Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

TRABAJO DE FIN DE GRADO

CONSTRUCCIÓN DE UN BRAZO ROBÓTICO DE BAJO COSTE Y MANIPULACIÓN INALÁMBRICA MEDIANTE TABLET

ALUMNA: CRISTINA NACARÍ TRUJILLO GORRÍN
TUTORA: MARTA SIGUT SAAVEDRA
GRADO: GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA
FECHA: JULIO 2018

Índice

I. Resumen

Este Trabajo de fin de Grado (TFG) consiste en el desarrollo de un prototipo de un manipulador antropomórfico a pequeña escala y de bajo coste, el cual es controlado por una placa Arduino y desde una Tablet Android.

La primera parte del proyecto consistió en la construcción, en madera, de la estructura del manipulador antropomórfico, así como la selección de los servomotores necesarios para mover la estructura.

Una vez construida la estructura procedí a programar la placa Arduino para que se comunique mediante bluetooth con una Tablet Android, que es la encargada de indicar a qué punto se ha de desplazar el manipulador. Conocido dicho punto se aplican las ecuaciones de la cinemática inversa del manipulador para obtener los valores que deben adoptar los ángulos de las articulaciones del brazo.

El siguiente paso en el desarrollo del proyecto consistió en desarrollar por completo la aplicación Android utilizada para comunicarse con la placa Arduino. Una vez comprobado que todas las partes eran completamente funcionales por separado, procedí a unificarlas. En un primer momento comprobé que la placa Arduino controlaba correctamente la estructura, para a continuación añadir la Tablet al sistema.

II. Abstract

This end-of-grade work (TFG) consists of the development of a prototype of a low-cost, small-scale anthropomorphic handler, which is controlled by an Arduino board and from an Android Tablet.

The first part of the project consisted of the construction, in wood, of the structure of the anthropomorphic manipulator, as well as the selection of the servomotors necessary to move the structure.

Once the structure was built, I proceeded to program the Arduino board to communicate via Bluetooth with an Android Tablet, which is responsible for indicating the point where the manipulator has to be moved. Knowing this point, the equations of the inverse kinematics of the manipulator are applied to obtain the values that the angles of the arm joints must adopt.

The next step in the development of the project was to fully develop the Android application used to communicate with the Arduino board. Once I had verified that all the parts were fully functional separately, I proceeded to unify them. I first checked that the Arduino board was controlling the structure correctly, and then added the Tablet to the system.

I. Memoria

La idea de realizar este Trabajo de Fin de Grado surgió durante la realización de las prácticas de la asignatura Ampliación de Sistemas Robotizados, de 4º curso del Grado de Ingeniería Electrónica Industrial y Automática. En dichas prácticas debíamos controlar un manipulador robótico para que hiciera un determinado recorrido según la imagen tomada de la zona de trabajo y procesada por el software matemático MATLAB. Durante esta práctica teníamos limitado el tiempo en el que podíamos disponer del manipulador. La parte del procesamiento de imagen, se podía simular en casa con una webcam y Matlab, pero no se podía comprobar el movimiento del manipulador. Es por esto, pensé que sería de gran utilidad construir mi propio prototipo y poder disponer de él todo el tiempo que quisiera.

Para adecuar esta idea a las características que yo quería para el Trabajo de Fin de Grado, decidí que el manipulador sería controlado desde un sistema Android, en este caso una Tablet BQ M8, para lo que estudié diferentes opciones a la hora de construir la estructura, diferentes tipos de servomotores y diferentes tipos de sistemas de comunicación.

Debido a la premisa de ser un prototipo bajo coste, parte del estudio fue dedicado a analizar diferentes opciones de diseño, materiales y proveedores. Como premisa personal, me propuse que todos los recursos utilizados para la realización de este TFG, desde el diseño elegido hasta el procesador de texto empleado para redactar esta memoria, sea software libre o de uso libre.

1. Desarrollo del prototipo

El desarrollo del proyecto de divide en cuatro partes: diseño del prototipo, desarrollo de la aplicación, programación y ensayos. En las siguientes páginas desarrollaré cada una de ellas.

1.1. Diseño del prototipo

La primera parte de este proyecto se centró en el prototipo que quería construir: Un manipulador antropomórfico de 5 grados de libertad (gdl), con articulaciones de revolución tal y como se describe en Fundamentos de Robótica [1].

En esta fase estudié los distintos diseños y materiales con los cuales podía construir la estructura del manipulador. También realicé una comparación del coste de los distintos elementos mecánicos y eléctricos que conformarían el prototipo. La parte final fue la construcción del manipulador robótico, que aproveché para analizar sus posibles mejoras, las cuales se implementaron en función de su beneficio y coste.

1.1.1. Diseños y materiales

La estructura se ideó para que fuera la más económica posible pero que también supusiera un complemento a la formación adquirida a lo largo de estos años. Debido a estos requisitos me planteé dos posibles materiales: madera y plástico. También me planteé, como último recurso, comprar una estructura de metal.

1.1.1.1. Diseño

Haciendo uso de la web Thingiverse [2], la cual es un portal de diseños digital libres para impresión 3D o cortadoras láser CNC (control numérico por ordenador), me encontré con 2 posibles diseños, que se muestran a continuación:

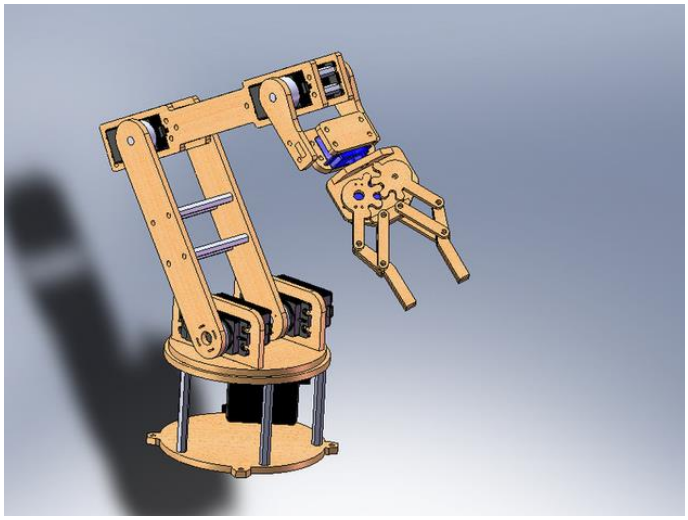


Figura 1: Diseño con 7 servomotores [3].

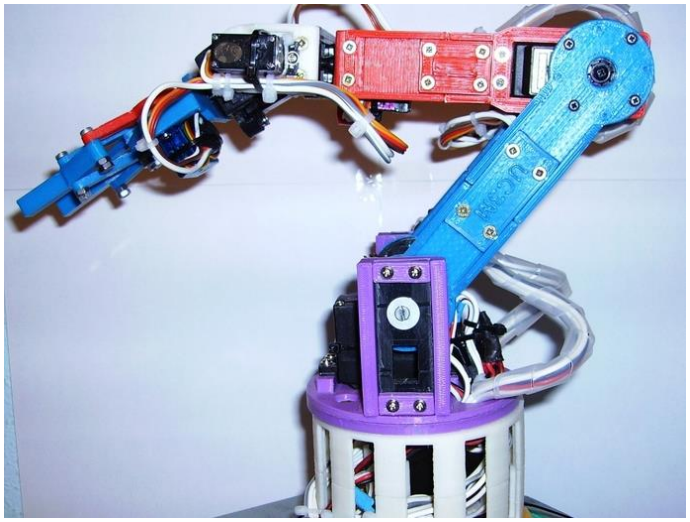


Figura 2: Diseño con 6 servomotores [4].

El diseño que se muestra en la figura 1 se puede fabricar tanto haciendo uso de una impresora 3D como de una cortadora por CNC, pero esta estructura está diseñada para hacer uso de 7 servomotores.

En cambio, la estructura de la figura 2 está diseñada para hacer uso de 6 servomotores, con la limitación de que solo puede ser construida con una impresora 3D.

A continuación, describiré las tres opciones de impresión consideradas: impresión 3D, en madera y en metal, para a continuación justificar la elección realizada.

1.1.1.1.1. Opción 1: impresión 3D

La impresión 3D consiste en realizar piezas con volumen, es decir, en 3 dimensiones, a partir de un diseño asistido por ordenador. Para ello el filamento del material con el que se va a construir la pieza se derrite y se va depositando en una superficie a menor temperatura según unas coordenadas correspondientes al largo (X) ancho (Y) y a la

altura (Z), las cuales son facilitadas por el ordenador, después de transformar el diseño en lenguaje de máquina.

El procedimiento que se describe en [5] lo resumo a continuación:

- Se parte de un modelo original, que puede ser un diseño previo, un boceto o incluso de una pieza real.
- A continuación, esta pieza es pasada a un modelo asistido por ordenador, para lo que se hace uso de un programa de diseño asistido por ordenador (CAD) o de un escáner 3D.
- Por último, la pieza es segmentada en capas según el grosor del filamento (normalmente entre 0,5 mm y 0,7 mm).

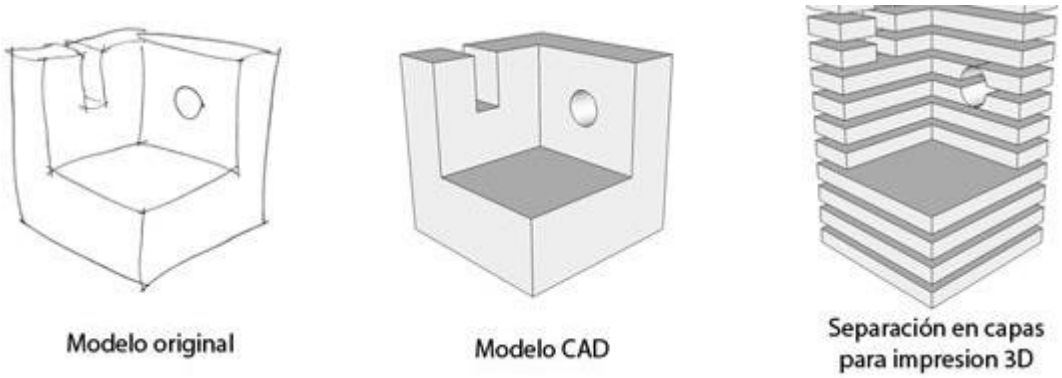


Figura 3: Ejemplo del procedimiento del diseño de una pieza 3D [5]

Una vez segmentado el diseño, se traducen las coordenadas X e Y del diseño en desplazamientos en dichos ejes del estructor (salida el material fundido). Una vez terminada una capa, se procede un desplazamiento en el eje Z para continuar con la capa siguiente.

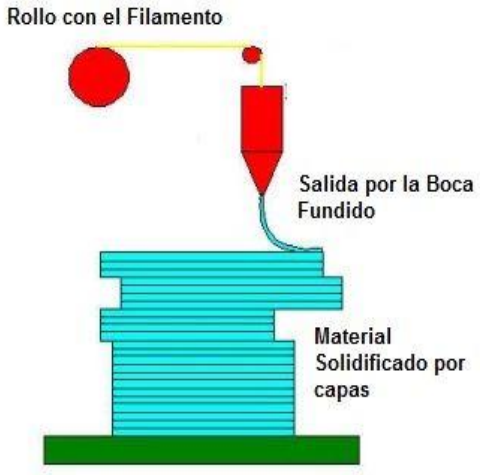


Figura 4: Dibujo funcionamiento de una impresora 3D. [5]

Para estudiar la viabilidad de este sistema y evaluar si era compatible con la especificación de bajo coste del proyecto, pedí presupuesto para imprimir las dos opciones de diseño (ver figuras 1 y 2). El más económico es el servicio de diseño digital de la ULL, que estimó un coste de 60€ por la impresión completa de la estructura de 6 servos y 50€ por la de 7 servos.

1.1.1.1.2. Opción 2: en madera

Realizar la estructura en madera supone limitar el diseño al que hace uso de 7 servomotores, pero es la opción más económica y accesible ya que a través de un familiar dispongo de una cortadora CNC, por lo que el coste sería sólo el de los materiales.

1.1.1.1.3. Opción 3: en metal

Esta opción solo la estudié por si se daba el caso de la imposibilidad de realizar ninguna de las anteriores. En este caso se compraría a través de un portal de compra online una estructura de metal cuyo coste variaba entre los 30€ y 50€, dependiendo del portal de compra.

1.1.1.1.4. Elección

Una vez analizadas las tres opciones de impresión, finalmente opté por realizar la estructura en madera básicamente por motivos económicos, como ya expliqué en el apartado. Para realizar los cortes utilicé una cortadora CNC. El procedimiento fue el siguiente:

- Primero se descargó el diseño en 2D seleccionado. Este diseño viene en formato dxf, que es un formato para diseños asistido por ordenador.
- Posteriormente este diseño se trasformó en lenguaje de programación gCode, que es el más usado en control numérico.
- Una vez transformado el diseño procedí cortar la madera.

En el primer intento utilicé una madera de 5 mm de grosor, pero el láser de la cortadora no era lo suficientemente potente como para atravesar toda la chapa. Por esto, finalmente, opte por utilizar una tabla de 3 mm de espesor. Esos 2 mm de diferencia hacen algo inestable la estructura, pero para la función destinada es aceptable.

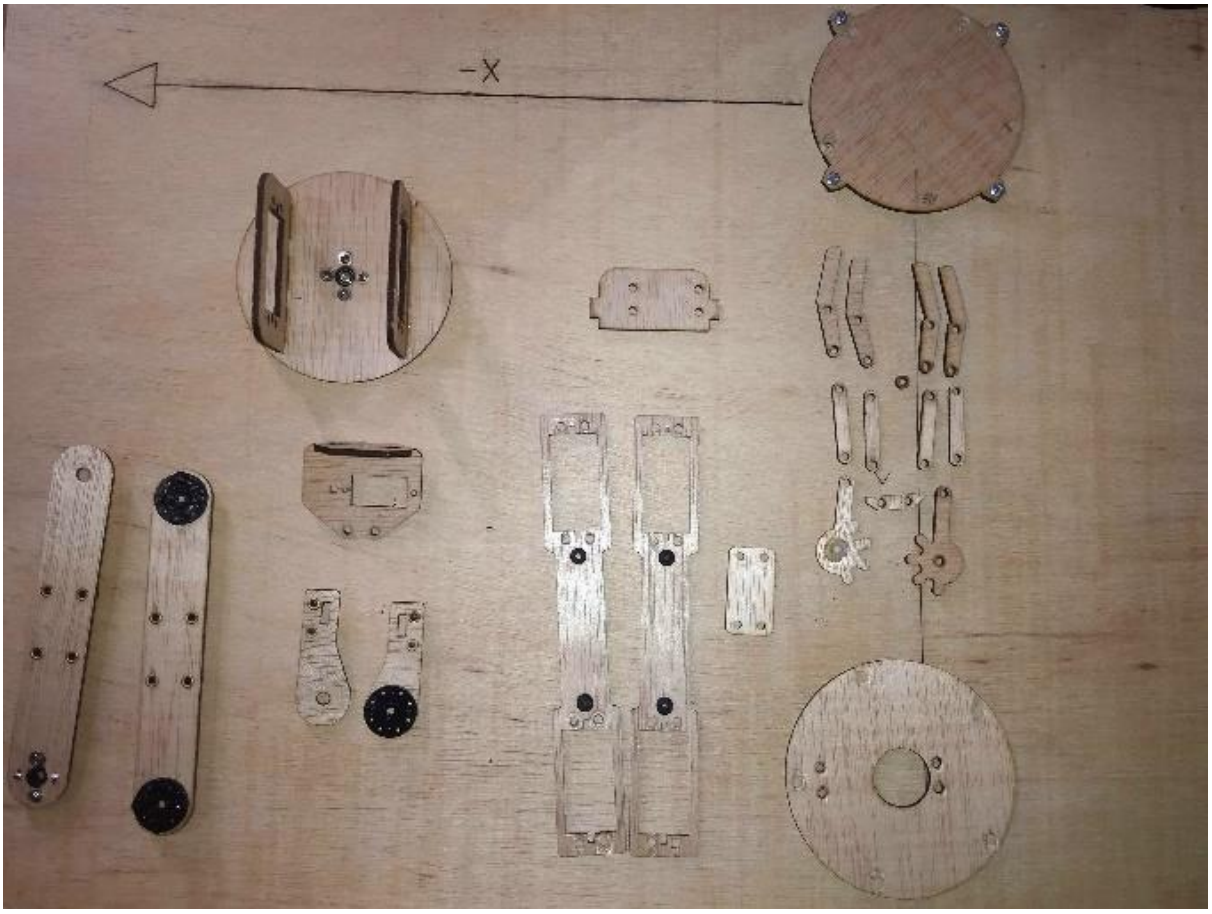


Figura 5: Piezas de la estructura cortadas.

1.1.1.2. Componentes: Servomotores y electrónica

Siguiendo la premisa de bajo coste he optado por componentes de manufactura china, evitando las marcas oficiales, aunque se buscaron también distribuidores españoles para disponer de ellos en un corto periodo de tiempo en caso de urgencia.

1.1.1.2.1. Servomotores

Los servomotores son motores con la capacidad de posicionarse en cualquier lugar dentro de un rango y mantenerse estable. Los que he utilizados son los comúnmente llamados ‘servos’, que son servomotores a menor escala, habitualmente utilizados en modelismo.

Estos servos están formados por un motor de corriente continua, una caja reductora y un circuito de control. A diferencia de los de uso industrial su margen de giro suele estar limitado a 180º, aproximadamente [6].

Su control se basa en la modulación por ancho de pulso (PWM), que consiste en que según el ancho del pulso en alta (según el tiempo en nivel alto), se mueve a una posición u otra. En la figura 6 se ilustra este funcionamiento con un ejemplo.

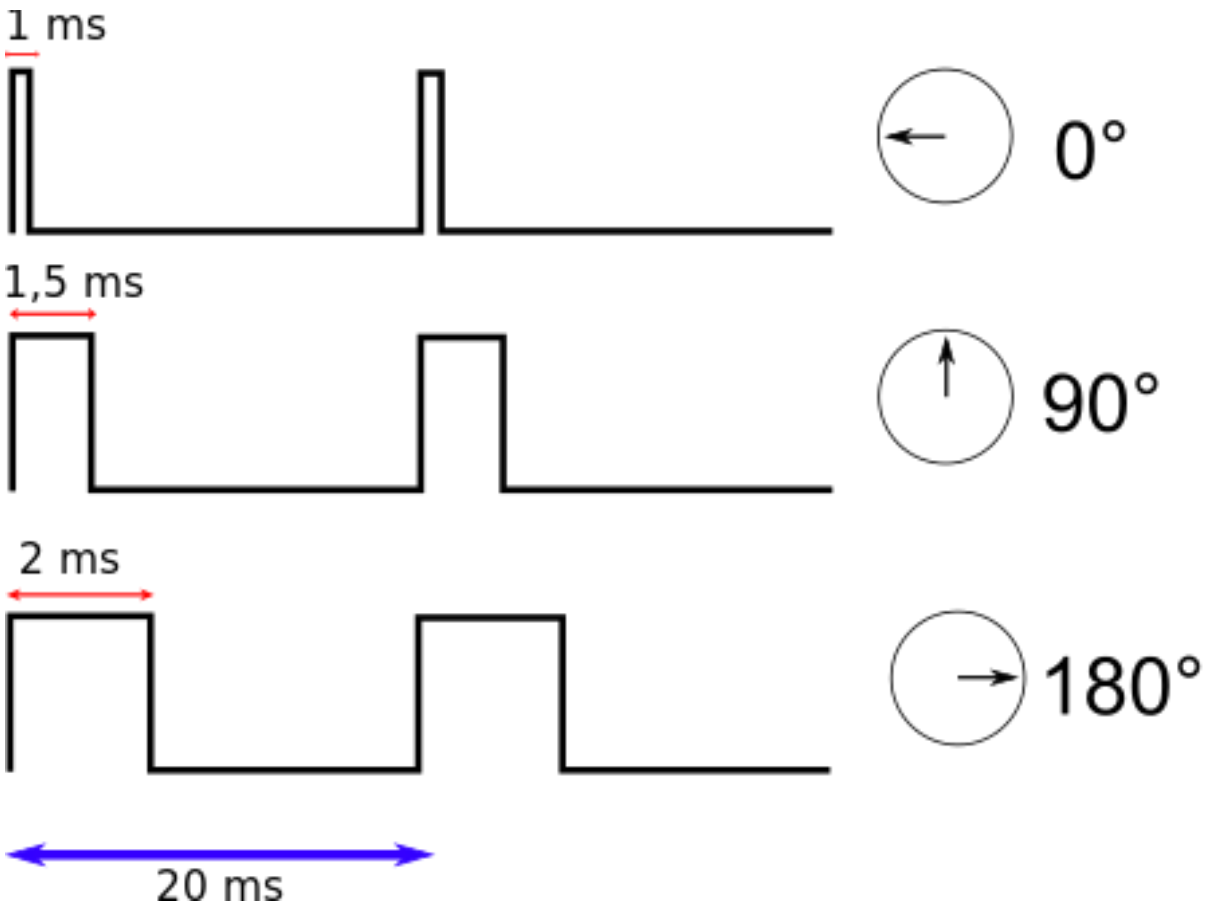


Figura 6: Ejemplo control modular de un servo. [6]

La estructura establece la necesidad de usar 7 servos: 1 para mover la articulación de la base, dos mover para articular el hombro, otro para el codo, dos para articular la muñeca y finalmente uno para abrir o cerrar la pinza.

A la hora de seleccionarlos tuve en cuenta dos factores: par y coste. En la siguiente

tabla se muestra las distintas opciones barajadas incluyendo el precio y el distribuidor.

Modelo	Torque(kg-cm)	Precio	Distribuidor Chino	Distribuidor europeo	Usos
HXT900	1,6	2,88	EBay (china)	Hobby king	Muñeca, muñeca
SG5010	5,5	8,95	EBay (china)	Hobby king	Base, codo, mano
MG956R	10	7,23	EBay (china)	Hobby king	Hombro

Finalmente opté por solo comprar dos modelos: el HXT900 para pinza y muñeca y el MG956R para el resto de la estructura.

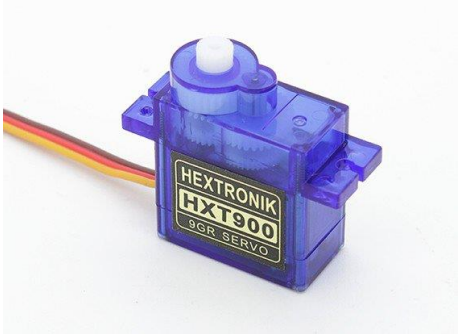


Figura 7: Servomotor HXT900 [7]


Figura 8: Servomotor MG956R[8]

1.1.1.2.2. Arduino

Arduino es una plataforma de "Open-source", por lo que cualquiera puede tomar el diseño original y modificarlo, según sus necesidades y gusto. Por todo ello, a día de hoy hay una gran variedad de placas y módulos.

Para nuestras necesidades haremos uso de la más básica, que está formada por una placa con un microcontrolador y unos puertos de entrada y salida.

Los usos posibles de Arduino con la gran cantidad de módulos compatibles que están disponibles son muy variados y van desde un simple controlador de encendido de un led, hasta el control del riego de un huerto según la humedad del suelo.

En este proyecto he hecho uso de la placa básica de Arduino: Arduino Uno, junto a un módulo bluetooth HC-06.

Figura 9: Arduino Uno [9]

Figura 10: Módulo HC-06[10]

1.1.1.2.2.1. Arduino Uno.

De entre todos los modelos de placas Arduinos, opte por el modelo Arduino Uno, porque me ofrece los puertos necesarios para este proyecto, con los que poder controlar los servos.

1.1.1.2.2.2. Módulo HC-06.

Existen dos modelos de este módulo: HC-05 y HC-06, en la web Promotec [11] indican que mientras a nivel de hardware ambos módulos son iguales se diferencian en el software, donde el modelo HC-06 solo puede actuar como esclavo, limitando así los comandos admitidos, mientras que el modelo HC-05 también puede funcionar como maestro.

1.1.2. Construcción

A la hora de la construcción del manipulador, el primer consistió en comprar todos los materiales necesarios: base, barniz, tornillos, placa perforada, cables, conectores, etc.

Una vez comprado todos los materiales, procedí a barnizar toda la madera y ponerle las patas a la base.

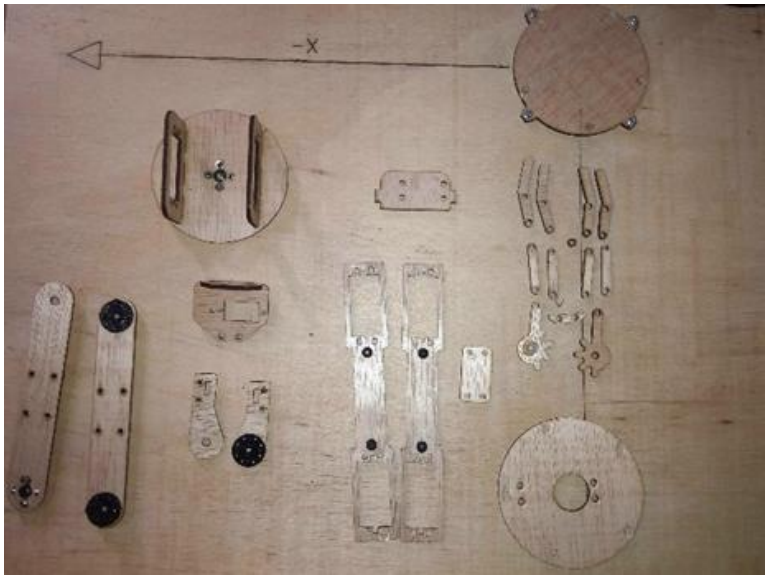


Figura 11: Piezas y base barnizadas.

Posteriormente usé una placa perforada para crear una placa de distribución de tal manera que la tensión de todos los servos partiera de ahí, añadiéndole la placa Arduino.

Figura 12: Placa de conexiones.

Por último, usé un pequeño programa (Anexo I) de Arduino para colocar los servos en la posición inicial (θ = 0º, siendo θ el ángulo del servomotor). Seguidamente comprobé el sentido de giro tomando θ = 180º. Una vez conocidos la posición en θ = 0º y el sentido de giro fui construyendo la estructura.

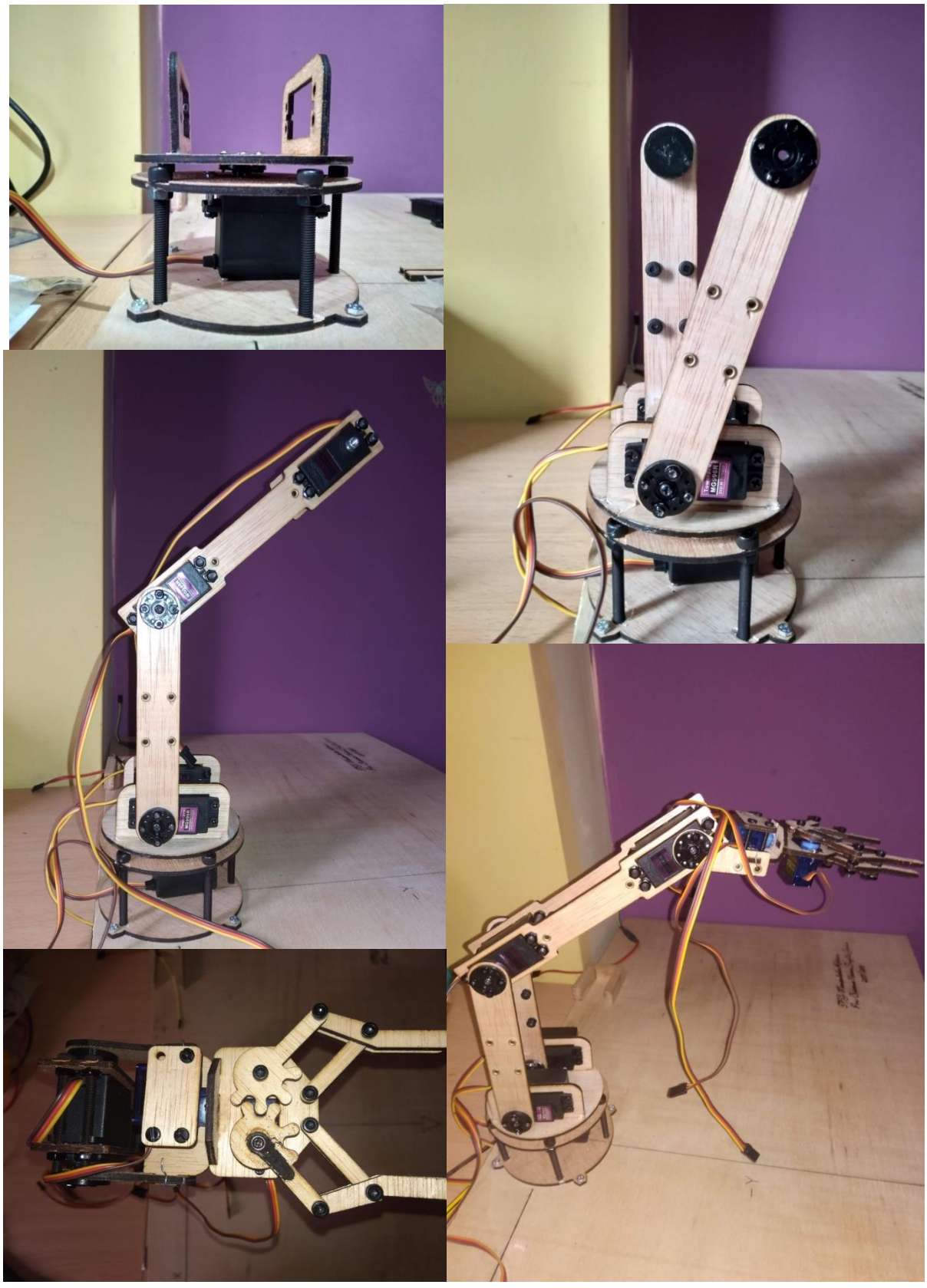


Figura 13: Procedimientos para construir el manipulador.

2. Desarrollo de la aplicación

El desarrollo de la aplicación para el control del brazo robótico desde la Tablet consta de dos partes claramente diferenciadas: el diseño visual y la programación. Ambas partes se realizaron empleando la plataforma MIT APP INVENTOR.

2.1. App Inventor

Para el desarrollo de la app que se ha utilizado para controlar el manipulador utilicé la plataforma desarrollada por el Massachusetts Institute of Technology (MIT) llamada APP Inventor[12].

Figura 14: Logo APP Inventor

Para acceder a la plataforma hay que registrarse, lo que se hace de manera gratuita. Una vez registrado, la interfaz es bastante intuitiva. Se distinguen dos módulos: Diseñador y Bloques. En la figura 15 se observa la pantalla inicial al comenzar una aplicación nueva, donde en el recuadro rojo se observa los dos módulos mencionadas.

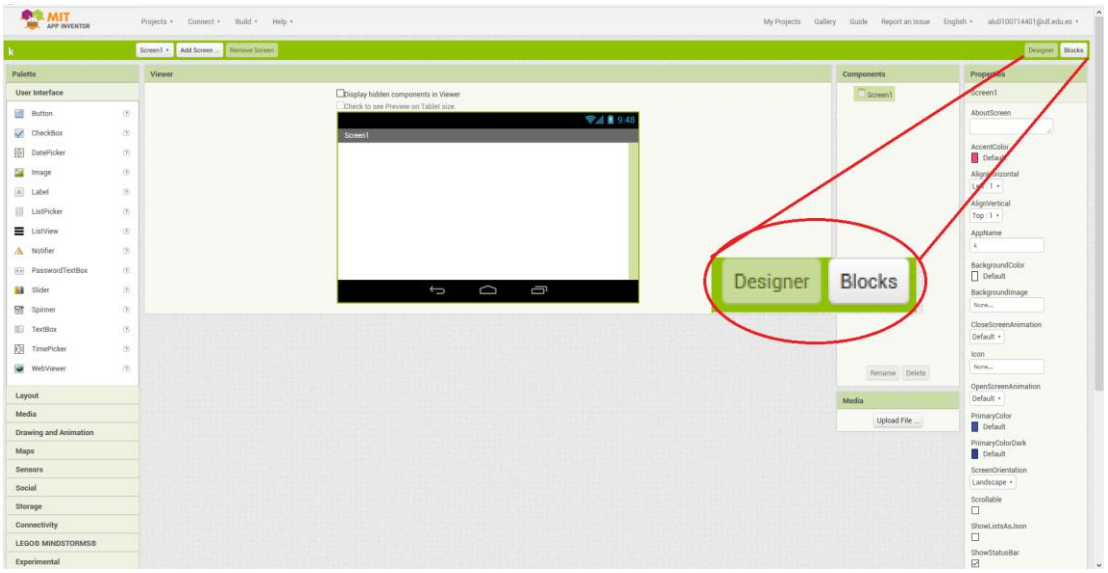


Figura 15: Pantalla de inicio de un proyecto nuevo.

2.1.1. Diseñador

En este apartado se diseña el aspecto visual de la aplicación, para lo que arrastramos los elementos que deseamos desde la parte izquierda a la plantilla.

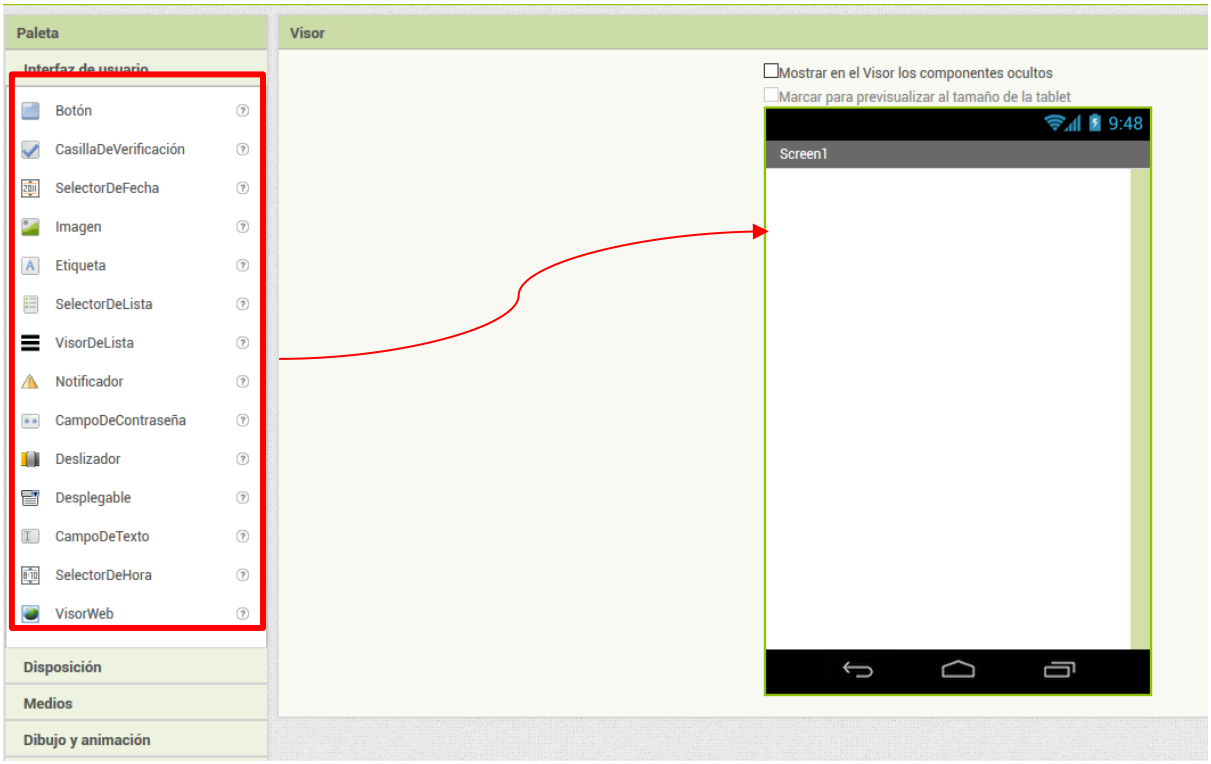


Figura 16: Elementos básicos de diseño.

Una vez seleccionado un elemento sus propiedades se pueden editar en la parte derecha, la sección de edición se puede apreciar en la figura 17.

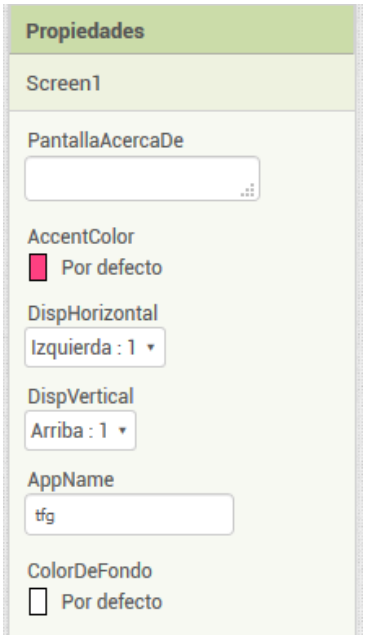


Figura 17 Ejemplo de propiedades editables.

Para este trabajo he hecho uso de los siguientes elementos:

- Botones: bloques que detectan cuándo se presiona.
- Selector de lista: es un botón que al ser presionado despliega una lista de texto definida en la programación.
- Disposición: Conjunto capaz de albergar varios elementos como botones, etiquetas, imágenes, etc. Hay 3 disposiciones: verticales, horizontales y tabular, dependiendo de cómo se quieran colocar los elementos.
- Lienzo: Imagen interactiva que detecta si es presionada y dónde.
- Spirite: Es un objeto que se puede colocar dentro del lienzo e interactúa con este.

2.1.2. Programación por bloques

En esta sección explico cómo he programado la aplicación. Puesto que la plataforma busca ser accesible a la mayor cantidad de personas posible, han optado por la programación mediante bloques, la cual consiste en unir fragmentos de código preescritos como si de un puzle se tratase.

En la figura 18 se presenta un ejemplo de la programación por bloques, en la cual se aprecian los distintos bloques de códigos, diferenciados por colores, que unidos forman un fragmento de código, en este caso, cuando se toca la pantalla táctil, se

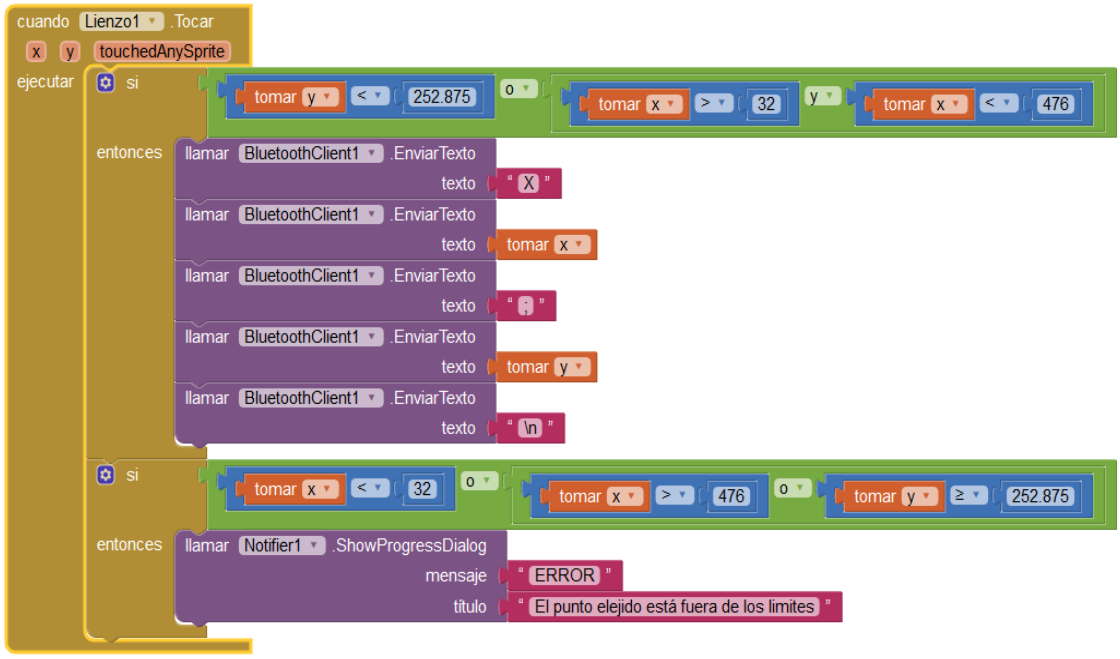


Figura 18: Ejemplo de programación por bloques

evalúan dos condiciones (en que rango se encuentran las coordenadas del punto que se tocó), y según la que sea cierta para ese punto se procede a enviar por bluetooth el punto presionado o mostrar un mensaje de alerta por la pantalla.

2.2. Aplicación

En lo que se refiere a la aplicación desarrollada, inicialmente se planteó de siguiente manera que se muestra en el diagrama de la figura 19:

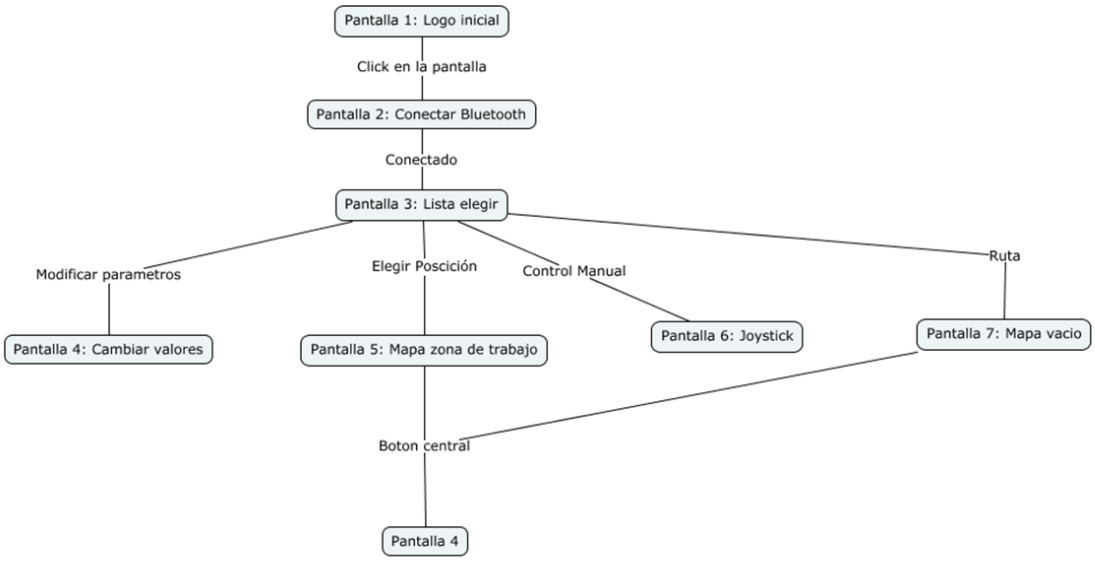


Figura 19: Esquema original.

Debido a que la conexión bluetooth no es estable, ya que en ocasiones al cambiar de pantalla se perdida la conexión bluetooth, por lo que opté por usar las disposiciones en vez de pantallas, lo que permite evitar tener que estar conectando y desconectándose al cambiar de pantalla. También se eliminaron las pantallas 6 y 7 (ver figura 19), optando por usar la pantalla 5 no solo para posiciones conocidas sino también para posiciones elegidas al presionar cualquier punto sobre la pantalla de la Tablet.

2.2.1. Bloque Bluetooth

Para conectar el bluetooth he usado un selector de lista, un botón y el módulo de cliente bluetooth.

El selector de lista es utilizado para desplegar los posibles dispositivos bluetooth a los cuales se puede conectar la aplicación.

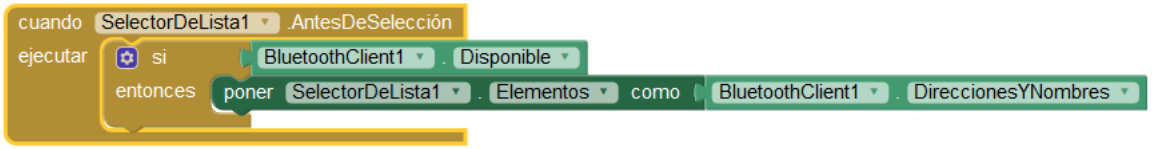


Figura 20: Configurador del selector de lista

Para configurar el selector de lista procedemos de la siguiente manera:
En primer lugar se elige en la barra de la derecha (ver figura 21) el selector de lista con la opción de .AntesDeSelección.

Figura 21: Opciones del selector de lista

Como se aprecia en la figura 21, el bloque remarcado en rojo, dispone de un hueco a la derecha de la palabra “ejecutar”, que es donde colocaremos todo lo que queremos que ocurra antes de que sea seleccionado el selector. En este caso completaremos la lista del selector con los datos del bluetooth, haciendo uso de un condicional, de tal manera que si el bluetooth esta desactivado, no aparece nada en la lista.

Para programar lo que ocurre al elegir una opción de la lista seguimos los mismos pasos que se acaban de describir, pero seleccionando la

opción .DespuésDeSelección, tal y como se observa en la figura 22.

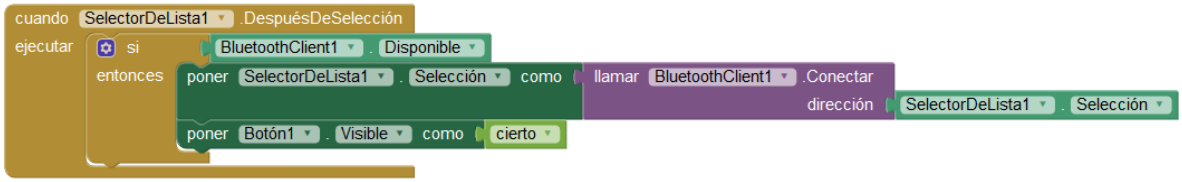


Figura 22: Bloque que establece la conexión Bluetooth.

En este caso utilizamos los bloques del módulo Bluetooth para conectarnos a la opción elegida, a la vez que hacemos visible el botón “Elegir”.

2.2.2. Bloque Visibilidad disposiciones

Para crear el efecto de pasar de una pantalla a otra utilizaremos la función de hacer visible un conjunto de elementos, mientras ocultamos el resto.

Esto es posible si en el diseñador ponemos en cada disposición los elementos que formarán una pantalla, por ejemplo:

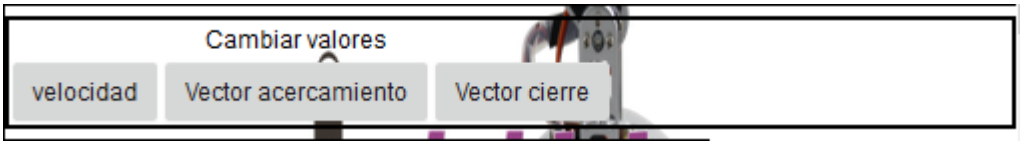


Figura 23: Conjunto de elementos que forman la "pantalla" de opciones.

La programación que efectúa el cambio de visibilidad de cada disposición se construye con el bloque que activa cada cambio, por ejemplo, el botón “velocidad” (Button1), lo que causa que todos los elementos que no sea el conjunto correspondiente al cambio de velocidad (DisposiciónTabular1) se oculten:

Figura 24: Cambio de visibilidad.

El resto de “pantallas” se programan de la misma manera cambiando los estados, según corresponda.

2.2.3. Bloque de envío de los datos introducidos por el usuario

Siguiendo con el ejemplo de la velocidad, la pantalla de diseño solicita al usuario que introduzca un nuevo porcentaje de velocidad:

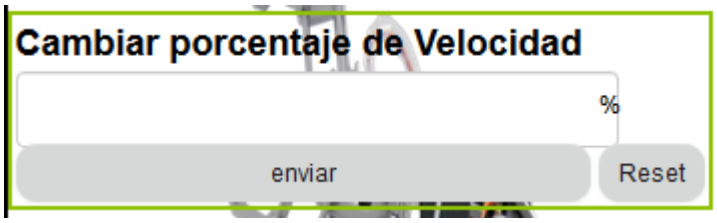


Figura 25: "Pantalla" para cambiar la velocidad.

Una vez se pulsa el botón “enviar” la información introducida es enviada al Arduino a través del bluetooth. Para ello, si la conexión bluetooth está establecida, se procede a ocultar el teclado y enviar una primera letra (en este caso “V”), que posteriormente le servirá al Arduino para saber qué significa el dato siguiente. Inmediatamente después se manda la información introducida, seguida por unos caracteres (“\n”) que el Arduino interpreta como final de la comunicación.

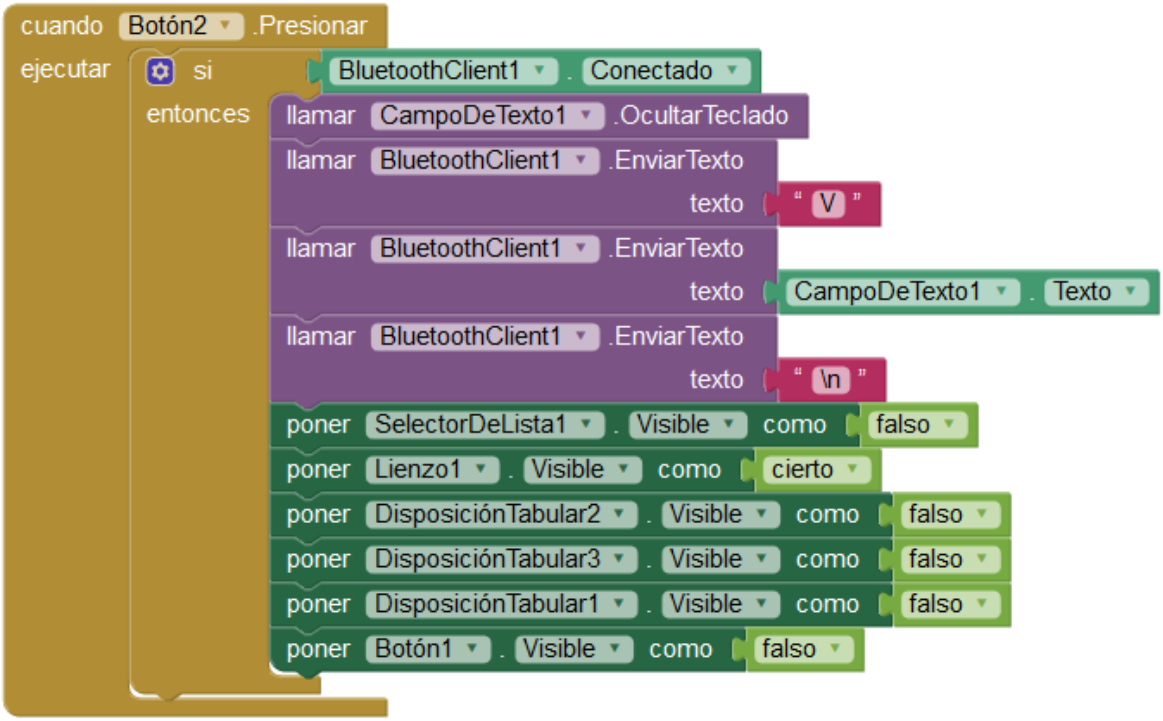


Figura 26: Envío de cambio de velocidad.

Como se puede observar en la figura 26, también se produce el cambio de visibilidad de las pantallas.

Por último, el botón reset está programado de la manera que se muestra en la siguiente figura:

Figura 27: Botón Reset.

En este caso (figura 27), una vez pulsado el botón, se procede a ocultar el teclado e introducir en el cuadro de texto la velocidad predeterminada (“20”). Para poder mandar esta información al Arduino habría que pulsar el botón “enviar”.

El mismo procedimiento se sigue al cambiar el vector de acercamiento o cierre y también el envío de las posiciones preestablecidas (figura 28), obviando en este último caso el cambio de visibilidad y el envío de los datos introducidos.

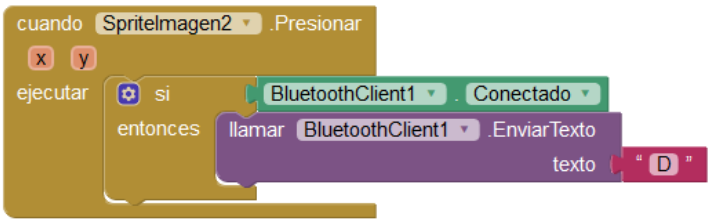


Figura 28: Envío de la posición "D"

Me gustaría destacar el hecho de que en este caso, como se desea también poder mandar posiciones desconocidas, se sustituye el uso del botón por un Sprite, para poder usar el lienzo como disposición.

2.2.4. Bloque Lienzo

En este último bloque hago uso del lienzo, el módulo bluetooth y el módulo de notificación.

Al colocar un lienzo en el diseñador, estamos colocando un elemento que, en nuestro caso, ocupará toda la pantalla de la Tablet. Usamos este elemento en vez de una disposición porque cuando pulsemos en cualquier punto del lienzo, obtendremos inmediatamente las coordenadas X e Y, en píxeles, de dicho punto.

Estas coordenadas se envían al Arduino, haciendo uso de los mismos bloques que en los casos anteriores, con la salvedad de que para identificar qué es lo que se envía se usa la siguiente secuencia:

X	coordenadas en pixel de X	;	Y	coordenadas en pixel de Y

Por tanto, el bloque queda como se muestra a continuación:

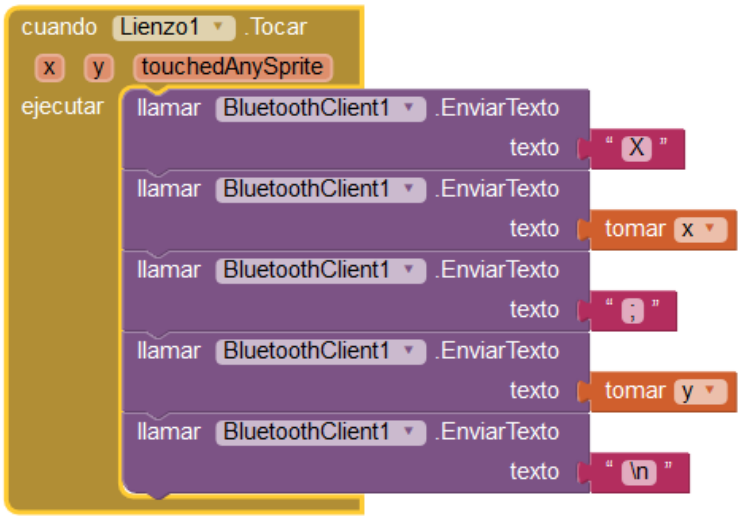


Figura 29: Bloque del lienzo

3. Programación del Arduino

La programación de la placa Arduino se puede dividir en 3 grandes bloques: cinemática inversa, control de los servos y comunicación con la aplicación. A continuación, se explicará con detalle cada uno de estos apartados, que forman el código (Anexo III) en C++ con el cual se programó la placa, así como el entorno de programación que se usó.

3.1. Entorno de programación. IDE Arduino

Las plataformas Arduino tienen su propio entorno de programación, denominado IDE Arduino, en el cual usaré el lenguaje C++ para programar.

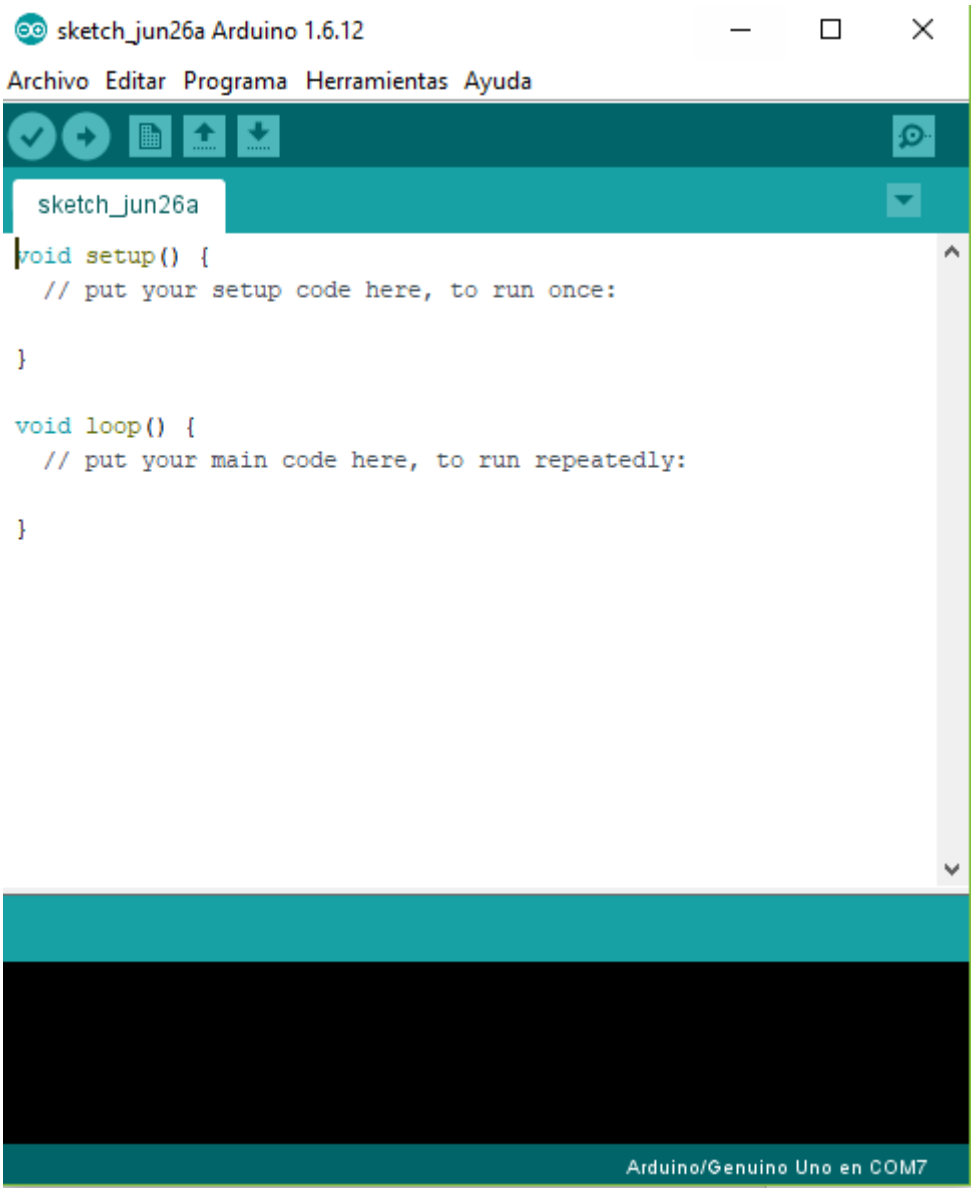


Figura 30: IDE Arduino

A pesar de que las distintas placas Arduino usan un mismo lenguaje de programación para facilitar su uso, se compilan de manera diferente.

Para indicar qué forma de compilación se utiliza, en el entorno hay que seleccionar la placa, tal y como se muestra en la figura 29, y el puerto al que está conectada.

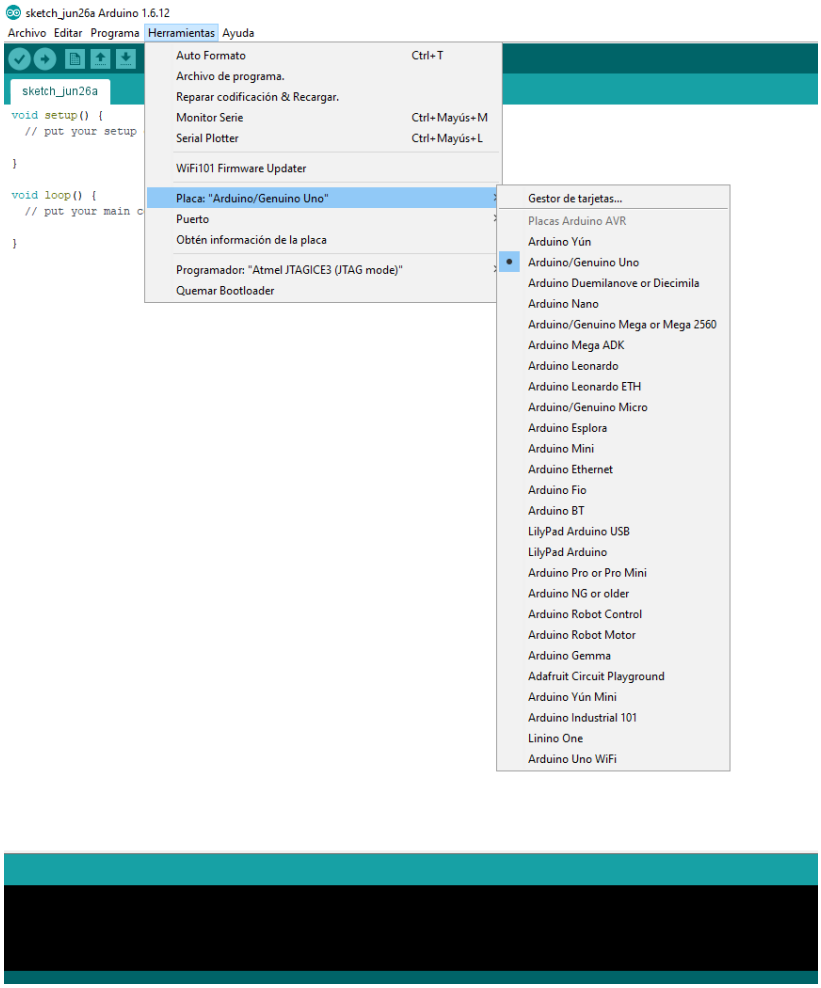


Figura 31: Selección de la placa

Una vez escogida la placa correcta se puede empezar a programar. Por defecto, al empezar un nuevo proyecto aparecen dos bloques de código: Void Setup y Void Loop. Es en estos dos bloques donde se suele incluir la mayoría de la programación.

3.1.1. Void setup

Todas las líneas de código escritas dentro de este bloque solo se utilizarán una vez al iniciar la placa. Es aquí donde se suelen asignar los pines de las variables deseadas o indicar si actúan como entradas o salidas.

3.1.2. Void loop

Al contrario que el bloque anterior, este actúa como un bucle, de tal manera que las líneas de código que contenga se ejecutarán continuamente. Es en este bloque donde se leen las entradas y se actúa sobre las salidas.

3.2. Control de los servos

Al usar el lenguaje C++ para programar disponemos de una gran variedad de librerías, que son conjuntos de líneas de código que evitamos incluir de manera repetida usando las asignaciones “keywords” que establecen.

Para controlar los servos hemos optado por utilizar la librería Varspeedservo, la cual nos permite controlar hasta 8 servos y variar la velocidad de cada uno por separado.

Esta librería nos establece que primero debemos asignar un nombre a cada servo que vayamos a usar, lo cual haremos después de incluir la librería, pero antes del bloque setup. Tal y como se muestra en la figura 32, la forma de declarar un servo es: VarSpeedServo nombreServo;

```
/*control motores
9/12/2017
cristina Nacarí Trujillo Gorrín
*/
#include <VarSpeedServo.h>

VarSpeedServo base;                    // servo de rotacion continua para mover la base.
VarSpeedServo hombroD;                 // servo de rotacion limitada derecho del hombro.
VarSpeedServo hombroI;                 // servo de rotacion limitada izquierdo del hombro.
VarSpeedServo codo;                    // servo de rotacion limitada que simula el codo.
VarSpeedServo MGiro;                   // servo de rotacion limitada encargado de girar la muñeca.
VarSpeedServo MInclinacion;            // servo de rotacion limitada encargado de inclinar la muñeca.
VarSpeedServo pinza;                    // servo de rotacion limitada que abre o cierra la pinza.

void setup() {

  }

void loop() {

}
```

Figura 32: Declaración de cada servo

En el bloque Setup asignamos a cada servo un pin digital de la placa arduino, para lo que usamos la cadena: nombreServo.attach(pinArduino);

```
void setup() {
  base.attach(13);
 hombroD.attach(12);
 hombroI.attach(11);
 codo.attach(10);
 MInclinacion.attach(9);
 MGiro.attach(8);
 pinza.attach(7);

 }
```

Figura 33: Asignación de los pines

Por último, para mover los servos usamos: nombreServo.write(posEnAngulos);. En este caso estas líneas de código van en el bloque Loop.

```
void loop() {
  base.write(0);
  hombroI.write(90);
  hombroD.write(90);
  codo.write(0);
  MInclinacion.write(180);
  MGiro.write(90);
  pinza.write(0);
}
```

Figura 34: Mover los servos.

Con estas líneas podemos controlar los servos de manera independiente. Para modificar la velocidad, bastaría con poner una coma seguida de un valor entre 0 y 255 (siendo 255 máxima velocidad) después de la posición del servo, por ejemplo: base.write(0,125);

3.3. Cinemática Inversa

La cinemática inversa consiste en obtener los ángulos correspondientes a cada articulación del manipulador a partir de las coordenadas cartesianas del punto en el que queremos que se posicione el elemento final del brazo, las longitudes de los eslabones y de los vectores de cierre y acercamiento.

En este caso, el manipulador dispone de 7 elementos rotativos: uno encargado de girar la base (θ_1); dos de mover el hombro, como el valor de ambos se considera igual, a efectos de cálculo solo tendremos en cuenta uno (θ_2); uno en el codo (θ_3); dos en

la muñeca, que tiene distinto eje de giro y por tanto denomino (θ_4 y θ_5) y por último el encargado de abrir y cerrar la pinza, que a efecto de la cinemática inversa no influye.

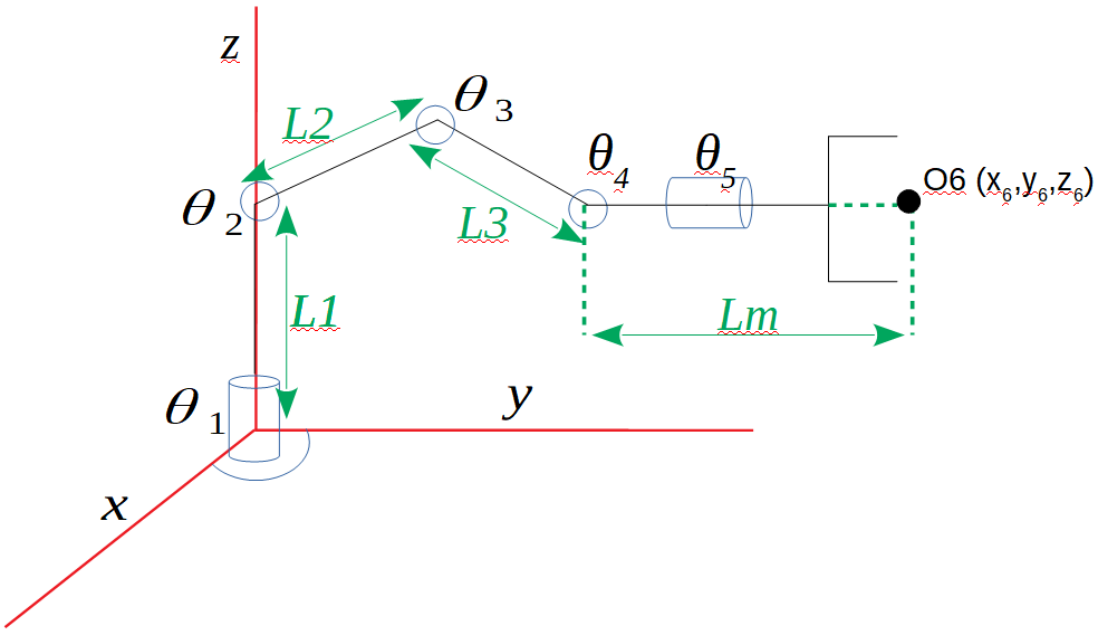


Figura 35 Estructura completa.

3.3.1. Cálculo de la cinemática inversa de los 3 primeros grados de libertad

Los tres primeros grados de libertad del manipulador corresponden a las articulaciones de la cintura (la base del robot), el hombro y el codo. Para calcular estos ángulos realizo el procedimiento descrito en [1]. Aquí se indica cómo calcular los ángulos θ_2 y θ_3, es decir fijando el valor de θ_1.

Pero como no conozco las coordenadas del punto O_3 sino las del punto O_6 (figura 35), así como las longitudes de los eslabones, y los vectores de acercamiento y cierre, por lo que puedo obtener dicho punto.

$$cos(\theta_3) = \frac{x^2 + y^2 - l_2^2 - l_3^2}{2 l_2 l_3} \qquad (1)$$

$$\theta_2 = \phi - \alpha \begin{Bmatrix} tan(\phi) = \frac{y}{x} \\ tan(\alpha) = \frac{l_3 \cdot sin(\theta_3)}{l_2 + l_3 \cdot cos(\theta_3)} \end{Bmatrix} \qquad (2)$$

Al añadir θ_1 la estructura pasa de estar en el plano (2D) a estar en el espacio(3D).

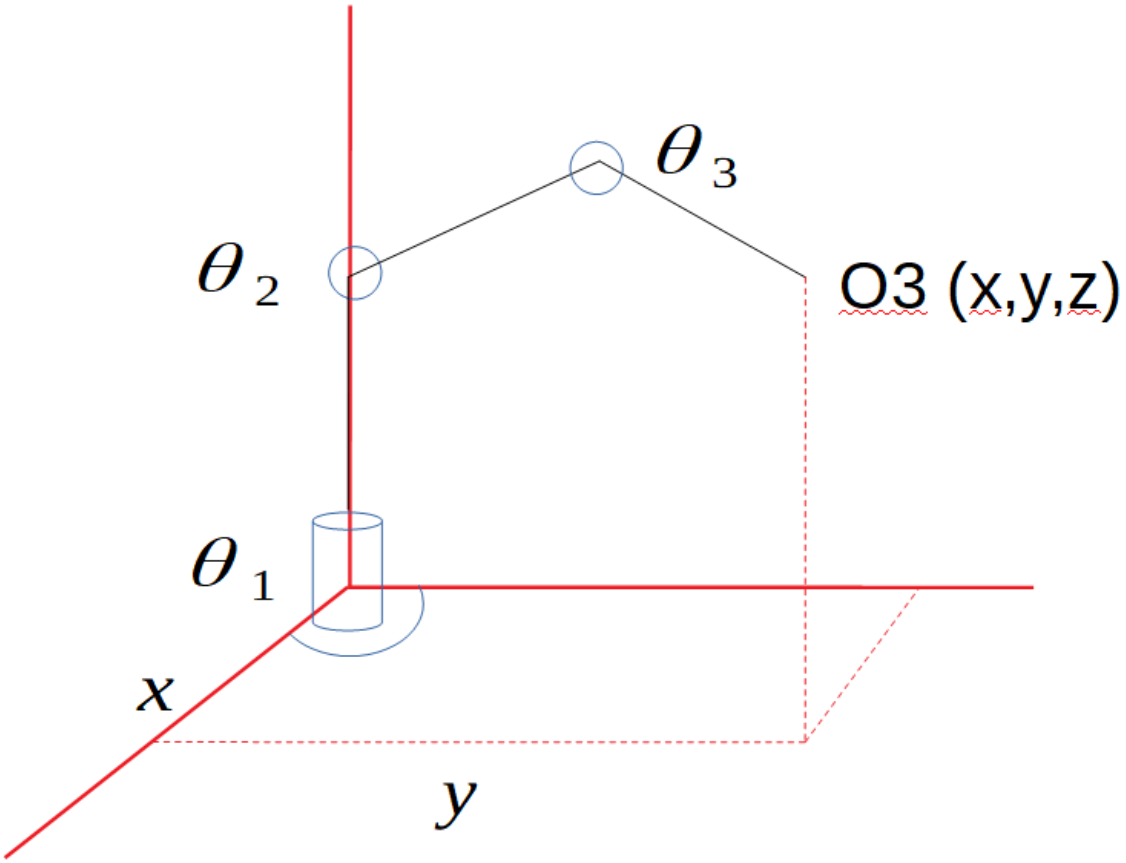


Figura 36 Ángulos de las 3 primeras articulaciones

A partir de las coordenadas (x,y) del punto O3 se calcula el ángulo θ_1:

$$tan(\theta_1) = \frac{y}{x} = arctan\left(\frac{y}{x}\right) \quad (3)$$

Podemos obtener las coordenadas del punto O3 (ver figura 35) a partior de las del punto O6 (donde queremos posicionar el punto final del manipulador), de la longitud de la muñeca y del vector de acercamiento, datos todos ellos conocidos, de la forma que se muestra a continuación:

$$\overrightarrow{O_3} = \overrightarrow{O_6} - \mathrm{Lm} \cdot \vec{a} \quad (4)$$

A continuación, generalizamos al espacio las expresiones para θ2 y θ3 ya obtenidas para el plano realizando los siguientes cambios de variables:

Caso 2D	Caso 3D
x	$\sqrt{x^2 + y^2}$
y	z-l_1

Tabla 1: De cambio de variables

Al realizar este cambio las fórmulas quedarían:

$$cos(\theta_3) = \frac{x^2 + y^2 + (z - l_1)^2 - l_2^2 - l_3^2}{2 l_2 l_3} \quad (5)$$

$$\theta_2 = \phi - \alpha \begin{Bmatrix} tan(\phi) = \dfrac{z - l_1}{\sqrt{x^2 + y^2}} \\ tan(\alpha) = \dfrac{l_3 \cdot sin(\theta_3)}{l_2 + l_3 \cdot cos(\theta_3)} \end{Bmatrix} \quad (6)$$

Obtenidos ya los tres primeros ángulos (θ_1, θ_2, θ_3), veamos cómo calcular los restantes. Para ello, hacemos uso de la matriz de transformación del punto 5 al punto 0. Si tenemos las coordenadas de un punto en el espacio referidas al punto O5 y las multiplicamos por esta matriz, obtenemos las coordenadas del mismo punto referidas ahora al punto O0, es decir, la base de nuestro manipulador. Esta matriz tiene la forma que se muestra a continuación:

$$o_{T_5} = \begin{bmatrix} s_1 & n_1 & a_1 & p_1 \\ s_2 & n_2 & a_2 & p_2 \\ s_3 & n_3 & a_3 & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Donde el vector $\vec{s}$ es el vector de cierre, el vector $\vec{a}$ es el de acercamiento y por último el vector $\vec{n}$ es un vector normal a los dos anteriores, y por tanto conocidos.

La submatriz formada por los tres vectores mencionados antes es la que contiene la información de las rotaciones por lo que es esa la que utilizare.

$$0_{T_5} = 0_{R_3} \cdot 3_{R_5} = [\vec{s}\vec{n}\vec{a}] \quad (8)$$

Esta expresión sale de descomponer la rotación del punto O5 al punto O0 como el producto de la rotación del punto O5 al punto O3 y la rotación del punto O3 al punto O0.

Entonces:

$$3_{R_5} = \left(3_{R_5}\right)^{-1} \cdot [\vec{s}\,\vec{n}\,\vec{a}] = A \quad (9)$$

Donde además sabemos que:

$$3_{R_5} = \begin{bmatrix} C_4 & 0 & S_4 \\ S_4 & 0 & -C_4 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} C_5 & -S_5 & 0 \\ S_5 & C_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

Para abreviar denominamos el coseno de θ_4 como C_4 y el seno como S_4, utilizando dicha abreviación también para θ_5.

$$3_{R_5} = \begin{bmatrix} \dots & \dots & S_4 \\ \dots & \dots & -C_4 \\ S_5 & C_5 & 0 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \quad (11)$$

Con lo que ya puedo obtener θ_4 y θ_5, obteniendo la magnitud del ángulo del coseno y el signo del seno:

$$\theta_4 = signo(A_{13}) \cdot arccos(-A_{23}) \quad (12)$$

$$\theta_5 = signo(A_{31}) \cdot arccos(A_{32}) \quad (13)$$

Por último, para el último grado de libertad (θ_6), que corresponde con la apertura y cierre de la pinza, utilizo una condición, de manera de que si la pinza está abierta se cierre y viceversa.

3.3.1.1. Problemas al programar en c++

A la hora de la programación, debemos tener en cuenta que en C++ tenemos la limitación de no poder realizar operaciones con matrices de manera directa, como ocurre en otros lenguajes como Matlab. Por ello debemos realizar los cálculos de la misma forma que lo haríamos en papel, lo que aumenta considerablemente las líneas de código.

```
float transpuesta[][3] = {
                         {adjunta[0][0], adjunta[1][0], adjunta[2][0]},
                         {adjunta[0][1], adjunta[1][1], adjunta[2][1]},
                         {adjunta[0][2], adjunta[1][2], adjunta[2][2]}
};
```

Figura 37: Traspuesta de una matriz en C++

3.3.2. Comunicación Bluetooth

Para la comunicación bluetooth debemos hacer uso de los pines seriales de la placa Arduino, por defecto los pines digitales 0 y 1.

Lamentablemente dichos pines son los utilizados durante la comunicación con el ordenador. La solución a este problema es utilizar la librería SoftwareSerial, que permite utilizar cualquier otro pin digital como pin serial.

Para implantar esta librería se sigue el mismo procedimiento que con la librería VarSpeedServo (figura 38), determinando primero cuales son los nuevos pines seriales, e indicando posteriormente la velocidad de la comunicación.

```
#include <SoftwareSerial.h>
#include <VarSpeedServo.h>                    // libreria para controlar los servos.

SoftwareSerial BT(2, 3);
void setup()
{
  BT.begin(9600);
```

Figura 38: Implementación de la librería

La comunicación con la Tablet parte del hecho de que la información se envía siguiendo la estructura de: Letra identificativa - Valores (separados por comas) - fin de comunicación. Por ejemplo: A0,1,0/n, donde A identifica el vector de acercamiento (ver Anexo II).

El procedimiento para procesar la información es el siguiente:

- Recibe toda la secuencia como un String, es decir, un conjunto de caracteres.
- Identifica el primer carácter y, según el que sea, se cumple la condición correspondiente. En el caso del ejemplo anterior (A0,1,0/n), se cumple la condición de que se va a cambiar el vector de acercamiento.
- Separa los valores por el símbolo separador, en este caso la coma.
- Cada fragmento de String, subString, es transformado en un numero entero.
- Cada número es asignado a la variable correspondiente.

```
if (cadena.charAt(0) == 'A') {
  int posComa1 = cadena.indexOf(',');
  int posComa2 = cadena.indexOf(',', posComa1 + 1);
  String a1 = cadena.substring(1, posComa1);
  String a2 = cadena.substring(posComa1 + 1, posComa2);
  String a3 = cadena.substring(posComa2 + 1);

  a[0] = a1.toInt();
  a[1] = a2.toInt();
  a[2] = a3.toInt();

  a1.remove(0);
  a2.remove(0);
  a3.remove(0);
  cadena.remove(0);

}
```

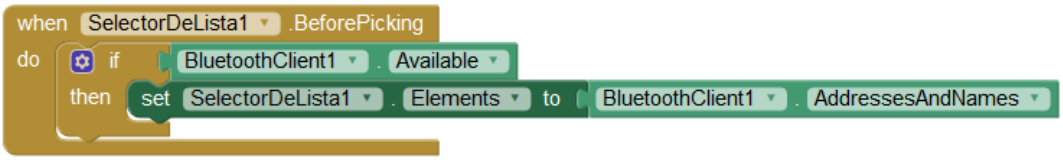
Figura 39: Procedimiento para cambiar el vector de acercamiento

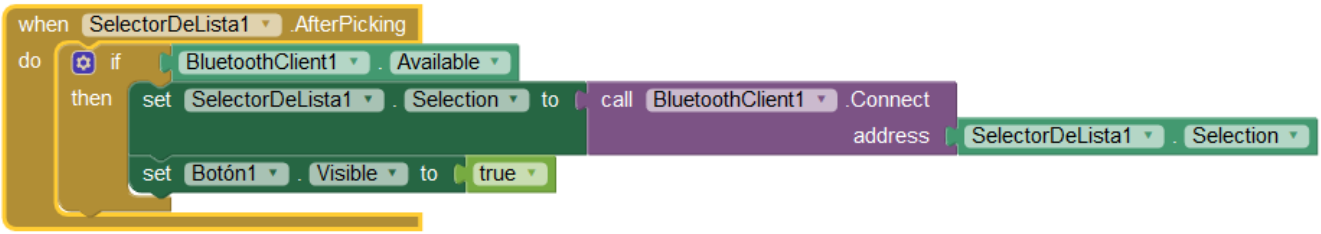
ANEXO I. Programa para posicionar servos

```
/*control motores
 9/12/2017
 cristina Nacarí Trujillo Gorrín
*/
#include <VarSpeedServo.h>

VarSpeedServo base; // servo de rotación continua para mover la base.
VarSpeedServo hombroD; // servo de rotación limitada derecho del hombro.
VarSpeedServo hombroI; // servo de rotación limitada izquierdo del hombro.
VarSpeedServo codo; // servo de rotación limitada que simula el codo.
VarSpeedServo MGiro; // servo de rotación limitada encargado de girar la
mu?ca.
VarSpeedServo MInclinacion; // servo de rotación limitada encargado de
inclinar la mu?ca.
VarSpeedServo pinza; // servo de rotación limitada que abre o cierra la
pinza.

void setup() {
 // asignación de pines a cada servo.
 base.attach(13);
 hombroD.attach(12);
 hombroI.attach(11);
 codo.attach(10);
 MInclinacion.attach(9);
 MGiro.attach(8);
 pinza.attach(7);
 // mover cada servo al ángulo indicado
 base.write(0);
 hombroI.write(90);
 hombroD.write(90);
 codo.write(0);
 MInclinacion.write(180);
 MGiro.write(90);
 pinza.write(0);

}

void loop() {

}
```

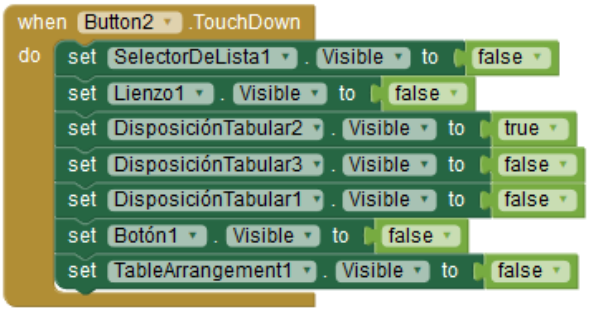
ANEXO II. Programación App

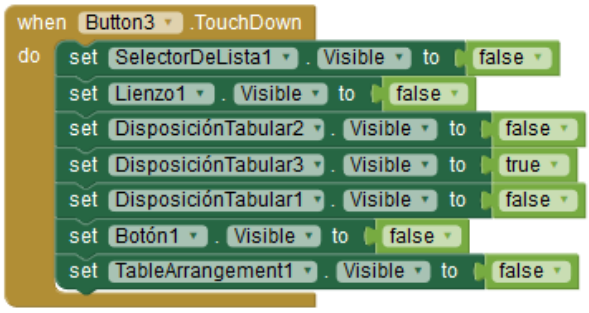
- Conexión bluetooth

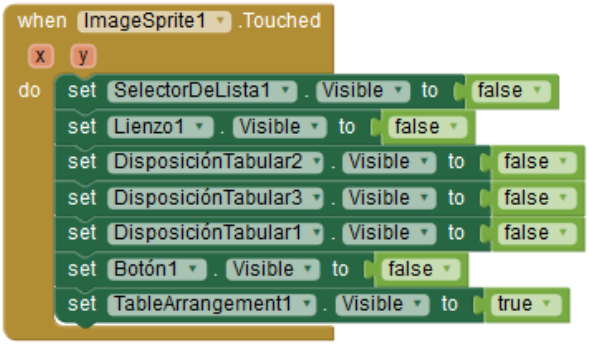


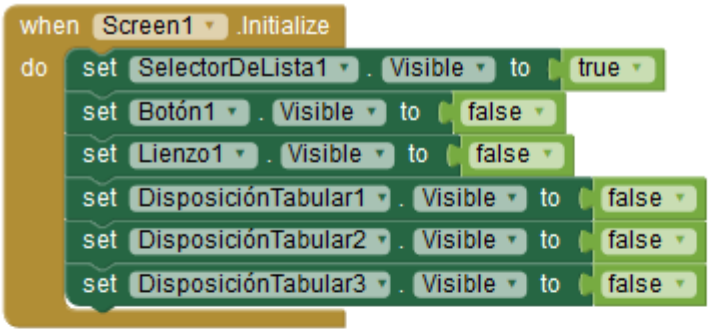


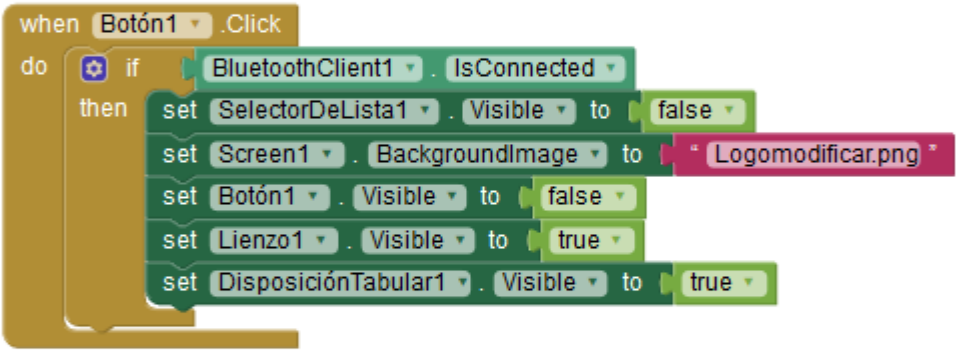
- Visualización / ocultación de elementos.

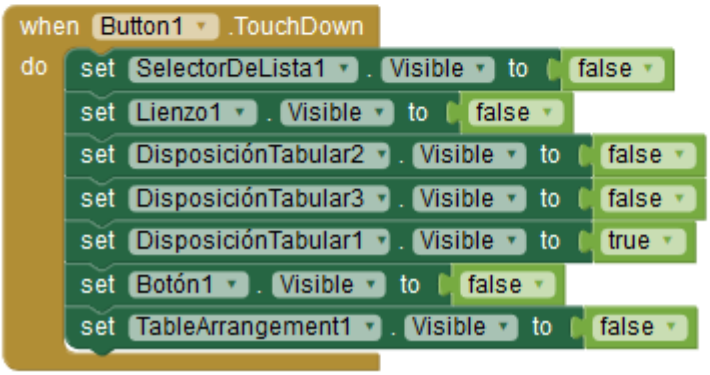




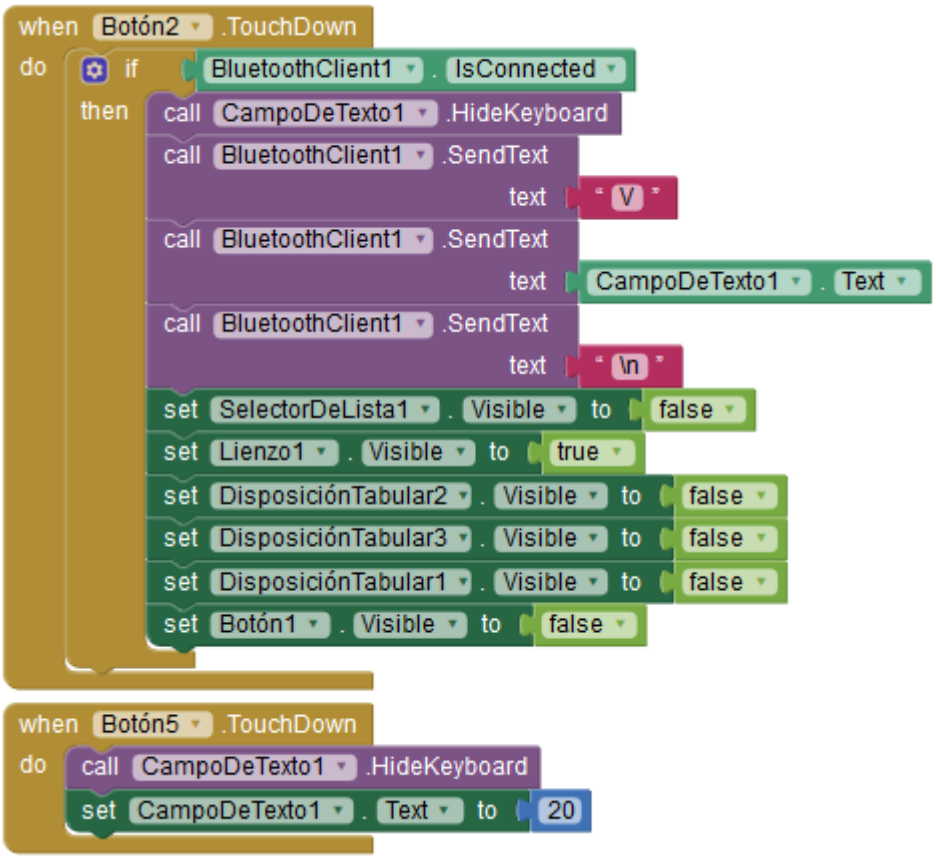



when Screen1 .Initialize
do
set SelectorDeLista1 . Visible to true
set Botón1 . Visible to false
set Lienzo1 . Visible to false
set DisposiciónTabular1 . Visible to false
set DisposiciónTabular2 . Visible to false
set DisposiciónTabular3 . Visible to false

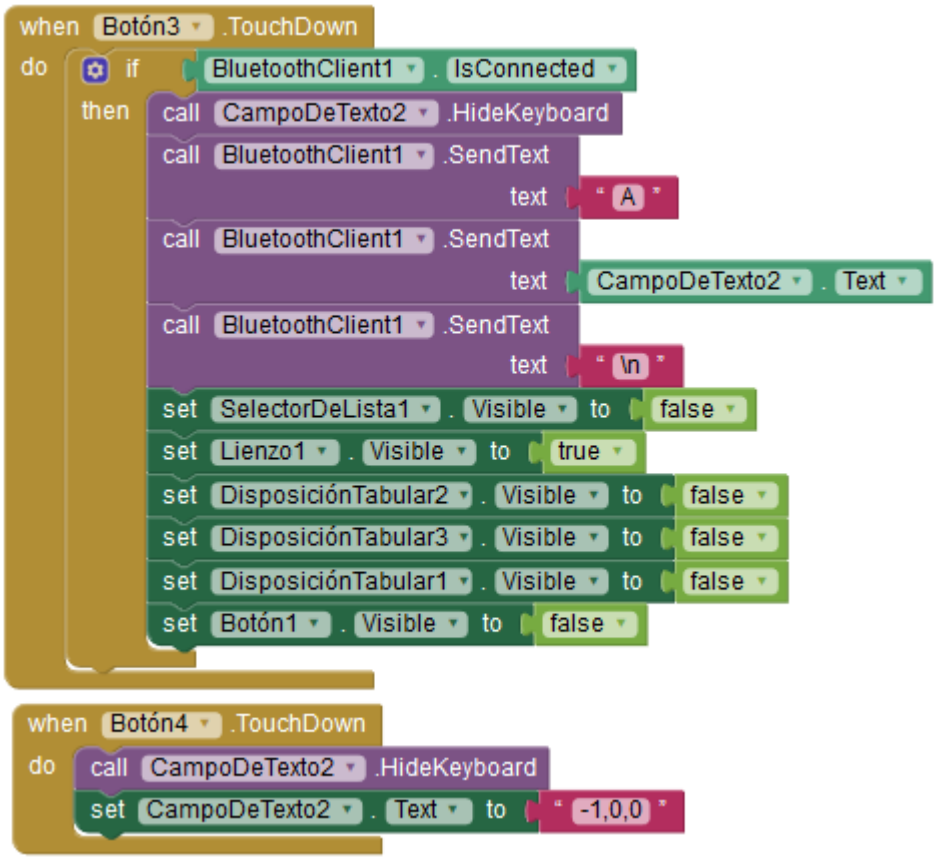

when Botón1 .Click
do
if BluetoothClient1 . IsConnected
then
set SelectorDeLista1 . Visible to false
set Screen1 . BackgroundImage to " Logomodificar.png "
set Botón1 . Visible to false
set Lienzo1 . Visible to true
set DisposiciónTabular1 . Visible to true


when Button1 .TouchDown
do
set SelectorDeLista1 . Visible to false
set Lienzo1 . Visible to false
set DisposiciónTabular2 . Visible to false
set DisposiciónTabular3 . Visible to false
set DisposiciónTabular1 . Visible to true
set Botón1 . Visible to false
set TableArrangement1 . Visible to false

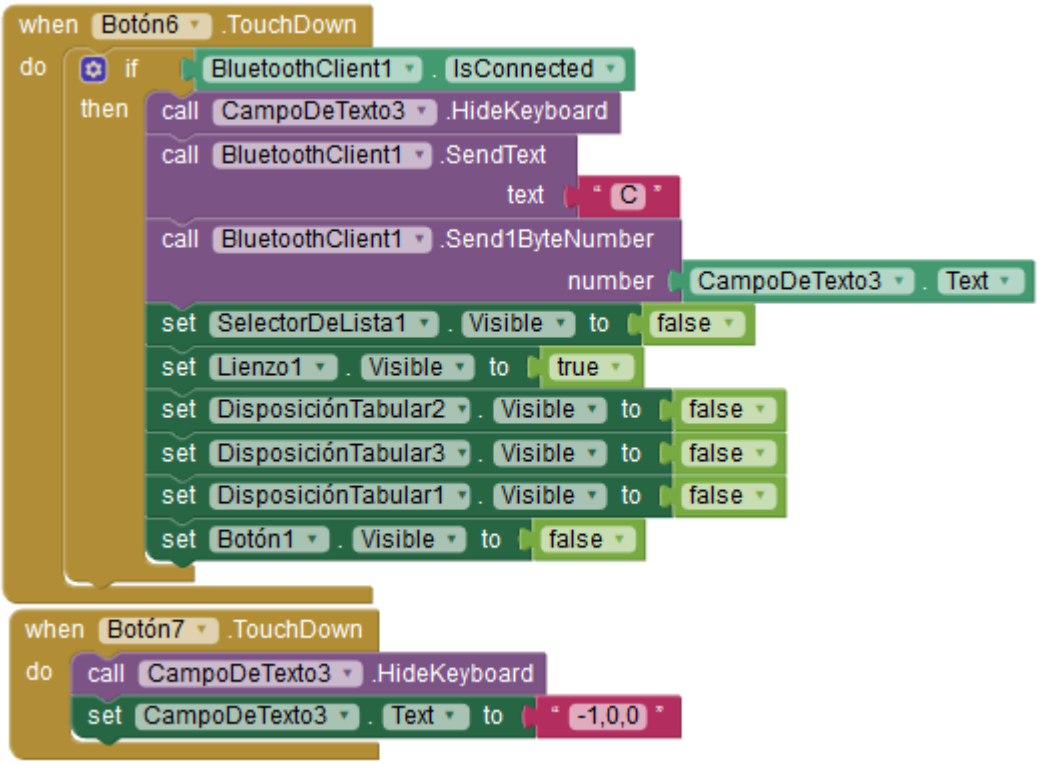
- Cambio de velocidad.



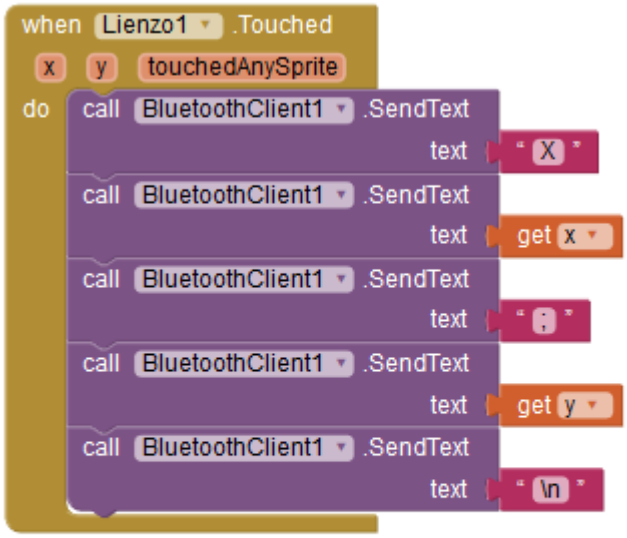
- Cambio vector acercamiento.

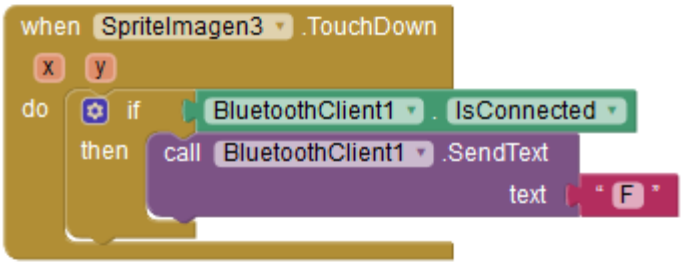


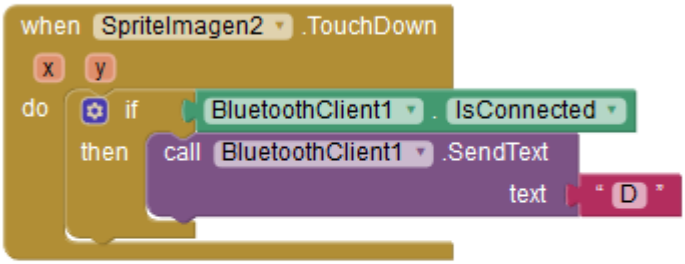
- Cambio vector de cierre.

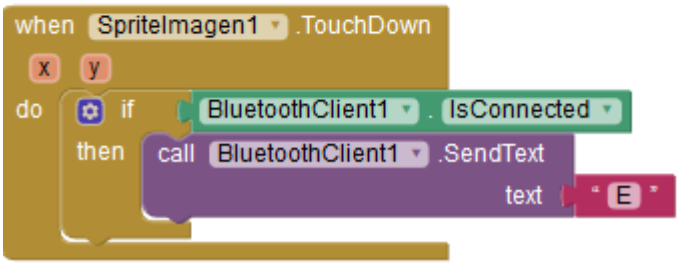


- Envío de posiciones.









ANEXO III. Programación Arduino

```
/*Programa completo
 25/2/2018
 cristina Nacar?Trujillo Gorr?
*/

#include <SoftwareSerial.h>// libreria para utilizar cualquier puerto como
pin de lectura-escritura
#include <VarSpeedServo.h> // libreria para controlar los servos.


VarSpeedServo base; // servo de rotacion continua para mover la base.
VarSpeedServo hombroD; // servo de rotacion limitada derecho del hombro.
VarSpeedServo hombroI; // servo de rotacion limitada izquierdo del hombro.
VarSpeedServo codo; // servo de rotacion limitada que simula el codo.
VarSpeedServo MGiro; // servo de rotacion limitada encargado de girar la
mu?ca.
VarSpeedServo MInclinacion; // servo de rotacion limitada encargado de
inclinar la mu?ca.
VarSpeedServo pinza; // servo de rotacion limitada que abre o cierra la
pinza.

SoftwareSerial BT(2, 3);//asignamo los pines 2 y 3 como lectura y
esctritura

String cadena;
String X;
String Y;

int l1 = 17; // longitud base - tronco mm
int l2 = 120; // longitud tronco - hombro mm
int l3 = 120; // longitud hombro - codo mm
int lm = 60; // longitud muñeca - pinza mm
int d6 = 37; // distancia punta pinza abierta al centro de la pinza mm



int o00[] = {0, 0, 0, 1}; // vector de origen
float a[] = { -1, 0, 0, 0}; // vector de acercamiento
float s [] = {0, -1, 0}; // vector de cierre
float CII[] = {180, 90, 90, 90, 90}; // valores iniciales
float CI[4];
float ptoFinal[] = {330, 225, 300}; // valores iniciales
float matriz[4][4];

float _X = 175; // coordenada X punto final
float _Y = 0.0; // coordenada Y punto final
float _Z = 300; // coordenada Z punto final mm


int velocidad = 51; //velocidad de movimiento m?ima.
```

```
bool estadoPinza;

bool estadoBrazo;

void pinzas (int posP) {
 if (posP == 1) { // abre la pinza
 pinza.write(90, 255);
 estadoPinza = 1;

 }

 else {
 pinza.write(0, 255);
 estadoPinza = 0;

 }
 delay(5000);
}

void moverBrazo(float pos[]) { //bloque de codigo encargado de mover cada
servo
 base.write(pos[0], velocidad);
 hombroD.write(pos[1], velocidad);
 hombroI.write(pos[1], velocidad);
 codo.write(pos[2], velocidad);
 MInclinacion.write(pos[3], velocidad);
 MGiro.write(pos[4], velocidad);

 delay(1000);

 if (estadoPinza == 1) {
 pinzas(0);
 }
 else {
 pinzas(1);
 }

}
void setup() // asigno cada servo a un pin, y coloco el manipulador en
posicion inicial
{
 BT.begin(9600);
 Serial.begin(9600);
 base.attach(13);
 hombroD.attach(12);
 hombroI.attach(11);
 codo.attach(10);
 MInclinacion.attach(9);
 MGiro.attach(8);
 pinza.attach(7);

 pinzas(1);

 base.write(0, velocidad);
```

```
 hombroD.write(90, velocidad);
 hombroI.write(90, velocidad);
 codo.write(90, velocidad);
 MInclinacion.write(90, velocidad);
 MGiro.write(90, velocidad);

}
void obtenerMatrizTransformacion(int di, float theta, int ai, float alpha)
{ // matriz de transformaci?
 float matrizTransformada[][4] = {{cos(theta), -1 * cos(alpha)*sin(theta),
sin(alpha)*sin(theta), ai * cos(theta)},
 {sin(theta), cos(alpha)*cos(theta), -1 * sin(alpha)*cos(theta), ai *
sin(theta)},
 {0, sin(alpha), cos(alpha), di},
 {0, 0, 0, 1}
 };
 for (int i = 0; i < 4; i++) {
 for (int j = 0; j < 4; j++) {
 matriz[i][j] = matrizTransformada[i][j];
 }
 }
}

void CinematicaInv(float X, float Y, float Z) { //calculo de la cinematica
inversa

 float raizA = sqrt(a[0] * a[0] + a[1] * a[1] + a[2] * a[2] + a[3] * a[3]);
//obtenemos un vector unitario

 for (int i = 0; i < 4; i++) {
 a[i] = a[i] / raizA;
 }

 float raizS = sqrt(s[0] * s[0] + s[1] * s[1] + s[2] * s[2]);

 for (int i = 0; i < 3; i++) {
 s[i] = s[i] / raizS;
 }
 //calculo del puno O3
 float o60[] = {0, 0, 0, 1};

 o60[0] = X;
 o60[1] = Y;
 o60[2] = Z;

 float LmA1[] = {0, 0, 0, 0};

 for (int i = 0; i < 4; i++) {
 LmA1[i] = a[i] * lm ;

 }

 float o30[4];
```

```
 for (int i = 0; i < 4; i++) {
 o30[i] = o60[i] - LmA1[i];
 }
 //Calculo del primer angulo
 float theta1 = atan(o30[1] / o30[0]);
 //Calculo de 3 angulo

  float theta3 = acos(((o30[0] * o30[0]) + ( o30[1] * o30[1]) + ((o30[2] -
l1) * (o30[2] - l1)) - (l2 * l2) - (l3 * l3)) / (2 * (l3 * l2)));
   //calculo del 2 angulo
  float fi = atan((o30[2] - l1) / sqrt((o30[0] * o30[0]) + (o30[1] *
o30[1])));

  float alpha = atan((l3 * sin(theta3)) / (l2 + l3 * cos(theta3)));

  float theta2 = fi - alpha;
  theta2=-1*abs(theta2);

//Calculo de las matrices de rotación
//Calculo la matriz de transformación de 3 a 0
//Para ello la descompongo en T de 3 a 2, T de 2 a 1 y T 1 a 0

 obtenerMatrizTransformacion(l1, theta1, 0, PI / 2);

 float T01[4][4];
 for (int i = 0; i < 4; i++) {
 for (int j = 0; j < 4; j++) {
 T01[i][j] = matriz[i][j];
 }
 }

 float T12[4][4];
 obtenerMatrizTransformacion(0, theta2, l2, 0);

 for (int i = 0; i < 4; i++) {
 for (int j = 0; j < 4; j++) {
 T12[i][j] = matriz[i][j];
 }
 }

 float T23[4][4];
 obtenerMatrizTransformacion(0, theta3, l3, 0);
 for (int i = 0; i < 4; i++) {
 for (int j = 0; j < 4; j++) {
 T23[i][j] = matriz[i][j];
 }
 }
 // vector perpendicular a y s
 float n[] = {((a[1]*s[2]) - (s[1]*a[2])), (-(a[0]*s[2]) + (s[0]*a[2])),
((a[0]*s[1]) - (s[0]*a[1]))};
 float R05[3][3] = {{s[0], n[0], a[0]}, {s[1], n[1], a[1]}, {s[2], n[2],
a[2]}};

 // matriz T01 por T12
 float T0112[4][4];
```

```
 for (int x = 0; x < 4; x++) {
 for ( int j = 0; j < 4; j++) {
 T0112[x][j] = 0;
 for (int i = 0; i < 4; i++) {
 T0112[x][j] = T0112[x][j] + T01[x][i] * T12[i][j];
 }
 }
 }

 float T03[4][4];
 // matriz T0112 por matriz T23
 for (int i = 0; i < 4; i++) {
 for ( int j = 0; j < 4; j++) {
 T03[i][j] = 0;
 for (int k = 0; k < 4; k++) {
 T03[i][j] = T03[i][j] + (T0112[i][k] * T23[k][j]);
 }
 }
 }

//Obtengo la matriz de rotación

 float R03[3][3];
 // nos quedamos con las 3 primeras columnas y filas
 for (int i = 0; i < 3; i++) {
 for ( int j = 0; j < 3; j++) {
 R03[i][j] = T03[i][j];
 }
 }

//Calculo el determinante de las 3 primeras filas y columnas

 float determinante = (R03[0][0] * R03[1][1] * R03[2][2]) + (R03[1][0] *
R03[2][1] * R03[0][2]) + (R03[2][0] * R03[0][1] * R03[1][2])
 - (R03[0][2] * R03[1][1] * R03[2][0]) - (R03[1][2] * R03[2][1] *
R03[0][0]) - (R03[2][2] * R03[0][1] * R03[1][0]);

//hago la matriz adjunta
if (determinante == 0) {
 Serial.println("ERROR Determinante = 0");
 }
 else {
 float adjunta[][3] = {{(R03[1][1]*R03[2][2] - R03[1][2]*R03[2][1]), -1 *
(R03[1][0]*R03[2][2] - R03[1][2]*R03[2][0]), (R03[1][0]*R03[2][1] -
R03[1][1]*R03[2][0])},
 { -1 * (R03[0][1]*R03[2][2] - R03[0][2]*R03[2][1]), (R03[0][0]*R03[2][2] -
R03[0][2]*R03[2][0]), -1 * (R03[0][0]*R03[2][1] - R03[0][1]*R03[2][0])},
 {(R03[0][1]*R03[1][2] - R03[0][2]*R03[1][1]), -1 * (R03[0][0]*R03[1][2] -
R03[0][2]*R03[1][0]), (R03[0][0]*R03[1][1] - R03[0][1]*R03[1][0])}
 };

//ahora la transpuesta

 float transpuesta[][3] = {{adjunta[0][0], adjunta[1][0], adjunta[2][0]},
```

```
 {adjunta[0][1], adjunta[1][1], adjunta[2][1]},
 {adjunta[0][2], adjunta[1][2], adjunta[2][2]}
 };

//matriz inversa
 float R03inv[3][3];

 for (int i = 0; i < 3; i++) {
 for (int j = 0; j < 3; j++) {
 R03inv[i][j] = (1 / determinante) * transpuesta[i][j];
 }
 }

// Multiplico para obtener la matriz de rotacion de 5 a 3
 float R35[3][3];

 for (int i = 0; i < 3; i++) {
 for ( int j = 0; j < 3; j++) {
 R35[i][j] = 0;
 for (int k = 0; k < 3; k++) {
 R35[i][j] = R35[i][j] + R03inv[i][k] * R05[k][j];

 }
 }
 }

//Calculo theta 4
 int signo;
 int valor;

 valor = -1 * R35[1][2];
 if (valor >= 0) {
 signo = -1;
 }
 else {
 signo = 1;
 }

 float theta4 = signo * acos(-1 * R35[1][2]);

// Calculo de theta 5
 valor = -1 * R35[2][0];
 if (valor >= 0) {
 signo = -1;
 }
 else {
 signo = 1;
 }

 float theta5 = signo * acos(R35[2][1]);
// Organizo los ángulos
 CI[0] = int(theta1 * 180 / PI);
 CI[1] = int(theta2 * 180 / PI);
 CI[2] = int(theta3 * 180 / PI);
```

```
 CI[3] = int(theta4 * 180 / PI);
 CI[4] = int(theta5 * 180 / PI);

// Los ángulos de 3 y 4 cuadrante los paso al 2 y 1.
 for (int i = 0; i < 5; i++) {
 if (CI[i] < 0) {
 CI[i] = 180 + CI[i];
 }
 else {
 CI[i] = CI[i];
 }
 }
 }
}
void loop() {

 //comprueba que haya conexion
 if (BT.available()) {

 char dato = BT.read(); // guarda el texto recibido
 Serial.print(dato);
 cadena += dato;
 //analiza la informaci? recibida
 if (dato == '\n') {
 if (cadena.charAt(0) == 'X') { // empieza con X es un punto nuevo al que
moverse
 X = cadena.substring(1, cadena.indexOf(';'));// divide la cadena desde la
X hasta el primer ;
 Y = cadena.substring((cadena.indexOf(';') + 1)); // divide la cadena desde
la Y hasta el final

 // pasa los valores a enteros
 int _X1 = X.toInt();
 int _Y1 = Y.toInt();
// Muestra los valores al pc conectado
 Serial.println(_X1);
 Serial.println(_Y1);
 //elimina la cadena
 X.remove(0);
 Y.remove(0);
 cadena.remove(0);
 //comprueba que se encuentren dentro de los limites
 if (_X1 > 254) {
 _X1 = map(_X1, 254.1, 508, 0, 400);
 }
 if (_X1 < 254) {
 _X1 = map(_X1, 254.1, 508, 0, 400);
 }
 if (_X1 == 254) {
 _X1 = 0.0;
 }
 _Y1 = map(_Y1, 0, 290, 0, -400);

 // Serial.println(_X1);
```

```
 // Serial.println(_Y1);
 //calcula la cinematica inversa
 CinematicaInv(_X1, _Y1, _Z);
 delay(1000);
 // desplaza el manipulador a punto calculado y luego al final
 moverBrazo(CI);
 moverBrazo(CII);
 }
 // si la cadena empieza por E,D o F calcula CI para cada punto y se
desplaza a dichos puntos
 if (cadena.charAt(0) == 'E') {

 _X = -280;
 _Y = -100;

 CinematicaInv(_X, _Y, _Z);
 delay(1000);
 moverBrazo(CI);
 moverBrazo(CII);

 cadena.remove(0);

 }
 if (cadena.charAt(0) == 'D') {

 _X = 0.0;
 _Y = -200;

 CinematicaInv(_X, _Y, _Z);
 delay(1000);
 moverBrazo(CI);
 moverBrazo(CII);
 cadena.remove(0);

 }
 if (cadena.charAt(0) == 'F') {

 _X = 200;
 _Y = -85;

 CinematicaInv(_X, _Y, _Z);
 delay(1000);
 moverBrazo(CI);
 // CinematicaInv(_X, _Y, _Z2);
 // moverBrazo(CI);
 moverBrazo(CII);

 cadena.remove(0);

 }
 // si la cadena empieza por V divide la cadena para obtener el porcentaje
de velocidad
 if (cadena.charAt(0) == 'V') {
 String _V = cadena.substring(1);
```

```
 // aplica el porcentaje a la velocidad
 velocidad = 255 * ( _V.toInt() / 100);

 _V.remove(0);
 cadena.remove(0);

 }
 // se la cadena empieza por A o C divide la cadena según la posición de
las , para obtener el valor del vector
 if (cadena.charAt(0) == 'A') {
 int posComa1 = cadena.indexOf(',');
 int posComa2 = cadena.indexOf(',', posComa1 + 1);
 String a1 = cadena.substring(1, posComa1);
 String a2 = cadena.substring(posComa1 + 1, posComa2);
 String a3 = cadena.substring(posComa2 + 1);

 a[0] = a1.toInt();
 a[1] = a2.toInt();
 a[2] = a3.toInt();

 a1.remove(0);
 a2.remove(0);
 a3.remove(0);
 cadena.remove(0);

 }

 if (cadena.charAt(0) == 'C') {
 int posComa1 = cadena.indexOf(',');
 int posComa2 = cadena.indexOf(',', posComa1 + 1);
 String s1 = cadena.substring(1, posComa1);
 String s2 = cadena.substring(posComa1 + 1, posComa2);
 String s3 = cadena.substring(posComa2 + 1);

 s[0] = s1.toInt();
 s[1] = s2.toInt();
 s[2] = s3.toInt();

 s1.remove(0);
 s2.remove(0);
 s3.remove(0);
 cadena.remove(0);

 }
 else {
 cadena.remove(0);

 }
 }
 }
}
```

Presupuesto.

- Materiales

Material	**Cantidad necesaria**	**Cantidad mínima**	**Coste paquete**	**Coste unitario**	**Coste Total**
Base de madera	1	1	4,86 €	4,86 €	4,86 €
Tornillos M3 de 14 mm	10	20	2,85 €	0,14 €	1,43 €
Tornillos M3 de 10 mm	10	10	1,13 €	0,11 €	1,13 €
Tornillos M4 de 40 mm	10	10	2,79 €	0,28 €	2,79 €
Tornillos M5 de 45 mm	5	10	3,13 €	0,31 €	1,16 €
MG996R	5	1	-	7,16 €	35,80 €
HXT900	2	1	-	2,85 €	5,70 €
Placa perforada	1	1	-	1,75 €	1,75 €
Barniz	0,05 L	0,25 L	6,40 €	-	1,28 €
Patas	6	2	2,79 €	0,70 €	4,20 €
Recoge cable	1	1	-	2,20 €	2,20 €
Tira pines	1	1	-	1,09 €	1,90 €
Arduino Uno	1	1	-	4,68 €	4,68 €
Modulo Bluetooth	1	1	-	4,74 €	4,72
Tablet	1	1	-	80 €	80 €
			TOTAL		**153,60 €**

- Estructura.

	Coste
Impresa en 3D	64,58 €
En madera	25 €
Prefabricada	34,79 €

- Mano de Obra.

	Horas totales	**Precio unitario**	**Coste**
Diseño y corte de piezas	10	20 €	200 €
Construcción del prototipo	15	20 €	300 €
Desarrollo de la aplicación	30	20 €	700 €
Programación del Arduino	45	20 €	900 €
		TOTAL	**2.100 €**

- Coste Total

	Coste
Materiales	153,60 €
Estructura	25 €
Mano de obra	2.100 €
TOTAL	**2.278,6 €**

Conclusiones y Líneas abiertas.

Este proyecto lo planteé como un complemento a la formación recibida durante el grado. Durante el desarrollo de este trabajo de fin de grado pude aplicar conocimientos adquiridos a lo largo del grado, como, por ejemplo: el diseño CAD de la asignatura de Expresión Gráfica y diseño asistido por ordenador, operaciones con matrices de la asignatura de Cálculo, la programación en C++ de las asignaturas de informática o la cinemática inversa de las asignaturas de Sistemas Robotizados, entre otros.

Incluso durante la realización de las prácticas externa trabajé con pantallas táctiles que se comunican con autómatas, lo que aplicado a este proyecto de manera industrial podría sustituir la placa Arduino y la Tablet.

A nivel personal consideré que el desarrollo de la aplicación sería lo más complejo, pero resulto ser bastante intuitivo y sencillo, resultando lo más complicado de todo el proyecto el desarrollo de la cinemática inversa en C++.

De cara a posibles mejoras o complementos se podría implementar las dos opciones que planteé, pero no llegue a desarrollar: controlar el manipulador haciendo uso de una pantalla en la aplicación que simule un joystick de videojuego y otra donde se pueda trazar una ruta y que el manipulador la siga. Estas dos opciones no se pudieron implementar por falta de tiempo.

A nivel estructural se podría sustituir el servomotor de la base por un motor paso a paso que permita un giro de 360°.

También se planteó la inclusión de una webcam que permita realizar un control por imagen, de tal manera que se pudiera desplazar el manipulador segmentando las imágenes por cuadrantes según formas o colores, e incluso llegar a incluir un control dinámico del mismo. Para poder incluir esta webcam habría que incluir unas placas tipo Raspberry Pi con la cual poder realizar el procesamiento de imágenes. Lamentablemente incluir la webcam y la Raspberry Pi encarecía enormemente el manipulador, ya que solo la Raspberry Pi y una webcam compatible con esta, cuesta más que todo el manipulador.

Por último, otra posible mejora es mostrar una alerta cuando se intente colocar el manipulador en una posición que no pueda alcanzar, permitiendo así al usuario saber porque el manipulador no se desplaza al punto indicado.

Conclusions and open lines.

I proposed this project as a complement to the training received during the degree. During the development of this end-of-degree work I was able to apply the knowledge acquired throughout the degree, such as: the CAD design of the Graphic Expression and Computer Aided Design course, operations with matrices of the Calculus course, the C++ programming of the computer science subjects or the inverse kinematics of the Robotized Systems subjects, among others.

Even during the external internship, I worked with touch screens that communicate with automatons, which applied to this project in an industrial way could replace the Arduino board and the Tablet.

On a personal level I considered that the development of the application would be the most complex, but it turned out to be quite intuitive and simple, being the most complicated of the whole project the development of the inverse kinematics in C++.

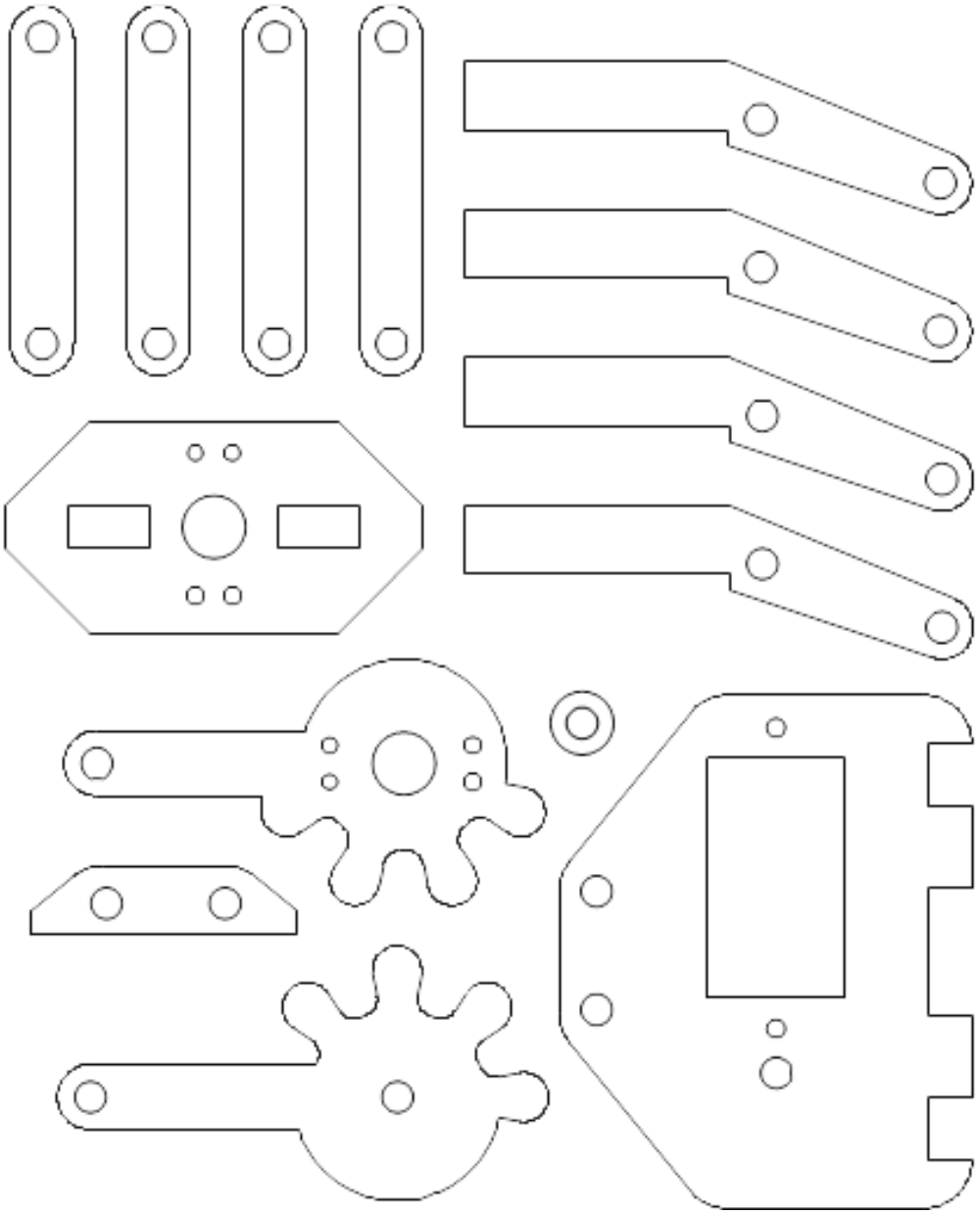
With a view to possible improvements or complements, the two options I proposed could be implemented, but I did not manage to develop them: control the manipulator using a screen in the application that simulates a videogame joystick and another where a route can be traced and the manipulator can follow it. These two options could not be implemented due to lack of time.

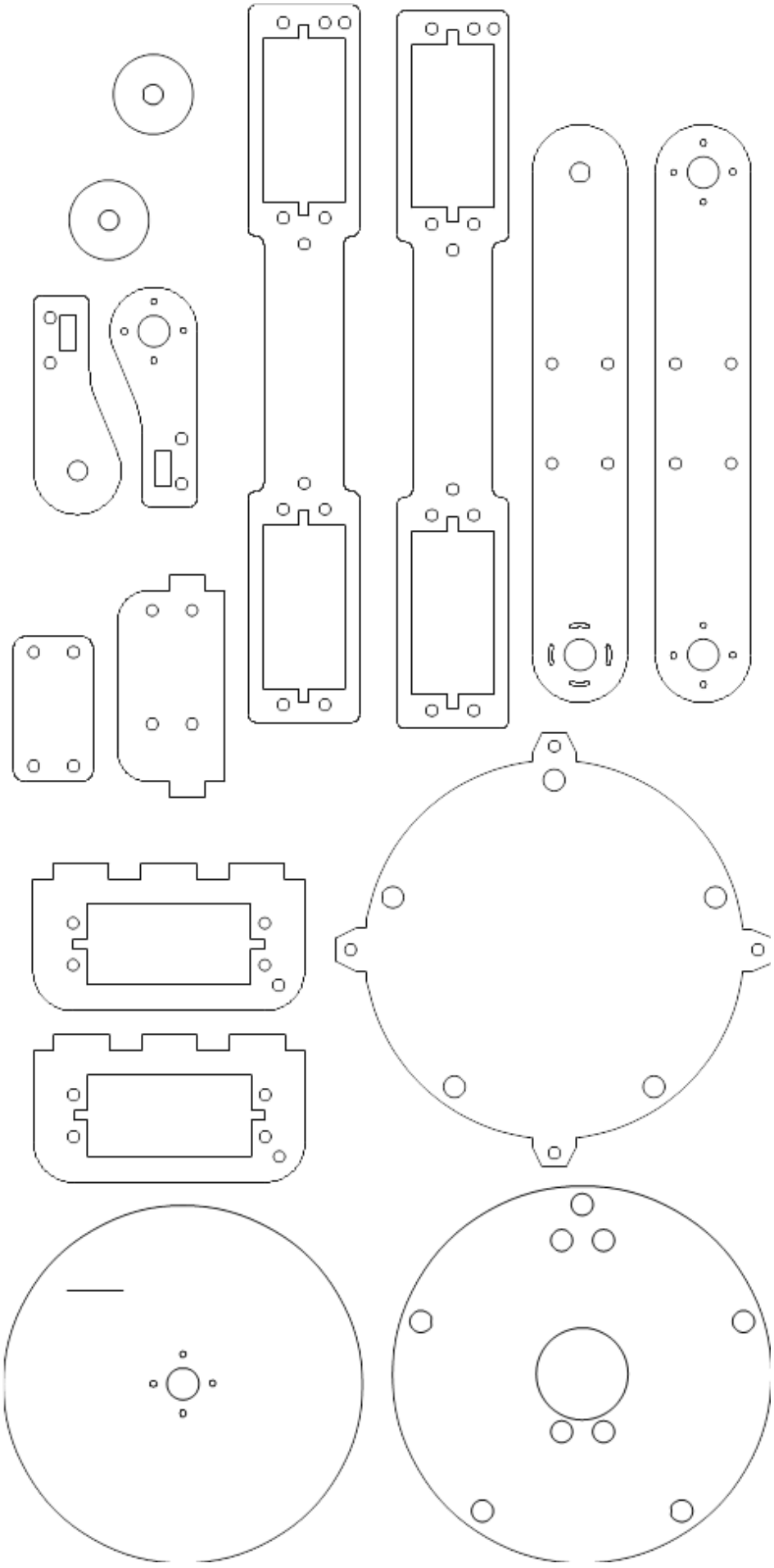
At the structural level, the servomotor of the base could be replaced by a stepper motor that allows a 360° rotation.

It was also proposed the inclusion of a webcam that allows an image control, so that the manipulator could be moved by segmenting the images by quadrants according to shapes or colors, and even to include a dynamic control of it. To be able to include this webcam you would have to include some Raspberry Pi type plates with which you could do the image processing. Unfortunately, including the webcam and the Raspberry Pi made the manipulator very expensive, since only the Raspberry Pi and a webcam compatible with it, costs more than the whole manipulator.

Finally, another possible improvement is to display an alert when attempting to place the manipulator in a position that it cannot reach, thus allowing the user to know why the manipulator does not move to the indicated point.

Planos: Diseño de las piezas.





Referencias.

[1] Fundamentos de Robótica - A. Barrientos, L. Peñin, C. Balaguer, R. Aracil – 2ed
[2] https://www.thingiverse.com/
[3] https://www.thingiverse.com/thing:30163
[4] https://www.thingiverse.com/thing:172706
[5] http://www.areatecnologia.com/informatica/impresoras-3d.html
[6] https://es.wikipedia.org/wiki/Servomotor_de_modelismo
[7] https://hobbyking.com/es_es/hxt900-micro-servo-1-6kg-0-12sec-9g.html
[8] https://hobbyking.com/es_es/towerpro-mg996r-10kg-servo-10kg-0-20sec-55g.html
[9] http://cdn-tienda.bricogeek.com/946-thickbox_default/arduino-uno.jpg
[10] https://images-na.ssl-images-amazon.com/images/I/41ZwOW%2BkxlL.jpg
[11] https://www.prometec.net/bt-hc06/
[12] http://appinventor.mit.edu/explore/index-2.html