Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Control del robot IRB120 mediante el casco de electroencefalografía Neurosky Mindwave

Autor: Laura García Martín

Tutor/es: Rafael Barea Navarro

2017

Universidad de Alcalá

Escuela Politécnica Superior

Departamento de Electrónica

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Control del robot IRB120 mediante el casco de electroencefalografía Neurosky Mindwave

Autor: Laura García Martín
Tutor/es: Rafael Barea Navarro

TRIBUNAL:
Presidente: José Luis Martín Sánchez
Vocal 1º: Álvaro Hernández Alonso
Vocal 2º: Rafael Barea Navarro

Fecha: 30 de Junio de 2017

Quiero dar las gracias a mi padre, a mi madre y a mi "Tati" por apoyarme en este periodo de mi vida, sin ellos no hubiera sido posible acabar.
A Fede, por escuchar mis charlas sobre temas que no entiende y aguantar mis momentos de agobio.
A mi prima Bárbara por acompañarme a la biblioteca todos los fines de semana. A mi "Titi" por esas ricas comidas de los sábados.
Y en especial a mi abuela, que aunque no me verá terminar sé que me está apoyando.

Índice general

Índice de figuras

Índice de tablas

I Resumen en Castellano

El propósito de este Proyecto es la creación de un sistema Brain Computer Interface (BCI). Para lograr este propósito se ha trabajado con el casco Mindwave de Neurosky, que recibe impulsos cerebrales a través de un electrodo y también con el robot IRB120 que es el dispositivo que realizará la acción.

Con el fin de capturar y procesar señales cerebrales, se utilizará el software matemático Matlab y se necesitará el simulador ABB RobotStudio para controlar el robot.

Por otro lado, se utilizará un socket TCP/IP de comunicación para establecer la comunicación entre Matlab y RobotStudio.

Además, se diseñará una interfaz gráfica en Matlab para que el usuario pueda comunicarse con el casco y el robot desde la misma ventana.

II Abstract

The purpose of this Project is the creation of a Brain Computer Interface system (BCI). To achieve this purpose it has been working with the Mindwave helmet of Neurosky, that receives brain impulses through an electrode, and also with the robot IRB120 which is the device that will implement the action.

In order to capture and process brain signals, the mathematical software Matlab will be used and the ABB RobotStudio simulator will be needed to control the robot.

Futhermore a communication TCP/IP Socket will be utilized to establish the communication between Matlab and RobotStudio.

In addition, a graphical interface is going to be designed in Matlab so that the user can communicate with both the helmet and the robot from the same window.

III Palabras clave

- ***Brain Computer Interface(BCI)***
- ***Casco Mindwave de Neurosky***
- ***Robot IRB120***
- **Interfaz**

IV Memoria

1. Introducción

Las tecnologías BCI (interfaz cerebro-computadora) constituyen un área de investigación relativamente joven. Fue en 1979 cuando empezaron a surgir los primeros proyectos de investigación debido a la relación entre las señales EEG y los movimientos reales e imaginarios de los usuarios. El objetivo principal de esta tecnología fue la creación de una interfaz que permitiera a las personas con graves discapacidades, motoras o cerebrales, poder comunicarse o realizar tareas gracias a dispositivos electrónicos.

Para el gran público los sistemas BCI aún son lentos, imprecisos, incómodos y caros, pero pueden ser de gran utilidad cuando el entorno no permita la comunicación con los interfaces habituales; cirujanos, mecánicos, soldados o pilotos pueden necesitar un canal de comunicación alternativo cuando sus manos están ocupadas o la comunicación vocal es imposible. También podrían recibir información del sistema BCI para concentrarse en una actividad o ayudar a conseguir un estado de relajación. [7]

1.1. Estado del arte

En este apartado se revisan algunos ejemplos relevantes de proyectos que permitan identificar *grosso modo* investigación de esta tecnología en auge. A continuación se hará un breve resumen de dichos proyectos:

I) Proyecto "*Brain-Computer Interface for cognitive training and domotic assistance against the effects of agering".* Desarrollado por el grupo de Ingeniería Biomédica de la Universidad de Valladolid. Se emplean los sistemas BCI que traducen las intenciones del usuario en comandos de control, como herramienta de entrenamiento cognitivo que ayude a prevenir los efectos del envejecimiento. También se desarrolla una aplicación BCI asistida que permite el control de dispositivos domóticos y electrónicos presentes en una vivienda. Estas aplicaciones permitirán entrenar diferentes procesos cognitivos y controlar múltiples dispositivos de climatización y calefacción, de iluminación, de entretenimiento (TV, DVD,). . Este BCI se basa en imágenes motoras y en potenciales P300. [12]

II) Proyecto "*Robust classification of EEG signal for brain-computer interface",* llevado a cabo por Lab. Inst. for Infocomm Res., Singapore. Se describe un BCI que presenta al usuario 36 letras organizadas en una matriz de 6 filas y 6 columnas que se iluminan de forma aleatoria aunque alternativa de forma que, tras 12 iluminaciones, todas las filas y columnas se han iluminado una única vez. Cuando el usuario se fija en una letra, la infrecuente iluminación de ésta (2 de cada 12 veces) hace que se produzca un pico de potencial en nuestras ondas cerebrales (P300), cada vez que se ilumina, que es registrado por el sistema BCI. [7] [13]

III) Proyecto europeo "*PRESENCCIA"*. Una de las actividades que se llevó a cabo en este proyecto fue la de mover una silla de ruedas en un entorno virtual, por un usuario tetrapléjico, sin más herramienta que pensar en el movimiento de sus pies.

El sistema BCI utilizado se basaba en la detección de la actividad cerebral en la banda de frecuencia 15-18 Hz (ritmos beta β), de forma que, en presencia de actividad en esa banda, el avatar que representa al usuario se desplaza hacia delante y en ausencia, permanece parado. [7]

IV) Proyecto "*MAIA"*. Sirvió como base para la aplicación de control de una silla de ruedas a través de un sistema BCI que, además, incorporaba sensores que captaban la información del entorno para convertir los comandos del usuario en movimientos de la silla.[7] [14]

V) Proyecto "*CT2WS o Sistema de Aviso de Amenazas basado en Tecnología Cognitiva"* desarrollado por el DARPA (Agencia de Investigaciones Avanzadas para la Defensa de Estados Unidos). Este proyecto pretende desarrollar unos binoculares que permitan la visión panorámica, tanto diurna como nocturna, y que utilicen la actividad cerebral para avisar a los soldados de infantería de posibles amenazas. De esta forma, el sistema contaría con un casco equipado con electrodos para registrar EEG, lo que permitiría a los soldados recibir información sobre los estímulos visuales de su entorno que hayan provocado una determinada respuesta neuronal. [7]

VI) Proyecto*, "Toward Self-Paced Brain-Computer Communication: Navigation Through Virtual Worlds".* Consiste en un juego experimental en el que el jugador debe navegar por un entorno virtual –*FreeSpace*– realizando tres posibles movimientos: girar a la izquierda, a la derecha o avanzar; con la finalidad de recoger los diferentes objetos que aparecen. Este sistema registra los eventos de sincronización y desincronización (ritmos sensoriomotores) que se producen cuando el usuario piensa en el movimiento de la mano izquierda, de la mano derecha y de un pie. [7] [15]

VII) En 2009, Mattel se asocia con NeuroSky y crean "*Mindflex*" un juego que utiliza un EEG para dirigir una bola a través de una carrera de obstáculos. Con gran diferencia es el producto más vendido, hasta la fecha, basado en EEG.
Neurosky ha desarrollado otro tipo de videojuegos que utilizan la capacidad de relajación o de concentración de los jugadores e incluso el estado emocional de éstos, para conseguir los retos propuestos o hacer que el videojuego se adapte al jugador.

1.2. Objetivos del TFG y campos de aplicación

El objetivo de este TFG es la creación de un sistema BCI "*Brain Computer Interface*" (en español, Interfaz Cerebro Computadora). El sistema se inicia con la medición y adquisición de señales cerebrales mediante la utilización del casco Mindwave de la empresa Neurosky. Posteriormente estas señales se procesan con el software Matlab. Se emplea un socket TCP/IP que permite la comunicación entre dicho software y el brazo robotizado IRB120 (que utiliza el software de la empresa ABB RobotStudio). Con el fin de que el usuario pueda llevar a cabo todos los pasos de este sistema de forma intuitiva y dinámica se ha creado una interfaz gráfica en Matlab.

En conclusión, lo que se persigue con este proyecto es poder controlar los movimientos del robot a partir de impulsos nerviosos.

En la figura siguiente (figura 1.1) se puede observar de forma esquematizada el objetivo a lograr en este proyecto:

Figura 1.1 *Esquema general del sistema BCI implementado*

Para lograr este objetivo se ha realizado un estudio exhaustivo del sistema: la generación de impulsos nerviosos, el tipo de ondas cerebrales recogidas, los tipos de sistemas de adquisición de esos impulsos y los tipos de BCI. Ha sido necesario también el aprendizaje del software de Matlab y Robotstudio, la utilización del casco Mindwave, el estudio del funcionamiento del socket de comunicación, la implementación de una interfaz gráfica y el procesado de imágenes, para la creación del BCI.

La tecnología BCI ha requerido la colaboración de múltiples disciplinas: biología, biotecnología, ingeniería biomédica, ciencia del conocimiento, tecnología de la información, robótica.

1.3. Estructura de la memoria

En este apartado se explicará cómo está estructurada la memoria de este "*Trabajo Fin de Grado*". Vista la memoria de forma general consta de dos partes diferenciadas: los **fundamentos teóricos** del proyecto y la **aplicación final**.

En el **primer capítulo** de la parte ***"IV Memoria"*** se encuentran la introducción al proyecto, el estado del arte y los objetivos del proyecto, así como la estructura de la memoria.

Los **fundamentos teóricos**, en la parte "***IV Memoria***", abarcan del capítulo 2 al 6. El **capítulo 2** resume la anatomía del encéfalo, haciendo gran hincapié en las neuronas. En el **capítulo 3** se hace un estudio del electroencefalograma, incluyendo la historia, los tipos de ondas que se pueden generar y procesar, los diferentes electroencefalogramas, los tipos de electrodos existentes, el sistema de colocación de electrodos (sistema 10-20), los posibles artefactos (ruidos) que pueden afectar a la medición de un electroencefalograma y, por último, los distintos montajes que se pueden realizar. El **capítulo 4** describe un sistema "Brain Computer Interface", incluyendo los tipos de BCI's, endógenos y exógenos, y los métodos de adquisición de la actividad neuronal, invasivos y no invasivos. En el capítulo 5 y 6 se presentan el software y hardware usado en la aplicación. Para la adquisición de los datos, se usa el casco Mindwave, recogido en el **capítulo 5** donde se comentan sus características y su protocolo de comunicación.

Por último, en el **capítulo 6**, se presenta de forma general el Robot Industrial de ABB, software y hardware.

El **capítulo 7** de la parte "***IV Memoria***" es la aplicación del proyecto, el **sistema BCI**. Este capítulo está dividido en **6 grandes apartados**. El **primer apartado** trata del casco Mindwave: adquisición de datos, programación y mejora de la actividad neuronal (atención y meditación). El **segundo apartado** detalla el desarrollo del procesamiento de imágenes, desde su captura hasta la conversión pixel-mm del centroide de las piezas. El **apartado tres** describe de forma general el socket de comunicación entre el software Matlab y el software RobotStudio, con la finalidad de dirigir el movimiento del robot desde el controlador creado en Matlab. En el **apartado cuatro** se detalla la estación de trabajo en RobotStudio del sistema simulado y la programación necesaria en el sistema real. En el **penúltimo apartado** se explican los aspectos necesarios para la creación de una interfaz de gráfica a través de GUI. Los resultados de la aplicación del sistema simulado y del sistema real se recogen en el **último apartado del capítulo**.

El **capítulo 8** recoge las **conclusiones** finales del proyecto y las posibles futuras líneas de investigación.

En la parte "***V Manual de Usuario***" se ha redactado un manual de uso de la interfaz del proyecto. Toda la programación relevante está recogida en la parte "***VI Planos***". En el apartado "***VII Pliego de condiciones***" y en "***VIII Presupuesto***" se detalla todo el software y el hardware usado y el coste final del proyecto.

En el apartado "***IX Bibliografía***" están todas las fuentes consultadas en el proyecto. Y por último, en la parte "***X Apéndices***" se ha creado el apéndice A, que recoge todas las abreviaturas utilizadas en el escrito.

Para situar al lector, a continuación (figura 1.2) se presenta un esquema que muestra los contenidos y apartados generales de este Trabajo fin de Grado.

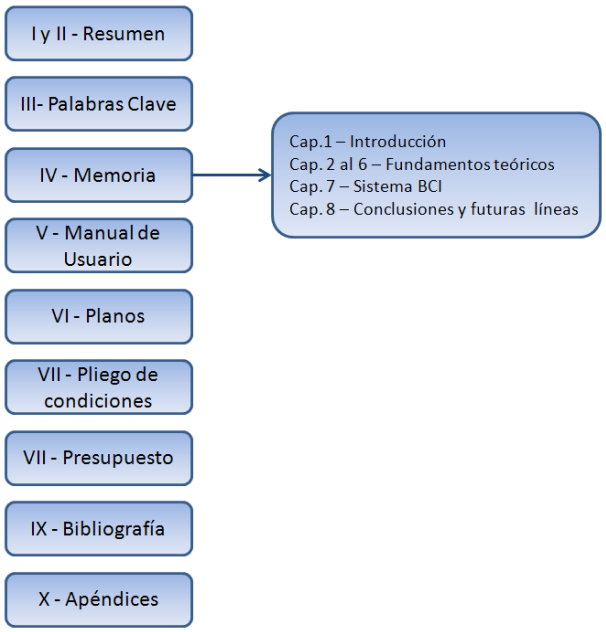


Figura 1.2 *Esquema de la estructura de la memoria*

2. Fundamentos fisiológicos

2.1. Anatomía del encéfalo

El **encéfalo** es el órgano encargado del correcto funcionamiento del cuerpo, realizando el control voluntario e involuntario del mismo, y junto con la **médula espinal** conforma el **sistema nervioso central**. Ambos están compuestos por varios millones de células especializadas llamadas **neuronas** (formadas por nervios sensitivos y motores), cuya funcionalidad es recibir y procesar las sensaciones recogidas por los órganos receptores y transmitir las órdenes de respuesta de forma efectiva a los distintos órganos efectores. El sistema nervioso central se encuentra protegido por tres membranas: duramadre (membrana externa), aracnoides (membrana intermedia) y piamadre (membrana interna), conocidas como meninges. Tanto la médula espinal como el encéfalo están protegidos por una envoltura ósea, la columna vertebral y el cráneo, respectivamente.

Anatómicamente, el encéfalo está formado por el cerebro, el cerebelo y el bulbo raquídeo (figura 2.1).

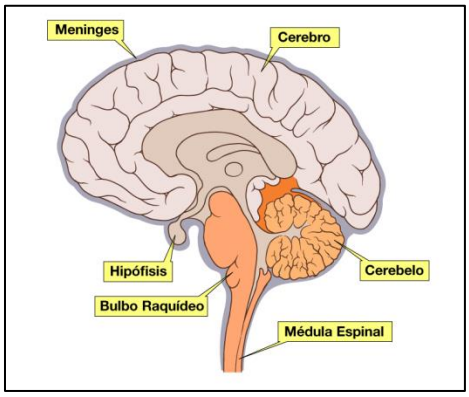


Figura 2.1 *Vista lateral del encéfalo* [22]

El **bulbo raquídeo** es la terminación de la parte superior de la médula espinal. Actúa sobre los movimientos involuntarios del corazón, interviene en el funcionamiento de las vías respiratorias, intestino delgado, hígado y participa en los mecanismos del sueño y la vigilia.

El **cerebelo** está localizado en la parte posterior e inferior del cerebro. La funcionalidad principal es, junto con el bulbo raquídeo, integrar el cerebro con la médula actuando de puente para la llegada y trasmisión de los impulsos neuronales entre ambos. Su función principal es integrar las vías sensitivas y las vías motoras. El cerebelo se encarga de mantener el equilibrio del cuerpo coordinando los movimientos musculares voluntarios. Actúa como "filtro paso bajo" facilitando que los movimientos musculares sean suaves y precisos, y filtrando los impulsos musculares espasmódicos.

El **cerebro** se sitúa en el centro del sistema nervioso, en la parte anterosuperior del encéfalo. Es capaz de realizar funciones muy complejas y regula los movimientos voluntarios y la actividad consciente. Es el generador de ideas, realiza las conexiones, es el centro de las funciones intelectuales, equilibra al organismo con el medio ambiente.

El cerebro está protegido por el cráneo, la duramadre, la piamadre y la aracnoides. Un cerebro humano tiene un peso aproximado de 1,5 kg -un 2% del peso total del cuerpo- y un tamaño de 1260 cm^3 en el hombre y 1130 cm^3 en la mujer; consume el 25% del total del oxígeno y el 20% de la sangre que bombea el corazón. Está formado por la sustancia blanca, que es la ramificación de las células nerviosas, **neuronas**, y por la sustancia gris que son los cuerpos neuronales que forman la corteza cerebral. Se alojan alrededor de cien mil millones de neuronas en él [10]. Se puede dividir en dos partes: **Telencéfalo** y **Diencéfalo**. El telencéfalo corresponde a los **hemisferios cerebrales**, mientras que el diencéfalo está formado por la hipófisis, el hipotálamo, el subtálamo, el tálamo y el epitálamo.

En el **diencéfalo** (figura 2.2) se procesan las emociones. Mientras el tálamo recibe la información de diversos órganos sensoriales, la filtra y se la envía al cerebro, el hipotálamo controla y regula diversos procesos de la vida como el ciclo sueño-vigilia, el hambre, la sed, la temperatura y la función reproductora. Desde el punto de vista fisiológico el hipotálamo es una de las piezas claves del sistema límbico que tiene importantes funciones en la regulación las emociones.

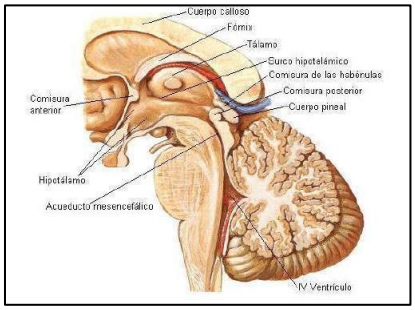


Figura 2.2 *Vista lateral del diencéfalo* [23]

En el **telencéfalo** se pueden diferenciar dos hemisferios, izquierdo y derecho (figura 2.3). Estos representan el 85% del peso total del cerebro. Los dos hemisferios están conectados por una gran cantidad de fibras celulares nerviosas formando el cuerpo calloso.

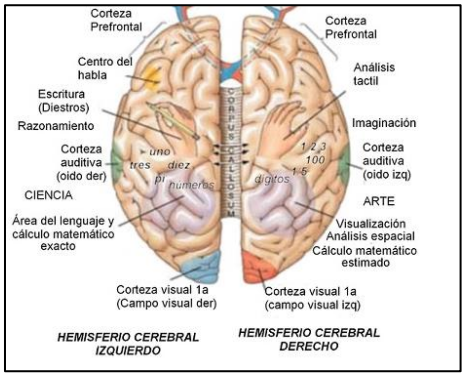


Figura 2.3 *Vista de los dos hemisferios y el cuerpo calloso* [21]

Actualmente, estudios científicos avalan que no hay diferencia en cuanto al trabajo que realiza cada hemisferio. La única disparidad recae en la manera en que cada uno tiene de procesar la información, y en que cada hemisferio controla y se relaciona con la mitad del cuerpo opuesto; las conexiones se cruzan, es decir, el hemisferio derecho interactúa con el lado izquierdo del cuerpo y viceversa.

El **hemisferio izquierdo** procesa la información de forma analítica, secuencialmente. Es la mitad más compleja y suele ser la dominante en la mayoría de los individuos. Este hemisferio se encarga de la expresión oral y la comprensión del lenguaje. También se le atribuye la capacidad de análisis, de razonar, de resolver problemas numéricos o hacer deducciones. El **hemisferio derecho** en cambio, procesa la información de forma global, une y relaciona varios tipos de datos (sonidos, imágenes, olores, sensaciones) y lo trasmite en un todo. Está especializado en sensaciones, sentimientos y habilidades visuales y sonoras, pero no verbales. El hemisferio derecho es el receptor e identificador de la orientación espacial, así como de la percepción del mundo (color, forma). Resumiendo, el hemisferio izquierdo lleva a cabo un pensamiento lógico y el derecho uno intuitivo.

Estos hemisferios están cubiertos por una capa externa compuesta por una sustancia gris llamada **corteza cerebral** o **córtex,** donde se encuentran la mayoría de las neuronas del cerebro humano. Esta capa se encarga de procesar la información sensorial recibida del mundo exterior, participa en numerosas funciones como el lenguaje, los impulsos motores voluntarios, la memoria o los sentidos.

La corteza cerebral se divide en cuatro lóbulos visibles (figura 2.4) bien limitados gracias a los surcos más prominentes, la **cisura central** y la **cisura lateral**, y un quinto lóbulo no visible desde el exterior cerebral.

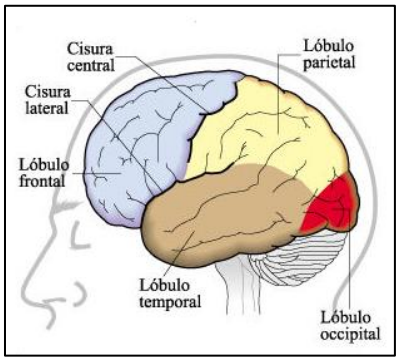


Figura 2.4 *Vista lateral lóbulos del cerebro* [24]

En la parte frontal del cerebro se encuentra el **lóbulo frontal** (figura 2.4) está delimitado por la cisura central o cisura de Rolando y por la cisura lateral o de Silvio. Incluye la corteza motora, controlando movimientos voluntarios de partes específicas. En él se llevan a cabo las "funciones ejecutivas" que son las que asociamos a la toma de decisiones (uso de la memoria, planificación, selección de objetivos...).

En el lóbulo frontal se encuentra el área de broca que aporta la capacidad lingüística. También se localiza en él una región denominada corteza prefrontal que se vincula con la personalidad del individuo, los sentimientos y el juicio, e interviene en el proceso de atención.

Detrás de la cisura central y sobre la cisura lateral se encuentra el **lóbulo parietal** (figura 2.4). Esta zona se encarga de la percepción, procesamiento y reconocimiento de la información sensorial procedente de varias partes del cuerpo como pueden ser la temperatura, el dolor, la presión, los estímulos táctiles. En este lóbulo reside el conocimiento de los números.

El **lóbulo temporal** (figura 2.4) se sitúa debajo de la cisura lateral, en el costado del cerebro y debajo del lóbulo frontal y parietal. Por su situación este lóbulo es el encargado de la percepción, procesamiento y reconocimiento de los estímulos auditivos y olfativos. Las neuronas que encontramos en esta región responden a las distintas frecuencias del sonido, situándose aquéllas de más alta frecuencia en la parte delantera y aquéllas de más baja en la parte trasera. El lóbulo temporal es el encargado de la formación y almacenamiento de los recuerdos. Se puede diferenciar el lóbulo temporal dominante, implicado en el recuerdo de palabras y el reconocimiento de objetos y el lóbulo temporal no dominante implicado en la memoria visual (recuerdo de caras, imágenes).

El **lóbulo occipital** (figura 2.4) se encuentra en la parte posterior del cerebro. Se ocupa de la percepción e interpretación de los estímulos visuales y del reconocimiento espacial.

El quinto lóbulo o **lóbulo ínsula**, no visible desde fuera del cerebro, está localizado en el fondo de la cisura lateral. Es el centro de conexión entre el sistema límbico y el neocórtex, encargado del razonamiento humano.

2.2. La neurona

Las neuronas (figura 2.5), en términos generales, reciben la información en forma de estímulos de naturaleza eléctrica y química procedente de otras neuronas u otros tipos de células a través medio de las dendritas. Éstas transmiten dicha información en forma de señales eléctricas denominadas **impulsos nerviosos**, hacia el núcleo o cuerpo neuronal que se encarga de procesar las señales de entrada y emitir una respuesta. Finalmente, la señal de repuesta es transmitida y propagada mediante el axón hacia los botones sinápticos, pudiendo ser esta señal, estímulo para otra neurona y formar así las redes neuronales presentes en el sistema nervioso. Estas respuestas se conocen como los procesos de excitación o inhibición de las neuronas. La conexión entre una neurona y otra se denomina sinapsis.

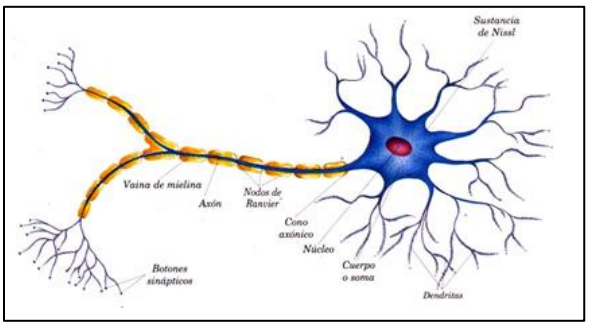


Figura 2.5 *Estructura general de una neurona* [27]

Una región sensorial capta un estímulo externo o interno; esta información es trasportada por las neuronas donde un componente integrador la analiza, procesa y genera una respuesta que será conducida por las neuronas hasta un componente motor que la ejecutará mediante contracciones musculares o secreciones glandulares.

Las neuronas transmiten ondas de naturaleza eléctrica originadas como consecuencia de un cambio transitorio de la permeabilidad en la membrana plasmática. Su propagación se debe a la existencia de una diferencia de potencial o potencial de membrana (que surge gracias a las distintas concentraciones de iones a ambos lados de la membrana) entre la parte interna y externa de la célula, por lo general de –70 mV. La carga de una célula inactiva se mantiene en valores negativos (el interior respecto al exterior) y varía dentro de unos estrechos márgenes. Cuando el potencial de membrana de una célula excitable se despolariza más allá de un cierto umbral (de 65 mV a 55 mV app) la célula genera un potencial de acción.

Un potencial de acción (figura 2.6) es un cambio muy rápido en la polaridad de la membrana que pasa de negativo a positivo ("**despolarización**") y vuelta a negativo ("**repolarización**") en un ciclo que dura unos milisegundos. [3]

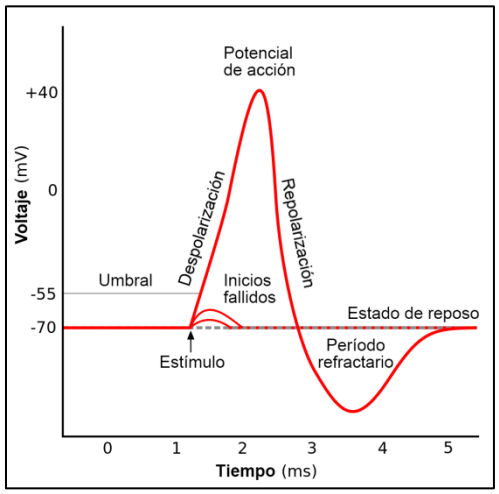


Figura 2.6 *Potencial de acción neuronal* [28]

Se calcula la diferencia de potencial por la diferencia absoluta entre las cargas positivas y negativas entre el interior y el exterior en relación a la membrana. La transmisión eléctrica en los axones se realiza mediante la apertura de ciertos canales de sodio y potasio (figura 2.7). Para que dicha transmisión sea eficiente, la carga de las células en reposo debe ser negativa. Si el estímulo supera el umbral mínimo, se abren los canales regulados por el voltaje Na+ lo que genera la entrada rápida de iones Na+, dando lugar a la despolarización de la membrana mientras que los canales K+ permanecen cerrados.

Cuando el potencial alcanza su máximo (valores positivos) se cierran los canales Na+ y se abren los canales de potasio generando la salida de los iones de K+, y así se reestablece la carga positiva en el exterior de la membrana, produciendo la repolarización de la misma. Por otro lado, para corregir las diferencias de concentración iónica y así reestablecer el potencial de reposo, actúan las bombas de Na/K, expulsando 3 iones de sodio e ingresando 2 de potasio.

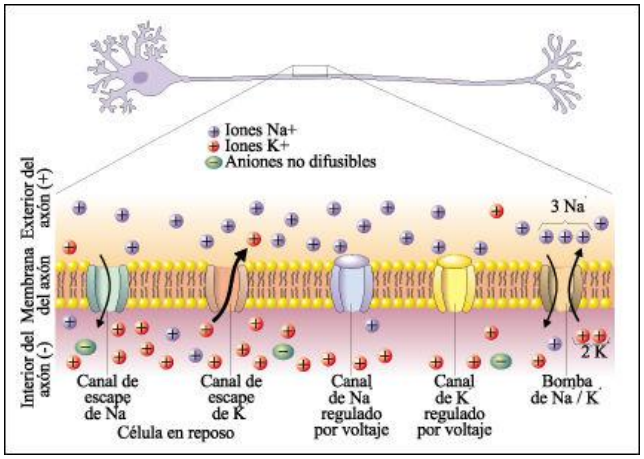


Figura 2.7 *Potencial de acción neuronal desde el punto de vista fisiológico* [29]

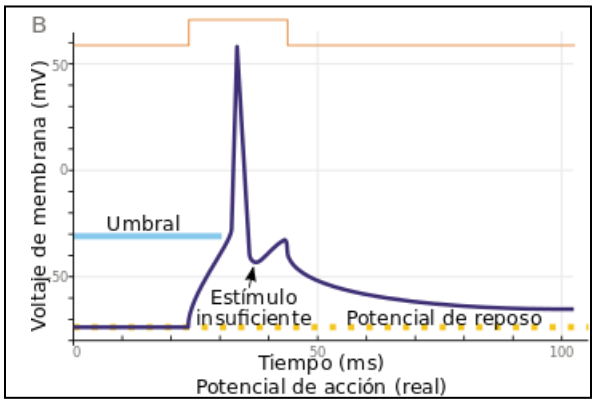


Figura 2.8 *Registro real de un potencial de acción [26]*

En la figura 2.8 se observa un registro real del potencial de acción, deformado debido a las técnicas electrofisilógicas. La electrofisiología estudia las propiedades eléctricas de las células, las medidas del cambio de voltaje o corriente eléctrica. Se incluyen las medidas de la actividad eléctrica de las neuronas (potencial de acción).

Un ejemplo de registro electrofisiológico seria la **electroencefalografía**.

3. Electroencefalograma (EEG)

El electroencefalograma es el registro de los potenciales eléctricos producidos por las neuronas, obtenidos a partir de **electrodos** colocados en la superficie del cuero cabelludo que captan la actividad eléctrica. En el cerebro hay miles de millones de neuronas y cada una genera un impulso eléctrico. El conjunto de todos ellos es lo que captan los electrodos, por lo que el EEG es la superposición de gran cantidad señales eléctricas individuales generadas por las neuronas.

3.1. Historia del EEG:

En 1870, Fritsch y Hitzig observaron que al inyectar corriente galvánica en uno de los dos hemisferios del cerebro se producían espasmos en el lado opuesto del cuerpo. Durante los años posteriores se confirmó que el cerebro era capaz de producir corrientes eléctricas que incluso se llegaron a registrar, en lo que se llamó electrocerebrograma.

Aunque se sospechaba que los impulsos eléctricos se podrían registrar desde el exterior del cráneo, hasta ese momento todos los experimentos se habían hecho en cerebros descubiertos, ya que las actividades eléctricas eran muy débiles y no se tenían los medios necesarios para amplificarlas.

En 1928, Hans Berger ideó un método para la captación de la actividad eléctrica, conocido como <<ritmo de Berger>>. La falta de conocimientos técnicos hizo que no pudiera seguir su investigación hasta el año 1934, cuando Adrian y Matthews verificaron el <<ritmo de Berger>> e introdujeron mejoras que permitieron a Berger llegar a observar que cuando un sujeto movía los ojos o resolvía un problema matemático el ritmo amplio y regular se alteraba. Adrian y Matthews llegaron a la conclusión de que el ritmo regular y amplio de 10 ciclos por segundo se generaba en las áreas visuales y no en todo el cerebro.

Años más tarde y gracias al descubrimiento de Berger, se avanzó mucho en este campo, sobre todo en el estudio de la epilepsia, llegando a la conclusión de que es necesario que estudiar el cerebro en su conjunto y que no se puede dividir ni aislar en funciones simples.

3.2. Electrogénesis cerebral

Todo el sistema nervioso tiene capacidad electrogénica, pero para los sistemas BCI no invasivos, en este caso los basados en el electroencefalograma, se consideran únicamente la corteza cerebral y las regiones directamente relacionadas con ellas.

La actividad eléctrica del tejido cortical se caracteriza por trenes de ondas lentas sobre las que se superponen ritmos rápidos. Entre un tren y otro aparecen períodos de silencio eléctrico. [2]

Las señales son producidas como consecuencia de la actividad sináptica general de regiones discretas del tejido: los PPSE (potenciales postsinápticos excitadores) y los PPSI (potenciales postsinápticos inhibidores) que se suman entre si dando origen a potenciales lentos y éstos son las ondas que se registran. [2]

3.3. Ondas cerebrales

Dentro de la actividad electroencefálica se pueden distinguir diferentes ondas o ritmos cerebrales dependiendo de su frecuencia y localización. Poseen amplitudes que van desde 10 mV en registros sobre el córtex a 100 µV en la superficie del cuero cabelludo. Las frecuencias de las ondas se mueven entre 0,5 y 100 Hz y dependen del grado de actividad del córtex cerebral. La mayoría de las veces estas ondas no poseen ninguna forma determinada: en algunas son ritmos normales que suelen clasificarse en ritmos **alfa**, **beta**, **theta**, **gamma** y **delta** (figura 3.1); en otras poseen características muy específicas de patologías cerebrales como la epilepsia. [2]

Estos ritmos estarán siempre presentes aunque, dependiendo de la actividad que estemos realizando, se distinguirán unos mejor que otros.

Las ondas cerebrales se diferencian por su frecuencia y amplitud. Una frecuencia alta es una alta velocidad de impulso eléctrico, una alta amplitud es una actividad de disparo sincronizada. En resumen, la frecuencia se relaciona con cuándo se disparan las neuronas mientras que la amplitud responde a cuántas neuronas se disparan en ese momento.

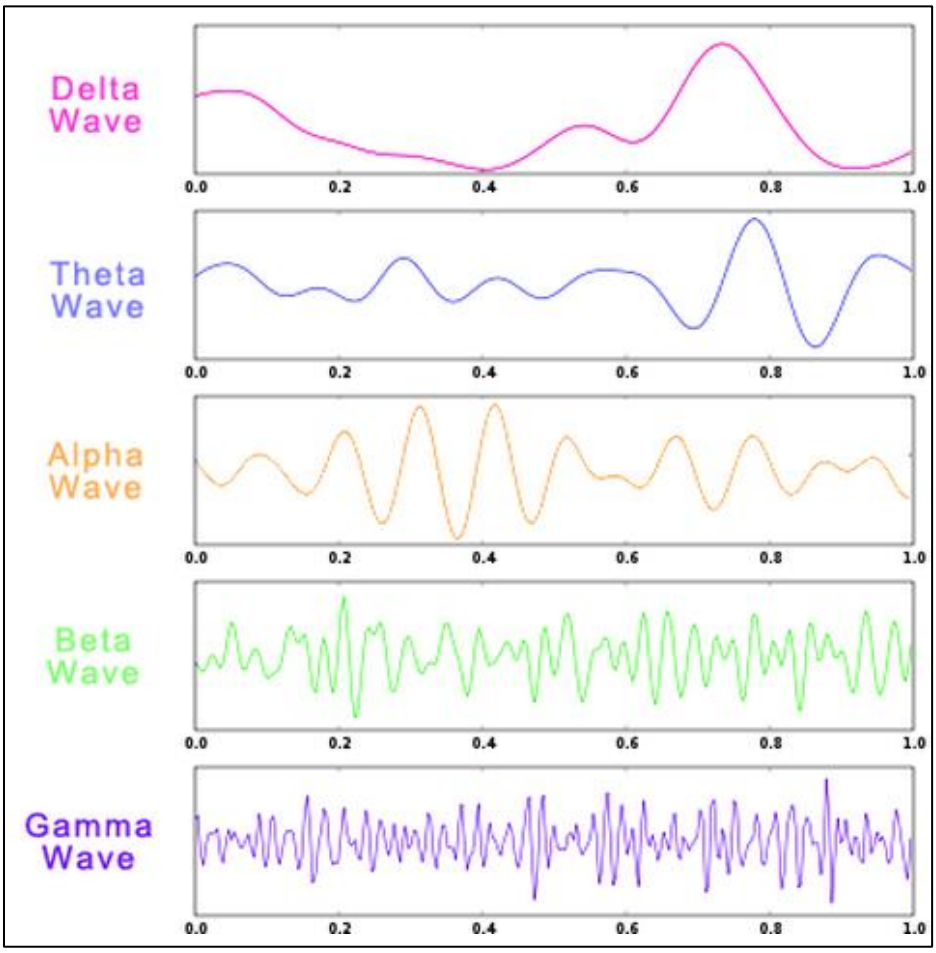


Figura 3.1 *Ondas cerebrales* [30]

Las **ondas delta (δ)** (figura 3.1) son las ondas de mayor amplitud y menor frecuencia. Tienen una frecuencia de oscilación inferior a 4 Hz, sin llegar nunca a cero, ya que eso significaría la muerte cerebral, y una amplitud que varía entre 20 y 200 μV. Se pueden observar en un estado de sueño profundo y en niños de menos de un año. Muy raras veces se observan en adultos despiertos pues esto podría significar una enfermedad cerebral grave. Estas ondas se suelen registrar en el lóbulo temporal (figura 3.2).

Las **ondas theta (θ)** (figura 3.1) están presentes en niños y en adultos en un estado de sueño ligero o en periodos de mucho estrés emocional y ansiedad. Su frecuencia varía entre 4 y 8 Hz y su amplitud entre 20 y 100 μV. Se localizan en la zona parietal y sobre todo en la temporal (figura 3.2).

Las **ondas alfa (α)** (figura 3.1) poseen frecuencias entre 8 y 13 Hz, y una amplitud entre 20 y 60 μV. Estas ondas se observan en adultos y registran un aumento de potencia cuando la persona se encuentra en un estado de no actividad (relajado, con los ojos cerrados). Por el contrario si la persona está realizando un esfuerzo mental o simplemente tiene los ojos abiertos, la amplitud de las ondas disminuye. Estas ondas son más prominentes en el lóbulo occipital (figura 3.2).

Las **ondas beta (β)** (figura 3.1) poseen frecuencias entre 14 y 30 Hz, aunque pueden llegar hasta 50Hz. Su amplitud varía entre 2 y 20 μV. Estas ondas se generan durante estados de concentración mental (realizando un problema de matemáticas, dando un discurso) o en estados de máxima alerta, siendo más amplia la onda cuando alfa está bloqueada. Estas ondas se registran normalmente en las regiones parietal y frontal (figura 3.2).

Las **ondas gamma (γ)** (figura 3.1) son las de mayor frecuencia y menor amplitud. Su frecuencia oscila entre 30 y 60Hz, siendo la frecuencia óptima alrededor de 40 Hz. Se han llegado a registrar ondas gamma de 100 Hz en un experimento llevado a cabo con monjes budistas en el cual se realizaron una serie de electroencefalogramas durante su proceso de meditación. Su amplitud no supera 2μV. Estas ondas se asocian a momentos de gran actividad mental, de extrema atención y concentración además, son las ondas que mejor registran los movimientos motores (figura 3.2).

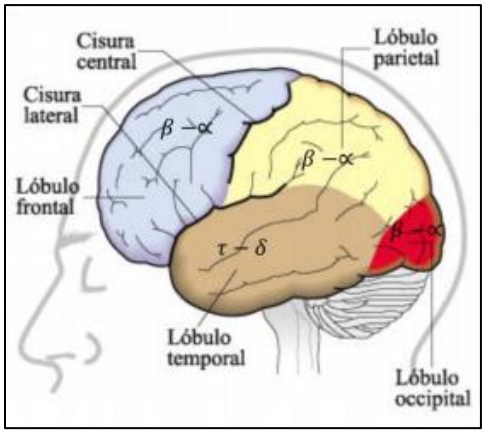


Figura 3.2 *Las zonas de captación de las ondas cerebrales* [8]

3.4. Tipos de EEG

Dependiendo de la necesidad de la prueba existen diferentes tipos de electroencefalogramas:

- **EEG en vigilia**: no es necesaria ninguna preparación previa; la única observación es la limpieza del cuero cabelludo el día anterior y no usar ni tintes, ni acondicionadores, ni laca. Esta prueba se hace en el más absoluto reposo, ya que cualquier movimiento puede alterar los resultados. En algunas ocasiones puede interesar activar el cerebro de algún modo: realizando respiraciones profundas, mirando una luz intensa y parpadeante o escuchando diferentes sonidos.

- **EEG en privación de sueño**: sí se necesita una preparación previa. Consiste en que el paciente se mantenga despierto durante las 24 horas anteriores a la realización de la prueba. Una vez comienza la prueba se recomienda al paciente la máxima relajación, propiciando el sueño, para que de esta forma aparezcan los trazados fisiológicos de las distintas fases del sueño, así como anomalías que no se puedan observar en un EEG normal.

- **EEG de sueño**: es un EEG convencional en el que el paciente es acomodado en una cama y se graba todo el proceso en video. El paciente se puede dormir o no durante la prueba pero no es necesaria una privación del sueño anterior. Se trata de tener constancia visual y eléctrica de crisis y es útil para casos de epilepsia o trastornos del sueño.

- **EEG de muerte cerebral**: es la técnica imprescindible para detectar actividad cortical cerebral o ausencia de la misma cuando el paciente no tiene pulso. Este es el primer paso que hay que dar para poner en marcha el funcionamiento de extracción y/o trasplante de órganos. Se utilizan electrodos de aguja para evitar resistencias y obtener una mejor señal.

3.5. Tipos de electrodos de EEG

Para captar las señales bioeléctricas de la corteza cerebral se usan **electrodos** que son los elementos situados en el punto de registro unidos por un hilo metálico al aparato amplificador.

Dependiendo del **tipo de registro de actividad neuronal** que se vaya a llevar a cabo, existen diferentes tipos de electrodos: **superficiales**, **basales** y **quirúrgicos**.

Para un electroencefalograma (EEG) se usan **electrodos superficiales** que se colocan sobre el cuero cabelludo. En algunos casos se usan los **electrodos basales**, éstos se sitúan sobre la base del cráneo sin necesidad de cirugía. Dentro de los electrodos superficiales se encuentran los adheridos, los de contacto y el gorro de electrodos y entre los basales, se tienen los electrodos esfenoidales.

Los **electrodos superficiales adheridos** (figura 3.3) son pequeños discos metálicos de 5 mm de diámetro. Se adhieren al cuero cabelludo con pasta conductora y se fijan con colodión que es un material aislante. Estos electrodos tienen bajas impedancias (1000-2000Ω). [2]

Figura 3.3 *Electrodo superficial adherido* [31]

Los **electrodos de contacto** (figura 3.4) consisten en pequeños tubos de plata clorurada roscados a soportes de plástico. Se sujetan al cráneo con bandas elásticas y se conectan con pinzas de <<cocodrilo>>. Son de colocación muy fácil pero incómodos para el paciente, por eso no se usan para registros de larga duración (figura 3.4). [2]

Figura 3.4 *Electrodos de contacto* [32]

Dentro de los electrodos de contacto se pueden diferenciar los **electrodos húmedos** -entran en contacto con la piel a través de un gel conductor- y los **electrodos secos** -se usan directamente sobre la piel-. Normalmente, para facilitar la conducción de corriente entre el interior del cuerpo y el metal del electrodo, es habitual usar un gel conductor que penetre en la epidermis. En estos casos se debe tener especial cuidado con no derramar el gel y provocar un cortocircuito con los electrodos próximos. Como este gel puede secarse no es útil para monitorizaciones prolongadas. En cuanto a los electrodos secos, la sudoración propia debajo del electrodo acaba reduciendo la alta impedancia inicial de la piel seca.

Los electrodos superficiales del **gorro de electrodos** son los más usados en la actualidad por su facilidad de uso y de colocación, y por su gran inmunidad a las interferencias. Se trata de una malla elástica que lleva insertados los electrodos de plata clorurada que siguen la colocación convencional del sistema 10-20 (sistema que se explicará en el apartado siguiente).

Mediante una jeringa se introduce en cada uno de los electrodos un gel conductor que facilita la recepción de la señal a través del cuero cabelludo. Todos los electrodos se unen a un conducto, y éste, al aparato que registra la actividad (figura 3.5).

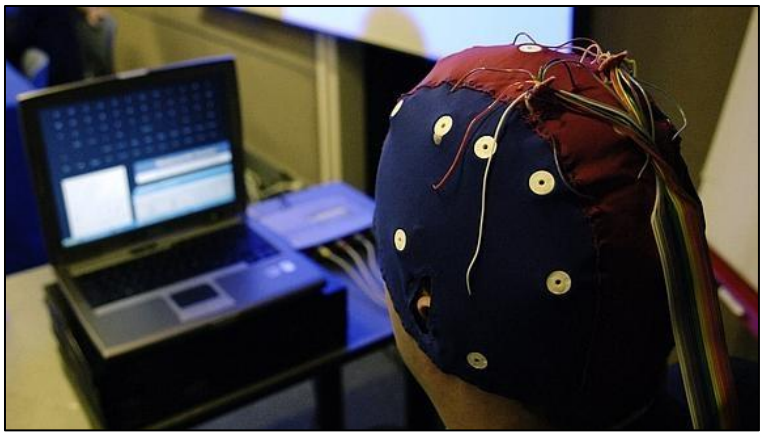


Figura 3.5 *Gorro de electrodos* [33]

Todos los electrodos descritos hasta aquí registran la parte superior de la corteza. Para el estudio de la cara basal del encéfalo se necesitan los **electrodos esfenoidales** que son los más comunes dentro de los electrodos basales y son semi-invasivos. Se usan para explorar la cara basal de la parte interna del lóbulo temporal. Se trata de una aguja hipodérmica de acero de 5 cm de largo, sin riesgo alguno a que se rompa durante las convulsiones. Estos electrodos se dejan fijos durante el electroencefalograma y buscan focos epilépticos para retirarlos y evitar convulsiones generadas por esta enfermedad.

3.6. Sistema internacional 10-20

Existen varios sistemas de colocación de electrodos pero, en la actualidad, el más usado es el sistema internacional <<Diez-Veinte>>, denominado así porque los electrodos están espaciados entre el 10% y el 20% de la distancia total entre puntos reconocibles en el cráneo que sirven como referencia. Esos puntos son: **nasion** (punto de unión entre frente y nariz), **inion** (protuberancia occipital) y punto **preauricular** (delante del trago de cada pabellón auditivo (figura 3.6).

El sistema 10-20 se diseñó para dar una cobertura total de la cabeza, pero existe la posibilidad de colocar electrodos adicionales, siempre y cuando se use la nomenclatura 10-20.

Para colocar los electrodos mediante este sistema, se mide la distancia entre el nasion y el inion pasando por el vertex. El 10% de esta distancia sobre el nasion señala el punto Fp (Frontal polar). El 10% de esta distancia sobre el inion señala el punto O (occipital) (figura 3.6). [2]

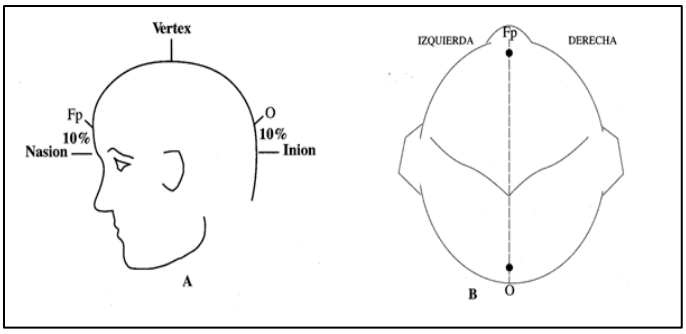


Figura 3.6 A: v*ista perfil,* **B***: vista superior. Fp: punto frontal polar, O: punto occipital* [2]

Entre los puntos FP y O se sitúan otros tres puntos espaciados a intervalos iguales (entre cada dos puntos, el 20% de la distancia nasion-onion). Estos tres puntos son, de delante hacia atrás: el Fz (Frontal), el Cz (Central o Vertex) y el Pz (Parietal) (figura 3.7). [2]

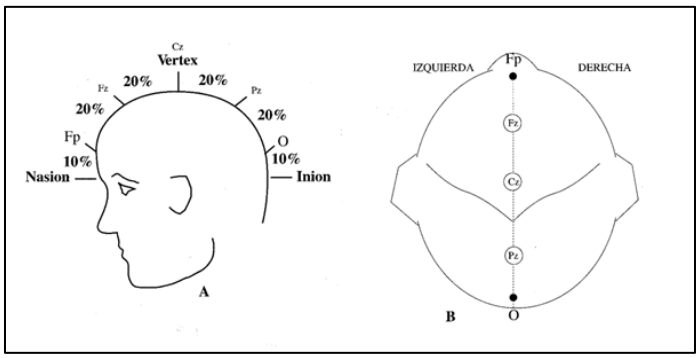


Figura 3.7 A*: vista perfil.* **B***: vista superior. Fz: punto frontal, Cz: punto central, Pz: punto parietal* [2]

Y así sucesivamente hasta tener todos los electrodos colocados (figura 3.8). Además de todos los electrodos que se colocan para captar los impulsos eléctricos existen dos electrodos que se colocan en el lóbulo de la oreja izquierda y derecha (A1, A2) que sirven de referencia para la adquisición de datos, porque experimentan el mismo ruido ambiental que cualquier punto, pero con un mínimo de actividad neuronal, haciendo así las veces de toma de tierra.

Como resumen en cuanto a nomenclatura de los electrodos del sistema, cabe destacar que este sistema divide la cabeza en seis zonas: frontal polar (Fp), frontal (F), central (C), parietal (P), occipital (O) y temporal (T). Los electrodos colocados en el hemisferio derecho llevan numeración par, los electrodos del hemisferio izquierdo numeración impar y los electrodos de la línea media llevan el subíndice “z” (“*zero*” en inglés) (figura 3.8).

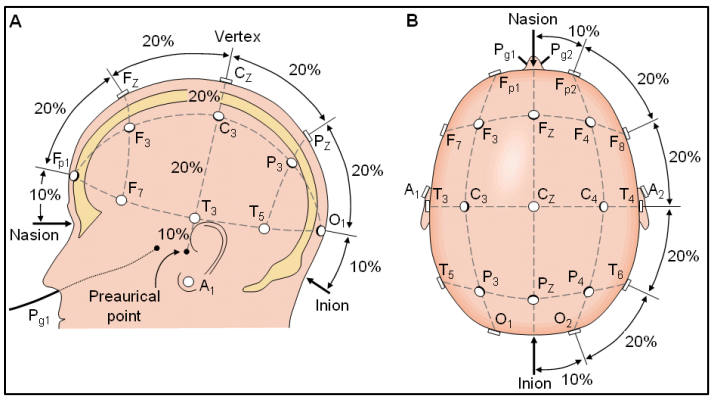


Figura 3.8 *Colocacion de los electrodos basado en el sistema 10-20* [34]

3.7. Artefactos del EEG

La amplitud de una señal EEG típica varía en un rango de 1µV a 200 µV (pp) en una persona adulta. Como las señales que se captan son muy débiles pueden ser fácilmente contaminadas por otras fuentes, denominadas **artefactos.** Un artefacto es cualquier señal EEG que no se ha originado en el cerebro. La finalidad de cualquier aparato de medición de impulsos cerebrales es conseguir un registro libre de artefactos, que no distorsionen la señal para medir, única y exclusivamente, impulsos eléctricos generados por un estado mental.

Se distinguen dos tipos de artefactos: externos (**no fisiológicos**) e internos (**fisiológicos**).

Los **artefactos no fisiológicos** son los derivados de la máquina de EEG, del instrumental o del medio ambiente. Los más frecuentes son los creados por el acoplamiento de la señal de la red eléctrica y los electrodos. Para asegurar una buena medida y eliminar la mayoría de los artefactos externos, tiene que darse un buen contacto entre el cuero cabelludo y el electrodo, evitando el deslizamiento, la suciedad, el exceso o déficit de pasta conductora y el balanceo de los cables, además de apantallar al máximo cualquier interferencia electro-magnética externa.

Los **artefactos fisiológicos** son los derivados del mismo paciente, campos eléctricos capaces de ser generados en cualquier parte del cuerpo.

A continuación se detallan algunos artefactos que deben ser reconocidos:

- Latidos del corazón (Electrocardiograma): el potencial de acción generado por el corazón se trasmite hasta el cuero cabelludo, dando lugar a puntas rítmicas.

- Pulso: la oscilación del pulso genera un impulso rítmico de vaivén donde el tiempo de ascenso es mayor que el de descenso. Este artefacto aparece cuando se coloca un electrodo sobre una arteria del cuero cabelludo.

- Respiración: provocando oscilaciones lentas y rítmicas que pueden durar varios segundos.

- Contracción de los músculos craneofaciales: estos artefactos son registrados en la mayoría de los EEG por los electrodos frontotemporales o parietales y originan ondas irregulares de amplitud y frecuencia elevada.

- Electromiograma o artefactos musculares: son potenciales aleatorios y muy variables,coincidentes con los movimientos del cuerpo. Estos pueden modificar el contacto entre la piel y el electrodo.

- Movimientos oculares: el parpadeo repetitivo da lugar a un movimiento en la zona frontal del cerebro que simula una actividad theta y delta.

- Movimiento de la lengua: dando lugar a la aparición de potenciales aleatorios y lentos similares a las ondas delta.

- Sudor: genera oscilaciones muy lentas en todos los canales, sobre todo en los electrodos frontales.

- Escalofríos: son ráfagas de potenciales puntiagudos rítmicos contaminados con el artefacto muscular y con una frecuencia de 10 a 14 Hz.

3.8. Montajes del EEG

Las señales se registran a partir de una serie de electrodos colocados cada uno en una posición exacta. Cada línea o canal EEG mide la diferencia de potencial eléctrico entre dos electrodos que se conoce como **derivación**. Un estudio electroencefalográfico adecuado requiere un mínimo de 10 canales. El sistema 10-20 coloca 22 electrodos. Hay que decidir entre el montaje para un **registro bipolar** o para uno **monopolar.**

Registro Bipolar

Los registros bipolares utilizan parejas de electrodos: tanto el situado en la posición uno como el electrodo situado en la posición dos, registran actividad cerebral y la diferencia entre los dos puntos es lo que va al amplificador para su registro (figura 3.9). Como se observa en el esquema de la figura 3.9, el electrodo a (posición 1) está a un potencial de +5V, el electrodo b (posición 2) a un potencial de +2V, registrándose así en el canal de amplificación un potencial de (+5)- (+2)= +3V

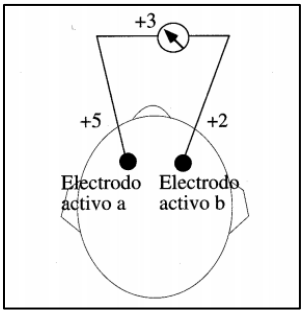


Figura 3.9 *Esquema de un montaje para un registro bipolar* [2]

Existen numerosas combinaciones de registros bipolares, cada combinación es un montaje. Se puede diferenciar entre montajes a **Larga Distancia**, cuando se registra el voltaje entre electrodos no contiguos, y montajes a **Corta Distancia** (figura 3.10), donde el voltaje de un determinado electrodo se compara con el de los electrodos adyacentes.

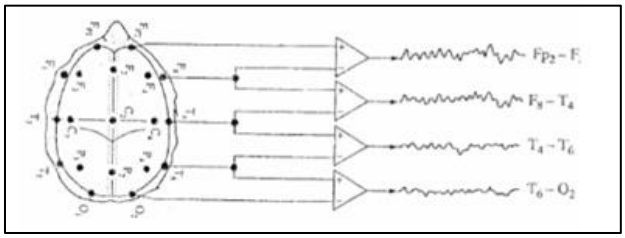


Figura 3.10 *Registro bipolar, montaje a Corta Distancia* [35]

Registro Monopolar o referencial

En el registro monopolar cada electrodo toma la señal de forma independiente. Los electrodos exploradores ocupan la posición uno del amplificador, mientras que la posición dos está ocupada por en electrodo relativamente inactivo y que, teóricamente, tendrá potencial cero (figura 3.11).

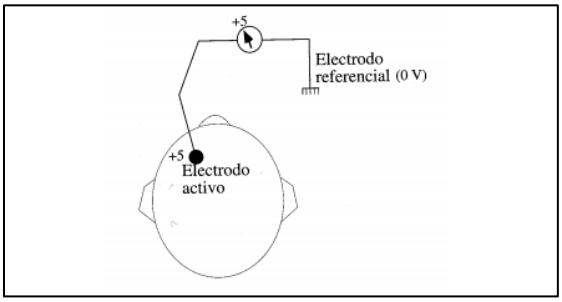


Figura 3.11 *Esquema de un montaje para un registro monopolar* [2]

El electrodo de referencia puede ser colocado en cualquier parte, pero existen algunas consideraciones a tener en cuenta. Si se coloca muy alejado de la zona de medición va a estar contaminado por el ruido generado por otros dispositivos eléctricos. Una de las localizaciones de referencia son los pabellones auriculares (A1, A2 nomenclatura del sistema 10-20) que es el punto común para todos los canales, ya que el lóbulo de la oreja es una zona que experimenta el mismo ruido ambiental que el resto de electrodos, pero con un mínimo de actividad neuronal (figura 3.12). Otra opción posible seria unir todos los electrodos y de esta forma, existiría un punto cuyo potencial sería la suma de los potenciales de cada electrodo.

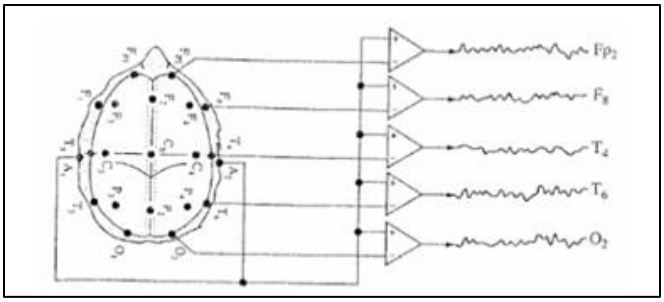


Figura 3.12 *Registro monopolar usando el pabellón auricular como electrodo de referencia* [35]

Tanto si se usa el montaje para un registro bipolar como si se usa el montaje para registro monopolar las señales que se obtienen son muy débiles por lo que hay que usar varios sistemas de amplificación y hacerlas pasar por diferentes procesos de filtrado y procesado de señal para mitigar el gran número de artefactos que les afectan (figura 3.13).

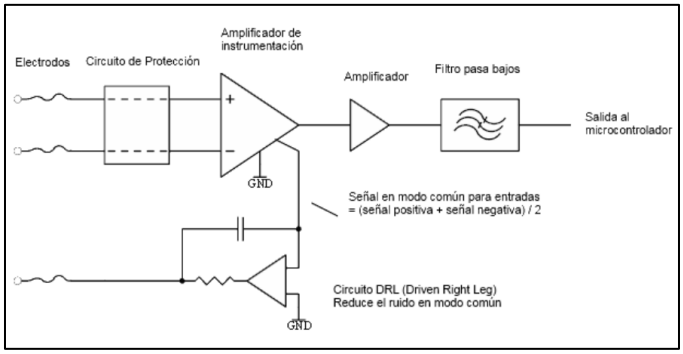


Figura 3.13 *Estructura básica de un electroencefalógrafo*

Los filtros que se emplean son un filtro paso alto (0,5-1 Hz) y un filtro paso bajo (hasta 35-70Hz). El filtro paso alto normalmente filtra los artefactos lentos y los artefactos de movimiento (ocular, lingual), mientras que el filtro paso bajo se usa para eliminar los artefactos de alta frecuencia, como las señales electromiográficas.

Los electrodos captan la señal que pasa a través de un circuito de protección por un hilo metálico que lo une al aparato de amplificación. El circuito de protección sirve para proteger los circuitos de las descargas electrostáticas (ESD) y para proteger al usuario de posibles cortocircuitos. La señal pasa al sistema de amplificación. Los amplificadores utilizados son diferenciales, es decir, reciben el impulso eléctrico de 2 puntos y magnifican la diferencia de potencial entre ellos. Dado que la señal es de amplitud muy pequeña, debe pasar por un segundo sistema de amplificación. Entre estas dos etapas, un filtro paso alto elimina el offset de tensión continua.

Después, la señal se amplifica una vez más y pasa por un filtro paso bajo. El filtrado se realiza para prevenir los efectos del *aliasing* en la digitalización. Finalmente, la señal pasa por un conversor analógico digital (ADC) que normalmente se encuentra dentro del microcontrolador.

Aliasing: es el efecto que causa que señales continuas distintas se tornen indistinguibles cuando se muestrean digitalmente. Cuando esto sucede, la señal original no puede ser reconstruida de forma unívoca a partir de la señal digital.[47]

4. Sistemas Brain Computer Interface (BCI)

Los sistemas Brain Computer Interface, en español interfaz cerebro-ordenador (figura 4.1), forman parte de una tecnología cuya finalidad es la de captar señales cerebrales mediante sensores, procesar la señal adquirida para obtener los datos de interés, y por último interaccionar con el entorno de la manera deseada solo mediante su actividad cerebral, sin utilizar el sistema muscular (sin utilizar su cuerpo). Establecen un canal de comunicación que conecta el cerebro con un ordenador o dispositivo externo.

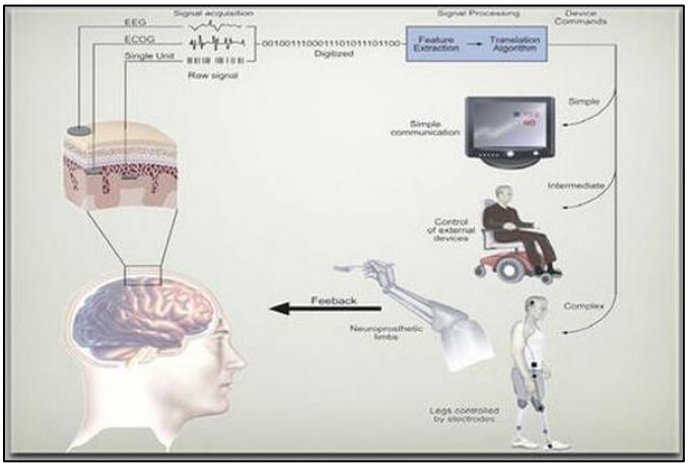


Figura 4.1 *Esquema de un sistema BCI* [36]

Al principio, el campo de investigación BCI se centró en ayudar, reparar o amplificar las funciones senso-motoras o cognitivas humanas mediante dispositivos eléctricos (ordenadores, neuropróteis..) u otras aplicaciones que sirven de ayuda y dan mayor independencia a las personas en su vida diaria.

En la actualidad, esta tecnología no solo ha servido para fines médicos, sino que también esta tecnología se ha centrado en la industria del videojuego, y se pueden encontrar las primeras interfaces cerebro-ordenador no invasivas de uso doméstico como las creadas por las empresas Neurosky o Emotiv.

4.1. Tipos de BCI

Los sistemas BCI se pueden clasificar en **sistemas endógenos** o **exógenos** según la naturaleza de la señal de entrada.

Los sistemas endógenos no necesitan estimulación externa (imágenes, sonidos...) para que se genere la actividad cerebral, dependen únicamente de la capacidad del usuario para controlar dicha actividad. Este tipo de BCI requiere un entrenamiento para mejorar el control.

Dentro de los **sistemas endógenos** se encuentran los BCI basados en **potenciales corticales lentos** (Slow Cortical Potencials, **SCP**) y los basados en **imágenes motoras o sensoriomotores**.

Los **Potenciales Corticales Lentos** corresponden a cambios de voltaje lentos que se pueden observar en intervalos superiores a 5 segundos y se dan en la corteza cerebral. Los SCP negativos se asocian al movimiento y a otras funciones que implican una actividad cortical. Los SCP positivos están relacionados con una reducción de la actividad del córtex. Estos potenciales se pueden controlar con entrenamiento (un par de horas al día durante varias semanas).

En un sistema BCI basado en SCP, se requiere que el sujeto regule su actividad cerebral de forma voluntaria, generalmente usando un paradigma de tipo binario en el que suele haber dos fases: la fase inicial de preparación seguida de la fase activa. Durante la fase activa se pide al sujeto que realice una actividad mental concreta (por ejemplo mover un cursor hacia arriba en la pantalla del ordenador) basándose en estados emocionales o imaginación. En primer lugar se mide el nivel de voltaje inicial durante la fase de preparación de unos 2 segundos, y en los siguientes 3 segundos el usuario realiza la actividad mental concreta, produciendo una disminución o un aumento, de dicho nivel de voltaje(figura 4.2). [17]

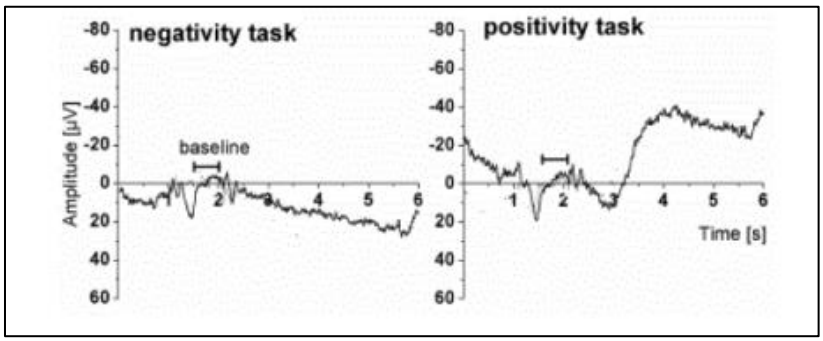


Figura 4.2 *Potencial cortical lento (SCP) negativo y positivo* [17]

Los **ritmos sensoriomotores** se basan en imágenes motoras (movimiento mano izquierda o derecha, de los pies…) u otros tareas mentales (resolución de problemas matemáticos). Este tipo de acciones producen cambios en la amplitud de las ondas beta, registrados en la zona motora y sensitiva de la corteza. Se aprecian los cambios en las ondas durante la ejecución del movimiento, como la "imaginación" del movimiento.

Los **sistemas exógenos** dependen de una estimulación externa y no necesitan ningún entrenamiento. Existen dos tipos de sistemas exógenos: los que se basan en **potenciales evocados P300** o en **potenciales evocados visuales o auditivos**.

Los **potenciales P300,** se registran en el lóbulo central y parietal. Son picos de amplitud que se observan en el EEG 300 ms después de producirse un estímulo visual o auditivo. Los estímulos se producen aleatoriamente dentro de un espacio de tiempo (figura 4.2).

Los **potenciales evocados visuales (VEP)** son los producidos por una estimulación luminosa, normalente se usa una imagen en forma de tablero de ajedrez (con cuadros blancos y negros).

Tambien se usan elementos parpadeantes a diferentes velocidades. En este caso el objetivo es que el usuario fije la atencion en uno de los elementos y que se registren potenciales de la misma

frecuencia que la del elemento que está mirando, para identificar una opcion. Estos potenciales se registran sobre todo en la parte visual del cerebro, en el lóbuo occipital.

Los **potenciales evocados auditivos (AEP)** tienen el mismo fundamento que los visuales, con la salvedad de que el estímulo son sonidos a diferentes frecuencias. El usuario se concentra en uno de ellos, generando una potencia de la misma frecuencia que el estímulo, identificándose así la opción.

Hay un sistema desarrollado en el CITIC-UGR que permite comunicarse a una persona por medio de estos estímulos. Se realiza una pregunta, cuya respuesta sea de carácter binario, y se ponen dos sonidos a diferentes velocidades uno en cada oído. El usuario se concentra en uno de los dos sonidos dependiendo de la respuesta, si la respuesta es sí se concentra en el estímulo oído derecho y si la respuesta es no se concentra en el estímulo del oído izquierdo. [8]

4.2. Métodos de adquisición

Los métodos de adquisición de la actividad neuronal se pueden clasificar en **métodos invasivos** o **no invasivos**; se diferencian en dónde y cómo se colocan los electrodos.

Los **métodos invasivos** introducen los sensores en el cerebro y captan las señales cerebrales. En comparación con los sistemas no invasivos, la intensidad de la señal es mucho mayor, debido a que los sensores se colocan directamente en la materia gris del cerebro (figura 4.3). Por otro lado, estos sistemas son más peligrosos, ya que necesitan de cirugía para ser implantados.

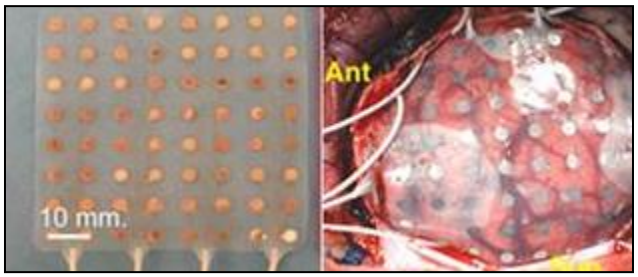


Figura 4.3 *Método de adquisición invasivo, macro red de electrodos* [37]

Normalmente, este tipo de sistemas se usan para personas con discapacidad motora con el fin de mejorar su calidad de vida. Los usuarios controlan dispositivos como sillas de ruedas, brazos robotizados, mediante impulsos eléctricos.

Los **sistemas no invasivos,** en su mayoría, se basan en la electroencefalografía (EEG). Es una técnica más segura, ya que los electrodos se colocan en la superficie externa del cráneo, normalmente en el cuero cabelludo, por lo que no necesitan cirugía.

El inconveniente de estos sistemas es que la señal es más débil y está contaminada por más artefactos que las señales de los invasivos. Estos sistemas usan tanto los **potenciales relacionados con eventos (ERP)** -cambios en el potencial eléctricos en respuesta a un evento-, como los **ritmos cerebrales** -oscilaciones de potencial que en su conjunto dan lugar a los electroencefalogramas-.

Los sistemas BCI que se basan en **potenciales relacionados con eventos**, son los que están relacionados con **estímulos auditivos (AEP)**, y **visuales (VEP)**. Como se explicó anteriormente, se producen estímulos a diferentes frecuencias y el usuario debe fijar su atención en una de las opciones, lo que generara un potencial de la misma frecuencia que la opción elegida. Estos sistemas son **síncronos**, es decir, es el sistema BCI quien presenta los estímulos al usuario, están sincronizados.

Los sistemas que procesan **potenciales corticales lentos (SCP)** para detectar los **ritmos de la actividad cerebral,** se fundamentan en la modificación voluntaria de los ritmos cerebrales que puede realizar un usuario concentrándose en alguna tarea: cálculo matemático, imagen. El usuario debe aprender a controlar sus estados cognitivos. Son sistemas **asíncronos**; el usuario es el que decide cuando enviar una orden. Dentro de este grupo, cabe destacar los BCI de imaginación motora donde el usuario "imagina" que está realizando el movimiento y se produce una actividad en la corteza motora que se puede reconocer, generando una señal que sirva de control para un dispositivo, por ejemplo, un brazo robotizado. Este tipo de sistemas son muy útiles para personas con discapacidad motora severa.

Dentro de los BCI no invasivos, existen una gran variedad de dispositivos comerciales para la detección de ondas cerebrales, destacando el casco de la empresa Emotiv Systems y el de la empresa Neurosky.

El casco de la empresa Emotiv Systems se conoce como Emotiv EPOC. Es un casco que cuenta con 14 electrodos y un sensor giroscópico. Asimismo este casco cuenta algoritmos que analizan el cerebro del usuario y trabajan en función de las características propias del mismo. En comparación, el de la empresa Neurosky, llamado **MindWave**, consta de un solo electrodo que extrae la información de una única zona cerebral. Pero de este casco hay que destacar que recoge todas las ondas cerebrales y que la empresa Neurosky ha creado unos algoritmos propios -eSense- que determinan los estados de atención y meditación.

En este proyecto se decantó por la utilización del **casco MindWave de Neurosky** (figura 4.4), por dos motivos: el primero, que este casco cumple todas las especificaciones necesarias para llevar a cabo la aplicación a desarrollar; el segundo, el precio, frente a los 400€ del casco Emotiv EPOC, este cuesta unos 120€.

Figura 4.4 *Casco Mindwave Mobile* [9]

4.3. Modelo funcional Genérico/Etapas de un sistema BCI

El modelo de un BCI, a grandes rasgos, se puede dividir en cuatro etapas: adquisición de la señal, procesado de la señal, aplicación y configuración (figura 4.5).

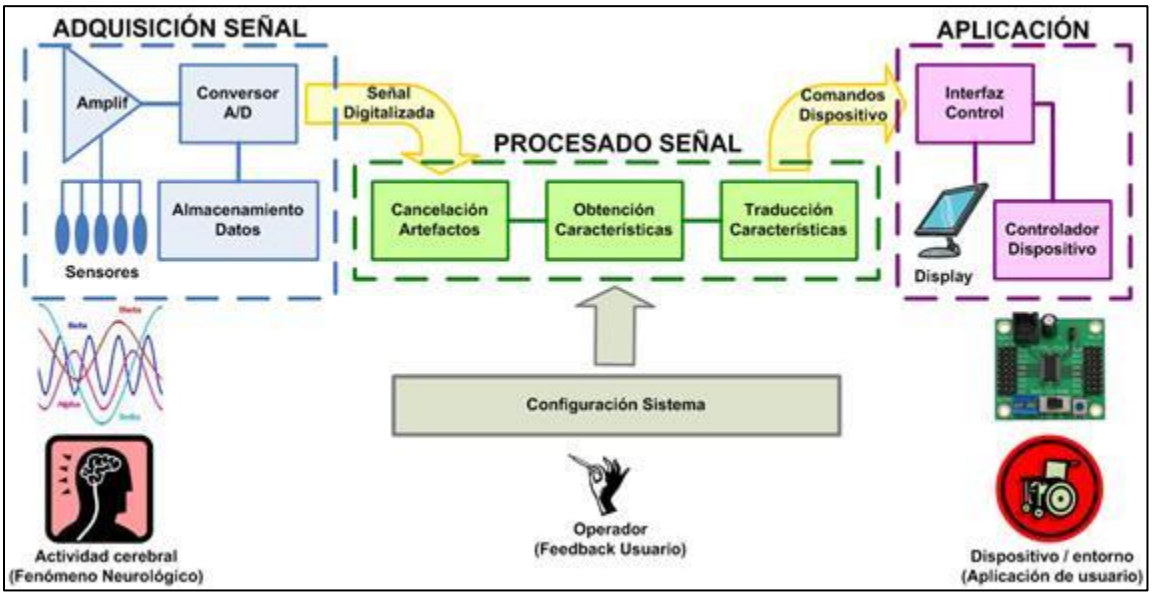


Figura 4.5 *Modelo genérico de los sistemas BCI* [7]

Adquisición de la señal: se registra la actividad cerebral, pasa por varias etapas de amplificación y se realiza la conversión analógica digital. Los BCI normalmente trabajan en tiempo real, pero esta etapa también permite que los resultados se almacenen para próximos análisis.

Procesado de la señal: se recibe la señal digitalizada y se llevan a cabo una serie de análisis para extraer las características de interés para que el dispositivo sobre el que está actuando el usuario sea capaz de interpretar las órdenes. Este bloque se divide en 3 etapas:

- **Cancelación de artefactos**. Esta etapa se dedica exclusivamente a la eliminación del ruido que distorsiona la señal que se quiere estudiar. Eliminan el ruido generado por el movimiento muscular, el movimiento ocular, el que provoca el sudor, los latidos del corazón.

- **Obtención de características**. En este bloque se traduce la señal cerebral de entrada (después de haber sido filtrada y eliminados la gran mayoría de los artefactos) en un vector de características de correlación con el fenómeno neurológico asociado a la señal.[10]

- **Traducción de características**. Se transforma el vector de características en una señal de control adecuada al dispositivo que se pretende controlar.[10]

Aplicación: esta etapa recibe la señal de control ya procesada y realiza las acciones correspondientes en el dispositivo del controlador del mismo. Este bloque también puede incorporar una pantalla que proporcione feedback al usuario. [10]

Configuración: se modifican y definen los parámetros del sistema si es necesario. El que modifica estos parámetros puede ser el propio usuario o un algoritmo automático que ajusta su comportamiento dependiendo de los resultados y del feedback del usuario.

5. Casco MindWave Neurosky

En apartados anteriores se ha descrito la interfaz cerebro-computador, explicando el proceso de funcionamiento, desde la adquisición (método), hasta el control de un dispositivo.

Este apartado, se centra en el dispositivo comercial Mindwave mobile de Neurosky utilizado en este proyecto (figura 5.1).

Figura 5.1 *Casco Mindwave mobile* [9]

5.1. Características

El dispositivo Mindwave es una diadema que recoge la actividad cerebral y divide la señal según la frecuencia. Basándonos en el sistema de colocación de electrodos 10-20, este casco dispone de un único sensor de aleación de acero seco (no necesita gel médico) colocado en la sección **FP_1** y de un electrodo de referencia colocado en la posición **A1** (lóbulo de la oreja izquierda). Este electrodo sirve de referencia, puesto que experimenta el mismo ruido ambiental que el situado en la parte frontal, pero con un mínimo de actividad neuronal. El registro se lleva a cabo mediante la configuración **referencia común** (figura 5.2).

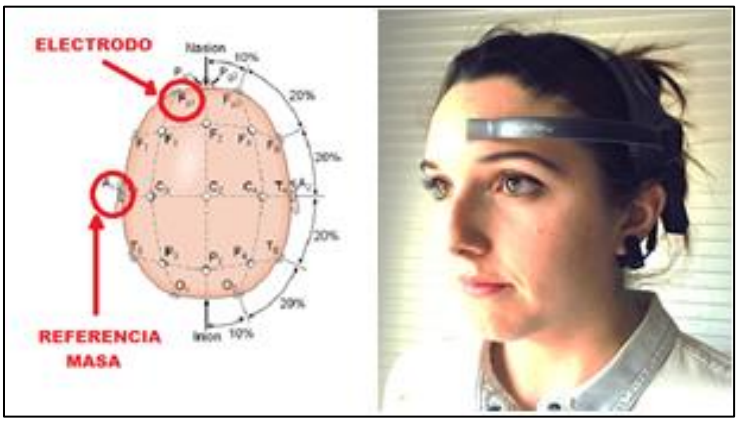


Figura 5.2 *Posición de los sensores del casco*

El sensor frontopolar obtiene los valores y el chip integrado **ThinkGear** procesa los datos de forma digital para que sean utilizados en distintas aplicaciones. En el chip ThinkGear se procesan las ondas cerebrales (**alpha**, **beta**, **tetha**, **gamma**) y el **algoritmo eSense**, dando como resultado valores del nivel de **Atención** y **Meditación** que se representan en una escala del 0 al 100. Por otra parte, este casco permite captar el **parpadeo** del usuario, herramienta que se usará en este proyecto.

La mayoría de las aplicaciones son no-científicas y se fundamentan en la capacidad del usuario para modificar su actividad cerebral de forma **voluntaria,** diferenciando **dos** estados mentales, la atención y la meditación (algoritmo eSense), dando lugar a **resultados binarios**, por ejemplo atención->0, meditación ->1.

Sin embargo, la compañía especifica en su página web que se han realizado estudios de precisión de los datos captados por este dispositivo que tienen una fiabilidad equivalente a aparatos empleados en la medicina actual.

Las medidas que realiza son [9]:

- Ondas cerebrales.
- Espectro de la frecuencia del EEG (alpha, beta…).
- Procesamiento y salida de los niveles de atención y meditación.
- Parpadeo.

Las características físicas son [9]:

- Peso: 90 gramos.
- Medidas: alto 225mm x ancho 155mm x profundidad 165 mm.

Las características del bluetooth son [9]:

- Versión: 2.1.
- Potencia de salida: clase 2.
- Voltaje mínimo: 1 V.
- Rango: 10 metros.
- Consumo de potencia: 80 mA (conectado y transmitiendo).
- UART (serie): VCC,GND,TX,RX.
- Tasa UART: 57600 baudios.

5.2. Protocolo de comunicación (chip ThinkGear)

El protocolo de comunicación del caso Mindwave, esta instaurado en el **chip ThinkGear**. El chip ThinkGear se encarga de atenuar el ruido ambiental para poder realizar mediciones en áreas no aisladas eléctricamente. El sensor frontopolar recoge las señales, las envía al chip, y este las procesa y las convierte en una secuencia de datos digitales. Una vez filtradas las interferencias las ondas cerebrales se amplifican y son cotejadas con los algoritmos que se encuentran codificados en la memoria del chip.

Datos ThinkGear

POOR_SIGNAL Quality

Se trata de un byte sin signo que describe como de "pobre" es la señal medida por el ThinkGear. El rango de valores es de 0 a 200; cualquier valor distinto de cero indica que hay ruido y si el valor es 200, quiere decir que no existe conexión entre el sensor y la piel.

Este valor es enviado cada segundo, está activo por defecto y su frecuencia de muestreo es 1 Hz. La calidad de la señal puede estar modificada por varias causas:

- Los contactos del sensor o la masa no están en contacto con la frente o la oreja respectivamente.
- Existe un contacto pobre entre el sensor y la piel.
- Excesivo movimiento del usuario
- Ruido electroestático ambiental
- Ruido generado por otras medidas bioeléctricas: EMG, EoG…

En cualquier caso, *ThinkGear* cuenta con un filtrado y un algoritmo para la medición de los valores de *Attention* y *Meditation*, que permiten detectar, corregir, compensar y tolerar muchos de estos tipos de error.

eSense Meters

Cuenta con dos valores: ***Attention*** y ***Meditation***, los cuales se recogen en una escala de 0 a 100. Cerca de cero tanto la atención como la meditación son bajas, y cerca de 80 a 100 son altas. El valor cero significa que el sensor no es capaz de calcular un valor (normalmente se da al principio, durante la calibración). Los valores de *attention* y *meditation* se actualizan cada segundo.

La razón de que el rango sea tan grande es debido a que el algoritmo que realiza estos cálculos cuenta con un aprendizaje dinámico para ajustarse a cada usuario.

Como la aplicación se basará en el algoritmo eSense, se desarrollará este tema en el apartado 5.3 del casco Mindwave.

RAW Wave Value

Está formado por dos bytes y representa una única muestra de la señal RAW. Se trata de un entero con signo, dentro del rango -32768 a 32767. El primer byte representa la parte alta y el segundo byte la parte baja.

Por defecto este valor también se encuentra siempre activo y se recoge 512 veces por segundo, una muestra cada 2ms.

ASIC_EEG_POWER

Representa el valor de 8 tipos de ondas cerebrales. Esta información va recogida en 8 grupos de 3 bytes, enteros con signo y en formato little-endian. Los 8 tipos de ondas cerebrales se recogen en el siguiente orden:

- Delta (0.5 – 2.75 Hz)

- Theta (3.5 – 6.75 Hz)
- Low-alpha (7.5 – 9.25 Hz)
- High-alpha (10 – 11.75 Hz)
- Low-beta (13 – 16.75 Hz)
- High-beta (18 – 29.75 Hz)
- Low-gamma (31 – 39.75 Hz)
- High-gamma (41 – 49.75 Hz)

Estos valores no tienen unidades y solo tienen sentido comparándolos con los demás o entre ellos mismos. Se recogen una vez por segundo y están activados por defecto.

BLINK STRENGTH

Este valor indica la intensidad del parpadeo de ojos del usuario. Es un byte cuyo rango va de 1 a 255. Se recoge cada vez que existe un parpadeo. El dato se actualiza cada segundo.

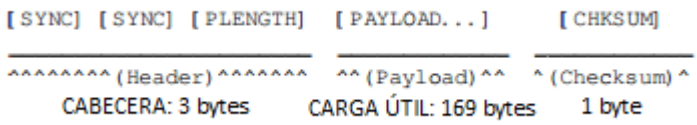
Paquetes ThinkGear

Los componentes ThinkGear envían los datos mediante un flujo de bytes en serie asíncrono. En el siguiente apartado se va a desarrollar una explicación de los diferentes campos que representan los ThinkGear Packets, con el fin de extraer e interpretar la información transmitida.

Estructura del paquete

Los paquetes se mandan en un flujo serie asíncrono de datos. El medio de transporte puede ser la UART, puerto serie COM, USB, Bluetooth, ficheros o cualquier otro mecanismo capaz de transportar bytes.

La estructura del paquete es la siguiente:

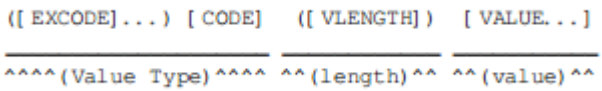


El tamaño mínimo del paquete es de 4 bytes (Header: 3 bytes, Payload: 0 bytes, Checksum: 1 byte), y el tamaño máximo 173 bytes (Header: 3 bytes, Payload: 169 bytes, Checksum: 1 byte).

- [SYNC]: 2 byte. Indica el comienzo de un nuevo paquete.

- [PLENGTH]: 1 byte. Indica la longitud de carga útil del paquete. Valores entre 0 y 169 (bytes de payload) como máximo. Cualquier valor mayor de 169 significa ERROR.

- [PAYLOAD]: carga útil, información. Este campo está formado por la siguiente estructura denominada **DATAROW**.



1. [EXCODE]: se utiliza para indicar el nivel de extensión de código utilizando el byte 0x55. Nivel 1 un byte de 0x55, nivel 2, dos byte de 0x55.
2. [CODE]: identifica el tipo de dato codificado. Se usa para enviar múltiples datos o para enviar datos que no tienen cabida en una sola trama.
3. [VLENGTH]: indica la longitud del dato.
4. [VALUE]: valor del dato definido por [CODE].

- [CHKSUM]: 1byte. Se emplea para verificar la integridad de la carga útil de los paquetes. El Checksum se calcula de la siguiente forma:
    1. Se suman todos los bytes del paquete [PAYLOAD].
    2. Se toman los 8 bits de menor peso de la suma.
    3. Se realiza la inversa de los 8 bits.

Si el Checksum calculado coincide con el Checksum del paquete, el receptor puede comenzar a analizar la carga útil de datos; si no coincide, el paquete se descarta.

A continuación, en la tabla 5.1, se plasma un breve resumen de los códigos del ThinkGear:

TABLA DE CÓDIGOS			
EXCODE	**CODE**	**VLENGTH**	**VALUE**
-	0x02	-	POOR_SIGNAL Quality (0 a 255)
-	0x04	-	ATTENTION eSense (0 a 100)
-	0x05	-	MEDITATION eSense (0 a 100)
-	0x16	-	Blink Strength (0 a 255)
-	0x80	2	RAW Wave Value (-32768 a 32767)
-	0x83	24	ASIC_EEG_POWER
-	0x55	-	[EXCODE]
-	0xAA	-	[SYNC]

Tabla 5.1 *Tabla de códigos del chip ThinkGear*

Paquete ejemplo:

```
Byte:    value  //  [CODE]   Explanation

[  0]: 0xAA   //   [SYNC]
[  1]: 0xAA   //   [SYNC]
[  2]: 0x20   //   [PLENGTH]   32 bytes
[  3]: 0x02   //   [POOR_SIGNAL]  Quality
[  4]: 0x00   //   Buena Señal (0/200)
[  5]: 0x83   //   [ASIC_EEG_POWER_INT]
[  6]: 0x18   //   [VLENGTH]   24 bytes
[  7]: 0x00   //   (1/3) Inicio bytes Delta
[  8]: 0x00   //   (2/3)
[  9]: 0x94   //   (3/3) Fin bytes Delta
[ 10]: 0x00   //   (1/3) Inicio bytes Theta
[ 11]: 0x20   //   (2/3)
[ 12]: 0x42   //   (3/3) Fin bytes Theta
[ 13]: 0x00   //   (1/3) Inicio bytes Low-alpha
[ 14]: 0x00   //   (2/3)
[ 15]: 0x0B   //   (3/3) Fin bytes Low-alpha
```

```
[ 16]: 0x00   //   (1/3) Inicio bytes High-alpha
[ 17]: 0x00   //   (2/3)
[ 18]: 0x64   //   (3/3) Fin bytes High-alpha
[ 19]: 0x00   //   (1/3) Inicio bytes Low-beta
[ 20]: 0x00   //   (2/3)
[ 21]: 0x4D   //   (3/3) Fin bytes Low-beta
[ 22]: 0x00   //   (1/3) Inicio bytes High-beta
[ 23]: 0x00   //   (2/3)
[ 24]: 0x3D   //   (3/3) Fin bytes High-beta
[ 25]: 0x00   //   (1/3) Inicio bytes Low-gamma
[ 26]: 0x00   //   (2/3)
[ 27]: 0x07   //   (3/3) Fin bytes Low-gamma
[ 28]: 0x00   //   (1/3) Inicio bytes Mid-gamma
[ 29]: 0x00   //   (2/3)
[ 30]: 0x05   //   (3/3) Fin bytes Mid.gamma
[ 31]: 0x04   //   [ATTENTION] eSense
[ 32]: 0x0D   //   Nivel de atencion de valor 13
[ 33]: 0x05   //   [MEDITATION] eSense
[ 34]: 0x3D   //   Nivel de meditacion de valor 61
[ 35]: 0x34   //   [CHKSUM]
```

Explicación paso a paso del *paquete ejemplo*:

1) Se lee la serie de bytes recibidos hasta que se encuentra uno de valor 0xAA[SYNC]

2) Se lee el segundo byte y se comprueba que es también 0xAA [SYNC]
   - Si no lo es, se vuelve al paso 1.
   - En caso contrario se continúa con el paso 3.

3) Se lee el siguiente byte correspondiente a PLENGTH:
   - Si tiene valor 170 (SYNC), se vuelve al paso 3.
   - Si tiene valor mayor de 170, se vuelve al paso 1, porque PLENGTH es demasiado largo.
   - Si tiene valor menor de 170, se continúa con el paso 4.

4) Se leen los bytes (indicados en PLENGTH) de la carga útil, PAYLOAD, y se almacenan en una tabla o vector. A su vez se suma cada byte que se lee a un acumulador (`checksum+=byte`), que se comprobará con el CHECKSUM al final.

5) Se cogen los últimos 8 bits del acumulador checksum y se invierten:
   ```
   checksum &=0xFF;
   checksum= ~checksum & 0xFF
   ```

6) Por último, se lee el siguiente byte de la cadena, el CHECKSUM:
   - Si el valor de CHECKSUM no coincide con el del acumulador, se ha producido un error y se descarta el paquete.
   - En otro caso, se considera como válido dicho paquete y se pasa a analizar la secuencia de la carga útil.
   - En cualquier paso, se vuelve al punto 1.

5.3. Algoritmo Esense

La empresa Neurosky ha creado un algoritmo denominado "**eSense**". Este algoritmo se compara con la señal que se ha obtenido con el electrodo que ha sido filtrada y amplificada. Esta comparación se hace con un rango de valores, por tanto, el valor eSense no corresponde a un valor numérico exacto.

La finalidad de este algoritmo es comprobar de manera cuántica la eficiencia que tiene el usuario cuando experimenta un estado de **atención** (concentración), o un estado de **meditación** (relajación). A base de entrenamiento pueden mejorarse el control y la meditación dedicando tiempo y esfuerzo.

Para cada parámetro, atención/meditación, se ha establecido una escala de valores relativa entre 1 y 100, donde 1 es el valor mínimo y 100 es el máximo, ya que el cero significa que el sensor no es capaz de realizar la medida. En esta escala de valores, un valor **entre 40 a 60**, se considera neutral y se relaciona con el concepto de "línea base" que se estable en las técnicas convencionales de medición de ondas.

Un valor **entre 60 y 80**, se considera levemente elevado, y puede empezar a interpretarse como indicador de uno de los estados. Entre los **80 y 100**, es un valor elevado, y ya no hay duda del estado experimentado.

Por el otro lado, el rango **entre 20 y 40** se considera un valor ligeramente bajo, interpretado como leve distracción. Mientras que los valores **entre 1 y 20** se estiman valores muy bajos, se relacionan con estados de distracción profunda, ansiedad o pensamientos errantes.

La razón para el empleo de estos amplios rangos para la interpretación de los resultados es que las señales recibidas del cerebro están sujetas a las reglas de varianza y sufren fluctuaciones por lo que es casi imposible obtener un valor exacto en las mediciones. Sin embargo, usando estos rangos se puede dar una interpretación a esos valores aunque varíen de forma dinámica.

Los valores umbrales y las fluctuaciones difieren en cada individuo. Por ello, gracias a algoritmos que actúan de forma dinámica y al mismo tiempo durante el procesado eSense, ajustando las fluctuaciones y las tendencias naturales de los usuarios, el chip ThinkGear es capaz de operar en una amplia gama de individuos bajo un amplio abanico de condiciones personales y ambientales, sin dejar de facilitar buena precisión y fiabilidad

6. Robot industrial ABB

ABB, siglas de Asea Brown Boveri, se crea en 1988 con la fusión de dos compañías: Asea, fundada en Estocolmo en 1883 y Brown Boveri, fundada en Zurich en 1891. ABB es una compañía dedicada a la ingeniería eléctrica y la automatización. En la actualidad tiene cuatro divisiones de negocio:

1. **Electrification Products**: buscan productos y soluciones adecuadas para múltiples aplicaciones eléctricas, incluyendo productos de control, conmutadores, sistemas de cableado.

2. **Discrite Automation and Motion**: proporcionan productos y soluciones que mejoran la productividad industrial y la eficiencia energética.

3. **Process Automation**: aportan soluciones para el control y optimización de industrias químicas, de gas, de petróleo.

4. **Power Grids**: buscan la eficiencia energética, mejorando la generación, el transporte y la distribución.

Además de todo lo mencionado anteriormente, ABB es un líder mundial en robótica. Ofrece a sus clientes todo tipo de robots industriales para la mejora y optimización de la productividad de las empresas (figura 6.1). Cuenta con un catálogo de unos 25 modelos de robot para diferentes prestaciones y necesidades en cada tipo de industria.

Figura 6.1 *Robots ABB en la producción en línea de chasis* [39]

6.1. Brazo robotizado IRB120

El brazo robotizado **IRB120** es el robot más pequeño de ABB, es muy flexible y compacto. Pesa unos 25 kg y puede manipular un peso desde 0,3 kg a 3 kg, incluso hasta 4 kg con la muñeca en vertical. Cuenta con un área de trabajo de 580mm. Este robot posee seis grados de libertad, los 3 primeros sirven para establecer la posición del efector final y los tres últimos para determinar la orientación del mismo.

En la figura 6.2 se observa que el robot IRB120 puede trabajar en diferentes posiciones dependiendo de las necesidades; en la figura 6.3 se aprecia el área de trabajo del robot.

Figura 6.2 *Brazo robotizado IRB120* [40]

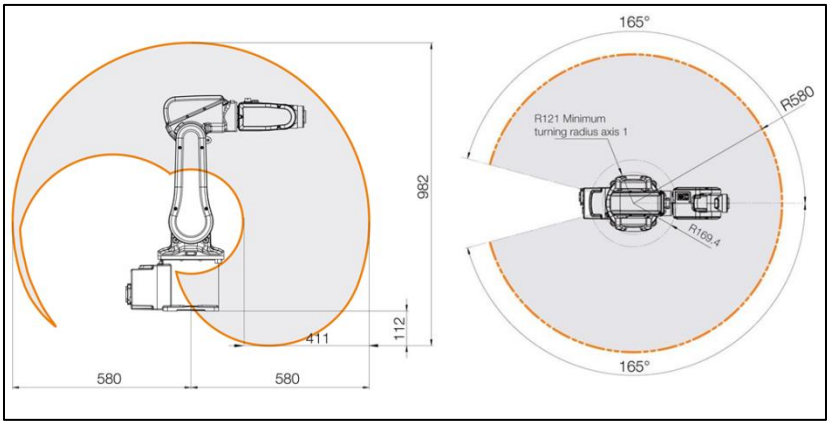


Figura 6.3 *Área de trabajo del robot IRB120* [16]

6.2. Software RobotStudio

ABB ha creado el software de simulación **RobotStudio** que permite crear, programar y simular células y estaciones de robots industriales ABB en 3D en un ordenador. Igualmente este simulador posibilita crear automáticamente cualquier tipo de estación e importar modelos 3D creados en otros programas. Tiene una gran facilidad de diseño y creación de células robóticas (robot y dispositivos). Además, permite exportar los resultados obtenidos en simulación a la estación real de tu robot.

Este software también permite simular el robot real, lo que concede muchas ventajas: reduces los riesgos, arranques más rápidos, menor tiempo para las modificaciones, aumento de la productividad.

El simulador RobotStudio (figura 6.4) funciona sobre **RobotWare**. Se trata de un software que contiene un conjunto de archivos necesarios para implementar todas las funciones (virtual-real), configuraciones, datos y programas necesarios para el control del sistema robot. [19]

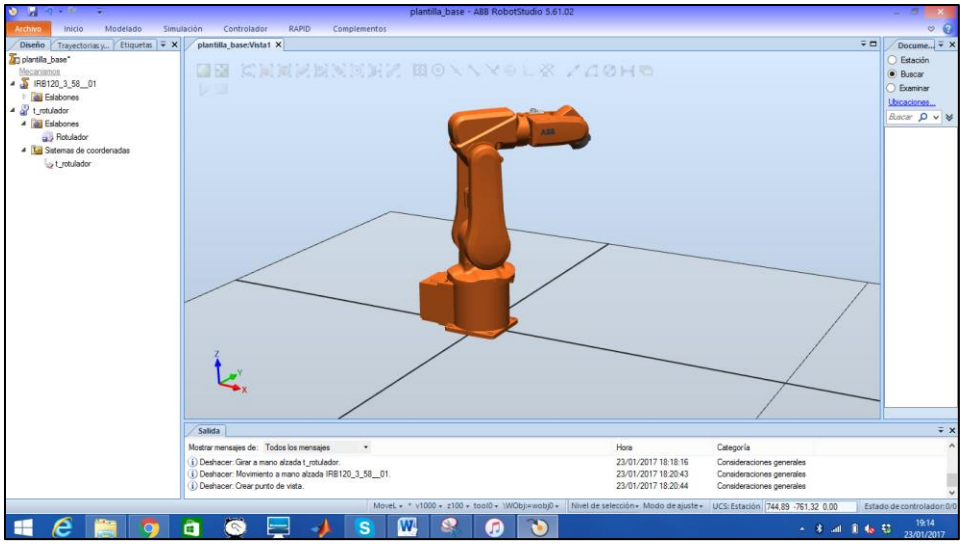


Figura 6.4 *Estación de trabajo del RobotStudio con el robot IRB120*

6.3. Lenguaje de programación RAPID

RAPID es el lenguaje de programación de alto nivel diseñado por la compañía ABB para el control de sus robots. Una aplicación basada en este lenguaje, consta de un programa y varios módulos. Un programa RAPID,contiene un conjunto de instrucciones que describen el comportamiento del robot, existen una serie de argumentos que son utilizados para definir el movimiento del robot.

El programa en sí, es un conjunto de instrucciones que constan de tres partes: **rutina** principal, **subrutinas** y **datos del programa** (figura 6.5).

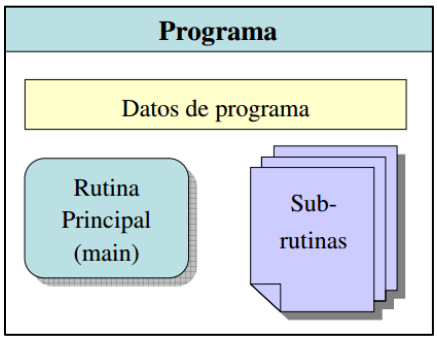


Figura 6.5 *Esquema básico de la estructura de un programa RAPID* [20]

- Una rutina principal (main): rutina donde se inicia la ejecución.

- Un conjunto de sub-rutinas: sirven para dividir el programa en partes más pequeñas a fin de obtener un programa modular (funciones).

- Los datos del programa: definen posiciones, valores numéricos, sistemas de coordenadas, etc.

6.4. Controlador IRC5 de ABB

Los controladores del robot industrial de ABB ofrecen un control de movimiento superior y permiten una rápida integración de hardware opcional. **El IRC5 es la quinta generación de controladores de robot de ABB** (figura 6.6). Su tecnología de control de movimiento, que combina TrueMove y QuickMove, es esencial para el rendimiento del robot en términos de precisión, velocidad, tiempos de ciclos, programación y sincronización con dispositivos externos. Entre otros elementos se incluyen el **FlexPendant** (figura 6.7) que consiste en una unidad de programación con pantalla táctil y movimiento del robot con joystick, lenguaje RAPID flexible y potentes opciones de comunicación.

Figura 6.6 *Controlador IRC5 Compact* [41]

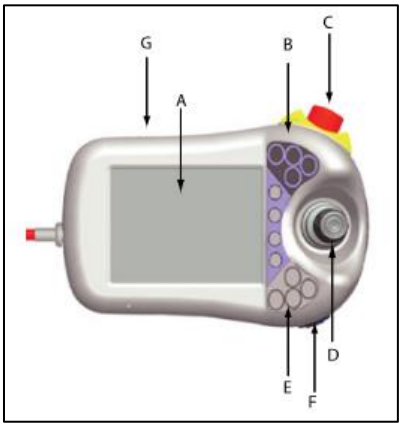


Figura 6.7 *Unidad Flexpendant* [41]

En la tabla 6.1, se resumen las partes de mayor relevancia del flexPendant.

TABLA FlexPendant	
POSICIÓN	**DESCRIPCIÓN**
A	Pantalla
B	Teclas programables
C	Botón paro de emergencia
D	Joystick
E	Teclas de ejecución de programas
F	Conexión de memoria portátil USB
G	Alojamiento de puntero

Tabla 6.1 *Partes del Flexpendant* [42]

De forma general, la familia de controladores IRC5 se compone de los siguientes módulos:

- **Módulo de accionamiento**: contiene el sistema de accionamiento.

- **Módulo de control**: contiene el ordenador principal, el panel del operador, el interruptor principal, las interfaces de comunicaciones, la conexión para el FlexPendant, los puertos de servicio y cierto espacio para los equipos del cliente, como por ejemplo tarjeta de E/S de ABB. El controlador también contiene el software del sistema, es decir RobotWare-OS, que incluye todas las funciones básicas de manejo y programación, sobre el cual, además, se pueden instalar un gran número de opciones con funcionalidad adicional. Por ejemplo, multitarea, transferencia de información de archivos al robot, comunicación con sistemas externos o tareas de movimiento avanzadas.

Asimismo, el controlador IRC5, mediante la tecnología **SafeMove**, marca un paso importante al eliminar el aislamiento de los robots industriales y facilitar la colaboración hombre-robot, siendo desarrollado y probado para cumplir con las normas de seguridad internacionales, garantizando un movimiento del robot seguro y predecible.

7. Sistema BCI “Control del robot IRB120 mediante Casco Mindwave”

El objetivo de este proyecto es crear un sistema BCI (figura 7.1). Esta aplicación se basa en la capacidad del usuario de modificar la actividad cerebral de forma voluntaria, diferenciando dos estados mentales: la atención y la meditación (algoritmo eSense), dando lugar a resultados binarios. Se tiene un tablero con dos piezas, una roja y una azul, asignándose un color a cada estado mental: el **rojo** a la **atención** y el **azul** a la **meditación**. Dependiendo del estado mental del usuario en ese momento, el robot IRB120 cogerá la pieza del color que corresponda, previa validación del usuario por medio del parpadeo y procesamiento de imágenes para determinar las coordenadas de la pieza.

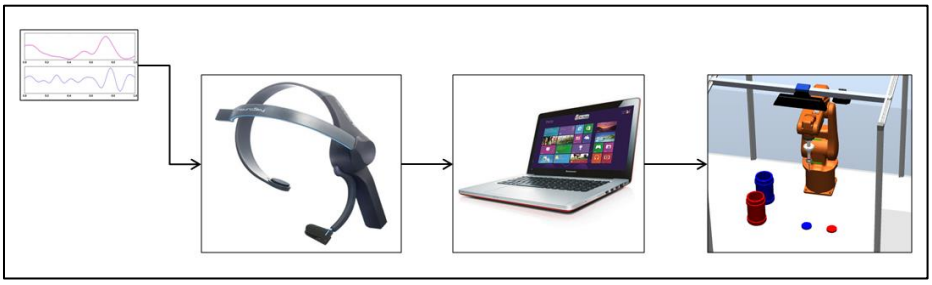


Figura 7.1 *Sistema Brain Computer Interface*

El diagrama de la aplicación será el mostrado en la figura 7.2:

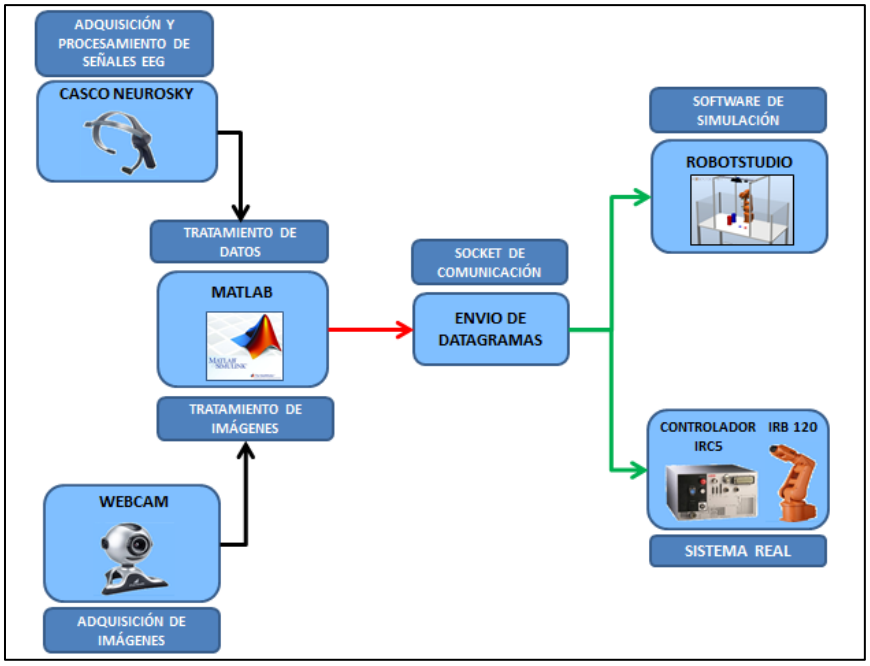


Figura 7.2 *Esquema de la arquitectura general del sistema*

La adquisición de datos se realiza con el casco Mindwave y los datos se procesan con Matlab en tiempo real. Se obtienen los datos de la atención y la meditación del usuario, además del parpadeo para la validación del color.

La adquisición de imágenes se lleva a cabo por una cámara web que está conectada al ordenador, la cual nos permite obtener capturas en **tiempo real**. Las imágenes obtenidas se transfieren directamente a la aplicación desarrollada en Matlab.

Posteriormente se realiza en Matlab el tratamiento de las imágenes que esto consiste en modificar ciertas características de la imagen para obtener información de interés. Una vez procesadas las imágenes, se procede a extraer las características de los objetos y se calculan sus coordenadas (centroides). Mediante la calibración se evalúa la relación pixel-mm para obtener las coordenadas en milímetros.

La información desde Matlab se envía al robot de ABB por medio del socket de comunicación TCP/IP en forma de datagrama, para simularlo en RobotStudio y/o cargarlo en el controlador IRC5 y así, ejecutarlo sobre el robot real.

7.1. Módulo Casco Mindwave de Neurosky

Esta es la primera y más importante parte de la aplicación. Se utiliza el casco Mindwave de Neurosky para captar la actividad cerebral del usuario, diferenciando dos estados mentales: la atención y la meditación. Esta parte del proyecto se centra en como captar esos estados y el parpadeo con Matlab, procesarlos y representarlos en tiempo real y en la capacidad del usuario para modificar la actividad cerebral de forma voluntaria.

Con el objetivo de familiarizarse con el dispositivo MindWave se han utilizado y estudiado las siguientes **aplicaciones** gratuitas disponibles en la página web de Neurosky:

- Math trainer.
- Brainwave Visualizer.

7.1.1. Aplicaciones Neurosky

Math trainer

Esta aplicación permite entrenar la habilidad aritmética para llegar a ser más preciso y eficiente. Una vez finalizada una prueba el programa devuelve una puntación calculada en base a los aciertos y fallos obtenidos en las distintas operaciones matemáticas.

Las operaciones a realizar pueden ser sumas, restas, divisiones o multiplicaciones. Se puede comprobar también con cuales de estas operaciones se tienen más problemas, dado que proporcionan el número de resultados correctos e incorrectos.

Aparece un gráfico que muestra la capacidad de atención del usuario independientemente de los errores o aciertos cometidos. Como se observa en las figuras 7.3 y figura 7.4, en la primera prueba, el resultado es más alto que el obtenido en la segunda prueba. Sin embargo, el usuario estaba más concentrado en la segunda prueba, ya que la gráfica de la atención tiene unos valores más altos.

Figura 7.3 *Resultado 1 prueba "Math Trainer"*

Figura 7.4 *Resultado 2 prueba "Math Trainer"*

Brainwave Visualizer

Se trata de una aplicación interactiva controlada por las ondas cerebrales que muestra una representación gráfica de la actividad cerebral (figura 7.5). Incluye un visualizador de ondas cerebrales y medidores del nivel de atención y meditación. También existe la posibilidad de escuchar música y observar cómo responde nuestro cerebro a determinadas melodías.

Figura 7.5 *Ejemplo aplicación "Brainwave Visualizer"*

Dentro de esta aplicación hay un juego basado en la atención. Cuando consigues llegar al valor máximo de atención, el barril explota (figura 7.6). Esta aplicación te permite mejorar la atención y llegar al valor máximo, cada vez en menor tiempo.

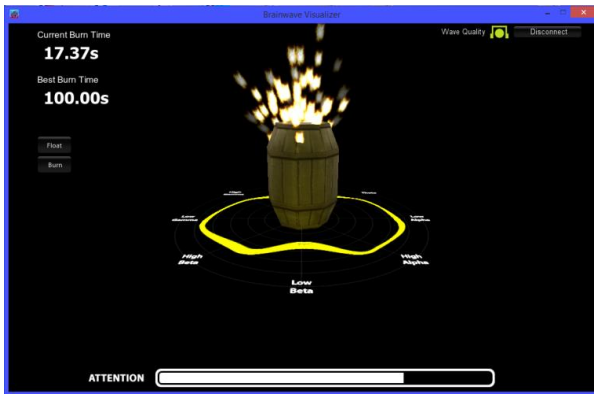


Figura 7.6 *Aplicación basada en la atención*

7.1.2. Algoritmo eSense: Atención y Meditación.

Como se ha visto a lo largo del proyecto, el casco Mindwave de Neurosky, permite captar y procesar la señal cerebral "cruda" raw , contiene los datos tal y como han sido adquiridos desde el sensor. Esta señal se desglosa en: alpha, tetha, gamma, beta. El casco permite captarlas individualmente, y también los estados mentales de atención y meditación, gracias al algoritmo eSense y el parpadeo del usuario.

En la siguiente figura (figura 7.7) se representan todas las ondas cerebrales individuales, en una prueba de 60 segundos.

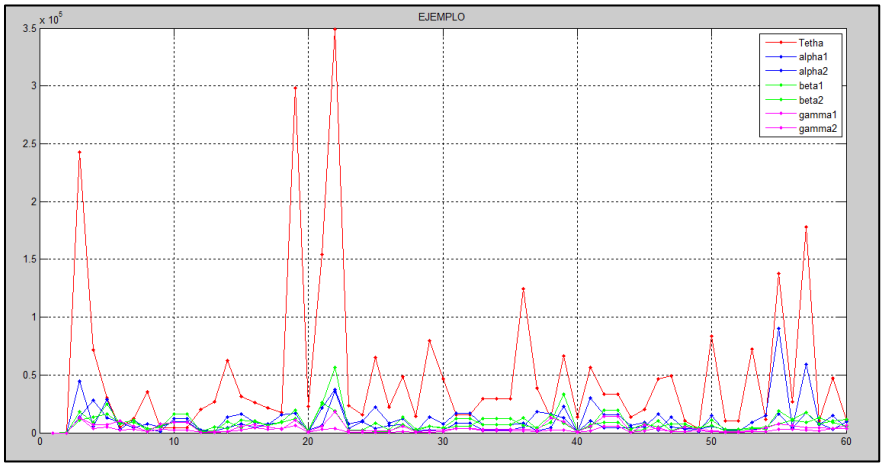


Figura 7.7 *Ondas cerebrales: tetha, alpha, beta, gamma*

En la siguiente figura (figura 7.8) se representa la atención, meditación y la señal raw (suma de las ondas cerebrales) del usuario en una prueba de 60 segundos de duración.

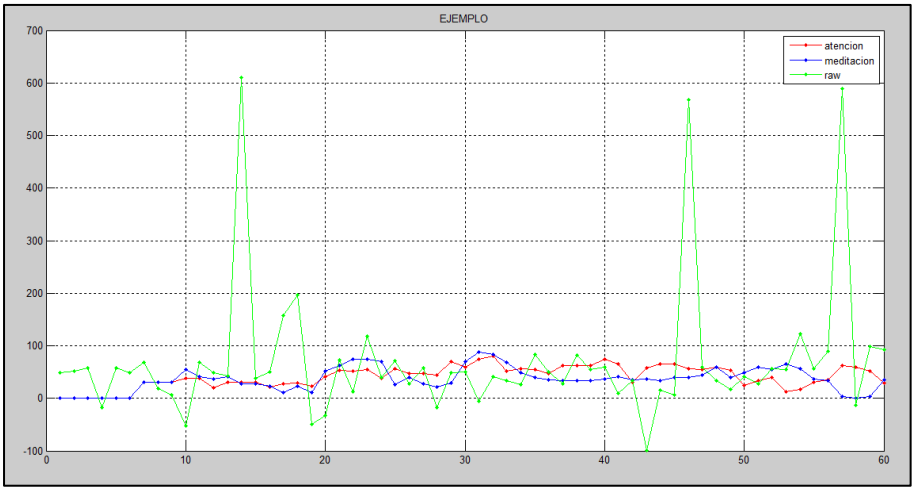


Figura 7.8 *Señal raw y los estados cognitivos: atención y meditación*

Los sistemas BCI **endógenos** se basan en la capacidad del hombre para modificar su actividad cerebral de forma voluntaria. Este tipo de BCI requiere un **entrenamiento** para mejorar el control. Los sistemas endógenos se fundamentan en **establecer dos estados**, si-no, delante-atrás, etc, por lo que, en este caso, se ha buscado una respuesta binaria. Una opción sería coger una de las señales cerebrales, estudiarla, filtrarla y establecer cuando se está pensando en el color rojo o en el color azul, pero se presentan dos problemas. El primero es que al tener solo un sensor en la parte frontal, no proporciona información suficiente para extraer resultados concluyentes, ya que un estudio electroencefalográfico adecuado necesita un mínimo de diez canales, veinte electrodos.

El segundo problema es que el sistema de reconocimiento de colores, está en la parte posterior del cerebro, en el lóbulo occipital, por lo que con el sensor del casco es imposible captar ese dato.

Por estas razones, se ha optado por los dos estados mentales que proporciona el **algoritmo eSense**, son señales filtradas y más estables, y aunque sufren fluctuaciones, se puede buscar una **respuesta binaria,** relacionando la **atención** con el **color rojo** y la **meditación** con el **color azul**, y así desarrollar el proyecto.

Estos dos datos del algoritmo eSense se nutren de las ondas cerebrales, ya que la **atención** se basa en las **ondas beta** y **gamma** y la **meditación** en las ondas **alpha** y **tetha**. Son señales ya filtradas y trabajadas, que nos dan un resultado exacto en un rango de valores de 0 a 100.

El **medidor eSense para la atención** indica la intensidad del nivel de un usuario de "foco" mental o "atención", tal como la que se produce durante la concentración intensa y dirigida. Su rango de valores va de 0 a 100. Las distracciones, pensamientos errantes, falta de concentración o ansiedad, pueden disminuir el nivel de atención. [43]

Generalmente, la atención se puede controlar a través de un enfoque visual como por ejemplo, centrarse en una idea singular. Una opción puede ser la de elegir un punto de la pantalla para focalizar e intentar imaginar que la acción que el usuario está tratando de lograr está sucediendo. Por ejemplo, mirar el medidor de atención eSense e imaginar que el marcador avance hacia números más altos.

Recomendaciones del fabricante que favorecen el control sobre los valores de atención:

- Identificar y mantener un pensamiento concreto.
- Enfocar la mirada en un objeto concreto.
- Concentrarte en algo que te gusta.
- Hacer un cálculo matemático.
- Dar un discurso.
- Cantar una canción silenciosamente.
- Imaginar una acción que estas tratando de que se cumpla.

A modo de resumen, los rangos de la atención se recogen en la siguiente tabla (tabla 7.1):

VALORES ATENCIÓN	
RANGO DE VALORES	**INTERPRETACIÓN**
0 - 20	Valor muy bajo. Se interpreta como estado de distracción profunda, ansiedad, y pensamientos errantes.
20 - 40	Se considera un valor ligeramente bajo. Interpretado como una leve distracción.
40 - 60	Valor neutral
60 - 80	Se interpreta como un estado de concentración aceptable, un valor ligeramente elevado.
80 - 100	Estado de concentración máxima. Se considera un valor muy elevado.

Tabla 7.1 *Interpretación rango de valores de la atención*

El **medidor eSense de Meditación** indica el nivel de "calma" mental de un usuario o de "relajación". Oscila entre 0 y 100. Se ha de tener en cuenta que la meditación es una medida de los estados mentales de una persona, no a nivel físico, por lo que simplemente relajar todos los músculos del cuerpo no conlleva inmediatamente un mayor nivel de meditación.

Sin embargo, para la mayoría de personas, en la mayoría de las circunstancias normales, relajar el cuerpo suele ayudar a la mente a relajarse. La meditación está relacionada con una actividad reducida por los procesos mentales activos en el cerebro. [43]

Se ha comprobado que el efecto de cerrar los ojos apaga las actividades mentales del procesado de imágenes a través de los ojos. Así que cerrar los ojos a menudo es un método eficaz para aumentar la meditación. Las distracciones, pensamientos errantes, ansiedad, agitación, y los estímulos sensoriales pueden reducir los niveles del medidor de meditación.

A continuación, algunas recomendaciones del fabricante para alcanzar el estado de meditación:

- Respirar y exhalar lento y profundo.
- Relajar todos los músculos.
- Dejar la mente en blanco.
- Cerrar los ojos.
- Imaginar que estas en la cama a punto de irte a dormir.
- Imaginar que estas flotando sobre el agua caliente.

A modo de resumen, los rangos de la meditación se recogen en la siguiente tabla (tabla 7.2):

VALORES MEDITACIÓN	
RANGO DE VALORES	**INTERPRETACIÓN**
0 - 20	Valor muy bajo. Se interpreta como estado consciente, activo, ausencia total de relajación.
20 - 40	Se considera un valor ligeramente bajo. Interpretado como una leve relajación.
40 - 60	Valor neutral.
60 - 80	Se interpreta como un estado de meditación aceptable, un valor ligeramente elevado
80 - 100	Estado de meditación máxima, cercano al sueño. Se considera un valor muy elevado.

Tabla 7.2 *Interpretación rango de valores de la meditación*

Hay que tener en cuenta que las señales recibidas del cerebro están sujetas a las reglas de varianza y sufren fluctuaciones, por lo que es casi imposible obtener un valor exacto en las mediciones. Sin embargo, usando los rangos (0-20, 20-40, 40-60, 60-80, 80-100) se puede dar una interpretación a esos valores aunque varíen de forma dinámica.

Cabe destacar que los valores umbrales y las mencionadas fluctuaciones pueden diferir en cada individuo con respecto a otros, por tanto no se debe esperar obtener los mismos valores en pruebas con las mismas condiciones, aunque si serán resultados bastante similares. Además, la capacidad de concentración o de meditación varía también de unos individuos a otros.

7.1.3. Funcionamiento algoritmo eSense

A continuación, se procede a realizar la primera prueba (figura 8.9). El sujeto que se somete al test no está realizando ningún tipo de ejercicio de relajación ni de concentración. El test se ha realizado durante 40 segundos, por lo que se recogerán 40 muestras, una muestra por segundo como describe el datasheet.

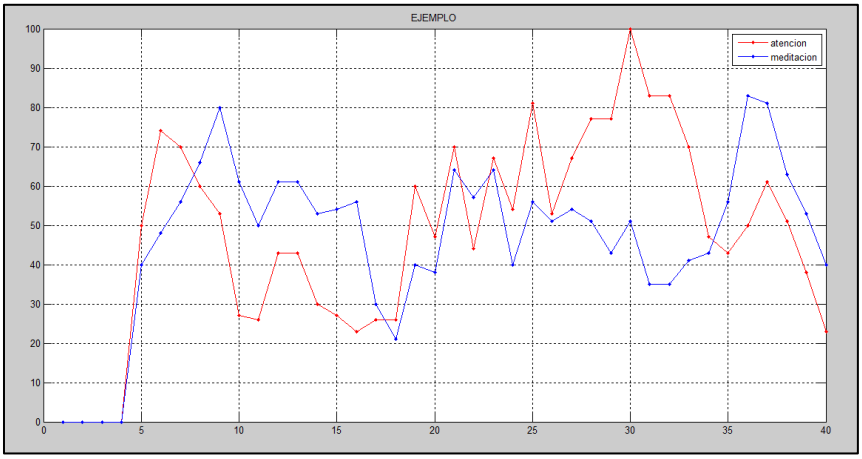


Figura 7.9 *Señal de atención y meditación*

Como se observa en el gráfica anterior (figura 7.9), los datos son completamente aleatorios ya que el sujeto se encontraba en una situación en la que no intentaba inferirse ningún estado cognitivo concreto. Lo que se observa es que los datos fluctúan mucho, esto habrá que tenerlo en cuenta en siguientes pruebas y en la aplicación final.

En la siguiente prueba el usuario está realizando un ejercicio de **concentración**, fijando su atención en el medidor de atención eSense e imaginando que la gráfica aumenta de valor. Los resultados de atención y meditación son los siguientes (figura 7.10).

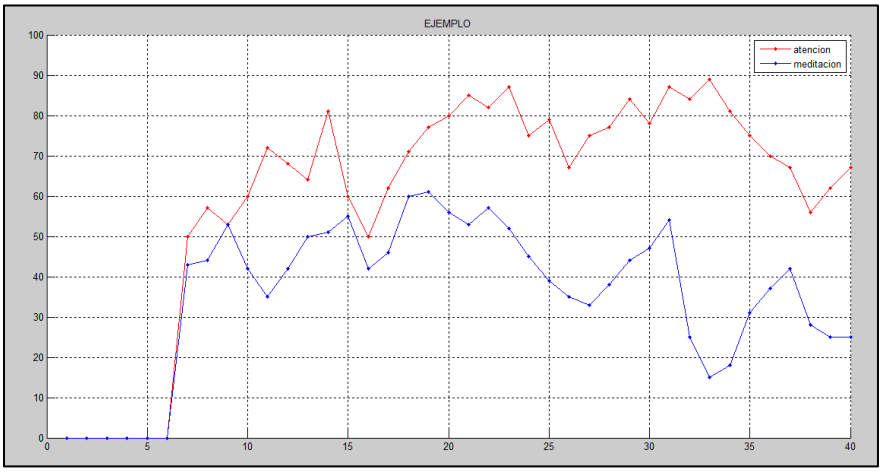


Figura 7.10 *Valores de atención y meditación durante la prueba*

En este caso, se aprecia claramente como el sujeto, debido a la concentración, eleva los niveles de atención y disminuye los de la meditación. Aun así se considera un estado de concentración relativamente bajo, ya que en ningún momento el sujeto llega a niveles entre 90 – 100.

También se puede observar que no hay una regularidad entre los valores ya que varían mucho a lo largo del tiempo, esto mejorará con el **entrenamiento**.

Como se ha hecho con la atención, en la siguiente prueba, se busca que el sujeto se **relaje** y entre en un estado de **meditación**. Para ello, se crea un escenario de tranquilidad, se apaga la luz, se pone música relajante. Una vez creado el ambiente el usuario cerrara los ojos, respirará profundamente e intentará dejar la mente en blanco. Hay que destacar que la persona que realiza la prueba no ha practicado nunca ningún tipo de meditación. Los resultados de la atención y la meditación son los siguientes (figura 7.11):

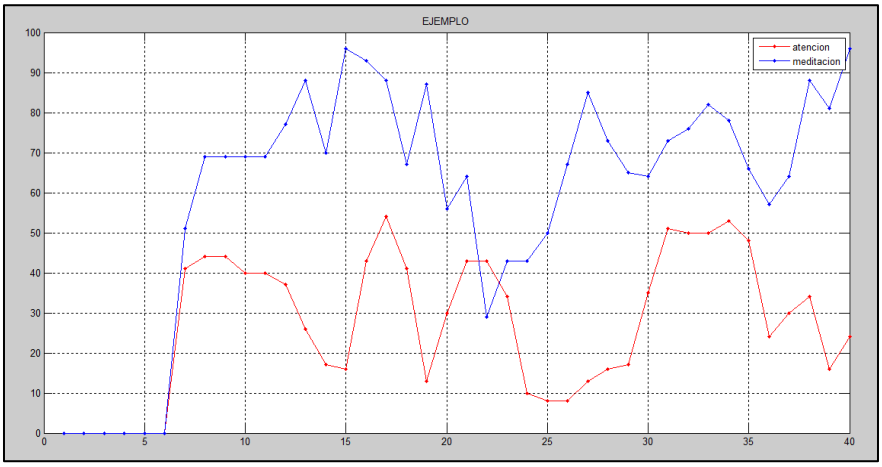


Figura 7.11 *Valores de meditación y atención durante la prueba*

Comparando ambas pruebas (figura 7.10 y 7.11), puede comprobarse que el usuario consigue cierta predominación del estado cognitivo, sin llegar a controlarlo en su totalidad. En los resultados se observa como los valores son elevados, pero sin llegar, en ninguno de los casos, al valor máximo. El individuo no tuvo tiempo suficiente para llegar a dominar el estado deseado y se puede observar cómo se mantenían niveles altos durante una cierta cantidad de tiempo y posteriormente estos valores disminuían de forma drástica.

Como ya se ha indicado, estos sistemas **endógenos** mejoran con el **entrenamiento**. Por esto motivo, se ha sometido al usuario a diferentes ejercicios para mejorar la atención y meditación media hora diaria durante un mes. Estos ejercicios consistían en realizar cálculos matemáticos, mantener un pensamiento concreto, fijar la mirada en un pensamiento concreto, imaginar una acción. Con el paso de los días, se pudo observar claramente como el sujeto tenía más facilidad para controlar sus niveles de atención. En el caso de la meditación, las mejoras tardaron más tiempo en visualizarse, pero al final el usuario era capaz de llegar al valor máximo de meditación y mantenerlo. Las actividades consistían en cerrar los ojos, intentar dejar la mente en blanco, respirar y exhalar lenta y profundamente.

Después de cierto tiempo de entrenamiento, el usuario fue capaz de elevar los niveles del medidor de **atención** eSense de manera voluntaria.

En la siguiente prueba se pretendió mantener los valores del medidor de la atención en el nivel más alto posible. Durante la prueba, el sujeto fija su atención en el medidor de atención eSense e imagina que va aumentando su valor, y cuando llega a 100 se mantiene. Los resultados son los siguientes (figura 7.12):

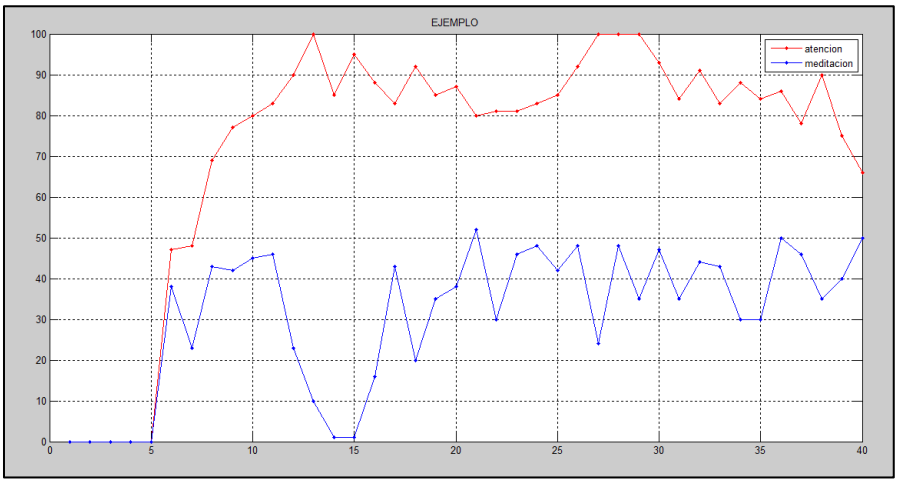


Figura 7.12 *Valores de atención y meditación después del entrenamiento*

Tal como se observa, el usuario ha sido capaz de controlar los niveles de atención. Se comprueba claramente la diferencia entre la primera gráfica y esta. Hay muchas menos fluctuaciones y existen tramos donde los valores eran máximos. Estos resultados se podrán usar sin problemas para el sistema, como respuesta binaria.

En la **siguiente prueba** se pretendió mantener los valores del medidor de la **meditación** también en el nivel más alto posible. Para conseguirlo, el usuario relajó su cuerpo, respiró lenta y profundamente e intentó dejar la mente en blanco. En la siguiente gráfica (figura 7.13) se observa el resultado:

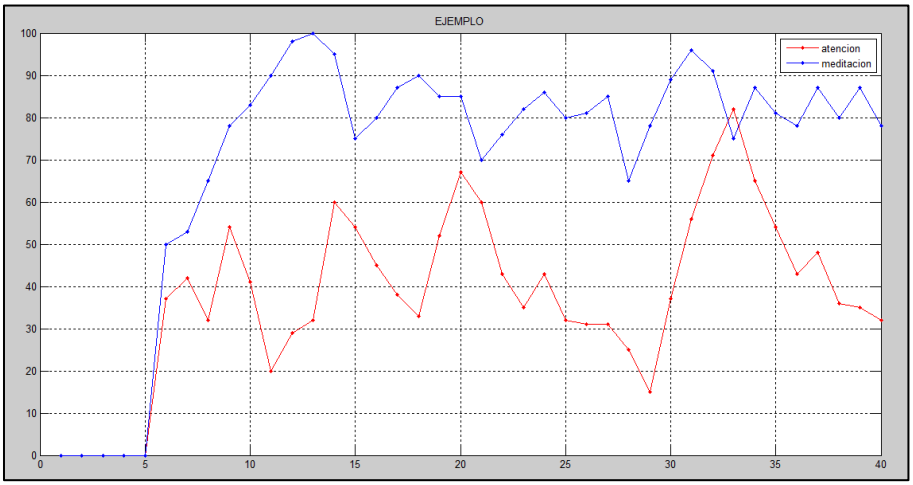


Figura 7.13 *Valores de meditación y atención después del entrenamiento*

El usuario es capaz de controlar el estado de meditación, de manera bastante constante a lo largo del tiempo. Al comparar las dos gráficas se comprueba que al usuario le supone más esfuerzo la meditación y es menos estable. Esta será la otra referencia binaria en el sistema.

Lo que se demuestra es que, debido a la **neuroplasticidad**, el cerebro tiene capacidad para formar nuevas conexiones nerviosas, a lo largo de nuestra vida, en respuesta a una información nueva, de nuestro cerebro y que como si de un músculo se tratase, es capaz de mejorar el rendimiento y el control de este a través de la práctica. En este caso se generan redes neuronales nuevas, influidas por una motivación externa, y al ser utilizadas de manera continua resulta mucho más fácil acceder a ellas una vez inducido el estímulo.

El código siguiente es el utilizado para realizar todas las pruebas anteriormente descritas, está basado en el ofrecido por la empresa Neurosky:

```
%Buffer
data_att = zeros(60,2);
%PUERTO
portnum1 = 4;
comPortName1 = sprintf('\\\\.\\COM%d', portnum1);
%Velocidad
TG_BAUD_57600  = 57600;
%Formato de datos para TG_Connect() and TG_SetDataFormat().
TG_STREAM_PACKETS =     0;
% Tipo de datos que devuelve TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_DELTA =           5;
TG_DATA_THETA =           6;
TG_DATA_ALPHA1 =          7;
TG_DATA_ALPHA2 =          8;
TG_DATA_BETA1 =           9;
TG_DATA_BETA2 =          10;
TG_DATA_GAMMA1 =         11;
TG_DATA_GAMMA2 =         12;
TG_DATA_BLINK_STRENGTH = 37;

% cargar thinkgear dll
loadlibrary('ThinkGear.dll');
fprintf('ThinkGear.dll loaded\n');
dllVersion = calllib('ThinkGear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version: %d\n', dllVersion );
% Obteniendo un identificador de ID de conexión a ThinkGear
connectionId1 = calllib('ThinkGear', 'TG_GetNewConnectionId');
    if ( connectionId1 < 0 )
        error( sprintf( 'ERROR: TG_GetNewConnectionId() returned
%d.\n', connectionId1 ) );
    end;
% Intentando conectar el identificador de la conexión al puerto serie
"COM3"
errCode = calllib('ThinkGear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
```

```
    if ( errCode < 0 )
        error( sprintf( 'ERROR: TG_Connect() returned %d.\n', errCode
) );
    end

fprintf( 'Connected.  Reading Packets...\n' );
    if (calllib('ThinkGear','TG_EnableBlinkDetection',
connectionId1,1)==0)
      disp('Blinkdetection');
    end
i=0;
j=0;
%To display in Command Window
disp('Reading Brainwaves');
t = timer('TimerFcn', 'stat=false; disp(''FIN'')','StartDelay',40);
errCode = calllib('ThinkGear', 'TG_EnableAutoRead',  connectionId1,-1
);
start(t)
stat=true;
while (stat==true)
if (calllib('ThinkGear','TG_ReadPackets',connectionId1,1) == 1)   %if
a packet was read...
   if
(calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_POOR_SI
GNAL) ~= 0) %leer las señales cada segundo
    j = j + 1;
    %leer atencion/meditacion.
    data_att(j,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
    data_att(j,2) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
    figure(1)
    hold on
    plot(data_att(:,1),'-r.','Linewidth',1)
    plot(data_att(:,2),'-b.','Linewidth',1)
    title('ATENCION/MEDITACION')
    legend('atencion','meditacion')
  end
end
end
csvwrite('C:\Users\laura\Documents\MATLAB\TFG\data.csv',data_att);
dlmwrite('C:\Users\laura\Documents\MATLAB\TFG\data.csv',data_att,'prec
ision', '%.6f');
%To display in Command Window
disp('Loop Completed')
%Release the comm port
calllib('ThinkGear', 'TG_FreeConnection', connectionId1 );
```

Como recoge el datasheet del chip ThinkGear, los valores de atención y meditación se actualizan cada segundo, frecuencia de muestreo de 1 Hz. Y como se observa en el código anterior, el captar estos estados depende del dato TG_DATA_POOR SIGNAL. Este valor es enviado cada segundo, tiene un rango de valores entre 0 y 200, cualquier valor distinto de cero indica que hay ruido y que se pueden captar señales.

Por lo que, cuando el valor **TG_DATA_POOR_SIGNAL** es distinto de cero, se recibe un valor de la atención y de la meditación, cada segundo.

Para que la aplicación sea más dinámica, como se verá en capítulos posteriores, se ha cambiado la condición para captar los estados de atención y meditación. En vez de depender de un dato que se actualiza cada segundo, se va a depender de la señal raw **TG_DATA_RAW**. Esta señal tiene una frecuencia de muestreo de 512 Hz, un valor cada 2 ms. Aunque la frecuencia de muestreo de la atención y la meditación no va a cambiar, aparecerán más muestras por segundo.

La parte del código modificada es la mostrada a continuación:

```
while (stat==true)
if (calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%leer las señales
    j = j + 1;
    %leer atencion/meditacion.
    data_att(j,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
    data_att(j,2) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);

 end
end
```

Teóricamente hablando, deberían aparecer, por cada valor nuevo de atención y meditación, 512 muestras iguales. Realizando varias pruebas, se observa que los 100-125 primeros datos son nulos, se entiende que forma parte de la calibración del sistema. Una vez calibrado el sistema, se empiezan a recopilar distintos valores, **cada 10-12 muestras se actualiza el dato** (figura 7.14). El número de muestras por dato es bastante inferior al esperado y esto es consecuencia de varias causas. La causa principal es que la condición del "TG_DATA_RAW ~=0" no siempre se cumple. En menor medida afectan otras causas como la cantidad de líneas de código, que durante la ejecución del programa se cargan imágenes y que se van mostrando los datos en tiempo real. Todos estos factores hacen que la velocidad disminuya.

	A	B						
127	43	44	137	45	53	147	40	41
128	43	44	138	45	53	148	40	41
129	43	44	139	45	53	149	40	41
130	43	44	140	45	53	150	40	41
131	43	44	141	45	53	151	40	41
132	43	44	142	45	53	152	40	41
133	43	44	143	45	53	153	40	41
134	43	44	144	45	53	154	40	41
135	43	44	145	45	53	155	40	41
136	43	44	146	45	53	156	40	41

Figura 7.14 *Los datos de data_att de atención y meditación, 10 muestras/dato*

Con todo ello, se ha reescrito el programa anterior, teniendo en cuenta que las **primeras 100 muestras no aportan valor**, y considerando **un dato nuevo cada 10 muestras**. Se considerará un dato **atención** cuando este en el **rango de 80-100**, y **meditación** cuando esté entre **80-100**. Para determinar si estás pensando en rojo o en azul, cualquiera de los dos debe **estar 5 datos por encima** del otro, en este caso, 50 muestras.

El código quedaría de la siguiente forma:

```
while (stat==true)
  if
(calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~=
0) %leer las señales
    j = j + 1;
    d=d+1;
    %leer atencion/meditacion.
    data_att(j,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
    data_att(j,2) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
    % Los ejes del PLOT
    if(j<70)
       xmin=0;
       xmax=69;
    else
       xmin=j-70;
       xmax=j;
    end
    plot(data_att(:,1),'-r','Linewidth',1);
    plot(data_att(:,2),'-b','Linewidth',1);
    drawnow;
    i=i+1;
    a=a+1; %DUDA
  end
  if((i>180)&&(a>12)) %los primeros 100 datos de calibracion
     a=0;
     aten=data_att((110+b:1:j),1);
     medi=data_att((110+b:1:j),2);
     atencion=length(aten(aten>=80)); %valores válidos mayor de 80
     meditacion=length(medi(medi>=80));
     b=b+10;
     if(atencion>(meditacion+50))
        ...
     elseif(meditacion>(atencion+50))
        ...
     else
     end
  else
  end
end
```

Usando el siguiente código y llevando a cabo una prueba de 40 segundos en la que el usuario pretende mantener el medidor de atención lo más alto posible, el resultado es el siguiente (figura 7.15):

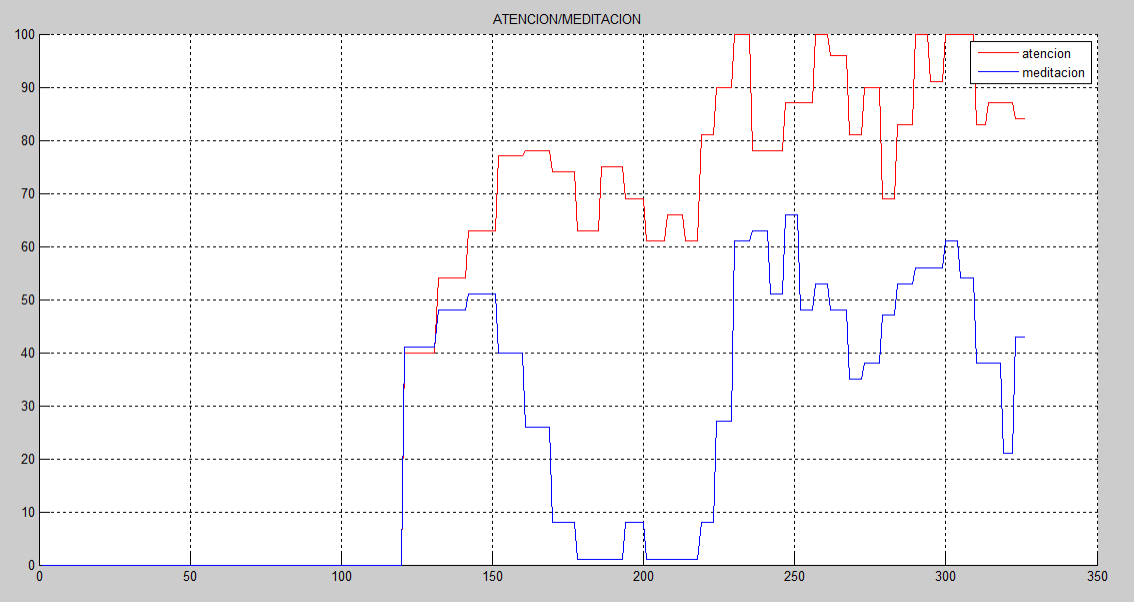


Figura 7.15 *Señal atención y meditación usando el código anterior*

7.1.4. Detección del parpadeo

Una vez obtenido el resultado del color rojo o azul, hay que **validarlo**. Es necesario ya que los datos captados con el casco Mindwave pueden fluctuar mucho y el resultado no ser el esperado. Para la validación del resultado se ha usado el **parpadeo.**

El simple hecho de **parpadear** no es indicativo de un estado mental o de una onda cerebral concreta, pero como se ha comentado con anteriormente, es un **potencial por acción muscular**, es decir, una señal detectada por el electrodo producida por la contracción del músculo frontal.

Dentro del protocolo de transmisión de datos el chip ThinkGear incorpora un algoritmo que da un valor directo de este pico de voltaje producido por el parpadeo, **DATA_BLINK_STRENGTH**.

```
TG_DATA_BLINK_STRENGTH = 37;
data_blink(i,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_BLINK_STRENGTH);
```

Este dato se actualiza cada segundo. Usando el primer código del apartado anterior, se realiza una prueba durante 40 segundos en la que el usuario está parpadeando con diferente "fuerza", obteniendo la siguiente figura (figura 7.16).

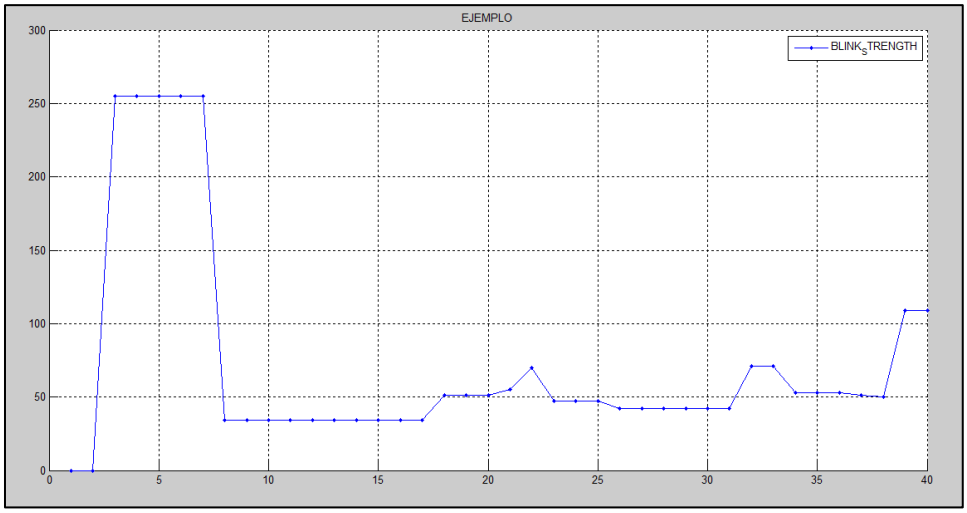


Figura 7.16 *Señal parpadeo (TG_DATA_BLINK_STRENGTH)*

La señal blink tiene un rango entre 0 y 255, llegandose a la conclusión que los valores más cercanos a 255, se producirán cuando el parpadeo era muy forzado. El problema de usar esta señal es que, al actualizarse cada segundo, no recoge todos los parpadeos, ya que tienes que parpadear en el momento exacto en el que se está adquiriendo el dato, y eso es problemático a la hora de tratar de usar esta señal como validación, ya que existe la posibilidad de pretender validar, y que no se reconozca ningún parpadeo.

Este artefacto interfiere la señal raw que lo está recibiendo de manera muy notable, produciendo un pico de voltaje característico. Se pueden apreciar los picos en la figura 8.17 donde se muestra la señal bruta captada con Matlab y alterada voluntariamente a partir del parpadeo. Se observa que a mayor contracción muscular estos picos serán más grandes.

En la siguiente figura (figura 7.17), se ha realizado la misma prueba anterior, pero adquiriendo también la señal raw.

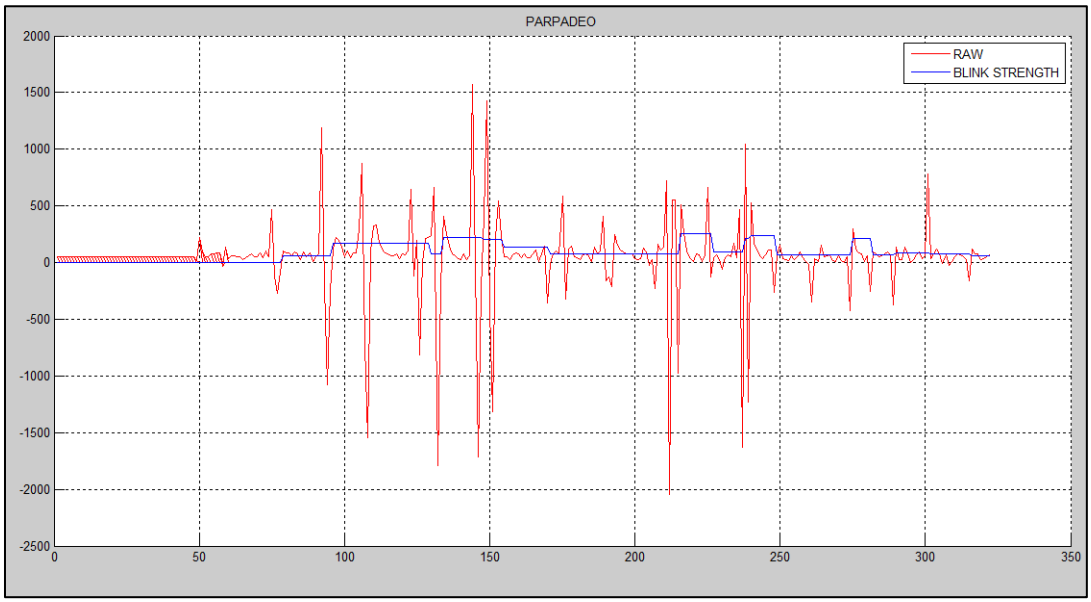


Figura 7.17 *Señal parpadeo y señal raw*

Si el usuario de manera voluntaria realiza la contracción muscular parpadeando forzadamente, se produce un pico de voltaje en la señal raw que destaca por encima del resto, y en la señal blink se observa un valor distinto de cero. Debido a que la señal blink se actualiza cada segundo, es menos real que la señal raw, y como esta contracción se necesita para validar una respuesta realizando dos parpadeos forzados en un tiempo de tres segundos, se usará la señal raw que actualiza los datos cada 2 ms, por lo que recogerá cualquier parpadeo.

En función de la fuerza que se ejerza en el parpadeo, este pico será más o menos pronunciado. Se han realizado varias pruebas con distintos usuarios, para determinar un rango estimado de lo que se considera **parpadeo voluntario e involuntario**. A modo de resumen (tabla 7.3):

UMBRAL PARPADEO (valores absolutos)	
Parpadeo involuntario	0 a 599
Parpadeo forzado	600 a 1499
Parpadeo muy forzado	Valores mayores que 1500

Tabla 7.3 *Rango de valores del parpadeo*

El código para la validación del resultado quedaría de la siguiente forma:

```
while(m<40) %validacion parpadeo
            if
(calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~=
0) %leer las señales
                k = k + 1;
                data_blink(k,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_RAW);
                m=m+1;
                plot(data_blink(:,1),'-k','Linewidth',1);
                drawnow;
              end
           end
           blink_senal=abs(data_blink(:,1)); %valor absoluto
           blink=length(blink_senal(blink_senal>600)); %valor de
parpadeo mayor que 600
           if(blink>8) %validado atencion rojo
              color=1;
              c=2;
              if (modo==1) %Robot conexion real simulacion
                 Socket_robot_simulado;
              else
                 Socket_robot_real;
              end
           else %no validado
             . . .
```

Usando el código anterior, aumentando el tiempo de adquisición de datos de 10 segundos, se recogen los siguientes resultados (figura 7.18): tres parpadeos forzados o muy forzados, por encima del valor absoluto de 600, y cuatro parpadeos involuntarios, por debajo de 500. El resto, son artefactos propios del casco.

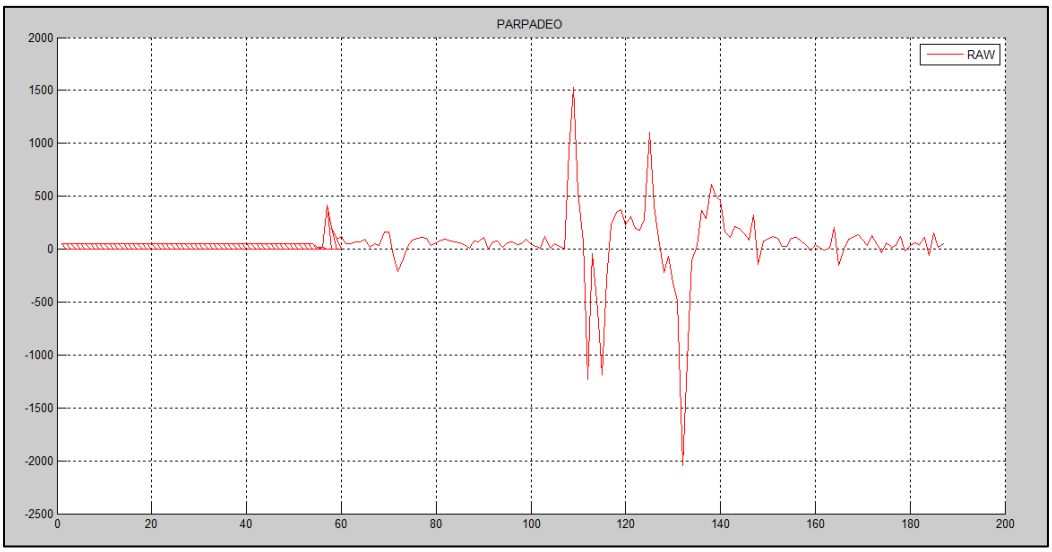


Figura 7.18 *Detección del parpadeo de la señal raw*

7.2. Módulo de adquisición y procesamiento de imágenes

El objetivo del sistema de visión para el robot, es el de transformar la imagen del medio ambiente, proporcionada por la cámara, en una descripción de los elementos presentes en el entorno del robot. Dicha descripción deberá contener la información necesaria para que el robot efectúe los movimientos que permitirán la ejecución de la tarea programada.

En este apartado se desarrollará todo el proceso que se ha seguido, desde el punto de vista teórico y práctico, así como la programación usada en cada apartado. Como se ha comentado en diversas partes del proyecto todo este apartado se ha desarrollado en Matlab. Los pasos que se van explicar son: captura de la imagen mediante WebCam, procesamiento de imagen para obtener las coordenadas de las piezas (características necesarias) y el proceso de calibración para trasladar la posición de las bolas del frame al sistema real.

Captura de imágenes

Con la finalidad de tomar las imágenes en tiempo real del tablero donde encontramos los objetos, se usa una Cámara Web. En este caso, se usa la WebCam USB Logitech C310, con una resolución de imagen de 640x480 pixeles.

Las capturas del tablero se hacen directamente desde Matlab, usando la toolbox "Image Acquisition" y el adaptador "winvideo", y seguidamente se procesan. El programa usado para las capturas desde Matlab es el siguiente:

```
%%Captura imagen
video = videoinput('winvideo', 2,'RGB24_640x480');
%adaptador:windwideo, formato:RGB24_640x480
start(video);
piezas = getsnapshot(video); %Captura de imagen para los colores
```

Procesamiento de imagen

El objetivo de esta primera etapa es separar al objeto que se quiere inspeccionar del fondo y del resto de objetos, para posteriormente extraer las características necesarias (posición, color…)

Se puede definir una imagen como una función bidimensional f(x,y) donde x e y son las coordenadas espaciales, y el valor de f en cualquier par de coordenadas (x,y) es la intensidad de la imagen en dicho punto.

En Matlab, las imágenes de color RGB se representan por **tres matrices bidimensionales**, correspondientes a los planos R, G, B. La siguiente figura (figura 7.19), representa la idea de la composición de una imagen digital.

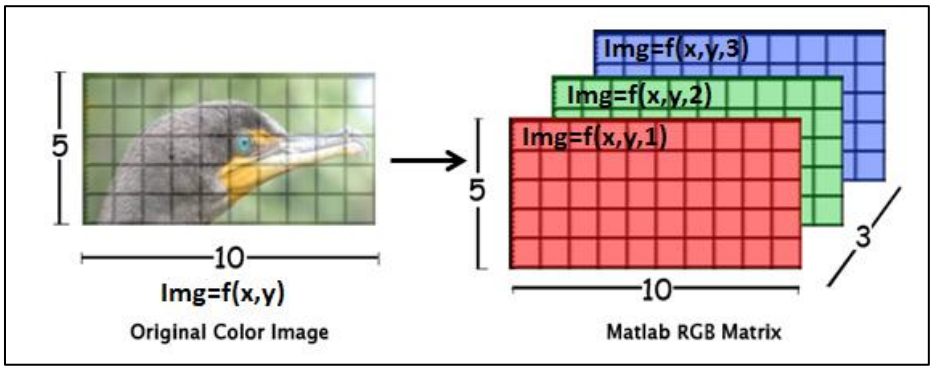


Figura 7.19 *Esquema de una imagen RGB en Matlab* [46]

Partiendo de una imagen original en color (RGB), es posible descomponerla en sus tres canales, usando el siguiente código:

```
img_colorR=piezas(:,:,1); %canal rojo
figure(2)
imshow(img_colorR)
img_colorV=piezas(:,:,2); %canal verde
figure(3)
imshow(img_colorV)
img_colorA=piezas(:,:,3); %canal azul
figure(4)
imshow(img_colorA)
```

El resultado del código anterior, es el mostrado en la siguiente figura (figura 7.20):

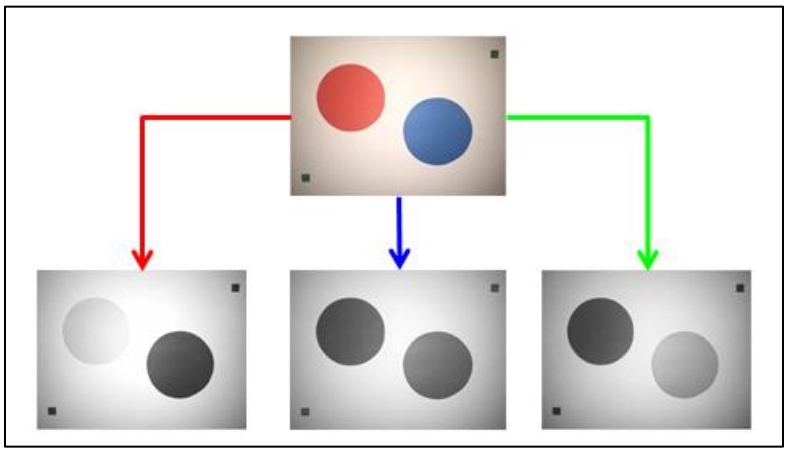


Figura 7.20 *Descomposición de la imagen "piezas" en sus tres canales RGB*

Conversión a escala de grises

La imagen a color adquirida (en valores RGB), es convertida a una imagen en escala de grises, teniendo 256 niveles de gris posibles (rango de 0 a 255) para cada uno de los pixeles que la componen, donde el nivel 255 es blanco, y el nivel 0 negro. La conversión se realiza usando la siguiente función:

```
gris=rgb2gray(piezas); %conversion rgb-gray
```

El siguiente paso es separar el objeto (color) de estudio, del fondo y del resto de los objetos de la imagen, y de esta forma extraer las características necesarias. Como se puede comprobar ni en las imágenes obtenidas de cada canal, ni en la imagen en escala de grises es posible separar los objetos.

Esto se debe a que las imágenes son matrices, por ejemplo, la imagen ***img_azul=(:,:,2),*** tendrá **pixeles** que se acerquen a **255** (azul muy elevado, y nada de rojo o verde; y los pixeles de color blanco ya que todas sus componentes son elevadas), y **pixeles** que se acerquen a **0** (en las zonas de la imagen donde estén los objetos verdes y rojos; y en las zonas oscuras).

Umbralización

La **umbralización** es un método de procesamiento cuyo objetivo es convertir una imagen en escala de grises en una **imagen binaria (1=blanco y 0=negro)**. Se define un límite, **umbral**, y todos los pixeles cuyo valor estén por encima de ese umbral, pasan a tener valor 1 y los pixeles que estén por debajo tiene un valor 0.

En Matlab el límite puede estar definido entre 0 y 255 o entre 0 y 1, dependiendo de la función elegida, en nuestro caso hemos usado la función ***im2bw*** en la que **el umbral se define entre 0 y 1**.

El siguiente código recoge todo lo comentado anteriormente: **conversión a escala de grises**, **aislar el objeto** y **umbralizar** la imagen con el objeto elegido. En este caso, se ha elegido el color azul:

```
umbralA=0.2;
gris=rgb2gray(piezas); %conversion rgb-gray
figure(1)
imshow(gris)
img_colorA=piezas(:,:,3); %canal azul
figure(2)
imshow(img_colorA)
IbwA=img_colorA-gris; % objeto azul
figure(3)
imshow(IbwA)
Im_binA=im2bw(IbwA,umbralA); %umbralizar la imagen
figure(4)
imshow(Im_binA)
```

El resultado del código anterior, se observa en la figura 7.21. El umbral elegido en este caso es 0.2. Una vez obtenemos esta imagen, se usan los operadores morfológicos para rellenar los huecos que hayan podido surgir.

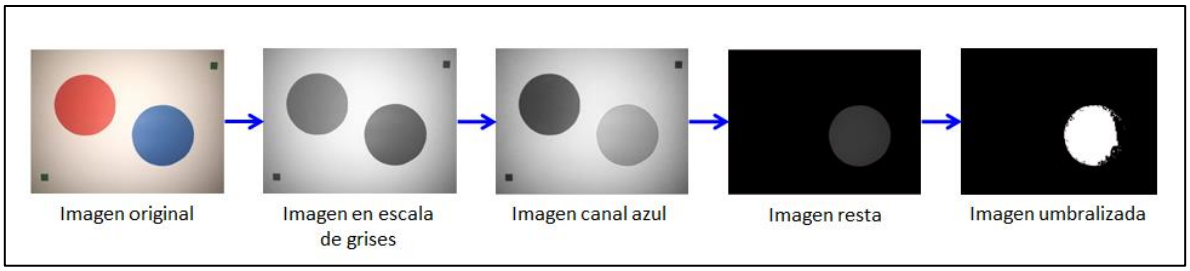


Figura 7.21 *Proceso desde la imagen RBG hasta la imagen binaria (umbralización)*

Operadores morfológicos

Una vez binarizada la imagen, el objeto aislado no es perfecto, debido al ruido en las imágenes y sobre todo a la iluminación de la sala. Por lo que se usan los operadores morfológicos, **erode** y **dilate**. Estos operadores sirven para mitigar estos errores y conseguir de esta forma una figura más definida.

Por otro lado, para eliminar el ruido "sal y pimienta" y que las piezas sean lo más parecidas a la realidad posible, se usa los dos operadores morfológicos: **open** y **close.**

Open (erode + dilate)

El operador OPEN está formado por el operador erode seguido del operador dilate. El operador OPEN elimina el ruido tipo sal con el erode. El ruido tipo pimienta no se amplifica gracias al dilate, pero no se consigue quitar. Los objetos no engordan ni adelgazan.

Close (dilate + erode)

El operador CLOSE está formado por el operador dilate seguido del operador erode. El operador CLOSE elimina el ruido tipo "pimienta" con el dilate. El ruido tipo sal no se amplifica gracias al erode, pero no se consigue eliminar. Los objetos ni engordan ni adelgazan.

Usando los dos operadores morfológicos "open-close" seguidos, se consigue eliminar el ruido tipo sal y pimienta y rellenar los huecos de los objetos aislados. Los objetos ni engordan ni adelgazan para que la figura sea lo más parecida a la real y poder calcular las características.

El código es el siguiente:

```
Im_binA=im2bw(IbwA,umbralA); %umbralizar la imagen
figure(1)
imshow(Im_binA)
ele=strel('disk',5);
Im_bin_A=imopen(Im_binA,ele); %operador open
figure(2)
Imshow(Im_bin_A)
ele2=strel('disk',9);
Im_bin_A=imclose(Im_bin_A,ele2); %operador close
figure(3)
Imshow(Im_bin_A)
```

El resultado de los operadores morfológicos se observa en la figura 7.22:

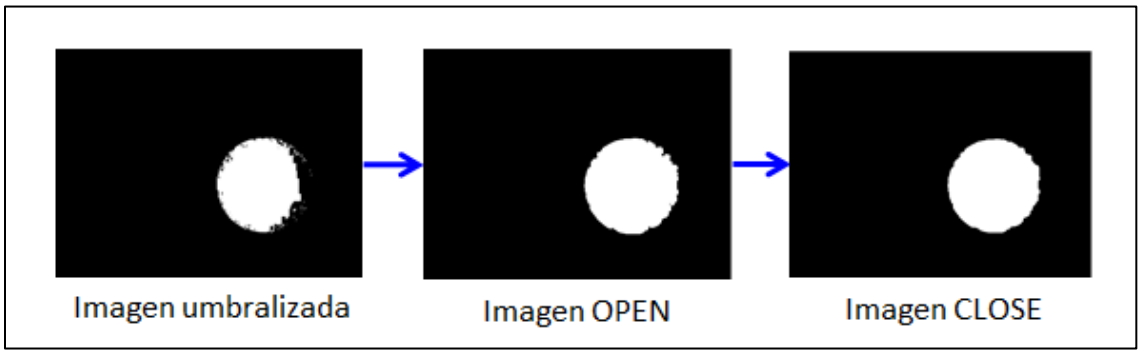


Figura 7.22 *Resultado de los operadores morfológicos*

Cálculo de características de la imagen

Una vez obtenida la imagen libre de ruido y conteniendo solo el objeto o los objetos de interés, se procede a la extracción de las características de cada objeto.

Lo primero es determinar el número de objetos del área de trabajo. En este caso, dos piezas verdes, una azul y una roja. Dependiendo del color filtrado, nos dará un número, verde 2, y azul o rojo 1.

Después de determinar el número de objetos presentes en el área de trabajo, su clasificación por colores y etiquetado; se procede a calcular los **centroides** de cada uno de los objetos.

A continuación se muestra el código para la extracción del centroide del objeto azul:

```
%numero de objetos
IlabelA=bwlabel(Im_bin_A);
num_objA=max(max(IlabelA));
%Centroide
gA=regionprops(IlabelA,'centroid')
imshow(piezas);
hold on
%pintar centroide pieza azul
if (num_objA==1)%numero maximo  azul=1
 for c=1:length(gA)
    plot(gA(c).Centroid(1),gA(c).Centroid(2),'k*');
    centro=gA(c).Centroid;
    Pos_x_objA=centro(1);
    Pos_y_objA=centro(2);
 end
```

El resultado final (figura 7.23), usando todo el código del procesamiento de imagen:

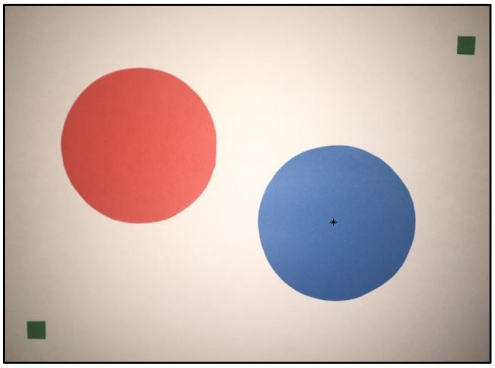


Figura 7.23 *Centroide pieza azul*

Sincronización entre las coordenadas Robot-Frame

La fase de sincronización está destinada a establecer la correspondencia entre el **sistema visual** y el **robot,** es decir, transformar las posiciones en el **sistema de coordenadas de visión** (cámara) a **posiciones en el sistema de coordenadas del robot**, permitiendo así manipular objetos que han sido identificados.

Las dimensiones del área de trabajo del robot (mesa de trabajo) son conocidas gracias a que hay colocada sobre ella una cuadrícula, facilitando la conversión entre las coordenadas del robot-frame.

Con el propósito de llevar a cabo la conversión se realizan los siguientes pasos:

1. Detección de los dos centroides de las piezas sobre la captura de la imagen. Una vez capturada la imagen, se toman como referencia los dos puntos seleccionados en el sistema real y con la ayuda de las herramientas de Matlab se obtienen las coordenadas (en pixeles) de dichos puntos.

   Se crea el siguiente código que calcula la longitud del tablero del sistema real (X e Y) en mm y la longitud del tablero de la foto en pixeles:

```
%LongituclosdX y LongitudY se calcula restando las coordenadas X de
las dos fichas
    LongitudX_foto=max(Pos_x(1),Pos_x(2))-min(Pos_x(1),Pos_x(2));
    LongitudY_foto=max(Pos_y(1),Pos_y(2))-min(Pos_y(1),Pos_y(2));
 %tamaño del tablero en la realidad
    LongitudX_real=max(real_x1,real_x2)-min(real_x1,real_x2);
    LongitudY_real=max(real_y1,real_y2)-min(real_y1,real_y2);
```

2. Una vez analizados dichos puntos, se desarrollan las ecuaciones pertinentes, las cuales relacionan los puntos del sistema real (LongitudX_real y LongitudY_real) y el frame (LongitudX_foto y LongitudY_foto).

Para efectuar la conversión de pixeles a mm se utiliza la **interpolación lineal**, esta función sirve para calcular los valores intermedios dentro de un límite máximo y otro mínimo tanto en x como en y.

En este caso, se usan dos interpolaciones lineales, la primera es para encontrar el valor en mm del centroide en X y la otra para encontrar el valor en mm del centroide en Y, todo con respecto al origen del robot. Para realizar la interpolación se usaron los valores de la posición en pixeles de los centroides de los límites del área de trabajo (**LongitudX_foto** y **LongitudY_foto**) y los valores de las coordenadas en mm de los límites del área de trabajo real (**LongitudX_real** y **LongitudY_real**).

$$y = y_1 + \frac{x - x_1}{x_1 - x_0}\,(y_2 - y_1)$$

El extracto del programa usado para realizar esta interpolación es la siguiente:

```
%diferencia longitud objeto centroide foto
    DiferenciaX_foto=max(Pos_x(1),Pos_x_obj)-min(Pos_x(1),Pos_x_obj);
    DiferenciaY_foto=max(Pos_y(1),Pos_y_obj)-min(Pos_y(1),Pos_y_obj);

%cálculo diferencia longitud objeto centroide real
  DiferenciaX_real=((LongitudX_real*DiferenciaX_foto)/LongitudX_foto);
  DiferenciaY_real=((LongitudY_real*DiferenciaY_foto)/LongitudY_foto);

%centroide en la posicion real para que lo coja el robot
    POSX_OBJ_REAL=real_x1+DiferenciaX_real;% el valor minimo siempre
sera la pieza 1 de referencia, que es la que marca el limite del
tablero
    POSY_OBJ_REAL=real_y1+DiferenciaY_real; % el valor minimo siempre
sera la pieza 1 de referencia, que es la que marca el limite del
tablero
```

7.3. Socket de comunicación

La finalidad de este apartado es el control del robot IRB120. Lo normal sería crear un programa en lenguaje RAPID, simularlo en RobotStudio y cargarlo con el controlador IRC5 para ejecutarlo posteriormente sobre el robot real. Pero, en este caso, el control se llevará a cabo desde el software de Matlab. Esto permite realizar operaciones más complejas, como por ejemplo, incluir una cámara de visión artificial. Por lo que se usará un **socket** para comunicarnos con el servidor, utilizando el **protocolo TCP/IP** (figura 7.24).

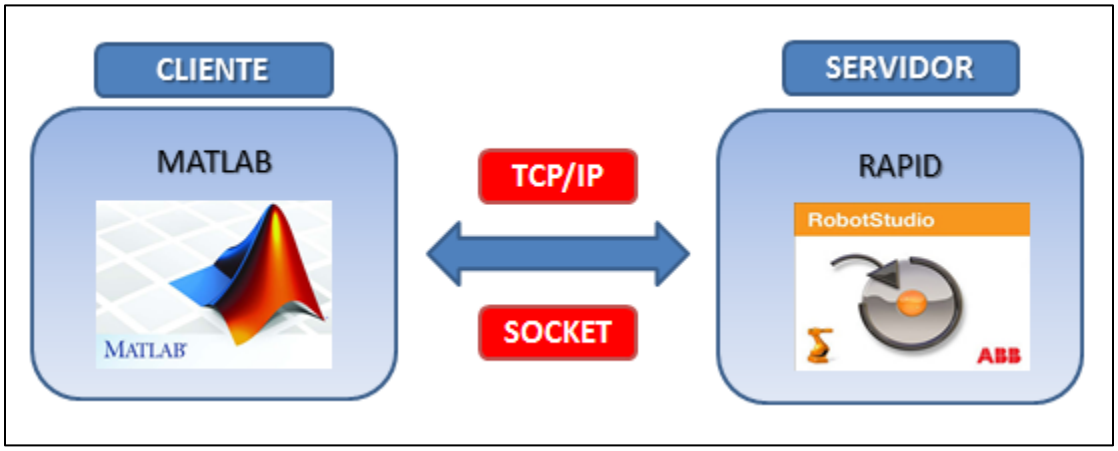


Figura 7.24 *Esquema básico del socket de comunicación Matlab – RAPID*

El **socket de comunicación** usado en este proyecto, fue desarrollado y expuesto en el TFG *"Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB",* por lo que en este apartado se abordará el tema de forma general y se explicarán partes de mayor relevancia, sin entrar en mucho detalle.

El **socket del cliente** (programa desarrollado en Matlab), contiene una serie de funciones que se encargan de construir los **datagramas** y enviarlos a través de la red al servidor. El **socket del servidor** (programa desarrollado en RAPID), se ha diseñado para recibir cada uno de los datagramas y realizar una acción específica.

Un **datagrama** es un paquete de datos preparados para ser enviados a través de la red. El protocolo utilizado para la comunicación, define ciertas normas de organización de los datagramas para normalizar el diseño de los mismos. Gracias a ello, será posible crear una gran colección de diversas aplicaciones que podrán procesar los datagramas recibidos y enviarlos de vuelta a través de una red.

El programa de RAPID identifica la cabecera (ID) de un datagrama entrante y según su valor, realizamos una acción u otra. A continuación, se explicarán en detalle los tipos de datagramas utilizados, su estructura y las acciones que el robot debe realizar.

7.3.1. Datagramas

Datagrama de Orientación y Posicionamiento del TCP

Este datagrama permite controlar al mismo tiempo la **posición del TCP** (Tool Central Point) y su **orientación** (figura 7.25) es decir, la orientación y la posición del efector final acoplado al robot, en este caso una ventosa.

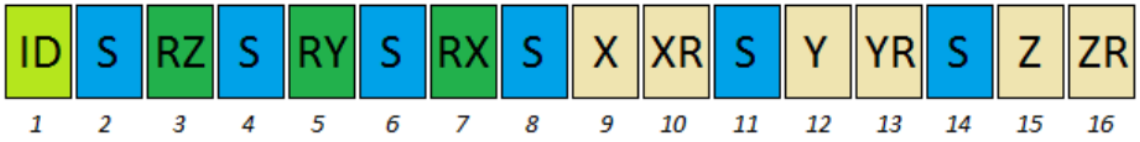


Figura 7.25 *Estructura del datagrama para el envío de orientación y posición* [44]

En la siguiente tabla (tabla 7.4), se puede observar el significado de cada campo del datagrama:

FUNCIÓN	CARACTERÍSTICAS	
ID	**Nombre:** **Tipo de datos:** **Valor:** **Anotaciones:**	Identificador del datagrama Byte 4 Contiene el identificador del datagrama de Orientación del TCP
S	**Nombre:** **Tipo de datos:** **Valor :** **Anotaciones**:	Valor del signo Byte 0 or 1 $sgn\,(J_{n+1}) = \begin{cases} -J_{n+1} & if\ Sn = 0 \\ J_{n+1} & if\ Sn = 0 \end{cases}$ Contiene el signo del siguiente valor de la orientación.
R_{ZYX}	**Nombre:** **Tipo de datos:** **Valor:** **Anotaciones:**	Valor absoluto de la orientación Byte 0 - 255 Un valor absoluto de la rotación alrededor del eje EulerZYX elegido, en referencia al sistema cartesiano TCP local. Sin embargo, el sistema local sin rotar corresponde a la global (los ejes paralelos). De este modo, el robot en la posición por defecto (valores de las articulaciones ponen a 0) tiene la orientación de sistema local del TCP que es igual a (0,90,0).

X / Y / Z **XR** **YR** **ZR**	**Nombre:** **Tipos de datos:** **Valor:** **Anotaciones:**	Valor absoluto de la posición Byte 0 – 255 X / Y / Z contienen los valores en el rango de 0 – 255. XR / YR / ZR contienen los valores de 0 – 99 Debido a la limitación del tipo byte, es necesario dividir el valor de la posición (en mm, en referencia al robot del sistema de coordenadas local) si se quiere alcanzar, por el TCP, cualquier posición dentro de la zona de trabajo. Los campos X y XR se refieren a la coordenada X; y los campos Y y YR a la coordenada Y, y asi sucesivamente. Fórmula matemática para la coordenada X: $X_{real} = 100 \cdot X_{frame} + XR$ Donde, X_{real} –> coordenada X referente al sistema del robot. X_{frame} –> define la cantidad de cientos de X_{real} XR –> el resto del valor (> 100)

Tabla 7.4 *Descripción del Datagrama de Orientación y Posicionamiento del TCP*

De esta manera, en el código RAPID tan solo hay que implementar un **case** para las diferentes ID y después interpretar la información de los bytes siguientes

Datagrama para el control de la herramienta

Este datagrama (figura 7.26), permite controlar la herramienta de trabajo. En este caso, la herramienta (figura 7.27) es una ventosa (tool_Ventosa_SMC_ZPT32BN_B01). Con este datagrama se controla la acción de activar ventosa (succión) o desactivar ventosa (no succión).

Figura 7.26 *Datagrama para control de la herramienta de trabajo* [44]

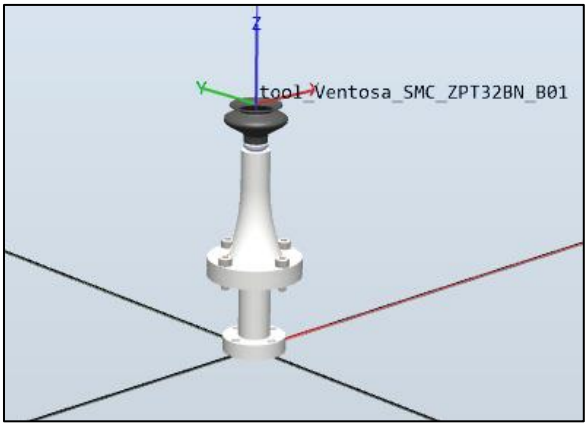


Figura 7.27 *Herramienta Ventosa*

En la siguiente tabla (tabla 7.5), se puede observar el significado de cada campo del datagrama:

FUNCIÓN	CARACTERÍSTICAS	
ID	**Nombre:** **Tipo de datos:** **Valor:** **Anotaciones:**	Identificador del datagrama Byte 6 Contiene el identificador del datagrama de succión
S	**Nombre:** **Tipo de datos:** **Valor :** **Anotaciones:**	Activar/Desactivar (succión) Byte 0 or 1 Si *S = 1*, se llama a la función para desactivar la ventosa Si *S = 0,* se llama a la función para activar la ventosa (succión)

Tabla 7.5 *Características del datagrama para control de la ventosa*

La interpretación de dicho datagrama en ***RobotStudio*** es el siguiente:

```
CASE 6: ! Manipulacion de Herramientas
   IF Data2Process{2} = 0 THEN  !!Activar ventosa (succion)
      setdo CogerPieza,1;   !Simulacion
      setdo CogerPieza,0;   !Simulacion
      SetDO DO10_9,1;   !Activar succion ventosa REAL
      waittime 1;
    ELSEIF Data2Process{2} = 1 THEN  !!Desactivar ventosa (no succion)
       setdo SoltarPieza,1;     !Simulacion
       setdo SoltarPieza,0;     !Simulacion
       SetDO DO10_9,0; !Desactivar succion ventosa REAL
       waittime 1;
     ENDIF
```

Del mismo modo, en el socket de comunicación de Matlab se agregó la función **TCPtool (r,x)** para el movimiento de la ventosa, el código se muestra a continuación:

```
function TCPtool(r,x) %Manipulacion de la Ventosa

            D6=uint8([6 x]);
            fwrite(r.conexion,D6);
            pause(2);
            fwrite(r.conexion,D6);
            pause(2);

         end
```

7.3.2. Conexión con el robot

Para conectar Matlab con el **simulador** de RobotStudio o con el **controlador real** (Controlador IRC5) del robot se usarán las clases creadas en el TFG *"Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB",* en el que se observa que para que se produzca la conexión hay que indicarle la **dirección IP** y el **puerto**.

```
%Conectar con el simulador
robot=irb120('127.0.0.1',1024);

%Conectar con el robot real
robot=irb120('172.29.28.185',1024);

%Conectar con el robot
robot.connect;
```

Además se utiliza la función **TCProtandpos** (también creada y explicada en el TFG *"Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB")* para enviar la posición y orientación del efector final del robot.

```
%Enviarle una posicion de destino
robot.TCProtandpos([180 0 180 X Y Z])
```

Los **tres primeros elementos (180 0 180)** indican la orientación del efector final, a partir de la rotación sobre el eje X, Y y Z respectivamente, es decir, una rotación de 180 sobre el eje X, 0 sobre el eje Y y 180 sobre el eje Z, en grados. Los **tres últimos elementos del vector (X Y Z)** indican la posición del efector final respecto a los ejes de coordenadas X, Y, y Z respectivamente, en milímetros.

Además, se utiliza la función **robot.TCPtool** para controlar la herramienta Ventosa:

```
%Control de herramienta
robot.TCPtool(1) %Desactivar la ventosa
robot.TCPtool(0) %Activar la ventosa
```

7.4. Módulo Robot

La acción final del sistema BCI la realiza el robot IRB120 de ABB. Para llevar a cabo esta parte, en la que se necesita el socket de comunicación del servidor, se ha partido del proyecto desarrollado en el TFG: "*Socket based communication in RobotStudio for controlling ABB-IRB120 robot. Design and development of a palletizing station*",realizado por Marek Jerzy Frydrysiak. Por lo que en este apartado se explicarán las partes que se han modificado y/o incluido.

7.4.1. Sistema simulado

Se ha creado la siguiente estación de trabajo en RobotStudio (figura 7.28), para simular la estación real y que cumpla las especificaciones de este proyecto, la mesa del laboratorio, con los dos discos rojo y azul, los botes donde depositar dichos discos, el robot modelo IRB120 y la herramienta ventosa (tool_Ventosa_SMC_ZPT32BN_B01).

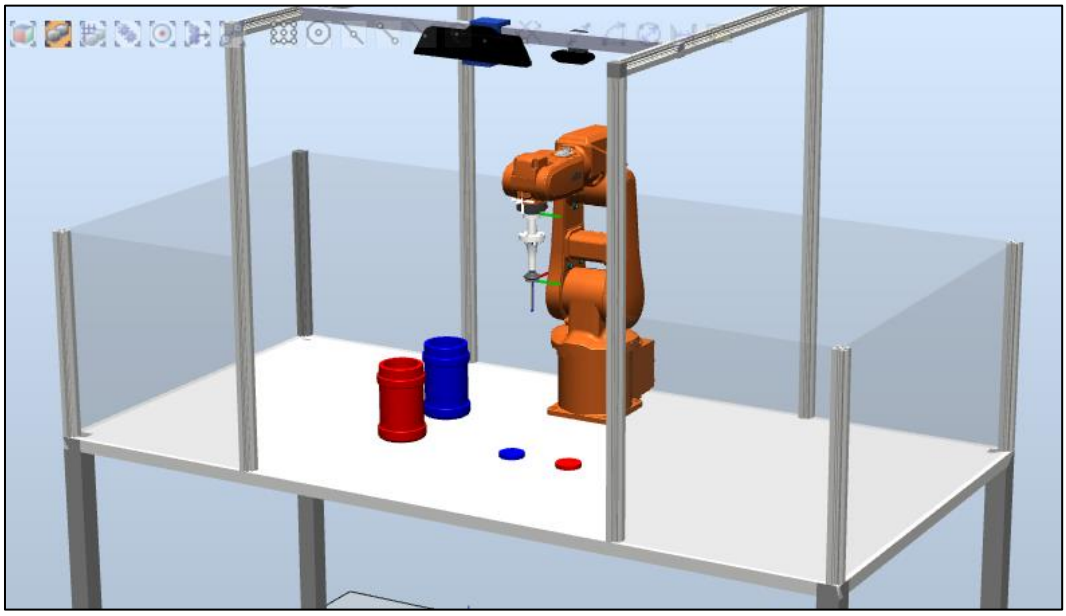


Figura 7.28 *Estación de trabajo de robotstudio de la aplicación final*

También se ha elaborado el "**componente inteligente ventosa"**; estos componentes sirven para controlar el comportamiento de las señales y de las propiedades de los componentes básicos. El componente inteligente debe encargarse de coger el objeto (efecto succión de la ventosa) y de soltar el objeto (desactivar el efecto succión de la ventosa) (figura 7.29).

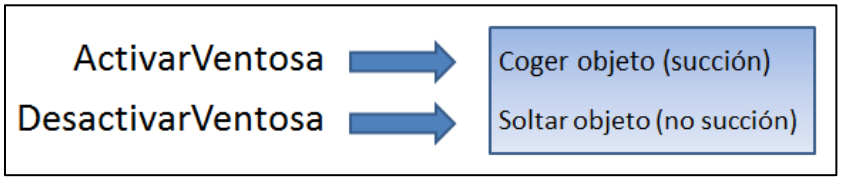


Figura 7.29 *Esquema general del efecto de la ventosa*

Para la creación del componente inteligente, se usan cinco módulos de acción:

- **LineSensor**: detecta el disco.
- **Attacher**: conecta un objeto, coge el disco.
- **Dettacher**: desconecta un objeto conectado, deja el disco.
- **JointMover:** mueve los ejes del robot a una posición definida durante un tiempo determinado.
- **LinearMover:** mueve un objeto en una trayectoria lineal, mueve el disco en la dirección z=-1, simulando la gravedad.

Además tendrá dos entradas y una salida:

- **ActivarVentosa** (entrada): se activará cuando se desee coger uno de los discos.
- **DesactivarVentosa** (entrada): se activará cuando se desee soltar uno de los discos.
- **Sensor** (salida): se activará cuando el sensor esté detectando uno de los discos.

El diseño quedará de la siguiente forma (figura 7.30):

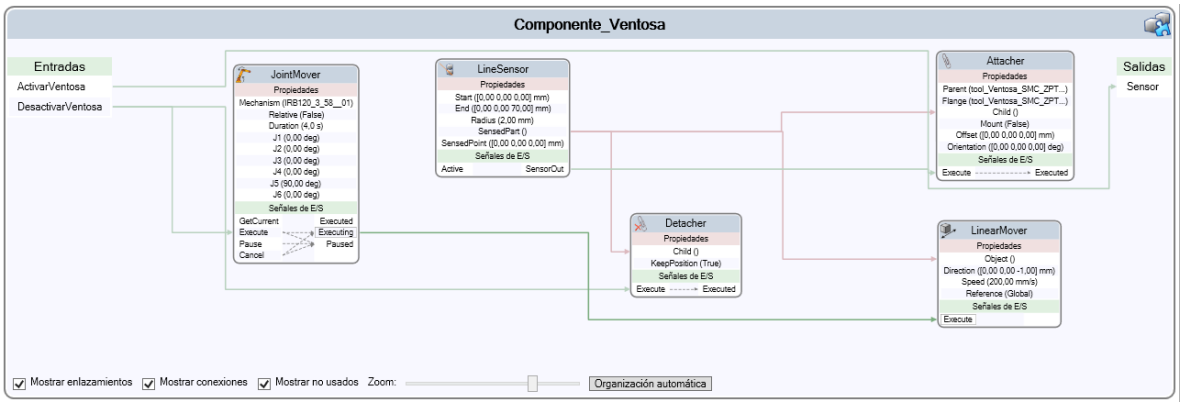


Figura 7.30 *Diseño final del componente inteligente ventosa*

Hay que establecer los comandos/ señales que determinarán los movimientos/acciones del manipulador IRB120 más ventosa cuando se activen los diferentes elementos o señales de los componentes inteligentes.

Las señales entrada-salida del sistema robot (figura 7.31), serán la de **CogerPieza,** señal de salida, y **SoltarPieza,** también señal de salida.

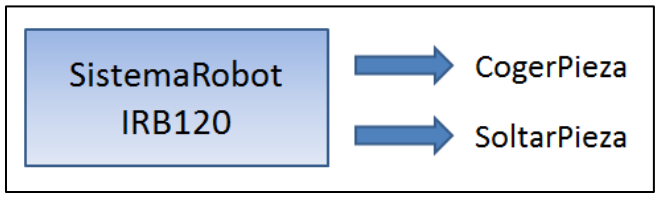


Figura 7.31 *Esquema general de las entradas y salidas del diseño*

Señales de E/S del sistema robot:

- **CogerPieza**: señal de salida que indica que el robot ha llegado a la posición del disco y debe proceder a cogerlo.
- **SoltarPieza**: señal de salida que indica que el robot ha llegado a la posición dejar disco y debe soltarlo.

Estas señales que se usaran en la simulación de la aplicación, se crean en la unidad virtual ABB (figura 7.32):

CH1	Digital Input	PANEL	Run Chain 1	22	safety	INTERNAL	0
CH2	Digital Input	PANEL	Run Chain 2	23	safety	INTERNAL	0
CierraGripper0	Digital Output	ABB		2		Default	0
CogerPieza	Digital Output	ABB		4		Default	0
DI10_1	Digital Input	BOARD10		0		DEFAULT	0
DI10_2	Digital Input	BOARD10		1		DEFAULT	0
DI10_3	Digital Input	BOARD10		2		DEFAULT	0
soGAP_SERVICE9	Digital Output	GAP	GAP Execution	79	internal	GAP_INTERFACE	0
SoltarPieza	Digital Output	ABB		3		Default	0
SS1	Digital Input	PANEL	Superior Stop chain(X6:4 to X6:6) and (X6:2 to X6:1)	19	safety	INTERNAL	0
SS2	Digital Input	PANEL	Superior Stop chain backup(X6:5 to X6:6) and (X6:3 to X6:1)	20	safety	INTERNAL	0
STLEDGR	Digital Output	PANEL	Set status LED to green color	9	safety	INTERNAL	0

Figura 7.32 *Unidad virtual de ABB*

Una vez configurados los componentes inteligentes y la tarjeta virtual de E/S del sistema robot, deben establecerse las conexiones entre los componentes inteligentes y el sistema robot. La lógica de la estación queda de la siguiente forma (figura 7.33):

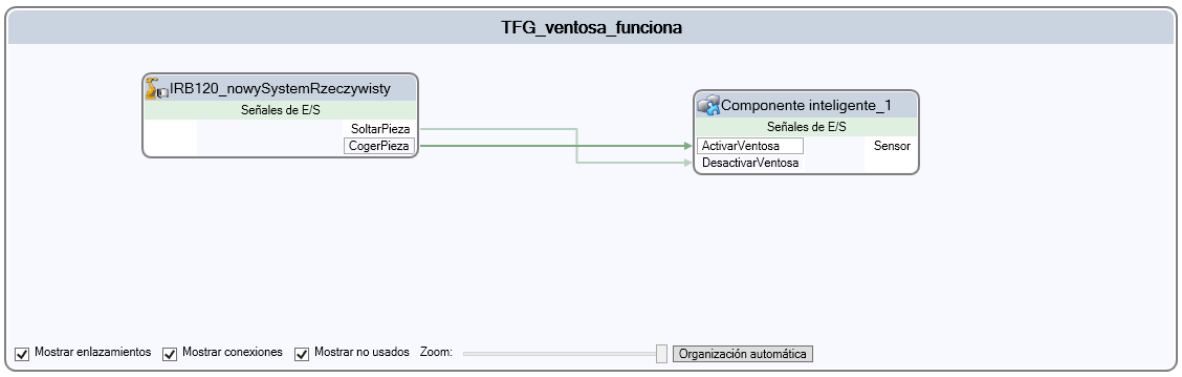


Figura 7.33 *Lógica de la estación del componente inteligente ventosa y el sistema robot*

Partiendo del código RAPID del proyecto: "*Socket based communication in RobotStudio for controlling ABB-IRB120 robot. Design and development of a palletizing station",* lo único que se ha modificado es en el módulo "IRB120_Control" el **case 6**, que corresponde a la **manipulación de la herramienta**, quedando:

```
CASE 6: ! Manipulacion de Herramientas
   IF Data2Process{2} = 0 THEN  !!Activar ventosa (succion)
      setdo CogerPieza,1;    !Simulacion
      setdo CogerPieza,0;    !Simulacion
      SetDO DO10_9,1;    !Activar succion ventosa REAL
      waittime 1;
   ELSEIF Data2Process{2} = 1 THEN  !!Desactivar ventosa (no succion)
       setdo SoltarPieza,1;      !Simulacion
       setdo SoltarPieza,0;      !Simulacion
       SetDO DO10_9,0; !Desactivar succion ventosa REAL
       waittime 1;
   ENDIF
```

7.4.2. Sistema real

En este caso, no hay estación de trabajo, sino el **sistema real** (figura 7.34). En él se encuentran las **dos piezas verdes**, que sirven para la calibración del espacio de trabajo, los dos **discos azul y rojo**, los **dos botes** donde depositar dichos discos, la **WebCam** conectada al ordenador para hacer las capturas, el **controlador IRC5**, el **robot IRB120** y, conectado a éste, la **ventosa**.

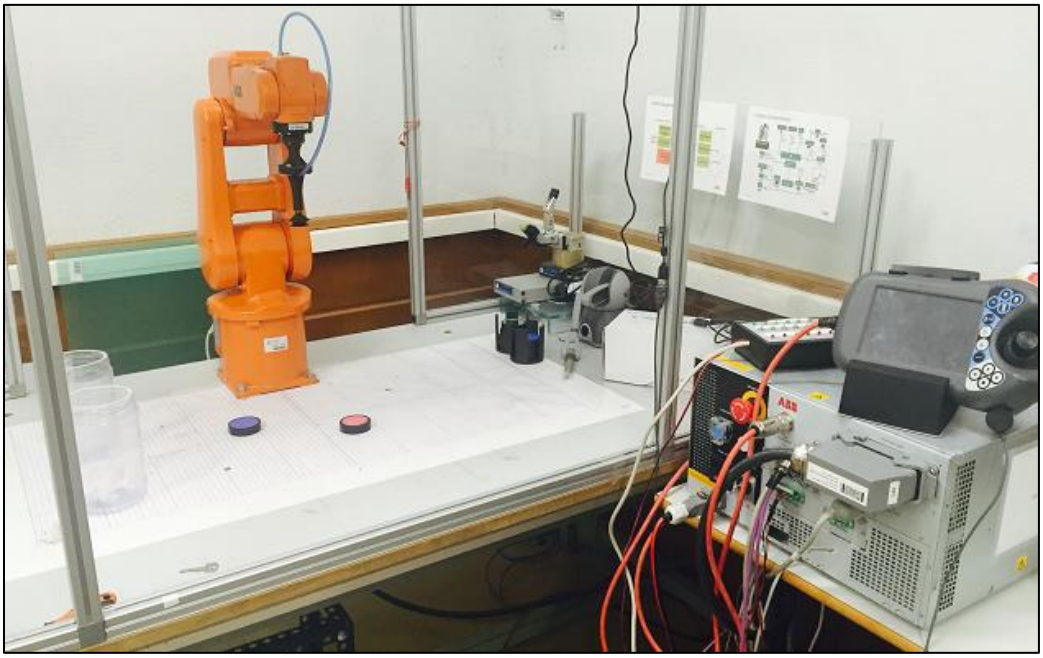


Figura 7.34 *Sistema real*

El armario de control IRC5 generará las órdenes de succión/no_succión (coger/dejar) a través de las E/S. En este caso, para la herramienta ventosa se utiliza la salida 9.

- Succión si DO10_9 = 1.
- No succión si DO10_9 =0.

```
CASE 6: ! Manipulacion de Herramientas
   IF Data2Process{2} = 0 THEN  !!Activar ventosa (succion)
      setdo CogerPieza,1;   !Simulacion
      setdo CogerPieza,0;   !Simulacion
      SetDO DO10_9,1;   !Activar succion ventosa REAL
      waittime 1;
   ELSEIF Data2Process{2} = 1 THEN  !!Desactivar ventosa (no succion)
       setdo SoltarPieza,1;     !Simulacion
       setdo SoltarPieza,0;     !Simulacion
       SetDO DO10_9,0; !Desactivar succion ventosa REAL
       waittime 1;
   ENDIF
```

7.5. Interfaz Gráfica

La interfaz desarrollada a través de la herramienta de Matlab GUIDE, es uno de los puntos clave de este proyecto.

Una interfaz gráfica de usuario (GUI) es una interfaz construida a través de objetos gráficos, menús, botones, barras de desplazamiento, figuras, etc. Todo bastante intuitivo.

A fin de diseñar una interfaz gráfica efectiva, se deben escoger los objetos gráficos que se desean ver por pantalla, distribuirlos y organizarlos de una manera lógica para que la interfaz sea fácil de usar.

Matlab permite desarrollar interfaces gráficas gracias a la herramienta **GUIDE** (GUI Development Environment). Esta herramienta permite diseñar la ventana y también genera un archivo de *extensión.m* que contiene el código para controlar el lanzamiento e inicialización de la aplicación GUI y realizar posibles modificaciones en su programación.

Los gráficos de Matlab tienen una estructura jerárquica, formada por objetos de distintos tipos (figura 7.35):

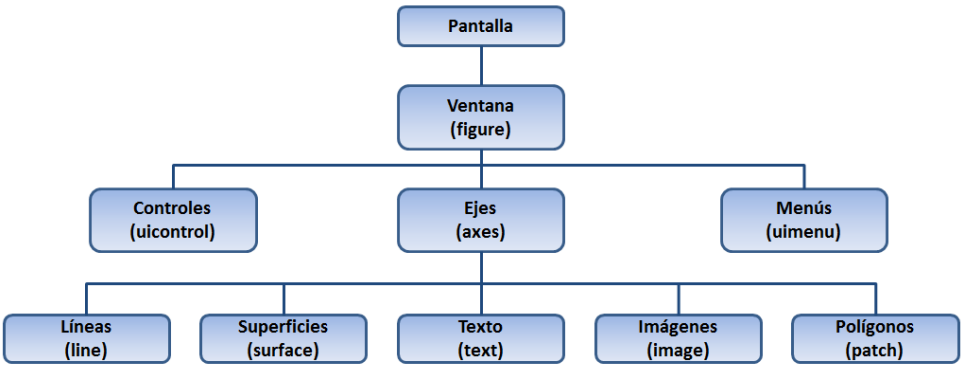


Figura 7.35 *Jerarquía de objetos gráficos de Matlab*

El objeto más general en una interfaz es la **pantalla**. Una pantalla puede contener una o varias ventanas (*figure*). Éstas, a su vez, pueden contener **controles** (*uicontrol*), como botones, barras de desplazamiento, etc, además de **menús** (*uimenus*), y uno o varios **ejes de coordenadas** (*axes*), en los que se pueden representar cinco tipos de elementos gráficos: *line*, *surface*, *text*, *image* y *patch.*

En la interfaz de Matlab se establece entre los objetos una relación padre-hijo. Por lo que cuando se borra un objeto en Matlab, automáticamente también se borran todos los objetos "hijos" del mismo. El comando a utilizar será: *delete* (*identificador*).

Cada objeto está asociado a un identificador (***handle***). A través de este identificador se pueden modificar las características del objeto gráfico. El identificador del objeto raíz, la pantalla, es siempre cero. El identificador de las ventanas es un entero que aparece en la barra de título de dicha ventana.

Así mismo, Matlab puede tener varias ventanas abiertas, pero siempre hay una única activa. Por otro lado, una ventana puede tener varios ejes abiertos pero solo uno activo.

Para crear una GUI, se selecciona la opcion "*Blank BUI(*Default)" (figura 7.36) y aparece una ventana que servirá como editor del nuevo GUI. En ella se colocarán los distintos objetos gráficos que se necesiten para la aplicación.

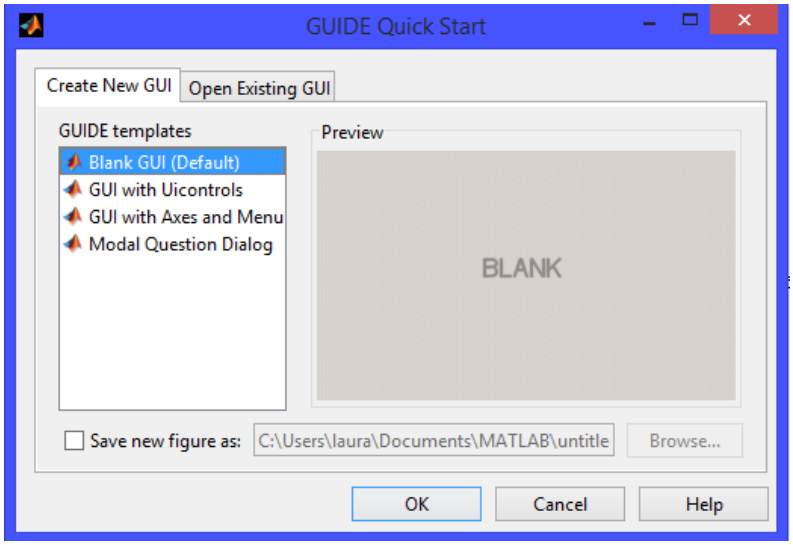


Figura 7.36 *Crear una GUIDE*

El desarrollo de la aplicación GUI se lleva a cabo en un área de trabajo (figura 7.37), donde se colocan los distintos objetos y elementos que se van a utilizar, y se definen funciones.

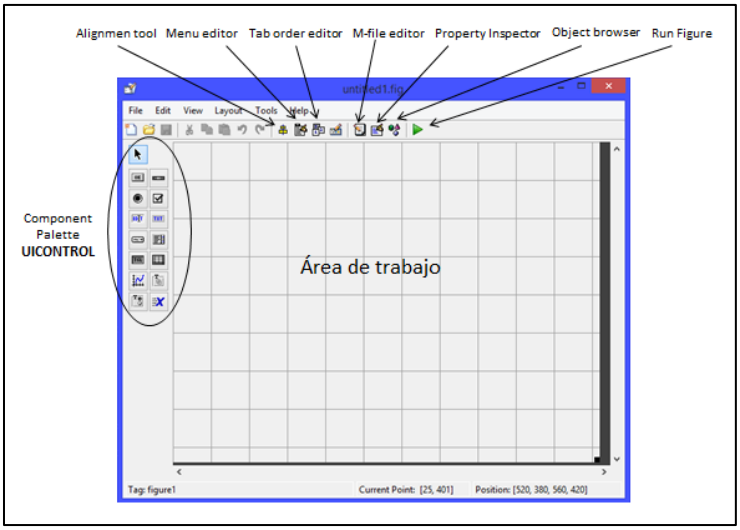
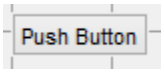


Figura 7.37 *Área de trabajo*

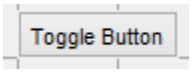
Como se observa en la figura anterior (figura 7.37), en el área de trabajo se colocaran los distintos elementos de control (*uicontrol*) de la aplicación. De igual modo, se han marcado en la figura las opciones más empleadas.

Los ***uicontrols*** son los denominados controles de usuario del Interface *(User Interface Controls)*. En el área de trabajo se pueden insertar los siguientes:

- ***Push Buttons***: permiten pasar de un estado al siguiente, en la aplicación, cuando el usuario pulsa sobre ellos (bien con el ratón o con la tecla *Enter*). Son los típicos botones de activación: Cancelar, OK…



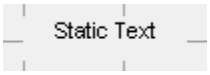
- ***Toggle Buttons***: generan una acción e indican un estado binario (*on*/*off*). Para ejecutar la *callback* asociada a este botón se necesita leer el valor con el comando: *get (gcbo,'value'*)



- ***Radio Buttons***: este tipo de botón, solo tiene dos estados: activado o desactivado.

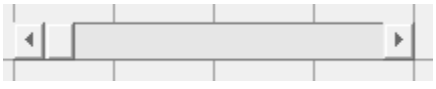
- ***Edit Text:*** sirve para modificar y escribir cadenas de texto. Dicho texto se puede modificar durante la ejecución.

- ***Static Text***: sirve para insertar un texto fijo en controles, botones o en la propia aplicación. Dicho texto no se puede modificar durante la ejecución.



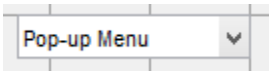
- ***Check Boxes***: cajas de selección. Se activan pulsando sobre el cuadrad y aparece un "*tick*". Tiene dos posibles valores: seleccionado o no seleccionado.

- ***Sliders***: se suelen utilizar para que el usuario de la aplicación seleccione valores numéricos. Aparece una barra que puede tener una orientación vertical u horizontal. Esta orientación se establece en el *property inspector*. El slider tiene asignadas cuatro variables: *value*, *max*, *min* y *sliderstep* (el salto de un valor al siguiente).

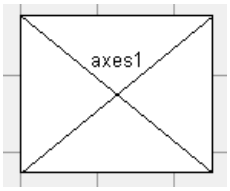


- ***List Boxes***: aparece una lista con varias opciones donde el usuario selecciona una de ellas.

- ***Pop-Up Menus:*** listas desplegables donde se muestran varias opciones para que el usuario de la aplicación pueda seleccionar una o varias de ellas.



- ***Axes y figures***: permiten insertar ejes y figuras en el GUI.



Cada objeto gráfico y/o figura que se crea en la interfaz, lleva asociado un ***callback***. Es una subrutina que permite definir la acción que llevará a cabo el objeto (*uicontrol*) cuando se active durante la ejecución la aplicación.

Una aplicación GUIDE consta de dos archivos: archivo.FIG, parte gráfica, y archvo.M, parte ejecutable. Cuando se salva la aplicación, automáticamente se crean estos dos archivos.

El **archivo.FIG** contiene la descripción gráfica de la interfaz. En él se guarda la figura con los objetos gráficos (*uicontrol*) insertados en ella, para su posterior modificación o uso. También contiene la información relativa a las propiedades de cada objeto que se pueden modificar in situ (nombre, color, datos…)

El **archivo.M**, es un fichero que contiene las funciones necesarias para controlar y ejecutar la GUI y las funciones *callbacks* de cada objeto del archivo.FIG. El programador, dependiendo de lo que quiera que haga dicho objeto introducirá y modificar el código pertinente. En el archivo.M generado se encuentran todas las funciones *callbacks* asociadas a los objetos *uicontrol*. Por ejemplo, la función ***Callback*** del ***push botton*** es la siguiente:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Las estructuras ***handles*** se usan como una entrada a cada *callback* para el intercambio de datos entre la aplicación (*archivo.FIG*) y el *archivo.M.*

Todos los valores de las propiedades de los elementos (valor, color, posición,..) se almacenan en una estructura y se accede a ellos mediante un único y un mismo puntero. El puntero se asigna en:

```
handles.output = hObject;
```

Handles es el puntero a los datos de la aplicación. Esta definición de puntero es salvada con la siguiente instrucción:

```
guidata(hObject, handles); % guidata, es la sentencia para guardar los
                             datos de la aplicacion
```

Guidata es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación, por lo tanto, en cada subrutina, se debe escribir en la última línea.

Por ejemplo, para almacenar los datos contenidos en una variable "A", se fija un campo de la estructura *handles* igual a "B", y se salva la estructura con *guidata* como se muestra a continuación:

```
handles.A = B;
guidata(hObject, handles);
```

En cualquier punto del programa, en cualquier *callback*, se puede recuperar el dato con el comando:

```
B = handles.A;
```

Es muy importante dominar el uso de la estructura *handles* para implementar y desarrollar aplicaciones GUI que requieran, continuamente, cambios en los parámetros de los botones según mandatos del usuario.

La asignación u obtención de valores de los componentes se realiza mediante las sentencias ***get*** y ***set***. Por ejemplo, si se quiere que la variable "C" tenga el valor del *uicontrol Slider* se escribe:

```
C = get(handles.slider1,'Value');
```

Notar que siempre se obtienen los datos a través del puntero *handles.*

Para asignar el valor a la variable "C" al *statictext* etiquetada como *text1* se escribe:

```
set(handles.text1,'String',C); %Escribe el valor del slider
                               %en el static-text
```

Creacion de las interfaces a través de GUI

A continuación se explicará en detalle la interfaz y el entorno de funcionamiento de la misma. Para esta aplicación se han creado dos interfaces: la primera, la ***InterfazMatlab*** y la segunda, la ***InterfazNeurosky*** (figura 7.38).

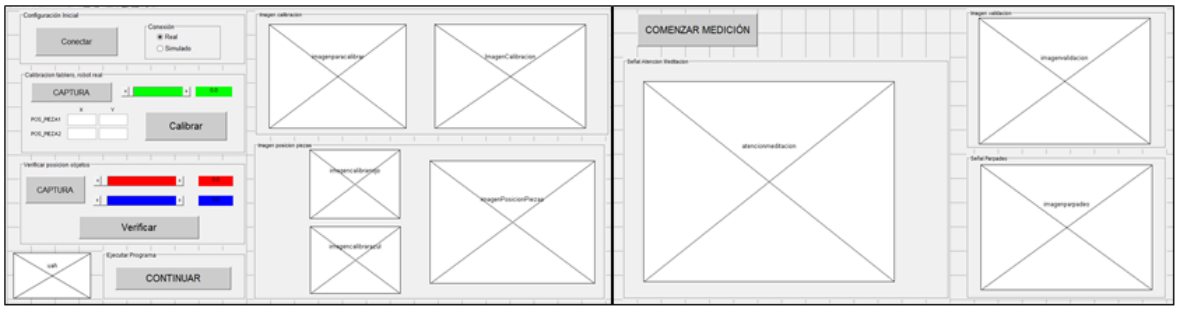


Figura 7.38 A*: InterfazMatlab y* **B***: InterfazNeurosky*

En la primera interfaz se realizan la conexión con el robot, la calibración con el tablero y el procesamiento de la imagen para determinar la posición de las piezas y la conversión de pixel-mm de dicha posición.

En la segunda interfaz se muestran los datos que se obtienen del casco Mindwave, la atencion y la meditacion, y el parpadeo, además se van cargando diferentes figuras conforme va avanzando la aplicación para orientar al usuario de lo que tiene o puede ir haciendo.

Cada una de las interfaces gráficas GUI se componen de dos archivos: un "*archivo.m*", el que se encuentra todo el código referente a la programación de controles y el "*archivo.fig",* en el que se localizan los controles propiamente dichos.

InterfazMatlab

En la imagen siguiente se puede observar la apariencia de la interfaz desde la pantalla de edición de la herramienta GUIDE (figura 7.39).

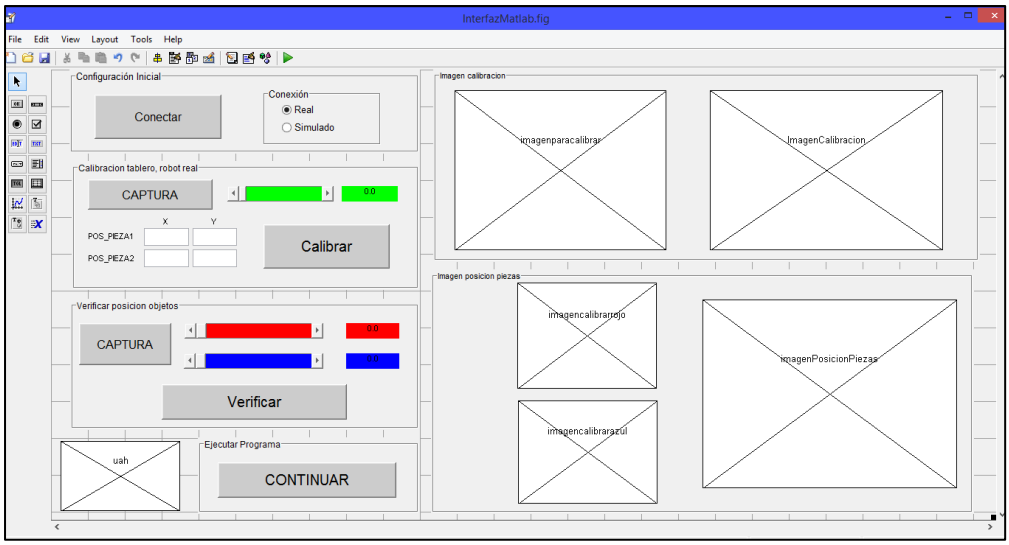


Figura 7.39 *InterfazMatlab en la pantalla de edición GUIDE*

A partir de la figura anterior se comenzarán a explicar los diferentes controles utilizados y el código implementado para cada uno de ellos.

Configuración inicial

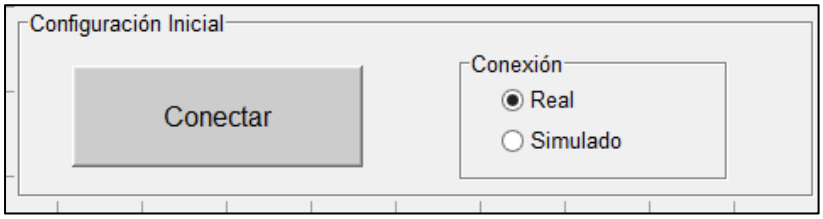


Figura 7.40 *Parte de la interfaz que conecta con el robot*

Como se observa en la "Configuración Inicial" (figura 7.40), en el apartado "Conexión" se elige la forma de conexión con el robot, real o simulada. El *push button* "Conectar" se encarga de conectar con el robot, dependiendo de lo que se haya marcado en la opción con la variable modo: **real** o **simulado**.

Es importante mencionar también que se ha hecho uso de variables globales, debido a que se deben utilizar ciertas variables para diversos controles y como ya se sabe, las variables utilizadas en una función (y no declaradas como globales), son consideradas por el programa como locales y por tanto no serían detectadas por otras funciones de nuestro código.

El código relacionado con la “Configuración Inicial” es el siguiente:

```
function conectar_Callback(hObject, eventdata, handles)
% hObject    handle to conectar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global modo comenzar
%%%Control de apariencia de botones en interfaz
set(handles.conectar,'Enable','off');
set(handles.real,'Enable','off');
set(handles.simulado,'Enable','off');
Conectar; %Conectar con robot
if modo==1 && comenzar==1
  set(handles.continuar,'Enable','on'); % Opcion simulacion
else
 set(handles.capturacalibracion,'Enable','on'); % Opcion real
end
```

En la función ***conectar*** se realiza la comunicación mediante el protocolo TCP/IP con sus correspondientes comandos. Se produce la conexión real y simulada dependiendo del valor de la variable ***modo***.

Y como se observa en el código, dependiendo de la opción, se conecta con una IP.

```
%CODIGO DE CONEXION CON BRAZO IRB120
global robot Simulado comenzar
%Obtenemos los datos sobre el modo de conexion
comenzar=0;
modo=get(handles.simulado,'Value');
if modo==1
  %Conectar con el simulador
  robot=irb120('127.0.0.1',1024);
else
  %Conectar con el robot real
  robot=irb120('172.29.28.185',1024);
end
```

Calibración tablero, robot real

Una vez seleccionada la opción real, se activa la parte de la calibración del tablero (figura 7.41).

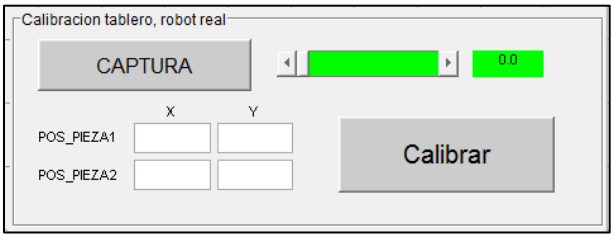


Figura 7.41 *Parte de la interfaz de la calibración del tablero*

Cuando se pulsa el *push button* "Captura", hace la foto de la mesa de trabajo. A continuación, se carga la imagen binarizada en la figura "*ImagenparaCalibracion"* de la interfaz.

El valor umbral inicial para la foto binarizada es 0. Dependiendo de la luz, el color de la pieza etc, se necesitará otro valor umbral. Este umbral está asociado al slider, el rango es de 0 a 1. Una vez se mueve el valor del slider, se carga el valor en el cuadro de texto y la imagen binarizada va cambiando en función del umbral marcado.

El código que recoge todo esto es el siguiente:

```
function capturacalibracion_Callback(hObject, eventdata, handles)
% hObject    handle to capturacalibracion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global umbralV Im_binV IbwV f
%%Inicializar video
vid = videoinput('winvideo', 2);
start(vid);
f=getsnapshot(vid); %Captura de imagen
umbralV=get(handles.sliderVerde,'Value');
gris=rgb2gray(f);
img_verde=f(:,:,2);
IbwV=img_verde-gris;
Im_binV=im2bw(IbwV,umbralV);
axes(handles.imagenparacalibrar);
imshow(Im_binV);
axis off
%guidata(hObject,handles)
set(handles.calibrar,'Enable','on'); %activar calibración
set(handles.capturacalibracion,'Enable','off'); %desactivar captura
```

Al final del código anterior, se activa el push button "*Calibrar*" y se deshabilita la opción de "*Captura*" ya que solo se calibra una vez.

Se han programado cuatro cuadros de texto para introducir las coordenadas de las esquinas de las piezas verdes "POS_PIEZA1 y POS_PIEZA2".

El código correspondiente a la "POS_PIEZA1", coordenada X es el siguiente:

```
function posx1_Callback(hObject, eventdata, handles)
% hObject    handle to posx1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of posx1 as text
%        str2double(get(hObject,'String')) returns contents of posx1
as a double
global err_X1
X1=str2double(get(hObject,'String')); %Almacenar valor ingresado y
transformar a tipo double
handles.posx1=X1; %Almacenar en puntero
```

Este código es igual para las cuatro variables. Además en la Callback se ha programado un error, que aparecerá si el valor introducido no es numérico, como se ve a continuación:

```
if isnan(X1)
    errordlg('El valor debe ser numérico','ERROR')
    err_X1=1;
else
    err_X1=0;
end
guidata(hObject,handles); %Salvar datos de la aplicación
```

Una vez introducidos los datos y en la "imagenparacalibrar" se aprecian los objeto aislados, se pulsa el *push button* "Calibrar". Calcula los centroides de las piezas verdes. El código de esta parte es el siguiente:

```
function calibrar_Callback(hObject, eventdata, handles)
% hObject    handle to calibrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global err_X1 err_X2 err_Y1 err_Y2 f Im_binV a b X1 X2 Y2 Y1
calibracion real_x1 real_x2 real_y1 real_y2 Pos_x Pos_y LongitudX_foto
LongitudY_foto LongitudX_real LongitudY_real real_x1 real_x2 real_y1
real_y2
X1=handles.posx1;
Y1=handles.posy1;
X2=handles.posx2;
Y2=handles.posy2;
if  (((err_X1==1) || (err_X2==1)) || ((err_Y1==1) || (err_Y2==1)))
    errordlg('Calibracion incompleta, datos introducidos
ERRONEOS','ERROR');
    axis off;
else
calibracion_tablero; % Procesamiento de imagen. Cálculo de centroides
b=a;
if ((b==3) && (LongitudX_real>=0) && (LongitudY_real>=0))
  msgbox('Calibracion completada satisfactoriamente','Calibracion');
  axis off;
  set(handles.capturaRojoAzul,'Enable','on');
  set(handles.verificar,'Enable','on');
else
    errordlg('No ha sido posible la calibracion','Calibracion');
    axis off;
end
```

Además, se han programado varios errores y mensajes, en relación con la calibración.

Verificar posición objetos

Una vez calibrado el tablero, se activa la opción de captura de la imagen y la verificación de las piezas (figura 7.42), para poder calcular los centroides de las piezas y que el robot pueda cogerlas.

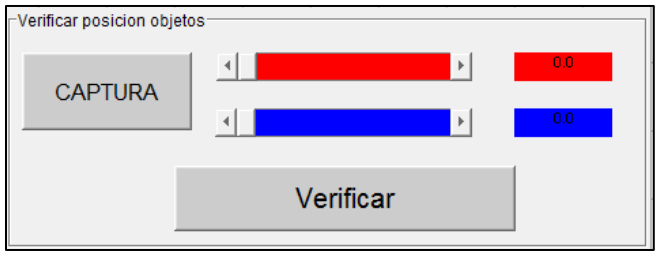


Figura 7.42 *Parte de la interfaz que verifica la posición de objetos*

El push button "*Captura*" tiene asociado el mismo procedimiento expuesto en el del apartado de la calibración. Una vez se pulsa se cargan dos imágenes binarias con el umbral inicial de 0.0. Existen dos slider: uno rojo que controla el umbral de la pieza roja, y uno azul que controla el de la pieza azul. Las imágenes binarias del canal azul y rojo se cargan en las figuras "*imagencalibraazul*" y "*imagencalibrarojo*", respectivamente. Según se va modificando el slider, va cambiando la imagen binaria. El código es el siguiente:

```
function capturaRojoAzul_Callback(hObject, eventdata, handles)
% hObject    handle to capturaRojoAzul (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global umbralR Im_binR IbwR umbralA Im_binA IbwA piezas
%%Inicializar video
vid = videoinput('winvideo', 2);
start(vid);
piezas = getsnapshot(vid); %Captura de imagen para los colores
umbralR=get(handles.sliderRojo,'Value');
umbralA=get(handles.sliderAzul,'Value');
gris=rgb2gray(piezas);
img_colorR=piezas(:,:,1);
img_colorA=piezas(:,:,3);
IbwR=img_colorR-gris;
IbwA=img_colorA-gris;
Im_binR=im2bw(IbwR,umbralR);
Im_binA=im2bw(IbwA,umbralA);
axes(handles.imagencalibrarrojo);
imshow(Im_binR);
axes(handles.imagencalibrarazul);
imshow(Im_binA);
axis off
%guidata(hObject,handles)
set(handles.calibrar,'Enable','off');
set(handles.verificar,'Enable','on');
```

Binarizadas las imágenes correctamente, se pulsa la opción de "*Verificar*". En la figura "*imagenPosicionPieza*", se carga la imagen capturada en color por la webcam, con los centroides de las dos piezas, rojo y azul, en color verde. Adicionalmente, se han programado errores, por si solo se calcula el centroide de una de las dos piezas o de ninguna.

El código siguiente recoge todo lo anterior:

```
function verificar_Callback(hObject, eventdata, handles)
% hObject    handle to verificar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clearvars a x num_obj g % se limpian las variables
global piezas color x POSX_OBJ_REAL POSY_OBJ_REAL num_objR num_objA
Pos_x Pos_y LongitudX_foto LongitudY_foto LongitudX_real
LongitudY_real umbralR umbralA
verificacion; %procesamiento de imagenes calculo de centroides
 if ((num_objR==1) && (num_objA==1))
  msgbox('Verificacion de los dos objetos completada','Verificacion');
  axis off;
 elseif ((num_objR~=1) && (num_objA==1))
    errordlg('No ha sido posible determinar la posicion del objeto
ROJO','Verificacion ROJO');
    axis off;
  elseif ((num_objR==1) && (num_objA~=1))
    errordlg('No ha sido posible determinar la posicion del objeto
AZUL',' Verificacion AZUL');
    axis off;
 else
    errordlg('No ha sido posible la verificacion','verificacion');
    axis off;
 end
%%%Control de apariencia de botones en interfaz
 set(handles.continuar,'Enable','on');
```

Ejecutar Programa

El push button "***Continuar***" (figura 7.43) se activa tanto en simulación como en real. La finalidad de esta *Callback* es llamar a la siguiente interfaz, la "***InterfazNeurosky***"

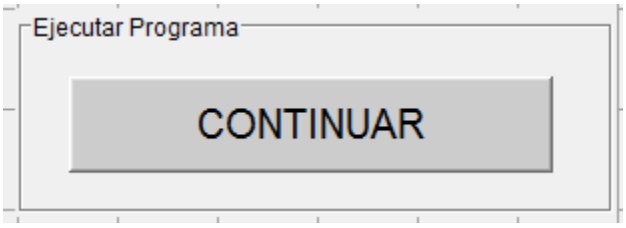


Figura 7.43 *Parte de la interfaz que permite continuar el programa*

El código es el siguiente:

```
function continuar_Callback(hObject, eventdata, handles)
% hObject    handle to continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global robot modo color Pos_x_objR Pos_y_objR Pos_x_objA Pos_y_objA
POS_X_OBJ POS_Y_OBJ LongitudX_real LongitudY_real LongitudX_foto
LongitudY_foto Pos_x Pos_y  real_x1 real_x2 real_y1 real_y2 robot

InterfazNeurosky; % INTERFAZ DATOS NEUROSKY
```

InterfazNeurosky

En la imagen siguiente se puede observar la apariencia de la interfaz desde la pantalla de edición de la herramienta GUIDE (figura 7.44).

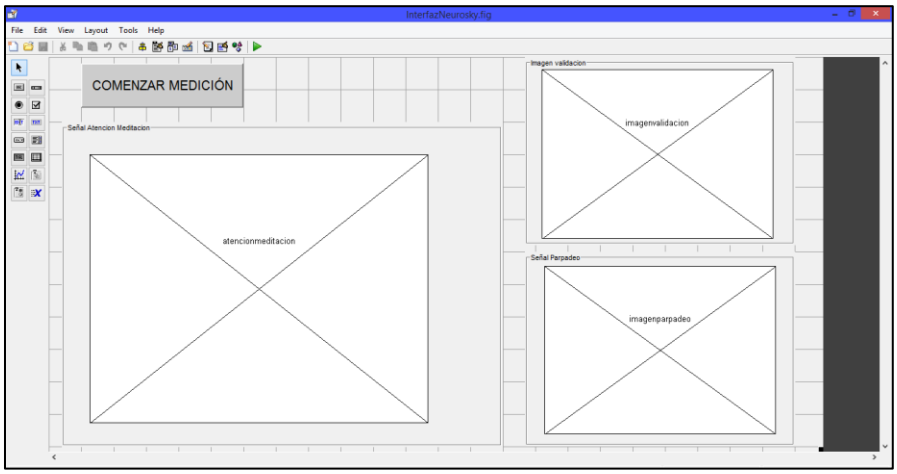


Figura 7.44 *InterfazNeurosky en la pantalla de edición GUIDE*

Esta interfaz está formada por un *push button* y tres figuras. El *push button* "***Comenzar Medición***" activa el programa que adquiere y procesa los datos del casco Mindwave. En la figura "***atencionmeditacion***" se muestran los datos que se recogen del casco, la atención y la meditación. En la figura "***imagenvalidación***" se van cargando las distintas imágenes necesarias para guiar al usuario: validar la respuesta, información sobre qué color se va a coger, información sino se valida la respuesta. Y, por último, en la figura "***imagenparpadeo***", se cargan los datos del parpadeo que se usa para validar el resultado (rojo o azul).

7.6. Resultados

7.6.1. Simulación

1er PASO: abrir el programa RobotStudio y la estación de trabajo "*TFG_ventosa_funcionajoint moverFINAL.rspag".*

2º PASO: en el apartado simulación, pulsar reproducir.

3er PASO: abrir en Matlab la "*InterfazMatlab.m*"

4º PASO: marcar la opción "*Simulación*" y el Push botton "*CONECTAR*".

5º PASO: ir a la ventana del robotStudio y comprobar si se conecta con el robot.

6º PASO: una vez el robot se ha conectado correctamente se pulsa el push botton "*CONTINUAR*"

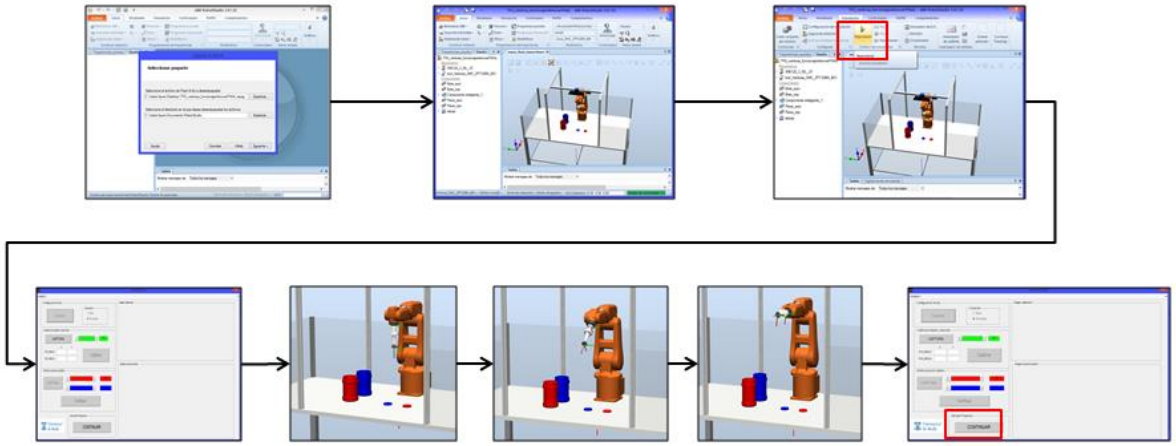


Figura 7.45 *Resultado conexión robot en simulación*

7º PASO: una vez pulsado "*CONTINUAR*" se carga en Matlab la "InterfazNeurosky".

8º PASO: pulsar el push botton "*COMENZAR MEDICIÓN*" y empiezan a cargarse las señales de la atención y de la meditación.

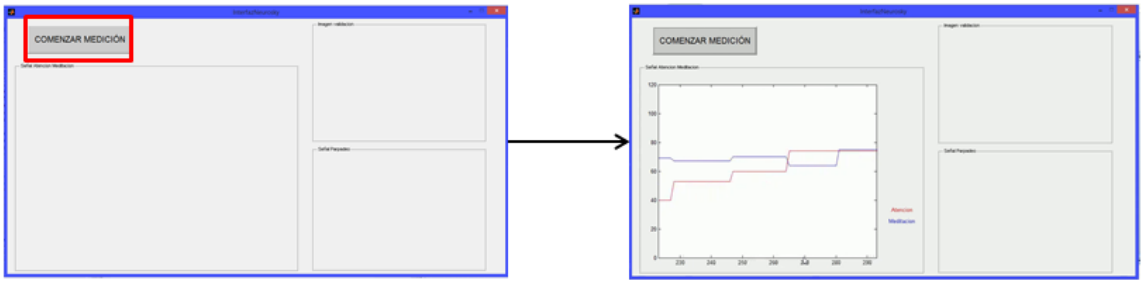


Figura 7.46 *Resultado comienzo medición con el casco*

9º PASO: cuando una de las dos señales destaca, aparece una imagen que pide la validación del resultado.

10º PASO: si se está de acuerdo con el resultado, se debe parpadear como mínimo dos veces de manera forzada apareciendo los datos en la interfaz.

11º PASO: si se ha validado el resultado, aparece una imagen con el texto: VALIDADO.

12º PASO: una vez validado, es necesario abrir la ventana de RobotStudio. De este modo se observa como el robot IRB120 coge el disco del color validado y lo deposita en el bote del color correspondiente.

RESULTADO MEDITACIÓN (COLOR AZUL)

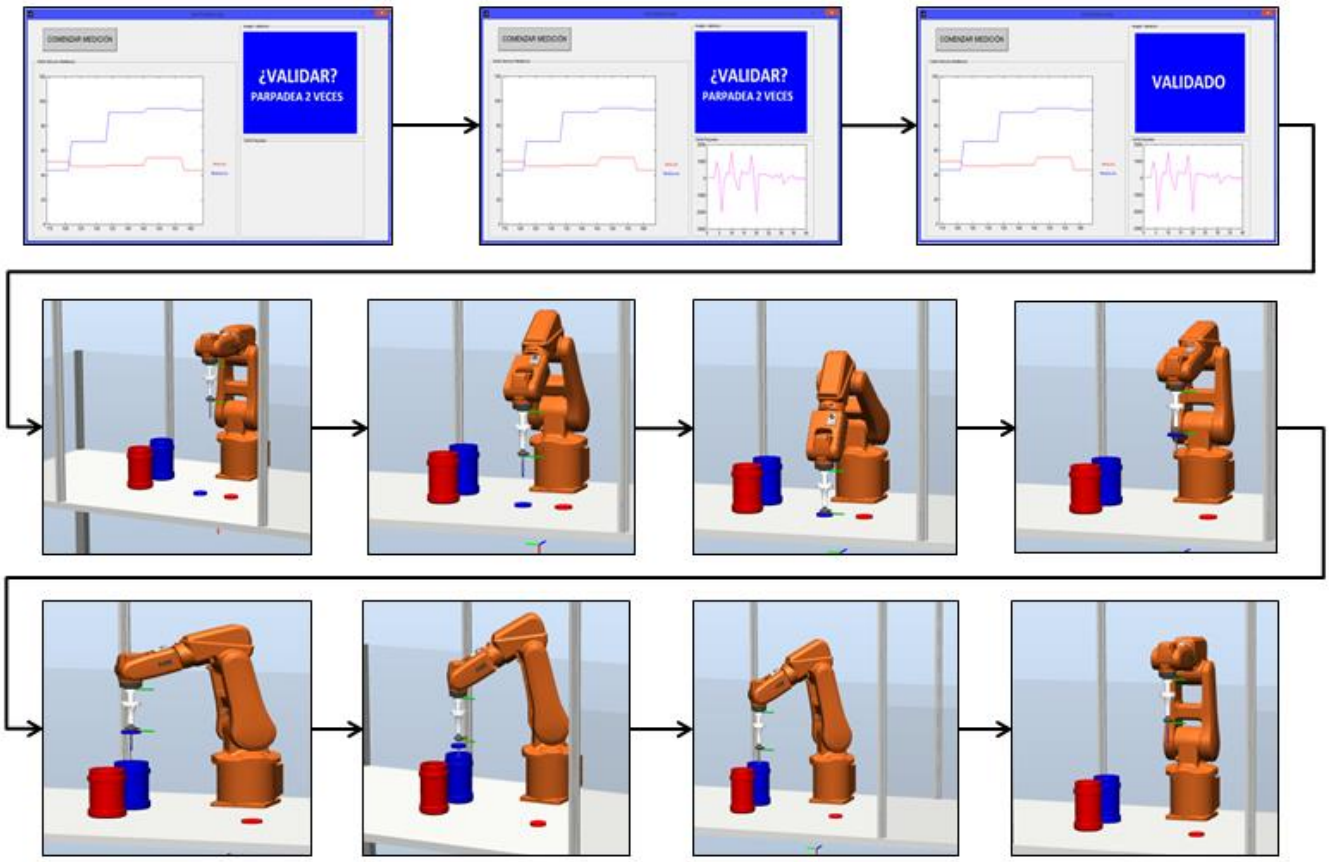


Figura 7.47 *Resultado con señal meditación*

RESULTADO ATENCIÓN (COLOR ROJO)

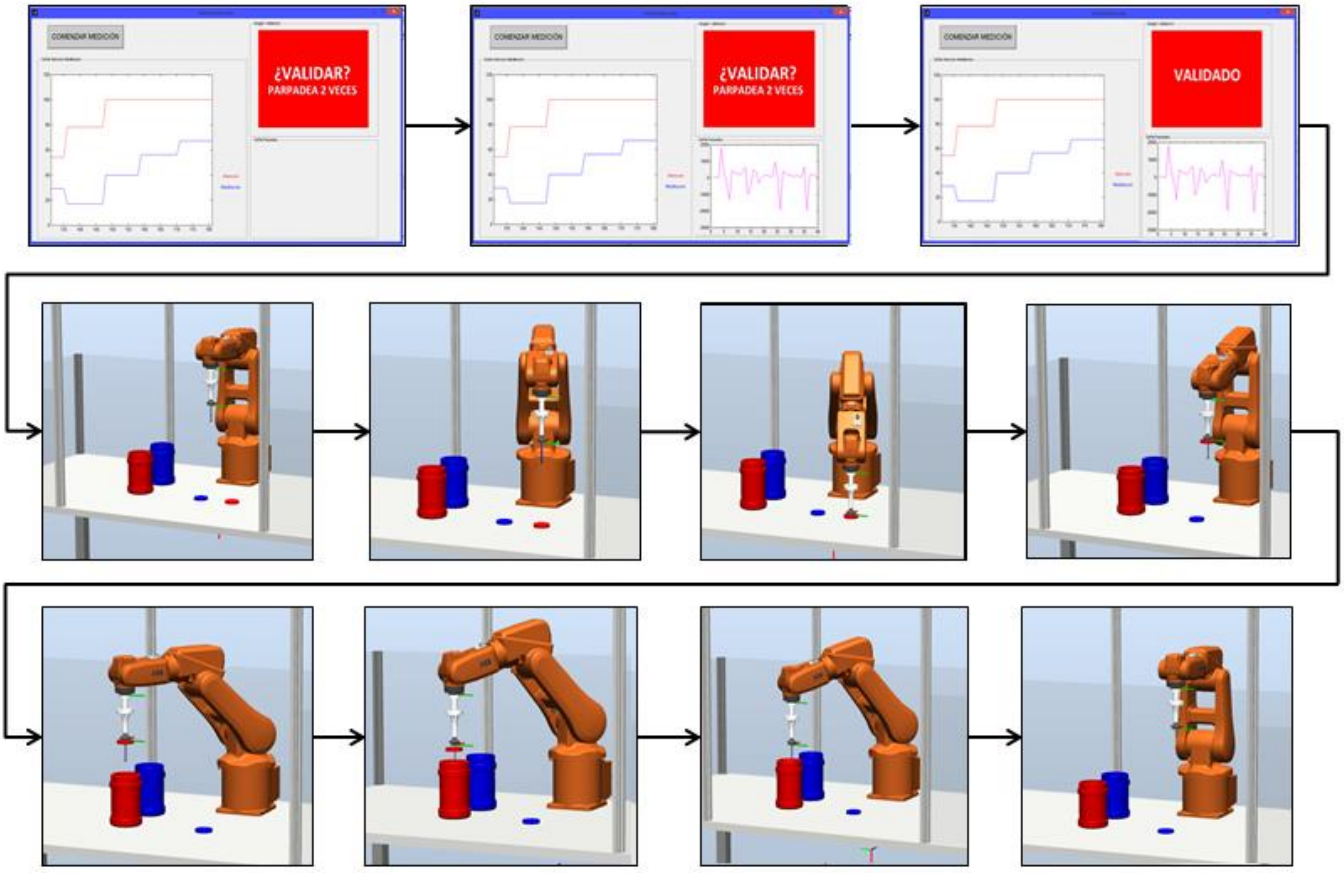


Figura 7.48 *Resultado con señal atención*

Si el resultado de las señales cerebrales no es el esperado, no se valida mediante el parpadeo y se siguen recogiendo datos.

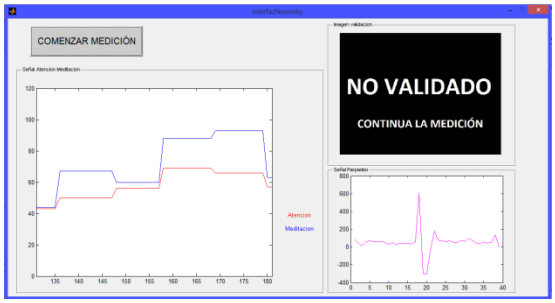


Figura 7.49 *Resultado no validado*

7.6.2. Real

1er PASO: encender el robot mediante el controlador IRC5.

2º PASO: abrir en Matlab la "*InterfazMatlab.m*".

3er PASO: marcar la opción "*Real*" y el Push botton "*CONECTAR*".

4º PASO: comprobar si conecta con el robot, mediante el movimiento del mismo.

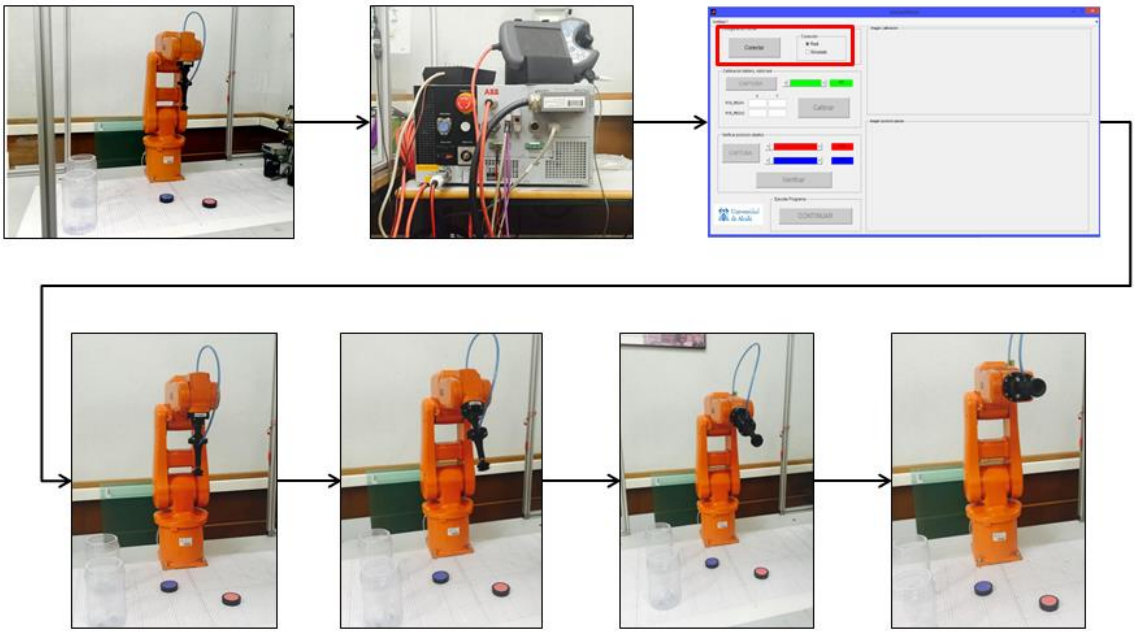


Figura 7.50 *Resultado conexión robot real*

5º PASO: se colocan las dos piezas verdes para la delimitación del tablero en esquinas opuestas.

6º PASO: en el apartado "calibración tablero", se pulsa el único botón activado "CAPTURA", hará una foto la webcam al tablero.

7º PASO: se introducen los valores de las esquinas de las piezas verdes, ayudándose de la plantilla.

8º PASO: se pulsa el botón "Calibrar", realizando la conversión pixel-mm para poder determinar las futuras posiciones de las piezas. Una vez realizada correctamente la calibración, se pasa a determinar la posición de las piezas. Las piezas verdes, se pueden dejar o quitar.

9º PASO: se colocan los dos discos, rojo y azul, donde se desee, pero siempre dentro del tablero delimitado por las piezas verdes.

10º PASO: en el apartado "verificar posición objetos", se pulsa el botón "CAPTURA", y se modifican los slider hasta conseguir una buena nitidez de las piezas.

11º PASO: se pulsa el botón "verificar", para determinar la posición exacta de las piezas. Esta parte se puede realizar todas las veces que se quiera.

Figura 7.51 *Resultado calibración tablero y verificación posición piezas*

12º PASO: una vez, se ha realizado correctamente la calibración y la verificación, se pulsa el botón "CONTINUAR".

Figura 7.52 *Continuar con la ejecución del programa*

13º PASO: una vez pulsado "*CONTINUAR*" se carga en Matlab, la "InterfazNeurosky".

14º PASO: pulsar el push botton "*COMENZAR MEDICIÓN*" y se empiezan a cargar las señales de la atención y de la meditación.

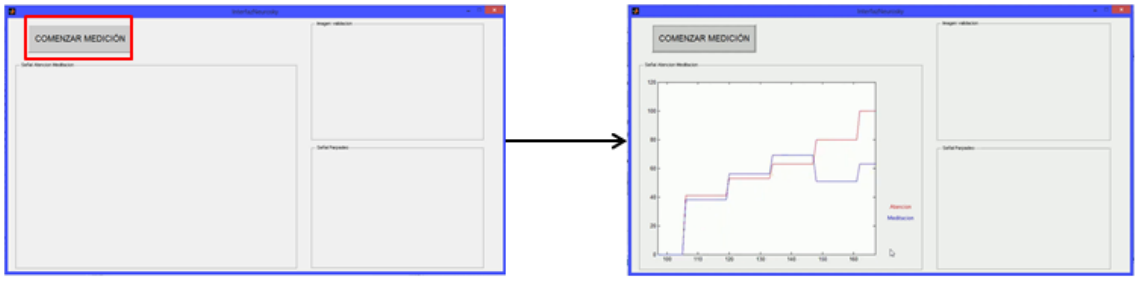


Figura 7.53 *Resultado comienzo medición con el casco*

15º PASO: cuando una de las dos señales destaca, aparece una imagen que pide la validación del resultado.

16º PASO: si se está de acuerdo con el resultado, se debe parpadear mínimo dos veces de manera forzada apareciendo los datos en la interfaz.

17º PASO: si se ha validado el resultado, aparece una imagen con el texto: VALIDADO.

18º PASO: una vez validado, se observa como el robot IRB120 coge el disco del color validado, y lo deposita en el bote del color correspondiente.

RESULTADO MEDITACIÓN (COLOR AZUL)

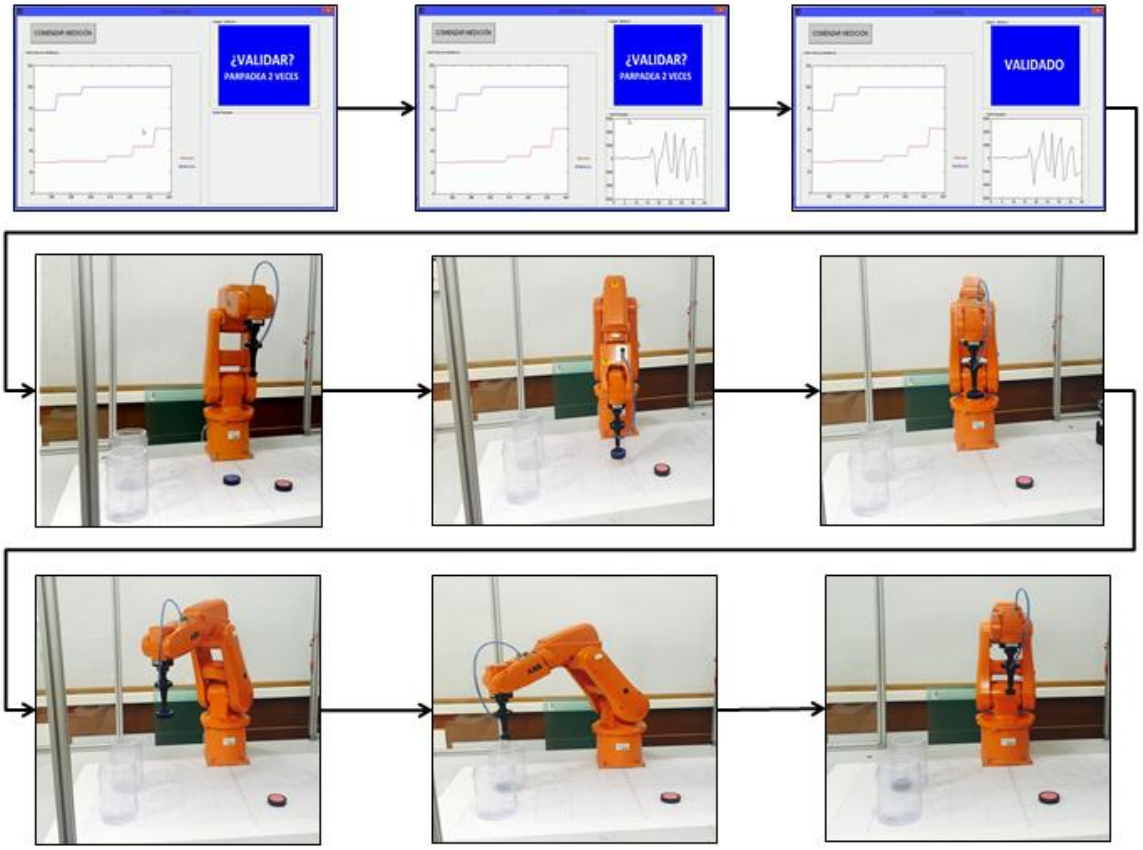


Figura 7.54 *Resultado con señal meditación en sistema real*

RESULTADO ATENCIÓN (COLOR ROJO)

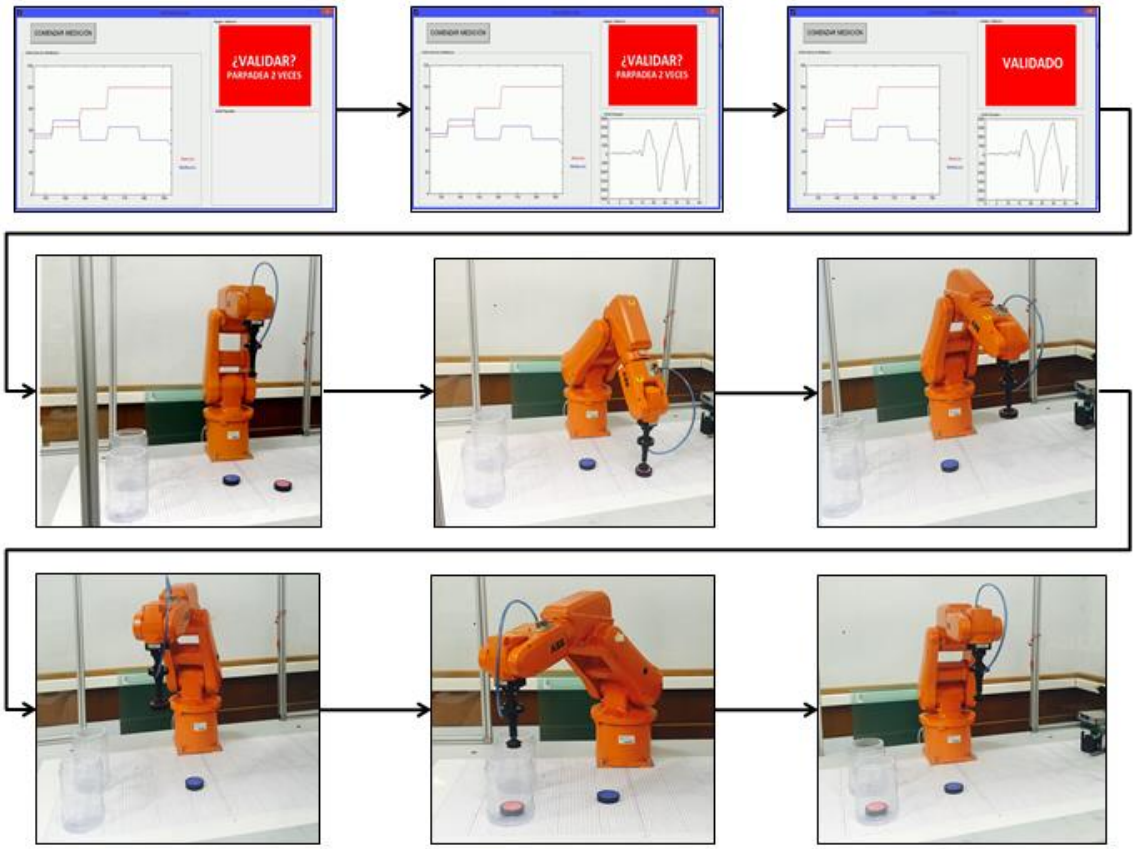


Figura 7.55 *Resultado con señal atención en sistema real*

8. Conclusiones y futuras líneas de trabajo

El objetivo principal del proyecto era la creación de un sistema Brain Computer Interface.

Los sistemas BCI **endógenos**, al igual que el estudiado en este proyecto, se basan en la capacidad del hombre para modificar su actividad cerebral de forma voluntaria. Este tipo de BCI requiere un **entrenamiento** a fin de mejorar el control. Los sistemas endógenos se basan en **establecer dos estados**: si-no, delante-atrás, etc. Por este motivo también se ha buscado una respuesta binaria en este proyecto.

La primera opción fue utilizar los colores rojo y azul como respuesta binaria. Se realizaron numerosas pruebas con el casco para lograr dicha respuesta, estudiando los diferentes tipos de ondas que nos proporcionaba el casco. Sin embargo, no se obtuvieron los resultados esperados debido a que los colores se perciben en la parte posterior del cerebro, y el casco Mindwave no tiene sensores en esa zona.

Posteriormente, en lugar diferenciar entre colores, se optó por diferenciar entre damero y no damero, ya que hay estudios en los que aparecen diferencias en las ondas provocadas por estas dos opciones. En este caso tampoco se obtuvieron resultados concluyentes al no obtenerse suficientes datos, dado que se utiliza un casco no-científico con un solo sensor.

Otra de las opciones que se barajaron fue el estudio del movimiento de la mano, basado en las ondas gamma que son las ondas que mejor registran los movimientos motores. La idea era realizar un movimiento continuo mano abierta (0,25 segundos), mano cerrada (1,75 segundos) y así durante 2 minutos seguidos, con el fin de comprobar si se apreciaban diferencias en dicha onda. Esta opción tampoco concluyó con la respuesta binaria esperada. El problema estribaba en que la señal gamma no se adquiría a la velocidad necesaria, con el inconveniente añadido de que los datos se cargan en un vector, lo que va ralentizando la adquisición de datos gamma. Esto generaba la imposibilidad de comparar la señal correspondiente al movimiento de abrir-cerrar al principio de la prueba y al final. Una de las soluciones que se intentó para resolver este problema fue estudiar la señal raw, en lugar de la señal gamma. La idea era filtrar la señal raw que se adquiría a la velocidad necesaria– en la frecuencia de la señal gamma (entre 30-60 Hz), que es la que verdaderamente proporciona la información del movimiento. Finalmente esta idea tampoco resulto útil, puesto que la señal raw también captaba el parpadeo, lo que ocasionaba un problema a la hora de filtrar esta señal para obtener la gamma, al sumarse parte de señal perteneciente al parpadeo (u otros artefactos), lo que impedia observar las diferencias entre los movimientos mano abierta-mano cerrada en la señal gamma filtrada.

Por tanto, después de mucho estudio y sabiendo las limitaciones que un casco con un solo sensor tenia, se optó por la utilización de las dos señales cognitivas ofrecidas por el casco, la atención y la meditación. Estas señales las proporciona el casco Mindwave utilizando el algoritmo eSense, obteniéndose un resultado fácil de manejar y de mucha utilidad para este tipo de proyectos. Al final, la respuesta binaria usada en el proyecto fue la atención, relacionada con el color rojo, y la meditación, relacionada con el color azul, usando como validación de respuesta el parpadeo forzado del usuario. Una vez validado el resultado, se enviaba toda la información al robot por medio de datagramas.

Como ya se ha indicado, al realizar el estudio con un casco de un único sensor, y como se ha comprobado a lo largo del proyecto, las señales fluctúan de forma significativa por lo que se hace imprescindible cualquier tipo de validación de la respuesta.

En cuanto a posibles futuras líneas de trabajo, siguiendo con la utilización del casco Mindwave de Neurosky, la mayoría están basadas en la atención y la meditación ya que son el resultado del algoritmo eSense creado por la empresa. En vez de relacionar la atención y la meditación con el movimiento de un robot entre dos piezas, esta respuesta binaria se podría utilizar para mover cualquier sistema teledirigido como coches o drones. Entre las múltiples posibilidades diferentes se consideran factibles relacionar la atención con el movimiento hacia delante, la meditación con el movimiento hacia atrás y el parpadeo para validar la respuesta. En este caso sería primordial controlar estos dos estados, con el objetivo de conseguir mantenerlos durante el mayor tiempo posible.

Actualmente y dentro del campo de la bioingeniería se están consiguiendo grandes avances relacionados con la creación de miembros biónicos. Un futuro campo de aplicación de este proyecto sería conseguir manejar el movimiento de una mano biónica usando otro casco como el *emotiv epoc* puesto que, al contar con un elevado número de sensores, facilitaría la obtención de numerosos datos, lo que permitiría realizar estudios más fiables y completos.

Otro de los campos de aplicación que en estos momentos están presentando un gran interés son las Redes Neuronales Artificiales. Mediante la utilización de éstas, seríamos capaces de proporcionar una gran base de datos de aprendizaje para el casco Mindwave, que serviría para obtener una respuesta de validación mucho más rápida y personalizada para cada usuario.

V Manual de usuario

9. Manual de Usuario

En este capítulo se detallará un manual para el usuario, con el cuál se podrá acceder a cualquier funcionalidad de la aplicación. Se explicarán los paneles de cada una de las aplicaciones desarrolladas y su función dentro del programa. También se elaborará una guía de botones con los que se ayudará al usuario a entender mejor el funcionamiento del sistema.

La interfaz GUI está desarrollada para que, al ser ejecutada la aplicación o simplemente escribiendo en la ventana de comandos de Matlab el nombre del archivo, aparezca la ventana de la Interfaz lista para ser usada (figura 9.1).

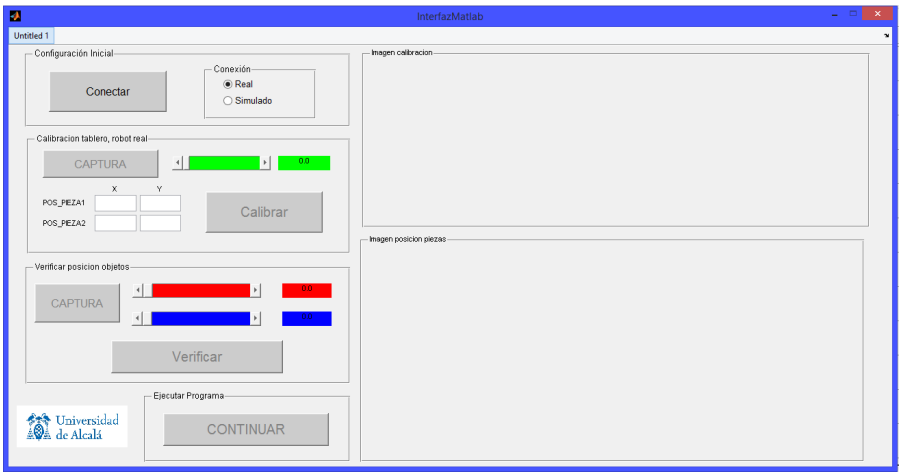


Figura 9.1 *Vista de la Interfaz al ser ejecutada*

En la figura anterior, figura xx, se puede observar lo que ocurre si desde la ventana de comandos de Matlab introducimos la siguiente línea de código:

```
>>InterfazMatlab
```

El siguiente paso es ejecutar la aplicación de RobotStudio para que el servidor se quede a la escucha. Esto se realiza pulsando el botón de *reproducir* en la pestaña de simulación (figura 9.2), o bien utilizando el controlador IRC5 y el servidor para el Robot real.

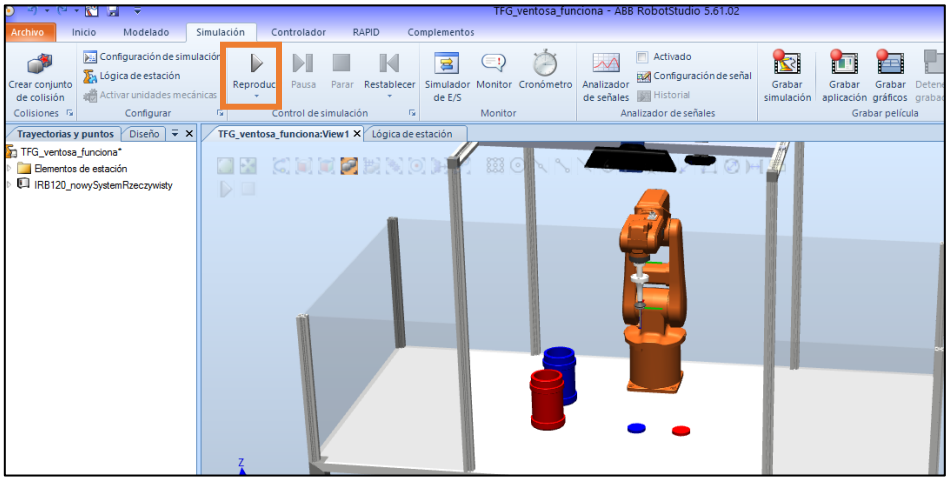


Figura 9.2 *Paso ejecutar la aplicación del sistema simulado*

El primer paso general es la **"Configuración inicial".** En este punto se selecciona el tipo de conexión con el robot, real o simulación. Se hace clic en **Conectar** (figura 9.3).

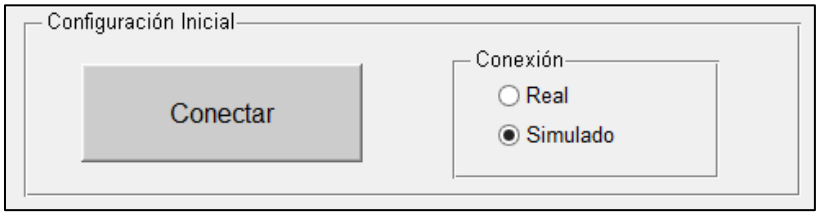


Figura 9.3 *Parte de la interfaz que conecta con el robot*

Una vez configurado el tipo de conexión, el siguiente paso es determinar la posición de los objetos. En simulación, la posición de los objetos es preestablecida, mientras que en el sistema real, la posición puede variar. Dependiendo de la conexión se activará el siguiente módulo (figura 9.4):

- **Conexión real**: Calibración tablero, robot real.

- **Conexión simulado**: Continuar.

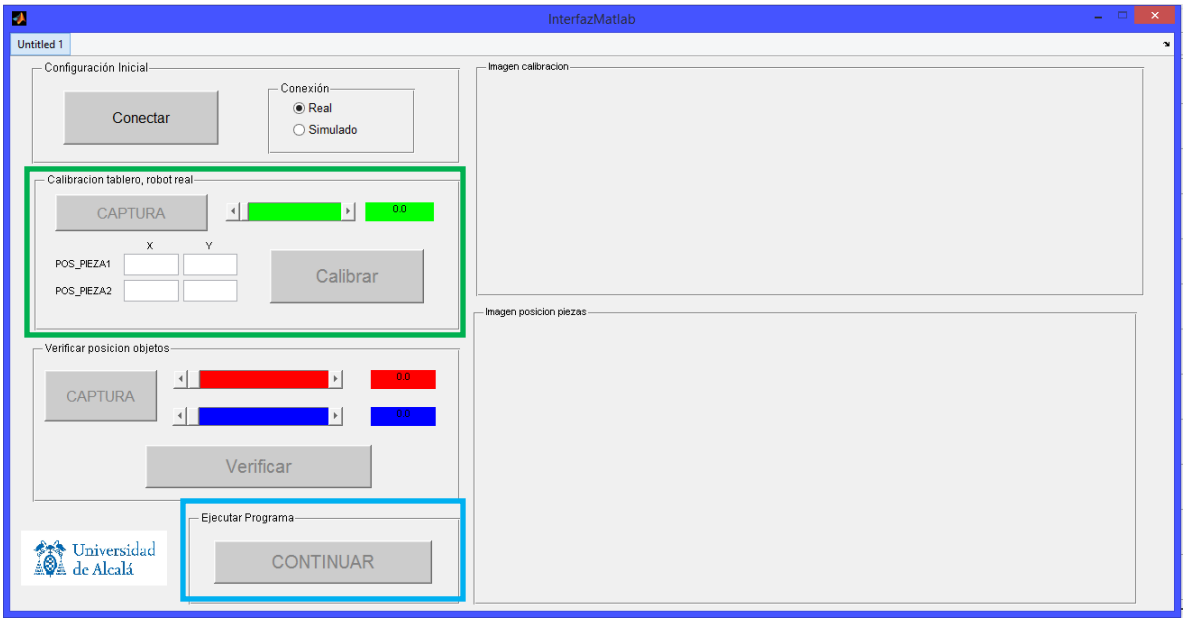


Figura 9.4 *Pasos a seguir dependiendo de la conexión*

Conexión real

Figura 9.5 *Sistema real completo*

1º. CALIBRACIÓN TABLERO, ROBOT REAL

1er PASO. Colocación piezas verdes

Para llevar a cabo la calibración del tablero y delimitar el área de trabajo, se usarán dos piezas cuadradas verdes de 10 mm de longitud. Estas piezas se colocarán en **diagonal**, quedando de la siguiente forma (figura 9.6).

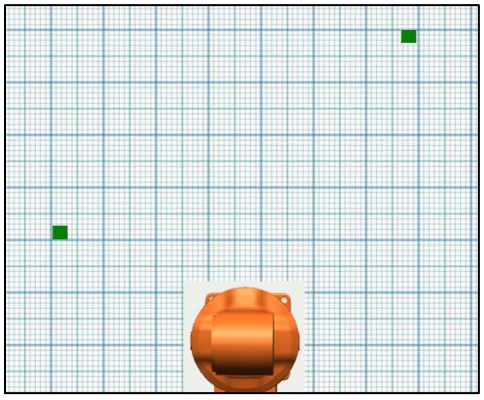


Figura 9.6 *Ejemplo colocación piezas verdes para calibración*

Una vez colocadas las piezas se procede a realizar la fase de calibración y delimitación del tablero.

2º PASO. CAPTURA imagen del área de trabajo. Se activa la opción una vez marcada la opción Conexión real (figura 9.7).

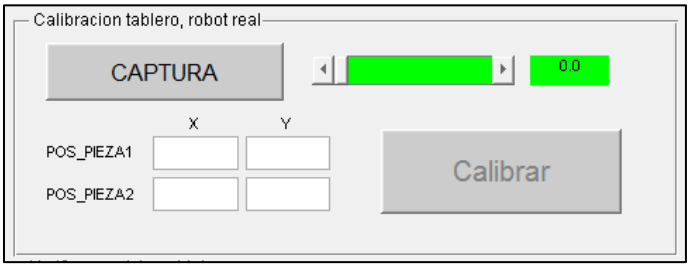


Figura 9.7 *Vista de la parte de calibración tablero*

La opción **CAPTURA**, hace una foto al área de trabajo para calibrar el tablero. La imagen que se carga en la interfaz es la imagen de las dos piezas binarizadas, con un umbral inicial 0,0 (figura 9.8).

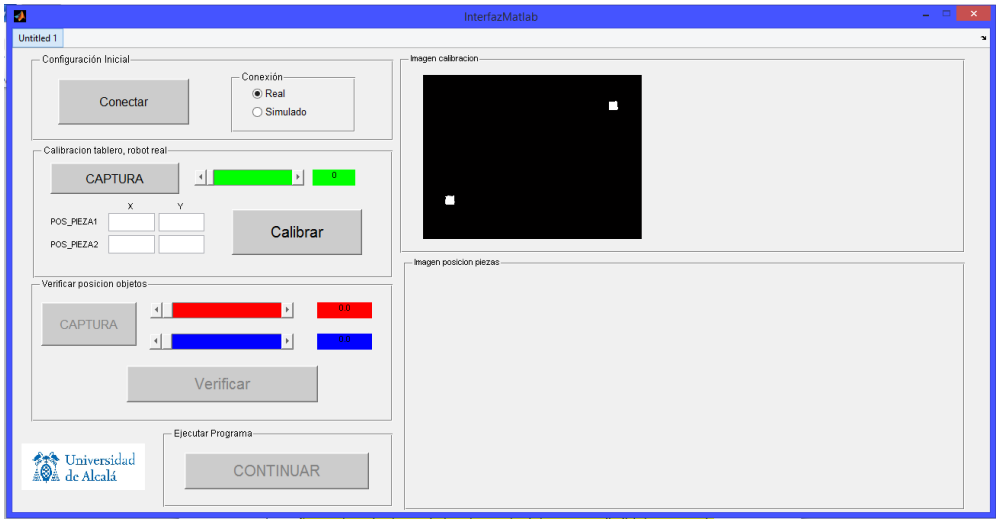


Figura 9.8 *Resultado de la captura de calibración*

Si las piezas no se ven correctamente, se usa el slider que cambia el **umbral de la imagen entre 0 y 1**, hasta que se vean más nítidas. El valor del umbral se muestra en el rectángulo verde al lado del slider (figura 9.9).

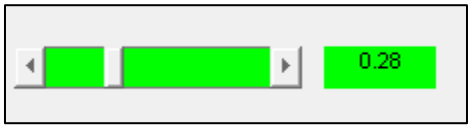


Figura 9.9 *Slider para modificar el umbral entre 0 y 1*

3er PASO. Posición real de las piezas (figura 9.10).

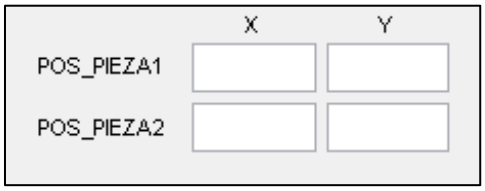


Figura 9.10 *Introducir los datos de las coordenadas por pantalla*

Las coordenadas de las esquinas (mm) de las piezas verdes se introducen por la interfaz gráfica. Se han creado cuatro cuadros de texto para las dos piezas "**POS_PIEZA1** y **POS_PIEZA2**" (la pieza 1 puede ser cualquiera de las dos).

Los datos que se usan son las **esquinas exteriores** (figura 9.11). Se usará la cuadrícula para ayudar a determinar los valores de las coordenadas.

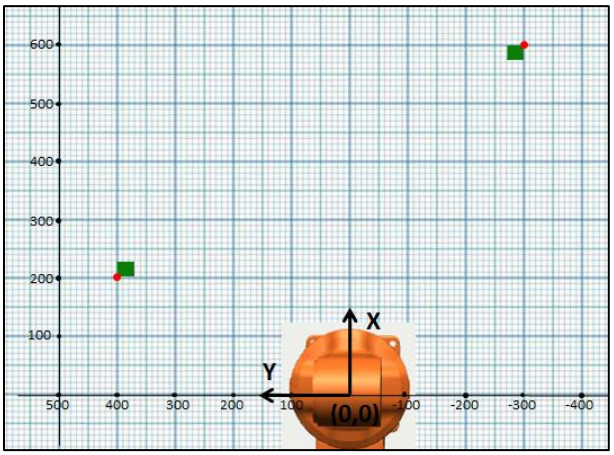


Figura 9.11 *Puntos necesarios para la calibración*

ADVERTENCIA *Las coordenadas son respecto al* ***origen del robot*** *(figura 9.11). Los valores X e Y se determinan teniendo en cuenta el* ***eje de coordenadas del robot*** *(figura 9.11).*

Los datos introducidos por la interfaz tienen que ser **NUMÉRICOS**, si se introduce otro tipo de dato, aparece un error (figura 9.12).

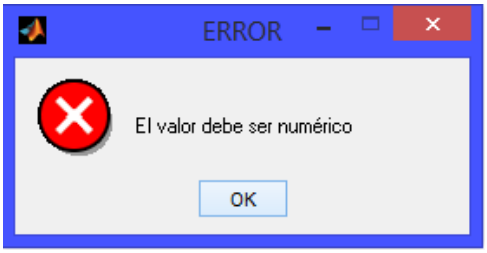


Figura 9.12 *Error dato no numérico*

4º PASO. Calibración

Una vez capturada la imagen e introducidos los datos de las coordenadas, se activa el *push button* **CALIBRAR** (figura 9.13).

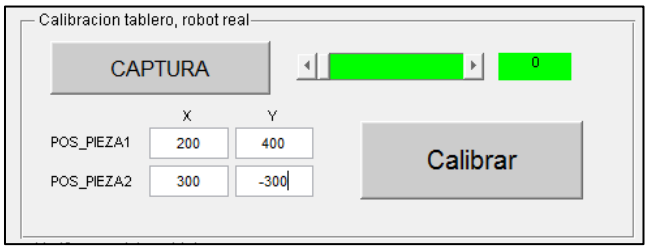


Figura 9.13 *Parte de la interfaz de la calibración con los push button activados y los datos de las coordenadas introducidos*

Una vez pulsado, se carga en la interfaz una imagen a color RGB con los centroides pintados (figura 9.14).

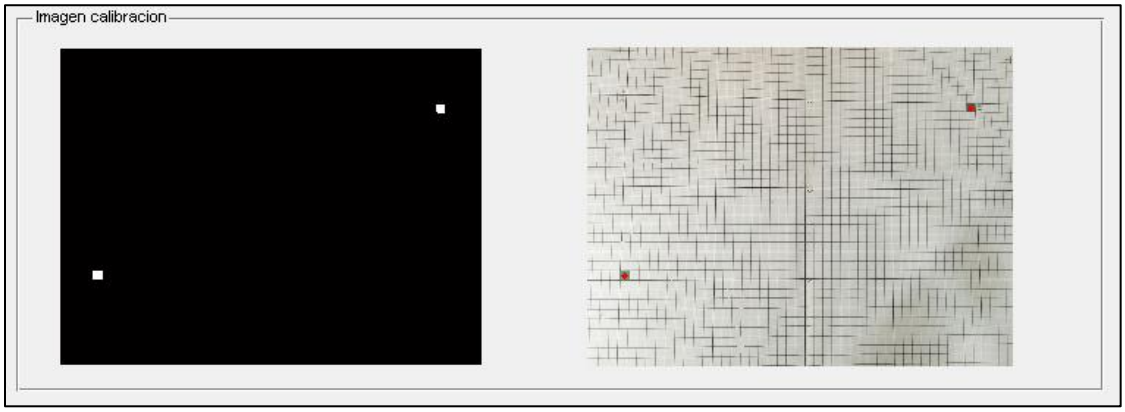


Figura 9.14 *Resultado final de la calibración*

Si la calibración se realiza satisfactoriamente, nos aparece un mensaje como el mostrado en la figura 9.15.

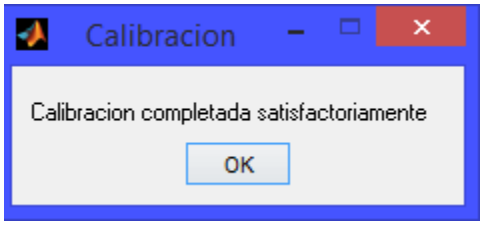


Figura 9.15 *Mensaje de calibración completada*

Si hay algún problema con la calibración, por ejemplo, los datos introducidos por pantalla son incorrectos o no se detecta alguna de las piezas verdes, aparece un error como el de la figura 9.16.

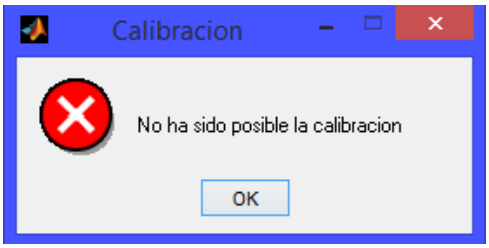


Figura 9.16 *Mensaje error de calibración*

Por otro lado, se calcula la relacion entre la **longitud del tablero real** (los datos introducidos en la interfaz), con la **longitud entre centroides** de las piezas en el frame, para determinar la posicion de los centroides de las piezas en pasos posteriores. Generando la **conversión pixel-mm**

2º. VERIFICACIÓN POSICIÓN PIEZAS:

1er PASO. Colocación piezas en el área de trabajo.

Cuando se ha calibrado y delimitado el área de trabajo, se colocan los dos discos en la posición que se quiera dentro del área de trabajo. En la siguiente figura (figura 9.17) se marca la **delimitación del área de trabajo** y una posible colocación de los discos.

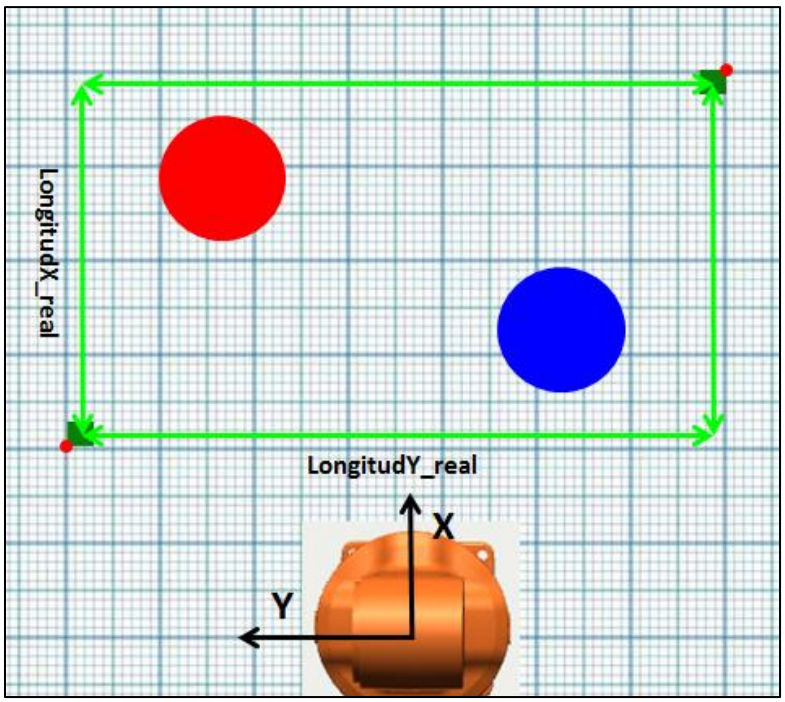


Figura 9.17 *Ejemplo colocación de los discos discos en el área de trabajo*

2º PASO. CAPTURA imagen del área de trabajo. Se activa el *push button* una vez realizada la calibración satisfactoriamente (figura 9.18).

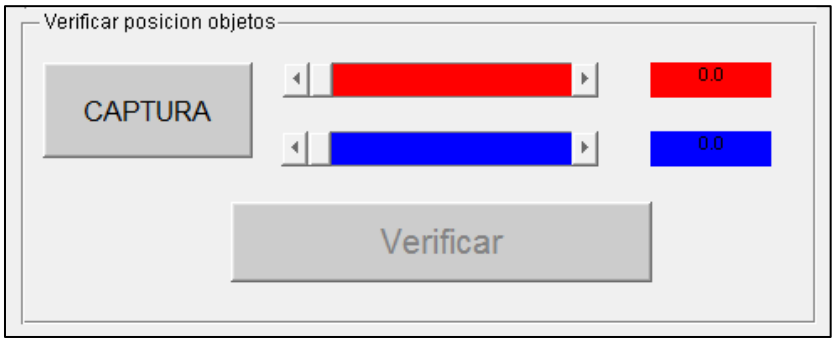


Figura 9.18 *Vista de la parte de Verificación de la posición de las piezas*

La opción **CAPTURA** hace una foto al área de trabajo para determinar la posición de las piezas y se carga la imagen **binaria roja** y la imagen **binaria azul** con un **umbral inicial 0,0** (figura 9.19).

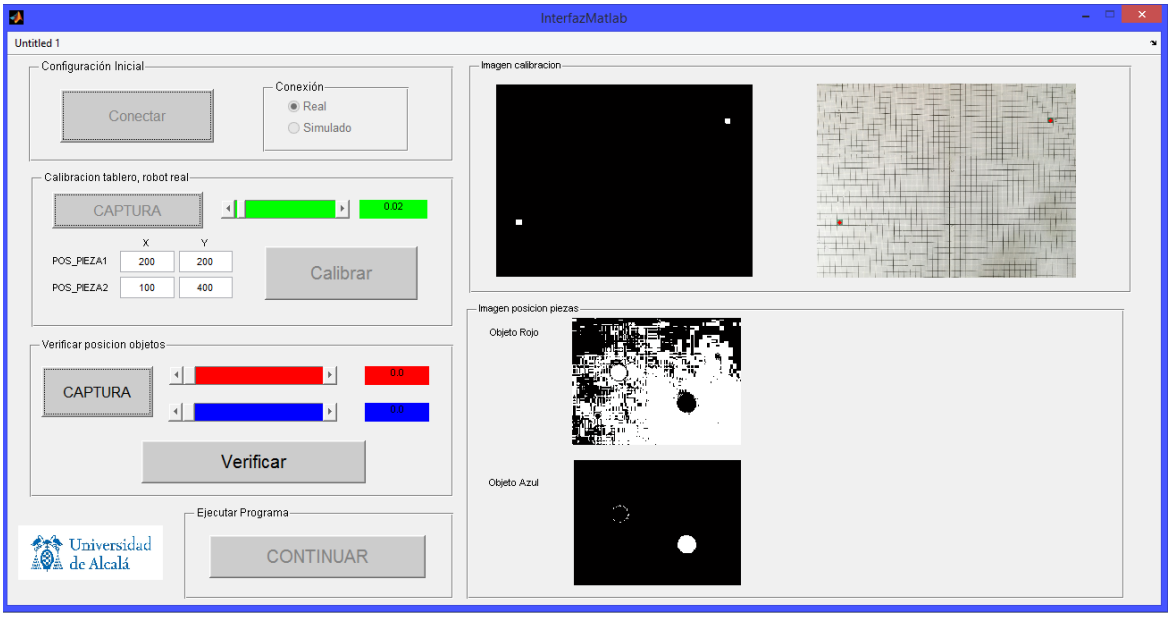


Figura 9.19 *Resultado de la captura de verificación*

En este caso, la **pieza azul** se ha detectado sin problemas y está perfectamente delimitada (no haría falta tocar el umbral), mientras que la **pieza roja** no se ha encontrado, por lo que se usa es slider que modifica el **umbral de la imagen** hasta que se visualice como la pieza azul (figura 9.20).

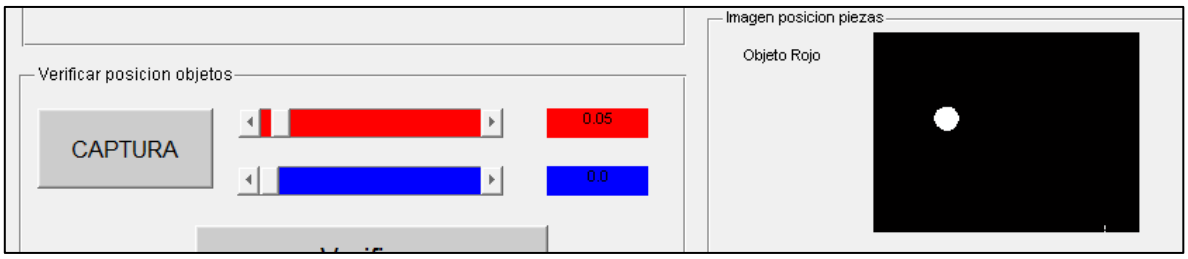


Figura 9.20 *Pieza roja después de usar el slider*

3er PASO. Verificación

En el momento en que las imágenes están binarizadas correctamente, se continúa con la determinación de la posición real, Pulsando el push button **Verificar** y cargando una imagen en color con los centroides pintados en los discos y recuadrados (figura 9.21)

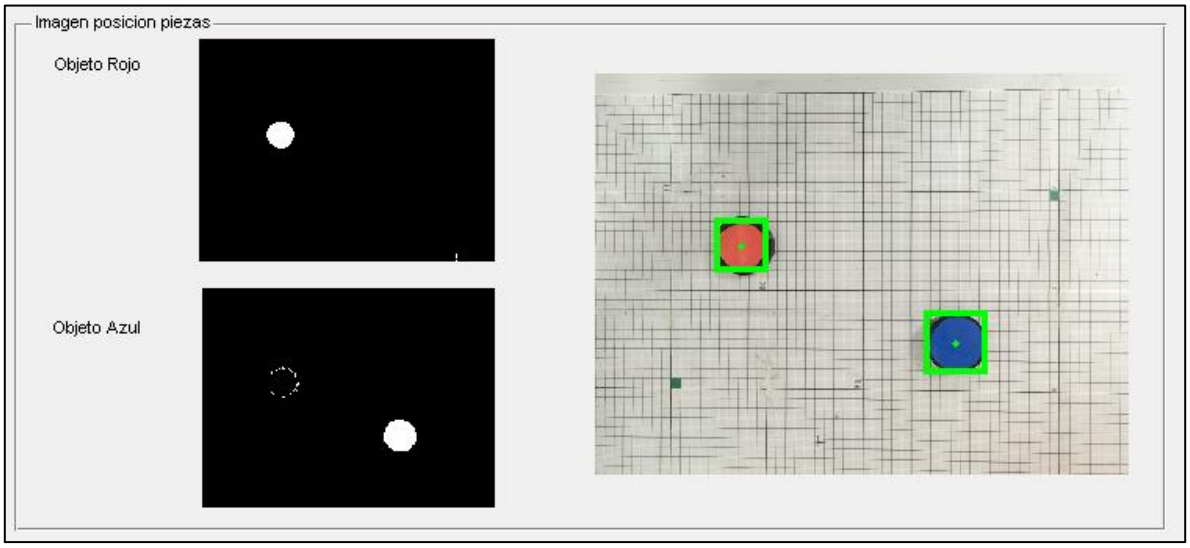


Figura 9.21 *Resultado de la verificación*

Si la verificación se realiza correctamente salta un mensaje (figura 9.22)

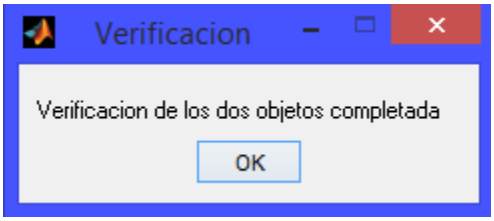


Figura 9.22 *Mensaje verificación completada*

Si sólo se determina la posición de una de las piezas, se carga la imagen en color con el centroide del disco detectado y mostrándose el siguiente error (figura 9.23).

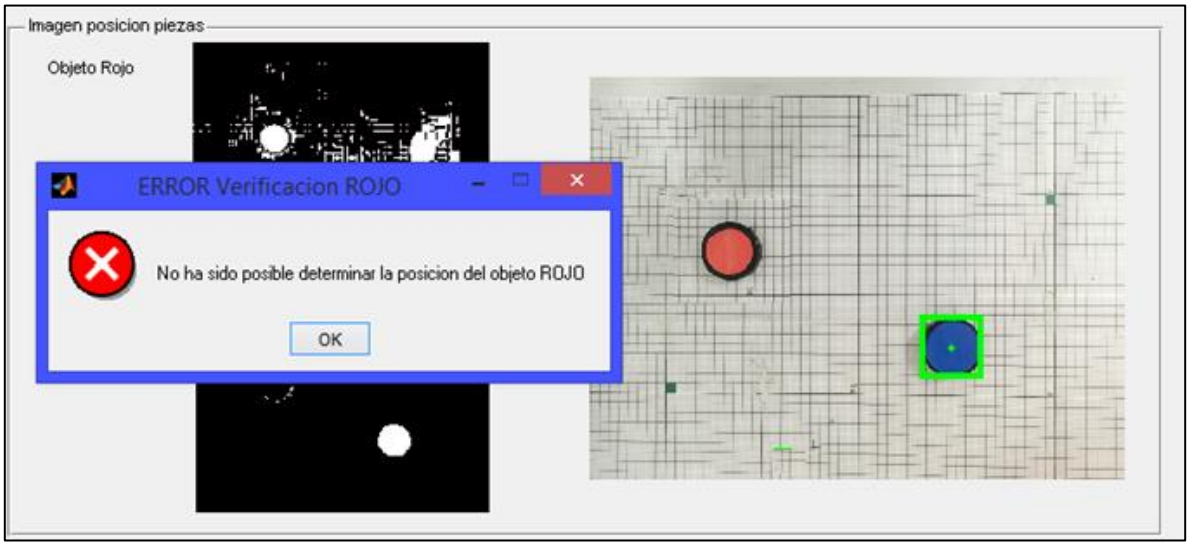


Figura 9.23 *Error de detección disco rojo*

Si por el contrario no se detecta ningún disco, o el centroide está fuera del área de trabajo, aparece el siguiente error (figura 9.24)

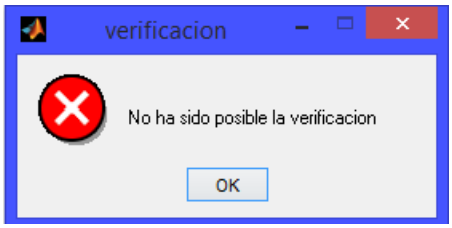


Figura 9.24 *Error de detección de las dos piezas*

Conexión real y simulada

3º CONTINUAR:

Este siguiente paso es común a la **conexión real y simulación**. Cuando se pulsa el botón **CONTINUAR** (figura9.25), se carga la siguiente interfaz "InterfazNeurosky" para ejecutar el programa de adquisición de la señal de la atención y la meditación del casco.

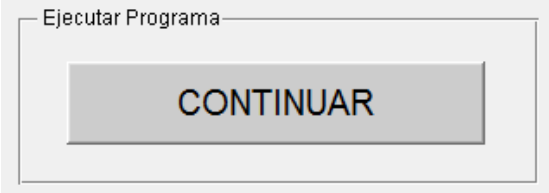


Figura 9.25 *Parte de la interfaz que permite continuar el programa*

Se necesita confirmación (figura 9.26) del paso a la siguiente interfaz, ya que luego no se puede volver a la actual.

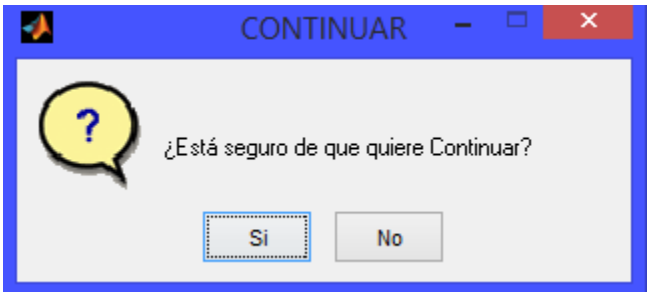


Figura 9.26 *Mensaje de confirmación*

InterfazNeurosky

Una vez, pulsado el *push button* **CONTINUAR**, se abre la "InterfazNeurosky" (figura 9.27). Esa parte recoge la adquisición y procesamiento de los estados cognitivos atención y meditación.

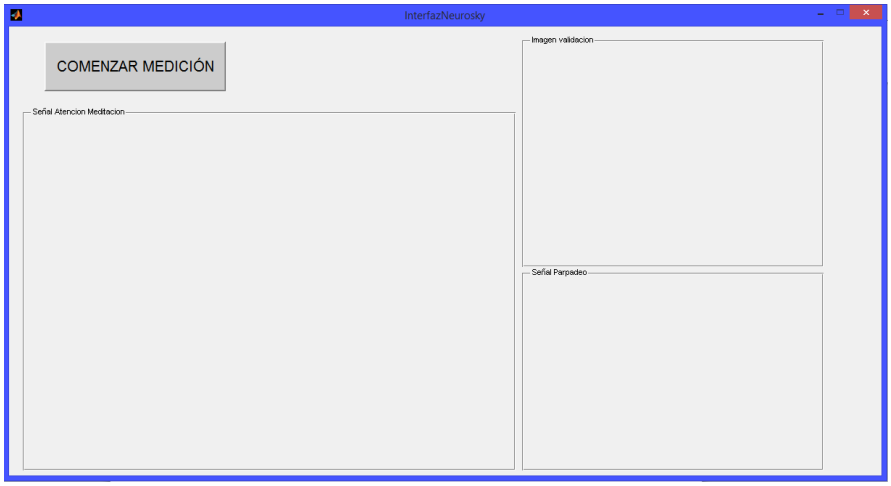


Figura 9.27 *Vista general de la Interfaz Neurosky*

Esta parte es igual tanto para el sistema simulado, como para el sistema real. Si se está en el sistema real ya se ha determinado el centroide de las piezas en la imagen y se ha hecho la conversión a mm. Y si se está en simulación la posición de las piezas es conocida.

1er PASO. COMENZAR MEDICIÓN.

Se pulsa el *push button* **COMENZAR MEDICIÓN** y en la parte de "***Señal Atención Meditación***" aparecen las señales de atención y meditación de forma dinámica moviéndose el medidor eSense (figura 9.28).

La señal de la atención es el color rojo y la señal de la meditación el color azul.

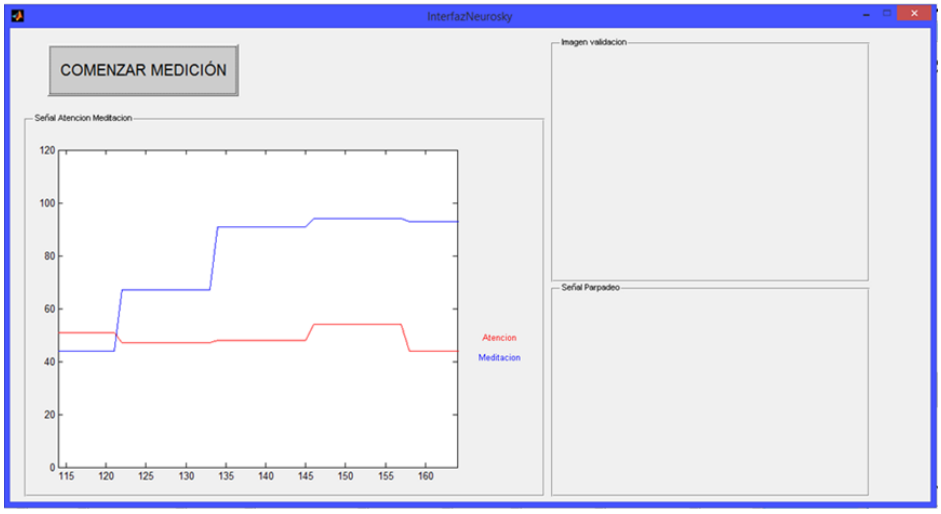


Figura 9.28 *Vista de la Interfaz Neurosky durante la adquisición*

En el momento que alguna de las dos señales destaca por encima de la otra, se pasa a la **validación del resultado.**

<u>2º PASO</u>. Validación.

Cuando una de las dos señales destaca por encima de la otra, aparece una imagen en el apartado "***Imagen Validación***". Las imágenes rojas están asociadas a la señal de la atención y las imágenes azules están asociadas a la señal de la meditación

En el siguiente ejemplo (figura 9.29) ha destacado la señal azul, el estado de meditación, por lo tanto en la "imagen validación" se carga la imagen con la pregunta: *si quieres validar el resultado parpadea dos veces.*

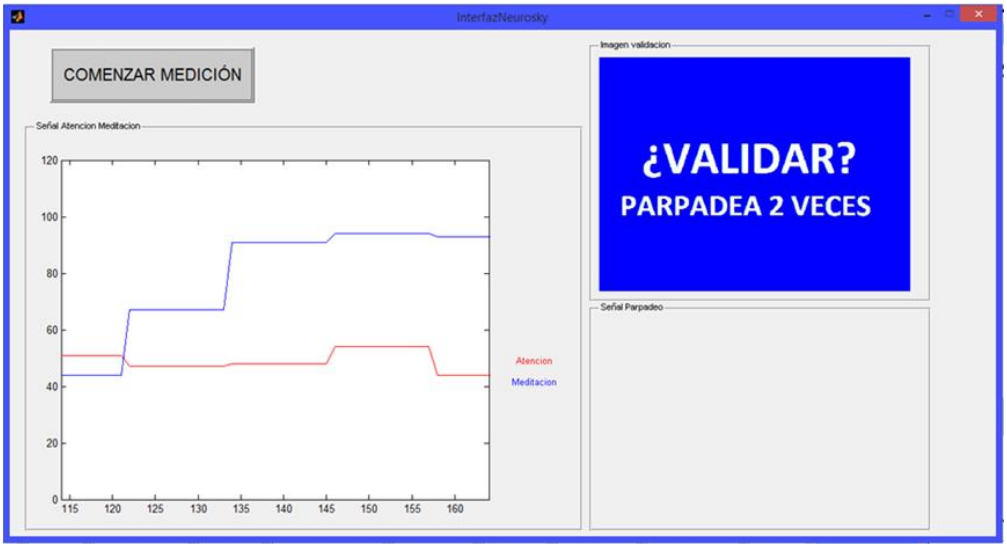


Figura 9.29 *Destaca la señal meditación ¿desea validar?*

Para que se considere validado, se debe **parpadear mínimo dos veces de manera forzada o muy forzada**, es decir por encima del **valor 600**, ya que si se realiza un parpadeo débil se puede confundir con un parpadeo involuntario.

Los valores se recogen en la siguiente tabla (tabla 9.1).

UMBRAL PARPADEO (valores absolutos)	
Parpadeo involuntario	0 a 599
Parpadeo forzado	600 a 1499
Parpadeo muy forzado	Valores mayores que 1500

Tabla 9.1 *Rango de valores del parpadeo*

Una vez mostrada la pregunta de validación, aparecerá la **señal parpadeo** debajo (figura 9.30). Se tienen 4 segundos para validar o no validar (figura 9.30).

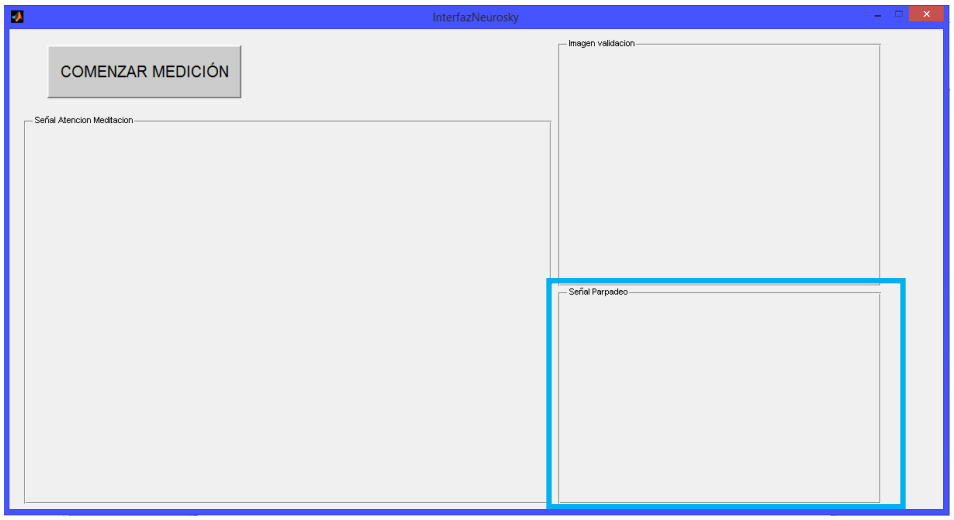


Figura 9.30 *Localización en la interfaz de la señal parpadeo*

Siguiendo con el ejemplo anterior, se quiere validar el resultado (figura 9.31)

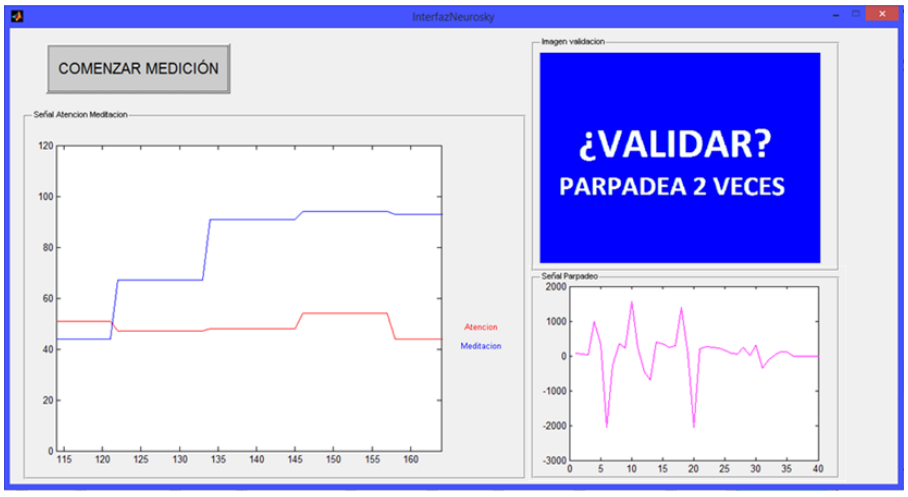


Figura 9.31 *Señal de parpadeo para validación*

Como se observa, en los 4 segundos se han producido tres parpadeos forzados, parpadeos involuntarios etc… Al haberse efectuado más de dos parpadeos forzados, se ha validado sin problema.

Si la respuesta ha sido validada correctamente y dentro de tiempo, en la "*ImagenValidación*" se carga una imagen con el mensaje "**VALIDADO**" y el fondo del color validado (figura 9.32)

Figura 9.32 *Validado el resultado (meditación)*

Si por el contrario, la respuesta no ha sido validada, bien porque no se ha querido validar ya que no era ese el resultado esperado, o bien porque no se ha llegado al valor de parpadeo necesario, aparece un mensaje en "*ImagenValidación*" de "**NO VALIDADO**" (figura 9.33) y se siguen adquiriendo las señales hasta que el resultado sea el deseado.

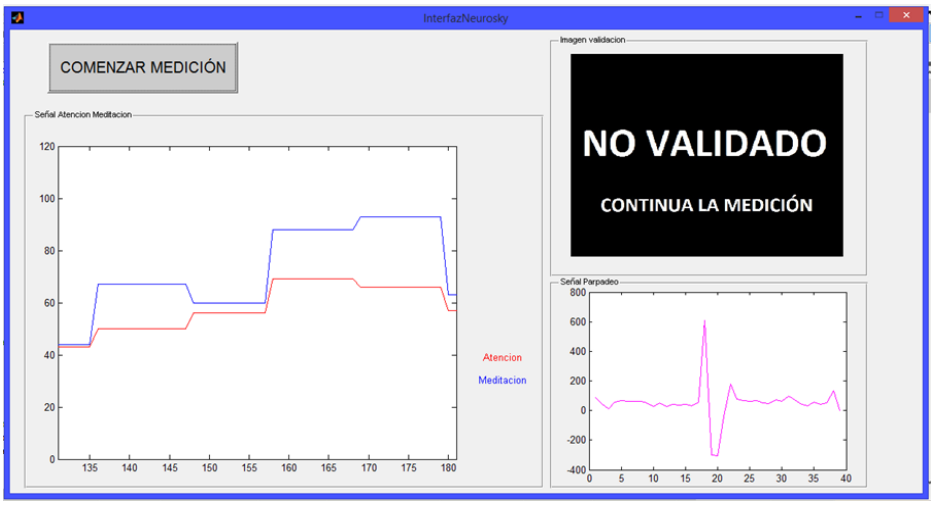


Figura 9.33 *Ejemplo de no validación*

ADVERTENCIA *Durante el tiempo de validación no se están recogiendo los datos de la atención y la meditación por eso cuando se vuelven a imprimir por pantalla pueden fluctuar mucho. Por otro lado, los últimos segundos almacenados se eliminan ya que el resultado en el siguiente segundo sería el mismo que no se ha querido validar.*

Los mensajes que se pueden cargar en el apartado "***ImagenValidación***" son los siguientes (figura 9.34):

Figura 9.34 *Mensajes posibles en el apartado "ImagenValidación"*

Una vez validado, el robot tanto en sistema real como en simulación comienza a moverse a la posición del color indicada.

VI Planos

En este capítulo se recoge todo el código de los archivos que se han realizado en todo el proyecto.

Conectar.m

```
%%CODIGO DE CONEXION CON BRAZO IRB120
global robot Simulado comenzar
%Obtenemos los datos sobre el modo de conexion
comenzar=0;
modo=get(handles.simulado,'Value');
if modo==1
   %Conectar con el simulador
   robot=irb120('127.0.0.1',1024);
else
   %Conectar con el robot real
   robot=irb120('172.29.28.185',1024);
end

%Conectar con el robot
robot.connect;
pause(2);
robot.TCPtool(1);   %Activar Ventosa
pause (2);
robot.TCPtool(0);   %Desactivar Ventosa
pause (2);
%Envio de coordenadas para mover el brazo a posiición intermedia
robot.TCProtandpos([180 0 180 300 0 300]);
comenzar=1;
```

Socket_robot_simulado.m

```
%%ROBOT SIMULADO
global modo color POS_X_OBJ POS_Y_OBJ X_FINAL Y_FINAL robot
if(color==1) %rojo
        X_FINAL=400;
        Y_FINAL=-300;
        POS_X_OBJ=400;
        POS_Y_OBJ=150;
elseif(color==3) %azul
        X_FINAL=200;
        Y_FINAL=-300;
        POS_X_OBJ=400;
        POS_Y_OBJ=0;
end

%%MOVIMIENTO ROBOT COGER PIEZAS
%Envio de coordenadas para mover el brazo a posición intermedia
rosbot.TCProtandpos([180 0 180 300 0 300]);
paue (1);
%Envio de coordenadas para mover el brazo a posición pieza con z=300
robot.TCProtandpos([180 0 180 POS_X_OBJ POS_Y_OBJ 300]);
pause (1);
%Envio de coordenadas para mover el brazo a posicion pieza z=180
robot.TCProtandpos([180 0 180 POS_X_OBJ POS_Y_OBJ 180]);
pause (1);
%Envio de coordenadas para mover el brazo a coger disco z=10
robot.TCProtandpos([180 0 180 POS_X_OBJ POS_Y_OBJ 10]);
pause (1);
robot.TCPtool(0);   %Activar Ventosa
%Envio de coordenadas para mover el brazo a posición intermedia
robot.TCProtandpos([180 0 180 300 0 300]);
pause (1);
%Envio de coordenadas para mover el brazo a posicion final con z=300
robot.TCProtandpos([180 0 180 X_FINAL Y_FINAL 300]);
pause (1);
%Envio de coordenadas para mover el brazo a posicion final con z=230
robot.TCProtandpos([180 0 180 X_FINAL Y_FINAL 230]);
pause(1);
robot.TCPtool(1);   %Desactivar Ventosa
pause(1);
%la posicion final la genera el jointmover de la estacion
```

Socket_robot_real.m

```
global color robot X_FINAL Y_FINAL Pos_x_objR Pos_y_objR Pos_x_objA Pos_y_objA
POS_X_OBJ POS_Y_OBJ LongitudX_real LongitudY_real LongitudX_foto LongitudY_foto Pos_x
Pos_y  real_x1 real_x2 real_y1 real_y2

if(color==1) %rojo
       Pos_x_obj=Pos_x_objR;
       Pos_y_obj=Pos_y_objR;
        X_FINAL=400;
        Y_FINAL=-300;

elseif (color==3) %azul
       Pos_x_obj=Pos_x_objA;
       Pos_y_obj=Pos_y_objA;
       X_FINAL=200;
       Y_FINAL=-300;

end

%diferencia longitud objeto centroide foto
DiferenciaX_foto=max(Pos_x(1),Pos_x_obj)-min(Pos_x(1),Pos_x_obj);
DiferenciaY_foto=max(Pos_y(1),Pos_y_obj)-min(Pos_y(1),Pos_y_obj);

%cálculo diferencia longitud objeto centroide real
DiferenciaX_real=((LongitudX_real*DiferenciaX_foto)/LongitudX_foto);
DiferenciaY_real=((LongitudY_real*DiferenciaY_foto)/LongitudY_foto);

%centroide en la posicion real para que lo coja el robot
POS_X_OBJ=real_x1+DiferenciaX_real;% el valor minimo siempre sera la pieza 1 de
referencia, que es la que marca el limite del tablero
POS_Y_OBJ=real_y1+DiferenciaY_real; % el valor minimo siempre sera la pieza 1 de
referencia, que es la que marca el limite del tablero

%%MOVIMIENTO ROBOT COGER PIEZAS
%Envio de coordenadas para mover el brazo a posición intermedia
robot.TCProtandpos([180 0 180 300 0 300]);
pause (1);
%Envio de coordenadas para mover el brazo a posición pieza con z=300
robot.TCProtandpos([180 0 180 POS_X_OBJ POS_Y_OBJ 300]);
pause(1);
%Envio de coordenadas para mover el brazo a posición pieza con z=180
robot.TCProtandpos([180 0 180 POS_X_OBJ POS_Y_OBJ 180]);
pause (1);
%Envio de coordenadas para mover el brazo a coger disco z=10
robot.TCProtandpos([180 0 180 POS_X_OBJ POS_Y_OBJ 10]);
pause(1);
robot.TCPtool(0);   %Activar ventosa
pause (1);
%Envio de coordenadas para mover el brazo a posición intermedia
robot.TCProtandpos([180 0 180 300 0 300]);
pause (1);
%Envio de coordenadas para mover el brazo a posición final con z=300
robot.TCProtandpos([180 0 180 X_FINAL Y_FINAL 300]);
pause (1);
%Envio de coordenadas para mover el brazo a posicion final con z=230
robot.TCProtandpos([180 0 180 X_FINAL Y_FINAL 230]);
pause(1);
robot.TCPtool(1);   %Desactivar ventosa
pause (1);
%Envio de coordenadas para mover el brazo a posición intermedia
robot.TCProtandpos([180 0 180 300 0 300]);
pause(1);
```

calibración_tablero.m

```
%Variables
global  Im_binV a b X1 X2 Y1 Y2 calibracion Pos_x Pos_y LongitudX_foto LongitudY_foto
LongitudX_real LongitudY_real real_x1 real_x2 real_y1 real_y2 umbralV
c=5;
real_x1=X1+c;
real_y1=Y1+c;
real_x2=X2+c;
real_y2=Y2+c;
a=1;

%Operadores morfológicos close y open
ele=strel('disk', 5);
Im_bin_V=imopen(Im_binV,ele);
ele2=strel('disk',9);
Im_bin_V=imclose(Im_bin_V,ele2);

%se limpian los bordes
Im_bin_V=imclearborder(Im_bin_V);

%relleno interior piezas
Im_binV=imfill(Im_bin_V,'holes');

%numero de objetos
Ilabel=bwlabel(Im_bin_V);
num_objV=max(max(Ilabel));

%centroide
g=regionprops(Ilabel,'centroid');
axes(handles.ImagenCalibracion);
imshow(calibracion);
hold on;

%%PINTAR CENTROIDES
if(num_objV==2) %numero de piezas verdes 2
   for x=1:length(g)
      plot(g(x).Centroid(1),g(x).Centroid(2),'r.');
      centro=g(x).Centroid;
      Pos_x(a)=centro(1);
      Pos_y(a)=centro(2);
      a=a+1;
   end
axis off;

%%CALCULO DE POSICION pixel-mm
   %LongituclosdX y LongitudY se calcula restando las coordenadas X de las
   %dos fichas
    LongitudX_foto=max(Pos_x(1),Pos_x(2))-min(Pos_x(1),Pos_x(2));
    LongitudY_foto=max(Pos_y(1),Pos_y(2))-min(Pos_y(1),Pos_y(2));
   %tamaño del tablero en la realidad
    LongitudX_real=max(real_x1,real_x2)-min(real_x1,real_x2);
    LongitudY_real=max(real_y1,real_y2)-min(real_y1,real_y2);
end
```

verificación.m

```
%Variables
global piezas Im_binA IbwA Im_binR IbwR  piezas num_objA num_objR  umbralR umbralA
Pos_x_objR Pos_y_objR Pos_x_objA Pos_y_objA

%%%%%ROJO%%%%
%Operadores morfológicos close y open
ele=strel('disk', 5);
Im_bin_R=imopen(Im_binR,ele);
ele2=strel('disk',9);
Im_bin_R=imclose(Im_bin_R,ele2);

%se limpian los bordes
Im_bin_R=imclearborder(Im_bin_R);
%relleno interior pieza
Im_bin_R=imfill(Im_bin_R,'holes');
%numero de objetos
IlabelR=bwlabel(Im_bin_R);
num_objR=max(max(IlabelR));
%Centroides y caja de pieza
gR=regionprops(IlabelR,'centroid','BoundingBox');

%%%%%AZUL%%%%
%Operadores morfológicos close y open
ele=strel('disk', 5);
Im_bin_A=imopen(Im_binA,ele);
ele2=strel('disk',9);
Im_bin_A=imclose(Im_bin_A,ele2);

%se limpian los bordes
Im_bin_A=imclearborder(Im_bin_A);
%relleno interior piezas
Im_bin_A=imfill(Im_bin_A,'holes');
%numero de objetos
IlabelA=bwlabel(Im_bin_A);
num_objA=max(max(IlabelA));
%Centroide y caja de pieza
gA=regionprops(IlabelA,'centroid','BoundingBox');
axes(handles.imagenPosicionPiezas);
imshow(piezas);
hold on

%pintar centroides pieza roja y azul
if(num_objR==1) || (num_objA==1)%numero maximo rojo=1 y azul=1
 for x=1:length(gR)
    plot(gR(x).Centroid(1),gR(x).Centroid(2),'g.');
    rectangle('position',gR(x).BoundingBox,'edgecolor','g','linewidth',3)
    centro=gR(x).Centroid;
    Pos_x_objR=centro(1);
    Pos_y_objR=centro(2);
 end
 for c=1:length(gA)
    plot(gA(c).Centroid(1),gA(c).Centroid(2),'g.');
rectangle('position',gA(c).BoundingBox,'edgecolor','g','linewidth',3)
    centro=gA(c).Centroid;
    Pos_x_objA=centro(1);
    Pos_y_objA=centro(2);
 end
end
```

InterfazMatlab.m

```
% --- Executes just before InterfazMatlab is made visible.
function InterfazMatlab_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for InterfazMatlab
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
%%%Botones de interfaz desactivados al iniciar el programa
set(handles.continuar,'Enable','on');
set(handles.continuar,'Enable','off');
set(handles.verificar,'Enable','off');
set(handles.calibrar,'Enable','off');
set(handles.capturacalibracion,'Enable','off');
set(handles.capturaRojoAzul,'Enable','off');
%%%Imagenes de logos de instituciones academicas
uah=imread('LogoUAH.jpg');
axes(handles.uah);
imshow(uah);
axis off;
% --- Executes on button press in conectar.
function conectar_Callback(hObject, eventdata, handles)
global modo robot comenzar
%%%Control de apariencia de botones en interfaz
set(handles.conectar,'Enable','off');
set(handles.real,'Enable','off');
set(handles.simulado,'Enable','off');
set(handles.continuar,'Enable','off');
Conectar; %Conectar con robot
%%Depndiendo si es REAL o SIMULADO
if modo==1 && comenzar==1
  set(handles.continuar,'Enable','on');
else
 set(handles.capturacalibracion,'Enable','on');
end
% --- Executes on slider movement.
function sliderRojo_Callback(hObject, eventdata, handles)
global umbralR Im_binR IbwR
umbralR=get(hObject,'Value');
set(handles.colorrojo,'String',umbralR); %Se asigna el valor del Slider al StaticText
Im_binR=im2bw(IbwR,umbralR);
axes(handles.imagencalibrarrojo);
imshow(Im_binR);
axis off
guidata(hObject,handles)
% --- Executes on slider movement.
function sliderAzul_Callback(hObject, eventdata, handles)
global umbralA Im_binA IbwA
umbralA=get(hObject,'Value');
set(handles.colorazul,'String',umbralA); %Se asigna el valor del Slider al StaticText
Im_binA=im2bw(IbwA,umbralA);
axes(handles.imagencalibrarazul);
imshow(Im_binA);
axis off
guidata(hObject,handles)
% --- Executes on slider movement.
function sliderVerde_Callback(hObject, eventdata, handles)
global umbralV Im_binV IbwV
umbralV=get(hObject,'Value');
set(handles.colorverde,'String',umbralV); %Se asigna el valor del Slider al StaticText
Im_binV=im2bw(IbwV,umbralV);
axes(handles.imagenparacalibrar);
imshow(Im_binV);
axis off
set(handles.calibrar,'Enable','on');
guidata(hObject,handles)
```

```
% --- Executes on button press in verificar.
function verificar_Callback(hObject, eventdata, handles)
)
global piezas color x POSX_OBJ_REAL POSY_OBJ_REAL num_objR num_objA Pos_x Pos_y
LongitudX_foto LongitudY_foto LongitudX_real LongitudY_real umbralR umbralA
verificacion;
if ((num_objR==1) && (num_objA==1))
  msgbox('Verificacion de los dos objetos completada','Verificacion'); %mensaje
  %%%Control de apariencia de botones en interfaz
  set(handles.continuar,'Enable','on');
  axis off;
elseif ((num_objR~=1) && (num_objA==1))
    errordlg('No ha sido posible determinar la posicion del objeto ROJO','Verificacion
ROJO'); %error
    axis off;
elseif ((num_objR==1) && (num_objA~=1))
    errordlg('No ha sido posible determinar la posicion del objeto AZUL',' Verificacion
AZUL'); %error
    axis off;
else
    errordlg('No ha sido posible la verificacion','verificacion'); %error
    axis off;
end
function posx1_Callback(hObject, eventdata, handles)
global err_X1
X1=str2double(get(hObject,'String')); %Almacenar valor ingresado y transformar a tipo
double
handles.posx1=X1; %Almacenar en puntero
if isnan(X1)
    errordlg('El valor debe ser numérico','ERROR')
    err_X1=1;
else
    err_X1=0;
end
guidata(hObject,handles); %Salvar datos de la aplicación
function posy1_Callback(hObject, eventdata, handles)
global err_Y1
Y1=str2double(get(hObject,'String')) %Almacenar valor ingresado y transformar a tipo
double
handles.posy1=Y1; %Almacenar en puntero
if isnan(Y1)
    errordlg('El valor debe ser numérico','ERROR')
    err_Y1=1;
else
    err_Y1=0;
end
function posx2_Callback(hObject, eventdata, handles)
global err_X2
X2=str2double(get(hObject,'String')); %Almacenar valor ingresado y transformar a tipo
double
handles.posx2=X2; %Almacenar en
puntero
if isnan(X2)
    errordlg('El valor debe ser numérico','ERROR')
    err_X2=1;
else
    err_X2=0;
end
guidata(hObject,handles); %Salvar datos de la aplicación

function posy2_Callback(hObject, eventdata, handles)
global err_Y2;
Y2=str2double(get(hObject,'String')); %Almacenar valor ingresado y transformar a tipo
double
handles.posy2=Y2; %Almacenar en puntero
if isnan(Y2)
    errordlg('El valor debe ser numérico','ERROR')
   err_Y2=1;
else
    err_Y2=0;
end
guidata(hObject,handles); %Salvar datos de la aplicación
```

```
% --- Executes on button press in capturacalibracion.
function capturacalibracion_Callback(hObject, eventdata, handles)
global umbralV Im_binV IbwV calibracion
%%Inicializar video
video = videoinput('winvideo', 2,'RGB24_640x480');
start(video);
calibracion= getsnapshot(video); %Captura de imagen calibracion
%%procesamiento de imágenes
umbralV=get(handles.sliderVerde,'Value');
gris=rgb2gray(calibracion);
img_verde=calibracion(:,:,2);
IbwV=img_verde-gris;
Im_binV=im2bw(IbwV,umbralV);
axes(handles.imagenparacalibrar);
imshow(Im_binV);
axis off
guidata(hObject,handles)

%%%Control de apariencia de botones en interfaz
set(handles.calibrar,'Enable','on')
set(handles.capturacalibracion,'Enable','off');
% --- Executes on button press in calibrar.
function calibrar_Callback(hObject, eventdata, handles)
global err_X1 err_X2 err_Y1 err_Y2 calibracion f Im_binV a b X1 X2 Y2 Y1 calibracion
real_x1 real_x2 real_y1 real_y2 Pos_x Pos_y LongitudX_foto LongitudY_foto
LongitudX_real LongitudY_real real_x1 real_x2 real_y1 real_y2

X1=handles.posx1;
Y1=handles.posy1;
X2=handles.posx2;
Y2=handles.posy2;
opc=questdlg ('¿Está seguro de Calibrar?', 'CALIBRACION','Si', 'No', 'Si'); %mensaje de
confirmacion final
axis off
if strcmp (opc, 'No')
    return;
else
  if  (((err_X1==1) || (err_X2==1)) || ((err_Y1==1) || (err_Y2==1))) %error de datos
    errordlg('Calibracion incompleta, datos introducidos ERRONEOS','ERROR');
    axis off;
  else
    calibracion_tablero;
    b=a;
    if ((b==3) && (LongitudX_real>=0) && (LongitudY_real>=0)) %mensaje de calibracion
completada
       msgbox('Calibracion completada satisfactoriamente','Calibracion');
       axis off;
       %%Control de apariencia de botones en interfaz
       set(handles.capturacalibracion,'Enable','off');
       set(handles.capturaRojoAzul,'Enable','on');
    else
       errordlg('No ha sido posible la calibracion','Calibracion'); %error de
calibracion
       axis off;
    end
  end
end

% --- Executes on button press in capturaRojoAzul.
function capturaRojoAzul_Callback(hObject, eventdata, handles)
global umbralR Im_binR IbwR umbralA Im_binA IbwA piezas
%%Inicializar video
video = videoinput('winvideo', 2,'RGB24_640x480');
start(video);
piezas = getsnapshot(video); %Captura de imagen para los colores
%%%Control de apariencia de botones en interfaz
set(handles.calibrar,'Enable','off'); %una vez calibrado ya no se vuelve a calibrar
set(handles.verificar,'Enable','on');
```

```
%%procesamiento de imagenes
umbralR=get(handles.sliderRojo,'Value');
umbralA=get(handles.sliderAzul,'Value');
gris=rgb2gray(piezas);
img_colorR=piezas(:,:,1);
img_colorA=piezas(:,:,3);
IbwR=img_colorR-gris;
IbwA=img_colorA-gris;
Im_binR=im2bw(IbwR,umbralR);
Im_binA=im2bw(IbwA,umbralA);
axes(handles.imagencalibrarrojo);
imshow(Im_binR);
axes(handles.imagencalibrarazul);
imshow(Im_binA);
set(handles.piezaroja
,'String','Objeto Rojo');
set(handles.piezaazul,'String','Objeto Azul');
axis off
%%%Control de apariencia de botones en interfaz
set(handles.verificar,'Enable','on'); % verificar tantas veces como queramos

% --- Executes on button press in continuar.
function continuar_Callback(hObject, eventdata, handles)
global robot modo color Pos_x_objR Pos_y_objR Pos_x_objA Pos_y_objA POS_X_OBJ POS_Y_OBJ
LongitudX_real LongitudY_real LongitudX_foto LongitudY_foto Pos_x Pos_y  real_x1
real_x2 real_y1 real_y2 robot

opc=questdlg ('¿Está seguro de que quiere Continuar?', 'CONTINUAR','Si', 'No', 'Si');
%mensaje de confirmacion
axis off
if strcmp (opc, 'No')
    return;
elseif strcmp (opc, 'Si')
    InterfazNeurosky;

end
```

InterfazNeurosky.m

```
function varargout = InterfazNeurosky(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @InterfazNeurosky_OpeningFcn, ...
                   'gui_OutputFcn',  @InterfazNeurosky_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDI
% --- Executes on button press in comenzar.
function comenzar_Callback(hObject, eventdata, handles)

%VARIABLES GLOBALES
global modo robot color Pos_x_objR Pos_y_objR Pos_x_objA Pos_y_objA POSX_OBJ_REAL
POSY_OBJ_REAL LongitudX_real LongitudY_real LongitudX_foto LongitudY_foto Pos_x Pos_y
real_x1 real_x2 real_y1 real_y2

%%Imagenes para la validacion
validar_rojo=imread('C:\Users\laura\Documents\MATLAB\TFG_INTERFAZ_VENTOSA\validar_rojo.
jpg','jpg');
validar_azul=imread('C:\Users\laura\Documents\MATLAB\TFG_INTERFAZ_VENTOSA\validar_azul.
jpg','jpg');
validado_rojo=imread('C:\Users\laura\Documents\MATLAB\TFG_INTERFAZ_VENTOSA\validado_roj
o.jpg','jpg');
validado_azul=imread('C:\Users\laura\Documents\MATLAB\TFG_INTERFAZ_VENTOSA\validado_azu
l.jpg','jpg');
novalidado=imread('C:\Users\laura\Documents\MATLAB\TFG_INTERFAZ_VENTOSA\novalidado.jpg'
,'jpg');
%%INICIALIZACION DEL CASCO MINDWAVE
%Buffer de datos
data_att = zeros(40,3);
data_blink = zeros(30,2);
portnum1 = 4;  %puerto
comPortName1 = sprintf('\\\\.\\COM%d', portnum1);
TG_BAUD_57600  = 57600; % velocidad
TG_STREAM_PACKETS =     0; % datos
% Datos TG_GetValue().
TG_DATA_POOR_SIGNAL =     1;
TG_DATA_ATTENTION =       2;
TG_DATA_MEDITATION =      3;
TG_DATA_RAW =             4;
TG_DATA_BLINK_STRENGTH = 37;
loadlibrary('ThinkGear.dll');
fprintf('ThinkGear.dll loaded\n');
dllVersion = calllib('ThinkGear', 'TG_GetDriverVersion');
fprintf('ThinkGear DLL version: %d\n', dllVersion );
%errores de conexión
% Obteniendo un identificador de ID de conexión a ThinkGear
connectionId1 = calllib('ThinkGear', 'TG_GetNewConnectionId');
if ( connectionId1 < 0 )
        error( sprintf( 'ERROR: TG_GetNewConnectionId() returned %d.\n', connectionId1
) );
end;
% Intentando conectar el identificador de la conexión al puerto serie "COM3"
errCode = calllib('ThinkGear', 'TG_Connect',
connectionId1,comPortName1,TG_BAUD_57600,TG_STREAM_PACKETS );
if ( errCode < 0 )
        error( sprintf( 'ERROR: TG_Connect() returned %d.\n', errCode ) );
end
```

```
fprintf( 'Connected.  Reading Packets...\n' );
if (calllib('ThinkGear','TG_EnableBlinkDetection', connectionId1,1)==0)
      disp('Blinkdetection');
end
disp('Reading Brainwaves');

%%declaramos variables
color=0; % rojo-> color=1 azul-> color=3
atencion=0;
meditacion=0;
blink=0;
aten=zeros(30,1); %hemos recogido 40 datos
medi=zeros(30,1);
blink_senal=zeros(20,1);
i=0;
j=0;
b=0;
c=1; %infinito el while
k=1;
a=1;
d=1;
n=0;
p=0;

%%leyenda de la señal atencion/meditacion
set(handles.color_rojo);
set(handles.rojo,'String','Atencion');
set(handles.color_azul);
set(handles.azul,'String','Meditacion');

errCode = calllib('ThinkGear', 'TG_EnableAutoRead',  connectionId1,-1 );
while (c==1)
  if (calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0) %leer
las señales
    j = j + 1;
    d=d+1;
    %leer atencion/meditacion.
    data_att(j,1) = calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_ATTENTION);
    data_att(j,2) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_MEDITATION);
    % Los ejes del PLOT
    if(j<70)
       xmin=0;
       xmax=69;
    else
       xmin=j-70;
       xmax=j;
    end
    plot(handles.atencionmeditacion,data_att(:,1),'-r','Linewidth',1);
    hold (handles.atencionmeditacion, 'on')
    plot(handles.atencionmeditacion,data_att(:,2),'-b','Linewidth',1);
    drawnow;
    %%Rango de X
    xlim(handles.atencionmeditacion,[xmin xmax]);
    ylim(handles.atencionmeditacion,[0  120]);
    i=i+1;
    a=a+1; %DUDA
  end

  if((i>180)&&(a>12)) %los primeros 100 datos de calibracion
     a=0;
     aten=data_att((110+b:1:j),1);
     medi=data_att((110+b:1:j),2);
     atencion=length(aten(aten>=80));
     meditacion=length(medi(medi>=80));
     b=b+10;
```

```
     if(atencion>(meditacion+40))
        axes(handles.imagenvalidacion);
        imshow(validar_rojo);
        m=1;
        k=0;
        while(m<40) %validacion parpadeo
           if (calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%leer las señales
              k = k + 1;
              data_blink(k,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_RAW);
              m=m+1;
              plot(handles.imagenparpadeo,data_blink(:,1),'-k','Linewidth',1);
              drawnow;
           end
         end
         blink_senal=abs(data_blink(:,1)); %valor absoluto
         blink=length(blink_senal(blink_senal>600)); %valor de parpadeo mayor que 600
         if(blink>7) %validado atencion rojo
            cla(handles.imagenvalidacion, 'reset');
            axes(handles.imagenvalidacion);
            imshow(validado_rojo);
            color=1;
            c=2;
            if (modo==1) %Robot conexion real simulacion
               Socket_robot_simulado;
            else
               Socket_robot_real;
            end
         else %no validado
            axes(handles.imagenvalidacion);
            imshow(novalidado);
            pause(2)
            i=120; %empezar a coger datos desde el 120 o el 110??
            b=60;
            cla(handles.imagenparpadeo, 'reset');
            data_blink=zeros(40,2);
            cla(handles.imagenvalidacion, 'reset');
         end
     elseif(meditacion>(atencion+40))
        axes(handles.imagenvalidacion);
        imshow(validar_azul);
        m=1;
        k=0;
        while(m<40) %validacion parpadeo
           if (calllib('ThinkGear','TG_GetValueStatus',connectionId1,TG_DATA_RAW) ~= 0)
%leer las señales
              k = k + 1;
              data_blink(k,1) =
calllib('ThinkGear','TG_GetValue',connectionId1,TG_DATA_RAW);
              m=m+1;
              plot(handles.imagenparpadeo,data_blink(:,1),'-k','Linewidth',1);
              drawnow;
           end
        end
        blink_senal=abs(data_blink(:,1));
        blink=length(blink_senal(blink_senal>600));
        if(blink>7) %validacion meditacion azul
            cla(handles.imagenvalidacion, 'reset');
            axes(handles.imagenvalidacion);
            imshow(validado_azul);
            color=3;
            c=2;
             if (modo==1) %Robot conexion real o simulacion
                Socket_robot_simulado;
            else
               Socket_robot_real;
             end
```

```
  else
            cla(handles.imagenvalidacion, 'reset');
            axes(handles.imagenvalidacion);
            imshow(novalidado);
            pause(2)
            cla(handles.imagenparpadeo, 'reset');
            data_blink=zeros(40,2);
            i=120;
            b=60;
            cla(handles.imagenvalidacion, 'reset');
        end
     else
     end
   else
   end
end
%%Desconectar el casco
calllib('ThinkGear', 'TG_FreeConnection', connectionId1 );
```

VI Pliego de condiciones

10. Pliego de condiciones

En este capítulo se detallan las herramientas utilizadas en el proyecto, con la finalidad de poder elaborar posteriormente el presupuesto final del proyecto.

10.1. Hardware

1. Portátil Lenovo U710
    - Procesador: Intel® Core™ i5.
    - RAM: 8 GB.
    - Tarjeta gráfica: NVIDIA GeForce 720M.

2. Robot industrial ABB-IRB120
    - Peso: 25 Kg.
    - Altura: 580 mm.
    - Carga soportada: 3 Kg (hasta 4kg con muñeca vertical).
    - Controlador IRC5 Compact. Tarjeta DeviceNet (Lean), con 16 entradas y 16 salidas digitales (0-1) de 24V.

3. Casco Mindwave de Neurosky
    - Peso: 90 gramos
    - Medidas: alto 225mm x ancho 155mm x profundidad 165 mm
    - Consumo de potencia: 80 mA (conectado y transmitiendo)
    - UART (serie): VCC,GND,TX,RX
    - Tasa UART: 57600 baudios

4. Periféricos
    - Ratón HP.
    - Pen Drive 16 GB TOSHIBA.

10.2. Software

1. Sistema operativo
    - Windows 8 © Microsoft Corporation.
2. Software de desarrollo
    - MATLAB R2010a
    - ABB RobotStudio versión 5.61
3. Procesador de textos y edición de imágenes
    - Microsoft Office Home and Student 2010
4. Software de edición de imágenes
    - Microsoft Paint 2010

VII Presupuesto

11. Presupuesto

Este capítulo proporciona la información detallada de los costes teóricos del desarrollo del proyecto, incluyendo los **costes materiales** y las **tasas profesionales**.

11.1. Costes materiales

En la tabla 11.1, se han desglosado de forma general los costes materiales de este proyecto.

Concepto		Cantidad	Precio Unidad (€)	Precio Total (€)
Hardware	***Portátil Lenovo***	1	600 €	600 €
	ABB-IRB120	1	10900 €	10900 €
	Casco Mindwave	1	120 €	120 €
	Periféricos	1	50 €	50 €
Software	***Microsoft Office 2010***	1	100 €	100 €
	Licencia RobotStudio 5.61	1	1100 €	1100 €
	Licencia Matlab r2010a	1	500 €	500 €
TOTAL				**13370 €**

Tabla 11.1 *Costes materiales (Hardware y Software) sin IVA*

11.2. Costes profesionales

El coste de profesional es el desglosado en la tabla 12.2, se ha hecho una estimación general de horas.

Concepto	Horas	Precio hora (€)	Precio Total (€)
Ingeniero Industrial	600	40 €	24000 €
TOTAL			**24000 €**

Tabla 11.2 *Costes profesionales sin IVA*

11.3. Costes totales

Los costes totales del Proyecto han sido obtenidos sumando los costes materiales y profesionales aplicando el IVA, como se recoge en la tabla 11.3. Además hay que tener en cuenta los costes de impresión y encuadernación.

Concepto	Precio Total (€)
Coste material	13370 €
Coste profesional	24000 €
SUBTOTAL	37450 €
IVA (21 %)	7864 €
TOTAL	**45314 €**

Tabla 11.3 *Costes totales con IVA*

VIII Bibliografía

Bibliografía

[1] http://www.teinteresa.es/ciencia/derecha-cerebro-creativa-izquierda__logica_0_866315262. Última fecha de consulta 27/12/2016

[2] Rafael Barea Navarro "Electroencefalografía" http://www.bioingenieria.edu.ar/academica/catedras/bioingenieria2/archivos/apuntes/tema%205%20-%20electroencefalografia.pdf

[3] https://plus.google.com/108366589298536076762/posts/ht6qaFUM2rE. Última fecha de consulta 31/12/2016

[4] http://antroporama.net/que-significan-las-ondas-gamma-cerebrales/. Última fecha de consulta 03/01/2017

[5] Teresa Talamillo García, "Manual Básico para enfermeros en electroencefalografía" http://www.juntadeandalucia.es/servicioandaluzdesalud/huvvsites/default/files/revistas/ED-094-07., pdf2011.

[7] https://lacofa.fundaciontelefonica.com/2008/12/15/introduccion-a-los-sistemas-brain-computer-interface/. Última fecha de consulta 15/01/2017

[8]Alberto Prieto, "Brain Computer Interfaces (BCI)" http://atc.ugr.es/pages/docencia/no_reglada/tendencias_ic/media/_doc/bci_aprieto_julio2013_v1/!.pdf, 2013

[9] Neurosky store, http://store.neurosky.com/pages/mindwave. Última fecha de consulta 30/01/2017.

[10] http://www7.uc.cl/sw_educ/neurociencias/html/004.html. Última fecha de consulta 03/02/2017

[11] Neurosky, "Mindset Communications Protocol", http://wearcam.org/ece516/mindset_communications_protocol.pdf ,2010.

[12] http://www.fgcsic.es/lychnos/es_es/articulos/Brain-Computer-Interface-aplicado-al-entrenamiento-cognitivo. Última fecha de consulta 21/01/17.

[13]http://ieeexplore.ieee.org/document/1605260/?reload=true&arnumber=1605260&_ga=1.30724281.761165472.148380547 Última fecha de consulta 21/01/17

[14] http://cnbi.epfl.ch/page-34069-en.html . Última fecha de consulta 21/01/17

[15]http://ieeexplore.ieee.org/document/4360109/?arnumber=4360109&_ga=1.197295273.761165472.1483805477. Última fecha de consulta 21/01/17

[16] http://new.abb.com/es/abb-in-spain/quienes-somos . Última fecha de consulta 21/01/17

[17] Hinterberger, T., Schmidt, S., Neumann, N., Mellinger, J.,Blankertz, B., Curio, G., and Birbaumer, N. “Brain-computercommunication and slow cortical potentials. Biomedical Engineering”,2004.

[18] http://new.abb.com/products/robotics/es/robots-industriales/irb-120. Última fecha de consulta 27/03/2017

[19] http://www.monografias.com/trabajos102/introduccion-utilizacion-robotstudio-abb/ introduccion -utilizacion-robotstudio-abb.shtml#ixzz4WbVSEbtV. Última fecha de consulta

[20] Dr. Alaa Khamis, “Lenguaje RAPID” http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/home/facilities/rapid.pdf , pdf, 2006

[21] http://facildeaprenderconnosotros.blogspot.com.es/p/blog-page.html . Última fecha de consulta 15/02/2017

[22] http://www.anatolandia.com/2013/10/caracteristicas-partes-funciones-encefalo.html. Última fecha de consulta 15/02/2017

[23] https://www.ecured.cu/Dienc%C3%A9falo. Última fecha de consulta 15/02/2017

[24] http://firedarkunamrock.blogspot.com.es/2012/09/funcion-de-los-lobulos-cerebrales.html. Última fecha de consulta 15/02/2017

[25] http://docentes.educacion.navarra.es/metayosa/1bach/rela3.html. Última fecha de consulta 15/02/2017

[26] http://www.datuopinion.com/potencial-de-accion. Última fecha de consulta 15/02/2017

[27] http://cuadrocomparativo.org/cuadros-sinopticos-sobre-las-neuronas/. Última fecha de consulta 15/02/2017

[28] https://percibir.wikispaces.com/Comunicaci%C3%B3n+neuronal. Última fecha de consulta 15/02/2017

[29] http://byg1b.blogspot.com.es/2011/04/3-transmision-nerviosa-impulso-y.html. Última fecha de consulta 17/02/2017

[30] http://www.ecodecrypt.com/brain-waves.html. Última fecha de consulta 20/02/2017

[31] http://es.slideshare.net/kattyo/electroencefalografia. Última fecha de consulta 25/02/2017

[32] http://es.slideshare.net/kattyo/electroencefalografia. Última fecha de consulta 25/02/2017

[33] http://www.revolu-tion.com.ar/foro/content/283-crean-una-app-para-navegar-gracias-la-actividad-cerebral.html. Última fecha de consulta 25/02/2017

[34] http://sapiensmedicus.org/eeg-interpretacion-para-mortales/. Última fecha de consulta 10/03/2017

[35] http://es.slideshare.net/RUDEROCKER/instrumentacion-medica-ii. Última fecha de consulta 10/03/2017

[36] http://www.eldiario.es/hojaderouter/tecnologia/sindrome_del_cautiverio-la_escafandra y_ la_mariposa-bci_0_335966571.html. Última fecha de consulta 18/03/2017

[37]http://www.monografias.com/trabajos99/estado-del-arte-interfaces-cerebro-computadora /estado-del-arte-interfaces-cerebro-computadora.shtml. Última fecha de consulta 18/03/2017

[38] Schalk, G. Leuthardt, E. "Brain-Computer Interfaces Using Electrocorticographic Signals. Clinical Aplication Review". IEE Reviews in Biomedical Engineering. Vol.4, 2011

[39] https://www02.abb.com/global/CNABB/CNABB050.NSF!OpenDatabase&db=/global/cnabb/ cnabb054.nsf&v=BF6&e=us&url=/global/seitp/seitp202.nsf/0/F528FEA6D4D93DB5482579130028 C06B!OpenDocument. Última fecha de consulta 23/03/2017

[40] http://www.infoplc.net/noticias/item/1375-abb-lanza-la-version-rapida-del-robot-irb-120. Última fecha de consulta 24/03/207

[41] ABB, "Especificaciones del producto Controller IRC5 with FlexPendant" https://library.e.abb.com/public/2b5b950d68a0503cc1257c0c003cb703/3HAC041344-es.pdf

[42] http://new.abb.com/products/3HAC020536-014/irc5-controller. Última fecha de consulta 27/03/2017

[43] Neurosky, "Mindwave User Guide" pdf, 2011. Última fecha de consulta 14/05/2017.

[44] Azahara Gutiérrez Corbacho, Trabajo Fin de Grado "Desarrollo de una interfaz para el control del robot IRB120 desde MATLAB",2014. Última fecha de consulta 18/05/2017

[45] Marek Jerzy Frydrysiak, "Socket based communication in RobotStudio for controlling ABB-IRB120 robot. Design and development of a palletizing station". fecha de consulta 18/05/2017

[46] http://www.comp.leeds.ac.uk/vision/vision_matlab_tutorial/section2/imgRep.html. fecha de consulta 18/05/2017

IX Apéndices

Apéndice A

TFG: Trabajo Fin de Grado

EEG: electroencefalografía

BCI: Brain Computer Interface (interfaz cerebro-ordenador?)

Hz: hercios

mV: milivoltios

µV: microvoltios

Fp: punto frontal polar

O: punto Occipital

Cz: punto central o vértex

Pz: punto parietal

Fz: punto frontal

ADC: Convertidor Analógico Digital

SCP: Slow Cortical Potencials (Potenciales corticales lentos)

VEP: Potenciales evocados visuales

AEP: Potenciales evocados auditivos

Kg: kilogramo

Mm: milímetros

mA: miliamperios

ABB: Asea Brown Boveri (empresa)

RAPID: Robotics Application Programming Interactive Dialogue

RGB: escala de color (RED, GREEN, BLUE)

GB: GigaByte

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá