UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

DISEÑO E IMPLEMENTACIÓN DE UN GUANTE SENSOR DE MOVIMIENTO Y MANO ROBÓTICA ANTROPOMÓRFICA

MEMORIA PRESENTADA POR:

Andrés Peiró Abreu

Máster Universitario en Ingeniería Mecatrónica

DIRECTOR:

Vicente Casanova Calvo

Diciembre de 2020

RESUMEN

El presente documento trata acerca del estudio, simulación e implementación de un dispositivo capturador de movimientos de una mano humana y una mano robótica controlada a distancia, cuya utilidad puede ser muy numerosa en entornos tan diversos como manipulación de materiales en entornos peligrosos, control de vehículos no tripulados o la sustitución de los mandos analógicos de maquinas o videoconsolas, entre otros.

Este proyecto engloba casi todas las competencias desarrolladas durante el máster en mecatrónica, ya que utiliza un microcontrolador electrónico y sensores analógicos para poder capturar los movimientos, programas de simulación y resolución matemática como Matlab, el diseño por ordenador de elementos en tres dimensiones, la comunicación entre dispositivos mediante bluetooth y la utilización de las señales capturadas para mover elementos mecánicos.

En el apartado Resultados se mostrará el prototipo final del guante con sus gráficas donde se podrá consultar cómo es capaz de detectar el movimiento de los dedos de la mano humana, así como las imágenes de la mano robótica impresa en 3D y cómo es capaz de moverse imitando fielmente la mano del usuario.

ABSTRACT

This document is about the development, simulation and implementation of a device capable of detect movements on a human hand, and control a wireless robotic hand, whose utility can be as numerous as manipulate objects in dangerous environments, unmanned vehicles control, or the substitution of a classic machine or videogames controls, among other things.

This project embraces almost all the competences developed during the master's degree in mechatronics, as it uses an electronic microcontroller, analog sensors to capture the motion of the fingers, complex simulation programs as Matlab, Computer Assisted Design for the 3D parts modeling, wireless communication using bluetooth, and how to use this signals for moving mechanical parts.

In the results chapter, there is exhibited the final prototypes of the motion detector with their graphics where is illustrated how it is capable of detecting the movements of the human hand, and the pictures of the 3D printed hand and how it moves imitating the user's hand.

Índice de contenido

Índice de ilustraciones

1. Introducción

El termino mecatrónica hace referencia a la unión de la electrónica, la mecánica, el control electrónico y la informática, para realizar procesos de control y automatización. Y lo que implica inmediatamente la unión de todos estos conceptos son los robots.
A lo largo de los años uno de los objetivos de la mecatrónica siempre ha sido el emular (y en la medida de lo posible mejorar) la forma de funcionar del cuerpo humano y de la naturaleza que le rodea. Asimismo, la tecnología se ha desarrollado con fines de facilitar la vida a las personas que la utilizan, dotándoles de capacidades sobrehumanas o haciendo más cómodas las tareas arduas o peligrosas.
Actualmente existen infinidad de robots antropomorfos con mayor o menor grado de autonomía, inteligencia y libertad de movimientos. Como ejemplo podríamos citar el robot bípedo ASIMO [1] de procedencia japonesa, uno de los primeros robots humanoides que era capaz de caminar sobre dos piernas, o el robot Atlas [2] de la empresa norteamericana Boston Dynamics, que esta desarrollado para tareas de búsqueda y rescate, entre otros muchos ejemplos.

Ilustración 1: Robot humanoide ASIMO. [3]

La idea que da origen a este proyecto es, basándose en la naturaleza del movimiento de una mano humana, ser capaz de capturar los movimientos de ésta y poder replicarlos y utilizarlos para realizar un trabajo especifico en el que por cualquier motivo este desaconsejado realizarlo directamente por un humano.

Son muchas las razones por las que no se desea exponer a una persona a realizar un trabajo, como puede ser la peligrosidad (entornos explosivos o corrosivos), las limitaciones físicas del ser humano (movimiento de grandes masas o acceso a lugares remotos) la repetitividad de una tarea, o la posibilidad de realizar una tarea de forma remota. Como ejemplo de ello, varios países están trabajando en robots para la Estación Espacial Internacional, con el objetivo de realizar experimentos y misiones de mantenimiento de las naves espaciales que requieran salir al espacio exterior, evitando así el riesgo que ello conllevaría para un ser humano [4].

Este dispositivo capturador de movimientos podría ser una forma útil de operar estos robots de manera precisa y totalmente segura desde el interior de la nave, o incluso desde la propia tierra. También podría usarse este dispositivo para controlar otros aparatos que requieran complejos sistemas de control, facilitando y acortando el periodo de adiestramiento de las personas que deban operar esta maquinaria.

Por otro lado, acoplando un modelo de la mano a un brazo robot comercial, podría conseguirse un sistema que imite los movimientos de cualquier persona para realizar trabajos, tal y como se muestra en la imagen en la que se ilustra un proyecto similar de la universidad de Lausana para facilitar la vida a personas con discapacidades [5].

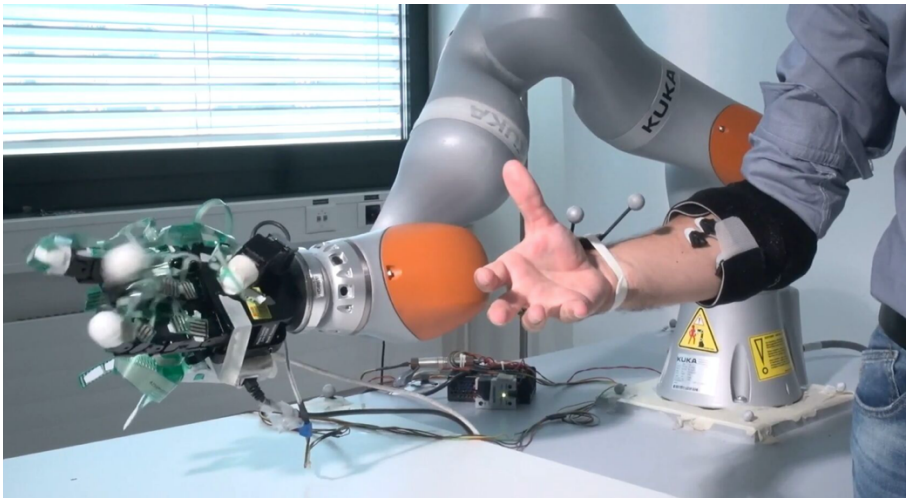


Ilustración 2: Proyecto de mano robótica, por la universidad de Lausana [6].

2.Objetivos y alcance del proyecto

El objetivo principal del proyecto es conseguir un instrumento capaz de capturar los movimientos de una mano humana, y aplicar estos datos a fines útiles como mover una mano robótica.

Para conseguir esto, el proyecto se dividirá en dos partes bien diferenciadas: el guante capturador de movimientos y el elemento móvil. Además, para completar el estudio, también será necesario realizar un modelo virtual por ordenador que nos permitan simular el comportamiento de la mano y adecuar las señales del capturador al fin requerido por el elemento móvil. Todo este trabajo estaría incompleto sin contemplar algún tipo de comunicación inalámbrica entre el capturador de movimientos y el elemento móvil. Esto se resuelve mediante una comunicación bluetooth entre ambos elementos.

Para realizar el guante capturador nos serviremos de un guante textil al que se acoplen sensores analógicos de flexión y un microcontrolador electrónico para tratar con estas señales y transmitirlas de forma inalámbrica.

Por otro lado, el modelo mecánico podría ser de diversa índole. Por cuestiones de extensión del proyecto, solo trataremos una de las posibles aplicaciones del guante, utilizando un modelo anatómico de una mano humana impresa en 3D a la que se añadirá otro microcontrolador y unos servomotores para recibir las señales y transformarlas en movimiento de los dedos mecánicos.

Por la complejidad del diseño de las piezas que componen una mano humana realista, y por existir multitud de ejemplos disponibles de forma gratuita en internet, así como por entender que la finalidad de este proyecto está más relacionada con la electrónica y el control que con el diseño en 3D, se ha optado por buscar un modelo de CAD con licencia abierta y realizar las simulaciones con ello. No obstante, se han realizado las adaptaciones y los cambios necesarios para poder utilizar estas piezas en las simulaciones y para realizar un modelo físico de mano móvil impresa en impresora 3D.

3.Estudio anatómico

Para poder imitar el movimiento de una mano, el primer paso es comprender su funcionamiento:

La mano humana es un complejo mecanismo compuesto por varios elementos, ensamblados entre ellos para realizar trabajos muy versátiles. Así pues, la mano humana es capaz de realizar multitud de movimientos complejos y de efectuar un mismo trabajo de varias formas distintas.

Internamente, una mano se compone de huesos unidos por cartílagos, que actúan como una unión elástica. Mecánicamente podríamos aproximar la mayoría de estas uniones como una articulación de tipo rotación, con un grado de libertad.

Cada hueso dispone de varios músculos, que son los encargados de mover los huesos según las necesidades. Estos músculos se unen entre ellos y a otros huesos utilizando tendones y fascias, de forma que resultan en un conjunto de tejidos blandos móviles unido a varios puntos que trabajan simultáneamente para aportar movimiento y estabilidad a cada articulación.

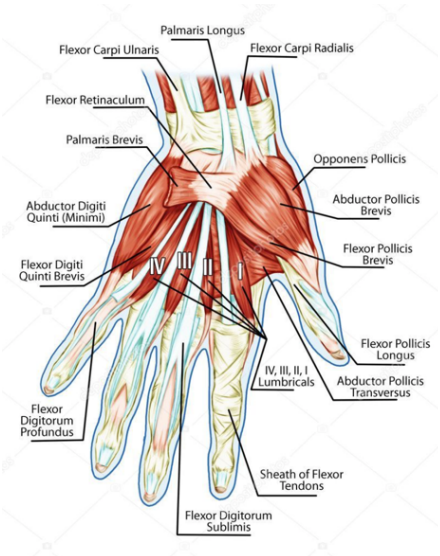


Ilustración 3: Estructura interna de la mano (Tejidos blandos). [7]

Para no exceder la extensión del trabajo, nos centraremos únicamente en el esqueleto que conforma la mano, dejando de lado otras partes y tejidos blandos como los músculos, tendones, ligamentos, etc.

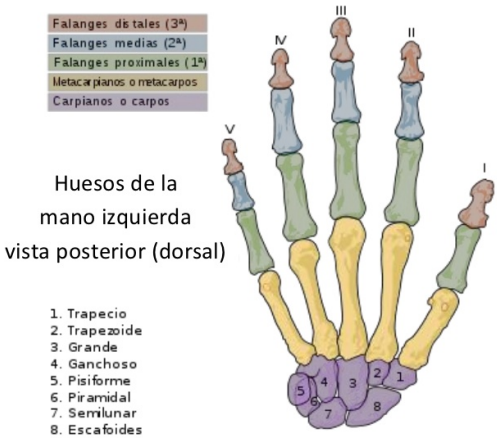


Ilustración 4: Huesos que componen la mano humana. [8]

En lo que al esqueleto respecta, la mano es un mecanismo compuesto por múltiples barras móviles unidas a una barra fija, que sería la muñeca. Las falanges de los dedos están unidas entre si mediante articulaciones, o pares cinemáticos de tipo R o revolución. Si miramos con más detenimiento las articulaciones de los metacarpos, podríamos observar que algunas tienen más de un grado de libertad, o que algunas de estas barras tienen más movimientos que otras, pero por simplicidad del prototipo nos vemos obligados a reducir los movimientos posibles a un único grado de libertad por cada articulación.

Analizando el movimiento natural de las manos al trabajar, podemos observar que las falanges proximales y medias se mueven independientemente de los metacarpos, no siendo así con las falanges distales. Por ello, el prototipo que realizaremos será un modelo en el que cada dedo podrá mover independientemente las falanges anteriormente descritas, mientras que no se tendrá en cuenta el movimiento de las falanges distales ni de los metacarpos. Esto nos deja con un modelo de cinco dedos en los que cada dedo dispone de dos grados de libertad, correspondientes a las dos primeras articulaciones de los dedos de la mano real.

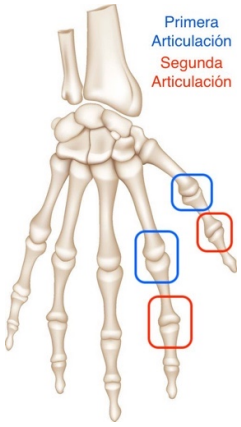


Ilustración 5: Dos grados de libertad en cada dedo.

En cuanto a la cantidad de dedos a simular, de nuevo se ha optado por reducir la carga de trabajo, utilizando únicamente dos dedos de la mano. Realizaremos esta simplificación ya que se entiende que los dedos índice, corazón, anular y meñique realizan un trabajo similar entre ellos, por lo que detectar el movimiento y simularlo en uno de ellos es extensible al resto. Por otro lado, el pulgar es un dedo distinto al resto, ya que es un dedo oponible que otorga una funcionalidad a la mano exclusiva de los humanos y primates.

Para poder realizar el movimiento de pinza tan importante y usual en la mano humana se opta por la simulación de los dedos índice y pulgar, que se considera suficiente para emular los movimientos básicos de una mano humana.

4.Hardware

El hardware que compone el proyecto está diferenciado en dos partes separadas: el guante capturador de movimiento y la mano robótica.

4.1 Guante capturador

El capturador de movimiento es el elemento que, ayudado de la electrónica, será capaz de detectar y transmitir los movimientos realizados por la mano humana que sirva de modelo. Se compone por los siguientes elementos:

4.1.1 Arduino Nano

La marca Arduino comprende una familia de microcontroladores de sistema abierto ampliamente conocida y utilizada en multitud de proyectos de diferente grado de dificultad técnica, desde pequeños proyectos escolares hasta sistemas de tecnología punta. Arduino se basa en la utilización de un microcontrolador de la familia Atmel, implementado en una placa impresa con todos los componentes necesarios para su funcionamiento, incluida fuente de alimentación, interfaz de comunicación USB, memoria RAM, conversor Analógico/Digital y puertos de conexión rápida para acoplar fácilmente multitud de hardware externo. Esta configuración le da una flexibilidad y una facilidad de uso que, unido a su potencia de procesamiento, lo ha hecho famoso en todo el mundo.

El modelo de Arduino utilizado en este proyecto es el Arduino Nano, el cual está compuesto por un microcontrolador de 8 bits ATmega168, 14 entradas/salidas digitales (de las cuales 6 pueden ser utilizadas como salida PWM), 8 entradas analógicas, interfaz USB, oscilador para el microcontrolador y botón de reset. El modelo de Arduino ha sido elegido por la cantidad de entradas analógicas disponibles, que hace la conexión de todos los sensores bastante simple, así como por su reducido tamaño y peso, con unas dimensiones de 45 x 18 milímetros y un peso de 7 gramos [9]. En conjunto, este modelo de Arduino es muy similar al clásico Arduino UNO, pero con un tamaño mucho más reducido.

Además, la frecuencia de trabajo del micro (16MHz) es adecuada para las necesidades del capturador de movimiento, ya que no se espera movimientos muy rápidos en una mano humana media.

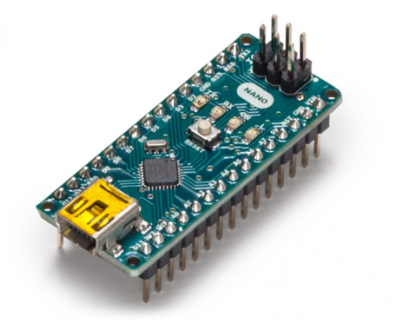


Ilustración 6: Arduino Nano, modelo utilizado en el guante capturador [10].

4.1.2 Sensor Flex

El sensor de flexión, o flex sensor, es el elemento utilizado para detectar el movimiento de los dedos, así como el ángulo de inclinación de cada articulación medida. Concretamente el modelo utilizado es el flex sensor 2.2" de Spectra Symbol. Este tipo de elementos cuentan con las ventajas de ser simples y robustos, de larga vida útil, poco voluminosos, se doblan con el elemento al que se han acoplado y son capaces de medir el ángulo de desplazamiento al que son sometidos. Se ha elegido este tipo de sensores por las características anteriormente mencionadas, así como porque son relativamente baratos y fáciles de conseguir.

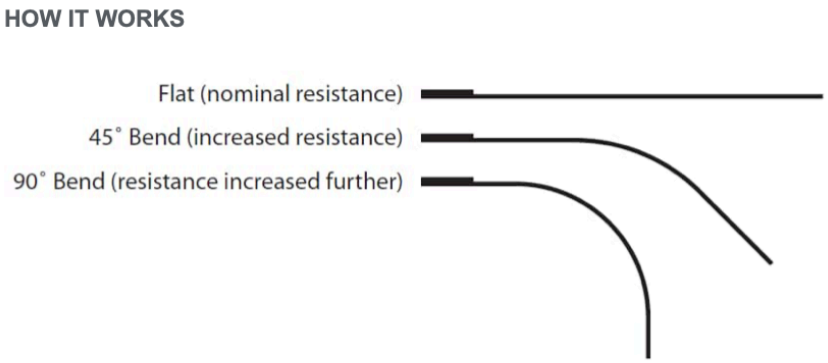


Ilustración 7: Comportamiento del flex sensor [11].

El comportamiento del flex sensor es similar a un potenciómetro, ya que varía su resistencia conforme se dobla el sensor. Esta respuesta es debida a que está compuesto por un material conductor con pequeñas partículas de otro material más conductor, de forma que ofrece una resistencia definida por el fabricante en condiciones normales. Al someter el material a una fuerza que lo deforma, las micropartículas internas se separan o acercan, variando su resistencia total. Este material se deposita sobre una película plástica flexible para dar resistencia al conjunto, y se acoplan dos conectores estandarizados a uno de los extremos de la lámina para poder conectar de manera rápida el sensor.

Existen varios tipos y modelos de flex sensor, concretamente el utilizado tiene un rango de medidas de entre 15 KΩ en reposo (recto) y 45 KΩ doblado a su ángulo máximo.

La forma de conectar estos sensores al microcontrolador puede ser de distintas formas. En este caso se ha optado por realizar un circuito divisor de tensión, de forma que el hardware necesario sea mínimo, únicamente una resistencia de 47 KΩ y los cables necesarios para su conexión.

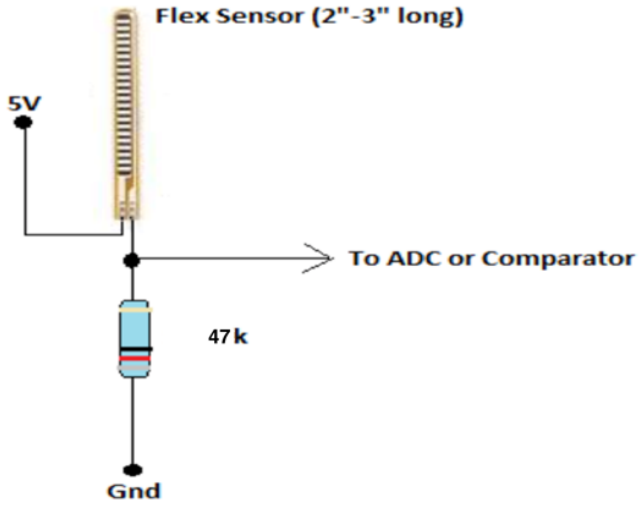


Ilustración 8: Circuito adaptador de señal [12]

Esta opción no es la mejor en términos de estabilidad de las lecturas, pero se ha mantenido por ser la más sencilla y la que me nos hardware adicional necesita. Se contemplarán las otras formas de conexión en el apartado mejoras propuestas.

En cuanto a su señal de salida, se realizaron varias pruebas durante la fase inicial de desarrollo del proyecto, para lo cual se utilizó un molde impreso en 3D con el que se colocaba la lámina en determinados ángulos, para poder realizar una medida estable y repetible a 0, 45, 90 y 180 grados.

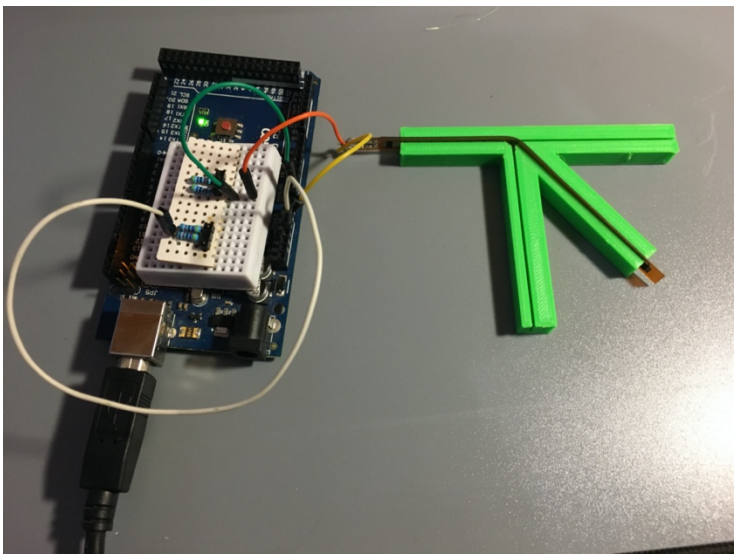


Ilustración 9: Ensayo del Flex Sensor.

Tras guardar las respectivas lecturas del conversor analógico digital disponible en la placa Arduino y transformarlas a tensión se obtuvo la siguiente gráfica:

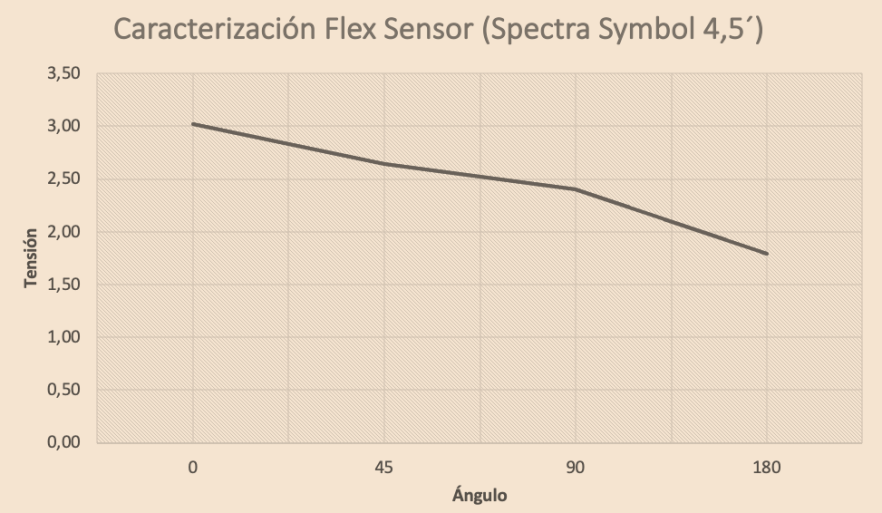


Ilustración 10: Gráfica Tensión/Ángulo del Flex Sensor.

En la gráfica se puede observar cómo la señal de salida no es perfectamente lineal, aunque se aproxima bastante a una recta, por lo que no será necesaria ninguna linealización matemática posterior.

4.1.3 Módulo Bluetooth

Bluetooth es una especificación industrial para redes inalámbricas de área personal (WPAN) creado por Bluetooth Special Interest Group, Inc. que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz [13].

Los objetivos de esta tecnología son eliminar los cables en las comunicaciones entre dispositivos móviles y crear pequeñas redes inalámbricas para sincronizar datos entre estos dispositivos.

Este tipo de comunicación ha aumentado mucho su utilización con el auge de los smartphones y los microcontroladores de precio reducido, por ello en el mercado existen varios módulos bluetooth diseñados específicamente para trabajar con Arduino, ejemplo de ellos son los conocidos módulos HC-05 y HC-06 que trabajan como esclavo y maestro respectivamente.

Por otro lado, existe otro módulo denominado HM-10, que puede trabajar como maestro y esclavo, y que cuenta con la tecnología bluetooth 4.0, lo cual supone una ventaja con respecto a sus antecesores en cuanto a velocidad, alcance y numero de dispositivos a conectar, así como ser de bajo consumo.

Este módulo cuenta con un precio reducido, buena fiabilidad y las facilidades de programación y configuración de los demás dispositivos de este tipo, por ello es el módulo seleccionado para este proyecto.

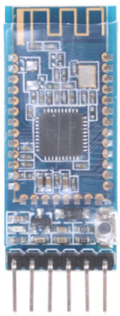


Ilustración 11: Modulo bluetooth HM-10. [14]

4.1.4 Guante

Todos los elementos anteriores necesitan un elemento que los una y permita el movimiento solidario a la mano, para esa labor se utiliza un guante textil de trabajo. Está compuesto por un tejido de nylon bañado en látex para aportar más resistencia y protección a las manos del usuario.

Sobre este guante se han cosido cuatro sensores de flexión, uno por cada articulación a registrar.

La posición de cada sensor es muy importante para poder detectar correctamente cada articulación, así como para evitar daños en los sensores si trabajan de forma incorrecta.

Ilustración 12: Guante de trabajo. [15]

4.2 Mano 3D

Como propuesta de utilización del capturador de movimientos se propone un modelo a escala de una mano humana controlada por un Arduino de forma inalámbrica. En este caso, y para abaratar costes, se ha realizado una simplificación del modelo de mano humana con un dedo pulgar e índice útiles, de forma que se pueda realizar el movimiento de pinza para recoger objetos. Este modelo se ha impreso en una impresora 3D utilizando PLA como material principal. Asimismo, se ha utilizado un servomotor para mover cada dedo y dotarlo de movimiento y control de posición todo integrado en un único elemento. Esta compuesta por los siguientes componentes:

4.2.1 Arduino UNO

Al igual que el guante capturador de movimientos, se utiliza un Arduino para gestionar las señales recibidas del mismo, pero en este caso se trata de un Arduino modelo UNO.

Ilustración 13: Arduino UNO. [16]

El Arduino UNO es uno de los modelos más usados por su versatilidad y bajo coste. Dispone de un microcontrolador ATmega 328p, con 14 entradas/salidas digitales, 6 de las cuales pueden ser utilizadas como salida PWM, 6 entradas analógicas y un reloj de 16 MHz. Sus medidas son 68,6 x 53,4 milímetros y pesa 25 gramos.

El motivo de la elección de este modelo es porque dispone de un circuito regulador de tensión de alimentación, lo que le permite ser alimentado por cualquier fuente de tensión de entre 6 y 20 voltios. Esto, unido a su bajo coste lo hace ideal para la mano robótica.

4.2.2 Batería

Para alimentar todos los elementos de la mano robótica se opta por utilizar una batería de litio, de forma que no necesite ningún cable para su funcionamiento.

La batería de litio es un elemento capaz de almacenar energía en su interior gracias a las sales de litio que contiene. Estas sales producen una reacción química que crea una diferencia de potencial entre el ánodo y el cátodo formando una corriente de electrones.

Para este proyecto se ha utilizado una batería cedida por el departamento de Ingeniería de Sistemas y Automática, que ya dispone del conector adecuado para alimentar el Arduino UNO y tiene capacidad suficiente para operar la mano durante horas. Sus características son: 7,4 V y 6000 mAh.

Ilustración 14: Batería para la mano robótica.

4.2.3 Servomotor

Los servomotores son elementos compuestos por un motor de corriente continua, un potenciómetro y un circuito interno. Tienen la capacidad de adoptar una posición estable ante variaciones externas dentro de su rango de posibilidades. La posición que adopta el eje del motor se determina mediante una señal de entrada modulada en ancho de pulso o PWM por sus siglas en inglés.

El modelo utilizado es el SG90, un servomotor de 9 gramos de peso, con un par de 1,8Kg/cm y reducido tamaño, elegido por ser un servomotor de pequeño tamaño y de bajo precio. Además, es uno de los servos mas versátiles y es usado en todo tipo de proyectos con Arduino.

Ilustración 15: Servo SG90. [17]

4.2.4 Modelo impreso

Son muchos los proyectos dedicados al modelado de partes humanas, con fines de realizar miembros protésicos, robots humanoides etc.

Durante la investigación necesaria para realizar este proyecto se encontró un proyecto llamado InMoov [18], que consiste en el diseño libre de un robot con forma humana capaz de imprimirse en

impresoras de tamaño estándar. En este proyecto, además de otras partes del cuerpo, existe una mano humanoide con capacidad de moverse gracias a servomotores, lo cual lo hace perfecto para utilizar con el guante capturador de movimientos. En su pagina web existen multitud de tutoriales detallados donde se explica cómo comenzar con un dedo móvil mediante Arduino [19], cómo montar la mano completa [20], y cómo ajustar su correcto funcionamiento [21], así como el montaje del resto de partes del robot.

Tal y como se avanzaba anteriormente, se ha tomado el modelo diseñado por ordenador de la mano y se ha adaptado a nuestros requerimientos, ya que el diseño original tiene muchas más piezas y elementos de los necesarios para este proyecto. El material utilizado es PLA, un tipo de termoplástico biodegradable proveniente del maíz [22], muy extendido como material para impresoras 3D por su baja temperatura de fusión y su resistencia mecánica a temperatura ambiente.

4.2.4.1 Montaje de las piezas

Tal y como se indicaba, en la web del diseñador de las piezas se pueden encontrar los pasos a seguir para ensamblar las piezas. Pero, al ser éste un modelo adaptado, no todos los pasos son iguales, por lo que se cree necesaria una pequeña descripción de los pasos seguidos para la realización de la mano robótica.

En primer lugar, se reúnen y clasifican todas las piezas impresas en 3D:

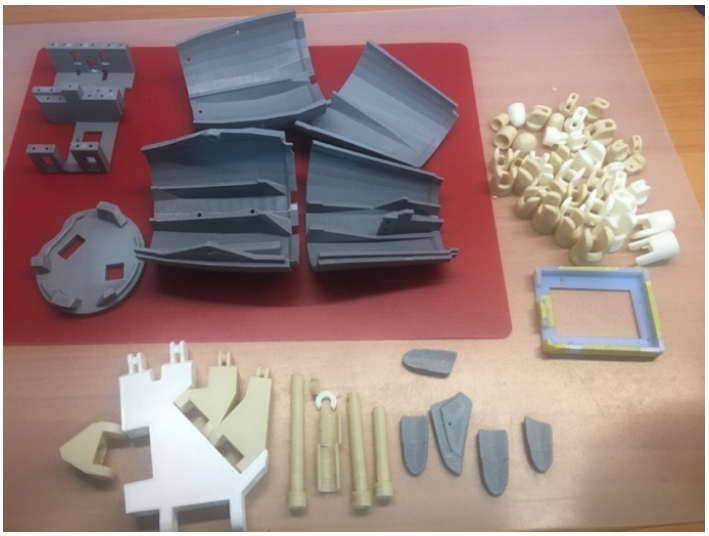


Ilustración 16: Despiece de la mano.

Se pegan las piezas del antebrazo, que están divididas por la mitad para poder imprimirse mas fácilmente (se utilizó cola de contacto por ser una unión mas elástica):

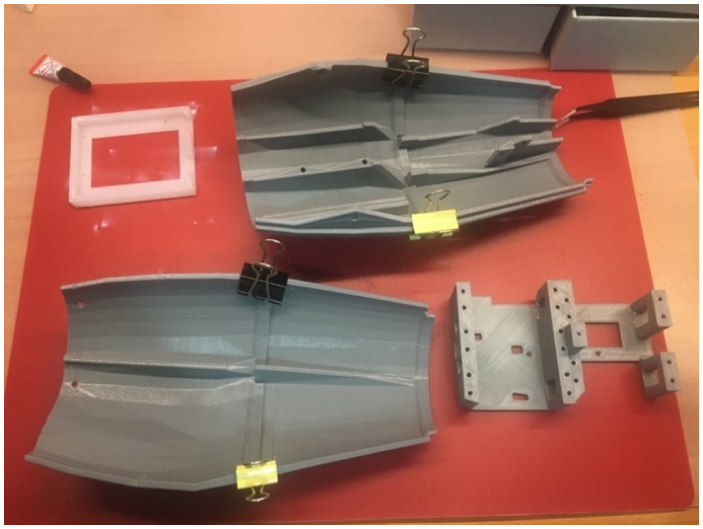


Ilustración 17: Pegado de las piezas del antebrazo.

Se pegan las piezas de los dedos, tal y como indica la web original utilizando pegamento con base de cianocrilato:

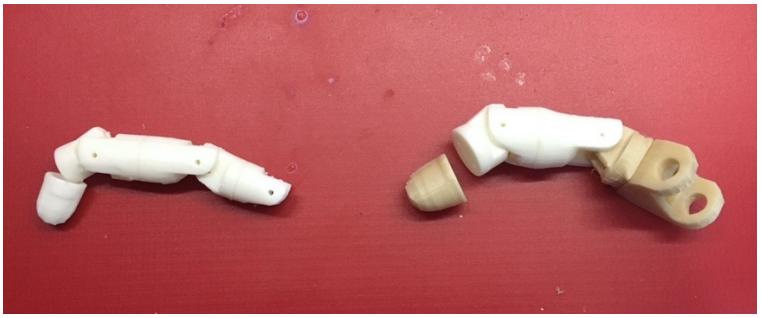


Ilustración 18: Pegado de las piezas de los dedos.

Con todas las piezas secas y preparadas para su ensamblaje, es muy importante lijar bien y eliminar cualquier rebaba en las uniones rotativas, para obtener un movimiento suave en la articulación y ofrecer la menor fricción posible. Para ello se recomienda papel de lija y un cúter afilado.

Una vez tenemos las piezas preparadas y estamos seguros de que las articulaciones giran suaves, procedemos a insertar el hilo de nylon que hará las veces de ligamentos:

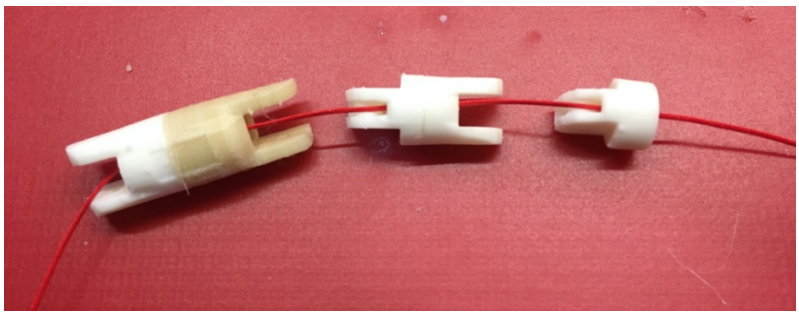


Ilustración 19: Inserción del cable de nylon.

Se inserta el hilo de pescar en dos tramos de unos 30 cm y se atan en la punta de cada dedo, tal y como se indica en la página web. Es importante que sea un hilo de nylon trenzado, ya que éste ofrece mínima elasticidad y una mejor resistencia a la abrasión que el hilo de nylon simple. El hilo escogido en este caso es un hilo hueco de 1,5 milímetros con una resistencia de 200 Kilogramos comprado en una tienda de pesca.

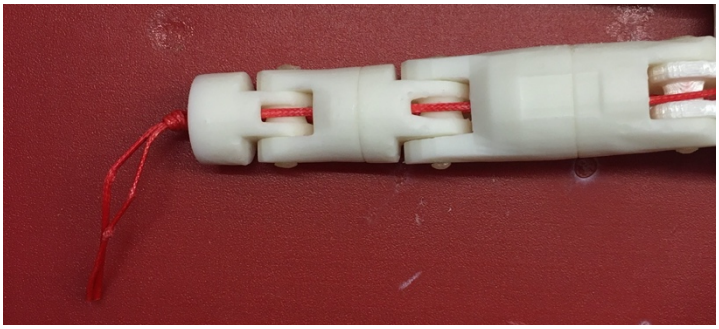


Ilustración 20: Fijación del cable de nylon.

Para unir los dedos, es necesario insertar un eje en los agujeros de que disponen las piezas. Originalmente se utiliza una barra roscada de acero, pero en este caso se prefirió utilizar un trozo de filamento para impresora 3D de 1,75 mm porque es una solución barata y sencilla de mecanizar y al ser un primer prototipo no se espera que ejerza una fuerza suficiente como para romper este filamento. Además, es un elemento que se encontraba disponible en el momento del ensamblaje y demostró su efectividad mas tarde.

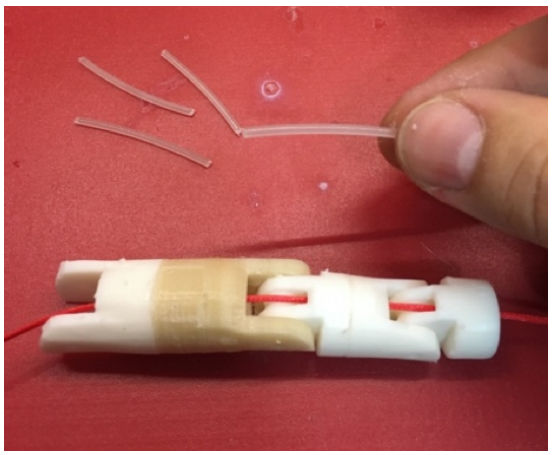


Ilustración 21: Montaje de los ejes de articulación.

Una de las ventajas de utilizar un filamento de PLA como eje es que es relativamente resistente y muy sencillo de mecanizar. Valiéndonos de un soldador de estaño y gracias a su baja temperatura de fusión, es muy sencillo fundir los extremos del filamento para que quede fijado a cada extremo de la articulación. Además, puesto que la propia mano es también de PLA, queda soldada a la misma pieza de forma definitiva.

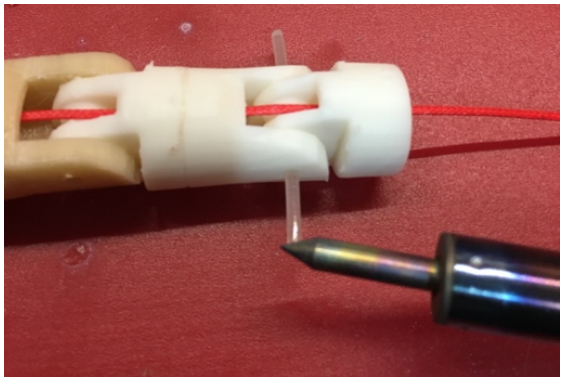


Ilustración 22: Fijación de los ejes por calor.

Con los dedos ensamblados, queda unirlos a la palma de la mano, utilizando el mismo método del filamento de 1,75 y los pasadores que hacen las veces de tornillo en las uniones mas grandes:

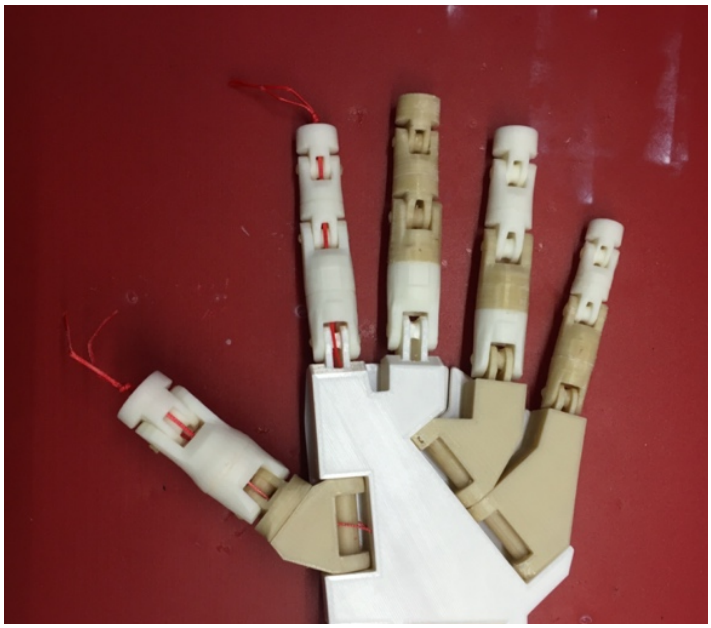


Ilustración 23: Palma de la mano.

Y el embellecedor del dorso de la mano, que también tiene una función estructural además de estética:

Ilustración 24: Dorso de la mano.

Nótese que las piezas finales de los dedos no se han colocado aun para permitir pequeñas modificaciones y ajustes posteriores, una vez colocados los servos. Estas piezas serán las últimas en colocarse ya que no cumplen una función estructural, únicamente estética.

Por último, se ensambla la mano con la pieza que la une al antebrazo, pasando los hilos por los agujeros correspondientes:

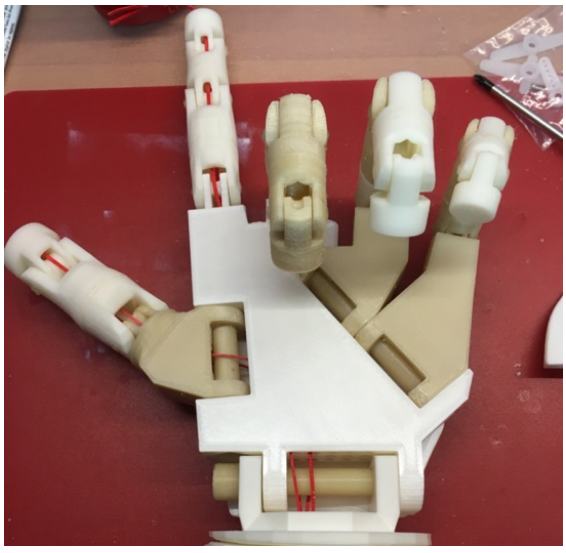


Ilustración 25: Unión de la muñeca.

La unión de la muñeca con la pieza de soporte se realiza, de nuevo, con un pasador que imita a un tornillo, proporcionado por el mismo diseñador. Nótese que solo se ha pasado el hilo de pesca por

los dedos índice y pulgar, ya que son los únicos que realizaran el movimiento controlado. Los demás dedos se han montado por estética, pero quedan libres.

Llegados a este paso, la mano esta terminada, pero queda el ensamblaje con las piezas del antebrazo y la electrónica interna.

El primer problema encontrado fue que la pieza que hace las veces de muñeca no correspondía con las del antebrazo:

Ilustración 26: Piezas del antebrazo.

Por algún error debido a las múltiples versiones que existen de este diseño, se imprimió una muñeca que no encaja con el antebrazo al ser demasiado grande. El problema se pudo solucionar cortando la pieza de la muñeca y pegándola a las piezas tanto del antebrazo como de la palma:

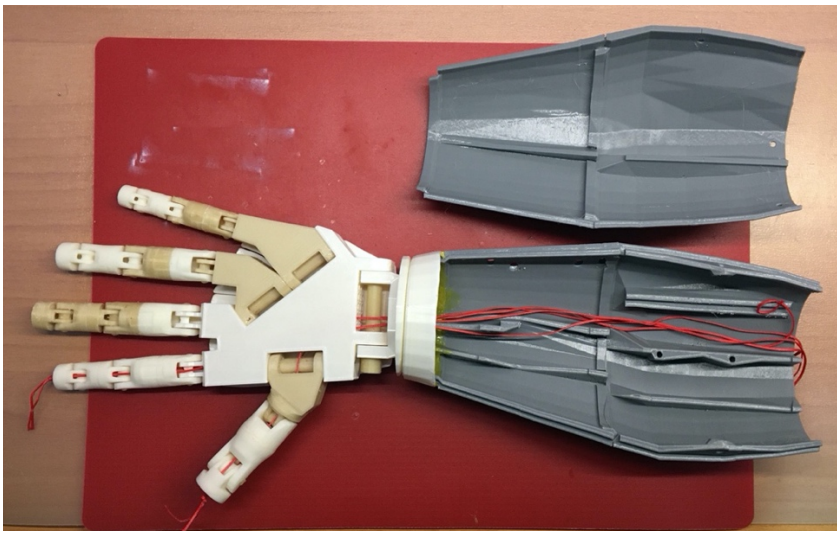


Ilustración 27: Estructura de la mano completa.

El siguiente paso es modificar el soporte de los servos y la electrónica, ya que nuestros servos son distintos de los originales y también es necesario hacer sitio a la batería:

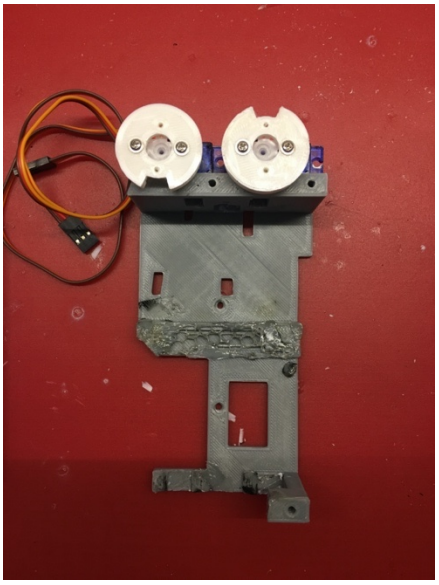


Ilustración 28: Bastidor modificado.

La modificación consiste en colocar los servos en la parte externa del bastidor y eliminar todos los salientes centrales, de forma que la batería se acople perfectamente sobre el nuevo bastidor.

A continuación, se pegan las piezas que sirven de soporte para el Arduino UNO:

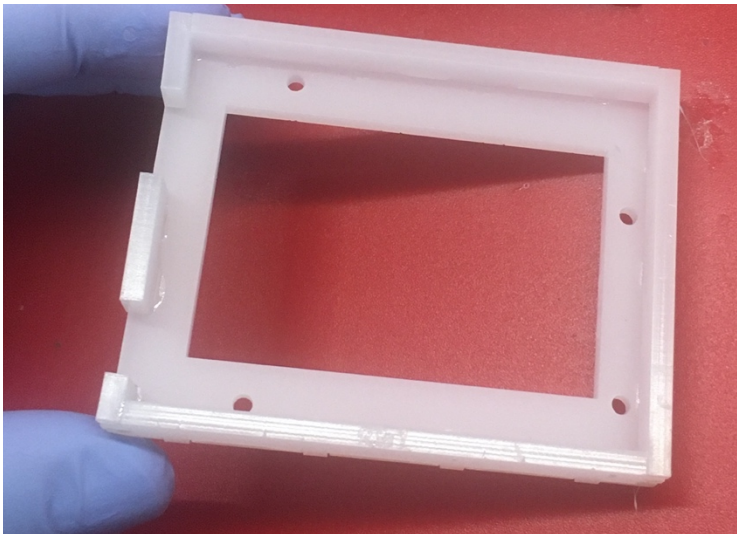


Ilustración 29: Soporte para el Arduino UNO

Estas piezas están realizadas en metacrilato de 3 mm mediante corte por láser, y pegadas con adhesivo cianocrilato.

Con el bastidor preparado y el soporte de Arduino pegado, queda todo preparado para montarlo:

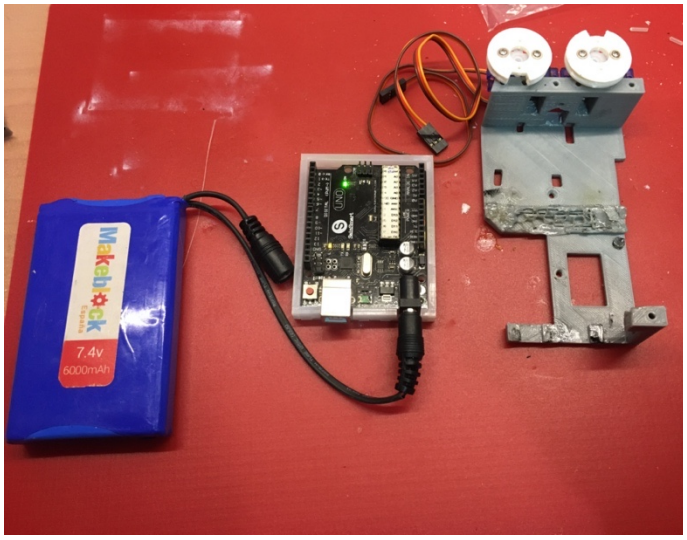


Ilustración 30: Piezas del bastidor.

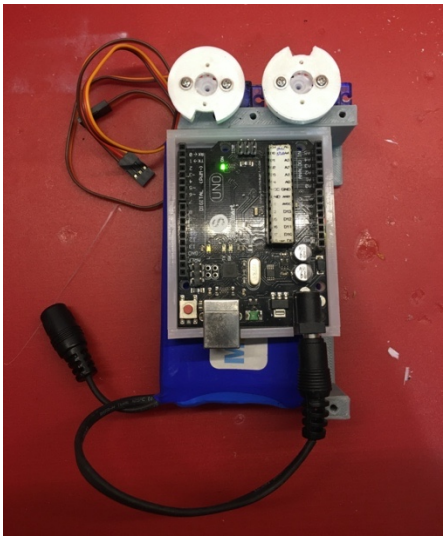


Ilustración 31: Bastidor preparado.

La forma de acoplar estas piezas es colocando la batería ajustada sobre el bastidor y el Arduino sobre ella. Todas estas piezas se aseguran con unas tiras de velcro, de forma que queden perfectamente fijas y puedan ser retiradas de forma fácil para mantenimiento o futuras mejoras en la mano.

Para finalizar con la electrónica, queda montar el bastidor sobre la pieza del antebrazo con dos tronillos y realizar las conexiones pertinentes tanto de los servos como del módulo bluetooth:

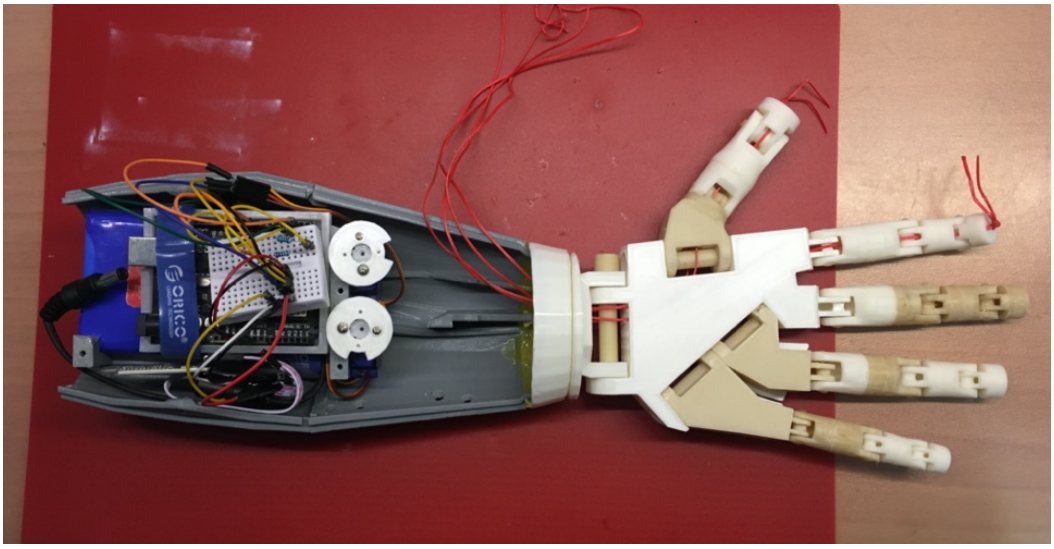


Ilustración 32: Montaje del bastidor.

El módulo bluetooth se ha dejado en un hueco lateral del bastidor y se asegura con al tira de velcro, ya que la no tener apenas masa no se espera que se mueva demasiado.

El paso final para terminar la mano es acoplar y tensar los hilos de nylon que hacen las funciones de tendones a las poleas de los servos. Para ello hay que realizar una pequeña mecanización de los agujeros de las poleas, ya que la impresión no ha conseguido el agujero definido del diámetro requerido. Estas poleas impresas se unen al eje original del servo con los tornillos proporcionados por el fabricante. Por suerte los agujeros de sujeción al eje del servo si acoplan bien así que no hay que mecanizarlos.

Se fijan los tendones haciéndolos pasar por los agujeros anteriormente descritos y después fijándolos a los tornillos de sujeción de la propia polea, procurando que queden tensos y las poleas centradas en el rango de movimientos del servo:

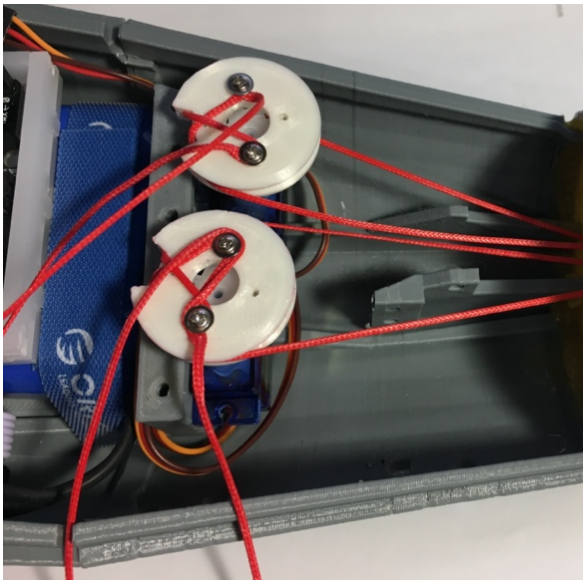


Ilustración 33: Detalle de los cables de nylon fijados a las poleas.

Una vez fijos y tensados los hilos de nylon, procederemos a pegar las puntas de los dedos de nuevo con cianocrilato.

Y por último se programa el Arduino para obtener una mano robótica funcional.

5.Software

En este apartado se explicará brevemente las funciones que realiza cada microcontrolador y los programas codificados para ello. Trataremos por separado la programación del microcontrolador, las comunicaciones de datos y la simulación por ordenador:

5.1 Software Arduino

El ID de Arduino es un programa de licencia gratuita ampliamente conocido que se utiliza para programar los microcontroladores de toda la familia Arduino. El lenguaje de programación es C++ y utiliza una gran cantidad de librerías para facilitar su uso y aprendizaje a cualquier persona iniciada o no en la electrónica digital.

En nuestro caso lo utilizaremos para programar los dos Arduino de forma que sean capaces de realizar dos tareas definidas y comunicarse entre ellos y con un ordenador.

5.1.1 Comunicaciones

Para realizar la simulación por ordenador se utilizará el protocolo de comunicación Serie, ya que el Arduino esta preparado para ello y es un sistema de comunicación efectivo y con suficiente velocidad para nuestros requerimientos. Es bastante usual y en internet abundan ejemplos de utilización de este tipo de comunicaciones con Arduino [23].

La comunicación serie es la transmisión de datos en forma de bits uno detrás de otro, utilizando para ello un único cable para la transmisión y otro para la recepción. Arduino utiliza para ello los pines Rx y Tx, o bien mediante el conector USB que tiene integrado en la placa.

En nuestro caso, al realizar la comunicación con el ordenador, tendremos que definir la transmisión de cuatro datos o caracteres en forma de byte que el ordenador recogerá de forma adecuada y transformará para realizar la simulación. Para ello nos valdremos del programa de simulación Matlab-Simulink, del que hablaremos más adelante, y que realiza las tareas de sincronización y gestión de datos de forma sencilla. En cuanto a la conexión entre Arduino y Matlab-Simulink, solo hay que tener en cuenta que se haya definido correctamente el puerto del ordenador al que se conecta al Arduino y que ambos elementos tengan la misma velocidad de transmisión.

Por otro lado, para realizar la comunicación entre el guante y la mano robótica, es un poco más complejo ya que se trata de dos microcontroladores. Una de las formas más comunes de realizar una comunicación de este tipo es la creación de una red Maestro-Esclavo, en el que un maestro inicia un requerimiento y el esclavo devuelve una respuesta utilizando un protocolo de comunicación preestablecido.

El guante capturador de movimientos será el Maestro, que generará una conexión mediante el módulo bluetooth y mandará información al esclavo, que será el Arduino colocado en la mano robótica.

Para realizar la comunicación bluetooth hay que configurar adecuadamente los módulos bluetooth, y para ello se utilizan los comandos AT.

Los comandos AT son unas instrucciones que se utilizan para configurar estos módulos bluetooth y que son heredadas de las instrucciones utilizadas para conectar los ordenadores a los primeros módems. Estos comandos se basaban en instrucciones sencillas de texto para configurar parámetros básicos de configuración y conexión. Se popularizaron tanto que hoy en día se siguen utilizando.

Para configurar y emparejar los módulos bluetooth lo primero es realizar las conexiones eléctricas adecuadas, siguiendo las instrucciones del datasheet del fabricante (que se podrá encontrar en el apartado de anexos), aunque en internet se pueden encontrar ejemplos de configuración y utilización [24].

Para comunicarnos con el módulo y poder configurarlo es necesario abrir un monitor serial desde el ID de Arduino y configurarlo para que no envíe carácter de final de línea en cada mensaje. Una vez hecho esto y cargado el programa de configuración del módulo (disponible en el apartado anexos) enviamos el comando “AT” y si todo es correcto debería respondernos OK. Ahora podremos configurar cosas como la velocidad de transmisión (“AT+BAUD”, que por defecto esta a 9600 baudios), el nombre del dispositivo (“AT+NAME”), la contraseña (“AT+PASS”, por defecto es 0000), o la dirección MAC del modulo (“AT+ADDR”). También se puede configurar el módulo como maestro o como esclavo, teniendo en cuenta que solo puede haber un maestro en la misma red y multitud de esclavos. Configuraremos el modulo del guante como maestro (utilizando el comando “AT+ROLE1”) y el modulo de la mano robótica como esclavo (“AT+ROLE0”). El resto de parámetros se pueden cambiar según las necesidades o dejarlos por defecto.

Desde el modulo que funcionará como maestro podremos realizar un escaneo de direcciones (“AT+DISC?”) que nos permitirá detectar todos los dispositivos bluetooth disponibles y en rango, de forma que podremos seleccionar el módulo que utilizaremos como esclavo para que se emparejen y una vez emparejados se volverán a emparejar por defecto cuando se enciendan, a nos ser que cambiemos la configuración.

Una vez los módulos bluetooth se han configurado y emparejado correctamente, la transmisión de datos entre ellos se puede realizar mediante la consola o bien por código, de forma análoga a las comunicaciones Serie anteriormente explicadas.

5.1.1 Arduino Maestro

Como explicábamos anteriormente el Arduino localizado en el guante capturador es el que realizará las funciones de maestro. Su función será la toma y envío de datos mediante cable o de forma inalámbrica. Veamos en mas detalle las funciones del programa:

La función principal del programa es la toma y escalado de los datos, mediante las funciones “analogRead” y “map” respectivamente. Este escalado es imprescindible para poder aprovechar toda la resolución del conversor digital analógico incluido en el propio microcontrolador, y no perder así

información u obtener medidas erróneas. De esta forma, tomando las medidas analógicas que se obtienen con los dedos estirados y encogidos al máximo, podemos enviar un dato de la medida del sensor en forma de byte con la mayor exactitud posible.

Otra parte importante del programa, de cara a la simulación por ordenador, es el control del tiempo de ejecución:

El microcontrolador tiene un cristal de 16 MHz, por lo que su tiempo de ejecución del bucle principal es bastante rápido. Por el contrario, el tiempo de transmisión de la información mediante protocolo Serie es mas lenta (configurado a 115000 baudios) así como los usos que le vamos a dar a esta información, por lo que será necesario realizar un control del tiempo de ejecución, para que el microcontrolador no vaya mas rápido de lo deseado y no produzca lecturas erróneas en el ordenador.

Para ello, se utiliza un sencillo bucle temporal sin ninguna función en su interior que detiene la ejecución de instrucciones sin detener completamente el programa. De esta forma no se realiza más de una medida y una transmisión de datos en un milisegundo, que se ha determinado como el periodo de muestreo establecido. Este tiempo de muestreo de un milisegundo se ha estimado por ser un compromiso entre velocidad y capacidad de respuesta de la mano mecánica.

De esta forma tenemos un programa que cada milisegundo realiza la lectura de cada sensor, la escala de forma conveniente, y la transmite mediante puerto serie al ordenador donde se realiza la simulación, o mediante bluetooth a la mano robótica.

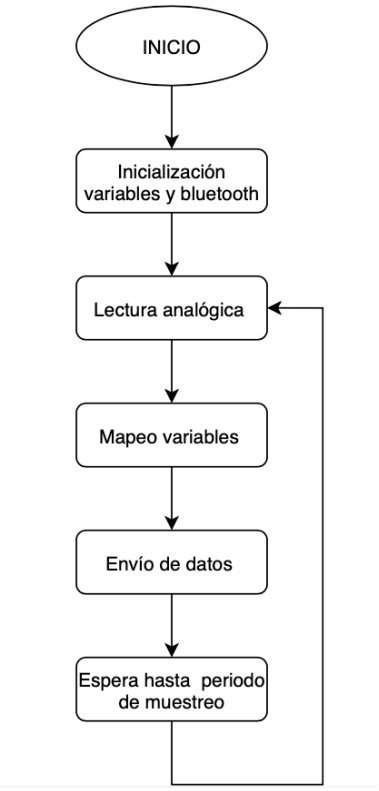


Ilustración 34: Diagrama de flujo del Arduino Maestro.

5.1.1 Arduino Esclavo

El Arduino localizado en la mano robótica cumple las funciones de esclavo, ya que su función es atender la información transmitida por el maestro cada vez que tenga una comunicación.

Para ello, se han configurado los módulos bluetooth para que se conecten automáticamente y se utiliza una función propia de la librería existente para la utilización del módulo bluetooth, que detecta cuando existe información entrante. Esta información, que se trata de un byte por cada dedo a mover, se asigna a una variable con la que después se calculará la posición a la que corresponde en el servo asignado, gracias a la función "map", y después se transmitirá al servo para que adopte dicha posición.

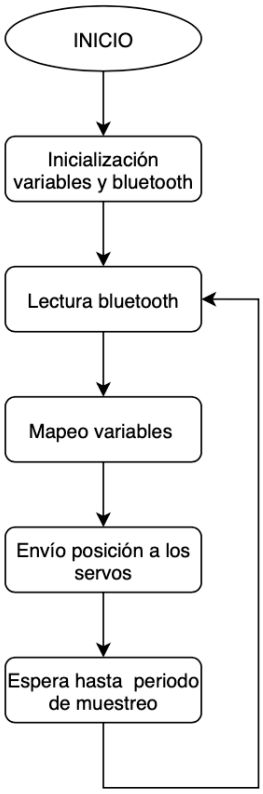


Ilustración 35: Diagrama de flujo del Arduino Esclavo.

El código completo de todos los programas utilizados se encuentra disponible en la sección ENEXOS.

5.2 Matlab-Simulink

El programa Matlab es un entorno de programación muy útil y potente para realizar cálculos matemáticos complejos. Fue creado a finales de los años 80 con el fin de realizar cálculos con matrices y representación de funciones [25]. A lo largo de los años, el propósito original de Matlab ha ido evolucionando para tratar otros aspectos más amplios de la ciencia, para tratar ramas tan diversas como la simulación de trayectorias aeroespaciales, la robótica, entre otros. Cómo no,

Arduino es una de esas ramas que no podían faltar en Matlab, por lo que se han ido añadiendo paquetes de programación que permiten el uso conjunto de ambos sistemas.

Valiéndonos de la compatibilidad de estas dos plataformas, utilizaremos la comunicación serie de Arduino para obtener la información de las posiciones de cada dedo en Matlab y, gracias al entorno Simulink, que permite realizar simulaciones de cuerpos solidos, realizar una simulación por ordenador de la mano robótica.

La simulación de la mano robótica hace uso de la comunicación serie entre Arduino y Matlab. Utilizando el bloque “Serial Configuration” se define el puerto donde se conecta el Arduino, así como el tipo de datos que espera recibir y la cantidad (esto debe coincidir con el formato de datos que hayamos programado en el microcontrolador Arduino). Por otro lado, con el bloque “Serial RCX” podemos extraer las señales recibidas del Arduino y utilizarlas para realizar los movimientos de la mano simulada.

En nuestro caso hemos definido en Arduino que se enviarán cuatro datos de tipo byte con la información de la lectura de cada uno de los sensores del guante. Además, estos datos los vamos a escalar entre +/-1 para que sea más sencillo tratar con ellos, de forma que -1 significa dedo completamente extendido y +1 totalmente encogido. Los valores decimales comprendidos entre estos dos números será cualquier posición intermedia entre las dos anteriormente descritas. Se ha decidido realizar esta transformación por comodidad, simplicidad y para dotar de cierta flexibilidad al programa de cara al uso de estos datos en otros propósitos, ya que siempre es mas sencillo tratar con datos normalizados.

Para realizar la parte visual de la simulación de la mano, lo primero que necesitamos es un sólido diseñado en 3D. De nuevo nos valdremos de los diseños compartidos en internet con licencia libre para realizar la simulación. Estos sólidos tendrán que unirse entre ellos mediante articulaciones, definidas por el entorno Simulink, y configuradas por el usuario. De esta forma podremos definir características físicas del cuerpo, como la masa, rigidez, aspecto visual, etc; además de definir cómo se unen entre ellos mediante articulaciones de diversos grados de libertad, cómo les afecta la gravedad, entre otros.

Puesto que los sólidos son cuerpos con formas dispares que se han diseñado en otro programa de CAD, será necesario definir en qué lugar se encuentran las uniones donde situaremos las articulaciones. Esto se realiza definiendo los marcos de referencia o “frames”. Una vez colocado un “frame” en cada lugar donde deseemos conectar con cuerpo con otro también será necesario orientarlo de manera adecuada, ya que, por ejemplo, la gravedad siempre afecta en el eje Z. Para ello nos valdremos de las transformaciones de coordenadas o “rigid transforms”, que nos permitirán orientar y trasladar los marcos de referencia según nuestras necesidades y poder conectar de forma correcta todos los cuerpos que componen la simulación. En nuestro caso, se define el eje Z como eje de revolución y no se tiene en cuenta la gravedad para simplificar la simulación, ya que tiene muchos elementos móviles y no es un elemento que aporte valor al sistema final.

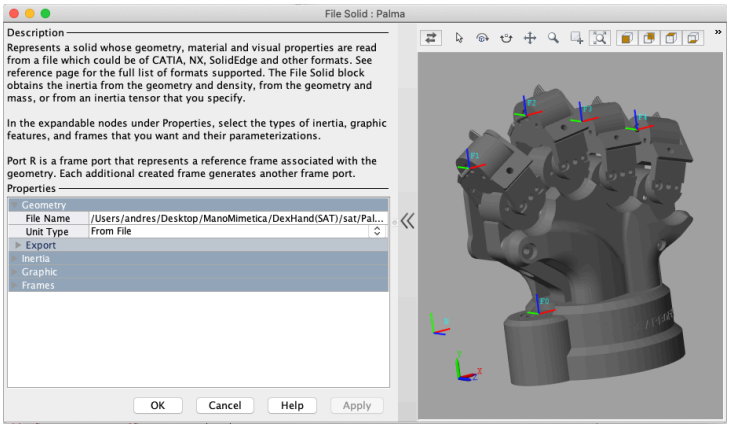


Ilustración 36: Colocación de ejes de coordenadas (frames).

Una vez conectados los sólidos de la manera deseada será necesario definir como se van a mover entre ellos. A cada articulación hay que definirle si tiene una señal de entrada, y si ésta es de movimiento o par. En nuestro caso se determina la entrada de movimiento (motion) por simplicidad. El caso más realista sería introducir una señal de par (torque) pero al estar destinado en el mundo real a mover servomotores, que disponen internamente de su control electrónico, podemos obviar la entrada de par y simplificar el sistema mandando la señal de posición a cada articulación.

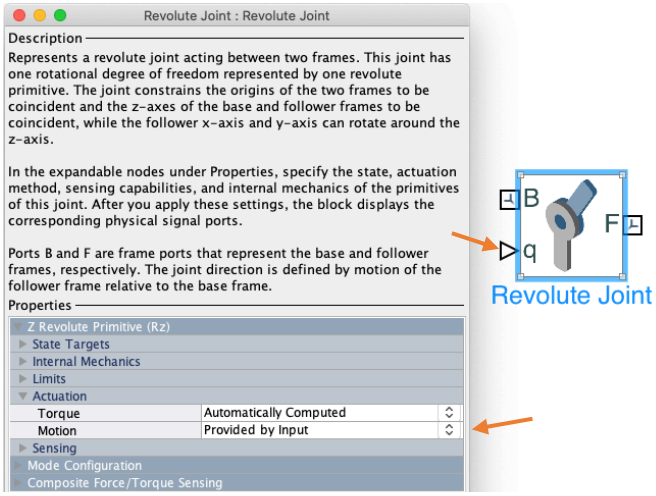


Ilustración 37: Configuración del par R.

Como primera prueba, se simula únicamente el sistema mecánico, de forma que se mueva con el ratón del ordenador y se verifique la correcta orientación de cada dedo y su articulación conectada.

Para ello, será necesario introducir una ganancia en cada señal de entrada. Se define una ganancia estática de 1, y una ganancia regulable “Slider Gain”, de forma que se pueda variar manualmente durante la simulación para adecuar los movimientos de la mano simulada a unos movimientos realistas compatibles con una mano real.

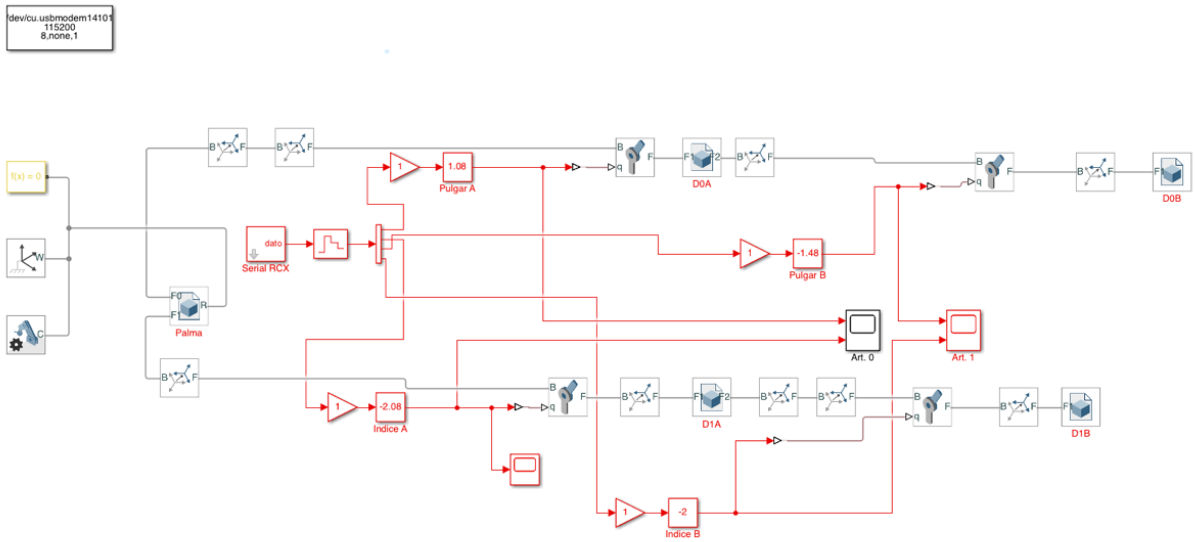


Ilustración 38: Primera prueba en Matlab-Simulink.

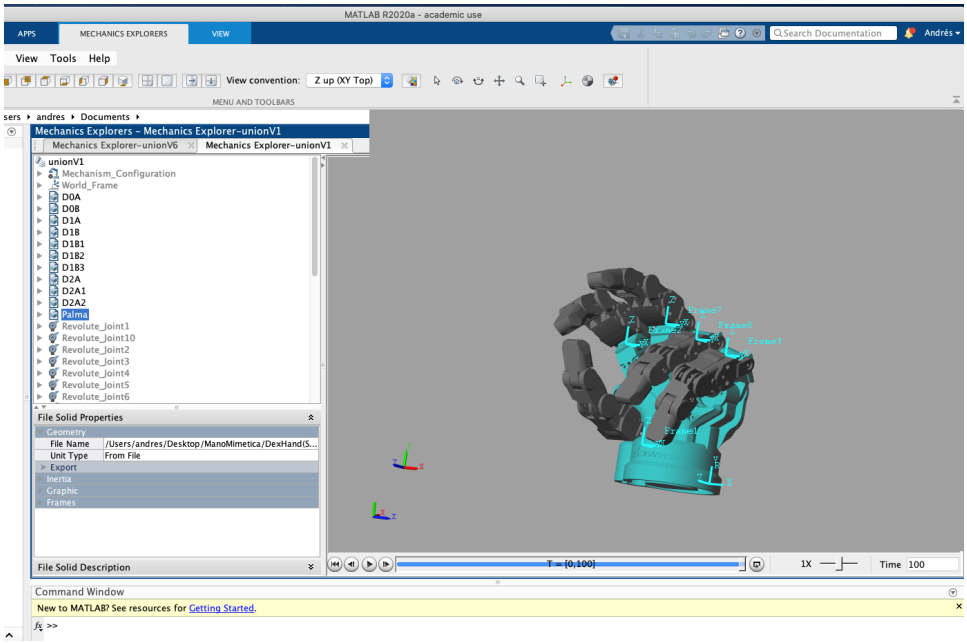


Ilustración 39: Simulación de la primera prueba en Matlab-Simulink.

Una vez comprobada la correcta visualización y funcionamiento del modelo podemos conectar el guante y pasar al ajuste de los movimientos, para que se adecuen al movimiento real capturado con el guante.

En primer lugar, se ha reducido el número de piezas móviles para agilizar la ejecución del programa, dejando únicamente los dos dedos que harán el movimiento de pinza.

Por otro lado, se han añadido dos bloques extra. El primero de ellos es el encargado de asegurar que las señales de entrada no exceden los valores esperados. Debido a la propia instalación de los sensores analógicos, detectamos ruidos electromagnéticos e interferencias que afectan a la

simulación. Este problema se soluciona fácilmente con un bloque “saturation” que limita la señal de entrada entre +/- 1.

Por otro lado, se utiliza un bloque “bias” que añade un nivel de continua a la señal inicial, de forma que los dedos quedan orientados para cada señal recibida según nuestras necesidades.

La siguiente imagen muestra cómo quedaría el programa final con todos los bloques descritos anteriormente.

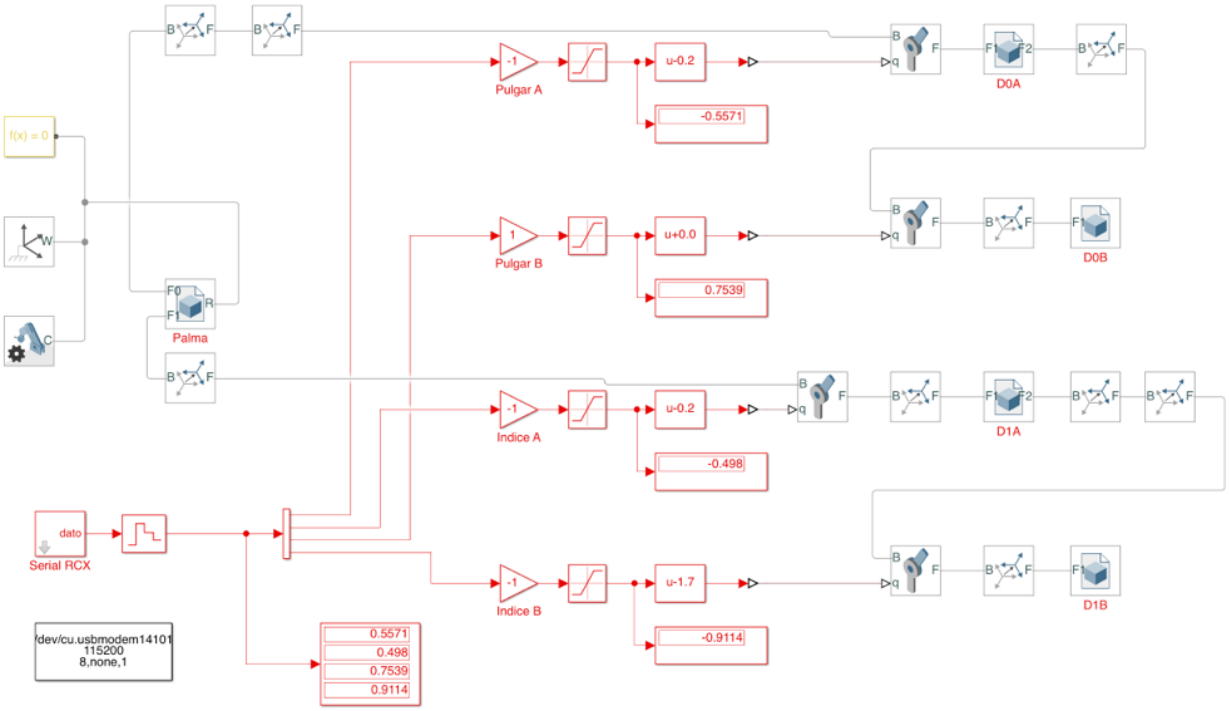


Ilustración 40: Simulación final.

6.Resultados

A continuación, se muestran los resultados obtenidos tras el desarrollo del proyecto. Se dividirá en las tres partes principales del proyecto, el capturador de movimientos, la simulación por ordenador y la mano robótica.

6.1 Guante sensor

Las pruebas realizadas únicamente al guante sensor fueron satisfactorias. Se configuró una visualización de las señales correspondientes a cada articulación con Simulink, obteniendo las gráficas que se muestran a continuación. Para clarificar los datos se han separado para obtener dos gráficas una por cada dedo, con sus dos articulaciones respectivas:

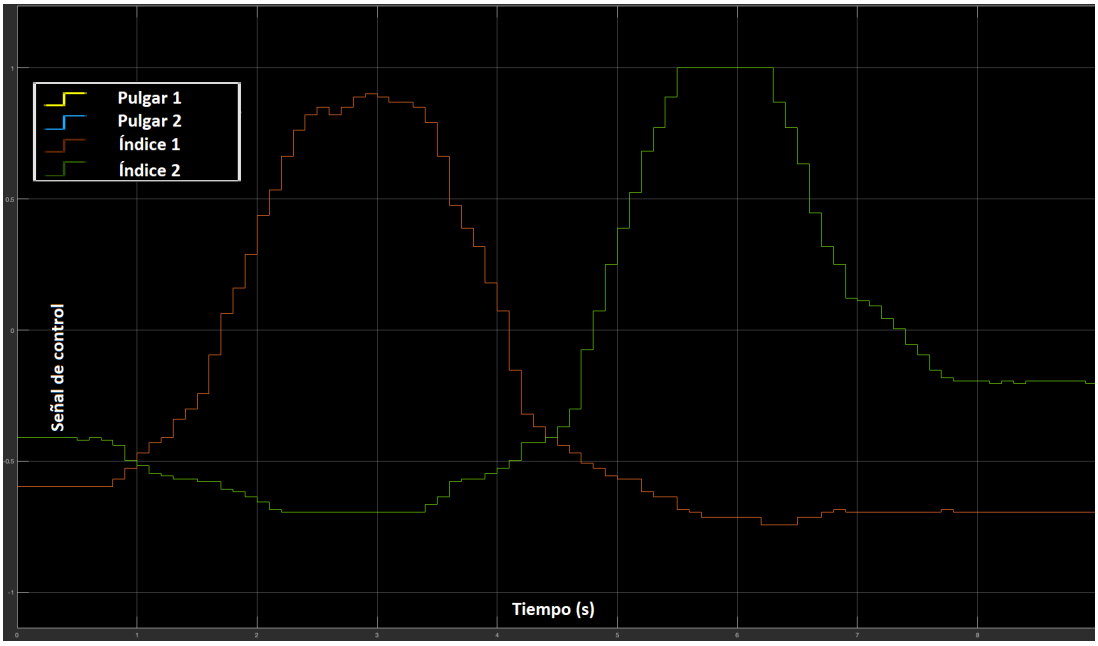


Ilustración 41: Señales de control del dedo índice.

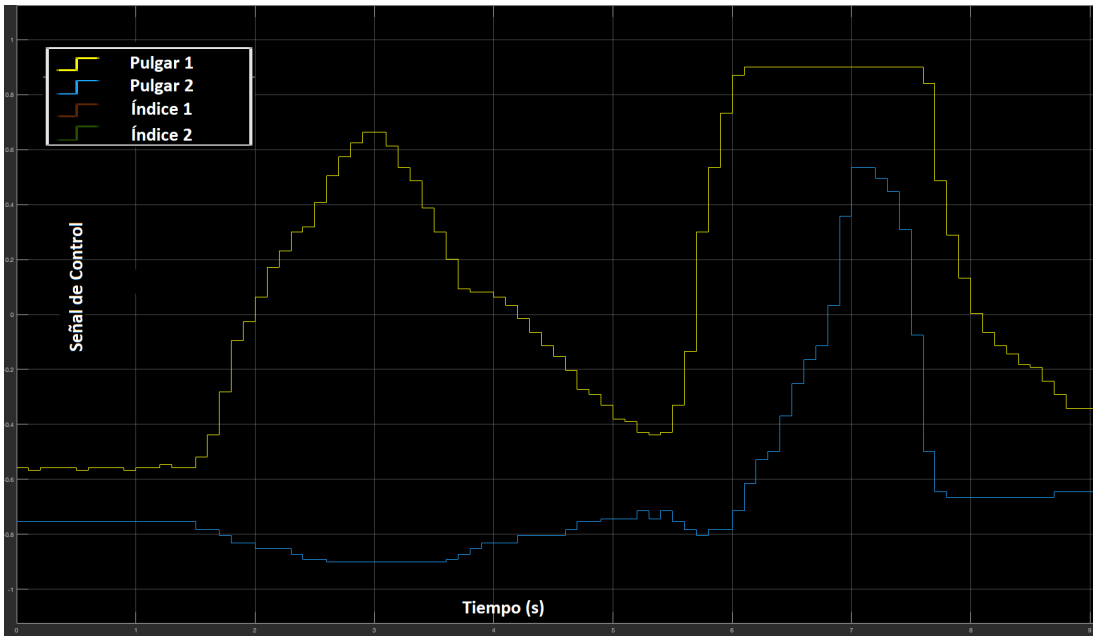


Ilustración 42: Señales de control del dedo pulgar.

En estas gráficas se puede ver como varía la señal recibida por el sensor de flexión asignado a cada articulación, así como la comprobación de que la señal aumenta conforma varía el ángulo de flexión de la articulación. También se puede comprobar cómo las señales no superan el margen preestablecido de +/-1. Por tanto, se puede afirmar que el guante es capaz de detectar el movimiento de cada dedo con dos grados de libertad y tiene capacidad de determinar el ángulo de giro de cada articulación.

Finalmente, y tras comprobar que el diseño era adecuado, se fijaron definitivamente todos los sensores, cables y microcontrolador al guante, para ello se utilizó hilo de coser y cola termofusible, obteniendo el prototipo final que se muestra a continuación:

Ilustración 43: Prototipo final del guante capturador de movimientos.

6.2 Simulaciones en Matlab-Simulink

El software Matlab-Simulink produjo los siguientes resultados:

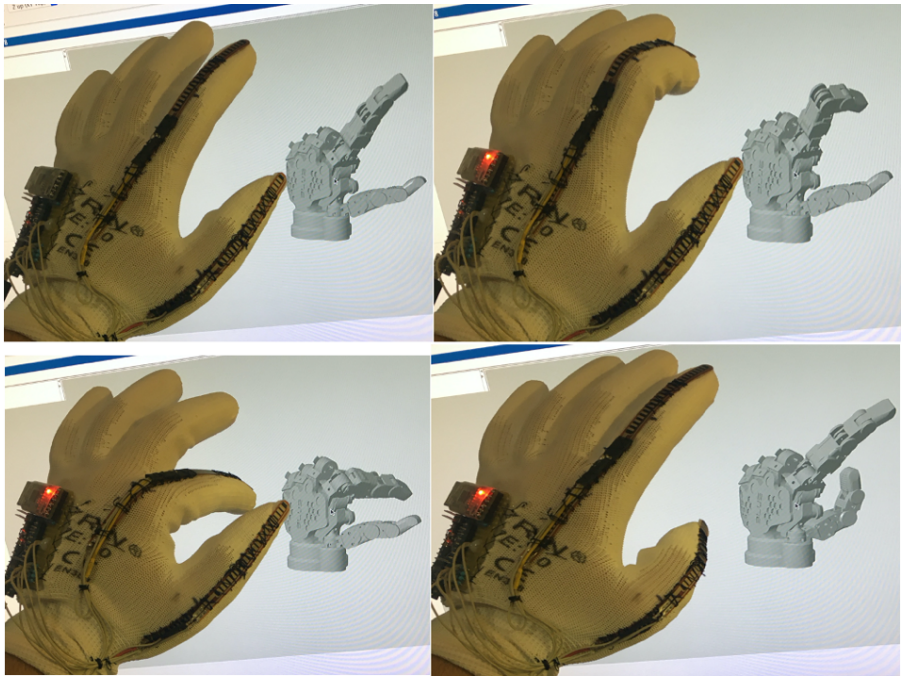


Ilustración 44: Simulación de la mano.

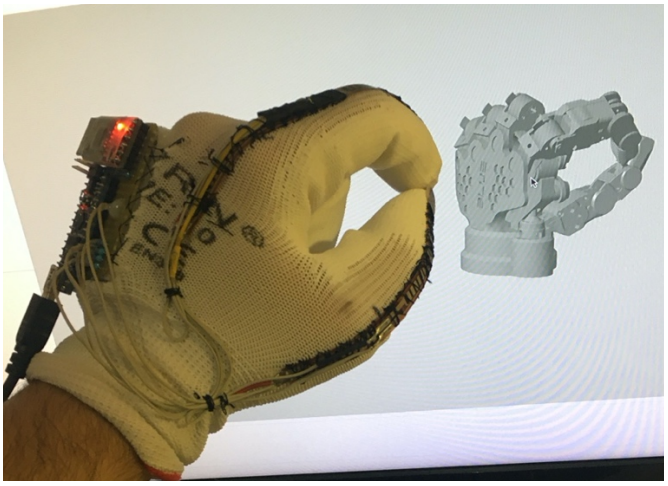


Ilustración 45: Simulación del movimiento de pinza.

En las imágenes se puede observar el cómo el modelo por ordenador imita a la perfección los movimientos efectuados con la mano que lleva el guante capturador. Se realiza el movimiento por separado de cada una de las cuatro articulaciones medidas y el movimiento de pinza característico de la mano para demostrar el correcto funcionamiento de la simulación.

Para comprobar el tiempo de simulación se fijó en Simulink un periodo de simulación de 60 segundos y se cronometró con un reloj por separado, obteniendo una desviación menor de un segundo, posiblemente debido a la incertidumbre propia del retraso que introduce Matlab al iniciar la simulación y la imposibilidad física de accionar los dos elementos exactamente al mismo tiempo. En cuanto a los movimientos se comprobó de manera visual que se corresponden de forma adecuada a los movimientos de la mano real.

Por tanto, podemos determinar que la simulación por ordenador en Simulink se comporta adecuadamente siguiendo los movimientos de la mano humana, de forma veraz y a tiempo real.

6.3 Mano 3D

Como se explicaba anteriormente, para mover la mano robótica se utiliza otro Arduino que transforma las señales recibidas por bluetooth en el ángulo que debe adoptar cada servo. Las gráficas obtenidas con este Arduino son las siguientes:

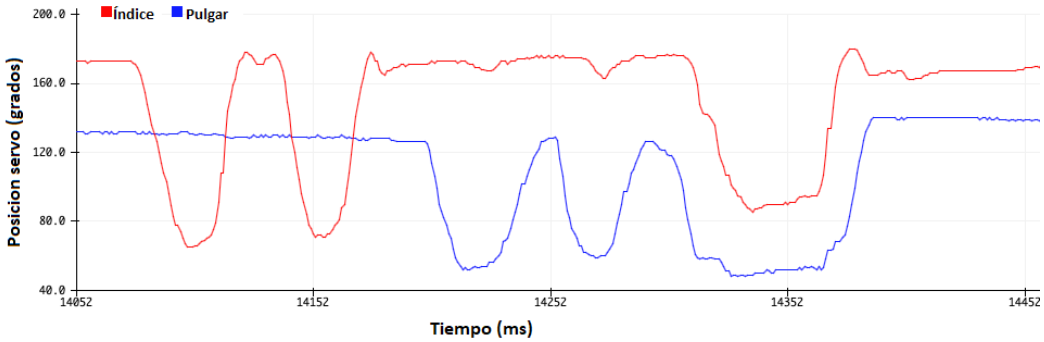


Ilustración 46: Señal de control de la mano robótica.

En estas gráficas podemos observar como los servos que mueven el pulgar y el índice toman un ángulo comprendido entre 60 y 180 grados aproximadamente, según la señal recibida del guante.

Las imágenes de la mano impresa en 3D totalmente ensamblada y en funcionamiento son las siguientes:

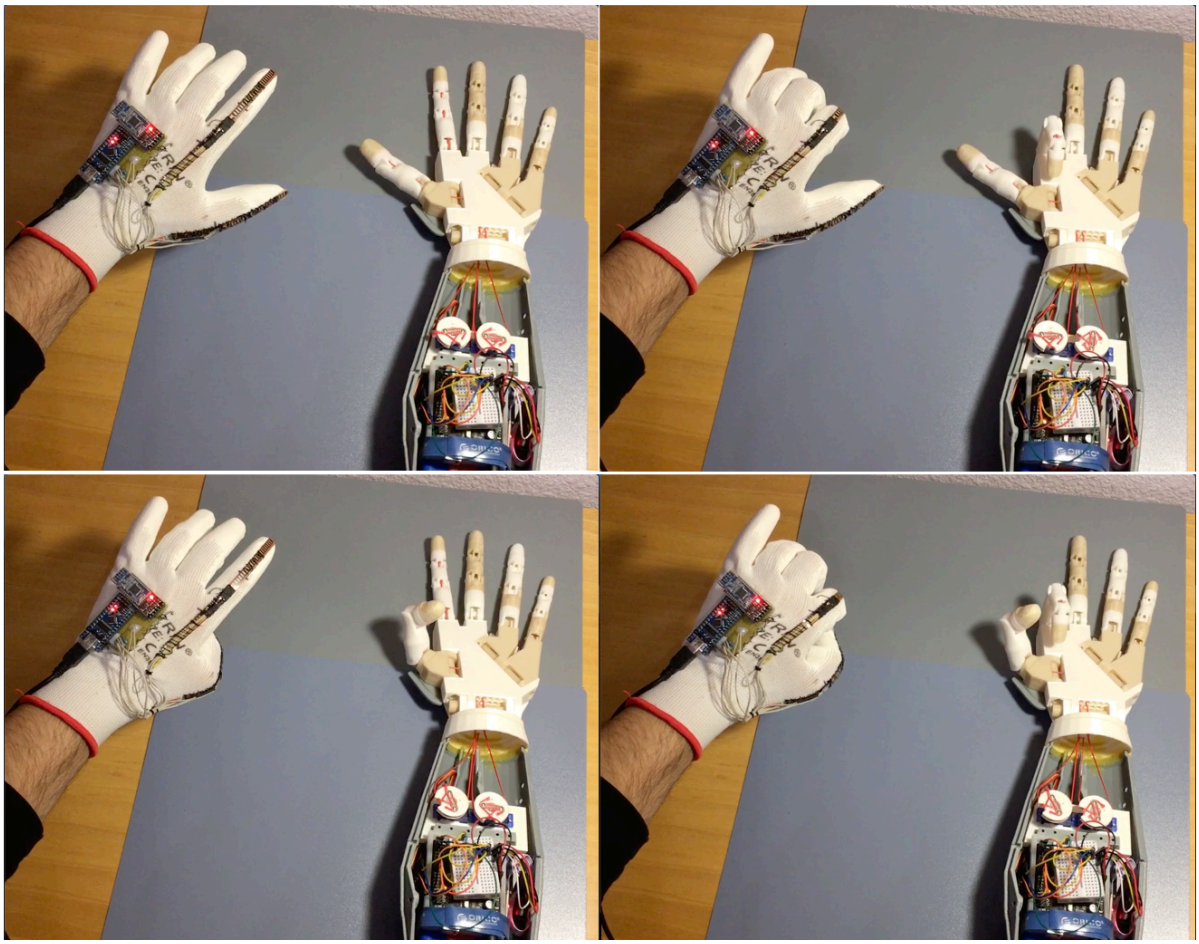


Ilustración 47: Mano robótica en funcionamiento.

El modelo mecánico es capaz de mover los dos dedos siguiendo los movimientos de la mano humana y hacer el movimiento de pinza que era el objetivo de la mano robótica.

Con los resultados obtenidos, podemos determinar que la mano mecánica se comporta de una manera adecuada, y con un tiempo de respuesta adecuado teniendo en cuenta sus propias limitaciones mecánicas.

7. Problemas encontrados y soluciones aplicadas

Durante la implementación del primer prototipo, así como del prototipo final, se encontraron varios problemas que debieron solventarse para finalizar el proyecto:

7.1. Elección del sensor

En el mercado se encuentran varios tipos de sensores, una vez decantados por el sensor de flexión de Spectra, el primero en ser probado fue el sensor de 4,5´ (11,43 cm), que dio un buen resultado en cuanto a comportamiento, pero era demasiado largo para nuestras necesidades, ya que abarcaba dos falanges. Para lograr los dos grados de libertad requeridos fue sustituido por el de 2,2´ (5,5 cm) que, al ser mas corto, podía abarcar solo una de las falanges de cada dedo. Aun así, la falange del dedo pulgar seguía siendo demasiado corta, por lo que hubo que prestar especial atención a la forma de posicionar este sensor en el guante textil.

7.2. Fijación de los sensores flex

Los sensores flex no tienen ningún punto de fijación como tornillos, pegamento, etc.

La primera opción meditada fue la aplicación de algún tipo de pegamento o cola termofusible, que responde muy bien a la unión de tejidos con otros materiales y es resistente, pero por la dificultad de posicionamiento de los sensores se prefirió coser directamente los sensores en cada dedo, de forma que permitiera cierta corrección una vez colocados. Finalmente, y una vez asegurada su exacta colocación en el guante se procedió a aplicar una pequeña cantidad de adhesivo para fijar completamente la posición de cada sensor.

Para esta labor y para la fijación del resto de componentes fue de gran ayuda la utilización de una mano articulada de madera, que normalmente se usa para modelar y demás trabajos artísticos, pero en esta ocasión servía para poder ver la forma que adopta el guante durante su uso y poder asegurar correctamente los cables y demás elementos delicados.

Ilustración 48: Mano de madera a escala humana.

7.3. Conexión de los sensores con el microcontrolador

Los sensores flex están pensados para ser conectados mediante una conexión estándar extraíble. Pero para nuestro propósito, una conexión extraíble supone un punto de falla probable al estar situados en una zona de movimiento habitual. El problema se solucionó soldando los cables directamente al sensor y conectándolos al microcontrolador. Estos cables a su vez deben de ser finos y flexibles, ya que los primeros que se utilizaron tendían a romperse tras poco uso por ser demasiado rígidos.

7.4. Carga de procesamiento en la simulación

Durante la elección del modelo a simular se escogió un modelo relativamente realista de una mano humana, que contenía multitud de piezas y huecos preparados para la inserción de motores de corriente contínua que le dotaran de movimiento. Esto suponía un gran coste computacional en la simulación del movimiento de la mano, siendo la mayoría de pequeñas piezas inútiles para el caso de este proyecto. Se procedió pues a la simplificación del modelo utilizando programas de CAD como Fusion 360 y Solidworks, tanto para la eliminación de elementos inútiles, simplificación de piezas y transformación de formatos de archivos para que Matlab lo pudiese simular.

7.5. Cambio de Modelo de Arduino Mega por Arduino NANO

La primera opción para realizar el guante capturador de movimientos fue utilizar un Arduino MEGA, ya que dispone de muchas entradas analógicas. Más tarde y conforma avanzaba el proyecto, las limitaciones económicas hicieron imposible colocar tantos sensores como eran necesarios para registrar el movimiento de todos los dedos con dos grados de libertad. Esto hizo posible cambiar el Arduino por otro de dimensiones más reducidas como el NANO. Este dispositivo tiene menos entradas, por lo que es mucho más pequeño, barato y de menor masa. De esta forma es más fácil de fijar y disimular en el guante capturador, y por tanto menos incómodo para el usuario. Además, al ser de la familia Arduino no supuso cambio en absoluto en el código del programa.

Este cambio de hardware tubo un cambio significativo en el desarrollo del guante, ya que el Arduino NANO no dispone de pines hembra de conexión rápida como el MEGA, por lo que fue necesario realizar un circuito soldado sobre una placa de baquelita micro perforada.

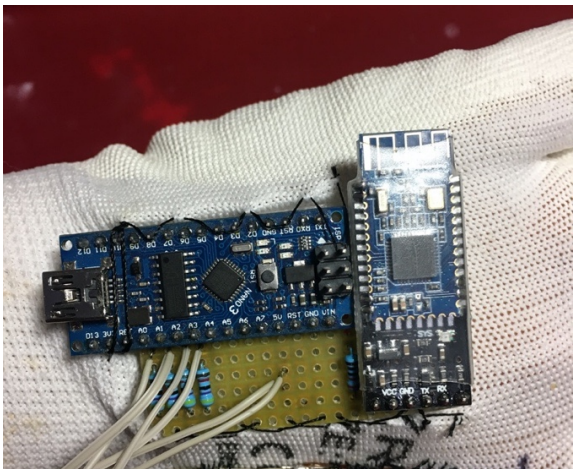


Ilustración 49: Detalle placa micro perforada.

7.5. Problemas subiendo programas a Arduino

Tras varios intentos fallidos de programación y sincronización del Arduino NANO con el ordenador, se descubrió que el modelo tiene varias versiones con distintos micros y era necesario configurar el modelo concreto de microcontrolador que llevaba en el IDE Arduino. Una vez consultada la documentación y configurado correctamente, la programación se realizó sin ningún problema de forma análoga a los otros modelos.

7.6. Niveles de tensión en el módulo bluetooth

Todos los módulos compatibles con Arduino normalmente tienen niveles de tensión de cinco voltios de alimentación. En el caso del módulo bluetooth, tras consultar la documentación y demás tutoriales disponibles en internet, se detectó que la señal de transmisión de datos no es oficialmente compatible con 5 voltios, sino que debe ser a 3,3 V. Esto se solucionó con un sencillo divisor de tensión en la patilla correspondiente del Arduino, pero produjo varios errores de comunicación en los primeros intentos de programación de estos módulos.

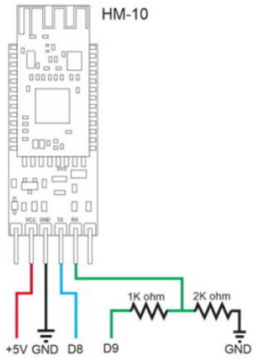


Ilustración 50: Diagrama de conexión del modulo HM-10. [26]

7.7. Versiones y modificaciones de la mano 3D

Las piezas impresas fueron obtenidas de un diseño Open Source disponible en la red, pero se hicieron varias modificaciones para adaptarla al proyecto. Un de ellas fue el recorte de la pieza de la muñeca que por problemas de versiones no encajaba con las del antebrazo. La siguiente fue la adaptación del bastidor con la electrónica interna, ya que el diseño original no pretendía tener una batería ni un Arduino para que trabajase independientemente del resto del robot.

Estas modificaciones tuvieron que realizarse mecanizando las piezas originales ya que, por problemas de limitación de acceso a la universidad, no se disponía del laboratorio de impresión de nuevo. Pero se tienen en cuenta las modificaciones para futuras mejoras y réplicas del proyecto.

8. Mejoras propuestas y futuras líneas de investigación

Puesto que la idea inicial del proyecto era conseguir una mano con cinco dedos funcionales, la primera propuesta de mejora será la inclusión de los sensores en los dedos restantes del guante y añadir tres servos más en la mano robótica para poder mover los cinco de dos de la mano independientemente. Este proceso no se detallará más puesto que es repetir tres veces lo ya realizado con el dedo índice y no supone ningún reto ni complicación mas allá de la económica y del consumo de tiempo.

A continuación, trataremos uno de los problemas encontrados durante la consecución del proyecto, que fue la inestabilidad de las señales transmitidas por los sensores de flexión. Al tratarse de sensores analógicos y por la proximidad de los sensores y los cables, se obtiene un ruido

electromagnético importante a la entrada del microcontrolador, que es necesario tener en cuenta cuando se requiere cierta precisión en las lecturas. Una buena opción para reducir este problema y aumentar la precisión en las lecturas es utilizar un circuito acondicionador de señal, compuesto normalmente por un amplificador operacional. La utilización de un amplificador operacional reduciría la incertidumbre de las medidas al no interferir (teóricamente) en la corriente que circula por el sensor, asimismo se podría mejorar la estabilidad de las señales colocando un integrado para transformar las señales analógicas en señales digitales, por ejemplo, utilizando el protocolo I2C, de forma que sean menos sensibles al ruido electromagnético.

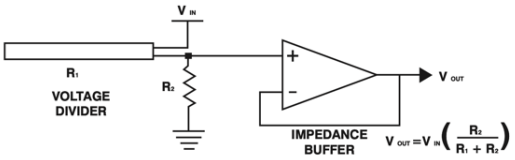


Ilustración 51: Circuito adaptador de señal con amplificador operacional. [27]

Ilustración 52: Sensor digitalizado para transmisión de datos por protocolo I2C. [28]

En cuanto a las comunicaciones, los módulos bluetooth utilizados tienen un alcance de alrededor de 10 metros, con lo cual su aplicación real estaría limitada a entornos reducidos. La utilización de otras tecnologías inalámbricas como la radiofrecuencia o un enlace mediante wifi permitirían la comunicación a kilómetros de distancia, ejemplo de ello serian los módulos Xbee o Wifi disponibles en tiendas de electrónica.

Por otro lado, la mano robótica tiene sus propias limitaciones características del material utilizado para construirla. El hecho de ser de polímero PLA hace que no pueda resistir grandes esfuerzos mantenidos en el tiempo. El plástico, así como otros materiales similares, sufre el efecto conocido como creep o “fluencia lenta”, puesto que sufre deformaciones en función del tiempo aunque la fuerza a la que se le someta no varíe. Este material tampoco es adecuado para trabajar a temperaturas altas ya que su transición vítrea se produce a partir de los 60ºC [29] y comienza a deformarse. Si se quisiera utilizar la mano robótica en un ambiente caliente o que realizase grandes fuerzas, sería necesario utilizar otros materiales, así como servomotores de mayor par.

Actualmente existen impresoras 3D capaces de imprimir piezas en una aleación de metales mucho más resistentes que el polímero, y también existen servomotores capaces de generar pares muy grandes, en comparación con los utilizados para este proyecto.

Estas tres propuestas permitirían mejorar sensiblemente el proyecto, manteniendo las características básicas del diseño. A continuación, se proponen algunas mejoras sobre el concepto propio del proyecto que permitirían continuar investigando sus aplicaciones:

Como se describe a lo largo de la memoria, los usos prácticos de la mano robótica podrían ser muchos, pero quedan limitados por el hecho de que la mano no tiene forma de desplazarse ni de orientarse hacia un objeto. Una posible ampliación del proyecto es la inserción de la mano robótica sobre un brazo robot comercial, o un vehículo. De esta forma se conseguiría mejorar mucho la funcionalidad y sus posibles aplicaciones.

Otra función añadida a esta mano sería la posibilidad de ampliar el número de articulaciones disponibles. En algunos trabajos podría ser útil tener más dedos disponibles (por ejemplo dos pulgares), que estos dedos se moviesen de forma peculiar o tener articulaciones distintas a las de la mano humana (más grados de libertad).

Para finalizar este apartado, la unión de este guante capturador con tecnologías de visión artificial podría darle una funcionalidad extra, de forma que el operador de un robot o un usuario de videojuegos pudiese sentirse dentro del robot o de la acción, permitiendo una experiencia inmersiva y favoreciendo el control de cualquier elemento sobre el que se quiera actuar.

8. Conclusiones

Durante este proyecto propuesto como trabajo de fin de máster se han podido reunir y desarrollar muchas de las competencias adquiridas y mejoradas durante el curso académico. Previamente a este máster, el autor no tenia ningún conocimiento de diseño de piezas en 3D ni de su adecuada fabricación en impresoras 3D. Este fue uno de los puntos críticos que hizo inclinar la balanza hacia la elección de este proyecto como propuesta en firme. Tampoco había trabajado con protocolos de comunicación inalámbrica, ni con montajes mecánicos compuestos por multitud de piezas, que es otro de los aspectos clave del proyecto.

Tras una pequeña presentación al tutor del proyecto, quedó patente que se trataba de algo muy adecuado para este trabajo y que, aunque se habían realizado diseños similares, eso no suponía un impedimento para realizarlo e incluso se podían utilizar estos proyectos previos como punto de apoyo y mejora al presente proyecto. De esta forma, se optó por investigar qué modelos se encontraban disponibles en internet con licencia libre para su uso académico y se encontraron multitud de prototipos y diseños disponibles. Esta metodología es muy utilizada en la ingeniería, ya que muchas veces el ingeniero no tiene que concebir una solución desde cero, sino que debe ser capaz de encontrar qué soluciones aproximadas se han desarrollado previamente y adaptarlas a sus propias necesidades. Esto permite al ingeniero encontrar soluciones a problemas complejos en un tiempo relativamente corto. Así pues, se decidió utilizar un modelo 3D existente e ir un paso mas allá e incluir de la manera mas fiel posible todos los movimientos de una mano humana.

Una vez definido el proyecto, se hizo evidente que había que limitar la carga de trabajo, para poder obtener un mínimo producto viable en el tiempo requerido. Además, tras las primeras pruebas satisfactorias del guante sensor, el autor tuvo que adaptarse a las condiciones externas que supusieron la restricción de acceso a la universidad y a los laboratorios donde se debían reproducir

las piezas correspondientes a la mano robótica. Esto hizo especialmente difícil el desarrollo del proyecto, aunque el tutor del proyecto seguía siendo accesible mediante E-mail, ya que puso a prueba la capacidad de desarrollo, de investigación y de resolución de problemas de forma autónoma.

Una vez finalizada la fase de diseño y pruebas de ambos elementos, los problemas encontrados y las soluciones dadas sirvieron de mucho al autor y permitieron ampliar su experiencia en el mundo de la impresión de piezas en 3D y la simulación por ordenador, sin dejar de lado el resto de aptitudes puestas a prueba durante el proyecto como la programación estructurada, o la electrónica analógica y digital.

Este proyecto intenta definir un dispositivo capaz de obtener los movimientos de una mano humana y trasladarlos a otra mano robótica, pero es evidente que esta aplicación es solo una de muchas posibilidades. Seria posible utilizar este guante para realizar muchos trabajos, por lo que pese a estar contento con el trabajo realizado, el autor prefiere poner un punto y seguido a la concepción de este proyecto y deja abierta la posibilidad de implementar otras aplicaciones con él en el futuro, tal y como se comentaba en apartados anteriores.

9. Presupuesto

En este apartado se detallará el coste de realizar este proyecto, teniendo en cuenta tanto el coste material como la mano de obra requerida y el software utilizado.

9.1 Hardware

Hace referencia al coste de todos los elementos necesarios para construir tanto el guante capturador de movimientos como la mano robótica. Puesto que se trata de dos elementos diferentes se realizará una valoración de cada elemento por separado.

Guante capturador		
Artículo	Coste unitario (€)	Cantidad
Arduino NANO	24,2	1
Módulo MH-10	4,24	1
Resistencias	0,15	6
Cable USB	2,92	1
Flex Sensor	10,83	4
Guante textil	1,39	1
Cables	1,78	1
Placa microperforada	3,58	1
COSTE TOTAL (€)	82,33	

Mano robótica		
Artículo	Coste unitario (€)	Cantidad
Arduino UNO	25,95	1
Módulo HM-10	4,24	1
Resistencias	0,15	2
Servo sg90	3,73	2
Cable	3,22	1
Bateria	22,95	1
Cables dupont	1,78	1
Protoboard	4,28	1
Tiras de velcro	2,99	1
Hilo de nylon	2,00	1
Pegamento	7,10	1
Tornilleria	5,00	1
Mano impresa	108,00	1
Antebrazo impreso	50,00	1
COSTE TOTAL (€)	245,27	

9.2 Software

Para la realización de las simulaciones y el diseño de las piezas se han utilizado varios programas con restricciones de uso. En este caso particular se ha utilizado licencias gratuitas de estudiante por ser un trabajo académico, pero se incluyen los costes comerciales a efectos de ofrecer un presupuesto acorde a las condiciones laborales normales.

Licencias software		
Artículo	Coste unitario (€)	Cantidad
Matlab	2.000,00	1
Solid works	6.600,00	1
Arduino	Gratuito	1
COSTE TOTAL (€)	8.600,00	

9.3 Recursos humanos

Este apartado hace referencia a la remuneración del trabajador que haya realizado las tareas necesarias para desarrollar el producto. Al tratarse de un trabajo académico el diseñador no ha recibido ninguna remuneración, por lo que no es un coste real, pero a efectos de veracidad se incluye un coste estimado de lo que costaría en horas de trabajo y en sueldo a un trabajador medio. Se determina que la labor se lleva a cabo por un solo ingeniero, sin tener en cuenta la ayuda prestada por el tutor del proyecto ya que su función se realiza en el plano académico y, aunque su ayuda ha sido de mucho valor, no se puede valorar de la misma forma ni bajo los mismos criterios que un ingeniero empleado.

Para valorar el trabajo del ingeniero se ha utilizado una estimación acerca de los sueldos medios que suelen percibir los ingenieros mecatrónicos, tasando el precio medio de la mano de obra en 20 €/h.

Recursos humanos			
Actividad	Coste por hora (€/h)	Horas (h)	Coste (€)
Análisis del problema	20,00	10	200,00
Pruebas iniciales	20,00	4	80,00
Simulacion	20,00	50	1000,00
Programacion	20,00	50	1000,00
Diseño de piezas	20,00	2	40,00
Montaje prototipos	20,00	50	1000,00
Depuracion de errores	20,00	10	200,00
COSTE TOTAL (€) (sin IVA)	3.520,00		
COSTE TOTAL (€)(con IVA)	4.259,20		

9.4 Presupuesto final

Para finalizar el apartado, realizaremos un presupuesto global que reúna todos los costes derivados del proyecto para poder valorarlo en conjunto desde el punto de vista económico.

Presupuesto global	
Concepto	Coste (€)
Material	327,60
Software	8.600,00
Personal	4.259,20
COSTE TOTAL (€)	13.186,80

10. Pliego de condiciones

En este apartado se contemplan todas las condiciones contractuales que deben cumplir este tipo de proyectos.

10.1 Definición y alcance del pliego

El objetivo del presente documento se centra en la propuesta, prueba y ejecución de un prototipo de guante capturador de movimientos y una mano robótica que imite los movimientos capturados por el guante. En el pliego de condiciones se recogen todos los aspectos y requisitos técnicos necesarios para la ejecución del proyecto tal y como se ha descrito durante todo el documento. Para ello se dividirá en distintos apartados que clasificará las condiciones que debe reunir este proyecto.

10.2 Condiciones generales

Se deben cumplir todas las normas generales en lo que a materia de seguridad e higiene en el trabajo respecta, así como las normas y recomendaciones particulares del entorno en el que se lleve a cabo el desarrollo del proyecto y las normas y recomendaciones de uso del fabricante de todos los equipos utilizados. El cumplimiento de estas normas es obligatorio tanto por la seguridad del propio individuo que lleva a cabo la tarea como por la seguridad de las demás personas que se encuentren alrededor, así como de los daños materiales y perjuicios que pueda ocasionar una negligencia o conducta irrespetuosa. El desconocimiento de la ley no exime de su cumplimiento, por lo que ante cualquier duda será responsabilidad del propio individuo el consultar la normativa vigente relativa a este ámbito de trabajo, así como el Reglamento Electrotécnico de Baja Tensión.

10.3 Especificaciones técnicas

10.3.1 Especificaciones de materiales y equipos

Los aparatos eléctricos y electrónicos necesarios para realizar este proyecto deberán ir acompañados de los certificados que acrediten el cumplimiento de las normativas por parte del fabricante. Se deberá seguir en todo momento las recomendaciones de éste, y ser utilizados para lo que han sido diseñados.

Para la fabricación de piezas en 3D se cumplirán todas las normativas en cuanto a fabricación aditiva, utilizando el material adecuado y bajo supervisión de personal debidamente formado y autorizado.

Para la realización del proyecto se han utilizado un ordenador con las siguientes características:

- MacBook Pro 13-inch, Late 2013
    - Procesador Intel Dual-Core i5 a 2,4GHz
    - Memoria RAM de 8 Gb
    - Tarjeta gráfica Intel Iris 1536 MB

En cuanto al software, reúne las siguientes características:

- Sistema operativo macOS Catalina 10.15.7
- Sistema operativo Windows 10 (como máquina virtual)
- Matlab 2020a
- SolidWorks 2019
- Arduino IDE

Estas características son suficientes para realizar los trabajos anteriormente descritos, aunque un equipo más potente podría producir resultados más rápidos y de forma más cómoda.

10.3.2 Especificaciones de ejecución

El proyecto se ha divide en varias fases bien diferenciadas:

1. Definición del proyecto y sus partes
2. Obtención y adaptación del modelo en tres dimensiones a simular
3. Construcción del prototipo de guante capturador
4. Implementación de la simulación por ordenador
5. Definición del modelo de mano robótica
6. Impresión de las piezas en 3D
7. Construcción y puesta a punto del modelo de mano robótica
8. Establecimiento de comunicación mediante bluetooth entre dos Arduinos
9. Depuración de errores
10. Redacción de la memoria

Cada uno de estos pasos deberá ser realizado por una persona capacitada para ello, que pueda garantizar su adecuado funcionamiento antes de pasar al siguiente apartado.

10.4 Condiciones legales

Dada la naturaleza didáctica del proyecto, la finalidad de este es la demostración práctica de las habilidades y competencias adquiridas durante el transcurso del Máster Universitario en Ingeniería Mecatrónica, por lo que no se establece ninguna obligación ni clausula legal sobre el diseñador del proyecto.

10.5 Gestión de residuos

El diseño de la mano robótica conlleva el uso de una batería de litio en su interior, la cual deberá ser convenientemente almacenada a media carga, en un lugar seco y alejada de fuentes de calor extremo, preferiblemente en una bolsa protectora ignifuga. Solo deberá usarse equipos certificados para su carga y bajo las condiciones que indica el fabricante de estos. Una vez finalizado su ciclo de vida deberá ser desechada correctamente en un punto limpio autorizado. Asimismo, los circuitos impresos que hayan cumplido su ciclo o que no funcionen correctamente deberán ser desechados convenientemente en los puntos limpios autorizados designados a tal efecto.

Bibliografía

[1] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/ASIMO.

[2] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Atlas_(robot).

[3] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/ASIMO.

[4] «Human Space Flight,» [En línea]. Available: https://spaceflight.nasa.gov/station/eva/robotics.html.

[5] M. Fernandez, «El Español,» [En línea]. Available: https://www.elespanol.com/omicrono/20190912/mano-robotica-usa-intenciones-funcionar-forma-remota/428707614_0.html.

[6] M. Fernandez, «El Español,» [En línea]. Available: https://www.elespanol.com/omicrono/20190912/mano-robotica-usa-intenciones-funcionar-forma-remota/428707614_0.html..

[7] «dreamstime,» [En línea]. Available: https://es.dreamstime.com/foto-de-archivo-anatom%C3%ADa-del-sistema-muscular-mano-músculo-de-la-palma-t-image27589400.

[8] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Mano.

[9] «Arduino,» [En línea]. Available: https://store.arduino.cc/arduino-nano?queryID=undefined.

[10] «Arduino,» [En línea]. Available: https://store.arduino.cc/arduino-nano?queryID=undefined.

[11] «spectrasymbol.com,» [En línea]. Available: https://www.spectrasymbol.com/product/flex-sensors/.

[12] «Hackaday.io,» [En línea]. Available: https://hackaday.io/project/26362-conomique-bionic-leg-system/log/66014-monitoring-system-design.

[13] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth#Bluetooth_v4.0_(2010).

[14] «Components 101,» [En línea]. Available: https://components101.com/wireless/hm-10-bluetooth-module.

[15] «Electrophone,» [En línea]. Available: https://www.electrophone.cl/insumos/1289-guante-antiestatico-talla-l.html.

[16] «arduino.cc,» [En línea]. Available: https://store.arduino.cc/arduino-uno-rev3.

[17] «Iberobotics,» [En línea]. Available: https://www.iberobotics.com/producto/micro-servo-towerpro-sg90-1-8kg9g0-12seg/.

[18] G. Langevin, «InMoov,» [En línea]. Available: http://inmoov.fr.

[19] G. Langevin, «InMoov,» [En línea]. Available: http://inmoov.fr/finger-starter/.

[20] G. Langevin, «InMoov,» [En línea]. Available: http://inmoov.fr/hand-and-forarm/.

[21] G. Langevin, «InMoov,» [En línea]. Available: http://inmoov.fr/lining-and-tighting-the-tendons/.

[22] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Ácido_poliláctico.

[23] «arduino.cc,» [En línea]. Available: https://forum.arduino.cc/index.php?topic=396450.0.

[24] M. Currey, «martyncurrey.com,» [En línea]. Available: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/.

[25] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/MATLAB.

[26] M. Currey, «Martyn Currey,» [En línea]. Available: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/.

[27] S. Symbol. [En línea]. Available: https://www.spectrasymbol.com/wp-content/uploads/2019/07/flexsensordatasheetv2019revA.pdf.

[28] «Sparkfun,» [En línea]. Available: https://www.sparkfun.com/products/14666.

[29] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Transición_V%C3%ADtrea.

[30] «Components101,» [En línea]. Available: https://components101.com/wireless/hm-10-bluetooth-module.

ANEXOS

Anexo I – Código

Código simulación con Matlab-Simulink

```
#define RT 13
#define MIN -1
#define MAX 1
long t1 = 0 ; long t2 = 0 ;
int T = 100  ;
float normal1 = 0; float normal2 = 0; float normal3 = 0; float normal4 = 0;
int Ain0 = A0; int Ain1 = A1; int Ain2 = A2; int Ain3 = A3;

void setup() {
  Serial.begin(115200);
  pinMode (Ain0, INPUT);
  pinMode (Ain1, INPUT);
  pinMode (Ain2, INPUT);
  pinMode (Ain3, INPUT);
}

void loop() {

//Toma de datos y mapeo entre posición extendida y encogida
  normal1 = map(analogRead(Ain0), 410, 590, MIN * 100, MAX * 100) / 100.0;
  normal2 = map(analogRead(Ain1), 380, 500, MIN * 100, MAX * 100) / 100.0;
  normal3 = map(analogRead(Ain2), 500, 610, MIN * 100, MAX * 100) / 100.0;
  normal4 = map(analogRead(Ain3), 340, 570, MIN * 100, MAX * 100) / 100.0;
// Envío de información por Puerto Serie
```

```
  Serial.write((byte)((normal1 - MIN) * (255.0 / (MAX - MIN)))) ;
  Serial.write((byte)((normal2 - MIN) * (255.0 / (MAX - MIN)))) ;
  Serial.write((byte)((normal3 - MIN) * (255.0 / (MAX - MIN)))) ;
  Serial.write((byte)((normal4 - MIN) * (255.0 / (MAX - MIN)))) ;
  Serial.flush() ;
// Bucle temporal para asegurar periodo de muestreo de 1 ms
  t2 = millis() ;
  if (t2 - t1 > T) {
    digitalWrite(RT, HIGH) ;
  }
  else {
    digitalWrite(RT, LOW) ;
  }
  while (t2 - t1 < T) {
    t2 = millis() ;
  }
  t1 = millis() ;
}
```

Código Maestro Bluetooth

```
//////////////////// ARDUINO MAESTRO /////////////////
#include <SoftwareSerial.h>
SoftwareSerial miBT(8, 9);
float normal1 = 0; float normal4 = 0;
int Ain0 = A0; int Ain3 = A3;
int raw1 = 0; int raw4 = 0;
int normal1int = 0; int normal4int = 0;
long t1 = 0 ; long t2 = 0 ; int T = 100  ;

void setup() {
  Serial.begin(9600);
  miBT.begin(9600);
  pinMode (Ain0, INPUT);
  pinMode (Ain3, INPUT);
}
void loop()
{
  // Lectura de datos de los sensores y mapeado entre posición extendida y encogida
  normal1 = map(analogRead(Ain0), 400, 605, MIN * 100, MAX * 100) / 100.0;
  normal4 = map(analogRead(Ain3), 300, 570, MIN * 100, MAX * 100) / 100.0;
// Cambio de tipo de dato a entero
  normal1int = (normal1 * 100) + 100;
  normal4int = (normal4 * 100) + 100;
// Envío de datos por bluetooth
  miBT.write(normal1int);
  miBT.write(normal4int);
  Serial.println(normal4int);
```

```
// Bucle temporal para asegurar periodo de muestreo de 1 ms
  t2 = millis() ;
  if (t2 - t1 > T) {
    digitalWrite(RT, HIGH) ;
  }
  else {
    digitalWrite(RT, LOW) ;
  }
  while (t2 - t1 < T) {
    t2 = millis() ;
  }
  t1 = millis() ;
}
```

Código Esclavo Bluetooth

```
//////////////////// ARDUINO ESCLAVO /////////////////
#include <Servo.h>
#include <SoftwareSerial.h>
long t1 = 0 ; long t2 = 0 ; int T = 100  ;
SoftwareSerial miBT(8, 9);
int dato1 = 0; int dato2 = 0;
int dato1map = 180; int dato2map = 180;
Servo Servopulgar;
Servo Servoindice;
void setup() {
 Servopulgar.attach(2);
 Servoindice.attach(3);
 miBT.begin(9600);
 Serial.begin(9600);
}
void loop() {
 if (miBT.available())  {  // Cuando detecta lectura bluetooth
   dato1 = miBT.read();
   dato2 = miBT.read();
 }
 dato1map = map(dato1, 50, 200, 80, 180);       //Escalado de variables
 dato2map = map(dato2, 0, 170, 65, 180);

 Servopulgar.write(dato1map);            //envío de posición a los servos
 Servoindice.write(dato2map);

t2 = millis() ;      //Bucle de espera
```

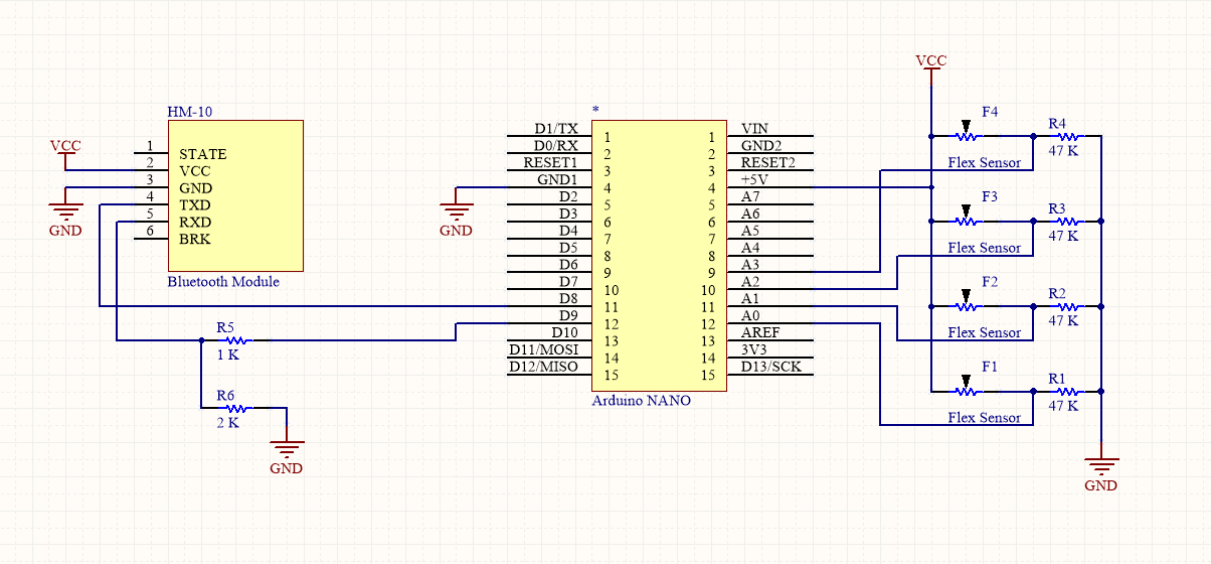
```
  if (t2 - t1 > T) {
    digitalWrite(RT, HIGH) ;
  }
  else {
    digitalWrite(RT, LOW) ;
  }
  while (t2 - t1 < T) {
    t2 = millis() ;
  }
  t1 = millis() ;
}
```

Código configuración Bluetooth

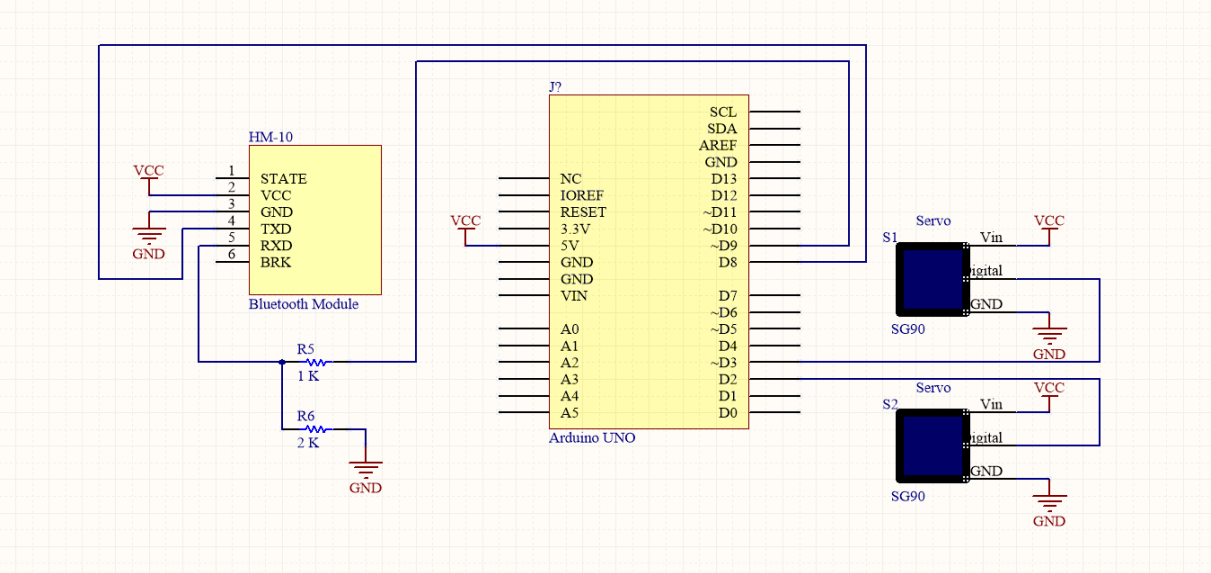
```
////////////////////// Configuración Bluetooth/////////////////
#include <SoftwareSerial.h>
SoftwareSerial HM10(8,9);
void setup(){
  Serial.begin(9600);
  Serial.println("INTRODUZCA COMANDOS AT:");
  HM10.begin(9600);
}
void loop(){
  if (HM10.available()){
    Serial.write(HM10.read());
  }
  if (Serial.available()){
  HM10.write(Serial.read());
  }
}
```

Anexo II – Diagrama de conexiones

Esquema eléctrico del guante capturador:

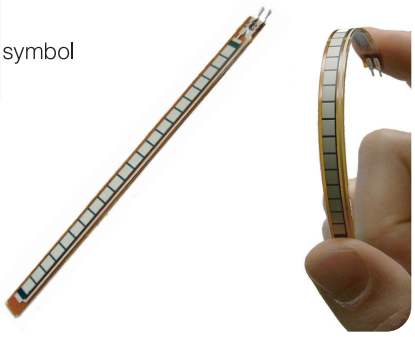


Esquema eléctrico de la mano robótica:



Anexo III – Datasheets

spectrasymbol



FLEX SENSOR FS

Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
- Simple Construction
- Low Profile

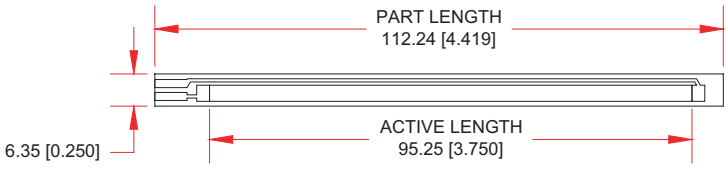
Mechanical Specifications

-Life Cycle: >1 million
-Height: ≤0.43mm (0.017")
-Temperature Range: -35°C to +80°C

Electrical Specifications

-Flat Resistance: 10K Ohms ±30%
-Bend Resistance: minimum 2 times greater than the flat resistance at 180° pinch bend *(see "How it Works" below)*
-Power Rating : 0.5 Watts continuous; 1 Watt Peak

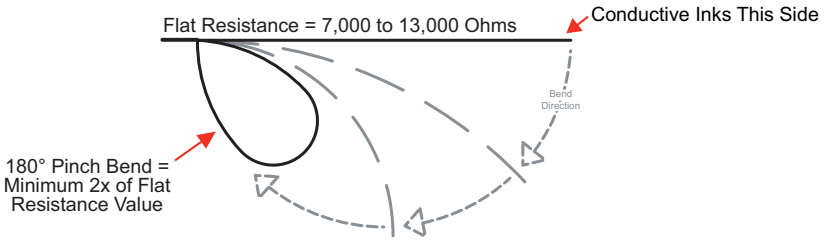
Dimensional Diagram - Stock Flex Sensor



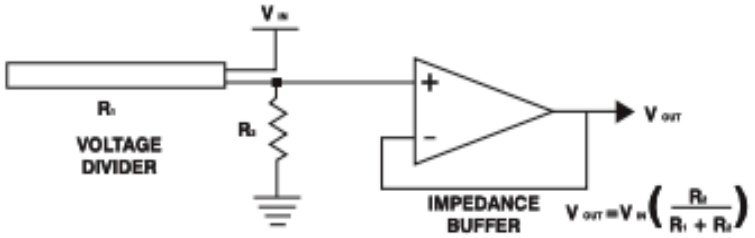
How to Order - Stock Flex Sensor

FS	–	L	–	0095	–	103	–	ST
Series		Model		Active Length		Resistance		Connectors
FS = Flex Sensor		L = Linear		0095 = 95.25mm		103 = 10 KOhms		ST = Solder Tab

How It Works



Schematics

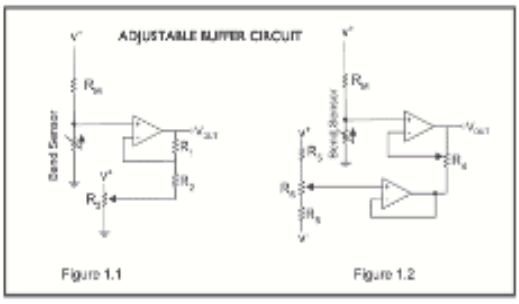


Following are notes from the ITP Flex Sensor Workshop

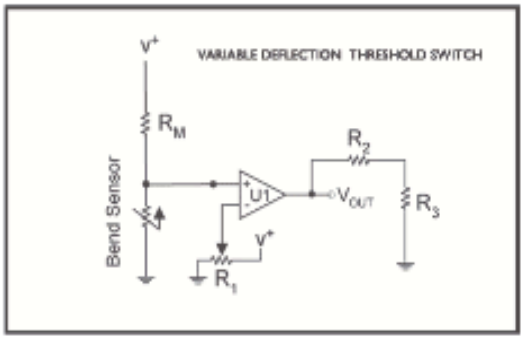
"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces errer due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324."

"You can also test your flex sensor using the simplest circut, and skip the op amp."

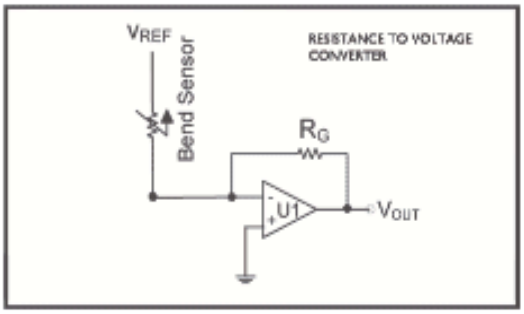
"Adjustable Buffer - a potentiometer can be added to the circuit to adjust the sensitivity range."



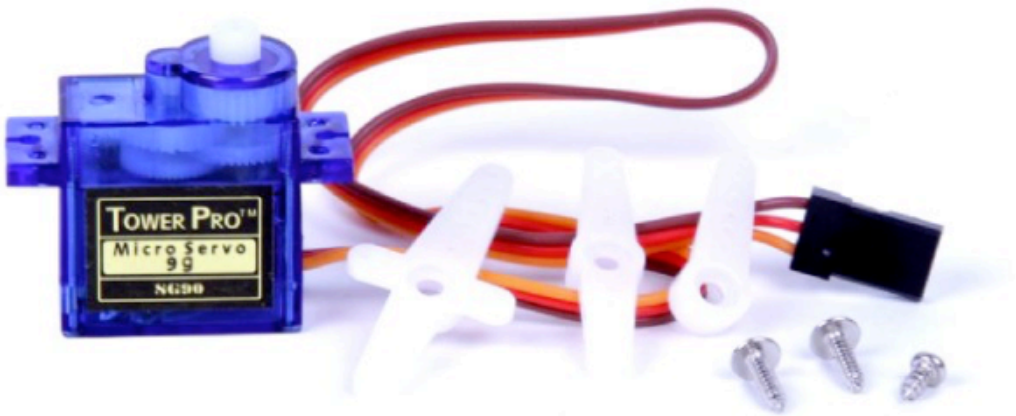
"Variable Deflection Threshold Switch - an op amp is used and outputs either high or low depending on the voltage of the inverting input. In this way you can use the flex sensor as a switch without going through a microcontroller."



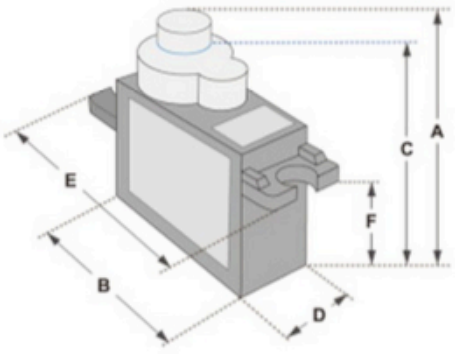
"Resistance to Voltage Converter - use the sensor as the input of a resistance to voltage converter using a dual sided supply op-amp. A negative reference voltage will give a positive output. Should be used in situations when you want output at a low degree of bending."



SERVO MOTOR SG90 DATA SHEET

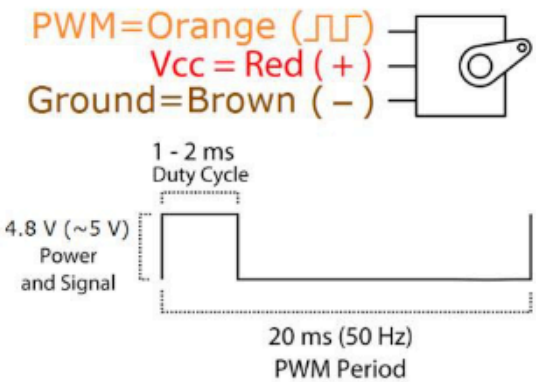


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

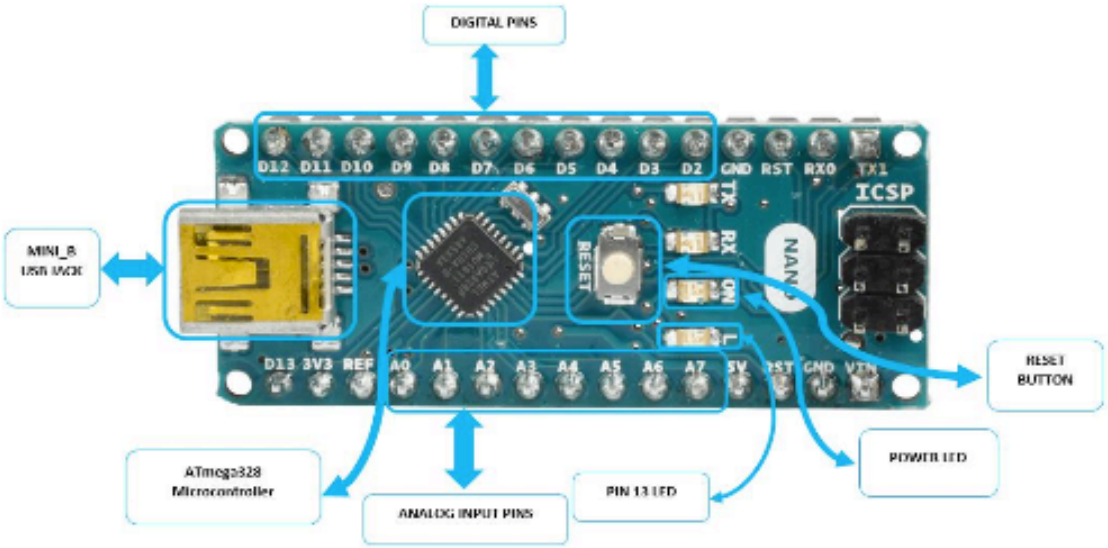


Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



ARDUINO NANO



INTRODUCTION

Arduino nano differ from other Arduino as it very small so it suitable for small sized projects and it supports breadboards so it can be plugged with other components in only one breadboard.

ARDUINO NANO PHYSICAL COMPONENTS

Microcontroller

In Arduino Nano 2.x version, still used ATmega168 microcontroller while the Arduino Nano 3.x version already used ATmega328 microcontroller.

1

ATmega168 Microcontroller

ATmega168 is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. And its features as follow:

- **High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family**
- **Advanced RISC Architecture**
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier

- **High Endurance Non-volatile Memory Segments**
  - 4K/8K/16KBytes of In-System Self-Programmable Flash Program Memory
  - 256/512/512Bytes EEPROM
  - 512/1K/1KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C/100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits

- **In-System Programming by On-chip Boot Program**
- **True Read-While-Write Operation**
  - Programming Lock for Software Security

- **Atmel® QTouch® Library Support**
  - Capacitive Touch Buttons, Sliders and Wheels
  - QTouch and QMatrix® Acquisition
  - Up to 64 sense channels

- **Peripheral Features**
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package

- **Temperature Measurement**
  - 6-channel 10-bit ADC in PDIP Package

- **Temperature Measurement**
  - Two Master/Slave SPI Serial Interface
  - One Programmable Serial USART
  - One Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - One On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

- **Special Microcontroller Features**
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

- **I/O and Packages**
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

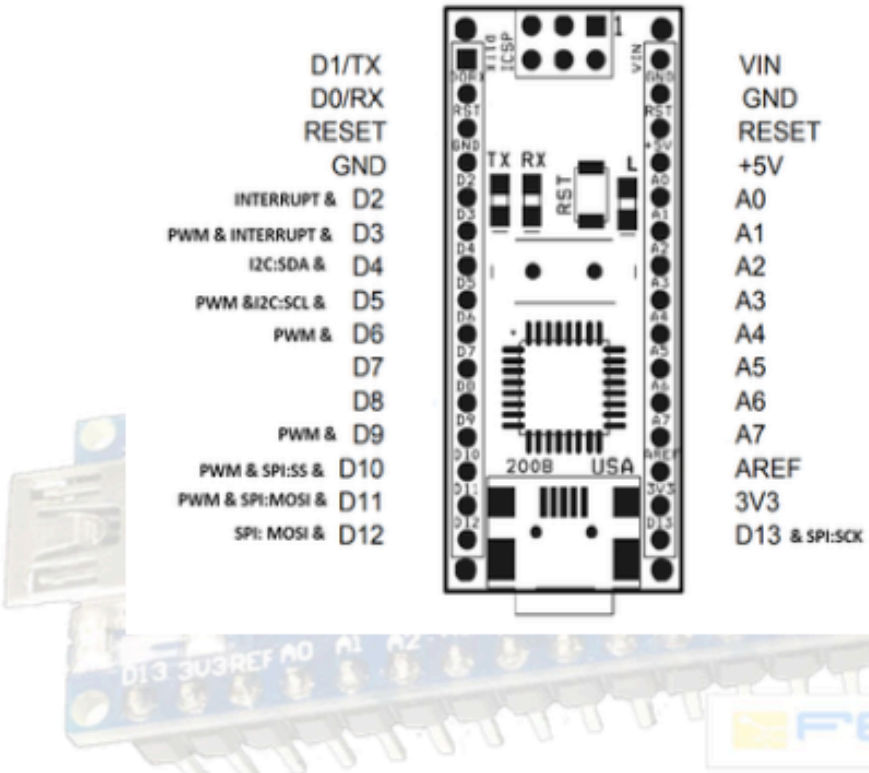
- **Operating Voltage:**
  - 2.7 - 5.5V for ATmega48/88/168
  - 1.8 - 5.5V for ATmega48V/88V/168V

- **Temperature Range:** -40°C to 85°C

- **Speed Grade:** 0 - 10MHz @ 2.7V - 5.5V, 0 - 20MHz @ 4.5V - 5.5V

- **Power Consumption at 1MHz, 1.8V, 25°C**
  - Active Mode: 0.3mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.8µA (Including 32kHz RTC)

ARDUINO NANO PIN CONFIGURATION



The Serial Peripheral Interface (SPI) IN PINS 7,8,13,14 AND 15

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.

With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically, there are three lines common to all the devices:

- MISO (Master In Slave Out) - The Slave line for sending data to the master,
- MOSI (Master Out Slave In) - The Master line for sending data to the peripherals,

8

- SCK (Serial Clock) - The clock pulses which synchronize data transmission generated by the master and one-line specific for every device:
- SS (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.

When a device's Slave Select pin is low, it communicates with the master. When it's high, it ignores the master. This allows you to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines.

Arduino Nano Specifications

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"
Length	45 mm
Width	18 mm
Weigth	5 g

9

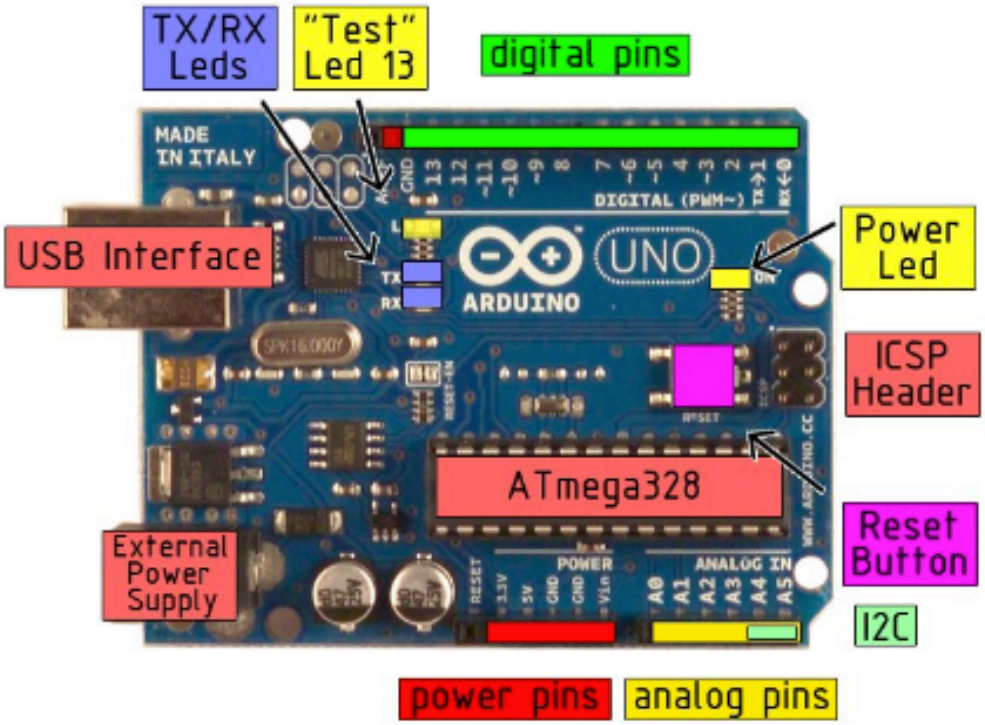
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

HM-10 DataSheet

Welcome to DSD ECH branded products!

If you have any questions, please contact us: Info@sihaicorp.com

1. Product Parameters

BT Version: Bluetooth Specification V4.0 BLE

Working frequency: 2.4GHz ISM band

Modulation method: GFSK(Gaussian Frequency Shift Keying)

RF Power: -23dbm, -6dbm, 0dbm, 6dbm

Speed: Asynchronous: 2-6K Bytes

Synchronous: 2-6K Bytes

Security: Authentication and encryption

Service: 0xFFE0 (Modifiable use AT+UUID command)

Characteristic: 0xFFE1 (Modifiable use AT+UUID command)

Characteristic: Notify and Write (Modifiable use AT+UUID command)

Power: +2.5V~3.3VDC 50mA

Power: Active state 8.5mA; Sleep state 50~200uA

Working temperature:–20 ~ +95 Centigrade

Size: HM-10 27mm x 13mm x 2.2 mm

Size: HM-11 18mm x 13mm x 2.2mm

Size: HM-15 65mm x 32mm x 16mm

2. Product overview

First of all, Thank you for choose our Bluetooth products.

HM Bluetooth modules use CSR or TI CC254x or cypress chips, Master and slave roles in one, data transmission version and remote control version and PIO status acquisition version in one, Support the AT commands modify module parameters, Convenient and flexible.

Data Transmission version:

Before connect:

You can configure module parameters with AT Commands through UART

After connect:

1) Send and receive Bluetooth data through UART.

Remote Control version:

Before connect:

You can configure module parameters with AT Commands through UART

After connect:

1) Send and receive Bluetooth data through UART.
2) Remote device could configure module parameters with AT Commands
3) Remote device could control PIO2~PIO11 output low or high

PIO acquisition version:

Before connect

You can configure module parameters with AT Commands

After connect

1) Send and receive Bluetooth data through UART.
2) Remote device could configure module parameters with AT Commands
3) Remote device could control PIO2, 3 output low or high with AT Commands
   HM-11 has no this function
4) Remote device could get PIO4 ~ 11 input status with AT Commands
   HM-11 only has PIO2 and PIO3.

Send	Receive	Parameter
AT	OK OK+LOST	None

If Module is not connected to remote device will receive: "OK"

If Module has connected, module will disconnected from remote device, if "AT + NOTI" is setup to 1, will receive: "OK+LOST"

2. Query module address

Send	Receive	Parameter
AT+ADC[P1]?	OK+GET:0.00	P1: 3,4,5,6,7,8,9,A,B map to PIO3~PIOB

HM-11 has no this function.

Add since V526.

3. Query module address

Send	Receive	Parameter
AT+ADDR?	OK+ADDR:MAC Address	None

3. Query/Set Advertising interval

Send	Receive	Parameter
AT+ADVI?	OK+ Get:[P1]	None
AT+ADVI[P1]	OK+ Set:[P1]	P1: 0 ~ F 0: 100ms 1: 152.5 ms 2: 211.25 ms 3: 318.75 ms 4: 417.5 ms 5: 546.25 ms 6: 760 ms 7: 852.5 ms 8: 1022.5 ms 9: 1285 ms

Send	Receive	Parameter
AT+BIT7?	OK+Get:[P1]	P1: bit7 switch.
AT+BIT7[P1]	OK+Set:[P1]	0-----Not compatible 1-----Compatible Default: 0

This command is used only for compatible uses 7 data bits, 2 stop bit device.

14. Query/Set baud rate

Send	Receive	Parameter
AT+BAUD?	OK+Get:[P1]	P1: Baud rate No.
AT+BAUD[P1]	OK+Set:[P1]	0---------9600 1---------19200 2---------38400 3---------57600 4---------115200 5---------4800 6---------2400 7---------1200 8---------230400 Default: 0(9600)

e.g.

Query baud:

Send: AT+BAUD?

Receive: OK+Get:0

Setup baud:

Send: AT+BAUD1

Receive: OK+Set:1

Note: If setup to Value 7, After next power on, module will not support any AT Commands, until PIO0 is pressed, Module will change Baud to 9600.

15. Query/Set Minimum Link Layer connection interval

		Default: 10

In mode 1, when PIO state is change, module will send OK+Col:[xx] to UART or remote side. This command is set send interval.

This command is added since V515 version.

26. Query/Set The switch of study function

Send	Receive	Parameter
AT+COMP?	OK+ Get:[P1]	P1: 0, 1, ? ?: Query; 0: Off; 1: On Default: 0
AT+COMP[P1]	OK+ Set:[P1]	

Please reference <How_To_Use_HM-1x_Study_function.pdf>

This command is added since V542 version.

27. Start a device discovery scan

Send	Receive	Parameter
AT+DISC?	OK+DIS[P0][P1]	P0: C,0, 1, 2 C: Common string 0~2: Address type P1: S, E, [MAC String] S: Start discovery E: End discovery MAC String : Device MAC string

Please set AT+ROLE1 and AT+IMME1 first.

e.g.

Send: AT+DISC?

Recv: OK+DISCS

Recv: OK+DIS[P0]:123456789012 (discovered device address information)

If AT+SHOW1 is setup, you will receive then Name information as follow

Recv: OK+NAME: xxx

After send Name value, will send two extra "\r\n" value ASCII byte

If the device not enable iBeacon function, P0, P1, P2 will use '0' fill.

Note: Added since V539

29. Connect to an Discovery device

Send	Receive	Parameter
AT+CONN[P1]	OK+CONN[P2]	P1: 0~5 P2: E, F, 0~5 E: Link error F: Link failed 0~5: Try to connect

This command is use after execute AT+DISC?

This command will clear all discovery data.

30. Query/Set iBeacon deploy mode

Send	Receive	Parameter
AT+DELO[P1]	OK+DELO[P1]	P1: 1, 2 1: Allowed to broadcast and scanning 2: Only allow broadcast

After receive OK+DELO[P1], module will reset after 500ms.

This command will let module into non-connectable status until next power on.

31. Remove bond information

Send	Receive	Parameter
AT+ERASE	OK+ERASE	

Note: Added in V524 version.

32. Set advertising data FLAG byte

Send	Receive	Parameter
AT+FLAG[P1]	OK+ Set:[P1]	P1: 0~FF (one byte)

Note: This command added in V530. Please ref to AT+BATT? Command.

33. Query/Set filter of HM modules

Send	Receive	Parameter

AT+IMME?	OK+ Get:[P1]	P1: 0, 1 1: When module is powered on, only respond the AT Command, don't do anything. Until AT + START, AT+CON, AT+CONNL commands is received. 0: When power on, module will start work immediately Default: 0
AT+IMME[P1]	OK+ Set:[P1]	

This command is only used for Central role.

39. Query/Set Module iBeacon switch

Send	Receive	Parameter
AT+IBEA?	OK+Get:[P1]	P1: 0, 1 0: Turn off iBeacon 1: Turn on iBeacon Default: 0
AT+IBEA[P1]	OK+Set:[P1]	

iBeacon UUID is: 74278BDA-B644-4520-8F0C-720EAF059935.

This command is added since V517 version.

40. Query/Set iBeacon UUID

Send	Receive	Parameter
AT+IBE0?	OK+Get:[P1]	P1: 00000001~ FFFFFFFE Default: 74278BDA
AT+IBE0[P1]	OK+Set:[P1]	

iBeacon UUID is: 74278BDA-B644-4520-8F0C-720EAF059935.

This command can change red color string in iBeacon UUID.

This command is added since V520 version.

e.g.: Send: AT+IBE012345678 change iBeacon UUID red color string to "12345678"

"12345678"

44. Query/Set Module iBeacon Marjor version

Send	Receive	Parameter
AT+MARJ?	OK+Get:[P1]	P1: 0x0001, 0xFFFE
AT+MARJ[P1]	OK+Set:[P1]	Default: 0xFFE0

E.g. Change marjor version to 0x0102

Send: AT+MARJ0x0102, if all is okay, module will send back OK+Set: 0x0102

This command is added since V517 version.

45. Query/Set Module iBeacon minor

Send	Receive	Parameter
AT+MINO?	OK+Get:[P1]	P1: 0x0001, 0xFFFE
AT+MINO[P1]	OK+Set:[P1]	Default: 0xFFE1

This command is added since V517 version.

46. Query/Set Module iBeacon Measured power

Send	Receive	Parameter
AT+MEAS?	OK+Get:[P1]	P1: 0x01~ 0xFF
AT+MEAS[P1]	OK+Set:[P1]	Default: 0xC5

This command is added since V519 version.

47. Query/Set Module Work Mode

Send	Receive	Parameter
AT+MODE?	OK+Get:[P1]	P1: 0, 1, 2
AT+MODE[P1]	OK+Set:[P1]	0: Transmission Mode 1: PIO collection Mode + Mode 0 2: Remote Control Mode + Mode 0 Default: 0

Mode 0:

Before establishing a connection, you can use the AT command configuration

module through UART.

After established a connection, you can send data to remote side from each other.

Mode 1:

Before establishing a connection, you can use the AT command configuration module through UART.

After established a connection, you can send data to remote side. Remote side can do fellows:

Send AT command configuration module.

Collect PIO04 to the PIO11 pins input state of HM-10.

Collect PIO03 pins input state of HM-11.

Remote control PIO2, PIO3 pins output state of HM-10.

Remote control PIO2 pin output state of HM-11.

Send data to module UART port (not include any AT command and per package must less than 20 bytes).

Mode 2:

Before establishing a connection, you can use the AT command configuration module through UART.

After established a connection, you can send data to remote side. Remote side can do fellows:

Send AT command configuration module.

Remote control PIO2 to PIO11 pins output state of HM-10.

Remote control PIO2, PIO3 pins output state of HM-11.

Send data to module UART port (not include any AT command and per package must less than 20 bytes).

48. Query/Set Notify information

Send	Receive	Parameter
AT+NOTI?	OK+Get:[P1]	P1: 0, 1
AT+NOTI[P1]	OK+Set:[P1]	0: Don't Notify

		1: Notify Default: 0

If this value is set to 1, when link ESTABLISHED or LOSTED module will send OK+CONN or OK+LOST string through UART.

49. Query/Set notify mode

Send	Receive	Parameter
Q: AT+NOTP?	OK+ Get[P1]	P1: 0, 1; default: 0 0: without address 1: with address
Q: AT+NOTP[P1]	OK+ Set[P1]	

This command must work with "AT+NOTI1", if this switch is open, when the module connect to disconnect, the prompt string will include the remote address.

OK+CONN:001122334455 String "001122334455" is the MAC address string

Added since V534

50. Query/Set Module name

Send	Receive	Parameter
AT+NAME?	OK+NAME[P1]	P1: module name, Max length is 12. Default: HMSoft
AT+NAME[P1]	OK+Set[P1]	

e.g.

change module name to bill_gates

Send: AT+NAMEbill_gates

Receive: OK+SetName:bill_gates

51. Query/Set output driver power

Send	Receive	Parameter
Query: AT+PCTL?	OK+Get:[P1]	None
Set: AT+PCTL[P1]	OK+Set:[P1]	P1: 0,1 0:Normal power output 1:Max power output Default: 1

59. Restore all setup value to factory setup

Send	Receive	Parameter
AT+RENEW	OK+RENEW	None

60. Restart module

Send	Receive	Parameter
AT+RESET	OK+RESET	None

61. Query/Set Master and Slaver Role

Send	Receive	Parameter
AT+ROLE?	OK+Get:[P1]	P1: 0, 1 0: Peripheral 1: Central Default: 0
AT+ROLE[P1]	OK+Set:[P1]	

62. Query RSSI Value

Send	Receive	Parameter
AT+RSSI?	OK+RSSI:[P1]	None

Require: AT+MODE value > 0

This command only used by Remote device query after connected.

63. Query Last Connected Device Address

Send	Receive	Parameter
AT+RADD?	OK+RADD:MAC Address	None

64. Query/Set Module Sensor work interval

Send	Receive	Parameter
AT+RAT??	OK+Get:[P1]	P1: 00~99 0:Save when connected 1:Don't Save Default: 0 Unit: minute
AT+RAT[P1]	OK+Set:[P1]	

Note: This command is only use for HMSensor

65. Query/Set Stop bit