Escuela Técnica Superior de Ingeniería y Diseño Industrial
Universidad Politécnica de Madrid

Memoria

Curso 2017/2018 - Grupo A-109

Datos de los miembros del grupo

Nombre	Apellidos	Matricula	Email
Sani	Mitkov Chandarov	54094	sani.mitkov.chandarov@alumnos.upm.es
Víctor	Peinado López	54146	v.peinado@alumnos.upm.es
Gabriel	Moreyra López	54117	gabriel.moreyra.lopez@alumnos.upm.es

Título y resumen

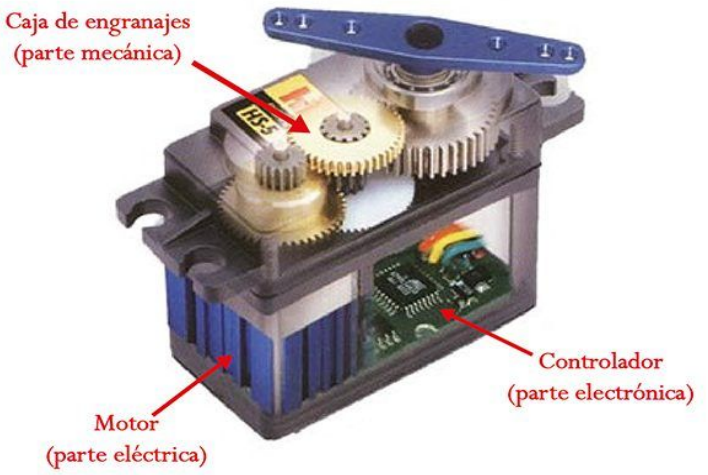
Sistema electrónico basado en un microcontrolador Arduino capaz de mover de forma controlada un brazo robótico. El objetivo del proyecto es llegar a mover el brazo mediante una secuencia programada para que sea capaz de llevar un objeto de una posición a otra,posteriormente se le añadirá un aplicación para poder controlar el brazo robótico desde un smartphone por bluetooth. El brazo tendrá 5 grados de libertad y estará compuesto por tres articulaciones controladas de manera simultánea para conseguir movimientos precisos y rápidos. Una pinza en el extremo del brazo será la encargada de sujetar el objeto y llevarlo de un lugar a otro.

Requisitos funcionales

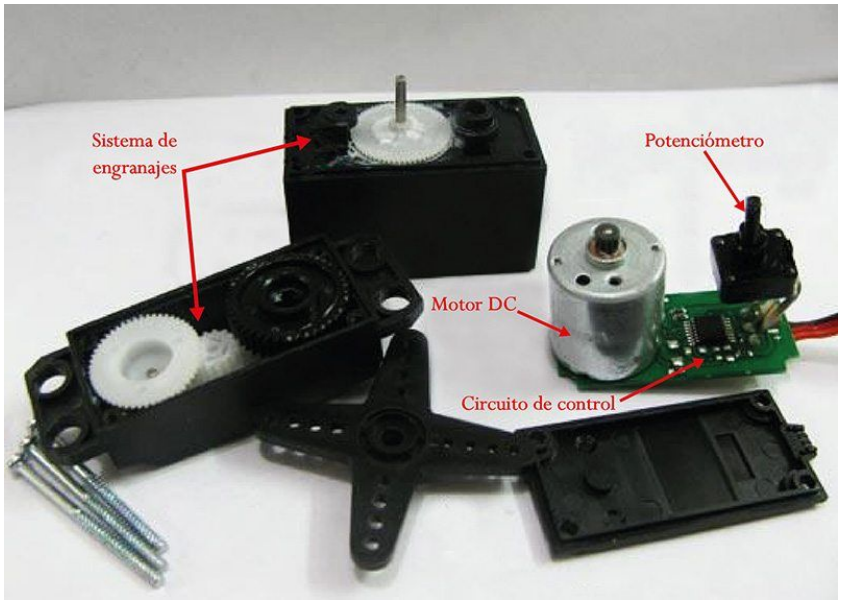
1º- El brazo deberá ser capaz de coger un objeto situado en unas coordenadas,que se facilitarán, y el programa calculará mediante cinemática inversa los grados que deben tener cada servomotor para que el brazo llegue a ese punto.
2º- Una vez conseguido el primer punto, se diseñará una aplicación para poder controlar el brazo mediante el smartphone.
3º- Para llevar a cabo todo el proyecto se necesitará el uso de un microcontrolador(Arduino) y varios servomotores, así como un módulo bluetooth y baterías.

HARDWARE

Plataforma de giro - Servomotor
fuente :http://panamahitek.com/que-es-y-como-funciona-un-servomotor/
Un servomotor (o servo) es un tipo especial de motor con características especiales de control de posición. Al hablar de un servomotor se hace referencia a un sistema compuesto por componentes electromecánicos y electrónicos.

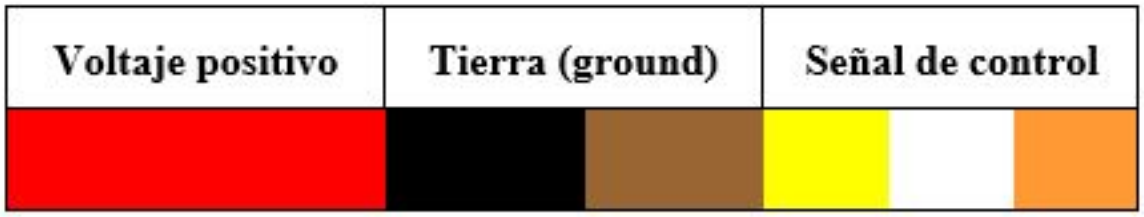


El motor en el interior de un servomotor es un motor DC común y corriente. El eje del motor se acopla a una caja de engranajes similar a una transmisión. Esto se hace para potenciar el torque del motor y permitir mantener una posición fija cuando se requiera. De forma similar a un automóvil, mayor velocidad, menor torque. El circuito electrónico es el encargado de manejar el movimiento y la posición del motor.

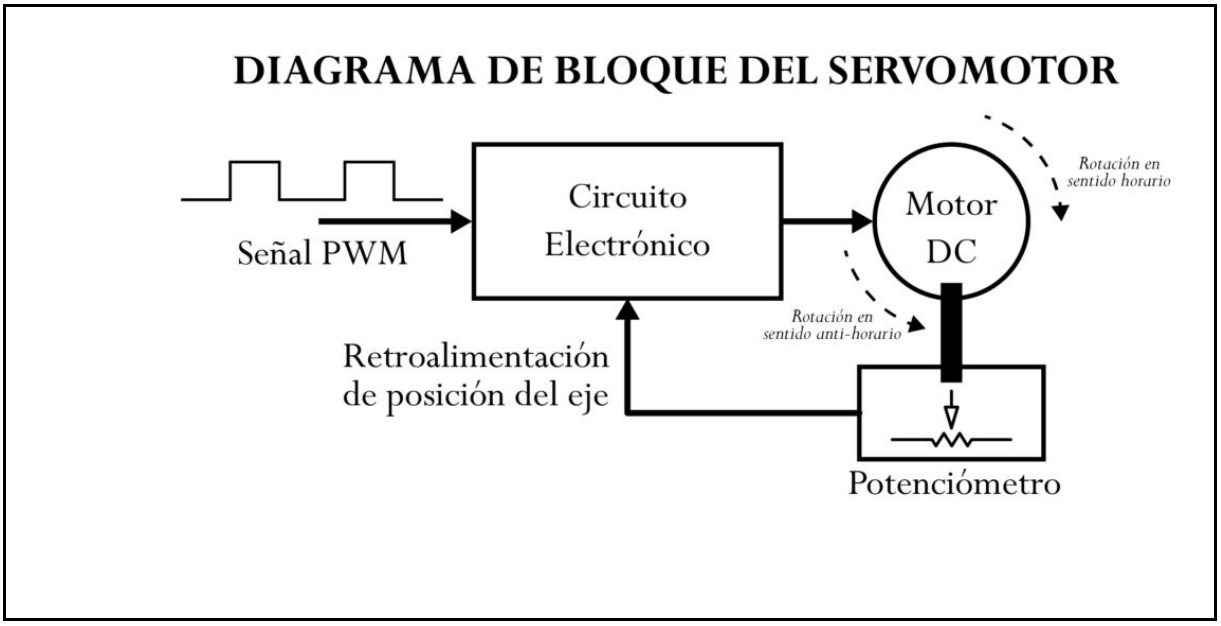


La presencia del sistema de engranajes como el que se muestra en la figura hace que cuando movemos el eje motor se sienta una inercia muy superior a la de un motor común y corriente. Observando las imágenes que hemos presentado nos podemos dar cuenta que un servo no es un motor como tal, sino un conjunto de partes (incluyendo un motor) que forman un sistema.

Los servomotores poseen tres cables, a diferencia de los motores comunes que sólo tienen dos. Estos tres cables casi siempre tienen los mismos colores, por lo que son fácilmente reconocibles.

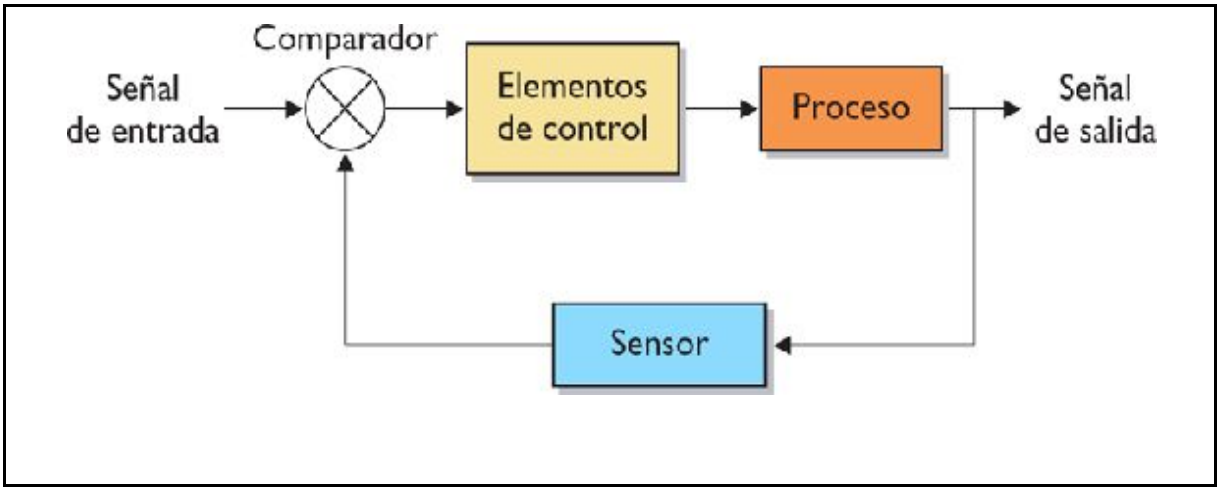


La necesidad de una señal de control para el funcionamiento de este tipo de motores hace que sea imposible utilizarlos sin un circuito de control adecuado. Esto se debe a que para que el circuito de control interno funcione, es necesaria una señal de control modulada. Para esto se utiliza modulación por ancho de pulsos, es decir, PWM.



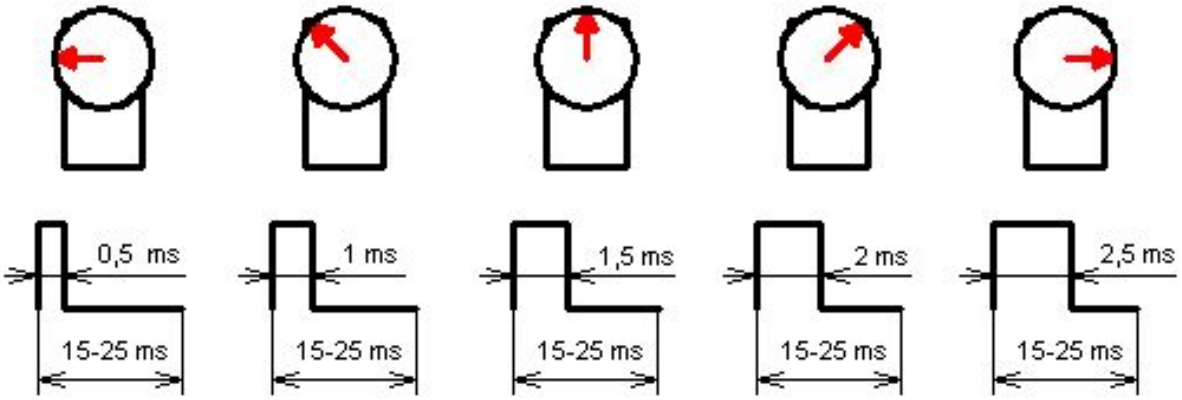
El diagrama de bloque del servomotor representa de forma visual el servomotor como un sistema. El circuito electrónico es el encargado de recibir la señal PWM y traducirla en movimiento del Motor DC. El eje del motor DC está acoplado a un potenciómetro, el cual permite formar un divisor de voltaje. El voltaje en la salida del divisor varía en función de la posición del eje del motor DC.

Cuando el eje del motor modifica la posición del potenciómetro, el voltaje en la terminal central varía. El potenciómetro permite que el circuito de control electrónico pueda retroalimentarse con la posición del motor en un momento dado. Esto, en Teoría de Control se conoce como un sistema de lazo cerrado.



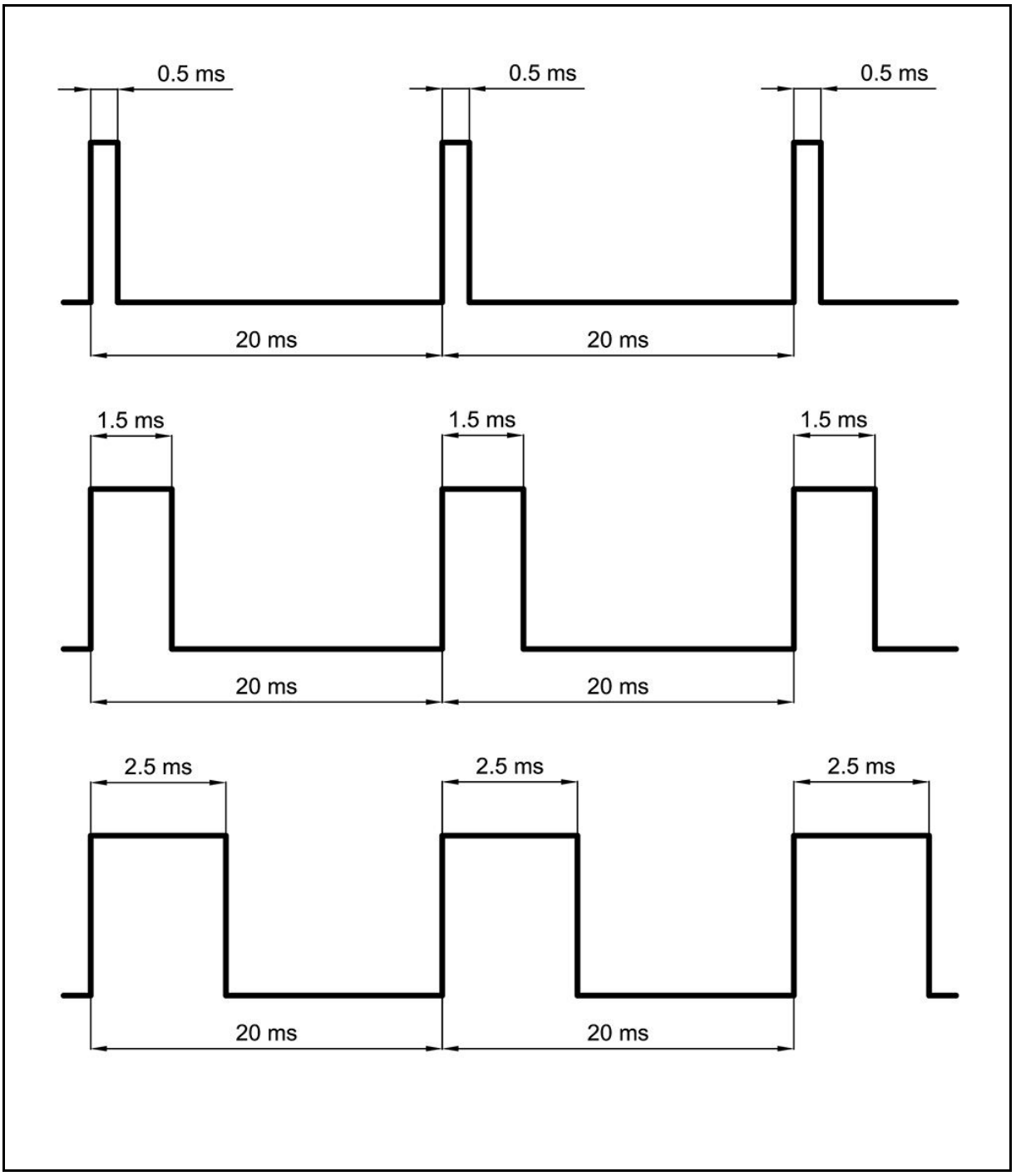
Los servomotores de rotación continua desacoplan el potenciómetro del eje del motor. Esto impide que el circuito de control pueda leer la posición del eje, por lo cual provoca un movimiento continuo al no ser capaz de cumplir la condición para que el servo se detenga. Los servomotores de rotación continua normalmente pueden girar en un sentido o en otro y detenerse. Podemos modificar la velocidad de giro, pero no podremos lograr, por ejemplo, que el servo se mueva una determinada cantidad de grados y luego se detenga.

Las señales de PWM requeridas para que el circuito de control electrónico son similares para la mayoría de los modelos de servo. Esta señal tiene la forma de una onda cuadrada. Dependiendo del ancho del pulso, el motor adoptará una posición fija.



Ancho de pulsos para lograr diferentes posiciones en un servomotor (180º, 135º, 90º, 45º y 0º)

Las señales que vemos en la imagen son las que permiten que el eje del motor adquiera determinada posición. Estas señales deben repetirse en el tiempo para que el motor mantenga una posición fija.


0.5 ms
0.5 ms
0.5 ms
20 ms
20 ms
1.5 ms
1.5 ms
1.5 ms
20 ms
20 ms
2.5 ms
2.5 ms
2.5 ms
20 ms
20 ms

Ejemplos de trenes de pulsos para las posiciones 180º, 90º y 0º en el eje de un servomotor

La duración del ciclo de trabajo varía entre 15 y 25 milisegundos. Las ondas mostradas en la imagen anterior representan ejemplos de trenes de pulsos con los que se puede mover un servomotor, utilizando un ciclo de trabajo de 20 milisegundos.

Este tren de pulsos puede ser generado por un circuito oscilador o por un microcontrolador. Es decir, con Arduino podemos controlar fácilmente un servomotor. De hecho en Arduino existen las librerías para el control de servos de forma nativa. No es necesario descargarlas.

```
#include <Servo.h>

Servo myservo;  // crea el objeto servo

int pos = 0;    // posicion del servo

void setup() {
   myservo.attach(9);  // vincula el servo al pin digital 9
}

void loop() {
   //varia la posicion de 0 a 180, con esperas de 15ms
   for (pos = 0; pos <= 180; pos += 1)
   {
      myservo.write(pos);
      delay(15);
   }

   //varia la posicion de 0 a 180, con esperas de 15ms
   for (pos = 180; pos >= 0; pos -= 1)
   {
      myservo.write(pos);
      delay(15);
   }
}
```

ARDUINO

https://www.arduino.cc/

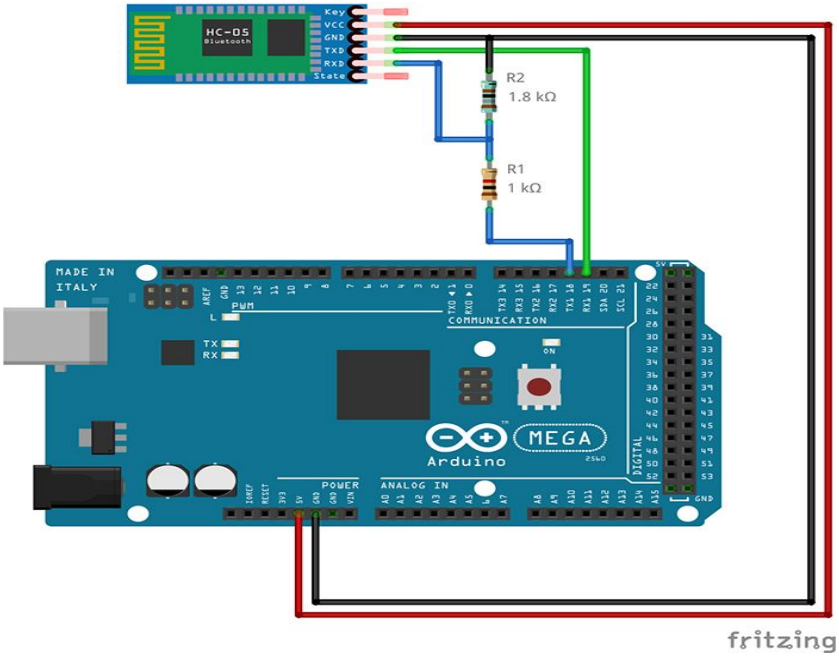
El microcontrolador que hemos elegido para mover los servomotores es Arduino.

Arduino (anteriormente conocido como **Genuino** a nivel internacional hasta octubre 2016), es una compañía open source y open hardware, así como un proyecto y comunidad internacional que diseña y manufactura placas de desarrollo de hardware para construir dispositivos digitales y dispositivos interactivos que puedan controlar objetos del mundo real. Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Los productos que vende la compañía son distribuidos como Hardware y Software Libre, bajo la Licencia Pública General Reducida de GNU (LGPL) o la Licencia Pública General de GNU (GPL),permitiendo la manufactura de las placas Arduino y distribución del software por cualquier individuo. Las placas Arduino están disponibles comercialmente en forma de placas ensambladas o también en forma de kits hazlo tú mismo (DIY, por sus siglas en inglés de "Do It Yourself").

Debido a que el arduino UNO del que disponíamos no es lo suficientemente potente para llevar a cabo tantas operaciones matemáticas y almacenar tanto código hemos decidido usar MATlab. Este programa se encarga de realizar todos los cálculos necesarios y devolver al arduino las posiciones que deben adoptar los servomotores

Bluetooth

https://aprendiendoarduino.wordpress.com/2016/11/13/bluetooth-en-arduino/



En nuestro trabajo hemos utilizado el modelo de Bluetooth HC-06, uno de los más versátiles para este tipo de proyectos.
Este módulo es apto para conectar como esclavo a un adaptador Bluetooth al PC, o un teléfono Bluetooth. Pueden ajustarse los parámetros del módulo mediante comandos de control emitidos a través de comandos AT. Dispone de dos conexiones de salida para indicar el estado: no conectado, conectado; así mismo se puede conectar a diodos LED o a un pin de entrada de un microcontrolador y analizar su estado. Es ideal como alternativa inalámbrica a la transmisión en serie, todos los parámetros se configuran con Comandos AT. Con la base de conexión es aún más fácil de utilizar, no es necesario adaptar los niveles lógicos de la alimentación a 3.3V (se puede alimentar hasta a 6V) y además de esto, puede adaptarse en una protoboard.

Código ejemplo Bluetooth control un servo

```
#include <Servo.h>0
Servo servo;
char a;
String readString;
void setup()
{
  servo.attach(3);
  Serial.begin(9600);
  servo.write(90);
  delay(10);
}
void loop()
{
  if(Serial.available())
  {
        a=Serial.read();
```

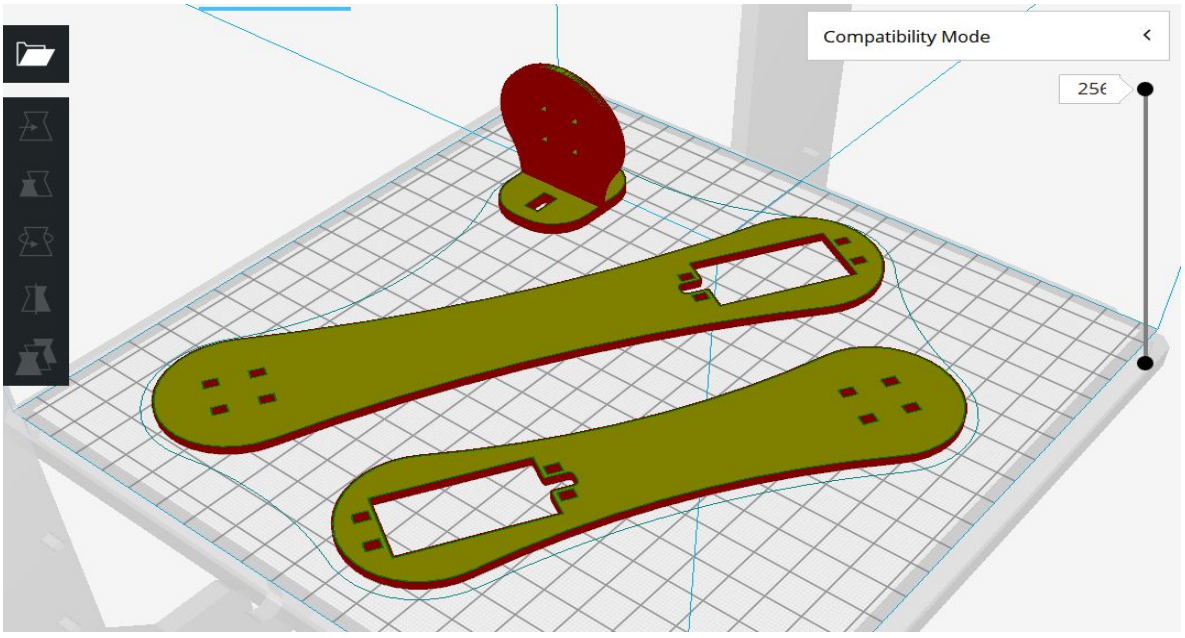
```
        if(a=='B')
        {
        motor2();
        }
  }
}
void motor2()
{
  delay(10);
  while(Serial.available())
  {
        char b = Serial.read();
        readString += b;
  }
  if(readString.length()>0)
  {
        Serial.println(readString.toInt());
        servo.write(readString.toInt());
        readString = "";
  }
}
```

DISEÑO

El brazo se compone de varias piezas fabricadas usando una impresora 3D, en concreto con el plástico PLA (ácido poliláctico), un plástico fácilmente extruible y además biodegradable. Todas las piezas han sido diseñadas en tres programas simultáneamente, AutoCAD, Inventor y SketchUp, debido a que cada programa se centra en distintos ámbitos del diseño. El diseño final se exporta como ".stl" y se abre con otro programa, Ultimaker Cura. Este se ocupa de ajustar los parámetros

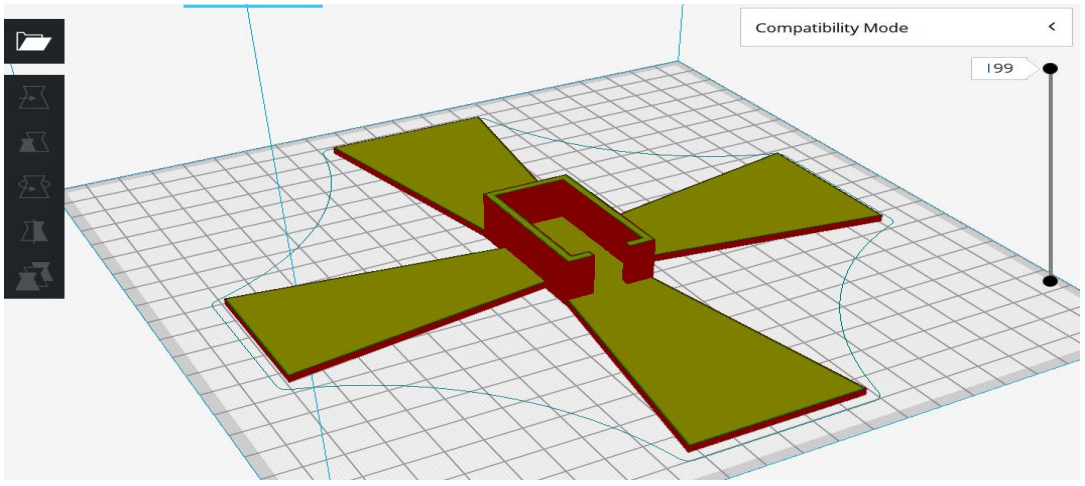
de impresión adecuados y exporta otro formato de archivo, “.gcode”, que es el que las impresoras manejan en su funcionamiento

El cuerpo principal lo componen tres piezas finas y alargadas, que funcionan como brazo, antebrazo y muñeca. Están diseñadas para acomodar el cuerpo de los servomotores y fijarlos con bridas en un extremo. En el extremo opuesto hay unos huecos que permiten colocar la cruceta del siguiente servo y así unir las tres piezas.



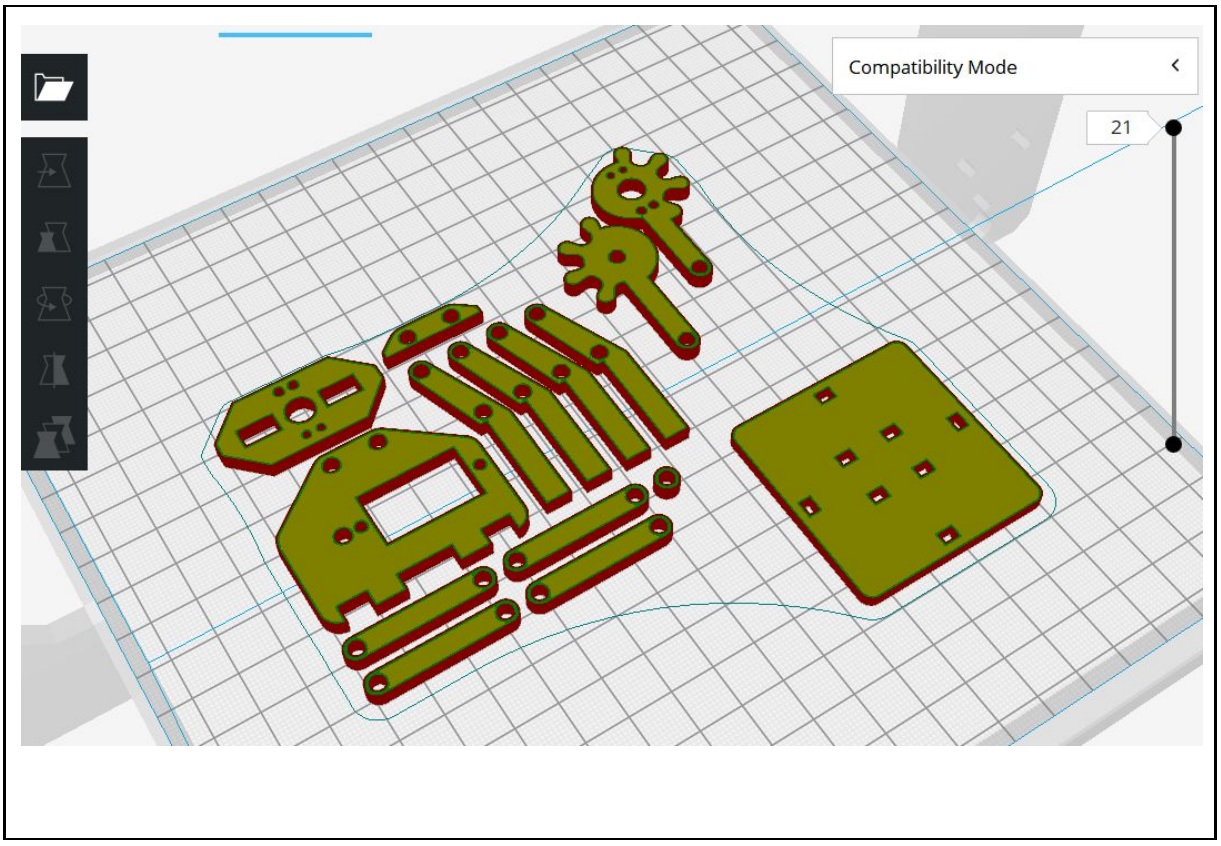
Conjunto Brazo, Antebrazo, Muñeca

La base tiene dos partes. La primera, más grande, y la que da soporte a todo el conjunto que aloja el servomotor que se encarga de la rotación del brazo sobre sí mismo. La segunda, más pequeña, se encarga de unir el servo anterior con el primero en posición horizontal.



Base soporte

Por último, la pinza, la pieza más compleja de todas. Está diseñada para mantener sus dedos en paralelo en cualquier posición, ya sea con la pinza abierta, cerrada o mientras coge un objeto. Tiene varios ejes, unidos con tornillos, y un servomotor que mueve ambos dedos de la pinza, gracias a unos engranajes.



Pinza y base pequeña

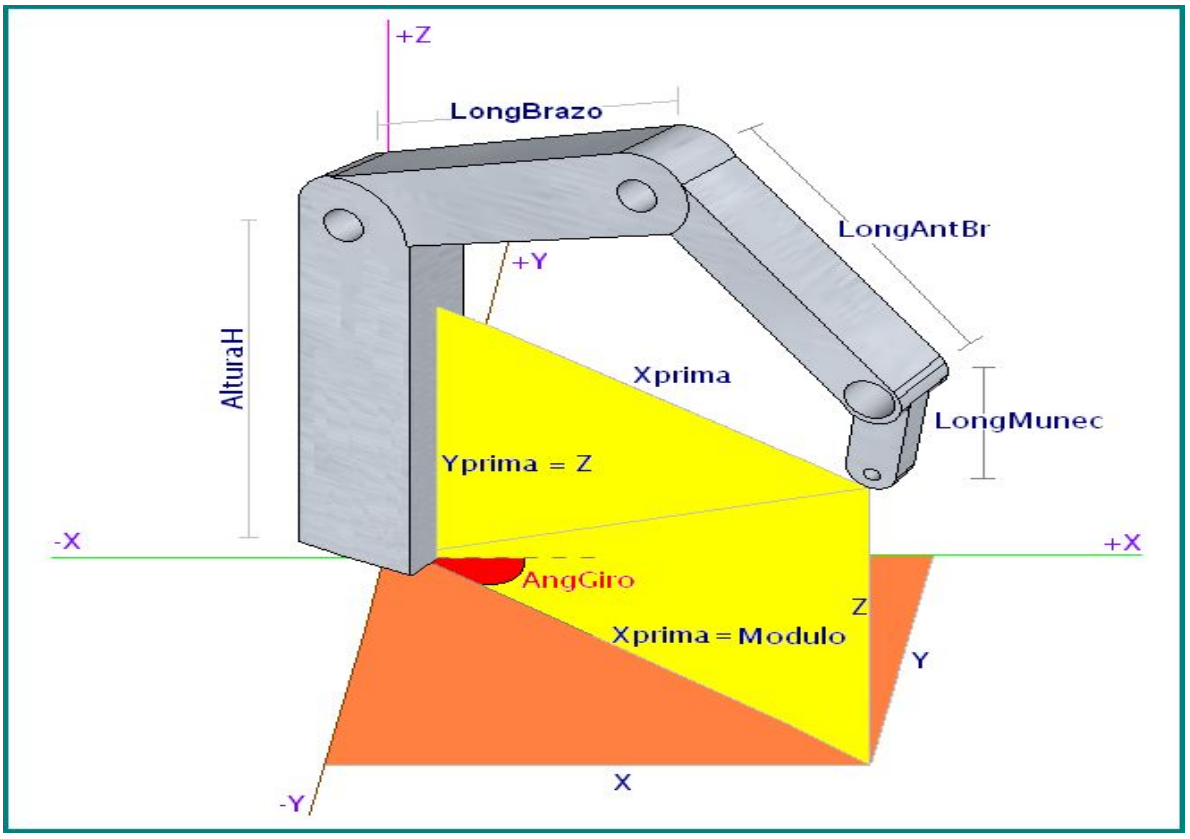
FUNDAMENTOS TEÓRICOS

Cinemática inversa.

Para llevar a cabo el proyecto se ha usado la cinemática inversa,en Robótica, la **Cinemática inversa (IK)** es la técnica que permite determinar el movimiento de una cadena de articulaciones para lograr que un actuador final se ubique en una posición concreta. El cálculo de la cinemática inversa es un problema complejo que consiste en la resolución de una serie de ecuaciones cuya solución normalmente no es única.

Existen muchos métodos para resolver el problema de la cinemática inversa,como metodos iterativos,metodos geometricos,métodos que usan la matriz de transformación homogénea,método diferencial matriz jacobiana… Nosotros por, sencillez hemos elegido el método geométrico:

Método geométrico



El método geométrico es adecuado para robots de pocos grados de libertad o para el caso de que se consideren solo los primeros grados de libertad para posicionar el extremo.

El procedimiento se basa en encontrar un número suficiente de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.

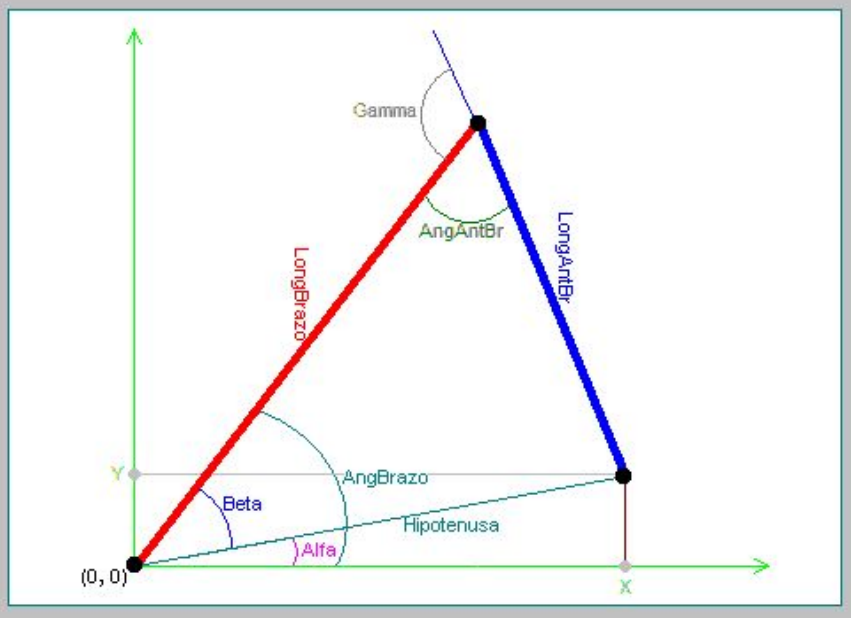
Para calcular la Cinemática Inversa de dos grados de libertad:

*** Conocemos:**

- **Longitud del brazo.**
- **Longitud del antebrazo.**
- **Eje X.**
- **Eje Y.**

*** Tenemos que averiguar los ángulos:**

- **Ángulo del brazo.**
- **Ángulo del antebrazo.**



Solución:

Hipotenusa = sqr((X^2)+(Y^2))

Alfa = Atan2(Y, X)

Beta = Acos((LongBrazo^2 - LongAntBr^2 + Hipotenusa^2) / (2 * LongBrazo * Hipotenusa))

AngBrazo = Alfa + Beta

Gamma = Acos ((LongBrazo^2 + LongAntBr^2 - Hipotenusa^2) / (2 * LongBrazo * LongAntBr))

AngAntBr = Gamma - 180

Función -CINEMATICAINV4

Hemos desarrollado una función que se encarga de calcular los ángulos que deben adoptar los servomotores a partir de las longitudes de las distintas partes del brazo así como el punto que debe alcanzar. Esta es la función que se usa durante el programa :

```
%% Funcion definitiva
function[angulo1,angulo2,angulo3,angulo4,Hipotenusa,ejeIMG,alfa,beta]=cin
ematicaINV4(x,y,z,Altura,LongBrazo,LongAntebrazo,LongMuneca)
%%funcion que a partir de un punto del espacio nos devuelve los angulos
que
%%deben tener los servos del brazo para llegar a ese punto
%% Calculo del angulo de la Base
ro= atan2(x,y)*180/pi;
ejeIMG=sqrt(y.^2+x.^2);
angulo1=ro;
%% Calculo del angulo del Brazo
Hipotenusa=sqrt((z-Altura).^2+(ejeIMG-LongMuneca).^2);

alfa=atan2((z-Altura),(ejeIMG-LongMuneca))*180/pi;
beta=acos((LongBrazo.^2+Hipotenusa.^2-LongAntebrazo.^2)/(2*LongBrazo*Hipo
tenusa))*180/pi;
angulo2=alfa+beta;
```

```
%% Calculo del angulo del Antebrazo

gamma=acos(((LongBrazo.^2)+(LongAntebrazo.^2)-(Hipotenusa.^2))/(2*LongBra
zo*LongAntebrazo))*180/pi;

angulo3=gamma;

%% Calculo del angulo de la Muñeca

theta=-90+gamma+beta+alfa;

angulo4=180-theta;
```

SOFTWARE

Matlab

https://es.mathworks.com/hardware-support/arduino-matlab.html

MATLAB (abreviatura de ***MAT****rix* ***LAB****oratory*, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario -

GUI). Además, se pueden ampliar las capacidades de MATLAB con las *cajas de herramientas* (*toolboxes*); y las de Simulink con los *paquetes de bloques* (*block sets*).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

La razón por la que hemos decidido usar matlab se debe a que el arduino UNO no es muy potente computacionalmente además de no tener espacio para almacenar mucho código.Por eso hemos optado por usar Matlab como “cerebro”, haciendo que se encargue de los cálculos y arduino se preocupe solo de controlar los servomotores.

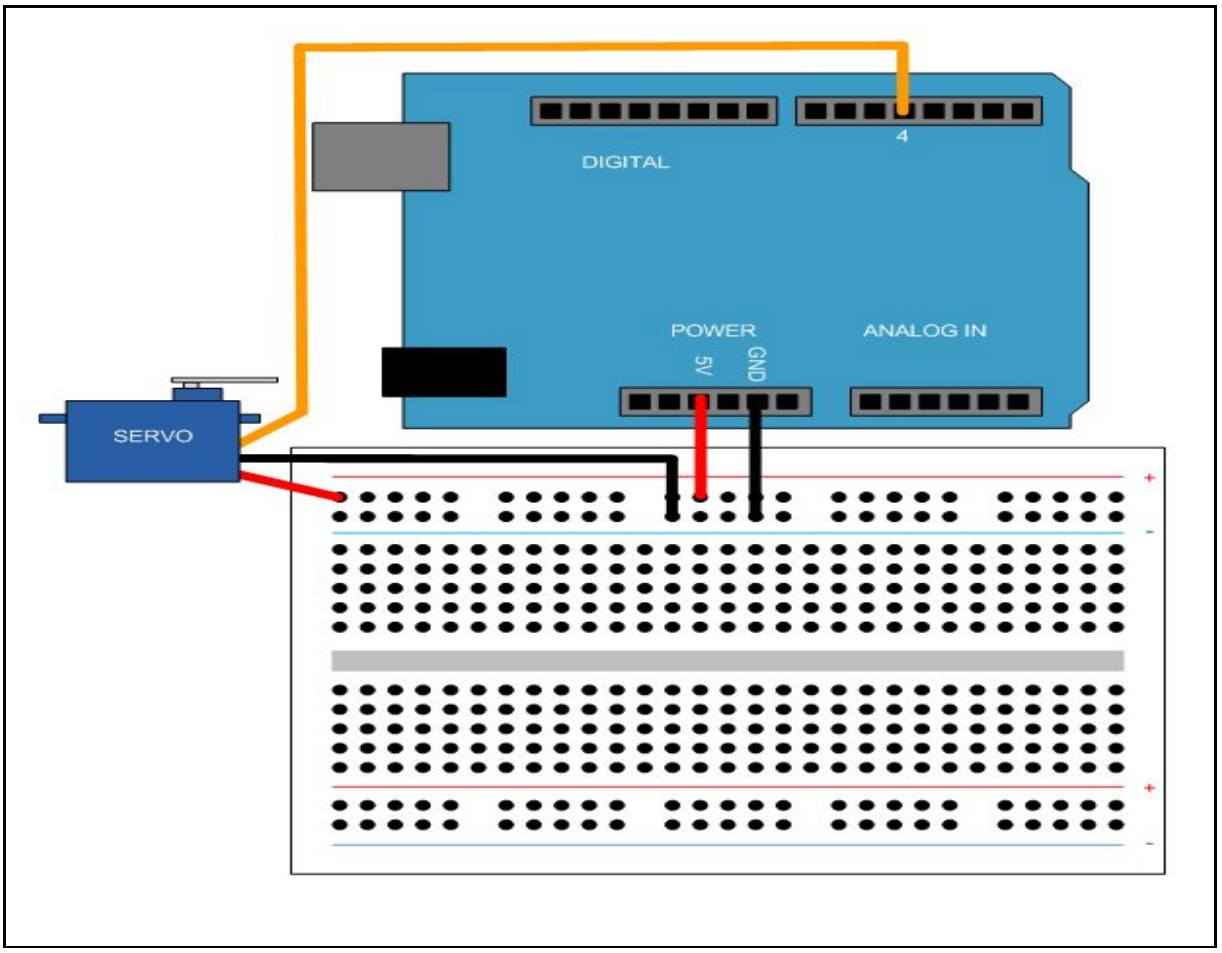
Para conseguir la comunicación Matlab-->Arduino se ha tenido que instalar unos serie de Drivers del siguiente link:

https://es.mathworks.com/hardware-support/arduino-matlab.html?requestedDomain=

Este ejemplo muestra cómo usar Matlab para para controlar un servo

Conectar un servo motor FS5106B a la placa Arduino.

1. Conectar el cable de alimentación al pin de 5V..
2. Conectar el cable de tierra al pin GND.
3. Conectar el cable de señal a un pin digital(en este caso el 4).


DIGITAL
4
POWER
ANALOG IN
5V
GND
SERVO

Definir el objeto servo y calibrar el motor

Crear un objeto arduino e incluir la librería Servo.

```
a = arduino();
```

También se puede definir explícitamente especificando en en el par Nombre-Valor de las librerías al crear el objeto arduino.

```
clear a;
a = arduino('COM4', 'Uno', 'Libraries', 'Servo');
```

```
Subir el código a la placa Arduino.
```

Crear el objeto servo.

```
s = servo(a, 'D4')
```

```
s =
  Servo with properties:

                Pins: D4
    MinPulseDuration: 5.44e-04 (s)
    MaxPulseDuration: 2.40e-03 (s)
```

```
Comprueba el rango de valores en la ficha de especificaciones del
servo para rotar el servo a las posiciones deseadas. Este ejemplo
usa 700^10^-6 y 2300*10^-6 para que el motor se mueva de 0 a 180
grados.
```

```
clear s;
s = servo(a, 'D4', 'MinPulseDuration', 700*10^-6,
'MaxPulseDuration', 2300*10^-6)
```

```
s =
  Servo with properties:

                Pins: D4
    MinPulseDuration: 7.00e-04 (s)
    MaxPulseDuration: 2.30e-03 (s)
```

Write and read Servo position

Cambia la posición del eje del servo desde 0 (mínimo) a 1(máximo) con incremento de 0.2,(36 grados) Muestra la posición cada vez que cambia.

```
for angle = 0:0.2:1
    writePosition(s, angle);
    current_pos = readPosition(s);
    current_pos = current_pos*180;
    fprintf('Current motor position is %d degrees\n', current_pos);
    pause(2);
end
```

```
Current motor position is 0 degrees
Current motor position is 36 degrees
Current motor position is 72 degrees
Current motor position is 108 degrees
Current motor position is 144 degrees
Current motor position is 180 degrees
```

Clean up

Una vez que no se necesita la conexión, se limpia el objeto asociado.

```
clear s a
```

PROGRAMA FINAL

Aquí tenemos una parte del código usado en el programa

```
%% Programa brazo con cinemática inversa

clc,clear;

%%Limpiamos lo anterior y establecemos la conexión con arduino
%%declaramos las variables servos

a=arduino();

Base=servo(a,'D3');
Brazo=servo(a,'D4');
Antebrazo=servo(a,'D5');
Muneca=servo(a,'D6');
Pinza=servo(a,'D7');

%%longitudes de las distintas partes del brazo

LongBrazo=16;
LongAntebrazo=12;
LongMuneca=12;
Altura=9;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Primera cajita

%% Punto

x=21;
y=16;
z=4;

a1=readPosition(Base);
a2=readPosition(Brazo);
a3=readPosition(Antebrazo);
a4=readPosition(Muneca);
a5=readPosition(Pinza);

[angulo1,angulo2,angulo3,angulo4,Hipotenusa,ejeIMG,alfa,beta]=cinematicaI
NV4(x,y,z,Altura,LongBrazo,LongAntebrazo,LongMuneca);

Hipotenusa;
ejeIMG;
alfa;
beta;

%% Pinza Abrir

      writePosition(Pinza,0);
```

```
%% servo Base

if(a1>((angulo1/180)))
      for i=a1:-0.005:((angulo1/180))
      writePosition(Base,i);
      end
else
      for i=a1:0.005:((angulo1/180))
      writePosition(Base,i);
      end

end

%% SERVO Muñeca

if(a4>(angulo4/180))
      for i=a4:-0.005:(angulo4/180)
      writePosition(Muneca,i);
      end
else
      for i=a4:0.005:(angulo4/180)
      writePosition(Muneca,i);
      end
end

%% servo Brazo

if(a2<(angulo2/180))
      for i=a2:0.005:(angulo2/180)
      writePosition(Brazo,i);
      end
else
      for i=a2:-0.005:(angulo2/180)
      writePosition(Brazo,i);
      end

end
%% SERVO Antebrazo

if(a3>(angulo3/180))
      for i=a3:-0.005:((angulo3/180))
      writePosition(Antebrazo,(1-i));
      end
else
      for i=a3:0.005:((angulo3/180))
      writePosition(Antebrazo,(1-i));
      end
end

pause(1);
```

```
%% Pinza Cerrar

writePosition(Pinza,0.5);

%%%%%%%%%%Volver

angulo1=0.35;
angulo2=0.62 ;
angulo3=0 ;
angulo4=0.300;

%% SERVO Antebrazo

if(a3>angulo3)
      for i=a3:-0.005:angulo3
      writePosition(Antebrazo,(1-i));
      end
else
      for i=a3:0.005:angulo
      writePosition(Antebrazo,(1-i));
      end
end
%% servo Brazo
if(a2<angulo2)
      for i=a2:0.005:angulo2
      writePosition(Brazo,i);
      end
else
      for i=a2:-0.005:angulo2
      writePosition(Brazo,i);
      end

end

%% servo Base

if(a1>angulo1)
      for i=a1:-0.005:angulo1
      writePosition(Base,i);
      end
else
      for i=a1:0.005:angulo1
      writePosition(Base,i);
      end

end

%% SERVO Muñeca
```

```
if(a4>angulo4)
      for i=a4:-0.005:angulo4
      writePosition(Muneca,i);
      end
else
      for i=a4:0.005:angulo4
      writePosition(Muneca,i);
      end
end
```