Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ÁREA DE TECNOLOGÍA ELECTRÓNICA

BRAZO ROBÓTICO CONTROLADO CON GESTOS MANUALES

D. JAVIER MARTÍNEZ BECERRA
TUTOR: D. RAMÓN RUBIO GARCÍA

FECHA: Junio 2023

ÍNDICE

TABLA DE ILUSTRACIONES

1. Introducción

1.1. IDENTIFICACIÓN DEL PROYECTO

Título	Control de brazo robótico mediante gestos manuales
Tutor	Ramón Rubio García
Autor	Javier Martínez Becerra
Fecha	Junio, 2023

1.2. ALCANCE

El siguiente Trabajo Fin de Grado consistirá en el diseño e implementación del sistema de control para un brazo robótico a partir de los gestos manuales realizados por el usuario.

1.3. MOTIVACIÓN

Esta idea de proyecto surge a partir de diversas fuentes y artículos en los que se pueden ver las diversas tecnologías y campos de conocimientos que abarca un brazo robótico, desde la estructura mecánica del mismo hasta los distintos algoritmos y métodos para su control.

Durante este proceso de recogida de información e investigación en este campo se ha comprobado que existen numerosos ejemplos y proyectos a cerca de brazos robóticos controlados mediante distintos algoritmos de control o modelos cinemáticos complejos, pero no tantos en los que el usuario pueda controlar el brazo robótico de forma sencilla sin más que mover su brazo.

Partiendo de esa base se cree que este proyecto puede ser una buena opción para trabajar con muy diversas tecnologías en el campo de la electrónica y la automática e incluso con algunas partes del campo de la ingeniería mecánica, a la vez que se aporta un material que puede servir de apoyo para acercar las tecnologías de forma fácil y vistosa a instituciones educativas y a la sociedad en general de forma que se motive a los estudiantes a formarse y desarrollarse en este ámbito.

1.4. ANTECEDENTES

Como se ha comentado en el punto anterior parte del interés en la realización de este proyecto es acercar la tecnología a la sociedad en general, y más concretamente a las instituciones educativas.

El diseño de este brazo robótico surge como idea para complementar el proyecto del "Rover Asturiosity" desarrollado en la Cátedra MediaLab de la Universidad de Oviedo.

MediaLab es un laboratorio universitario de tecnología y diseño que surge de la colaboración entre el Ayuntamiento de Gijón, Gijón Impulsa y la Universidad de Oviedo. Su sede se encuentra en la Escuela Politécnica de Gijón y tiene por objetivo acercar las nuevas tecnologías a las personas.

Dentro de este espacio de trabajo surge el proyecto del "Rover Asturiosity". Este proyecto consiste en la recreación de un astromóvil marciano o Rover marciano. Un Rover es un vehículo motorizado diseñado para moverse mediante ruedas sobre la superficie del planeta Marte. Además, dispone de diversos sensores y tecnologías que le permiten explorar y estudiar la superficie de dicho planeta. Este proyecto tiene por objetivo diseñar y desarrollar un prototipo de un robot de este tipo para un uso educativo, con fines de acercar las nuevas tecnologías a colegios e institutos con un proyecto vistoso que sirve de demostración de diversas tecnologías de forma vistosa y educativa a la vez. Este proyecto se ha ido desarrollando a lo largo del tiempo por varios residentes del espacio MediaLab mediante la implementación de diversas tecnologías en él tales como, accionamientos electrónicos, comunicaciones, sistemas de almacenamiento de energía, sensores, etc. Este proyecto está en continuo desarrollo mediante la implementación de nuevas funcionalidades y tecnologías sobre el prototipo básico.

Es aquí donde surge la idea del brazo robótico. Además de otras tecnologías que se le están implantando como visión por computador o sensores de diversos tipos se plantea la idea de colocarle encima un brazo robótico con intención de simular de la forma más rigurosa posible el modelo real de un Rover marciano el cual habitualmente cuenta con un brazo robótico. O simplemente complementando las funcionalidades que se pueden enseñar desde el Rover con algunas nuevas a partir del brazo.

La integración del brazo en el Rover no forma parte de este proyecto. Pero se podría hacer posteriormente ya que como se ha comentado anteriormente el proyecto del Rover se realiza de forma modular y está en constante proceso de desarrollo y mejora.

Partiendo de esta base, inicialmente se plantea el diseño mecánico del brazo robótico, esta tarea se ha llevado a cabo por un estudiante de Ingeniería Mecánica de la Escuela Politécnica de Gijón en años anteriores.

1.5. ESTRUCTURA MECÁNICA

Como se ha comentado se parte de un diseño mecánico del brazo realizado por un antiguo alumno de la Escuela Politécnica de Ingeniería de Gijón. Es un brazo diseñado para un funcionamiento con fines educativos o recreativos por lo que no podrá levantar grandes pesos o realizar tareas que requieran de mucha precisión.

Este brazo ha sido fabricado en su totalidad mediante impresión 3D. Es una estructura con 6 grados de libertad que le permiten alcanzar una gran variedad de configuraciones y posiciones en el espacio. Se compone de 5 eslabones principales:

- Eslabón 1: encargado de rotar toda la estructura, permitiendo así que el brazo llegue a distintas posiciones en un radio de 360º. Accionado por un motor paso a paso.
- Eslabones 2, 3, 4: encargados de subir o bajar adaptando distintas configuraciones y permitiendo así llegar a los distintos puntos espaciales al brazo. Todos ellos accionados por servomotores.
- Eslabón 5: eslabón final (herramienta) encargado de realizar la labor para la que esté diseñado el brazo robótico, en este caso una pinza. Accionado por un servomotor.

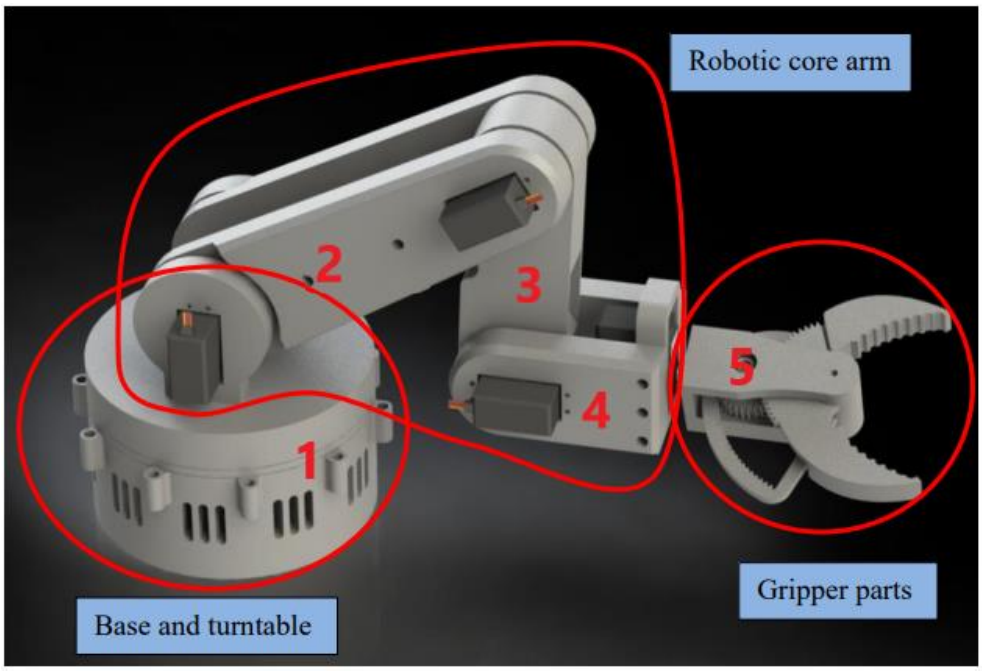


Ilustración 1. Estructura del brazo robótico.

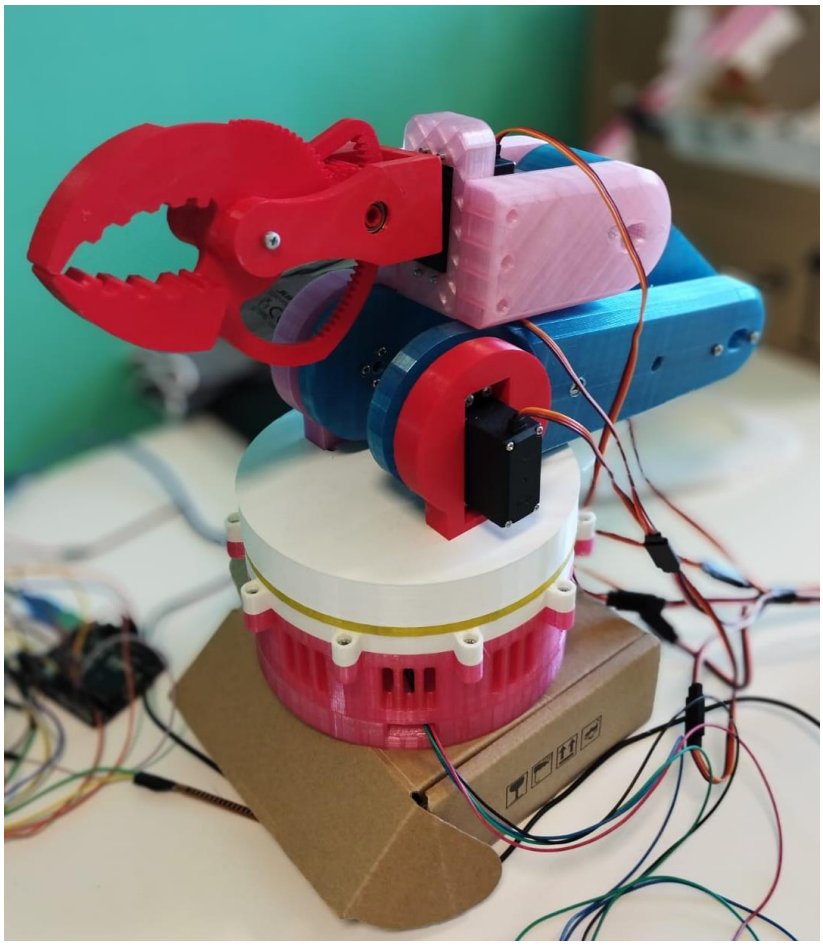


Ilustración 2. Estructura brazo real

1.6. OBJETIVO

En este punto es donde entra el trabajo desarrollado por este proyecto, el cual, partiendo de esta base de estructura mecánica pretende conseguir el control del movimiento de dicha estructura a partir de diversos componentes electrónicos y el código necesario para su programación.

El objetivo final del trabajo será conseguir que el brazo robótico mueva sus distintas articulaciones en función de los movimientos que el usuario realice con la mano, los cuales serán captados por sensores de distinto tipo.

Para conseguirlo se utilizarán diversas tecnologías y componentes dentro del campo de la tecnología electrónica tales como, sensores, accionamientos electrónicos y módulos de comunicación. Además del lenguaje de programación correspondientes para el control de dichos componentes.

1.7. ESTRUCTURA DEL DOCUMENTO

El presente Trabajo Fin de Grado está compuesto por los siguientes documentos:

- Memoria.
- Anexos.
    - Planificación temporal.
    - Códigos.

2. Estado del arte

2.1. ROBÓTICA

La robótica es una disciplina técnica que combina la ingeniería mecánica, la electrónica y la informática para diseñar y construir robots capaces de realizar tareas de forma autónoma o semiautónoma. Estos robots se basan en sistemas complejos de sensores, actuadores y algoritmos de control que les permiten interactuar con el entorno y llevar a cabo tareas específicas de manera precisa y eficiente.

Uno de los aspectos fundamentales de la robótica es el control de movimiento, que implica la planificación de trayectorias y la programación de los movimientos del robot para que sean precisos y fluidos. Para ello, se utilizan algoritmos de control de retroalimentación que permiten corregir los errores y ajustar los movimientos del robot en tiempo real.

Otro aspecto importante de la robótica es la percepción, que se refiere a la capacidad del robot para entender el entorno y detectar objetos, obstáculos y otros elementos relevantes. Para ello, se utilizan diferentes tipos de sensores, como cámaras, sensores de distancia, micrófonos y sensores táctiles, que permiten al robot recopilar información sobre el entorno y ajustar su comportamiento en consecuencia.

La interacción humano-robot también es un área importante de la robótica, y se refiere a la capacidad del robot para comunicarse y colaborar con las personas. Para ello, se utilizan interfaces de usuario intuitivas y sistemas de detección de voz y gestos que permiten al robot interactuar de forma natural con los humanos. Esta idea es la que tiene por objetivo y en la que se basa el presente TFG.

En la actualidad, la robótica se aplica en una amplia variedad de campos, desde la industria y la fabricación, hasta la medicina, la exploración espacial y el entretenimiento. Los avances en la inteligencia artificial y el aprendizaje automático están permitiendo el desarrollo de robots cada vez más sofisticados y capaces de adaptarse y aprender de su entorno, lo que abre nuevas posibilidades en términos de automatización y colaboración entre humanos y robots.

2.1.1. TIPOS DE ROBOTS

Se pueden diferenciar los distintos robots en función de algunos factores con el fin de clasificarlos. Según su cronología:

- 1ª Generación: Los robots de primera generación son los primeros robots que se desarrollaron y comercializaron. Estos robots se caracterizan por ser simples y limitados en su capacidad para realizar tareas, y se utilizaban principalmente en la industria para realizar trabajos repetitivos y peligrosos.

Los robots de primera generación se basan en sistemas mecánicos y neumáticos y utilizan una programación secuencial para realizar tareas específicas. Por lo general, se les programaba para realizar una tarea determinada en un ciclo repetitivo, y no podían adaptarse a cambios en el entorno o realizar tareas diferentes sin ser reprogramados. En general, los robots de primera generación eran muy rígidos en su comportamiento y no eran capaces de aprender o adaptarse a situaciones nuevas.

Además, eran muy costosos y requerían de una gran cantidad de mantenimiento y reparación para mantener su funcionamiento. A pesar de sus limitaciones, los robots de primera generación representaron un gran avance en la automatización de la industria y la mejora de la seguridad en el lugar de trabajo. Su uso en la fabricación permitió aumentar la eficiencia y reducir los costos de producción, lo que impulsó el crecimiento de la industria.

Estos robots se caracterizan por ser simples y limitados en su capacidad para realizar tareas, pero representaron un gran avance en la automatización de la industria y la mejora de la seguridad en el lugar de trabajo.

- 2ª Generación: los robots de segunda generación utilizan sistemas electrónicos y de control de retroalimentación, lo que les permite ser más precisos y flexibles en su comportamiento. Los robots de segunda generación se caracterizan por tener sistemas de control más avanzados y sensores más sofisticados, lo que les permite adaptarse mejor al entorno y a situaciones cambiantes. Por ejemplo, pueden detectar obstáculos y ajustar su trayectoria para evitarlos, o ajustar su velocidad y posición en respuesta a cambios en la carga o el peso de los objetos que manipulan.

  Además, los robots de segunda generación son más programables que los de primera generación, lo que les permite realizar tareas más complejas y variadas. Se pueden programar para realizar diferentes tareas en diferentes secuencias, y también pueden recibir entradas de sensores y ajustar su comportamiento en consecuencia. Otra característica importante de los robots de segunda generación es su capacidad para interactuar con humanos de forma más natural. Se pueden integrar interfaces de usuario más intuitivas y sistemas de detección de voz y gestos, lo que les permite comunicarse y colaborar mejor con las personas.

  Estos robots son una evolución de los robots de primera generación, que utilizan sistemas electrónicos y de control de retroalimentación más avanzados y sensores más sofisticados. Son más precisos, flexibles y programables que los de primera generación, y se utilizan en una amplia variedad de aplicaciones en la industria, la medicina y otras áreas.

- 3ª Generación: estos robots se caracterizan por su capacidad para aprender y adaptarse a situaciones nuevas y desconocidas, gracias a sistemas de inteligencia artificial y de aprendizaje automático.

  Los robots de tercera generación se basan en sistemas de sensores y cámaras más avanzados, y en algoritmos de inteligencia artificial que les permiten procesar y analizar grandes cantidades de datos. Estos sistemas les permiten detectar y reconocer objetos y situaciones, y tomar decisiones en tiempo real basadas en esta información.

  Pueden aprender y adaptarse a su entorno y a las situaciones cambiantes de forma autónoma, sin necesidad de ser programados específicamente para cada tarea. Esto les permite realizar tareas más complejas y variadas, y también les permite mejorar su rendimiento y eficiencia con el tiempo.

  También tienen capacidad para colaborar con humanos de forma más estrecha y natural. Se pueden integrar sistemas de reconocimiento de voz y gestos, y se pueden utilizar interfaces de usuario más intuitivas, lo que facilita la comunicación y la colaboración entre humanos y robots.

  Actualmente se utilizan en campos que van desde la industria y la fabricación hasta la medicina y la exploración espacial. Se utilizan para tareas que requieren inteligencia, adaptabilidad y autonomía, como la exploración de terrenos desconocidos, la realización de cirugía robótica y la producción de bienes personalizados y adaptados a las necesidades específicas de cada cliente.

Actualmente también existen algunas fuentes que dividen los tipos de robots en cinco generaciones, diferenciando ligeramente más entre los comportamientos y características de cada robot.

Ilustración 3. Evolución robots

También se puede diferenciar los robots a partir de su función o modo de funcionamiento:

- Robots articulados: los robots articulados son robots que tienen una serie de articulaciones en su estructura que les permiten moverse con libertad y precisión. Estos robots constan de varios brazos o manipuladores que están conectados a través de articulaciones, lo que les permite realizar movimientos complejos y precisos. Cada brazo está formado por varios segmentos que se pueden mover independientemente uno del otro. Cada segmento está conectado a través de articulaciones, que pueden ser rotativas o lineales, y los movimientos de los segmentos se controlan a través de motores eléctricos o hidráulicos.

  Los robots articulados se utilizan en una amplia variedad de aplicaciones industriales, como la soldadura, la pintura, el ensamblaje y la manipulación de materiales.

- Robots móviles: son robots que tienen la capacidad de moverse de forma autónoma y sin restricciones en su entorno. A diferencia de los robots industriales que se mueven en una posición fija y realizan tareas específicas, los robots móviles están diseñados para moverse de forma autónoma en un entorno dinámico y realizar diversas tareas según las necesidades.

  Los robots móviles pueden ser de varios tipos, como los robots terrestres, los robots aéreos o los robots submarinos, según el medio en el que se desplacen. Estos robots pueden utilizar diferentes métodos de locomoción, como ruedas, patas, hélices o propulsores.

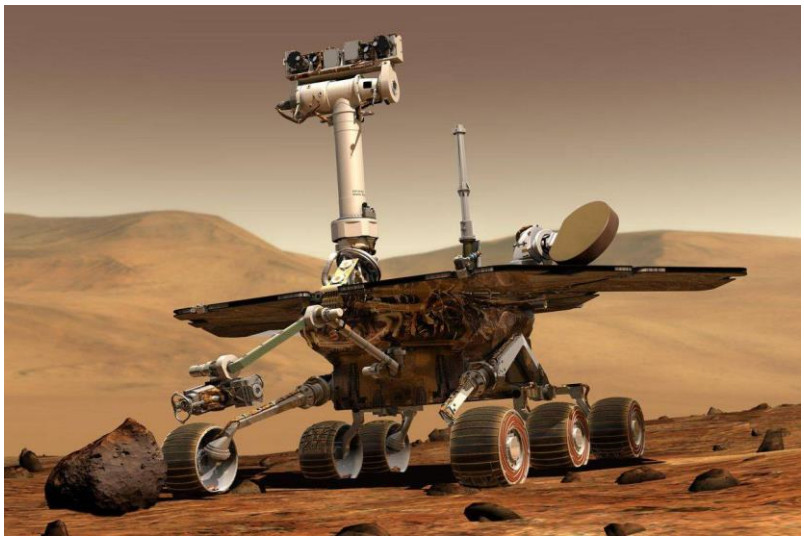


Ilustración 4. Robot móvil

- Robots humanoides: robots diseñados para tener una apariencia y movimientos similares a los de los seres humanos. Estos robots suelen tener una cabeza, torso, brazos y piernas, y están diseñados para imitar el comportamiento humano en términos de movimientos y habilidades. Los robots humanoides utilizan tecnología avanzada para imitar la locomoción, la percepción y la manipulación

humana. Esto incluye la utilización de sensores, cámaras y procesamiento de imagen para la percepción del entorno, así como sistemas de control de movimiento para replicar los movimientos humanos.

Tienen usos en el campo de la investigación, la educación el entretenimiento y la asistencia en tareas domésticas.

Ilustración 5. Robot humanoide

- Cobots: los cobots, o robots colaborativos, son robots diseñados para trabajar junto a los seres humanos en un entorno de trabajo compartido. A diferencia de los robots industriales tradicionales, que se utilizan para realizar tareas peligrosas o repetitivas sin la presencia de trabajadores humanos, los cobots están diseñados para trabajar en equipo con los trabajadores humanos para mejorar la eficiencia y seguridad del trabajo.

  Los cobots están diseñados para ser fáciles de programar y configurar, lo que permite a los trabajadores humanos controlar su funcionamiento y realizar tareas colaborativas. Los cobots suelen estar equipados con sensores y sistemas de seguridad que les permiten interactuar con los seres humanos de forma segura y detectar cuando hay un riesgo potencial de colisión.

  Los cobots se utilizan en una amplia variedad de aplicaciones, como la manipulación y ensamblaje de piezas, el embalaje y paletizado de productos, la inspección de calidad y la asistencia en tareas de fabricación.

Ilustración 6. Robot colaborativo

- Híbrido: estos robots combinan una serie de funcionalidades y características combinadas de algunos de los anteriores por lo que no se pueden clasificar específicamente como ninguno ellos.

Se podrían hacer más clasificaciones de los tipos de robots en función de otras características como, por ejemplo, su estructura, pero no se profundizará más en ello.

Una vez hecha una panorámica general de la robótica y de los tipos de robots que hay, se podría considerar que el robot desarrollado en este proyecto es un robot híbrido, ya que, es un brazo robótico, lo cual, cumple las características para estar en la categoría de robots articulados y a su vez es un robot colaborativo, ya que, está pensado para trabajar conjuntamente con los seres humanos, de hecho, en este caso particular este cobot no podría funcionar sin la actuación del movimiento de control por parte del ser humano.

Por ello se profundizará un poco más en esta categoría de robot, particularizando en los brazos robóticos con los que se trabaja en este proyecto.

2.2. BRAZOS ROBÓTICOS

Se podría definir un brazo robótico como un autómata programable, cuyo objetivo es el de imitar el funcionamiento de un brazo humano, con las ventajas de un robot en cuanto a diversos factores, como fuerza o precisión. Habitualmente este tipo de dispositivos disponen de siete segmentos y seis articulaciones, proporcionando al dispositivo seis grados de libertad que le permite realizar movimientos y configuraciones diversas similares a las que se pueden realizar con un brazo humano. Estas articulaciones simularían las correspondientes de un brazo humano, el cual dispone de siete grados de libertad. El extremo del brazo robótico a menudo está equipado con herramientas especializadas, como pinzas, cámaras, láseres o sensores, que permiten que el robot realice una tarea específica. El brazo robótico puede ser programado para realizar movimientos precisos y repetitivos, lo que lo convierte en una herramienta valiosa en la industria de la fabricación, la medicina, la exploración espacial y muchas otras

aplicaciones donde se requiere una alta precisión y velocidad en las tareas realizadas. A la vista de sus características se podría clasificar este tipo de robot en la categoría de robot articulado.

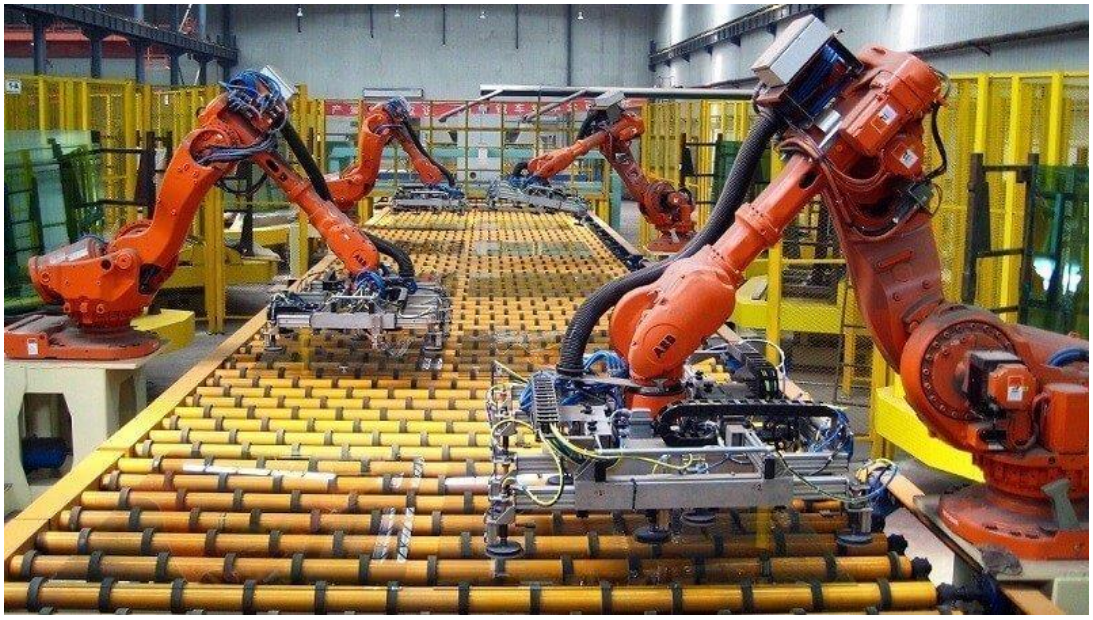


Ilustración 7. Brazos robóticos industriales

A la hora de seleccionar un brazo robótico se deben tener en cuenta dos factores importantes: los elementos que conforman el robot y sus parámetros característicos, ya que, en función de estos parámetros el robot será apto o no apto para realizar según que tareas.

El primer factor considera las partes o subsistemas de un brazo robótico: manipulador, elemento terminal, actuadores, sistemas de control, sistemas de percepción.

El segundo factor se refiere al conjunto de características del robot, tales como: grados de libertad, precisión, velocidad, capacidad de carga, entre otras.

Ilustración 8. Subsistemas de un brazo robótico

2.2.1. ELEMENTOS DE UN BRAZO ROBÓTICO

- Manipulador: constituye la estructura mecánica del robot a excepción del elemento terminal. Está formado por un conjunto de eslabones en serie unidos mediante articulaciones que permiten al robot adoptar una serie de configuraciones en el espacio para realizar la tarea que le sea programada mediante algún algoritmo de control o directamente por un operador mediante algún dispositivo de control remoto.

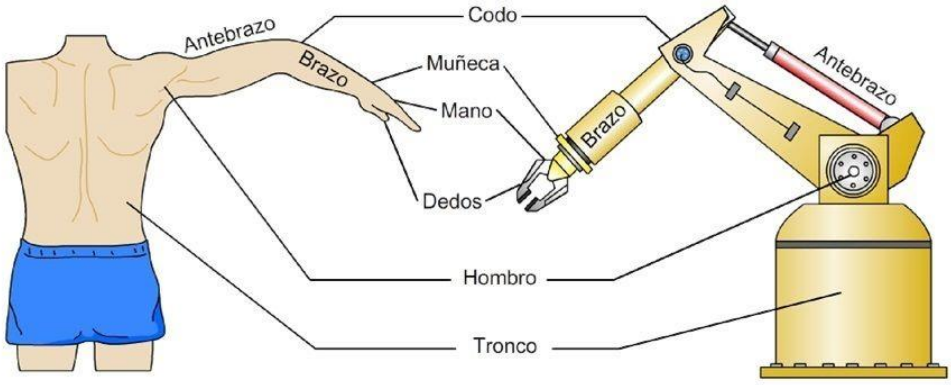


Ilustración 9. Morfología de un brazo robótico

En función del tipo de articulaciones que a su vez caracterizará las posibilidades de movimiento del brazo se pueden dividir los brazos robóticos en:

- Robot cartesiano: es un tipo de robot industrial que se mueve en coordenadas cartesianas (X, Y, Z) en un espacio tridimensional. Está compuesto por tres pares de ejes lineales que se mueven a lo largo de los tres ejes cartesianos y su diseño se asemeja a una mesa con un brazo perpendicular que se mueve hacia arriba y hacia abajo.

El movimiento del robot cartesiano es muy preciso y se puede programar para seguir una trayectoria específica. Por lo general, se utilizan en aplicaciones de ensamblaje, corte, soldadura y pintura, en las que se requiere una alta precisión en los movimientos realizados. También son fáciles de programar y pueden ser reprogramados para diferentes tareas, lo que los hace muy versátiles.

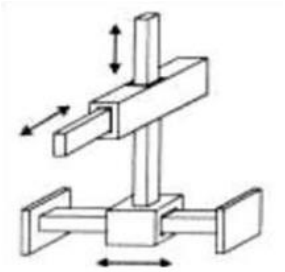


Ilustración 10. Robot cartesiano

- Robot cilíndrico: es un tipo de robot que utiliza un sistema de coordenadas cilíndricas para mover su brazo. Estos robots tienen una estructura tubular con una base giratoria y un brazo que se extiende hacia afuera, con una junta de rotación en el extremo del brazo. El movimiento de este tipo de robots se realiza a lo largo de los ejes de coordenadas cilíndricas, es decir, en la dirección radial, de elevación y de rotación.

  Los robots cilíndricos se utilizan en aplicaciones en las que se requiere una gran movilidad en un espacio limitado, como en la manipulación de objetos en entornos pequeños y la automatización de procesos.

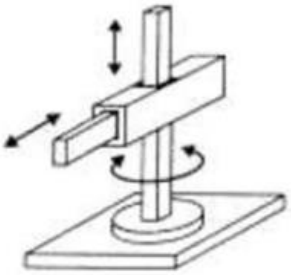


Ilustración 11. Robot cilíndrico

- Robot esférico/polar: utiliza un sistema de coordenadas esféricas para mover su brazo. Estos robots tienen una estructura esférica o semiesférica con tres articulaciones que se extienden desde la base de la esfera. El extremo del brazo está equipado con una herramienta de trabajo y puede moverse en cualquier dirección en un espacio de trabajo esférico.

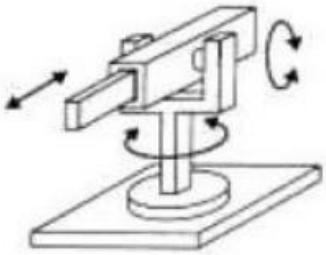


Ilustración 12. Robot esférico

- Robot SCARA: un robot SCARA es un tipo de robot industrial que utiliza un sistema de coordenadas cartesianas para mover su brazo. La palabra SCARA significa "Selective Compliance Assembly Robot Arm". Estos robots tienen una estructura de brazo articulado con cuatro ejes de movimiento, que se mueven en coordenadas cartesianas X, Y, Z, con un eje rotativo adicional para la orientación de la herramienta. Debido a su diseño, estos robots pueden trabajar en un rango muy amplio de áreas de trabajo.

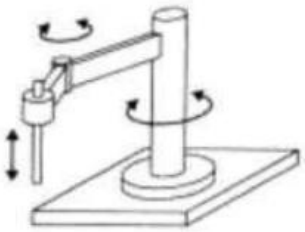


Ilustración 13. Robot SCARA

- Robot articulado: estos robots tienen varias articulaciones que se asemejan a la estructura de un brazo humano, con varios grados de libertad que permiten una gran movilidad y flexibilidad en su movimiento. Este es el tipo de robot que se utilizará para el desarrollo de este proyecto.

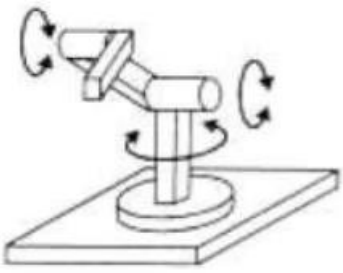


Ilustración 14. Robot articulado

En cuanto a los componentes de los brazos robóticos se pueden diferenciar:

- Elemento terminal: este es el dispositivo que se acopla en el último eslabón del brazo articulado. Se seleccionará en función de la tarea que se quiera realizar.

Un mismo brazo robótico podría tener varios elementos terminales intercambiables para realizar distintas tareas.

- Sistemas motrices: los sistemas motrices son los componentes mecánicos y eléctricos que permiten que el robot se mueva y realice tareas específicas. Los sistemas motrices incluyen motores, reductores, sistemas neumáticos, hidráulicos.

  Los motores son los componentes eléctricos que generan la fuerza necesaria para mover los brazos del robot. Los motores más comunes utilizados en los robots son los motores de corriente continua y los motores de paso a paso.

  Los reductores son dispositivos mecánicos que reducen la velocidad y aumentan el torque del motor. Estos componentes son necesarios porque el brazo robótico para determinadas aplicaciones necesita un alto torque para levantar objetos pesados y una velocidad más baja para una mayor precisión en la realización de tareas.

- Sistemas de control: son los componentes que permiten programar y controlar el movimiento del robot. Estos sistemas se dividen en dos categorías principales: control de bajo y alto nivel.

  El control de bajo nivel se refiere al hardware físico que se utiliza para controlar los motores y sensores del robot. Este tipo de control incluye las tarjetas de controladores de motor, sensores y actuadores, que se conectan al hardware del robot.

  El control de alto nivel se refiere al software utilizado para programar el movimiento del robot y definir sus comportamientos. Este tipo de control utiliza lenguajes de programación como C++, Python, Java y otros para crear programas que controlan el robot. Los programas de control de alto nivel también pueden incluir bibliotecas de software que proporcionan funciones comunes para el control de robots, como la planificación de trayectorias y la detección de obstáculos.

- Sistemas de percepción: estos son los componentes que permiten al robot comprender y adaptarse al entorno que lo rodea. Estos sistemas incluyen sensores y algoritmos de procesamiento de señales que permiten al robot detectar objetos, medir distancias y capturar información visual.

  Los sensores más comunes utilizados en los robots incluyen cámaras, sensores de proximidad, sensores táctiles, sensores de fuerza y sensores de presión.

2.2.2. PARÁMETROS DE UN BRAZO ROBÓTICO

Hay una serie de parámetros que caracterizan el brazo robótico y que son de vital importancia a la hora de su diseño, ya que determinarán si el robot es apto o no para realizar determinadas tareas de forma correcta.

- Grados de libertad: el número de grados de libertad (DOF, por sus siglas en inglés) de un brazo robótico se refiere a la cantidad de movimientos independientes que puede realizar el brazo. Cada grado de libertad representa una dirección de movimiento en la que el brazo puede moverse.

  Un brazo robótico típico puede tener entre 2 y 7 grados de libertad. Los grados de libertad más comunes son:

  - Rotación de la base: el brazo puede girar sobre su base.
  - Elevación del hombro: el brazo puede levantarse y bajarse.
  - Flexión del codo: el brazo puede doblarse y estirarse.
  - Rotación de la muñeca: la mano puede girar alrededor de la muñeca.
  - Flexión de la muñeca: la mano puede doblarse hacia arriba y hacia abajo.
  - Abducción / aducción: la muñeca puede moverse hacia afuera y hacia adentro.
  - Apertura / cierre de la pinza: la mano puede abrir y cerrar para agarrar objetos.

  Cada grado de libertad adicional aumenta la complejidad y capacidad del brazo robótico, lo que permite realizar tareas más complejas y precisas. A la vez que dificulta su control.

- Capacidad de carga: se refiere a la cantidad máxima de peso que el brazo puede manipular de manera segura y efectiva. La capacidad de carga depende del diseño y la construcción del brazo, así como del tamaño y tipo de herramienta de agarre que se utiliza en la extremidad final del brazo.

- Precisión: es la capacidad del brazo para repetir movimientos con una precisión y exactitud consistentes. La precisión es esencial en muchas aplicaciones donde se requiere una alta precisión en la colocación de objetos, como en la fabricación de componentes electrónicos o en la cirugía.

  La precisión de un brazo robótico se ve afectada por varios factores, incluyendo la rigidez y la estabilidad mecánica del brazo y la resolución de los sensores utilizados para medir la posición y la orientación del brazo, y la precisión y la

resolución de los controladores utilizados para controlar el movimiento del brazo.

La precisión de un brazo robótico se suele medir en términos de la desviación estándar de la posición y la orientación del brazo en relación con una posición y orientación deseada. Los fabricantes de brazos robóticos suelen proporcionar especificaciones de precisión para sus productos, que pueden incluir la precisión absoluta (la diferencia máxima permitida entre la posición deseada y la posición real del brazo) y la repetibilidad (la capacidad del brazo para repetir el mismo movimiento con precisión consistente).

Es importante tener en cuenta que la precisión de un brazo robótico puede verse afectada por factores ambientales, como las vibraciones, la temperatura y la humedad, por lo que es importante controlar y minimizar estos factores en la medida de lo posible para lograr la mayor precisión posible.

- Velocidad: la rapidez con la que el brazo robótico puede realizar los movimientos con una precisión aceptable. La velocidad de un brazo robótico se mide típicamente en términos de la velocidad de punta, que es la velocidad máxima a la que la extremidad final del brazo puede moverse. Los fabricantes de brazos robóticos suelen proporcionar especificaciones de velocidad para sus productos, que pueden incluir la velocidad y la aceleración máximas del brazo. Al igual que la precisión se puede ver afectada por diversos factores ambientales.

- Programabilidad: se refiere a las distintas posibilidades de control que el brazo robótico soporta. Esto viene determinado tanto por el hardware como por el software elegido para su control.

2.2.3. MÉTODOS DE CONTROL DE UN BRAZO ROBÓTICO

Como se comenta anteriormente una de las principales subpartes dentro de un brazo robótico es el sistema de control. Como el desarrollo del presente Trabajo Fin de Grado consiste principalmente en el diseño e implementación de un sistema de control para el brazo robótico se profundizará un poco más en las distintas posibilidades que existen en cuanto al control de brazos robóticos.

Existen diversos parámetros que se pueden controlar en el funcionamiento de un brazo robótico, tales como, la velocidad, la fuerza o la trayectoria que sigue el mismo, pero habitualmente la variable sobre la que se quiere tener control es la posición. En concreto, la posición del actuador final del brazo, que será el encargado de realizar la tarea. Normalmente los brazos robóticos profesionales o industriales implementan el control de varias de estas características.

En cuanto al control de posición de los brazos robóticos existen dos posibilidades principales:

- Modelos cinemáticos: La cinemática de un brazo robótico se refiere al estudio del movimiento y la posición de sus partes móviles, como juntas y eslabones, sin tener en cuenta las fuerzas que los producen (dinámica). Se enfoca en la descripción matemática del movimiento de los elementos del brazo robótico en términos de posición, velocidad, aceleración y dirección. Establece una relación entre la posición y orientación del extremo del robot y los valores del resto de sus coordenadas articulares. Existen dos modelos cinemáticos principales para un brazo robótico:

  - Cinemática directa: este método consiste en determinar la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas de referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.

  - Cinemática inversa: determina la configuración que deben adoptar las articulaciones del robot para alcanzar una posición y orientación del extremo conocidas.

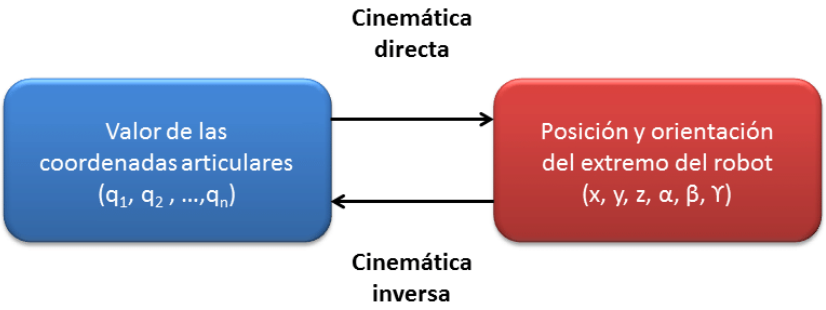


Ilustración 15. Modelos cinemáticos

Existen diversos métodos para la resolución de estos modelos, tales como, métodos geométricos o resolución por el método de Danevit-Hartenberg. No se profundizará en estos modelos y sus métodos de resolución, ya que, para este proyecto el control del brazo robótico no se realizará haciendo uso de estos métodos.

- Control remoto: se refiere a la capacidad de controlar un dispositivo electrónico desde una ubicación remota, sin la necesidad de estar cerca del dispositivo. Esto se logra mediante el uso de señales inalámbricas o por cable, que transmiten los comandos del usuario al dispositivo, permitiendo que se realicen funciones específicas.

  Existen diferentes tipos de control remoto, desde los más simples hasta los más avanzados. Los más simples son los que se utilizan en televisores, reproductores de DVD y otros dispositivos similares, que

utilizan señales infrarrojas para transmitir comandos al dispositivo. Estos controles suelen tener botones para controlar la potencia, el volumen, el cambio de canales, entre otras funciones.

Los controles remotos más avanzados se utilizan en aplicaciones industriales, como en sistemas de control de procesos o en robots industriales. Estos controles pueden ser inalámbricos o por cable y pueden ser capaces de controlar una gran cantidad de funciones y parámetros en tiempo real.

En general, el control remoto tiene la ventaja de permitir el control de dispositivos electrónicos desde una ubicación remota, lo que puede ser muy útil en una variedad de situaciones. Además, los controles remotos avanzados pueden ser programados para realizar una gran cantidad de tareas y para adaptarse a diferentes necesidades y requisitos de la aplicación.

Este control remoto se puede realizar de diversas formas, mediante mandos a distancia o desde un computador. Este es el control que se va a desarrollar para este proyecto, y se profundizará en él durante toda la duración del mismo.

3. Diseño electrónico

Hasta ahora se ha hecho una introducción en la que se han explicado los objetivos de este Trabajo Fin de Grado y los aspectos que han llevado a su realización. Además, tras un proceso de recogida de información e investigación, se ha proporcionado una panorámica general sobre la robótica y los robots hasta el día de hoy y más concretamente se ha profundizado en los brazos robóticos que es el tema de estudio de este proyecto. También se ha hecho un resumen sobre los fundamentos mecánicos del brazo, ya que, se considera necesaria una comprensión básica de los aspectos mecánicos de los que se parte para su posterior control. Como se ha comentado, esta estructura ha sido realizada previamente por otro alumno de ingeniería mecánica como Trabajo Fin de Grado, en el cual se desarrollan más a fondo los aspectos de diseño del mismo. En este punto comienza el desarrollo del grueso del presente Trabajo Fin de Grado, que consistirá en la implementación de toda la electrónica que proporcionará el control y el movimiento del brazo. En esta parte se profundizará en los distintos aspectos que han llevado a la selección de los componentes utilizados, sus características técnicas y electrónicas y el software utilizado para su control.

3.1. INTRODUCCIÓN

Como se ha comentado anteriormente existen dos métodos principales para el control de brazos robóticos, para este trabajo se ha seleccionado el control remoto como modo de funcionamiento.

Este control remoto se fundamentará en la utilización de un ‘guante robótico’, este constará de una serie de sensores y módulos de comunicación que permitirán recoger la información del movimiento realizado por la mano del usuario y enviársela al brazo robótico en tiempo real, el cual tendrá que moverse en función de estos datos recibidos. Se ha seleccionado este método de control porque se ha visto que existen muchos menos ejemplos de su implementación que en el caso por ejemplo de modelos cinemáticos, por lo tanto, resulta algo más innovador, aunque ya existan también ejemplos de su implementación. Además, en el caso de la previsión de aplicación para el que se quiere utilizar (usos recreativos y educativos) resulta un método de control sencillo y vistoso para el usuario.

Por lo tanto, el siguiente proyecto se dividirá en dos subsistemas principales claramente diferenciados:

- El brazo robótico: formado por la estructura mecánica, los accionadores y el módulo de comunicación.
- El guante robótico: formado por los sensores que recogerán la información y el módulo de comunicación.

4. Brazo robótico

Se puede ver un brazo robótico como un conjunto de motores que trabajan a la vez y de forma coordinada para alcanzar una posición establecida. Por lo tanto, si se quiere controlar la posición de un brazo robótico se necesitará controlar la posición de los motores que lo mueven. Para esto se requerirá una serie de componentes que realicen esta función. Para el funcionamiento del brazo robótico se utilizarán los siguientes componentes:

- Arduino Uno.
- 6x Servomotor MG996R Series.
- Servo driver PCA9685.
- Motor paso a paso NEMA 17.
- Driver para motor paso a paso 8825.
- Batería 4.8 V 2400 mAh.
- Batería 12 V 2500 mAh.
- Módulo bluetooth HC-05.
- PCB
- Cables.

4.1. ARDUINO

Inicialmente, como parte lógica principal para la programación se utilizará Arduino. Arduino es un proyecto de hardware y software libre para la creación de prototipos y proyectos electrónicos, esto significa que cualquier persona puede acceder a los esquemas, diseño y códigos con los que la empresa ha creado sus placas y replicarlas o modificarlas en función de sus necesidades. También dispone de un software fácil y flexible para la programación de sus placas, el Arduino IDE, este software utiliza lenguaje de programación C. Este proyecto nace en 2003 de la mano de unos estudiantes italianos que pretendían posibilitar a todo el que quisiera el acceso a la realización de proyectos electrónicos de una forma poco costosa y sencilla.

Actualmente las placas de Arduino son unas de las más extendidas para la realización de proyecto electrónicos y prototipos, y debido a esto existe una importante cantidad de información y recursos para este tipo de placas que hacen a su vez más sencillo el aprendizaje y la realización de proyectos. Además, existen numerosos periféricos y componentes compatibles con las placas Arduino que posibilitan la realización de proyectos muy variados. Estos periféricos cuentan normalmente con librerías

desarrolladas por usuarios o desarrolladores de código que facilitan mucho el control de estos periféricos. En programación una librería es un conjunto de funciones y subrutinas predefinidas que pueden ser reutilizadas en múltiples programas o proyectos sin necesidad de volver a escribir el código de cero, con el objetivo de ser llamadas desde el programa principal y que estas realicen una función.

Por todos estos factores se ha decidido que este tipo de placa es la adecuada y será sobre la que se fundamente el control del brazo de este proyecto.

Esto se decide así porque como se ha explicado anteriormente el uso que se pretende hacer del brazo es un uso educativo y recreativo, en caso de no ser así Arduino no sería el sistema de control más adecuado, ya que esta placa tiene por objetivo la realización de prototipos y proyectos de forma fácil y barata, pero para un entorno industrial o de mayor complejidad esta placa tiene diversas limitaciones que no le permitirían cumplir su cometido de forma adecuada, ya que, Arduino utiliza un microcontrolador que no es lo suficientemente potente para manejar tareas complejas o procesos intensivos en términos de recursos. Además, la placa de circuito impreso de Arduino no está diseñada para ser resistente a la intemperie, ni para soportar altas temperaturas, vibraciones, interferencias electromagnéticas, ni otros tipos de condiciones ambientales adversas que son comunes en entornos industriales.

4.1.1. ARDUINO UNO

Entre las distintas placas de Arduino para este proyecto se ha decidido utilizar una placa de Arduino Uno que es su modelo más utilizado, tiene un precio económico y cumple perfectamente con las funcionalidades requeridas para este proyecto.

Arduino Uno es una placa de circuito impreso basada en un microcontrolador ATmega328P de la compañía Atmel. La placa Arduino Uno incluye una serie de pines digitales y analógicos que permiten a los usuarios conectar y controlar una variedad de componentes electrónicos, como sensores, motores, pantallas, leds y otros dispositivos. También cuenta con un puerto USB que se utiliza para cargar el código en el microcontrolador y para comunicarse con la placa desde un ordenador y desde el cual también se puede alimentar la placa en el caso en el que se le vaya a dar usos en los que el consumo de corriente por parte de los periféricos sea mínimo, ya que, en caso contrario el conector USB no podría suministrar suficiente corriente y daría lugar a funcionamientos erráticos. Las principales características técnicas son las siguientes:

- Microcontrolador: Microchip ATmega328P6
- Voltaje de funcionamiento: 5 voltios
- Voltaje de entrada: 7 a 20 voltios
- Pines de E/S digitales: 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica: 6

- Corriente DC por Pin de E/S: 20 mA
- Corriente DC para Pin de 3.3V: 50 mA
- Velocidad del reloj: 16 MHz
- Longitud: 68.6 mm
- Ancho: 53.4 mm
- Peso: 25 g

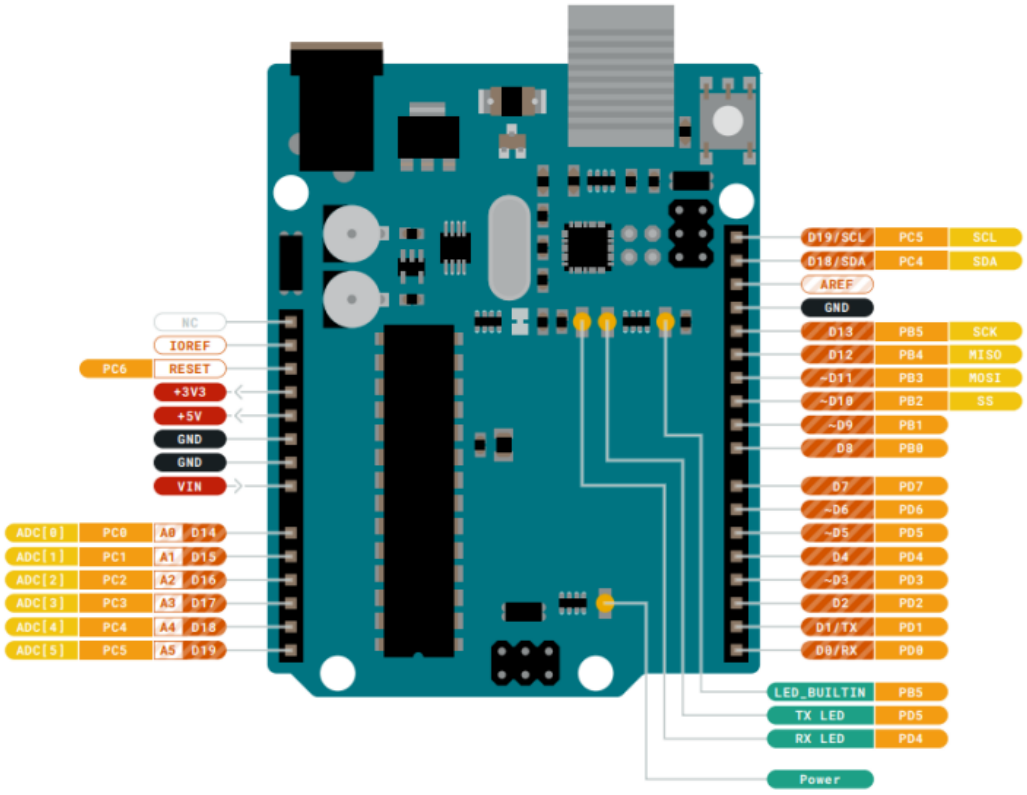


Ilustración 16. Esquemático Arduino Uno

Como se puede observar hay pines que pueden tener diferentes formas de funcionamiento en función de la configuración que se le introduzca al igual que en la mayoría de los microcontroladores. Cabe destacar el funcionamiento de cuatro de los pines que tendrán gran importancia en cualquier proyecto con Arduino que se realice y en cualquier proyecto electrónico en general:

- Pin Vin: desde este pin se puede suministrar voltaje en un rango de 7-20 V o si se suministra voltaje desde el conector de alimentación, acceder a él a través de este pin.

- Pin 5V: Este pin emite 5 V regulado desde el regulador de la placa. La placa se puede alimentar con el conector de alimentación de CC (7-20 V), el conector USB (5 V) o el pin VIN de la placa (7-20 V). El suministro de voltaje a través de los pines de 5 V o 3.3 V evita el regulador y puede dañar la placa.
- Pin 3.3V: un suministro de 3.3 voltios generado por el regulador de la placa. El consumo máximo de corriente es de 50 mA.
- Pin GND: Pines de tierra.

Es importante tener claro el funcionamiento de estos pines y las capacidades que pueden soportar o suministrar en cuanto a valores de corriente y tensión, ya que, sin una alimentación estable y suficiente para cualquier proyecto electrónico, este fallará. Por lo tanto, siempre hay que tener en cuenta las características de alimentación que la placa puede recibir desde la fuente de alimentación y suministrar a los periféricos. Habrá que tener en cuenta el consumo de corriente y el rango de voltajes con los que los periféricos que se conecten a la placa pueden trabajar.

También se recalca el pin de tierra, ya que, casi para cualquier proyecto electrónico es importante en general tener una tierra estable y unificada entre los distintos componentes que se utilicen para el proyecto, ya que en caso de no tener esto en cuenta se producirán funcionamientos erráticos.

Se ha querido recalcar este punto, ya que, no tener una correcta fuente de alimentación y una masa estable, es fuente de numerosos fallos y funcionamientos erróneos en prototipos y proyectos electrónicos.

4.2. SERVOMOTORES

Serán los encargados de proporcionar el movimiento al brazo. Un servomotor es un tipo de motor eléctrico que se caracteriza por su capacidad de control preciso de la posición angular. Está diseñado para recibir señales de control de posición y velocidad, y convertirlas en movimiento mecánico de precisión en el eje de salida.

El funcionamiento de un servomotor se basa en un sistema de realimentación. Un sensor de posición, típicamente un potenciómetro o un codificador, se monta en el eje del motor y proporciona información en tiempo real sobre la posición angular actual del eje. Esta información se compara con la posición de destino que se desea alcanzar, y el controlador del servomotor ajusta la energía eléctrica suministrada al motor para que el eje se mueva en la dirección adecuada y alcance la posición deseada con precisión. Estos accionadores son los ideales para este tipo de proyectos en los que se necesita un movimiento controlado y alcanzar posiciones concretas de forma precisa, por lo que son muy típicos en robótica. Además, están ampliamente extendidos por lo que hay gran variedad de recursos e información sobre ellos además de tener un precio económico. Otra ventaja es que tienen un consumo bastante bajo.

Para controlar un servo motor, se necesita enviar una señal de control al mismo, esta señal se conoce como señal PWM (Pulse Width Modulation). La señal PWM consiste en una serie de pulsos digitales (onda cuadrada) de ancho variable que se utilizan para controlar la posición del servo.

Para enviar la señal PWM al servo, se puede utilizar un microcontrolador o un circuito integrado especializado. El microcontrolador o circuito integrado genera una señal PWM que se envía al servo a través de tres cables: uno para la señal PWM, uno para la alimentación y uno para la masa.

La señal PWM envía información al servo sobre la posición en la que debe estar. La señal consiste en una serie de pulsos eléctricos con una duración que varía entre 1 y 2 milisegundos habitualmente, aunque esto depende del servo en concreto y de su hoja de características. El tiempo de duración del pulso determina la posición del servo.

Para controlar el servo, se puede programar el microcontrolador o circuito integrado para generar la señal PWM adecuada. También se pueden utilizar bibliotecas de software para facilitar la programación. Cabe destacar también que la mayoría de estos dispositivos trabajan a una frecuencia de 50Hz (periodo 20ms).

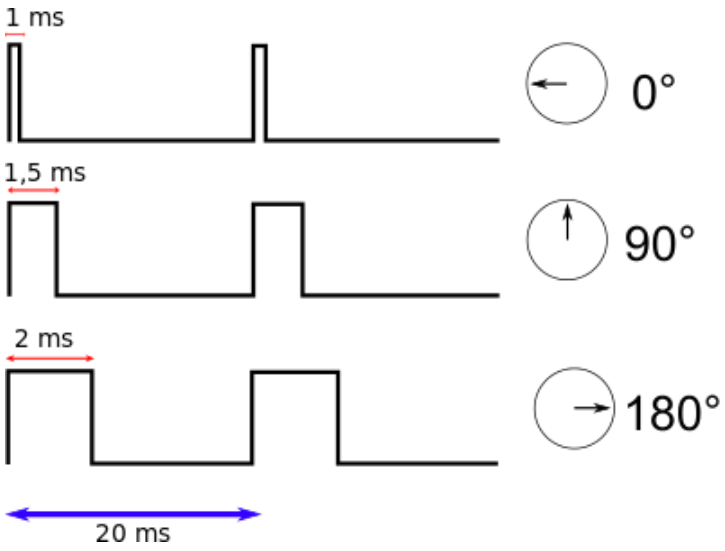


Ilustración 17. PWM para control de servo

Para este proyecto se ha decidido utilizar los servomotores MG996R que tiene suficiente potencia para mover todas las articulaciones del brazo robótico excepto la rotación de la base para la cual se ha decidido utilizar un motor paso a paso, el cual se explicará más adelante en detalle y las razones que han llevado a esta decisión. Este servomotor a diferencia de otros que tienen 180º de rango de movimiento efectivo tiene 120º, aunque será suficiente para este proyecto. Además en el caso de la articulación que une la base con el primer eslabón se han colocado dos servomotores trabajando conjuntamente, uno enfrente de otro, habrá que tener esto en cuenta para su programación ya que el sentido de giro será el contrario para uno que para otro por lo

que a uno habrá que meterle la señal complementaria del otro para que se muevan solidarios, además debido a que al montarlos no se ha montado el brazo exactamente en la misma posición sobre el servomotor de un lado que sobre el de otro tendrá un pequeño desfase que también habrá que tener en cuenta a la hora de su programación, pero este procedimiento se explicará más en detalle posteriormente en la parte de pruebas y experimentos.

Este servomotor trabaja a una tensión de entre 4.8-7.2 V y en su funcionamiento puede consumir hasta 900 mA, aunque se ha comprobado que en la mayoría de las situaciones un solo servo consumo en el rango de 100 a 350 mA. Como se requerirá utilizar 6 servomotores para el movimiento del brazo el consumo de corriente en determinados momentos puede ser bastante alto, por este motivo se ha decidido colocar una fuente de alimentación externa y un driver para los servomotores. Esto se hace así porque la placa Arduino no será capaz de suministrar tanta corriente y en caso de conectar directamente los motores al Arduino daría lugar a funcionamientos erráticos. Además, trabajando directamente desde Arduino se consumirían seis pines, en caso de trabajar con el driver solo dos. En este caso, los motores y el Arduino funcionan en los mismos rangos de tensión, 5V aproximadamente, pero en caso de no ser así el driver nos serviría también para adaptar los niveles tensión de forma que se tendría la alimentación de control para la placa controladora (típicamente de 3.3 o 5 V) y la alimentación de potencia de la tensión necesaria aislada de la de control.

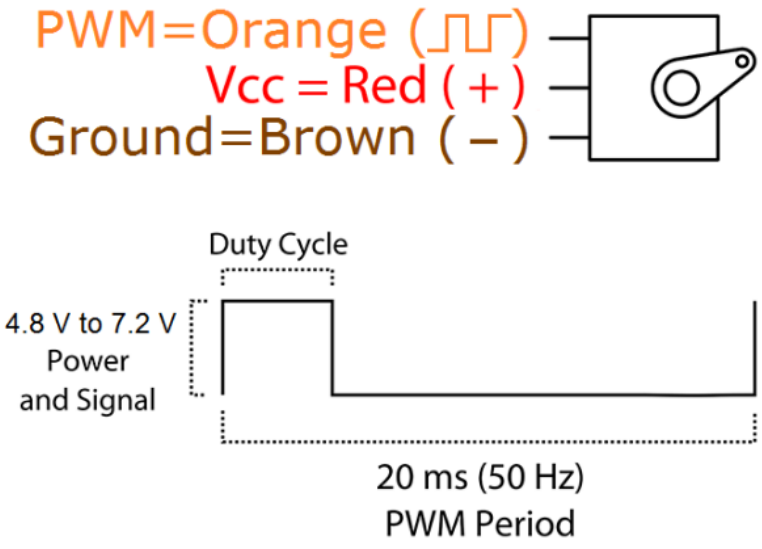


Ilustración 18. Conexionado de los servomotores

4.3. DRIVER SERVOMOTORES

Como se comenta en el punto anterior el motivo principal por el que se ha utilizado el driver en este proyecto es debido a que las exigencias de corriente de los servomotores no podían ser suplidas por el Arduino. Pero en general es una buena praxis utilizar un

driver en la mayoría de las situaciones en las que se requiera el uso de motores, ya que, estos componentes están especialmente diseñados para conectarse directamente a los motores, a diferencia de los microcontroladores.

El driver se utiliza para controlar uno o varios servomotores de forma más eficiente y sencilla que si se hiciera directamente desde un microcontrolador o circuito integrado, además de integrar protecciones y otros elementos para un mejor funcionamiento.

El driver para servos proporciona una interfaz de control simplificada para los servomotores, lo que hace que sea más fácil programar y controlar los movimientos del motor. Además, los drivers para servos también pueden proporcionar funciones adicionales, como protección contra sobrecargas, ajuste de la velocidad y la posición, y la capacidad de controlar varios servomotores al mismo tiempo.

Por todos estos motivos se ha decidido utilizar un driver, en este caso el PCA9685. El PCA9685 es un driver de 16 canales PWM (Pulse Width Modulation) que se utiliza comúnmente para controlar servomotores y luces LED. Cada canal del PCA9685 tiene una resolución de 12 bits, lo que significa que puede generar 4096 niveles. Este dispositivo trabaja entre 40 y 1000 Hz (habitualmente 50 para servos y 1000 para Leds).

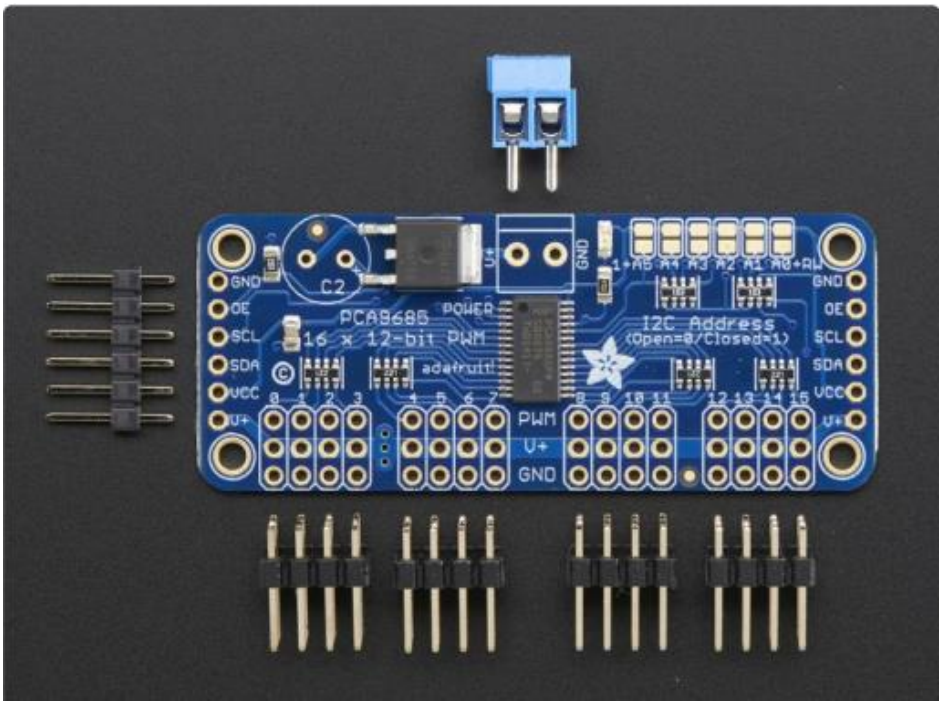


Ilustración 19. Driver PCA9685

El PCA9685 es un circuito integrado que se comunica con un microcontrolador a través del protocolo I2C (Inter-Integrated Circuit).

El protocolo I2C (Inter-Integrated Circuit) es un protocolo de comunicación serie sincrónico que se utiliza para interconectar dispositivos electrónicos en un circuito integrado o en una placa de circuito impreso. El protocolo I2C utiliza dos líneas de

señal para la transmisión de datos: la línea de datos (SDA) y la línea de reloj (SCL). Los dispositivos en un bus I2C se dividen en dos categorías: maestros y esclavos. El dispositivo maestro es el que inicia la comunicación, envía comandos y controla el flujo de datos. Los dispositivos esclavos reciben comandos y envían respuestas al dispositivo maestro. El protocolo I2C utiliza una dirección de 7 bits para identificar cada dispositivo en el bus. Cada dispositivo I2C tiene una dirección única que se utiliza para comunicarse con él. Si hay varios dispositivos en el bus con la misma dirección, se pueden seleccionar diferentes direcciones para cada dispositivo a través de puentes de selección de dirección o mediante la configuración de pines específicos. Este protocolo es interesante principalmente porque permitirá conectar y controlar varios periféricos a un mismo microcontrolador ocupando solamente dos líneas de transmisión SDA y SCL. En este caso no habrá muchos dispositivos, simplemente se enviarán los datos necesarios en serie para controlar los distintos canales del driver.

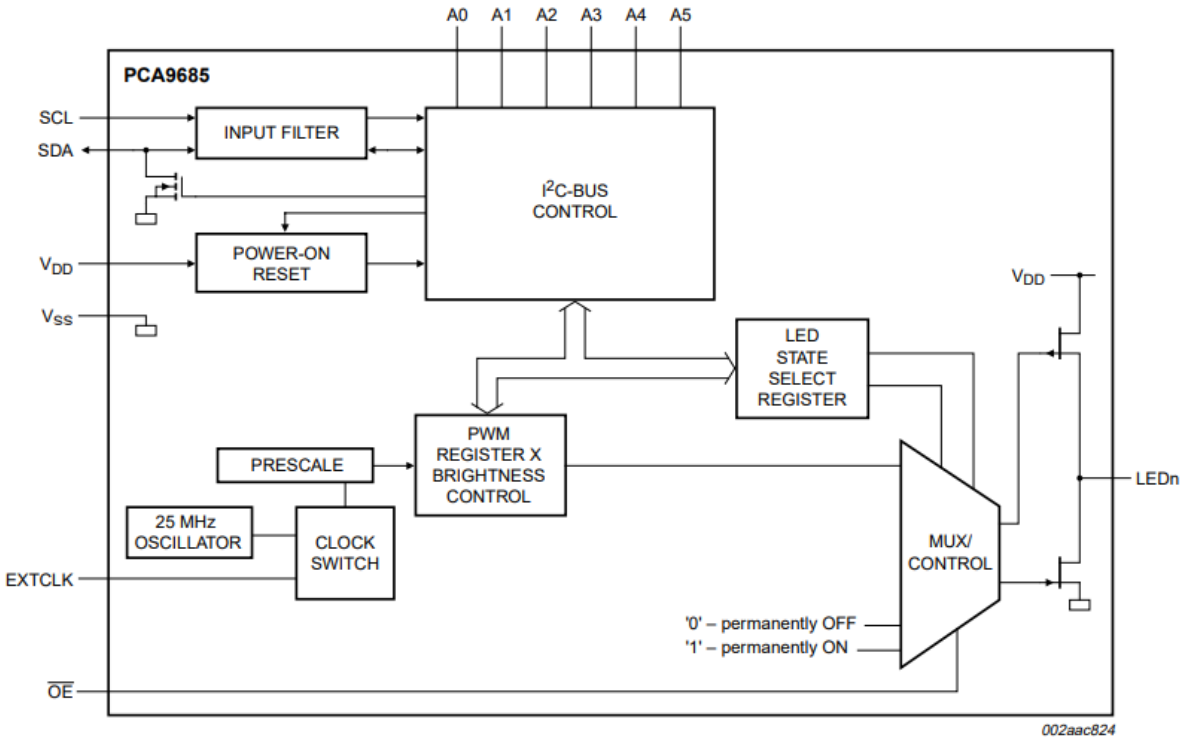


Ilustración 20. Diagrama de bloques del PCA9685

Se ha seleccionado este driver ya que se van a utilizar 6 servomotores para este proyecto de forma que utilizar 6 servocontroladores sería mucho menos cómodo que poder controlar todos desde un mismo dispositivo conectado directamente al microcontrolador y utilizando solo dos pines.

Además, otro factor importante es, que este driver dispone de una librería propia para el control de servomotores, y se podrá utilizar para controlarlos de forma sencilla. Esta librería se llama 'HCPCA9685.h'. Como funciones principales a tener en cuenta para su uso para una aplicación básica habrá que:

- Definir la dirección I2C del driver, que por defecto si no lo cambias es 0x40 (#define I2cAdd 0x40).
- Declarar una variable de tipo HCPCA9685 que se utilizará para controlar los servos. (HCPCA9685 MiServo(I2CAdd))
- Iniciar la librería en modo servo. (MiServo.Init(SERVO_MODE))
- Establecer la frecuencia de trabajo en 50Hz para servos. (MiServo.SetPeriodFreq(50))
- Mover el servo a la posición deseada. (MiServo.Servo(canal, duty))
    - El valor duty no será la posición en grados del servomotor sino en valor entre 0-4095 (12 bits) en función del cual el servo alcanzará una u otra posición. Esto se hará mediante un programa que se explicará más adelante en el cual se calibrará el valor de este parámetro para cada servomotor en función de las restricciones del servo y también de las restricciones mecánicas del propio brazo. Recalcar que cada servomotor tendrá parámetros de duty distintos, en un rango parecido, pero no el mismo, para realizar el mismo movimiento, a pesar incluso de ser del mismo modelo.

Por último, en cuanto al driver, para alimentarlo se utilizará una batería de 4.8 V y 2400 mAh que resulta suficiente en cualquier situación del brazo, aunque durante las pruebas se ha utilizado una fuente de alimentación.

En cuanto al conexionado general del driver junto con el Arduino y la alimentación quedaría de la siguiente forma.

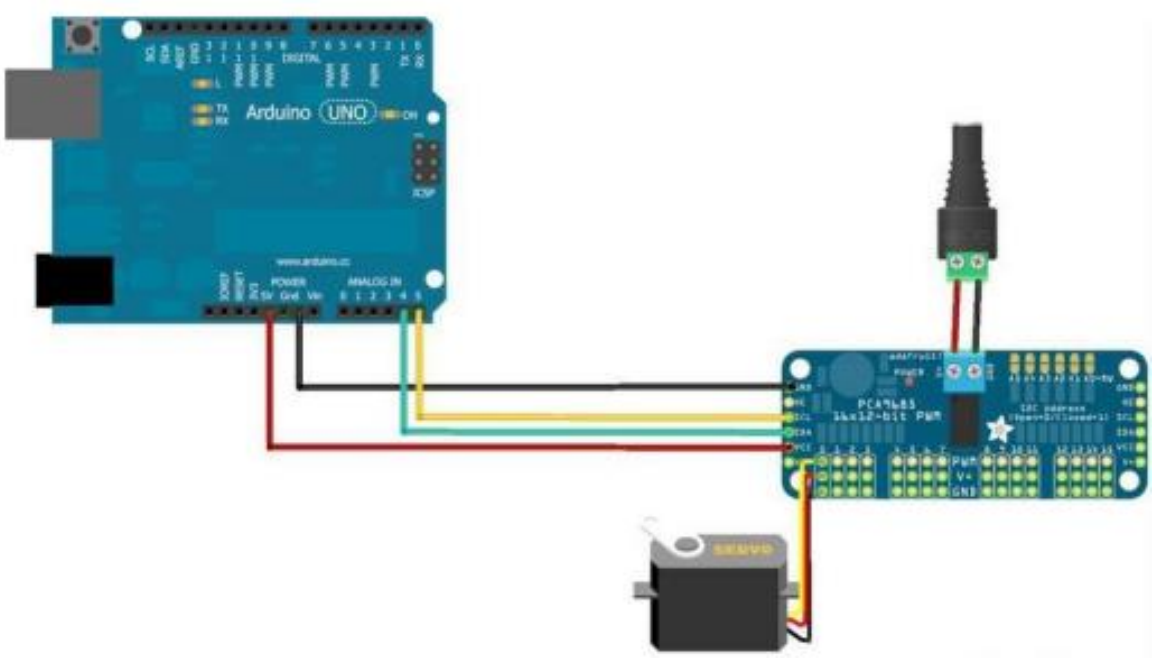


Ilustración 21. Conexionado PCA9685, servo y Arduino

- Vcc (Arduino) -> Vcc (PCA9685)
- GND (Arduino) -> GND (PCA9685)
- Analog 4 (Arduino) -> SDA (PCA9685)
- Analog 5 (Arduino) -> SCL (PCA9685)

En el caso del conexionado de la alimentación conectada a la batería mediante los pines V+ y GND, no se conectará de esa forma por razones que se explicarán más adelante en la sección de pruebas y experimentos.

4.4. MOTOR PASO A PASO

Como se ha comentado más arriba, para la rotación de la base se ha decidido utilizar un motor paso a paso. Esto se ha decidido así ya que el brazo tiene un peso considerable y ya que el motor de la base será el encargado de mover y sostener todo el peso del brazo se requiere un motor más robusto y con un par mayor que permita realizar este cometido sin problemas, además el motor paso a paso permite rotar el brazo en un rango completo de 360º y llegar así a cualquier punto del espacio, cosa que no sucede en el caso de usar un servomotor, por tanto en este caso el motor paso a paso sería mejor opción que el servo.

Un motor paso a paso es un tipo de motor eléctrico que convierte señales eléctricas en movimientos mecánicos discretos, llamados pasos.

En función del número de pasos el motor será más o menos preciso, lo más habitual son motores de 200 pasos, que tendrán una precisión en sus movimientos de 1.8º (360/200),

pero existen motores con otro tipo de rangos de pasos. Además, existe un tipo de funcionamiento en el cual el motor puede funcionar desplazándose solamente medio paso o un cuarto de paso de forma que se podría aumentar su sensibilidad aún más, estos modos de funcionamiento se conocen como microstep.

Hay varios tipos de motores paso a paso, pero el diseño básico implica un rotor con varios dientes magnéticos y un estator con bobinas electromagnéticas. Cuando se aplican señales eléctricas a las bobinas del estator, los campos magnéticos resultantes hacen que el rotor gire en una dirección específica, avanzando un paso a la vez.

Es decir, se alimentarán las bobinas del estator siguiendo una secuencia que creará un campo magnético giratorio que los dientes magnéticos del rotor seguirán. Estos motores tienen por característica principal que pueden girar hasta alcanzar una posición angular específica al igual que los servomotores por lo cual son útiles para esta aplicación.

En función de su modo de funcionamiento y su construcción se pueden diferenciar varios tipos de motores paso a paso como son: de reluctancia variable, de imanes permanentes (dentro de este tipo se pueden distinguir además entre unipolares y bipolares.) o híbridos. No se profundizará en el funcionamiento específico de cada uno de ellos.

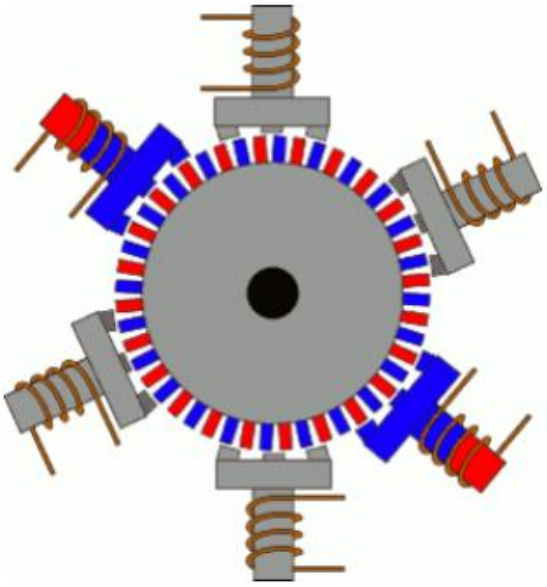


Ilustración 22. Motor paso a paso

Para este proyecto se ha escogido un motor paso de imanes permanentes bipolar. Estos motores tienen mayor par y precisión que los unipolares, en contrapartida su control es más complejo y tienen un mayor consumo de energía. En concreto se ha seleccionado el NEMA 17 JK42HS34-0844 que trabaja a 12 V y tiene un consumo de corriente por fase de 0.8 A.

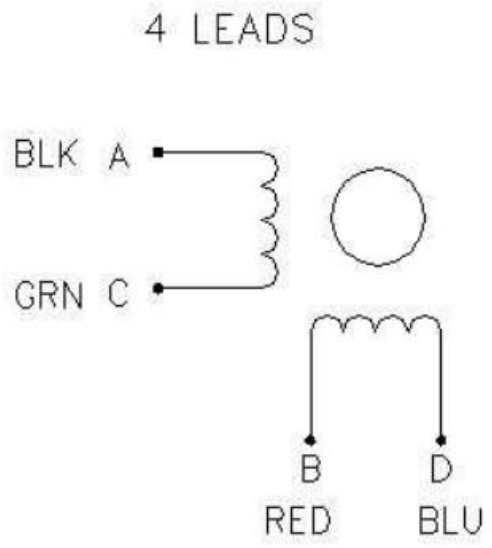


Ilustración 23. Bobinas motor paso a paso

4.5. DRIVER MOTOR PASO A PASO

Como se ha comentado anteriormente este tipo de motor requiere de un control complejo, por lo que se utilizará para su control un driver específico para este tipo de motores y una librería. En este caso el driver servirá para adaptar los niveles de tensión entre el Arduino que trabaja a 5 V y el motor que trabaja a 12 V, además de facilitar el control del motor e integrar las protecciones necesarias.

Este driver está compuesto internamente por dos puentes en H (Motor driver A y B en la imagen) que serán la estructura principal encargada del control del motor. Además, disponen de otra serie de componentes que aportarán funcionalidades útiles para el funcionamiento del motor como un regulador de corriente para ajustar la corriente por cada bobina, protección térmica y contra cortocircuitos y posibilidad de control mediante micro-stepping. Para su control únicamente requieren dos salidas digitales, una para indicar el sentido de giro (DIR) y otra para comunicar que queremos que el motor avance un paso (STEP). Este driver es una evolución del A4988 un driver bastante popular para este tipo de motores, ambos funcionan de la misma forma con la única diferencia de que el 8825 tiene algunas características superiores: hasta 2.5 A por fase, hasta 45 V, 32 microsteps. En caso de superar 1.5 A será necesario colocar un disipador de calor, en este proyecto no será necesario.

Para regular la corriente que circula por las bobinas del motor habrá que seguir el siguiente procedimiento:

- Medir la tensión (Vref) entre el potenciómetro y GND.

- Ajustar la Vref con el potenciómetro para que se cumpla que la Imax despejada de la siguiente fórmula sea igual a la Imax por bobina de la hoja de características del motor. Imax = Vref / (5 * Rs)

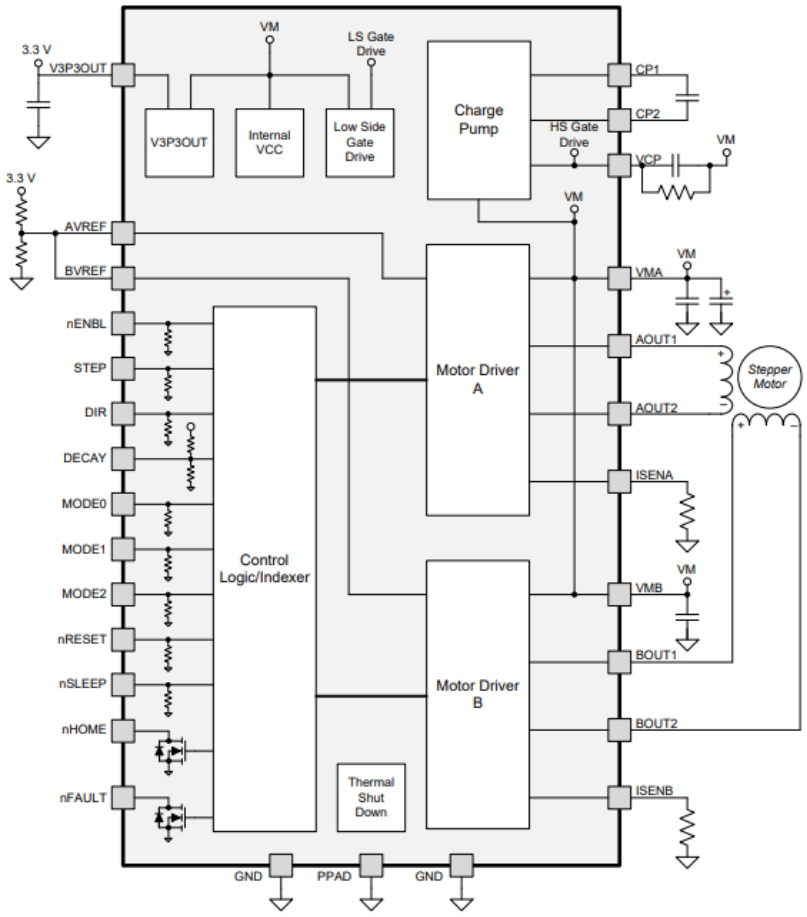


Ilustración 24. Estructura interna DRV8825

En cuanto al conexionado del driver con el Arduino y el motor quedaría de la siguiente forma:

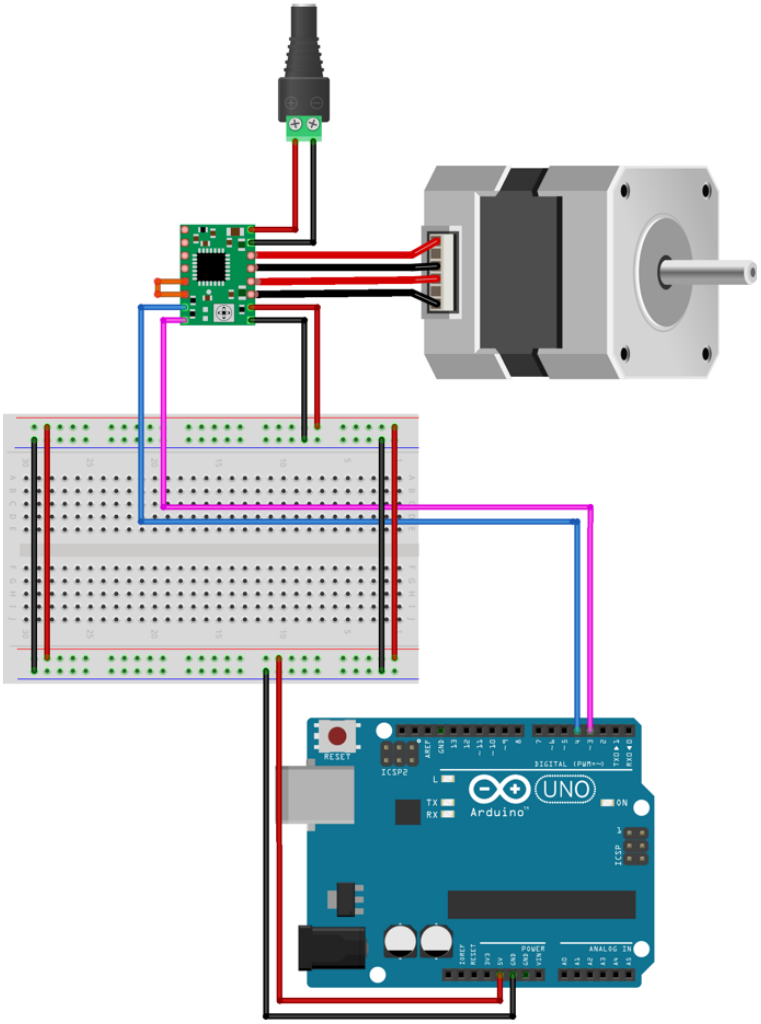


Ilustración 25. Conexionado DRV8825

Como se ha comentado anteriormente para mayor facilidad en el control del motor paso a paso se ha utilizado una librería de Arduino. La librería es AccelStepper.h. Esta librería contiene un conjunto de funciones que permite controlar uno o varios motores paso a paso. Además, tiene algunas funcionalidades más allá del simple control de la posición del motor paso a paso tales como el control de la velocidad entre pasos o un arranque y frenado suave para no generar grandes pares de inercia en los arranques y paradas, lo cual es muy interesante en este proyecto. Como se puede comprobar, el hecho de que haya tanta documentación y recursos para esta plataforma (Arduino) facilita mucho la implementación de tareas que en ocaso de partir de cero serían bastante complejas. Los pasos que seguir y funciones básicas a utilizar para el uso de esta librería serían los siguientes:

- Declarar una variable de tipo AccelStepper, asignado el pin de paso y de dirección. (AccelStepper stepper(AccelStepper::DRIVER, stepPin, dirPin);)
- Establecer la velocidad (pasos/s). (stepper.setSpeed(velocidad))

- Establecer la aceleración (stepper.setAcceleration(aceleración))
- Establecer cantidad de pasos a moverse. (stepper.moveTo(pasos))

Con todo lo visto hasta ahora se podrían mover los motores que moverían a su vez el brazo robótico, pero sin un sistema de control definido.

Para poder implementar el control se necesita comunicar el brazo con el guante robótico que servirá de elemento de control. Este será el siguiente punto a tratar.

4.6. MÓDULO BLUETOOTH ESCLAVO

La comunicación entre el brazo y el guante robótico se va a hacer de forma inalámbrica y más concretamente por bluetooth. El bluetooth es un tipo de tecnología de comunicación inalámbrica que permite a dos dispositivos electrónicos que están en un rango de distancia determinado (normalmente no más de 10 metros) intercambiar datos mediante ondas de radiofrecuencia en la banda de 2.4 GHz.

Para implementar esta tecnología se ha utilizado el módulo bluetooth HC-05. El HC-05 es un módulo de transceptor[1] Bluetooth que se comunica a través del protocolo serial universal (UART), utiliza la versión 2.0 de la especificación Bluetooth.

El protocolo UART (Universal Asynchronous Receiver/Transmitter) es un protocolo de comunicación serie asíncrono utilizado para la transmisión de datos entre dispositivos electrónicos. Es una interfaz de hardware que permite la comunicación entre dos dispositivos electrónicos mediante la transmisión y recepción de datos en serie. En una comunicación UART, el dispositivo que envía los datos se conoce como transmisor y el dispositivo que recibe los datos se conoce como receptor. El transmisor envía los datos a través de un canal de comunicación en serie, que puede ser un cable o un enlace inalámbrico, en paquetes de bits. Estos bits se envían secuencialmente a una velocidad determinada, conocida como tasa de baudios (bits/segundo). El receptor, por otro lado, lee los bits de datos entrantes y los decodifica en caracteres o números que pueden ser procesados por el dispositivo. La comunicación UART es asíncrona, lo que significa que los dispositivos no necesitan sincronizarse antes de la transmisión de datos. En cambio, los dispositivos utilizan marcas de inicio y de finalización de paquetes de datos para sincronizar la comunicación. La mayoría de los microcontroladores y microprocesadores tienen un hardware UART incorporado, lo que hace que la comunicación UART sea una forma común y fácil de comunicación entre dispositivos.

El módulo HC-05 se puede configurar para trabajar en modo maestro o esclavo. Maestro es el dispositivo que inicia y controla el intercambio de información y los esclavos (puede ser uno o varios) responden a las solicitudes del maestro (comunicación bidireccional).

[1] Emisor y receptor

El HC-05 se puede alimentar con voltajes de 3.3 V a 5 V y su consumo de energía es bajo.

La comunicación entre el HC-05 y otros dispositivos se realiza mediante el establecimiento de una conexión Bluetooth. La conexión se establece mediante el proceso de "emparejamiento", que implica que el HC-05 y el otro dispositivo intercambian claves de seguridad para establecer una conexión segura y codificada. Una vez que se establece la conexión, los dispositivos pueden intercambiar datos a través del protocolo UART.

Para este proyecto se utilizarán dos módulos HC-05, uno para el brazo y otro para el guante. El guante trabajará como dispositivo maestro enviando datos de los sensores y controlando la comunicación y el brazo recibirá los datos del guante.

El conexionado del módulo HC-05 se hará como sigue:

- Vcc (Arduino) -> Vcc (HC-05)
- GND (Arduino) -> GND (HC-05)
- Pin 10 (Arduino) -> RX (HC-05)
- Pin 11 (Arduino) -> TX (HC-05)

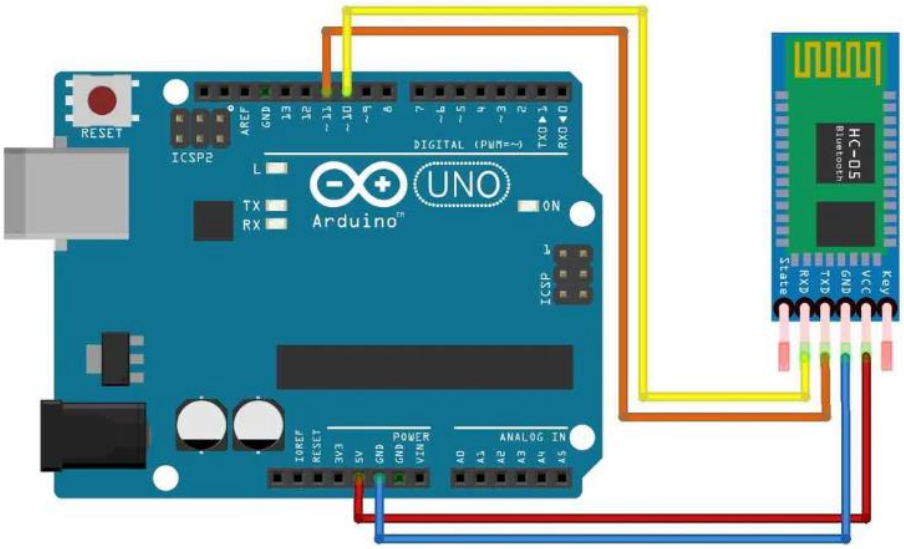


Ilustración 26. Conexionado HC-05

El motivo por el cual no se conectan los pines RX y TX del módulo HC-05 a los pines RX y TX del Arduino, que es lo que parece más lógico en un principio, se explicará más adelante, en la parte de pruebas y experimentos.

Para configurar inicialmente el funcionamiento del módulo bluetooth habrá que cargar el siguiente código en la placa:

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10,11); //RX / TX

void setup() {
Serial.begin(9600);
BTSerial.begin(38400);
Serial.println("Dispositivo listo para recibir comando AT");
}

void loop() {
  if(Serial.available())
    BTSerial.write(Serial.read());
  if(BTSerial.available())
    Serial.write(BTSerial.read());
}
```

Ilustración 27. Código configuración BT

Nótese que la tasa de baudios para trabajar en el modo configuración es 38400 la recomendada por el fabricante. Además, se ha utilizado una librería llamada SoftwareSerial.h que permite implementar un puerto serie en cualquiera de los pines del microcontrolador cosa que es útil para no ocupar los principales, ya que el Arduino Uno solo tiene un puerto serie.

Una vez cargado ese código habrá que seguir los siguientes pasos:

- Mientras se mantiene presionado el botón del módulo se conecta la alimentación (no antes). En este momento el módulo entra en modo de configuración y preparado para recibir comandos AT.
- Poner el monitor serie de Arduino en modo ‘Ambos NL & CR.
- Escribir el monitor serie los siguientes comandos (si todo funciona el monitor devolverá OK sino Error (0)):
    - AT -> Si devuelve OK el dispositivo está esperando comandos AT
    - AT+NAME=Nombre -> Si se le quiere cambiar el nombre
    - AT+PSWD=Contraseña -> La contraseña que se utilizará para comunicarse con el dispositivo maestro (Debe ser la misma para ambos).
    - AT+UART=4800,0,0 -> Para configurar la tasa de baudios de la comunicación (En este caso 4800).
    - AT+ROLE=0 -> Configurar el dispositivo en modo esclavo (1 para modo maestro).
    - AT+ADDR -> Devuelve la dirección MAC del dispositivo esclavo que será necesaria para configurar el módulo maestro.

Una vez hecho esto se ha configurado correctamente el módulo bluetooth para comunicarse en modo esclavo. Para configurar el módulo bluetooth del guante habrá que seguir unos pasos similares que se explicarán en el apartado del guante robótico.

Una vez terminado esto todos los componentes electrónicos del guante están configurados y listos para funcionar con un código adecuado.

4.7. CONEXIONADO GENERAL DEL BRAZO

A continuación, se muestra el esquema de conexionado con todos los componentes del subsistema del brazo:

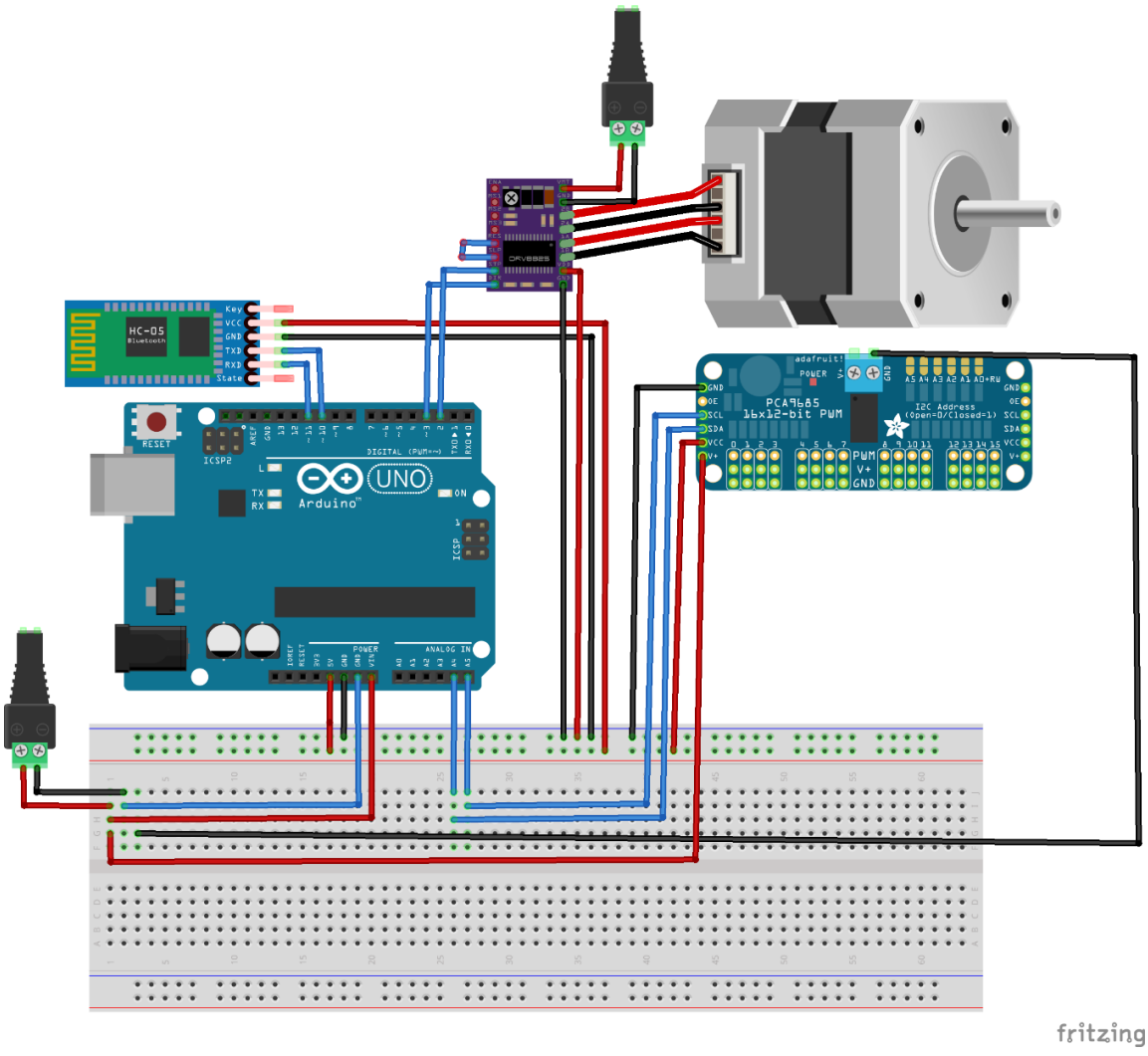


Ilustración 28. Conexionado general del brazo

4.8. PRUEBAS Y EXPERIMENTOS

En este apartado se explicarán las pruebas y procedimientos que se llevaron a cabo para ir comprobando el correcto funcionamiento de todas las partes y los problemas que surgieron durante la realización de dichas pruebas.

4.8.1. SERVOMOTORES

Inicialmente se intentó hacer funcionar un servomotor desde el Arduino con la librería Servo.h, para entrar un poco en contacto con el funcionamiento de los servos y comprobar su funcionamiento, este paso funcionó rápido y sin problemas.

Posteriormente se intentó hacer funcionar un servomotor ahora ya sí conectando el Arduino al PCA9685 y este al servomotor, utilizando la librería HCPCA9685.h específica de este dispositivo. Es aquí donde surge el primer problema, ya que a pesar de realizar todos los conexionados y programación tal y como se decía en cualquiera de las fuentes consultadas, el motor no se movía nada. Tras muchos intentos y búsqueda del posible fallo en los distintos recursos disponibles sin éxito (utilización de otras librerías, distintos métodos de programación, distintos motores, etc) se comprobó el estado de los componentes hardware del sistema, el Arduino y el motor ya se sabía que funcionaban de las pruebas anteriores, así que el fallo en caso de ser un problema hardware debería estar en el driver. Se comprobó con ayuda de un polímetro las distintas tensiones y corrientes que llegaban a los pines del driver, en este punto se comprobó que al pin V+ (alimentación de potencia de los motores) no llegaba ningún tipo de tensión y corriente a pesar de que sí se le estaba alimentando desde la fuente. Esta prueba no se realizó antes, ya que, se habían comprado dos drivers nuevos del mismo tipo y se habían hecho pruebas con ambos sin éxito, por lo que se supuso que el fallo no podía ser de dos componentes nuevos. Tampoco se había probado a alimentar el pin V+ por el otro pin disponible para ello, ya que, en toda la información y recursos a cerca de este dispositivo recomendaban utilizar el otro punto de alimentación ya que la pista conectada al otro puerto de alimentación era más ancha y en caso de necesidad de mucha corriente por parte de los motores el funcionamiento sería más óptimo (menos caída de tensión, calentamiento, etc). Una vez descubierto esto se probó a alimentar el driver desde el pin lateral que no es específico para dicho propósito, pero puede servir para él, a partir de ahí todo comenzó a funcionar perfectamente, sin dar ningún error ni si quiera en el caso de tener los seis servomotores que serían necesarios para el proyecto funcionando todos a la vez. Se cree que puede existir algún error en el proceso de soldadura de las patillas del driver ya que de fábrica este driver viene sin soldar, y cuando se adquirió dicho dispositivo en este caso sí que venía ya soldado, lo cual puede haber surgido de algún problema durante su soldadura. Además, se comprobó también la posible existencia de una habilitación de dicho pin mediante algún comando software, ya que es un procedimiento que a veces se da para este tipo de dispositivos, sin éxito también.

Otro procedimiento a seguir ajustar el correcto funcionamiento de los servomotores fue el proceso de calibración de los mismos. Para ellos se utilizó un programa que aumentaba de uno en uno el valor de la señal pwm que recibía el motor, moviéndolo así poco a poco hasta que se comprobaba que llegaba a su extremo de movimiento permitido, para este extremo se tenían en cuenta tanto restricciones físicas de la propia estructura del brazo, como electrónicas del rango de movimiento de los servomotores.

Durante este proceso también hubo que desmontar y montar el brazo varias veces debido a que no se podía comprobar el movimiento de los servomotores ya montados sobre la estructura del brazo, ya que las restricciones mecánicas no permitían ver el rango de movimiento correcto de los servomotores. Además la articulación dos, en la cual trabajan dos servomotores conjuntamente no se movía en un inicio, ya que los eslabones que van unidos a cada uno de estos servomotores deben ser montados específicamente con ambos servomotores en la misma posición para que trabajen conjuntamente, en este caso no se había hecho así por lo que a pesar de que se programaban para alcanzar la misma posición si partieran de un mismo punto, como esto no era así se trababan y no se movían ya que no podían trabajar solidarios. Aún, desmontándolo y volviéndolo a montar se tuvo que programar un pequeño desfase entre ambos en la señal pwm ya que los motores entre sí pueden no tener exactamente el mismo rango de movimiento a pesar de ser del mismo modelo, como pasaba en este caso.

4.8.2. MOTOR PASO A PASO

Para comprobar el funcionamiento del motor paso a paso directamente se conectó el Arduino al driver y este al motor paso a paso siguiendo los conexionados descritos anteriormente. Se programó para hacerlo funcionar siguiendo los procedimientos descritos anteriormente y con el uso de la librería AccelStepper.h, sin éxito inicialmente, ya que, el motor solo vibraba sin realizar ningún giro. Después de una exhaustiva búsqueda de información acerca de los motores paso a paso se descubrió que las bobinas de los motores paso a paso no siempre vienen con la misma configuración de pines externos por los cuales se conectan al driver, para comprobar que pines iban juntos dos a dos para cada bobina se midió la resistencia entre los terminales del motor paso a paso con el polímetro, en caso de no ser dos pines conectados por la bobina daría resistencia infinita (circuito abierto). Se comprobó con este proceso que los pines del cable que conectaba las bobinas del motor al driver no estaba configurado correctamente, este cable no venía con el motor, sino que se le había puesto durante el montaje del brazo. Una vez cambiada la conexión de dicho cable el motor comenzó a funcionar correctamente.

5. Guante robótico

Una vez se han explicado todos los componentes del brazo robótico, que son, cuáles son sus características y se han configurado para su correcto funcionamiento, se procederá a realiza lo mismo para el guante.

El guante robótico consistirá en una serie de sensores que recogerán información de los movimientos y gestos que el usuario realice con la mano. Esta información será procesada y se enviarán comandos de forma inalámbrica al brazo que controlen su movimiento. Para poder cumplir con estas funciones el guante utilizará:

- Arduino Uno.
- Guante de constructor.
- Módulo bluetooth HC-05.
- 2x MPU6050 (Acelerómetro).
- 3x Sensores flexibles.
- 4x Resistencias de 10k.
- Batería de 4.8 V 2200 mAh.
- Interruptor
- LED
- PCB
- Cables.

Se utilizará de nuevo un Arduino Uno como sistema de control y una serie de sensores para recabar información de los movimientos de la mano, además de algunos elementos complementarios como la protoboard, cables y resistencias.

5.1. MÓDULO BLUETOOTH MAESTRO

A pesar de haber hablado de este módulo en la parte del brazo robótico en esta parte se detallará el procedimiento para configurar el módulo en modo maestro y de forma que se conecte con el esclavo de la placa de control del brazo. El conexionado a seguir y el programa a cargar en la placa para realizar la configuración serán los mismos que para la configuración del esclavo. Una vez hecho esto se seguirán los siguientes pasos para configurar el módulo en modo maestro:

- Mientras se mantiene presionado el botón del módulo se conecta la alimentación (no antes). En este momento el módulo entra en modo de configuración y preparado para recibir comandos AT.

- Poner el monitor serie de Arduino en modo 'Ambos NL & CR.

- Escribir el monitor serie los siguientes comandos (si todo funciona el monitor devolverá OK sino Error (0)):

    - AT -> Si devuelve OK el dispositivo está esperando comandos AT

    - AT+NAME=Nombre -> Si se le quiere cambiar el nombre

    - AT+PSWD=Contraseña -> La contraseña que se utilizará para comunicarse con el dispositivo esclavo (Debe ser la misma para ambos).

    - AT+UART=4800,0,0 -> Para configurar la tasa de baudios de la comunicación (En este caso 4800).

    - AT+ROLE=1 -> Configurar el dispositivo en modo esclavo.

    - AT+BIND=Dirección MAC del esclavo -> Para conectarse con el dispositivo esclavo que está en esa dirección.

Una vez configurados ambos módulos como se ha indicado, se podrá comprobar que ambos módulos están emparejados porque al configurar ambos módulos, apagarlos y volverlos a encender (sin presionar el botón esta vez) se verá como el parpadeo del led de ambos módulos es distinto al anterior y ambas están sincronizados, de forma que primero parpadea el módulo maestro y poco después el módulo esclavo.

5.2. SENSORES FLEXIBLES

Este componente es un tipo de sensor que se encarga de detectar el grado de curvatura del elemento al que está unido. Está diseñado para cambiar su resistencia eléctrica en respuesta a la flexión, lo que permite medir el grado de flexión o movimiento de un objeto.

El sensor flexible consiste en una película delgada y flexible con una resistencia eléctrica variable que cambia en función de la cantidad de flexión o curvatura a la que está sometido. Cuando el sensor se flexiona, la distancia entre los electrodos dentro del sensor cambia, lo que a su vez cambia la resistencia eléctrica del sensor. Esto permite al sensor convertir la flexión o curvatura en una señal eléctrica que puede ser procesada y utilizada para controlar un dispositivo o enviar datos a un sistema de registro. Son típicamente utilizados en guantes robóticos ya que pueden registrar la cantidad de movimiento y curvatura de los dedos.

Este componente consta de dos patillas de conexión. Para su utilización habrá que conectar una de sus patillas a uno de los pines analógicos del microcontrolador, ya que

este sensor proporciona valores analógicos en un cierto rango, y a su vez este pin en paralelo a tierra mediante una resistencia de 10k, por último, conectar la otra patilla del sensor a Vcc del microcontrolador. El conexionado sería el siguiente:

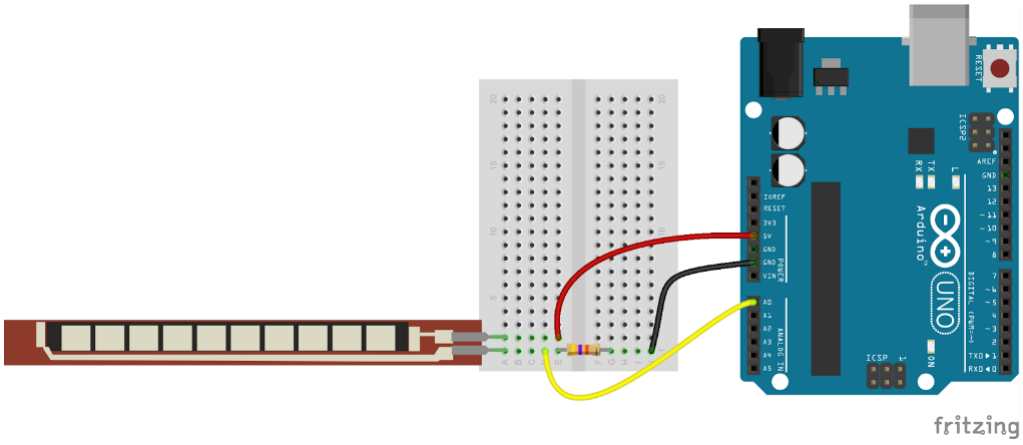


Ilustración 29. Conexionado sensor flexible

En cuanto al software para su utilización solo habrá que definir los pines de conexión y leer el valor proporcionado (analogRead()) por el sensor y utilizarlo para lo que se requiera.

5.3. MPU6050

EL MPU6050 es una unidad de medición inercial o IMU (Inertial Measurment Units) de 6 grados de libertad, con un acelerómetro y un giroscopio integrados. Este componente sirve para medir distintas variaciones de movimiento gracias a sus dos módulos, acelerómetro y giroscopio.

Un acelerómetro es un dispositivo de medición que se utiliza para medir la aceleración lineal de un objeto. Es capaz de detectar cambios en la velocidad o en la dirección de movimiento en una o más direcciones. Los acelerómetros suelen ser sensores electrónicos que miden la fuerza que actúa sobre un objeto en una dirección específica. Un acelerómetro consta de una masa unida a un resorte, que se encuentra en un estado de equilibrio en reposo. Cuando el objeto en el que se encuentra el acelerómetro se acelera en una dirección específica, la masa se desplaza de su posición de equilibrio debido a la fuerza que actúa sobre ella. Esta desviación de la masa del estado de equilibrio es medida por el sensor y se convierte en una señal eléctrica que puede ser procesada para obtener información sobre la aceleración.

Un giroscopio es un dispositivo de medición que se utiliza para medir la velocidad angular o la tasa de rotación de un objeto en torno a un eje. El giroscopio mide la velocidad angular del objeto en relación con un punto de referencia establecido, lo que permite la detección de cambios en la dirección y la velocidad de la rotación del objeto. El giroscopio consta de un rotor que gira alrededor de un eje central. Cuando el giroscopio es sometido a una rotación en torno a otro eje, la fuerza de Coriolis provoca

una desviación en el eje de rotación del rotor. Esta desviación se mide y se convierte en una señal eléctrica que puede ser procesada para obtener información sobre la velocidad angular.

Además, el MPU dispone de un módulo de comunicación I2C (explicada previamente) que le permite comunicarse de forma sencilla y sin necesidad de utilización de muchos pines con el microcontrolador.

Las IMU son típicamente usadas en dispositivos o sistemas que requieren de un control y mediciones de su posición y velocidad, como por ejemplo drones.

En este caso harán falta dos MPU6050, ya que, solo serán útiles dos valores obtenidos de su medición, la aceleración en el eje x y la aceleración en el eje y. La aceleración en el eje z no proporciona información útil, ya que está afectada por la aceleración de la gravedad, y los valores medidos por el giroscopio tampoco proporcionan valores fiables y útiles para el control de la posición de la mano en este proyecto.

En cuanto al conexionado para su funcionamiento, habrá que conectar los dos pines destinados a la comunicación I2C y los pines de alimentación. La conexión sería la siguiente:

- Vcc (Arduino) -> Vcc (MPU6050)
- GND (Arduino) -> GND (MPU6050)
- A4 (Arduino) -> SDA (MPU6050)
- A5 (Arduino) -> SCL (MPU6050)
- Vcc o GND (Arduino) (0x69, 0x68) -> AD0 (MPU6050)

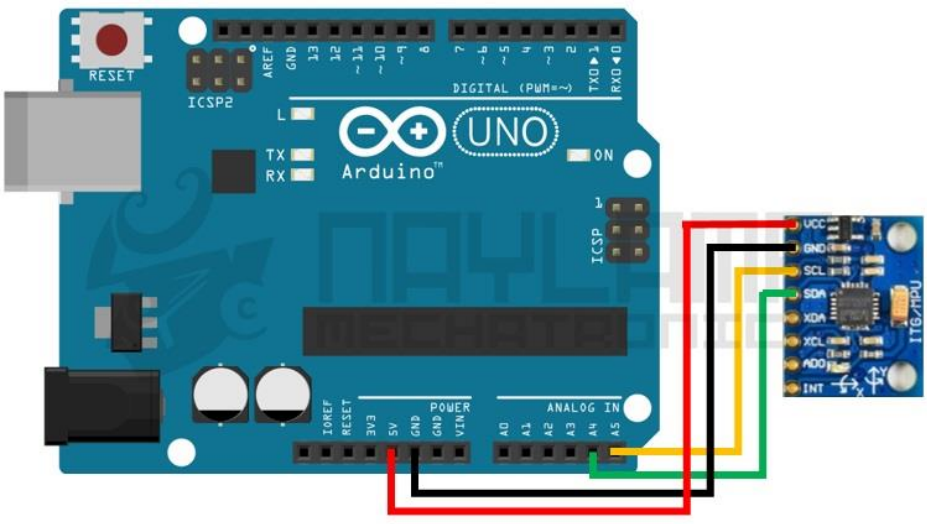


Ilustración 30. Conexionado MPU6050

En cuanto al software necesario para su funcionamiento, primero habrá que definir la dirección I2C de los módulos, ya que, por defecto viene la dirección 0x68 y al utilizar dos MPU en este proyecto, controlados con el mismo Arduino, en caso de no cambiar la dirección de uno de los dos MPU a 0x69 (la otra dirección válida para el módulo) los datos de ambos módulos se mezclarían y daría lugar a funcionamientos erráticos.

- #define MPU6050_ADDRESS_1 0x68 //Dirección del primer MPU
- #define MPU6050_ADDRESS_2 0x69. //Dirección del segundo MPU

Aunque se podría utilizar directamente este módulo de forma 'manual', lo interesante de utilizar Arduino como se ha comentado anteriormente es la gran variedad de recursos que existen, en este caso en forma de librerías, por lo tanto, para la utilización de este módulo se utilizará en librería realizada específicamente para él, la librería MPU6050.h.

Para la utilización de esta, primero habrá que definir dos instancias de tipo MPU6050 (una para cada MPU) asociadas a las dos direcciones I2C previamente definidas:

- MPU6050 mpu0(MPU6050_ADDRESS_1);
- MPU6050 mpu1(MPU6050_ADDRESS_2);

Una vez se tiene esto, para leer los datos recogidos por el MPU solo habrá que hacer uso de la función:

- mpu.getMotion6(&ax0, &ay0, &az0, &gx0, &gy0, &gz0);

Que recogerá los valores de aceleración lineal y angular y los guardará en las variables previamente definidas para este uso.

5.4. CONEXIONADO GENERAL DEL GUANTE

A continuación, se muestra el conexionado general de todos los componentes del subsistema guante.

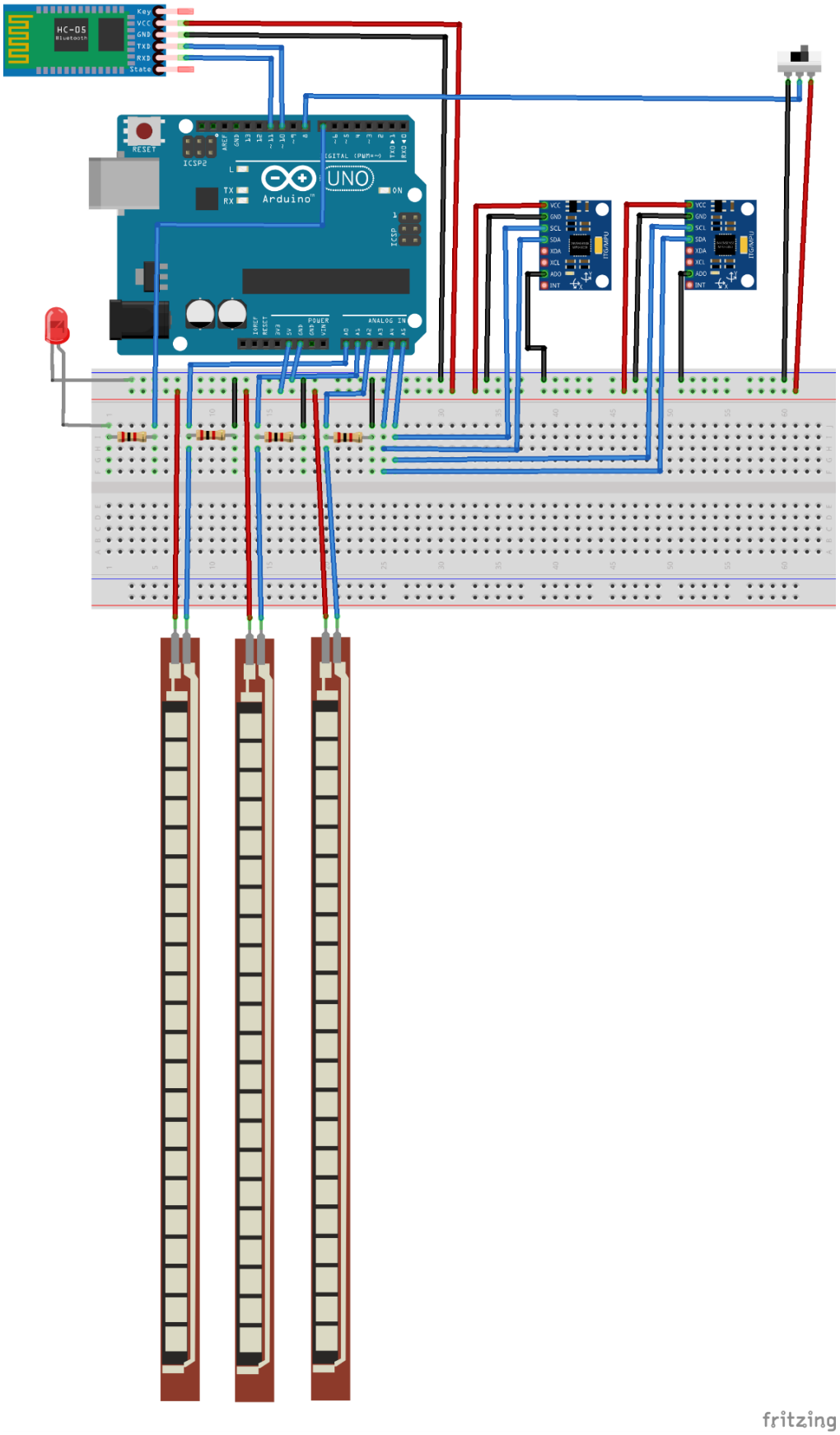


Ilustración 31. Conexionado general del guante

5.5. PRUEBAS Y EXPERIMENTOS

Al igual que para el brazo para la calibración y comprobación del correcto funcionamiento de los componentes se hicieron una serie de pruebas y experimentos.

5.5.1. COMUNICACIÓN BLUETOOTH

Inicialmente tras configurar ambos módulos bluetooth y comprobar que ambos estaban emparejados se probó a enviar un carácter desde el módulo maestro (guante) al (esclavo). Inicialmente los pines RX y TX del Arduino y los pines RX y TX del HC-05 se habían conectado juntos, esto da lugar a que durante el proceso de programación del Arduino el programa dé un error. Para evitar esto se hizo uso de la librería SoftwareSerial.h y de esta forma designar dos pines que actuaran a modo de puerto serie del módulo bluetooth.

El programa funcionaría utilizando los puertos RX y TX del Arduino, pero habría que desconectar esos dos pines cada vez que se quisiese reprogramar el Arduino, lo cual no resulta cómodo, además con esta librería se podría crear más de un puerto serie lo cual podría ser útil en posibles ampliaciones o mejoras del proyecto. Este proceso se realizó tanto para el HC-05 maestro como para el esclavo.

Una vez hecho esto se programaron ambos Arduino (de la parte emisora y de la receptora) y se comprobó que la emisión y la recepción de datos era correcta.

5.5.2. PRUEBAS MPU6050

Para comprobar el funcionamiento del MPU se conectó y se realizó un programa inicial que imprimiera por pantalla los valores recibidos del sensor. Durante este proceso se comprobaron dos cosas.

Muchos de los valores obtenidos por el MPU no eran útiles, ya que para los movimientos que se iban a realizar para controlar el brazo no aportaba datos fiables. Primero, el valor de aceleración en el eje z, estaba afectado directamente por el valor de la gravedad por lo que no servía. Y los valores proporcionados por el giroscopio proporcionaban valores instantáneos durante la rotación del MPU que no eran útiles, ya que, para el control del brazo, interesaban valores constantes según el movimiento que se realizase.

Además, se comprobó que en la posición que se consideraría de reposo para el brazo y para el guante el MPU no proporcionaba valores nulos, esto se debe a que el durante el montaje sobre la protoboard o incluso la soldadura de las patillas de conexión no quede perfectamente perpendicular a la vertical que es la posición de reposo que tiene el sensor, ya que, está afectado por la gravedad. Además, interesa que el brazo no se mueve por cualquier mínimo movimiento involuntario, por todo ello viendo los valores recibidos por el sensor en reposo y el rango de datos que ofrece (16 bits) se le aplicó un valor de offset positivo y negativo, de forma que los valores en el rango de más menos ese offset se consideran nulos y no provocan reacción en el brazo.

5.5.3. PRUEBAS SENSOR FLEXIBLE

Para el sensor flexible de igual forma que para el MPU se comprobó que en su situación de reposo se leía un cierto valor por el canal analógico, si se doblaba el sensor ese valor bajaba. De la misma forma que para el sensor inercial se utilizó un offset de forma que si el valor leído recibido del sensor no estaba por debajo de dicho valor no se reaccionaba.

El valor recibido del sensor flexible dependerá de la resistencia colocada para el circuito del sensor.

6. Software control

Para el software se utilizará lenguaje C para Arduino, en el entorno desarrollado para el propio Arduino como se ha comentado anteriormente.

El código, al igual que en la parte hardware se dividirá en dos zonas bien diferenciadas, una para cada uno de los microcontroladores utilizados, el del guante y el del brazo.

El funcionamiento general del sistema consistirá en un microcontrolador que se encargará del control del funcionamiento del guante. Este estará continuamente leyendo la información recibida por los sensores integrados en el guante, la procesará y la enviará de forma inalámbrica a través de bluetooth al microcontrolador encargado del control del brazo. El microcontrolador encargado del control del brazo recibirá los datos obtenidos de los sensores, procesará esta información y enviará órdenes a los motores encargados del movimiento del brazo en función de los datos recibidos.

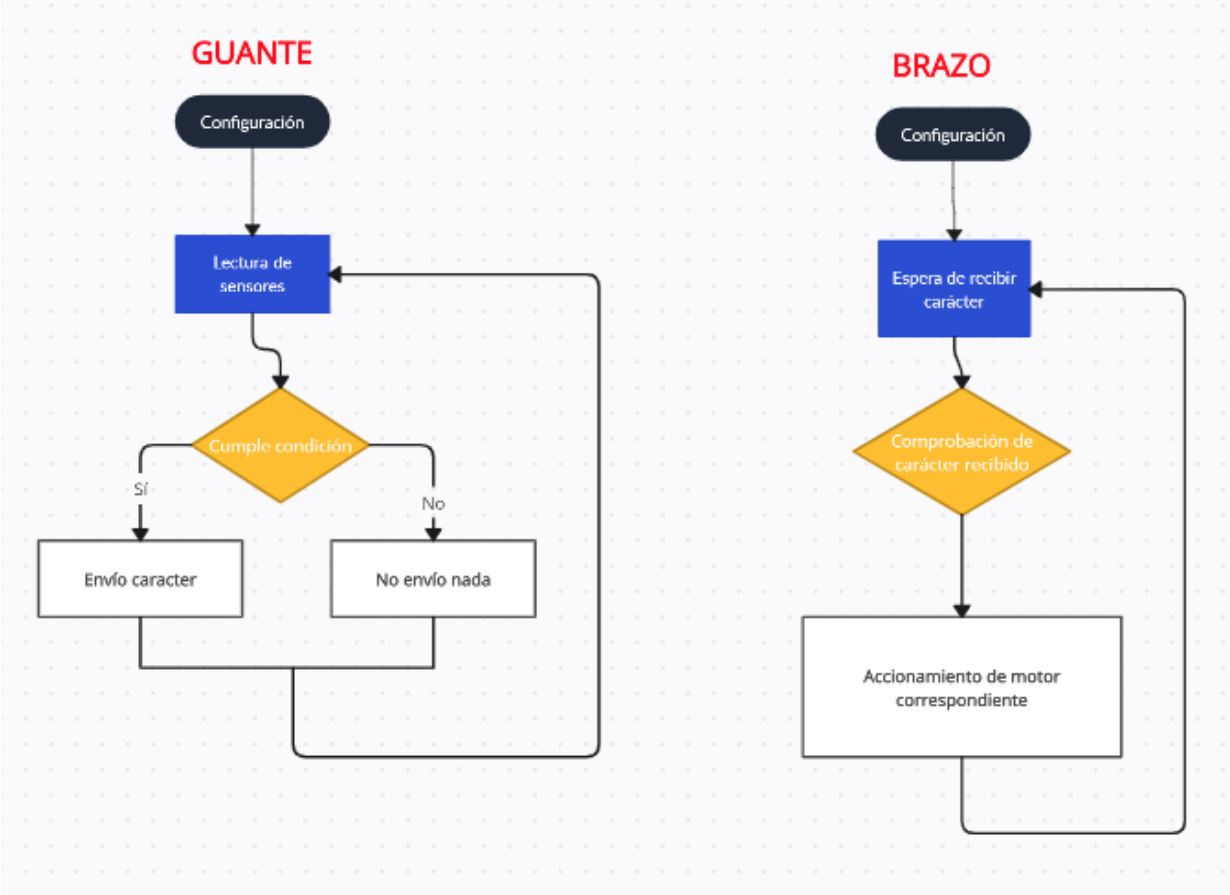


Ilustración 32. Flujograma

6.1. CÓDIGO GUANTE

Como se ha dicho este programa estará constantemente recibiendo datos de los sensores y enviándolos al brazo de forma inalámbrica.

Para ello inicialmente, se incluyen las librerías a utilizar que serán:

- SoftwareSerial.h
- MPU6050.h
- I2Cdev.h
- Wire.h.

Como ya se ha comentado la primera se utilizará para crear nuevos puertos serie y la segunda para el control de los MPU. Las dos últimas se incluyen porque son utilizadas para realizar la comunicación I2C por parte de la librería MPU6050.h.

Posteriormente se procede a la definición de las instancias y variables que se utilizarán en el programa:

- Se define la dirección de los MPU.
- Se definen las instancias para el módulo bluetooth y los dos MPU.
- Se definen las variables donde se guardarán los datos de los MPU.
- Se definen los valores de offset de los sensores.
- Se definen los pines para el interruptor y el led de ON/OFF del guante.
- Se definen los pines que usarán los sensores flexibles y la variable donde guardarán los datos recibidos.

En la función de configuración setup() se configuran los parámetros del programa:

- Se inicia la comunicación serie y se declaran los baudios de esta.
- Se inicia la comunicación bluetooth y se declaran los baudios.
- Se inician los dos MPU.
- Se declara como entrada el pin del interruptor.
- Se declara como salida el pin del led.

En el bucle de ejecución loop():

- Se comprueba el estado del interruptor. Si está en estado alto se enciende el led y se ejecutará todo el resto del programa de forma normal. En caso contrario, se apaga el led y no se hace nada.
- Una vez se ha comprobado el interruptor en estado alto. Se leen los valores de todos los sensores (2 MPU y 3 Flex).

- Posteriormente, se procede a comprobar si los valores de interés obtenidos por los sensores son superiores o inferiores a los offset fijados previamente, en función de si son superiores o inferiores se enviarán diferentes caracteres.
    - Flexión flex1 -> ‘A’
    - Flexión felx2 -> ‘a’
    - Rotación adelante MPU0 -> ‘B’
    - Rotación atrás MPU0 -> ‘b’
    - Rotación adelante MPU1 -> ‘C’
    - Rotación atrás MPU1 -> ‘c’
    - Rotación horaria MPU0 -> ‘D’
    - Rotación antihoraria MPU0 -> ‘d’
    - Rotación horaria MPU1 -> ‘E’
    - Rotación antihoraria MPU1 -> ‘e’
    - Flexión flex0 -> ‘F’

Con este programa se obtendrían los valores de los sensores y se enviarían los datos necesarios para el control del brazo.

En varios puntos del programa hay instrucciones comentadas que se encargarían de imprimir por pantalla los valores medidos de los sensores. Estas instrucciones se descomentarían ya que serían útiles en caso de tareas de mantenimiento o ajuste de los valores de los sensores. Pero que para el funcionamiento normal del programa no pueden ejecutarse, ya que, son un número considerable de instrucciones de print que consumen recursos del programa y se ha comprobado que retrasan considerablemente el funcionamiento del mismo.

6.2. CÓDIGO BRAZO

El programa de la parte del brazo se encargará de recibir los datos enviados por parte del guante y enviar las órdenes de movimiento a los motores del brazo en función de dichos datos.

Inicialmente se incluyen las librerías:

- SoftwareSerial.h
- HCPCA9685.h
- AccelStepper.h

La primera igual que en el caso del guante es para crear nuevos puertos serie para el Arduino uno, el cual solo dispone de uno. La segunda se utiliza para controlar los servomotores a través del driver PCA9685 de forma sencilla. La última sirve para el control del motor paso a paso de la base.

Posteriormente se pasa a la parte de definición de variables y parámetros:

- Se define la dirección I2C del driver para servos, y los pines de control del driver para el motor PaP.
- Se crean instancias para los datos de tipo HCPCA9685, AccelStepper y SoftwareSerial.
- Se crean varias variables auxiliares que se usarán en el programa principal varias veces, de forma que queda una sintaxis más clara del programa y en caso de querer cambiarlas se cambiarán todas a la vez.
- Se define el valor para las posiciones máximas y mínimas de los servomotores. Y se declara la variable de control para cada uno de los servomotores (duty).

En la parte de configuración setup():

- Se inicia el driver para servos y se fija su frecuencia de trabajo.
- Se fija la velocidad y aceleración máxima del motor paso a paso.
- Se inicia la comunicación serie y se fijan los baudios de la misma en 9600.
- Se inicia la comunicación bluetooth y se fijan los baudios en 4800.

En el bucle principal del programa, inicialmente se ejecuta una secuencia de instrucciones para fijar el brazo en la posición inicial, esta fase solo se ejecuta en el arranque del brazo. Posteriormente, se dispone a comprobar de forma continua si hay mensajes disponibles en el módulo bluetooth recibidos del guante. En caso negativo se sigue esperando por un mensaje. En caso positivo se lee dicho mensaje y en función de este, se procederá a darle una orden de movimiento u otra a los motores del brazo:

- Si ‘A’ -> Movimiento horario de la base
- Si ‘a’ -> Movimiento antihorario de la base
- Si ‘B’ -> Movimiento hacia arriba de la articulación 2
- Si ‘b’ -> Movimiento hacia abajo de la articulación 2
- Si ‘C’ -> Movimiento hacia arriba de la articulación 4
- Si ‘c’ -> Movimiento hacia abajo de la articulación 4

- Si ‘D’ -> Movimiento hacia arriba de la articulación 5
- Si ‘d’ -> Movimiento hacia abajo de la articulación 5
- Si ‘E’ -> Movimiento horario de la articulación 6
- Si ‘e’ -> Movimiento antihorario de la articulación 6
- Si ‘F’ -> Apertura pinza
- Si no ‘F’ -> Cierre pinza

Con este programa se reciben las órdenes de control por parte del guante y se envían las órdenes para el control de los motores del brazo.

7. Diseño PCB

Para el ensamblaje de todos los componentes electrónicos tanto del brazo como del guante se diseñaron dos PCBs para su montaje.

Para el diseño se utilizó el programa easyeda. Este programa es un software online para diseño de PCBs muy útil, ya que, tiene una librería muy amplia con numerosos componentes creados por los usuarios además de los propios del programa. Se utiliza de forma bastante sencilla a la vez que tiene funcionalidades de programas bastante sofisticados.

Las PCBs se han diseñado con objetivo de ser un modelo de shield de Arduino que se colocará sobre él, y a su vez sobre esta shield se conectarán el resto de los componentes del circuito.

7.1. PCB GUANTE

El diseño de la PCB para integrar los componentes del guante sería el siguiente:

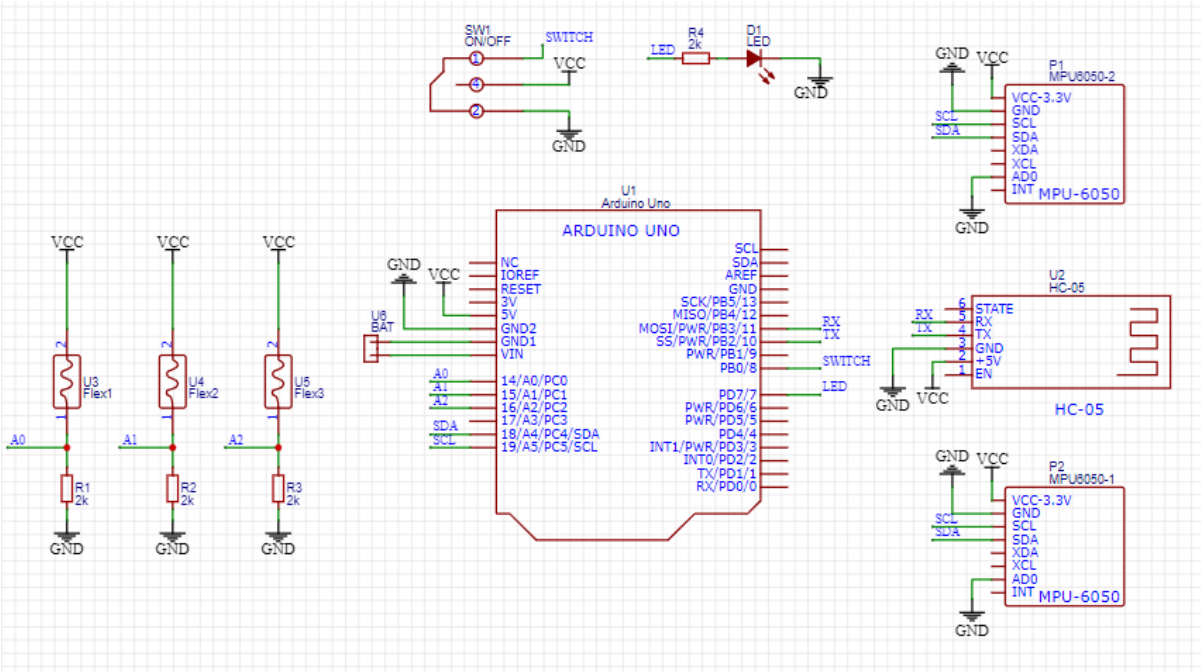


Ilustración 33. Esquemático guante

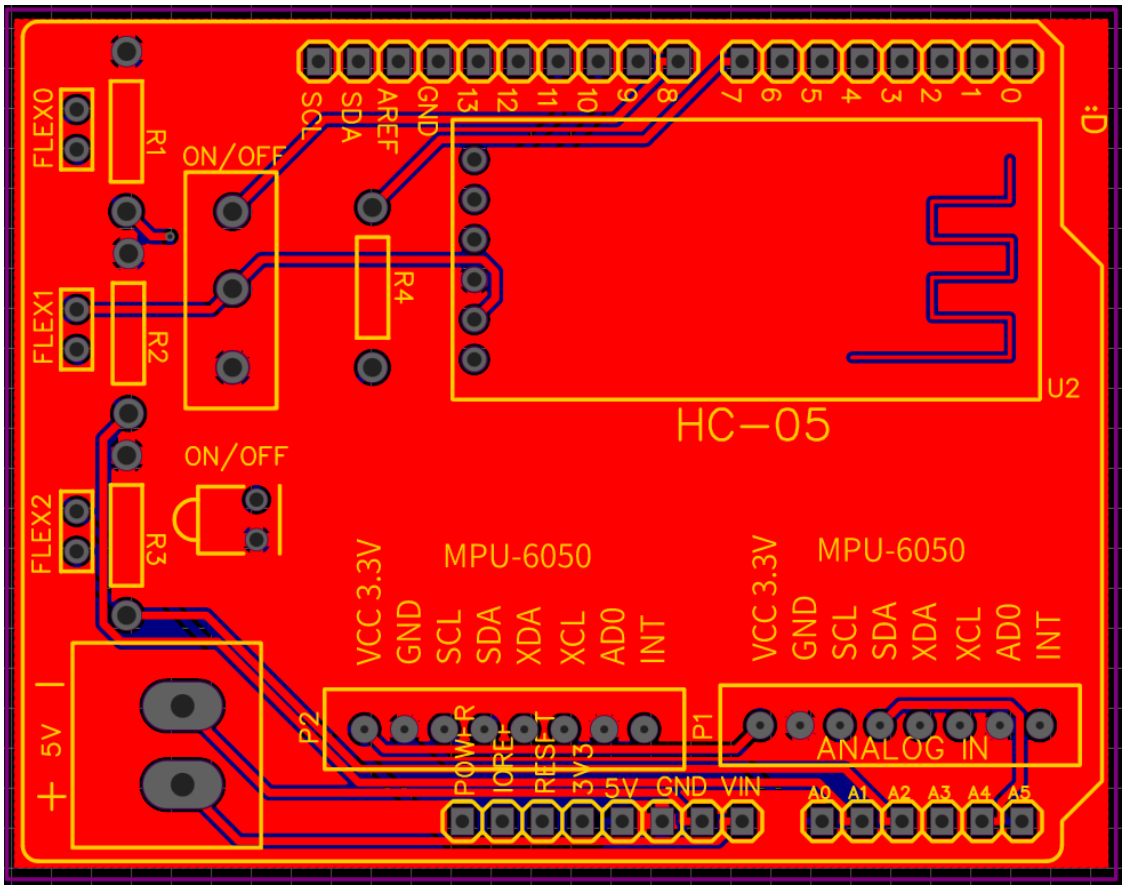


Ilustración 34. PCB guante

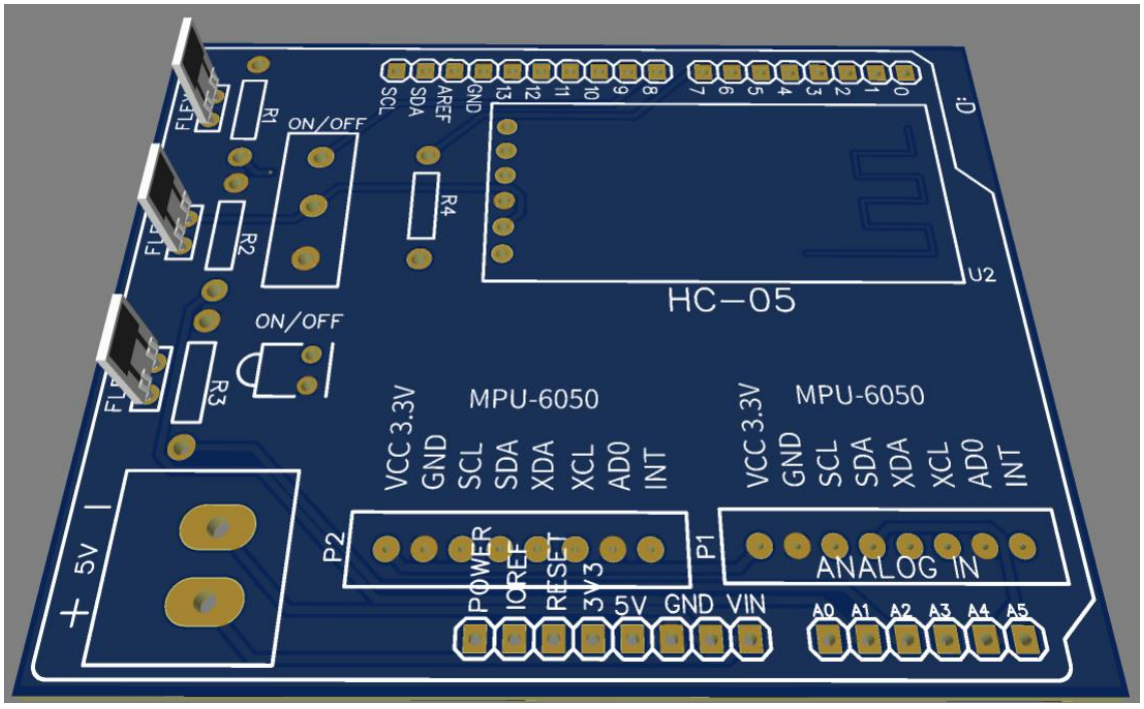


Ilustración 35. 3D PCB guante

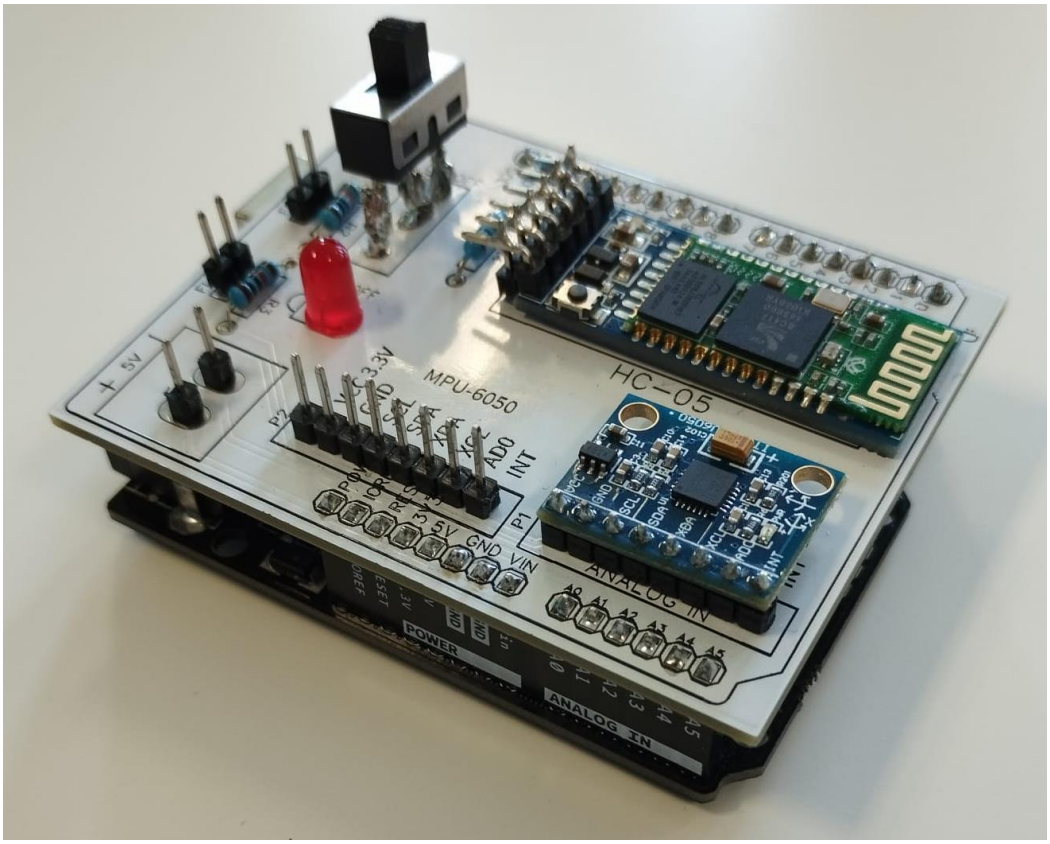


Ilustración 36. PCB guante real

La mano uno será el componente principal del guante, donde estarán la mayoría de los componentes como se puede ver. Sobre ella irán colocados dos sensores flexibles, uno de los MPU, el módulo bluetooth, un interruptor con el que se podrá apagar o encender el funcionamiento del guante y un led que indicará el estado de encendido o apagado del guante. Aunque aparezcan sobre la PCB también el segundo MPU y el tercer flexómetro, estos solo representan conexiones, ya que, en realidad irán colocados en la mano dos por los motivos que se explican a continuación.

En la mano dos irán colocados solamente uno de los acelerómetros y uno de los flexómetros para que el acelerómetro dos no detecte los mismos datos que el uno, que sería la que pasaría en caso de ir los dos sobre la misma mano. Se conectarán mediante cables largos, que permitan un rango cómodo de movimientos, a la mano uno donde se encuentra el Arduino que controlará y procesará los datos de los sensores.

7.2. PCB BRAZO

Por último, la PCB diseñada para el brazo también está diseñada para ir colocada sobre el Arduino Uno, y sobre ella irán conectados los componentes que controlan el sistema del brazo: el módulo bluetooth, el driver 8825 para el motor paso a paso, el motor paso a paso, el driver PCA9685 para los servomotores (el propio driver ya trae los conectores para los servomotores, por eso no hace falta colocarlos sobre la PCB) y los conectores de alimentación (5V para los servomotores y 12V para el motor paso a paso).

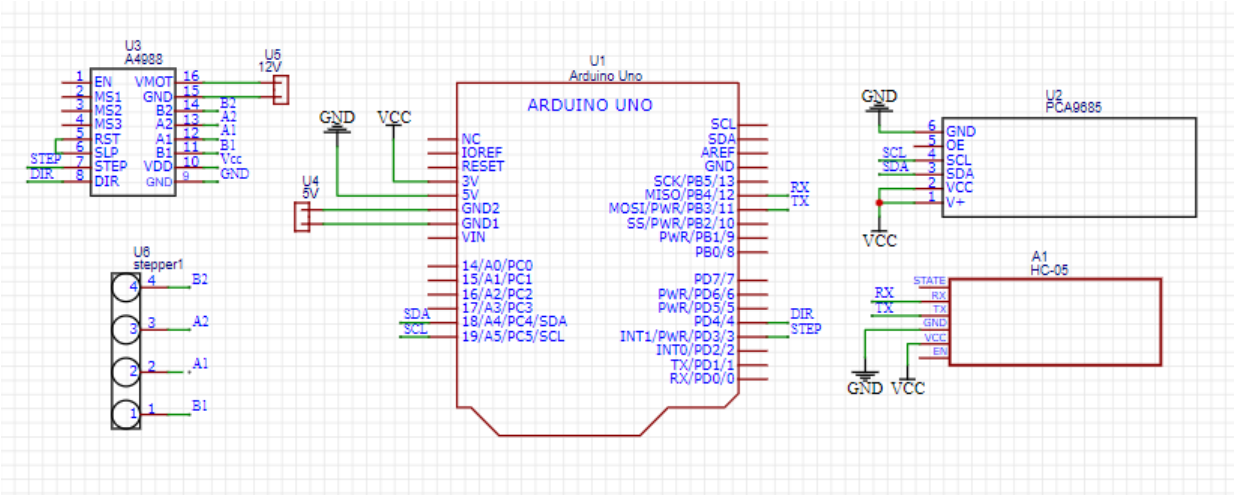


Ilustración 37. Esquemático brazo

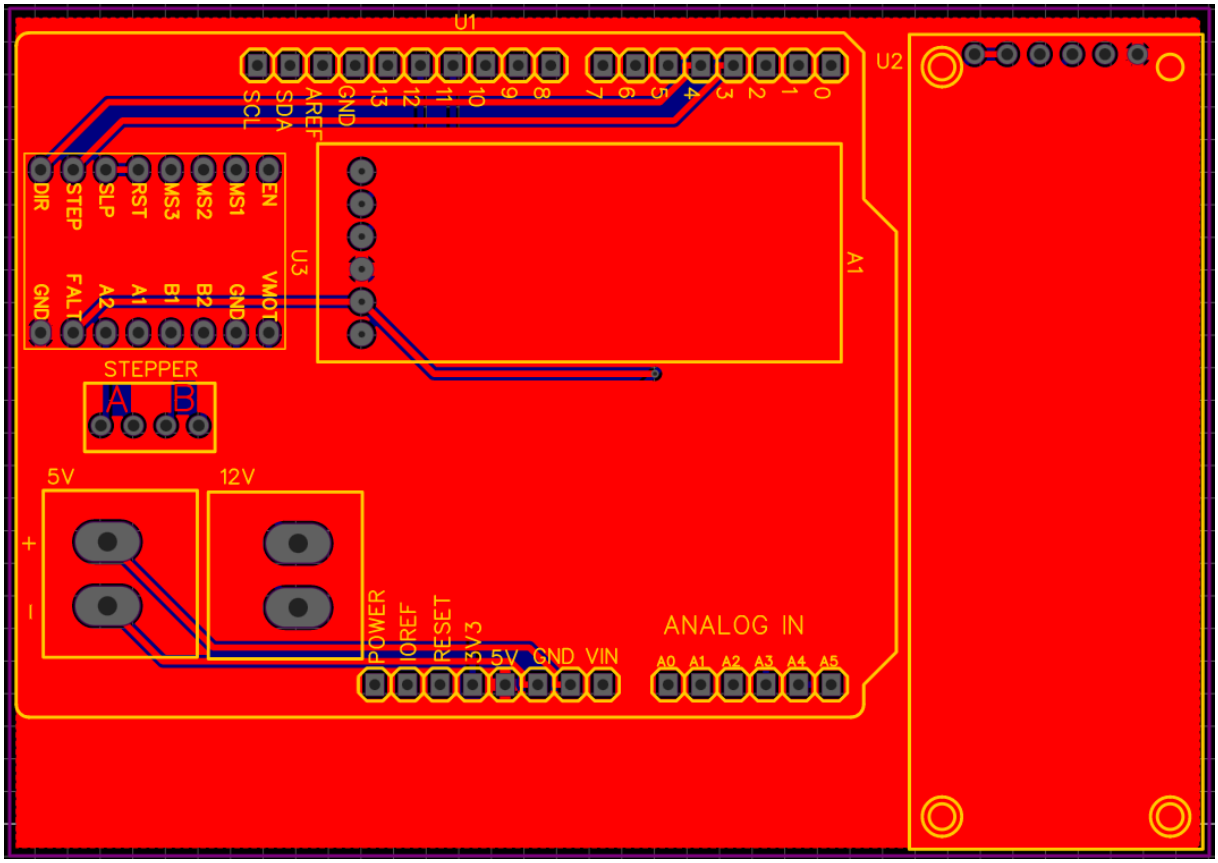


Ilustración 38. PCB brazo

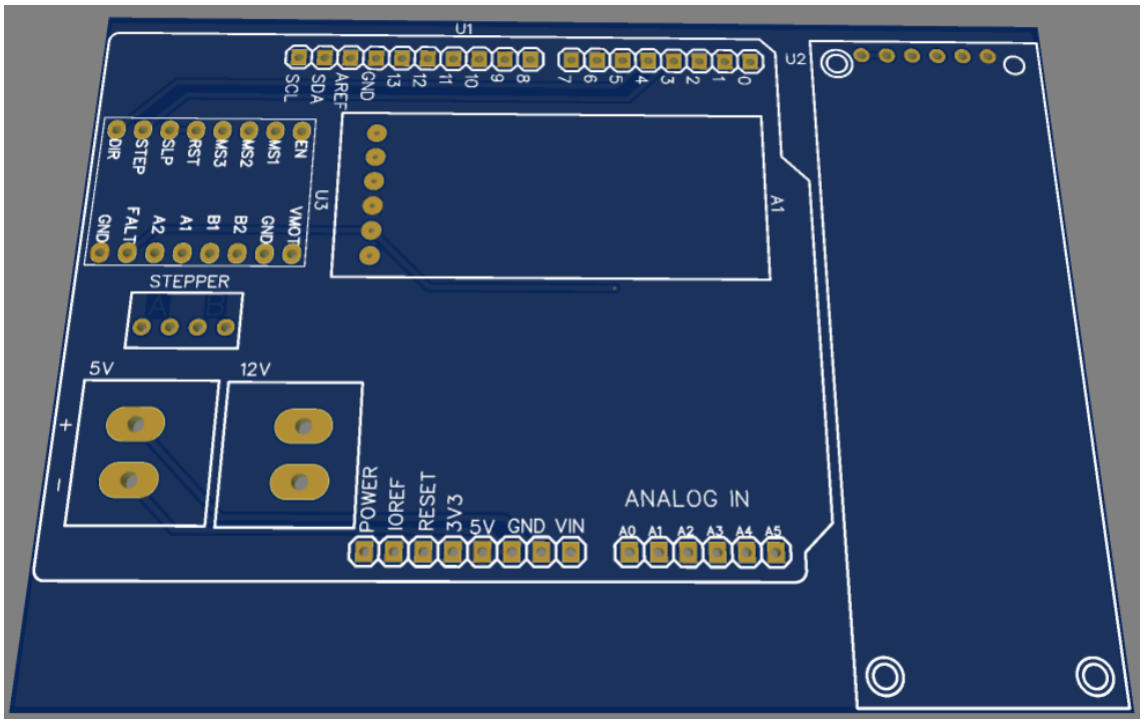


Ilustración 39. 3D PCB brazo

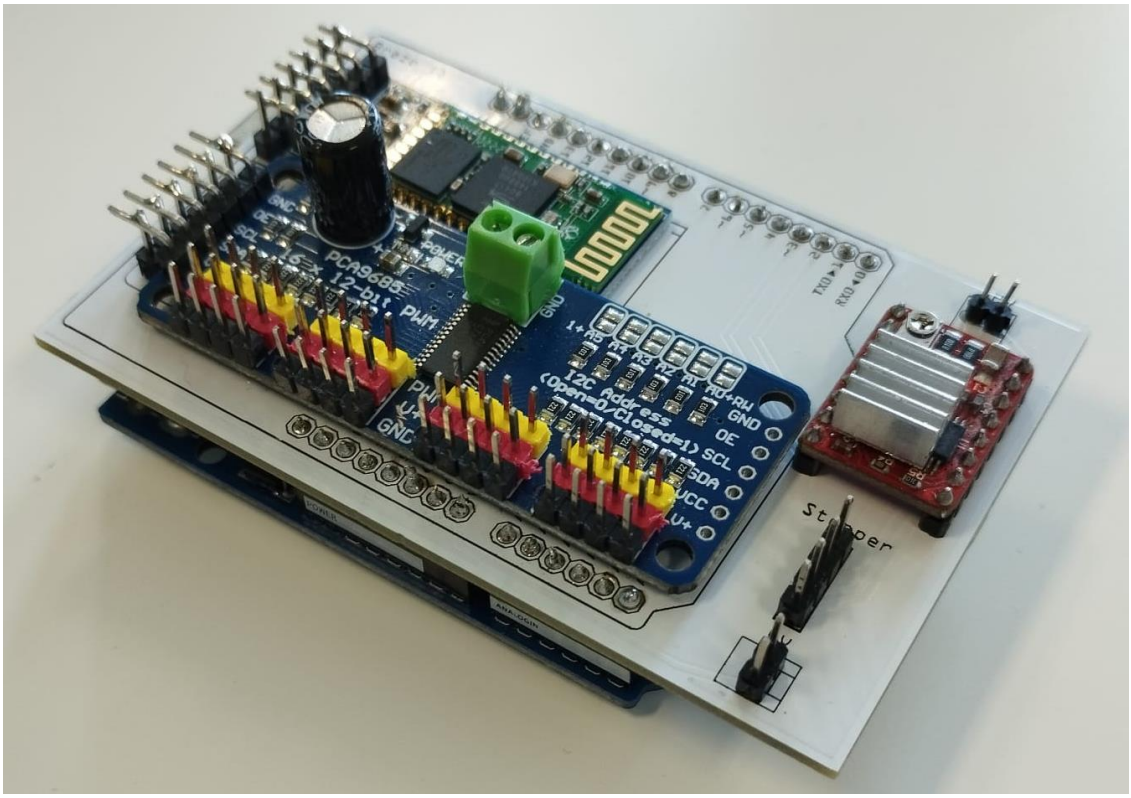


Ilustración 40. PCB brazo real

8. Conclusiones y trabajos futuros

El objetivo desde el inicio del presente Trabajo Fin de Grado ha sido el diseño y desarrollo de un sistema de control para un brazo robótico. Este brazo robótico está diseñado para su utilización en entornos de entretenimiento o educativos. Con el desarrollo de este prototipo también se pretendía tener contacto con diversos componentes electrónicos no utilizados durante el desarrollo del grado, por lo cual el desarrollo de este proyecto serviría como un buen complemento final para la realización de los estudios.

Para el primero de ellos, se ha diseñado y desarrollado un sistema de control que se puede considerar bastante adecuado para los fines para los que está diseñado el brazo robótico, la educación y el entretenimiento, ya que, el sistema de control mediante gestos manuales es un método vistoso y fácil para poder controlar el brazo robótico. En caso de un control para otro tipo de usos, como un uso industrial, esto no sería lo más adecuado. Por lo tanto, con fin de cumplir este objetivo se ha desarrollado un algoritmo que se ajusta correctamente y consigue el funcionamiento explicado con éxito.

En cuanto al segundo objetivo también se puede considerar cumplido ya que, prácticamente la totalidad de los componentes utilizados para el desarrollo de este proyecto no se habían utilizado durante el desarrollo del Grado. Se han utilizado numerosos componentes electrónicos, tales como, acelerómetros, sensores flexibles, servomotores, motor paso a paso, etc. Ninguno de ellos se había utilizado con anterioridad, por lo que hubo que realizar un proceso de estudio y recopilación de información sobre dichos componentes. También se han utilizado varios softwares nuevos, tales como, Arduino IDE para la programación o Easyeda para el diseño de las PCBs.

Además, se ha aprendido el cual sería el proceso de diseño completo de un prototipo electrónico siguiendo todos sus pasos, recogida de información, ideas de diseño, selección de componentes óptimos, programación, diseño de PCBs, etc.

Como parte complementaria a los objetivos principales, también se ha tenido que trabajar dentro de un proyecto que involucra más partes de la ingeniería además de la electrónica, en este caso la mecánica. Esta idea también es importante, ya que, en el desarrollo normal de un proyecto real de la industria es habitual trabajar en conjunto con distintas áreas de la ingeniería, las cuales habrá que tener en cuenta para realizar los trabajos encomendados en el área de la electrónica, como en este caso.

Por todo ello, se considera que los objetivos que se tenían desde un inicio se han cumplido y el proyecto se ha desarrollado con éxito.

Como algunas posibles mejoras o ampliaciones en cuanto al sistema de control del brazo robótico se considera la posibilidad de implementar nuevos sistemas de control para el brazo robótico.

El objetivo de esta idea sería desarrollar un sistema de control con el cual se pudiese seleccionar el sistema de control actual (gestos manuales), además de otros tipos de sistemas. Por ejemplo, se podría utilizar un interruptor con varias posiciones con el cual se seleccionase el tipo de sistema de control.

En cuanto a los sistemas de control a implementar, se podría realizar algunos de los comentados durante el desarrollo del trabajo, tales como los desarrollados mediante modelos cinemáticos o mediante control remoto a través de un mando de control remoto o una computadora.

Además, en cuanto a una mejora necesaria en el campo de la ingeniería mecánica, habría que diseñar y fabricar la estructura que soporte los sensores de la parte del ‘guante robótico’.

Esto haría más vistoso el proyecto para un uso recreativo, y en caso del uso educativo enseñaría diversos métodos de control, algunas más orientados a un uso recreativo como el implementado para este trabajo y otros más industriales como es el caso del control mediante modelos cinemáticos.

9. Bibliografía

[1] I. C. González, «TFG Diseño de un brazo robótico fabricable mediante impresora 3D,» 2018.

[2] J. C. R. Zambrana, «Modelo cinemático y control de un brazo robótico imprimible».

[3] V. P. Muñiz, «TFG Brazo robótico de 3 GDL de bajo coste controlado con Arduino,» 2019.

[4] J. I. Trujillo, «TFG Brazo robot de bajo costo con motores paso a paso que escribe,» 2020.

[5] Á. G. López, «TFG Brazo robótico controlado por gestos humanos planos,» 2022.

[6] Á. G. López, «TFG brazo robótico controlado por gestos manuales memoria,» 2022.

[7] «16 salidas servodriver PCA9685,» 2023. [En línea]. Available: https://www.luisllamas.es/controlar-16-servos-o-16-salidas-pwm-en-arduino-con-pca9685/.

[8] «A000066,» 2023. [En línea]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf.

[9] «Arduino,» 2023. [En línea]. Available: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno.

[10] «GitHub,» 2023. [En línea]. Available: https://github.com/.

[11] «Módulo HC-05,» 2023. [En línea]. Available: https://naylampmechatronics.com/inalambrico/43-modulo-bluetooth-hc05.html.

[12] «Motor paso a paso,» 2023. [En línea]. Available: https://www.tme.eu/es/news/library-articles/page/41861/Motor-paso-a-paso-tipos-y-ejemplos-del-uso-de-motores-paso-a-paso/.

[13] «MPU6050,» 2023. [En línea]. Available: https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html.

[14] «PCA9685,» 2023. [En línea]. Available: https://naylampmechatronics.com/blog/41_tutorial-modulo-controlador-de-servos-pca9685-con-arduino.html.

[15] «PCA9685,» 2023. [En línea]. Available: https://www.aranacorp.com/es/usando-un-modulo-pca9685-con-arduino/.

[16] «Robot arm,» 2023. [En línea]. Available: https://smartbuilds.io/diy-robot-arm-arduino-hand-gestures/.

[17] «Robótica,» 2023. [En línea]. Available: https://revistaderobots.com/robots-y-robotica/que-es-la-robotica/.

[18] «Robótica,» 2023. [En línea]. Available: https://concepto.de/robotica/.

[19] «Servodriver 16 channel,» 2023. [En línea]. Available: https://cdn-learn.adafruit.com/downloads/pdf/16-channel-pwm-servo-driver.pdf.

[20] «Robótica,» 2023. [En línea]. Available: https://www.edsrobotics.com/blog/que-es-la-robotica/.

[21] «Servomotores,» 2023. [En línea]. Available: https://www.cursosaula21.com/que-es-un-servomotor/.

[22] «Datasheet PCA9685,» [En línea].

[23] «Datasheet DRV8825,» [En línea].

[24] «Datasheet MG996R,» [En línea].

[25] «Datasheet NEMA 17 JK42Hs,» [En línea].

[26] «Datasheet HC-05,» [En línea].

[27] «Datasheet MPU-6050,» [En línea].

[28] «Datasheet Arduino Uno,» [En línea].

10. Anexos

10.1. PLANIFICACIÓN TEMPORAL

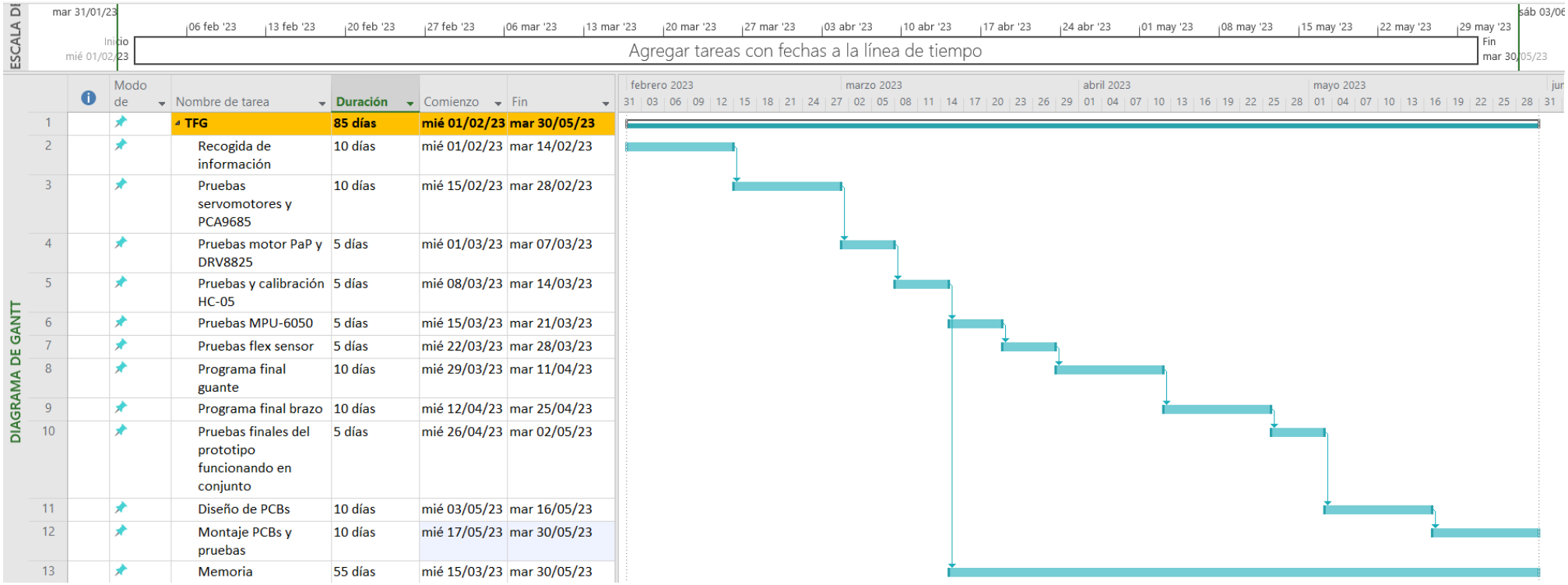


Ilustración 41. Planificación temporal

10.2. CÓDIGOS

10.2.1. CÓDIGO GUANTE

```
////////////////////////////////////////////////////////////
//  PROGRAMA GUANTE ROBÓTICO PARA CONTROL DE BRAZO ROBÓTICO
//
//            Javier Martínez Becerra                 //
////////////////////////////////////////////////////////////

// LIBRERÍAS
#include "SoftwareSerial.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

#define MPU6050_ADDRESS_1 0x68  //Dirección I2C del primer MPU
#define MPU6050_ADDRESS_2 0x69  //Dirección I2C del segundo MPU

SoftwareSerial BTSerial(10,11); //TX / RX auxiliares para no utilizar el puerto del
Arduino

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 mpu0(MPU6050_ADDRESS_1);
MPU6050 mpu1(MPU6050_ADDRESS_2);

// Valores de los dos MPU en los ejes x,y,z
int ax0, ay0, az0;
int gx0, gy0, gz0;
int ax1, ay1, az1;
int gx1, gy1, gz1;

int t=1;                  // Tiempo de respuesta
int offset=3500;      // Offset de los MPU
int offset_flex0=170; // Offset de los flex0
int offset_flex1=160; // Offset de los flex1
int offset_flex2=170; // Offset de los flex2

int interruptor=8;  // Pin interruptor de ON/OFF del guante
int led=7;           // Pin led ON/OFF del guante

int flexPin0 = A0;    // Pin analógico del primer flex sensor
int flexValue0 = 0;   // Valor leído del primer flex sensor
int flexPin1 = A1;    // Pin analógico del segundo flex sensor
int flexValue1 = 0;   // Valor leído del segundo flex sensor
int flexPin2 = A2;    // Pin analógico del tercer flex sensor
int flexValue2 = 0;   // Valor leído del tercer flex sensor

void setup() {
```

```
Serial.begin(9600);     // Baudios comunicación serie
BTSerial.begin(4800);          // Baudios comunicacion BT
Wire.begin();
mpu0.initialize();       // Inicializar el primer MPU-6050
mpu1.initialize();       // Inicializar el segundo MPU-6050
pinMode(interruptor,INPUT);// Interruptor entrada
pinMode(led,OUTPUT);          // Led salida
delay(1000);
}

void loop() {
  //Comprobacion apagado o encendido
  if(digitalRead(interruptor)==HIGH) {

  //Serial.println("ENCENDIDO");  // Guante ON
  digitalWrite(led, HIGH);

  // Lectura de valores del primer MPU
  mpu0.getMotion6(&ax0, &ay0, &az0, &gx0, &gy0, &gz0);

  delay(t);

  // Lectura de valores del segundo MPU
  mpu1.getMotion6(&ax1, &ay1, &az1, &gx1, &gy1, &gz1);

  delay(t);
  // Lectura del primer flex sensor
  flexValue0 = analogRead(flexPin0);

  delay(t);
  // Lectura del segundo flex sensor
  flexValue1 = analogRead(flexPin1);

  delay(t);
  // Lectura del tercer flex sensor
  flexValue2 = analogRead(flexPin2);
  delay(t);

  // Descomentar para comprobar valores medidos por los sensores


/*Serial.println("==========================================
=======");

  Serial.println(flexValue0);
  Serial.println(flexValue1);
  Serial.println(flexValue2);

  Serial.print("MPU1: ");
```

```
Serial.print("ax1=");
Serial.print(ax1);
Serial.print("ay1=");
Serial.print(ay1);
Serial.print("az1=");
Serial.print(az1);
Serial.print("gx1=");
Serial.print(gx1);
Serial.print("gy1=");
Serial.print(gy1);
Serial.print("gz1=");
Serial.println(gz1);

Serial.print("MPU0: ");
Serial.print("ax0=");
Serial.print(ax0);
Serial.print("ay0=");
Serial.print(ay0);
Serial.print("az0=");
Serial.print(az0);
Serial.print("gx0=");
Serial.print(gx0);
Serial.print("gy0=");
Serial.print(gy0);
Serial.print("gz0=");
Serial.println(gz0);

delay(1000);*/

// Comandos enviados por BT para el control del brazo robótico
if(flexValue1<offset_flex1){          // Flexión flex 1
  BTSerial.write('A');
  delay(t);
}

if(flexValue2<offset_flex2){          // Flexión flex 2
  BTSerial.write('a');
  delay(t);
}

if(ax0>offset){                  // Rotación adelante MPU0
  BTSerial.write('B');
  delay(t);
}

if(ax0<-offset){                 // Rotación atrás MPU0
  BTSerial.write('b');
  delay(t);
}
```

```
 if(ax1>offset){              // Rotación adelante MPU1
  BTSerial.write('C');
  delay(t);
 }

 if(ax1<-offset){             // Rotación atrás MPU1
  BTSerial.write('c');
  delay(t);
 }

  if(ay0>offset){             // Rotación horaria MPU0
  BTSerial.write('D');
  delay(t);
 }

 if(ay0<-offset){             // Rotación antihoraria MPU0
  BTSerial.write('d');
  delay(t);
 }

 if(ay1>offset){              // Rotación horaria MPU1
  BTSerial.write('E');
  delay(t);
 }

 if(ay1<-offset){             // Rotación antihoraria MPU1
  BTSerial.write('e');
  delay(t);
 }

 if(flexValue0<offset_flex0){        // Flexión flex 0
  BTSerial.write('F');
  delay(t);
 }
 }

 else{
  //Serial.println("APAGADO");  // Guante OFF
  digitalWrite(led, LOW);
 }
}
```

10.2.2. CÓDIGO BRAZO

```
//////////////////////////////////////////////////////////

//  PROGRAMA PARA CONTROL DE BRAZO ROBÓTICO MEDIANTE GESTOS//

//          Javier Martínez Becerra                 //

//////////////////////////////////////////////////////////


// LIBRERÍAS

#include <SoftwareSerial.h>

#include <HCPCA9685.h>

#include <AccelStepper.h>


#define  I2CAdd 0x40  // Dirección I2C del driver de servos


#define stepPin 2     // Define los pines de control del driver 8825

#define dirPin 3


HCPCA9685 MiServo(I2CAdd);  // Variable de tipo HCPCA9685 para controlar servos


AccelStepper stepper(AccelStepper::DRIVER, stepPin, dirPin);       // Variable AccelStepper para controlar el motor  PaP


char mensaje; // Variable para guardar mensaje leído por BT del guante


SoftwareSerial BTSerial(10,11); //TX / RX auxiliares


int Fase0=1;  // Variable para activar la posición inicial del brazo

int t=10;     // Tiempo entre pasos

int t1=200;   // Tiempo inicial

int pasos=1;  // Pasos por comando
```

```
int pasos_pinza=10;  // Pasos por comando pinza
int aux=0;

// Posiciones máximas y mínimas de los servomotores de las articulaciones
int duty2=230;
int pos0_2=0;
int pos120_2=250;

int duty3=250-duty2;
int pos0_3=-20;
int pos120_3=250;

int duty4=100;
int pos0_4=0;
int pos120_4=120;

int duty5=350;
int pos0_5=140;
int pos120_5=420;

int duty6=200;
int pos0_6=25;
int pos120_6=400;

int duty7=0;
int pos0_7=0;
int pos120_7=400;
```

```
 MiServo.Servo(9,duty7);
 delay(t1);

 Fase0=0;
}

// Comprobar que hay mensaje enviado por BT
if (BTSerial.available()){
mensaje=BTSerial.read();
Serial.println(mensaje);

// Comandos para mover el brazo (servos y PaP) en función de los datos recibidos del guante
if(mensaje=='A'){
  stepper.move(10);       // Movimiento horario de la art 1
  stepper.run();
  delay(t);
}

if(mensaje=='a'){
  stepper.move(-10);   // Movimiento antihorario de la art 1
  stepper.run();
  delay(t);
}

if(mensaje=='B' and (pos0_2<duty2 and pos120_2>=duty2+10)){
 duty2=duty2+pasos;
 MiServo.Servo(0,duty2+22);          // Movimiento hacia arriba de la art 2
```

```
void setup()
{
  MiServo.Init(SERVO_MODE); //Inicializar librería HCPCA9685 en modo Servomotor
  MiServo.Sleep(false);     // Despertar el driver
  MiServo.SetPeriodFreq(50);  // Fijar frecuencia de trabajo
  stepper.setMaxSpeed(50); // Configura la velocidad máxima en pasos por segundo
  stepper.setAcceleration(50); // Configura la aceleración en pasos por segundo por segundo
  Serial.begin(9600);         // Baudios comunicación serie
  BTSerial.begin(4800);       // Baudios comunciación BT
}

void loop()
{
// Fijar la posición inicial del brazo
if(Fase0==1){
 MiServo.Servo(0,duty2+22);
 MiServo.Servo(1,250-duty2);
 delay(t1);

 MiServo.Servo(2,duty4);
 delay(t1);

 MiServo.Servo(3,duty5);
 delay(t1);

 MiServo.Servo(8,duty6);
 delay(t1);
```

```
 MiServo.Servo(1,250-duty2);
 delay(t);
}

if(mensaje=='b' and (pos0_2<duty2-10 and pos120_2>=duty2)){
 duty2=duty2-pasos;
 MiServo.Servo(0,duty2+22);          // Movimiento hacia abajo de la art 2
 MiServo.Servo(1,250-duty2);
 delay(t);
}

if(mensaje=='C' and (pos0_4<duty4 and pos120_4>=duty4+10)){
 duty4=duty4+pasos;
 MiServo.Servo(2,duty4);      // Movimiento hacia arriba de la art 4
 delay(t);
}

if(mensaje=='c' and (pos0_4<duty4-10 and pos120_4>=duty4)){
 duty4=duty4-pasos;
 MiServo.Servo(2,duty4);      // Movimiento hacia abajo de la art 4
 delay(t);
}

if(mensaje=='D' and (pos0_5<duty5 and pos120_5>=duty5+10)){
 MiServo.Servo(3,duty5);      // Movimiento hacia arriba de la art 5
 delay(t);
 duty5=duty5+pasos;
}
```

```
if(mensaje=='d' and (pos0_5<duty5-10 and pos120_5>=duty5)){
 MiServo.Servo(3,duty5);      // Movimiento hacia abajo de la art 5
 delay(t);
 duty5=duty5-pasos;
}

if(mensaje=='E' and (pos0_6<=duty6 and pos120_6>=duty6+10)){
 duty6=duty6+pasos;           // Movimiento horario de la art 6
 MiServo.Servo(8,duty6);
 delay(t);
}

if(mensaje=='e' and (pos0_6<=duty6-10 and pos120_6>=duty6)){
 duty6=duty6-pasos;           // Movimiento antihorario de la art 6
 MiServo.Servo(8,duty6);
 delay(t);
}

if(mensaje=='F' and (duty7<pos120_7)){
 duty7=duty7+pasos_pinza;   // Abrir pinza
 MiServo.Servo(9,duty7);
 delay(t);
}

aux=1;

}
else if(duty7>pos0_7 and aux==0){
 duty7=duty7-pasos_pinza;    // Cerrar pinza
```

```
 MiServo.Servo(9,duty7);
 delay(t);
}
aux=0;
}
```

Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ÁREA DE TECNOLOGÍA ELECTRÓNICA

BRAZO ROBÓTICO CONTROLADO CON GESTOS MANUALES

D. JAVIER MARTÍNEZ BECERRA
TUTOR: D. RAMÓN RUBIO GARCÍA

FECHA: Junio 2023

Presupuesto

ÍNDICE

1. Presupuesto de ingeniería

1.1. COSTES SOFTWARE

Para este proyecto se han utilizado principalmente dos programas. Para el desarrollo del software del proyecto se ha utilizado el Arduino IDE el cual es gratis. Para el diseño de las PCBs se ha utilizado easyeda, el cual es un programa online y también es gratis. Como programa extra para la realización de los esquemáticos se ha utilizado el programa Fritzing.

Software	Meses	€/mes	Coste (€)
Arduino IDE	4	0	0
Easyeda	4	0	0
Fritzing	4	0	0
Costes totales (€)			0

1.2. COSTES PERSONAL

En esta parte se incluyen todos los costes de personal derivados de las actividades de diseño del proyecto.

Actividad	Horas	€/h	Costes (€)
Recogida de información	30	12	360
Prueba de componentes hardware	70	15	1050
Pruebas software y programación	50	15	750
Diseño PCB	30	15	450
Montaje PCB	20	15	300
Memoria	50	12	600
Costes totales (€)			3510

1.3. COSTES EQUIPO

Costes derivados del equipo y las herramientas necesarias para el diseño y desarrollo del proyecto.

Herramienta	Coste (€)
PC gama media	1000
Multímetro	40
Fuente de alimentación	60
Costes totales (€)	1100

1.4. COSTES TOTALES INGENIERÍA

Sección	Coste (€)
Software	0
Personal	3510
Equipo	1100
Costes indirectos (11%)	507
Beneficio industrial (8%)	369
IVA (21%)	968
Costes totales (€)	6454

2. Costes de ejecución

En este apartado se muestran los costes derivados de los materiales de ejecución de dicho proyecto.

Componente	Cantidad	€/unidad	Precio (€)
Arduino Uno	2	30	60
MG996R	6	6	36
PCA9685	1	13	13
NEMA 17	1	16	16
DRV8825	1	13	13
Batería 4.8V 2400mAh	2	16	32
Batería 12V 2500mAh	1	28	28
HC-05	2	10	20
MPU6050	2	6	12
Flex sensor	3	10	30
PCB	2	5	10
Jumpers	45	0.1	4.5
LED	1	0.15	0.15
Resistencia 10k	4	0.05	0.2
Interruptor	1	0.15	0.15
Costes totales (€)			275

Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ÁREA DE TECNOLOGÍA ELECTRÓNICA

BRAZO ROBÓTICO CONTROLADO CON GESTOS MANUALES

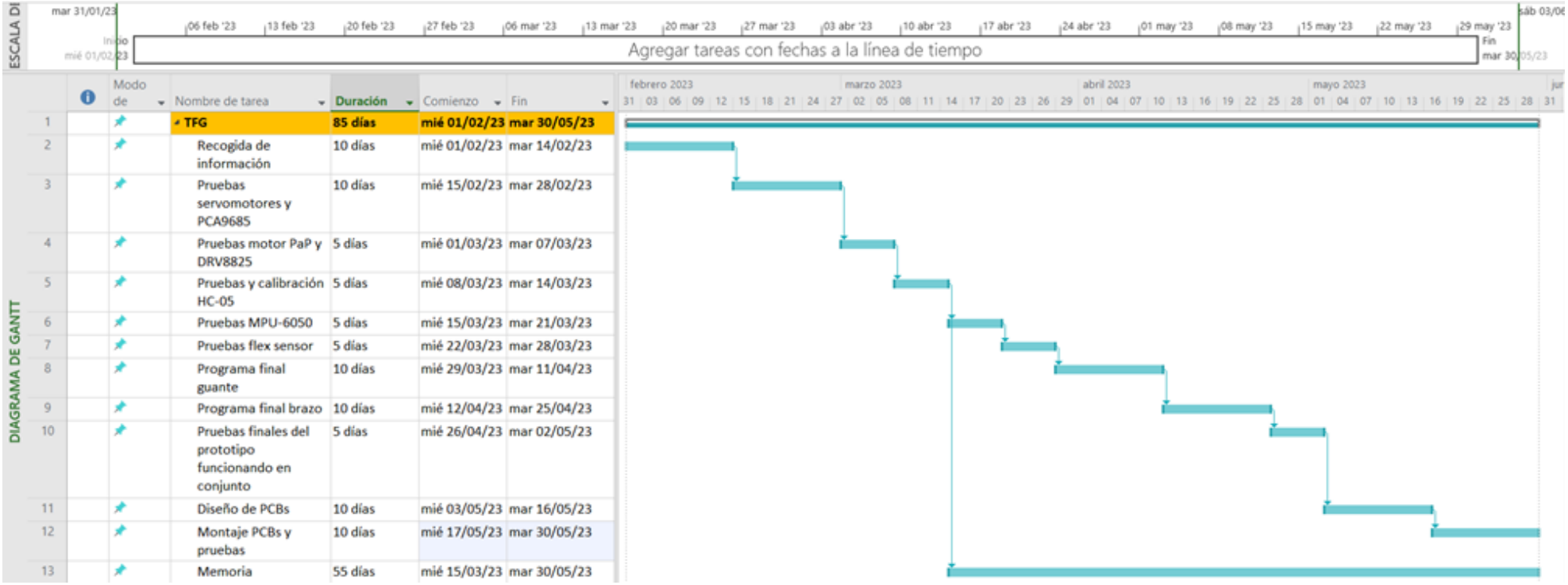
D. JAVIER MARTÍNEZ BECERRA
TUTOR: D. RAMÓN RUBIO GARCÍA

FECHA: Junio 2023

Anexos

ÍNDICE

1. Planificación temporal



2. Códigos

2.1. CÓDIGO GUANTE

```
//////////////////////////////////////////////////////////
// PROGRAMA GUANTE ROBÓTICO PARA CONTROL DE BRAZO ROBÓTICO
//
//            Javier Martínez Becerra                   //
//////////////////////////////////////////////////////////

// LIBRERÍAS
#include "SoftwareSerial.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

#define MPU6050_ADDRESS_1 0x68  //Dirección I2C del primer MPU
#define MPU6050_ADDRESS_2 0x69  //Dirección I2C del segundo MPU

SoftwareSerial BTSerial(10,11); //TX / RX auxiliares para no utilizar el puerto del Arduino

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 mpu0(MPU6050_ADDRESS_1);
MPU6050 mpu1(MPU6050_ADDRESS_2);

// Valores de los dos MPU en los ejes x,y,z
int ax0, ay0, az0;
int gx0, gy0, gz0;
int ax1, ay1, az1;
int gx1, gy1, gz1;

int t=1;                  // Tiempo de respuesta
int offset=3500;      // Offset de los MPU
int offset_flex0=170; // Offset de los flex0
int offset_flex1=160; // Offset de los flex1
int offset_flex2=170; // Offset de los flex2

int interruptor=8;  // Pin interruptor de ON/OFF del guante
int led=7;            // Pin led ON/OFF del guante

int flexPin0 = A0;    // Pin analógico del primer flex sensor
int flexValue0 = 0;   // Valor leído del primer flex sensor
int flexPin1 = A1;    // Pin analógico del segundo flex sensor
int flexValue1 = 0;   // Valor leído del segundo flex sensor
int flexPin2 = A2;    // Pin analógico del tercer flex sensor
int flexValue2 = 0;   // Valor leído del tercer flex sensor
```

```
void setup() {
Serial.begin(9600);    // Baudios comunicación serie
BTSerial.begin(4800);          // Baudios comunicacion BT
Wire.begin();
mpu0.initialize();       // Inicializar el primer MPU-6050
mpu1.initialize();       // Inicializar el segundo MPU-6050
pinMode(interruptor,INPUT);// Interruptor entrada
pinMode(led,OUTPUT);          // Led salida
delay(1000);
}

void loop() {
  //Comprobacion apagado o encendido
  if(digitalRead(interruptor)==HIGH) {

  //Serial.println("ENCENDIDO");  // Guante ON
  digitalWrite(led, HIGH);

  // Lectura de valores del primer MPU
  mpu0.getMotion6(&ax0, &ay0, &az0, &gx0, &gy0, &gz0);

  delay(t);

  // Lectura de valores del segundo MPU
  mpu1.getMotion6(&ax1, &ay1, &az1, &gx1, &gy1, &gz1);

  delay(t);
  // Lectura del primer flex sensor
  flexValue0 = analogRead(flexPin0);

  delay(t);
  // Lectura del segundo flex sensor
  flexValue1 = analogRead(flexPin1);

  delay(t);
  // Lectura del tercer flex sensor
  flexValue2 = analogRead(flexPin2);
  delay(t);

  // Descomentar para comprobar valores medidos por los sensores


/*Serial.println("================================================
=======");

  Serial.println(flexValue0);
  Serial.println(flexValue1);
  Serial.println(flexValue2);
```

```
Serial.print("MPU1: ");
Serial.print("ax1=");
Serial.print(ax1);
Serial.print("ay1=");
Serial.print(ay1);
Serial.print("az1=");
Serial.print(az1);
Serial.print("gx1=");
Serial.print(gx1);
Serial.print("gy1=");
Serial.print(gy1);
Serial.print("gz1=");
Serial.println(gz1);

Serial.print("MPU0: ");
Serial.print("ax0=");
Serial.print(ax0);
Serial.print("ay0=");
Serial.print(ay0);
Serial.print("az0=");
Serial.print(az0);
Serial.print("gx0=");
Serial.print(gx0);
Serial.print("gy0=");
Serial.print(gy0);
Serial.print("gz0=");
Serial.println(gz0);

delay(1000);*/

// Comandos enviados por BT para el control del brazo robótico
if(flexValue1<offset_flex1){          // Flexión flex 1
  BTSerial.write('A');
  delay(t);
}

if(flexValue2<offset_flex2){          // Flexión flex 2
  BTSerial.write('a');
  delay(t);
}

if(ax0>offset){                  // Rotación adelante MPU0
  BTSerial.write('B');
  delay(t);
}

if(ax0<-offset){                 // Rotación atrás MPU0
  BTSerial.write('b');
  delay(t);
```

```
 }

 if(ax1>offset){              // Rotación adelante MPU1
  BTSerial.write('C');
  delay(t);
 }

 if(ax1<-offset){             // Rotación atrás MPU1
  BTSerial.write('c');
  delay(t);
 }

  if(ay0>offset){             // Rotación horaria MPU0
  BTSerial.write('D');
  delay(t);
 }

 if(ay0<-offset){             // Rotación antihoraria MPU0
  BTSerial.write('d');
  delay(t);
 }

 if(ay1>offset){              // Rotación horaria MPU1
  BTSerial.write('E');
  delay(t);
 }

 if(ay1<-offset){             // Rotación antihoraria MPU1
  BTSerial.write('e');
  delay(t);
 }

 if(flexValue0<offset_flex0){        // Flexión flex 0
  BTSerial.write('F');
  delay(t);
 }
 }

 else{
  //Serial.println("APAGADO");  // Guante OFF
  digitalWrite(led, LOW);
 }
}
```

2.2. CÓDIGO BRAZO

```
//////////////////////////////////////////////////////////
//  PROGRAMA PARA CONTROL DE BRAZO ROBÓTICO MEDIANTE GESTOS//
//          Javier Martínez Becerra                 //
//////////////////////////////////////////////////////////
// LIBRERÍAS
#include <SoftwareSerial.h>
#include <HCPCA9685.h>
#include <AccelStepper.h>

#define  I2CAdd 0x40  // Dirección I2C del driver de servos
#define stepPin 2     // Define los pines de control del driver 8825
#define dirPin 3

HCPCA9685 MiServo(I2CAdd);  // Variable de tipo HCPCA9685 para controlar servos

AccelStepper stepper(AccelStepper::DRIVER, stepPin, dirPin);  // Variable AccelStepper para controlar el motor  PaP

char mensaje; // Variable para guardar mensaje leído por BT del guante

SoftwareSerial BTSerial(10,11); //TX / RX auxiliares

int Fase0=1;  // Variable para activar la posición inicial del brazo
int t=10;     // Tiempo entre pasos
int t1=200;   // Tiempo inicial
int pasos=1;  // Pasos por comando
int pasos_pinza=10;  // Pasos por comando pinza
int aux=0;
```

```
// Posiciones máximas y mínimas de los servomotores de las articulaciones
int duty2=230;
int pos0_2=0;
int pos120_2=250;

int duty3=250-duty2;
int pos0_3=-20;
int pos120_3=250;

int duty4=100;
int pos0_4=0;
int pos120_4=120;

int duty5=350;
int pos0_5=140;
int pos120_5=420;

int duty6=200;
int pos0_6=25;
int pos120_6=400;

int duty7=0;
int pos0_7=0;
int pos120_7=400;



void setup()
{
```

```
 MiServo.Init(SERVO_MODE); //Inicializar librería HCPCA9685 en modo Servomotor
 MiServo.Sleep(false);     // Despertar el driver
 MiServo.SetPeriodFreq(50);  // Fijar frecuencia de trabajo
 stepper.setMaxSpeed(50); // Configura la velocidad máxima en pasos por segundo
 stepper.setAcceleration(50); // Configura la aceleración en pasos por segundo por segundo
 Serial.begin(9600);         // Baudios comunicación serie
 BTSerial.begin(4800);       // Baudios comunciación BT
}

void loop()
{
// Fijar la posición inicial del brazo
if(Fase0==1){
 MiServo.Servo(0,duty2+22);
 MiServo.Servo(1,250-duty2);
 delay(t1);

 MiServo.Servo(2,duty4);
 delay(t1);

 MiServo.Servo(3,duty5);
 delay(t1);

 MiServo.Servo(8,duty6);
 delay(t1);

 MiServo.Servo(9,duty7);
 delay(t1);
```

```
 Fase0=0;
}

// Comprobar que hay mensaje enviado por BT
if (BTSerial.available()){
mensaje=BTSerial.read();
Serial.println(mensaje);

// Comandos para mover el brazo (servos y PaP) en función de los datos recibidos del guante
if(mensaje=='A'){
  stepper.move(10);       // Movimiento horario de la art 1
  stepper.run();
  delay(t);
}

if(mensaje=='a'){
  stepper.move(-10);   // Movimiento antihorario de la art 1
  stepper.run();
  delay(t);
}

if(mensaje=='B' and (pos0_2<duty2 and pos120_2>=duty2+10)){
 duty2=duty2+pasos;
 MiServo.Servo(0,duty2+22);          // Movimiento hacia arriba de la art 2
 MiServo.Servo(1,250-duty2);
 delay(t);
}
```

```
if(mensaje=='b' and (pos0_2<duty2-10 and pos120_2>=duty2)){
 duty2=duty2-pasos;
 MiServo.Servo(0,duty2+22);          // Movimiento hacia abajo de la art 2
 MiServo.Servo(1,250-duty2);
 delay(t);
}

if(mensaje=='C' and (pos0_4<duty4 and pos120_4>=duty4+10)){
 duty4=duty4+pasos;
 MiServo.Servo(2,duty4);      // Movimiento hacia arriba de la art 4
 delay(t);
}

if(mensaje=='c' and (pos0_4<duty4-10 and pos120_4>=duty4)){
 duty4=duty4-pasos;
 MiServo.Servo(2,duty4);      // Movimiento hacia abajo de la art 4
 delay(t);
}

if(mensaje=='D' and (pos0_5<duty5 and pos120_5>=duty5+10)){
 MiServo.Servo(3,duty5);      // Movimiento hacia arriba de la art 5
 delay(t);
 duty5=duty5+pasos;
}

if(mensaje=='d' and (pos0_5<duty5-10 and pos120_5>=duty5)){
 MiServo.Servo(3,duty5);      // Movimiento hacia abajo de la art 5
 delay(t);
```

```
 duty5=duty5-pasos;
}

if(mensaje=='E' and (pos0_6<=duty6 and pos120_6>=duty6+10)){
 duty6=duty6+pasos;            // Movimiento horario de la art 6
 MiServo.Servo(8,duty6);
 delay(t);
}

if(mensaje=='e' and (pos0_6<=duty6-10 and pos120_6>=duty6)){
 duty6=duty6-pasos;            // Movimiento antihorario de la art 6
 MiServo.Servo(8,duty6);
 delay(t);
}

if(mensaje=='F' and (duty7<pos120_7)){
 duty7=duty7+pasos_pinza;   // Abrir pinza
 MiServo.Servo(9,duty7);
 delay(t);
}
aux=1;
}
else if(duty7>pos0_7 and aux==0){
 duty7=duty7-pasos_pinza;    // Cerrar pinza
 MiServo.Servo(9,duty7);
 delay(t);
}
aux=0;
}
```