UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Utilización del Servicio EMOTIV Cortex API para Controlar un Brazo Robótico Mediante Señales, Obtenidas del Casco Insight Neuro-Headset

Artículo Académico.

Iván Nicolás Zamora Avilés

Ingeniería Electrónica

Trabajo de titulación presentado como requisito
para la obtención del título de
Ingeniero Electrónico

Quito, 25 de febrero de 2019

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERÍAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

Utilización del Servicio EMOTIV Cortex API para Controlar un Brazo Robótico Mediante Señales, Obtenidas del Casco Insight Neuro-Headset

Iván Nicolás Zamora Avilés

Calificación:

Nombre del profesor, Título académico Diego Benítez, Ph.D.

Firma del profesor _____

Quito, 25 de febrero de 2019

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Iván Nicolás Zamora Avilés

Código: 00118675

Cédula de Identidad: 0604029546

Lugar y fecha: Quito, 25 de febrero de 2019

RESUMEN

En este trabajo se muestra la manera en la que se utilizó el servicio CORTEX de Emotiv para construir un software de adquisición de datos en el lenguaje de programación Python para el control de un brazo robótico utilizando el headset Emotiv Insight. Se presentan rutinas de programación para la adquisición de datos que se reciben a través de streaming mediante un Websocket, los cuales son luego procesados y analizados para la detección de eventos. Posteriormente los eventos disparan señales de control hacia un brazo robótico el cual es controlado por un Raspberry Pi 3 mediante el protocolo de Internet de las cosas MQTT.

Palabras clave: MQTT, Emotiv Insight, Python, Websocket, API, Suscripción, Publicación

ABSTRACT

This paper presents the way that Emotiv Cortex Service was used for building a data acquisition software based on Python in the controlling of a robotic arm using Emotiv Insight headset. It shows programming routines for data acquisition that are collected through streaming in a Websocket. Then the data is processed for event detection. The event detection algortihm triggers control signals to a robotic arm that is controlled by a Raspberry Pi 3 through IoT MQTT protocol.

Key words: MQTT, Emotiv Insight, Python, Websocket, API, Subscription, Publication.

TABLA DE CONTENIDO

Utilización del Servicio EMOTIV Cortex API para Controlar un Brazo Robótico Mediante Señales Obtenidas del Casco Insight Neuro-Headset

Ivan Zamora, Diego S. Benítez

Universidad San Francisco de Quito USFQ
Colegio de Ciencias e Ingenierías "El Politécnico"
Campus Cumbayá, Casilla Postal 17-1200-841, Quito, Ecuador
email: ivan.zamora@estud.usfq.edu.ec, dbenitez@usfq.edu.ec

***Abstract*—En este trabajo se muestra la manera en la que se utilizó el servicio CORTEX de Emotiv para construir un software de adquisición de datos en el lenguaje de programación Python para el control de un brazo robótico utilizando el headset Emotiv Insight. Se presentan rutinas de programación para la adquisición de datos que se reciben a través de streaming mediante un websocket, los cuales son luego procesados y analizados para la detección de eventos. Posteriormente los eventos disparan señales de control hacia un brazo robótico el cual es controlado por un Raspberry Pi 3 mediante el protocolo de Internet de las cosas MQTT.**

I. Introducción

En los últimos años, ha proliferado el desarrollo de herramientas que estén al alcance de personas con discapacidades motrices, por el ejemplo se han desarrollado nuevas técnicas de control que estén a disposición de una mayor parte de la población, basadas en el uso de señales provenientes de sensores de Electroencefalograma (EEG) y Electrogmiograma(EMG) [6].

En este sentido, el trabajo presenta una prueba de concepto para el control de un brazo robótico enfocado a personas con discapacidades, mediante el uso de señales de control obtenidas de un casco EEG de bajo costo disponible en el mercado (Emotiv Insight). Cabe mencionar, que el sistema de desarrollo de software (SDK) proporcionado por la compañía Emotiv para sus cascos con sensores electroencefalográficos (EEG) (Emotiv SDK) fue recientemente dado de baja; por lo que, Emotiv saco al mercado un nuevo sistema de desarrollo basado en API (Application Programming Interface) llamado Cortex, razón por la cual, las aplicaciones utilizando el casco ECG Emotiv EPOC y otros productos de Emotiv previamente desarrollados en el SDK se deben trasladar a Cortex.

La API de Cortex esta basada en JSON (Java Script Object Notation) y Websockets, de esta manera se puede acceder al servicio usando múltiples lenguajes y/o plataformas. Una conexión Websocket establece una conexión full duplex, de comunicación con canales bidireccionales que operan sobre un único socket en la Web [4], de esta manera el usuario debe ahora contratar el servicio para obtener información del casco directamente con la compañía.

En este trabajo se ha optado por desarrollar un programa capaz de aprovechar las bondades del servicio proporcionado, tales cómo: adquisición de señales sin procesar en tiempo real, uso de las señales de los sensores de movimiento, etc. Para esto, se eligió el lenguaje de programación Python, el cual es un lenguaje amigable con los usuarios y de múltiple compatibilidad con otros entornos y servicios. Python es de libre acceso y cuenta con una gran base de librerías para el tratamiento de señales y datos.

El API Cortex responde a un servicio de peticiones que son enviadas en mensajes de tipo JSON. Mediante los cuales se accede al servicio de streaming, esto se da solamente, luego de un exitoso inicio de sesión y autenticación del registro en la base de datos de Emotiv. La transmisión de datos una vez abierto el websocket, es de 128 datos por segundo, por cada stream que se haya solicitado.

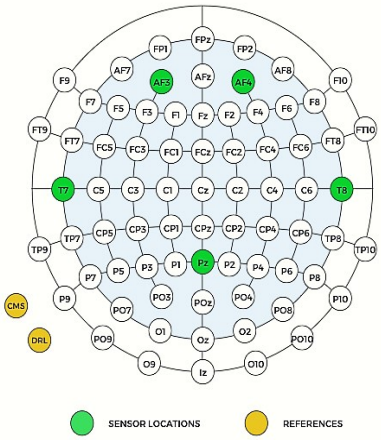

Fig. 1: Emotiv Insight Sensor Location

El headset utilizado en este proyecto es el Emotiv Insight, que cuenta con cinco sensores de EEG estos se ubican en las

posiciones: AF3, T7, PZ, T8 y AF4, de acuerdo al sistema internacional 10-20, tal como se muestra en la Fig. 1. La resolución del casco es de 14 bits con 1 LSB = 0.51μV [1]. Además de esto, el casco cuenta con sensores de movimiento tales como: giroscopio, magnetómetro y un acelerómetro. Una característica importante del casco es su poco peso, sus señales robustas y su forma amigable con el usuario, tal como se muestra en la Fig. 2.

Fig. 2: Headset Emotiv Insight

Mediante el uso de este casco se busca acceder a las señales crudas, es decir sin procesar, tanto de los de sensores de movimiento como de los canales de EEG de los que dipsone el casco. Estas señales, una vez procesadas servirán como señales de control para el movimiento del brazo robótico. El medio de comunicación entre el brazo robótico y las señales de control se hará a través del protocolo de comunicación MQTT (Message Queue Telemetry Transport), un protocolo de mensajería ligero para pequeños sensores y dispositivos móviles, a través del internet optimizado para redes de alta latencia o poco confiables, basado en IOT (internet de las cosas) [7]. Este protocolo presenta una gran ventaja dado que usa el principio de mensajeria basado en publicación/suscripción de tópicos en los que esta interesado el usuario. EL protocolo MQTT a través de un broker permitirá la comunicación entre la interfaz desarrollada en Python y un microcontrolador Raspberry, el cual se usará para controlar el brazo robótico. En la Fig.3 se muestra un ejemplo del funcionamiento del protocolo.

II. Metodologia

A. Acceso al Servicio

Para el acceso al servicio se utilizó toda la información y documentación para desarrolladores proporcionada por Emotiv en su guía Cortex API disponible online. El primer paso es conectarse al servidor Emotiv en la dirección 'wss://emotivcortex.com:54321' mediante una conexión tipo websocket segura, para el envió y recepción de mensajes.

Posteriormente se debe enviar el método *login* de inicio de sesión, para lo cual se debe encapsular el mensaje en tipología JSON, y mediante el websocket se envía el mensaje. El puerto

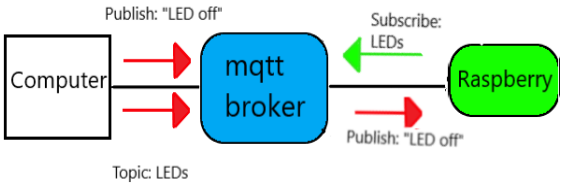

Fig. 3: MQTT protocol example diagram

está siempre escuchando, por lo que para recibir un mensaje se debe, decodificar de JSON cada mensaje que llegue. Esto se repite mediante varios métodos donde se debe establecer los parámetros de usuario para recibir con una autenticación única (token) que sea capaz de mantener abierta la sesión del usuario en el servicio. En la Fig. 4 se muestra un diagrama de flujo de los métodos que se debe enviar.

Fig. 4: Methods sequence to access Cortex

Para el flujo de datos (data stream) de los sensores de EEG se debe tener una suscripción de paga; sin embargo, para el flujo de datos de los sensores de movimiento se tiene acceso gratuito. Todo este procedimiento se da a través de una única identificación de Emotiv llamada **client secret** la que se otorga luego de un registro como desarrollador de aplicaciones en la página web de Emotiv. El método final es llamado **session** donde se deben especificar los tipos de stream que se requieren. En este caso son descritos como *eeg* para las señales de sensores EEG y *mot* para las señales de los sensores de movimiento [2].

B. Adquisición de datos

Luego de la petición de datos se crea un bucle de tipo "while" que permite escuchar constantemente todo lo que llega del puerto, en este caso lo que llega son los streams solicitados, la rapidez del stream es de 128 mensajes JSON por segundo por cada stream solicitado. Cada segundo se deben manejar 256 mensajes, los mismos que deben ser decodificados de JSON para ser tratados como listas o tuplas. Para clasificar los datos de ingreso se usa un condicional el cual busca una palabra *eeg* y *mot*, tal como se muestra en la Fig.5.

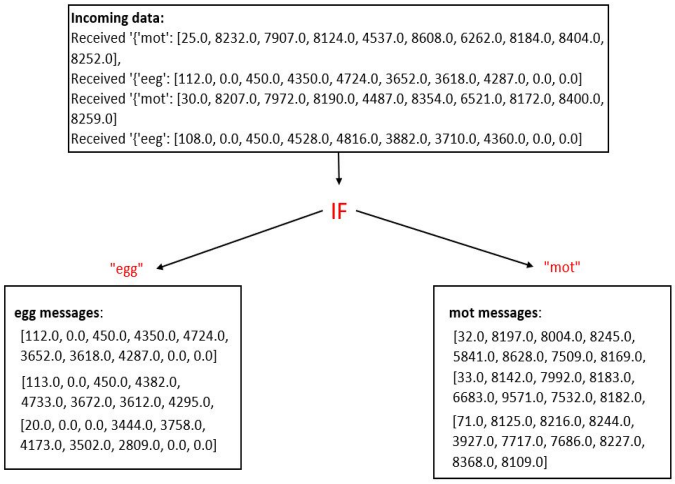

Fig. 5: Data stream filtering routine

Dentro de cada lista se encuentran los valores de cada sensor. En la primera clasificación se encuentran los valores de señales eeg de los canales de los sensores, cada canal se encuentra en una posición específica del arreglo, que para eeg es de 10 elementos. De la misma manera el segundo grupo de datos son los provenientes del stream mot, dentro de esta lista está contenida la información de los canales de los sensores de movimiento, la misma que cuenta con 10 elementos.

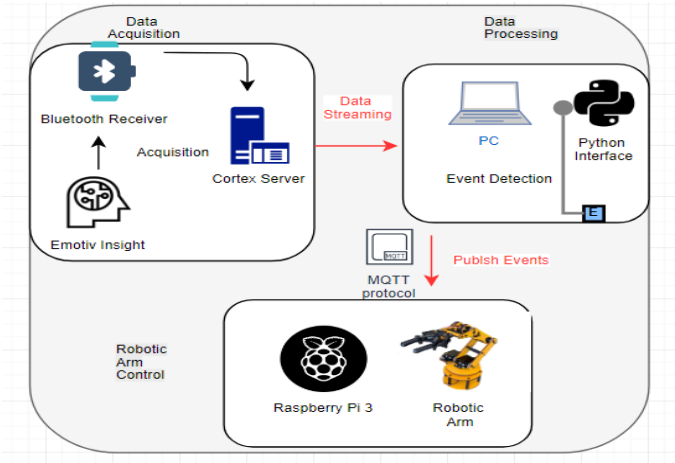

Fig. 6: Data Acquistion and Data Processing

En la Fig. 6 se muestra el camino de flujo de datos, desde su adquisición hasta su representación en señales de control en el brazo robótico.

C. Sensores de interés

En un principio se evaluaron los datos de entrada con el casco en uso para así obtener información de los sensores y determinar que señales se puede usar para reconocer eventos.

Se observó que las señales más representativas se produjeron en los canales AF3 y AF4, los cuales respondían a cambios marcados de amplitud debido a pestañeos y movimientos faciales correspondientes a la parte frontal de la cabeza [5]. Por lo que se descarto el uso de los otros sensores para esta aplicación, el sensor AF3 tuvo un cambio marcado de amplitud con un pestañeo del ojo izquierdo. Por otro lado, el sensor AF4 respondía en amplitud al pestañeo del ojo derecho. De esta manera para el procesamiento de la señal se usan los elementos que se encuentran en las posiciones del arreglo EEG(3) y EEG(7). El arreglo se muestra de la siguiente manera luego de la decodificación del mensaje en JSON : 'eeg': [116.0, 0.0, 450.0, 4438.0, 4847.0, 3801.0, 3530.0, 4313.0, 0.0, 0.0],

Para el stream mot de los sensores de movimiento, las cambios en amplitud no estaban marcadas en solo un canal para un mismo evento como en el caso de eeg. Las respuestas en los sensores de movimientos se dieron en pares, por lo que se observó que varios canales respondían para un mismo evento. Es así, que se pudo determinar que los canales que se debian usar serían: Girox, Giroy, Giroz, Accx, Accy, Accz. Los mismos que corresponden a los valores del acelerómetro y giroscopio del headset Emotiv insight. Las posiciones de los elementos dentro del arreglo son MOT(1), MOT(2), MOT(3), MOT(4), MOT(5), y MOT(6), respectivamente. El arreglo se recibe de la siguiente manera: 'mot': [114.0, 8311.0, 8362.0, 8149.0, 6274.0, 10487.0, 5342.0, 8226.0, 8416.0, 8155.0],

D. Empaquetamiento de datos y tratamiento de señal

Para tratar los datos de las señales se usó la estructura de datos "Queue" para almacenar un número determinado de datos en un arreglo controlado a manera de FIFO (First In First Out) [12]. De esta manera, se pueden empaquetar los datos para procesarlos en pequeños arreglos dependiendo de los segundos con los que se quiera trabajar. En este caso se observó que para el control del brazo robótico se requería empaquetar en pedazos de 256 datos, es decir 2 segundos de stream. Dado que en esta longitud de arreglos se tenía una mayor respuesta en la recolección de eventos de los canales, cada nuevo dato que llega se posiciona a la cola después del dato anterior, cuando se alcanza los 256 datos estos son procesados en bloque para la detección de eventos. El siguiente procesamiento se realizará cuando lleguen otros 256 datos nuevos y así se seguirá repitiendo el proceso.

En el caso de las señales de los sensores AF3 y AF4, para determinar eventos, mediante una función se buscan dentro de los arreglos valores máximos que superen un valor umbral, el cual se determinó experimentalmente. En este caso luego de empaquetar los datos se los convierte de tuplas a arreglos para, de esta manera, implementar funciones matemáticas capaces

de buscar valores umbral, (máximos y mínimos) dentro de pequeños arreglos.

Para la búsqueda de valores máximos y mínimos se utilizaron las funciones de la librería numpy, de esta manera, cada vez que las señales en los canales superan los umbrales, la función envía un mensaje de control hacia el brazo robótico por medio del protocolo MQTT [8]. Por ejemplo, un pestañeo simultáneo en los canales generará una acción de control. En la Fig.7 se muestra un ejemplo de la manera como las señales de EEG superan los valores umbral debido a un pestañeo, una señal de control se enviará al brazo robótico en caso de una condición de pestañeo simultaneo con los dos ojos.

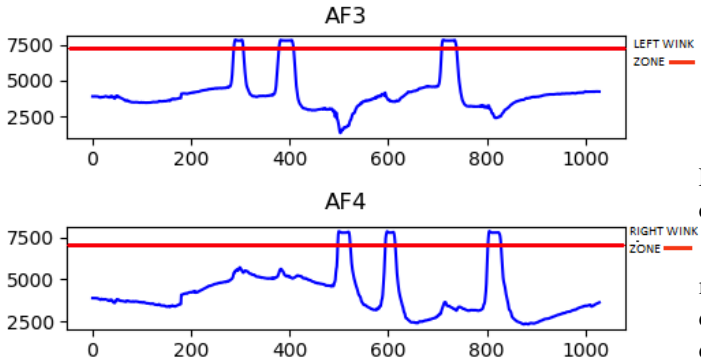

Fig. 7: Los máximos en los cambios de amplitud en AF3 muestran pestañeos izquierdos y en AF4 muestran pestañeos derechos

Para el stream de los sensores de movimiento *mot* se usa el mismo sistema FIFO. El procesamiento para la detección de eventos se da de tal forma que, se evalúan pares de canales de distinto tipo de sensores por cada evento. Para detectar un evento, se necesitan de dos condiciones en dos canales diferentes. La característica principal para los movimientos de rotación (izquierda y derecha) [11] de la cabeza del usuario son los cambios en amplitud de los elementos (1) y (5) del mensaje "mot" que corresponden al valor del giroscopio en X y del acelerómetro en Y, respectivamente. Tal como se muestra en la Fig. 8. En este caso se busca en los 2 arreglos, para determinar que el movimiento es a la izquierda los máximos en los dos canales deben superen un valor umbral. Similarmente para determinar que el movimiento es a la derecha, se debe identificar que los mínimos de las señales de los dos canales se den en el mismo arreglo.

La siguiente señal de control utilizada fue los movimientos de flexión y extensión de la cabeza, producidos al direccionar la barbilla hacia el tórax y el movimiento opuesto. En este caso la característica principal del evento fueron los cambios en amplitud en los elementos giroscopio en Y, elemento (2) y acelerómetro en Z elemento (6). En este caso los eventos se identifican de la siguiente manera: arriba, cuando se genera un máximo en la señal del acelerómetro en Z y un mínimo en la señal del giroscopio en Y, ambos dentro del mismo arreglo. Para determinar que el movimiento es hacia abajo, se usa el

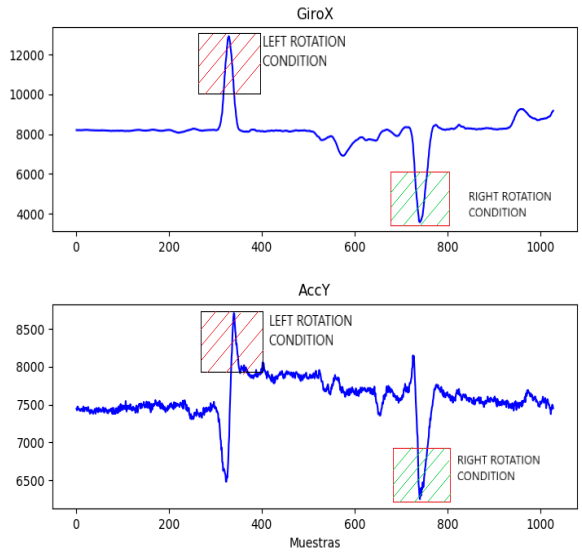

Fig. 8: Condiciones para detectar un evento de rotación, en canales GiroX, y AccY

máximo de la señal del giroscopio y el mínimo de la señal del acelerómetro, los mismos que también deben producirse dentro del mismo arreglo. Tal como se muestra en la Fig. 9.

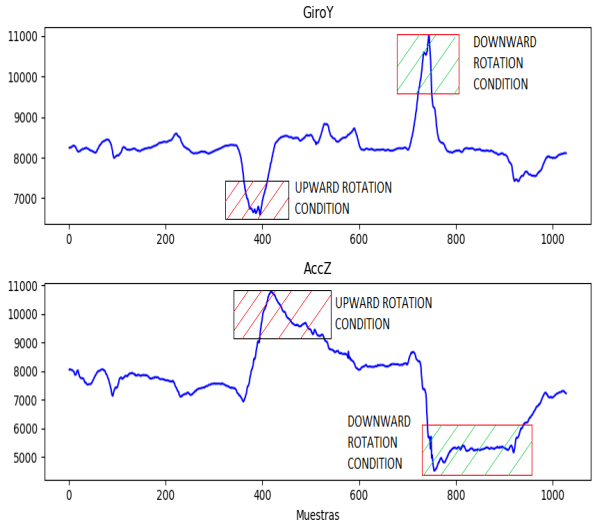

Fig. 9: Condiciones para detectar un evento de flexión y extensión

Se implementó además una tercera señal de control por movimiento descrita como rotación, la cual controla los servo motores del brazo que soportan la estructura de la pinza. En este caso se usaron las señales del giroscopio en Z y acelerómetro en Y. Elementos (3) y (5) respectivamente. Los cuales reaccionaban en amplitud frente a movimientos de flexión lateral, movimientos donde la cabeza se acerca a los hombros del individuo. La rutina de detección busca que se cumplan condiciones simultaneas de máximos y mínimos en las señales de los 2 canales. Los picos de amplitud generan

acciones de control donde el servo gira hacia los cero grados con respecto a la posición mecánica del mismo, en este caso se lo llamó sentido horario, y en su movimiento contrario (movimientos antihorarios), se genera en el servo motor un giro opuesto de 90 grados. Estas señales se muestran en la Fig. 10.

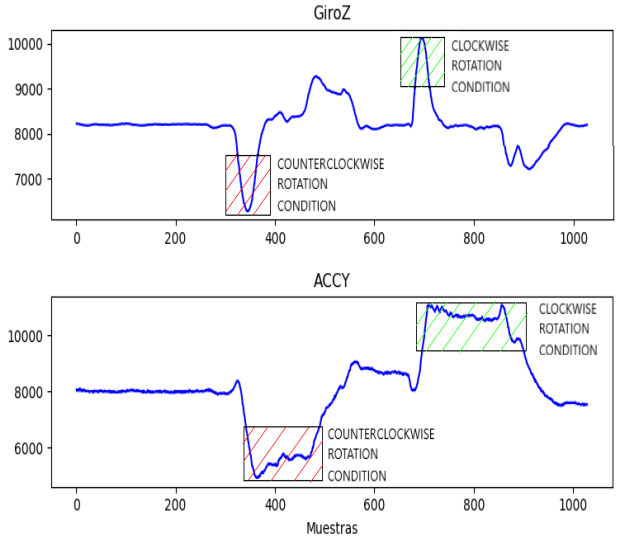

Fig. 10: Condiciones para detectar un evento de flexión lateral

E. Control de servos

Los servomotores del brazo robótico, mostrados en la Fig. 11, son controlados mediante streams de mensajes por medio del protocolo MQTT. Para ello se utilizó una microcomputadora Raspbberry Pi 3, en la cual se ejecuta el sistema operativo Raspbian de Linux. De igual manera, el sistema de control está basado en Python. Cada servo es controlado por una señal de control distinta, la cual llega luego de haberse suscrito al stream correspondiente. El control de los servos está dividido en dos partes. El primer control esta dado por los streams de las señales correspondientes a los sensores de EEG, dentro de los cuales se implementó un algoritmo para cambio de estado. Es decir, cada vez que llega una señal del stream de pestañeo los servos cambian entre dos posiciones establecidas, es decir la misma señal de control abre y cierra la pinza del brazo robótico con la misma señal de control, pero dependiendo de la posición en la que se encuentra el servomotor.

Fig. 11: Brazo robótico y posición de servomotores

El segundo control está dado por los streams provenientes de las señales de movimiento. De la misma manera, se implementó un algoritmo de detección de cambio de estado el cual permite cambiar la posición de los servos que controlan el movimiento del brazo. Sin embargo, en este caso llegan señales de control diferentes para cada uno los servos.

La Tabla I, muestra los canales utilizados, la subscripción al tópico del que se desea recibir los mensajes, el mensaje que se recibe y el servomotor sobre el que se realiza la acción de control.

Robotic Arm Control			
Channels	Subscription	Message	Action
AF3,AF4	Double Wink W	Wink	Servo 6(Claw)
GiroX,AccY	RLControl	Right, Left	Servo 1
GiroY,AccZ	UDControl	Up, Down	Servo 4
GiroZ,AccY	Rotation	Clockwise Counter-clockwise	Servo 5

TABLE I: Tabla de Control de Brazo Robótico

Se agregó además una interfase de usuario donde se pueden observar las acciones que se han realizado en la consola PowerShell de Windows, tal como se muestra en la Fig. 12.

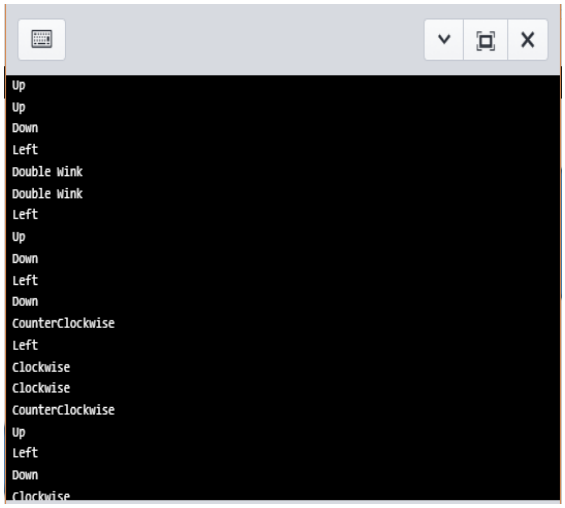

Fig. 12: Interfaz de usuario en consola

III. Resultados y Experimentación

Para comprobar la eficacia de los algoritmos implementados además del correcto uso del servicio de Cortex, se realizaron pruebas con 10 sujetos los cuales debían completar la tarea de levantar una esponja con la pinza y llevarla al lado opuesto

de la posición inicial del brazo. En la Fig.13 se muestra la configuración utilizada en el experimento.

Fig. 13: Implementación de Experimento

Las pruebas se realizaron de manera exitosa en los 15 sujetos, sin embargo, el tiempo de duración para completar la tarea varia mucho entre las distintas personas que participaron en el experimento, tal como se muestra en la siguiente Fig.14.

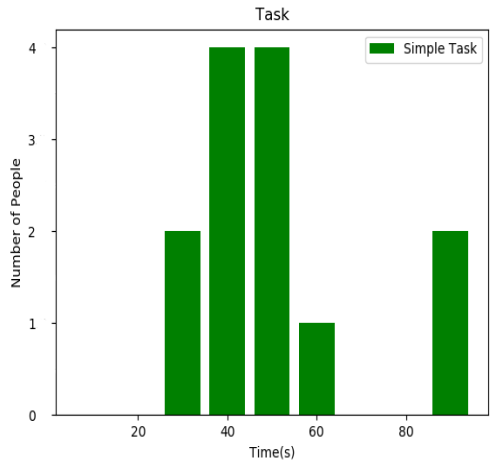

Fig. 14: Diagrama de barras. Se muestra el tiempo en que realizaron la rutina propuesta y cuanto tiempo les tomo a 15 sujetos de prueba

Los sujetos mas rápidos completaron la tarea en 30 segundos, sin embargo se observa una marcado promedio de 8 individuos los cuales realizaron la tarea asignada en un tiempo de entre 40 a 60 segundos. Por lo que se puede inferir que el tiempo en el que se puede completar esta tarea, la mayoría de veces sera menos de un minuto.

El principal inconveniente que se tiene, es la familiarización inicial que requiere cada individuo con cada acción que puede ser realizada por el brazo robótico. Las acciones con menor precisión se dieron, en el servo controlado por los movimientos de flexión lateral. Por otro lado, el control del brazo mediante los movimientos de cabeza hacia arriba, y hacia abajo e izquierda a derecha resultaron muy fáciles de ejecutar por los usuarios. El accionamiento de la pinza disparada por el pestañeo simultaneo de los dos ojos fue muy variante, lo cual puede deberse a la diferencia en el tamaño de las cabezas de las personas; ya que, al momento de colocarse el casco, la posición de los sensores AF3 y AF4 varía según la persona. En unas personas el proceso resulta ser adecuado y en otras se requiere un poco mas de esfuerzo hasta que se familiaricen con la acción.

Una segunda prueba se realizó con 3 sujetos, los cuales debían realizar movimientos para comandar las acciones de control para todos los posibles movimientos del brazo robótico por 5 veces, en un orden pre-establecido, los resultados obtenidos en esta prueba se muestran en la Tabla II.

Accuracy Test						
Rondas	**1**	**2**	**3**	**4**	**5**	**Accuracy**
Sujeto 1	7/8	6/8	8/8	7/8	8/8	90%
Sujeto 2	8/8	6/8	7/8	8/8	7/8	90%
Sujeto 3	7/8	7/8	6/8	6/8	7/8	82%

TABLE II: Prueba de Precisón de Aciertos por Ronda

Como se puede observar, los resultados de la Tabla II reflejan el fácil control del brazo robótico a través del uso de las señales del casco, ya que se obtuvieron 36 aciertos de 40 acciones posibles en los sujetos 1 y 2. En el caso del sujeto 3, se obtuvo una menor cantidad de aciertos, principalmente en la acción de control controlada por el movimiento de flexión lateral. El número de aciertos por acción se muestran en la Tabla III. El peor resultado obtenido fue de 7 errores en 40 acciones.

Accuracy Test 2							
Action	**Up**	**Down**	**Left**	**Right**	**CW**	**ACW**	Double Wink
Sujeto 1	5/5	5/5	3/5	5/5	5/5	4/5	4/5
Accuracy	100 %	100%	60%	100%	100%	80%	80%
Sujeto 2	3/5	5/5	4/5	4/5	5/5	5/5	5/5
Accuracy	60%	100%	80%	80%	100%	100%	80%
Sujeto 3	5/5	5/5	5/5	3/5	2/5	4/5	4/5
Accuracy	100%	100%	100%	60%	40%	80%	80%

TABLE III: Prueba de Precisión de Aciertos por Acción

IV. Conclusión

En este trabajo se demostró el potencial de trabajar con el casco Emotiv Insight y el Cortex API, como una nueva alternativa para obtener señales EEG a bajo costo para el control de un brazo robótico. Se ha implementado una interfase capaz de trabajar con cualquier casco headset de Emotiv y

obtener señales crudas tanto de los sensores EEG como de los sensores de movimiento disponibles en el casco. En este trabajo se muestra el correcto uso del API mediante el uso de paquetes computacionales livianos y gratuitos además de aprovechar la gran cantidad de aplicaciones que estos tienen, como por ejemplo la, conexión a un raspberry pi mediante un protocolo MQTT, el cual brinda ventajas como el control a distancia sin necesidad de conexiones físicas. Como trabajo futuro se pueden explotar los otros servicios disponibles en el API CORTEX, dado que, al desarrollar este trabajo se observó que simplemente al cambiar el tipo de stream o solicitud se puede también trabajar con información ya procesada en los servidores de EMOTIV tales como: expresiones faciales, comandos mentales, niveles de estados mentales, etc. Adicionalmente, se puede aplicar las señales de control a otros tipos de actuadores.

REFERENCES

[1] Emotiv. (2018). Recuperado de https://www.emotiv.com/insight/

[2] Emotiv, (2017). Cortex API Documentation. Available in :https://emotiv.github.io/cortex-docs/introduction

[3] Krell Mario, S. S. (2013). pySPACE—a signal processing and classification environment in Python. Frontiers in Neuroinformatics, , vol. 7 40. Recuperado el 30 de agosto de 2018 desde: https://www.frontiersin.org/article/10.3389/fninf.2013.00040

[4] V. Pimentel and B. G. Nickerson, (2012). Communicating and Displaying RealTime Data with WebSocket, in IEEE Internet Computing, vol. 16.

[5] S. Aguiar, W. Yánez and D. Benítez, "Low complexity approach for controlling a robotic arm using the Emotiv EPOC Headset," 2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, 2016, pp. 1-6

[6] D. Benitez, S. Toscano and A. Silva, "On the use of the Emotiv EPOC neuroHeadset as a low-cost alternative for EEG signal acquisition," 2016 IEEE Colombian Conference on Communications and Computing (COLCOM), Cartagena, 2016, pp. 1-6.

[7] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," The Journal of Open Source Software, vol. 2, no. 13, May 2017, DOI: 10.21105/joss.00265

[8] McKinney W,(2012).Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.USA: O'Reilly Media, Inc.

[9] Python Software Foundation. Python Language Reference, version 3.7. Available at http://www.Python.org

[10] Sanei, S., Chambers, J. (2013). EEG Signal Processing. New York, NY: John Wiley Sons.

[11] Palastanga, N., Field, D., Soames, R., Gonzalez del Campo Roman, P. (2007). Anatomia y movimiento humano estructura y funcionamiento. Barcelona, Espana: Editorial Paidotribo.

[12] Lutz, M. (2011). Programming Python. Sebastopol, CA: O'Reilly Media Inc.