ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA (IEM)

CONTROL DE UN BRAZO ROBÓTICO PARA EMULACIÓN DE LOS MOVIMIENTOS DE UNA MANO

Autor: Carlos Ripoll Ramzi

Directores: Juan Luis Zamora Macho

José Porras Galán

Madrid, Agosto de 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. CARLOS RIPOLL RAMZI

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: CONTROL DE UN BRAZO ROBÓTICO PARA EMULACIÓN DE LOS MOVIMIENTOS DE UNA MANO que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 27 de AGOSTO de 2018

ACEPTA

Fdo: CARLOS RIPOLL RAMZI

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título CONTROL DE UN BRAZO ROBÓTICO PARA EMULACIÓN DE LOS MOVIMIENTOS DE UNA MANO en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el Curso académico 2017/2018 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Carlos Ripoll Ramzi Fecha: 27/08/2018

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Juan Luis Zamora Macho Fecha: 27/08/2018

CONTROL DE UN BRAZO ROBÓTICO PARA EMULACIÓN DE LOS MOVIMIENTOS DE UNA MANO

Autor: Carlos Ripoll Ramzi

Directores: Juan Luis Zamora Macho y José Porras Galán

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

Resumen del proyecto

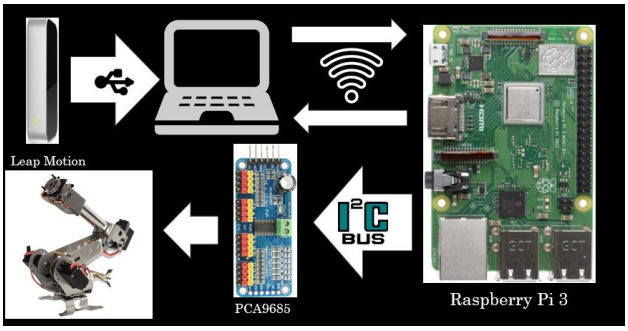
Hoy en día, la gran precisión de los brazos robóticos ha hecho de ellos un gran aliado en el ámbito de las operaciones quirúrgicas [1]. Además, su versatilidad a la hora de trabajar en diferentes horarios o en sesiones ininterrumpidas les ha abierto un hueco en cadenas de montajes. Sin embargo, cuando se controlan estos de forma libre, su control es lento y siempre a través de una emisora o mando. Con el fin de aprovechar las ventajas del uso de los brazos robóticos en el ámbito doméstico y de la pequeña y mediana empresa, especialmente para personas con discapacidades físicas, se necesita un control intuitivo y barato.

Palabras clave: Brazo robótico, Leap Motion

1. INTRODUCCIÓN

Los brazos robóticos son unas herramientas robustas y muy versátiles. Con el tiempo, han ido apareciendo en el mercado una gama de brazos robóticos de una calidad más que aceptable y a precios más asequibles, capaces de mover más de 100kg de carga. Los precios de estos brazos oscilan alrededor de los mil euros (fuente: https://www.roscomponents.com/es/15-brazos-roboticos), lo que los hace accesibles en el ámbito de la pequeña y mediana empresa, y en hogares de personas con alguna discapacidad física.

Para hacer estos robots más accesibles a las personas “de a pie”, es necesario poder controlarlos de una forma sencilla, intuitiva y de bajo coste, como pueden ser gestos corporales recogidos por un sensor infrarrojos. El objetivo del proyecto es el de poder controlar un brazo robótico mediante gestos corporales, que se recogerán con un sensor infrarrojo Leap Motion, y se enviarán a los servos mediante la tarjeta PCa9685 conectada a una Raspberry Pi Modelo 3B. En la siguiente figura se puede observar el esquema de conexiones y comunicaciones del sistema.



2. MODELADO DEL SISTEMA

2.1 Brazo robótico

El robot que se usará en el proyecto será una variante del estándar PUMA de seis grados de libertad, hecho en aluminio y de unos 600 gramos, con una región de alcance de unos 25cm.. El modelado de este brazo puede hacerse mediante la matriz de Denavit-Hartenberg. Esta matriz (que posteriormente nos dará seis, una por cada grado de libertad) nos permitirá, conociendo la posición angular de cada uno de los servos, obtener la posición y orientación de cada uno de los nodos del brazo (articulaciones + pinza). Las ecuaciones de cinemática inversa (obtener las posiciones angulares de cada servo para lograr una determinada posición y orientación en la pinza del brazo ya han sido resueltas previamente para el PUMA estándar, así que se han estudiado y modificado las ecuaciones para su uso con el brazo de estudio. Estos cálculos se explicarán más adelante. Todos los cálculos se realizarán mediante una S-function, que nos permitirá ejecutar en el entorno de SIMULINK un script escrito en lenguaje MATLAB.

2.2 Actuadores: servos

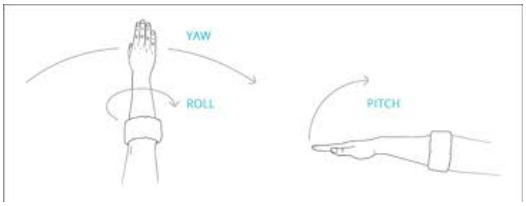
Se emplearán 3 servos de 180 grados de recorrido (tipo MG995), uno de 270 grados de recorrido (tipo DS3218), y dos micro-servos de 180 grados de rango (tipo MG90S). Todos ellos son controlados mediante señales moduladas por ancho de pulso de 50Hz (20ms de periodo) cuyo ancho de pulso varía entre los 500 y los 2500 microsegundos. Como la posición angular correspondiente a 0º se consigue con un ancho de pulso de 500us, podemos modelar el factor de servicio de la señal enviada a cada servo mediante la ecuación:

$$FS = 500 + \frac{2000}{\text{Rango servo}(180\ o\ 270)} * \theta$$

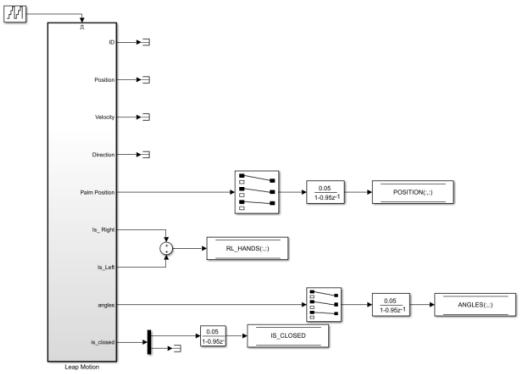
Esta ecuación consiste en un offset más la referencia multiplicada por una sensibilidad. Puede variar en función del rango de la entrada de posición angular ([-90,90]o [0,180], etc…) y ha sido especificada para cada servo, ya que sus distintas condiciones de trabajo han resultado en comportamientos que difieren sensiblemente de la ecuación ideal.

2.3 Sensor: Leap Motion

El sensor Leap Motion logra detectar con gran precisión la posición, velocidad y orientación de manos y objetos pequeños. Consta de dos cámaras infrarrojas y tres LEDs infrarrojos. Este sensor es idóneo para detectar las manos, ya que tiene un rango de alcance similar al rango de movimiento del brazo humano (80cm de alcance y hasta 120º en el eje Z y 150º en los ejes X e Y). Las entradas que necesitaremos del sensor serán el número de manos detectadas y la posición, orientación y apertura de la mano que denominaremos principal (explicado más adelante). La orientación de la mano se definirá mediante una terna de ángulos muy utilizada en aeromodelismo: guiñada, alabeo y cabeceo (Yaw, Pitch y Roll). Esto puede observarse en la siguiente figura:



Dado que Simulink no posee una toolbox para el procesado de las entradas provenientes del sensor y el Leap Motion no puede ser controlado mediante la Image Acquisition Toolbox de SIMULINK, se ha tenido que buscar una solución alternativa, que se ha encontrado en el conjunto de archivos denominado Matleap [2]. Estos son un conjunto de Mex-Files, o scripts escritos en lenguaje C++ pero que son ejecutables en MATLAB mediante el uso de un compilador (que incluye la versión 2018A de MATLAB). Estos scripts obtienen frames (o fotos) tomadas por el sensor y extraen su información. Para poder ejecutar estos scripts en el entorno de SIMULINK será necesario el uso de una S-function de segundo orden, que puede verse en la siguiente figura.

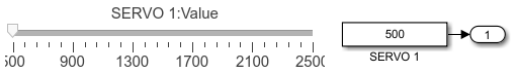


2.4 Comunicación: Buses y protocolos

Toda la información se organizará en buses para obtener una interfaz más cómoda. El bus más importante será el de control, y en él se incluirán las variables de entrada (medidas del sensor), las intermedias o de control (ángulos en grados) y los parámetros de salida (ancho de pulso de cada señal PWM). Los medios de comunicación (que pueden observarse en la figura 1) serán USB para el Leap Motion, y Wi-Fi e I2C para la Raspberry y la tarjeta PCA9685. En la comunicación vía Wi-Fi se empleará el protocolo MAVLink, ya que es muy ligero y óptimo para comunicaciones con ancho de banda limitado.

3. METODOLOGÍA. ENSAYOS Y RESULTADOS

Una vez montado el brazo con los servos, se controló cada uno de ellos empleando un probador digital de servos para estudiar el rango de movimiento de cada articulación, así como para calibrar las ecuaciones de cada servo. Las ecuaciones corregidas para cada uno de ellos se encuentran en el interior del bloque que se muestra en la siguiente figura. Todas ellas son similares a la ideal, pero con cambios en algunos parámetros (sensibilidad, offset). Posteriormente, se empleó un módulo UART y una tarjeta de servos PS2 para controlar los servos desde Simulink (ya que la primera intención del proyecto era controlarlo mediante un módulo UART). La referencia para cada servo se creará mediante una deslizadera asociada a un generador de valor, y se puede observar en la siguiente figura.



Finalmente, cuando se decidió introducir la tarjeta PCA9685 y la Raspberry y cambiar el medio de comunicación de UART a Wi-Fi, se repitió el proceso anterior.

El bloque de control, al constar de dos apartados (cinemática directa e inversa) que constituían un circuito cerrado, fue comprobado a la vez. Primero, se ejecutaba la cinemática inversa

introduciendo una posición y orientación cualesquiera, y posteriormente se introducían los valores obtenidos para los servos en el script de cinemática directa. Una vez se corrigieron errores de signos en los ejes de coordenadas tomados, se consiguió obtener de nuevo la posición y orientación de partida.

Cabe destacar que, debido a los problemas de paso de configuración flip a no flip y viceversa, se optó por una solución más intuitiva para los tres ángulos de la muñeca. Esto se explicará más adelante.

4. CONCLUSIONES

Las conclusiones de este proyecto son muy positivas, dado que no solo se ha conseguido un control preciso de la posición y orientación para el brazo y sensor empleados, sino que la estructura modular en la que se ha dividido el proyecto permite el uso de cualquier tipo de control, brazo o sensor, siempre y cuando se consiga obtener de ellos las variables necesarias (p.ej, posición, orientación y apertura de la mano en el caso del sensor).

Además, el sensor Leap Motion ha causado una impresión excelente. A pesar de su reducido tamaño y peso, y de su bajo precio, tiene una gran precisión.

Con respecto al brazo robótico, su estructura de aluminio y bajo precio hacen de él un buen brazo para aficionados a la robótica, pero su diseño (en especial, la relación no lineal entre los ángulos θ_2 y θ_3) comprometen en gran medida su precisión. Para trabajos en los que se requiera precisión, sería recomendable obtener un robot más robusto (como puede ser el Dobot Magician [2]) y unos servos tipo “Smart Control”, interconectados por comunicación serie.

1. BIBLIOGRAFÍA

[1] A. Alfageme, «Conoce al ‘supercirujano’ de las manos de acero,» 15 Noviembre 2016. [En línea]. Available: https://elpais.com/elpais/2016/11/01/talento_digital/1478021936_889288.html. [Último acceso: 29 08 2018].

[2] «Dobot Magician Overview,» Dobot, [En línea]. Available: https://www.dobot.cc/dobot-magician/product-overview.html. [Último acceso: 20 08 2018].

[3] A. Colgan, «How does the Leap Motion Controller work?,» Leap Motion Blog, 9 08 2014. [En línea]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. [Último acceso: 10 07 2018].

[4] Wikipedia, «Leap Motion - Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Leap_Motion#History. [Último acceso: 20 07 2018].

[5] T. Foster, «Will these guys kill the computer interface as we know it?,» Popular science, 22 July 2013. [En línea]. Available: https://www.popsci.com/technology/article/2013-07/will-these-guys-kill-computer-interface-we-know-it. [Último acceso: 1 08 2018].

[6] K. Hay, «Designing the Leap Motion Controller,» Blog Leap Motion, 3 April 2013. [En línea]. Available: http://blog.leapmotion.com/designing-leap-motion-controller/. [Último acceso: 18 July 2018].

[7] G. Leal, «Owi arm controlled by a LEAP Motion,» [En línea]. Available: https://hackaday.io/project/28551-owi-arm-controlled-by-a-leap-motion#menu-description. [Último acceso: 25 04 2018].

[8] J. Belda, «Leap Motion (II): Principio de funcionamiento,» ShowLeap Blog, 04 05 2015. [En línea]. Available: blog.showleap.com/2015/05/leap-motion-ii-principio-de-funcionamiento/. [Último acceso: 15 01 2018].

[9] J. Perry, «Git Hub,» 2013. [En línea]. Available: https://github.com/jeffsp/matleap. [Último acceso: 16 02 2018].

[10] M. M. Botella, «CONTROL DE NAVEGACIÓN DE UN CUADRICÓPTERO MEDIANTE GESTOS,» *Trabajo de fin de grado curso 2017/2018. Escuela técnica superior de ingeniería ICAI.* Agosto 2018.

[11] «MAVLink Developer Guide,» [En línea]. Available: https://mavlink.io/en/. [Último acceso: 20 08 2018].

[12] M. Morales, «Fundamentos del protocolo I2C,» Teslabem, 2017 Febrero 2017. [En línea]. Available: https://teslabem.com/learn/fundamentos-del-protocolo-i2c-aprende/. [Último acceso: 8 Agosto 2018].

Project summary

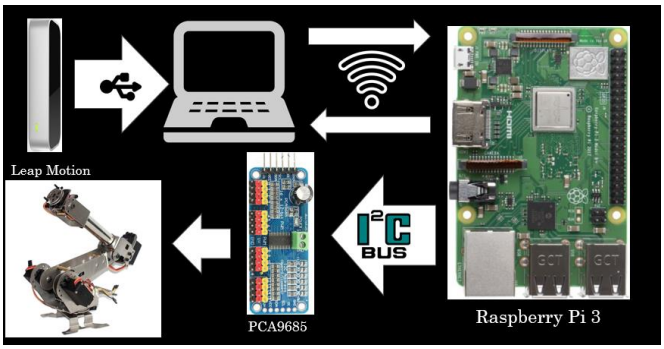
Nowadays, robotic arms' precision makes them perfect tools for surgery [1]. What is more, the fact that they can work at any time for any amount of time has earned them a spot in assembly lines. However, when it comes to remote controlling them, the existing control methods are too slow and all include remote controls. In order to make use of all the perks involving robotic arm application in small businesses and at physically disabled people's houses, a simple and cheap, but fast and intuitive way of controlling them is needed.

Key terms: Robotic arm, Leap Motion

1. INTRODUCTION

Robotic arms are very powerful and versatile tools. Over time, a new stock of affordable, high-quality robotic arms which are capable of lifting over 100kg has appeared. Their price usually approaches a thousand euros (source: https://www.roscomponents.com/es/15-brazos-roboticos), which makes them affordable for small and medium-sized companies, or at homes where a person with physical disabilities lives.

To make these robots more suitable for unexperienced people, there needs to be a way to control them in a simple, cheap and intuitive way, for example, body gestures and movements captured by an infrared sensor. The goal of this project is to control a robotic arm by hand gestures captured by a Leap Motion sensor and will be sent to the robot's servos by a PCA9685 servo driver board, which will be connected to a raspberry pi model 3B. The following figure shows the connection diagram.



2. SYSTEM MODELING

2.1 Robotic arm

The used robotic arm corresponds to a variation of a PUMA 6 degrees-of-freedom robotic arm. It is made of aluminum and weights over 600 grams without the servos. This robot can reach objects located closer than 25cm to its base. Robotic arms can be modeled by the Denavit-Hartenberg matrix. This matrix (which will be used to calculate other 6 more, one for every degree of freedom) allows us to know the position and orientation of every single joint of the robotic arm, given the servo angular positions.

Inverse kinematics equations (the ones who solve for the servo angles, given a final position and orientation of the robot's claw) have already been solved for the PUMA robot, but they need to be studied and changed for the specific arm that is used. All the calculations will be explained further on. All these

calculations will be calculated at a S-Function, which allows us to run MATLAB code inside SIMULINK.

2.2 Actuators: Servos

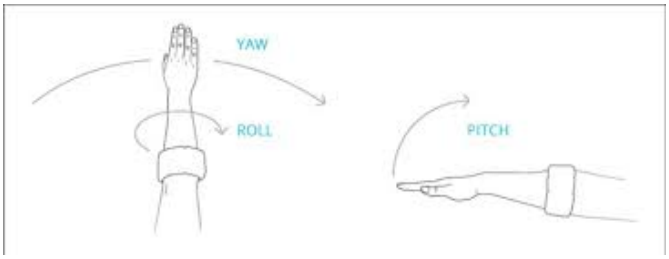
Three 180-degree servos (as the MG995 model), one 270-degree servo (as the DS3218 model) and two 180-degree micro servos (as the MG90s model) will be used. All of them are controlled by 50Hz Phase-Width-Modulated signals. Their duty cycle can take any value from 500 to 2500us, which will result in an angle between 0 and 180 or 270 degrees. Therefore, the calculation of the needed duty cycle to obtain a desired angle will be done using the following equation:

$$DC = 500 + \frac{2000}{\text{Range } (180 \; or \; 270)} * \theta$$

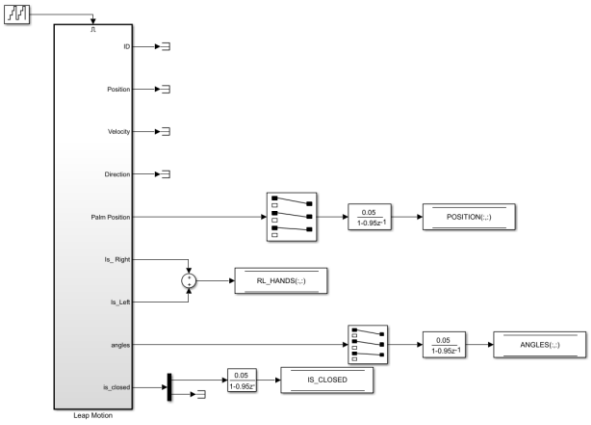
This equation is formed by an offset plus a gain (sensitivity) multiplied by the reference. These values may vary depending on the servo angle range ([-90,90], [0,180] …). After testing, this equation has been changed to meet as precise as possible every single real servo's behavior, since they all work under different conditions.

2.3 Sensor: Leap Motion

The Leap Motion sensor can measure precisely the speed, position and orientation of hands or small objects located above it. It consists of three infrared LEDs and two infrared cameras. This sensor is perfect for detecting hands, since its detecting range is similar to a human arm's movement range (80cm within 120º of the Z-axis and 150º of the X and Y axis). The parameters that will be taken from the sensor will be the number of hands detected over it and the position, orientation and opening of the so called controlling hand (this will be explained further below). Hand's orientation will be defined by three angles commonly used in aeromodelling: Yaw, pitch and roll. These angles can be seen on the following figure:



Because SIMULINK does not own a Leap Motion toolbox, and this sensor can't be controlled by SIMULINK's image acquisition toolbox, an alternative way was needed. The solution came from GitHub forum, in the form of a bunch of Mex-files called Matleap [2]. Mex-files are C or C++ files which include a function or subroutine, and can be used in MATLAB using a compiler, like the MinGW compiler add-on available for MATLAB 2018A. These scripts are able to obtain frames taken by the Leap Motion and extract their information. In order to execute these files in SIMULINK, a 2nd order S-Function will be used. The following figure shows the multiplexed outputs of the s-function.



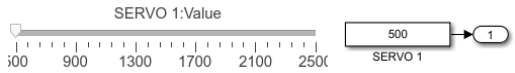
2.4 Communication: Bus structures and protocols

All the information will be stored in data buses to obtain a simpler and more

intuitive interface. The most important bus is the control bus, where all the sensor, control and output parameters are stored. In order to keep the all of the system components communicated, three physical communication structures will be used: Wi-Fi between the control station (PC) and the Raspberry Pi (using MAVLink message coding protocol, since it is a very light weight protocol, optimal for connections through poor bandwidth), I2C between the servo board and the Raspberry Pi and USB between the sensor and the control station. All this structure can be seen on the first figure.

3. Methodology: Tests and results

Once the robotic arm was assembled with all the servos, each of them was tested using a digital servo tester. This helped to acknowledge their real movement range and to set them at the optimal angular position (centered). Finally, the behavior equations could be specified for each of them, all of them similar to the generic equation presented previously, but with slight variations. After that, UART communication system and a PS2 Servo board were used to control the servos from the control station using sliders. Each of the sliders was connected to a constant generator, this can be seen on the figure below:



Since the control unit is made up of two blocks that provide a way back and forth between position + orientation and servo angles, once one of them is proven correct, it can be used to check the second one. The Denavit-Hartenberg matrix was tested by hand providing different servo angles and calculating the final position and orientation. After that, it was used to check the results of the inverse kinematics equation.

It is important to emphasize that, since problems arose when flip and no flip wrist configurations were used, an alternative control (which will be explained below) was used instead.

4. Conclusions

The overall conclusions drawn from the project were positive, since not only a precise position and orientation control was designed, but the modular structure created makes possible the use of different control systems (such as a speed control), sensors and any 6 DOF robotic arm, as long as all the variables needed from every block are provided.

Furthermore, the Leap Motion sensor has proven itself to be a cheap, yet precise and reliable sensor. This is really commendable given its small size and price.

Concerning the robotic arm, its aluminum structure and cheap price make it a very good option for robotic hobbyists, but its lack of sturdiness and especially the nonlinear relation concerning the angles θ_2 and θ_3 compromise its overall precision. Therefore, if precision is wanted to be held paramount, other sturdier robotic arms (such as the Dobot Magician [3]) and smart-controlled, more powerful servos should be used.

5. BIBLIOGRAPHY

[1] A. Alfageme, «Conoce al 'supercirujano' de las manos de acero,» 15 Noviembre 2016. [En línea]. Available: https://elpais.com/elpais/2016/11/01/talento_digital/1478021936_889288.html. [Último acceso: 29 08 2018].

[2] «Dobot Magician Overview,» Dobot, [En línea]. Available: https://www.dobot.cc/dobot-magician/product-overview.html. [Último acceso: 20 08 2018].

[3] A. Colgan, «How does the Leap Motion Controller work?,» Leap Motion Blog, 9 08 2014. [En línea]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. [Último acceso: 10 07 2018].

[4] Wikipedia, «Leap Motion - Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Leap_Motion#History. [Último acceso: 20 07 2018].

[5] T. Foster, «Will these guys kill the computer interface as we know it?,» Popular science, 22 July 2013. [En línea]. Available: https://www.popsci.com/technology/article/2013-07/will-these-guys-kill-computer-interface-we-know-it. [Último acceso: 1 08 2018].

[6] K. Hay, «Designing the Leap Motion Controller,» Blog Leap Motion, 3 April 2013. [En línea]. Available: http://blog.leapmotion.com/designing-leap-motion-controller/. [Último acceso: 18 July 2018].

[7] G. Leal, «Owi arm controlled by a LEAP Motion,» [En línea]. Available: https://hackaday.io/project/28551-owi-arm-controlled-by-a-leap-motion#menu-description. [Último acceso: 25 04 2018].

[8] J. Belda, «Leap Motion (II): Principio de funcionamiento,» ShowLeap Blog, 04 05 2015. [En línea]. Available: blog.showleap.com/2015/05/leap-motion-ii-principio-de-funcionamiento/. [Último acceso: 15 01 2018].

[9] J. Perry, «Git Hub,» 2013. [En línea]. Available: https://github.com/jeffsp/matleap. [Último acceso: 16 02 2018].

[10] M. M. Botella, «CONTROL DE NAVEGACIÓN DE UN CUADRICÓPTERO MEDIANTE GESTOS,» *Trabajo de fin de grado curso 2017/2018. Escuela técnica superior de ingeniería ICAI.* Agosto 2018.

[11] «MAVLink Developer Guide,» [En línea]. Available: https://mavlink.io/en/. [Último acceso: 20 08 2018].

[12] M. Morales, «Fundamentos del protocolo I2C,» Teslabem, 2017 Febrero 2017. [En línea]. Available: https://teslabem.com/learn/fundamentos-del-protocolo-i2c-aprende/. [Último acceso: 8 Agosto 2018].

MEMORIA

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1 Descripción del proyecto

El objetivo de este proyecto es el de crear una estructura modular para controlar cualquier brazo robótico de seis grados de libertad mediante gestos corporales, y desarrollar un control de posición mediante mimetización para el brazo robótico PUMA de seis grados de libertad. Como sensor se ha elegido el Leap Motion, capaz de detectar la posición y orientación de las manos, lo que eliminará la dificultad de aprender a usar un mando o controlador. El sensor se conectará al PC mediante un cable USB, y los servos se controlarán mediante la tarjeta PWM PCA9685, comunicada a una Raspberry Pi modelo 3B mediante el protocolo I2C. La comunicación entre la Raspberry y el PC se hará por conexión Wi-Fi, usando el protocolo UDP y empleando el protocolo de mensajería diseñado para drones MAVLink.

1.2 Motivación del proyecto

Las razones por las que se ha decidido realizar este proyecto son varias, y se resumen en:

- **Afición por la robótica:** Es de gran ayuda tener interés por el proyecto en el que se está trabajando, ya que se podrá trabajar en mejores condiciones durante períodos de tiempo más largos
- **Auge de las tareas asistidas o desarrolladas por robots:** En especial en el ámbito de la pequeña y mediana empresa, herramientas más versátiles y polivalentes pueden desarrollar distintas tareas en función de las necesidades del momento
- **Potenciales beneficios del uso de robots en la vida cotidiana:** Personas mayores o con discapacidad podrían ver su independencia incrementada ayudándose de un robot para tareas que no sean capaces de realizar por sí mismos o que conlleven un riesgo (coger algo de una estantería alta, abrir una ventana en una buhardilla…).

1.3 Objetivos del proyecto

El proyecto tiene como objetivo establecer un punto de partida para el diseño de controles para brazos robóticos de seis grados de libertad mediante la creación de la estructura modular que permita la rápida sustitución de sus módulos. Además, se diseñará un control de posición y orientación empleando el sensor Leap Motion.

Los factores determinantes del éxito del proyecto serán los siguientes:

- La estructura creada permite la implantación de distintos controles mediante un cambio en el módulo de control.
- La estructura creada permite la implantación de los controles en cualquier brazo de seis grados de libertad, introduciendo sus datos en el módulo de modelo.

- El brazo robótico empleado es capaz de reproducir la posición y orientación del brazo humano situado sobre el sensor Leap Motion. Además, la máquina de estados proporcionará la posibilidad de encenderlo y apagarlo.

1.4 Recursos utilizados

Para llevar a cabo el proyecto, se hará uso de los siguientes recursos:

- Brazo robótico de 6 grados de libertad y cuerpo de aluminio.
- 3 servo motores 180º (DS3218) y uno de 270º (MG995)
- 2 micro servos estándar (MG90S)
- Tarjeta de 16 servos PCA9685
- Raspberry Pi 3 modelo B
- Ordenador personal con MATLAB y SIMULINK, versiones 2018A.
- Sensor Leap Motion
- Cable micro USB
- Fuente de alimentación 6V 3A.
- Herramientas (destornilladores, llaves Allen

1.5 Metodología de trabajo

Para facilitar la depuración de errores, y desarrollar una estructuración temporal coherente, se dividirá el proyecto en cinco bloques/fases, independientes entre sí, más una última interconexión entre sí. Estos serán:

- **Montaje del brazo robótico. Prueba de servos y movilidad:** El brazo empleado llegará completamente desarmado, por lo que se armará al completo. Además, se comprobará que cada uno de los servos empleados funciona en su rango de pulsos teórico (500 us-2500us), y que, al montarlo, se consigue una total movilidad de cada una de las articulaciones.
- **Comunicación entre el ordenador y los servos.** Previa descarga de los programas Matlab y Simulink en el ordenador personal, se establecerá la comunicación con los servos. Esta fase se considerará completa cuando, introducida una referencia en microsegundos en Simulink, Se cree el comando (Número de servo + duración del pulso) y este sea enviado a través del módulo UART.
- **Traducción entre referencia en grados y ancho de pulso:** La matriz de Denavit-Hartenberg nos proporcionará los ángulos en grados para cada servo, por lo que se deberá establecer, para cada uno de ellos, la ecuación que nos dará el ancho de pulso necesario para obtener el ángulo deseado. Para cada servo,

mediante toma de datos (tabla), se creará la función, por aproximación lineal. (polinomio de primer orden).

- **Elaboración de la matriz de Denavit-Hartenberg:** Se estudiará, para el brazo de estudio, los elementos de la matriz de Denavit-Hartenberg, que, dada una posición en el espacio y una orientación, proporcionará los ángulos necesarios en cada una de las articulaciones para conseguir la misma posición y orientación en la punta del brazo robótico.

- **Obtención de posición y orientación con el sensor Leap Motion:** Mediante los ficheros de Simulink provistos por un TFG anterior a este, se obtendrán los valores de altura y orientación de la mano, así como también se establecerán algunos comandos de apagado de emergencia y encendido.

- **Interconexión de los bloques 2, 3, 4 y 5:** La posición y orientación obtenidas por el sensor Leap Motion se introducirán en la matriz de Denavit-Hartenberg, que suministrará los ángulos de cada articulación. Estos ángulos se introducirán en el bloque de cálculo de factor de servicio de los servos, y finalmente se enviarán estos factores de servicio a los servos.

Este proyecto se empezó en junio de 2017 con la intención de finalizarlo en mayo de 2018 siguiendo la progresión que se puede observar en el siguiente diagrama de Gannt:

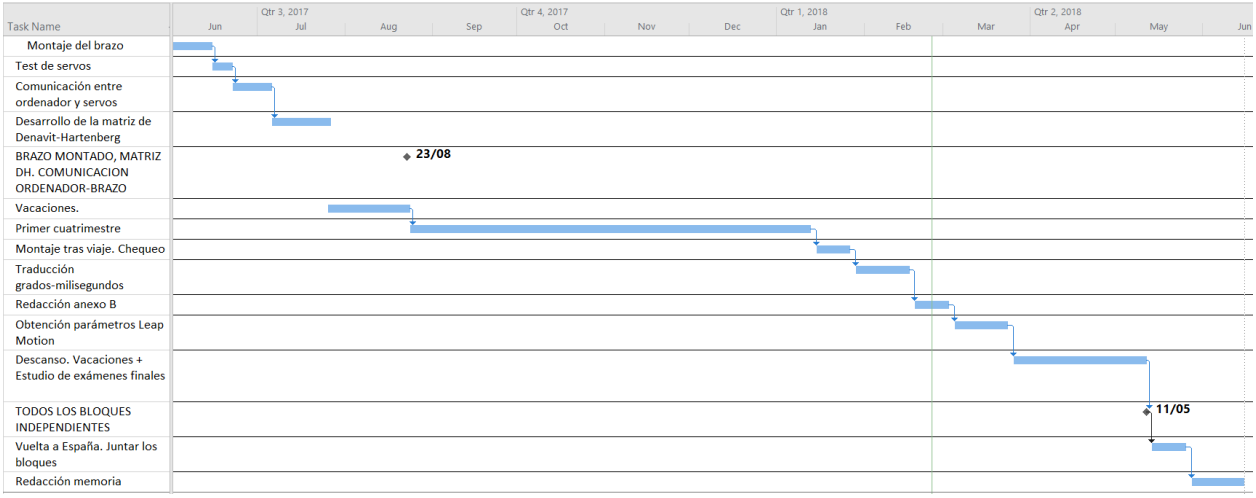


Figura 1. Diagrama de Gannt.

Sin embargo, debido al trabajo extra que supuso el cambio de la estructura de todo el proyecto, al cambio de tarjeta de servos (que supuso también un cambio en el protocolo de comunicación) y a la incorporación de la Raspberry, se tuvo que aplazar la fecha de finalización a Julio.

2.ESTADO DEL ARTE

2.1 Componentes

2.1.1 Brazo robótico de seis grados de libertad

El brazo robótico que se empleará en el proyecto (y que se puede observar en la figura 1) corresponde a la configuración estándar PUMA de 6 grados de libertad, el más usado actualmente. Además de los grandes brazos industriales, se comercializan modelos en aluminio a escala para amantes de la robótica a un precio asequible (50€ sin servos, unos 200€ con ellos). Estos servos son ligeros y fáciles de transportar, a la par de resistentes.

Una ventaja del método empleado para el control (Matriz de Denavit-Hartenberg) es que, en caso de querer cambiar de brazo, bastará con cambiar los parámetros de la matriz (medida de los brazos y posición de las articulaciones) para conseguir los resultados de posición y orientación deseados. Esto nos permitirá desarrollar un sistema de gran adaptabilidad.

Figura 2. Brazo robótico de seis grados de libertad.

2.1.2 Sensor Leap Motion

El sensor Leap Motion permite conocer la posición, velocidad y orientación de manos y objetos pequeños. Consta de dos cámaras infrarrojas monocromáticas y tres LEDs infrarrojos, y detecta luz infrarroja de 850 nm de longitud de onda a una frecuencia máxima de 200 frames por segundo [3]. Las principales ventajas del Leap Motion frente a su principal competencia (Kinect V1) eran su pequeño tamaño y la gran precisión que ofrecía.

Figura 3. Tamaño del Leap Motion

Su tecnología fue primeramente desarrollada en 2008 por David Holz, mientras estudiaba para su doctorado en matemáticas. A este se unió su amigo de la infancia Michael Buckwald en 2010 [4]. Su empresa consiguió una elevada suma de dinero por parte de empresas inversoras y posteriormente por su asociación con ASUS en 2013.

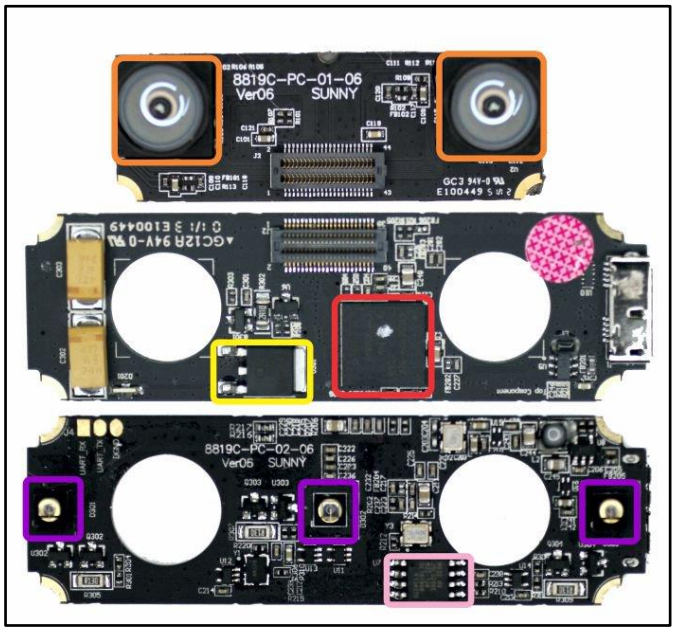


Figura 4. despiece de sensor Leap Motion

Su principal objetivo era el de optimizar la forma en la que los humanos se "comunicaban" con los ordenadores, ya que el conjunto teclado-ratón no siempre es intuitivo y rápido [5]. Mediante una interacción más intuitiva y completa con el ordenador, determinadas tareas podrían hacerse de una forma más rápida y sencilla.

El dispositivo, que fue primeramente llamado The Leap, fue disminuyendo progresivamente de tamaño hasta el modelo de diciembre de 2012, que es el empleado para el proyecto [6].

Figura 5. Evolución del tamaño del Leap Motion

2.1.3 Servos (DS3218MG) y micro servos (MG90S) estándar

Los servos que se emplearán (que se pueden encontrar en la figura 3) son los modelos DS3218MG y MG90S. Ambos van alimentados a 5V y tienen un par máximo de 15kg*cm y 1.8kg*cm, respectivamente.

Estos servos son controlados por ondas cuadradas de período fijo (20 milisegundos) cuyo ancho de pulso (o ciclo de trabajo) determina la posición angular. Estos anchos de pulso pueden tomar cualquier valor entre 500 y 2500 microsegundos, que corresponderán, respectivamente, a 0 grados y 180 o 270 (dependiendo del servo) grados.

Figura 6. Servos y micro servos

2.2 Sistema de control

Desde la implantación de los brazos robóticos en el mundo de la industria y posteriormente en la medicina, se ha tratado de simplificar el control de estos.

Inicialmente, los brazos han sido programados para realizar tareas repetitivas, inculcadas mediante programas. Poco a poco se ha ido avanzando hacia el control en tiempo real de los brazos para otorgar una mayor versatilidad a estos.

Recientemente, con la aparición de los modernos y avanzados sensores de posición (Leap Motion, Kinect…), la aplicación de estas tecnologías a controlar robots y al entorno de la realidad virtual se está extendiendo a una velocidad de vértigo. Se tiene constancia de proyectos para el control de brazos robóticos mediante gestos con los brazos mediante Arduino [7]. Sin embargo, su existencia es puramente experimental, y no se ha dado el salto a la comercialización del sistema.

3. COMPONENTES DEL SISTEMA

3.1 Hardware

El esquema de componentes hardware, así como de las conexiones que se establecen entre sí pueden observarse en la siguiente figura:

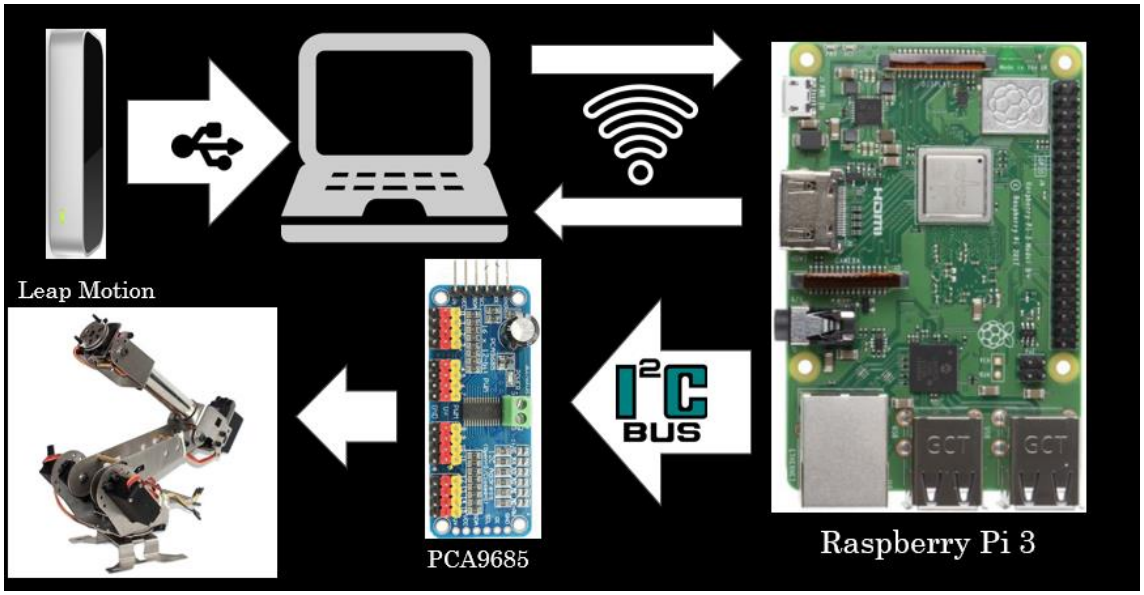


Figura 7. Esquema del hardware

Como puede observarse, el sensor Leap Motion se conectará al PC mediante un cable USB (incluido en su caja). La conexión entre la Raspberry y el PC se realizará mediante un Router Wi-Fi siguiendo el protocolo UDP. Finalmente, la Raspberry enviará las órdenes a la tarjeta controladora de servos mediante un bus I2C. Los servos irán alimentados (a través de la tarjeta) a 6V y 3A, obtenidos mediante un reductor 12-6V conectado a una fuente de tensión de 12V. La Raspberry Pi se alimentará por medio de su puerto micro USB utilizando un adaptador de 5V y 2A.

3.1.1 Modelado del brazo robótico: Ecuaciones de Denavit-Hartenberg.

El brazo robótico que se empleará corresponde a una variante del modelo estándar PUMA de seis grados de libertad. Esto quiere decir que se podrán usar tanto la matriz de Denavit-Hartenberg como las ecuaciones de cinemática inversa ya resueltas para el robot PUMA, ambas ligeramente modificadas para representar la geometría exacta del brazo. La matriz de Denavit-Hartenberg para el brazo de estudio es la siguiente:

```
% DH_Matrix = [theta    alpha     a         d]
  DH_matrix = [theta1   -90      46         0
               theta2     0     117         0
               theta3    90     -17         0
               theta4   -90       0       131.5
               theta5    90       0         0
               theta6     0       0       100];
```

Figura 8. Matriz de Denavit-Hartenberg.

El procedimiento de cálculo que se llevará a cabo para hallar los ángulos de los servos será el siguiente:

1. Cálculo de los ángulos θ_1, θ_2 y θ_3 mediante cinemática inversa.
2. Obtención de la orientación de la muñeca mediante cinemática directa
3. Cálculo de los ángulos θ_4, θ_5 y θ_6 mediante cinemática inversa, para la que será necesaria la orientación de la muñeca.
4. Cálculo del ángulo θ_7 en función de is_closed.

3.1.2 Servos

Los servos son dispositivos capaces de proporcionar una posición angular en función del ancho de pulso de la señal (de 50Hz de frecuencia) que los controla. Este ancho de pulso puede estar comprendido entre 500 y 2500 microsegundos, para un rango de recorrido que puede ser de 180 o 270 grados. Por lo tanto, la señal que se envía a cada servo será una señal de 50Hz y con factor de servicio determinado por la ecuación:

$$FS = 500 + \frac{2000}{\text{Rango servo}(180\ o\ 270)} * \theta$$

Esta fórmula funcionará para un rango de valores de θ [0, (180 o 270)]. Sin embargo, para algunos servos se emplearán ángulos comprendidos en el intervalo [-90, +90], por lo que se cambiará ligeramente la fórmula para esos casos.

3.2 Software

Los bloques de software, encargados de la obtención, interpretación envío de información trabajarán secuencialmente para obtener los anchos de pulso de las señales PWM a partir de la entrada procedente del Leap Motion. La estructura de software se puede observar en la siguiente figura:

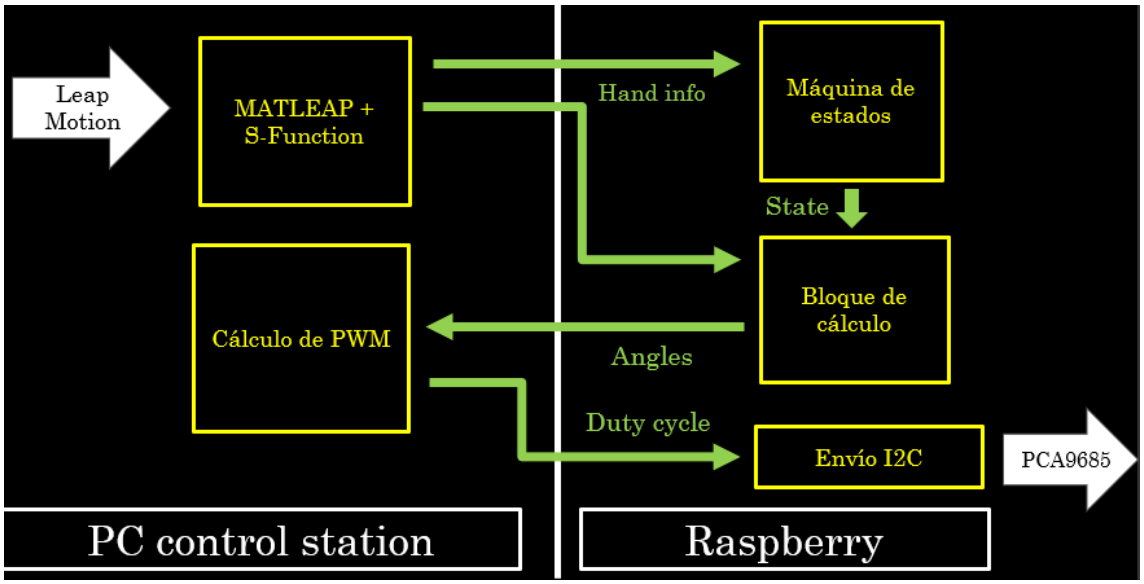


Figura 9. Esquema del software

Como puede observarse, la información proveniente del Leap Motion será obtenida empleando los archivos de Matleap y serán enviados a la máquina de estados y al bloque de cálculo. En la máquina de estados se determinará el estado en el que se encuentra el sistema, y será enviado al bloque de cálculo. En el bloque de cálculo, en

función de el estado en el que se encuentre, se determinará la salida. Esta será enviada al bloque de generación de PWM situada en el PC (En un futuro se puede instalar este bloque en el lado de la Raspberry). Finalmente, el ancho de pulso se enviará desde la Raspberry a la tarjeta controladora de servos.

4. TOMA DE DATOS Y CONTROL: BLOQUES

4.1 Sensor: Leap Motion

4.1.1 Introducción

El sensor Leap Motion consta de dos cámaras infrarrojas y tres LEDs infrarrojos. Estos últimos emiten una luz infrarroja (de 850nm de longitud de onda), que rebota en los objetos y es captada por las dos lentes. Cada una de ellas toma una imagen en escala de grises, y estas son enviadas al ordenador (cabe destacar que una de las razones por las que el Leap Motion es tan rápido es porque las imágenes tomadas solo sufren ajustes de resolución en el dispositivo, y el procesado de ellas se lleva a cabo en el ordenador. [8]

El Leap Motion es capaz de manos u objetos pequeños situados a una distancia máxima de 80cm con respecto al sensor. Además, este radio de detección se mantiene hasta un ángulo de 150º con el ángulo Z y de 120º con respecto a los ejes X e Y. Esto puede observarse en la siguiente figura:

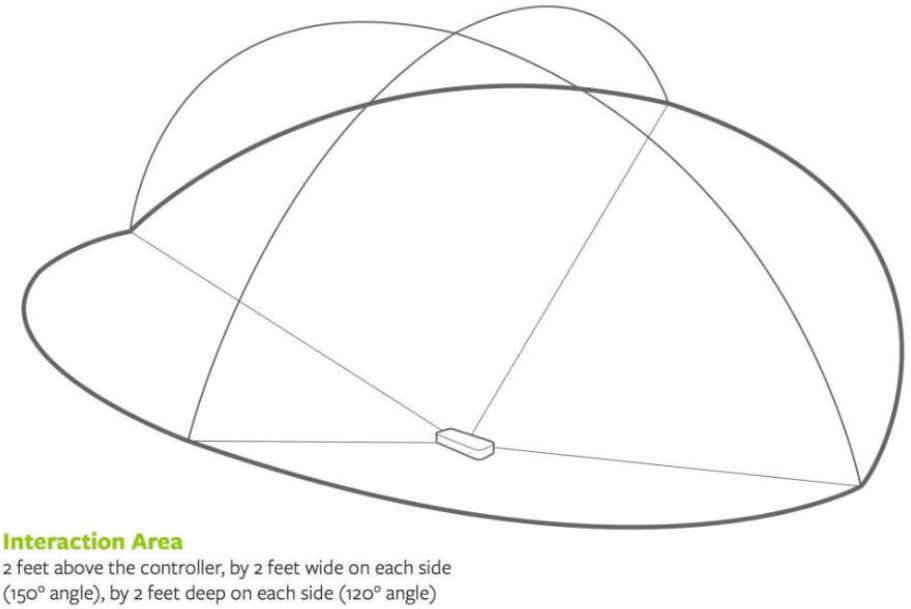


Figura 10. Rango de visión del Leap Motion

4.1.2 Uso del Leap Motion en el proyecto

Las salidas del Leap Motion que se emplearán serán las siguientes:

- **Posición de la mano (Palm Position):** Expresadas en mm en ejes del sensor.

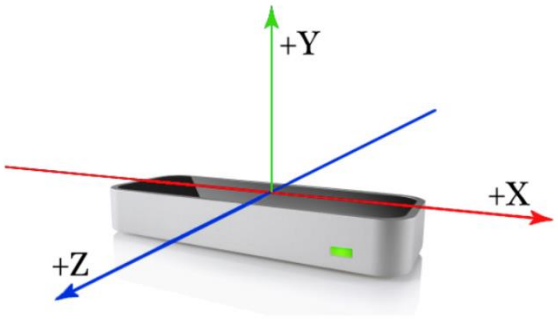


Figura 11. Ejes Leap Motion

- **Orientación de la mano (angles):** Debido a la imprecisión de trabajar con el vector normal a la mano, el sensor nos proporciona la orientación de la palma de la mano mediante tres ángulos: Guiñada (yaw), cabeceo (pitch) y alabeo (roll). Estos ángulos, de los que se obtendrá la matriz de rotación mediante el comando **eul2rotm**.

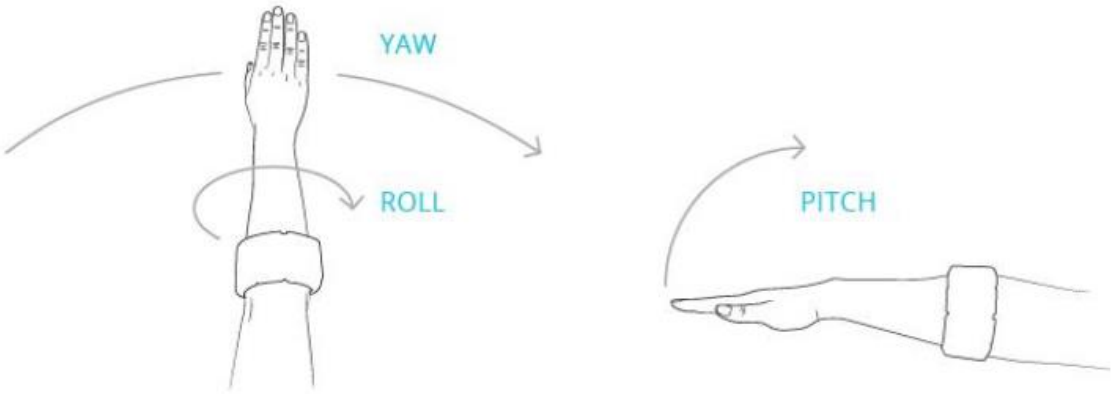


Figura 12. Ángulos Leap Motion.

- **Porcentaje de cierre de mano:** La variable is_closed proporciona, en tanto por uno, cómo de cerrada está la mano (0 para abierta, 1 para cerrada). Esta variable se empleará para el manejo de la pinza.
- **Hand_id:** Variable de 2x1 (ya que su estructura se ha fijado para detectar 2 manos), determinará la columna de cada variable (posición, orientación…) de la mano que se detecta (valdrá [0 0] si no detecta ninguna).
- **is_left e is_right:** Ambas variables de 2x1, tendrán un '1' en una columna si la mano con ese número de identificación (si hand_id vale 1 en esa columna) es izquierda o derecha. Ambas se sumarán para obtener la variable **rl_hand**, que valdrá [0 0] si no se detecta mano, [1 0] si se detecta una mano, o [1 1] si hay dos manos (una será izquierda y la otra derecha. Ya que el sensor está pensado para ser usado por un solo operario, no se contempla la posibilidad de que haya dos manos izquierdas o derechas sobre el sensor.

Dado que el brazo solo se controlará con una mano, se mandarán exclusivamente las variables de posición y orientación de la primera columna (id = 1). La escritura en el bus de control se puede observar en la siguiente figura. Adicionalmente, se ha añadido

un filtro digital paso bajo para la eliminación del ruido proveniente del sensor. Como se explicará más adelante, la S-Function proporciona 19 salidas vectoriales, y se han unido en matrices mediante multiplexores las que correspondían a la misma variable.

4.1.3 Software: Matleap y S-Function

Un problema que se encontró con el uso del Leap Motion fue el no contar MATLAB con una Toolbox para su uso. La solución a este problema llegó de la mano del foro GitHub [9]. En él se encontraba un conjunto de Mex-Files denominado Matleap. Una Mex-File es un programa escrito en C o C++ contiene una función o subrutina y puede ser en el entorno de MATLAB mediante el uso de un compilador. Se descargó el MinGW Compiler para MATLAB del gestor de Add-Ons para su utilización.

Los dos programas que se emplearon para probar el sensor fueron:

- **Matleap_Frame:** Devuelve una variable llamada frame, con estructura de bus. En ella se incluye la información de lo que ha captado el sensor en una "foto" (frame).
- **Test_Matleap:** Esta función toma un frame tras otro (ejecuta la función matleap_frame en bucle) durante 10 segundos, y devuelve la información obtenida por el sensor en cada uno de los frames.

Una vez se comprobó que la lectura del Leap Motion era correcta y precisa, se pudo proceder a integrarlo en el fichero de Simulink.

Las funciones incluidas en Matleap nos permiten ejecutarlas en el entorno de MATLAB, pero no en SIMULINK. Dado que el sistema correrá en un fichero de SIMULINK, será necesario obtener ejecutar la función matleap_frame continuamente. Para ello, se ha obtenido del TFG de Marta Menéndez [10] una **S-Function de segundo orden**. Las S-Functions permiten ejecutar un código escrito en lenguaje de MATLAB (fichero.m) indefinidamente en el entorno de SIMULINK. En nuestro caso, esta función se ejecutará una vez cada 20 milisegundos. Esto nos proporcionará 50 frames por segundo. Dado que el fabricante recomienda usar una frecuencia menor a 200 fps para evitar pérdidas de precisión, la frecuencia empleada no debería suponer ningún problema.

El principal problema de emplear una s-function radica en las salidas. Estas deberán tener una estructura fija (recordamos que las salidas del Leap Motion tienen una estructura dinámica), y estas no podrán tener más de una dimensión (vectores).

La función se llamará leapmotion.m, e instanciará al programa frame.m contenido en el conjunto de ficheros matleap. Por ello, será necesario que los ficheros de matleap estén presentes en la carpeta de trabajo (en el path) de MATLAB.

El cuerpo de esta función consta de varias instancias, de las que usaremos dos:

- **Setup**: define el bloque, su dimensión y las propiedades de sus puertos de salida (tipo de variable, dimensión…)

- **Outputs**: Obtiene frames del sensor mediante la función matleap_frame, inicializa todas las variables de salida, obtiene algunos parámetros de interés

(como los ángulos) mediante cálculos con las entradas del sensor (figura 16) y finalmente coloca los parámetros de salida en los puertos de salida.

La función Outputs comienza tomando un frame e inicializando todas las variables a cero. Como se ha comentado anteriormente, una S-function requiere el uso de variables de dimensión estática. Para conseguirlo, se inicializarán todas las variables a cero, y las dimensiones que se les darán serán las máximas que podrían alcanzar en una estructura variable (por ejemplo, para la variable is_left, que es una matriz fila que puede tener una o dos columnas, se inicializará a [0 0]. Por lo tanto, si el sensor no actualiza el valor de una variable (por ejemplo, cuando el sensor detecte solo una mano, los valores correspondientes a la segunda mano) mantendrán su valor anterior. Estos dos primeros pasos pueden observarse en la siguiente figura:

```
function Outputs(block)

frame=matleap_frame;
id=zeros(10,1);
position=zeros(10,3);
velocity=zeros(10,3);
direction=zeros(10,3);
palm_position=zeros(2,3);
is_right=zeros(2,1);
is_left=zeros(2,1);
angles=zeros(2,3);
yaw=zeros(2,1);
pitch=zeros(2,1);
roll=zeros(2,1);
open=zeros(2,1);
```

Figura 13. Toma de un frame de las variables.

Posteriormente, se introducen los valores de los parámetros medidos en variables locales intermedias. Estas variables serán empleadas para calcular las salidas. Esto puede verse en la siguiente figura:

```
if isempty(frame.pointables)== 0

    for i=1: min(length(frame.pointables),10)
            if (frame.pointables(i).is_finger == 1)
            id(i)=frame.pointables(i).id;
            position(i,:)=frame.pointables(i).position;
            velocity(i,:)=frame.pointables(i).velocity;
            direction(i,:)=frame.pointables(i).direction;
            end
     end
    for j=1: min(length(frame.hands),2)
        palm_position(j,:)=frame.hands(j).palm_position;
        is_right(j)=frame.hands(j).is_right;
        is_left(j)=frame.hands(j).is_left;
        d=frame.hands(j).direction;
        dn=frame.hands(j).palm_normal;
        yaw(j)=180/pi*atan(-d(1)/d(3));
        pitch(j)=180/pi*atan(-d(2)/d(3));
        roll(j)=180/pi*atan(-dn(1)/dn(2));
        open(j)=frame.hands(j).pinch_strength;
    end
end
angles=[yaw pitch roll];
```

Finalmente, se introducen las salidas en los puertos de salida. Debido a la limitación en la dimensión de los parámetros de salida, estos se deberán enviar en forma de filas o columnas, por lo que, tras deshacer las matrices de posición, velocidad y dirección en sus respectivas filas, necesitaremos un total de 19 variables de salida, que posteriormente se agruparán en Simulink mediante multiplexores. Las variables de salida pueden observarse en la siguiente figura:

```
block.OutputPort(1).Data = id;
block.OutputPort(2).Data = position(:,1);
block.OutputPort(3).Data = position(:,2);
block.OutputPort(4).Data = position(:,3);
block.OutputPort(5).Data = velocity(:,1);
block.OutputPort(6).Data = velocity(:,2);
block.OutputPort(7).Data = velocity(:,3);
block.OutputPort(8).Data = direction(:,1);
block.OutputPort(9).Data = direction(:,2);
block.OutputPort(10).Data = direction(:,3);
block.OutputPort(11).Data = palm_position(:,1);
block.OutputPort(12).Data = palm_position(:,2);
block.OutputPort(13).Data = palm_position(:,3);
block.OutputPort(14).Data = is_right;
block.OutputPort(15).Data = is_left;
block.OutputPort(16).Data = angles(:,1);
block.OutputPort(17).Data = angles(:,2);
block.OutputPort(18).Data = angles(:,3);
block.OutputPort(19).Data = open;
```

Figura 14. Salida de la S-function.

4.1.4 Pruebas y calibrado

Para garantizar un control estable y preciso, se ha estudiado

- **Posición.** Con el objetivo de conocer el área de operación del sensor, se han buscado, para los tres ejes X, Y y Z, los valores máximos y mínimos que se pueden conseguir de una forma estable (que el sensor no sufra pérdidas momentáneas), precisa y lineal. Esto se realizará añadiendo un scope a la variable "position", y evaluando los valores. La siguiente figura muestra la prueba que se ha realizado sobre la prueba para el eje Z, donde se aprecia que el rango de valores estables es el [150,700]. Tanto para el eje X como el eje Y, este rango es el [-350,+350].

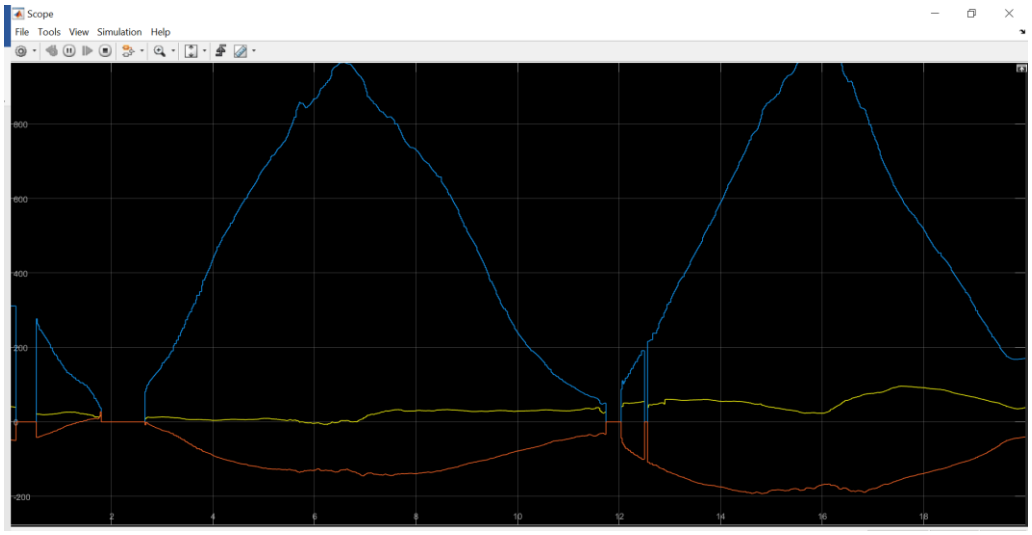


Figura 15. Prueba de rango del eje Z del sensor.

- **Orientación (ángulos).** Usaremos los ángulos de Euler que nos proporciona el sensor Leap Motion. Al igual que para cada una de las coordenadas de la posición, se determinará el rango estable de valores de cada uno de los ángulos mediante la prueba y la lectura por un scope. La siguiente figura muestra la prueba del ángulo correspondiente a la guiñada (yaw).

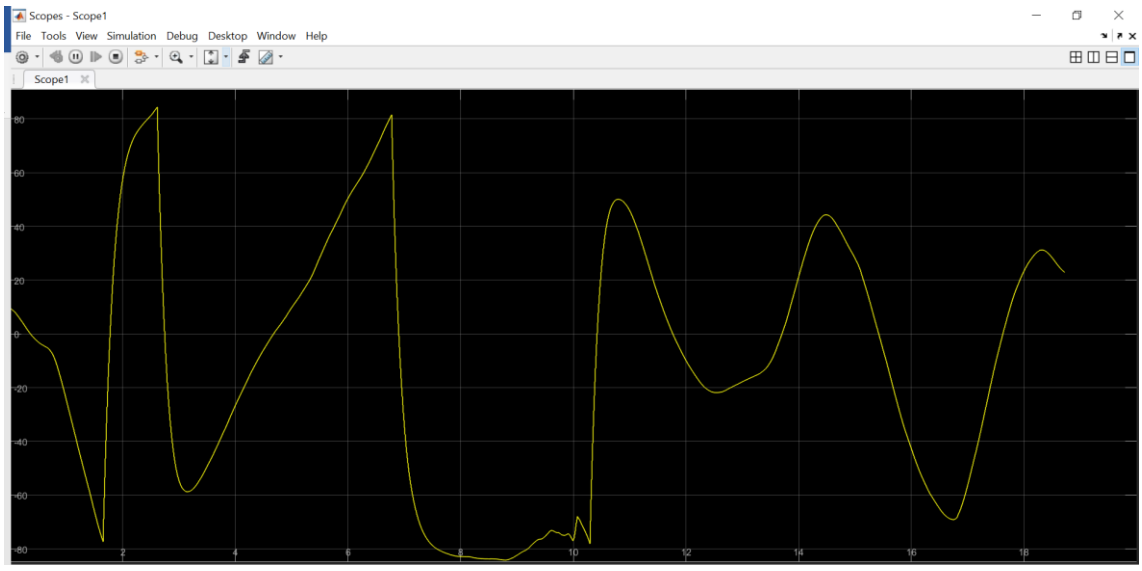


Figura 16. Prueba del ángulo de guiñada del Leap Motion.

Los tres ángulos medidos siguen el mismo patrón: Al aplicar el filtro digital, su medida se vuelve muy estable entre -80 y 80 grados. Un gran problema que se ha detectado es que **el valor de los ángulos "salta" desde -80 a 80 y viceversa.** Este problema será muy difícil de solventar ya que este "salto" no es directo (lo cual se podría arreglar por medio de código), sino que toma todos los valores comprendidos entre los extremos (como si se girase la mano en sentido contrario muy rápido).

- **Apertura de la mano (is_closed):** Se comprobará la sensibilidad y precisión del sensor para detectar cómo de cerrada está la mano. Esta estabilidad será tan importante como las demás, ya que, para proporcionar seguridad y confianza en la herramienta, esta debe ser lo más precisa y suave posible. La siguiente figura muestra las medidas del sensor cuando se abre y cierra la mano de una forma lenta y controlada. Las medidas son lo suficientemente precisas como para no necesitar un filtro digital. Se hará, sin embargo, la media entre la muestra actual y la anterior en el código de control para evitar inestabilidad y añadir suavidad a su movimiento.

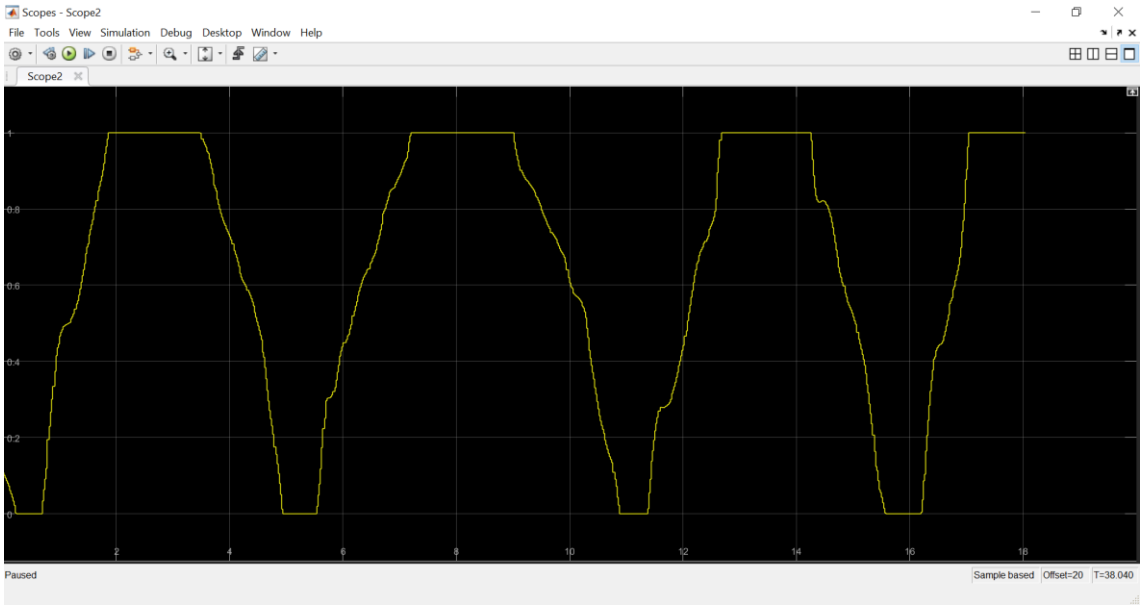


Figura 17. Prueba de la señal is_closed del sensor Leap Motion.

4.2 Control: Máquina de estados

4.2.1 Transición de estados

Para contar con funciones de encendido y apagado del brazo, se ha diseñado una máquina de estados de Moore. Esta contará con los siguientes estados:

0. **OFF:** El brazo está apagado en una posición de reposo.
1. **RESET:** El brazo continúa en la posición de reposo, se inicializa el sistema.
2. **ON:** Control sobre el brazo, que sigue la trayectoria deseada.
3. **Apagado:** El brazo vuelve a la posición de reposo.

La máquina de estados se implantará en el "lado" de la Raspberry (Raspi control system) y leerá y escribirá en el bus de control, que cuenta con variables relacionadas a los estados (estado actual, estado siguiente), así como con las entradas del sensor y las referencias de salida (todas ellas enviadas por MAVLINK. La posición de reposo se ha obtenido por tanteo, tratando de conseguir que el centro de masas del brazo estuviese lo más centrado posible y los servos en una posición angular centrada, permitiendo un arranque suave. Esta posición es la correspondiente a los siguientes ángulos:

- $\Theta_1 = 0^o$
- $\Theta_2 = 235^o$
- $\Theta_3 = 270^o$
- $\Theta_4 = 0^o$
- $\Theta_5 = 0^o$
- $\Theta_6 = 0^o$
- Pinza: Indiferente (180º = cerrada, 90º = abierta).

Las condiciones para los cambios de estado serán las siguientes:

1. **Paso de OFF a reset:** La mano 1 (controladora) se encuentra cerrada sobre el sensor.

```
if is_closed == 1
    NEXT_STATUS = uint8(1); % INIT
end
```

2. **Paso de reset a ON**: Transcurren cinco segundos.

```
timer = timer + SAMPLING_TIME;
if timer >= 5
      NEXT_STATUS = uint8(2); % CONTROL
end
```

3. **Paso de ON a Apagado:** Se detecta una segunda mano sobre el sensor.

```
if rl_hands == [1 1]
    NEXT_STATUS = uint8(3); %APAGADO
end
```

4. **Paso de Apagado a OFF:** No se detecta ninguna mano en el sensor.

```
if rl_hands == [0 0]
    NEXT_STATUS = uint8(0);
end
```

La transición 3 (de apagado) se realiza de esta manera como medida de seguridad, ya que permite el apagado sin tener que mover el brazo que está controlando al robot. Además, no se permitirá el encendido hasta que no se haya despejado el espacio sobre el sensor. Esto permitirá, junto con la condición de encendido, conocer con seguridad con qué mano se controlará el robot y evitar cualquier problema que pudiese ocurrir si se detecta una segunda mano en el entorno del sensor.

La asignación de salidas se realizará en otra función paralela a la máquina de estados, en el bloque llamado inverse_kinematics. Las salidas se actualizarán de la siguiente manera:

- **Estado ON:** Se realizarán los cálculos de cinemática inversa para obtener los ángulos de los servos en función de la entrada de posición y orientación proveniente del sensor. Este vector (llamado angles) se enviará de vuelta a la estación de control.
- **Resto de estados:** Los ángulos de la posición de reposo están fijados, si bien sería posible establecer una dupla de posición y orientación de reposo y realizar los cálculos de cinemática inversa. Dado que estos siempre obtendrían el mismo resultado (conocido) y suponen una carga computacional constante, se ha decidido enviar los ángulos directamente.

4.2.2 Control de salidas

Las salidas dependen exclusivamente del estado en el que se encuentre el sistema (máquina de Moore).

La asignación de salidas se realizará en otra función paralela a la máquina de estados, en el bloque llamado inverse_kinematics. Las salidas se actualizarán de la siguiente manera:

- **Estado ON:** Se realizarán los cálculos de cinemática inversa para obtener los ángulos de los servos en función de la entrada de posición y orientación proveniente del sensor. Este vector (llamado angles) se enviará de vuelta a la estación de control.
- **Resto de estados:** Los ángulos de la posición de reposo están fijados, si bien sería posible establecer una dupla de posición y orientación de reposo y realizar los cálculos de cinemática inversa. Dado que estos siempre obtendrían el mismo resultado (conocido) y suponen una carga computacional constante, se ha decidido enviar los ángulos directamente.

4.3 Control: Bloque de cálculo

4.3.1 Primera aproximación

El bloque de cálculo obtiene los valores de ángulos de cada servo para obtener una posición y orientación determinadas. Esto se realiza mediante cálculos de cinemática inversa. Como se ha explicado previamente en el apartado de modelado del brazo, el procedimiento que realiza este método es el siguiente:

1. Cálculo de los ángulos θ_1, θ_2 y θ_3 mediante cinemática inversa.
2. Obtención de la orientación de la muñeca mediante cinemática directa
3. Cálculo de los ángulos θ_4, θ_5 y θ_6 mediante cinemática inversa, para la que será necesaria la orientación de la muñeca.
4. Cálculo del ángulo θ_7 en función de is_closed.

En el cálculo de los tres primeros ángulos, se empleará la configuración “mano izquierda” y “Codo arriba”, ya que las limitaciones físicas de movimiento del brazo hacen imposible que adopte las posiciones correspondientes a las configuraciones “brazo derecho” y “codo abajo”.

4.3.2 Segunda aproximación: muñeca

El procedimiento explicado anteriormente tiene un problema, y es el constante cambio de configuración de la muñeca a la hora de realizar un control por mimetización. Como puede observarse en las ecuaciones correspondientes a la muñeca, una misma posición y orientación de la pinza puede obtenerse mediante dos ternas de ángulos. (Además, existe un punto (aquél en el que Z3 y Z6 son coincidentes) en el que hay infinitas soluciones. Este último no supone ningún problema dado que MATLAB añade errores de cálculo que, aunque son muy pequeños (del orden de 1e-10), bastan para que no se pueda conseguir una alineación perfecta). Los cálculos cuentan con dos configuraciones posibles (en las que θ_4 y θ_6 de una configuración son suplementarios respectivamente a

sus homólogos de la otra configuración, y θ_5 es opuesto a su homólogo). Cuando se produce un paso por el punto en el que Z3 y Z6 son coincidentes, se produce un cambio de configuración. Este cambio ha dado problemas a la hora de implantar, ya que se producían vibraciones y temblores en el brazo robótico en el momento del cambio.

La solución a la que se ha llegado es emplear directamente los ángulos de entrada del sensor, en concreto los ángulos de alabeo y cabeceo. Ya que el robot pretende mimetizar la posición de la mano, los ángulos Yaw Pitch y Roll que debe mostrar el robot deberán ser los mismos que aquellos que presenta el brazo humano. Por lo tanto, se asignará a θ_4 el valor del cabeceo, y a θ_6 el valor del alabeo de la mano.

El valor de los ángulos se introduce en el vector SERVO_ANGLES, que es enviado de vuelta a la estación de control.

4.4 Creación de señales para los servos

Una vez tenemos las referencias en grados en la estación de control, se obtendrá, mediante las ecuaciones obtenidas a partir de la genérica presente en el apartado 2.2 del abstracto:

$$Factor\ de\ Servicio = 500 + \frac{2000}{\text{Rango servo}(180\ o\ 270)} * \theta[º]$$

Estas ecuaciones se obtendrán mediante sumadores y ganancias, como puede observarse en la siguiente figura:

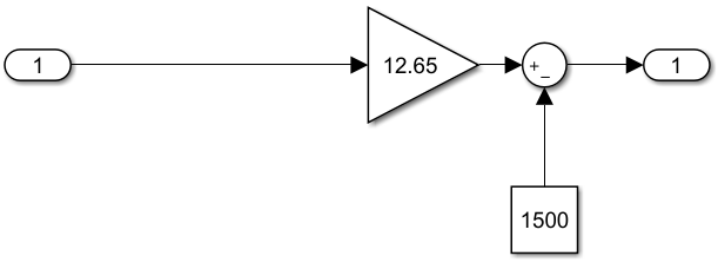


Figura 18. Modelado de servos.

Puede observarse que, para este servo, un ángulo de 0 grados se obtendrá con un ancho de pulso de 1500us, y, por tanto, su rango de recorrido es [-90º, +90º]

Además, se controlará la velocidad a la que se mueven los servos, para que su movimiento sea lo más suave posible. Esto se hará mediante un filtro digital muy sencillo, de la siguiente ecuación:

$$Y[k] = y[k-1] + signo * max(abs(velocidad\ límite), abs(y[k] - y[k-1])$$

Por lo tanto, se limita la cantidad que pueda variar la referencia de ángulo a un máximo, que se determinó mediante prueba y error.

Un caso especial fue el ángulo θ_3, ya que el mecanismo de palanca por el que se obtenía su posición angular (puede observarse en la siguiente figura) generaba una dependencia del ángulo θ_2. Por lo tanto, fue necesario elaborar una tabla con diferentes referencias en ambos ángulos para determinar una ecuación para θ_3. Dado que la relación entre estos dos ángulos no es lineal, y, para simplificar los cálculos se ha aproximado a una función

lineal, existe una falta de precisión en este ángulo. Este será un factor determinante para el uso de otro modelo de brazo robótico en futuros proyectos si se desea aumentar la precisión.

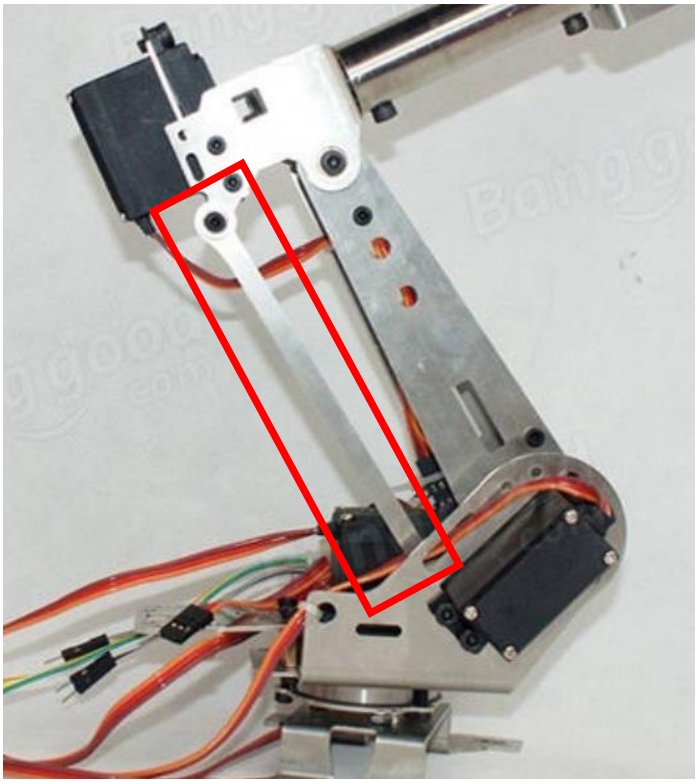
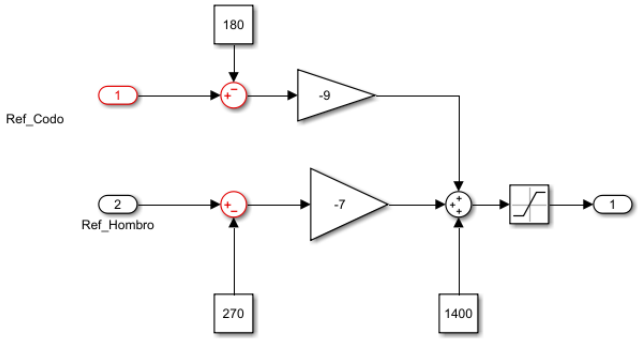


Figura 19. Movimiento de theta 3.



Estos anchos de pulso se envían a la tarjeta PCA9685, lo que supone el final de la etapa de toma de información y control.

4.5 Tarjeta PCA9685

La tarjeta PCA9685 se empleará para generar las señales PWM que se enviarán a cada uno de los servos. La comunicación de esta tarjeta con la Raspberry se realizará por el protocolo I2C. Dado que la tarjeta pertenece a la familia Fast-Mode plus (datasheet), admiete frecuencias de hasta 1MHz en la comunicación.

Esta tarjeta cuenta con un oscilador interno de 25MHz, que podrá obtener cualquier frecuencia múltiplo de esta mediante el pre-escalador. Dado que la frecuencia (o update rate) que queremos obtener es 50Hz, por medio de la fórmula:

$$prescale\ value = round\left(\frac{osc\_clock}{4096 \times update\_rate}\right) - 1$$

Figura 20. Fórmula cálculo prescalador

Obtenemos un valor de 121 para el registro del prescalador.

Lo primero que haremos será inicializar la tarjeta de los servos, es decir, introducir la configuración de funcionamiento, empleando los registros MODE 1 (00), MODE 2 (01) y prescaler. Los pasos que se deben seguir son los siguientes:

1. Poner a 1 el bit de sleep mode en el registro MODE 1
2. Poner a 1 el bit de configuración en tótem en el registro MODE 2
3. Indicar el valor del prescalador en su correspondiente registro
4. Revertir el paso 1 para volver al modo normal en el registro MODE 1

Para comprobar que la inicialización se realizaba con éxito, se comprueba que los cuatro pasos se han realizado (status del bloque I2C write = 0) y se manda como señal de enable al siguiente paso. Esto se puede observar en las siguientes figuras:

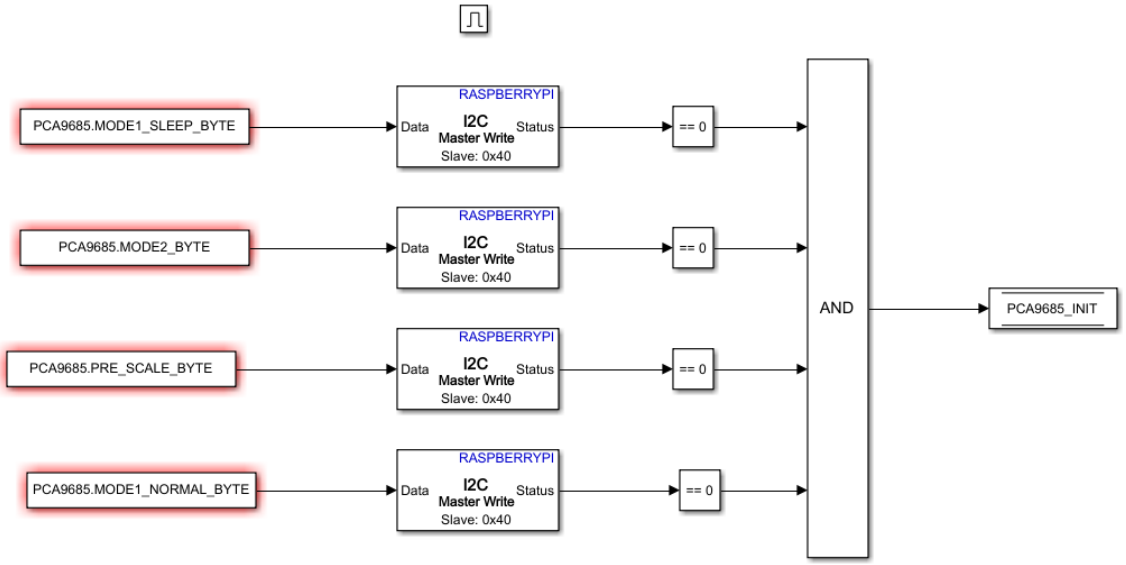


Figura 21. Proceso de inicialización de la tarjeta PCA9685

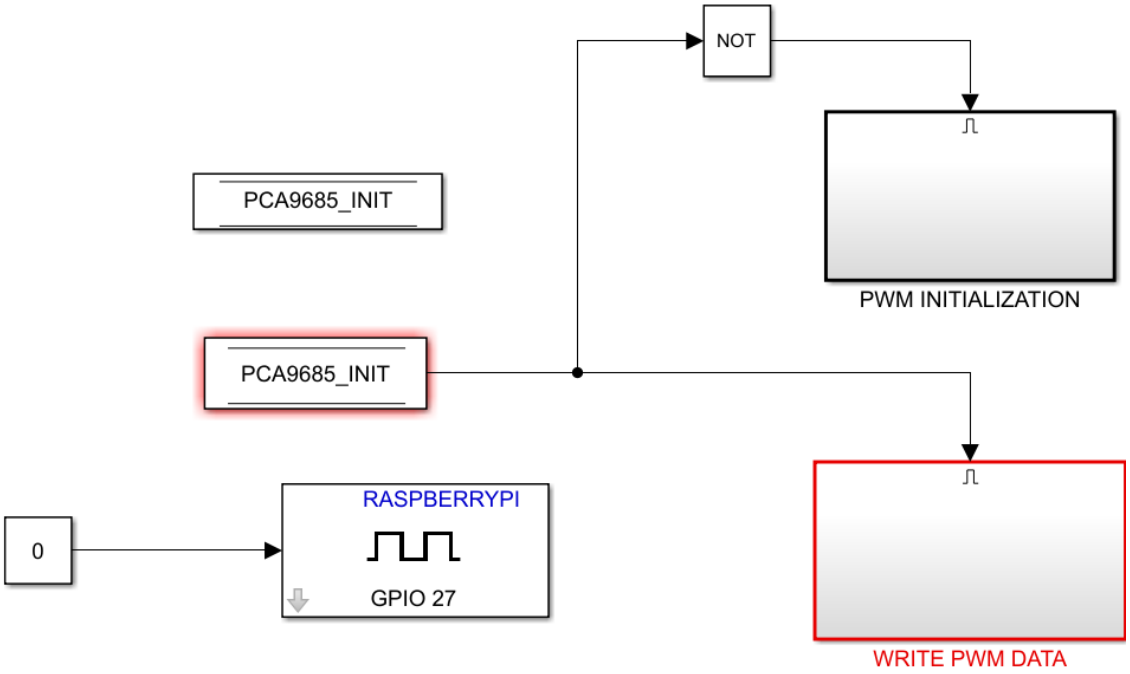


Figura 22. Comprobación de inicialización de la tarjeta PCA9685

Una vez se ha inicializado la tarjeta, se mandarán los anchos de pulso de cada señal (SERVO_RAW) y la frecuencia de estas (PWM_FREQ) a un bloque de cálculo que generará los mensajes que se enviarán a la tarjeta.

Las señales PWM se generan a partir de dos valores representados en 2 bytes cada uno:

- Tiempo de ON: paso de '0' lógico a '1' lógico
- Tiempo de OFF: Paso de '1' lógico a '0' lógico

Ambos números representan el "tanto por 4096" de ciclo en el que se produce el cambio, por lo que, para nuestro caso, un valor de 0x00 en estos registros equivaldrá a t=0, mientras que el valor 0xFF equivaldrá a t=20ms (el período de nuestra PWM).

Como no queremos ningún tipo de offset en el tiempo de encendido (es innecesario ya que la comunicación con los servos es asíncrona) la señal T_ON se fijará a 0, es decir, queremos que la señal cambie de 0 a 1 nada más comenzar un ciclo. Por lo tanto, el tiempo de off será el ancho de pulso deseado partido por la frecuencia de la onda (lo que nos daría un tanto por uno) multiplicado por 4096 (para expresar este valor en 2 bytes).

La información se enviará en formato uint8, es decir, en bytes. En el script empleado (diseñado para controlar 8 señales PWM) se convertirá cada dupla T_ON, T_OFF correspondiente a cada señal a formato uint16. Esto nos proporcionará una matriz de 16 filas (8 canales x 2 variables) y dos columnas (ya que un número uint16 contiene 2 bytes). Finalmente, se escribirá cada byte en una fila independiente, obteniendo de esta manera la matriz final PWM_I2C (32x1, uint8).

Esta última matriz será la que se envíe a la tarjeta PCA9685, a los registros correspondientes a cada una de las señales (la variable PWM.Address incluida en el fichero CONFIG_PCA9685.m contiene las direcciones de cada uno de los 16 registros de señales PWM).

5. COMUNICACIONES

En este apartado se explicará de qué manera se ha establecido cada una de las distintas comunicaciones entre los dispositivos que se han utilizado. Como se puede observarse en la siguiente figura, existen tres medios físicos por los que se establece comunicación:

- **USB:** El sensor Leap Motion se comunica con el ordenador mediante un cable USB
- **Wi-Fi:** La comunicación entre la estación base (PC) y la Raspberry se realiza mediante protocolo UDP. Para establecer esta conexión es necesario que los dos dispositivos se encuentren conectados al mismo enrutador Wi-Fi. Se ha utilizado un teléfono móvil, que ejerce el papel de enrutador. Los mensajes enviados por este medio irán codificados según el protocolo de comunicaciones MAVLink, que se explicará a continuación.
- **I_2C:** La Raspberry enviará la información a la tarjeta PCA9685 mediante una comunicación I_2C.

Figura 23. Diagrama de conexiones y comunicaciones.

Dado que la lectura de los datos enviados por el Leap Motion ha sido explicada en el apartado correspondiente al sensor, limitaremos el apartado a la explicación de los protocolos MAVLink e I2C.

5.1 Comunicación Wi-Fi

5.1.1 Herramientas de envío y recibimiento de mensajes

Tanto la biblioteca de procesamiento de señales digitales como la biblioteca de Raspberry para SIMULINK contienen bloques para enviar y recibir información por UDP. Se ha elegido este protocolo ya que la pérdida de un mensaje no supone un gran problema, y proporciona una mayor rapidez de comunicación. Para configurar estos bloques debemos primero decidir qué puertos del PC se utilizarán para la transmisión en cada uno de los sentidos. Estos dos puertos deben ser diferentes entre sí. En este caso, se han usado los puertos 25001 para la comunicación PC-Raspberry y 25000 para la comunicación inversa. Además, debemos incluir la máxima longitud que podrá tener un mensaje en bytes. Dado que el mensaje MAVLink más largo (véase el siguiente apartado) es de 44+45 = 89 bytes, fijaremos un tamaño máximo de 95 bytes para dar un margen.

5.1.2 Protocolo MAVLink [11]

MAVLink (Micro Air Vehicle) es un protocolo de comunicación muy usado en el ámbito de los drones y, en general, cuando el ancho de banda disponible no es muy grande. La principal razón por la que se usa es porque es muy ligero y eficiente. Sus principales características son:

- MAVLink1 solo emplea 8 bytes de overhead, y su versión mejorada MAVLink2 usa 14 (dado que incrementa la seguridad y es más versátil).
- Es un protocolo muy seguro, ya que proporciona maneras de detectar pérdida o corrupción de los paquetes (secuencia de paquete y CRC).
- Admite la programación en diferentes lenguajes y en hasta 255 sistemas entrelazados (un byte de identificación de sistema).
- Permite comunicaciones entre distintos sistemas, y entre componentes de un mismo sistema (un byte de identificación de componente de sistema).

La versión que emplearemos será MAVLink1, por lo que nuestro mensaje tendrá la siguiente estructura:

```
uint8_t magic;               ///< protocol magic marker
uint8_t len;                 ///< Length of payload
uint8_t seq;                 ///< Sequence of packet
uint8_t sysid;               ///< ID of message sender system/aircraft
uint8_t compid;              ///< ID of the message sender component
uint8_t msgid;               ///< ID of message in payload
uint8_t payload[max 255];    ///< A maximum of 255 payload bytes
uint16_t checksum;           ///< X.25 CRC
```

Se emplearán los ficheros ya existentes con mensajes predefinidos, que han sido proporcionados por el director del proyecto.

La simple estructura de los mensajes en este protocolo nos permitirá crear mensajes. Para ello, cada estación (ordenador y raspberry) tendrán codificadores y decodificadores, en los que se definirá la estructura de cada mensaje (cómo descomponer todos los bytes del payload en los distintos parámetros que se están enviando).

En el cuerpo (payload) del mensaje, dado que los mensajes se envían en formato uint8 (bytes), necesitaremos 4 bytes (o un número uint32) para representar los números en coma flotante. Estos son todos los valores menos las variables rl_hands e is_left, is_right.

En nuestro caso, la comunicación entre la estación de control y la raspberry es bilateral. Para cada mensaje enviado, se definirá el mensaje (orden y longitud de cada parámetro) tanto en el codificador del remitente como en el decodificador del receptor.

Esta comunicación se llevará a cabo mediante coders y decoders. En los coders (emisor) se moverán variables del bus de control al bus de MAVLink, y en los decoders (receptor) se realizará el proceso opuesto. Dado que en este proyecto la comunicación será bilateral, ambos entornos contarán con codificadores y decodificadores.

Los mensajes que se desean enviar son declarados el fichero de configuración del sistema, dentro del apartado de MAVLink.

```
%% MAVLINK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UART transfer rate (bits/s)
MAVLINK_TRANSFER_RATE = 115200;
fprintf('MAVLINK TRANSFER RATE = %d bits/s\n',MAVLINK_TRANSFER_RATE);
% MAVLINK buffer size
MAVLINK_BUFFER_SIZE = floor((MAVLINK_TRANSFER_RATE/12)*SAMPLING_TIME);
fprintf('MAVLINK BUFFER SIZE = %d bytes\n',MAVLINK_BUFFER_SIZE);
% List of parameters
PARAM_LIST_LEN = 1;
cd ../SOFTWARE_COMPONENTS/MAVLINK
% SYSTEM MAVLINK configuration
disp('MAVLINK FOR CONTROL SYSTEM:')
MAVLINK_SYS_INIT = CONFIG_MAVLINK([190],MAVLINK_BUFFER_SIZE,1,0,PARAM_LIST_LEN);
% PC MAVLINK configuration
disp('MAVLINK FOR PC CONTROL STATION:')
MAVLINK_PC_INIT = CONFIG_MAVLINK([191 36],MAVLINK_BUFFER_SIZE,0,0,PARAM_LIST_LEN);
cd ../../CONFIGURATION
```

Los mensajes que se enviarán en este sistema serán:

- De la estación de control a la raspberry
  - Valores medidos por el sensor: Se ha creado un mensaje en el número 191. Este constará de 36 bytes de payload:
    - Posición de la mano: 3xFloat (12 bytes)
    - Ángulos de orientación: 3xFloat (12 bytes)
    - Apertura de la mano: Float (4 bytes)
    - RL_Hands: 2xFloat
  - Orden de la estación de control a los servos: Este mensaje ya está predefinido en el número 36 (SERVO_OUTPUT_RAW). Consta de:
    - Tiempo desde encendido: Float
    - Ancho de pulso en microsegundos para 16 canales de servos: números enteros (integers), enviados cada uno en dos bytes.
    - Número de puerto: Se emplea por si se quieren controlar más de 16 servos. Valdrá uno para este proyecto, pero se dejará puesto para futuros desarrollos. Es un número entero

    El tamaño del cuerpo del mensaje será de 4+(16*2)+1 = 37 bytes.

- De la Raspberry a la estación de control:
  - Ángulos de los servos en grados: Se creará un mensaje en el hueco 190, muy similar al número 36 usado anteriormente. Este mensaje tendrá un cuerpo de 68 bytes, dado que incluye:
    - Tiempo desde inicialización: Float.
    - Ángulos en grados: 16*Float.

A continuación, se muestra a modo de ejemplo la codificación y decodificación del mensaje 191, correspondiente al envío de la información obtenida por el sensor de la estación de control a la Raspberry:

Codificación: En el codificador de la estación de control, se introduce el valor del índice del mensaje, MESSAGE_INDEX (que será 192, ya que posteriormente será

reducido en una unidad). Posteriormente, se introducen las variables (previamente cambiadas a columnas de uint8) en la matriz PAYLOAD. Es importante observar que, dado que el mensaje se enviará en una matriz columna, los datos ordenados en forma matricial serán descompuestas en sus valores individuales. Finalmente, se crea el mensaje final en la rutina MAVLINK_CODER, introduciendo los datos necesarios. El valor del número de secuencia se irá aumentando en una unidad cada vez que se envíe el mensaje, volviendo a 0 cuando se alcance 255.

```
if MESSAGE_ID(192) % LEAP_MOTION ( #191 ).
    MESSAGE_INDEX = uint8(192);
    % Payload definition
    PAYLOAD = [ typecast(single(CONTROL.INPUT.LEAP_MOTION.POSITION(1)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.POSITION(2)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.POSITION(3)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.ANGLES(1)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.ANGLES(2)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.ANGLES(3)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.IS_CLOSED),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.RL_HANDS(1)),'uint8')'
                typecast(single(CONTROL.INPUT.LEAP_MOTION.RL_HANDS(2)),'uint8')'

    % MAVLINK message codification
    [MESSAGE,MESSAGE_LEN] = MAVLINK_CODER(MESSAGE_INDEX-1,SYSTEM_ID,...
                            COMPONENT_ID,PAYLOAD,SEQUENCE_NUMBER,...
                            MAVLINK_MESSAGE_CRCS(MESSAGE_INDEX));
```

```
HEADER=[hex2dec('FE') ; SIZE_PAYLOAD ; SEQUENCE_NUMBER ; SYSTEM_ID ; COMPONENT_ID ; MESSAGE_ID];
```

```
    % CRC computation
    CRC_VALUE = uint16(65535);
    for nn=1:length(AUX)
        TMP = bitxor(AUX(nn),bitand(CRC_VALUE,uint16(255)));
        TMP4 = bitand(bitshift(TMP,4),uint16(255));
        TMP = bitxor(TMP,TMP4);
        TMP8 = bitshift(TMP,8);
        TMP3 = bitshift(TMP,3);
        TMP_4 = bitshift(TMP,-4);
        CRC_VALUE = bitxor(bitshift(CRC_VALUE,-8),bitxor(TMP8,bitxor(TMP3,TMP_4)));
        CRC_VALUE = bitand(CRC_VALUE,uint16(65535));
    end
    CRC1 = uint8(bitand(CRC_VALUE,uint16(255)));
    CRC2 = uint8(bitand(bitshift(CRC_VALUE,-8),uint16(255)));
    % Final MAVLink message
    MESSAGE = [uint8(HEADER(:)) ; uint8(PAYLOAD(:)) ; CRC1 ; CRC2];
    MESSAGE_LEN = uint8(length(MESSAGE));
    return
```

Figura 24. Creación de un mensaje MAVLink

5.2 Comunicación I2C [12]

La comunicación I2C (del inglés Inter-Integrated Circuit) es un protocolo de comunicación serie síncrona generalmente empleado con controladores que interactúan (leen o escriben) con una serie de dispositivos periféricos. En esta comunicación, uno de los dispositivos (generalmente el controlador) es el “maestro”, y el resto son los “esclavos”.

Además de las conexiones de alimentación y la tierra, la comunicación se lleva a cabo mediante dos pines: un reloj SCL que comparten maestro y esclavos (y que convierte la

comunicación en síncrona) y una línea de datos SCA, por la que se transmite el mensaje. Ambas líneas deben estar conectadas mediante una resistencia de pull-up a una línea de tensión alta. En la siguiente figura se muestra un ejemplo de conexionado para controlar 3 esclavos con un maestro:

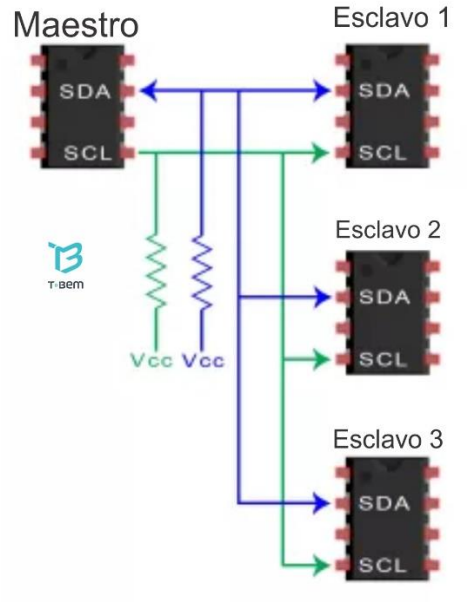


Figura 25. Conexionado para comunicación I2C

La estructura de los mensajes enviados por comunicación I2C se muestra en la siguiente figura:

Figura 26. Estructura de un mensaje I2C

En los instantes previos a la comunicación, ambas líneas SCA y SCL permanecen en estado alto. La comunicación comienza cuando la vía SCA pasa a nivel bajo. Tras la dirección del esclavo con el que se desea comunicarse, se determina con el bit read/write (1=write, 0=read) si se desea escribir en el esclavo o leer de él. Posteriormente, se envía el cuerpo del mensaje, con sus respectivos bits de ACK/NACK que comprueban si el mensaje se ha recibido con éxito. Una desventaja de este protocolo es que el tamaño del cuerpo del mensaje está limitado a 8 bits (ya que el esclavo debe confirmar el recibimiento de cada mensaje con un ACK).

En este proyecto, la Raspberry se comunicará con la tarjeta PCA9685 mediante comunicación I2C.

6. CONCLUSIONES Y FUTUROS DESARROLLOS

6.1 Trabajo realizado

Este proyecto ha combinado trabajo manual y trabajo con el ordenador, y cabe destacar que la mitad del proyecto se realizó en el extranjero. El robot vino completamente desmontado, y con unas instrucciones que dejaban bastante que desear (y que se encuentran en el anexo). Una vez montado el brazo y los servos, empleando un script que se comunicaba con la tarjeta PS2 para generar señales PWM, se estableció la comunicación con los servos. Finalmente, se modeló cada uno de los servos para adecuar la ecuación general para obtener el ancho de pulso necesario para una referencia angular deseada.

Dado que no se había realizado ningún proyecto con brazos robóticos previamente en ICAI, fue necesario estudiar de cero las ecuaciones de robótica de Denavit-Hartenberg.

En los primeros compases del proyecto, la intención era controlar el brazo robótico mediante la tarjeta de servos PS2 comunicada por un módulo UART a la estación de control. Sin embargo, a finales de mayo se decidió cambiar la comunicación por Wi-Fi hacia una Raspberry, y utilizar la tarjeta PCA9685 comunicada por I2C. Además, se decidió cambiar la estructura del proyecto a una modular, para así poder aprovechar partes de él en futuros proyectos.

Una vez se consiguió comunicarse con los servos, se empezó el estudio del sensor Leap Motion. Esto constituyó un gran problema, ya que el uso de mex-files requiere tener un compilador instalado. Se intentó descargar el compilador MinGW (Add-on para MATLAB), pero se dio por imposible tras un mes de errores y de conversaciones con el servicio técnico. Finalmente, se descargó la versión 2018A de MATLAB, la cual incorpora un parche que soluciona el error obtenido en la versión 2017A.

Tras comprobar que los archivos de Matleap eran capaces de comunicarse con el Leap Motion, se pasó a probar su funcionamiento en SIMULINK con la S-Function. Mediante scopes, se realizaron las pruebas de rango de funcionamiento fiable.

A la vuelta del intercambio se obtuvo, de parte del subdirector del proyecto, las ecuaciones generales de cinemática inversa para el robot PUMA estándar. Tras su estudio, se logró desarrollar las ecuaciones específicas para el brazo utilizado.

Tras comprobar que las ecuaciones estaban formuladas en el fichero correctamente, se desarrolló la estructura de buses y se definieron los mensajes en protocolo MAVLink, para establecer la comunicación entre la estación base y la Raspberry.

Una vez se conectó todo, se procedió al control del robot con el sensor para estudiar rango de movimiento, detectar problemas de cálculo y encontrar la posición de reposo. Finalmente, se desarrolló una máquina de estados para poder encenderlo y apagarlo.

6.2 Conclusiones del proyecto

Tras el trabajo realizado, se ha llegado a unas conclusiones positivas, ya que se han cumplido los objetivos del proyecto:

- Se ha creado una estructura de archivos que permite cualquier tipo de control, ya sea de posición (como el desarrollado en el proyecto) como de velocidad (lo que permitiría el uso de emisoras, o del propio sensor Leap Motion funcionando como un joystick tridimensional. El único fichero que sería necesario actualizar sería el de actualización de salidas de la máquina de estados (el bloque inverse_kinematics).
- Esta estructura permite, además, el uso de cualquier sensor (por ejemplo, Kinect, Play Station Move, Eye Toy…). La única condición impuesta es que el sensor debe proporcionar la posición y orientación de la mano, ya que son las salidas requeridas para ese módulo.
- Mediante un cambio en el módulo de parámetros del modelo (es decir, los parámetros de Denavit-Hartenberg del brazo), se puede controlar cualquier brazo de seis grados de libertad.
- A pesar de las imprecisiones y limitaciones debidas al uso de un brazo robótico destinado al mundo del hobby y de servos en lazo abierto, se ha conseguido un control de posición con una precisión más que aceptable.

Además, se ha concluido que el sensor Leap Motion es un sensor asequible de gran precisión y pequeño tamaño. Esto es esencial para la motivación del proyecto, que es fomentar su uso en ambientes más modestos. Además, debido a que el sistema de control se puede cambiar, con el mismo hardware se pueden desarrollar distintos controles.

6.3 Futuros desarrollos

Este proyecto puede servir como punto de partida para el control de brazos robóticos. El mayor problema que se ha encontrado es la falta de robustez y precisión del brazo, que requería revisiones y aprietes de tornillos cada media hora de uso. Con el fin de ampliar los resultados que se han obtenido en este proyecto, hay tres bloques en los que se puede explorar y mejorar:

- **Brazo robótico:** El uso de un robot más robusto mejoraría la precisión, el rango de movimiento estable y la carga que puede soportar. Esto haría posible la realización de numerosas actividades que este brazo y estos servos no pueden hacer debido a su debilidad y estructura. Cabe destacar que, en determinadas posiciones, el brazo no puede ni con su propio peso.
- **Sensor Leap Motion:** A pesar de que se piensa que este sensor es el más adecuado para controlar la posición de las manos (principal controlador del robot), hay numerosos sensores en el mercado (Kinect, PlayStation Move…) que pueden ser explorados e integrados, para, por ejemplo, controlar dos robots al mismo tiempo, uno con cada brazo.
- **Modos de control:** Un control de velocidad en lugar de posición dotaría a la mano de las funciones de un joystick tridimensional: Para aplicaciones que requieran de una gran precisión, se podría añadir a la máquina de estados un modo de control de velocidad con un aún mayor incremento del factor de escala

(por ejemplo: 10cm de movimiento de la mano se traducirían en una velocidad de 1cm/s).

- **Automatización de rutinas:** Una función que podría ser de gran utilidad sería la de contar con un modo de grabación de movimientos, en la que se grabase la secuencia de movimientos que realiza el usuario, para su posterior ejecución con solo pulsar un botón (por ejemplo, grabar los gestos que realiza el usuario para abrir una ventana, y que el robot pueda luego realizarlos sin necesidad de que el usuario tenga que repetir la rutina).

CONTROL DE UN BRAZO ROBÓTICO PARA EMULACIÓN DE LOS MOVIMIENTOS DE UNA MANO

PRESUPUESTO

1.RECURSOS EMPLEADOS

En este apartado se desglosarán los recursos empleados, ya sean de componentes principales del proyecto, como recursos humanos, y las distintas herramientas auxiliares que se han empleado durante todo el período de trabajo.

1.1 Componentes principales

Componentes	Cantidad
Brazo robótico	1
Servos MG995	1
Servos DS3218	3
Micro servos MG90S	2
Tarjeta PCA9685	1
Raspberry Pi Modelo 3B	1
Leap Motion 2.0	1

1.2 Herramientas físicas y de software

Componente	Cantidad	Horas de proyecto	Horas de uso al año
Cable micro USB	1	5	1000
Cable HDMI	1	1	3
Ordenador	1	500	1200
MATLAB+SIMULINK	1	400	1500
Fuente de alimentación	1	50	500
Microsoft Office	1	60	600
Servo tester	1	2	6
Router inalámbrico	1	50	800

2.RECURSOS HUMANOS. MANO DE OBRA

Actividad	Horas destinadas
Estudio	50
Creación de la estructura software	50
Montaje y mantenimiento brazo	50
Ensayos y pruebas	100
Redacción de documentos	50

3. COSTES UNITARIOS

En este apartado se expondrán los costes unitarios de cada uno de los componentes que se han empleado para este proyecto. Conjuntamente con el primer apartado, se obtendrán los costes totales del proyecto.

3.1 Componentes principales

Componentes	Precio (€/ud)
Brazo robótico	56
Servos MG995	4.40
Servos DS3218	9.12
Micro servos MG90S	3.09
Tarjeta PCA9685	2.83
Raspberry Pi Modelo 3B	35
Leap Motion 2.0	69.99

3.2 Herramientas físicas y de software

Componente	Precio (€/ud)
Cable micro USB	3
Cable HDMI	10
Ordenador	1200
MATLAB+SIMULINK	200
Fuente de alimentación	12.92
Microsoft Office	89
Servo tester	33.49
Router inalámbrico	20

3.3 Mano de obra

Actividad	Precio (€/h)
Estudio	40
Creación de la estructura software	40
Montaje y mantenimiento brazo	30
Ensayos y pruebas	50
Redacción de documentos	50

4. SUMAS PARCIALES

En este apartado se obtendrán los costes totales por bloque, empleando los datos contenidos en los dos apartados anteriores.

4.1 Componentes principales

Componentes	Cantidad	Precio	Total
Brazo robótico	1	56	56
Servos MG995	1	4.4	4.4
Servos DS3218	3	9.12	27.36
Micro servos MG90S	2	3.09	6.18
Tarjeta PCA9685	1	2.83	2.83
Raspberry Pi Modelo 3B	1	35	35
Leap Motion 2.0	1	6.99	6.99
		Total	138.76€

4.2 Herramientas físicas y de software

Componente	Cantidad	Precio (€/ud)	Horas de proyecto	Horas de uso al año	Amortización anual	Total
Cable micro USB	1	3	5	1000	25%	0.00375
Cable HDMI	1	10	1	3	25%	0.8333333
Ordenador	1	1200	600	1200	25%	150
MATLAB+SIMULINK	1	200	400	1500	25%	13.333333
Fuente de alimentación	1	12.92	50	500	25%	0.323
Microsoft Office	1	89	60	600	25%	2.225
Servo tester	1	33.49	2	6	25%	2.7908333
Router inalámbrico	1	20	50	800	25%	0.3125
					TOTAL	169.822€

4.3 Mano de obra

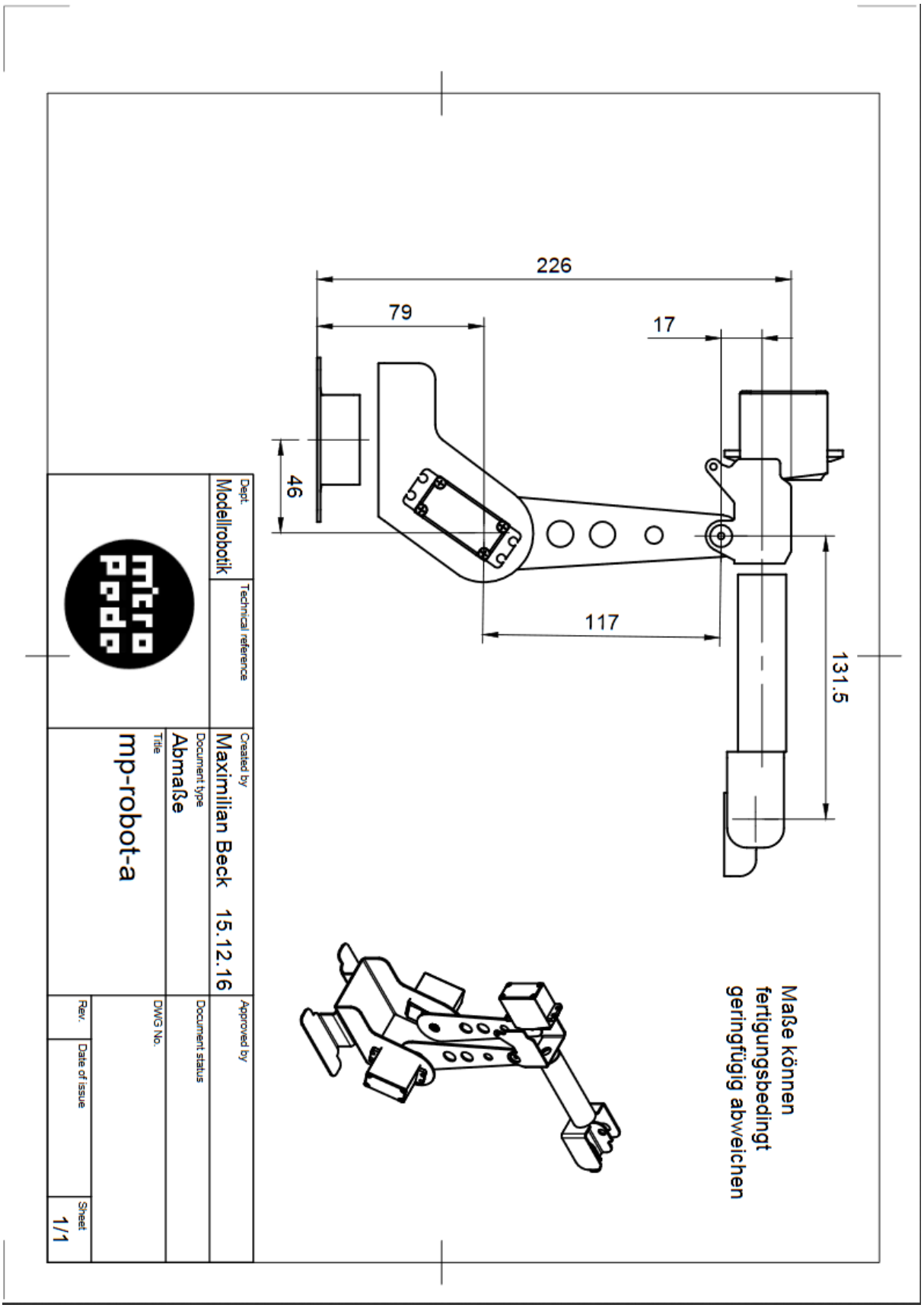
Actividad	Precio (€/h)	Precio (€/h)	Total(€)
Estudio	40	40	1600
Creación de la estructura software	40	40	1600
Montaje y mantenimiento brazo	30	30	900
Ensayos y pruebas	50	50	2500
Redacción de documentos	50	50	2500
		TOTAL	9100€

5. PRESUPUESTO GENERAL

Sumando los tres valores obtenidos en el apartado anterior, obtenemos el presupuesto (coste) total que ha supuesto el proyecto.

Bloque	Coste (€)
Componentes principales	138.76
Herramientas físicas y de software	169.82
Mano de obra	9100
TOTAL	9408.58€

ANEXO I: PLANOS BRAZO ROBÓTICO



6.BIBLIOGRAFÍA

[1] A. Alfageme, «Conoce al 'supercirujano' de las manos de acero,» 15 Noviembre 2016. [En línea]. Available: https://elpais.com/elpais/2016/11/01/talento_digital/1478021936_889288.html. [Último acceso: 29 08 2018].

[2] «Dobot Magician Overview,» Dobot, [En línea]. Available: https://www.dobot.cc/dobot-magician/product-overview.html. [Último acceso: 20 08 2018].

[3] A. Colgan, «How does the Leap Motion Controller work?,» Leap Motion Blog, 9 08 2014. [En línea]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. [Último acceso: 10 07 2018].

[4] Wikipedia, «Leap Motion - Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Leap_Motion#History. [Último acceso: 20 07 2018].

[5] T. Foster, «Will these guys kill the computer interface as we know it?,» Popular science, 22 July 2013. [En línea]. Available: https://www.popsci.com/technology/article/2013-07/will-these-guys-kill-computer-interface-we-know-it. [Último acceso: 1 08 2018].

[6] K. Hay, «Designing the Leap Motion Controller,» Blog Leap Motion, 3 April 2013. [En línea]. Available: http://blog.leapmotion.com/designing-leap-motion-controller/. [Último acceso: 18 July 2018].

[7] G. Leal, «Owi arm controlled by a LEAP Motion,» [En línea]. Available: https://hackaday.io/project/28551-owi-arm-controlled-by-a-leap-motion#menu-description. [Último acceso: 25 04 2018].

[8] J. Belda, «Leap Motion (II): Principio de funcionamiento,» ShowLeap Blog, 04 05 2015. [En línea]. Available: blog.showleap.com/2015/05/leap-motion-ii-principio-de-funcionamiento/. [Último acceso: 15 01 2018].

[9] J. Perry, «Git Hub,» 2013. [En línea]. Available: https://github.com/jeffsp/matleap. [Último acceso: 16 02 2018].

[10] M. M. Botella, «CONTROL DE NAVEGACIÓN DE UN CUADRICÓPTERO MEDIANTE GESTOS,» *Trabajo de fin de grado curso 2017/2018. Escuela técnica superior de ingeniería ICAI.*, p. 125, Agosto 2018.

[11] «MAVLink Developer Guide,» [En línea]. Available: https://mavlink.io/en/. [Último acceso: 20 08 2018].

[12] M. Morales, «Fundamentos del protocolo I2C,» Teslabem, 2017 Febrero 2017. [En línea]. Available: https://teslabem.com/learn/fundamentos-del-protocolo-i2c-aprende/. [Último acceso: 8 Agosto 2018].