most unexplored and mysterious part.

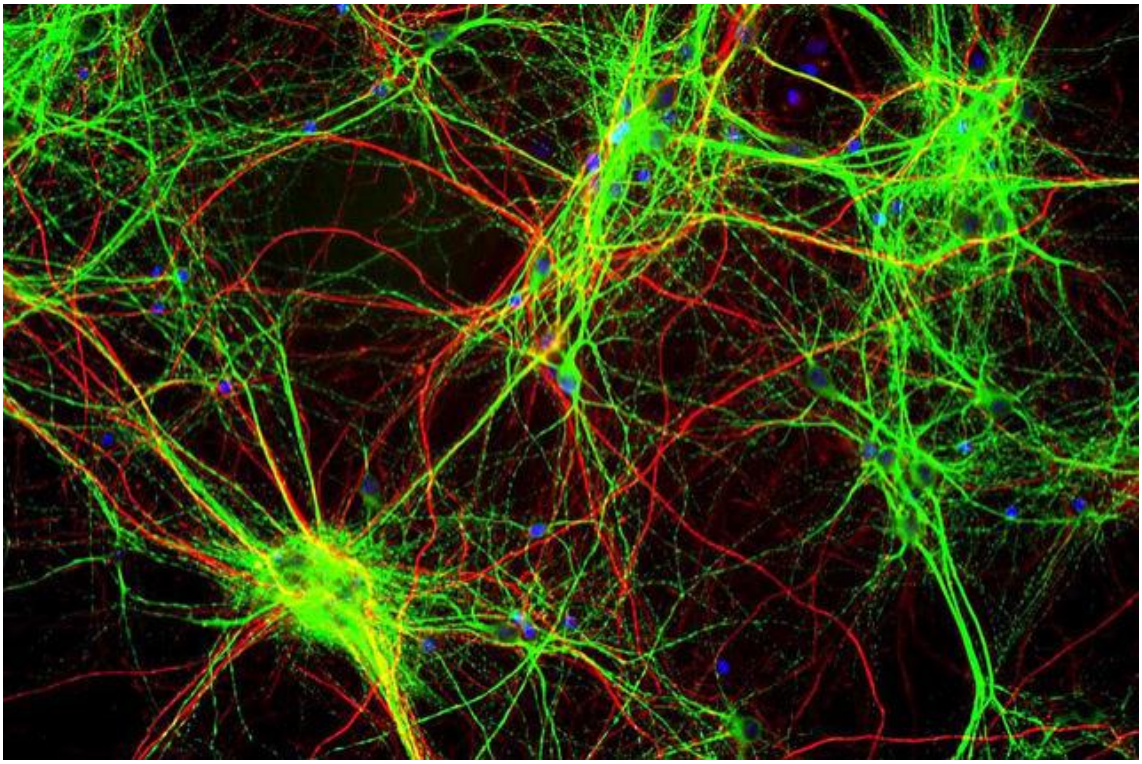


FIGURE 2: NEURONES

Technology nowadays

Neuronal headsets are able to read an electroencephalography (EEG) and send it to another device. An EEG is a brain activity register and can provide information to be interpreted as emotions, feelings, concentration and relaxation states and more.

The most innovative element in this project is the neuronal headset. Choosing the right device for the project has been complicated because of the different devices that exists nowadays. At the beginning of the project, there were four headsets to compare and select: *Emotiv EPOC+*, *Emotiv Insight*, *NeuroSky* and *Open BCI*.

NeuroSky

Having a different way to think, *NeuroSky* Company tries to reach a widest consumer's range selling a cheaper headset. The limitations are bigger than the previous ones, of course, as this device is not ready to get a trusty user's EEG, with only one channel from the frontal's to the ear's sensor.

FIGURE 3: *NEUROSKY* HEADSET

Checking the device's efficiency some tests have been done.

The first tests were with the tiny drone provided with it and its smartphone's application. The results were not satisfactory as the application broke few seconds after the application run, it was thought that the external connector was broken and the company sent a new one, but nothing changed with it.

The second tests were done with *NeuroSky* EEG reading application called *BrainWave Visualizer*. The applications shows the user's brain waves in a beautiful graphics and meditation and concentration levels in two indicators from 0 to 100. Even if the application was very well designed, the results were also not satisfactory because of the trusty of the results, the waves read were very changeable and unstable to use in this project.

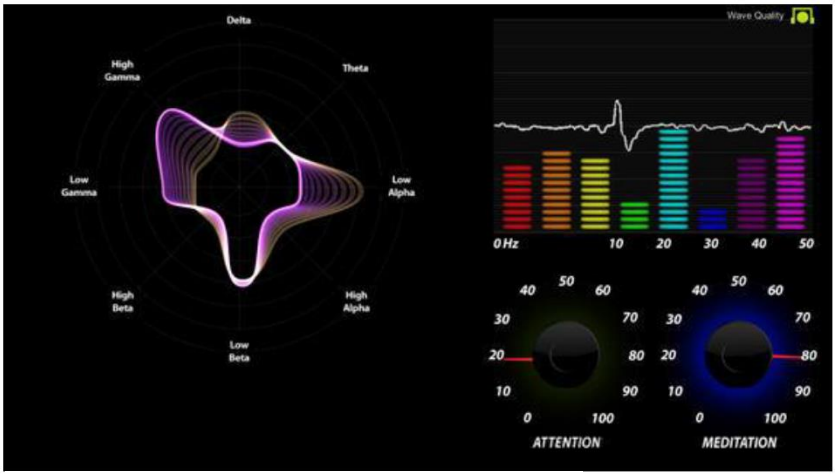


FIGURE 4: BRAINWAVE VISUALIZER

Open BCI

Another brain-computer interface device that it is going to be used in the project for studying brain reactions is a new device from kick-starter which can read and write neuronal micro voltages and the body's device is full 3D printable, so the customer can choose the positions for reading and/or writing. It is available with 6 or 12 connectors but, as it is a new device, it has to be tried and made some probes before using it as the main body of the project.

FIGURE 5: *OPEN BCI* ELECTRODES

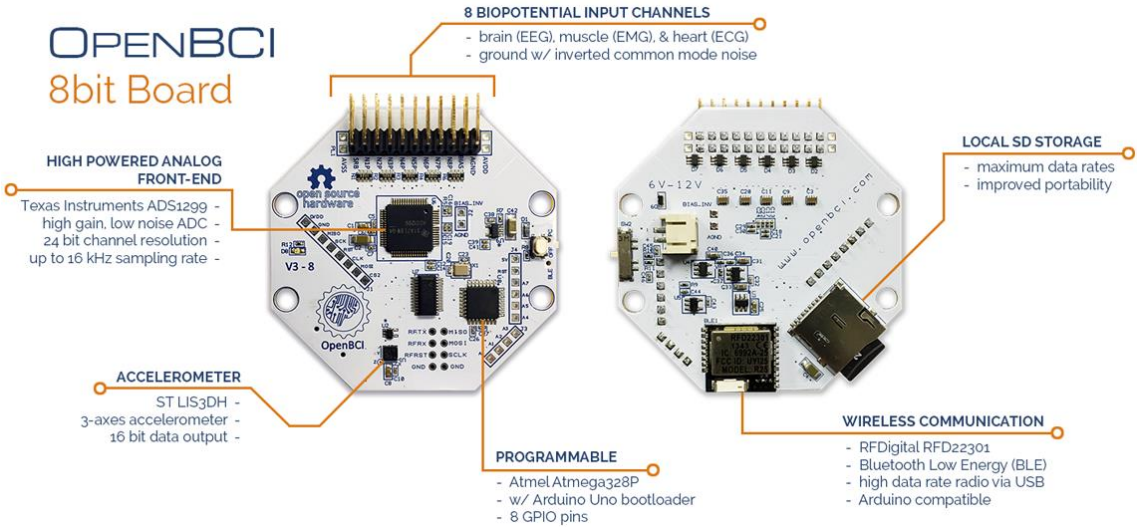


FIGURE 6: *OPEN BCI*

Emotiv neuroheadset.

It is called neuroheadset the instrumentation with neurosensors, which are used for the reading of the *Electroencephalography* (EEG), the brain made waves. The *EPOC Emotiv*, can bring us the differential voltage information between eighteen points and fourteen channels available, divided by frequency, so it is easy to understand and manage it.

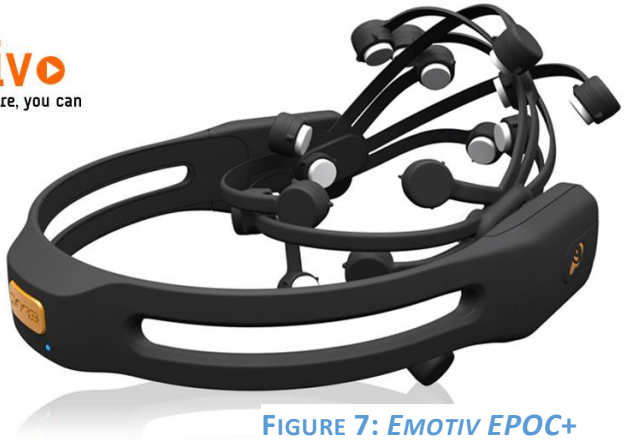


FIGURE 7: *EMOTIV EPOC+*

The sensors of the actual *EPOC* headset are a wet type, which has to be soaked every few time, at least every time you use it, so could be useful for a first prototype but no for a real daily use. On the other hand, the company is developing a new one with dry sensors, which means that we do not have to take as care as much as the other sensors, but the final product could not be on time for the project.

FIGURE 8: *EMOTIV INSIGHT*

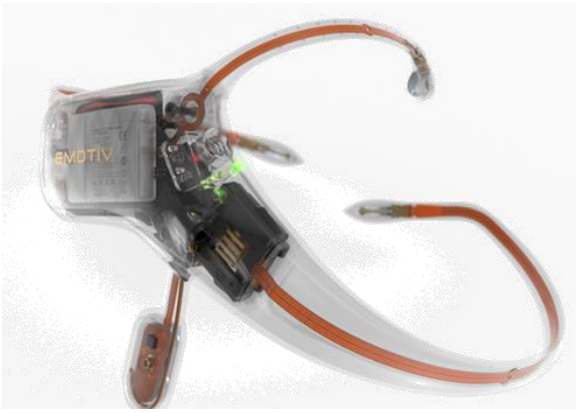


FIGURE 9: *EMOTIV INSIGHT* INSIDES

Understanding Basic Brainwaves

As an important part of this project, the brainwaves are read, getting parameters divided in frequency and amplitude. But, what are the meaning of them? The most basic ones are going to be explained.

Beta are the waves of our normal waking consciousness. It is associated with a heightened state of alertness, logical thinking, problem-solving ability, concentration, when the mind is actively engaged in mental activities. Like a person in active conversation, playing sports or making a presentation would be in a Beta state.[1]

Alpha waves are associated with a fully awake and conscious but relaxed state of mind. It is a normal pattern in creative people or when someone is meditating. This frequency enhances your imagination, memory, concentration and creativity, so this is the best state for learning and performance.

[1] Extracted from: http://fractalenlightenment.com/

Theta brainwaves are an indicator light sleep or lucid dreaming including the REM dream state, daydreaming or drowsiness, and imaginative and creative thinking led by subconscious. Also have been identify as key in learning, memory and reduction of stress.

Delta waves are associated with fast sleep or a deep meditation. Certain frequencies of delta brainwaves have been shown to trigger the body's healing and growth mechanisms.

Gamma are harder to archive waves because of its high frequencies. These are related with complex motor processes but can also be associated with very anxious or panic attack type behaviour.

It is found on The Internet or books that are some differences between brainwaves descriptions, this is normal because brain is the most unexplored, and difficult to explore, human organ.

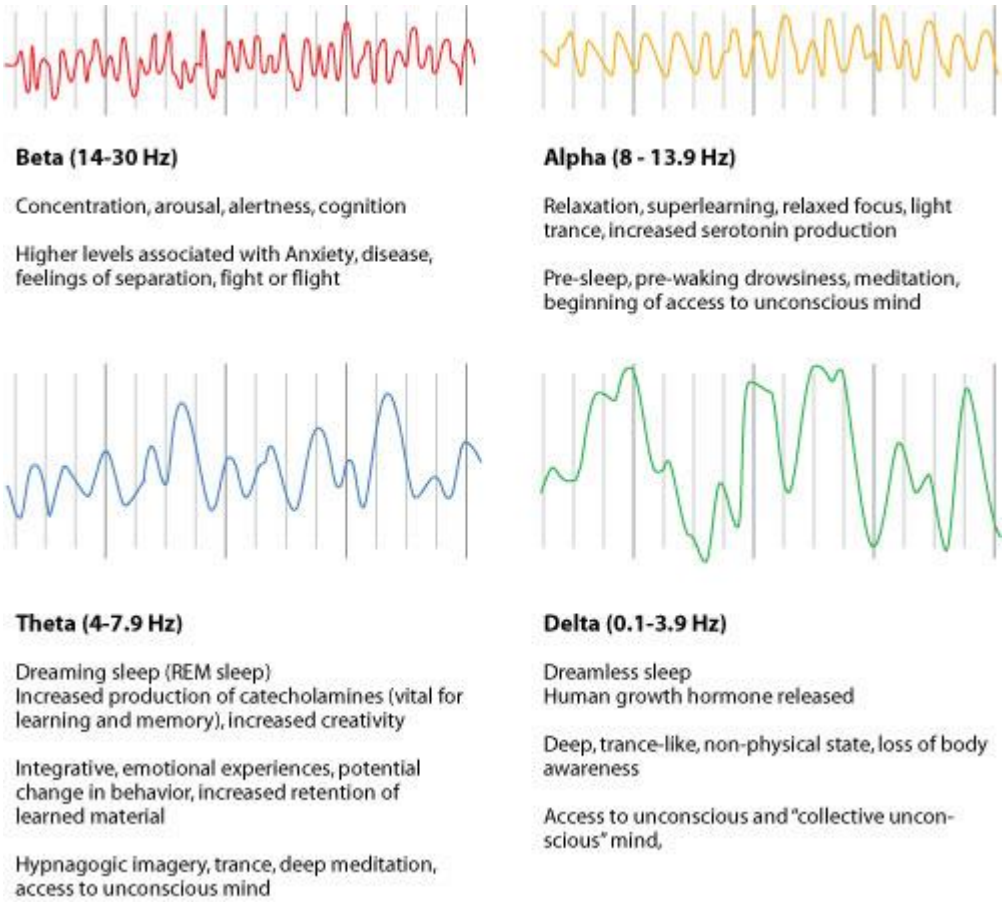


Figure 10: Brainwave frequency chart showing the different frequencies and its state of mind

Extracted from: http://fractalenlightenment.com/

Emotiv Control Panel Interface

Emotiv has its own application to get information from the headset. Here the basic interactions are shown: the connection's state, headset setup, sensors' state and managing user profiles. Also we have a small guide for setting up the headset.

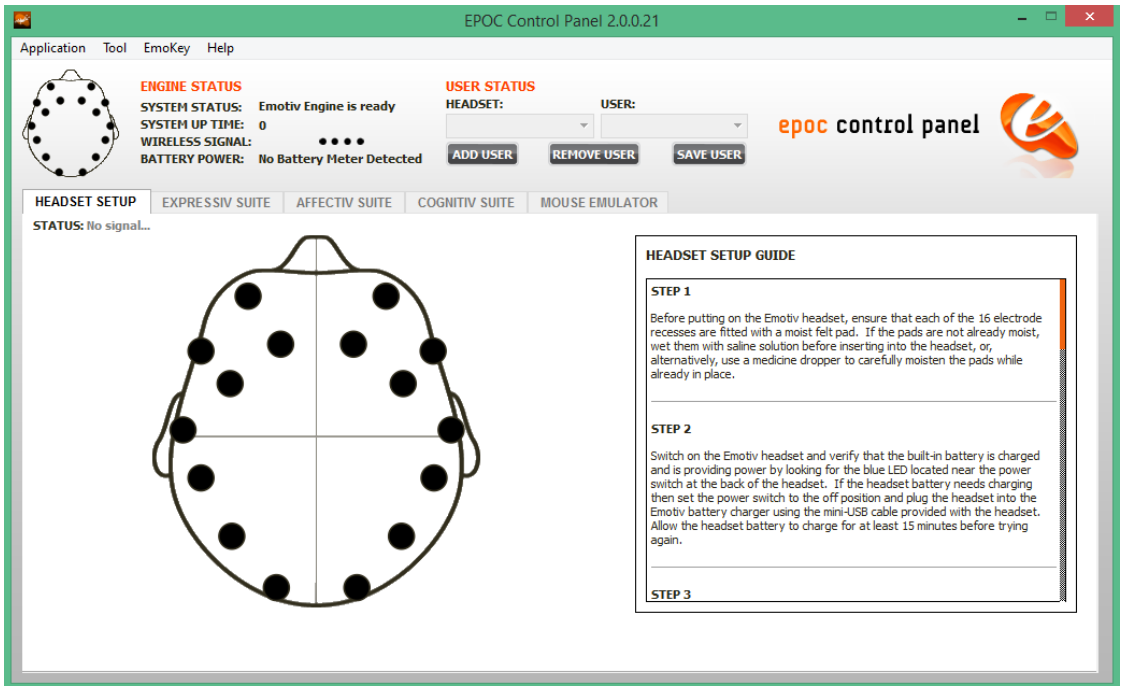


FIGURE 11: *CONTROL PANEL* HEADSET SETUP

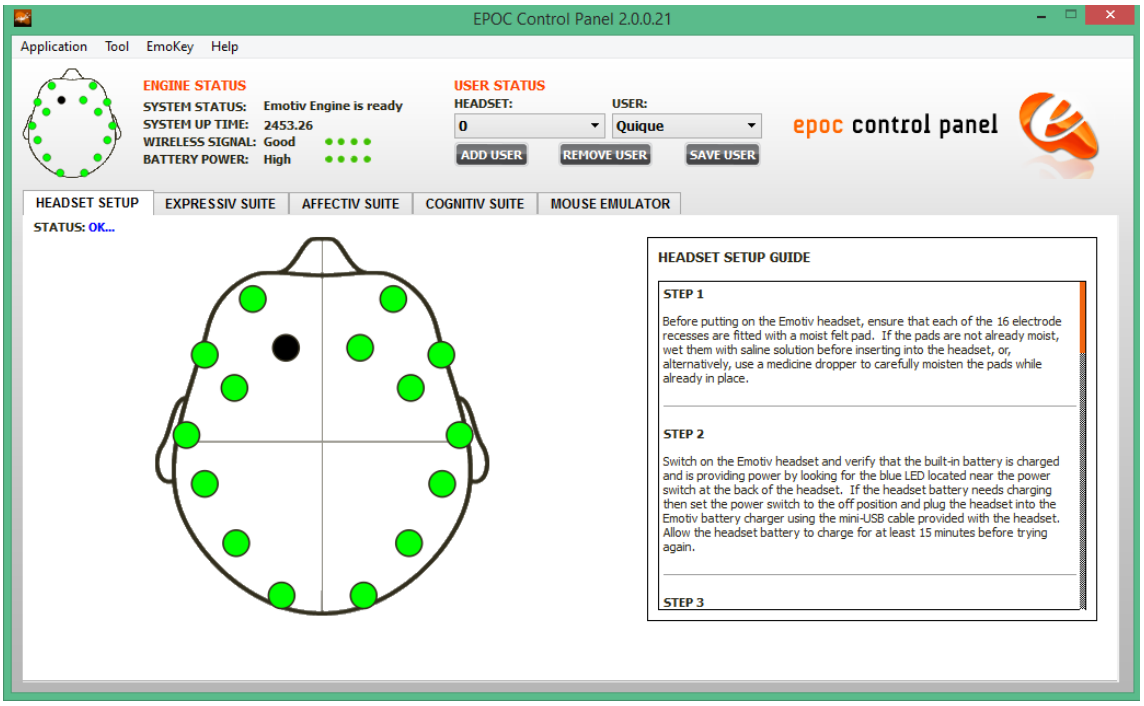


FIGURE 12: *CONTROL PANEL* HEADSET SETUP ENABLED

EmoEngine are the drivers that receive data from the headset, once installed, put the Bluetooth dongle in your PC and initiated the connection, *EmoEngine* is going to acquire data automatically.

Emotiv can provide data related to facial expressions and affective states directly to *EmoEngine*, so the *Control Panel* can get them and those are shown in *Expressiv* and *Affectiv* Suites, as seen in figures 13 and 15.

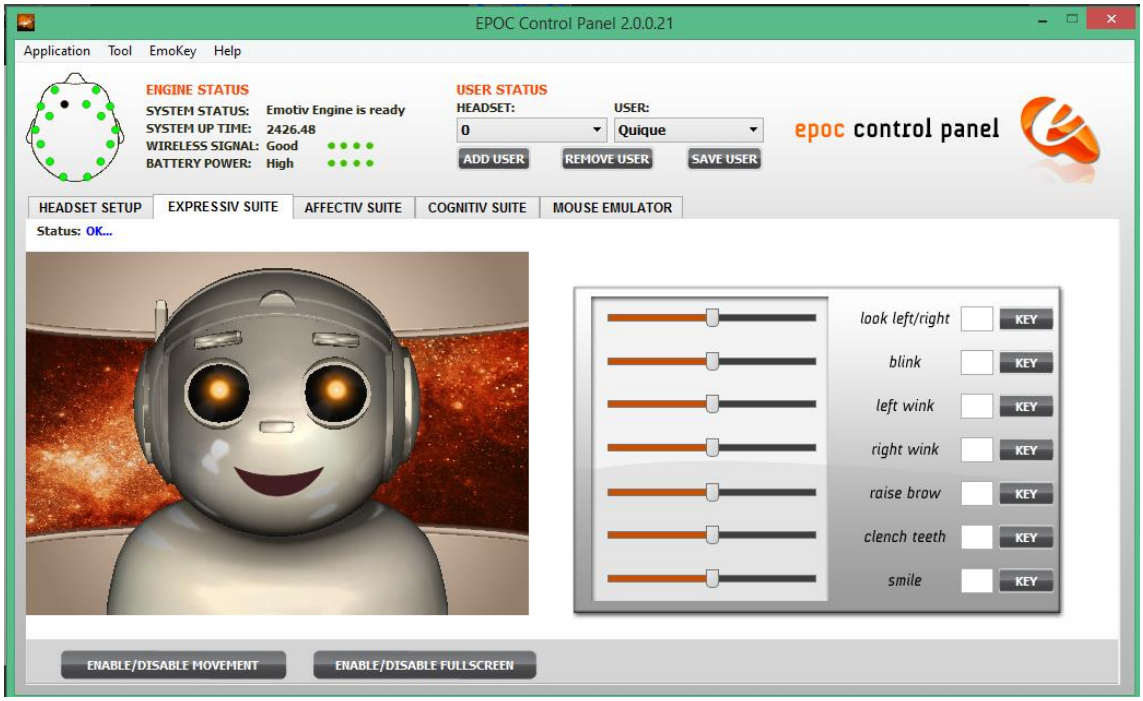


FIGURE 13: *CONTROL PANEL* EXPRESSIV SUIT

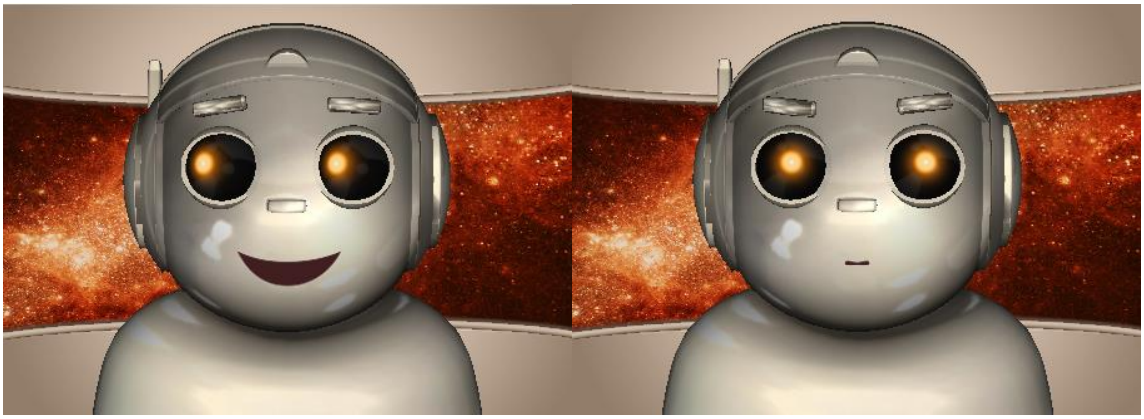


FIGURE 14: EXPRESSIV SUIT CHANGING WITH USER'S EXPRESSIONS

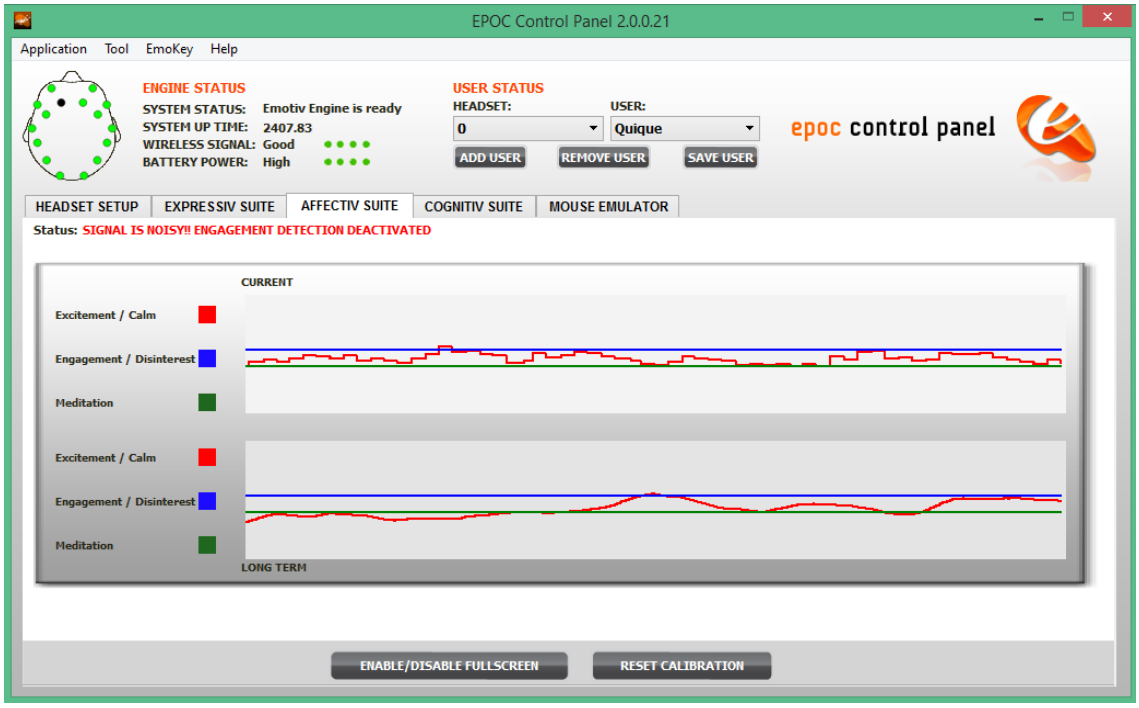


FIGURE 15: *CONTROL PANEL* AFFECTIV SUIT

Cognitiv Suit is the most interesting section of *Emotiv Control Panel* application, as that part is the one that is going to be used in this project. Here can be seen some actions to interact with the interface. First of all, there is a 3D cube that is floating in a window, which is nothing but a visual representation to help user to train and see cognitive states, so the cube is going to change direction in the space in relation with the action linked to user's thoughts. Second, there's an interface to train actions, it compares the brainwaves read with brainwaves recorded and attached to actions, so the application approximates what is being thought. That is going to be further explained in next section. Finally, the application has a percent with every action, that percent shows to the user how good is the action trained, based on the experiments made in this project, it can be said that the action is well trained when exceeds 40% more or less. That means that, every ten seconds, there are four that match with the action wanted, but that does not mean that the other six seconds match with another action, most time those are related to nothing, so that is not a real problem to control the drone.

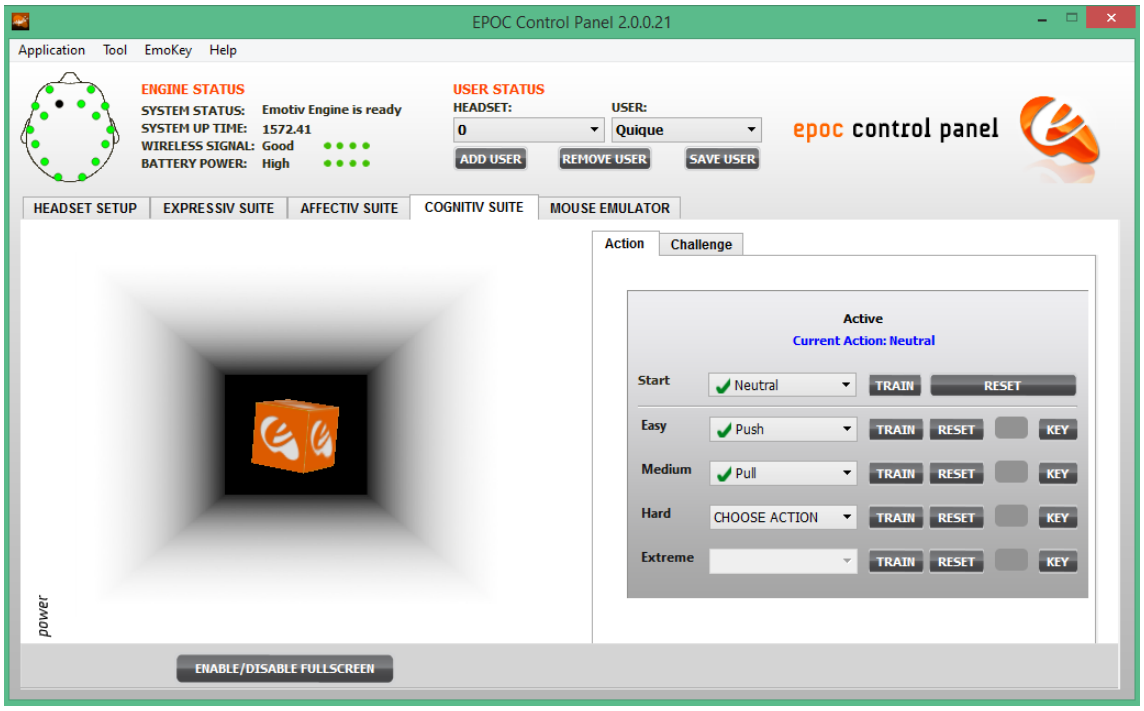


FIGURE 16: *CONTROL PANEL* COGNITIV SUIT

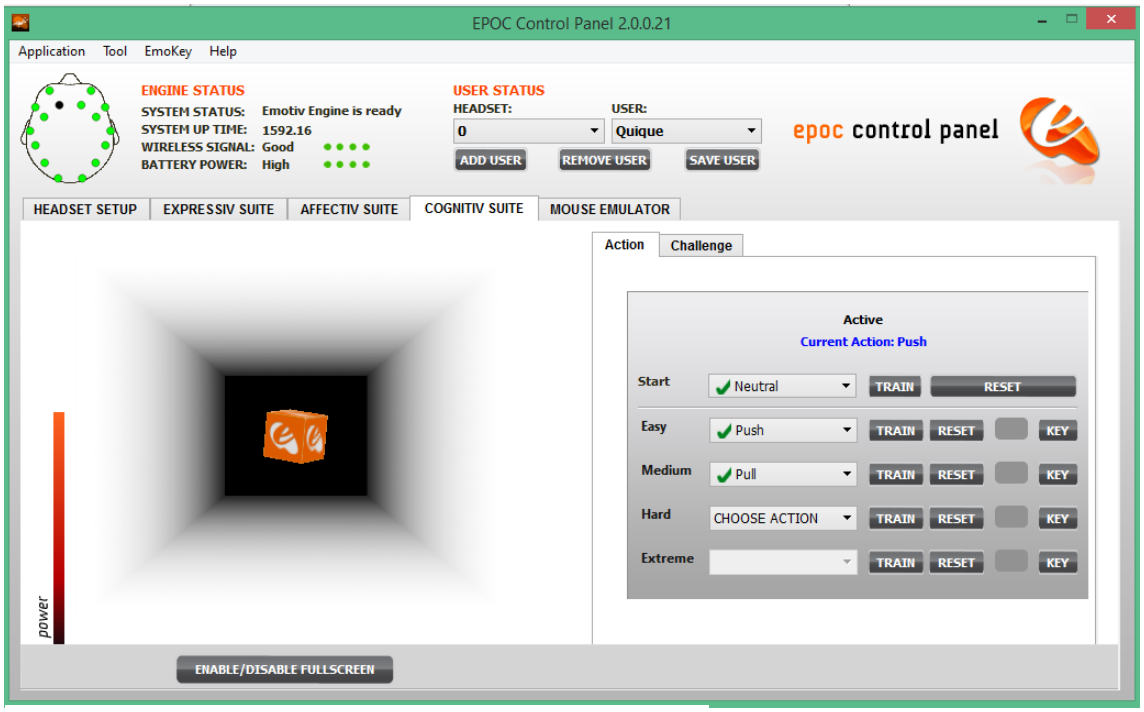


FIGURE 17: *CONTROL PANEL* COGNITIV SUIT PUSH STATE

Training to pilot

It has been said before that it is compulsory to train your mind to archive a good actions' control, and the way it is done is going to be explained in this section. Is it going to work if it is thought "forward"? What is the best way to train the user's brain? Can the user control everything with thoughts? What are the limitations?

First of all, actual technology limits cannot read the people's mind as a newspaper is read. Even if it is true that are some information that can be interpreted, like facial expressions (as said in the chapter before), the biggest part of brainwaves have not been translated or understood, so *EPOC*, the device used in this project, is not further than that. The *Emotiv* working way is to record the brainwaves and associate them to an action, so every time the user is generating brainwaves similar to any trained action, the application "understands" them. But there is a big limitation for that technology, each action trained complicates the reading and comparison further than expected. So, in this project, it has not been trained all the actions but the most relevant, even if, with more time, it could be fully controlled.

"The best way to train cognitiv is to start with recording a neutral state (relaxed, act naturally). And then train 1 action. You need to maintain the thought consistently over the 8 seconds while training is in progress. Once you accept the training, the cube is live and you can try to see if you are getting results.

Key things about Mental Commands training are (1) relax, so the algorithms don't have to cope with variable muscle signals from straining and (2) keep the visualisations as consistent and distinct as possible. More experienced users tend to be more comfortable and confident and can invoke the visualisations almost unconsciously, like moving an arm or leg. I tend to visualise a ball of fluid in the centre of my head which is squeezed forward through imaginary bars behind my eyes to make a PUSH, floats out through the top of my head for LIFT etc.

Training for an action can take up to 8 hours of training before the system recognizes your specific pattern."[2]

[2] https://*Emotiv*.zendesk.com/hc/en-us/articles/201222435-How-do-I-train-multiple-actions-

In this project has been imagined the 3D cube provided by *Emotiv*'s software and moving it from the head. It was tested different ways to obtain a better training, but imagine a cube or a ball of water are the best methods for training.

Starting point

AR Drone

Drones are the common name for unmanned aerial vehicle (UAV), more concrete it is referred as a Remotely Piloted Aircraft (RPA), which is an aircraft without a human pilot on board, as it is going to be used in the project.

Having four helixes that gives stabilization and easy control to the vehicle, the instructions used to command it are very simple, having the direction and the speed variables, which are the easiest to be used in that control.

FIGURE 18: AR DRONE 2.0

The cuadricopter's System Development Kit (SDK) provided by the company has the raw instructions given and variables gotten from the drone, thing that makes easier the integration in the project's main control program.

The drone's flight is controlled by six inputs, three axis position and rotation variables. The rotation over X axis is known as *Roll*, over Y axis is known as *Yaw* and over Z axis as *Pitch*, as seen in the figure 19. That variables are inside the drone and they can be changed by the user. Other parameters as velocity or acceleration are directly dependent from the rotation over the axis.

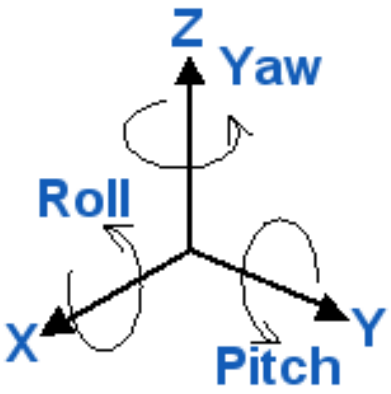


FIGURE 19: CONTROL PARAMETERS

Neuronal headset

After few testing with the headsets available in the market nowadays, as descripted before, *Emotiv* was chosen to be the used in the project as the company's devices suited better inside the idea and development process. It is going to analyse the data received by the headset in the main application program and then, the transformed data ready to send to drone. The data received is going to be thrown in the screen as a graphic, so it could be seen and managed easily.

The following two pictures are placed in order to understand the sources of the 14 data sensors that *Emotiv* have, which are rightly distributed to obtain a low error EEG

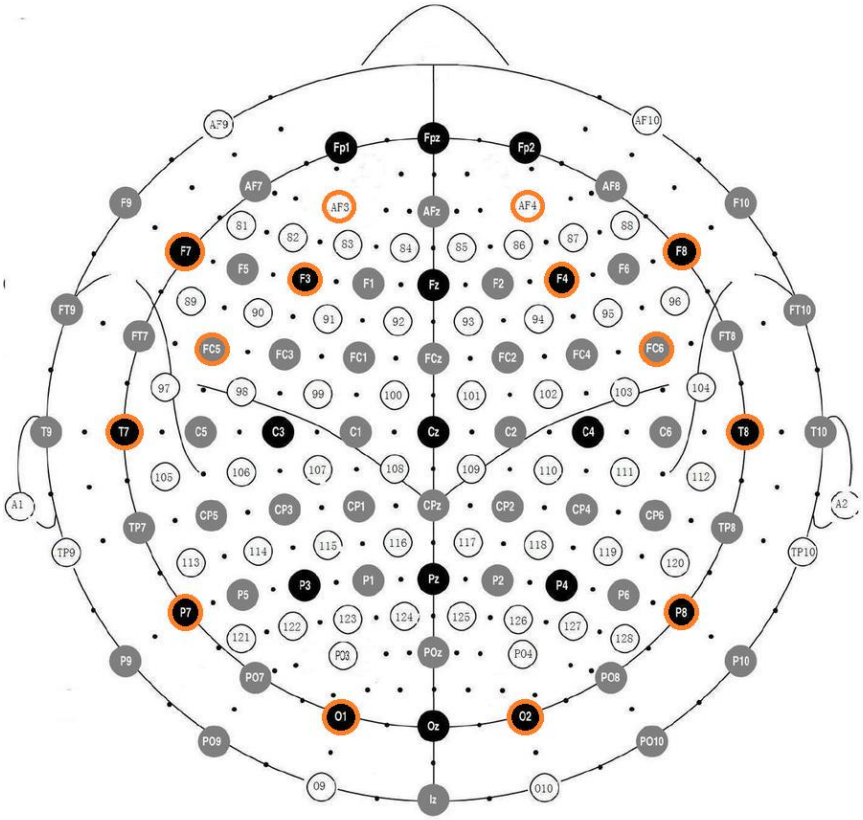


FIGURE 20: FULL EEG MAP

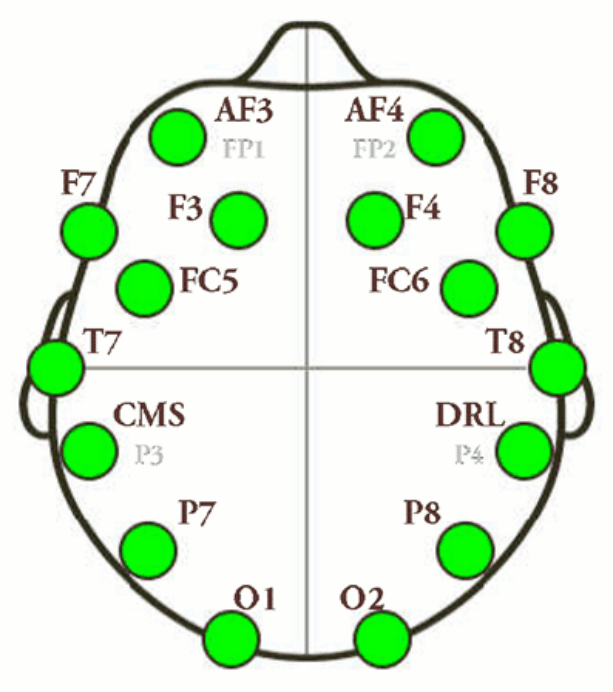


FIGURE 21: *EMOTIV* EEG MAP

Java, Android or C#?

Although the headset's API is prepared only for Java, the future version will have also the android API but, as said before, there is no assurance that the product will be on time for the project, so a problem of compatibility has appeared. There are different ways of solving the problem. One possibility is the rewriting of the API so it could be used for android. The other way is use Java as our background language, which means that the project will not be able to get used by smartphones, that was the original idea.

Programming for Windows is always the safest choice and using C Sharp (C#) language could ease the work, as it is used for that. Visual Studio, the program develop environment of Microsoft has its own section of User Interface (UI) design.

The choice

At the beginning, the better choice was to program the application for Android, but after the logistic problems in the background, the headset is not able to be used, so the best solution found is designing the application for Windows and, if the newest headset is on time, make another application for android. Finally, an automatic following by drone could be an option to be implemented, so the drone could be staying near the user without sending instructions and follow its position close.

The Develop Environment used for this project is going to be Unity 3D, a graphic motor that can ease long the work of integrating the headset into the application, can blend some program languages as JavaScript and C#, can export for Windows, Android and more and user graphic interface is not going to be hard work.

Software Development Kit (SDK)

The Software Development Kit [3] is a set of tools for the device's applications development, so everybody can write its own program to interact with the device. In this project, two different SDK are going to be used.

AR Drone SDK

The SDK of the AR Drone 2.0 has its own version for android and for Windows, which is made in C. The API references can be found in the attached document[4].

Emotiv SDK

Emotiv Insight has SDK for android and Windows (also iOS and Linux but those are not relevant for this project) and *EPOC* has SDK for Windows.

[3] SDK: Set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.

[4] ARDrone_Developer_Guide

Unity 3D

Unity 3D is one of the most used graphic motors nowadays to develop games and some applications, because it has a very complete develop environment, scripting integration and interface, and for the wide range of plugins available in The Internet.

Developing with it is completely free while the product is not sold. If your earnings exceed an amount set, beneficiaries are required to give them the 5%. That is not a problem for this project as it is not planned to be a lucrative application.

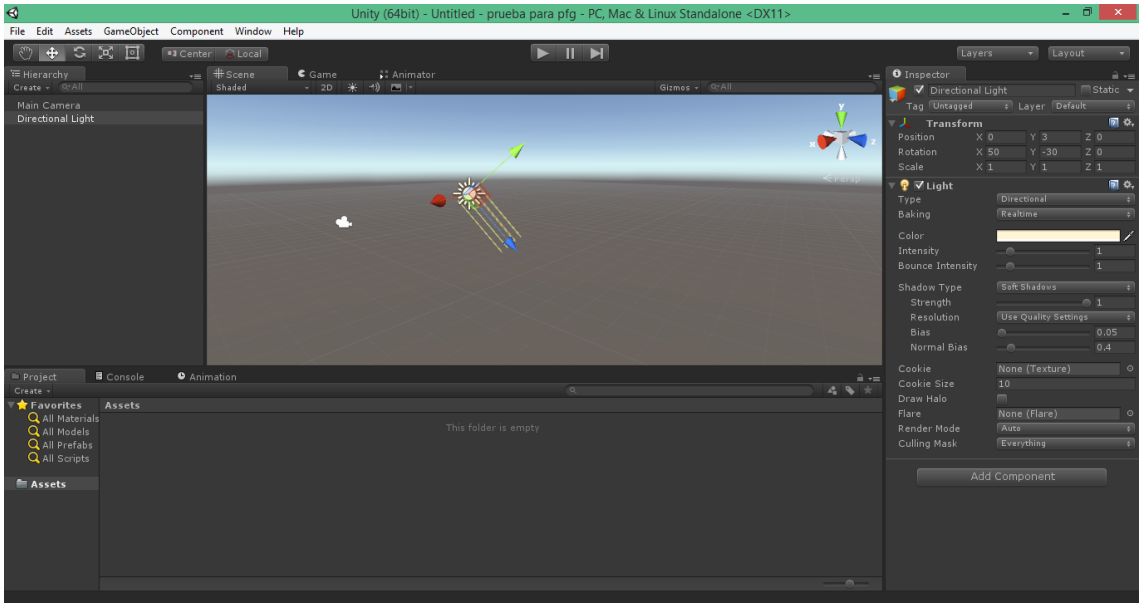


FIGURE 22: UNITY 3D INTERFACE

Using Unity 3D is much easy, so that the user only has to drop the external files into the interface and will be imported automatically. The same working way is used for linking scripts and the objects used in the scene (GameObjects).

Programming

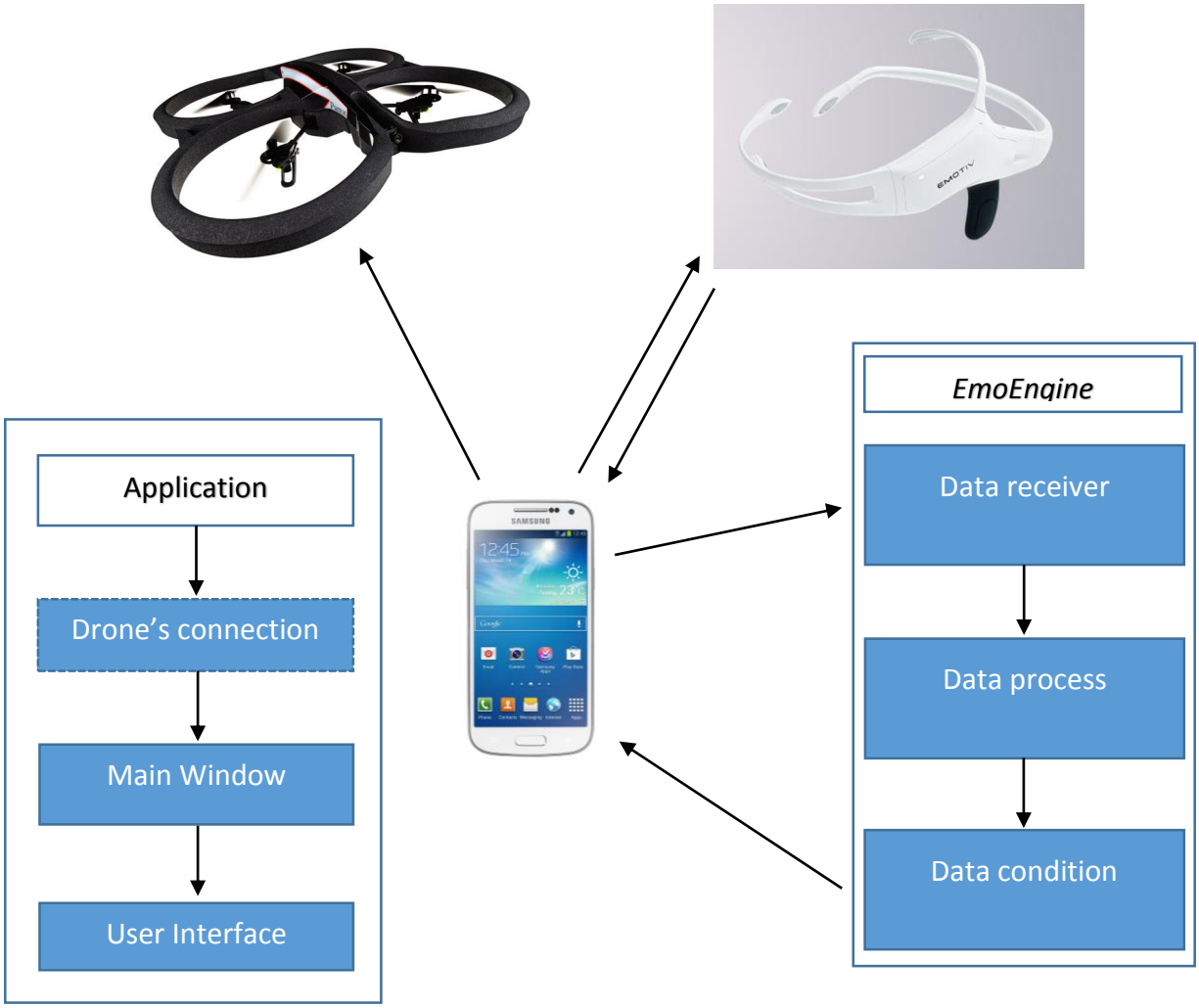
In this chapter the main program as well as the problems which came up along the project and related with it will be explained. First of all, Android programming and then Unity 3D, which is the one that has been used to build the project, are going to be explained.

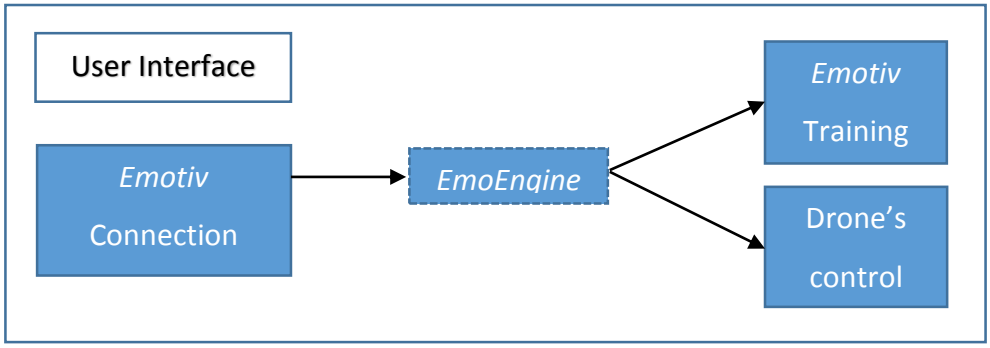
The main program has been created from another drone's control project[5] which allows the user to control it with the keyboard and live the first person experience with the virtual reality device for PC, *Oculus Rift*, which has an easy plugin integration for Unity 3D.

[5] https://github.com/scopus777/RiftDrone

Android

At the beginning, the original idea was to create an *Android* application to connect the neuronal headset with drone's movement. To have a better understanding of this written part, a connection's scheme has been done.





The Android application has been made on *Eclipse* develop environment with Android SDK libraries to export it as an Android application. In this section, the program built is going to be explained and commented.

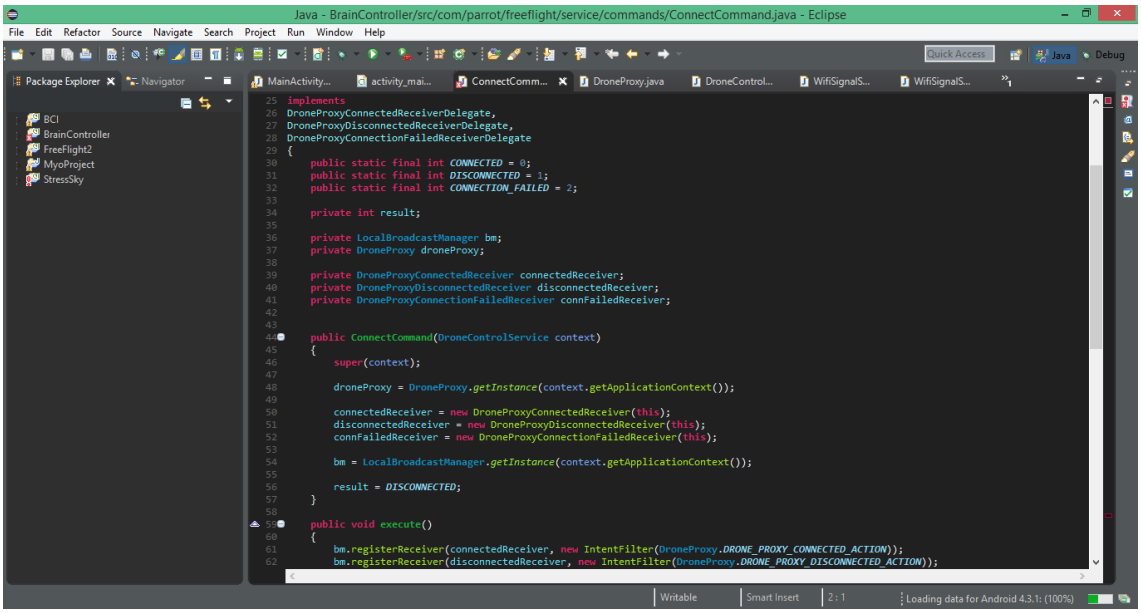


FIGURE 23: ECLIPSE INTERFACE

Reading and understanding the drone's application took more time than expected because of the different classes that were used to creating the User Interface (UI) design.

First of all, the *Emotiv* part, which creates a Bluetooth connection between the Android device and the user's headset. The connection function uses the libraries associated to *Emotiv Insight*'s development kit, which has a written Bluetooth protocol connection. This protocol is called from the main program to create a connection event.

LEdk.IEE_EngineConnect (this, ProfileLocation);

After that, to use the received data, a function is called to send a request and store the information into an *EmoState* variable, which is the one that also stores the cognitive states of the user's brain, sending the brainwaves to *EmoEngine* and getting the right cognitive, expressive and affective states.

In the *Emotiv* library there are not only functions, but also enumerations with tags. Once the cognitive state is read, this is compared with that tag list, in case the generated brainwaves fits an action-related tag, then the signal gets conditioned and the

application sends the appropriate command to the drone, which receives the data and the movement starts.

droneProxy.setControlValue(CONTROL_SET_PITCH, power);

At the end, the new headset did not arrive on time, as it was the newer technology available in the market and were on development state when the idea of this project began, so the code that had been built is useless from this point: the drone's control commands and the headset connection interface, just lacked the training and profile management interface and the testing.

As it was previously mentioned, the code was useless for this project, but can be used to go with a further development when device got available. To conclude, even if the connection and commands built, the *Emotiv* interface did not work because needed the other code part. But, as the other controls were not deleted, the drone could be controlled by a Smartphone or Tablet.

Even that, the code done is going to be used to continue with the project when the new headset arrives. The code made builds and it is no disturbing the main drone application. So, even if there is no chance to test the neuronal headset connection, the project can be continued from this point.

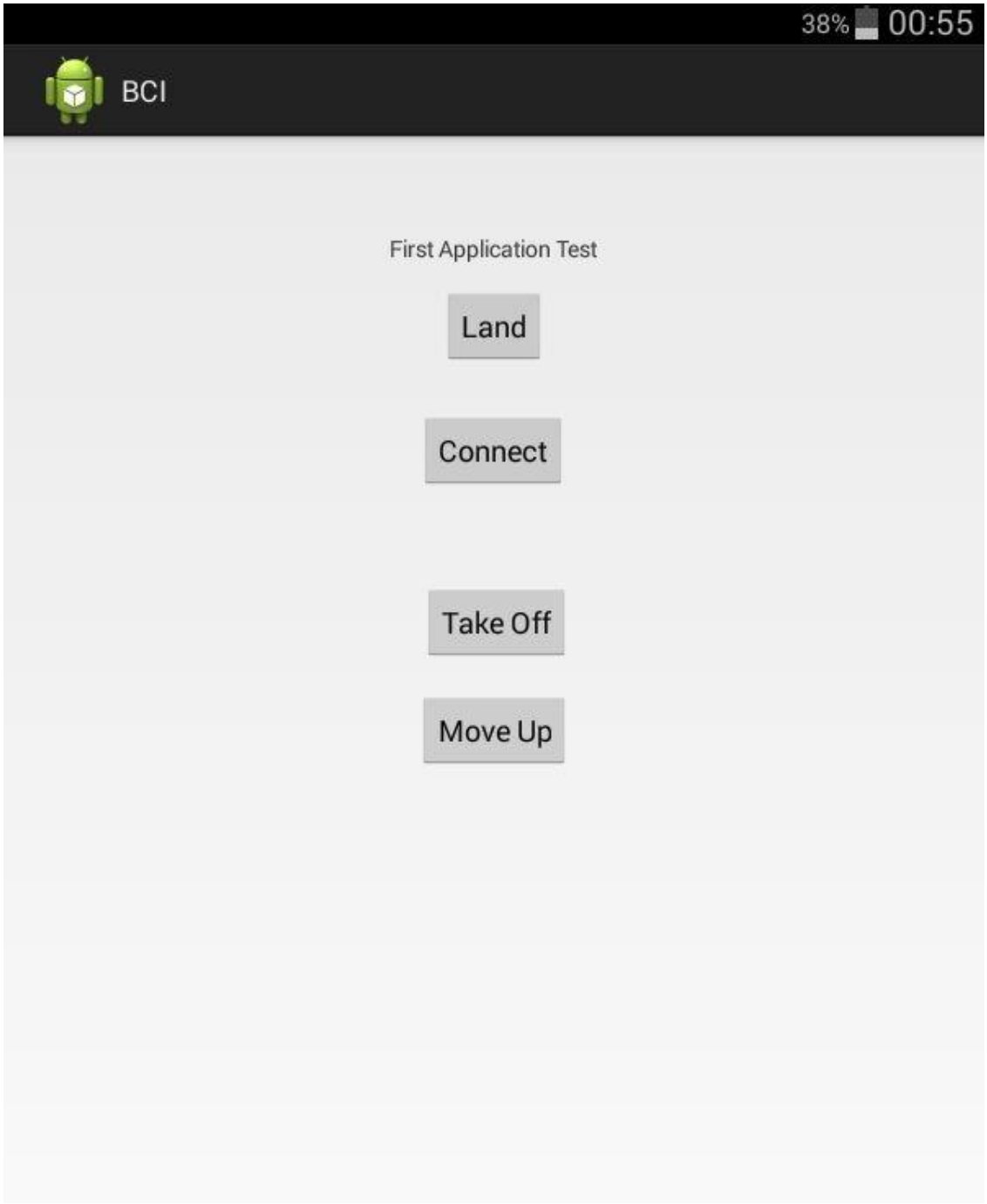
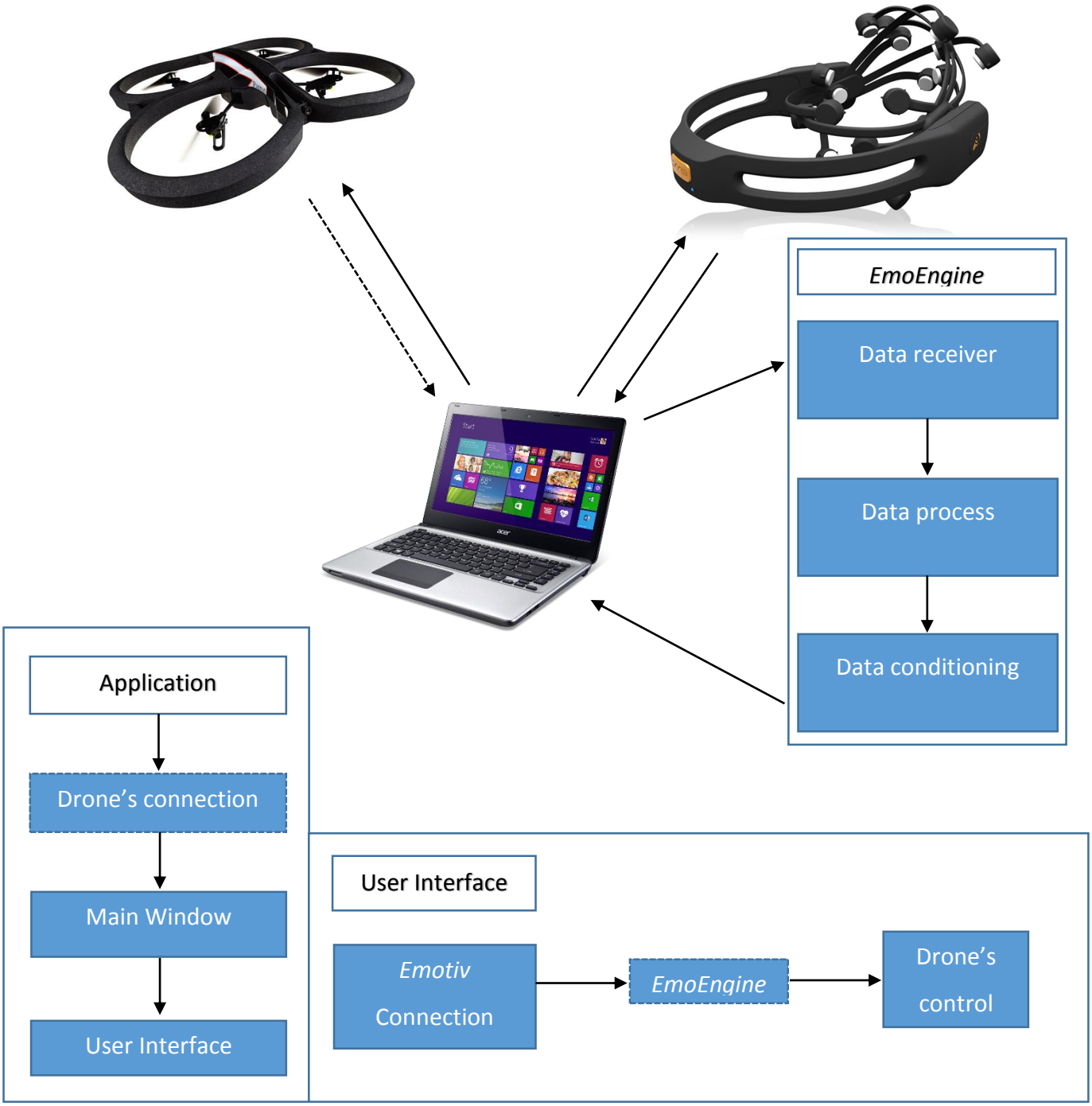


FIGURE 24: ANDROID APPLICATION INTERFACE

Unity (C#)

The project program has been done using Unity 3D, as said before, and the connection between the user's device and the neuronal headset has been written in C# language. To understand this section, a connection scheme has been drawn.

EmoEngine are the drivers included with each *Emotiv* device; it manages all the brainwaves data and internally transformed data.



Using thoughts in the application

To understand how the application works, it has been mandatory to create a signal processing using Matlab, a mathematics program that helps with calculations and it can also run program files.

The Matlab code built reads the 14 raw sensor data for 10 seconds at 128Hz and stores them into a data array, then this data are plotted to see the brainwaves generated.

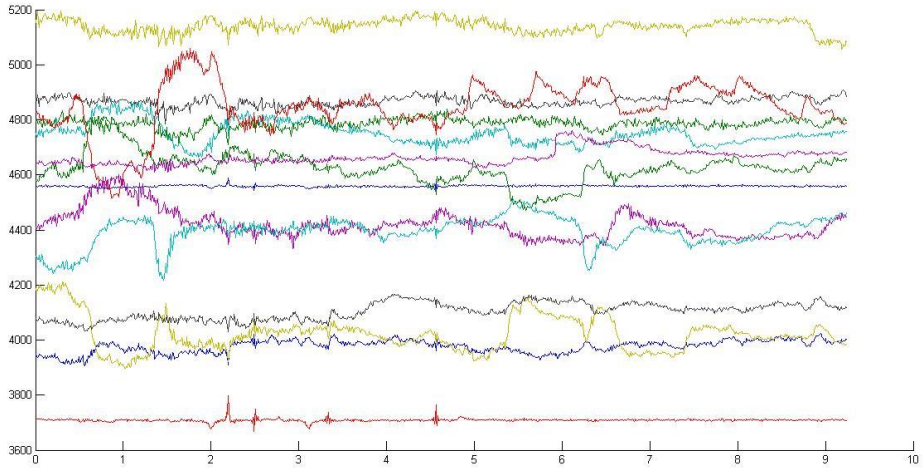


FIGURE 25: *EMOTIV* EEG

The data received are interesting for investigation but has no relevant information for the project with this form, so it has been stored and compared with next 10 seconds brainwaves (figures 26 and 27). The figures shows 14 data plots that correspond to each headset neurosensor. The voltage difference is because of the sensor distinction.

The data compared is understood as the error between thoughts (brainwaves), so the next step was conditioning the error using two virtual Schmitt Triggers that supply stability to the thought comparison. So, it understands that two thoughts are similar when having a ±2µV difference and continue being similar until having more than a ±5 µV difference (figure 29).

The Error EEG plot deals with the difference between both two saved data, which are broken down to get the full information, even if they are not important for the project.

After that, the figure 29 shows the mean error between both saved data with the full 14 sensors. The data are the used with Schmitt Trigger values to obtain the cognitive states.

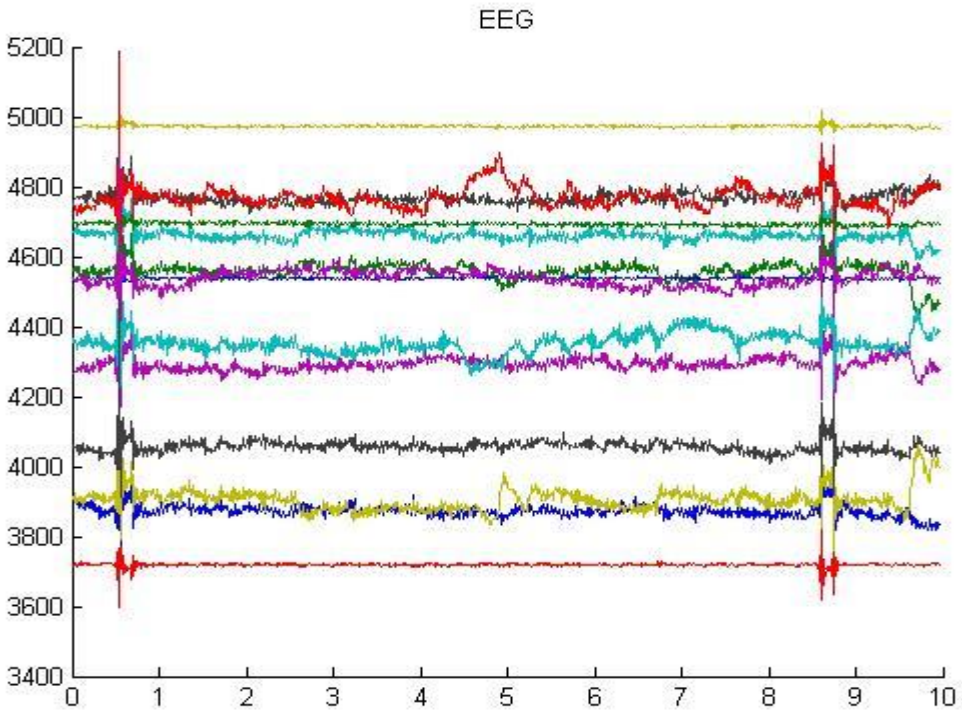


FIGURE 26: ORIGINAL EEG

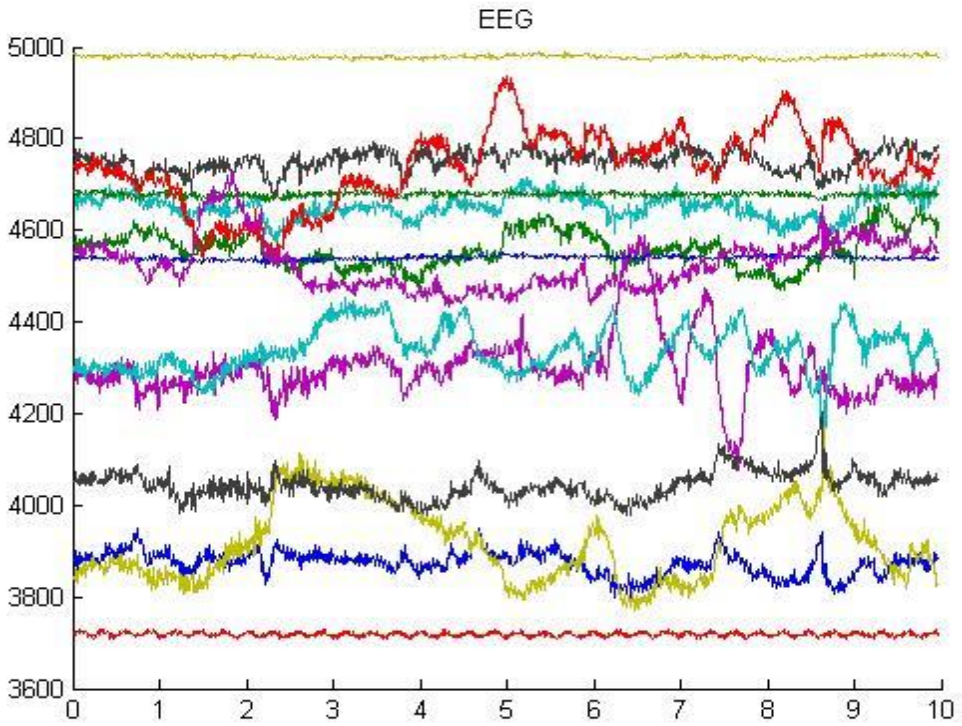


FIGURE 27: COMPARED EEG

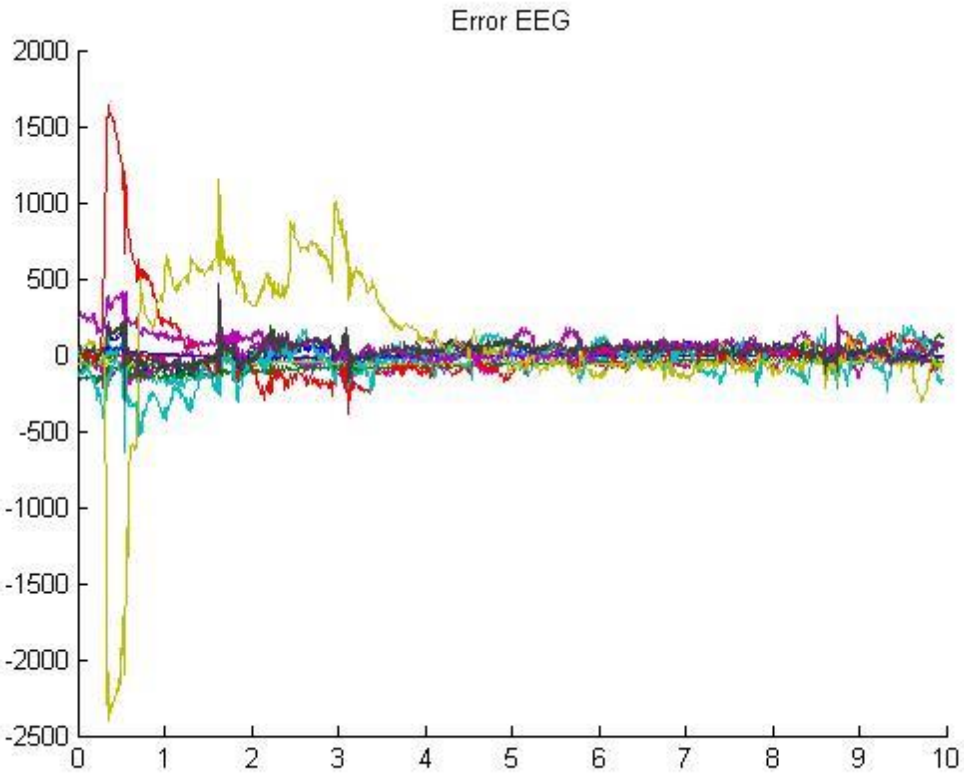


FIGURE 28: MEAN ERROR FROM EACH SENSOR

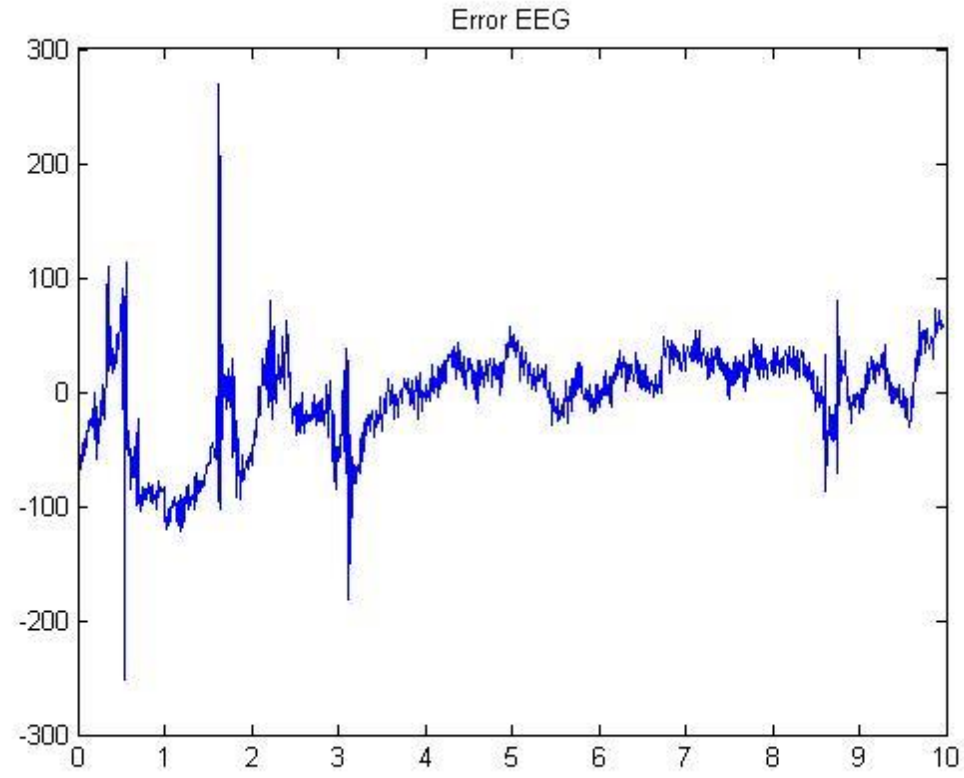


FIGURE 29: TOTAL MEAN ERROR

The data received is processed by the *Emotiv* drivers installed and returned as a cognitive state. The cognitive states return a value between 0 and 5, corresponding to the signal strength, which has been calculated dealing with the error between the brainwaves. The EEG Error and the Neutral and Push States generated when error EEG has crossed the Schmitt Trigger values can be seen in figure 31.

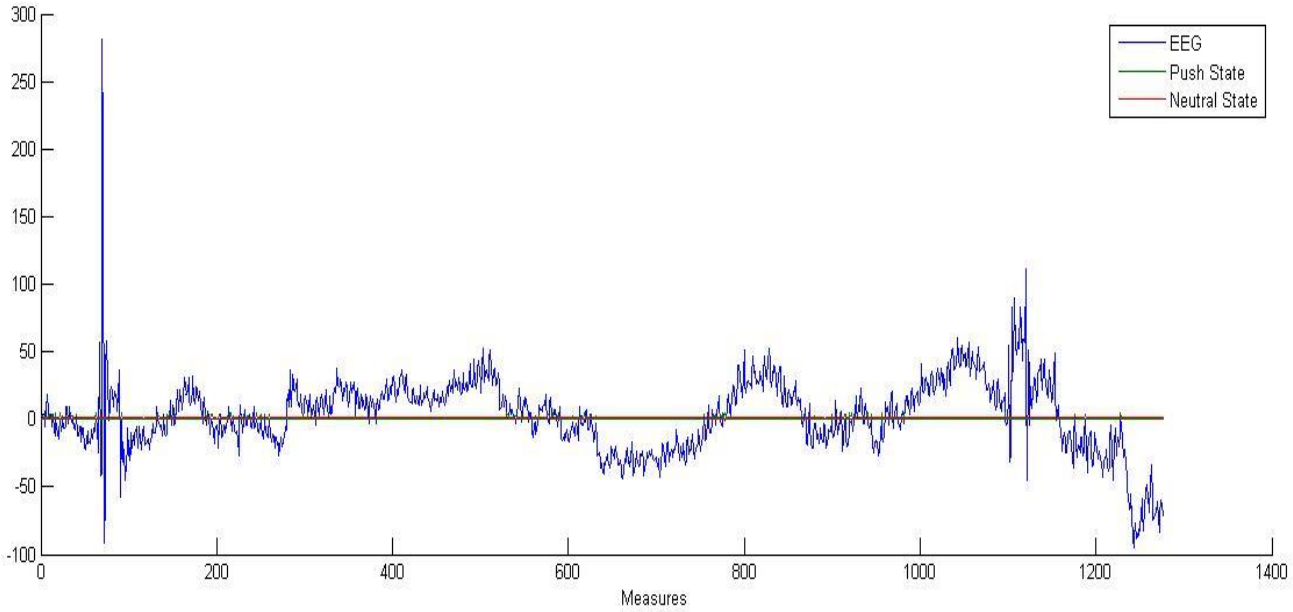


FIGURE 30: COGNITIVE PROCESSED DATA

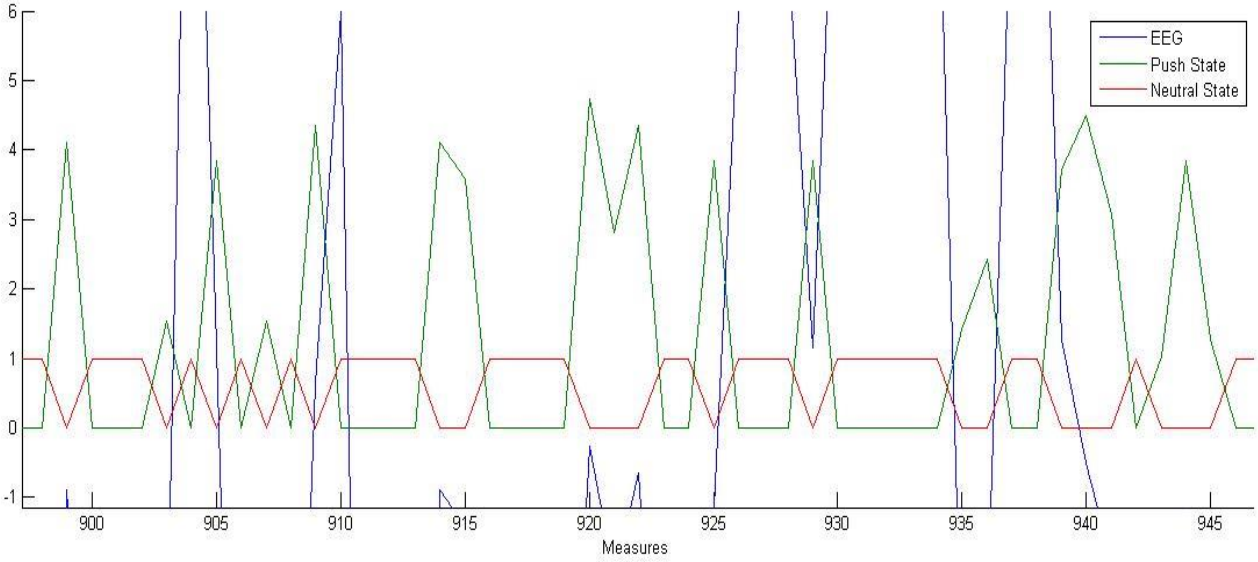


FIGURE 31: ZOOMED COGNITIVE PROCESSED DATA

If the data corresponding to alpha, beta, gamma and delta values were important, they could be obtained with the right interpretation, dividing brainwaves by frequencies applying filters, though there was no reason to do that in this project.

The SDK comes with *Edk.dll*, a library with functions and variables. This library has different enumerators with tags that are used to compare cognitive, expressive and affective states.

For this application a comparison between the user's actual cognitive state and the tags provided from SDK has been done, so every time the user's thoughts are similar to one recorded brainwave sequence in relation to a tag, the program is going to understand that the user is thinking of an action. After that, the program is going to trigger the brainwaves and translate them into the drone's corresponding variables (roll, pitch, yaw and gaz), which are sent to the user's drone. The drone's input in relation with cognitive state "Push", which change pitch value from 0 to 1 (15 degrees) if the state is active or not, can be seen in figure 35.

EmoState emoState = getCognitiveState();
If (emoState.CognitivGetCurrentAction() == EdkDll.EE_CognitivAction_t.COG_RIGHT)

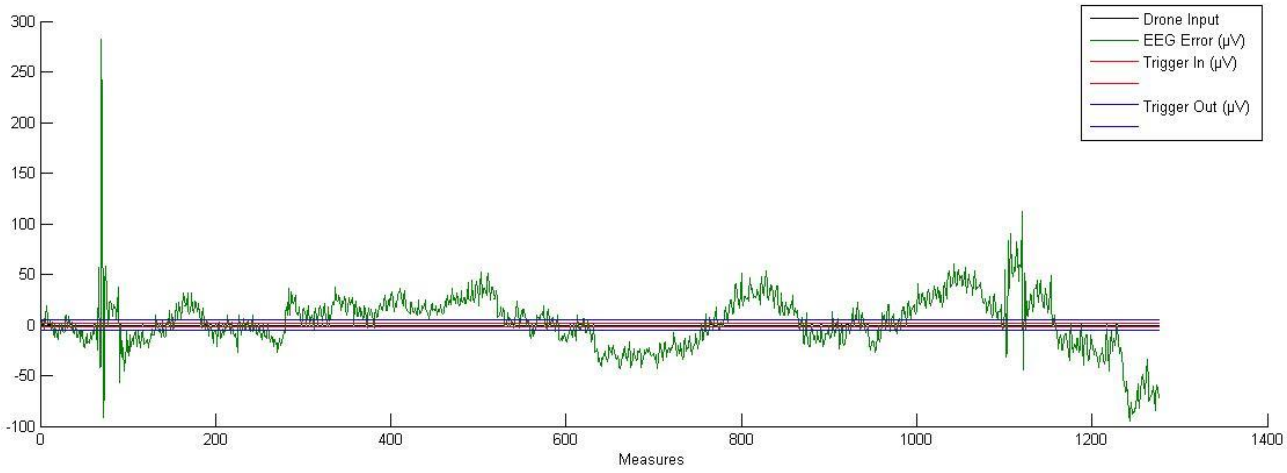


FIGURE 32: DATA CONDITIONED

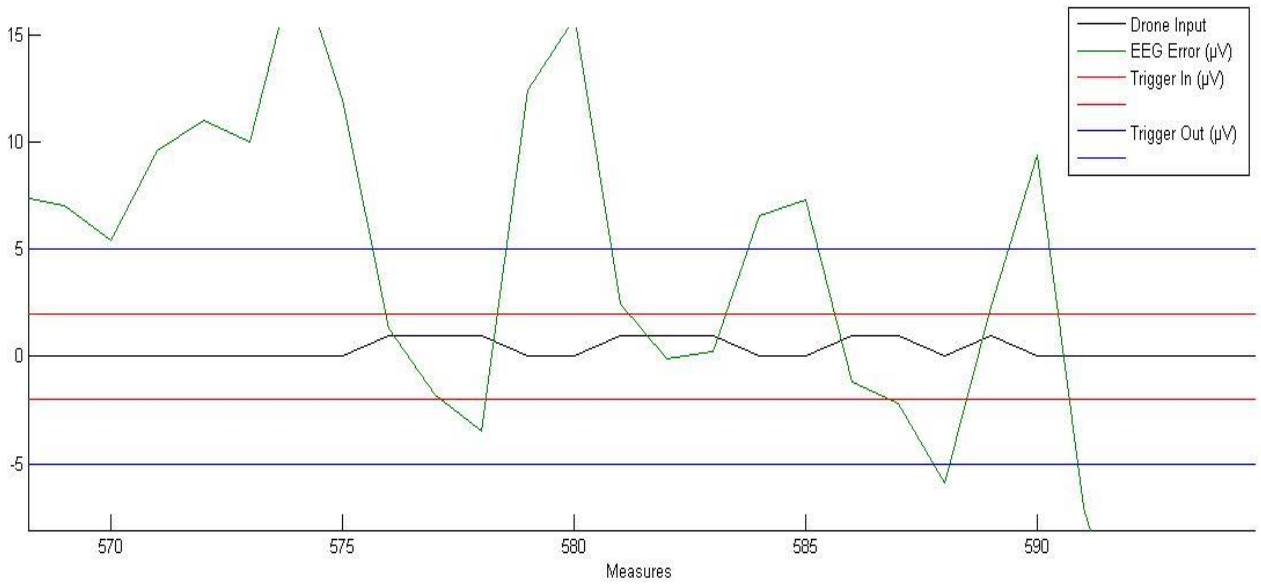


FIGURE 33: ZOOMED CONDITIONED DATA

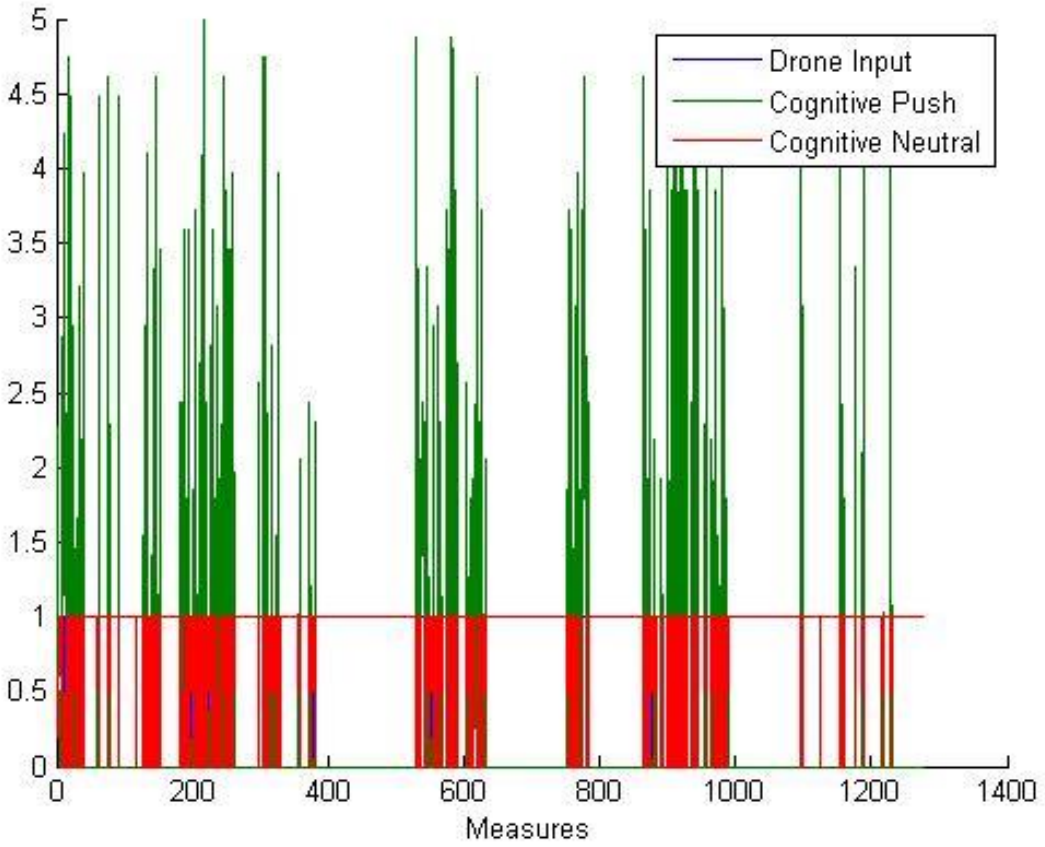


FIGURE 34: OUTPUT DATA

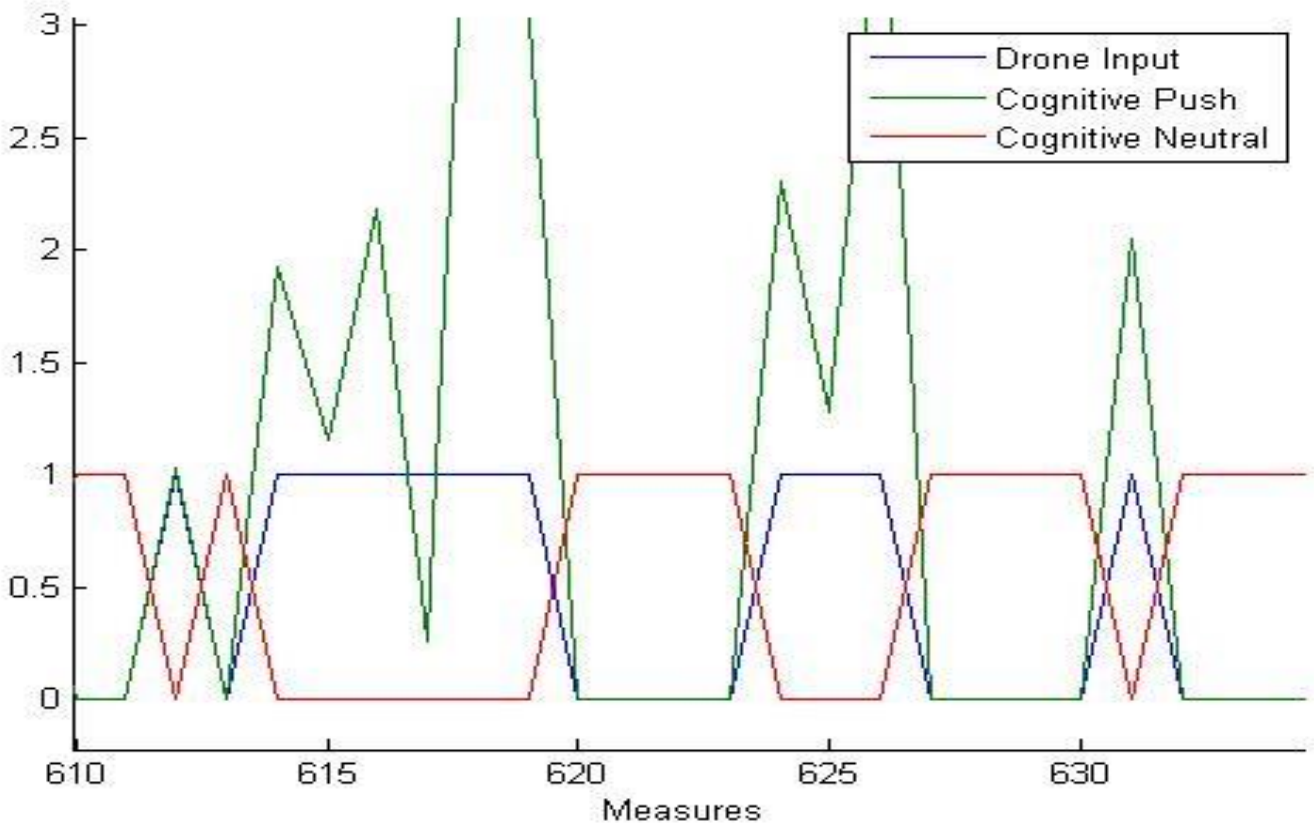


FIGURE 35: ZOOMED OUTPUT DATA

The last part is the user profile's loading, so the application can translate the received brainwaves into the cognitive states trained before. This scripting part gave some problems as using the provided functions the program didn't load correctly the "*.emu*" files, which is the file extension for *Emotiv* profiles.

engine.LoadUserProfile (userID, "profile.emu");

Different methods were tried to load *Emotiv* files. Firstly, it has been tried to load the user directly with the *Emotiv* library (*EdkDll*) function, but it was found that some problems happened with functions related to get the user profile and to set it, so that method was rejected.

Then, another solution found was to load the profile directly from *Emotiv* library, but using *EmoEngine* class in order to manage the profile information, getting and setting it into the scene created.

It has been supposed that load user's function has some bug inside as the *EmoEngine* class function is nothing but an easier interface between the application and the library. So *EmoEngine* uses the *Emotiv* library to send the user's profile. In the end, the application worked fine.

Even after building the application, nothing but *Expressiv* worked when testing with headset, so this solution was not enough.

After some search that *Emotiv* has two SDK versions, Research and Developer Editions was found[6]. The last one has no limitations meanwhile Research Edition has some, and it is not done to receive cognitive actions. After checking the version used, it was the Developer Edition, which was supposed not to have any problems.

Another solution found was to use a remote connection, and link the application to the *Control Panel*, which can manage profiles, train and more functions previously explained. The problem was having another application open so that one of them worked, which was not the initial idea. But, even having built the application, it didn't work.

It was found in the official forum[7] that one solution is to load the profile into a byte array and then convert it into an *Emotiv* Profile variable class. But it was not thought for Unity integration and has some problems with the linker interface class, throwing a convert file extension error.

Fortunately, the *Emotiv*'s technical service helped with this problem and has provided the right function to use when using Unity 3D, which is the last test improved version, loading the profile into a byte array and use it as a profile file.

[6] http://198.176.29.113/forum/forum12/topic1938/messages/
[7] http://*Emotiv*.com/forum/forum15/topic1371/messages/

AR Drone 2.0

The program used to understand the connection between the device and the drone, functions and main commands of this was *Parrot FreeFlight 2.0,* which took more time to read and understand than expected. Even that, it could be possible to extract rewarding information, the way to connect and send commands.

To start, the connection between drone and user's device (PC) is done by Wi-Fi. This protocol connection came programmed with the base project that has been chosen to work. The user has to connect the device to drone's generated Wi-Fi and, when starting the application, the data transfer initializes.

Navigation Data *(Navdata)* is the information about the drone that is returned for it, as the position, speed and status. This data is sent by drone to its client on UDP port 5554.

AT Commands, sent on UDP port 5556, are the data sent to drone to control and configure it. All the values can vary between -1 and 1, which corresponds to a maximum angle set, 15 degrees in this project for pitch and roll. That values are also used to set a height increment (gaz) and to set a turn angle speed (yaw). Another value out of range is going to throw an error.

AT command	Arguments[1]	Description
AT*REF	input	Takeoff/Landing/Emergency stop command
AT*PCMD	flag, roll, pitch, gaz, yaw	Move the drone
AT*PCMD_MAG	flag, roll, pitch, gaz, yaw, psi, psi accuracy	Move the drone (with Absolute Control support)
AT*FTRIM	-	Sets the reference for the horizontal plane (must be on ground)
AT*CONFIG	key, value	Configuration of the AR.Drone 2.0
AT*CONFIG_IDS	session, user, application ids	Identifiers for AT*CONFIG commands
AT*COMWDG	-	Reset the communication watchdog
AT*CALIB	device number	Ask the drone to calibrate the magnetometer (must be flying)

TABLE 3: AT COMMANDS

Drone's control

The drone's control script used is one AT Command that sends five variables related to roll, pitch, yaw and gaz, which are values between -1 and 1, and the flight type.

```
droneClient.Progress (AR.Drone.Client.Command.FlightMode.Progressice, pitch:pitch,
                      roll: roll, gaz:gaz, yaw:yaw);
```

FlightMode refers to flight type that is going to carry out. In the project only first and second mode are going to be used because of the limitation of our device, that is a Personal Computer (PC), which has not got gyroscope, and because of the target of this project.

There are four modes in this application script:

```
[Flags]
Public enum FlightMode
{
        //The hover.
        Hover = 0,

        //The progressive is a flight motion enabling bit.
        Progressive = 1 << 0,

        //The combied yaw.
        CombinedYaw = 1 << 1,

        //The absolute control.
        AbsoluteControl = 1 << 2
}
```

Hover, which is used to make drone's flight stable at one point.

Progressive, that is a command used to set flight parameters and make drone move. It has no effect when the drone lies on the ground.

Combined Yaw is a new mode that combines yaw with roll to get an easier control mode for racing games.

Absolute Control, when all the commands will be considered in the controller frame instead of the drone frame, for example, using it to link drone's control to gyroscope's device.

Hover is set every time the cognitive state is neutral, so the drone is going to be stable between commands. Because of some problems during test part, explained later, take off and land will be safer linked to a keyword.

If (Input.GetKey("space"))

In order to control yaw variable (turning around), a new concept is going to be introduced, linking yaw to eye position. With this improvement, changing orientation is as easy as looking to right or left direction. The same integration has been done to control gaz variable, looking up and down to control height.

EmoState emoState = getCognitiveState();
If (isLookingRight == true)

It was thought that this data acquisition was automatically enabled from *EmoEngine*. Nothing further from the truth, because its own handler had been written. After that, the application returned data from your facial expressions.

In conclusion, the drone has been controlled using cognitive states in relation to forward and backward (pitch) and right and left (roll), to control the drone's height (gaz) and turning around (yaw), the expressive position of the user's eyes has been placed. Finally, the drive power has been set between 0 and 1 because of the difficulties to maintain a constant cognitive state.

User Interface (UI)

The user interface design done is based on a main Window with three components. The first one is one big screen that is used to render the drone's frontal camera. The second is another screen smaller in the right-down corner, which is used to render the drone's bottom camera, but none of both screens are fully developed and fixed to obtain and project the drone's videos. The last component is a button that initiates the connection protocol between the neuronal headset and the application.

There are two more components and they are texts. Both components are used to showing application status information, one to Wi-Fi power signal and the other to battery and flight measures, in general, the drone's status information.

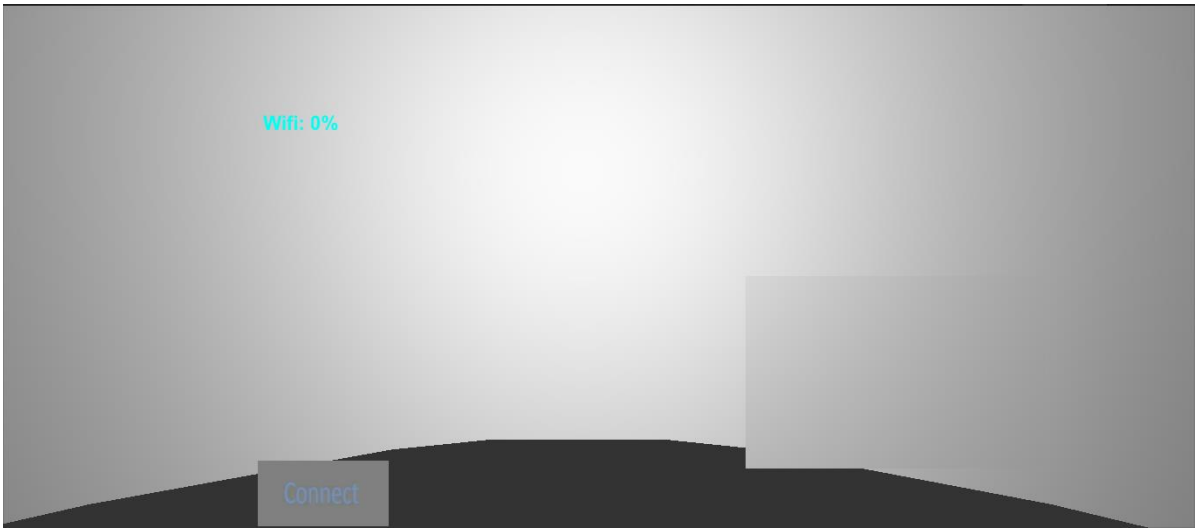


FIGURE 36: APPLICATION INTERFACE

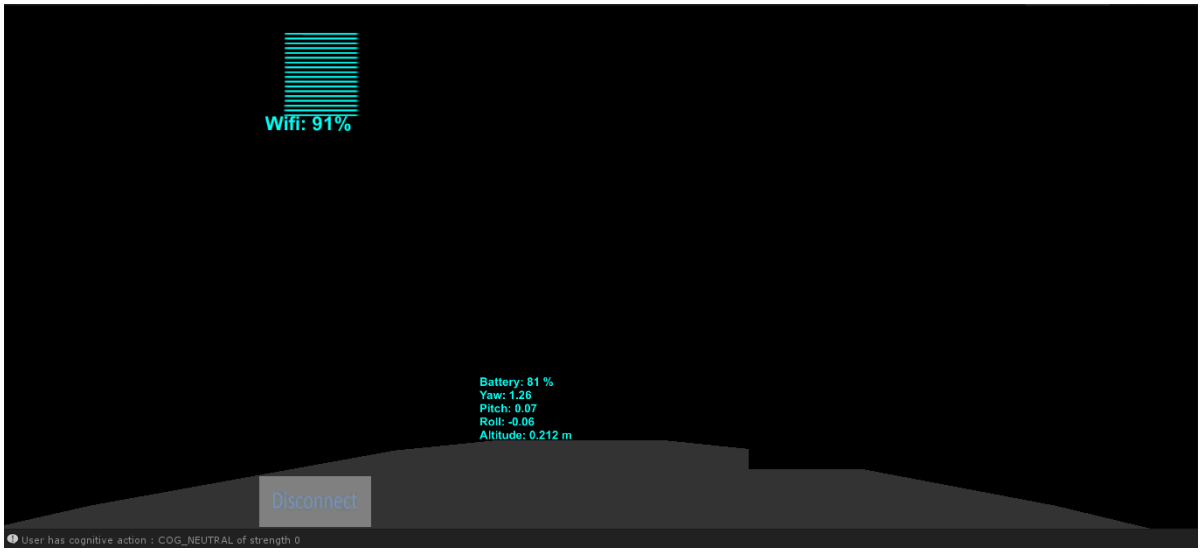


FIGURE 37: WORKING APPLICATION INTERFACE

Testing

Through the development of the project it has been done several tests to know the current state of the application, so the main problems could be fixed and tested again. In this section it is not going to explain each test because every part was tested with every change.

First of all, it has been tested the connection between the application and the drone. The testing was early satisfactory as the drone answered fine the commands, the most problematic was the drone's battery, because it has about 10 minutes of autonomy and, further than that, the charger it has been used was broken, so the batteries were not charging and it has been needed to borrow another one to the University project's tutor, who, with drone's section director consent, lent it without problems.

After that, it has been added the neuronal headset connection and tested. Because of discrepancies between C++ and C#, which is the one it has been used, there were some problems to connection. At the beginning, it was tried to make a remote connection, linking it through *Control Panel*'s interface, which could ease the work, but, after making a table of decision, a direct connection was chosen, because of the user's facility when using the application and because of making a complete and compact application project.

Inside the neuronal headset connection test, there were two points to prove. On the one hand, program the headset connection as a condition to control the drone, so, if the headset is not connected, the application doesn't send commands to drone's flight. And, on the other hand, testing if the orders sent by headset are being interpreted in the right way. The last one has been done making a debug application that gives you feedback about the cognitive state, besides the application writes in the screen the last "thought" the user had.

This was a very tricky part and some problems appeared with the brainwave recognition, as explained before. Therefore, when testing flight, drone was hovering always and nothing but keyboard control worked. Even expressive control didn't worked, because of forgetting about handle data. But, after some changes, the expressive control worked.

First Expressiv Test Video.

The last test has been done with the whole application finished, trying to control the drone with the neuronal headset interface's application made. First time it was turned on, drone went directly to the wall and an exception was thrown, so drone stopped. Another time, drone was taking off and landing intermittently, result of having the commands linked to thoughts, so the script was changed. Finally, drone flight well with only a problem with the program base, which breaks after running few minutes, thing that can be a problem when flying a while.

First Cognitiv Test Video.

Conclusions about the final tests were that, even if the drone's movement is controlled by the user's brainwaves, not all the axes can be controlled, as the more axes are added, the more difficult is the application to control. So, in the end, the application has been made to control four axes (forward, backward, right and left) and letting height (gaz) controlled by keyboard, the same as taking off and landing, what also helped to gain some security measures, because it is not a good user experience if drone lands suddenly because of any instant reading error.

Second Cognitiv Testing Video.

Third Cognitiv Testing Video.

FIGURE 38: TESTING EXPRESSIV NEUTRAL

FIGURE 39: TESTING EXPRESSIV LOOKING RIGHT

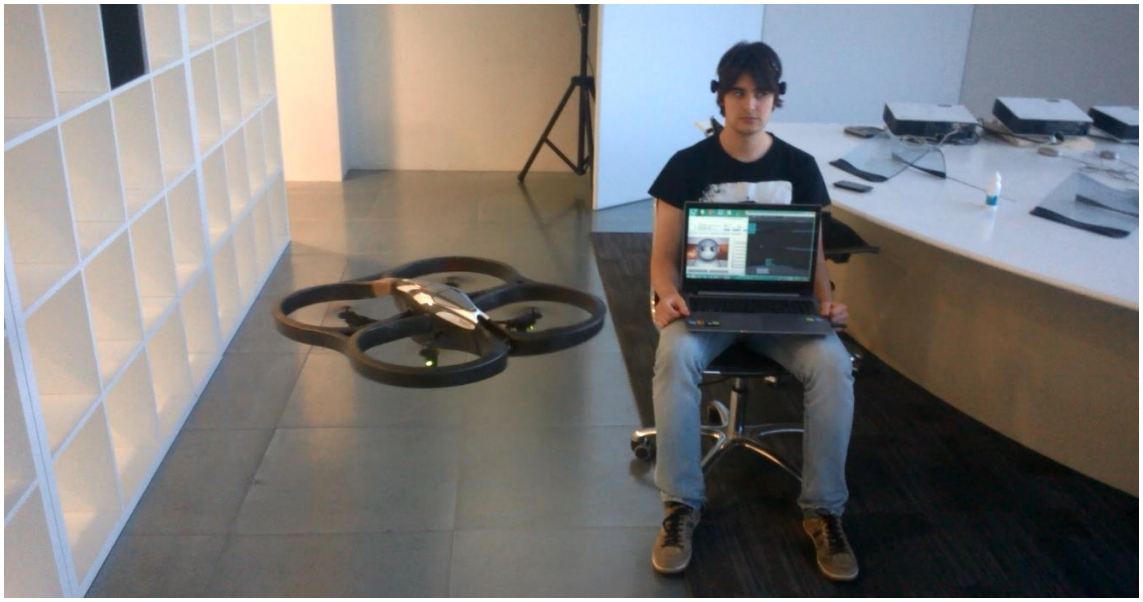


FIGURE 40: TESTING EXPRESSIV LOOKING LEFT

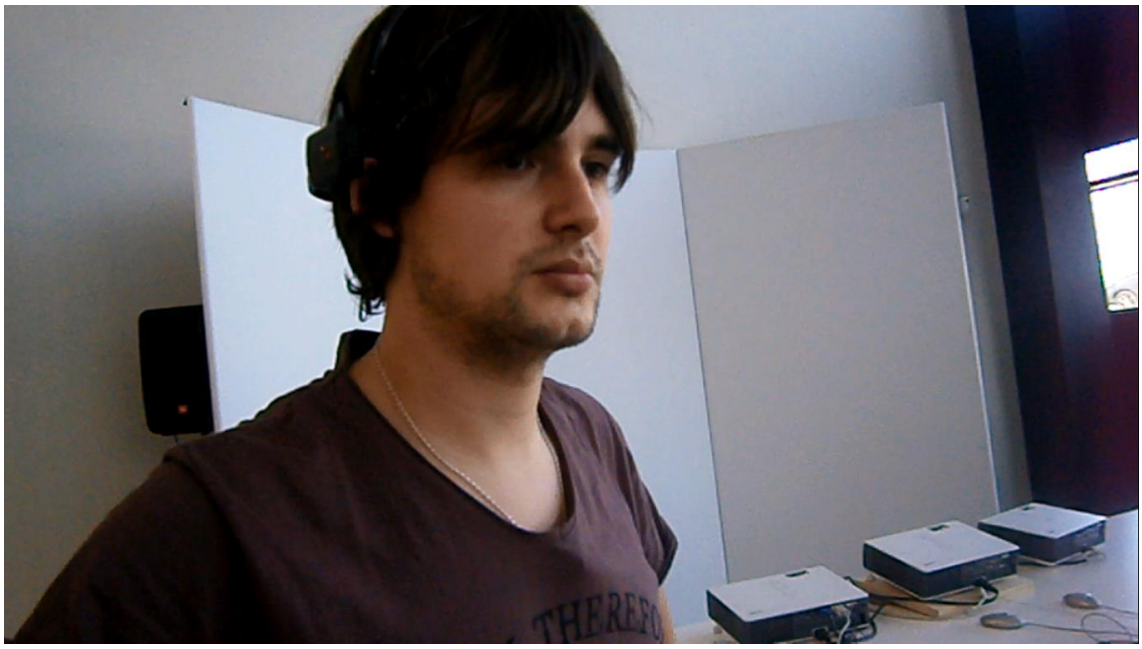


FIGURE 41: WEARING EMOTIV HEADSET WHILE TESTING

FIGURE 42: TESTING COGNITIV NEUTRAL

FIGURE 43: TESTING COGNITIV BACKWARDS

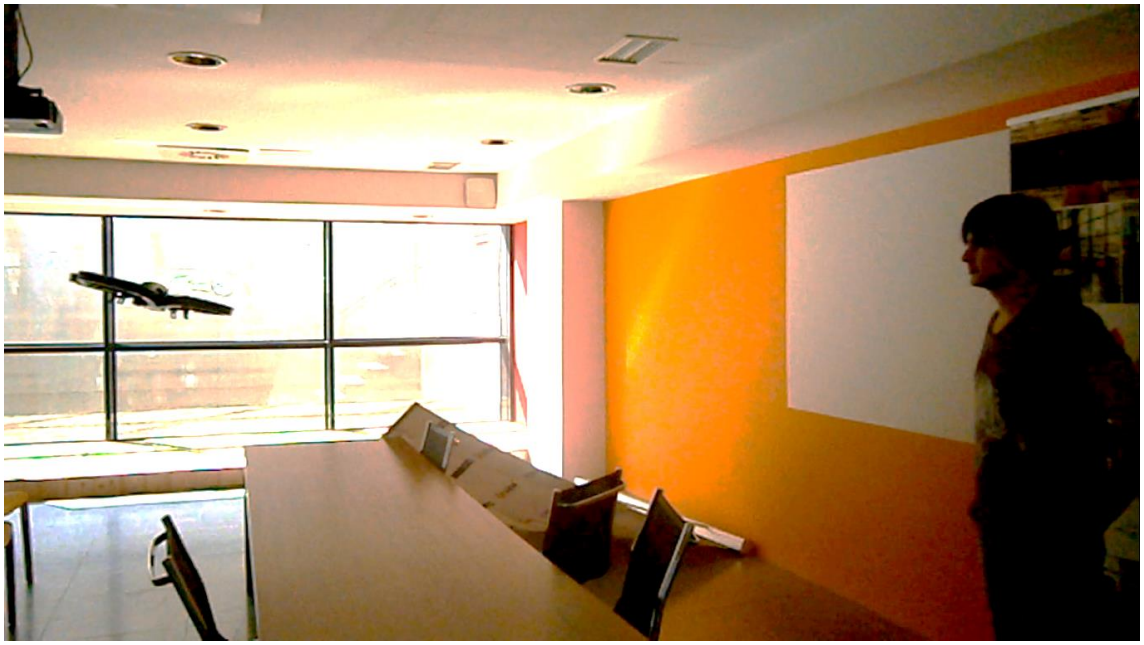


FIGURE 44: TESTING COGNITIV LEFT

Conclusions

This Project began with the idea of being a bridge between medics and engineering and the goals were placed in order to achieve that target. The project growth had an investigation and neuronal technology testing part, which has an important relevance when decided the material that has been used. Also there was a development part, first with a brainwave investigation interface (Matlab code) and then with the main project application (Unity code), which used the data obtained from headset and processed by *EmoEngine* drivers.

After that, the conclusion about using this technology is that, on the one hand, it can be a chance for medical uses and investigation even with the actual limits, but engineering requires a precision that nowadays is not available. This project, on the other hand, is orientated to investigation and improving for that technology, and the initial goals has been achieved as the application built connects the drone and the neuronal headset, received brainwave data, process the data and sends command controls to drone depending on what is being thought.

In conclusion, the project development, even with the problems found, went ahead and became into a Windows application and a Matlab investigation code. Also, it has the beginning code for an Android application development. The project finished but also opened the way to more neuronal applications and investigations, both in medics as in engineering.

Economy and time real costs

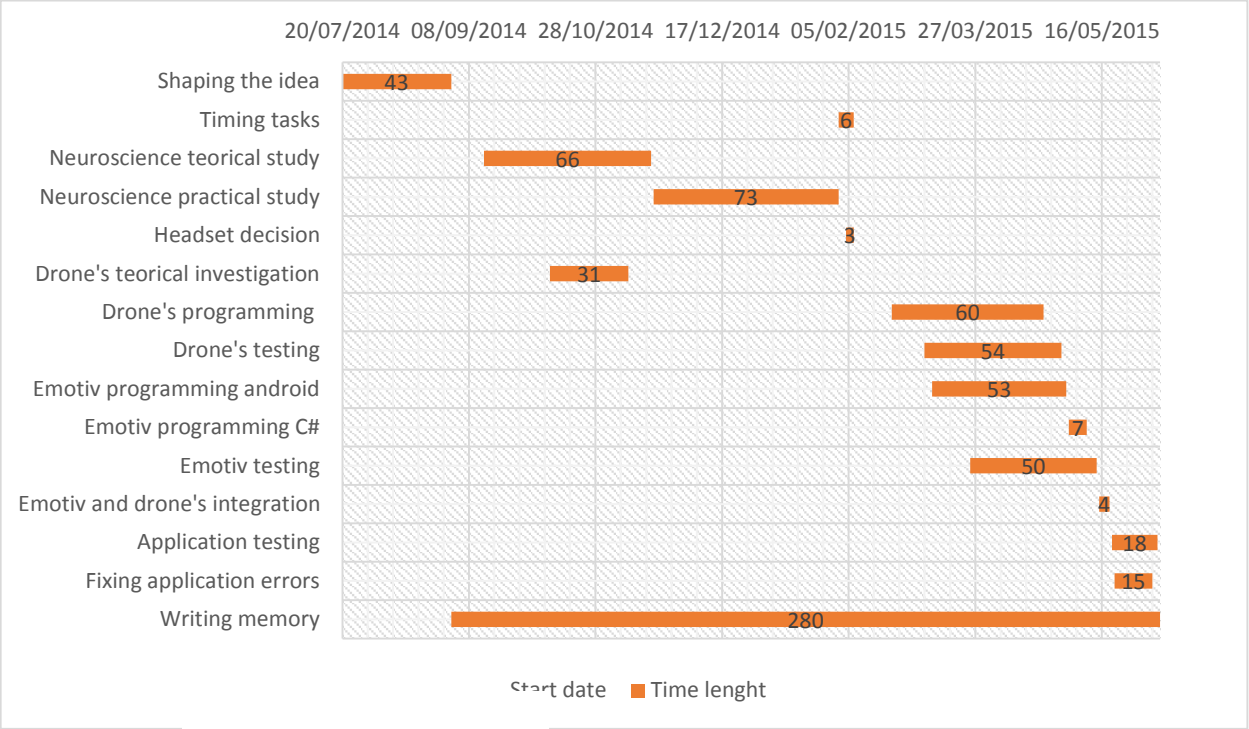


TABLE 4: FINAL GANTT TABLE

Budget estimation final	
Title	**Cost ($)**
Working time	45780
Emotiv headset	750
Drone	250
Laptop	750
Total	**47530**

TABLE 5: FINAL BUDGET ESTIMATION

Working time estimation		
Time (hours)	**Cost ($)**	**Total**
3052	15	45780

In the end, comparing the money and time estimation costs and the real ones that this project had had if it was a company project, it can be seen two different results.

First of all, time estimations were not as close as expected as the project needed more time and work when programming and fixing errors. Also, the investigation time was long, but not real because of the not working time through 2014. The real project began on 2015, January.

Finally, the budget estimation was closer than expected, only working time had an error, but it is not a problem in a real project as the money difference is not exceeding the error border, usually between 25% and 30%.

Improving ideas for upcoming projects

Finishing this project there were some improvement ideas that could be nice integrations for further upcoming projects, holding the idea of BCI and engineering management.

Adding the automatic user following: At the beginning of this project, it was said that another function to develop is the automatic following but, in the end, there was not enough time to do it and there will be a big problem with it. Integrating acceleration from drone to get the position is generating an error that gets bigger on time and useless for the precision required. The new headset *Insight* has a magnetometer, the same as the drone used, so using it can be an option and the improving could be implemented.

Adding the whole training part in the main application so the user do not need to use another external application to get a profile. This can be improved with some tutorials and instructions, getting an intuitive application that guides the user through the full mental training process.

Outdoor (or indoor) positioning has not been implemented because of time, but can be a technological improvement if it is aimed to military context, but Wi-Fi or Bluetooth beacons are required and that technology has not been tested in this project.

BCI project ideas

Continuing with investigation and health development there are some projects that can be done to contribute building this bridge between medics and engineering. The following ideas have to be done together with a specialist or requires some previously learning and investigation in neuroscience.

Robot arm control: Thought to improve one-handed or no-handed people life's level, an arm or hand control is a very difficult engineering project, but could create a big media impact if it is built with low cost neuronal headset. Some predefined robot movements could be thrown by brain control. This project integrates an engineering challenge as a robot programmer and investigation with mind control improvement.

Driving control system: Based on mind controlled wheelchair[8], the improvement of the application changing the laptop for a Smartphone can ease life adding comfort and reducing costs. The project has an electric part to power the motor and a program part to create the brain-computer interface between headset and Arduino[9].

[8] http://www.instructables.com/id/Brain-Controlled-Wheelchair/
[9] http://www.arduino.cc/

References

Emotiv Inc. (2014). *Emotiv EPOC User Manual.* https://*Emotiv*.zendesk.com/hc/en-us/article_attachments/200343895/*EPOC*UserManual2014.pdf

Emotiv Inc. (2014). *Emotiv Software Development Kit. User Manual for Release 2.0.0.20.* http://es.scribd.com/doc/232799269/SDK-UserManual-*Emotiv*Ltd-CE#scribd

Emotiv Inc: Technic support [online] support@*Emotiv*.zendesk.com
[consultation: 4th June 2015]

http://answers.unity3d.com/

http://ardrone2.parrot.com/

http://developer.*NeuroSky*.com/docs/doku.php?id=developer_tools_2.5_development_guide#thinkgear_sdk_for_net

http://docs.unity3d.com/Manual/index.html

http://eeg.dev.isib.be/2012/06/19/*Emotiv-EPOC*-eeg-raw-signals/#more-488

http://*Emotiv*.com/

http://*Emotiv*.com/forum/forum15/topic1371/messages/

http://*NeuroSky*.com/

http://stackoverflow.com/

http://store.*NeuroSky*.com/

http://wiki.*Emotiv*.com/

http://www.instructables.com/id/Brain-Controlled-Wheelchair/

http://www.lua.org/

http://www.mathworks.com/

http://www.mathworks.com/matlabcentral/fileexchange/36111-*Emotiv*eeg-headset-toolbox

http://www.parrot.com/es/

http://www.wordreference.com/es/

https://code.google.com/

https://*Emotiv*.com/blog/brain-controlled-wheelchair/

https://github.com/

https://projects.ardrone.org/boards/1/topics/show/4806

https://unity3d.com/es

Oweiss, K. G. (2010). *Statistical signal processing for neuroscience and neurotechnology*. Burlington, MA: Academic Press.

Parrot SA. (2013). *Parrot AR.Drone2.0 User guide.* http://parrotcontact.parrot.com/website/user-guides/download-user-guides.php?pdf=ar-drone-2/AR-Drone-2_User-guide_Android_UK.pdf

Piskorski, Stephane; Brulez, Nicolas; Eline, Pierre; D'Haeyer, Frederic. (2012). *AR.Drone Developer Guide.* http://www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf

Squire, L., Berg, D., Bloom, F. E., du Lac, S., Ghosh, A. & Spitzer, N.C. (2013). *Fundamental neuroscience.* California, San Diego: Academic Press.

Full Project videos on cloud:
https://drive.google.com/folderview?id=0B2nNZxsJTLBrfndmNG02XzN6dm9kdWpvV1JVdWJnZERhcXZrVVNNbThYbkZ2dWczQnVpRnM&usp=sharing

Annexes

Android Code

```
package com.example.bci;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

import com.example.Emotiv.*;
import com.example.freeflight.drone.DroneProxy;
import com.example.freeflight.drone.DroneProxy.EVideoRecorderCapability;
import com.example.freeflight.utils.DeviceCapabilitiesUtils;
import com.example.freeflight.receivers.WifiSignalStrengthChangedReceiver;

public class MainActivity extends Activity {

	private Led tgDevice;
	private DroneProxy droneProxy;
	boolean conected = false;
	TextView v1;
	Button takeOff;
	Button ConnectButton;
	Button MoveUp;
	private WifiSignalStrengthChangedReceiver Wifi;
	Intent intent;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		//final DroneServiceCommand droneServiceCommand = null;

		ConnectButton = (Button) this.findViewById(R.id.ConnectButton);
		OnClickListener connectListener = new View.OnClickListener(){
			Boolean conn = false;
			@Override
			public void onClick(View v) {
				// TODO Auto-generated method stub
				v1.setText("caca de vaca flaca D:");
				//droneProxy.triggerTakeOff();
				//if(Wifi.onReceive
(getApplicationContext(),intent) > 0)
				droneProxy =
DroneProxy.getInstance(getApplicationContext());
				 EVideoRecorderCapability recorderCapability =
DeviceCapabilitiesUtils.getMaxSupportedVideoRes();

				if (recorderCapability ==
EVideoRecorderCapability.NOT_SUPPORTED) {
					// Giving user a chance to record the video
					recorderCapability =
EVideoRecorderCapability.VIDEO_360P;
```

```
                        }

                        //Hay que cambiar el context, creo, pero
primero debes hacer el move up para comprobar que funciona.
                        droneProxy.doConnect(getApplicationContext(),
recorderCapability);
                }

        };


        //droneProxy = DroneProxy.getInstance(getApplicationContext());
        //droneProxy.doConnect(context, recorderCapability);
        v1 = (TextView)this.findViewById(R.id.tv1);
        takeOff = (Button) this.findViewById(R.id.TakeOff);
        OnClickListener takeOffListener = new View.OnClickListener(){
                boolean tOff = false;
                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        v1.setText("caca de vaca flaca D:");
                        droneProxy.triggerTakeOff();
                        if(tOff){
                                takeOff.setText("Land");
                        } else {
                                takeOff.setText("Take Off");
                        }
                }

        };

        MoveUp = (Button) this.findViewById(R.id.MoveUp);
        OnClickListener moveUpListener = new View.OnClickListener(){
                Boolean mUp = false;
                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        v1.setText("Hello World.");
                        //droneProxy
                }

        };


        MoveUp.setOnClickListener(moveUpListener);
        takeOff.setOnClickListener(takeOffListener);
        ConnectButton.setOnClickListener(connectListener);
    }



    public void Sky (){

        if(!conected){
                //tgDevice.connect(true);
        }
        else{
        }
    }
```

```
	void Update(){
		Sky();
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is
present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_settings) {
			return true;
		}
		return super.onOptionsItemSelected(item);
	}
}
```

Matlab Code (investigation)

```
function [AllOK output_matrix nS] = eeglogger(rectime,varargin)
%
%
% Enrique Fernández Serra, 15th March 2015.
%
% This function uses the Emotiv EPOC headset dll (Edk.dll) to acquire
the
% data contained in the EE_DataChannels_enum structure. The function
mimics
% the EEGLogger.exe function that can be compiled and used to acquire
the
% data using a C++ compiler.
% Additionally, it checks to make sure that the library hasn't been
loaded
% yet. The function requires no inputs (4 optional) and produces 3
outputs
%
% Optional Inputs
% rectime: this is time, in seconds, of the data buffer size, default
= 1
% acqtime: acquisition time, in seconds, default = 10 (for testing).
% lib_flag_popup: 1 = activates the libfunctionsview window, useful
for
% looking at all the functions that were loaded from the dll. default
= 0.
% plot_popup: 1 = plots the GyroX and GyroY signals after the data was
% recorded; 0 = no plot, default = 1.
%
% Outputs:
% AllOK: Everything worked fine in loading the library. If this is the
case
% a ZERO (0) should be returned.
% output_matrix: a 25 (=length(EE_DataChannels_enum) by n matrix where
n =
% sampling_frequency * acquisition time. The sampling_frequency, as
defined
% by Emotiv, is effectively 128 Hz.
% nS: provides you with the number of samples acquired (equivalent to
nSamplesTaken
% in the EEGLogger main.cpp function).

% data structures, copied and pasted from EPOCmfile.m
structs.InputSensorDescriptor_struct.members=struct('channelId',
'EE_InputChannels_enum', 'fExists', 'int32', 'pszLabel', 'cstring',
'xLoc', 'double', 'yLoc', 'double', 'zLoc', 'double');
enuminfo.EE_DataChannels_enum=struct('ED_COUNTER',0,'ED_INTERPOLATED',
1,'ED_RAW_CQ',2,'ED_AF3',3,'ED_F7',4,'ED_F3',5,'ED_FC5',6,'ED_T7',7,'E
D_P7',8,'ED_O1',9,'ED_O2',10,'ED_P8',11,'ED_T8',12,'ED_FC6',13,'ED_F4'
,14,'ED_F8',15,'ED_AF4',16,'ED_GYROX',17,'ED_GYROY',18,'ED_TIMESTAMP',
19,'ED_ES_TIMESTAMP',20,'ED_FUNC_ID',21,'ED_FUNC_VALUE',22,'ED_MARKER'
,23,'ED_SYNC_SIGNAL',24);
enuminfo.EE_CognitivTrainingControl_enum=struct('COG_NONE',0,'COG_STAR
T',1,'COG_ACCEPT',2,'COG_REJECT',3,'COG_ERASE',4,'COG_RESET',5);
enuminfo.EE_ExpressivAlgo_enum=struct('EXP_NEUTRAL',1,'EXP_BLINK',2,'E
XP_WINK_LEFT',4,'EXP_WINK_RIGHT',8,'EXP_HORIEYE',16,'EXP_EYEBROW',32,'
EXP_FURROW',64,'EXP_SMILE',128,'EXP_CLENCH',256,'EXP_LAUGH',512,'EXP_S
MIRK_LEFT',1024,'EXP_SMIRK_RIGHT',2048);
enuminfo.EE_ExpressivTrainingControl_enum=struct('EXP_NONE',0,'EXP_STA
RT',1,'EXP_ACCEPT',2,'EXP_REJECT',3,'EXP_ERASE',4,'EXP_RESET',5);
enuminfo.EE_ExpressivThreshold_enum=struct('EXP_SENSITIVITY',0);
```

```
enuminfo.EE_CognitivEvent_enum=struct('EE_CognitivNoEvent',0,'EE_Cogni
tivTrainingStarted',1,'EE_CognitivTrainingSucceeded',2,'EE_CognitivTra
iningFailed',3,'EE_CognitivTrainingCompleted',4,'EE_CognitivTrainingDa
taErased',5,'EE_CognitivTrainingRejected',6,'EE_CognitivTrainingReset'
,7,'EE_CognitivAutoSamplingNeutralCompleted',8,'EE_CognitivSignatureUp
dated',9);
enuminfo.EE_EmotivSuite_enum=struct('EE_EXPRESSIV',0,'EE_AFFECTIV',1,'
EE_COGNITIV',2);
enuminfo.EE_ExpressivEvent_enum=struct('EE_ExpressivNoEvent',0,'EE_Exp
ressivTrainingStarted',1,'EE_ExpressivTrainingSucceeded',2,'EE_Express
ivTrainingFailed',3,'EE_ExpressivTrainingCompleted',4,'EE_ExpressivTra
iningDataErased',5,'EE_ExpressivTrainingRejected',6,'EE_ExpressivTrain
ingReset',7);
enuminfo.EE_CognitivAction_enum=struct('COG_NEUTRAL',1,'COG_PUSH',2,'C
OG_PULL',4,'COG_LIFT',8,'COG_DROP',16,'COG_LEFT',32,'COG_RIGHT',64,'CO
G_ROTATE_LEFT',128,'COG_ROTATE_RIGHT',256,'COG_ROTATE_CLOCKWISE',512,'
COG_ROTATE_COUNTER_CLOCKWISE',1024,'COG_ROTATE_FORWARDS',2048,'COG_ROT
ATE_REVERSE',4096,'COG_DISAPPEAR',8192);
enuminfo.EE_InputChannels_enum=struct('EE_CHAN_CMS',0,'EE_CHAN_DRL',1,
'EE_CHAN_FP1',2,'EE_CHAN_AF3',3,'EE_CHAN_F7',4,'EE_CHAN_F3',5,'EE_CHAN
_FC5',6,'EE_CHAN_T7',7,'EE_CHAN_P7',8,'EE_CHAN_O1',9,'EE_CHAN_O2',10,'
EE_CHAN_P8',11,'EE_CHAN_T8',12,'EE_CHAN_FC6',13,'EE_CHAN_F4',14,'EE_CH
AN_F8',15,'EE_CHAN_AF4',16,'EE_CHAN_FP2',17);
enuminfo.EE_ExpressivSignature_enum=struct('EXP_SIG_UNIVERSAL',0,'EXP_
SIG_TRAINED',1);
enuminfo.EE_Event_enum=struct('EE_UnknownEvent',0,'EE_EmulatorError',1
,'EE_ReservedEvent',2,'EE_UserAdded',16,'EE_UserRemoved',32,'EE_EmoSta
teUpdated',64,'EE_ProfileEvent',128,'EE_CognitivEvent',256,'EE_Express
ivEvent',512,'EE_InternalStateChanged',1024,'EE_AllEvent',2032);

DataChannels = enuminfo.EE_DataChannels_enum;
DataChannelsNames =
{'ED_COUNTER','ED_INTERPOLATED','ED_RAW_CQ','ED_AF3','ED_F7','ED_F3','
ED_FC5','ED_T7','ED_P7','ED_O1','ED_O2','ED_P8','ED_T8','ED_FC6','ED_F
4','ED_F8','ED_AF4','ED_GYROX','ED_GYROY','ED_TIMESTAMP','ED_ES_TIMEST
AMP','ED_FUNC_ID','ED_FUNC_VALUE','ED_MARKER','ED_SYNC_SIGNAL'};



data_value(10000)=0;
optargin = size(varargin,2);
rectime = 1;
acqtime = 10;
lib_flag_popup = 1;
plot_popup = 1;

if optargin > 4
    error('Too many inputs');

elseif optargin == 4
    rectime = varargin{1};
    acqtime = varargin{2};
    lib_flag_popup = varargin{3};
    plot_popup = varargin{4};
elseif optargin == 3
    rectime = varargin{1};
    acqtime = varargin{2};
    lib_flag_popup = varargin{3};
elseif optargin == 2
    rectime = varargin{1};
    acqtime = varargin{2};
```

```
elseif optargin == 1
    rectime = varargin{1};
end
% Check to see if library was already loaded
if ~libisloaded('Edk')
    [nf, w] = loadlibrary('Edk','Edk', 'addheader', 'EmoStateDLL',
'addheader', 'EdkErrorCode');
    disp(['EDK library loaded']);
    if( lib_flag_popup )
        libfunctionsview('Edk')
        nf % these should be empty if all went well
        w
    end
else
    disp(['EDK library already loaded']);
end
sampFreq = 128;
default = int8(['Emotiv Systems-5' 0]);
AllOK = calllib('Edk','EE_EngineConnect', 'Emotiv Systems-5'); %
success means this value is 0

hData = calllib('Edk','EE_DataCreate');
calllib('Edk','EE_DataSetBufferSizeInSec',rectime);
eEvent = calllib('Edk','EE_EmoEngineEventCreate');
readytocollect = false;
cnt = 0;


% initialize outputs:
output_matrix = zeros(acqtime*sampFreq,length(DataChannelsNames));
nS = zeros(acqtime*sampFreq,1);

% For this next part, see the eeglogger main.cpp file for a better
% understanding of what's happening here.

tic
while(toc < acqtime)
    state = calllib('Edk','EE_EngineGetNextEvent',eEvent); % state = 0
if everything's OK
    eventType = calllib('Edk','EE_EmoEngineEventGetType',eEvent);
    %disp(eventType);
    userID=libpointer('uint32Ptr',0);
    calllib('Edk','EE_EmoEngineEventGetUserId',eEvent, userID);


    if strcmp(eventType,'EE_UserAdded') == true
        User_added = 1;
        userID_value = get(userID,'value');
        calllib('Edk','EE_DataAcquisitionEnable',userID_value,true);
        readytocollect = true;
    end

    if (readytocollect)

        calllib('Edk','EE_DataUpdateHandle', 0, hData);
        nSamples = libpointer('uint32Ptr',0);
        calllib('Edk','EE_DataGetNumberOfSample',hData,nSamples);
        nSamplesTaken = get(nSamples,'value') ;
        if (nSamplesTaken ~= 0)
            data = libpointer('doublePtr',zeros(1,nSamplesTaken));
```

```
   %          for sampleIdx=1:nSamplesTaken
                for i =
1:length(fieldnames(enuminfo.EE_DataChannels_enum))
                    calllib('Edk','EE_DataGet',hData,
DataChannels.([DataChannelsNames{i}]), data, uint32(nSamplesTaken));
                    data_value = get(data,'value');
%                    output_matrix(cnt+1,i) = data_value(sampleIdx);
                    output_matrix(cnt+1:cnt+length(data_value),i) =
data_value;
                    disp(data_value);
                end
                %for i =
1:length(fieldnames(enuminfo.EE_DataChannels_enum))
                 %   calllib('Edk','EE_DataGet',hData,
DataChannels.([DataChannelsNames{i}]), data, uint32(nSamplesTaken));
                  %  eeg_value = get(data,'value');
                   % eeg_matrix(cnt+1:cnt+length(eeg_value),i) =
eeg_value;
                    %disp(eeg_value);
                %end

                nS(cnt+1) = nSamplesTaken;
%               cnt = cnt + 1;
                cnt = cnt + length(data_value);
%             end
        end
    end
    %    pause(0.1);    % haven't played with this much...

end
% extract sampling rate (should be 128)
sampRateOutPtr = libpointer('uint32Ptr',0);
calllib('Edk','EE_DataGetSamplingRate',0,sampRateOutPtr);
sampFreqOut = get(sampRateOutPtr,'value') % in Hz

calllib('Edk','EE_DataFree',hData);
end_time = find(output_matrix(:,20)==0,1) - 1;
if plot_popup
    figure;
    plot([0:1/sampFreq:(end_time-
1)/sampFreq],output_matrix(1:end_time,18))
    hold on
    plot([0:1/sampFreq:(end_time-
1)/sampFreq],output_matrix(1:end_time,19),'r')
    xlabel('time (s)')
    title('GyroX GyroZ values') % I call it gyroZ instead of gyroY ...
                                % because if I move the headset up and
...
                                % down it tracks.
    legend('GyroX','GyroZ');
    figure;
    plot3([0:1/sampFreq:(end_time-
1)/sampFreq],output_matrix(1:end_time,18),output_matrix(1:end_time,19)
)
    xlabel('time (s)')
    ylabel('X coord')
    zlabel('Y coord')
end

title('EEG');
figure;
```

```
for i=4:17
plot([0:1/sampFreq:(end_time-1)/sampFreq],output_matrix(1:end_time,i))
hold on
end

calllib('Edk','EE_EngineDisconnect');
calllib('Edk','EE_EmoEngineEventFree',eEvent);

% unloadlibrary('Edk'); % unload the library after having turned off
% [int32, uint32Ptr] EE_DataGetSamplingRate (uint32, uint32Ptr)
% int32 EE_DataSetSychronizationSignal (uint32, int32)
% [int32, string] EE_EnableDiagnostics (cstring, int32, int32)
```

Unity Code (C#)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System;
using System.Runtime.InteropServices;

using AR.Drone.Client;
using AR.Drone.Video;
using AR.Drone.Data;
using AR.Drone.Data.Navigation;
using FFmpeg.AutoGen;
using XInputDotNetPure;
using NativeWifi;

using Emotiv;
//using DotNet;


public class DroneController : MonoBehaviour {
	// Stick which is moved analogical to the movement of the gamepad stick
	//public Transform Stick;
	// Modifies the stick rotation
	// public float StickRotationModifier = 0.15f;
	// Plane on which the main camera is mapped
	public Renderer MainRenderer;
	// Plane on which the secondary camera is mapped
	public Renderer SecondaryRenderer;
	// Rotation limit for the switch between the main camera and the secondary camera
	public float SwitchRotation = 0.4f;
	// The camera used for the switch test
	public Transform CameraForSwitchCheck;
	// Ambient Light
	public Light[] AmbientLights;
```

```
// Status text
public TextMesh StatusText;
// Wifi status
public TextMesh WifiText;
public TextMesh WifiChart;
public int maxChartBars = 20;

// Gamepad variables
/*private bool playerIndexSet = false;
private XInputDotNetPure.PlayerIndex playerIndex;
private GamePadState state;
private GamePadState prevState;*/

// Indicates that the drone is landed
private bool isLanded = true;
// Indicates that the startButton is Pressed
private bool startButtonPressed = false;
// Texture used for the camera content
private Texture2D cameraTexture;
// A black texture used for the inactive plane
private Texture2D blackTexture;
// byte array which will be filled by the camera data
private byte[] data;
// Drone variables
private VideoPacketDecoderWorker videoPacketDecoderWorker;
private DroneClient droneClient;
private NavigationData navigationData;

//Emotiv variables
protected EmoEngine engine; // Access to the EDK is viaa the EmoEngine
private uint userID; // userID is used to uniquely identify a user's headset
private EmoState cogState = null;
private Dictionary<EdkDll.EE_DataChannel_t, double[]> emoData;
private bool EmoCon = false;
private bool isFlying = false;
```

```
public Profile prof;
private   string mStr;
private string debugProfileDir;
public Profile profileID;
private Profile userProfiles;
//public Byte[] pro;

//Expressiv
//public var clenchExtent = e.emoState.ExpressivGetClenchExtent();
//var eyebrowExtent = e.emoState.ExpressivGetEyebrowExtent();
//float leftEyelid, rightEyelid;
//e.emoState.ExpressivGetEyelidState(out leftEyelid, out rightEyelid);
//float leftEyeLocation, rightEyeLocation;
//e.emoState.ExpressivGetEyeLocation(out leftEyeLocation, out rightEyeLocation);
//var lowerFaceAction = e.emoState.ExpressivGetLowerFaceAction();
//var lowerFaceActionPower = e.emoState.ExpressivGetLowerFaceActionPower();
//var smileExtent = e.emoState.ExpressivGetSmileExtent();
//var upperFaceAction = e.emoState.ExpressivGetUpperFaceAction();
//var upperFaceActionPower = e.emoState.ExpressivGetUpperFaceActionPower();
public bool isBlinking;
public bool isEyesOpen;
public bool isLeftWink;
public bool isRightWink;
public bool isLookingDown;
public bool isLookingLeft;
public bool isLookingRight;
public bool isLookingUp;

//Affectiv
public float excitementShortTermScore;
public float excitementLongTermScore;
public float engagementBoredomScore;
public float frustrationScore;
public float meditationScore;
```

```
	public int incomingPower = 1;
	//public float modifier = 0.1f;
	public string debugKey;

	public GUITexture ConnectButton;
	public Texture2D connectTexture;
	public Texture2D disconnectTexture;

	//public GUILayer test;
	bool connectPressed = false;

	private static float bufferSize = 1.0f;

	private float elapsedTime = 0;

	private static DroneController instance;
	public static DroneController Instance
	{
		get
		{
			if (instance == null) {
				instance = new GameObject("EmotivHandler").AddComponent<DroneController>();
			}

			return instance;
		}
	}



	// Width and height of the camera
	private int width = 640;
	private int height = 360;
```

```
// wlanclient for signal strength
private WlanClient client;

// Use this for initialization
void Start () {
        Debug.Log("Start");
        // initialize data array
        data = new byte[width*height*3];
        //profileID = (Profile)pro;
        // set textures
        MainRenderer.material.mainTexture = cameraTexture;
        SecondaryRenderer.material.mainTexture = blackTexture;
        cameraTexture = new Texture2D (width, height);
        blackTexture = new Texture2D (1, 1);
        blackTexture.SetPixel (0, 0, Color.black);
        blackTexture.Apply ();

        // Initialize drone
        videoPacketDecoderWorker = new VideoPacketDecoderWorker(PixelFormat.BGR24,
true, OnVideoPacketDecoded);
        videoPacketDecoderWorker.Start();
        droneClient = new DroneClient("192.168.1.1");
        droneClient.UnhandledException += HandleUnhandledException;
        droneClient.VideoPacketAcquired += OnVideoPacketAcquired;
        droneClient.NavigationDataAcquired += navData => navigationData = navData;
        videoPacketDecoderWorker.UnhandledException += HandleUnhandledException;
        droneClient.Start ();

        //ConnectButton = GameObject.Find("ConnectButton");

        //var xPosition1 = (Screen.width);

        //ConnectButton.texture = connectTexture;
        //ConnectButton.pixelInset = new Rect((20),20,50,50);
```

```
        // activate main drone camera
        switchDroneCamera (AR.Drone.Client.Configuration.VideoChannelType.Vertical);

        // determine connection
        client = new WlanClient();

        mStr = System.IO.Directory.GetCurrentDirectory();
        debugProfileDir = "C:/Users/Quique/Desktop/quique.emu";
        //debugProfileDir.Replace (@"\",@"/");
                //"C://Users/Quique/Desktop/Lord.emu";
}

// Update is called once per frame
void Update () {

        var yaw = 0;
        var roll = 0;
        var pitch = 0;
        var gaz = 0;

        if(Input.GetMouseButtonDown(0) && ConnectButton.HitTest(Input.mousePosition)){

                if(!connectPressed){
                        //ConnectButton.pixelInset.x = 155;
                        EmoConnect();
                        ConnectButton.texture = disconnectTexture;
                        connectPressed = true;
                }
                else{
                        EmoDisconnect();
                        ConnectButton.texture = connectTexture;
                        connectPressed = false;
                }
```

```
}


convertCameraData ();

//updateGamepadState ();
//getRawData ();
//moveStick ();
if (EmoCon) {

        if (engine == null)
                return;
        else {
                // Handle any waiting events
                engine.ProcessEvents ();
                // This should be called every second...
                elapsedTime += Time.deltaTime;
                if (elapsedTime > bufferSize) {
                        emoData = engine.GetData (userID);
                        elapsedTime = 0;

                        if (data == null)
                                return;
                }
        }
}
        // Start or land the drone

                if (Input.GetKeyDown ("space")) {
                Debug.Log("Trying to take off");
                        if (isLanded){
                                Debug.Log ("Flying");
                                droneClient.Takeoff ();
                                        isFlying = true;
                        }
```

```
                    else{
                        Debug.Log ("Landing");
                        droneClient.Land ();
                        isFlying = false;
                    }
                    isLanded = !isLanded;
                    startButtonPressed = true;
                }
                else{
                    startButtonPressed = false;
                }



        // exit application
        if (Input.GetKey ("escape")) {
            droneClient.Land();
            droneClient.Stop();
            droneClient.Dispose ();
            videoPacketDecoderWorker.Stop ();
            videoPacketDecoderWorker.Dispose();
            Application.Quit ();
        }

        // Move the drone with Emotiv

        if (isConnected ()&& isFlying) {
            EmoState emoState = getCognitiveState ();
            if(emoState != null && emoState.CognitivGetCurrentAction() ==
EdkDll.EE_CognitivAction_t.COG_NEUTRAL){
                yaw = 0;
                pitch = 0;
                roll = 0;
                gaz = 0;
```

```
		}
	}

	// Move Right
	if (isConnected() && isFlying) {
		EmoState emoState = getCognitiveState();

		if (emoState != null && emoState.CognitivGetCurrentAction() == EdkDll.EE_CognitivAction_t.COG_RIGHT || Input.GetKey (KeyCode.D) || isRightWink) {
			Debug.Log ("Derecha");
			roll = 1;
			//roll = (int)(emoState.CognitivGetCurrentActionPower());
		}
	}

	// Move Left
	if (isConnected() && isFlying) {
		EmoState emoState = getCognitiveState();

		if ( emoState != null && emoState.CognitivGetCurrentAction() == EdkDll.EE_CognitivAction_t.COG_LEFT || Input.GetKey (KeyCode.A) || isLeftWink) {
			Debug.Log ("Izquierda");
			roll = -1;
			//roll = (int)(emoState.CognitivGetCurrentActionPower());
		}
	}

	// Move Forward
	if (isConnected()&& isFlying) {
		EmoState emoState = getCognitiveState();
		bool forw = false;

		if ( emoState != null && emoState.CognitivGetCurrentAction() == EdkDll.EE_CognitivAction_t.COG_PUSH || Input.GetKey (KeyCode.W) || excitementShortTermScore > 0.40 && !isEyesOpen) {
			Debug.Log ("Adelante");
```

```
                pitch = -1;
                //pitch = (int)(emoState.CognitivGetCurrentActionPower());
            }
        }

        // Move Backwards
        if (isConnected()&& isFlying) {
            EmoState emoState = getCognitiveState();
            if ( emoState != null && emoState.CognitivGetCurrentAction() == EdkDll.EE_CognitivAction_t.COG_PULL || Input.GetKey (KeyCode.S) || excitementShortTermScore < 0.4 && !isEyesOpen) {
                Debug.Log ("Atras");
                pitch = 1;
                //pitch = (int)(emoState.CognitivGetCurrentActionPower());
            }
        }

        // Turn Arround with your eyes
        if (isConnected()&& isFlying) {
            if (isLookingRight == true || Input.GetKey(KeyCode.E)) {
                Debug.Log ("Girando a la derecha");
                //yaw = (int)(emoState.CognitivGetCurrentActionPower());
                yaw = 1;
            }
        }

        if (isConnected()&& isFlying) {
            if (isLookingLeft == true || Input.GetKey(KeyCode.Q)) {
                Debug.Log ("Girando a la izquierda");
                //yaw = -(int)(emoState.CognitivGetCurrentActionPower());
                yaw = -1;
            }
        }

        // Moving up and down
```

```
		if (isConnected()&& isFlying) {
			if (isLookingDown == true || Input.GetKey(KeyCode.X)) {
				Debug.Log ("Bajando");
				//yaw = (int)(emoState.CognitivGetCurrentActionPower());
				gaz = -1;
			}
		}

		if (isConnected()&& isFlying) {
			if (isLookingUp == true || Input.GetKey(KeyCode.Z)) {
				Debug.Log ("Subiendo");
				//yaw = -(int)(emoState.CognitivGetCurrentActionPower());
				gaz = 1;
			}
		}



		droneClient.Progress(AR.Drone.Client.Command.FlightMode.Progressive,  pitch:  pitch,
roll: roll, gaz: gaz, yaw: yaw);

		// Switch drone camera

			if (MainRenderer.material.mainTexture != cameraTexture) {
				SecondaryRenderer.material.mainTexture = blackTexture;
				MainRenderer.material.mainTexture = cameraTexture;
				switchDroneCamera
(AR.Drone.Client.Configuration.VideoChannelType.Horizontal);
			}



		// set status text
		if (navigationData != null) {
			StatusText.text = string.Format("Battery: {0} % \nYaw: {1:f} \nPitch: {2:f} \nRoll:
{3:f} \nAltitude: {4} m",
```

```
                                        navigationData.Battery.Percentage,navigationData.Yaw,
navigationData.Pitch,
                                        navigationData.Roll,navigationData.Altitude);
                }

                // determine wifi strength
                determineWifiStrength ();
        }

        // Called if the gameobject is destroyed
        void OnDestroy(){
                droneClient.Land();
                droneClient.Stop();
                droneClient.Dispose ();
                videoPacketDecoderWorker.Stop ();
                videoPacketDecoderWorker.Dispose();
        }

        /// <summary>
        /// Log the unhandled exception.
        /// </summary>
        /// <param name="arg1">Arg1.</param>
        /// <param name="arg2">Arg2.</param>
        void HandleUnhandledException (object arg1, System.Exception arg2)
        {
                Debug.Log (arg2);
        }

        /// <summary>
        /// Switchs the drone camera.
        /// </summary>
        /// <param name="Type">Video channel type.</param>
        private void switchDroneCamera(AR.Drone.Client.Configuration.VideoChannelType Type){
                var configuration = new AR.Drone.Client.Configuration.Settings();
                configuration.Video.Channel = Type;
```

```
            droneClient.Send(configuration);
        }

        /// <summary>
        /// Converts the camera data to a color array and applies it to the texture.
        /// </summary>
        private void convertCameraData(){
            int r = 0;
            int g = 0;
            int b = 0;
            int total = 0;
            var colorArray = new Color32[data.Length/3];
            for(var i = 0; i < data.Length; i+=3)
            {
                colorArray[i/3] = new Color32(data[i + 2], data[i + 1], data[i + 0], 1);
                r += data[i + 2];
                g += data[i + 1];
                b += data[i + 0];
                total++;
            }
            r /= total;
            g /= total;
            b /= total;
            cameraTexture.SetPixels32(colorArray);
            cameraTexture.Apply();
            foreach (Light light in AmbientLights)
                light.color = new Color32 (System.Convert.ToByte(r),
System.Convert.ToByte(g), System.Convert.ToByte(b), 1);
        }

        public static EmoEngine getEmoEngine(){
            if (instance.engine == null) return EmoEngine.Instance;
            else return instance.engine;

        }
```

```
	public bool isConnected() {
		return (engine != null);
	}

	public Dictionary<EdkDll.EE_DataChannel_t, double[]> getRawData() {
		//Debug.Log (emoData);
		return emoData;
	}

	public double[] getDataChannel(EdkDll.EE_DataChannel_t channel) {
		return emoData[channel];
	}

	//**********************************************************

	/// <summary>
	/// Function to save byte array to a file
	/// </summary>
	/// <param name="_FileName">File name to save byte array</param>
	/// <param name="_ByteArray">Byte array to save to external file</param>
	/// <returns>Return true if byte array save successfully, if not return false</returns>
	public bool ByteArrayToFile(string _FileName, byte[] _ByteArray)
	{
		try
		{
			System.IO.FileStream _FileStream = new System.IO.FileStream(_FileName,
System.IO.FileMode.Create, System.IO.FileAccess.Write);
			_FileStream.Write(_ByteArray, 0, _ByteArray.Length);
			_FileStream.Close();
			return true;
		}catch (Exception _Exception){
			Console.WriteLine("Exception caught in process: {0}",
_Exception.ToString());}return false;
```

```
}
/// <summary>
/// Save all profiles in arraylist to file
/// </summary>
public void SaveProfilesToFile()
{
	Debug.Log("Save file");
	//---------------------
	string mStr = System.IO.Directory.GetCurrentDirectory();
	if (!System.IO.Directory.Exists(mStr + @"/EmotivUserProfile"))
	{
		System.IO.Directory.CreateDirectory(mStr + @"/EmotivUserProfile");
	}
	System.IO.Directory.SetCurrentDirectory(mStr + @"/EmotivUserProfile");
	for (int i = 0; i < userProfiles.Count; i++)
	{
		UserProfile tmp = (UserProfile)userProfiles[i];
		//UserProfile tmp = (UserProfile)userProfiles;
	UserProfile tmp = null;
		ByteArrayToFile(tmp.profileName + ".up", tmp.profile.GetBytes());
	}
	System.IO.Directory.SetCurrentDirectory(mStr);
}
/// <summary>
/// Get Profile List
/// </summary>
/// <returns></returns>
public static string[] GetProfileList()
{
	string mStr = System.IO.Directory.GetCurrentDirectory();
	if (!System.IO.Directory.Exists(mStr + @"/EmotivUserProfile"))
	{
		System.IO.Directory.CreateDirectory(mStr + @"/EmotivUserProfile");
	}
	mStr += @"/EmotivUserProfile";
```

```
            string[] strArrFiles = System.IO.Directory.GetFiles(mStr, "*.up");
            for (int i = 0; i < strArrFiles.Length; i++)
            {
                strArrFiles[i] = strArrFiles[i].Substring(0, strArrFiles[i].Length - 3);
                strArrFiles[i] = Path.GetFileName(strArrFiles[i]);
            }
            return strArrFiles;
        }
        /// <summary>
        /// Load all profile file in EmotivUserProfile folder into arraylist
        /// </summary>
        public void LoadProfilesFromFile()
        {
            Debug.Log("Load file");
            //----------------------
            string mStr = System.IO.Directory.GetCurrentDirectory();
            if (!System.IO.Directory.Exists(mStr + @"/EmotivUserProfile"))
            {
                System.IO.Directory.CreateDirectory(mStr + @"/EmotivUserProfile");
            }
            mStr += @"/EmotivUserProfile";
            string[] strArrFiles = System.IO.Directory.GetFiles(mStr, "*.up");
            for (int i = 0; i < strArrFiles.Length; i++)
            {
                System.IO.FileStream _FileStream = new System.IO.FileStream(strArrFiles[i],
System.IO.FileMode.Open, System.IO.FileAccess.Read);
                Byte[] buffer = new Byte[_FileStream.Length];
                _FileStream.Read(buffer, 0,(int)_FileStream.Length);
                _FileStream.Close();
                engine.SetUserProfile((uint)userID, buffer);
                UserProfile tmp = new UserProfile();
                tmp.profileName = strArrFiles[i].Substring(0, strArrFiles[i].Length -3);
                tmp.profileName = Path.GetFileName(tmp.profileName);
                tmp.profile = engine.GetUserProfile((uint)userID);
                //userProfiles.Add(tmp);
```

```
            }
}
/// <summary>
/// Set user profile , remember to save current profile before or lose it
/// </summary>
/// <param name="prName">Profile Name</param>
/// <returns>Return false if no profile with this name</returns>
public Boolean SetUserProfile(string prName)
{
            int i = FindProfileName(prName);
            int i = 1;
            if (i != userProfiles.Count)
            if (i != 1)
            {
                        UserProfile tmp = (UserProfile)userProfiles[i];
                        UserProfile tmp = null;
                        engine.SetUserProfile((uint)userID, tmp.profile);
                        currentIndex = i;
                        Debug.Log(prName);
                        EmoCognitiv.cognitivActionsEnabled = CheckCurrentProfile();
                        return true;
            }
            else
            {
                        Debug.Log("Set user profile failed");
                        return false;
            }// have no profile with this name
}
//*************************************************************



void engine_EmoEngineConnected(object sender, EmoEngineEventArgs e) {

            Debug.Log("EmoEngine Connected!");
```

```
userID = e.userId;
Debug.Log (userID);
//uint user = System.Convert.ToUInt32 (userID);
//byte prof = new byte[0];
//LoadProfilesFromFile ();
//EmoEngine.Instance.LoadUserProfile(0, debugProfileDir);
//engine.SetUserProfile(userID,);
userID = 0;
engine.DataAcquisitionEnable(userID, true);
engine.EE_DataSetBufferSizeInSec(1);

// Load the user profile with my training data.
EdkDll userEDK;
//userEDK.EE_LoadUserProfile(userID, "quique.emu");
//Profile usuario;
//usuario = userEDK.EE_GetUserProfile (userID);
//userEDK.EE_SetUserProfile (userID, usuario);
//engine.LoadUserProfile (userID, "quique.emu");

EdkDll.EE_LoadUserProfile(userID, "quique.emu");
Profile usuario;
//IntPtr ptr = ;
//usuario = EdkDll.EE_GetUserProfile (userID, IntPtr.Zero);
//EdkDll.EE_SetUserProfile (userID, usuario);

//engine.LoadUserProfile(userID,"quique.emu");
Debug.Log ("caca");
//EdkDll.EE_SetUserProfile ();
//EdkDll.EE_SaveUserProfile (userID, "caca.emu");
//engine.LoadUserProfile (userID, "caca.emu");
prof = engine.GetUserProfile((uint)userID);
engine.SetUserProfile(userID, prof);

//userID = e.userId;
//engine.LoadUserProfile (userID, @"C:\Users\Quique\Desktop\quique.emu");
```

```
            Debug.Log ("User ID: " + userID);
            EmoCon = true;
        }

        void engine_CognitiveEmoStateUpdated(object sender, EmoStateUpdatedEventArgs args) {
            cogState = args.emoState;
            EmoState emoState = args.emoState;
            //Debug.Log (cogState);

            Debug.Log("User has cognitive action : " +
emoState.CognitivGetCurrentAction().ToString() + " of strength " +
emoState.CognitivGetCurrentActionPower().ToString() );
        }

        void engine_ExpressivEmoStateUpdated(object sender, EmoStateUpdatedEventArgs e)
        {
            isBlinking = e.emoState.ExpressivIsBlink();
            isEyesOpen = e.emoState.ExpressivIsEyesOpen();
            isLeftWink = e.emoState.ExpressivIsLeftWink();
            isRightWink = e.emoState.ExpressivIsRightWink();
            isLookingDown = e.emoState.ExpressivIsLookingDown();
            isLookingLeft = e.emoState.ExpressivIsLookingLeft();
            isLookingRight = e.emoState.ExpressivIsLookingRight();
            isLookingUp = e.emoState.ExpressivIsLookingUp();
            //Debug.Log ("User is blinking?: " + e.emoState.ExpressivIsBlink());
            //Debug.Log ("User is looking left?: " + e.emoState.ExpressivIsLookingLeft());
        }

        public void engine_AffectivEmoStateUpdated(object sender, EmoStateUpdatedEventArgs e){
            excitementShortTermScore = e.emoState.AffectivGetExcitementShortTermScore();
            excitementLongTermScore = e.emoState.AffectivGetExcitementLongTermScore();
            engagementBoredomScore = e.emoState.AffectivGetEngagementBoredomScore();
            frustrationScore = e.emoState.AffectivGetFrustrationScore();
            meditationScore = e.emoState.AffectivGetMeditationScore();
            Debug.Log ("User actual excitement is: " + excitementShortTermScore);
```

```
        }

        public void EmoConnect (){

                engine = EmoEngine.Instance;

                engine.EmoEngineConnected += new
EmoEngine.EmoEngineConnectedEventHandler(engine_EmoEngineConnected);

                //To record affectiv suite data
                engine.AffectivEmoStateUpdated += new
EmoEngine.AffectivEmoStateUpdatedEventHandler(engine_AffectivEmoStateUpdated);

                //for response to expressiv motions
                engine.ExpressivEmoStateUpdated += new
EmoEngine.ExpressivEmoStateUpdatedEventHandler(engine_ExpressivEmoStateUpdated);

                //to get Cognitive state
                engine.CognitivEmoStateUpdated += new
EmoEngine.CognitivEmoStateUpdatedEventHandler(engine_CognitiveEmoStateUpdated);

                engine.Connect ();
                //engine.RemoteConnect("127.0.0.1",3008);
                //EmoCon = true;
        }

        public void EmoDisconnect(){
                try {
                        engine.Disconnect();
                        EmoCon = false;
                } catch {}

                engine = null;

        }
```

```
        public EmoState getCognitiveState() {
                return cogState;
        }

        /// <summary>
        /// Updates the state of the gamepad.
        /// </summary>
        /*
        void updateGamepadState(){
                // Find a PlayerIndex, for a single player game
                // Will find the first controller that is connected and use it
                if (!playerIndexSet || !prevState.IsConnected)
                {
                        for (int i = 0; i < 4; ++i)
                        {
                                PlayerIndex testPlayerIndex = (PlayerIndex)i;
                                GamePadState testState = GamePad.GetState(testPlayerIndex);
                                if (testState.IsConnected)
                                {
                                        Debug.Log(string.Format("GamePad found {0}",
testPlayerIndex));
                                        playerIndex = testPlayerIndex;
                                        playerIndexSet = true;
                                }
                        }
                }

                prevState = state;
                state = GamePad.GetState(playerIndex);
        }
        */
        /// <summary>
        /// Determines what happens if a video packet is acquired.
        /// </summary>
        /// <param name="packet">Packet.</param>
```

```
private void OnVideoPacketAcquired(VideoPacket packet)
{
	if (videoPacketDecoderWorker.IsAlive)
		videoPacketDecoderWorker.EnqueuePacket(packet);
}

/// <summary>
/// Determines what happens if a video packet is decoded.
/// </summary>
/// <param name="frame">Frame.</param>
private void OnVideoPacketDecoded(VideoFrame frame)
{
	data = frame.Data;
}

/// <summary>
/// Sets the rotation of the stick
/// </summary>
/*
private void moveStick(){
	var newRotation = Stick.rotation;
	newRotation.x = StickRotationModifier* state.ThumbSticks.Left.Y;
	newRotation.z =  -StickRotationModifier * state.ThumbSticks.Left.X;
	Stick.rotation = newRotation;
}
*/
/// <summary>
/// Determine the wifi strength.
/// </summary>
private void determineWifiStrength(){
	int signalQuality = 0;
	foreach (WlanClient.WlanInterface wlanIface in client.Interfaces)
	{
		try {
```

```
                                        signalQuality                                                   =
(int)wlanIface.CurrentConnection.wlanAssociationAttributes.wlanSignalQuality;
                                }
                                catch (System.Exception e ){
                                        Debug.Log ("No Wifi Connection");
                                }
                        }

                        if (signalQuality != 0) {
                                WifiChart.text = new string('|',signalQuality / (100 / maxChartBars));
                                WifiText.text = "Wifi: " + signalQuality.ToString() + "%";
                        }
                        else {
                                WifiChart.text = "";
                                WifiText.text = "Wifi: 0%";
                        }
                }
}
```