Universidad Zaragoza
1542

Trabajo Fin de Grado

DISEÑO ELECTRÓNICO Y SOFTWARE DE CONTROL PARA BRAZO ROBOTICO DE 6 GRADOS DE LIBERTAD

ELECTRONIC DESIGN AND CONTROL SOFTWARE ROBOTIC ARM OF 6 DEGREES OF FREEDOM

Autor

José Javier Alonso Montesinos

Director

Javier Esteban Escaño

Escuela Universitaria Politécnica La Almunia

2017

Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

DISEÑO ELECTRÓNICO Y SOFTWARE DE CONTROL PARA BRAZO ROBÓTICO DE 6 GRADOS DE LIBERTAD

ELECTRONIC DESIGN AND CONTROL SOFTWARE ROBOTIC ARM OF 6 DEGREES OF FREEDOM

424.17.65

Autor: José Javier Alonso Montesinos

Director: Javier Esteban Escaño

Fecha: 27/06/2018

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE ECUACIONES

1. RESUMEN

El presente proyecto tiene su origen en el trabajo de fin de grado desarrollado el curso pasado. Su autor, realizó un proyecto que consistía en la construcción de un brazo robótico de 6 grados de libertad mediante tecnologías de impresión 3D.

Así pues, el objetivo de este trabajo fin de grado va a consistir en darle "vida" a este brazo robótico, diseñando el sistema electrónico y el software de control necesario para su funcionamiento, obteniendo así un brazo robótico didáctico completo.

La estructura que vamos a seguir para el desarrollo del proyecto va a ser la siguiente:

- Estudio teórico del prototipo diseñado en el proyecto anterior, y en el que se va a basar nuestro proyecto.
- Al estar finalizado ya el diseño mecánico en el proyecto anterior, nos centramos en el diseño electrónico y software de control, para ello realizaremos un estudio de los distintos métodos de control, y plantearemos y justificaremos el seleccionado.
- Una vez elegido el método de control, seleccionaremos el hardware electrónico y actuadores necesarios para su ejecución y la explicación de estos.
- Finalmente se realizará el desarrollo del software y la interconexión del sistema.

1.1. PALABRAS CLAVE

Brazo robótico, hardware, electrónico, software, control.

2. ABSTRACT

The present project has its origin in the end-of-degree work developed last year. Its author, realized a project that consisted of the construction of a robotic arm of 6 degrees of freedom by means of technologies of 3D impression.

Therefore, the aim of this final degree project will be to give "life" to this robotic arm, designing the electronic system and the control software necessary for its operation, obtaining a complete didactic robotic arm.

The structure that we will follow for the development of the project will be the following:

• Theoretical study of the prototype designed in the previous project, and on which our project will be based.

• We focus on the electronic design and control software, for this we will carry out a study of the different control methods, and we will raise and justify the selected one.

• Once the control method has been chosen, we will select the electronic hardware and actuators necessary for its execution and the explanation of these.

• Finally, software development and system interconnection will be carried out.

3. INTRODUCCIÓN

Actualmente nos encontramos en una sociedad en la que nuestra industria tiende cada vez más hacia la automatización, haciendo que esta mejore y aportando ventajas en diferentes campos:

Uno de estos campos es la economía, gracias a las mejoras en las cadenas de producción se consigue una reducción de costes de producción, reducción de personal, dependiendo del grado de automatización permite responder eficientemente ante cambios (ajustes internos en la cadena productiva, manejo eficiente de los procesos), en definitiva, el resultado de la automatización se ve reflejado en la economía de la empresa.

También se consigue aumentar la seguridad, aportando una mayor seguridad en las operaciones, puesto que la intervención del hombre es menor. Al reducir la intervención del operario se aumenta la comodidad de su trabajo (evitando así las operaciones que producen cansancio del operario pudiendo afectar también a la calidad final del producto).

En la actualidad la información en las empresas es de gran importancia ya que aumenta el conocimiento del proceso pudiendo realizar mejoras sobre este, basadas en los datos obtenidos gracias a la automatización (analizando datos, generando modelos etc.)

Por todo esto la implantación de robots industriales ha crecido considerablemente desde su llegada a la industria, ya que son una herramienta cada vez más importante.

Actualmente encontramos diversas definiciones a la hora de establecer lo que es un robot industrial, incluso con el tiempo se ha tenido que ir modificando la definición debido a los avances que ha ido sufriendo este apartado, a pesar de ello más formalmente se encuentra el estándar ISO según el cual: (ISO 8373:1994,I. Robots industriales manipuladores-Vocabulario) define un robot industrial como un manipulador programable en tres o más ejes multipropósito, controlado automáticamente y reprogramable.

Son muchas las aplicaciones de un robot industrial, y no siempre es ventajoso la introducción de este, por lo que hay que examinar las ventajas e inconvenientes que ocasionaría incorporar un robot industrial en el citado proceso. Algunas de estas aplicaciones pueden ser: Trabajos en fundición, soldadura, aplicación de materiales, procesado, corte, montaje etc.

Incluyendo nuevos sectores de aplicación en auge, como puede ser la agricultura, construcción, domésticos, entornos peligrosos, medicina y salud. Por todo ello y a la vista de que es un campo en constante mejora no es de extrañar que sea necesario el estudio de este tipo de robots y todo lo que ello conlleva. Es aquí donde encontramos la motivación y el sentido de este trabajo fin de grado, cuyo objetivo es precisamente este:

Realizar el sistema de control y diseño electrónico de un brazo robótico de bajo coste para un uso docente, permitiendo así al estudiante una mayor capacidad de aprendizaje y motivación de poder aplicar lo aprendido de una forma más práctica.

3.1. ESTADO DEL ARTE

Una vez obtenidos los modelos cinemáticos o dinámicos del robot, se puede abordar el problema del control de los mismos. Definir el movimiento del robot implica controlar dicho robot de manera que siga un camino planificado. El objetivo es, por tanto, establecer cuáles son las trayectorias que debe seguir cada articulación a lo largo del tiempo para conseguir los objetivos fijados, cumpliendo con una serie de restricciones físicas impuestas por los actuadores y de calidad de la trayectoria, suavidad, precisión, etc.

Por lo tanto, las funciones del control cinemático son:

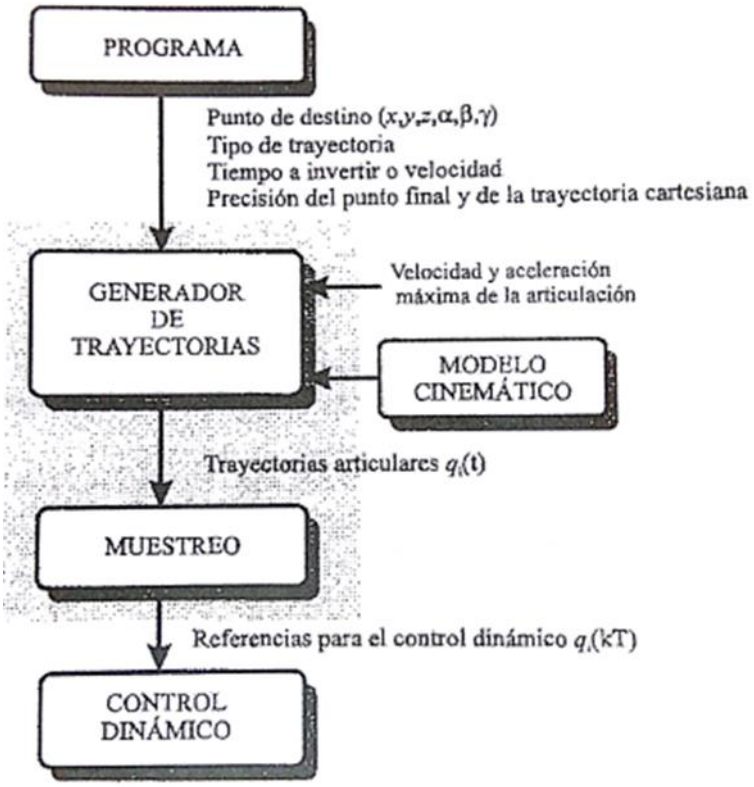


Ilustración 1 Funciones del control cinemático

1. Convertir la especificación del movimiento dada en el programa en una trayectoria analítica en el espacio cartesiano.
2. Muestrear la trayectoria cartesiana obteniendo un número finito de puntos de dicha trayectoria. Cada uno de estos puntos vendrá dado por una 6-tupla, típicamente (x,y,z,α,β,γ).
3. Utilizando la transformación homogénea inversa, convertir cada uno de estos puntos en sus correspondientes coordenadas articulares (q_1,q_2,q_3,q_4,q_5,q_6).
4. Interpolación de los puntos articulares obtenidos, generando para cada variable articular una expresión $q_i(t)$ que pase o se aproxime a ellos de modo que, siendo una trayectoria cartesiana lo más próxima a la especificada por el programa del usuario (en cuanto a precisión, velocidad, etc)
5. Muestreo de la trayectoria articular para generar referencias al control dinámico.

3.1.1. Tipos de trayectorias

Para poder realizar una determinada tarea el robot debe moverse desde un punto inicial a un punto final. Este movimiento es lo que vamos a implementar, y este puede ser realizado según infinitas trayectorias espaciales, de todas ellas hay algunas que, bien por su sencillez de implementación por parte del control cinemático o bien por su utilidad y aplicación a diversas tareas, son las que en la práctica incorporan los robots comerciales. De este modo puede encontrarse que los robots disponen de trayectorias punto a punto, coordinadas y continuas.

3.1.1.1. Trayectorias punto a punto

En este tipo de trayectorias cada articulación evoluciona desde su posición inicial a la final sin realizar consideración alguna sobre el estado o evolución de las demás articulaciones. Normalmente, cada actuador trata de llevar a su articulación al punto de destino en el menos tiempo posible, pudiéndose entonces distinguir dos casos: movimiento eje a eje y movimiento simultaneo de ejes.

- Movimiento eje a eje: solo se mueve un eje cada vez, comenzará a moverse la primera articulación, y una vez que esta haya alcanzado su punto final lo hará la segunda, y así sucesivamente. Este tipo de movimiento da obviamente como resultando un mayor tiempo de ciclo, obteniéndose como única ventaja un menor consumo de potencia instantánea por parte de los actuadores.

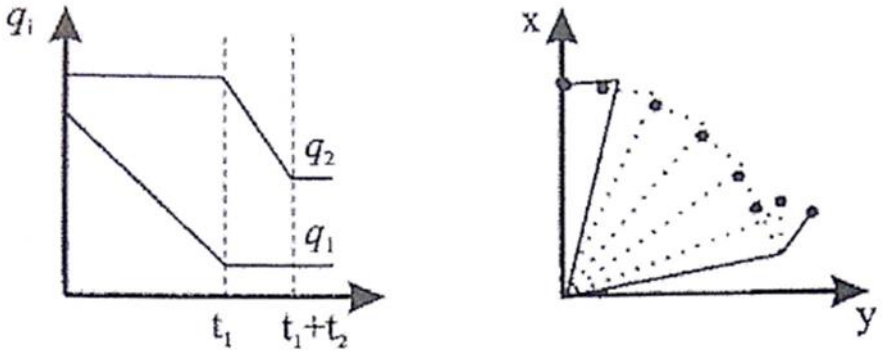


Ilustración 2 Movimiento eje a eje

- Movimiento simultaneo de ejes: en este caso todos los actuadores comienzan simultáneamente a mover las articulaciones del robot a una velocidad específica para cada una de ellas. Dado que la distancia a recorrer y las velocidades suelen ser diferentes cada una acaba su movimiento en un instante diferente. Por lo tanto, el tiempo total del movimiento coincidirá con el del eje que más tiempo emplee en realizar su movimiento.

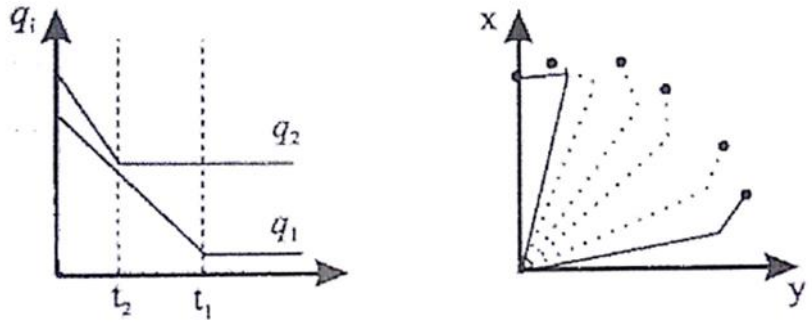


Ilustración 3 Movimiento simultaneo de ejes

Por estos motivos expuestos, las trayectorias punto a punto solo suelen estar implementadas en robots muy simples o con unidades de control muy limitadas.

3.1.1.2. Trayectorias coordinas o isócronas

Para evitar que algunos actuadores trabajen de forma que fuercen sus velocidades y aceleraciones, teniendo que esperar a que finalice el movimiento de la articulación más lenta, puede hacerse un cálculo previo, averiguando cual es esta articulación y que tiempo invertirá. Se ralentizará entonces el movimiento del resto de los ejes para hacer que los tiempos de movimiento coincidan y utilicen todos los mismos tiempos, acabando simultáneamente. De esta forma el tiempo total invertido es el menor posible y no se piden aceleraciones y velocidades elevadas a los actuadores de manera inútil.

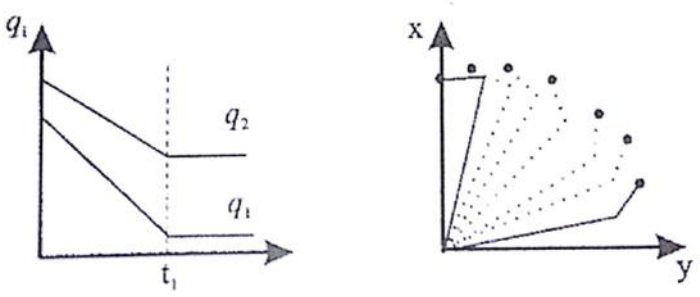


Ilustración 4 Trayectoria coordinada

3.1.1.3. Trayectorias continuas

Cuando se pretende que la trayectoria que sigue el extremo del robot sea conocida por el usuario (trayectoria en el espacio cartesiano o de la tarea), es preciso calcular de manera continua las trayectorias articulares. Típicamente, las trayectorias que el usuario pretende que el robot describa son trayectorias en línea recta o en arco de círculo. El resultado será que articulación sigue un movimiento aparentemente caótico con posibles cambios de dirección y velocidad y sin coordinación con el resto de las articulaciones. Sin embargo, el resultado conjunto será que el extremo del robot describirá la trayectoria deseada.

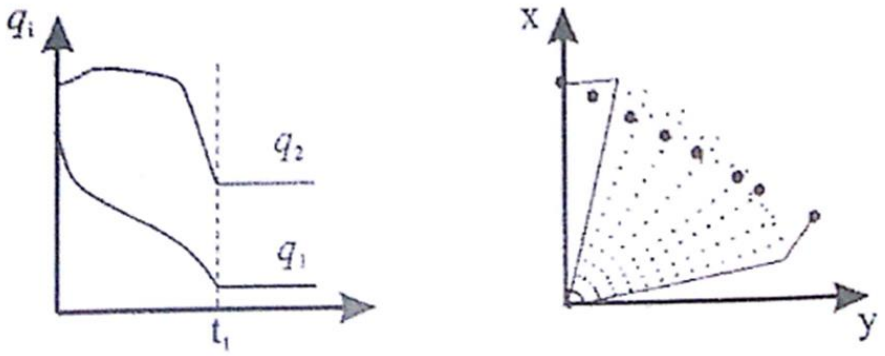


Ilustración 5 Trayectoria continua rectilínea

3.1.2. Interpolación de trayectorias

Como se ha indicado, una de las funciones del control cinemático es la de unir una sucesión de puntos en el espacio articular por los que se quiere que pasen las articulaciones del robot en un instante determinado. Además, junto con las condiciones de contorno: posición-tiempo, es conveniente añadir restricciones en la velocidad y aceleración de paso por los puntos, de manera que se asegure la suavidad de la trayectoria y se limiten las velocidades y aceleraciones máximas.

Se presentan a continuación las funciones interpoladoras utilizadas con mayor frecuencia. Cada una de ellas desarrollada para un solo grado de libertad, debiendo repetir el mismo cálculo para cada uno de los grados de libertad del robot.

3.1.2.1. Interpoladores lineales

$$q(t) = a * t + b$$

Condiciones:

$$q(t^{i-1}) = a * t^{i-1} + b = q^{i-1}$$

$$q(t^{i}) = a * t^{i} + b = q^{i}$$

Despejando a y b obtenemos

$$q(t) = (q^i - q^{i-1})\frac{t - t^{i-1}}{T} + q^{i-1}$$

$$T = t^i - t^{i-1}$$

Ecuación 1 Interpolador lineal

Esta selección de paso por t^i por los puntos q^i podrá haberse hecho según los diferentes criterios teniendo en cuenta los tipos de trayectoria expuestos.

La velocidad cambia bruscamente y la aceleración es infinita, por lo que en la práctica no es posible.

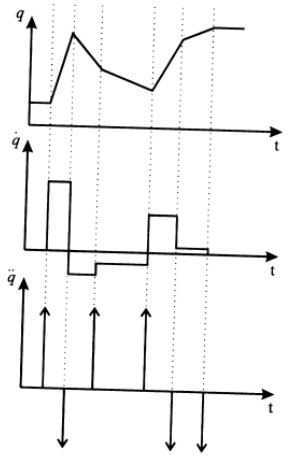


Ilustración 6 Interpoladores lineales

3.1.2.2. Interpoladores cúbicos

Para asegurar que la trayectoria que une los puntos por los que tiene que pasar la articulación considerada presente continuidad en velocidad, puede recurrirse a utilizar un polinomio de grado 3 que una cada pareja de puntos adyacente.

Fórmula de interpolación:

$$q(t) = a + b(t - t^{i-1}) + c(t - t^{i-1})^2 + d(t - t^{i-1})^3$$

$$a = q^{i-1}$$

$$b = \dot{q}^{i-1}$$

$$c = \frac{3}{T^2}(q^i - q^{i-1}) - \frac{2}{T^2}\dot{q}^{i-1} - \frac{1}{T^2}\dot{q}^i$$

$$d = -\frac{2}{T^3}(q^i - q^{i-1}) + \frac{1}{T^2}(\dot{q}^{i-1} + \dot{q}^i)$$

$$T = t^i - t^{i-1}$$

Ecuación 2 Interpolador cúbico

3.1.2.3. Interpoladores a tramos

En los interpoladores anteriores se utiliza un polinomio de un grado determinado para unir dos puntos consecutivos de la trayectoria. Una alternativa que nos permite una solución intermedia consiste en descomponer en tres tramos consecutivos la trayectoria que une dos puntos. Utilizando así en el tramo central un interpolador lineal, manteniendo la velocidad constante y en los tramos inicial y final utilizar un polinomio de segundo grado, de modo que en el tramo 1 la velocidad varíe linealmente desde la velocidad de la trayectoria anterior a la de la presente, y en el tramo 3 varíe desde la velocidad de la trayectoria presente hasta la siguiente. Así pues, en los tramos inicial y final la aceleración toma valores constantes distintos de cero, mientras que en el tramo intermedio la aceleración es nula.

Dentro de la interpolación a tramos encontramos el denominado ajuste parabólico, y es que en el caso de tener una trayectoria formada por varios puntos, la velocidad de paso no debería ser nula, puesto que daría lugar a un movimiento discontinuo del robot. Esto se puede evitar haciendo que la trayectoria no pase exactamente por los puntos sino tan cerca de él como lo permita la aceleración máxima.

Siguiendo esta técnica la ecuación de los tres tramos que componen la trayectoria que une dos puntos consecutivos sería:

$$q(t) = \begin{cases} q^0 + \frac{q^1 - q^0}{T_1} & 0 \le t \le T_1 - \tau \\ q^1 + \frac{q^1 - q^0}{T_1}(T - T_1) + \frac{a}{2}(t - T_1 + \tau)^2 & T_1 - \tau < t < T_1 + \tau \\ q^1 + \frac{q^2 - q^1}{T_2}(t - T_1) & T_1 + \tau \le t \le T_1 + T_2 \end{cases}$$

Ecuación 3 Interpolador a tramos

Siendo a la aceleración constante que permite cambiar la velocidad de un tramo al siguiente siendo su valor:

$$a = \frac{T_1(q^2 - q^1) - T_2(q^1 - q^0)}{2T_1T_2\tau}$$

3.2. OBJETIVOS

Realizar el sistema de control y diseño electrónico de un brazo robótico de bajo coste para un uso docente, que sirva así al estudiante para obtener una mayor capacidad de aprendizaje y motivación de poder aplicar lo aprendido de una forma más práctica.

Este es el objetivo principal, pero para lograr eso hay que exponer con detalle las diferentes partes que abarca el proyecto:

- **Montaje y diseño electrónico**

  Aprovechándonos del diseño mecánico del TFG previo, lo primero para poder cumplir nuestro objetivo, es realizar el diseño electrónico completo, lo que incluye el acople de los servos al diseño mecánico, como alimentar estos, cablearlos sin que el brazo vea afectado su diseño mecánico, y por último conectarlos al microcontrolador para que puedan recibir las ordenes.

- **Comunicación**

  El usuario podrá mandar órdenes de movimiento al robot ya sea introduciendo las ordenes a través del teclado del ordenador, o mediante algún otro dispositivo que se comunicará con el microcontrolador y será el encargado de mover los servomotores hasta la posición indicada.

4. DESARROLLO

4.1. DISEÑO ELECTRÓNICO Y CONTROL

4.1.1. Servomotores

Para dar movimiento al brazo robótico vamos a emplear distintos servomotores, en este caso necesitaremos 7 para conseguir los distintos grados de libertad, como al comienzo del proyecto íbamos a continuar con el prototipo empezado durante el año anterior al comienzo de este, algunos servomotores de los elegidos coinciden con los seleccionados en el anterior TFG, pero otros ha sido necesario modificarlos y elegir otro modelo, para coincidir con el nuevo diseño.

Para la base utilizamos el servomotor TowerPro MG995, de forma que lo aprovecharemos del proyecto anterior ya que vamos a aprovechar la base y esto nos favorecerá el montaje del mismo al ser las dimensiones de este servomotor las que emplea el autor del TFG anterior

Características:

Material engranajes: Metal

Peso: 55 gr

Torque: 9.4kg/cm (4.8v); 11kg/cm (6v)

Ilustración 7 Tower Pro MG 995

(«Tower Pro MG995 Metal Gear Servo», s. f.)

Para el hombro empleamos el K-Power DM4000 debido a su elevado par ya que este va a ser el encargado de soportar todo el peso del brazo, este servomotor había sido elegido en el anterior TFG y aprovechamos para emplear el mismo modelo.

Características:

Material Engranajes: Metal

Peso: 146 gr

Torque:42kg-cm/6.0V;50kg cm/ 7.4V; 54kg-cm/8.4V

Ilustración 8: K-Power DM4000

(«K-Power DM4000 Servo 54kg-cm/0,11 s/6 Volt till 8,4 volt - rc-car-online Onlineshop Hobbythek», s. f.)

Para el codo empleamos un nuevo servomotor que no estaba incluido en el TFG anterior, se trata del servo futaba S3003

Características

Material engranajes: Nylon

Voltaje de operación 4.8-6 Volts.

Peso: 38 gr.

Torque

3.2 kg/cm (4.8 volts)

4.1 kg/cm (6 Volts)

Ilustración 9 Futaba S3003

(«13-Servomotor FUTABA [S3003].pdf», s. f.)

Para los dos giros de la muñeca y el giro de la pinza utilizamos el mismo servo aprovechando así las mismas dimensiones de acople del servo para todas las piezas, evitándonos así el tener que cambiar dimensiones, el servo va a ser el Tower Pro MG90S.

Características:

Material engranajes: Metal

Peso: 13,4 g

Torque: 1,8 kg/cm (4,8 V)

2,2 kg/cm (6 V)

Ilustración 10 Tower pro MG90S

(«ds-servo-mg90s.pdf», s. f.)

Por último, para realizar el agarra de la pinza empleamos el servomotor indicado por el creador del diseño subido a la página web opensource thingiverse, e indica utilizar el servo HITEC HS-55.

Características:

Material engranajes: Nylon

Peso: 8 g

Torque: 1,1 Kg/cm (4,8 V)

1,3 Kg/cm (6 V)

Ilustración 11 Hitec HS-55

(«hs55.pdf», s. f.)

4.1.2. Arduino

En este caso, se opta por la plataforma Arduino, esta es una plataforma de código abierto (open source) basado en una placa con un sencillo microcontrolador y un entorno de desarrollo que permite la creación de programas para luego ser cargados en la memoria del mismo.

Dentro de la plataforma Arduino tenemos muchos modelos de microcontroladores, pero para la realización de este brazo robótico el seleccionado es el Arduino Mega 2560 R3. La elección de este se debe principalmente a que dispone de más entradas/salidas que el modelo más famoso de Arduino, el Arduino UNO.

Introducción general a la placa Arduino Mega

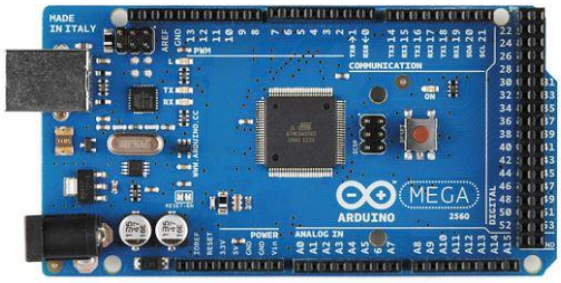


Ilustración 12 Arduino Mega

Características técnicas Arduino Mega:

- Tensión de alimentación: 5V
- Tensión de entrada recomendada: 7-12V

- Límite de entrada: 6-20V
- Pines digitales: 54 (14 con PWM)
- Entradas analógicas: 16
- Corriente máxima por pin: 40 mA
- Corriente máxima para el pin 3.3V: 50 mA
- Memoria flash: 256 KB
- SRAM: 8 KB
- EEPROM: 4 KB
- Velocidad de reloj: 16 MHz

Una razón muy importante por la cual se selecciona Arduino, es debido a estas entradas PWM que son capaces de generar y gestionar automáticamente esta señal PWM en sus pines, y así poder gobernar los servos del brazo sin demasiada complicación.

Cada uno de los 54 pines digitales en el Mega pueden utilizarse como entradas o como salidas usando las funciones pinMode(), digitalWrite(), y digitalRead(). Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad maxima de 40mA y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50kOhms. Además, algunos pines tienen funciones especializadas:

- Serie: 0 (RX) y 1 (TX),Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Usados para recibir (RX) transmitir (TX) datos a través de puerto serie TTL.
- Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2). Estos pines se pueden configurar para lanzar una interrupción en un valor LOW(0V), en flancos de subida o bajada (cambio de LOW a HIGH(5V) o viceversa), o en cambios de valor. Ver la función attachInterrupt() para más detalles.
- PWM: de 0 a 13.Proporciona una salida PWM (Pulse Width Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a través de la función analogWrite().

(«Arduino Mega 2560 R3 ~ Arduino.cl», s. f.)

4.1.3. Módulo bluetooth HC-06

Se trata de un módulo que utiliza el protocolo UART RS 232 serial, es perfecto para esta aplicación inalámbrica en la que queremos emplearlo como esclavo para transmitir las instrucciones a nuestro módulo Arduino mediante un dispositivo móvil.

Ilustración 13 Módulo bluetooth HC-06

(«Módulo Bluetooth HC06 HC-06 » IBEROBOTICS», s. f.)

Las características de este módulo bluetooth son las siguientes:

Bluetooth V2.1 + EDR

Bluetooth Clase 2

Antena de radiofrecuencia PCB incorporada

Admite la interfaz UART

Potencia de 3.3V

(«Módulo BlueTooth HC-06 | Tienda y Tutoriales Arduino», s. f.)

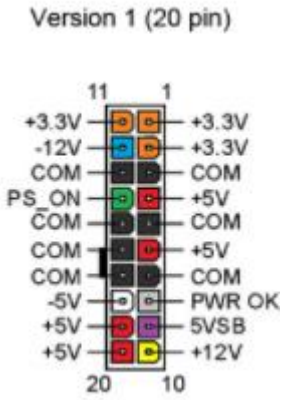
4.1.4. Alimentación

Para la alimentación del brazo robótico distinguiremos dos subsistemas, Arduino y el sistema de movimiento cuya alimentación proviene del USB del pc.

Y la parte de servomotores que irán alimentados mediante una fuente de alimentación externa. Para la cual empleamos una fuente de alimentación de un ordenador antiguo, reutilizándola y convirtiéndola en una fuente de alimentación para nuestro proyecto, en este caso la fuente que reutilizamos se trata de la fuente ATX 2.03 Model: LPK2-30P4, ya que como sabemos presentan una gran potencia y salidas para voltajes habituales en la electrónica que vamos a usar (3.3V, 5 V Y 12 V). Para poder emplear esta fuente de alimentación como fuente externa para nuestro proyecto necesitamos realizarle una serie de modificaciones, las cuales explicamos a continuación.

Ilustración 14 Fuente ATX

Lo primero que vemos es gran número de cables que salen de la fuente, el conector más grande es el conector ATX, que es el que en ordenadores se conecta y alimenta a la placa base. Este conector antiguamente tenía 20 pines y posteriormente paso a 24 en el caso de mi fuente todavía es de 20 pines.



Como vemos cada cable se encuentra codificado por colores, de forma que cables con el mismo color tendrán la misma función. Elegimos los cables que nos interesan de todos los que están disponibles para el resto cortarlos.

En mi caso vamos a emplear:

- Conductor de encendido: Verde (PS_ON)
- Salidas de voltaje: Amarillo (+3V), Rojo(+5V) y amarillo (+12V).
- Tierra: Negro (GROUND)

Además del conector ATX, la fuente dispone de otros muchos conectores. Podemos aprovechar estos cables para conectar clavijas o tomas rápidas para otros dispositivos, pero en este caso no va a ser necesario y vamos a recortar todos los sobrantes.

Lo ideal sería emplear conectores de banana y agujerear la carcasa para realizar una fuente de alimentación completa y que el acabado fuera mucho mejor, pero para realizar las pruebas de nuestros servos vamos a dejar los cables que necesitamos sueltos, y usaremos la placa de conexiones y los empalmes necesarios entre cables para realizar la alimentación, a posteriori si funcionara correctamente se podría realizar esta última modificación.

4.1.5. Conexionado

Para realizar el conexionado entre la fuente de alimentación servomotores y Arduino emplearemos una protoboard de marca Ariston, y cables de la marca Elegoo que son los comúnmente empleados para trabajar con Arduino.

Ilustración 15 Cables Elegoo

(«ELEGOO 120pcs Multicolored Dupont Wire – Elegoo Industries,Ingenious & fun DIY electronics and kits», s. f.)

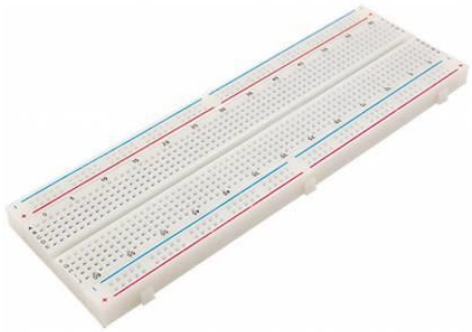


Ilustración 16 Protoboard Ariston

(«Electrònica Analògica 7: Pràctiques amb "Protoboard" – tecnoclasse4t», s. f.)

Una vez seleccionado el módulo de Arduino a utilizar, la fuente y los servos, hay que diseñar la forma en que se va a conectar la placa con todos los servos y señales de posición que deben ser gobernadas por el módulo.

El brazo se compone de un total de 7 servos, cuya señal PWM se generará desde un pin digital y una señal de posición por cada servo, es decir hay 7 señales de posición que deben ser leídas por el módulo Arduino, que recibirá las señales a través de un módulo bluetooth.

4.2. DISEÑO MECÁNICO

El diseño del brazo robótico forma parte del TFG de Cristian Doñate Lahuerta, en este TFG se utiliza dicho diseño con la intención de dar continuidad a ese proyecto realizando el control del brazo.

Todas las piezas se encuentran ya impresas y realizadas con plástico PLA (Ácido poliláctico)

A continuación, muestro una fotografía del brazo completo.

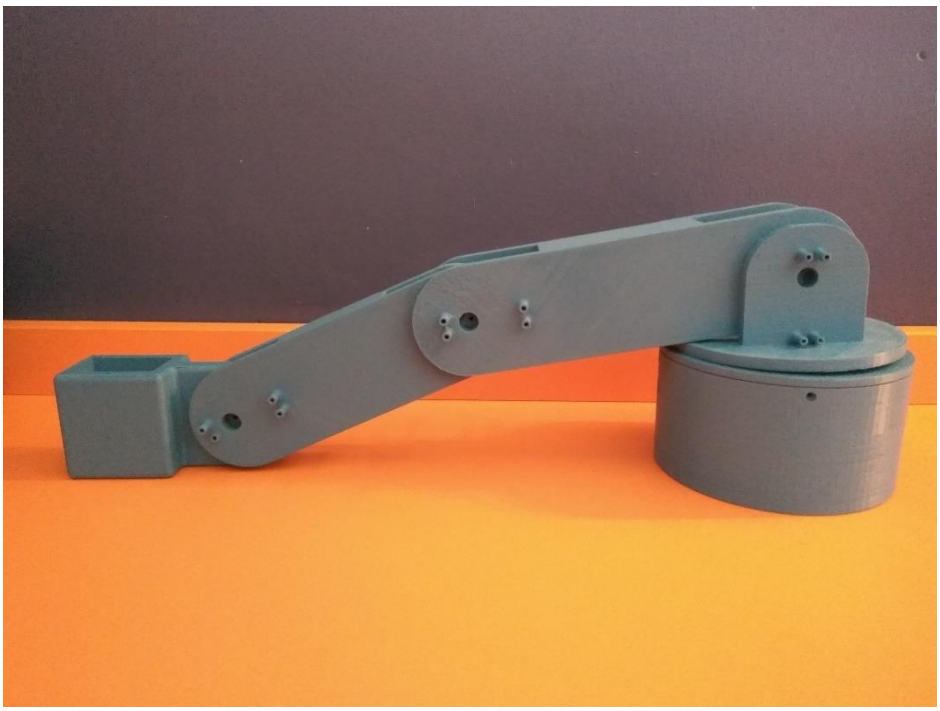


Ilustración 17 Brazo Robótico Completo

4.3. ENSAMBLAJE ELECTRÓNICO Y MECÁNICO

Para el ensamblaje electrónico y mecánico de nuestro brazo, realizamos dos pequeños programas en Arduino únicamente para comprobar los movimientos de los servos sin ser este el programa final que emplearemos para así poder comprobar que funciona correctamente tras el acople del servo al brazo. El primero de ellos refleja en estas líneas de unas de código que hacen que el servo se mueva grado a grado con la ayuda de un potenciómetro, así sabemos en qué grado se encuentra cada momento el servo, este programa no solo nos servirá para la comprobación, sino para a posteriori poder realizar el ensamblaje final del brazo con los servos ya colocados exactamente en la posición que queremos que coincida con un número determinado de grados

```
#include <Servo.h>

Servo Prueba_Servo;  // Creamos la variable servo.

int Pin_Potenciometro = 0;  // El pin de analógico que empleamos para el potenciómetro
int Valor_Potenciometro;    // El valor del potenciometro que vamos a leer convertido a grados en pantalla.

void setup() {
  Serial.begin(9600);
  Prueba_Servo.attach(2);  // Asignamos el servomotor al pin 2.
}

void loop() {
  Valor_Potenciometro = analogRead(Pin_Potenciometro);            // Leemos el valor del potenciometro ( Valor entre 0 y 1023)
  Valor_Potenciometro = map(Valor_Potenciometro, 0, 1024, 0, 180);     // Escalamos ese valor para poder usarlo con el servo(Valor entre 0 y 180)
  Prueba_Servo.write(Valor_Potenciometro);  // Introducimos ese valor del potenciometro al servo
  Serial.println(Valor_Potenciometro);  //Reflejamos por pantalla el valor en angulos
  delay(30);                           // Pequeño delay hasta que el servo llegue a la posicion
}
```

Ilustración 18 Programación prueba potenciómetro

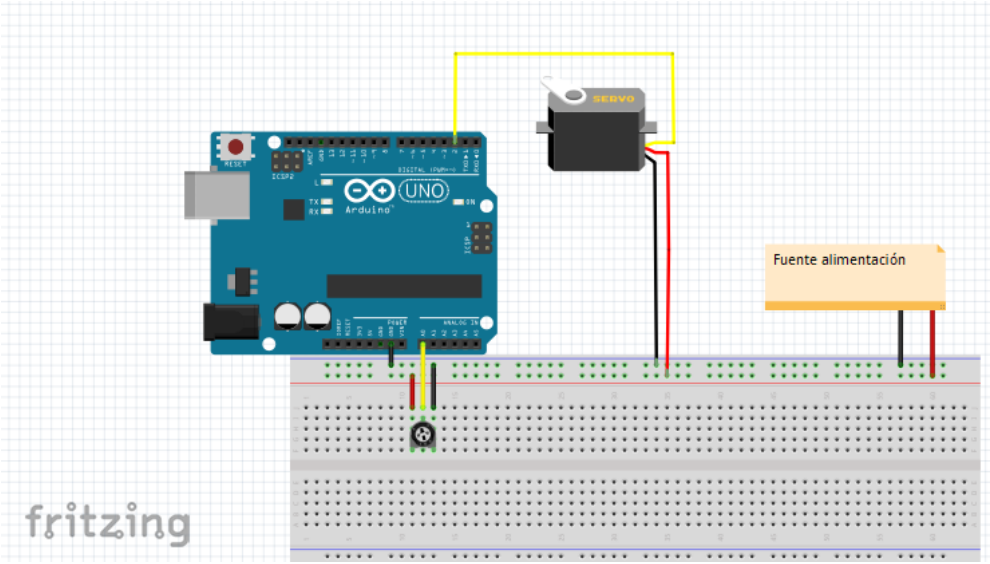


Ilustración 19 Conexionado prueba potenciómetro

El segundo código de programación que comentaba es simplemente una programación en la que nombrando previamente las secuencias que queremos que haga el servo. Es decir se le introducen los grados que queremos que mueva dentro de una determinada secuencia a la cual le asignamos un nombre, es simplemente otra forma de comprobar y una posible solución para el diseño final de la programación.

```
void ControlServo(int angulomin, int angulomax)
 {
  for(int angulo = angulomax; angulo > angulomin; angulo --)//150 5
  {   //  un ciclo para mover el servo entre los 0 y los 180 grados
        ServoMuneca.write(angulo);              //  manda al servo la posicion
        delay(30);                        //  espera unos milisegundos para que el servo llegue a su posicion
  }
  delay(1000);
 }
```

Ilustración 20 Programación movimiento autónomo

4.3.1. Base

Para el ensamblaje del subconjunto de la base, vamos a necesitar el servo TowerPro MG995, 4 tornillos de M3X30 y sus correspondientes tuercas M3 para el anclaje del servo a la base.

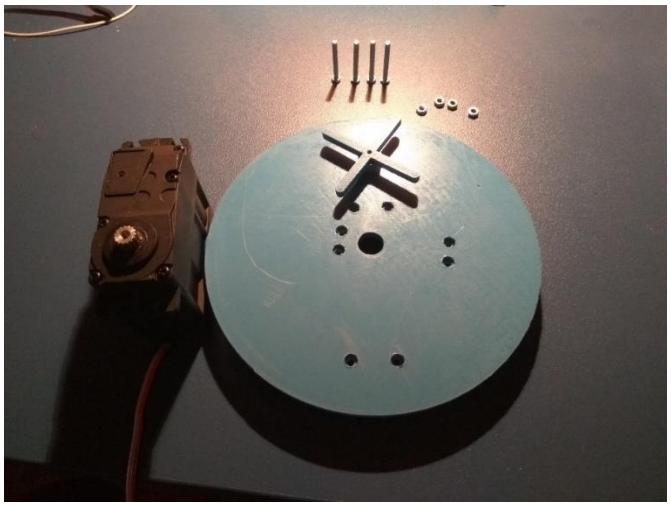


Ilustración 21 Tapa base

El soporte de la base se mantiene igual, como antes necesitaremos 4 tornillos, pero en este caso serán de M3X25 para unir la tapa de la base con el soporte de la base.

Ilustración 22 Soporte base

Por ultimo y donde encontramos el mayor problema de todos es en el Cover base, ya que al tratarse de un prototipo tiene el soporte realizado como sucede con la tapa de la base para el servo de menor tamaño, y en este caso a diferencia que con la tapa de la base la solución no es tan sencilla como hacer agujeros nuevos, la única solución viable es el rediseño de la pieza y reimprimirla con el tamaño adecuado para el servo.

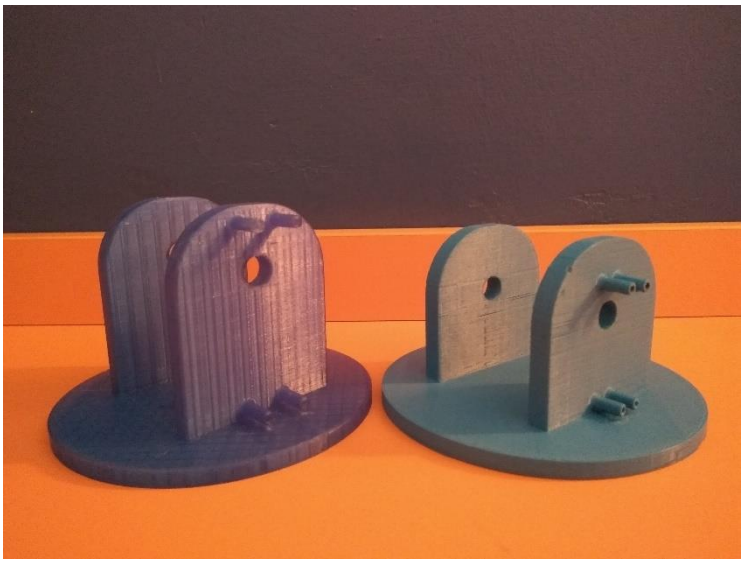


Ilustración 23 Rediseño Cover Base

Realizamos la reimpresión de la pieza, como se aprecia en la foto, el material es el mismo, utilizamos PLA, pero con un color y acabado distinto, más traslucido que el inicial.

Ilustración 24 Ensamblaje Final Base

Con esto habríamos terminado con el ensamblaje mecánico para la base, ahora procedemos con el ensamblaje electrónico.

Para el ensamblaje electrónico en esta primera parte al ser un único servomotor podríamos usar Arduino como fuente de alimentación y no tendríamos problema, pero como más adelante vamos a usar más servos utilizamos ya la fuente de alimentación. El esquema de conexión completo sería el siguiente:

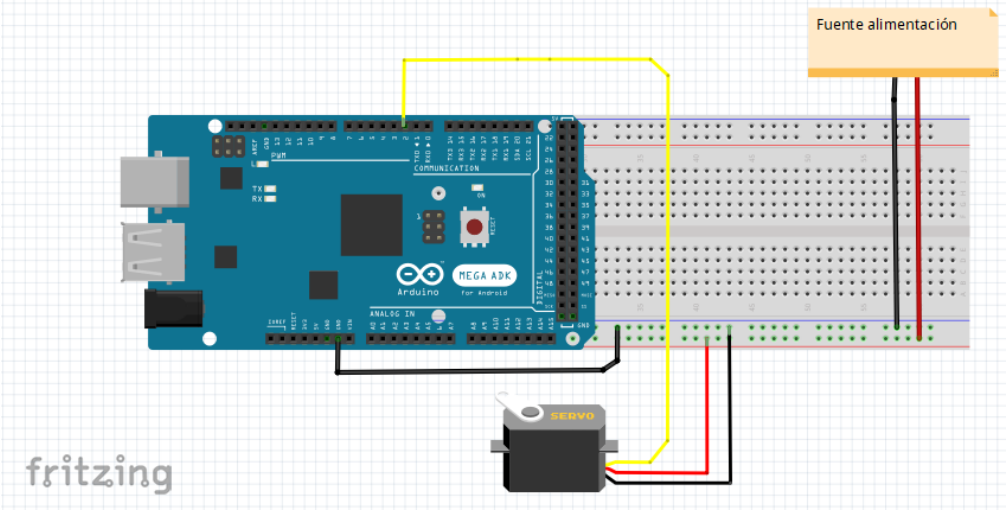


Ilustración 25 Esquema de conexión base

Con el ensamblaje electrónico y mecánico del conjunto base finalizado procedemos a realizar la prueba de funcionamiento para verificar que no tenemos problemas poder seguir con el ensamblaje.

4.3.2. Hombro

Para el hombro vamos a utilizar el servo K-POWER DM4000 y 4 tornillos de M3X30 para anclar el servo a la nueva pieza que hemos reimpreso, entre la unidad funcional del brazo con sus respectivos acoples que tenemos ya impresos del TFG anterior, conseguimos un tornillo de M3x70 para poder realizar la unión entre el servo y la pieza, con todos los materiales seguimos el mismo proceso de ensamblaje, primero acoplamos el servo a la pieza y posteriormente procedemos al ensamblaje electrónico.

Unimos de forma mecánica todas las piezas y empezamos a observar que el peso de la unidad funcional del brazo va a ser excesivo, no para el servo y su par, sino para la forma de transmitir el par del servo al brazo pensada con un único tornillo, pero seguimos ensamblando para poder realizar las pruebas de funcionamiento y ver si esto se confirma.

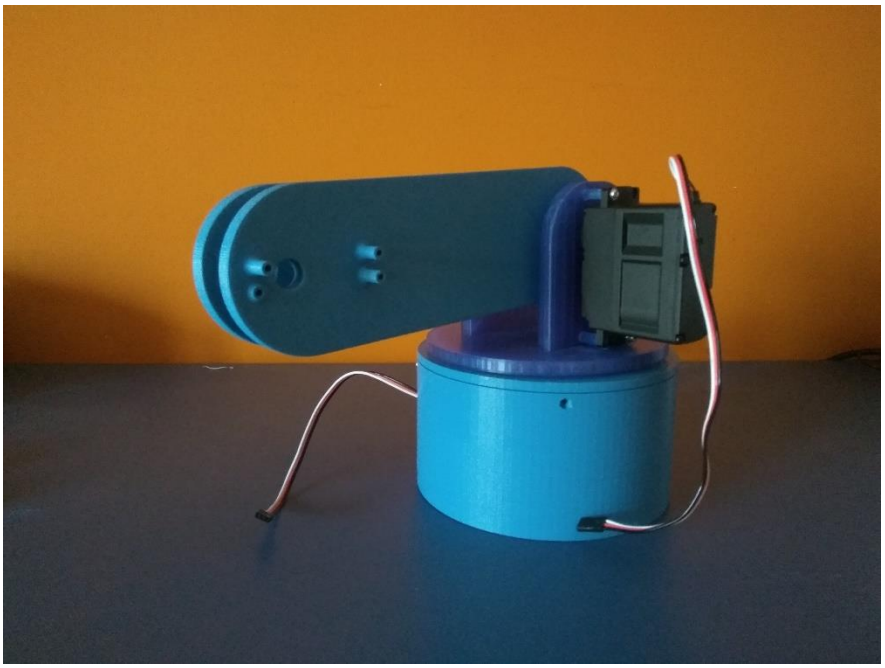
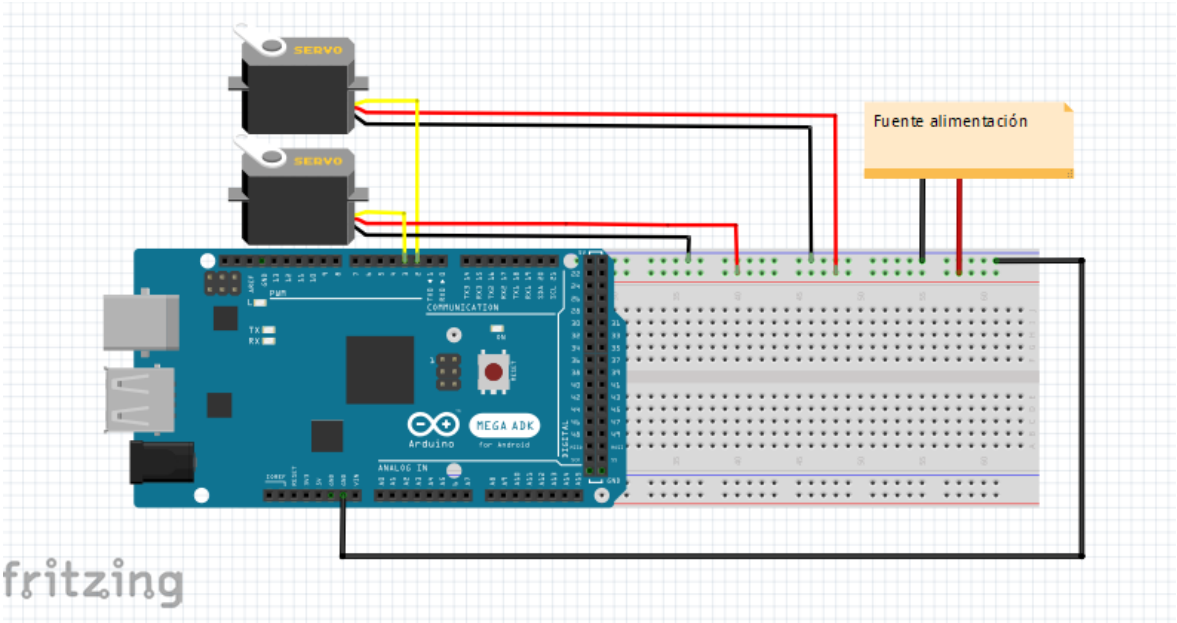


Ilustración 26 Unidades funcionales, base, hombro

Por último, solo nos queda añadir como antes, las conexiones electrónicas para su funcionamiento:



Comenzamos a realizar las pruebas una vez terminado por completo el ensamblaje y observamos que el peso de la pieza es excesivo para la forma de unión elegida para este proyecto, lo que se observa es que la pieza "resbala" sobre el tornillo encargado de transmitir el par del servo al brazo. Este punto es crítico ya que no hay ninguna alternativa a este modo de unión.

A pesar de esto hacemos varias pruebas empleando la pieza del antebrazo (más pequeña que la del brazo, pero con su mismo modo de unión) y conseguimos que se mueva y obtener los 2 GDL, pero al añadir a esta última la muñeca vuelve a suceder lo mismo, el peso es excesivo y vuelve a "resbalar" sobre el tornillo. A pesar de haber modificado la pieza para poder acoplar el servo KPOWER DM4000 que era para el cual se habían hecho los cálculos de movimiento, resulta imposible realizar el movimiento con este sistema de engranaje entre servo y pieza. Además el diseño de todos los soportes del prototipo del brazo robótico están diseñados para el mismo servo el Tower Pro MG995 y no para cada uno de los calculados por separado, por lo tanto ante la imposibilidad de poder seguir con el objetivo inicial de diseñar el software de control para el brazo no nos queda otra que modificar el objetivo del TFG, pasando a ser este el rediseño del brazo sobre todo en su forma de transmitir el par de los servomotores a las piezas para lograr que se mueva.

4.4. REDISEÑO PROTOTIPO

Para poder explicar lo que sucede es necesario acudir al coeficiente de rozamiento entre los materiales, debido a lo observado durante el montaje electrónico y las pruebas. Todos los demás parámetros han sido tenidos en cuenta a la hora de realizar el diseño mecánico en el prototipo, y según los cálculos previos, este debería moverse aplicando los torques necesarios en cada articulación, pero no es así.

Podemos decir que el problema se encuentra en la forma elegida para transmitir el par entre el tornillo y la pieza de PLA, este problema se debe al rozamiento generado entre ambos, puesto que tendría que haber una fuerza mucho mayor a la que se produce entre la pieza de un elevado peso y longitud generándose un momento mayor con respecto al tornillo.

Una fuerza de rozamiento es toda fuerza opuesta al movimiento, que se manifiesta en la superficie de contacto de dos cuerpos siempre que uno de ellos se mueva o tienda

a moverse sobre otro, en nuestro caso práctico el movimiento de uno tiene que transferir el movimiento al otro por lo que nos interesa un coeficiente de rozamiento alto entre ambos, o una forma de unión que no dependa de este rozamiento.

Ante esto el objetivo de nuestro TFG se ve obligado a cambiar y comenzar con un nuevo diseño mecánico en la forma de transmitir el par para lograr que se mueva y poder aprovecharnos de los servos que ya disponemos y añadiendo a estos los nuevos servos necesarios.

4.4.1. Rediseño mecánico

Para el diseño mecánico únicamente vamos a poder aprovechar del anterior proyecto la base y la tapa de esta, el resto de las piezas vamos a tener que rediseñarlas al presentarse el primer problema en el cover base que es el componente principal al que se une el hombro y sujeta el resto del brazo, al tener que modificarlo va a ser necesario adaptar el resto del brazo para evitar problemas como el que nos surge anteriormente.

A la hora del rediseño de las piezas no podemos hacerlo con las mismas dimensiones que el prototipo previo ya que tenemos una serie de limitaciones en cuenta. Lo primero de todo es necesario reducir el tamaño de algunas piezas para poder imprimirlas en la nueva impresora que vamos a emplear para la creación de estas. La limitación que se nos presenta es que las piezas no pueden tener una longitud superior a 20 cm por motivos físicos de la impresora.

La impresora utilizada se trata de la Anet A8, una impresora que se puede adquirir por un precio de 150€ y la cual te viene en kit para montarla y configurarla en casa, como material para las piezas seguiremos usando PLA.

Ilustración 27 Anet A8

(«IMPRESORA 3D ANET A2 DIY [anet-a2] : MasToner», s. f.)

Una vez explicado el material que vamos a usar para el diseño de las piezas comenzamos con su explicación

4.4.1.1. Base y tapa base

Estas piezas son las únicas que vamos a reaprovechar del proyecto anterior sin ninguna modificación, el resto se van a intentar asemejar en la mayor medida posible, pero van a ser necesarios cambios.

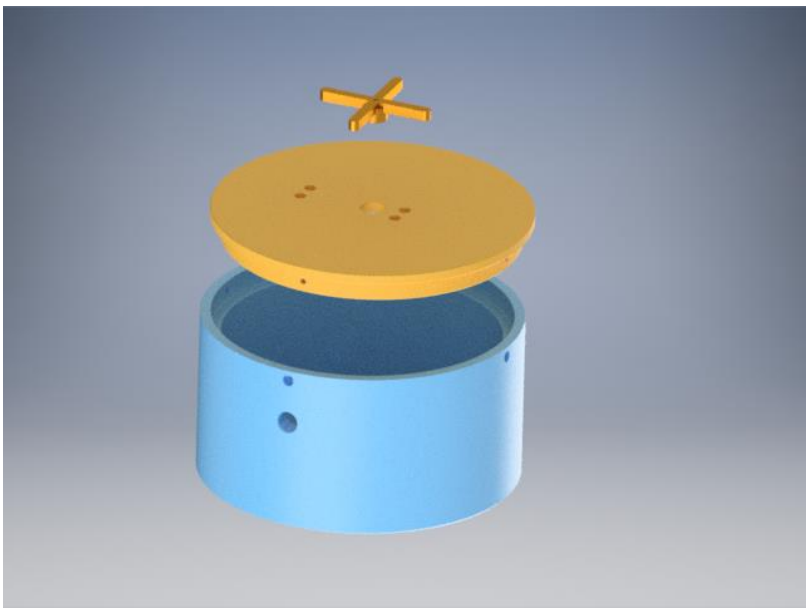


Ilustración 28 Montaje Base, tapa base

4.4.1.2. Cover base

En este caso ya introducimos modificaciones partiendo de la nueva forma de transmitir el par de nuestros servos al brazo, aprovechando que ya contamos con los servomotores que van a aplicar la fuerza para mover el hombro, diseñamos la pieza a medida de estos, y no al revés.

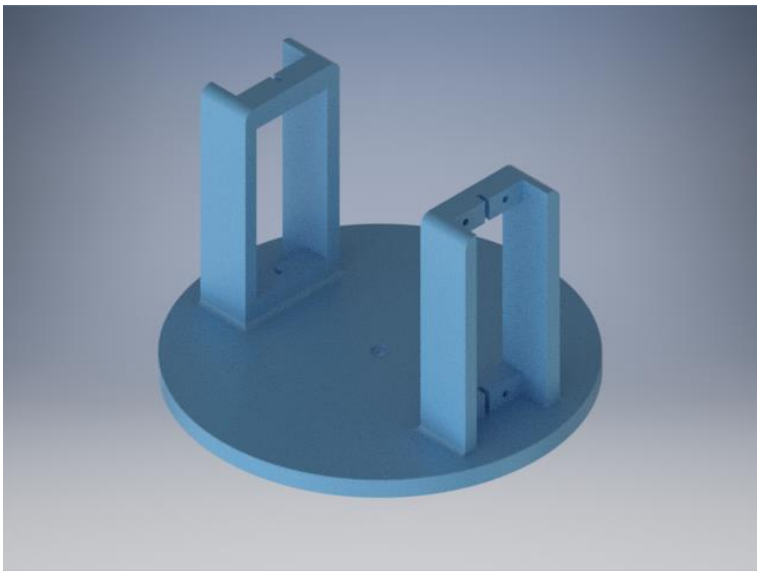


Ilustración 29 Cover Base

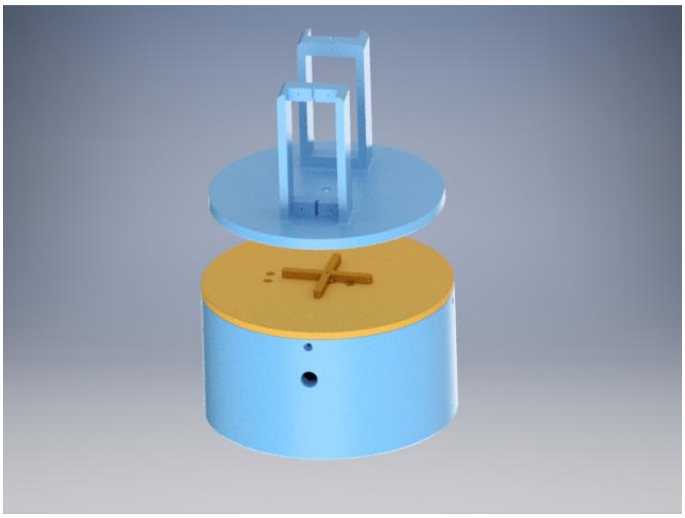


Ilustración 30 Unidad funcional Base

4.4.1.3. Hombro y unión hombro

En este caso cambiamos por completo el diseño del hombro, pasando a ser formado por tres piezas en vez de una única sólida y por supuesto también diseñada respecto a las dimensiones de los servomotores.

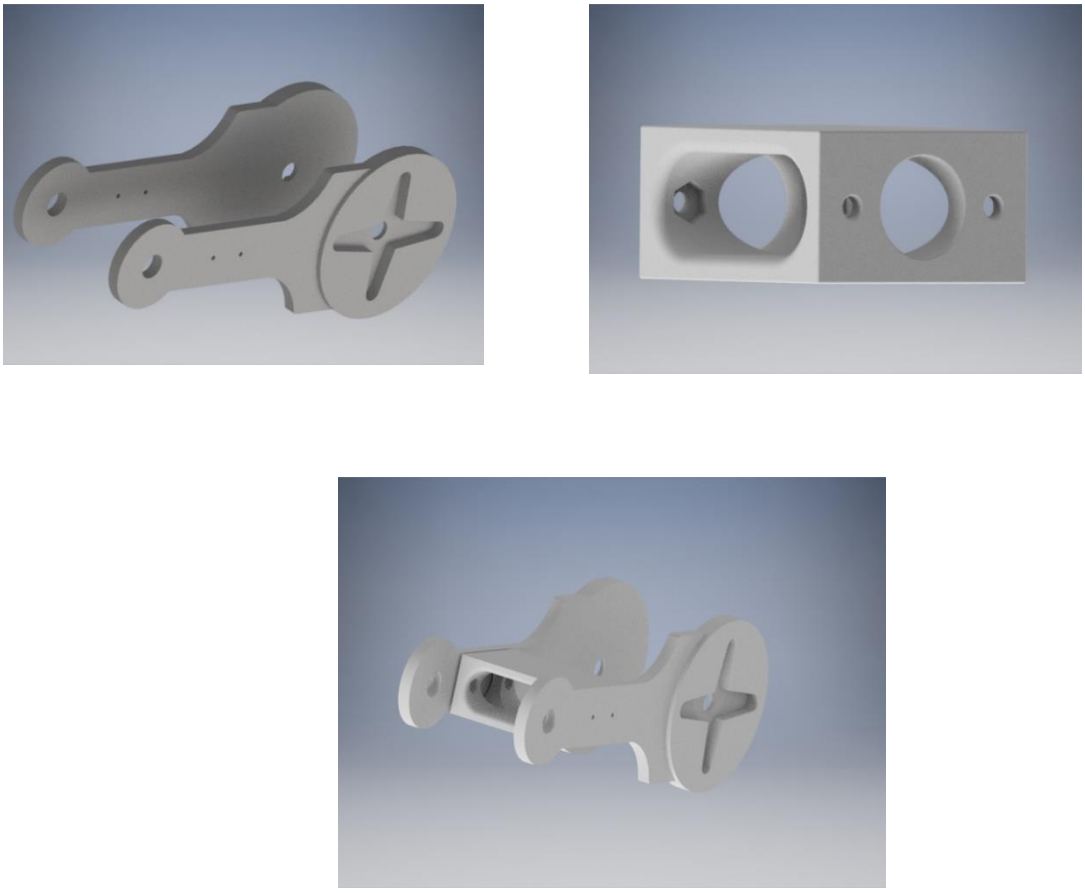
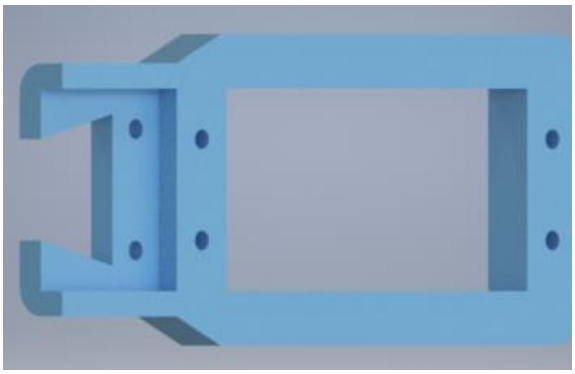
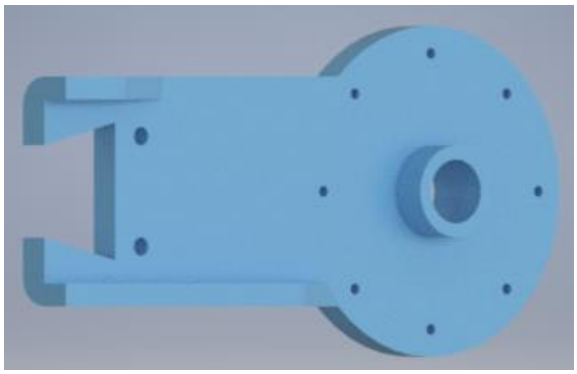
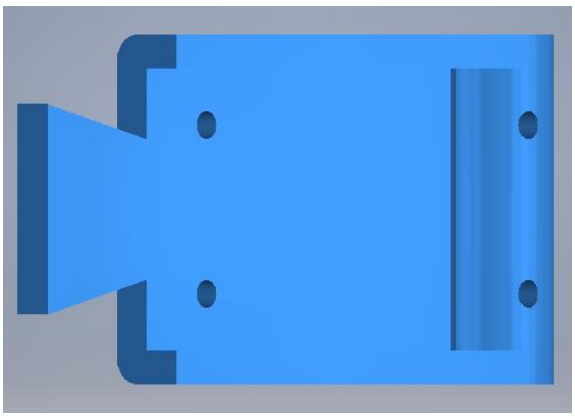
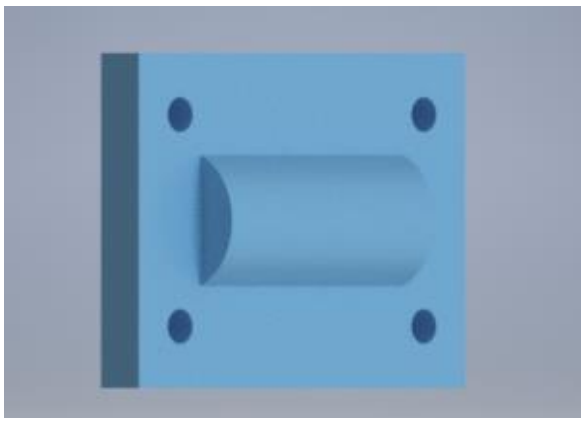
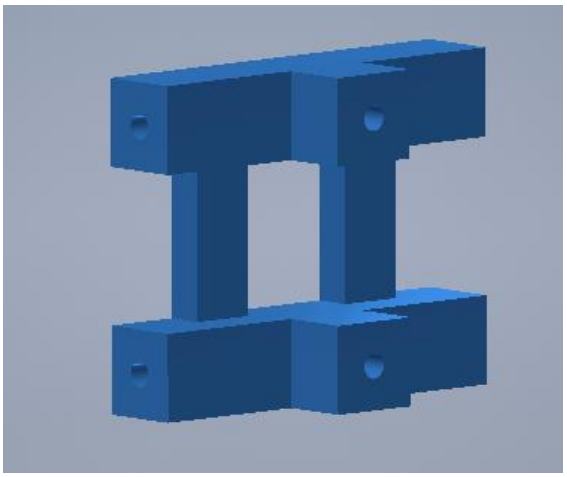
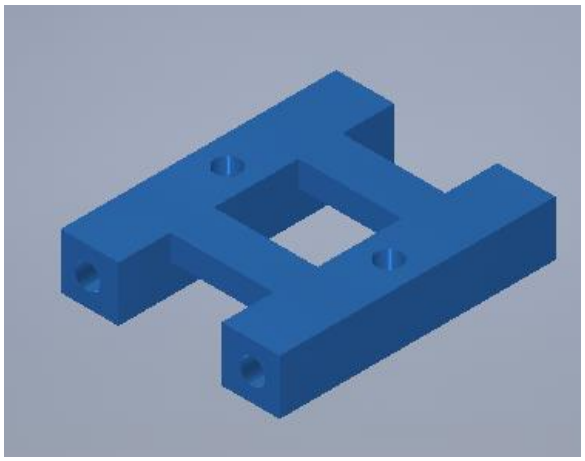
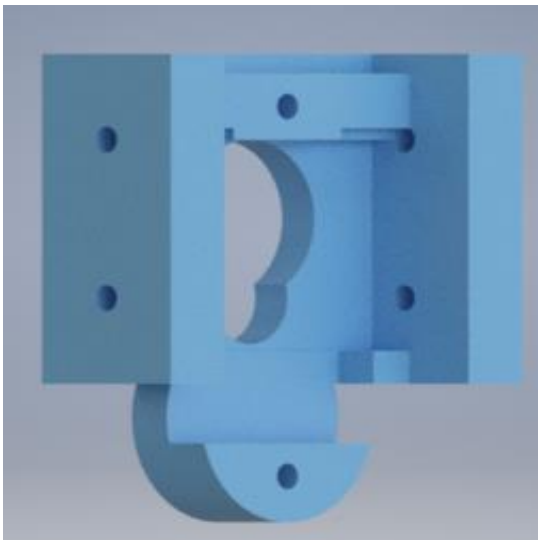


Ilustración 31 Unidad funcional Hombro

4.4.1.4. Codo

Al igual que en el hombro hemos optado por hacerlo en varias piezas para posteriormente ensamblarlas.



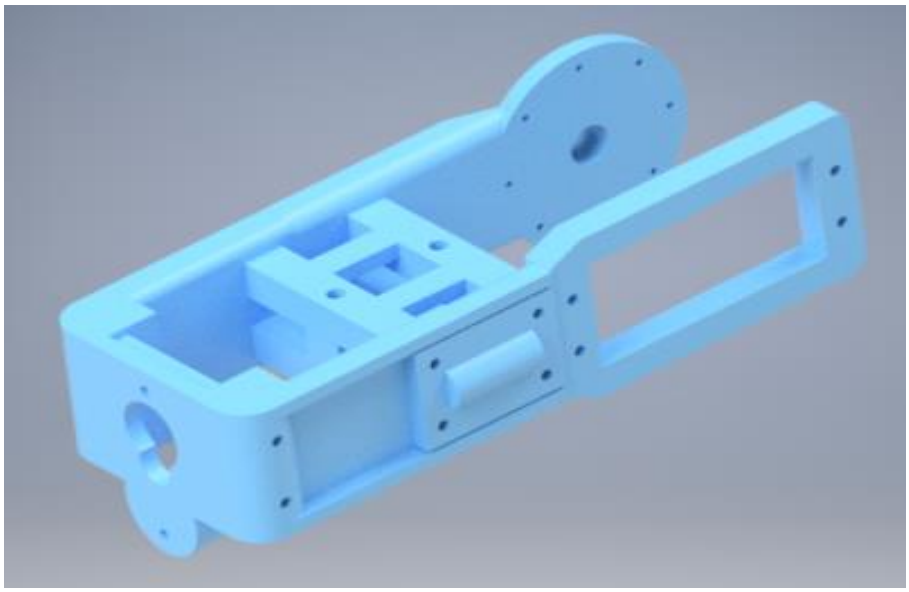
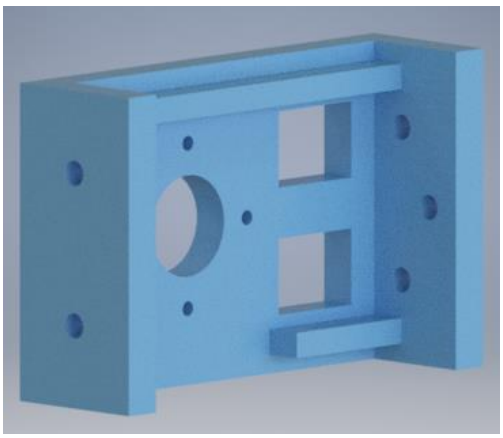
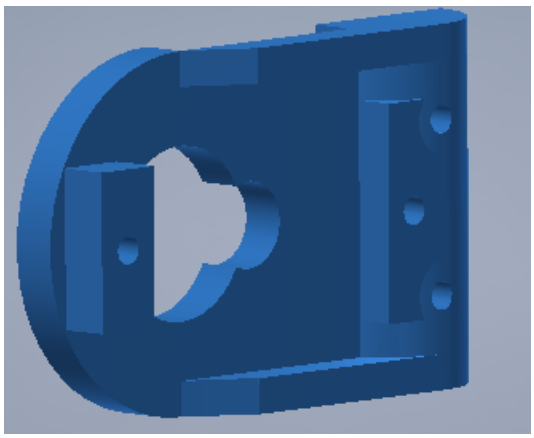
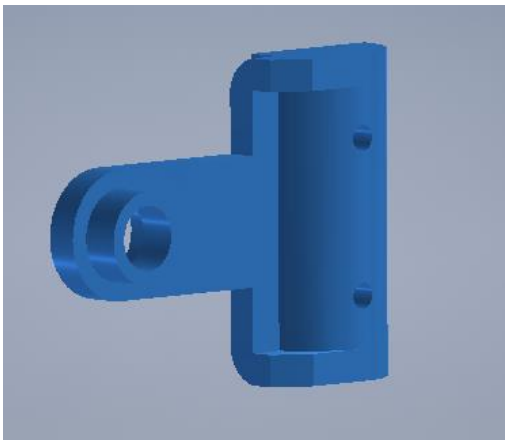
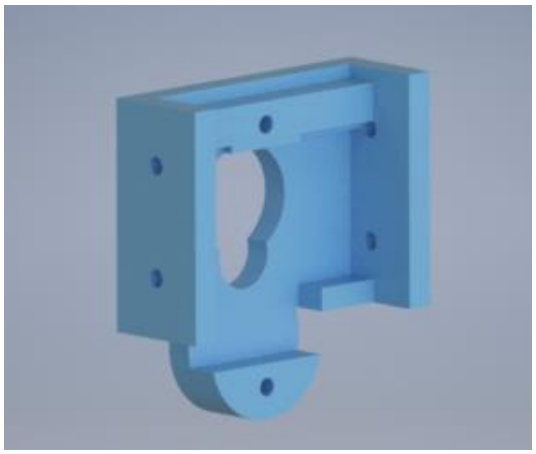
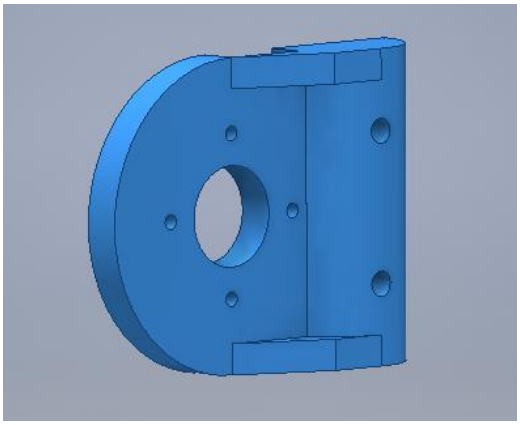
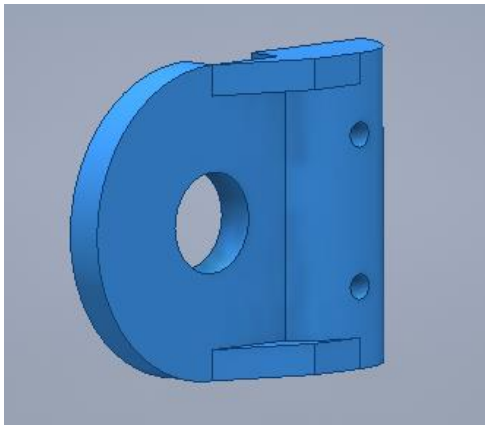


Ilustración 32 Unidad Funcional Codo

4.4.1.5. Muñeca

Siguiendo con los ejemplos anteriores de hombro y codo, la muñeca también la realizamos en varias partes que posteriormente ensamblaremos para facilitar su montaje y movimiento.

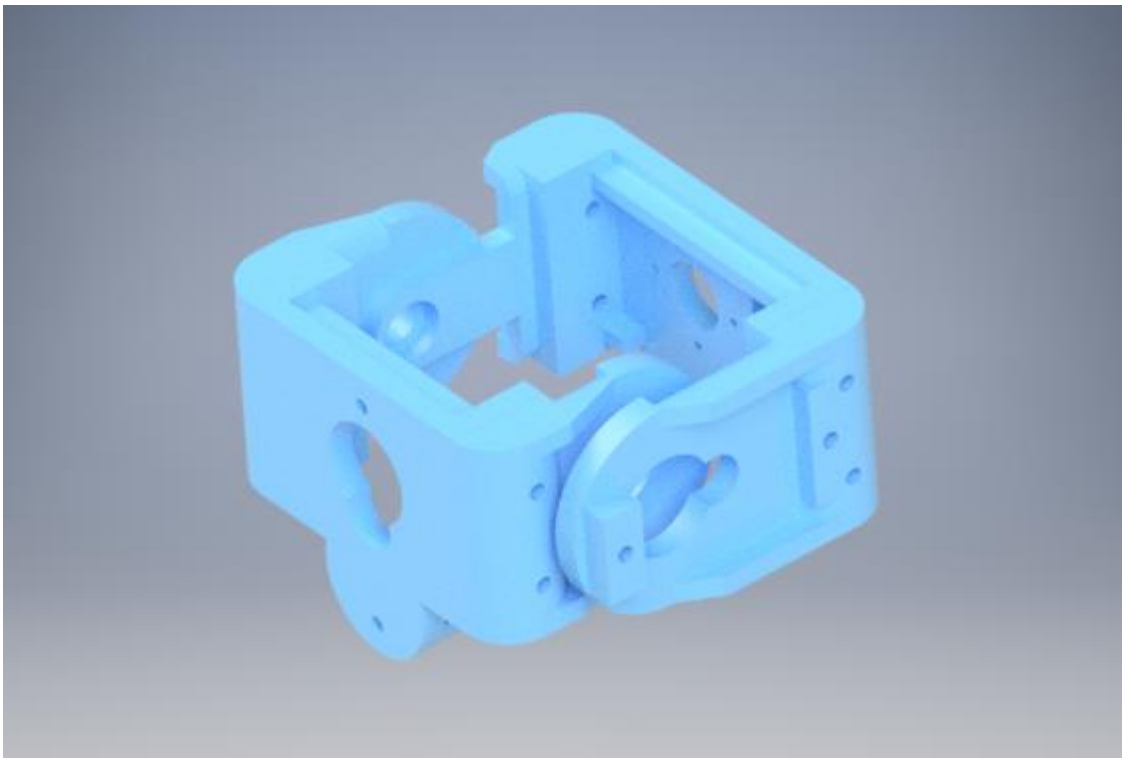


Ilustración 33 Unidad funcional muñeca

4.4.1.6. Pinza

Para la pinza vamos a aprovecharnos de un diseño subido a la plataforma thingiverse, comunidad en la que los usuarios suben los diseños de manera libre para que cualquiera pueda descargarlos e imprimirlos.

El enlace de esta:

https://www.thingiverse.com/thing:7109

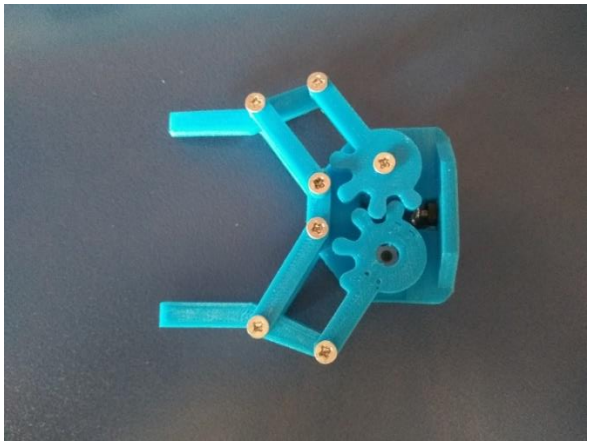


Ilustración 34 Pinza Impresa

4.4.2. Ensamblaje mecánico y electrónico nuevo diseño

4.4.2.1. Hombro

Previamente habíamos ensamblado y simulado las conexiones eléctricas del acople entre la base y el hombro del diseño antiguo, como la base y el cover base son únicamente las piezas del diseño anterior que vamos a aprovechar, seguimos con las del nuevo diseño para comprobar que funcionan correctamente y no tenemos problemas.

El hombro es la primera pieza que rediseñamos, material necesario para el ensamblaje:

- Hombro impreso
- Anclajes impresos para los servos

  Servomotor KPOWER DM4000 (Se usarán dos, pero uno de ellos es totalmente inoperativo, se le solicito al vendedor si podía proporcionarnos uno que no funcionará con la compra del otro servo, con el unico objetivo de que el brazo quedará más estético)

- Acoples Del brazo y tornillos de sujeción de los servos
- Plato para acoplar el servomotor que moverá el codo, en este caso el servo FUTABAS3003
- 4 tornillos M3X20mm, 4 tornillos M3X12mm, 8 tuercas M3.

Ilustración 35 Ensamblaje Hombro

4.4.2.2. Codo

Material necesario para el ensamblaje del codo:

- Piezas del codo impresas
- Servomotor FUTABAS3003 (Tornillos de sujeción, el plato ya ha sido insatalado en el hombro previamente)
- Anclajes servo FutabaS3003
- Servo MG90S (Tornillos de sujeción y acople, este se encargará del movimiento de giro de la muñeca)
- 12 Tornillos aglomerado M2,5x16mm

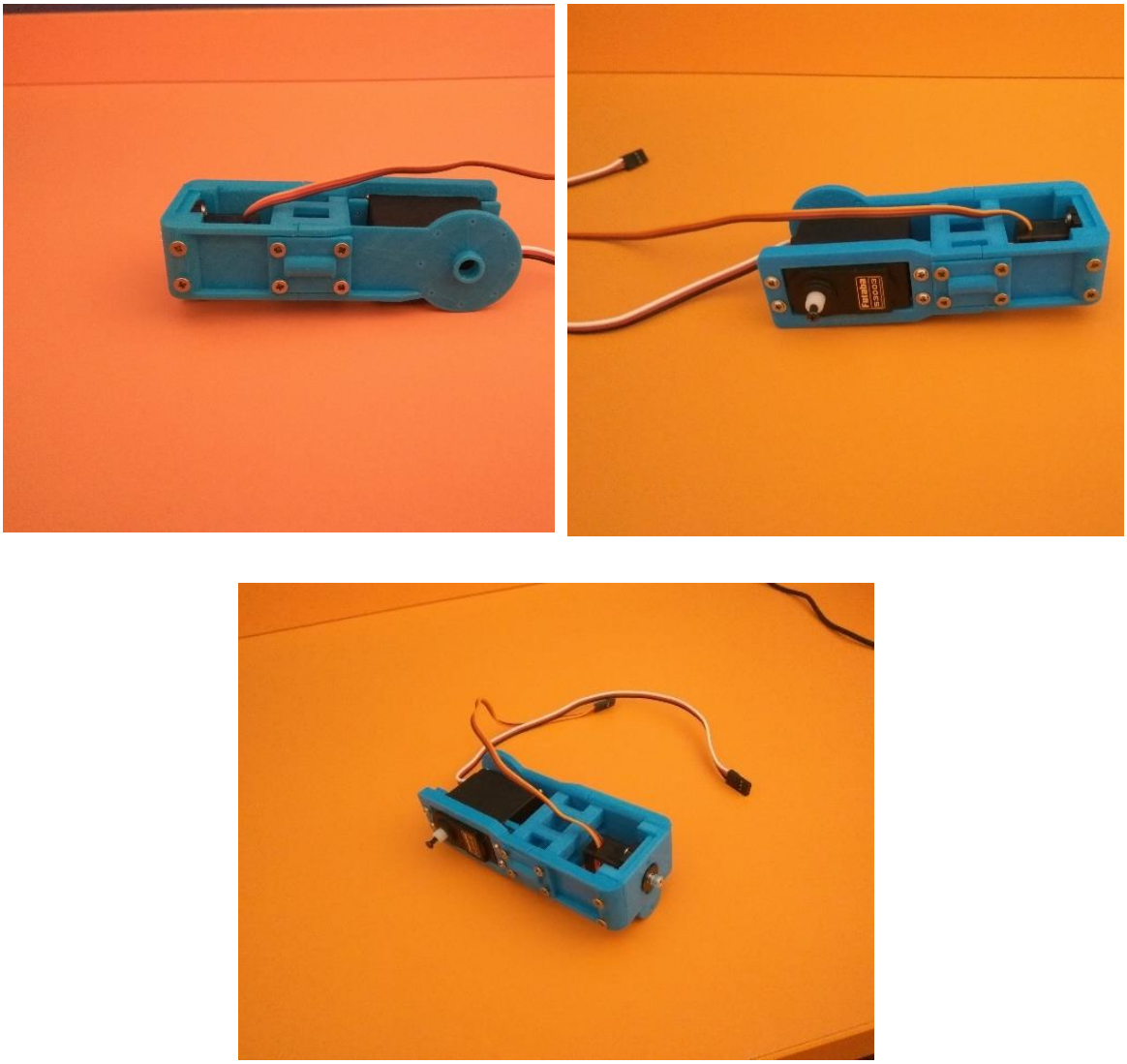


Ilustración 36 Ensamblaje Codo

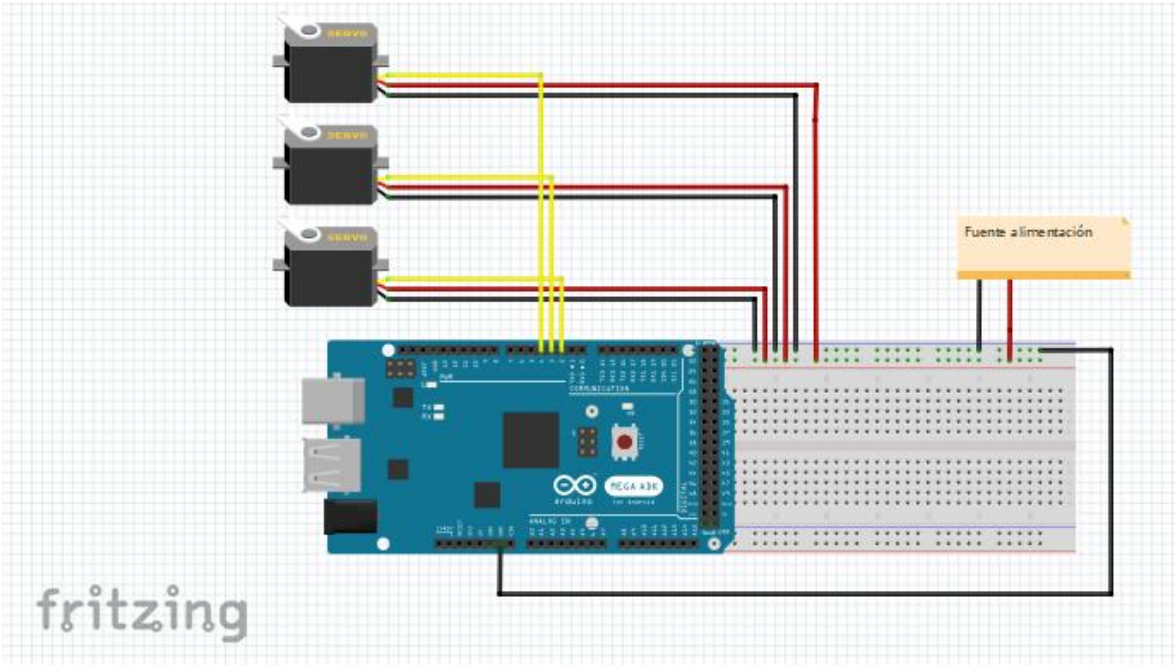


Ilustración 37 Esquema conexión base, hombro y codo

Este sería el conexionado eléctrico con los cuatro servos, base, hombro, y los dos que irían en el codo, como vemos no aparece el módulo bluetooth conectado, y esto es porque como explicamos en el apartado del diseño electrónico y control, para el ensamblaje probaremos el funcionamiento de los servomotores con la prueba del potenciómetro y así poder ajustarlos en la posición que queremos. En el ensamblaje final incluiremos el módulo bluetooth.

4.4.2.3. Muñeca

Material necesario para el ensamblaje de la muñeca:

- Piezas de la muñeca impresas
- Servomotor MG90S encargado de subir y bajar la muñeca (Acople y tornillos de sujeción)
- Servomotor MG90S que se encargara de aportar giro a la pinza (Acople y tornillos de sujeción)
- 8 Tornillos aglomerado M2,5x16mm.

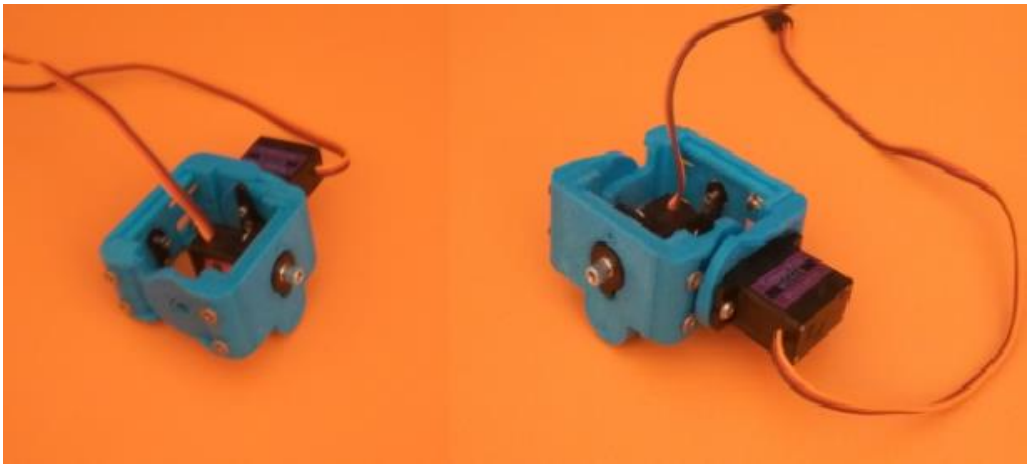


Ilustración 38 Ensamblaje muñeca

No adjunto el diseño electrónico ya que es seguir el esquema de las conexiones anteriores (base, hombro y codo), adjuntare el esquema final con todos los servomotores y el módulo bluetooth, es decir, el diseño final.

4.4.2.4. Pinza

Material necesario para el ensamblaje de la pinza:

- Piezas de la pinza impresas
- Servomotor HITEC HS-55 (Acople y tornillos de sujeción)
- 3 Tornillos M3x20mm, 4 Tornillos M3x25mm, 7 Tuercas M3

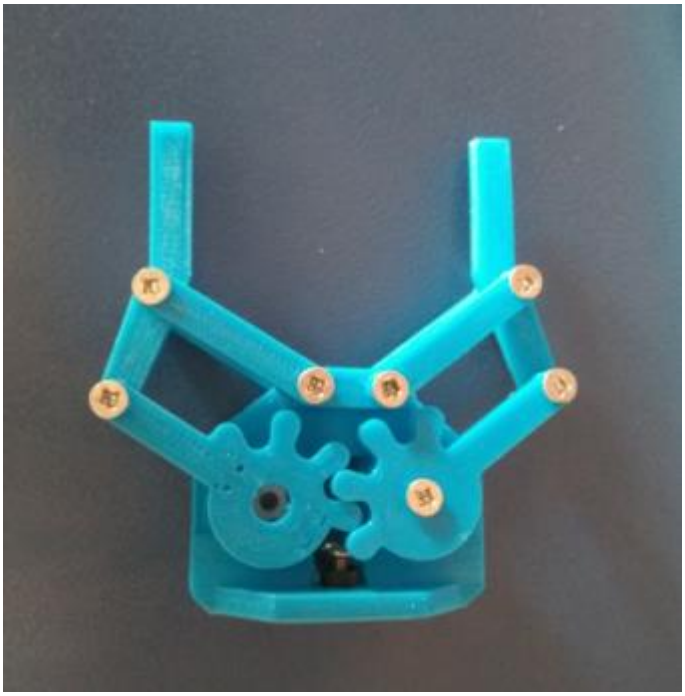
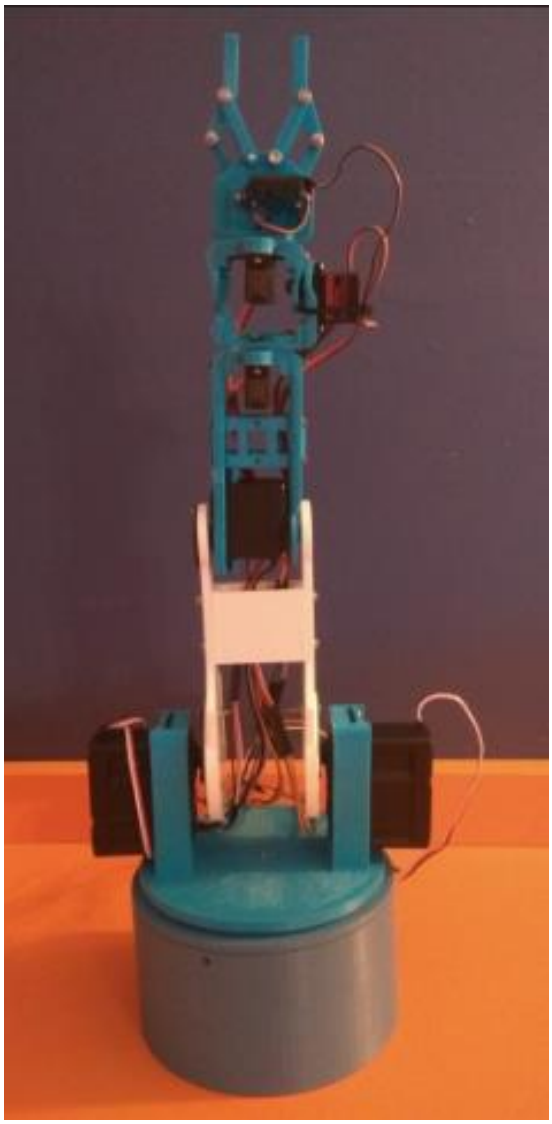


Ilustración 39 Ensamblaje pinza

Cada unidad funcional por separado no lleva mucho trabajo ensamblarla, lo complicado es realizar el ensamblaje completo del brazo robótico con cada servo a la siguiente unidad funcional, ya que como explicábamos es necesario hacerlo individualmente con cada servo con ayuda de la prueba del potenciómetro para poder aprovechar los 180º que nos ofrece cada servo físicamente en cada articulación.

Además de cablear todo de tal manera que no dificulte el movimiento del brazo y permita realizar todos los movimientos sin dificultades.

4.4.2.5. Brazo completo



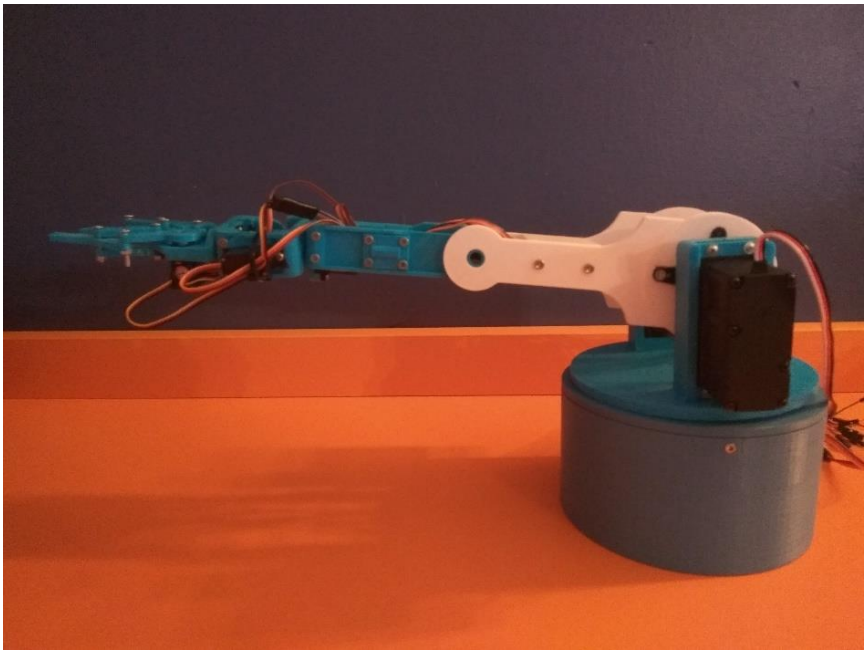


Ilustración 40 Ensamblaje final

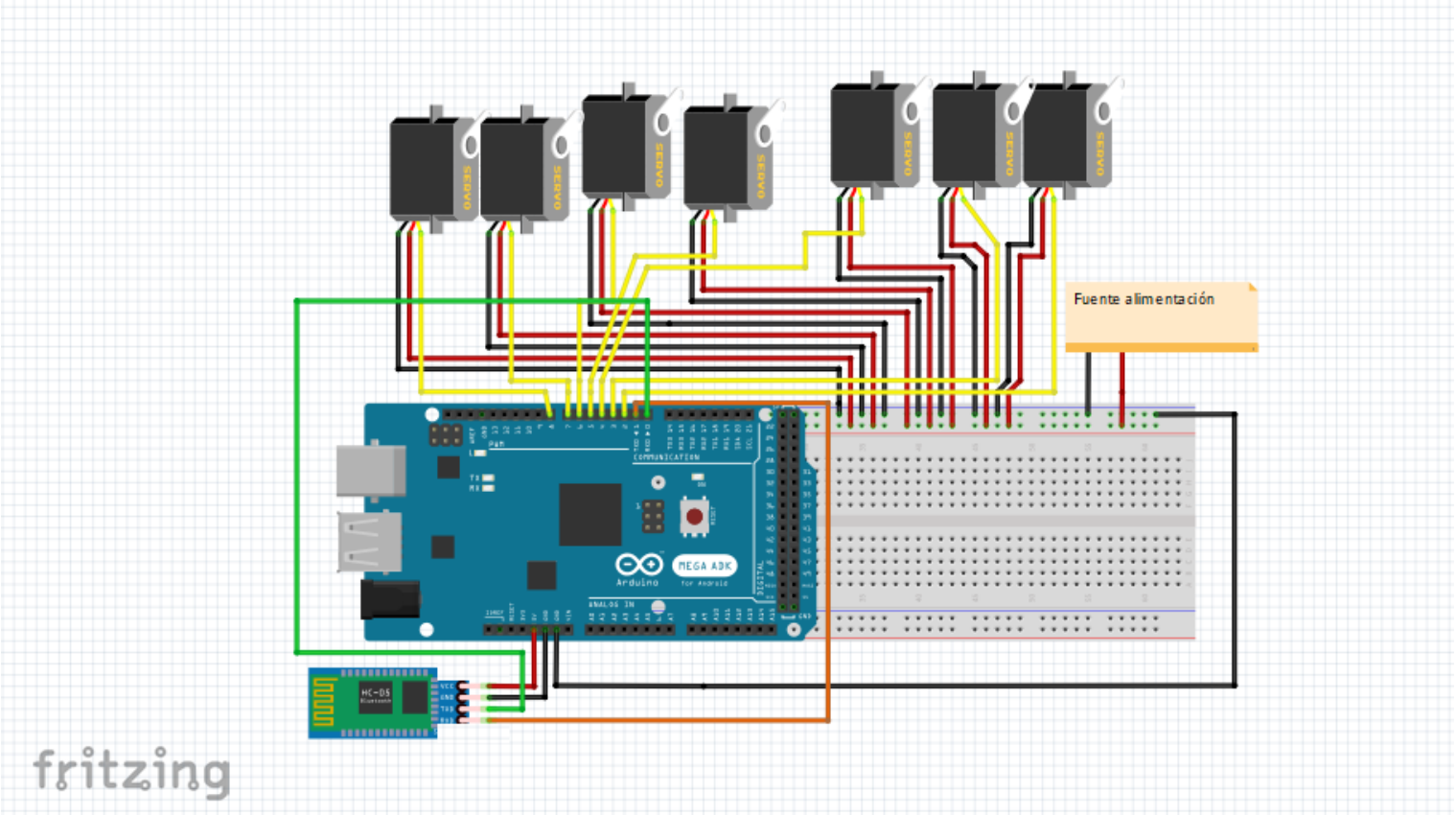


Ilustración 41 Ensamblaje electrónico final

En el diseño electrónico incluyo todos los elementos que van a ser usados en el sistema final, el módulo bluetooth aparece por primera vez y aparecen todos los servomotores que previamente habíamos obviado por ser repetitivo el diagrama de conexiones para cada servomotor.

4.5. SISTEMA DE CONTROL

Se opta por el control del brazo robótico mediante un dispositivo móvil, se realiza el diseño de una app con la que podremos controlar cada eje del servomotor independientemente.

Dividimos en dos fases el desarrollo del sistema de control, el diseño y programación de la aplicación para el dispositivo móvil y el diseño del programa en Arduino.

4.5.1. Diseño aplicación dispositivo móvil

Está realizado con el programa online MITT APP INVENTOR, emplean un entorno gráfico para el diseño y un entorno de bloques para la programación, es un programa con el que se puede realizar cualquier tipo de aplicación, aunque con ciertas limitaciones.

Tanto el sistema de movimiento como la aplicación, van a ser ambas muy sencillos por temas de tiempo ya que al tener que rediseñar el robot entero la carga del TFG aumentaba considerablemente.

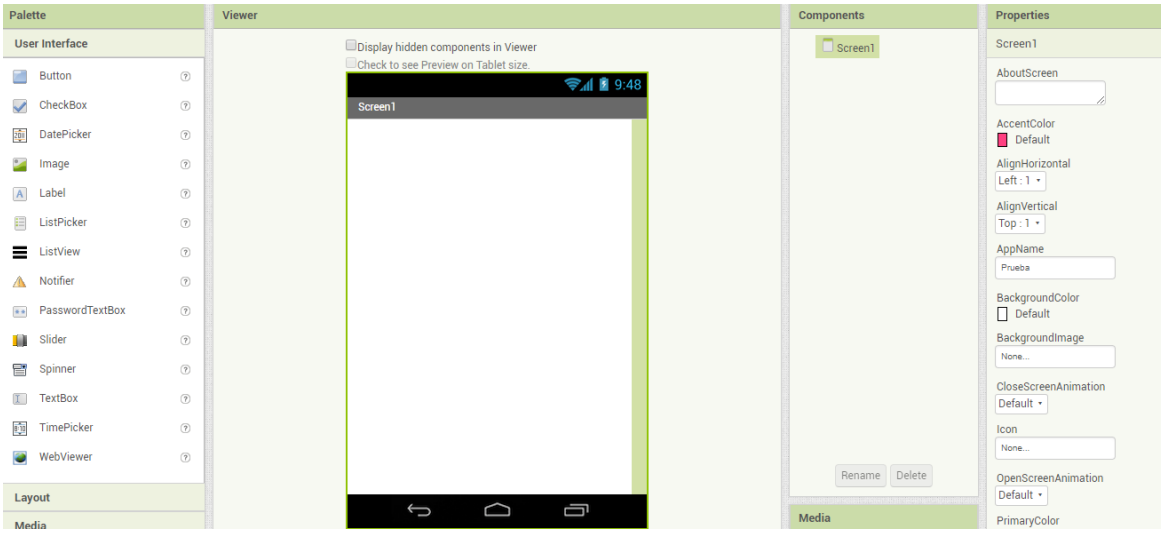


Ilustración 42 Interfaz página web diseño app

Si observamos la interfaz de la página web, a la izquierda vemos las opciones que nos da para insertar en la zona central que simula la pantalla del móvil y como lo vamos a ver, arrastrando cada elemento al interior de la pantalla podremos visualizarlo y posteriormente programar cada función de lo que insertemos. Cada elemento que

añadamos también se insertará automáticamente en la zona derecha "Components", la zona de la derecha es en la cual podremos editar de forma física como queremos que se visualice en pantalla.

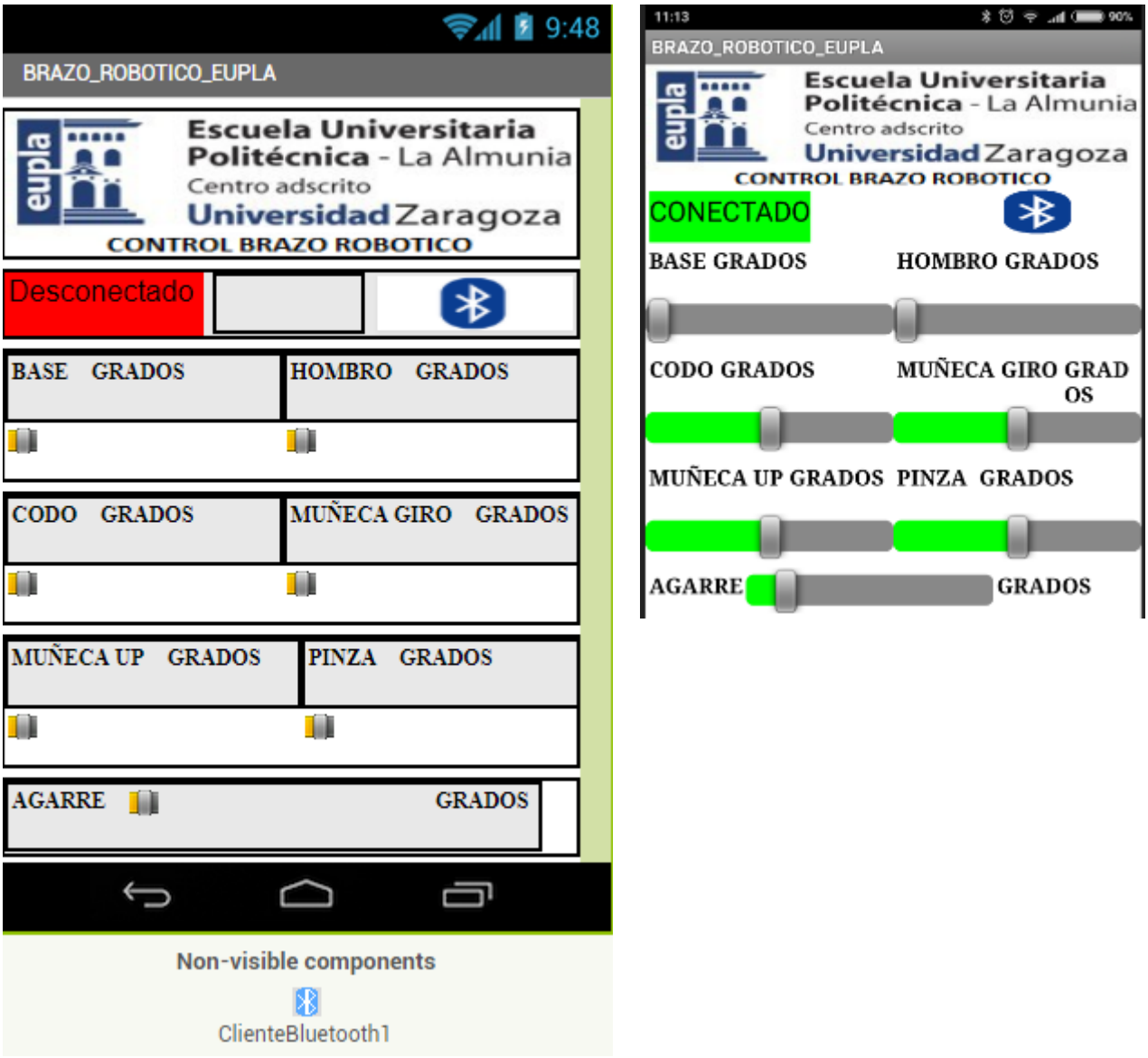


Ilustración 43 Interfaz app

A la izquierda vemos el diseño del interfaz visto desde el ordenador, a la derecha aparece la aplicación vista desde el dispositivo móvil y una vez conectado al dispositivo bluetooth para manejar cada eje.

Es un diseño muy sencillo que nos permite controlar los grados de cada eje mediante deslizadores que van de 0 a 180, inicializándolo en la posición que nosotros programemos

EN la zona superior de la aplicación vemos el indicador de conectado o desconectado, es simplemente un cuadro de texto que cambiará de color y texto en función de cómo se encuentre, en la zona central es simplemente un cuadro horizontal de separación, y a la derecha tenemos un botón al que le hemos puesto la imagen del bluetooth, ya que cuando pulsemos este nos aparecerá una lista con todos los dispositivos bluetooths cercanos.

Ilustración 44 Zona superior App

Cada eje se controla mediante un deslizador, y como hemos dicho previamente es muy sencillo, cada eje únicamente se compone de su título y el apartado grados que cambiará de texto a números cuando se inicializar la aplicación y movamos el deslizador, indicándonos en cada momento como se encuentra el eje seleccionado.

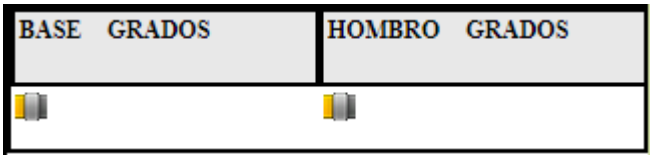


Ilustración 45 Movimiento ejes

Todos los demás ejes se controlan de la misma manera y están diseñados de igual forma, simplemente cambian los grados en los que fijamos que aparezcan cuando iniciamos la aplicación, esto se hace desde el apartado gráfico y no desde la programación, que ahí se le asignara la posición física a la que queremos que vayan cuando inicialicemos.

A cada elemento que incluimos en la interfaz gráfica tenemos que asignarle un nombre en la parte derecha de componentes que como comentaba se insertan

automáticamente ya que estos nombres son los que vemos luego en la zona de programación de bloques.

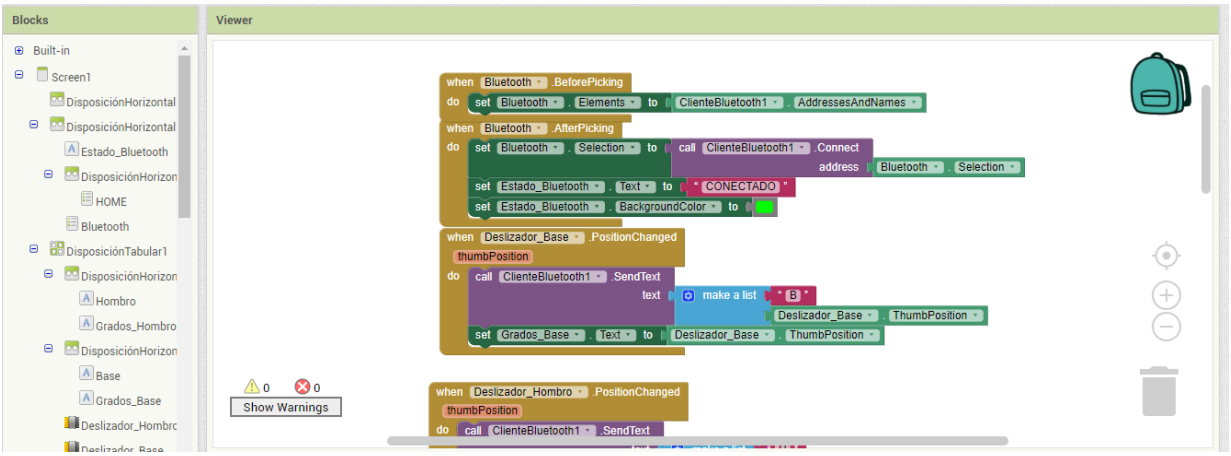


Ilustración 46 Interfaz página web programación mediante bloques

Como comentaba en la parte derecha aparecen todos los elementos que hemos insertado en la parte gráfica que simula la pantalla de nuestro dispositivo y aquí es donde realizamos la programación, esta programación hay que realizarla previo a la programación de Arduino, pero pensando también en como realizaremos la programación en Arduino para luego evitarnos tener que hacer muchos cambios en ambos diseños.

Únicamente se programa el botón bluetooth y cada deslizador por separado, para el botón bluetooth la programación es la siguiente:

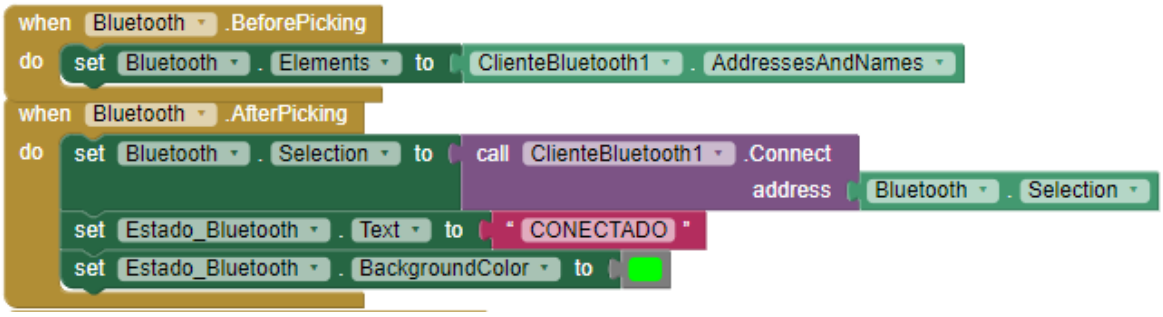


Ilustración 47 Programación en bloques bluetooth

La primera parte en la que cuando pulsemos el botón nos muestre las direcciones de todos los clientes bluetooth a los que nos podemos conectar, y la segunda en la que una vez seleccionado se conecta al cliente seleccionado y cambia el cuadro de texto en

el que previamente ponía desconectado y aparecía en rojo, para pasar a poner conectado y aparecer en verde.

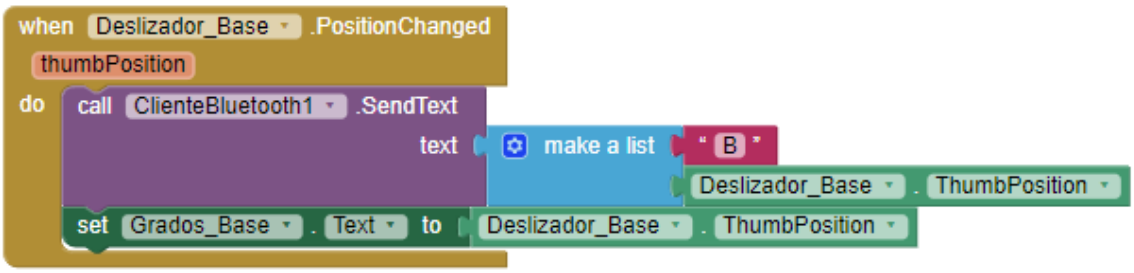


Ilustración 48 Programación en bloques deslizador

Lo siguiente que programamos son los deslizadores, ponemos de ejemplo el de la base, como vemos esta seleccionado con el nombre Deslizador_Base, y lo que consiste es que cuando cambiamos de posición el deslizador, el dispositivo móvil que esta conectado al cliente bluetooth y asimismo con Arduino envía el carácter B, que después mediante la programación en Arduino haremos que ese carácter sea el que mueva el servomotor de la base, además cambiamos el texto grados del deslizador por los grados en los que se encuentra realmente el servomotor.

El resto de deslizadores siguen el mismo principio, pero cambiando el carácter que enviamos.

4.5.2. Diseño programa Arduino

Una vez terminado el diseño de la aplicación para el dispositivo móvil tenemos que hacer el diseño en Arduino que al final es el encargado de transmitir las ordenes a los servomotores.

Con ayuda de la pagina https://www.prometec.net/ en la que aparecen muchos tutoriales de Arduino y te enseñan a manejar los distintos componentes que vamos a usar, el módulo bluetooth y los servos.

Lo explicamos por partes como anteriormente, con explicar un eje el comportamiento del resto es el mismo.

```
#include <Servo.h>

Servo Servo_Base;
Servo Servo_Hombro;
Servo Servo_Codo;
Servo Servo_Muneca_Giro;
Servo Servo_Muneca_Up;
Servo Servo_Pinza;
Servo Servo_Agarre;

char Parametro;
String readString;
```

Ilustración 49 Variables Servos Arduino

Lo que vemos en la imagen es como incluimos la biblioteca Servo.h y posteriormente creamos las variables que van a ser de tipo Servo de cada servomotor, además de una variable tipo carácter y otra variable tipo cadena.

```
void setup() {
Servo_Base.attach(2);
Servo_Hombro.attach(3);
Servo_Codo.attach(4);
Servo_Muneca_Giro.attach(5);
Servo_Muneca_Up.attach(6);
Servo_Pinza.attach(7);
Servo_Agarre.attach(8);
Serial.begin(9600);

Servo_Base.write(0);
Servo_Hombro.write(0);
Servo_Codo.write(90);
Servo_Muneca_Giro.write(90);
Servo_Muneca_Up.write(90);
Servo_Pinza.write(90);
Servo_Agarre.write(40);
delay(10);
}
```

Ilustración 50 Void setup Arduino

En el void setup del Arduino estas las instrucciones que solo se ejecutan una vez. Indicamos a que puerto se encuentra asociado cada servomotor, como vemos van del 2 al 8 el 0 y el 1 están reservados para el módulo bluetooth ya que son los pines TX y RX.

El segundo párrafo de instrucciones es a la posición a la que mandamos los servos cuando cargamos el programa en Arduino,

```
void loop() {

if (Serial.available())
{
  Parametro = Serial.read();
if (Parametro=='B'){
  Movimiento_Base();
}
if (Parametro=='H'){
  Movimiento_Hombro();
  }
  if (Parametro=='C'){
    Movimiento_Codo();
  }
  if (Parametro=='M'){
    Movimiento_Muneca_Giro();
  }
  if (Parametro=='U'){
    Movimiento_Muneca_Up();
  }
  if (Parametro=='P'){
    Movimiento_Pinza();
  }
   if (Parametro=='A'){
    Movimiento_Agarre();
  }
}}
```

Ilustración 51 Void loop Arduino

Como sabemos este es el ciclo que se repite constantemente en el Arduino, no como el anterior que solo se ejecuta cuando cargamos el programa, aquí ponemos los condicionales, es decir las condiciones que tiene que a ver para ejecutar una instrucción o otra.

El primer condicional es para comprobar que hay caracteres en el puerto serial, y si es así que los lea en la variable tipo carácter que hemos creado previamente llamada Parametro,

Los siguientes son las condicionales con las letras que hemos puesto en la aplicación diseñada para el móvil cada vez que tocábamos un deslizador, es decir, si este parámetro que estamos leyendo hemos movido el deslizador de la base, le llegara

el carácter B, entonces ejecutaremos la función Movimiento_Base() que pasamos a explicarla ahora y que será la misma función para el resto de servos.

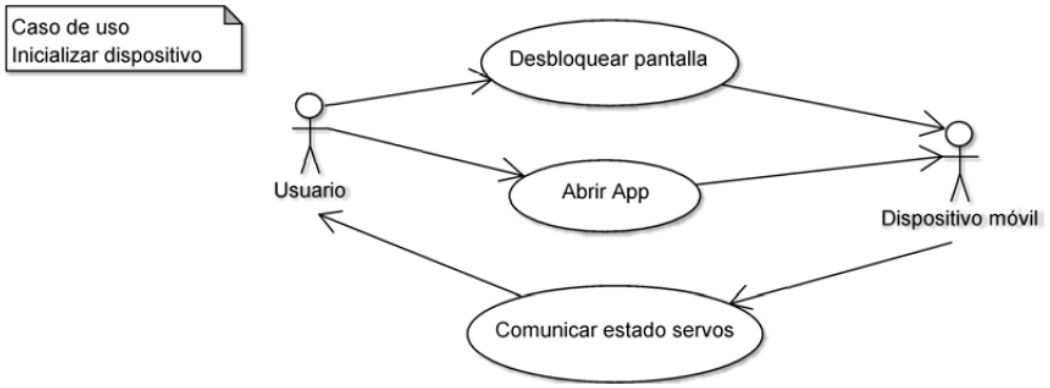
```
void Movimiento_Base()
{
  delay(10);
  while (Serial.available()){
    char Parametro_Temporal = Serial.read();
    readString += Parametro_Temporal;
  }
if (readString.length() >0) {
  Serial.println(readString.toInt());
  Servo_Base.write(readString.toInt());
  readString="";
}
}
```

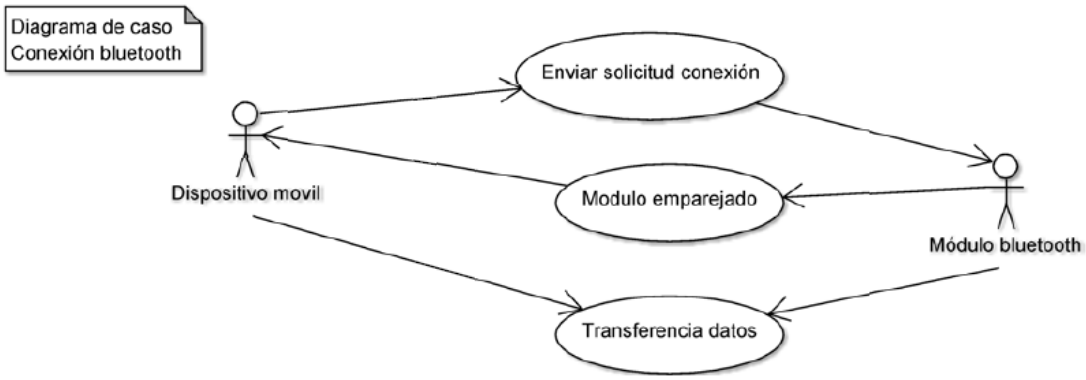
Ilustración 52 Función movimiento servos

Todas estas funciones empiezan con el condicional while, cuyo funcionamiento es, que mientras estén llegando caracteres por el puerto serial, los guardo en una nueva variable llamada "Parametro_Temporal", y con el readString nos permite crear una cadena de caracteres, introducimos otro if con el que verificamos el tamaño de la cadena, y usamos el readString.toINT para que nos lo convierta en un número entero, y mandamos ese valor a dicho servo, en este caso Servo_Base, esto es lo único cambia en las demás funciones para los otros servos, y es es donde mandamos este valor.

4.6. DIAGRAMAS CASO DE USO, UML

4.6.1. Diagramas de caso





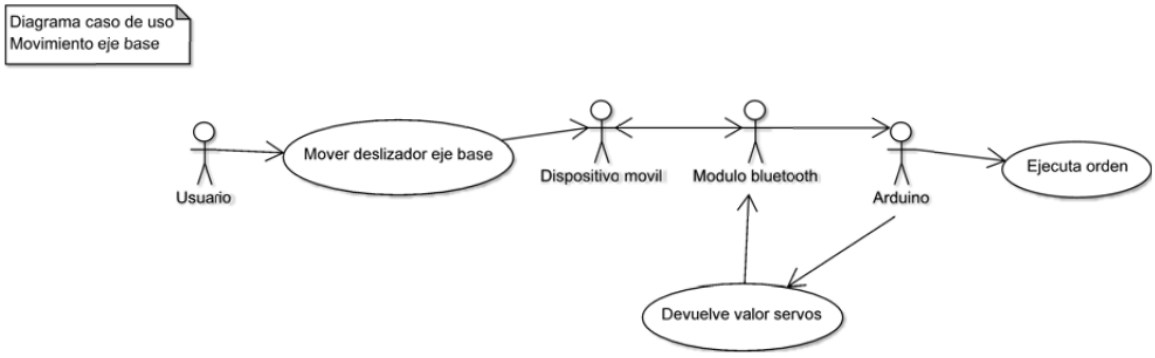


Diagrama caso de uso
Movimiento eje base
Usuario
Mover deslizador eje base
Dispositivo movil
Modulo bluetooth
Arduino
Ejecuta orden
Devuelve valor servos

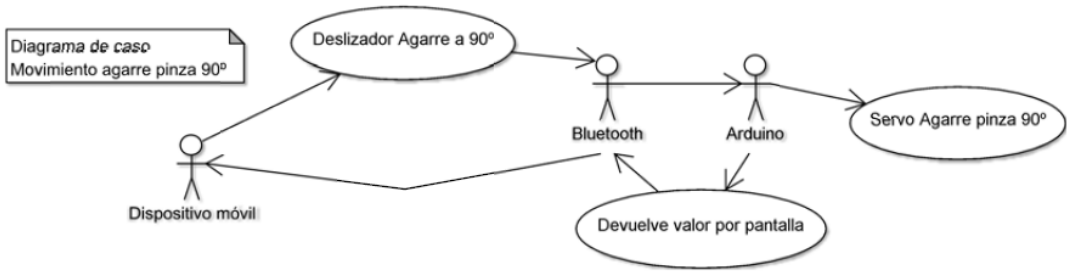


Diagrama de caso
Movimiento agarre pinza 90º
Deslizador Agarre a 90º
Bluetooth
Arduino
Servo Agarre pinza 90º
Dispositivo móvil
Devuelve valor por pantalla

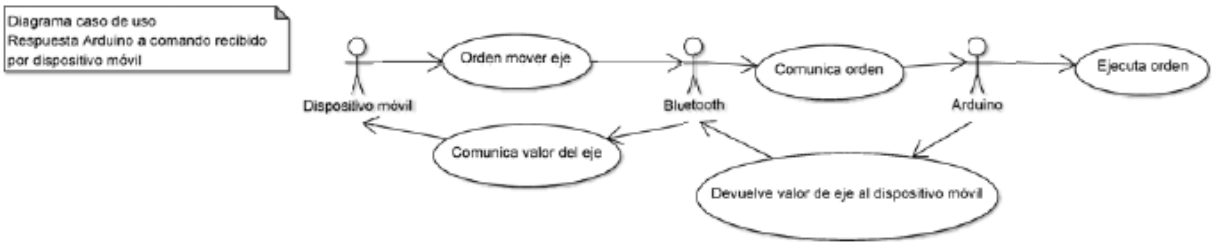


Diagrama caso de uso
Respuesta Arduino a comando recibido por dispositivo móvil
Dispositivo móvil
Orden mover eje
Bluetooth
Comunica orden
Arduino
Ejecuta orden
Comunica valor del eje
Devuelve valor de eje al dispositivo móvil

4.6.2. UML

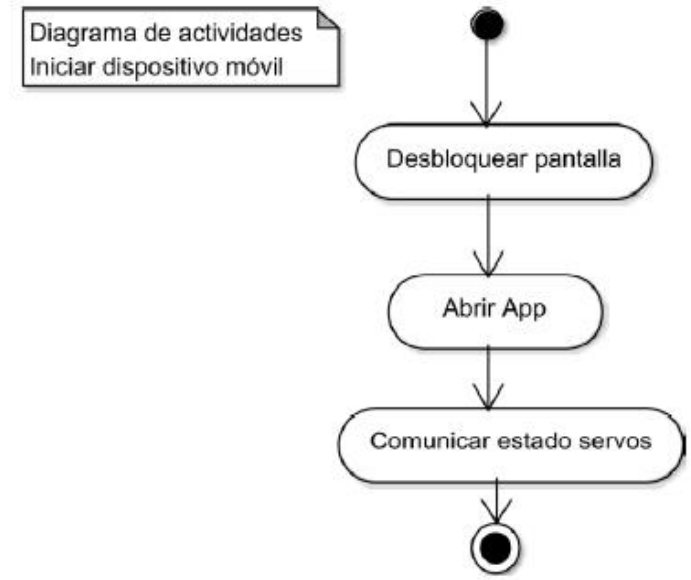
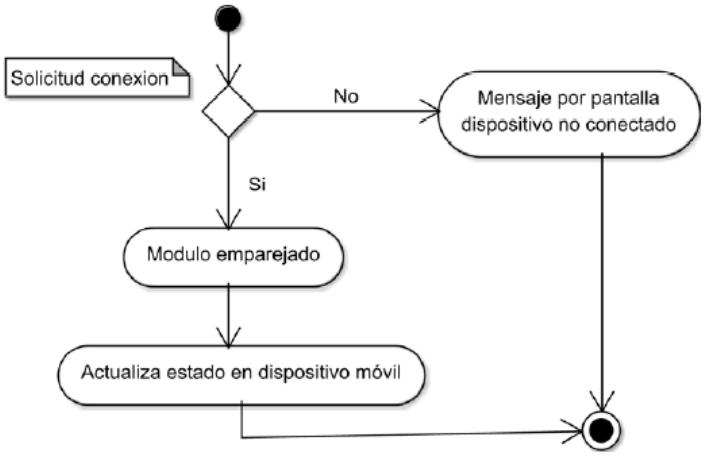


Diagrama de actividades
Iniciar dispositivo móvil
Desbloquear pantalla
Abrir App
Comunicar estado servos


Solicitud conexion
No
Mensaje por pantalla
dispositivo no conectado
Si
Modulo emparejado
Actualiza estado en dispositivo móvil

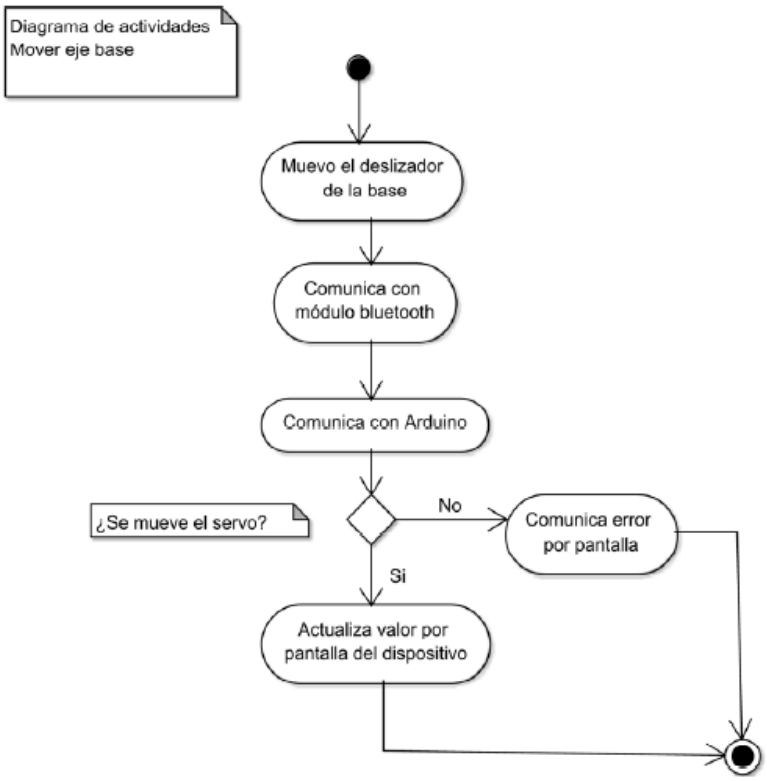


Diagrama de actividades
Mover eje base
Muevo el deslizador de la base
Comunica con módulo bluetooth
Comunica con Arduino
¿Se mueve el servo?
No
Comunica error por pantalla
Si
Actualiza valor por pantalla del dispositivo

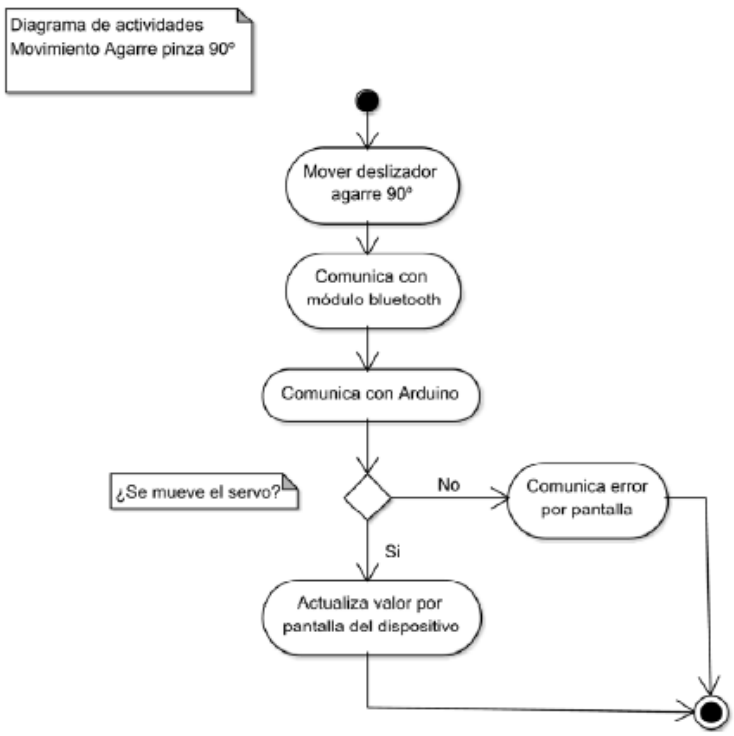


Diagrama de actividades
Movimiento Agarre pinza 90°
Mover deslizador agarre 90°
Comunica con módulo bluetooth
Comunica con Arduino
¿Se mueve el servo?
No
Comunica error por pantalla
Si
Actualiza valor por pantalla del dispositivo

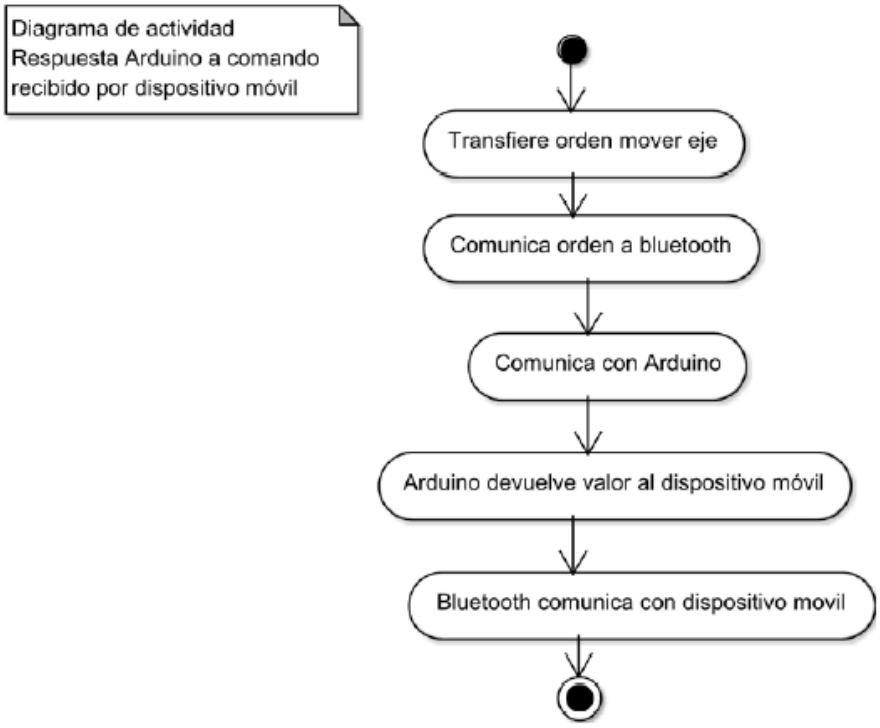


Diagrama de actividad
Respuesta Arduino a comando recibido por dispositivo móvil
Transfiere orden mover eje
Comunica orden a bluetooth
Comunica con Arduino
Arduino devuelve valor al dispositivo móvil
Bluetooth comunica con dispositivo movil

5. CONCLUSIONES

Con respecto a los objetivos marcados al inicio de este proyecto, puedo confirmar que han sido cubiertos, incluso teniendo que añadir objetivos extra debido a los problemas que se presentaron durante la realización de este. Inicialmente era un proyecto que aparentemente solo abarcaba el campo de la electrónica y la programación, pero ha terminado convirtiéndose en un proyecto que, además, ha incluido en gran medida la mecánica, el diseño, y un campo hasta ahora desconocido para mí como es la impresión en 3D.

El principal objetivo que se buscaba como se indica en el título era el diseño electrónico, y como se puede ver ha sido logrado a través de la selección y montaje de los servomotores, placa Arduino, módulo bluetooth y sistema de alimentación interconectando todos ellos, consiguiendo así el primer punto de nuestro proyecto.

El segundo punto que indica nuestro título es el software de control para un brazo robótico de 6 grados de libertad, también ha sido logrado desarrollando una App para el teléfono móvil con la que podemos mover cada grado de libertad de nuestro brazo robótico de forma independiente.

Como objetivo añadido se ha incorporado el rediseño mecánico del prototipo por completo, que ha sido lo que ha abarcado en mayor parte nuestro proyecto, sin este rediseño no hubiera sido posible el montaje físico del prototipo por los problemas expuestos en la memoria.

Como objetivos personales conseguidos desde el comienzo del proyecto hasta ahora, puedo decir que he podido demostrar conocimientos adquiridos durante el grado, como el manejo de Inventor para el rediseño del prototipo mecánico, programación en Arduino para el desarrollo de la App, etc. Pero también cabe señalar que he adquirido nuevas aptitudes durante el proceso del proyecto, sobre todo los conocimientos adquiridos en impresión 3D, técnicas utilizadas, software empleado, y la familiarización con desarrollo de aplicaciones para dispositivos móviles.

5.1. FUTURAS MEJORAS

Como posibles mejoras para el trabajo veo que hay bastantes posibilidades de añadir e implementar, entre ellas terminar de editar el rediseño del prototipo y modificar la base también, para poder introducir los cables por esta, y el Arduino también dentro y tener todo en un solo bloque sin necesidades de tener los cables impidiendo e incomodando los movimientos del brazo robótico, esto desde el punto de vista mecánico.

En cuanto al software para el manejo del brazo robótico creo que hay muchas posibilidades de mejora, se podría diseñar un software que monitorice en todo momento la posición de los servomotores, poder introducir secuencias y memorizarlas en el propio software para después ejecutarlas, añadir nuevos sensores al robot para evitar choques con obstáculos, como comento hay muchas posibilidades que para un posible futuro de TFG se podría aplicar, ya que ahora sí que únicamente tendría que centrarse en el desarrollo de este.

6. BIBLIOGRAFÍA

174.pdf. (s. f.). Recuperado a partir de http://intranet.ceautomatica.es/old/actividades/jornadas/XXIV/documentos/ro/174.pdf

Acosta, L., & Sigut, M. (2005). Matemáticas y robótica, 11.

Baturone, A. O. (2005). *Robótica: manipuladores y robots móviles*. Marcombo.

Cárdenas, M. M., Barrios, P. P., Moreno, K. M. G., Arismendy, J. F. S., & Ordoñez, M. C. (2015). Diseño y Construcción del Prototipo de un Brazo Robótico con Tres Grados de Libertad, como Objeto de Estudio. *Ingeniare*, (18), 87-94.

Cedillo, C., & Raúl, A. (2017). Caracterización mecánica de piezas de PLA fabricadas mediante impresión 3D. Recuperado a partir de http://tesis.ipn.mx:8080/xmlui/handle/123456789/23751

Córdoba-López, A.-J. (2016). Puesta en marcha de brazo robótico y desarrollo de aplicaciones. Recuperado a partir de http://tauja.ujaen.es/jspui/handle/10953.1/3605

Cruceira, R. C. (2017, agosto 2). Conceptos básicos en impresión 3D. Recuperado 21 de junio de 2018, a partir de https://ingenierate.com/2017/08/02/conceptos-basicos-en-impresion-3d/

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE BRAZO ROBÓTICO EN 3D PARA LA MANIPULACIÓN DE CAJAS EN UNA MATRIZ DE ALMACENAMIENTO | Rey Molina | Redes de Ingeniería. (s. f.). Recuperado 18 de junio de 2018, a partir de https://revistas.udistrital.edu.co/ojs/index.php/REDES/article/view/6380/7899

DYOR: Introducción a Fritzing. (s. f.). Recuperado 15 de mayo de 2018, a partir de https://riunet.upv.es/handle/10251/102867

Fritzing Fritzing. (s. f.). Recuperado 15 de mayo de 2018, a partir de http://fritzing.org/home/

guia-iniciacion-app-inventor.pdf. (s. f.). Recuperado a partir de http://codeweek.eu/resources/spain/guia-iniciacion-app-inventor.pdf

López-Para, P. R., & Soto, J. L. (2011). TECNOLOGÍAS ADITIVAS, UN CONCEPTO MAS AMPLIO QUE EL DE PROTOTIPADO RÁPIDO., 14.

PLANIFICACIÓN DE TRAYECTORIAS. (s. f.), 25.

T-027.pdf. (s. f.). Recuperado a partir de https://s3.amazonaws.com/academia.edu.documents/37945579/T-027.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1529606143&Signature=FIqHedLuWW%2BJA0WYxd2VdB0FyL0%3D&response-content-disposition=inline%3B%20filename%3DModelado_de_un_brazo_robotico_de_dos_art.pdf

Thingiverse - Digital Designs for Physical Objects. (s. f.). Recuperado 26 de mayo de 2018, a partir de https://www.thingiverse.com/

Tutorials for App Inventor | Explore MIT App Inventor. (s. f.). Recuperado 4 de abril de 2018, a partir de http://appinventor.mit.edu/explore/ai2/tutorials.html

Relación de documentos

(X) Memoria ..... 60 páginas

(_) Anexos ..... 25 páginas

Planos ..... 26 páginas

La Almunia, a 27 de Junio de 2018

Firmado: José Javier Alonso Montesinos