DISEÑO Y DESARROLLO DE UNA PLATAFORMA DE INTEGRACIÓN DIGITAL ENTRE UN DRONE Y UN DISPOSITIVO DE RECONOCIMIENTO GESTUAL

EDWIN ALFONSO CUBILLOS BOHÓRQUEZ

UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
FACULTAD SEDE SECCIONAL SOGAMOSO
ESCUELA DE INGENIERÍA ELECTRÓNICA
GRUPO DE INVESTIGACIÓN EN ROBÓTICA Y AUTOMATIZACIÓN INDUSTRIAL
GIRA-UPTC
2016

DISEÑO Y DESARROLLO DE UNA PLATAFORMA DE INTEGRACIÓN DIGITAL ENTRE UN DRONE Y UN DISPOSITIVO DE RECONOCIMIENTO GESTUAL

EDWIN ALFONSO CUBILLOS BOHÓRQUEZ

MONOGRAFÍA

DIRECTOR:

Ing.LUIS ARIEL MESA MESA, MSc.

UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
FACULTAD SEDE SECCIONAL SOGAMOSO
ESCUELA DE INGENIERÍA ELECTRÓNICA
GRUPO DE INVESTIGACIÓN EN ROBÓTICA Y AUTOMATIZACIÓN INDUSTRIAL
GIRA-UPTC
2016

AGRADECIMIENTOS

A mi familia, en especial a mi madre quienes han brindado su apoyo incondicional.

A las comunidades de desarrollo internacional, quienes siempre estarán dispuestas a dar apoyo mutuo y soporte a desarrolladores de todo el mundo.

A la Uptc, en especial a mis compañeros y docentes de la Escuela de Ingeniería electrónica.

CONTENIDO

LISTA DE FIGURAS 6
1. INTRODUCCIÓN 7
1.1 ESTADO DEL ARTE 8
1.2 IDENTIFICACIÓN DEL PROBLEMA 10
1.3 OBJETIVOS 10
1.3.1 OBJETIVO GENERAL 10
1.3.2 OBJETIVOS ESPECÍFICOS 10
1.4 MARCO CONCEPTUAL 11
1.4.1 PLATAFORMA DIGITAL 11
1.4.2. DRONE/DRON 11
1.4.3 CUADRICOPTERO 11
1.4.4 RECONOCIMIENTO GESTUAL 11
1.4.5 LENGUAJE DE PROGRAMACIÓN 11
1.4.6 LICENCIAS DE SOFTWARE 11
1.4.7 PROGRAMAS PREDECESORES / ANTECEDENTES 12
2. HARDWARE Y SOFTWARE 14
2.1 DISPOSITIVOS DE RECONOCIMIENTO GESTUAL 14
2.1.1 BRAZALETE MYO DE THALMIC LABS 15
2.2 DRONE 16
2.2.1 PARROT BEBOP DRONE 17
2.3 SELECCIÓN DE SOFTWARE 18
2.3.1 UNITY 5 18
2.3.2 PYTHON 18
3. DISEÑO DE LA PLATAFORMA DIGITAL 20
3.1 SOFTWARE PYTHON 20
3.2 INSTALACIÓN Y CALIBRACIÓN DEL MYO 20
3.3 INSTALACIÓN DE LA LIBRERÍA MYO PARA PYTHON 21
3.4 INTEGRACIÓN MYO-PYTHON 21
3.4.1 PROCESAMIENTO DE DATOS 24
3.5 PRUEBA FUNCIONAL DRONE 28
3.6 INTEGRACIÓN PYTHON - BEEBOP DRONE 29
3.7 INTEGRACIÓN FINAL 33

4. DESARROLLO DE LA PLATAFORMA DE INTEGRACIÓN DIGITAL 34
4.1 FUNCIONAMIENTO 35
4.2 PROTECCIONES DE LA PLATAFORMA Y ALERTAS 37
4.3 LICENCIA 39
4.4 VERSIÓN MEJORADA 39
5. PRUEBAS FUNCIONALES 40
6. RESULTADOS Y TRABAJOS FUTUROS 44
6.1 RESULTADOS 44
6.2 TRABAJOS FUTUROS 44
7. CONCLUSIONES 45
8. LOGROS 46
BIBLIOGRAFÍA 47
LISTA DE ANEXOS 49

LISTA DE FIGURAS

Figura 1. Interfaz AutoFlight 12
Figura 2. Interfaz MyoPilot 13
Figura 3. Myo + Parrot 3.0 13
Figura 4. Sensor Leap Motion 14
Figura 5. Dispositivo brazalete MYO de Thalmic Labs 14
Figura 6. Myo de Thalmic Labs 15
Figura 7. Gestos principales que detecta el MYO 16
Figura 8. Parrot Bebop drone 17
Figura 9. Calibración Myo 21
Figura 10. Ejes de referencia 22
Figura 11. Diagrama de flujo del programa Myo_Input.py 22
Figura 12. Clase principal para la lectura de datos del Myo 23
Figura 13. Método mejorado para calcular roll, pitch y yaw 24
Figura 14. Funciones para calcular roll, pitch y yaw actuales. 25
Figura 15. Función para estableces rango y zonas de muerte en los ángulos. 25
Figura 16. Función auxiliar para escalar los ángulos 26
Figura 17. Funciones de filtrado, y escalado de ángulos. 27
Figura 18. Función para la ejecución y procesamientos de datos del Myo 27
Figura 19. Valor obtenido para rotación en eje de referencia roll 28
Figura 20. Vista desde el drone a 100m, Monterrey Casanare 29
Figura 21. Relación de rotación Bebop drone 29
Figura 22. Diagrama de flujo primera prueba 30
Figura 23. Botones de prueba 31
Figura 24. Diagrama de flujo Drone_Int.py 31
Figura 25. Cabecera del programa Drone_int.py 32
Figura 26. Funciones para envió de comando al drone 33
Figura 27. Interfaz final creada sobre Python. 34
Figura 28. Estructura del Programa final 35
Figura 29. Carga de imágenes para la interfaz gráfica 35
Figura 30. Diagrama de flujo principal main.py 36
Figura 31. Gestos y movimientos para volar el drone 37
Figura 32. Líneas de código de protección 37
Figura 33. Mensajes de alerta y errores 38
Figura 34. Licencia escogida 39
Figura 35. Código alojado en GitHub 39
Figura 36. Conexión del brazalete Myo con el PC 40
Figura 37. Gestos observados en el Computador con la aplicación Myo Connect 41
Figura 38. Configuración de velocidad en rotores del drone 42
Figura 39. Pasos para desactivar la opción “Presentation Mode” 42
Figura 40. Combinación de gestos para despegar el drone. 43
Figura 41. Combinación de gestos para aterrizar el drone. 43

1. INTRODUCCIÓN

Este informe presenta los resultados del proyecto titulado “Diseño y desarrollo de una plataforma de integración digital entre un drone y un dispositivo de reconocimiento gestual”, el cual se desarrolla de forma consecutiva de la siguiente forma:

Primero se realiza una introducción de los dispositivos, software y sistema operativo que se van a utilizar en el desarrollo del proyecto destacando las razones por las cuales se seleccionan estos elementos respecto a otros presentes en el mercado.

Se procede a establecer la comunicación entre el dispositivo de reconocimiento gestual (MYO) con la plataforma de integración; se estudia el kit de desarrollo de software creado por Thalmic Labs y la librería disponible para el lenguaje seleccionado.

A continuación se realiza la integración de la plataforma en lenguaje Python con el drone Bebop y luego se realiza la integración final MYO-BEBOP-DRONE analizando los parámetros de funcionamiento y conectividad.

Durante cada fase del proyecto la plataforma estará en constante desarrollo tanto en forma estética como funcional.

Al finalizar, se presenta la conclusión sobre el desarrollo de la plataforma y se dan algunas recomendaciones para trabajos futuros que pueden potencializar el desarrollo de la plataforma propuesta.

1.1 ESTADO DEL ARTE

Uno de los retos en ingeniería en cuanto a la relación hombre-máquina es poder traducir la basta cantidad de movimientos que tiene el ser humano en señales digitales que pueda interpretar la máquina para realizar una tarea determinada. Para este propósito y con la ayuda de la técnica de la electromiografía (EMG) [1], se han desarrollado sensores capaces de seguir el movimiento en tiempo real, realizando un reconocimiento de patrones que pueden ser entendidos por una máquina.

Por ejemplo en [2], se utilizó Matlab para realizar la interfaz entre el sistema de reconocimiento gestual, en donde se crea una interacción hombre maquina a través de los movimientos de la mano y la cara para controlar el reproductor multimedia del computador. En [3] se utiliza el dispositivo Leap Motion para realizar el reconocimiento de diferentes gestos manuales y se crea una librería de reconocimiento de patrones de movimiento estáticos y dinámicos que sustituyen algunos comandos del teclado. Con esta misma visión, en [4] se realizó un proyecto para reemplazar el mouse y que la interacción con el entorno grafico del computador fuera más dinámica, diseñando un brazalete casero con sensores EMG y un sensor de unidad de movimiento inercial IMU [5] logrando un diseño aceptable para el reemplazo del mouse.

Un producto que cumple con las características físicas de un brazalete casero pero robusto y eficiente salió a la venta en el 2013 denominado “Myo” diseñado por Thalmic Labs. Este brazalete es capaz de detectar el movimiento de cada uno de los dedos gracias a los sensores basados en la técnica EMG, que evalúa y registra la actividad eléctrica producida por los músculos esqueléticos [6]. Este dispositivo puede tener diferentes usos, por ejemplo, al utilizar este brazalete y patrones de movimiento determinados, se puede controlar una presentación en Power Point® eliminando así el uso de controles o el teclado; también puede reemplazar el joystick de los videos juegos. En [7] sugieren algunas pautas sobre el uso del dispositivo para el control de mapas interactivos y resaltan ventajas como la ergonomía, facilidad de uso y confort que aporta el brazalete. La diversidad de aplicaciones del Myo va según la necesidad; por ejemplo en [8], está en estudio la viabilidad de leer movimientos ergonómicos en pacientes realizando auto exámenes y enviando posteriormente estos datos en tiempo real con el Myo.

Un uso interesante es convertir los movimientos de los dedos en comandos para controlar robots. Por ejemplo en [9] se realiza una plataforma con la ayuda de Autodesk Inventor, Matlab y Labview para operar un cuadricoptero con los gestos de las manos mostrando que es posible entablar una conexión entre un dispositivo de reconocimiento gestual y un AR Drone. En [10] se crea un sistema de reconocimiento de gestos con base en la plataforma FIRMWARE, este sistema recibe los datos obtenidos por los sensores de movimiento ya sea de un Leap Motion o Kinect y arroja un flujo de datos con los vectores que describen el movimiento del cuerpo humano. Un sistema más avanzado se da en [11], en donde se utilizó Java para realizar la plataforma que conecta el Kinect con un AR Drone y de esta manera controlar el dispositivo con movimientos simples de la mano y el cuerpo.

Con las aplicaciones de los drones no solo se amplía el campo de la investigación sino también se aprovechan las herramientas que se pueden enlazar con estos dispositivos. Un enlace se realiza en [12], donde se pretende mejorar la enseñanza de la aplicación del filtro de Kalman por el desarrollo de una matriz de sensores en un avión no tripulado Bebop, cuyo objetivo es detectar elementos dentro de un rango determinado para reducir el ruido de fondo en las mediciones obtenidas. En [13], se emplea un nuevo método de

estabilización de imagen basado en un filtro de respuesta de impulso finito (FIR) que pretende que las cámaras de video montadas sobre un Drone, capten imágenes limpias, sin alteraciones ocasionadas por el viento o vibraciones. Uno de los principales objetivos de las cámaras que llevan los drones, es poder tener un control de vuelo autónomo de interior para vehículos aéreos no tripulados, como en el caso de [14], donde se utilizó un AR Drone, una cámara Vicon y un algoritmo de control diseñado por Mathworks bajo el software Matlab Simulink para lograr un vuelo autónomo a partir de la posición.

En este proyecto se pretende entrelazar el Myo con un Parrot Drone, el cual es un vehículo aéreo no tripulado radio controlado que funciona propulsado por cuatro motores eléctricos [15]. Para esto se requiere realizar la interfaz en un software libre, comenzando con la conexión entre este y el Myo, para captar los movimientos de los dedos de la mano que tenga el brazalete. Luego, de acuerdo con estas señales, controlar el movimiento de los motores del Drone, reemplazando así el control para manejar el Parrot, otorgándole libertad al usuario.

1.2 IDENTIFICACIÓN DEL PROBLEMA

La evolución de la tecnología exige que la relación hombre-máquina ya no se vea limitada por un control, mouse o joystick en un entorno 2D sino que se presente en un ambiente 3D donde el usuario tenga libertad en sus movimientos y que estos puedan ser comprendidos por la máquina. Para lograr esto, se requieren plataformas de integración digital entre dispositivos de reconocimiento gestual y el elemento que se desee controlar.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Diseñar y desarrollar una plataforma de integración digital entre un drone y un dispositivo de reconocimiento gestual.

1.3.2 OBJETIVOS ESPECÍFICOS

1. Adquirir el hardware y software necesarios para la implementación.

2. Diseñar una plataforma digital en software libre para la integración de sensores y sistemas mecánicos por medio de señales inalámbricas.

3. Desarrollar un programa que permita al usuario operar un drone mediante los gestos de las manos.

4. Realizar pruebas de funcionamiento.

1.4 MARCO CONCEPTUAL

1.4.1 PLATAFORMA DIGITAL

Es un sistema desarrollado para conectar diferentes módulos de hardware y/o software, estableciendo parámetros de funcionamiento, sistema operativo, lenguaje de programación e interfaz de usuario (GUI), que son necesarias para lograr una correcta integración de los módulos. Una plataforma digital puede ser programada, personalizada o modificada por desarrolladores para innumerables usos ligados con licencias y derechos de copia.

1.4.2. DRONE/DRON

Vehículo aéreo no tripulado, también conocido con las siglas (VANT) y en inglés (UAV), es una aeronave que vuela sin tripulación y puede ser comandado a distancia. En un principio eran utilizados con fines militares pero con la evolución llego a tener usos civiles orientados a diversas finalidades.

1.4.3 CUADRICOPTERO

Un cuadricoptero es un helicóptero multi-rotor con cuatro brazos, los cuales tienen en su parte final un motor y una hélice; son similares a los helicópteros aunque la elevación y el empuje lo realizan con cuatro hélices, a menudo reciben otros nombres como: quadcopter, quadrotor, quad-copter, drone, UAV, entre otros.

1.4.4 RECONOCIMIENTO GESTUAL

El reconocimiento gestual tiene como objetivo interpretar gestos humanos con la ayuda sistemas compuestos de sensores y algoritmos matemáticos, los gestos son movimientos corporales como la postura y el caminar, pero principalmente los gestos se realizan con la cara y las manos.

1.4.5 LENGUAJE DE PROGRAMACIÓN

Es un lenguaje con el cual podemos describir un conjunto de acciones consecutivas que un equipo deberá interpretar y traducir para que el equipo final pueda comprenderlas, los lenguajes de programación fueron diseñados para que los humanos puedan dar órdenes a las maquinas.

1.4.6 LICENCIAS DE SOFTWARE

Una licencia de software establece el contrato entre el dueño del software y el usuario final. Las licencias están protegidas internacionalmente y especifican lo que el usuario tiene derecho a realizar con el software.

A continuación, se presentan algunas de las licencias de software:

- GNUGPL(General Public License): Esta licencia garantiza los derechos de autor, permitiendo la libre distribución, modificación y comercialización, con la condición que si se realiza modificaciones al software, debe ser distribuido bajo licencia GNU; es decir que el código debe estar disponible totalmente gratis, por ejemplo el sistema operativo Linux y sus distribuciones.

- BSD (Berkeley Software Distribution): La licencia creada para las distribuciones de Berkeley software, impone pocas restricciones sobre su uso, modificación y distribución. El software puede ser vendido y no hay obligación de incluir el código fuente ni la misma licencia, pero establece que se debe incluir los derechos de autor en el nuevo software, por ejemplo el sistema operativo MAC OS X Contiene código distribuido bajo licencia BSD.
- APACHE: Esta licencia permite al usuario distribuir y modificar, pero debe conservar el copyright y la exclusión de responsabilidades (disclaimer). No exige que las modificaciones sigan usando la licencia apache pero se debe informar que la fuente utiliza esta licencia.
- LICENCIA MIT: Licencia creada por el Instituto Tecnológico de Massachusetts, se considera una licencia permisiva al igual que la licencia BSD. La licencia concede permiso de forma gratuita, para modificación venta, sub licenciar, fusionar, publicar redistribuir el código, salvo la única condición de incluir el texto de la licencia en las nuevas copias. Por ejemplo, algunos software que usan esta licencia son Ruby on Rail y Bitcoin.

1.4.7 PROGRAMAS PREDECESORES / ANTECEDENTES

Actualmente existe una comunidad amplia desarrollando software de uso general con el Myo, combinado con drones y tecnologías emergentes. A continuación, presentamos los proyectos predecesores con una breve descripción.

1.4.7.1 AUTOFLIGHT

El software oficial lanzado en Abril de 2015 por Thalmic Labs, capaz de controlar el AR. DRONE 2.0, puede correr sin problema en Windows. La descarga se puede hacer desde la página oficial o en el blog oficial [16]. Este software combina movimientos y gestos para controlar el drone, el cual está programado en LUA (lenguaje madre de Myo). En la Figura 1 se observa la interfaz del software.

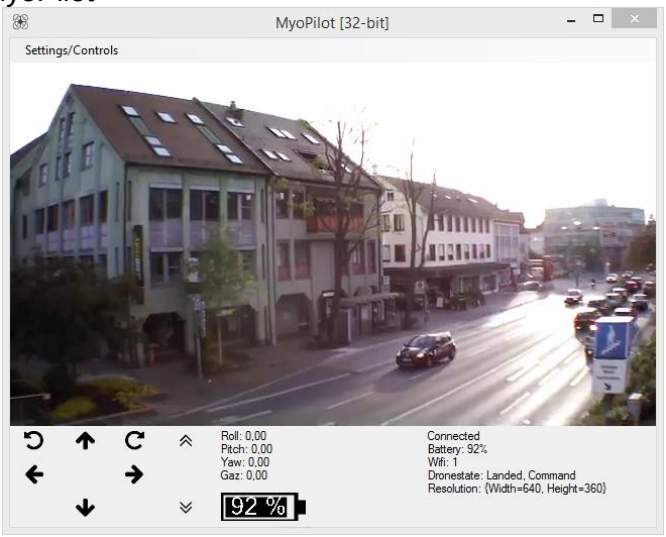
Figura 1. Interfaz AutoFlight

Fuente**:** Autoflight software

1.4.7.2 MYOPILOT

Desarrollado por Andreas Deguele, es un software no oficial y cumple con el mismo propósito de AutoFlight. A diferencia que este puede controlar el drone con el teclado y con un control de Xbox. Está programado en lenguaje de programación C# con interfaz de usuario en Visual Basic, está disponible en GitHub y se distribuye con licencia MIT. Fue publicado en julio de 2015 en el blog oficial de desarrolladores de Myo [17]. En la Figura 2 se observa su interfaz.

Figura 2. Interfaz MyoPilot

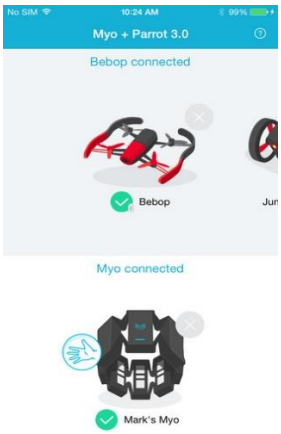


Fuente**:** MyoPilot software

1.4.7.3 MYO + PARROT 3.0

MYO+PARROT 3.0 es una aplicación desarrollada por Thalmic Labs para IOS y Android. Permite controlar todos los drones actuales de la marca Parrot vía WIFI y establece comunicación con el Myo vía Bluetooth. La aplicación no ofrece vista de cámara al usuario y los gestos están diseñados para acciones de control. Está disponible en la tienda oficial de Thalmic Labs [18]. En la Figura 3 se observa su interfaz principal.

Figura 3. Myo + Parrot 3.0



Fuente**:** Myo Market

2. HARDWARE Y SOFTWARE

Para la selección de hardware se tiene en cuenta la valoración de los usuarios a los dispositivos y su aceptación en el marcado.

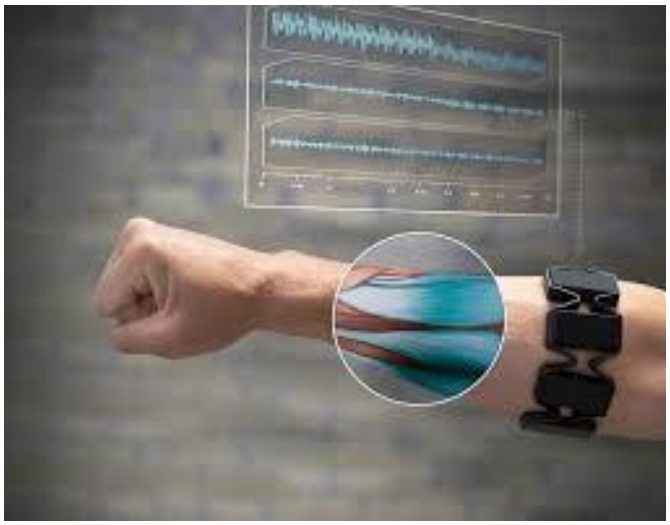
2.1 DISPOSITIVOS DE RECONOCIMIENTO GESTUAL

En la actualidad hay bastantes dispositivos para el reconocimiento gestual, este proyecto se focalizó en dispositivos capaces de leer gestos de las manos, para lo cual, los dos dispositivos que cuentan con mejor aceptación de los desarrolladores y que cumplen con los requerimientos son el sensor Leap Motion Figura 4, y el brazalete Myo, Figura 5.

Figura 4**.** Sensor Leap Motion

Fuente**:** Leapmotion.com

Figura 5. Dispositivo brazalete MYO de Thalmic Labs



Fuente**:** Thalmic Labs

En la tabla 1 se observa las principales características de cada dispositivo.

Tabla 1.Caracteristicas de los sensores seleccionados

Dispositivos	LEAP MOTION	MYO
Características principales	-Conexión USB - Reconocimiento múltiple - Software y SDK's disponible para multiples Sistemas Operativos y Plataformas - Sistema abierto para desarrolladores - Amplia documentación y soporte ofrecida - Dirigido Principalmente a desarrolladores	-Conexión Bluetooth - Software y SDK's disponible para multiples Sistemas Operativos y Plataformas - Sistema abierto para desarrolladores -Amplia documentación y soporte ofrecida -Dirigido a público en general y Desarrolladores
Precio	79.99 USD	199 USD

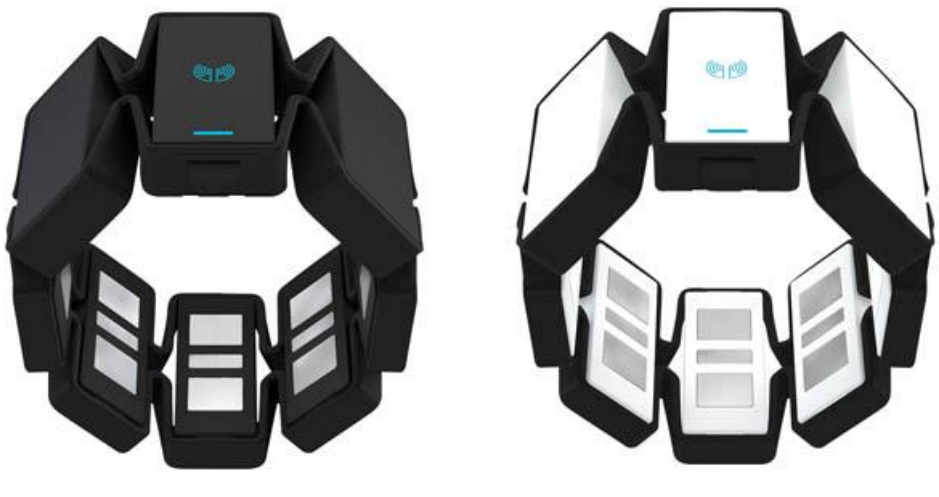
Fuente**:** Autor

Se elige el brazalete Myo principalmente porque permite al usuario trasmitir datos de forma inalámbrica y no limita al usuario a permanecer a determinada distancia como lo hace el Leap Motion.

2.1.1 BRAZALETE MYO DE THALMIC LABS

El Myo es un dispositivo desarrollado por Thalmic Labs, el cual salió al mercado a principios de 2013 con el propósito de controlar diferentes dispositivos con gestos de los brazos, dirigido principalmente a desarrolladores e innovadores [19]. En la Figura 6 se observa los dos modelos disponibles.

Figura 6**.** Myo de Thalmic Labs



Fuente**:** Thalmic Labs

El MYO se coloca en el antebrazo debido a que en este punto se pueden recibir los movimientos del brazo, la muñeca y los dedos, creando un abanico de posibilidades a la hora de configurar el dispositivo.

El brazalete recoge los impulsos eléctricos del brazo, y los envía directamente al ordenador, cuando esto sucede, depende del programador configurar las órdenes. El MYO viene configurado para cinco gestos principales, como se observa en la Figura 7.

Figura 7**.** Gestos principales que detecta el MYO

Fuente: Thalmic Labs, modificada con comentarios en español.

Además de esto el MYO posee un giroscopio y un acelerómetro que permiten entregar valores de rotación en los tres ejes x, y, z; estas rotaciones son denotadas como roll, pitch & yaw respectivamente.

MYO de Thalmic Labs cuenta con varios SDK (Software Development Kit) compatibles con sistemas operativos como iOS®, Android®, Mac OS®, Windows® entre otros, que permiten a los desarrolladores crear aplicaciones dependiendo el campo de desarrollo, que puede variar desde aplicaciones a la medicina, hasta usos militares. La Tabla 2 muestra las principales características del brazalete.

Tabla 2.Principales características brazalete Myo

Peso	93 gr
Circunferencia expandible	19-34 cm
Procesador	ARM Cortex M4
Indicadores	Led e indicador de respuesta por medio de vibración
Batería	Ion litio / 24h
Conexión	Bluetooth con tecnología Smart Wireless

Fuente**:** Autor

2.2 DRONE

Frente al amplio mercado de drones, se seleccionan las tres marcas principales y sus equipos insignia, evaluando sus características y precio. La Tabla 3 muestra las principales características de estos drones.

Se elige el Bebop drone ya que sus características y precio se ajustan a los requerimientos del proyecto.

Tabla 3. Principales características de los drones seleccionados

Dispositivo	Características principales	Precio
Phanton DJI	-Alcance de 1Km -25 Minutos de Vuelo -Facil Manejo -SDK's para desarrolladores -Disponible para público en general	1200 USD
SOLO (3DR Robotics)	-Facil Manejo -Diseño Robusto -SDK para desarrolladores -20 Minutos de vuelo -Alcance 800 m	999 USD
PARROT BEBOP	-Fácil Manejo -Diseño esquemático -Soporte y reemplazo de partes -Abierto a desarrolladores -10 Minutos de vuelo -Alcance de 1 KM -Cámara de alta resolución integrada	499 USD

Fuente**:** Autor

2.2.1 PARROT BEBOP DRONE

En la Figura 8 se muestra el Parrot Bebop Drone el cual es un equipo de la empresa francesa Parrot, la cual desarrolla vehículos no tripulados, radio controlados de uso recreativo civil, el cual cuenta con varios SDK en distintas plataformas y sistemas operativos para los desarrolladores. El Bebop drone [20] es el siguiente lanzamiento después del AR. DRONE 2.0

Figura 8**.** Parrot Bebop drone

Fuente**:** www.parrot.com

Se selecciona la versión Bebop drone que cuenta con las siguientes características:

- ✓ Tienen un peso ligero (400g) y estructura reforzada con fibra de vidrio que hace el dispositivo sólido.
- ✓ En caso de impacto, las hélices se detienen automáticamente por lo que se convierte en un dispositivo seguro a la hora de volar.
- ✓ Procesador dual Core Parrot P7, procesador gráfico Quad Core y memoria flash interna de 8G.
- ✓ El circuito está montado en un soporte de magnesio que cumple el papel de blindaje electromagnético y sistema de refrigeración.
- ✓ Cuenta con una conexión Wi-Fi MIMO (Multiple Input Multiple Output): El Parrot Bebop drone tiene 2 antenas de doble banda Wi-Fi que le permiten gestionar las dos frecuencias 2,4 GHz y 5 GHz en MIMO. Genera su propia red, según las últimas normas Wi-FI IEEE 802.11.
- ✓ Todas las piezas son desmontables para facilitar el transporte del Bebop drone.
- ✓ Los pilotos pueden concentrarse en sus vuelos y realizar imágenes y vídeos de alta calidad.
- ✓ Cuenta con una cámara de 14 mega-pixeles integrada y puede girar en un Angulo de 180°
- ✓ Cuenta con un GPS embebido.

2.3 SELECCIÓN DE SOFTWARE

En este proyecto se estudiaron dos opciones para la selección de software realizando la instalación y comprobación de cada software que se describe a continuación.

2.3.1 UNITY 5

Es una plataforma de desarrollo flexible y poderosa para crear juegos y experiencias interactivos 3D y 2D multiplataforma. Es un ecosistema completo para todo aquel que busque desarrollar un negocio a partir de la creación de contenido de alta gama y conectarse con sus jugadores y clientes más fieles y entusiastas; es posible programar los script de esta plataforma en C# o JavaScript.

Siguiendo los pasos para el desarrollo del proyecto se inició integrando el dispositivo Myo con Unity, usando el SDK disponible. Con este software se logró la integración, pero se desistió de usar este software ya que no se contaba con una librería compatible para lograr la conexión adecuada con el drone.

2.3.2 PYTHON

Python es un lenguaje de programación de alto nivel, interpretado y multipropósito, el cual está orientado a objetos y puede usarse para desarrollo Web. Puede ser utilizado en diversas plataformas y sistemas operativos, cómo Windows, Mac OS y Linux, así como también puede funcionar en Smartphones, gracias a las librerías y su constante actualización basado en el modelo OPEN SOURCE.

Actualmente Python es uno de los lenguajes más utilizados, por ejemplo, Walt Disney, la NASA, Google, Yahoo!, Red Hat, Nokia, Yale University y la Universidad de California,

hacen uso de esté lenguaje para desarrollar sus productos, servicios, páginas web, complementos y juegos, lo que demuestra el gran potencial de Python a nivel mundial.

Se eligió este lenguaje de programación ya que el drone contaba con una librería creada y probada por terceros, sustentado de pruebas básicas que comprueban su viabilidad con el drone seleccionado y el dispositivo Myo.

3. DISEÑO DE LA PLATAFORMA DIGITAL

Durante esta fase se instalan el lenguaje de programación seleccionado, el driver, el software necesarios para cada fase y se integran parcialmente los módulos antes de pasar al desarrollo final.

3.1 SOFTWARE PYTHON

La instalación de Python es ágil y sencilla; el software está disponible en la página oficial https://www.python.org/, donde se encuentra las diferentes versiones y actualizaciones dependiendo el sistema operativo. Para este proyecto se eligió la versión 2.7.11, ya que es la versión en la que se desarrolló la librería para el drone. Si se usan versiones más recientes, se debe ajustar el código a los cambios de la nueva versión lo que implicaría líneas de código que extenderían el proceso y alargarían el desarrollo del proyecto.

Para programar en Python se puede ingresar al símbolo de sistema y escribir "*Python"***,** de esta manera, se direcciona directamente al modo de programación escribiendo línea por línea. Es necesario, contar con un editor de texto, como el bloc de texto de Windows o uno avanzado para programación tal como Notepad++. Los archivos se deben guardar con la extensión ***".py",*** para ser reconocidos por el lenguaje, y para ejecutarlo se escribe en el símbolo de sistema, "*python name.py*", y el programa se ejecutara.

3.2 INSTALACIÓN Y CALIBRACIÓN DEL MYO

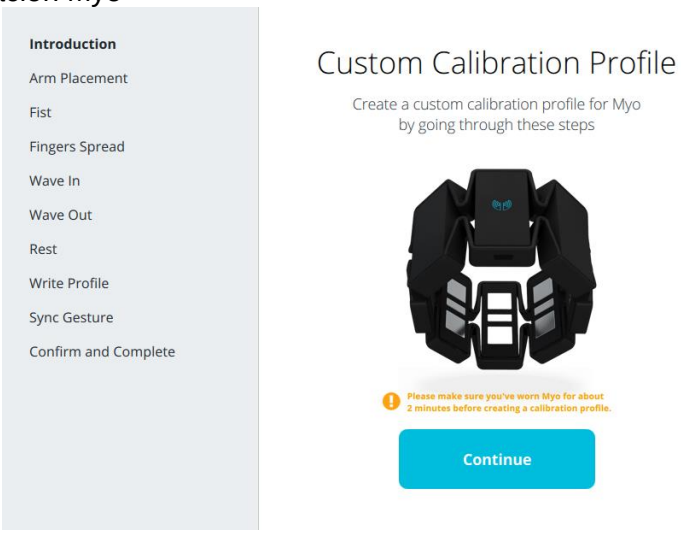
Primero se procede a descargar de la página oficial de MYO el software "Myo Connect" el cual viene con los drives y un programa para enlazarse con el computador.

Cuando está instalado el software se procede a realizar la calibración. Se debe tener en cuenta que el Myo se considera un sensor personal, ya que cada usuario tiene comportamientos ligeramente diferentes en sus gestos; por tanto la calibración se logra realizando una serie de movimientos y gestos predeterminados por cada usuario que vaya a utilizar el sensor.

A continuación, se realiza un test de funcionamiento donde se comprueba el correcto envió y recepción de datos traducidos a gestos. En la Figura 9 se observa la pantalla de calibración del MYO. Con la ayuda de Unity 5, se realiza un script para probar la recepción de datos en tiempo real, esta prueba se puede apreciar en [21].

Para finalizar se realiza la prueba de alcance del dispositivo (MYO-PC), en la cual se determina que el rango de distancia óptima para garantizar la conexión es de 0-10m con línea de vista.

Figura 9. Calibración Myo



Fuente**:** Thalmic Labs

3.3 INSTALACIÓN DE LA LIBRERÍA MYO PARA PYTHON

Para este proyecto se utiliza la librería desarrollada por Niklas Rosenstein [22]. La librería se puede instalar manualmente o ejecutando el comando *-pip install myo-* en el símbolo de sistema de Windows. La librería necesita acceder al archivo de conexión del Myo, *myo32.dll* o *myo64.dll*, según sea el sistema operativo. Estos archivos están disponibles en el SDK oficial distribuido por Thalmic Labs. La librería también hace uso del complemento "*six*", el cual se puede obtener ejecutando el comando *–pip install six-***.** Cumpliendo los anteriores requerimientos se obtiene el correcto funcionamiento de la librería Myo para Python.

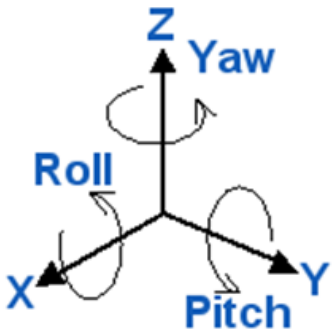
3.4 INTEGRACIÓN MYO-PYTHON

Se procede a programar en Python una secuencia que muestra un valor de entre [-1,1], correspondiendo al grado de cambio de posición del Myo; es decir, la medida del ángulo relativo entre dos posiciones, lo cual se logra obteniendo los valores enviados por el Myo y convirtiéndolos en valores separados para roll, pitch & yaw. La Figura 10 muestra la relación de cada rotación con su eje respectivo.

El código necesario para adquirir y procesar los datos está disponible en distintas versiones, una de ellas es la versión en C# usado en el MyoPilot. Este código sirve de guía para lograr la correcta lectura de datos y posterior procesamiento para el desarrollo de la plataforma.

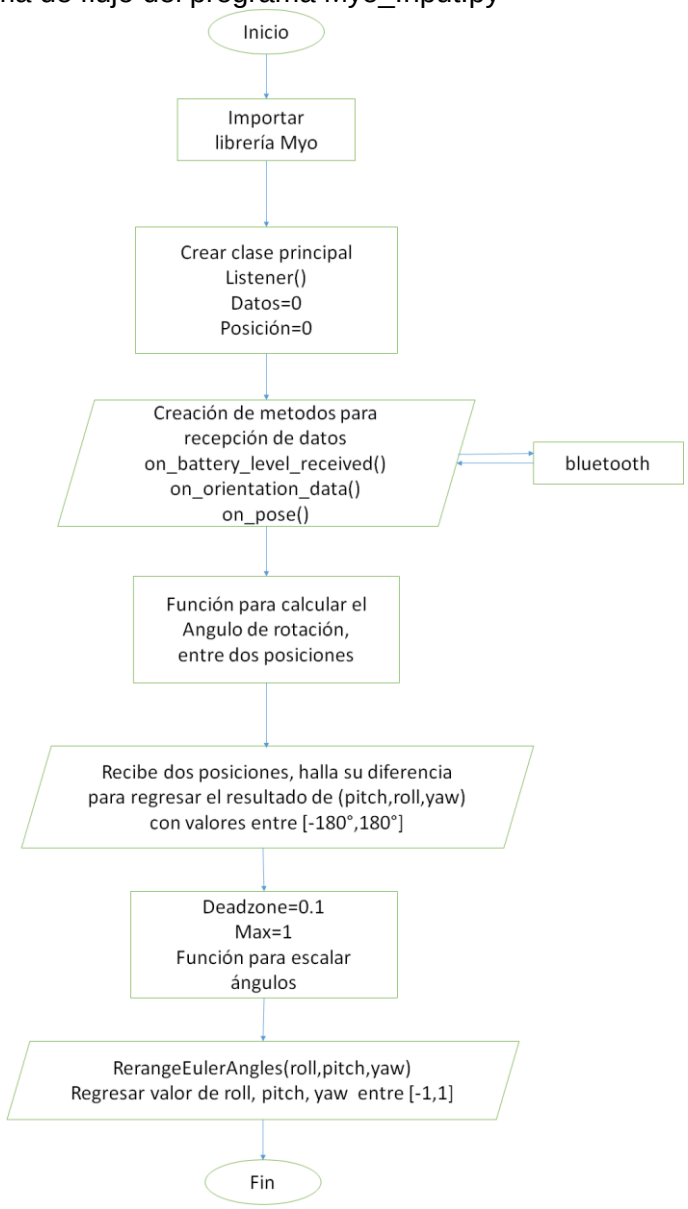
El sub-programa encargado de gestionar la entrada de datos del MYO, se denomina "*Myo_Input.py*". Este código contiene los atributos necesarios para iniciar la aplicación, leer los datos del MYO y ajustarlos para ser enviados, en el diagrama de flujo mostrado en la Figura 11 muestra la estructura y secuencia del programa.

Figura 10. Ejes de referencia



Fuente: Thalmic Labs

Figura 11. Diagrama de flujo del programa Myo_Input.py



Fuente: Autor

Para el desarrollo de este programa se parte del ejemplo divulgado por el desarrollador de la librería, el cual está disponible en [22], este ejemplo captura los datos del Myo y los imprime dependiendo una determinada posición, en la Figura 12, se muestra las líneas iniciales obtenidas después de eliminar secuencias, funciones y procesos básicos propios del código "*hello_myo.py",* dejando líneas estructurales para la construcción de aplicaciones e incluyendo nuevas instrucciones para la el procesamiento de datos del programa.

Figura 12**.** Clase principal para la lectura de datos del Myo

```
class Listener(libmyo.DeviceListener):
    """
    Listener implementation. Return False from any function to
    stop the Hub.
    """
    def __init__(self):
        super(Listener, self).__init__()
        self.orientation = None
        self.pose = libmyo.Pose.rest
        self.emg_enabled = False
        self.locked = False
        self.rssi = None
        self.emg = None
        self.last_time = 0
        self.currentOrientation=[0,0,0,0]
        self.referenceOrientation=[0,0,0,0]
        self.level_battery=None  #0
        self.myo_sync=False
        self.pair=False

    def on_connect(self, myo, timestamp, firmware_version):
        myo.vibrate('short')
        myo.request_rssi()
        myo.request_battery_level()
        #self.myo=myo

    def on_pose(self, myo, timestamp, pose):
        if (pose == libmyo.Pose.fingers_spread or pose==libmyo.Pose.fist):
            self.referenceOrientation=self.currentOrientation

        self.pose = pose

    def on_orientation_data(self, myo, timestamp,quat):

        self.currentOrientation=[quat.x, quat.y, quat.z, quat.w]
        #self.rpy=quat.rpy

    def on_unlock(self, myo, timestamp):
        self.locked = False

    def on_lock(self, myo, timestamp):
        self.locked = True
```

Fuente**:** Autor

Cada una de estas funciones se ejecutan al suceder el evento, por ejemplo la función *on_pose* , se ejecuta cuando el brazalete detecta un cambio en algunos de sus gestos, para este caso, cuando se realiza el gesto, *dedos separados* o *puño,* se actualiza los valores registrados en variables que guardan la orientación, con esta clase llamada *Listener,* se captura los datos del sensor, como lo son la orientación (datos del giroscopio), niveles de

batería y el gesto leído, estos datos son enviados desde el sensor por medio de señales bluetooth, posteriormente estos son procesados por medio de funciones para procesamiento de datos.

Los datos de orientación son alojados en un vector de tipo quaternion, estos datos contienen los valores del giroscopio, lo cuales son procesados para obtener los ángulos y posteriormente valores entre [-1,1], la librería ya contiene el método para calcular los ángulos roll, pitch y yaw, pero al presentarse una falla o bug en este método se hace necesario reescribirlo en el código con sus respetivas restricciones para evitar las fallas detectadas, la Figura 13 muestra este método con las adiciones.

Figura 13**.** Método mejorado para calcular roll, pitch y yaw

```
129  def CalculateAnglesm(quat):
130      """
131      calculate ROLL, PITCH and YAW
132      """
133
134      x, y, z, w = quat[0], quat[1], quat[2], quat[3]
135
136      roll = math.atan2(2.0*y*w - 2.0*x*z, 1.0 - 2.0*y*y - 2.0*z*z)
137      pitch = math.atan2(2.0*x*w - 2.0*y*z, 1.0 - 2.0*x*x - 2.0*z*z)
138      try:
139          yaw = math.asin(2.0*x*y + 2.0*z*w)
140      except:
141          yaw =0
142      return roll,pitch,yaw
143
144
145  def RerangeEulerAngle(angle,deadzone,max1):
```

Fuente**:** Autor

3.4.1 PROCESAMIENTO DE DATOS

Después de la captura de datos, se obtienen dos vectores que alojan las posición actual y la posición anterior guardados en cuaterniones, a partir de estos se obtienen los valores de roll, pitch y yaw, de cada vector orientación, esta posición se actualiza con cada lectura del sensor, posteriormente se calcula el ángulo relativo obtenido en cada eje, es decir, un ángulo entre 180° y -180°, la Figura 14 muestra las funciones desarrolladas para el procesamiento de estos datos, estas funciones son una traducción a lenguaje Python de las funciones usadas en el SDK disponible para *Unity 5* del Myo, que también están presentes en el desarrollo del proyecto *Myo Pilot*. Estas funciones arrojan los ángulos en radianes los cuales serán posteriormente convertidos en comandos para ser enviados.

Figura 14. Funciones para calcular roll, pitch y yaw actuales.

```
192
193  def CalculateRelativeEulerAngles(currentOrientation,referenceOrientation):
194
195      """
196       Calculate all relative angles between the current orientation and the reference orientation
197      """
198      currentRoll,currentPitch,currentYaw = CalculateAnglesm (currentOrientation)
199      referenceRoll, referencePitch, referenceYaw=CalculateAnglesm (referenceOrientation)
200      roll = CalculateRelativeAngle(currentRoll, referenceRoll)
201      pitch = CalculateRelativeAngle(currentPitch, referencePitch)
202      yaw = CalculateRelativeAngle(currentYaw, referenceYaw)
203      return roll,pitch,yaw
204
205  def CalculateRelativeAngle(current, reference ):
206      """
207      Calculate one relative angle between the current angle and the reference angle
208      """
209      ## Because our domain is circular (-PI and +PI are the same) we calculate
210      # the smallest difference and return
211
212      angle = current - reference
213
214      if (angle > math.pi):
215          return angle - 2*math.pi
216
217      if (angle < -math.pi):
218          return angle + 2*math.pi
219      return angle;
220
```

Fuente: Autor

A continuación se establecen valores fijos para la zona de no lectura, lo que permite calibrar sensibilidad en la captura de los ángulos, esto se realiza por medio de dos métodos que se encargar de escalar y posteriormente arrojar los valores en un rango de [-1,1], la Figura 15 muestra la función encargada de realizar esta tarea, que a su vez realiza el llamado a una sub-función mostrada en la Figura 16, los valores pueden modificarse según los requerimientos.

Figura 15. Función para estableces rango y zonas de muerte en los ángulos.

```
165  def RerangeEulerAngles(roll,pitch,yaw):
166      """
167           Rerange the angles to [-1 - +1]
168           If one angle is in the deadzone it is set to 0
169           Angles are cubed for better control
170           Deadzone and max is configurable
171      """
172      # dead Zona
173      rollDeadzone = 0.3   #0.1    #0.25
174      pitchDeadzone = 0.3
175      yawDeadzone = 0.3
176      rollMax = 0.8      #0.2     0.65
177      pitchMax = 0.95    #.35     .8
178      yawMax = 0.8       #       .8
179      roll= RerangeEulerAngle(roll,rollDeadzone,rollMax)
180      pitch=RerangeEulerAngle(pitch, pitchDeadzone, pitchMax)
181      yaw= RerangeEulerAngle(yaw, yawDeadzone, yawMax)
182      return (roll,pitch,yaw)
183
```

Fuente: Autor

Figura 16. Función auxiliar para escalar los ángulos.

```
126  def RerangeEulerAngle(angle,deadzone,max1):
127
128      """
129          Rerange the angles to [-1 - +1]
130          If one angle is in the deadzone it is set to 0
131          Angles are cubed for better control
132          Deadzone and max is configurable
133              max1 ---/
134                     /:
135                    / :
136                   /  :
137         deadzone-/   :
138                  :   :
139                  0   1
140      """
141      sign=math.copysign(1, angle)
142      value = abs(angle)
143
144      if (value < deadzone):
145          angle = 0
146          return angle
147      else:
148              # current range of value: [deadzone - inifinite]
149          value = float(min(value, max1))
150              #current range of value: [deadzone - max1]
151          value -= deadzone
152              #current range of value: [0 - (max1 - deadzone)]
153          value /= (max1 - deadzone);
154              #current value [0-1]
155
156      # m= max1 - deadzone        # (y2-y1)/(x2-x1)
157      # value=m*value+deadzone    # Y=m(X-x1)+y1
158
159      angle = float(sign*value)
160      return angle
161
```

Fuente: Autor

Estas funciones siguen los algoritmos desarrollados en otros programas, y han sido programados en Python con diferentes métodos. Los ángulos obtenidos pasan posteriormente por un filtro adicional, que solo permite el paso del ángulo con mayor variación, esta función no está incluida en otros programas del mismo propósito. El ángulo final se multiplica por 100 para ser manejado el siguiente paso, en la Figura 17 se muestran las funciones descritas.

Para el verificar el código se desarrolla una función que llama las rutinas para el procesamiento de datos, este código se puede apreciar en la Figura 18.

Figura 17. Funciones de filtrado, y escalado de ángulos.

```
209   def Filter_values(val1,val2,val3):
210       if (val1==0 and val2==0 and val3==0):
211           return 0,0,0
212       sg1,sg2,sg3=math.copysign(1, val1), math.copysign(1, val2),math.copysign(1, val3)
213       val1,val2,val3=abs(val1),abs(val2),abs(val3)
214       if(val1>val2 and val1>val3):
215           val2=0
216           val3=0
217       elif(val2>val3):
218           if(val3!=0):
219               val3=0
220           if(val1!=0):
221               val1=0
222       else:
223           if(val1!=0):
224               val1=0
225           if(val2!=0):
226               val2=0
227       return sg1*val1,sg2*val2,sg3*val3
228
229
230   def conver_grade(angle):
231       angle= (angle*100)
232
233       return angle
```

Fuente: Autor

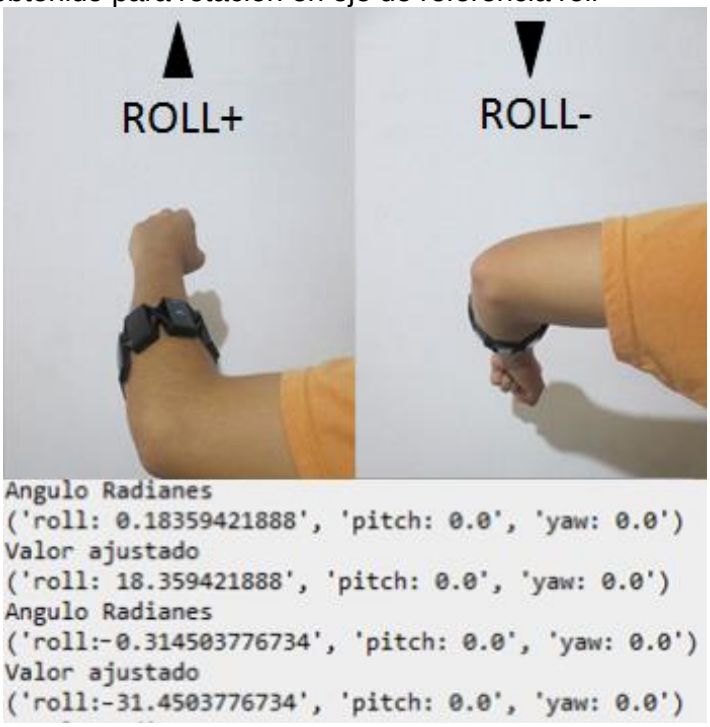
Figura 18. Función para la ejecución y procesamientos de datos del Myo

```
def  proccesOutRobotEdw(app):
    """
    configure set points
    """
    myoRoll,myoPitch,myoYaw=CalculateRelativeEulerAngles (app.currentOrientation,app.referenceOrientation)
    myoRoll_, myoPitch_, myoYaw_=RerangeEulerAngles (myoRoll, myoPitch, myoYaw)
    myoRoll1, myoPitch1, myoYaw1=Filter_values(myoRoll_, myoPitch_, myoYaw_)

    if (app.pose==libmyo.Pose.fist):
        print ("Angulo Final")
        print ("roll: "+  str(round(myoRoll1,5)),"pitch: "+ str(round(myoPitch1,5)),"yaw: "+ str(round(myoYaw1,5)))
        print ("Valor Ajustado")
        print ("roll: "+  str(conver_grade(myoRoll1)),"pitch: "+ str(conver_grade(myoPitch1)),"yaw: "+ str(conver_grade(myoYaw1)))

def main():
    print("Connecting to Myo ... Use CTRL^C to exit.")
    try:
        global hub
        hub = libmyo.Hub()
    except MemoryError:
        print("Myo Hub could not be created. Make sure Myo Connect is running.")
        return
    myApp=Listener()
    hub.set_locking_policy(libmyo.LockingPolicy.none)
    hub.run(1000, myApp)
    # Listen to keyboard interrupts and stop the hub in that case.
    try:
        while hub.running:
            time.sleep(0.25)
            proccesOutRobotEdw (myApp)
    except KeyboardInterrupt:
        print("\nQuitting ...")
    finally:
        print("Shutting down hub...")
        hub.shutdown()
```

Fuente: Autor

Como resultado obtenemos un ángulo entre -100 y 100, dependiendo el sentido que giremos el brazo, en la Figura 19 se muestra el valor obtenido cuando se gira el brazo manteniendo el gesto puño (Fist); este giro alrededor del el eje X y se denota como roll.

Figura 19**.** Valor obtenido para rotación en eje de referencia roll



Fuente**:** Autor

3.5 PRUEBA FUNCIONAL DRONE

Antes de proceder a la integración se realiza una prueba de funcionamiento para familiarizarse con el drone y sus características a tener en cuenta.

La prueba se realizó una mañana con una temperatura de 30º y sin viento en el municipio de Monterrey Casanare, el cual se encuentra situado a 481 m sobre el nivel del mar. Se probó el drone con una batería de 11.4W, alcanzando una altura máxima recomendada de 100m, con un tiempo de subida de 65s, tiempo establecimiento con reposo de 7 min y tiempo de bajada de 67s, aterrizando con una batería restante del 10%.

En otra prueba con la misma batería se elevó el drone a la máxima altitud con un tiempo de subida de 65s, tiempo de establecimiento y sobrevuelo en círculos de 6.5 min, tiempo de bajada 64s, aterrizando con un porcentaje de batería de 9%.

Posteriormente se elevó el drone en un lugar amplio con ráfagas de viento, comprobando que el drone al ser liviano es muy susceptible a movimientos a causa de ráfagas de viento.

De acuerdo a las pruebas se selecciona una altura máxima de 80m y un tiempo total de vuelo de 8min para asegurar la estabilidad, la seguridad y garantizar la duración de la batería, la cual depende de la altitud de vuelo, el terreno por el cual sobrevuela y las maniobras realizadas.

Figura 20**.** Vista desde el drone a 100m, Monterrey Casanare

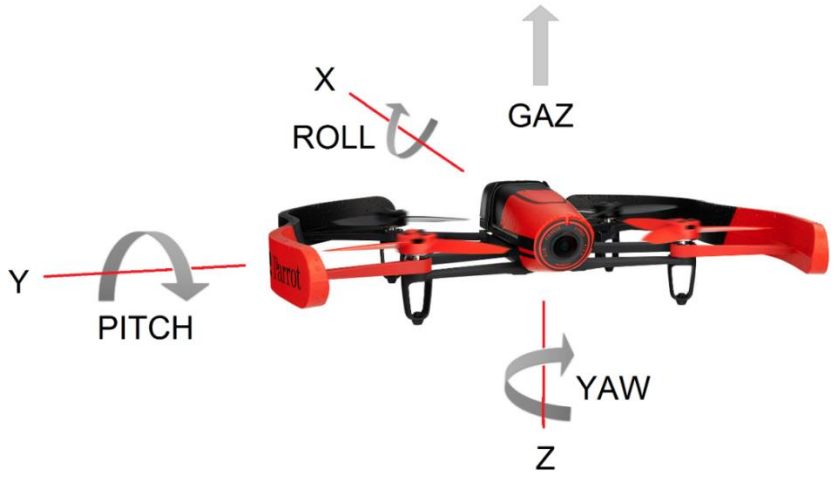
Fuente**:** Autor

3.6 INTEGRACIÓN PYTHON - BEEBOP DRONE

Para la integración del drone se usa la librería en desarrollo por el grupo de robótica de Republica Checa [23], la cual es una librearía experimental que aún está en desarrollo. La librería cuenta con los métodos suficientes para volar el drone y se encuentra disponible en [24], distribuida bajo licencia MIT y puede instalarse manualmente.

El drone utiliza como referencia cuatro variables principales: pitch, roll, yaw y gaz. La Figura 21 muestra la relación de las variables con los ejes de referencias.

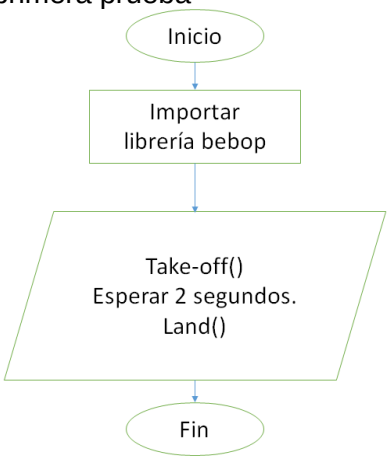
Figura 21. Relación de rotación Bebop drone



Fuente: Thalmic Labs

En la primera prueba se inicia la comunicación con el drone, se eleva por 2 segundos y se aterriza para comprobar él envió de comandos. Por defecto, el drone siempre despega y alza vuelo a un metro por encima de su zona de despegue.

Figura 22**.** Diagrama de flujo primera prueba



Fuente**:** Autor

Posteriormente, se procede a modificar el tiempo de duración, es decir se despegara y durara 10 segundos en el aire, para después aterrizar.

Al ejecutar esta prueba se pudo apreciar que el drone no mantiene su posición espacial; es decir, si una ráfaga de viento mueve el drone este se moverá con la ráfaga, manteniendo la altura y la estabilidad de su vuelo, pero seguirá la trayectoria del viento. Esto quiere decir que los métodos de vuelo no estas optimizados para mantener la estabilidad espacial.

En el siguiente paso, la plataforma enviara los comandos necesarios para volar, moverse en los ejes y aterrizar, para lograr esto se disponen 10 botones en la plataforma para las diez instrucciones las cuales se observan en la Tabla 4:

Tabla 4**.** Instrucciones

Instrucción	Descripción
Despegue-Aterrizaje	Take off -Land
Aterrizaje suave	Land++
Atrás	Pitch-
Adelante	Pitch+
Izquierda	Roll-
Derecha	Roll+
Giro en sentido horario	Yaw+
Giro en sentido anti-horario	Yaw-
Subir	Gaz+
Bajar	Gaz-

Fuente**:** Autor

El drone está configurado para recibir valores en cada instrucción; estos valores varían en el Intervalo [-1,1], y determinan la velocidad de movimiento cuando se pulse los botones. La velocidad máxima a su vez está limitada por la configuración predeterminada.

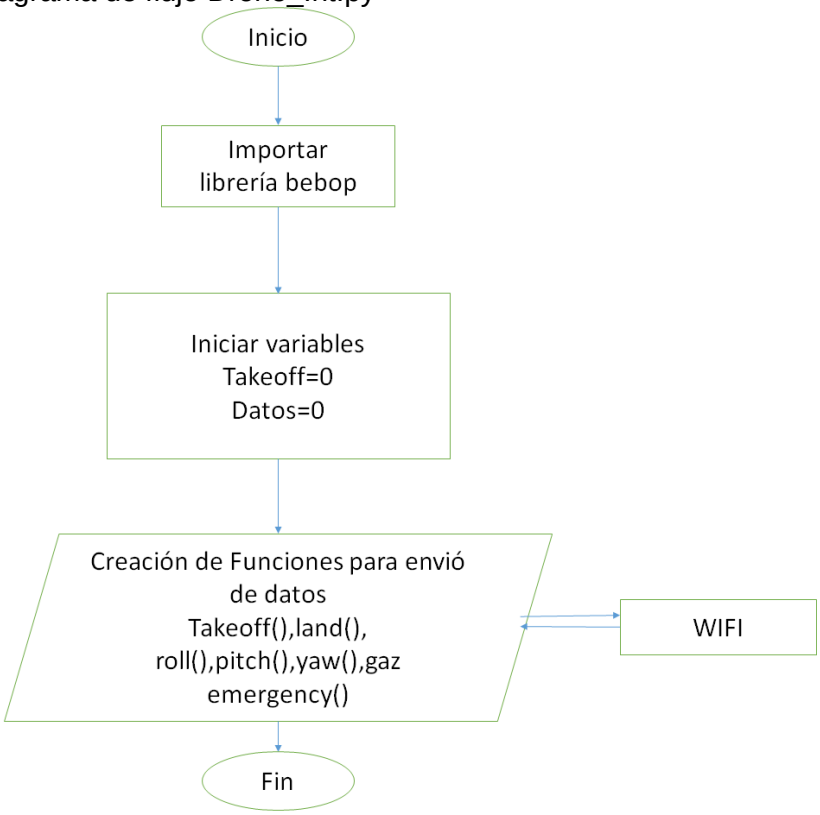
Figura 23**.** Botones de prueba

Fuente**:** Autor

En la Figura 24 se aprecia el diagrama de flujo que describe el comportamiento de este script. Al realizar la prueba, el drone responde adecuadamente a los movimientos.

El código necesario para gestionar las órdenes enviadas al drone, se denomina "*Drone_Int.py*", el cual contiene todos los métodos para enviar las órdenes al drone, incluido comandos de despegue y aterrizaje.

Figura 24. Diagrama de flujo Drone_Int.py



Fuente**:** Autor

Para el desarrollo de este código se estudió, las funciones requeridas para enviar comandos al drone. Posteriormente se crean métodos para cada movimiento que serán llamados por un botón determinado, aplicando las restricciones necesarias para evitar errores en código y posibles daños físicos en el drone, al tratarse de una librería en desarrollo no se cuenta con manuales y para su funcionamiento se requiere copiar toda la carpeta que contiene los métodos necesarios para interactuar con el drone, posteriormente esta carpeta se importa para su uso, en la Figura 25 se muestra la cabecera principal de este script, y la creación de la clase principal que contiene las funciones para cada comando que será envido al drone.

Figura 25. Cabecera del programa Drone_int.py

```
35  from core.bebop import *
36  import tkFont
37
38
39  class Control():
40      """
41      Class for control proccess bebop
42      """
43      def __init__(self):
44          self.takeoff_land=False
45          self.active=False
46          self.d_time=0.3 #0.25
47          self.info_drone="No connected"
48          #self.dron_init()
49
50      def dron_init(self):
51          ##drone
52          self.drone=Bebop()
53          self.speed=20
54          self.active=not(self.active)
55          self.info_drone="Ready to fly"
```

Fuente**:** Autor

Se pudo observar que el acceso al drone es sencillo, así como también él envió de comandos, a diferencia del script utilizado en Myo_Input.py, para este script no se tuvo ningún modelo ni algoritmo, ya que hasta el momento la información disponible es limitada.

En la Figura 26 se puede observar la forma como se acede a los comandos, y las restricciones que se hicieron. Se crearon métodos para los 10 botones iniciales, adicionando un método de aterrizaje alternativo (Land++), ya que el comando natural de aterrizaje no ofrece un descenso suave, también se adiciono el comando de parada de emergencia que detiene instantáneamente los rotores, los comandos enviados son mantenidos por el drone por esta razón al enviar una instrucción de debe mantener durante un determinado tiempo y posteriormente enviar el comando de cero movimiento.

Figura 26. Funciones para envió de comando al drone

```
def roll_i(self,speed):
    if(self.takeoff_land):
        self.info_drone="Right"
        self.drone.update( cmd=movePCMDCmd( True, speed , 0, 0, 0 ) )
        time.sleep(self.d_time)
        self.drone.hover()

        print ("roll left")
    else:
        tkMessageBox.showinfo(title="Alert",message="Takeoff")

def roll_d(self,speed):
    if(self.takeoff_land):
        self.info_drone="Left"
        self.drone.update( cmd=movePCMDCmd( True, -speed , 0, 0, 0 ) )
        time.sleep(self.d_time)
        self.drone.hover()
        print ("roll right")
    else:
        tkMessageBox.showinfo(title="Alert",message="Takeoff")

def ateb(self):
    if (self.takeoff_land):
        print ("Landing....")
        self.info_drone="Landing ++"
        self.drone.flyToAltitude(.7, timeout=13)
        time.sleep(0.3)
        self.drone.land()
        self.takeoff_land=False
    else:
        pass
        #tkMessageBox.showinfo(title="Alert",message="Takeoff")

def Emergency(self):

    if (self.takeoff_land):
        self.drone.emergency()
        self.info_drone="Stop Emergency"
        tkMessageBox.showinfo(title="Alert",message="Stop Emergency")
        time.sleep(3)
```

Fuente: Autor

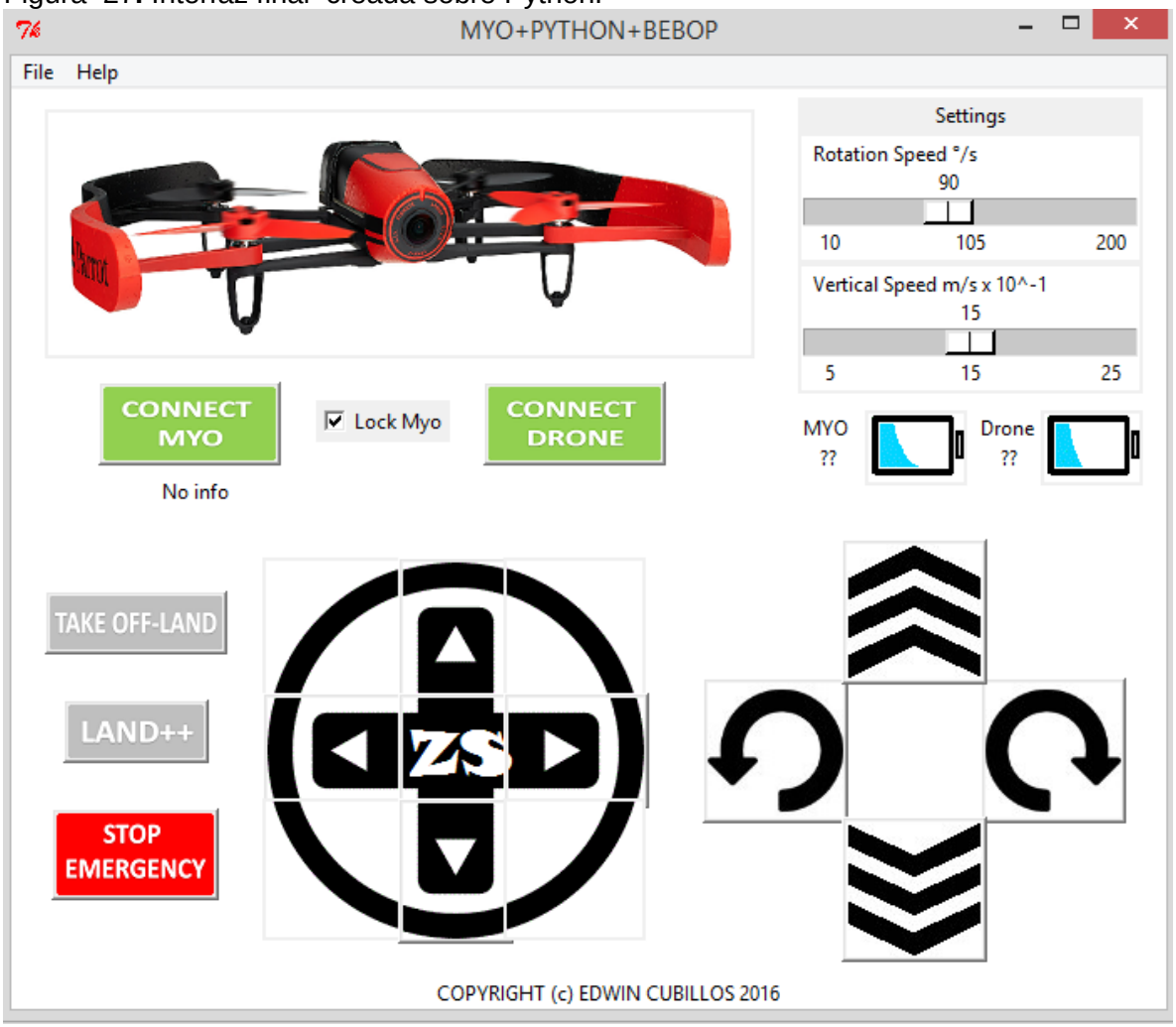
3.7 INTEGRACIÓN FINAL

Para lograr la integración de todos los dispositivos se crea un script principal que recoge los códigos antes desarrollados y se adicionan nuevos elementos a la interfaz final de usuario. También se tienen en cuenta protecciones en el código principalmente para salvaguardar el drone de posibles colisiones.

4. DESARROLLO DE LA PLATAFORMA DE INTEGRACIÓN DIGITAL

Para la integración final se programa una interfaz de usuario, conocida como GUI en programación. Para esto se utilizó la librería "*Tkinter*", que viene por defecto con Python, la cual permite crear ventanas, botones, menús, acciones, y demás elementos básicos en una interfaz.

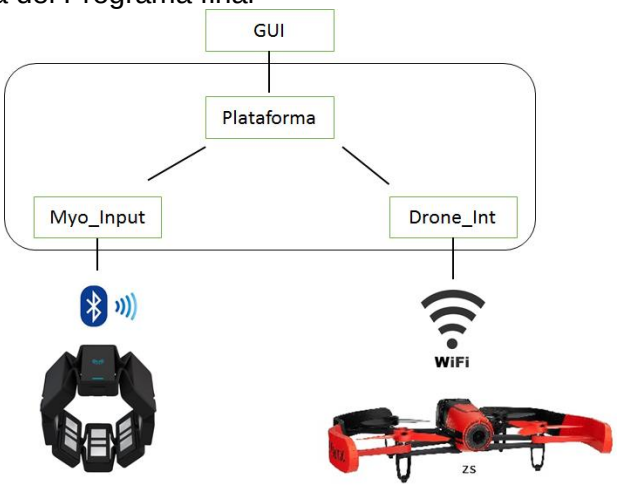
Figura 27. Interfaz final creada sobre Python.



Fuente: Autor

La interfaz plataforma presentada, contiene los métodos anteriormente desarrollados para cada módulo, realizando la conexión y proceso de los mismos, en la Figura 28 se aprecia la estructura de la plataforma, adicionalmente la interfaz contiene métodos para captura de datos, y muestra de información básica.

Figura 28. Estructura del Programa final



Fuente: Autor

Para el desarrollo de esta Interfaz de usuario se crearon todos los botones y elementos gráficos, que se pueden apreciar en la Figura 27, para posteriormente ser cargados, esto se logra con líneas básicas de código para carga de imágenes en formatos compatibles con la librería, la Figura 29 muestra un fragmento del código capaz de cargar elementos gráficos y ubicarlos en la interfaz.

Figura 29. Carga de imágenes para la interfaz gráfica

```
##image
im_bat=PhotoImage(file="Bib_ima/bat1_57x44.gif")
im_bat2=PhotoImage(file="Bib_ima/bat_57x44.gif")
ima_bat=Label(raiz, image=im_bat, anchor="center",padx=2)
ima_bat2=Label(raiz, image=im_bat2, anchor="center",padx=2)

##

y5_inicial=200
txtleveldrone.place(x=x5_inicial,y=y5_inicial-8)
txtlevelmyo.place(x=x5_inicial+106,y=y5_inicial-8)
ima_bat.place(x=x5_inicial+40,y=y5_inicial-8)
ima_bat2.place(x=x5_inicial+102+43,y=y5_inicial-8)
level_myo.place(x=x5_inicial-20,y=y5_inicial-8+18)
level_drone.place(x=x5_inicial-20+108,y=y5_inicial-8+18)

#imagen drone
raiz.title("MYO+PYTHON+BEBOP")
raiz.geometry("690x550+300+70")

raiz.configure(background='white')

im_drone3 =PhotoImage(file="Bib_ima/bebop_420x145.ppm")
im_drone = Label(raiz, image=im_drone3, anchor="center",padx=2)
```

Fuente: Autor

4.1 FUNCIONAMIENTO

Para realizar las conexiones con los dispositivos y controlar el drone adecuadamente se debe tener en cuenta los siguientes parámetros que se muestran en la Tabla 5.

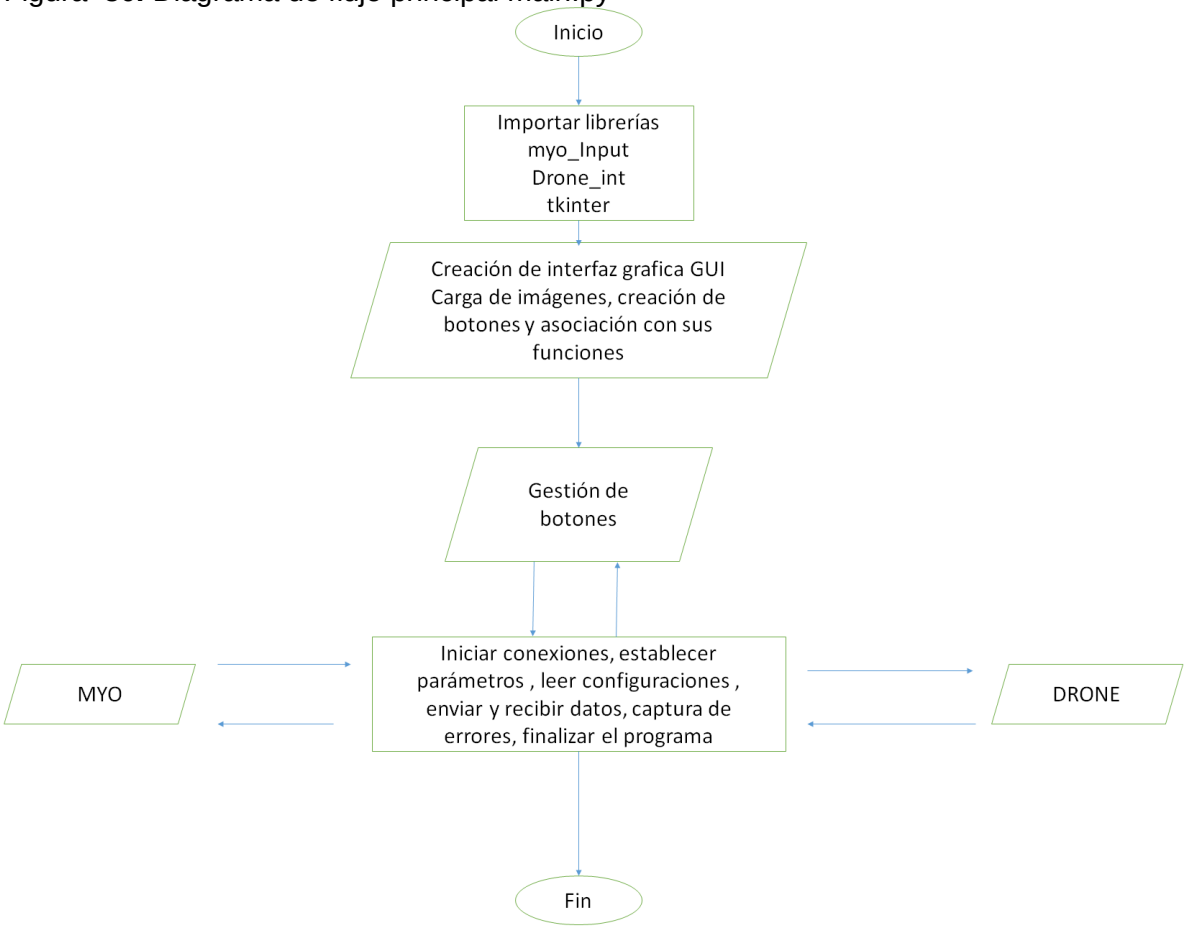
Tabla 5. Parámetros de funcionamiento

Parámetro	Función
CONNECT DRONE	Inicia la conexión con el drone
CONNECT MYO	Inicia la conexión con el brazalete MYO
LOCK MYO	Bloquea en envió de datos provenientes del brazalete Myo
TAKE OFF-LAND	Despegue y aterrizaje el drone
LAND ++”	Aterriza suave el drone
STOP EMERGENCY	Botón de parada de emergencia para los motores
SETTINGS	Configura la velocidad vertical y de rotación
JOYSTICK 1	Controla las direcciones adelante, atrás, izquierda y derecha
JOYSTICK 2	Controla el ascenso, descenso, rotación en sentido horario y anti-horario
Niveles de baterías	Muestra el porcentaje actual de los dispositivos conectados

Fuente: Autor

El código que llama a los métodos encargados de manejar al MYO y al drone se denomina “*main.py”*. Este código contiene la importación y gestión de métodos, la interfaz de usuario (GUI) y establece la conexión entre las imágenes y/o botones con los métodos correspondientes que serán enviados como comandos al drone. La figura 30 se muestra el diagrama de flujo principal.

Figura 30**.** Diagrama de flujo principal main.py



Fuente**:** Autor

Los controles y gestos necesarios para manejar el drone con el brazalete se observan en la Figura 31. Estos gestos son convertidos en comandos por medio de los métodos diseñados en el código *Myo_Input.py*, y la plataforma se encarga de realizar la asociación de estos comandos con los métodos descritos en el código *Drone_Int.py,* con los cuales se logra el control drone.

Figura 31. Gestos y movimientos para volar el drone

Fuente**:** Autor

4.2 PROTECCIONES DE LA PLATAFORMA Y ALERTAS

La plataforma cuenta con diversas protecciones y mensajes de alerta que informan al usuario sobre posibles problemas y fallos.

- **Protección contra fallo en el software**: Está protección se dispara ante cualquier error en el código principal o en las librerías. Al dispararse se detiene el proceso en el brazalete Myo y si el drone está volando lo aterriza suavemente, para finalmente cerrar la aplicación.

Figura 32**.** Líneas de código de protección

```
#for any error land and exit
except :
    print ("error")
finally:
    #before to out
    app_drone.ateb()
    try:
        hub.shutdown()
    except:
        print ("Cannot do")
    exit()
```

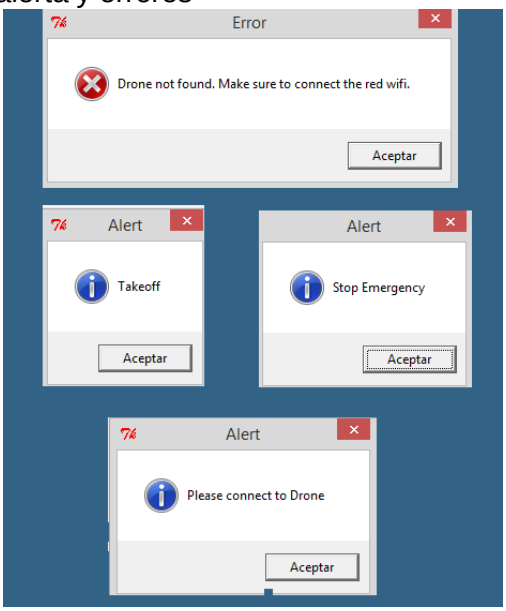
Fuente**:** Autor

- **Protección batería baja**: El drone ya viene protegido contra batería baja, cuando se detecta un porcentaje igual o menor al 5% de carga; si el drone está volando se aterriza automáticamente y si está en el suelo no permite alzar el vuelo. Adicionalmente, la plataforma despliega un mensaje de alerta cuando la batería alcance el 10% de su carga y envía una vibración al brazalete para alertar al usuario.

  El brazalete Myo tiene una batería de larga duración (24 horas) y un led indicador; cuando el led se torna naranja la batería es baja y en la plataforma se informa al usuario cuando la carga este por debajo del 5%.

  Mensajes de alerta: La plataforma cuenta con mensajes desplegables de alerta, errores, peligro e información. Por ejemplo, un mensaje de alerta aparece cuando el usuario desea despegar el drone pero no ha realizado la conexión o cuando no se puede establecer conexión con el drone o el Myo. En la Figura 33 se observan algunos mensajes que despliega la plataforma.

Figura 33. Mensajes de alerta y errores



Fuente: Autor

- **Protección contra fallas en comunicación**: Se crearon los botones de aterrizaje y parada de emergencia en la plataforma para proteger al drone de posibles fallas que se presenten como lo son una mala lectura del Myo o su desincronización.
- **Protección de rotores del drone**: El drone puede detener sus rotores por seguridad, por ejemplo cuando sus hélices chocan cuando un objeto por precaución se genera una parada de emergencia.

4.3 LICENCIA

Se escoge la licencia MIT, por sus características, compatibilidad y por el hecho de ser una licencia de uso público, tendrá más aceptación y retroalimentaciones. Se debe resaltar que no necesariamente un código derivado o que use código con licencia MIT deberá llevarla.

Figura 34. Licencia escogida

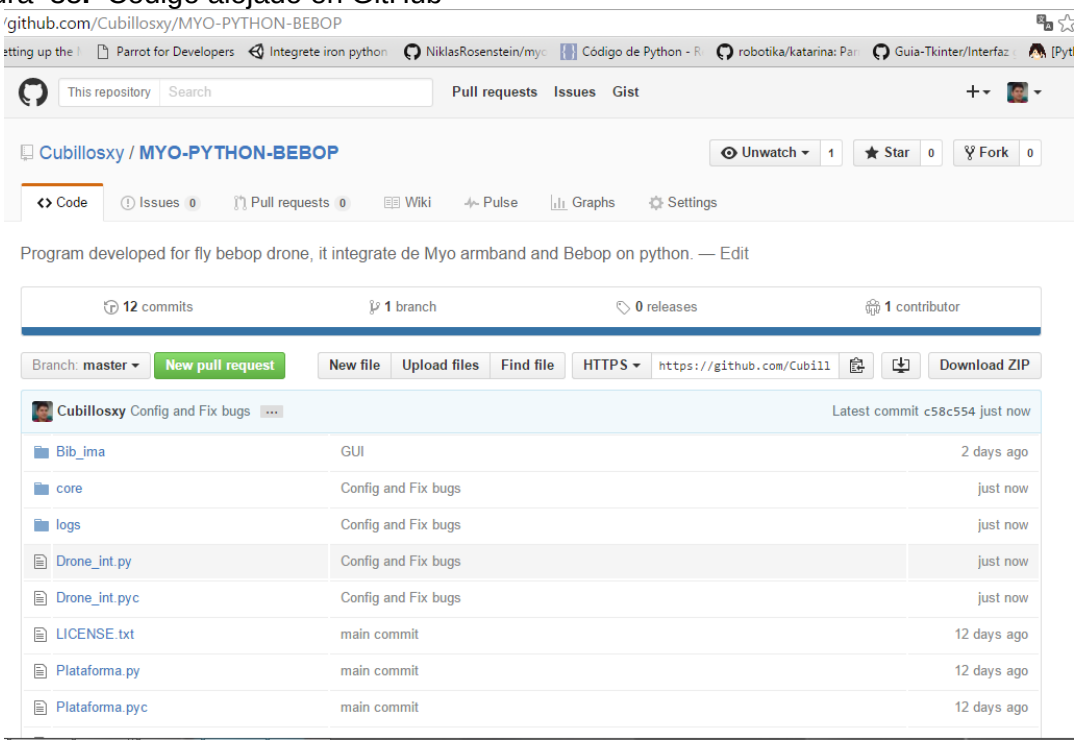
```
Copyright (c) 2016 Edwin Cubillos https://github.com/Cubillosxy

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Fuente: Autor

4.4 VERSIÓN MEJORADA

Dado que el programa puede corresponder a una continuación en el desarrollo de software, esta versión se pone a disposición de la comunidad de desarrolladores para ser mejorada. Por tanto es una versión libre, distribuida bajo licencia MIT y disponible en GitHub [25].

Figura 35. Código alojado en GitHub



Fuente: Autor

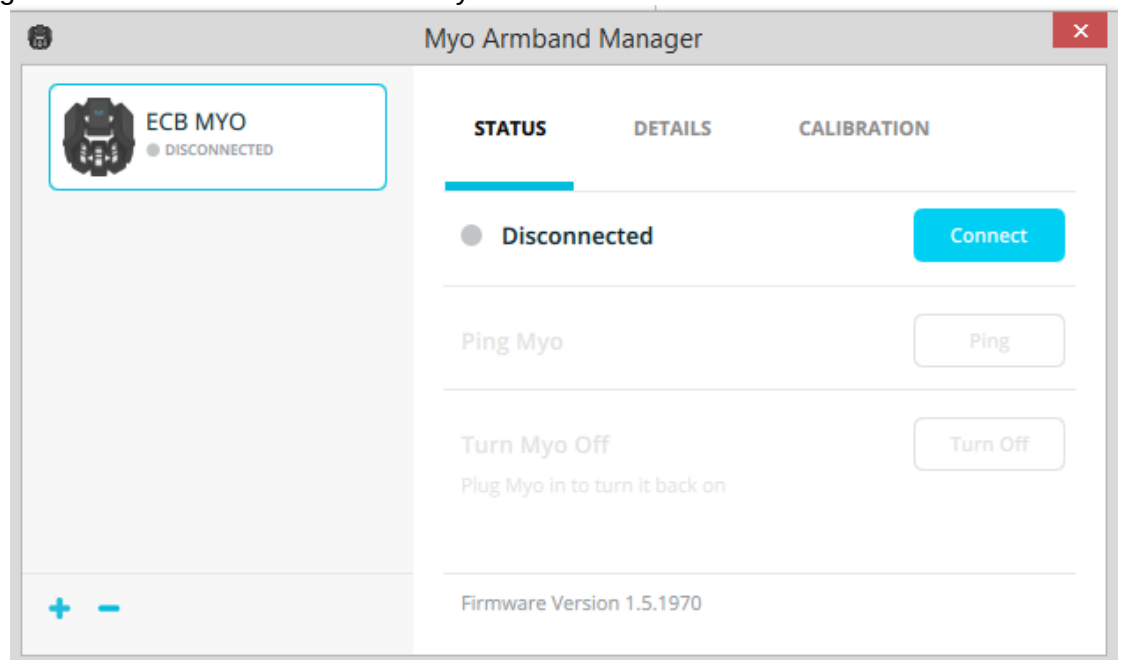
5. PRUEBAS FUNCIONALES

Para realizar las pruebas se debe seguir el protocolo descrito a continuación en un lugar despejado, tomando en cuenta la regulación colombiana para el vuelo de un drone con fin deportivo [26]. También se recomienda tener experiencia en el manejo del Myo, el drone y sus respectivos accesorios.

1. Realizar la conexión a la red Wifi del drone desde el computador. Se debe asegurar que no existan otros dispositivos conectados.

2. Conectar el dispositivo USB del MYO al PC e iniciar la conexión “Myo-Connect”.

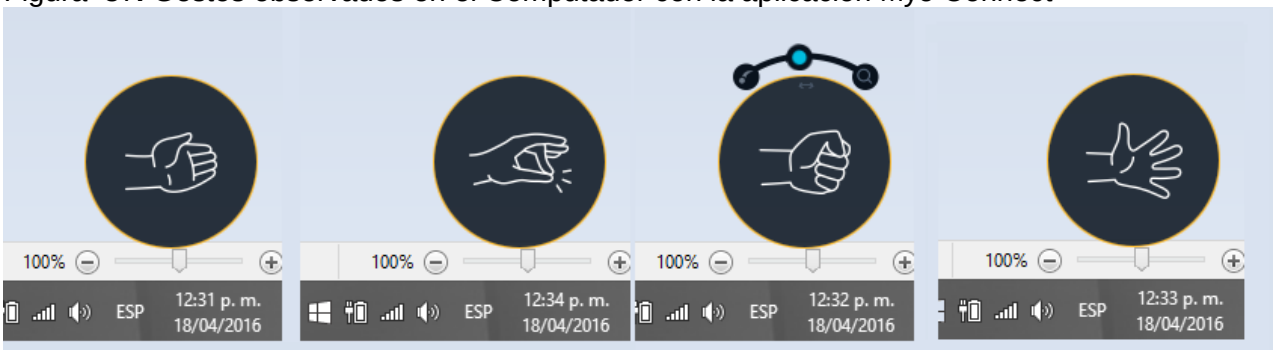
Figura 36. Conexión del brazalete Myo con el PC



Fuente: Autor

Después de conectarse al sensor Myo, el usuario puede verificar que el brazalete este sincronizado y leyendo correctamente los gestos, con la ejecución de un test en el que se realizan gestos con el brazo y se observa la respuesta en el computador a través de un punto en la pantalla que se mueve según las rotaciones en Roll, Pitch y Yaw. Para realizar este test el usuario deberá activar la opción “*Presentation Mode*”. La Figura 37 muestra las imágenes que proporciona esta prueba.

Figura 37**.** Gestos observados en el Computador con la aplicación Myo Connect



Fuente**:** Autor

Si se presentan problemas con el reconocimiento de los gestos, es recomendable realizar una calibración del Myo, con el cual se logra una correcta lectura de los gestos a excepción del gesto “Double Tap” , el cual no tiene opción de calibrarse.

3. Iniciar el programa desarrollado MYO+PYTHON+BEBOP.

La aplicación se corrió en Windows 8. Si desea trabajar en otro sistema operativo se debe tener previamente instalado Python en su versión 2.X con las librerías correspondientes. Seguido a esto, se puede utilizar cualquier editor para abrir el archivo principal “*main.py*”, el cual se corre por medio de la consola de Windows escribiendo -*python main.py*- **.**

4. Pulsar los botones para conectar el drone y el brazalete.

La plataforma desplegara un mensaje para indicar que la acción se completó satisfactoriamente o saltara un mensaje de error indicando una falla.

El label debajo del botón “CONNECT MYO” muestra información actual de los comandos leídos por el brazalete.

5. Revisar que los dos dispositivos cuenten con la batería suficiente para ser usados sin inconvenientes.

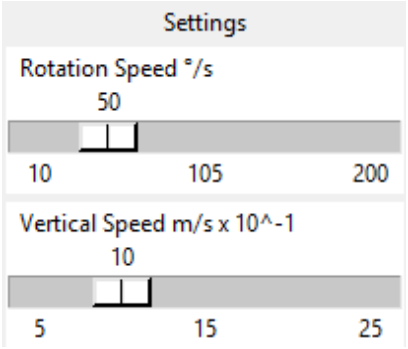
6. Realizar prueba con los botones

Sin desmarcar la casilla “*Lock Myo*”, realizar una prueba rápida con los botones primero despegando con el botón “Take-off Land”, y presionando los botones de control para validar que efectivamente el drone está recibiendo órdenes de la plataforma, así como también para descartar alguna falla técnica en el drone. Puede probarse a criterio del usuario todos los botones y movimientos respectivamente, antes de aterrizar el drone.

Para realizar esta prueba siga los siguientes pasos:

- Configure la velocidad del drone: Esta configuración solo tiene efecto si se vuela con los botones de la interfaz. Si se vuela con el brazalete Myo la velocidad es proporcional a la velocidad con la que se mueva el brazo.

Figura 38. Configuración de velocidad en rotores del drone



Fuente: Autor

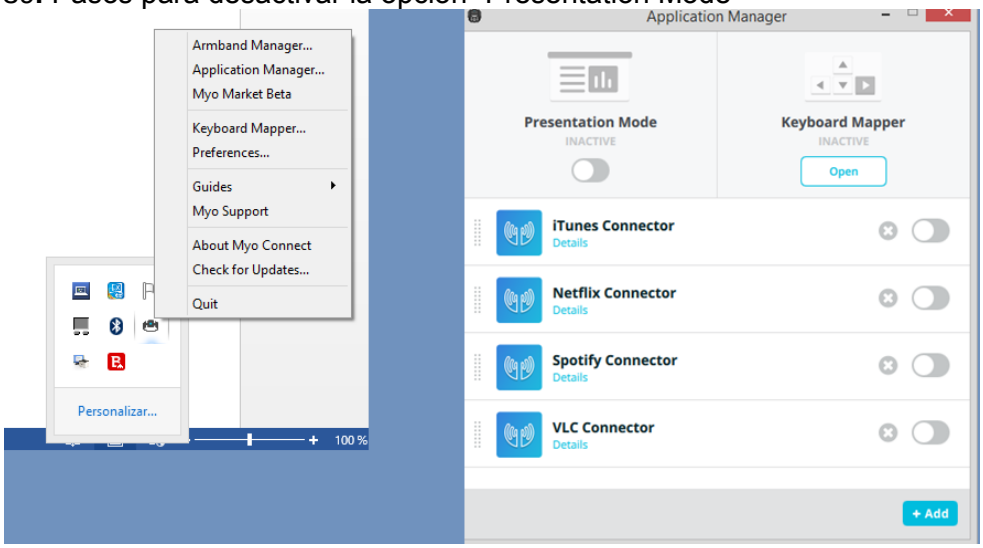
- Presione el botón “Take off- land”.
- Mueva el drone con los botones

Si presenta algún problema aterrice el drone con cualquiera de los tres botones, recordando que el mejor aterrizaje se obtiene con el botón “Land++”.

7. Despegar el drone y controlarlo adecuadamente con el Myo

Se recomienda desactivar la opción “Presentation Mode”, para desactivarla abra la ventana “Applicaction Manager” y a continuación desactive la casilla de esta opción.

Figura 39. Pasos para desactivar la opción “Presentation Mode”

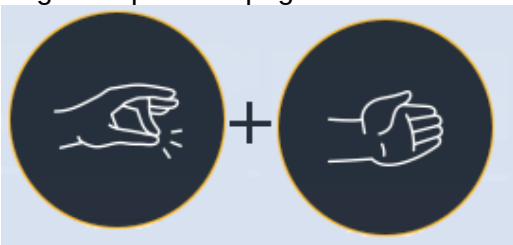


Fuente: Autor

La desactivación es opcional puesto que no representa ningún riesgo con la plataforma ni crea ningún conflicto de comunicación, pero si puede alterar la pantalla (efecto lupa), y puede ocasionar problemas.

Para despegar el drone el usuario debe realizar la combinación de gestos descritos en la Figura 40, o puede presionar el botón de despegue. Previamente debe desmarcar la casilla “Lock Myo” para poder enviar datos al drone.

Figura 40**.** Combinación de gestos para despegar el drone.



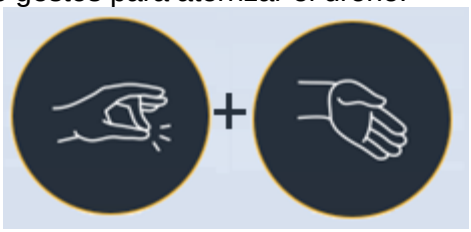
Fuente**:** Autor

Para el control del drone el usuario debe realizar los gestos y movimientos descritos en la Figura 31, en caso de desorientación del drone se puede utilizar como respaldo el botón de aterrizaje de emergencia. Es necesario recordar a los usuarios novatos que los vuelos deben ser en espacios abiertos debido a que se requiere práctica para aprender los gestos y movimientos adecuadamente y no confundirse entre los mismos. Puede verse un video de prueba funcional del equipo en los anexos de este documento.

8. Aterrizar el drone

La Figura 41 contiene la combinación de gestos que el usuario debe seguir para aterrizar el drone.

Figura 41. Combinación de gestos para aterrizar el drone.



Fuente: Autor

Si se presenta algún problema, puede aterrizar el drone con cualquiera de los tres botones anteriormente descritos, “Take off- Land”, “Land++”, “Stop Emergency”.

6. RESULTADOS Y TRABAJOS FUTUROS

6.1 RESULTADOS

- Se realizó la integración de dos dispositivos de fabricación reciente, el brazalete Myo y el Parrot Bebop drone, lo permite realizar un aporte a la comunidad internacional de desarrollo y contribuir al avance tecnológico.
- El desarrollo de este proyecto implica unas habilidades específicas que se adquieren manejando distintos software y lenguajes de programación. En este caso antes de llegar a un diseño final, se pasó por diseños con UNITY 5, que es un software de desarrollo de juegos y muy arraigado al diseño con sensores como el Myo, pero con pocas herramientas para drones como el Bebop.
- Se fortaleció la interacción con comunidades de desarrollo y foros como Stack Overflow [27], Parrot Developer [28], Unity Community [29], GitHub [30], Myo Developers [31] lo que permite a futuro realizar trabajos de cooperación internacional desarrollando software y aplicaciones.

6.2 TRABAJOS FUTUROS

Con la presentación de este trabajo se deja abierto el camino a continuar mejorando la plataforma o a darle una aplicación. A continuación mencionamos algunas adiciones futuras.

- Adicionar más entradas para controlar el drone, como una entrada mapeando las teclas del PC para controlar el drone o adicionar con la ayuda de librerías la entrada de datos con un control de Xbox como lo hace el MyoPilot.
- Mejorar la interfaz de la plataforma agregándole más variables del drone, como la cámara la posición GPS, la visualización de datos en tiempo real.
- Se puede combinar el trabajo y agregar la vista de la cámara del drone por medio gafas 3D para una vista en primera persona, haciendo aún más interactiva la aplicación.
- Generar un programa mucho más robusto, capaz de proveer la posición del drone gracias al GPS embebido, y generar un mapa tridimensional de su trayectoria.
- Agregar otros dispositivos de para controlar el drone, por ejemplo control por mandos de voz, o con la ayuda de un sensor Myo adicional.
- Adicionar con la ayuda de un GPS adicional o reconocimiento de imágenes la opción de “Sigueme” en inglés “Follow me”, que aumentaría la versatilidad del drone y la plataforma.
- Basándose en los mismos algoritmos se puede Realizar el mismo proceso para integrar y controlar el nuevo drone de Parrot, el Bebop 2.

7. CONCLUSIONES

- Durante el desarrollo del proyecto la información disponible y fuentes de consulta fueron limitadas, estas dificultades sobresalen al usar herramientas de última generación ya que todas estas están sujetas a nuevas mejoras y los usuarios son los elegidos para probar el dispositivo y reportar errores.

- La integración entre el MYO y el drone Bebop responde a una suma de procesos y pruebas necesarias para poder llegar un producto final. El eje de desarrollo fue el drone y el determinante al momento de programar en *Python*, ya que este dispositivo cuenta con menor investigación respecto al MYO.

- El gesto " doublé tap", es el único gesto que no se puede calibrar al usar el programa *Myo connect*, esto ocasionando errores en el desarrollo de software con esta aplicación, para este caso la opción de despegue y aterrizaje son de difícil lectura. Actualmente *Thalmic Labs* trabaja sobre este problema.

- Se comprobó que al tratarse de dispositivos de última generación el campo de investigación está presente sobre todo en países desarrollados, y por ejemplo la información es español es totalmente reducida, requiriendo una habilidad esencial en la actual generación como lo es la buena compresión del idioma inglés.

8. LOGROS

- Presentar un software desarrollado por terceros capaz de manejar el drone Bebop 1

- Presentar un software desarrollado por terceros capaz en integrar vía Python el MYO y el drone Bebop 1.

- Presentar un software multiplataforma desarrollado por terceros para PC capaz de controlar el drone Bebop 1 con el brazalete Myo

- Comprobar el correcto funcionamiento y aplicabilidad de librerías desarrolladas por terceros y disponibles para los desarrolladores futuros.

BIBLIOGRAFÍA

[1] T. Labs. *Thalmic MYO*. Available: https://www.thalmic.com/myo/

[2] R. Azad, B. Azad, N. Belhaj, and S. Jamali, "Real-time human-computer interaction based on face and hand gesture recognition," 4, International Journal in Foundations of Computer Science & Technology (IJFCST), No.4, July 2014.

[3] M. Nowicki, O. Pilarczyk, J. Wasikowski, and K. Zjawin., "GESTURE RECOGNITION LIBRARY FOR LEAP MOTION CONTROLLER," Thesis,Computing Science, Poznan University of Technology, Poznan, 2014.

[4] T. Forbes, "MOUSE HCI THROUGH COMBINED EMG AND IMUBA," Thesis, Master of science in electrical engineering, University of rhode island, 2013.

[5] S. Godha and M. E. Cannon, "Integration of DGPS with a Low Cost MEMS - Based Inertial Measurement Unit (IMU) for Land Vehicle Navigation Application ", Position Location and Navigation Group (PLAN), Department of Geomatics Engineering, University of Calgary, Calgary, Alberta, Canada, 2005.

[6] O. Kainz and F. Jakab, "Approach to Hand Tracking and Gesture Recognition Based on Depth-Sensing Cameras and EMG Monitoring," Acta Informatica Pragensia, Technical University of Košice, Košice Letná 9, 042 00 Košice, Slovakia, 2014.

[7] M. Sathiyanarayanan and T. Mulling, "Map Navigation Using Hand Gesture Recognition: A Case Study Using MYO Connector on Apple Maps," *Procedia Computer Science,* vol. 58, pp. 50-57, // 2015.

[8] M. Sathiyanarayanan and S. Rajan, "MYO Armband for physiotherapy healthcare: A case study using gesture recognition application," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, 2016, pp. 1-6.

[9] R. B. Guerrero and F. X. Coronel, "Sistema de navegación de un cuadricóptero guiado por el movimiento de las manos para operaciones de busqueda y rescate," Monografia, Ingenieria electronica, Universidad politécnica salesiana, Cuenca-Ecuador, 2013.

[10] A. Preventis, K. Stravoskoufos, S. Sotiriadis, and E. Petrakis, "Personalized motion sensor driven gesture recognition in the FIWARE cloud platform," Intelligent Systems Laboratory, Department of Electronic and Computer Engineering, Technical University of Crete (TUC) Chania, Greece, GR-73100, 2013.

[11] K. Boudjit, C. Larbes, and M. Alouache, ""Control of Flight Operation of a Quad rotor AR. Drone Using Depth Map from Microsoft Kinect Sensor," in international Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 3, 2013.

[12] J. S. G. Guerrero, A. F. C. González, J. I. H. Vega, and L. A. N. Tovar, "Instrumentation of an Array of Ultrasonic Sensors and Data Processing for Unmanned Aerial Vehicle (UAV) for Teaching the Application of the Kalman Filter," *Procedia Computer Science,* vol. 75, pp. 375-380, // 2015.

[13] R. Y. Park, J. M. Pak, C. K. Ahn, and M. T. Lim, "Image stabilization using FIR filters," in *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, 2015, pp. 1234-1237.

[14] S. A. Habsi, M. Shehada, M. Abdoon, A. Mashood, and H. Noura, "Integration of a Vicon camera system for indoor flight of a Parrot AR Drone," in *Mechatronics and its Applications (ISMA), 2015 10th International Symposium on*, 2015, pp. 1-6.

[15] T. Stuart and C. Anderson, "Disrupting the drone marke," Berkeley-haas case series 3d robotics, California management review, vol. 57, No. 2, 2015.

[16] T. Labs. *Autoflight*. Available: http://developerblog.myo.com/myocraft-fly-your-parrot-ardrone-2-0-with-autoflight/
[17] T. Labs. *Flying drones with Myopilot*. Available: http://developerblog.myo.com/flying-drones-with-myopilot/
[18] T. Labs. *Myo + Parrot 3.0*. Available: https://market.myo.com/app/559ec0cee4b0f2c8982c9164/myo--parrot-30
[19] T. Labs. *Myo - Real Life Applications of the Myo Armband*. Available: https://www.youtube.com/watch?v=te1RBQQlHz4
[20] Parrot. *Bebop Drone - LAUNCH VIDEO*. Available: https://www.youtube.com/watch?v=WUpmPUcnuBY&nohtml5=False
[21] E. Cubillos. *Myo armband + Unity 5+ SDK*. Available: https://youtu.be/emamv5gsQ5A
[22] NiklasRosenstein. *Myo Python*. Available: https://github.com/NiklasRosenstein/myo-python
[23] Robotika.cz. *Robotika*. Available: http://robotika.cz/cs
[24] Robotika. *Katarina, Parrot drone Bebop*. Available: https://github.com/robotika/katarina
[25] E. Cubillos. *MYO+PYTHON+BEBOP*. Available: https://github.com/Cubillosxy/MYO-PYTHON-BEBOP
[26] E. TIEMPO. *Nueva Reglamentación para los drones* Available: http://www.eltiempo.com/tecnosfera/novedades-tecnologia/nueva-reglamentacion-para-los-drones/16353316
[27] S. Exchange. *Stackoverflow*. Available: http://stackoverflow.com/
[28] Parrot. *Parrot for Developers*. Available: http://forum.developer.parrot.com/
[29] U. Technologies. *Unity Community*. Available: http://forum.unity3d.com/
[30] GitHub. *Github*. Available: https://github.com/
[31] T. Labs. *Myo Developer*. Available: https://developer.thalmic.com/forums/
[32] R. A. ESPAÑOLA, "RAE."
[33] E. Howell, "What is A Drone?."

LISTA DE ANEXOS

1. Manual de Operación
2. Programa MYO-PYTHON-BEBOP
3. Costo del proyecto
4. Prueba Funcional (Video)
5. Otros