SNEAKY PEEKY

Sergio Florido Becerra

Andrés Domínguez Aguilar

MEI

ÍNDICE

1. INTRODUCCIÓN 3
2. ELECTRÓNICA 4
3. PROGRAMACIÓN 8
4. MONTAJE DE LA PARTE MECÁNICA 13
5. INCIDENCIAS 20
   5.1. Diseños 3D 21
   5.2. Programación en PIC: 30
6. DOCUMENTACIÓN 34

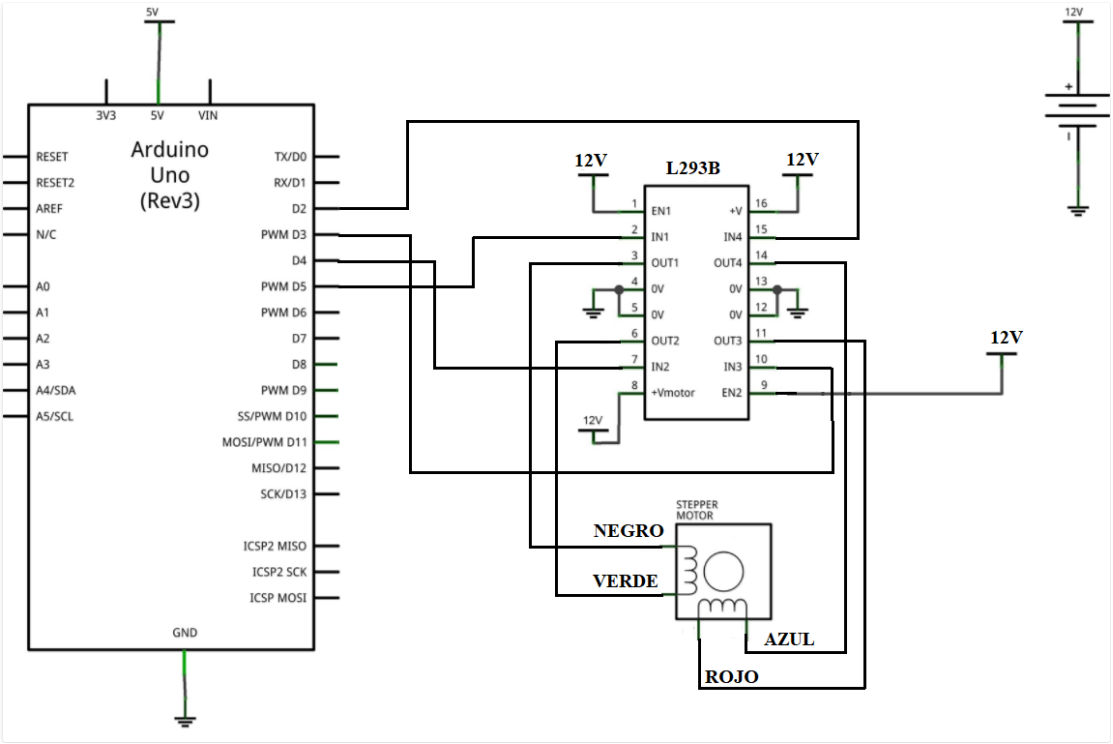
1. INTRODUCCIÓN

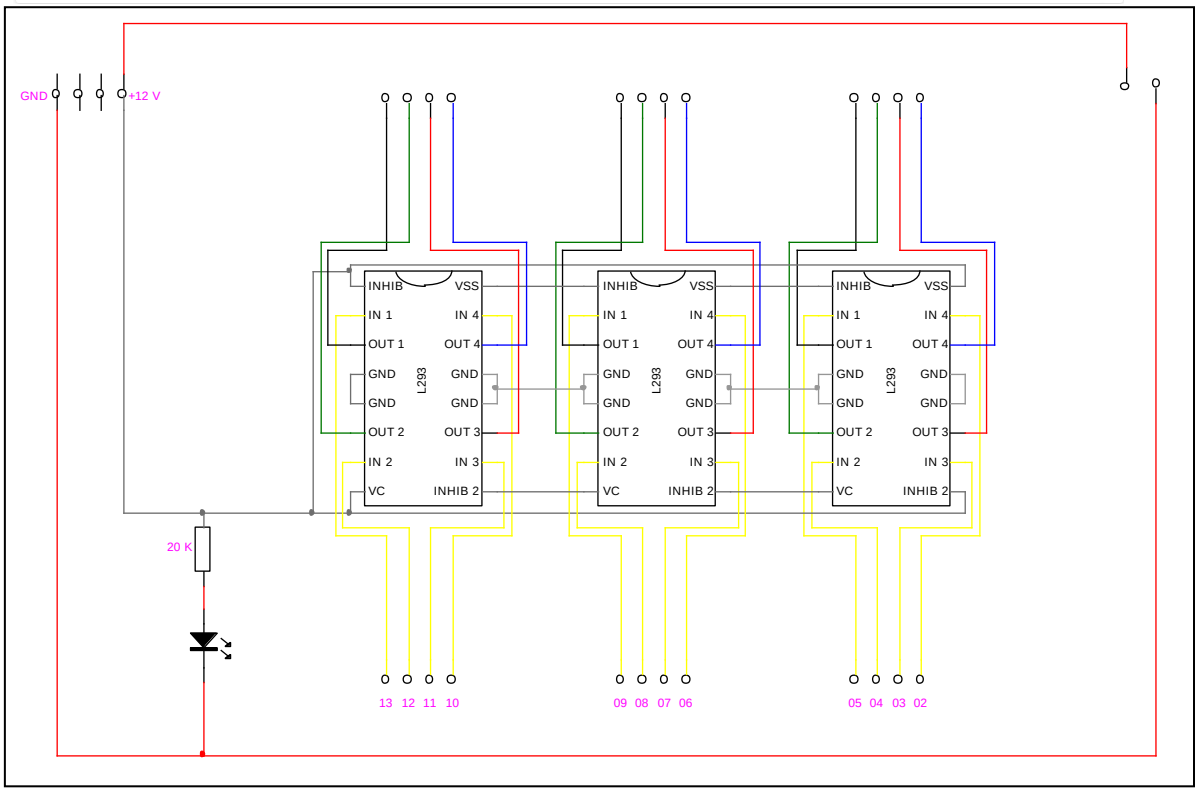
Sneaky-Peeky es un proyecto del ciclo superior de Mantenimiento Electrónico basado en las garras de Lobezno. El principal objetivo de este proyecto es controlar con la mano, tres cuchillas que van ancladas al brazo de la misma mano, de tal forma que las cuchillas salgan o entren en el brazo dependiendo de la flexión de los dedos de la mano, es decir, cuando los dedos de la mano estén totalmente flexionados (el puño cerrado), las cuchillas estarán completamente fuera, y si la mano está abierta, las cuchillas estarán dentro del antebrazo. En caso de que los dedos estén flexionados en un término intermedio, las cuchillas también lo estarán de forma proporcional a estos.

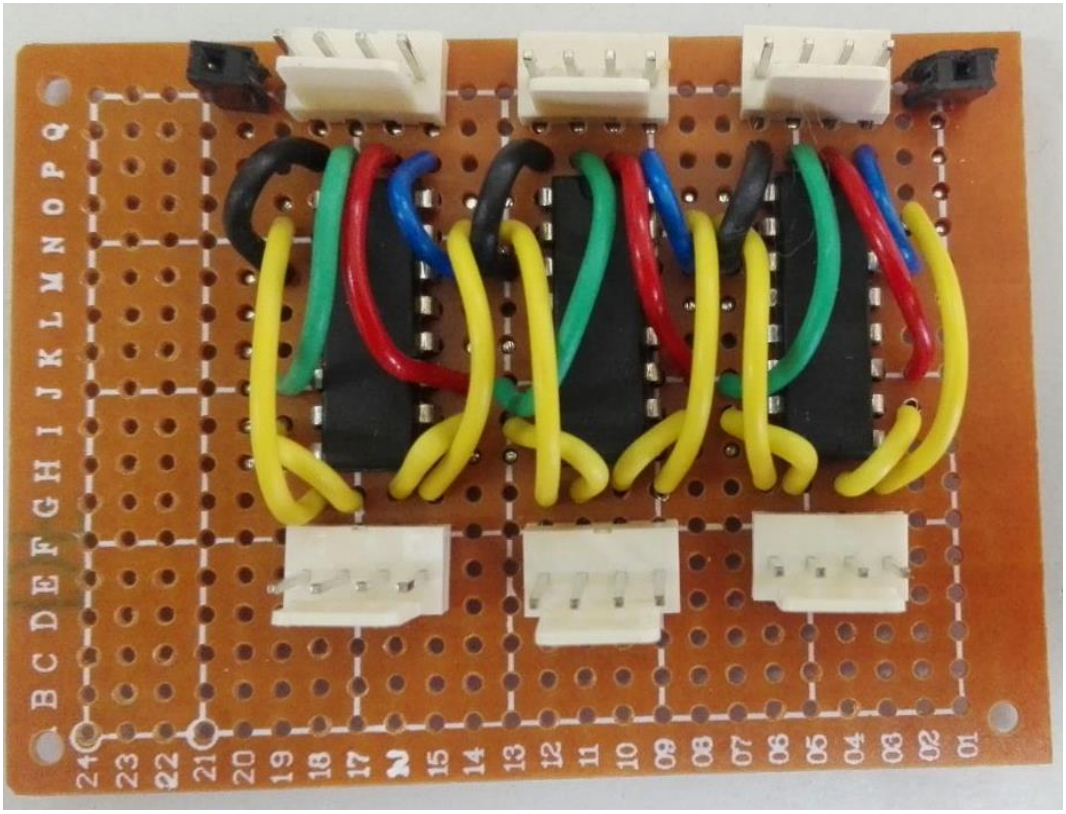
Esto se realizará con unos sensores o resistencias flexibles que irán sujetos a los dedos, tres motores paso a paso, tres cuchillas impresas con una impresora 3D, una correa con un rodamiento y una guía de cajón para cada motor. El funcionamiento general del mecanismo es que cuando la resistencia flexible cambie de valor, el motor paso a paso con la correa girará en una dirección, por lo tanto, la correa avanzará en un sentido o en el contrario, moviendo las cuchillas enganchadas a esta.

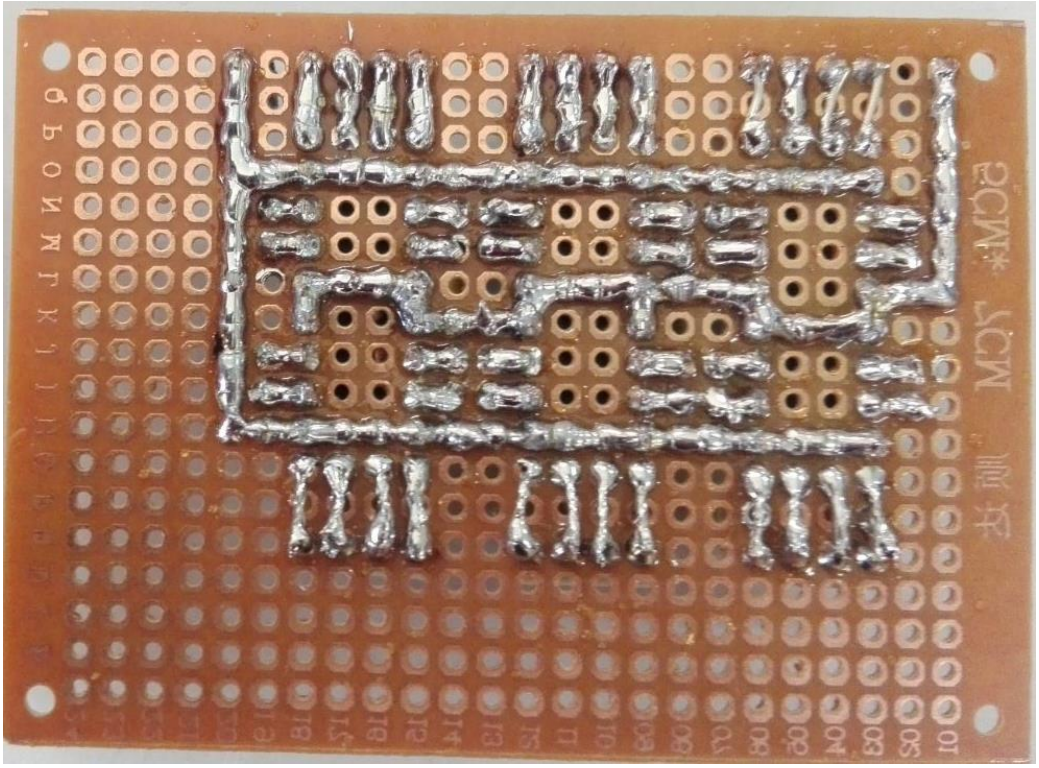
2. ELECTRÓNICA

Para la electrónica hemos usado 3 drivers L293B para el control de los motores paso a paso (bipolares) con una placa de Arduino UNO como control de estos. Va conexionado de la siguiente forma:

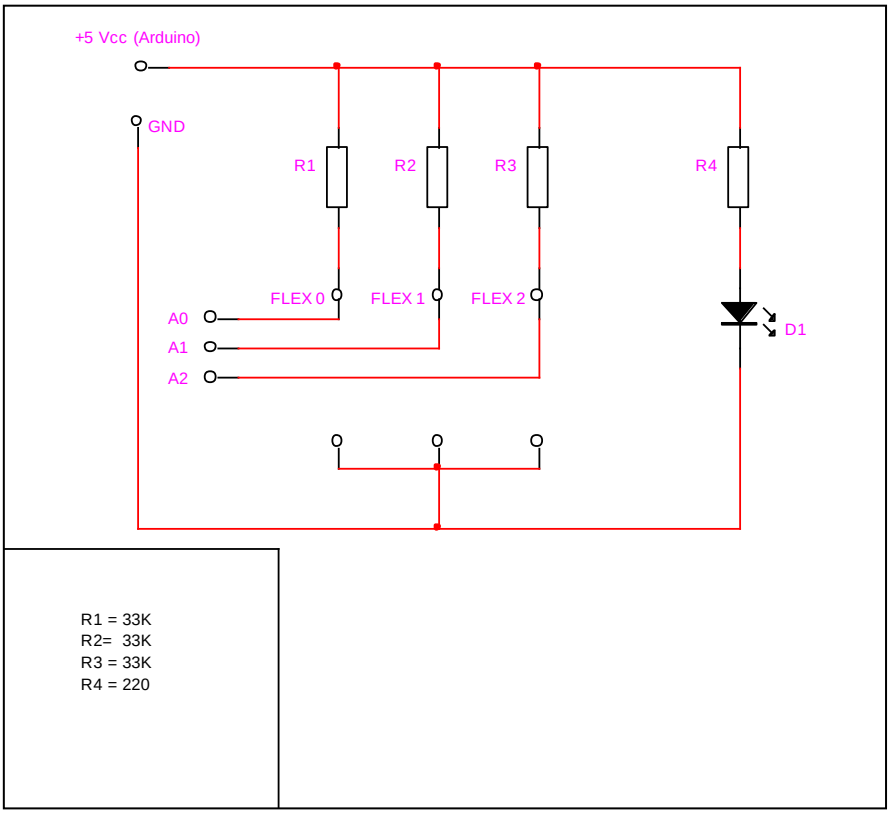


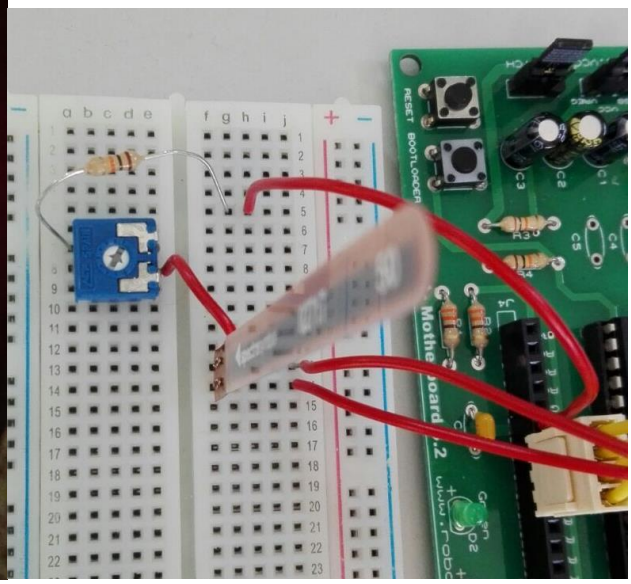


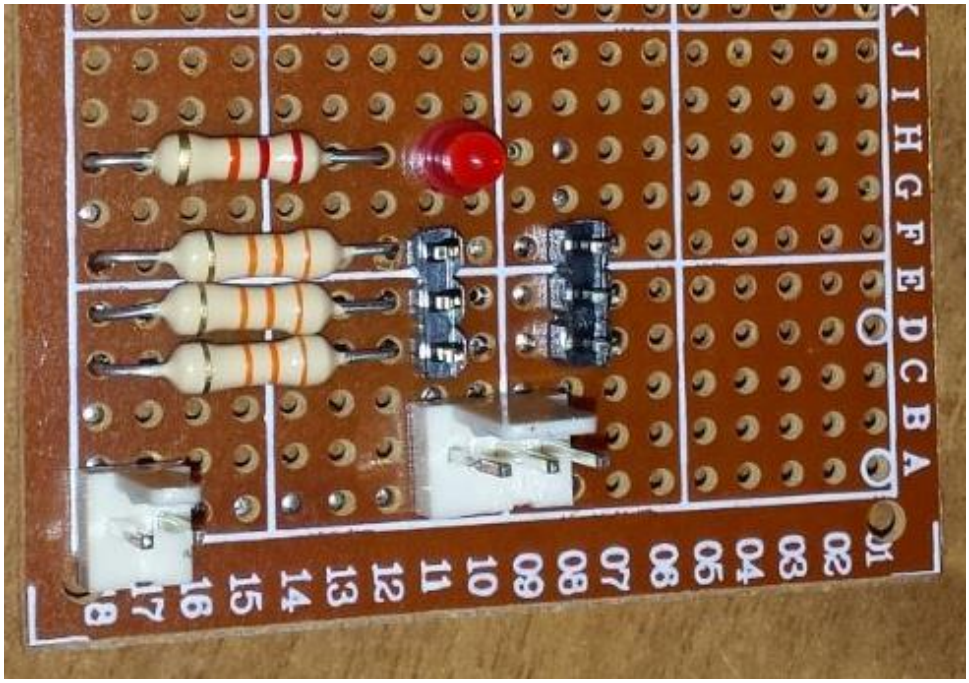


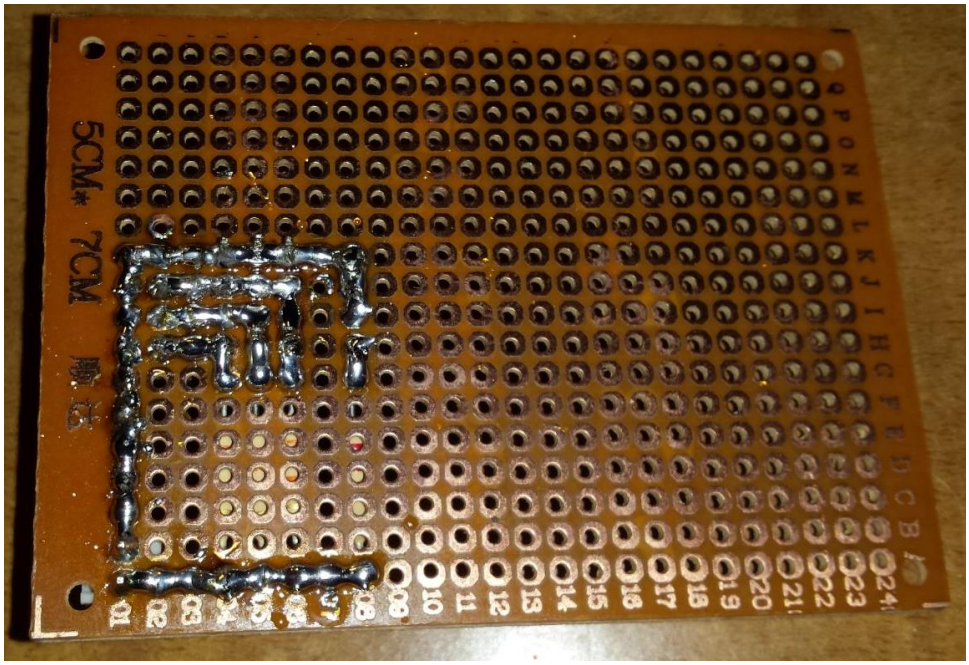


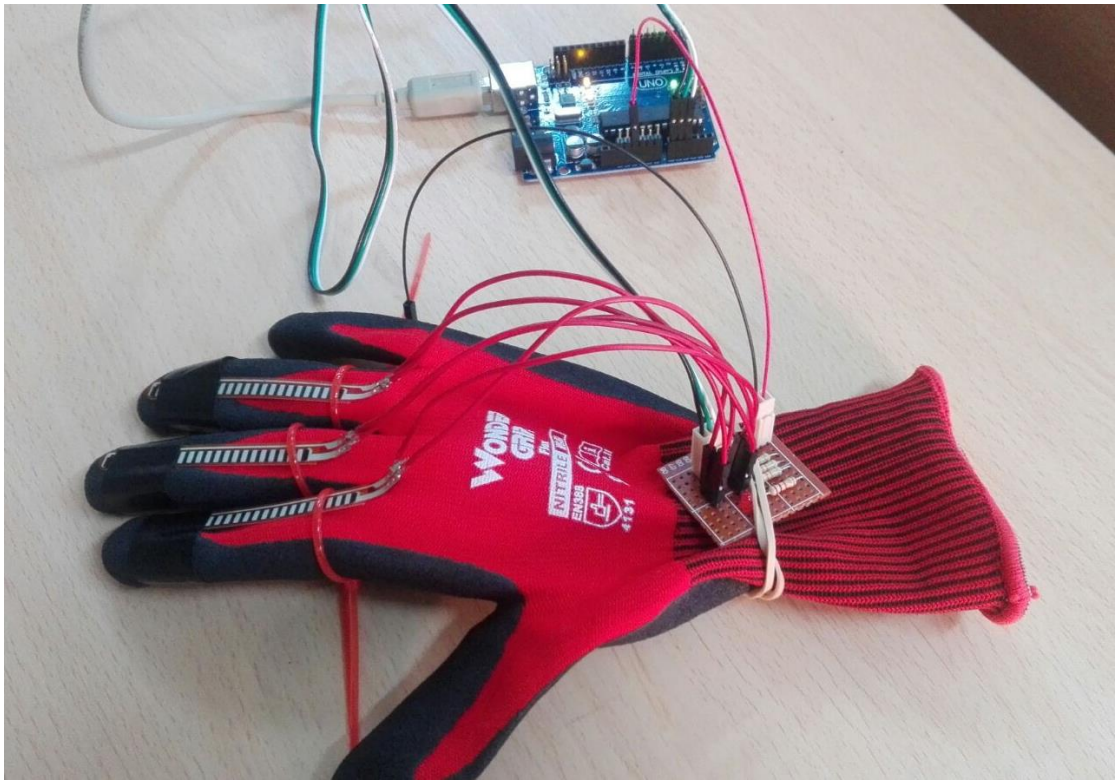
Por otra parte las resistencias flexibles van conexionadas de la siguiente forma en una pequeña placa que hemos diseñado junto con unas resistencias de 33k para hacer un divisor de tensión:






K J I H G F E D C B A
J1
02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18


5CM* 7CM



3. PROGRAMACIÓN

```
#include <Stepper.h> // Importamos la librería para controlar motores paso a paso


#define STEPS 100

// Ponemos el número de pasos que necesita para dar una vuelta. 100 en nuestro caso


// Ponemos nombre al motor, el número de pasos y los pins de control

Stepper cabesa0(STEPS, 2, 3, 4, 5);

// Stepper nombre motor (número de pasos por vuelta, pins de control)

Stepper cabesa1(STEPS, 6, 7, 8, 9);

Stepper cabesa2(STEPS, 10, 11, 12, 13);
```

```
const float VCC = 4.98; // Voltaje de 5V proporcionado por la placa de Arduino UNO

const float R_DIV = 47500.0; // Resistencia medida de la resistencia de 33k


const float STRAIGHT_RESISTANCE0 = 26600.0;

// Este es el valor de resistencia que nosotros tomamos como inicial en la resistencia
flexible, es decir, es el valor del sensor flexible cuando este no está flexionado.

const float BEND_RESISTANCE0 = 8100.0;

// Este es el valor de resistencia que  nosotros tomamos como final en la resistencia
flexible, es decir, es el valor del sensor flexible cuando flexionamos este lo máximo que
nos permiten los dedos.


const float STRAIGHT_RESISTANCE1 = 37000.0;

const float BEND_RESISTANCE1 = 7500.0;

// Tenemos un valor inicial y final para cada resistencia.

const float STRAIGHT_RESISTANCE2 = 20500.0;

const float BEND_RESISTANCE2 = 5000.0;


int cont0 = 0, cont1 = 0, cont2 = 0;

// Inicializamos un contador para cada motor que va a representar la posición de cada
cuchilla


void setup()

{


    cabesa0.setSpeed(400);

    cabesa1.setSpeed(400); // Establece la velocidad de los motores

    cabesa2.setSpeed(400);
```

```
}

void loop()
{
    int flexADC0 = analogRead(0);
// Lee el valor analógico de la entrada analógica PIN 0
    float flexV0 = flexADC0 * VCC / 1023.0;
// Calcula el voltaje equivalente al valor analógico
    float flexR0 = R_DIV * (VCC / flexV0 - 1.0);
// Calcula la resistencia del sensor flexible
    float angle0 = map(flexR0, STRAIGHT_RESISTANCE0,
BEND_RESISTANCE0, 0, 18.0);
// A partir de la resistencia calculada, establezco con un mapeado la
STRAIGHT_RESISTANCE como inicial y la BEND_RESISTANCE como final y
colocamos el valor de la resistencia flexible entre estos dos, después, lo extrapolamos al
rango que determinemos.

    int flexADC1 = analogRead(1);
    float flexV1 = flexADC1 * VCC / 1023.0;
    float flexR1 = R_DIV * (VCC / flexV1 - 1.0);
    float angle1 = map(flexR1, STRAIGHT_RESISTANCE1,
BEND_RESISTANCE1, 0, 18.0);

    int flexADC2 = analogRead(2);
    float flexV2 = flexADC2 * VCC / 1023.0;
    float flexR2 = R_DIV * (VCC / flexV2 - 1.0);
    float angle2 = map(flexR2, STRAIGHT_RESISTANCE2,
BEND_RESISTANCE2, 0, 18.0);
```

// Cuando el contador sea menor que angle-2 (para evitar movimientos innecesarios del motor por la alta sensibilidad de las resistencias flexibles) el contador suma 1 y el motor gira en sentido horario 30 pasos

```
if(cont0 < (angle0-2)){


    cont0 ++;

    cabesa0.step(30);

}
```

// Si el contador es mayor que angle+2, el contador resta 1 y el motor gira en sentido antihorario 30 pasos.

```
if(cont0 > (angle0+2)){


    cont0 --;

    cabesa0.step(-30);

}
```

// Si el valor del contador está entre los dos rangos de angle (incluyendo 0), el motor no se mueve, para poder parar el motor en caso de que no s edoble la resistencia.

```
if(cont0 > (angle0-2) && cont0 < (angle0+2)){

  cabesa0.step(0);

}


if(cont1 < (angle1-2)){


    cont1 ++;

    cabesa1.step(30);

}

if(cont1 > (angle1+2)){
```

```
        cont1 --;
        cabesa1.step(-30);
    }
    if(cont1 > (angle1-2) && cont1 < (angle1+2)){
      cabesa1.step(0);
    }

    if(cont2 < (angle2-2)){

        cont2 ++;
        cabesa2.step(30);
    }

    if(cont2 > (angle2+2)){

        cont2 --;
        cabesa2.step(-30);
    }
    if(cont2 > (angle2-2) && cont2 < (angle2+2)){
      cabesa2.step(0);
    }
```

// El proceso se repite en un bucle loop de tal forma que el contador correspondiente a cada motor siempre intenta alcanzar al angle de cada motor, por lo tanto al cambiar la flexión de la resistencia flexible, cambia la posición de la cuchilla.

// Los motores parecen que funcionan al mismo tiempo, pero realmente funcionan de forma consecutiva solo que la velocidad de trabajo es muy alta para que parezca que funcionan al mismo tiempo.

```
    }
```

4. MONTAJE DE LA PARTE MECÁNICA

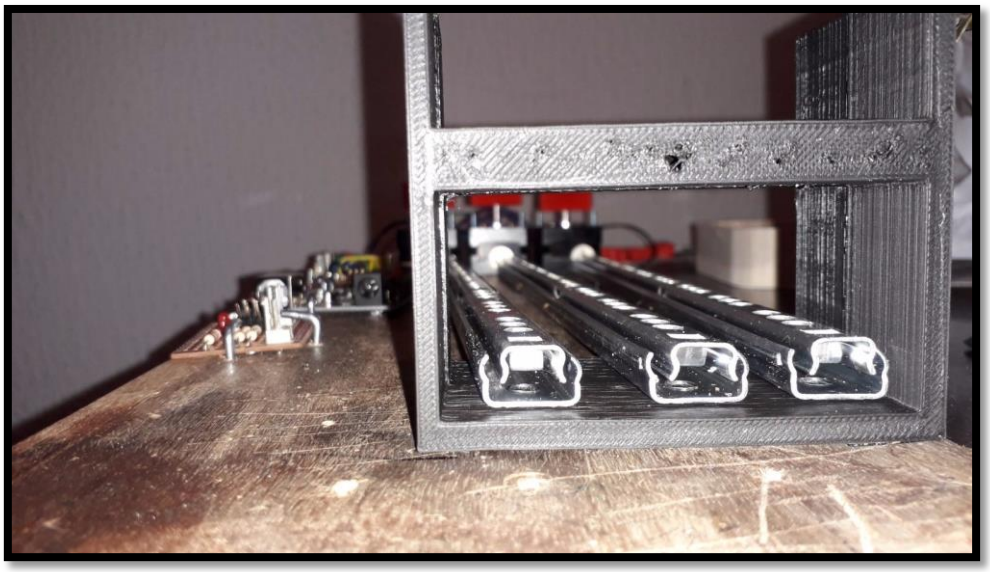
Para el montaje de la parte mecánica, se debería contar con las impresiones de los diseños adjuntados junto al proyecto. Esto queda mejor detallado en el apartado de incidencias.

En primer lugar, se ha procedido a anclar los motores paso a paso en el soporte donde irá montada toda la parte mecánica.

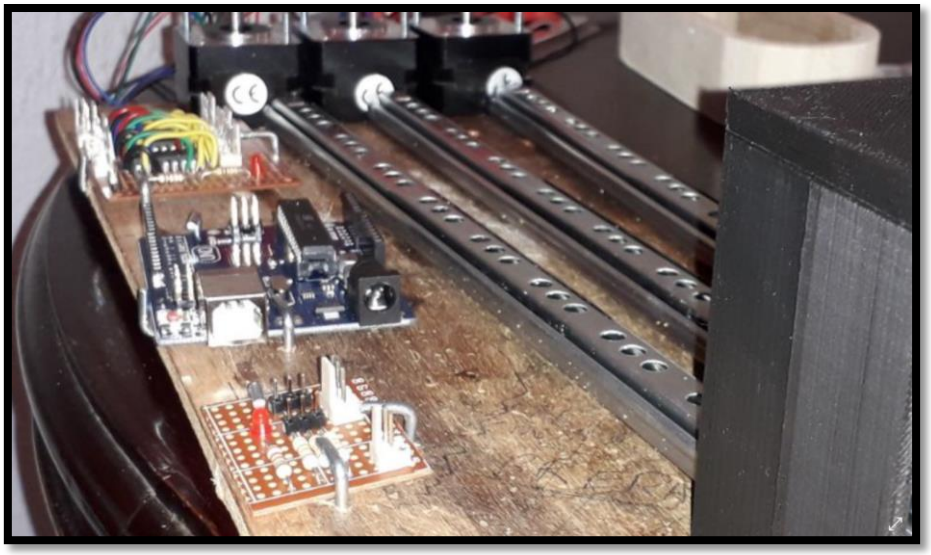
Se han taladrado los huecos en los que se van a anclar los motores, con tornillos de métrica 3, que son los que encajan en los huecos de los motores.

Una vez anclados los motores, se ubicarán los raíles de cajón, para marcar los huecos en los que se practicarán los taladros pertinentes.

Para la colocación de los raíles, habrá que situar la pieza que contendrá los rodamientos, ya que irá anclada en el mismo punto que el extremo final de los raíles, quedando todas las partes fijadas en los mismos 3 puntos, tal y como se ven en la imagen.



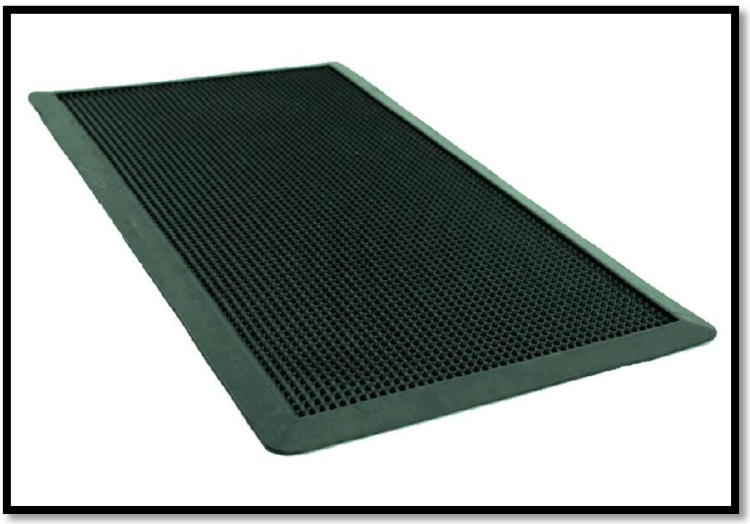
Una vez practicados todos los taladros, situamos las placas en su sitio, para afianzarlas al prototipo con ayuda de alcayatas.



Una vez fijadas las placas al prototipo pasamos a la construcción del brazal.

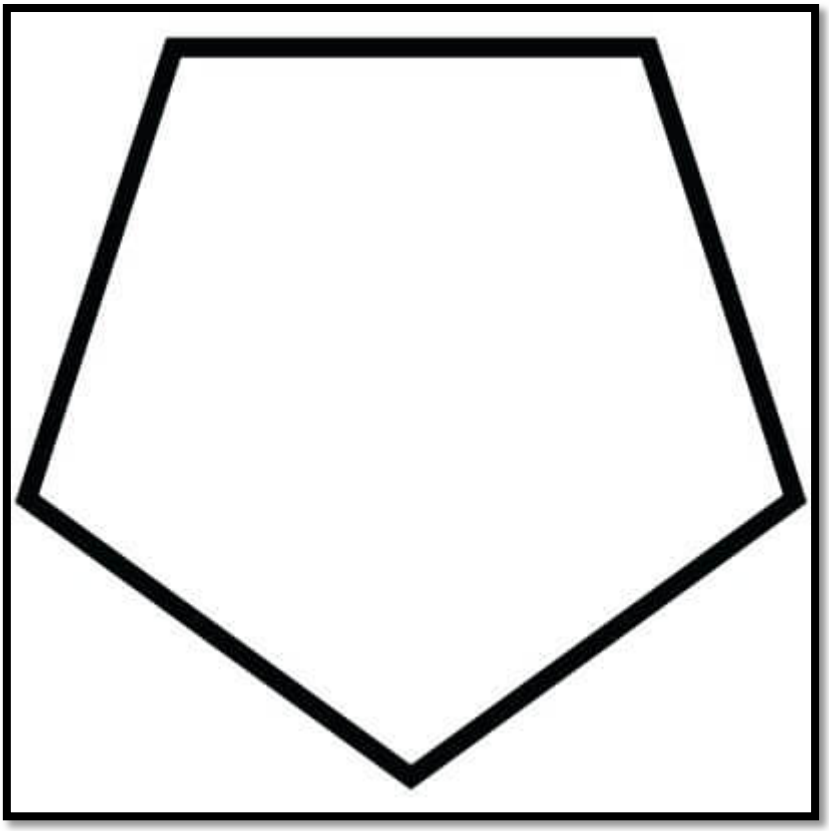
Tomaremos una pieza da caucho y la recortaremos a la misma medida que el soporte del prototipo, unos 35cm aproximadamente.

En nuestro caso, hemos usado un felpudo de caucho.



Recortaremos los bordes más gruesos, porque nos servirán para reforzar ciertas partes en las se harán una serie de perforaciones.

Una vez que tenemos la pieza de 35cm de largo, trazaremos un pentágono, que será lo que le dé la forma de brazal.



Una vez tenemos el pentágono, y con ayuda de un pegamento de contacto fijaremos los bordes que hemos reservado anteriormente. Los recortaremos y fijaremos en ambos lados de la base superior del pentágono. Ahí será donde se perforarán los huecos por los que pasaremos los cordones.

Con ayuda de un rotulador, marcaremos los puntos a perforar, intentando que quede lo más simétrico posible.

Cuando tengamos hecho los huecos, pasaremos los cordones, con cuidado, ya que el caucho se puede rajar aun estando reforzado.

Con esto ya tendremos la forma básica del brazal.

Lo siguiente que haremos, será colocar la batería en el lateral del brazal.

Para ello ubicaremos la batería en su posición final y marcaremos los puntos a perforar con ayuda de un rotulador, o con un marcador blanco.

Nosotros hemos hecho 7 puntos de unión a cada lado de la batería.

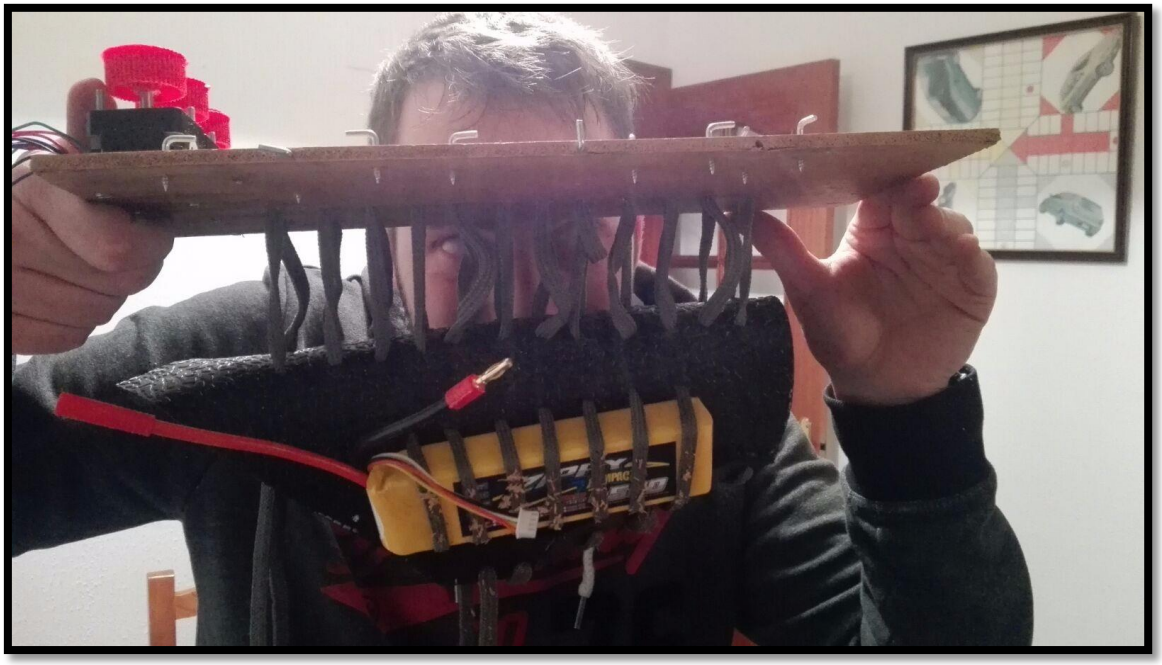
Una vez hechos los boquetes, "coseremos" la batería al brazal.

Ahora, nos falta unir la base del prototipo con el brazal.

Para unirlos, colocaremos la base sobre el brazal, y marcaremos a ambos lados del rail central puntos a taladrar cada 2cm.

Retiraremos los raíles para realizar las perforaciones.

Una vez realizadas, situaremos el brazal bajo la base, y con ayuda de un punzón, iremos agujereando el caucho en los mismos puntos que en la base del prototipo. Después, pasaremos los cordones, que "coserán" ambas partes, quedando completamente fijas.



Una vez fijado el prototipo, afianzaremos las ruedas dentadas en los ejes de los motores y procederemos a colocar los rodamientos en su soporte correspondiente.

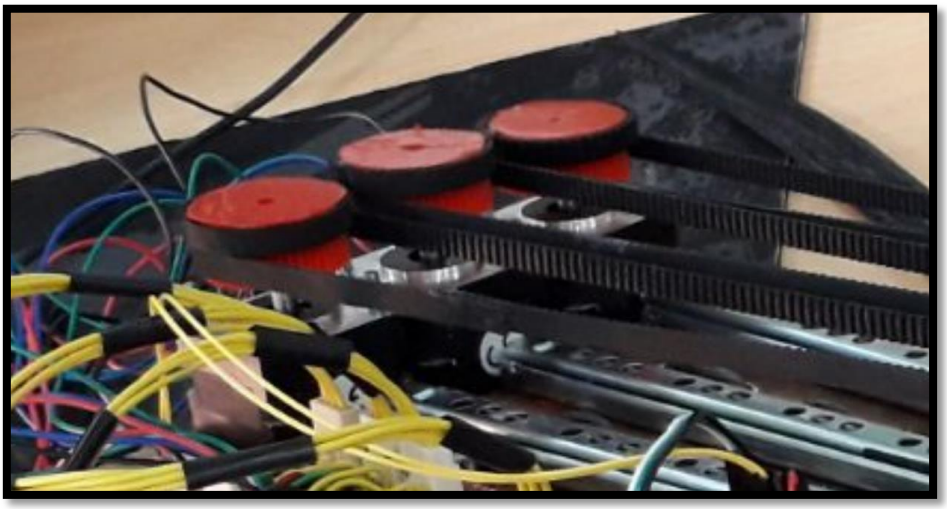
Para evitar la fricción de los rodamientos con el soporte, se colocarán unas arandelas, que también servirán para ejercer la presión necesaria para evitar que los rodamientos se muevan de su posición.

Los rodamientos serán atravesados con tornillos de métrica 3, y se pasarán por el hueco existente en el soporte para rodamientos. Serán necesarias un mínimo de 6 arandelas de un tamaño suficiente para afianzar el montaje de los mismos.

Completaremos el montaje del soporte de los rodamientos, con ayuda de unas bisagras pequeñas y un elemento de cierre, que mantenga la tapa fija en su sitio. Esto ayudará a prevenir que las correas se salgan.

Ahora que ya tenemos los rodamientos y las ruedas dentadas, se procede a colocar las correas.

Ajustaremos las correas a la medida que necesitemos, y las fijaremos con un pegamento de contacto. Si por casualidad, alguna de las correas tiende a salirse de la rueda dentada, fijaremos un trozo de correa a modo de anillo, en la parte superior de la misma. De esta forma, cuando la correa suba, chocará con el anillo, evitando que se salga.



Con esto damos por terminado el montaje de la base del prototipo. Ahora pasaremos al montaje de los soportes para las resistencias flexibles.

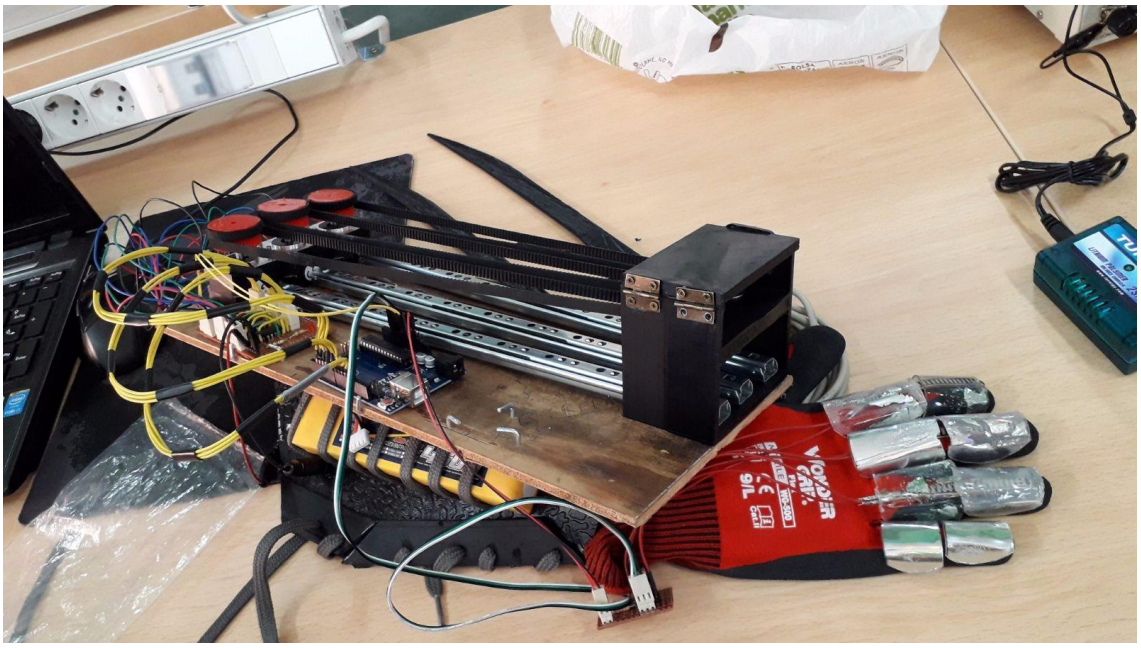
Recortaremos unos CD's evitando pasar por el hueco central ya que, si se recorta llegando al centro, se suelen resquebrajar dejando el disco inservible.

Con ayuda de soplador de aire caliente, moldearemos las piezas para darle forma de dedos.

Habrá que trabajar con un margen alto, ya que hay que situarlos en un guante y no directamente en la mano, con lo que deben de ser de una superficie mayor.

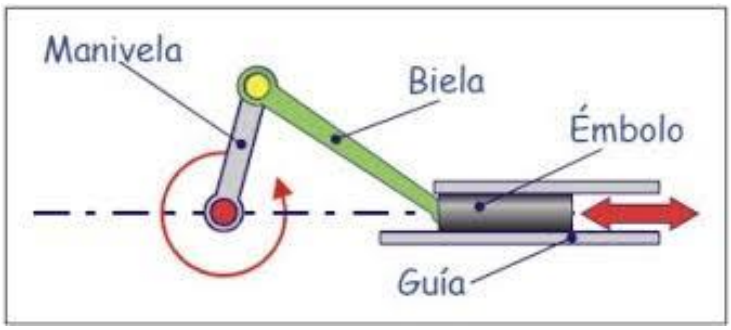
Una vez moldeados, los uniremos al guante pegándolos con pegamento de contacto, pero sólo por los laterales. Es importante no aplicar adhesivo en la parte del dorso de los dedos, ya que en esa zona es en la que introduciremos las resistencias flexibles.

El resultado será un prototipo similar al que se muestra acontinuación:



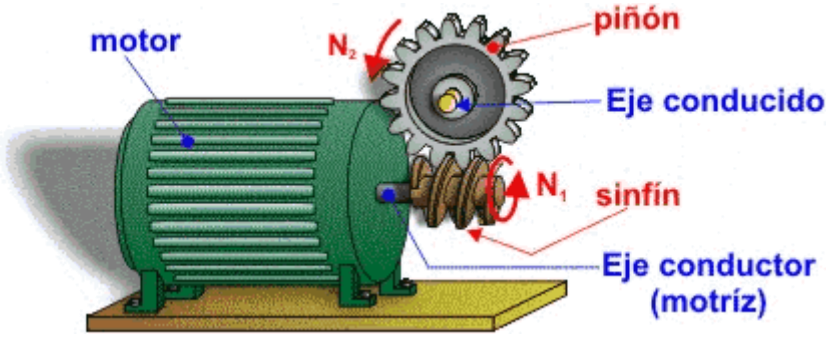
5. INCIDENCIAS

En primer lugar, nuestra intención fue realizar el proyecto con tres servomotores y un mecanismo de biela-manivela, pero aparte de que se nos complicaba mucho, el mecanismo debía ser muy grande y saldría bastante alto por encima del brazo cuando las garras estuvieran recogidas.



Entonces Enrique, nuestro profesor, nos dio la idea de usar un tornillo sin fin que, al girar mediante un motor paso a paso, movía una tuerca en un movimiento rectilíneo y uniforme a través del tornillo, es decir, la garra iría pegada a la tuerca y podríamos controlar su movimiento hacia delante o hacia atrás.

El principal problema de este diseño fue el tornillo sin fin. El que encontramos era de la impresora 3D por lo tanto tenía muchísimas vueltas, lo que ralentizaba el movimiento de las cuchillas muchísimo cuando necesitábamos que fuese a una velocidad considerable para poder sacar y meter las garras rápidamente. También pensamos en hacer el tornillo sin fin con la impresora 3D, pero nos iba a dar muchos fallos de impresión.



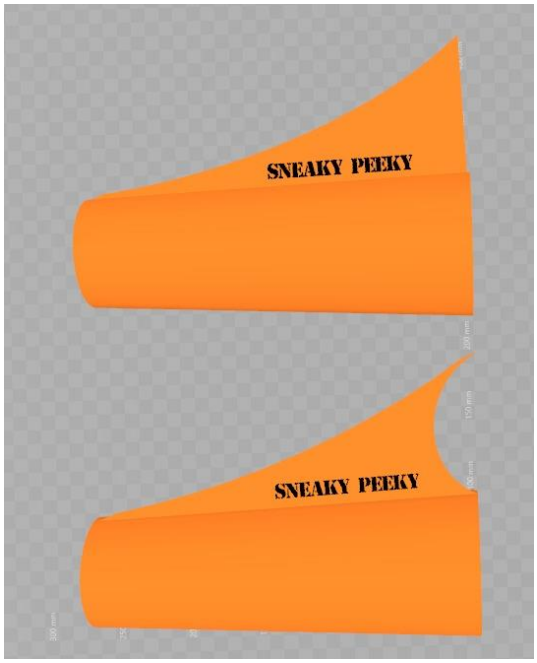
Finalmente, llegamos a la idea de usar un motor paso a paso, como con el tornillo sin fin, pero con una correa en los motores paso a paso hacia un rodamiento al final del recorrido, de tal forma que la cuchilla estuviese unida a la correa mediante una brida y la moviera a través del recorrido.

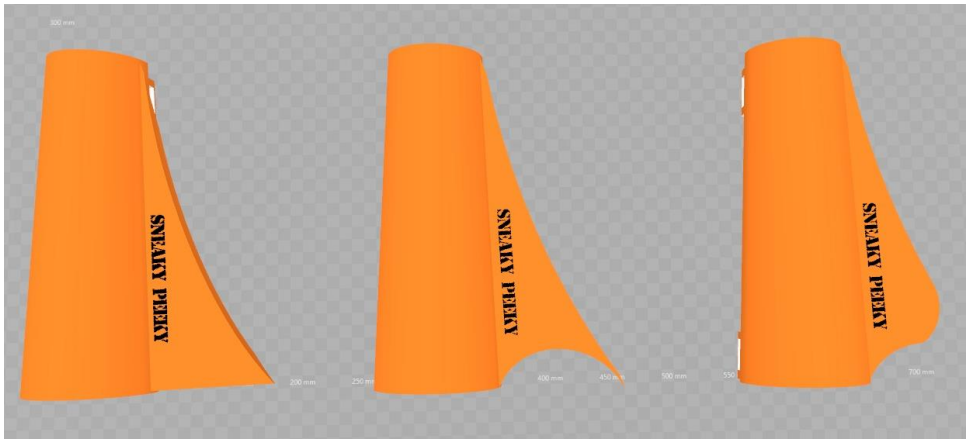
Antes de realizar el proyecto nos hemos encontrado con muchos problemas. En un principio queríamos realizar la parte mecánica en primer lugar, ya que era lo que nos ofrecía más conflictos, así que nos pusimos directamente a hacer diseños 3D para el brazal que no llegaríamos a tener. Tras un tiempo de espera sin hacer nada para la llegada de estas impresiones decidimos ponernos con la electrónica y la programación.

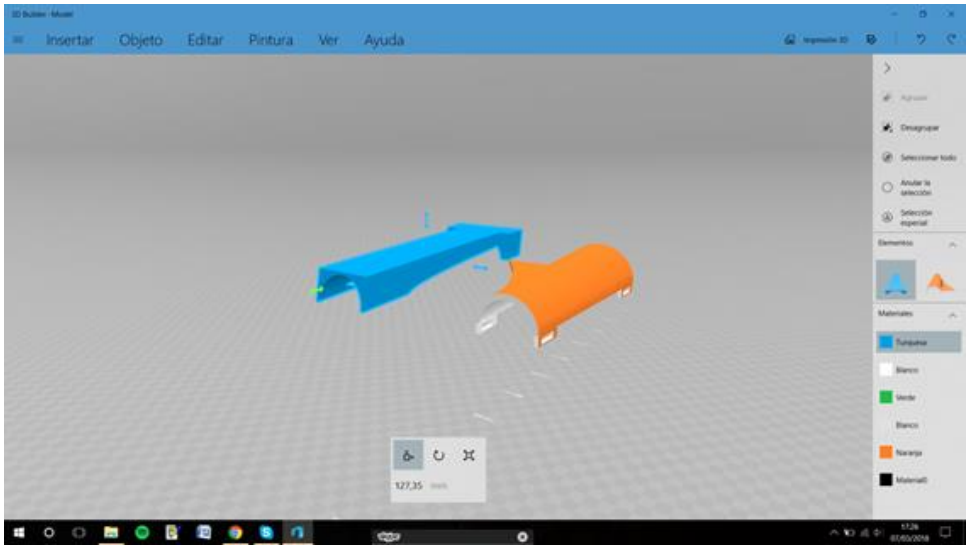
Además, se hizo un pedido de una serie de piezas, como puedan ser los rodamientos para las correas, las correas y la batería de 12 V.

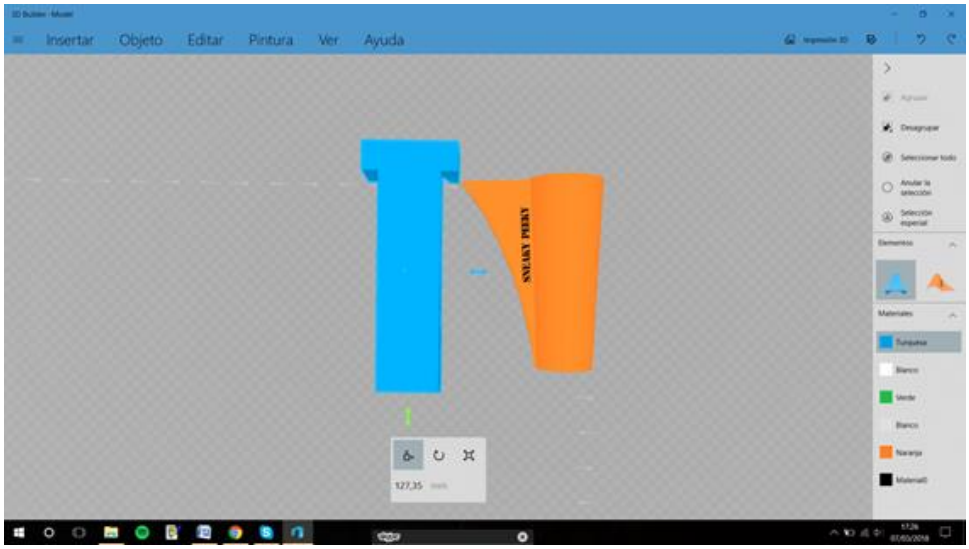
5.1. Diseños 3D

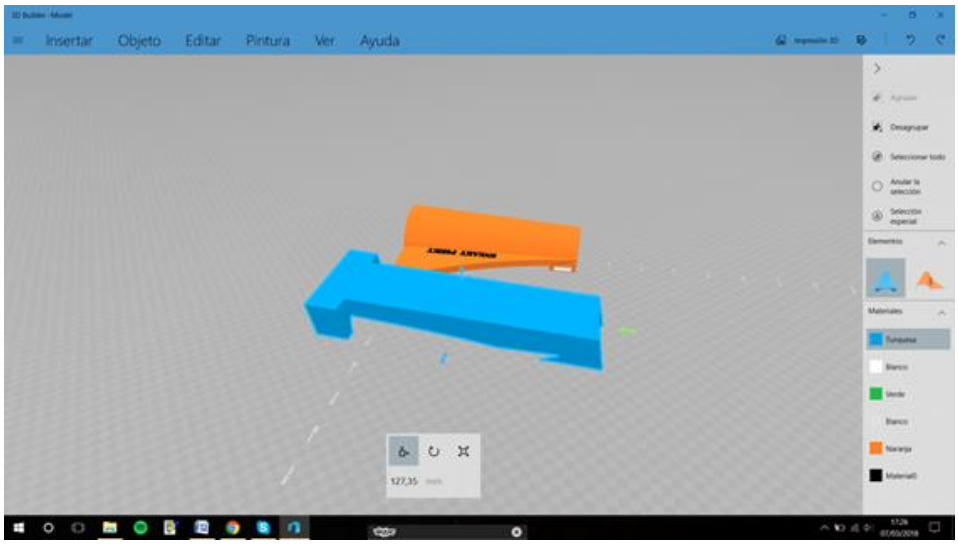
Diseño 3D de brazal:

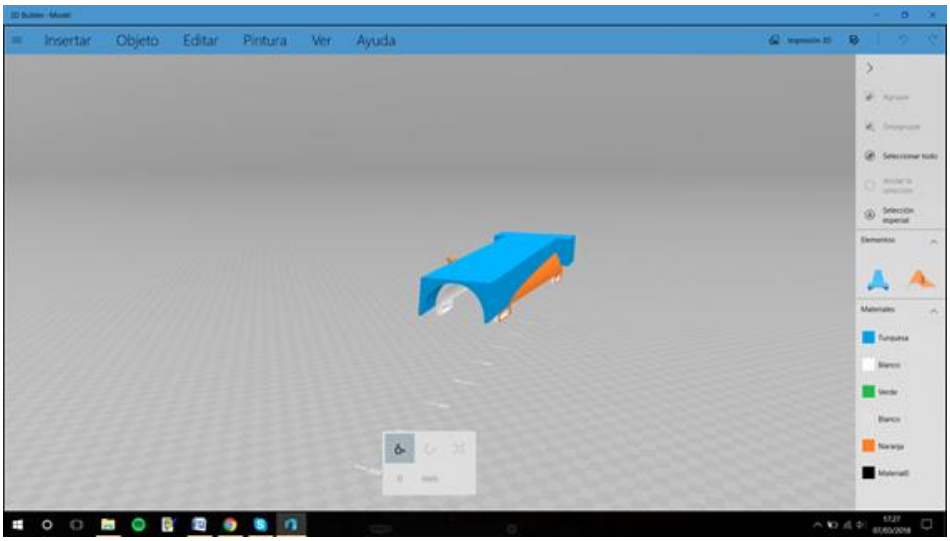


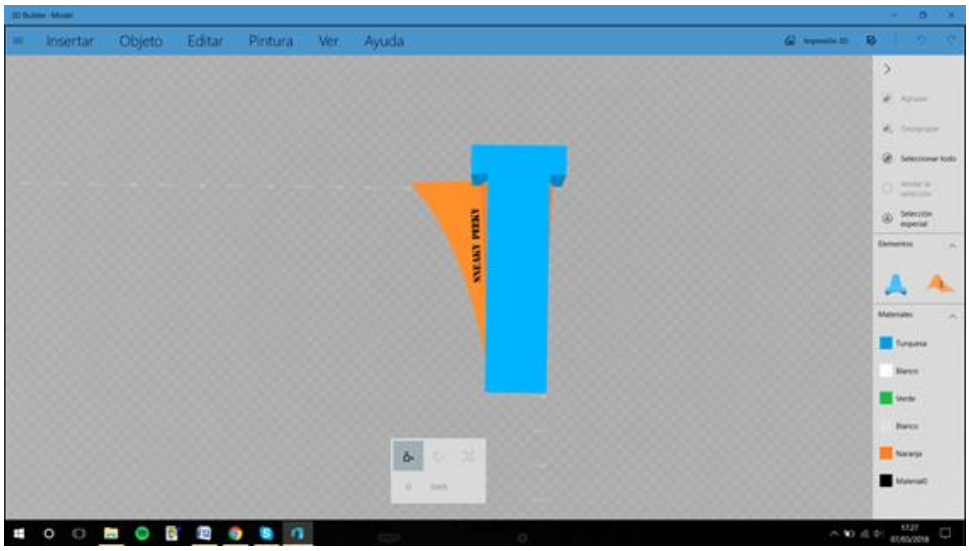

SNEAKY PEEKY
SNEAKY PEEKY
SNEAKY PEEKY

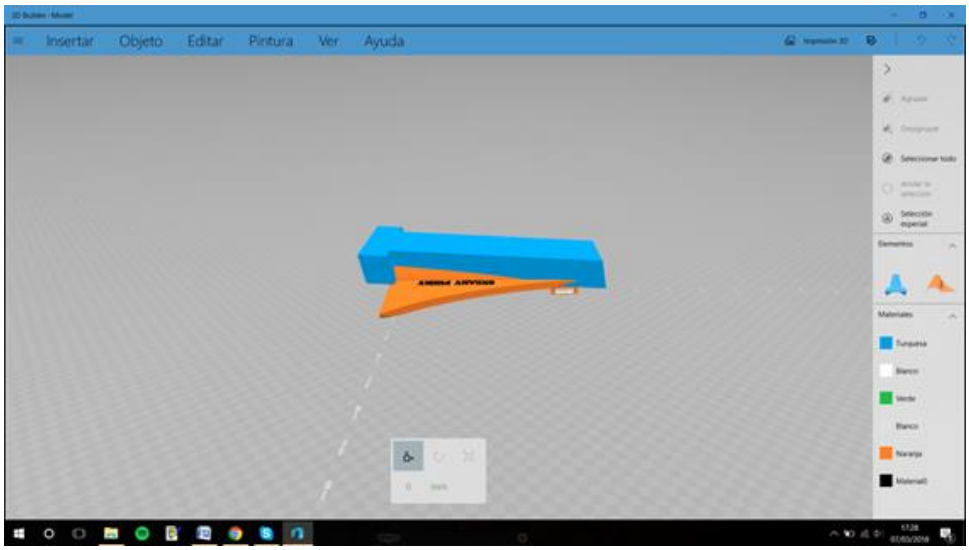

Insertar
Objeto
Editar
Pintura
Ver
Ayuda


Insertar
Objeto
Editar
Pintura
Ver
Ayuda
SNEAKY PEEKY

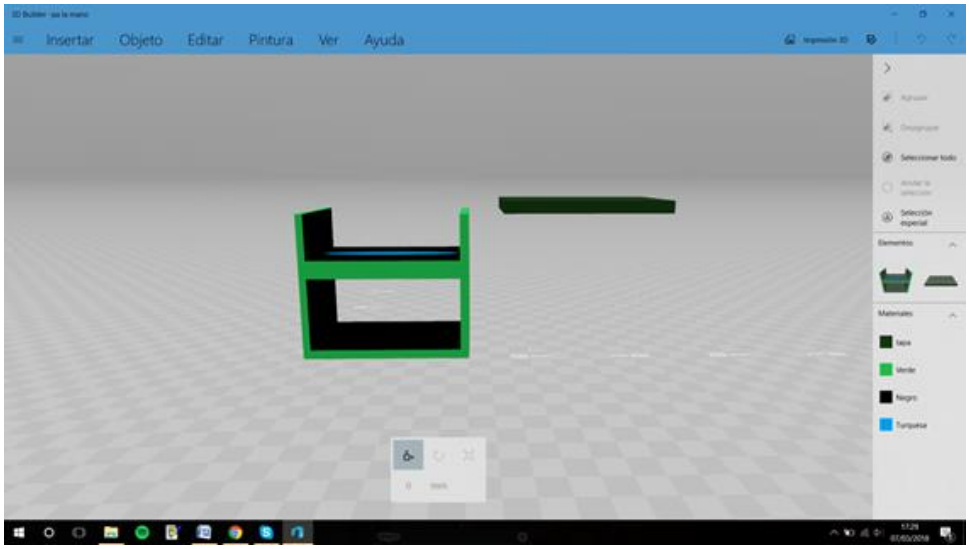

Insertar
Objeto
Editar
Pintura
Ver
Ayuda

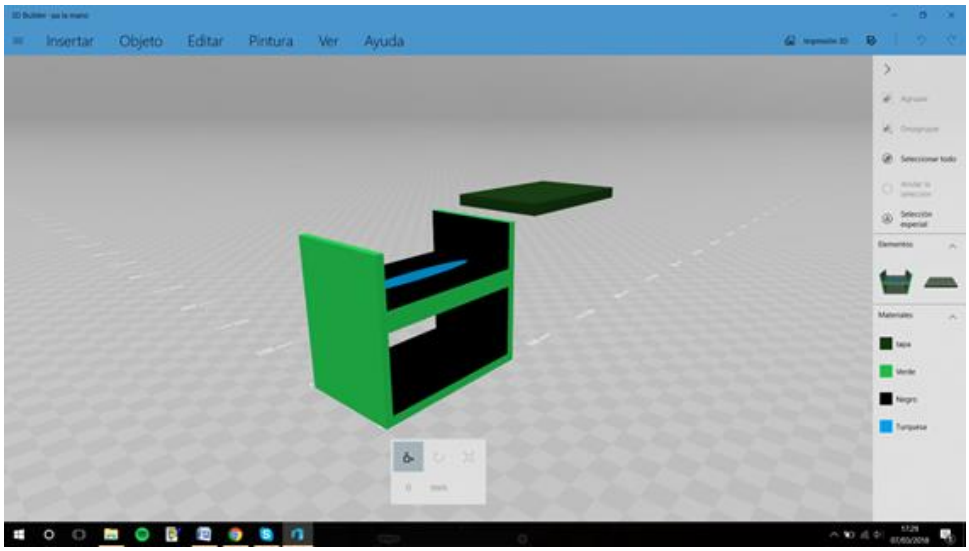

Insertar
Objeto
Editar
Pintura
Ver
Ayuda

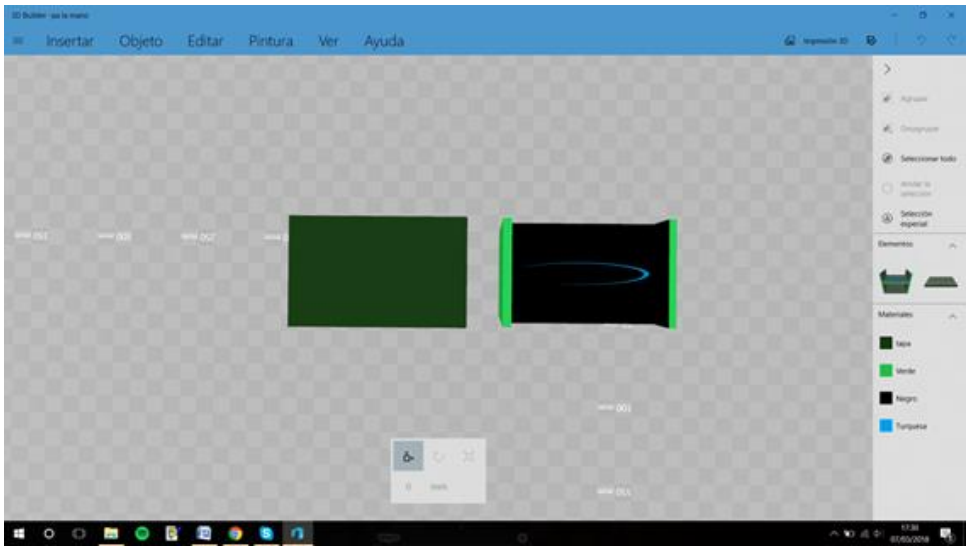

Insertar
Objeto
Editar
Pintura
Ver
Ayuda



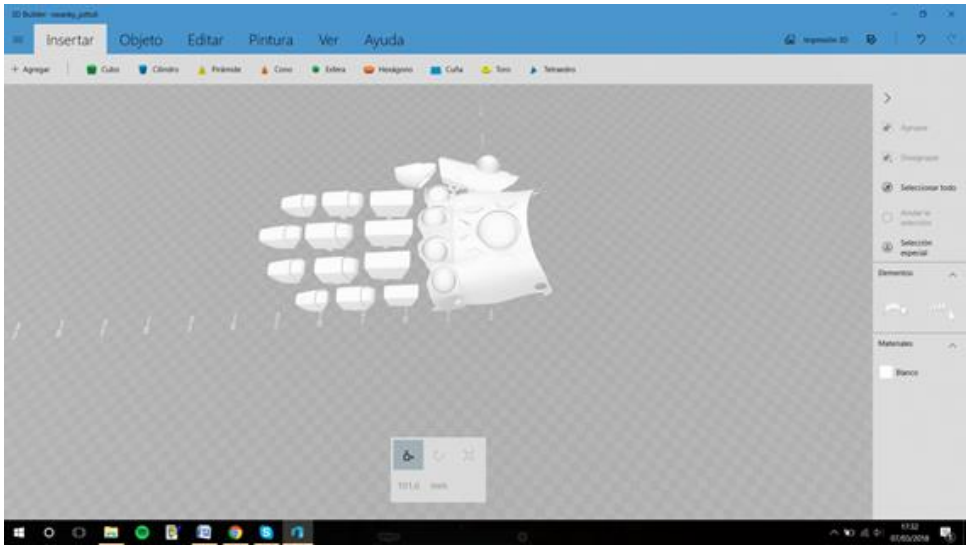
Diseño 3D de soporte para rodamientos:

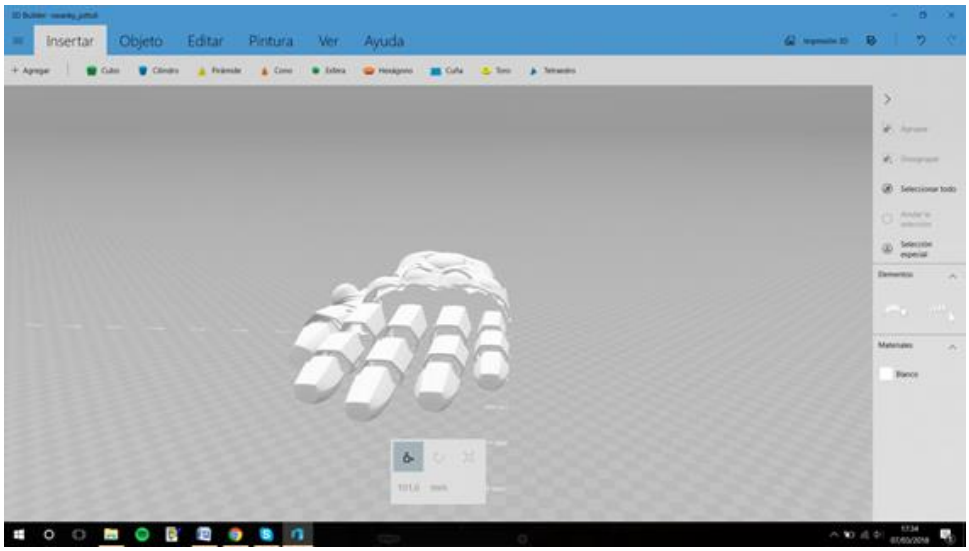


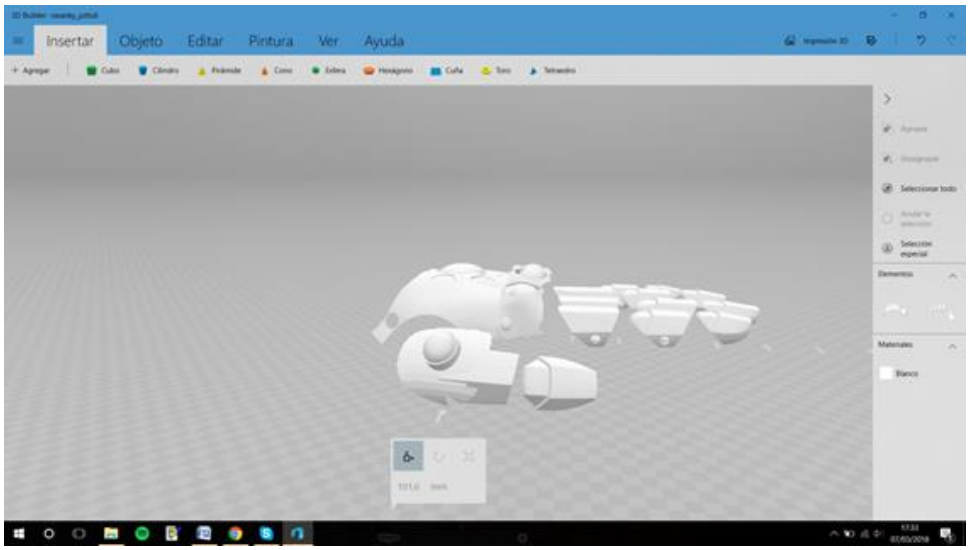




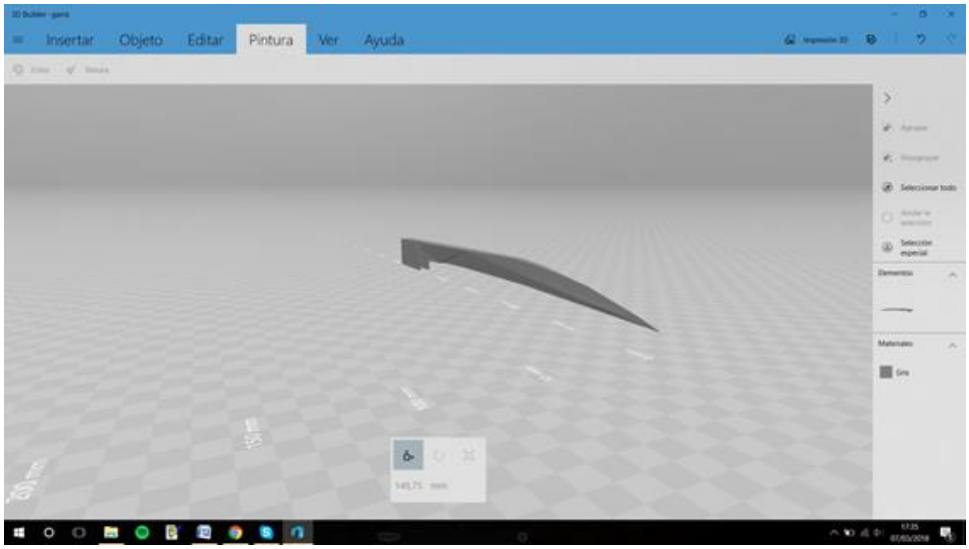
Diseño 3D para los guantes:

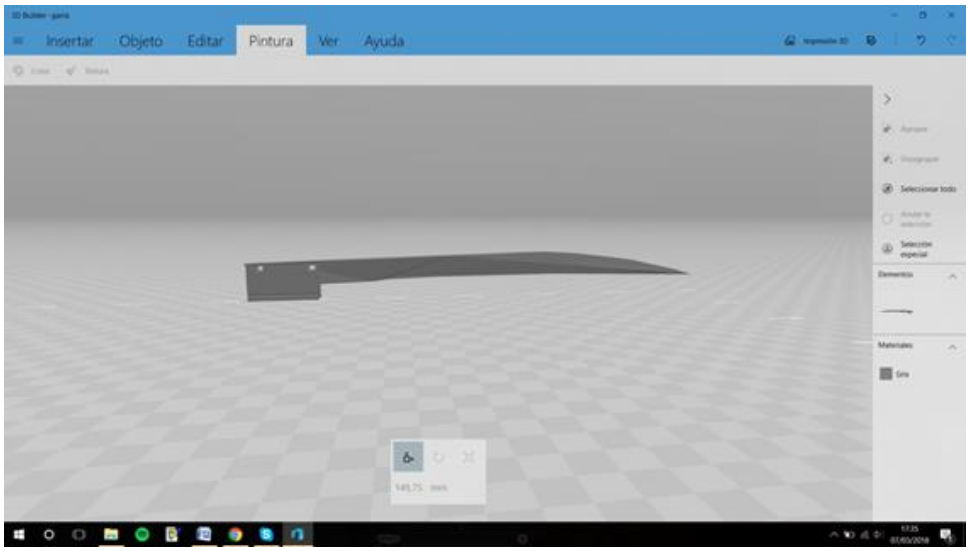


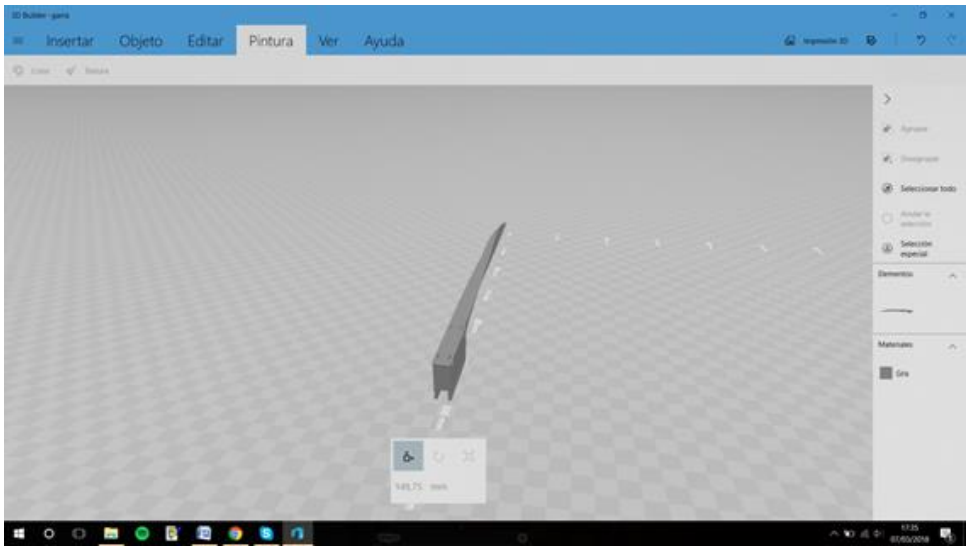




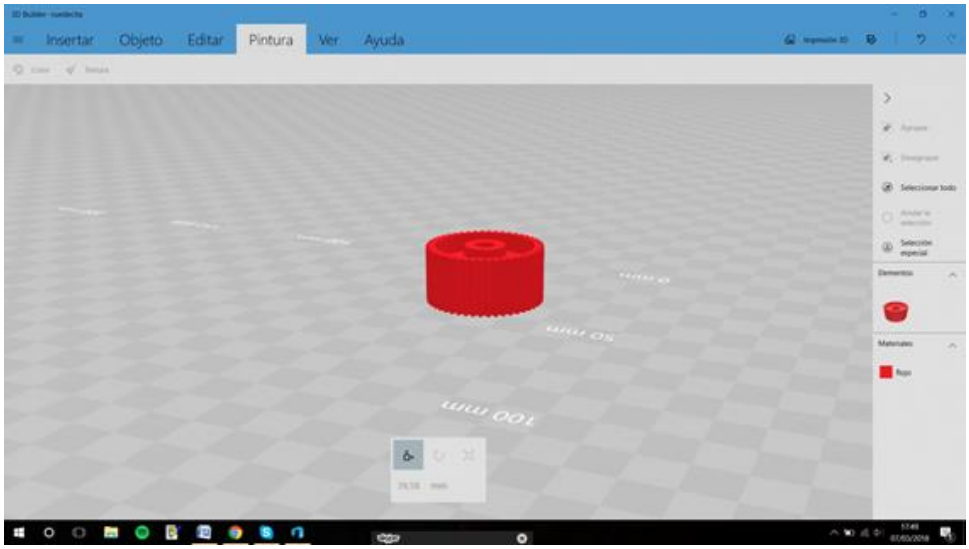
Diseño 3D de las cuchillas:

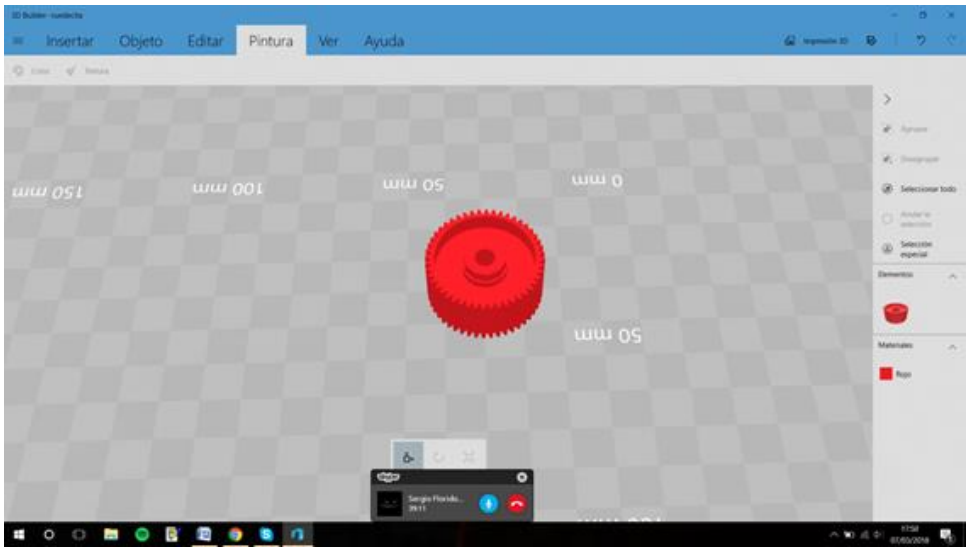






Diseño 3D de las ruedas dentadas:

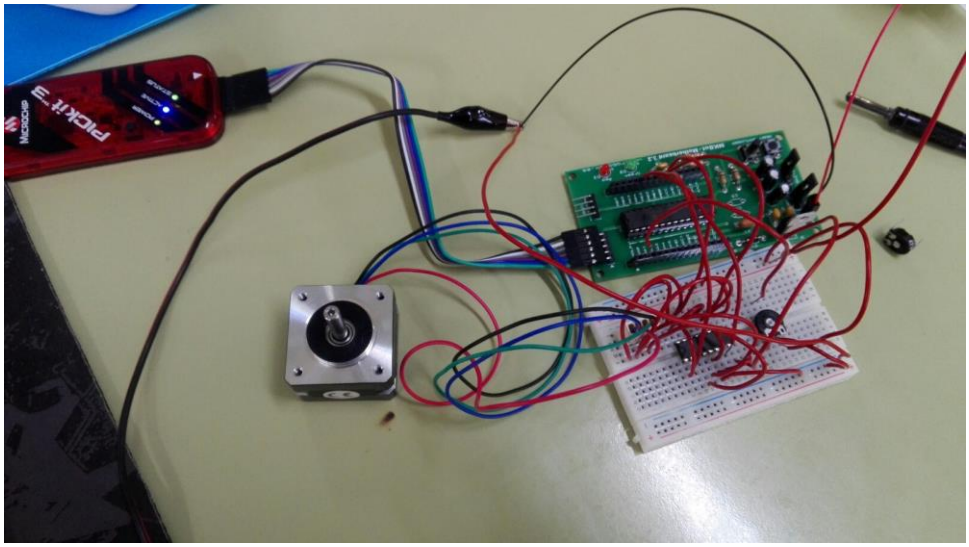




Antes de nada, queríamos probar los motores y su control, al menos de uno, para ver su potencia, su consumo y su control para ver si necesitaríamos otro tipo de motores, y así fue. Tuvimos que cambiar nuestros motores paso a paso por unos con mayor torque ya que teníamos que tirar en el proyecto de la correa con las cuchillas. Después descubrimos que no nos servía con una batería de 7V para que los motores tuvieran suficiente torque trabajando al mismo tiempo, así que tuvimos que pedir una batería de 12V.

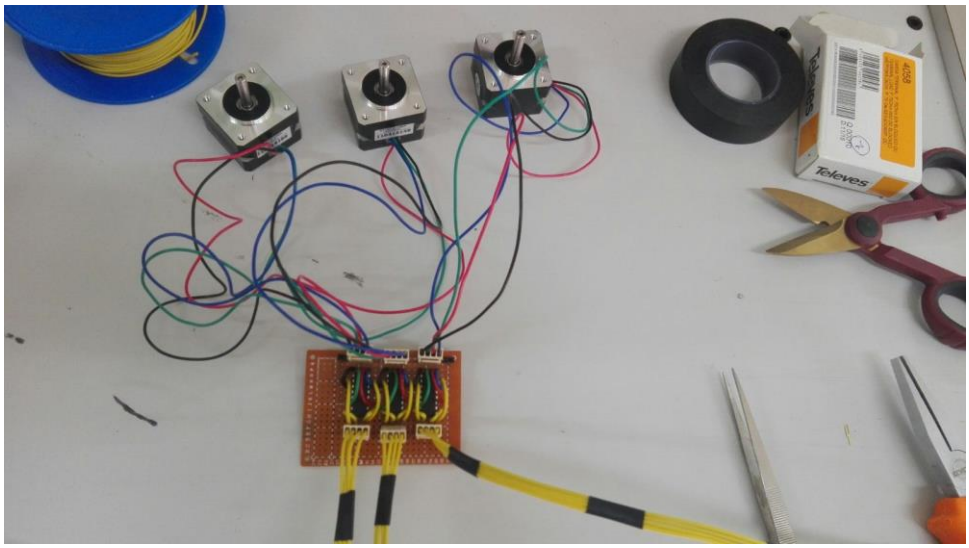
Para el control de los motores, estuvimos usando la placa Motherboard 3.2 con su respectivo PIC18F2550 junto con los drivers L293B que encontramos por internet que podían usarse para el control de los motores paso a paso.

Estuvimos un tiempo liados con los esquemas de conexionado con el Driver, ya que el único esquema de conexionado que encontramos por internet no estaba correctamente conectado, tenía dos salidas intercambiadas entre sí. Una vez que lo conseguimos arreglar, creamos nuestra placa electrónica donde teníamos los tres Drivers y sus conexiones.



El siguiente problema surgió con la programación. Después de muchos intentos, nos dimos cuenta de que nos era prácticamente imposible realizar este proyecto con PIC, ya que el control de un motor no iba fluido de ninguna manera, no conseguíamos que el motor se quedase 100% quieto cuando debía, sino que oscilaba o vibraba y el control de 3 motores al mismo tiempo era imposible, ya que el control de estos motores en PIC se hace mediante delays, es decir, bloqueos completos del PIC. Por lo tanto, nos cambiamos a Arduino.

Además, cuando el PIC trataba de mover los 3 motores a la vez, la Motherboard entraba en corto.



En una ocasión y tras haber realizado previamente algunas pruebas de control de motores, el sistema no funcionaba. Empezamos a verificar el funcionamiento de todos los elementos sin encontrar ningún error aparente. Tras varias horas buscando el fallo, detectamos que había un corto en la placa de los drivers de los motores. Una vez solucionado, todo funcionó correctamente.

También, hemos tenido que cambiar los conectores de los cables en varias ocasiones, ya que la conexión que ofrecían era bastante pobre.

Hubo otra ocasión en que la crimpadora desapareció del taller, y tuvimos que crimpar los cables pinzas y alicates, ralentizando notablemente el proceso.

El montaje de la placa de drivers, se realizó contando con utilizar el conector de carga de la batería, ya que teníamos acceso a 3 niveles distintos de tensión, de esta forma podíamos alimentar el PIC con 7V y los motores con 12V, pero luego tuvimos que optar por elegir el conector principal de la batería, ya que se nos aconsejó dejar de usar el conector de carga, para que dicho uso no afecte a la integridad de la batería.

Tuvimos que adaptar los conectores de la batería, porque no había conectores compatibles con nuestra placa.

5.2. Programación en PIC:

```
7
8
9   #include <xc.h>
10  #include <pic18f2550.h>
11  #pragma config FOSC = HS // Oscillator Selection bits (HS oscillator (HS))
12  #pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabl
13  #pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is plac
14  #define _XTAL_FREQ 20000000 // 20MHz
15
16  void conf_analog()
17  {
18   TRISAbits.TRISA0 = 1;
19   TRISAbits.TRISA1 = 1;
20   ADCON1bits.VCFG = 0b00; // Configura tensiones de referencia a 0 y 5V
21   ADCON1bits.PCFG = 0b1101;// Configura AN0 analógico y el resto digitales
22   ADCON2bits.ADCS = 0b111; // Configura tiempo de conversión para 20MHz
23   ADCON2bits.ACQT = 0b011; // Configura tiempo de adquisición 6TAD
24   ADCON2bits.ADFM = 0; // Dato justifica a la izquierda, resolución 255
25  }
26
27  void horario1(){
28          LATB=0b00001010;
29          __delay_ms(2);
30          LATB=0b00001001;
31          __delay_ms(2);
32          LATB=0b00000101;
33          __delay_ms(2);
34          LATB=0b00000110;
35          __delay_ms(2);
36  }
37
38  void antihorario1(){
39          LATB=0b00000110;
40          __delay_ms(2);
41          LATB=0b00000101;
42          __delay_ms(2);
43          LATB=0b00001001;
44          __delay_ms(2);
45          LATB=0b00001010;
46          __delay_ms(2);
47  }
```

```
48
49  void horario2(){
50          LATC=0b10100000;
51          __delay_ms(2);
52          LATC=0b10010000;
53          __delay_ms(2);
54          LATC=0b01010000;
55          __delay_ms(2);
56          LATC=0b01100000;
57          __delay_ms(2);
58  }
59
60  void antihorario2(){
61          LATC=0b01100000;
62          __delay_ms(2);
63          LATC=0b01010000;
64          __delay_ms(2);
65          LATC=0b10010000;
66          __delay_ms(2);
67          LATC=0b10100000;
68          __delay_ms(2);
69  }
70  void main(void) {
71      int abc = 0;
72      int def = 125;
73      int ghi = 0;
74      int jk = 125;
75      conf_analog();
76      TRISB=0b00000000;
77      TRISC=0b00000000;
78      while(1){
79          ADCON0bits.ADON = 1; // Habilita el módulo convertidor
80         ADCON0bits.CHS = 0b0000; // Selecciona el AN0 para la conversión A/D
81         ADCON0bits.GO_DONE = 1; // Lanzamos la conversión
82         while (ADCON0bits.GO_DONE == 1) // Espera que se realice la conversión
83   {
84   }
85   abc = ADRESH; // Cargo el valor de 8 bits en la variable adc (0-255)
```

```
86
87  if(abc>(def+5) || abc<(def-5)){
88
89      if(abc>def){
90
91              for(def; def<=ADRESH; def++){
92                  horario1();
93              }
94
95          }
96      if(abc<def){
97
98              for(def; def>=ADRESH; def--){
99                  antihorario1();
100             }
101
102         }
103     }
104         ADCON0bits.ADON = 1; // Habilita el módulo convertidor
105         ADCON0bits.CHS = 0b0001; // Selecciona el AN1 para la conversión A/D
106         ADCON0bits.GO_DONE = 1; // Lanzamos la conversión
107         while (ADCON0bits.GO_DONE == 1) // Espera que se realice la conversión
108 {
109 }
110 ghi = ADRESH; // Cargo el valor de 8 bits en la variable adc (0-255)
111
112 if(ghi>(jk+5) || ghi<(jk-5)){
113
114     if(ghi>jk){
115
116             for(jk; jk<=ADRESH; jk++){
117                 horario2();
118             }
119
120         }
121     if(ghi<jk){
122
123             for(jk; jk>=ADRESH; jk--){
124                 antihorario2();
125             }
126
127         }
128     }
```

Una vez que empezamos a usar Arduino, todo empezó a ir un poco mejor, hasta que nos topamos con el uso de las resistencias flexibles y los motores al mismo tiempo. Resultó que no podíamos usar la entrada y salida de puerto serie, como un puerto normal, por lo tanto, siempre nos daba error el programa a la hora de subir el programa a la placa de Arduino UNO.

Conseguimos arreglarlo, pero ahora teníamos un problema con las resistencias flexibles y es que variaban demasiado y de forma distinta dependiendo de la dirección hacia la que la doblábamos, por lo tanto, tuvimos que buscar una forma por internet de calibrar las resistencias flexibles.

Finalmente, tras mucha espera, nos llegó algo de las impresiones 3D, aunque mucho menos de lo que esperábamos, solo el soporte de los rodamientos, que, aunque era de lo más imprescindible, nos dejaba sin parte mecánica en el proyecto. Por lo tanto, a falta de tiempo, nos tuvimos que poner a inventarnos un brazal nuevo sin impresión 3D barato y factible. Básicamente fuimos a un local de venta de productos chinos y con un poco (bastante) de ingenio e imaginación acabamos comprando los materiales con los que al final haríamos nuestro brazal provisional.

Otro de los fallos que observamos trabajando con arduino, era que, aparentemente no se movían los motores, pero sí estaban trabajando. En un principio, colocamos una brida a modo de banderilla en los ejes de los motores, para controlar la dirección en que estaban girando. Cuando colocamos las ruedas dentadas, el hueco para el eje de las mismas no presentaba oposición al giro del eje del motor, con lo cual el eje se movía, pero la rueda no. Esto se solucionó en el momento en que fijamos la rueda dentada al eje.

6. DOCUMENTACIÓN

Control de motores paso a paso con Arduino:

http://diymakers.es/mover-motores-paso-paso-con-arduino/

Montaje de brazal de caucho:

https://youtu.be/WqiqKVSCn44

Calibración de resistencia flexible:

http://www.lunegate.net/2016/07/tutorial-aprender-usar-un-sensor-flex_31.html#.WqAuQejOVPY