



## **“Control de un brazo mecánico diseñado con tecnología 3D”**

**“Robotín”**

**Francisco José Marín Zurita**

Trabajo Fin de Grado  
Ingeniería Radioelectrónica  
Tutor: Prof. Dr. Carlos Corrales Alba

Escuela de Ingenierías Marina, Náutica y Radioelectrónica

SEPTIEMBRE 2017



## AGRADECIMIENTOS

*Quisiera agradecer a varias personas y entidades la ayuda que se ha prestado en la realización de este proyecto. Entre ellas, y en primer lugar, a mi director de proyecto Don Carlos Corrales Alba, por todo lo que nos ha enseñado y lo que me ha transmitido durante estos años, así como depositar su confianza en mí para la realización del presente proyecto.*

*A Agustín Carmona Lorente por su inestimable ayuda y enseñanzas, sin las cuales hubiera sido más difícil la realización de este trabajo. Ante todas las dificultades que se han presentado, siempre ha estado dispuesto a dedicarme su tiempo y su conocimiento.*

*Y a toda mi familia: en especial a mis padres, Francisco Marín Torrejón y Josefa Zurita Viruez y mis hermanos, Ismael y Desirée. Han sido un apoyo constante, y nos lo han dado todo y siguen inspirándonos, apoyándonos cada día.*

*Agradecer también a Don Joaquín Moreno Marchal, mi tutor de Alumnos Colaboradores del departamento en Automática, Electrónica, Arquitectura y Redes de Computadores, que junto con Agustín Carmona me han abierto las puertas de sus laboratorios y sus recursos dentro del proyecto Crealab.*

*Muchas gracias a todas las personas que he nombrado y a otras tantas que no lo he echo por su apoyo incondicional, esta etapa de mi vida llega a su fin pero me llevo muchos amigos y compañeros, así como el cariño de muchos profesores a los cuales no olvidaré.*



# Índice de contenidos

1. Resumen.....	1
Abstract.....	1
2. Objetivo.....	2
3. Descripción del problema.....	2
3.1. Comprensión del problema.....	2
3.2. Restricciones.....	2
3.3. Especificaciones de diseño.....	4
3.3.1. Estructura.....	4
3.3.2. Procedimiento.....	4
4. Elección de Arduino.....	5
5. Programación.....	7
5.1. Arduino.....	7
5.1.1. Software.....	7
5.1.1.1. Introducción al entorno Arduino.....	7
5.1.2. Comenzando con Arduino.....	8
5.1.2.1. Estructura.....	8
5.1.2.2. Funciones.....	9
5.1.2.3. Librerías.....	10
5.1.3. Código.....	10
5.1.3.1. Código paso a paso.....	16
5.2. MIT APP Inventor (Aplicación Android).....	26
5.2.1. ¿Por qué MIT APP Inventor?.....	26
5.2.2. Entorno MIT APP Inventor.....	26
5.2.2.1. Entorno MIT APP Inventor. Diseñador.....	27
5.2.2.2. Entorno MIT APP Inventor. Bloques.....	33
6. Anotaciones.....	42
6.1. Pruebas con servomotores y potenciómetros.....	42
6.2. Uso de fuentes de alimentación conmutadas.....	42
6.3. Diseño en papel de las piezas del robot y primeras impresiones.....	43
6.4. Módulo Bluetooth. Configuración y Comandos AT.....	43
6.5. Modos de estado y de uso.....	45
6.6. Estructura.....	45
6.7. Piezas Robotin.....	45
7. Esquemas.....	47
7.1. Esquema Montaje. Pruebas Iniciales 1.....	47
7.2. Esquema Montaje. Pruebas Iniciales 2.....	49
7.3. Esquema Montaje. Pruebas Iniciales 3.....	51
7.4. Esquema Montaje. Configuración Módulo Bluetooth.....	53
7.5. Esquema Montaje. Final.....	55
7.6. Esquema control SG90.....	57
8. Materiales usados.....	58
Impresora 3D.....	58
PLA.....	58
Servomotor SG90.....	59
Servomotor S3003.....	59
Potenciómetro.....	60
Breadboard.....	61
Cableado.....	61
Arduino UNO.....	62
LED.....	62

Resistencias.....	62
Fuente de alimentación conmutadas.....	63
Botón pulsador.....	63
HC-05.....	64
9. Presupuesto.....	65
10. Competencias.....	66
11. Conclusiones.....	67
12. Referencias.....	68
12.1 Presupuesto.....	68
12.2 Información.....	69
13. Anexos.....	71
13.1. Data Sheet servomotor SG90.....	71
13.2. Data Sheet servomotor S3003.....	73

## Índice de tablas

Tabla 1: Algunos servos del mercado - BRICOGEEK.....	3
Tabla 2: Algunos servos del mercado - PROMETEC.....	3
Tabla 3: Algunos servos del mercado - LA FÁBRICA DIGITAL.....	4
Tabla 4: Conexiones Arduino UNO.....	6
Tabla 5: Comandos AT.....	44
Tabla 6: Información sobre piezas del robot.....	46
Tabla 7: Características comunes botón pulsador.....	63

## Índice de ilustraciones

Ilustración 1: Placa Arduino UNO.....	5
Ilustración 2: Pines Microprocesador ATmega328P – Arduino UNO.....	6
Ilustración 3: Entorno Arduino.....	7
Ilustración 4: Operaciones lógicas en C.....	19
Ilustración 5: Entorno MIT APP Inventor. Diseñador.....	26
Ilustración 6: Entorno MIT APP Inventor. Bloques.....	27
Ilustración 7: Opciones del menú Interfaz de usuario.....	28
Ilustración 8: Opciones del menú Conectividad.....	28
Ilustración 9: Opciones del menú Disposición.....	28
Ilustración 10: Aplicación Android. Vista superior.....	29
Ilustración 11: Aplicación Android. Vista inferior.....	29
Ilustración 12: Componentes. Selector de lista o "listPicker1".....	29
Ilustración 13: Componentes. Etiqueta Estado_Bluetooth.....	30
Ilustración 14: Componentes. Etiqueta INFO.....	30
Ilustración 15: Componentes. Elementos de Disposición, Botones y Etiquetas.....	30
Ilustración 16: Componentes. Elementos de disposición para memorizar varios estados.....	31
Ilustración 17: Botón Salir de la aplicación.....	32
Ilustración 18: Logo Universidad de Cádiz.....	32
Ilustración 19: Logo CreaLab.....	32
Ilustración 20: Menú Bloques.....	33
Ilustración 21: Bloque LisPicker1 y algunas de las opciones que nos muestra la Interfaz de programación.....	34
Ilustración 22: Bloques. Salir de la aplicación.....	35
Ilustración 23: Bloques. Bluetooth.....	35
Ilustración 24: Bloques. Deslizador 1.....	36
Ilustración 25: Bloques. Deslizador 2 al Deslizador 5.....	37
Ilustración 26: Bloques. Botón memoria Deslizador 1.....	38
Ilustración 27: Bloques. Botones de memoria Deslizadores del 2 al 5.....	38
Ilustración 28: Bloques. Botones "+" y "-" Deslizador 1.....	39
Ilustración 29: Bloques. Botones "+" y "-" Deslizadores 2 al 5.....	39
Ilustración 30: Bloques. Botón memoria "Pos. 1".....	40
Ilustración 31: Bloques. Botones memoria "Pos. 2", "Pos. 3" y "Pos. 4".....	41
Ilustración 32: Esquema Montaje. Pruebas Iniciales 1.....	47
Ilustración 33: Esquema Montaje. Pruebas Iniciales 2.....	49
Ilustración 34: Esquema Montaje. Pruebas Iniciales 3.....	51
Ilustración 35: Esquema Montaje. Módulo Bluetooth.....	53
Ilustración 36: Esquema Montaje. Final.....	55
Ilustración 37: Esquema control SG90.....	57
Ilustración 38: PRUSA I3 HEPHESTOS de BQ.....	58
Ilustración 39: Rollos de PLA.....	58
Ilustración 40: Servomotor SG90.....	59
Ilustración 41: Servomotor S3003.....	59
Ilustración 42: Potenciómetro.....	60
Ilustración 43: Esque potenciómetro.....	60
Ilustración 44: Breadboard.....	61
Ilustración 45: Cables Jumper.....	61
Ilustración 46: Placa Arduino UNO.....	62
Ilustración 47: LED.....	62
Ilustración 48: Resistencia.....	62
Ilustración 49: DC - DC Step Down.....	63

Ilustración 50: Esquema control pulsador.....63  
Ilustración 51: Botón pulsador.....63  
Ilustración 52: HC-05 Modulo Bluetooth.....64





# 1. Resumen

*A través del presente documento se pretende hacer llegar al lector el proceso y la metodología que se ha seguido para el diseño de un robot que simula el funcionamiento de un brazo humano, la programación tanto de la placa Arduino como de la App para controlar dicho brazo y el uso de los elementos de los que forman parte.*

*En este documento podrá encontrar la descripción en detalle de la programación paso a paso del control del robot, el uso de la web MIT APP Inventor que se ha usado para la creación de la App para el control Android por medio de Bluetooth.*

*Para el diseño de las piezas del robot se utilizó la herramienta gráfica AutoCAD, y para su impresión se ha dispuesto de una Impresora 3D. El control del robot se dispuso de Potenciómetros, Servomotores y Fuentes de Alimentación Conmutadas.*

*Palabras clave: Robot, Arduino, App, MIT APP Inventor, AutoCAD, Impresora 3D, Potenciómetro, Servomotor, Fuente Alimentación Conmutada.*

## Abstract

*Through this document is intended to let know the reader the procedure and methodology that has been followed for the design of a robotic arm that simulates the behaviour of an human arm, the programming of the Arduino with the App for the control of said robotic arm and the use of each element.*

*It can be found in this document the detailed description of the programming step by step to control the robot, the use of the webpage MIT APP Inventor which has been used for the creation of the App for controlling the Android device through Bluetooth.*

*For the piece's design it has been used the design software AutoCAD, for creation of the pieces a 3D printer has been used and for the control of the robot potentiometer, servomotors and conmutated power supply have been used.*

*Keywords: Robot, Arduino, App, MIT APP Inventor, AutoCAD, 3D printer, potentiometer, servomotor, conmutated power supply.*

## 2. Objetivo

El objetivo del presente proyecto es realizar el diseño y programación de un robot que simule un brazo humano, así como el diseño e impresión en 3D del mismo.

Utilizaremos un programa de diseño gráfico, AutoCAD, y para la elaboración de dichas piezas se utilizará una impresora 3D.

Dicho robot, se debe controlar de dos formas distintas:

1. Mediante control manual.
2. Por medio de una App, de creación propia para este proyecto.

## 3. Descripción del problema

### 3.1. Comprensión del problema

La finalidad del presente proyecto es realizar el diseño y montaje de un robot que simulara a un brazo humano, para ello se diseñó con AutoCAD las diferentes piezas articuladas y se imprimieron en 3D.

El control manual de las diferentes posiciones del robot se realiza mediante potenciómetros lineales, siendo éstos los sensores de posición, disponiéndose verticalmente, asemejando un brazo humano, de tal forma, que al mover cualquier parte de éste, el otro elemento siga dichos movimientos. Éste segundo brazo constituiría el conjunto actuador, que formado por servomotores y dispuestos de forma vertical.

En cuanto al control por Bluetooth, sólo se necesita un dispositivo Android compatible con la aplicación que se ha preparado para el control del robot con su módulo receptor de Bluetooth correspondiente.

### 3.2. Restricciones

1. **Grados de libertad de los servomotores.** Para que se asemejara a un brazo humano se han escogido servomotores con 180° de libertad. (Ver tabla 1. Algunos servos del mercado).
2. **Material para las piezas de la impresora 3D.** Como hemos tenido acceso a la impresora 3D del departamento en Ingeniería en Automática, Electrónica,



Arquitectura y Redes de Computadores, y Crealab que dispone de una Prusa I3 Hephestos, se ha utilizado material del tipo PLA.

3. **Tamaño de las piezas a imprimir.** No deben superar el espacio de impresión de la Prusa I3, en este caso 20x20 cm.
4. **Fuente de alimentación para los servomotores.** Para poder usar el par máximo de los servomotores deben estar alimentados a una tensión específica. Para ello se han usado fuentes de alimentación conmutadas.
5. **Alcance de la señal Bluetooth.** Máximo unos 20 metros aproximadamente, según especificaciones del fabricante.
6. **Peso de la estructura.** Al estar los servomotores apilados verticalmente se debe tener en cuenta el peso de estructura.

SERVOMOTORES						
TIENDA	BRICOGEEK					
MODELO	S05NF	DF15RSMG	DS65HB	HD 6001 HB	HD 1160A	3001 HB
CARACTERÍSTICAS						
Alimentación	2,8 – 6,0 V	5,0 – 7,4 V	4,8 – 6,0 V	4,8 – 6,0 V	4,8 – 6,0 V	4,8 – 6,0 V
Rotación	180°	360°	180° ± 10°	180° ± 10°	180° ± 10°	180° ± 10°
Torque	2,8 – 3,2 Kg-cm	19,3 kg-m	1,3 – 1,5 kg-cm	5,8 – 6,7 kg/m	2,0 – 2,7 kg-cm	3,5 – 4,4 kg-cm
Dimensiones (mm)	30,2 x 13,8 x 28,8	40 x 40 x 20	20 x 11,5 x 30	43,5 x 20,5 x 54,5	29,6 x 13,2 x 40	43,5 x 54,5 x 20,5
Peso	20 gr	ND	25 gr	43 gr	16 gr	43 gr

Tabla 1: Algunos servos del mercado - BRICOGEEK

SERVOMOTORES				
TIENDA	PROMETEC			
MODELO	S3003	MG90S	SG90	MG995
CARACTERÍSTICAS				
Alimentación	4,8 – 6,0 V	4,5 – 6 V	4,8 V	4,8 – 7,2 V
Rotación	180°	180°	180°	180°
Torque	3,2 – 4,1 kg-cm	1,8 – 2,2 kg-cm	1,8 kg-cm	8,5 – 10 kg-cm
Dimensiones (mm)	41 x 20,1 x 36,1	22,5 x 12 x 35,5	22,2 x 11,8 x 31	40,6 x 19,8 x 42,9
Peso	37 gr	13,4 gr	9 gr	55 gr

Tabla 2: Algunos servos del mercado - PROMETEC

SERVOMOTORES						
TIENDA	FÁBRICA DIGITAL					
MODELO	S3003	SG90	ES09MD	MG996R	MG90S	ES08MDII
CARACTERÍSTICAS						
Alimentación	4,8 – 6,0 V	4,8 V	4,8 – 6,0 V	4,8 – 7.2 V	4,5 – 6 V	4,8 – 6,0 V
Rotación	180°	180°	180°	120°	180°	??
Torque	3,2 – 4,1 kg-cm	1,8 kg-cm	2,3 – 2,6 kg-cm	9,7 – 11 kg-cm	1,8 – 2,2 kg-cm	1,6 – 2 kg-cm
Dimensiones (mm)	41 x 20,1 x 36,1	22,2 x 11,8 x 31	23 x 12 x 24,5	40,7 x 19,7 x 42,9	22,5 x 12 x 35,5	23,0 x 11,5 x 24,0
Peso	37 gr	9 gr	14,8 gr	55 gr	13,4 gr	12 gr

Tabla 3: Algunos servos del mercado - LA FÁBRICA DIGITAL

### 3.3. Especificaciones de diseño

#### 3.3.1. Estructura

Por medio de un programa de diseño gráfico, en este caso AutoCAD, se realiza el diseño de las piezas que constituirán la estructura del robot, que posteriormente serán impresas en la impresora 3D, Prusa I3 Hephestos, utilizando para ello el software necesario para dicha tarea, en este caso se utilizó Repetier.

El controlador donde se conectan los sensores y actuadores será una placa de Arduino en este supuesto el tipo Uno. Para la programación de dicha placa de Arduino se utilizará un lenguaje de programación de alto nivel conocido como C++.

Para la programación de la interfaz para Android conectada mediante Bluetooth con la placa de Arduino se utilizará MIT APP Inventor.

#### 3.3.2. Procedimiento

Al conectar el sistema a la red, se encenderá un Led de color Rojo, este indicará que el sistema está listo, no se podrá controlar ni de forma manual, ni de forma inalámbrica a través del Bluetooth, este Led indica que el robot se encuentra en la posición de StandBy o modo reposo.

Al pulsar sobre el pulsador, P1, el Led rojo se apagará y se encenderá otro de color Verde y a su vez otro Led más de color Blanco. El Led verde indica que el robot se encuentra en modo de funcionamiento activo y el led blanco indica que el modo de control es del tipo manual, únicamente el robot se controlará por medio de la estructura manual.



Al pulsar sobre el pulsador, P2, el Led verde continuará encendido, pero el led blanco se apaga y se enciende el Led de color Azul. Indica que se activará el módulo Bluetooth para que la estructura del brazo pueda ser controlada únicamente por medio de la aplicación Android creada para dicha función.

En caso de una emergencia se pulsa nuevamente sobre P1, el led verde se apagará y volverá a encenderse el Led Rojo. Ocasionará el apagado automático del robot, podría decirse que es un paro de emergencia, dejando el robot en modo StandBy.

## 4. Elección de Arduino

Para este proyecto se decidió utilizar Arduino por varios motivos:

1. Permite la creación de entornos interactivos para el usuario de manera sencilla e intuitiva, y es perfecta para todo tipo de diseñadores tanto profesionales como para aficionados, ya que el nivel de programación y el precio lo pone el usuario.
2. Utilización de código abierto. Permite que los usuarios puedan compartir sus proyectos con la comunidad, así otros usuarios pueden tomar partes de esos códigos y manipularlos para sus propios fines.

Dentro de los distintos tipos de placas que ofrece Arduino, la más popular es el **Arduino UNO**. Que ha sido la placa que se ha utilizado en este proyecto. Utiliza un microprocesador ATmega328P, dispone de 14 pines de entrada/salida digitales, 6 pines de entrada analógicos. Conexión USB, un jack de corriente y un botón reset.

Una de las grandes ventajas de esta placa es que se puede utilizar sin miedo a que le pase algo, ya que en el peor de los casos bastaría con reemplazar el microprocesador, que además tiene un precio muy asequible. Lo cual proporciona tranquilidad a la hora de experimentar con la placa. Ideal para usuarios que estén empezando.



*Ilustración 1: Placa Arduino  
UNO*

Las especificaciones técnicas de Arduino UNO son las siguientes:

<b>Microcontrolador</b>	ATmega328P
<b>Corriente de trabajo</b>	5V
<b>Corriente de entrada (recomendada)</b>	7-12V
<b>Corriente de entrada (límite)</b>	6-20V
<b>Pines Digitales E/S</b>	14
<b>Pines Entrada Analógicos</b>	6
<b>DC Current per I/O Pin</b>	20 mA
<b>DC Current for 3.3V Pin</b>	50 mA
<b>Memoria Flash</b>	32 KB (ATmega328P) de los cuales 0.5 KB son usados para el arranque
<b>SRAM</b>	2 KB (ATmega328P)
<b>EEPROM</b>	1 KB (ATmega328P)
<b>Velocidad de Reloj</b>	16 MHz
<b>Longitud</b>	68.6 mm
<b>Anchura</b>	53.4 mm
<b>Peso</b>	25 g

Tabla 4: Conexiones Arduino UNO

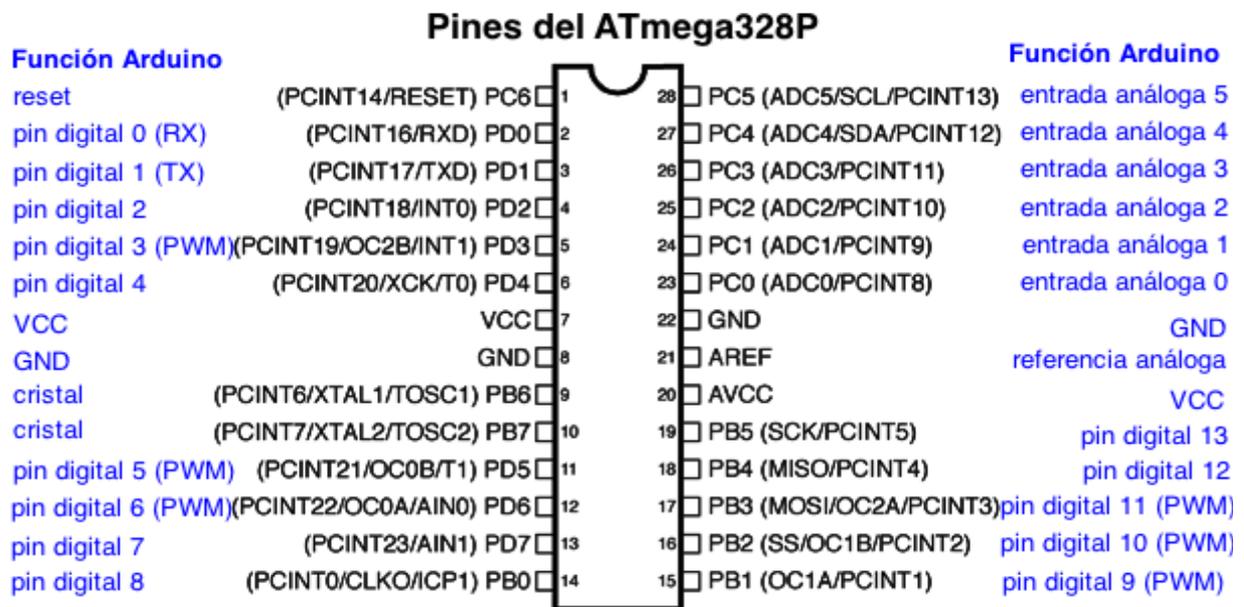


Ilustración 2: Pines Microprocesador ATmega328P – Arduino UNO



## 5. Programación

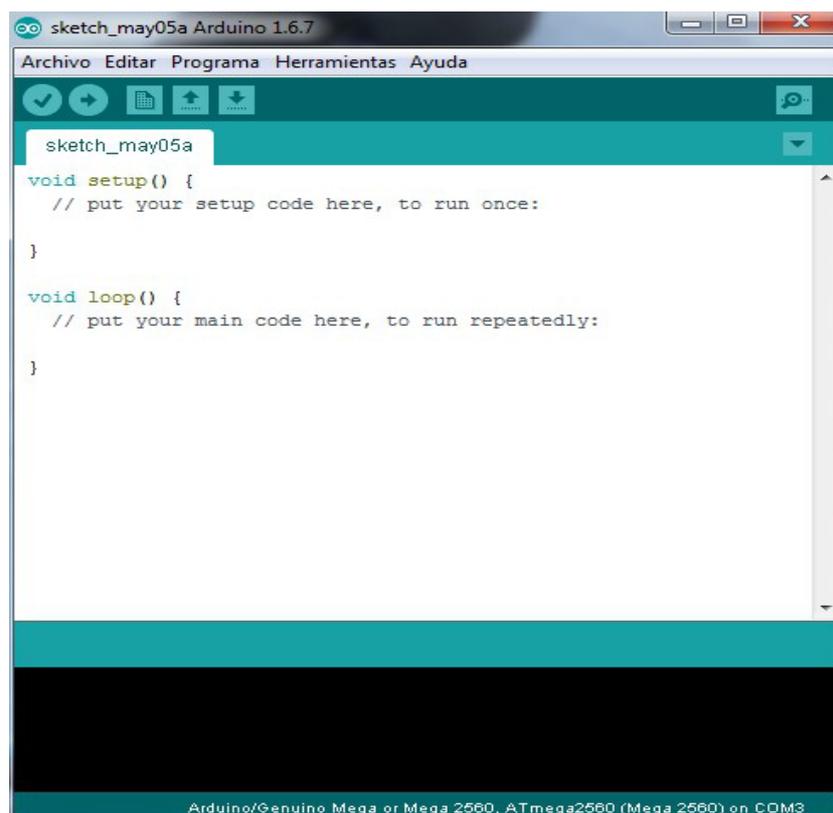
La programación de este proyecto consta de dos partes bien diferenciadas pero que a su vez se relacionan para su correcto funcionamiento, y se describirán a continuación.

### 5.1. Arduino

Tal como se ha mencionado anteriormente, la programación en la que se basa Arduino es del tipo C++, pero tiene unas características específicas las cuales se describen a continuación.

#### 5.1.1. Software

##### 5.1.1.1. Introducción al entorno Arduino



*Ilustración 3: Entorno Arduino*

En esta imagen se puede observar el entorno de la aplicación Arduino, con todos sus menús, los más característicos tienen sus propios botones de acceso rápido, los cuales se explican a continuación:

- |   |   |
|---|---|
|  | 1. Verificar / Compilar<br>Comprueba el código en busca de errores.   |
|  | 2. Subir<br>Envía el código a la placa arduino.                       |
|  | 3. Nuevo<br>Crea una nueva rutina.                                    |
|  | 4. Abrir<br>Muestra un menú con todas las rutinas de tu "sketchbook". |
|  | 5. Salvar<br>Guarda tus rutinas.                                      |

## 5.1.2. Comenzando con Arduino

### 5.1.2.1. Estructura

Los programas de Arduino están divididos en tres partes: estructura, valores (variables y constantes) y funciones.

La estructura de Arduino es bastante simple y se organiza en al menos dos partes o funciones, cada una con sus correspondientes bloques de declaraciones.

```
void setup()
{
    declaraciones;
}
void loop()
{
    declaraciones;
}
```

Ambas funciones son imprescindibles para que el programa funciones.

#### Setup()

La función *setup* debería contener la declaración de cualquier variable al comienzo del programa. Es la primera función que el programa ejecuta, y sólo se ejecuta una vez, se suele utilizar para asignar *pinMode* o inicializar las comunicaciones serie.

```
void setup()
{
    pinMode (pin, OUTPUT);    //ajustamos 'pin' como salida
}
```



## Loop()

Esta función se ejecuta a continuación de setup, en ella se incluye el código que se ejecutará continuamente (leyendo entradas, activando salidas, etc). Es el núcleo de todos los programas y es donde se va a realizar la mayor parte del trabajo.

```
void loop()
{
    digitalWrite (pin, HIGH);    //Activa 'pin'
    delay (1000);                //espera un segundo
    digitalWrite (pin, LOW);    //desactiva 'pin'
    delay (1000);                //espera un segundo
}
```

### 5.1.2.2. Funciones

Son bloques de código que tienen un nombre y un grupo de declaraciones que realizan cuando se llama a dicha función.

Se utilizan para ejecutar tareas repetitivas y así disminuir el tamaño del código, y en consecuencia, el desorden en la programación. En primer lugar declararemos el tipo de función, que será el valor que devolverá dicha función (int, void...). A continuación del tipo, se declara el nombre de la función y, entre paréntesis, los parámetros que le pasamos a la función.

```
tipo_función nombre_función (parámetros)
{
    declaraciones;
}
```

La siguiente función `int delayVal()`, asigna un valor de retardo en un programa por lectura del valor de un potenciómetro.

```
Int delayval()
{
    int v;                        //crea una variable temporal 'v'
    v = analogRead(pot)          //lee el valor del potenciómetro
    v /= 4;                       //convierte 0-1023 a 0-255
    return v;                     //devuelve el valor final de 'v'
}
```

### Llaves {}

Las llaves definen el comienzo y el final de bloques de función, y deben estar balanceadas (a una llave de apertura '{' debe seguirla una llave de cierre '}', de lo contrario se producen errores de compilación):

**Punto y coma ;**

Un punto y coma debe usarse al final de cada declaración y separa los elementos del programa.

```
int x = 13; //declara la variable 'x' como el entero 13
```

**Bloques de comentarios /\*...\*/**

Los bloques de comentarios, son zonas de texto que ignora el programa a la hora de compilar, pueden abarcar varias líneas pero deben empezar por `/*` y acabar con `*/`, al igual que las llaves, deben de estar balanceados, de lo contrario se producen errores de compilación.

**Comentarios de línea //**

Lo que se escriba detrás de `//` hasta el final de la línea, el programa lo ignorará a la hora de compilar. Se utiliza para hacer anotaciones puntuales.

**5.1.2.3. Librerías**

Arduino tiene implantada una serie de librerías que son de uso específico. Por ejemplo, si queremos utilizar la pantalla LCD, el propio Arduino ya tiene codificada la librería y solo haría falta incluirla en la cabecera del programa.

En la bibliografía se muestra un enlace con el listado completo de librerías que ofrece Arduino.

**5.1.3. Código**

```
#include <Servo.h>

#define BT 2 //Bluetooth
#define Pulsador_1 8 //modo funcionamiento, standby / on
#define Pulsador_2 11 //modo empleo, manual / automático
#define Led_1 9 //color rojo
#define Led_2 10 //color verde
#define Led_3 12 //color Azul
#define Led_4 13 //color Blanco

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;

int angulo1 = 0; // variable que almacena la posición del servo
```



```

int angulo2 = 0;
int angulo3 = 0;
int angulo4 = 0;
int angulo5 = 0;

const int PIN_SERVO1 = 7;
const int PIN_SERVO2 = 6;
const int PIN_SERVO3 = 5;
const int PIN_SERVO4 = 4;
const int PIN_SERVO5 = 3;

const int PIN_POTENCIOMETRO1 = A0; // pin al que conectamos el potenciómetro
const int PIN_POTENCIOMETRO2 = A1;
const int PIN_POTENCIOMETRO3 = A2;
const int PIN_POTENCIOMETRO4 = A3;
const int PIN_POTENCIOMETRO5 = A4;

char caracter;
String readString; //asignamos la palabra readString a una variable tipo cadena

int inicio = 0;
int estado_actual = 0;
int estado_anterior = 0;

int inicio_2 = 0;
int estado_actual_2 = 0;
int estado_anterior_2 = 0;

void setup() {

    pinMode (Pulsador_1, INPUT);
    pinMode (BT, OUTPUT);
    pinMode (Led_1, OUTPUT);
    pinMode (Led_2, OUTPUT);

    servo1.attach(PIN_SERVO1); // vinculo el objeto con el servo al pin introducido arriba
    servo2.attach(PIN_SERVO2);
    servo3.attach(PIN_SERVO3);
    servo4.attach(PIN_SERVO4);
    servo5.attach(PIN_SERVO5);
    Serial.begin(9600); //para inicializar la comunicación

    digitalWrite (Led_1, HIGH);
    digitalWrite (Led_2, LOW);
    digitalWrite (Led_3, LOW);
    digitalWrite (Led_4, LOW);
}

```

```

void loop() {

    estado_actual = digitalRead (Pulsador_1); //asigno a estado actual el valor que entra por la
                                                //entrada digital "Pulsador"
    if (estado_actual && estado_anterior == 0) //multiplicamos estado anterior por estado actual y
                                                //si es 0 entra dentro del if
    {
        inicio = 1 - inicio;
        delay(100); //retardo para evitar el efecto rebote del pulsador
    }
    estado_anterior = estado_actual; //reseteamos el valor de las variables

    if (inicio == 1)
    {
        digitalWrite (Led_1, LOW);
        digitalWrite (Led_2, HIGH);

        estado_actual_2 = digitalRead (Pulsador_2); //asigno a estado actual el valor que entra
                                                    //por la entrada digital "Pulsador"
        if (estado_actual_2 && estado_anterior_2 == 0) //multiplicamos estado anterior por
                                                    //estado actual y si es 0 entra
                                                    //dentro del if
        {
            inicio_2 = 1 - inicio_2;
            delay(100); //delay para evitar el efecto rebote del pulsador
        }
        estado_anterior_2 = estado_actual_2; //reseteamos el valor de las variables

        if (inicio_2 == 1) //funcionamiento mediante BT
        {
            digitalWrite (Led_3, HIGH);
            digitalWrite (Led_4, LOW);
            comprobar(); //Llamada a la función de control mediante
                        //Bluetooth
        }
        else
        {
            digitalWrite (Led_3, LOW);
            digitalWrite (Led_4, HIGH);

            digitalWrite (BT, LOW);

            manual();
        }
    }
    else
    {

```



```

        digitalWrite (Led_1, HIGH);
        digitalWrite (Led_2, LOW);
        digitalWrite (Led_3, LOW);
        digitalWrite (Led_4, LOW);

        digitalWrite (BT, LOW);
    }
}

void comprobar()
{
    digitalWrite (BT, HIGH);
    if (Serial.available())
    {
        //compruebo el primer carácter del puerto serie
        caracter = Serial.read();

        if(caracter=='A') //si leo A voy a la función motor1
            motor1();

        if(caracter=='B') //si leo B voy a la función motor2
            motor2();

        if(caracter=='C') //si leo C voy a la función motor3
            motor3();

        if(caracter=='D') //si leo D voy a la función motor4
            motor4();

        if(caracter=='E') //si leo E voy a la función motor5
            motor5();
    }
}

void motor1()
{
    delay(10);
    readString="";
    while (Serial.available()) //se reciben los datos numéricos del ángulo del servomotor
    {
        char c = Serial.read(); // Se leen los caracteres que ingresan por el puerto
        readString += c; //cada carácter se construye en una cadena
    }
    if (readString.length() >0) //se verifica la longitud de la cadena
    {
        Serial.println(readString.toInt()); //mandamos la información al servo
        servo1.write(readString.toInt());
    }
}

```

```
}  
  
void motor2()  
{  
    delay(10);  
    while (Serial.available())  
    {  
        char c = Serial.read();  
        readString += c;  
    }  
    if (readString.length() >0)  
    {  
        Serial.println(readString.toInt());  
        servo2.write(readString.toInt());  
        readString="";  
    }  
}  
  
void motor3()  
{  
    delay(10);  
    while (Serial.available())  
    {  
        char c = Serial.read();  
        readString += c;  
    }  
    if (readString.length() >0)  
    {  
        Serial.println(readString.toInt());  
        servo3.write(readString.toInt());  
        readString="";  
    }  
}  
  
void motor4()  
{  
    delay(10);  
    while (Serial.available())  
    {  
        char c = Serial.read();  
        readString += c;  
    }  
    if (readString.length() >0)  
    {  
        Serial.println(readString.toInt());  
        servo4.write(readString.toInt());  
        readString="";  
    }  
}
```



```

void motor5()
{
    delay(10);
    while (Serial.available())
    {
        char c = Serial.read();
        readString += c;
    }
    if (readString.length() >0)
    {
        Serial.println(readString.toInt());
        servo5.write(readString.toInt());
        readString="";
    }
}

void manual()
{
    int valorPotenciometro1 = analogRead(PIN_POTENCIOMETRO1);
    //hacemos que la posición del servo dependa del potenciómetro
    int valorPotenciometro2 = analogRead(PIN_POTENCIOMETRO2);
    int valorPotenciometro3 = analogRead(PIN_POTENCIOMETRO3);
    int valorPotenciometro4 = analogRead(PIN_POTENCIOMETRO4);
    int valorPotenciometro5 = analogRead(PIN_POTENCIOMETRO5);

    int angulo1 = map(analogRead(A0), 0, 1023, 0, 179);
    int angulo2 = map(analogRead(A1), 0, 1023, 0, 179);
    int angulo3 = map(analogRead(A2), 0, 1023, 0, 179);
    int angulo4 = map(analogRead(A3), 0, 1023, 0, 179);
    int angulo5 = map(analogRead(A4), 0, 1023, 0, 179);

    servo1.write(angulo1);           //para mandar el valor al servo
    servo2.write(angulo2);
    servo3.write(angulo3);
    servo4.write(angulo4);
    servo5.write(angulo5);
    delay(15);
}

```

Ese sería el código que se ha utilizado para el presente proyecto y a continuación se describirá cada una de sus partes para una mayor comprensión.

Se debe recordar que la estructura del programa está dividida en tres partes, una primera donde se introducen librerías y se declaran e inicializan variables de índole general, es decir, que estarán presente en

todo código; una segunda “`void setup() {}`” donde se inicializa el estado de los pines así como de las comunicaciones necesarias; y una tercera “`void loop() {}`” donde irá el programa en sí, el cual se ejecutará de forma cíclica.

Se debe mencionar también que al basarse el lenguaje de programación “C++” como se puede comprobar en el código escrito anteriormente hay llamadas a funciones en la tercera parte, pero el código de esas funciones se encuentra escrito fuera del “loop”. Esto se hace para evitar escribir código de forma redundante, por estética y sobre todo para optimizar la programación ya que al final se reduce un problema grande en otros más pequeños que sean más fácil de resolver.

### 5.1.3.1. Código paso a paso

En la primera parte del código nos encontramos con las librerías así como la declaración de variables globales:

```
#include <Servo.h>
```

Sirve para incluir librerías, de esta forma se pueden utilizar los comandos de control incluidos en esa librería de forma fácil y sencilla.

```
#define BT 2  
#define Pulsador_1 8  
#define Pulsador_2 11  
#define Led_1 9  
#define Led_2 10  
#define Led_3 12  
#define Led_4 13
```

Este grupo de líneas sirve para crear variables y asignarles unos pines específicos de la placa Arduino, por ejemplo `#define BT 2` nos indica que hemos definido / creado una variable con el nombre BT, de Bluetooth, y que le asignamos el pin digital número 2, y así sucesivamente con el resto de variables que estén precedidas del comando `#define`.

```
Servo servo1;  
Servo servo2;  
Servo servo3;  
Servo servo4;  
Servo servo5;
```

Si observan estas líneas, la palabra “*Servo*” es de color naranja. Son de este color para identificar las variables de control de Arduino. En concreto estas líneas sirven para crear cinco variables en este caso, `servo1`, `servo2`, ... , `servo5`, que se utilizarán para definir los servomotores que se usan en el proyecto.

```
int angulo1 = 0;  
int angulo2 = 0;  
int angulo3 = 0;  
int angulo4 = 0;  
int angulo5 = 0;
```



La palabra clave “**int**” indica que lo que se escriba a continuación es una variable de tipo entero, en la cual almacenaremos números de tipo entero, utilizando para ello 2 Bytes de memoria, pudiendo tomar valores de entre -32768 y 32767.

En este caso se están declarando variables con el nombre “*anguloX*”, siendo “X” el número del servomotor.

Existen varias formas de declarar variables, otra forma sería:

```
const int PIN_SERVO1 = 7;
const int PIN_SERVO2 = 6;
const int PIN_SERVO3 = 5;
const int PIN_SERVO4 = 4;
const int PIN_SERVO5 = 3;
```

```
const int PIN_POTENCIOMETRO1 = A0;
const int PIN_POTENCIOMETRO2 = A1;
const int PIN_POTENCIOMETRO3 = A2;
const int PIN_POTENCIOMETRO4 = A3;
const int PIN_POTENCIOMETRO5 = A4;
```

Al igual que con “define”, se puede utilizar el comando o palabra reservada “**const int**” de esta forma se están asignando los pines digitales del 7 al 3 para los servos del 1 al 5, y los pines analógicos desde A0 a A4 a los potenciómetros del 1 al 5.

```
char caracter;
```

Sirve para declarar la palabra “caracter” como variable de tipo carácter.

```
String readString;
```

Asignamos la palabra reservada “**readString**” a una variable de tipo cadena de caracteres.

```
int inicio = 0;
int estado_actual = 0;
int estado_anterior = 0;
```

```
int inicio_2 = 0;
int estado_actual_2 = 0;
int estado_anterior_2 = 0;
```

Se le asigna un valor de inicio a estas variables que posteriormente servirán para controlar el estado y modo de funcionamiento del robot.

En la segunda parte del programa se encuentra el “setup”, en el cual se inicializa el estado de algunos pines y las comunicaciones:

```
void setup() {

    pinMode (Pulsador_1, INPUT);
    pinMode (BT, OUTPUT);
    pinMode (Led_1, OUTPUT);
    pinMode (Led_2, OUTPUT);
```

La palabra reservada “*pinMode* ()” se utiliza para inicializar los pines digitales, en concreto en estas cuatro sentencias, se asigna el pin vinculado a “Pulsador\_1” como entrada, y los pines vinculados a “BT”, “Led\_1” y “Led\_2” como salidas. En lugar de poner el nombre de Pulsador\_1, BT, Led\_1 o Led\_2, se podría haber puesto directamente el número del pin a los cuales hacen referencia, Pulsador\_1 → 8, BT → 2, Led\_1 → 9 o Led\_2 → 10. Pero de esta forma si se decide cambiar de placa o si se decide cambiar el lugar donde se conectarán dichos componentes bastaría sólo con cambiar el número de pin en la declaración de variables, de otro modo, habría que buscar en todo el código y cambiar el número e pin de cada sentencia.

```
servo1.attach(PIN_SERVO1);
servo2.attach(PIN_SERVO2);
servo3.attach(PIN_SERVO3);
servo4.attach(PIN_SERVO4);
servo5.attach(PIN_SERVO5);
```

Por medio de estas sentencias se vincula el objeto con el servo a través del pin que se introdujo en la declaración de variables, es decir. Servo1 está vinculado con PIN\_SERVO1, que si recuerda hace referencia al pin digital 7, servo2 de igual forma estará vinculado con PIN\_SERVO2, que hace referencia al pin digital 6, servo3 estará vinculado con PIN\_SERVO3, a través del pin digital 5, servo4 estará vinculado con PIN\_SERVO4, a través del pin digital 4 y servo5 estará vinculado con PIN\_SERVO5, por medio del pin digital 3.

Al igual que antes si se decide cambiar de placa o de salida digital bastará con cambiar dicha referencia en la parte de declaración de variables.

```
Serial.begin(9600);
```

“*Serial.begin()*” sirve para inicializar el puerto serie de la placa Arduino y en el interior del paréntesis se introduce la velocidad en baudios del dispositivo. En este caso se ha utilizado para ayudar en la programación y ver si el programa, cuando estaba en fase de creación, se ejecutaba correctamente y a través del monitor serie del programa Arduino poder ver la ejecución del programa.

```
digitalWrite (Led_1, HIGH);
digitalWrite (Led_2, LOW);
digitalWrite (Led_3, LOW);
digitalWrite (Led_4, LOW);
}
```

A través del comando “*digitalWrite*” se puede escribir en un pin digital. A través de estas sentencias, al ser salidas digitales, se pueden poner en “*HIGH*” o en “*LOW*”. De esta forma al ponerlas, en alto = HIGH lo que se consigue es mandar 5 voltios y si las ponemos en bajo = LOW, se manda a dicho pin digital 0 voltios.



De esta forma, se asegura de forma inicial, que se encienda el led rojo, Led\_1, para indicar que el robot se encuentra en stand by, y de este modo no se pueda controlar ni de forma manual ni de forma inalámbrica.

Hasta aquí sería la parte del setup, donde se ha establecido la inicialización de algunas de las variables, se han iniciado las comunicaciones por el puerto serie, 9600 baudios, que es la velocidad de comunicación del Bluetooth en estado de funcionamiento normal, y se han asignado o vinculado los servomotores que se van a utilizar.

En la tercera parte tiene el “loop”, la parte más importante ya que es donde se encuentra el grueso del programa y la parte del código que se repetirá una y otra vez.

```
void loop() {
```

```
    estado_actual = digitalRead (Pulsador_1);
```

Por medio de esta sentencia se lee el valor del pin digital vinculado a Pulsador\_1, que si se recuerda, es la entrada digital 8.

```
    if (estado_actual && estado_anterior == 0){
```

Por medio de este comando de control “if()” se evalúa la condición que se introduce en el paréntesis, si al evaluar dicha condición se comprueba que se cumple, el condigo entrará a ejecutar lo que se encuentre dentro de los corchetes, en caso contrario, no los leerá por lo que pasará a la siguiente línea de código del programa.

Esta condición comprueba por medio de unos operadores el estado de las variables, u operaciones digitales entre dichas variables. En este caso, “&&” hace referencia a la operación lógica AND, que equivale al producto lógico entre dichas variables, y por medio del operador “==” compara el resultado de dicha operación lógica con el valor “0”.

<b>A</b>	<b>B</b>	<b>!A</b>	<b>A &amp;&amp; B</b>	<b>A    B</b>
0	0	1	0	0
1	0	0	0	1
0	1	1	0	1
1	1	0	1	1

Ilustración 4: Operaciones lógicas en C

```
inicio = 1 – inicio;
```

En caso de haberse cumplido la condición del *if* se ejecutaría esta sentencia la cual hace una operación aritmética básica de resta, y la almacena en la variable “inicio”.

```
delay(100);
```

Esta sentencia es un pequeño retardo de 100 milisegundos que se introduce para evitar el efecto rebote del pulsador. De este modo se consigue que al pulsar y soltar se detecte si cambia el valor de la variable Pulsador\_1.

```
    }
    estado_anterior = estado_actual;
```

Mediante esta sentencia se resetean el valor de ambas variables.

```
if (inicio == 1) {
```

De nuevo se comprueba el valor de una variable para acceder al conjunto de sentencias que se encuentran delimitadas por las llaves “{}”. Si en este caso el valor de la variable inicio es 1 se ejecutan estas sentencias:

```
    digitalWrite (Led_1, LOW);
    digitalWrite (Led_2, HIGH);
```

Por medio de estas dos sentencias, se pondría Led\_1 en bajo, lo cual apagaría el Led rojo y Led\_2 en alto, lo cual encendería el Led verde, indicando que el estado de funcionamiento ha cambiado de modo reposo o standby a modo activo. A su vez se encenderá el Led\_4, indicando que el modo de funcionamiento es manual.

```
    estado_actual_2 = digitalRead (Pulsador_2);
```

Se lee la entrada digital que hace referencia al Pulsador\_2 y se almacena ese valor en la variable estado\_actual\_2.

```
if (estado_actual_2 && estado_anterior_2 == 0) {
```

Este “if” evaluará el estado del Pulsador\_2 con la variable estado\_anterior\_2, y para que se ejecuten las sentencias descritas entre sus corchetes debe cumplirse dicha condición, en caso contrario, se saltará dicho bloque de sentencias.

```
    inicio_2 = 1 – inicio_2;
```

Servirá para acceder al modo de funcionamiento manual o inalámbrico.

```
    delay(100);
}
```



Un nuevo retardo de 100 milisegundos para evitar el efecto rebote del Pulsador\_2.

```
estado_anterior_2 = estado_actual_2;
```

Se igualan las variables y de este modo se resetean sus valores.

```
if (inicio_2 == 1) {
```

Si dicha condición se cumple se accede al modo de funcionamiento inalámbrico.

```
digitalWrite (Led_3, HIGH);
digitalWrite (Led_4, LOW);
```

Se pone el alto el Led\_3, de color azul, el cual indica que el modo de funcionamiento es por medio del Bluetooth y se apaga el Led\_4 (de color Blanco), el cual indicaba el modo de funcionamiento manual.

```
comprobar();
}
```

Esta sentencia indica una llamada a la función “comprobar()”, la cual se encarga del funcionamiento y control del robot por medio de la aplicación Android creada para este proyecto.

Esta función así como otras que se han implementado para este código serán explicadas después de terminar con la parte del “loop”.

```
else{
```

El comando “else” se puede utilizar después de utilizar un “if”, y significa “si no”, por lo que si no se entra en el modo de funcionamiento inalámbrico, se ejecutan estas sentencias:

```
digitalWrite (Led_3, LOW);
digitalWrite (Led_4, HIGH);
```

Se escribe en las salidas digitales de tal forma que se pone en alto Led\_4, lo cual hace que se encienda el led de color blanco y se pone el Led\_3 en bajo, haciendo que permanezca el led de color azul apagado.

```
digitalWrite (BT, LOW);
```

Al poner esta salida digital en bajo, lo que se consigue es apagar el módulo Bluetooth.

```
manual();
}
}
```

Esta sentencia indica la llamada a la función con nombre “manual()”, la cual se ha usado para el control del robot de forma manual. Al igual que la sentencia “comprobar()” será descrita después de terminar con el bloque “loop”.

```
else {
```

Este “else” hace referencia al primer “if”, al entrar en esta parte del código se estaría en el modo de reposo, el cual está definido por:

```
digitalWrite (Led_1, HIGH);
digitalWrite (Led_2, LOW);
digitalWrite (Led_3, LOW);
digitalWrite (Led_4, LOW);
digitalWrite (BT, LOW);
    }
}
```

Cuando el robot se encuentra en este estado, el Led\_1, de color rojo está en alto, por lo que está encendido y el resto de Leds, así como el módulo Bluetooth están en bajo o apagados.

Con esto acabaría el “loop” y el programa en sí, quedan por explicar las funciones “comprobar()” y “manual()” que se describirán a continuación.

**void comprobar() {**

Por medio de esta función se alimentará el módulo Bluetooth y se tomará el control de los servos a través de la aplicación Android que se ha diseñado para ello. El código de esta función es el siguiente:

```
digitalWrite (BT, HIGH);
```

A través de esta sentencia se alimenta el módulo Bluetooth, ya que dicho módulo necesita una tensión de 5 voltios y al poner en alto (HIGH) el pin que hace referencia a BT, pin digital número 2, de esta forma se consigue que sólo al entrar en esta función esté alimentado el módulo Bluetooth.

```
if (Serial.available()) {
```

Por medio de esta sentencia se comprueba si el puerto serie está disponible, para ello se utiliza “*Serial.available()*” que es un comando de control que lee los caracteres que llegan al puerto serie.

```
    caracter = Serial.read();
```

De este modo se almacena el carácter que entra por el puerto serie en la variable “caracter”, de este modo en las siguientes sentencias comprobando el valor almacenado de esa variable se ejecutará la función que corresponda.

```
        if(caracter=='A')
            motor1();
```

Se comprueba que el carácter recibido sea el carácter ‘A’ y si coinciden se hará la llamada a la función “motor1()”.

```
        if(caracter=='B')
            motor2();
```



```

if(caracter=='C')
    motor3();

if(caracter=='D')
    motor4();

if(caracter=='E')
    motor5();
    }
    }

```

En caso de que el carácter que entra por el puerto serie, y que se almacena en la variable “caracter” no sea ‘A’, se pasaría a la segunda sentencia de control, que hace la comparación con la letra ‘B’. En caso afirmativo se haría la llamada a la función “motor2()”, y así sucesivamente. Si el carácter es una ‘C’ se llamaría a la función “motor3()”, con el carácter ‘D’ se llamaría a la función “motor4()” y con el carácter ‘E’ se llamaría a la función “motor5()”.

Estos caracteres serán enviados por un dispositivo con Android y serán recibidos por el módulo Bluetooth conectado a la placa de Arduino y serán enviados al puerto serie.

A continuación se explicará el funcionamiento de la función “motor1()”, omitiendo explicar el resto de funciones con el nombre “motor2()” a “motor5()”, ya que el funcionamiento interno es el mismo, lo único que cambia es el servo el cual controla cada función.

```
void motor1() {
```

```
    delay(10);
```

Se introduce un pequeño retardo de 10 milisegundos.

```
    ReadString="";
    while (Serial.available()) {
```

El comando “*while()*” se utiliza para el control lógico, y es un “mientras”, es decir, mientras se cumpla la sentencia u orden entre sus paréntesis se ejecutará el resto de sentencias anidadas dentro de sus corchetes “{}”.

En este caso comprueba que el puerto serie esté disponible y que esté llegando información a través de él.

```
    char c = Serial.read();
```

En esta parte del programa se declara una variable “c” del tipo “char”, esta variable es de carácter local, sólo tendrá uso dentro de esta función y en ella se almacena el carácter que esté entrando en el buffer del puerto serie.

```
    readString += c;
```

```
}
```

Se almacena cada carácter en una cadena de caracteres. El “+=” se utiliza para sumar y almacenar el valor de la variable “c” a dicha cadena, de este modo, cada carácter que vaya entrando por el buffer del puerto serie que es almacenado en la variable “c”, a continuación es almacenada en la variable “readString” que formará una cadena de caracteres.

```
if (readString.length() >0) {
```

Se comprueba la longitud de la cadena, debiendo ser mayor de 0 para poder entrar dentro de la sentencia de control de este “if”.

```
Serial.println(readString.toInt());
```

Se envía la información al monitor serie del Arduino, de esta forma se puede observar que valores son los que están llegando y que posteriormente se enviarán al servomotor.

```
servo1.write(readString.toInt());
    }
}
```

En esta última sentencia se manda la información que llega al servomotor, de esta manera se envía un número entre 0 y 179 que si se recuerda son los grados de libertad del servomotor, de este modo conseguimos el control de dichos servomotores.

La última función que se va a explicar es la que se encarga del control manual de los servomotores, los cuales serán controlados por medio de potenciómetros.

```
void manual(){
```

```
int valorPotenciometro1 = analogRead(PIN_POTENCIOMETRO1);
int valorPotenciometro2 = analogRead(PIN_POTENCIOMETRO2);
int valorPotenciometro3 = analogRead(PIN_POTENCIOMETRO3);
int valorPotenciometro4 = analogRead(PIN_POTENCIOMETRO4);
int valorPotenciometro5 = analogRead(PIN_POTENCIOMETRO5);
```

Por medio de estas sentencias se leerá la posición del potenciómetro a través de una lectura analógica y se almacenarán dichas posiciones en las variables “valorPotenciometroX”, siendo “X” el número de cada potenciómetro.

Debido a que los valores analógicos van de 0 a 1023 y los servomotores sólo pueden tomar valores entre 0 y 179 (valor máximo 180 grados), se debe realizar alguna operación para equiparar dichos valores, esa función se conoce como “escalar” y de esta forma se ajusta el movimiento de los potenciómetros a los servomotores.

Existen varios métodos para resolver este problema, el más sencillo, consiste en dividir esos 1024 pasos entre 6, quedando así un valor de 170, con lo que se consigue que el movimiento del potenciómetro quede dentro de los grados de maniobra del servo, perdiendo eso sí, algo de precisión.



El propio Arduino dispone de una función para escalar, la cual es la siguiente “`map(analogRead(A0), 0, 1023, 0, 179)`” con ella se hace la lectura analógica del pin A0, se le dice que es un valor entre “0 , 1023” y se convierte a un valor entre “0, 179”, de esta forma se consigue un escalado más preciso el cual se ha utilizado.

```
int angulo1 = map(analogRead(A0), 0, 1023, 0, 179);  
int angulo2 = map(analogRead(A1), 0, 1023, 0, 179);  
int angulo3 = map(analogRead(A2), 0, 1023, 0, 179);  
int angulo4 = map(analogRead(A3), 0, 1023, 0, 179);  
int angulo5 = map(analogRead(A4), 0, 1023, 0, 179);
```

A través de estas sentencias se crean y se almacenan al mismo tiempo las variables en las cuales se almacenará el valor escalado del potenciómetro el cual se va a corresponder con el valor del ángulo del servomotor.

```
servo1.write(angulo1);  
servo2.write(angulo2);  
servo3.write(angulo3);  
servo4.write(angulo4);  
servo5.write(angulo5);  
    delay(15);  
}
```

Por medio de estas sentencias se escribe el valor del ángulo en cada servo, y se añade un pequeño retardo al final.

Con esto quedarían explicadas todas la funciones del código que se han preparado para el presente proyecto.

## 5.2. MIT APP Inventor (Aplicación Android)

### 5.2.1. ¿Por qué MIT APP Inventor?

Para la aplicación Android se ha decidido utilizar “MIT APP Inventor” (MIT viene de Massachusetts Institute of Technology), que es un entorno de programación a través de Internet, debido a su sencillez y a que utiliza un método intuitivo de utilización, además existe mucha información y tutoriales y es gratuito.

Para utilizar dicho entorno de programación, es necesario tener una cuenta de Gmail.

Para acceder sólo hay que entrar en la siguiente URL: [ai2.appinventor.mit.edu/](http://ai2.appinventor.mit.edu/) y acceder al espacio con los datos de la cuenta de correo de Gmail. A continuación se hace clic en nuevo proyecto y listo, ya se puede comenzar con la creación y diseño de tu aplicación Android.

### 5.2.2. Entorno MIT APP Inventor

MIT APP Inventor cuenta con dos partes bien diferenciadas a la hora de diseñar tu aplicación, en la primera de ellas se muestra una pantalla, que simula un dispositivo Android, se llama “Diseñador”; y una segunda pantalla, en la cual se programarán las acciones de control que se programen, se llama “Bloques”.

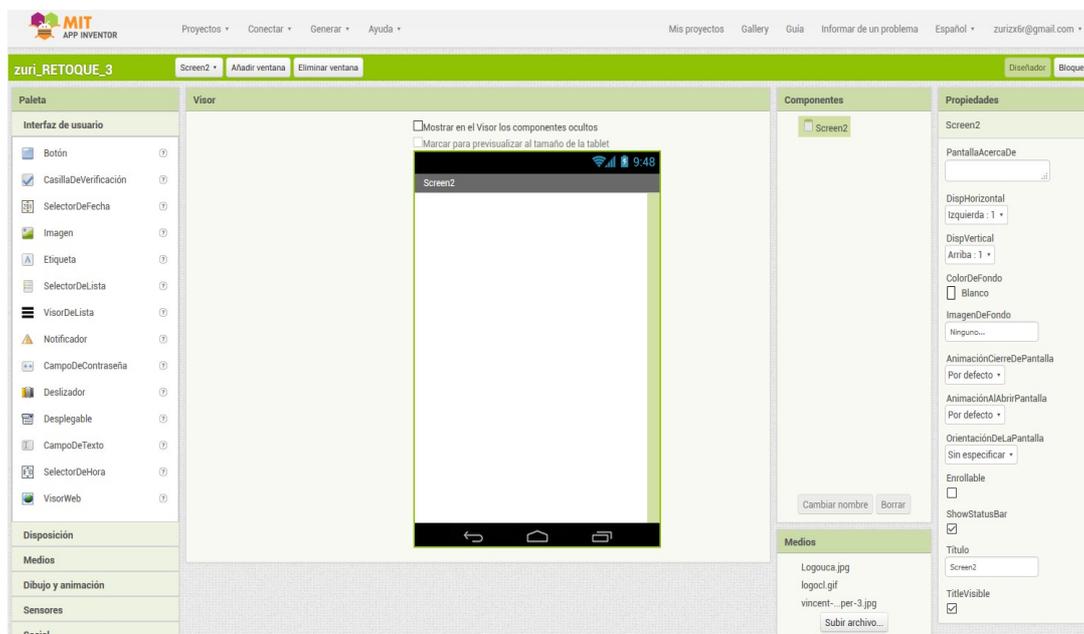
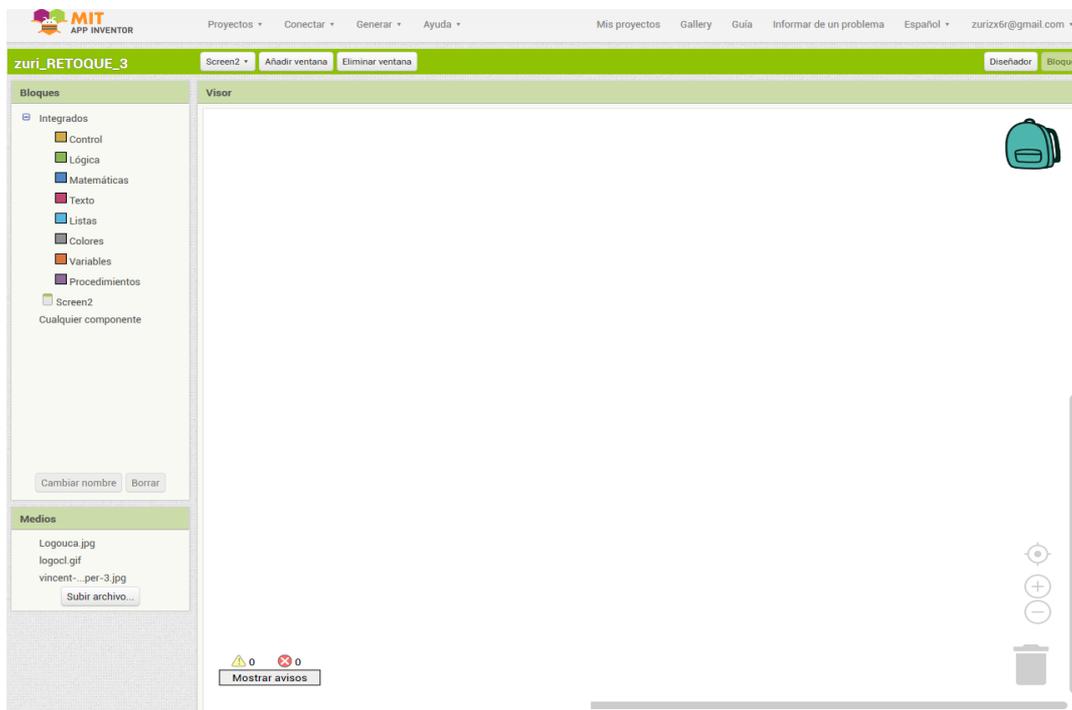


Ilustración 5: Entorno MIT APP Inventor. Diseñador.



*Ilustración 6: Entorno MIT APP Inventor. Bloques*

### 5.2.2.1. Entorno MIT APP Inventor. Diseñador

En el “Diseñador” se pueden apreciar 4 componentes principales:

- Paleta
- Visor
- Componentes y Medios
- Propiedades

Dentro de “Paleta” se encuentran varios menús: como Interfaz de usuario, Disposición, Medios... Los más relevantes en este proyecto son Interfaz de usuario, Disposición y Conectividad ya que con las opciones que presentan estos dos menús se ha realizado el diseño de la App para el control de los servomotores.

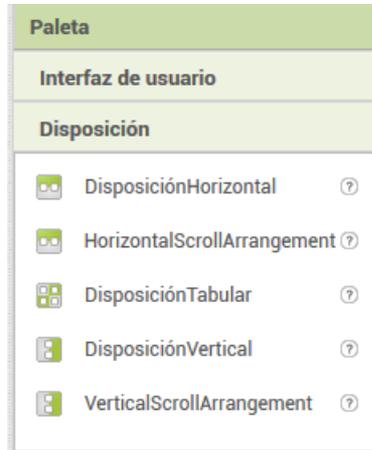
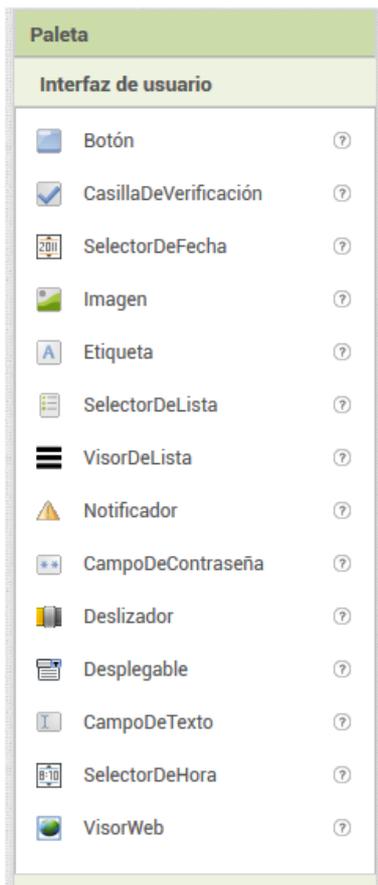


Ilustración 9: Opciones del menú Disposición.

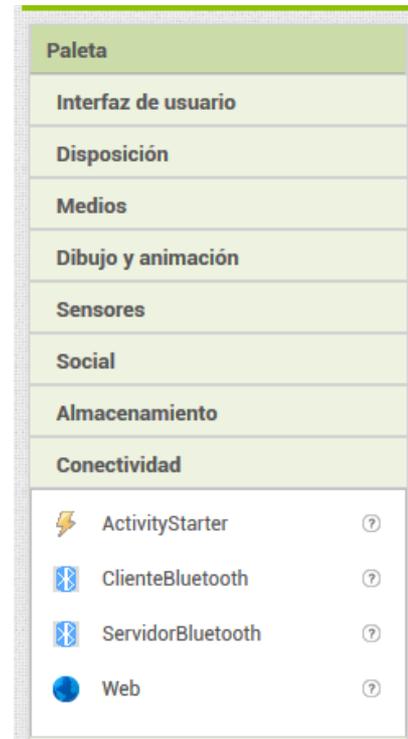


Ilustración 8: Opciones del menú Conectividad.

Ilustración 7: Opciones del menú Interfaz de usuario.

Para utilizar cada una de estas herramientas basta con hacer clic y arrastrar hasta la pantalla. Y una vez se tengan los elementos que se desea en la pantalla, éstos se pueden modificar a través del menú Propiedades situado a la derecha en el cual aparecerán todas propiedades disponibles para ese elemento en cuestión.

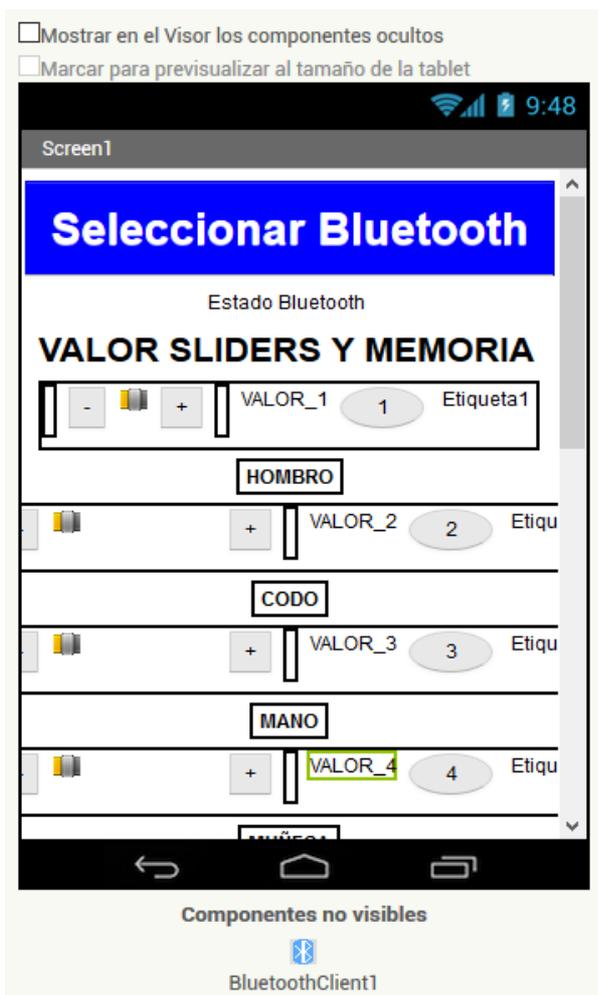


Ilustración 10: Aplicación Android. Vista superior.

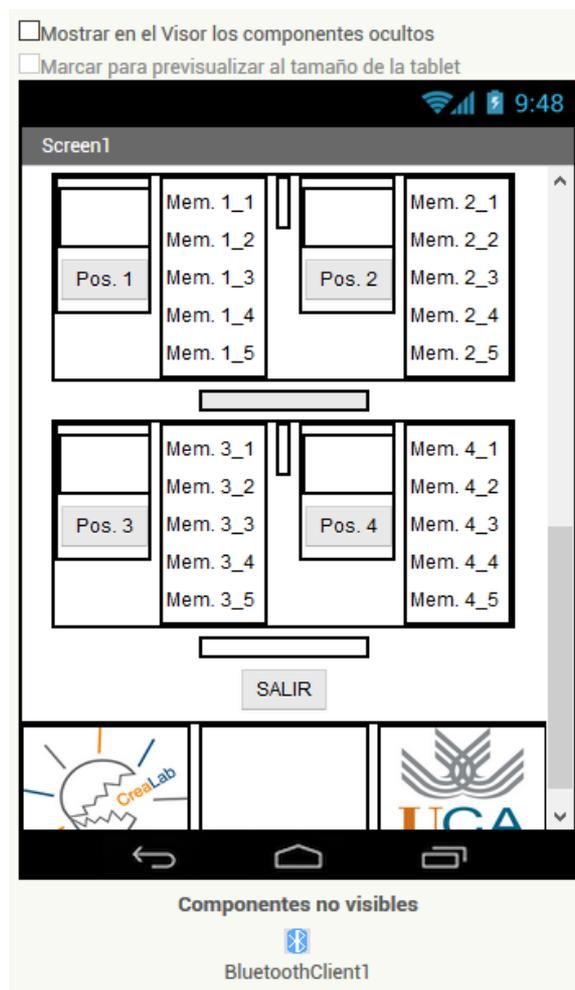


Ilustración 11: Aplicación Android. Vista inferior.

Lo primero que se encuentra en la aplicación es un “SelectorDeLista” al cual hemos llamado “Seleccionar Bluetooth”, esto no es más que un botón que al pulsar sobre él, mostrará una lista de textos entre los cuales podremos elegir uno. Si se pulsa sobre el componente a la derecha aparecerá un listado completo de todas las propiedades que se pueden modificar de este botón selector, desde tamaño de letra, color de fondo, tipo de letra, etc.

Éste selector de lista en concreto nos mostrará la lista de Bluetooth sincronizados con el dispositivo Android donde se encuentre instalada la aplicación, ésto se consigue por medio de la programación de bloques que se explicará más adelante.



Ilustración 12: Componentes. Selector de lista o “listPicker1”.

Lo siguiente que aparece es una “Etiqueta” nombrada como “Estado Bluetooth”. Al igual que el botón anterior, podemos modificar su tamaño, fuente, etc.

### Estado Bluetooth

*Ilustración 13: Componentes.  
Etiqueta Estado\_Bluetooth.*

Al iniciar la aplicación desde el dispositivo Android en la etiqueta aparecerá el nombre “Estado Bluetooth”, pero si se pulsa sobre el selector de lista y dentro de él, y se selecciona un dispositivo Bluetooth de dicha lista podrá aparecer uno de los dos siguientes mensajes en el lugar donde se encuentra “Estado Bluetooth”:

1. Conexión CORRECTA. (texto en color verde)
2. Sin Conexión. (texto en color rojo)

A continuación aparece una nueva etiqueta la cual se ha nombrado como “VALOR SLIDERS Y MEMORIA”, esa etiqueta se ha nombrado como “INFO” de forma interna en la parte de componentes, ya que se pueden modificar los textos de los componentes que usas para su diferenciación en el caso de usar varios elementos iguales.

## VALOR SLIDERS Y MEMORIA

*Ilustración 14: Componentes. Etiqueta INFO.*



*Ilustración 15: Componentes. Elementos de Disposición, Botones y Etiquetas.*

En la ilustración número 15 se pueden observar varios elementos de disposición, en concreto de disposición horizontal, no son más que rectángulos que se han usado para separar los elementos entre sí, así como ordenar en su interior el resto de elementos.

Observando de arriba abajo y de izquierda a derecha, en la ilustración anterior se pueden ver 2 elementos de disposición horizontal, el primero está relleno de otros elementos, como son a su vez: un elemento de disposición horizontal, un botón, un deslizador (o slider), un botón, un elemento de disposición horizontal, una etiqueta, un botón y por último una etiqueta; y el segundo elemento de disposición horizontal con una etiqueta en su interior, que nombrará el servo al cual controla.



Los elementos de disposición horizontal que se encuentran vacíos se han usado como separadores en su mayor parte. De estos elementos podemos modificar su tamaño en el menú de propiedades, su color de fondo, así como insertar imágenes en ellos entre otras cosas.

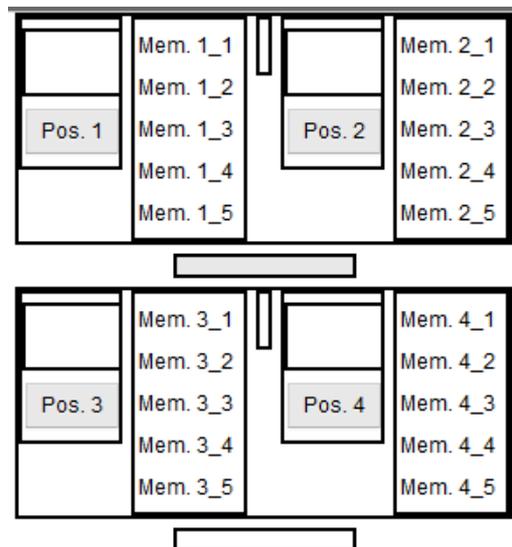
Con los respecto a los botones “+” y “-” se utilizan para mover el “deslizador” cuyos valores se han establecido para que oscilen entre 0 y 179, al pulsar sobre “+” se sumará un salto de 5 unidades con respecto a la posición del deslizador, y al pulsar sobre “-” se restará un salto de 5 unidades.

La etiqueta “VALOR\_X” nos informará de la posición numérica del deslizador, siendo “X” el número del servomotor que controle dicho deslizador.

El siguiente elemento que nos encontramos es un botón, al que se le ha dado forma ovalada para distinguirlo del resto de botones, ya que este botón, tiene una doble función. Si dicho botón se deja pulsado durante unos segundos, lo que se denomina en este entorno como “clic largo”, almacenará en memoria la posición del deslizador, es decir, el valor numérico que aparece en la etiqueta “VALOR\_X” será copiado en la siguiente etiqueta con nombre “EtiquetaX”. Si únicamente se pulsa el botón, y el deslizador se encuentra en una posición distinta a la que muestra el valor de “EtiquetaX” automáticamente la posición del deslizador cambiará y se moverá hasta el valor numérico que se corresponda con el que nos muestra dicha etiqueta, y por lo tanto el valor numérico que aparecerá nuevamente en la etiqueta “VALOR\_X” coincidirá con el de la etiqueta “EtiquetaX”.

Esto proceso es análogo para los 5 bloques siguientes, ya que con cada uno de estos bloques controlaremos uno de los servomotores del robot.

Lo siguiente que se puede encontrar en la aplicación, son varios bloques de disposición horizontal, en concreto 4, uno relleno, otro vacío, otro relleno y el último vacío.



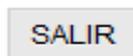
*Ilustración 16: Componentes. Elementos de disposición para memorizar varios estados.*

Tal como se puede apreciar en la ilustración los bloques son semejantes, a través de ello conseguimos memorizar las 5 posiciones de los servomotores en un único botón, se han dispuesto 4 de estos botones, para la configuración de varios estados del robot, por si por ejemplo se quiere secuenciar la carga y descargar de algún objeto. De esta forma bastaría con almacenar dichas posiciones y pulsar de forma secuencial dichos botones para que se realice dicha acción.

El modo de almacenar en memoria dichas posiciones es igual que antes, al dejar pulsado el botón se guarda en memoria la posición numérica de todos los servos, y al pulsar una única vez, si los deslizadores no estaban en esa posición irán automáticamente hacia ella. Para almacenar una nueva posición bastaría hacer nuevamente clic largo en el botón.

Al iniciar la aplicación dichas memorias estarán vacías, ya que al salir de la aplicación se libera la memoria utilizada para almacenar dichos valores.

Lo siguiente en la aplicación es un botón que se ha habilitado para salir de la aplicación, del tal forma que al pulsarlo volvemos al escritorio del dispositivo Android.



*Ilustración 17: Botón Salir de la aplicación*

Por último se han añadido un par de imágenes corporativas:



*Ilustración 19: Logo CreaLab.*



*Ilustración 18: Logo Universidad de Cádiz.*



### 5.2.2.2. Entorno MIT APP Inventor. Bloques

En la parte de “Bloques” o programación, se pueden encontrar dos partes bien diferenciadas:

1. Bloques
2. Visor

Dentro del menú bloques, aparece un menú llamado Integrados, el cual consta de: Control, Lógica, Matemáticas, Texto, Listas, Colores, Variables, Procedimientos, y además todos los elementos que se han arrastrado a la pantalla en la parte “Diseñador”.

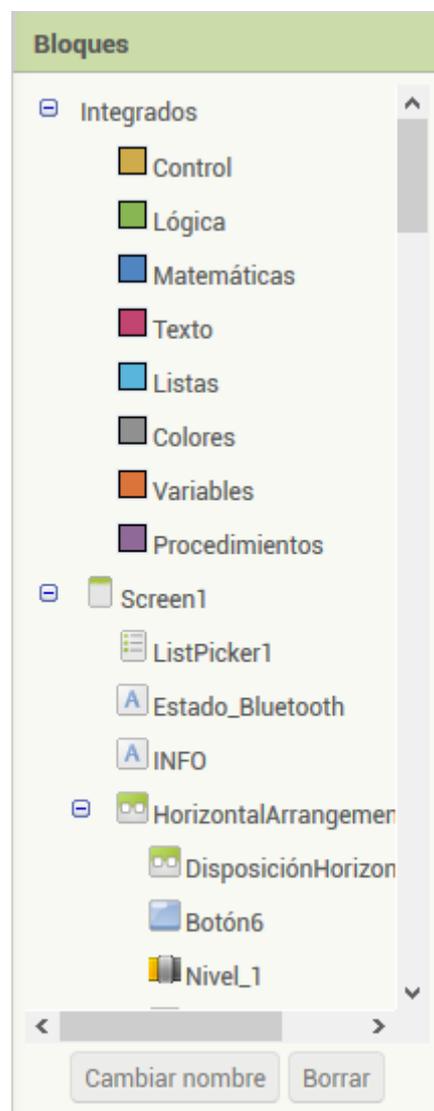


Ilustración 20: Menú Bloques.

Al hacer clic sobre cualquier elemento ubicado dentro de “Bloques” aparecerá un submenú con todas las opciones disponibles para esa “bloque”, como se puede observar en la siguiente imagen, tienen formas y colores distintos, pero la peculiaridad de dichos bloques, es que se pueden unir como si de un puzzle se tratase, de tal forma que la programación sea fácil e intuitiva para el usuario.

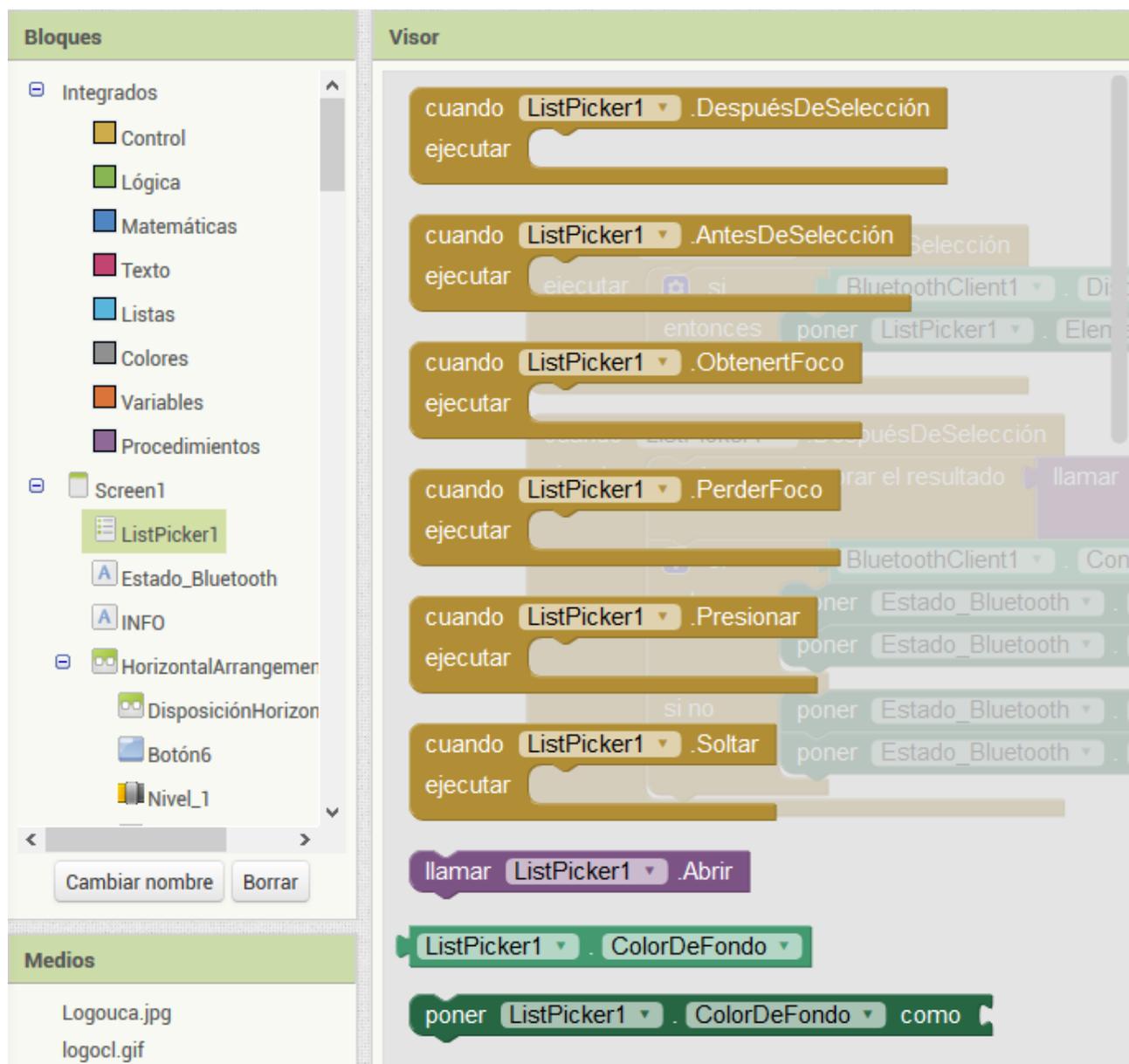


Ilustración 21: Bloque LisPicker1 y algunas de las opciones que nos muestra la Interfaz de programación.

Si se observa detenidamente la ilustración anterior, se puede apreciar cómo dentro de cada bloque de operaciones hay elementos con listas, por lo que se pueden hacer varias modificaciones dentro de cada bloque.



En la parte del “visor” será donde se añadan dichas piezas de puzzle con la que se va a realizar la programación de la App. El “visor” no tiene tamaño máximo, por lo que no importa el orden de colocación de los puzzles, pero siempre es mejor llevar cierto orden para poder identificar mejor los puzzles que no realicen correctamente la acción que hagan y así poder corregirlos.

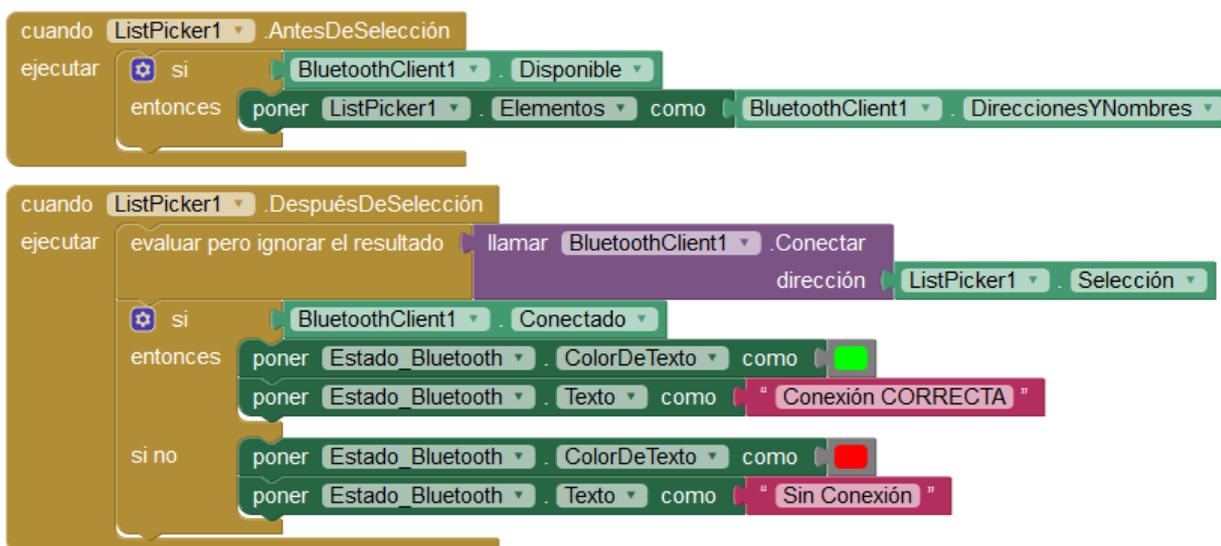
A continuación se describirán todos los puzzles que se han usado para el diseño de la App. Estos puzzles o bloques, se corresponden con la solución a los problemas o especificaciones de este proyecto, pero se debe mencionar que no existe una única solución a dichos problemas por lo que se podría utilizar otros elementos o funciones para resolverlos.



*Ilustración 22: Bloques. Salir de la aplicación.*

Tal como se aprecia en la ilustración 22, al hacer clic sobre el botón denominado “Salir”, se ejecutará la acción “cerrar aplicación”. Para ello en la sección de “Bloques”, se busca en el listado el botón “salir” y se despliegan todas las posibles opciones de uso para botones, y la primera de ellas, es la que se muestra en la ilustración 22.

Para el uso del Bluetooth se han usado dos puzzles:



*Ilustración 23: Bloques. Bluetooth.*

En el primer puzzle, el superior, se comprueba si el Bluetooth está activo, y en caso afirmativo se accede al listado de dispositivos vinculados a nuestro dispositivo Android. Dicha lista mostrará las direcciones MAC y nombres de esos dispositivos.

Por medio del segundo bloque la App muestra en pantalla si se ha podido vincular el Bluetooth del dispositivo, con el módulo Bluetooth al que se está llamando, en este caso, al módulo Bluetooth conectado a la placa Arduino.

Si la vinculación ha sido satisfactoria se mostrará el texto “**Conexión CORRECTA**”, pero si por el contrario no ha sido posible la vinculación entre elementos, el mensaje que se muestre será “**Sin Conexión**”.

En la siguiente ilustración se puede observar la programación para los deslizadores, en ella se puede observar que cuando cambie la posición de “Nivel\_1”, nombre que tiene asignado el deslizador encargado del servomotor 1, se ejecutará la siguiente sentencia de acciones:

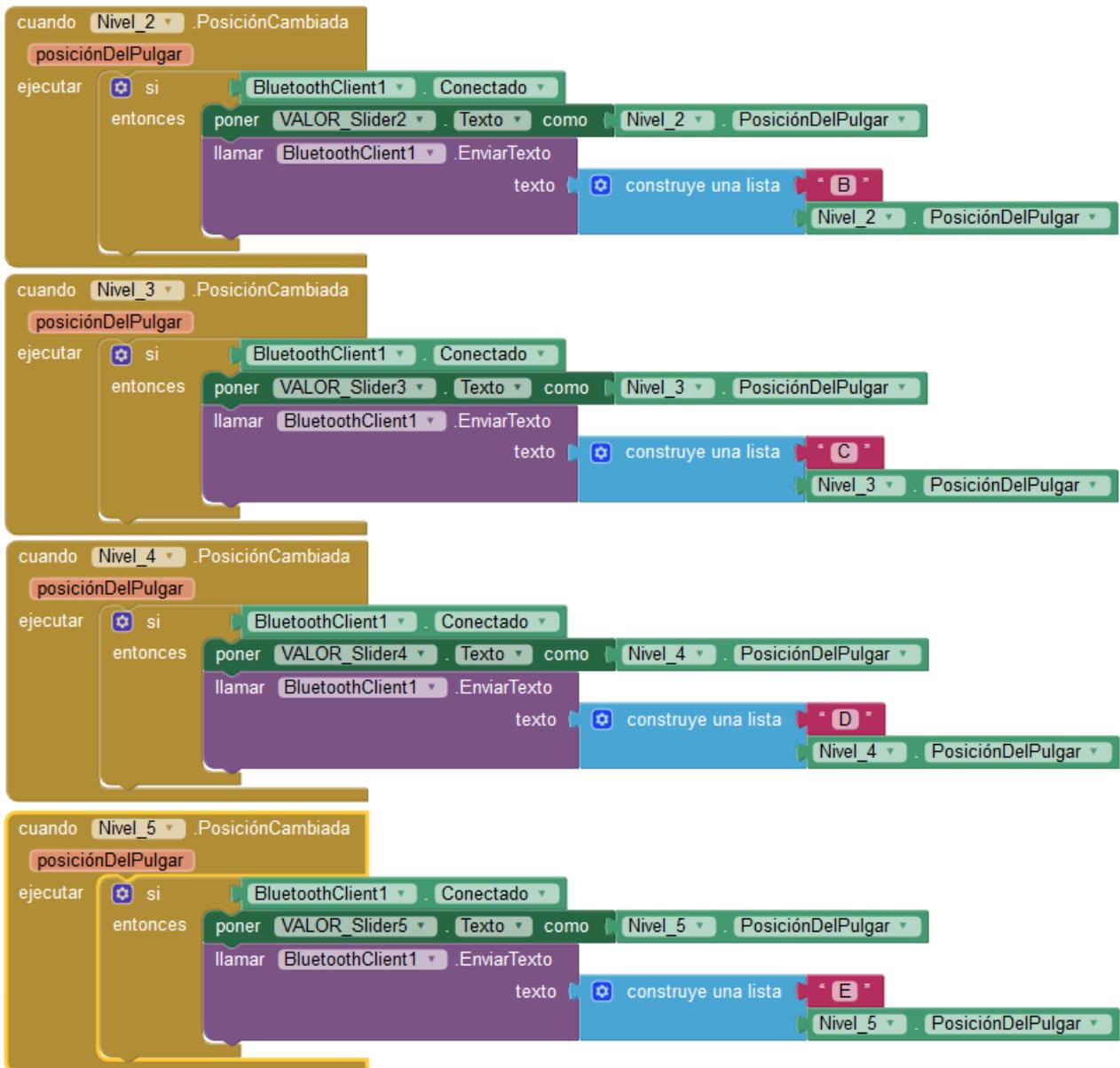
- Primero se comprueba la conexión mediante el cliente Bluetooth, es decir, que esté vinculado a la placa Arduino.
- En caso afirmativo se toma el valor numérico del deslizador 1 y se manda a través del módulo Bluetooth, previamente mandando el carácter ‘A’.



Ilustración 24: Bloques. Deslizador 1.

Se ha utilizado el carácter ‘A’ como una referencia, para diferenciar el movimiento de cada deslizador, por eso, en la parte de programación del código de Arduino, a leer el carácter A, se hacía una llamada a la función motor1().

Análogamente se han programado el resto de deslizadores, pero cada uno de esos puzzles envía un carácter distinto para diferenciar cada deslizador.



*Ilustración 25: Bloques. Deslizador 2 al Deslizador 5.*

Para la configuración del Botón que almacena en memoria la posición de cada deslizador se han utilizado los siguientes puzzles:

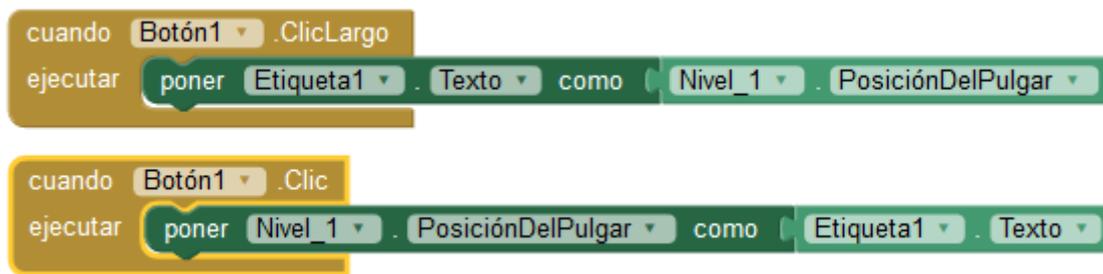


Ilustración 26: Bloques. Botón memoria Deslizador 1.

Como se puede observar en la ilustración, al hacer clic largo sobre “Botón1”, se guarda el texto de la posición del pulgar (posición del deslizador), en la “Etiqueta1”. Y al hacer clic sobre “Botón1” se cambia la posición del deslizador hacia la posición que indica el valor numérico almacenado en “Etiqueta1”.

De forma análoga se programa para el resto de botones de memoria de los deslizadores.



Ilustración 27: Bloques. Botones de memoria Deslizadores del 2 al 5.



A continuación se explica la programación de los botones "+" y "-". Para ello simplemente se ha utilizado una operación matemática, en la que al hacer clic sobre el botón "+", se toma la posición numérica del deslizador y se le suma 5, y para el caso del botón "-", se toma la posición numérica del deslizador y se le resta 5. Se le suman o restan 5 unidades para que el usuario al pulsar sobre dicho botón vea que la barra del deslizador se mueve, ya que dar saltos de 1 en 1 es poco significativo.



Ilustración 28: Bloques. Botones "+" y "-" Deslizador 1.



Ilustración 29: Bloques. Botones "+" y "-" Deslizadores 2 al 5.

Al pulsar sobre uno de estos botones se obliga al deslizador a cambiar su posición, por lo que esta parte de la programación interactúa el puzzle / programación de la ilustración 24, por la que al mover el deslizador se envía un carácter y la posición del deslizador a través del Bluetooth.

Ya sólo queda por explicar los puzzles de "Pos. 1", "Pos. 2", "Pos. 3", "Pos. 4". Estos botones están preparados para almacenar la información de los 5 servomotores al mismo tiempo, esto se ha realizado para poder ejecutar maniobras específicas de movimiento. El funcionamiento es idéntico al botón que almacena la posición de un único servomotor, con la única diferencia de que almacenará la posición de los 5 servomotores al hacer clic largo, y si nuevamente se pulsa sobre él hará que todos los servomotores vayan a las posiciones almacenadas en memoria.

Al hacer clic largo tomará el valor numérico de las etiquetas con las posiciones de los deslidores y las almacenará en las etiquetas de "Mem. X\_Y", haciendo referencia "X" al número del botón e "Y" al número del deslizador.

Al hacer clic se moverán los deslidores hasta las posiciones numéricas que contienen las etiquetas de "Mem. X\_Y", haciendo referencia "X" al número del botón e "Y" al número del deslizador.



Ilustración 30: Bloques. Botón memoria "Pos. 1".

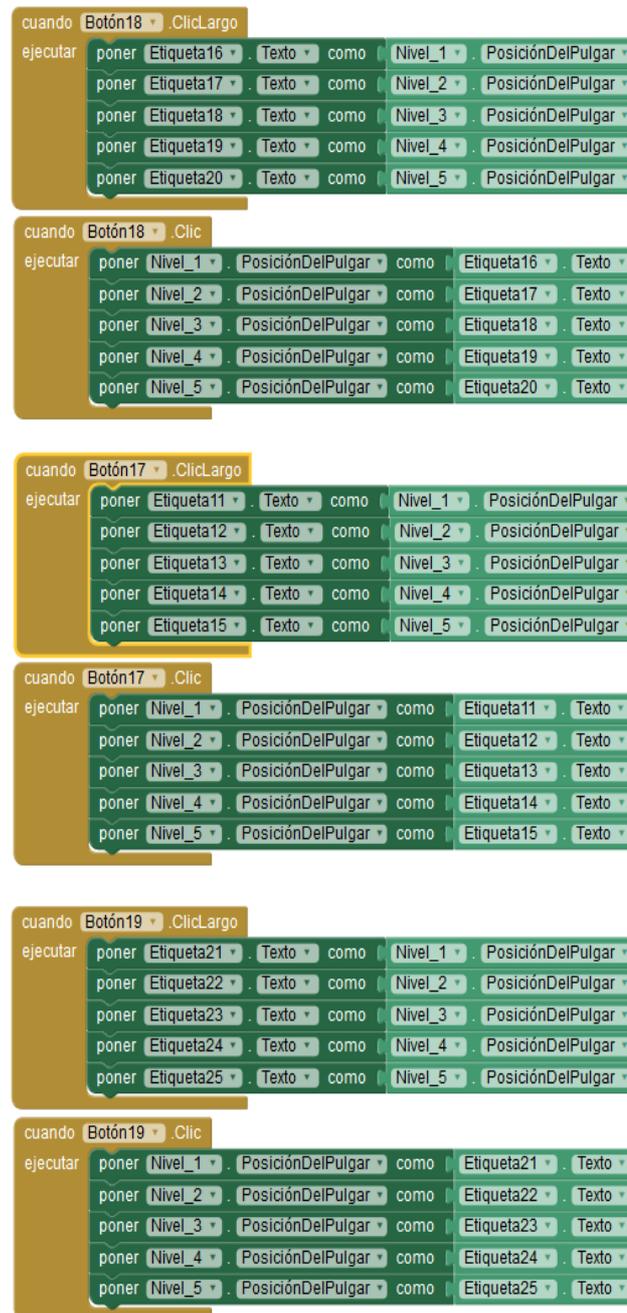


Ilustración 31: Bloques. Botones memoria "Pos. 2", "Pos. 3" y "Pos. 4".

Una vez finalizada la creación de la App, basta con pulsar el botón de generar el archivo ".APK", que será el instalador de nuestra aplicación, el cual se debe instalar en el dispositivo Android el cual se desee usar.

## 6. Anotaciones

En esta sección se comenta la evolución del proyecto así como las modificaciones que se han ido llevando a cabo y el porqué.

### 6.1. Pruebas con servomotores y potenciómetros.

Lo primero que se tuvo en cuenta para la realización del presente proyecto, fue la elección de los servomotores, y gracias a que en el laboratorio 111 se disponía de dos servomotores Futaba 3001HB, similares a los que se han usado en este trabajo (S3003), se hicieron pruebas para su control con potenciómetros de 10 K ohmios.

Al conectar los dos servomotores, 3001HB, a la placa Arduino y tratar de controlarlos mediante dos potenciómetros. Usando como alimentación el cable USB y conectándolo al ordenador (ver “Esquema Montaje. Pruebas Iniciales 1”), se observó que al mover los potenciómetros, los servomotores giraban de forma descontrolada, sin embargo, al alimentar la placa Arduino mediante una fuente de alimentación externa, cuyas salidas son de 12 voltios y 1000 mA, llega un momento en el que los servomotores se estabilizan al mover los potenciómetros. Esto ocurre debido a que Arduino tiene una única salida de 5 voltios, y una corriente limitada a 40 mA, por lo que controlar estos dos servomotores, de forma simultánea, no es posible, debido a que la corriente es limitada ya que en el momento en que a un servomotor se le aplica una carga, se produce un aumento de consumo, causando que la placa de Arduino se venga abajo.

La solución planteada para este problema fue el uso de reguladores de tensión, para intentar ofrecer la máxima tensión posible a los servomotores y así disponer de su máximo par posible.

### 6.2. Uso de fuentes de alimentación conmutadas.

Para solucionar el problema anteriormente expuesto, se implementó el uso de condensadores, y los reguladores de tensión, en concreto, los LM7805 (ver “Esquema Montaje. Pruebas Iniciales 2”). Así mismo, se necesitaban dos fuentes de alimentación una de 12 voltios para alimentar la placa que se había creado, y la segunda podría ser el cable USB para conectar la placa Arduino al PC.

Con esta solución se conseguía el control de ambos servomotores, con una tensión en cada servomotor de 5 voltios.

A medida que se profundizó en es estudio de componentes, (con consejo del tutor del proyecto, Don Carlos Corrales Alba), se optó por el uso de fuentes de alimentación conmutadas, en



concreto las fuentes DC – DC Step Down, basadas en los LM2596S, estos, a través de un condensador regulable son capaces de regular la tensión de salida en función de la tensión de entrada (ver “Esquema Montaje. Pruebas Iniciales 3”). Con esta solución sólo se necesitaba una única fuente de alimentación externa a 12 voltios, que no fuera la alimentación propia Arduino.

Así mismo, los reguladores tipo LM7805 tienen poca eficiencia, ya que tienen un rendimiento energético, para estos valores de tensión, del 41% (desperdician en forma de calor gran cantidad de energía consumida). Por otra parte, dado la forma de funcionamiento de las fuentes conmutadas basadas en los circuitos LM2596S, se puede llegar a una eficiencia del 85%.

### 6.3. Diseño en papel de las piezas del robot y primeras impresiones.

El siguiente paso fue la realización de diseño de las piezas del prototipo a papel, antes de plasmarlas en el ordenador usando AutoCAD. Se realizó un catálogo, a mano alzada, para ir viendo como serían las piezas que posteriormente se realizarían en AutoCAD.

A medida que se fueron dibujando e imprimiendo con la impresora 3D, fueron apareciendo ciertas anomalías, que se fueron corrigiendo. Y así, aprendiendo de esos fallos.

Con prueba y error, se corrigieron tanto formas de las piezas, grosores, etc. todo ello para llegar a lo que se estima un resultado final óptimo.

### 6.4. Módulo Bluetooth. Configuración y Comandos AT.

Una vez conseguido el control de los servomotores por medio de los potenciómetros, el siguiente paso fue el uso y control a través del Bluetooth. Al principio se trabajó de forma aislada con un único servomotor que no estaba montado en el prototipo final.

Para la configuración del módulo Bluetooth se utilizan los llamados comandos AT, y dependiendo del módulo que se disponga, se utilizan unos u otros códigos. En este trabajo, para la codificación se usó una placa Arduino independiente (ver “Esquema Montaje. Módulo Bluetooth”).

La velocidad de conexión de los módulos Bluetooth es normalmente de 9600 baudios, pudiendo ser modificada y para ello se utilizan los comando AT. Para configurar un módulo Bluetooth a través de los comandos AT la velocidad de comunicación habitual es de 38400 baudios.

El código que se usó para dicha configuración fue el siguiente:

```
#include <SoftwareSerial.h>  
SoftwareSerial BT(0, 1); // RX, TX
```

```

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  BT.begin(38400);

  Serial.println("configuracion!");
  // set the data rate for the SoftwareSerial port
}

void loop() { // run over and over
  if (BT.available()) {
    Serial.write(BT.read());
  }
  if (Serial.available()) {
    BT.write(Serial.read());
  }
}

```

Dicho código está algo modificado con respecto al código que trae el menú de ejemplos de Arduino. Por medio de este código se puede acceder al menú de configuración de los módulos de Bluetooth HC-05. Basta con quitar el cable de alimentación del módulo, presionar el botón pulsador, conectar la placa Arduino al PC y una vez enviado el código, conectar la alimentación, y sin dejar de pulsar el pulsador, se aprecia como cambia el modo de parpadeo del Bluetooth, de forma rápida a forma más pausada, es entonces cuando por medio del Monitor Serie de la aplicación Bluetooth podremos introducir los comandos AT, para su configuración, entre los cuales se podrá cambiar, el nombre, pin, velocidad de comunicación, etc.

AT COMMAND LISTING	
COMMAND	FUNCTION
AT	Test UART Connection
AT+RESET	Reset Device
AT+VERSION	Query firmware version
AT+ORGL	Restore settings to Factory Defaults
AT+ADDR	Query Device Bluetooth Address
AT+NAME	Query/Set Device Name
AT+RNAME	Query Remote Bluetooth Device's
AT+ROLE	Query/Set Device Role
AT+CLASS	Query/Set Class of Device CoD
AT+IAC	Query/Set Inquire Access Code
AT+INQM	Query/Set Inquire Access Mode
AT+PSWDAT+PIN	Query/Set Pairing Passkey
AT+UART	Query/Set UART parameter
AT+BIND	Query/Set Binding Bluetooth Address
AP+POLAR	Query/Set LED Output Polarity
AT+PIO	Set/Reset a User I/O pin

Tabla 5: Comandos AT.



## 6.5. Modos de estado y de uso.

El siguiente paso fue la creación de la aplicación Android, así como la unificación de la programación en un único código.

A la hora de unificar todo el código se decidió usar cuatro LEDs de distintos colores para identificar tanto el modo como el estado de funcionamiento del robot, siendo:

- LED Rojo → Modo Reposo.
- LED Verde → Modo Activo.
- LED Blanco → Control Manual. (Potenciómetros)
- LED AZUL → Control Inalámbrico. (Bluetooth)

Todas las pruebas iniciales se realizaron en una placa del tipo Arduino Mega, pero al finalizarlas se pudo comprobar que con una placa del tipo Arduino Uno se disponía de todos los puertos necesarios para el montaje del proyecto, por lo que finalmente se decidió usar dicha placa.

Con respecto al montaje final se decidió usar una fuente de alimentación conmutada DC – DC Step Down por cada dos servomotores del tipo S3003 y regularlos a 6 voltios, y el servomotor SG90 se conectó a una salida de tensión de 5 voltios de la placa Arduino (ver “Esquema Montaje. Final”). El motivo se ha explicado en el apartado anterior.

## 6.6. Estructura.

Todos los servomotores se fijaran por medio de tornillería, ya que dichos servomotores disponen de bastante fuerza, y, las piezas podrían salir despedidas.

## 6.7. Piezas Robotin.

En la siguiente tabla se exponen los tiempos y materiales de cada pieza. El software Repetier-Host, que se ha usado para la impresión de las piezas del robot.

Dicho software es gratuito y a través de él se pueden configurar las propiedades de impresión de las piezas, tales como relleno de la pieza, velocidades de impresión, uso de soportes para las piezas que sea necesario, así el escalado y muchos otros parámetros.

Este software realiza una separación por capas del modelo realizado en AutoCAD, de esta forma a la hora de imprimir, se va imprimiendo capa a capa.

Pieza	Tiempo estimado de Impresión	Número de capas	Total de Líneas	Filamento necesario (mm)	Relleno
Potenciómetro Nivel 1	1h:7m:11s	65	41780	4099	50,00 %
Servomotor Nivel 1	4h:40m:35s	196	174048	16887	50,00 %
Tapa Servomotor Nivel 1	26m:36s	10	6127	1589	50,00 %
Potenciómetro Nivel 2	1h:1m:0s	315	58164	3766	50,00 %
Servomotor Nivel 2	3h:16m:50s	153	64032	12548	50,00 %
Tapa Servomotor Nivel 2	26m:36s	10	6127	1589	50,00 %
Potenciómetro Nivel 3	35m:4s	125	13863	2086	50,00 %
Servomotor Nivel 3	6h:38m:20s	271	146741	24139	40,00 %
Tapa Servomotor Nivel 3	26m:36s	10	6127	1589	50,00 %
Potenciómetro Nivel 4	32m:49s	125	12560	1886	40,00 %
Servomotor Nivel 4	4h:36m:6s	150	89364	16882	40,00 %
Tapa Servomotor Nivel 4	26m:36s	10	6127	1589	50,00 %
Potenciómetro Nivel 5	42m:44s	125	16029	2457	40,00 %
Pinza Servomotor Nivel 5	13m:10s	40	8719	709	40,00 %
Servomotor Nivel 5	2h:24m:9s	218	90213	8673	40,00 %
Tapa Servomotor Nivel 5	15m:50s	13	5456	939	40,00 %

<b>Total tiempo Impresión</b>	<b>1d:3h:5m:12s</b>
<b>Total Filamento necesario (mm)</b>	<b>101427</b>
<b>Total Filamento necesario (m)</b>	<b>101,427</b>

*Tabla 6: Información sobre piezas del robot.*

Tal y como se aprecia en la tabla, para la totalidad de impresión de las piezas, siempre que no haya que repetir alguna debido a algún fallo de impresión se utilizará un total de 101,427 metros de filamento y se tardaría un tiempo de estimado de unas 27 horas, tiempo al que habría que sumarle los tiempos de calibración, así como los de quitar las piezas finalizadas y preparar la impresora para la siguiente impresión.



## 7. Esquemas

### 7.1. Esquema Montaje. Pruebas Iniciales 1.

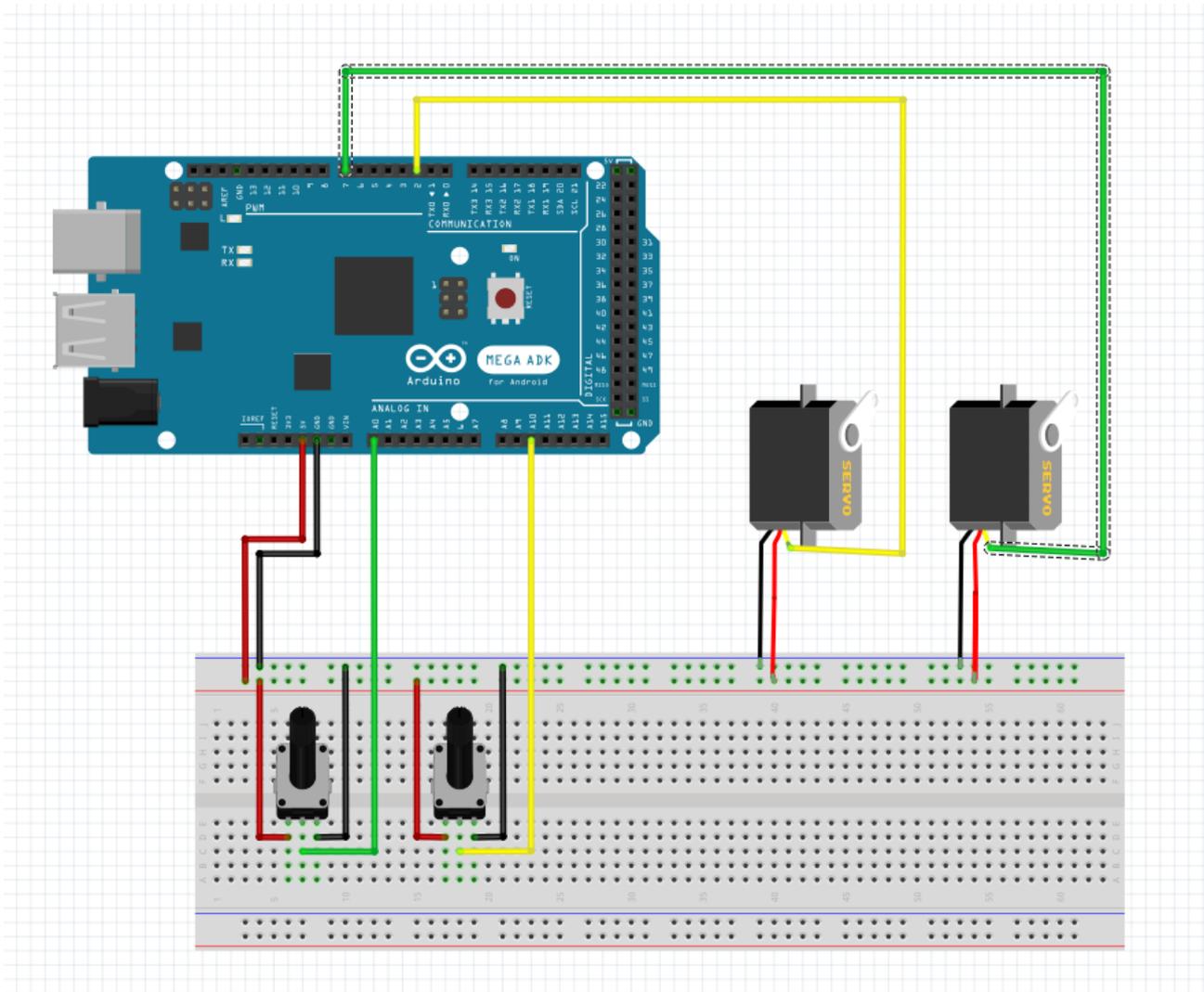


Ilustración 32: Esquema Montaje. Pruebas Iniciales 1.





## 7.2. Esquema Montaje. Pruebas Iniciales 2.

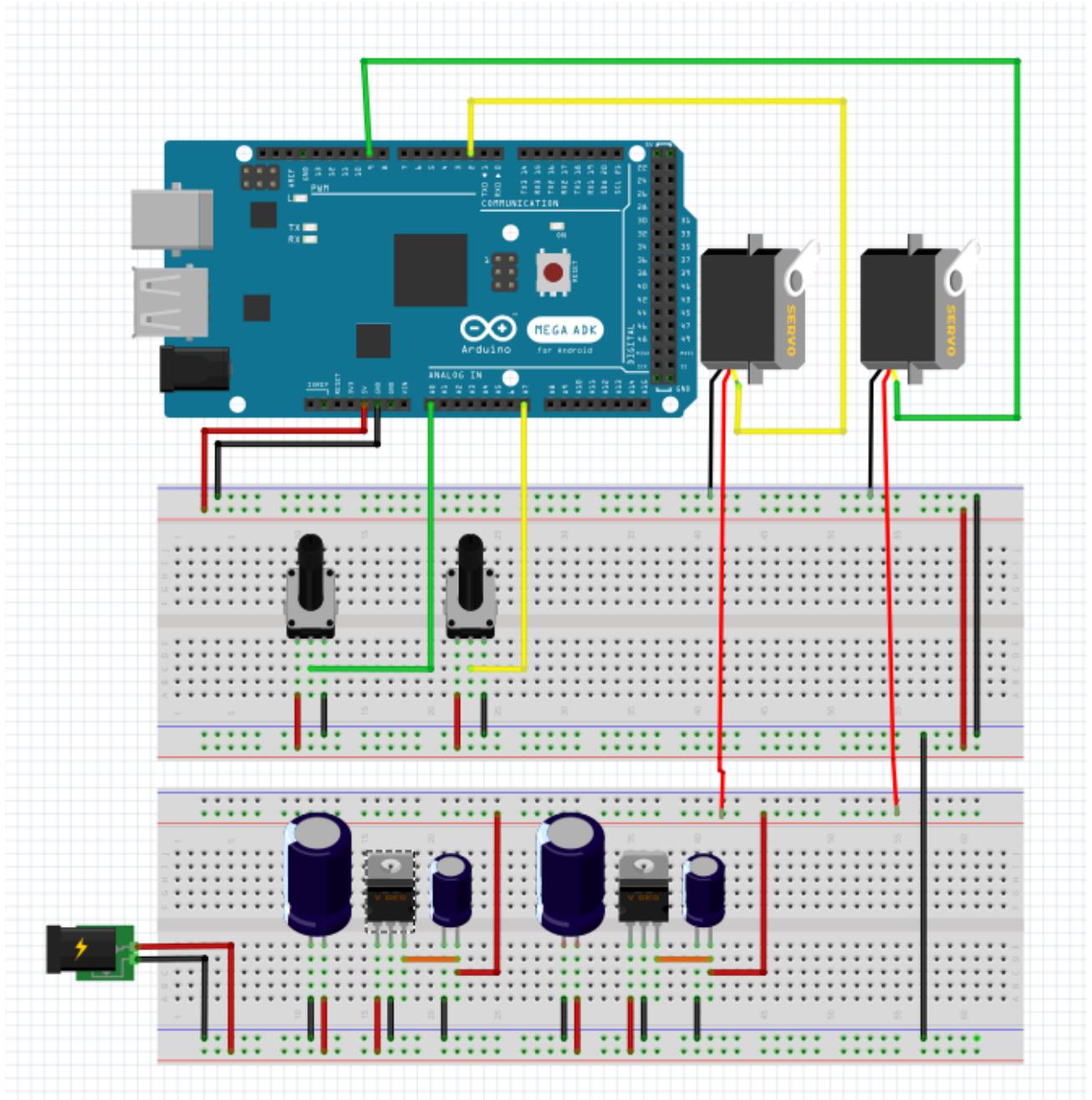


Ilustración 33: Esquema Montaje. Pruebas Iniciales 2.





### 7.3. Esquema Montaje. Pruebas Iniciales 3.

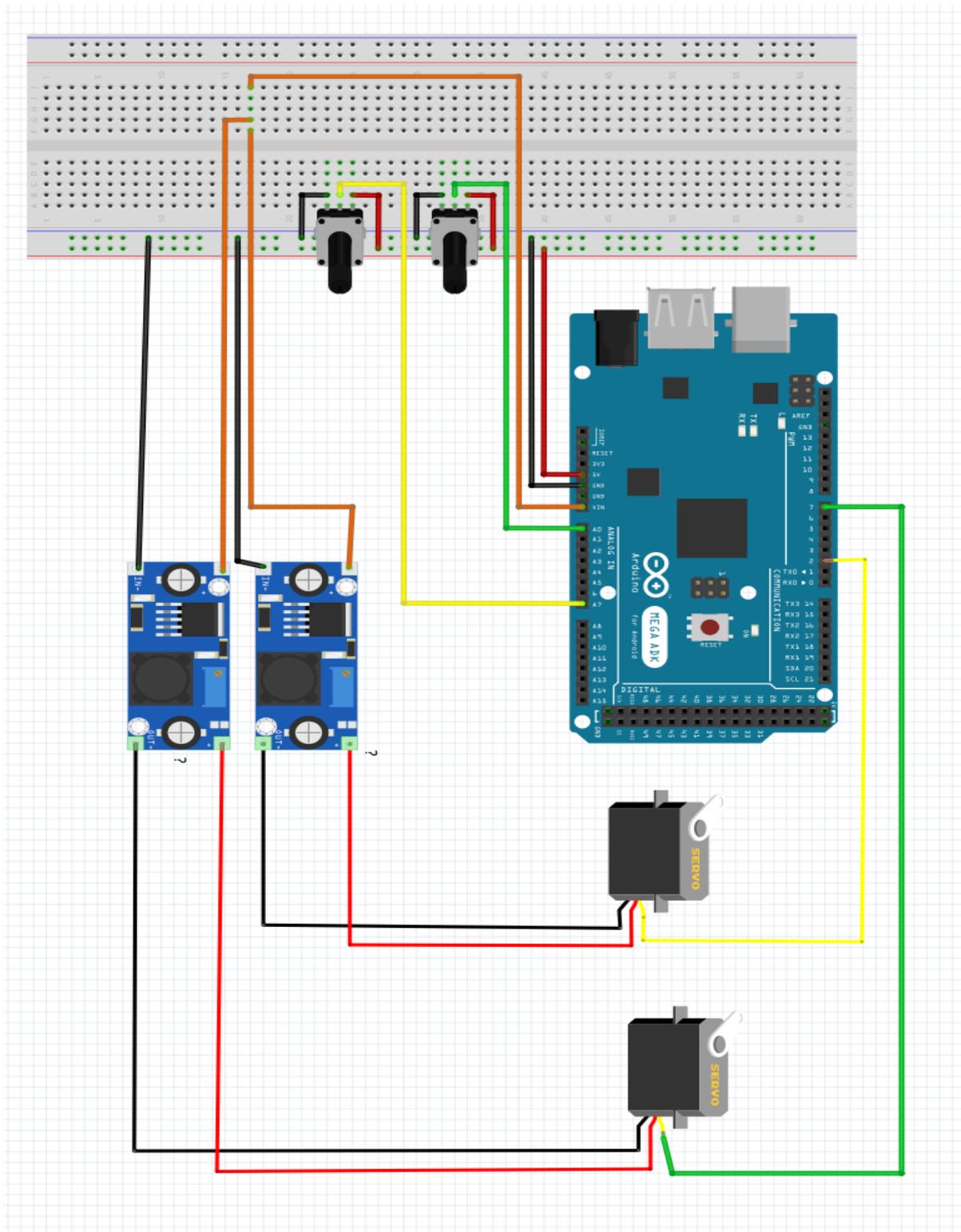


Ilustración 34: Esquema Montaje. Pruebas Iniciales 3.





## 7.4. Esquema Montaje. Configuración Módulo Bluetooth.

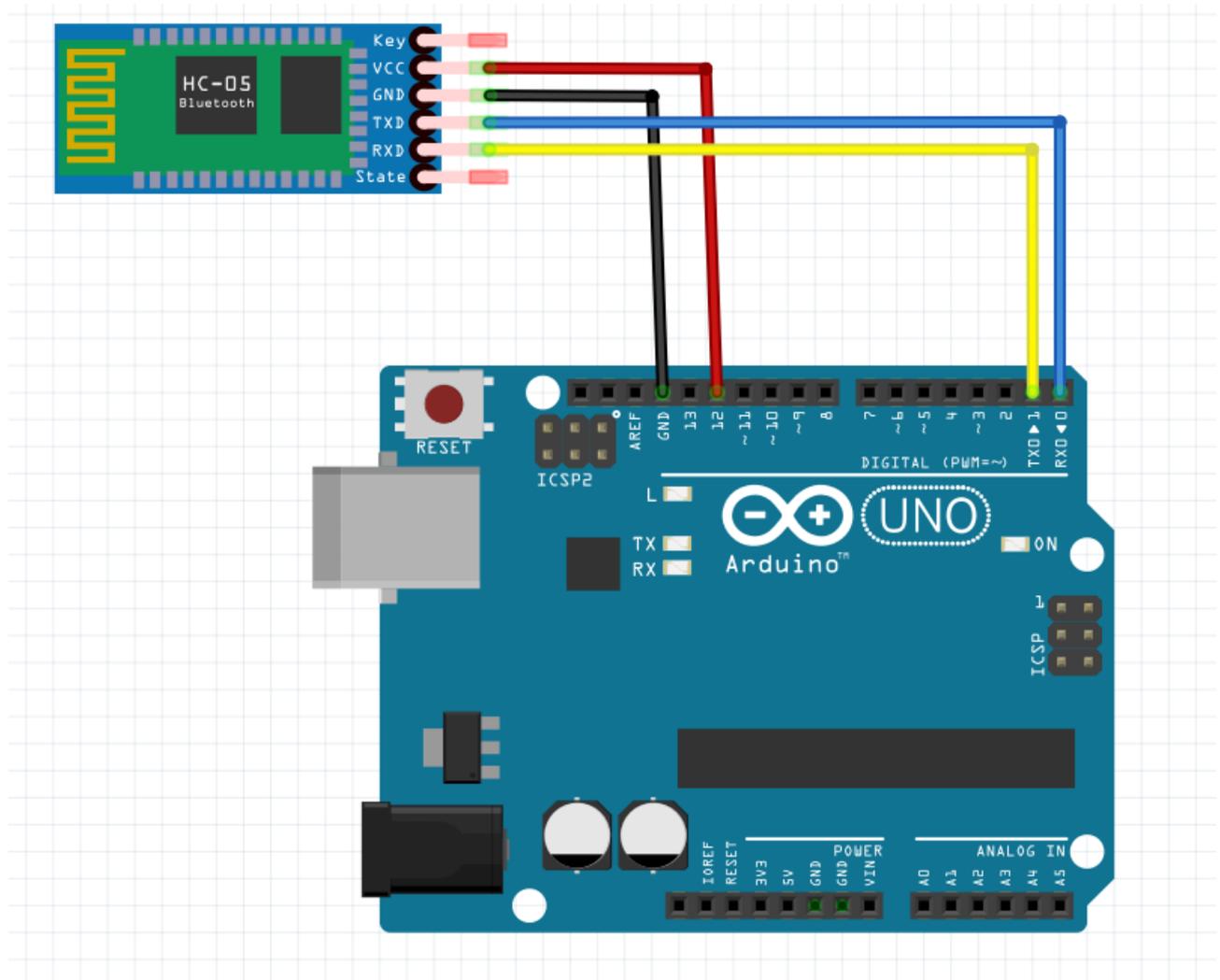


Ilustración 35: Esquema Montaje. Módulo Bluetooth.





## 7.5. Esquema Montaje. Final.

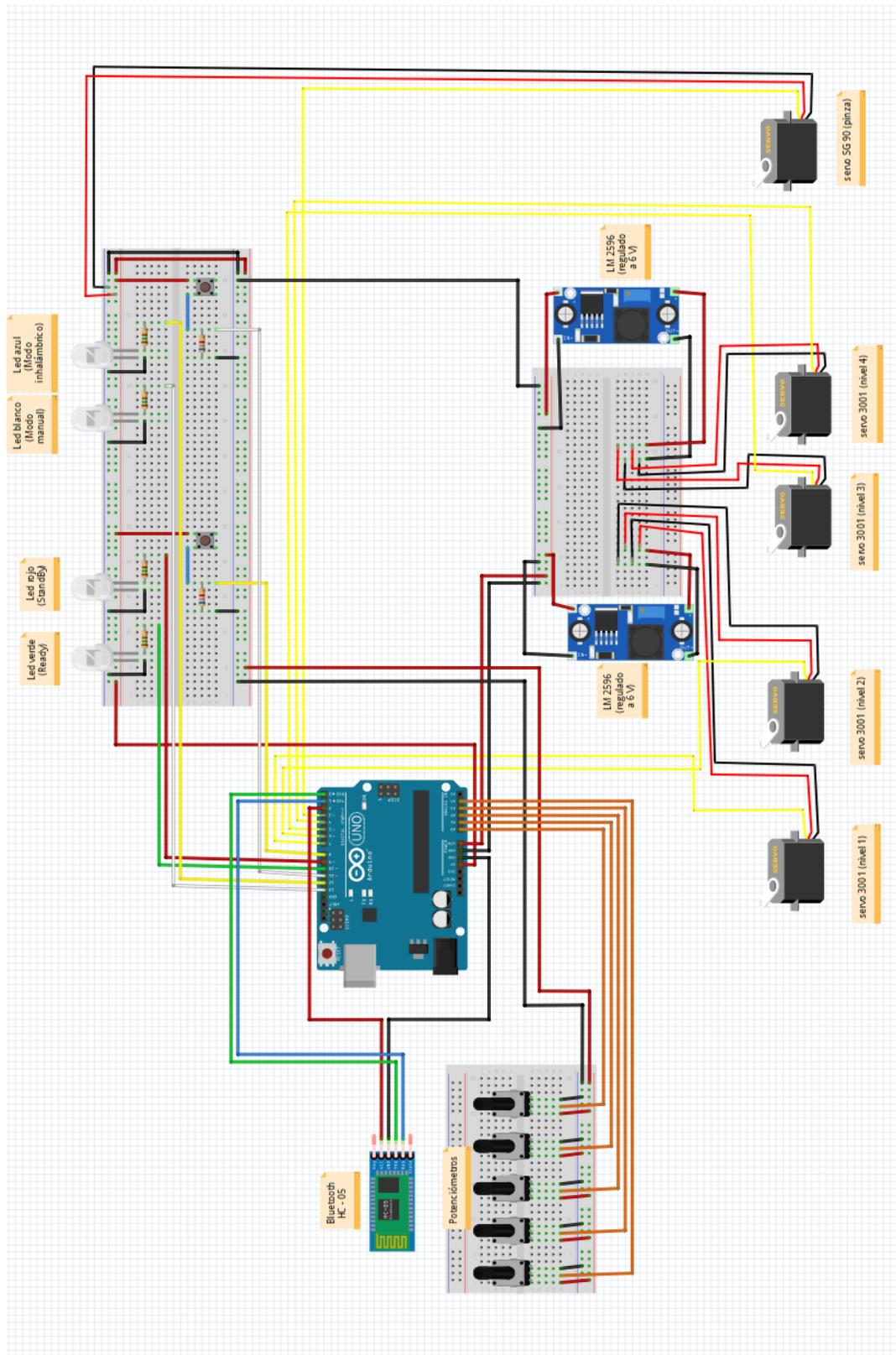


Ilustración 36: Esquema Montaje. Final.





## 7.6. Esquema control SG90.

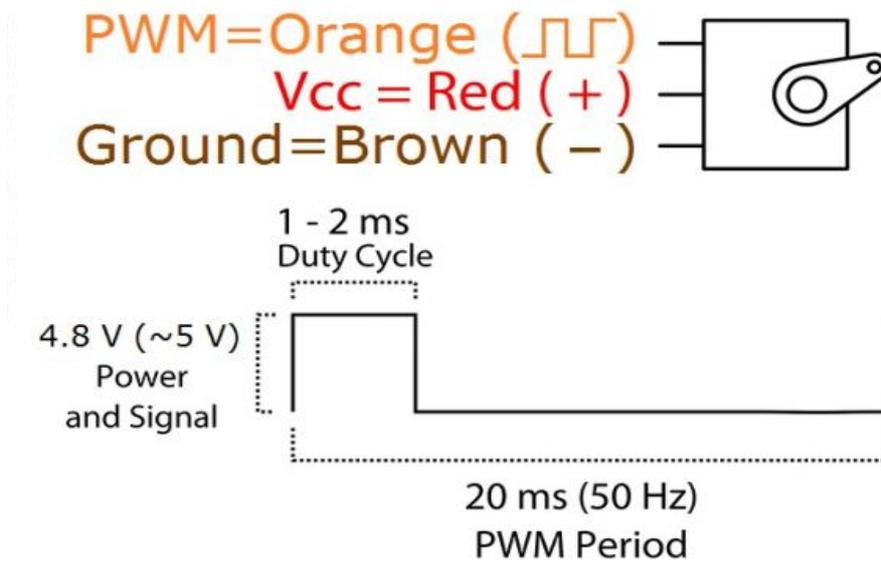


Ilustración 37: Esquema control SG90.

## 8. Materiales usados

### Impresora 3D



*Ilustración 38: PRUSA I3 HEPHESTOS de BQ*

El modelo de impresora que se ha utilizado, es la Prusa i3 Hephestos de BQ, gracias a que el departamento en Automática, Electrónica, Arquitectura y Redes de Computadores disponía de una unidad y se permitió su uso.

En cuanto a sus especificaciones, las más características son:

- Resolución máxima de 60 micras.
- Filamentos de 1,75 mm de diámetro: PLA, madera, bronce, cobre y Filaflex.
- Boquilla del extrusor de 0,4 mm.

### PLA

Las siglas PLA, vienen de la palabra Ácido Poli Láctico, que es un material biodegradable.

Este recurso será el que se utilice para la creación de las piezas que se utilicen en el presente proyecto.

Tal como se aprecia en la imagen, existen filamentos de varios colores.



*Ilustración 39: Rollos de PLA.*



## Servomotor SG90



Un servomotor lo define la Real Academia Española (RAE) como un sistema electromecánico que amplifica la potencia reguladora.

Este pequeño servomotor dispone de 180 grados de libertad para su rotación, y dispone de un torque máximo de 1,6 kg por cm.

*Ilustración 40: Servomotor SG90*

## Servomotor S3003

Dispone de 180 grados de libertad, con un torque mínimo de 3,17 kg – cm para una tensión de 4.8 voltios y un torque máximo de 4,10 kg – cm a 6 voltios.



*Ilustración 41: Servomotor S3003*

## Potenciómetro

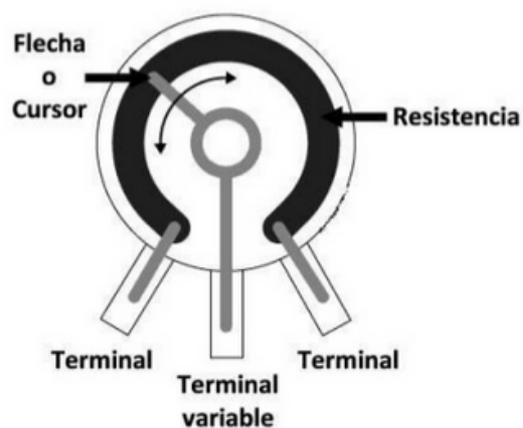


Según la RAE, un potenciómetro no es más que una resistencia regulable en un circuito eléctrico.

Para el proyecto se han utilizado potenciómetros lineales de 10 K de resistencia.

Consta de tres terminales, siendo el del centro, el que controla la señal variable, en función del material resistivo por el que está formado el potenciómetro; y por medio de los otros terminales se establece la polaridad del potenciómetro.

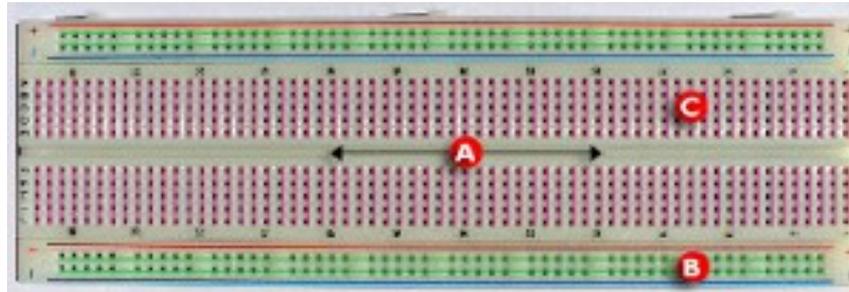
*Ilustración 42: Potenciómetro.*



*Ilustración 43: Esque potenciómetro.*



## Breadboard



*Ilustración 44: Breadboard.*

La Breadboard o protoboard es un tablero con orificios, en la cual se pueden insertar componentes electrónicos y cables para montar circuitos.

Se divide en tres regiones:

- A) **Canal central:** Situada en la zona central, se utiliza para colocar los circuitos integrado.
- B) **Buses:** Los podemos localizar en ambos extremos, se representan normalmente por líneas rojas (positivos o de voltaje) y azules (negativos o de tierra). Normalmente aquí es donde se colocan los terminales de las fuentes de alimentación.
- C) **Pistas:** Localizadas en la parte central, se representan y conducen según las líneas rosas (verticalmente).

## Cableado



Compuesto por un hilo o conjunto de hilos metálicos que sirven como conductor, y que poseen una envoltura aislante.

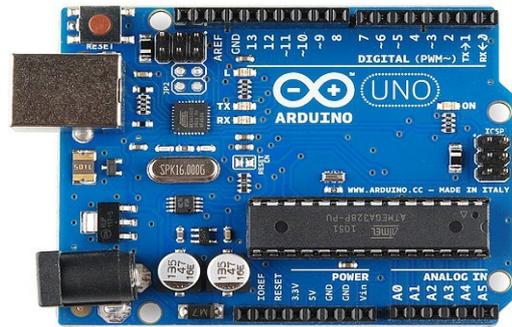
En función del terminal, dicho conjunto de cables los podemos encontrar como:

- Macho – macho.
- Macho – hembra.
- Hembra – hembra.

*Ilustración 45: Cables Jumper.*

## Arduino UNO

Es la placa hardware que se ha decidido utilizar para este proyecto debido a su pequeño tamaño y grandes prestaciones.



*Ilustración 46: Placa Arduino UNO.*

## LED



Las siglas del acrónimo LED, vienen de Light Emitting Diode, que traducido significa Diodo Emisor de Luz. Consiste básicamente en un material semiconductor que es capaz de emitir una radiación electromagnética en forma de luz.

*Ilustración 47: LED.*

## Resistencias

Componente electrónico diseñado para introducir una resistencia eléctrica determinada entre dos puntos de un circuito.



*Ilustración 48: Resistencia.*



## Fuente de alimentación conmutadas



El módulo DC – DC Step Down usado para este proyecto, está basado en los reguladores de tensión LM2596S, el cual nos permite regular la salida en función de la señal de entrada.

En nuestro caso, la señal de entrada es de 12 voltios y a la salida se regulaba para obtener 6 voltios.

Ilustración 49: DC - DC Step Down.

## Botón pulsador



Ilustración 51:  
Botón pulsador.



Pulsador normalmente abierto

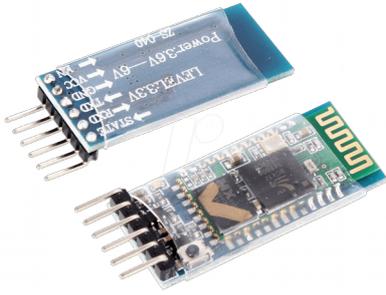
Ilustración 50: Esquema control pulsador.

Es un interruptor que al ser accionado de forma manual cambia de estado y al soltarlo regresa a su estado inicial.

<b>Funcionamiento</b>	OFF-ON (abierto en reposo)
<b>Carga</b>	12 VDC / 50 mA
<b>Vida útil</b>	$1 \times 10^5$ op.
<b>Fuerza de maniobra</b>	130 g
<b>Resistencia de contacto máxima</b>	30 M $\Omega$
<b>Carrera</b>	0'3 mm

Tabla 7: Características comunes botón pulsador.

## HC-05



*Ilustración 52: HC-05 Modulo Bluetooth.*

Módulo Bluetooth para trabajar tanto en modo maestro como en modo esclavo.

Utiliza protocolo de comunicaciones serie, UART RS 232.

Por defecto la velocidad e trabajo del módulo es de 9600 baudios, pudiéndose modificar.



## 9. Presupuesto

<i>Nombre</i>	<i>Robotín</i>		
<b>Materiales</b>	<b>Cantidad</b>	<b>Precio unidad</b>	<b>Total</b>
Kit cableado	10	2,48 €	24,80 €
LED	4	0,10 €	0,40 €
Fuente DC-DC Step Down	2	4,20 €	8,40 €
Arduino UNO	1	20,66 €	20,66 €
Breadboard	3	2,89 €	8,67 €
HC-05 (Módulo Bluetooth)	1	8,02 €	8,02 €
Kit botones pulsadores	1	1,65 €	1,65 €
kit resistencias	1	0,99 €	0,99 €
PLA	2	20,00 €	40,00 €
Potenciómetros	5	1,00 €	5,00 €
Servo SG90	1	2,81 €	2,81 €
Servo S3003	4	4,96 €	19,84 €
Mano de obra	30	10,00 €	300,00 €
		<b>Total</b>	<b>441,24 €</b>
		<b>Tipo IVA</b>	<b>21,00 %</b>
			92,66 €
		<b>Total Presupuesto</b>	<b>533,90 €</b>

La impresora 3D no formaría parte del presupuesto, pero si no se dispone de una en propiedad habría que arrendar los servicios de una para la impresión del material.

## 10. Competencias

El objetivo de un Trabajo Fin de Grado es alcanzar el mayor número de competencias de la titulación. Con tal fin, en el presente trabajo se han pretendido cumplir las siguientes competencias del Grado en Ingeniería Radioelectrónica impartido por a Universidad de Cádiz:

B1. Capacidad para la resolución de los problemas matemáticos que puedan plantearse en la ingeniería. Aptitud para aplicar los conocimientos sobre: álgebra lineal; geometría; geometría diferencial; cálculo diferencial e integral; ecuaciones diferenciales y en derivadas parciales; métodos numéricos; algorítmicos numéricos; estadísticos y optimización.

B3. Conocimientos básicos sobre el uso y programación de ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.

B5. Capacidad de visión espacial y conocimiento de las técnicas de representación gráfica, tanto por métodos tradicionales de geometría métrica y geometría descriptiva, como mediante las aplicaciones de diseño asistido por ordenador.

C1. Conocimiento, utilización y aplicación al buque de los principios de teoría de circuitos y máquinas eléctricas marinas.

C2. Conocimiento, utilización y aplicación al buque de los principios de electrónica aplicada al buque e instalaciones marinas.

C11. Conocimiento, utilización y aplicación al buque de los principios de legislación y normativa marina.

E1. Conocimientos en materias fundamentales y tecnológicas, que le capaciten para el aprendizaje de nuevos métodos y teorías, así como que le doten de una gran versatilidad para adaptarse a nuevas situaciones.

E2. Capacidad para resolver problemas con iniciativa, toma de decisiones, creatividad, razonamiento crítico y de comunicar y transmitir conocimientos habilidades y destrezas.

E7. Capacidad para el manejo de especificaciones, reglamentos y normas de obligado cumplimiento.

E12. Conocimientos y capacidad para calcular, diseñar y proyectar, de acuerdo con el Convenio STCW, circuitos y sistemas de comunicaciones.



E13. Conocimientos y capacidad para calcular, diseñar y proyectar, de acuerdo con el Convenio STCW, esquemas, planos de circuitos, sistemas e instalaciones eléctricas y electrónicas.

E15. Conocimientos y capacidad para calcular, diseñar y proyectar, de acuerdo con el Convenio STCW, normas, especificaciones técnicas de componentes, circuitos y sistemas electrónicos, automatismos.

E19. Conocimientos y capacidad para calcular, diseñar y proyectar, de acuerdo con el Convenio STCW, gestión del mantenimiento de sistemas eléctricos y electrónicos.

W14. Capacidad de toma de decisiones.

## 11. Conclusiones

Es un trabajo original, con aplicación en la docencia de microcontroladores, electrónica, automatización, programación y diseño gráfico.

En la construcción de los distintos elementos físicos, se ha visto que de los distintos polímeros que soporta la impresora 3D, el PLA es el más adecuado por la rigidez que presenta, aumento la robustez del mismo.

Uno de los problemas más importantes encontrado en este trabajo es la búsqueda de un compromiso entre el tamaño y la forma de las piezas del brazo y el par motor producido por los servomotores. Un tamaño pequeño quita robustez y aplicabilidad, y uno grande hace muy complicado el movimiento.

La colocación de los potenciómetros del brazo de control de forma vertical, uno encima del otro, hace que el sistema gane en sencillez y robustez, pero por el contrario, pierda en maniobrabilidad y ergonomía.

Existen multitud de herramientas gráficas para el diseño de piezas en 3D, se utilizó AutoCAD debido a su amplia gama de herramientas así como por su compatibilidad con el formato ".stl" (proviene de STereoLithography), necesario para la impresión de modelos en 3D.

Repetier-Host se utilizó para la impresión de las piezas por su sencillez y multitud de configuraciones, permitiendo la creación de soportes para las piezas de difícil impresión.

## 12. Referencias

### 12.1 Presupuesto

*Bq Hephestos Impresora 3D*

[https://www.pccomponentes.com/bq-hephestos-impresora-3d-roja?gclid=Cj0KCQjwub7NBRDJARIsAP7wIT8sJfFKy7WtZ04iRd6ZRitsxtIVY2TJA2UrOPwuHxOz7uQchmvv\\_HUaAuGvEALw\\_wcB](https://www.pccomponentes.com/bq-hephestos-impresora-3d-roja?gclid=Cj0KCQjwub7NBRDJARIsAP7wIT8sJfFKy7WtZ04iRd6ZRitsxtIVY2TJA2UrOPwuHxOz7uQchmvv_HUaAuGvEALw_wcB)

*Microservo SG90*

<https://fabricadigital.org/productos/microservo-sg90/>

*Servo S3003*

<https://fabricadigital.org/productos/servo-s3003-de-37g/>

*Cables Jumper para prototipar*

<https://fabricadigital.org/productos/20-cables-jumper-para-prototipar/>

*LED*

<https://fabricadigital.org/productos/led-blanco-difuso-de-5mm/>

*HC-05*

<https://fabricadigital.org/productos/modulo-bluetooth-puente-serie-masteresclavo-hc-05/>

*Fuente DC-DC Step Down*

<https://www.promotec.net/producto/regulador-lm2596s-dc-dc-step-down/>

*Breadboard*

<https://fabricadigital.org/productos/breadboard-400-puntos/>

*Kit resistencias*

<https://fabricadigital.org/productos/resistencias-de-pelicula-de-carbono-0-25w-5/>

*Cableado*

<https://fabricadigital.org/productos/20-cables-jumper-para-prototipar/>

*Arduino UNO*

<https://fabricadigital.org/productos/genuino-uno-rev-3/>

*LED*

<https://fabricadigital.org/productos/led-blanco-difuso-de-5mm/>



*Resistencias*

<https://fabricadigital.org/productos/resistencias-de-pelicula-de-carbono-0-25w-5/>

*Botón pulsador*

<https://fabricadigital.org/productos/boton-switch/>

## 12.2 Información

*Pines Microprocesador ATmega328P – Arduino UNO*

<https://comprendiendoarduino.wordpress.com/2016/02/01/armar-un-arduino-uno-en-una-protoboard/>

*Lista de librerías Arduino*

<https://www.arduino.cc/en/Reference/Libraries>

*Características variables C++*

<http://es.ccm.net/faq/3169-las-variables-en-c>

*Operaciones lógicas en C*

<https://www.jmramirez.pro/codigofuente/016-los-logicos-y-enumerados/>

*HC-05*

<https://electronilab.co/tienda/modulo-bluetooth-hc-05-serial-rs232/>

*Comandos AT módulo Bluetooth HC-05*

<http://www.prometec.net/bt-hc05/>

*HC-05. DATASHEET*

[http://cdn.makezine.com/uploads/2014/03/hc\\_hc-05-user-instructions-bluetooth.pdf](http://cdn.makezine.com/uploads/2014/03/hc_hc-05-user-instructions-bluetooth.pdf)

*Referencias comandos AT*

<http://cdn.instructables.com/ORIG/FKY/Z0UT/HX7OYY7I/FKYZ0UTHX7OYY7I.pdf>

*Prusa I3 Hephestos de BQ*

<https://www.bq.com/es/hephestos-prusa>

*PLA*

<http://www.eis.uva.es/~biopolimeros/alberto/pla.htm>

*Definición servomotor*

<http://dle.rae.es/srv/fetch?id=Xi6qRXB>

*Información SG90*

<https://fabricadigital.org/productos/microservo-sg90/>

*Información S3003*

<https://fabricadigital.org/productos/servo-s3003-de-37g/>

*Definición potenciómetro*

<http://dle.rae.es/srv/search?m=30&w=potenci%C3%B3metro>

*Arduino UNO*

<https://store.arduino.cc/usa/arduino-uno-rev3>

*LED*

<https://www.mastermagazine.info/termino/5554.php>

*Resistencias*

<https://es.wikipedia.org/wiki/Resistor>

*Fuente alimentación conmutada*

<http://www.prometec.net/fuentes-step-down/>

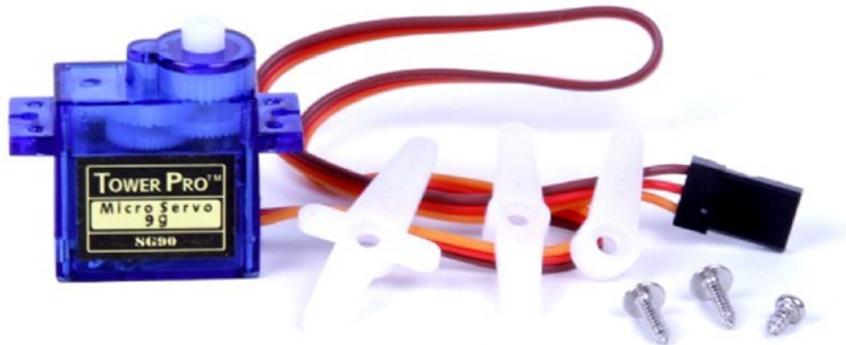


# 13. Anexos

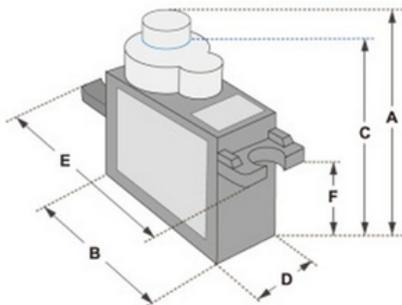
## 13.1. Data Sheet servomotor SG90

### SERVO MOTOR SG90

### DATA SHEET

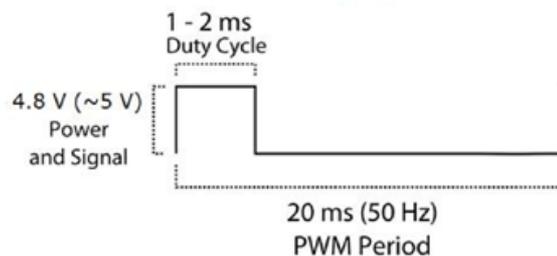
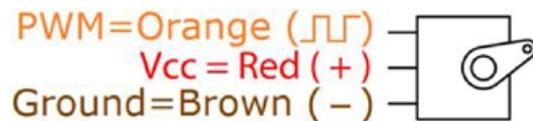


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building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



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

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

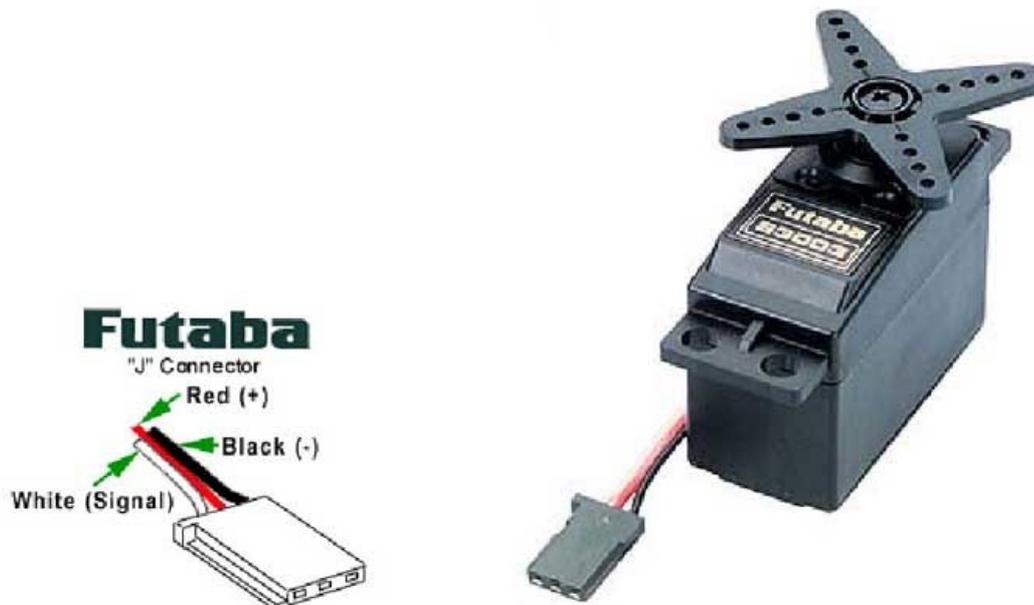






## 13.2. Data Sheet servomotor S3003

# S3003 FUTABA SERVO



### Detailed Specifications

Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 15201900usec
Operating Temperature Range:	-20 to +60 Degree C	Motor Type:	3 Pole Ferrite
Operating Speed (4.8V):	0.23sec/60 degrees at no load	Potentiometer Drive:	Indirect Drive
Operating Speed (6.0V):	0.19sec/60 degrees at no load	Bearing Type:	Plastic Bearing
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)	Gear Type:	All Nylon Gears
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)	Connector Wire Length:	12"
Operating Angle:	45 Deg. one side pulse traveling 400usec	Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
360 Modifiable:	Yes	Weight:	1.3oz. (37.2g)