# Codename Betaglo

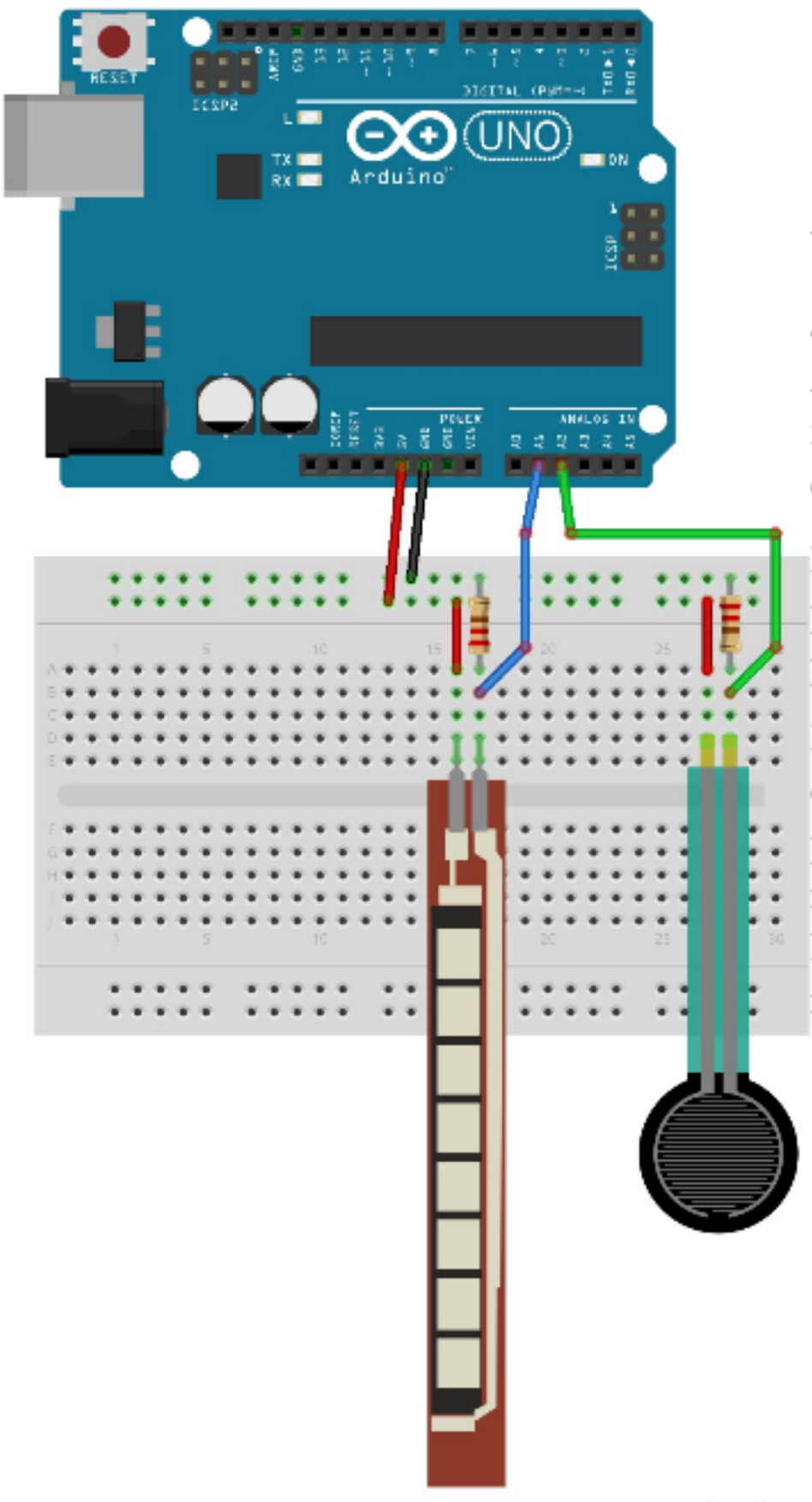
Antonio Molina - https://github.com/amolid00
Dorian Hawkmoon - https://github.com/DorianHawkmoon
Javain - https://github.com/Javain

# Índice

# Primera fase: El guante mágico

La idea de controlar con la mano nos gustaba, pero nos parecía una aplicación un poco pobre, asi que echándole imaginación, llegamos a la idea de un guante con el que poder controlar algo, como pudiera ser un ordenador, que era el que más juego daba para programar.

Consideramos pues los sensores adecuados al proyecto, que eran unos flexores para detectar la flexión de los dedos, y unos sensores de presión/fuerza para detectar los toques en las puntas de los dedos. El esquema de conexión para ambos sensores es el siguiente:




Ambos sensores disponen de una conexión a 5V mientras que en el otro tienen una conexión a masa con una resistencia de unos 10K. Algunos sitios comentan 22K o 30K. La influencia de la resistencia viene dada por el divisor de tensión.

El divisor de tensión es un circuito linear pasivo, que produce una tensión de salida proporcional al voltaje de entrada.

Los sensores de por si son resistencias y cambian según la presión/flexión recibida, pero para medir esos cambios necesitamos una resistencia fija (divisor de tensión). Se ajustan los valores a un rango menor y se mide la resistencia del sensor.

Al comprar los materiales, seleccionamos unos flexores de 7cm y unos sensores de fuerza de 2cm de diámetro junto a las resistencias de 10K. Todos los componentes se testearon por separado y en conjunto para comprobar su correcto funcionamiento. El programa en arduino para dicho testeo es bastante trivial:

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    int value=analogRead(2); //pin of the sensor
    Serial.println(value);
}
```

Comprobando que los sensores se comportaban como era de esperar y eran lo suficientemente precisos y fiables para el uso que le íbamos a dar, el siguiente paso era realizar el guante que controlaría el ordenador para posteriormente programarlo y probarlo.

Como detalle, aunque se veía un proyecto fácil y sencillo, decidimos que como era la primera vez, sólo haríamos un guante en vez de la pareja, y dejar el segundo guante como ampliación del proyecto.

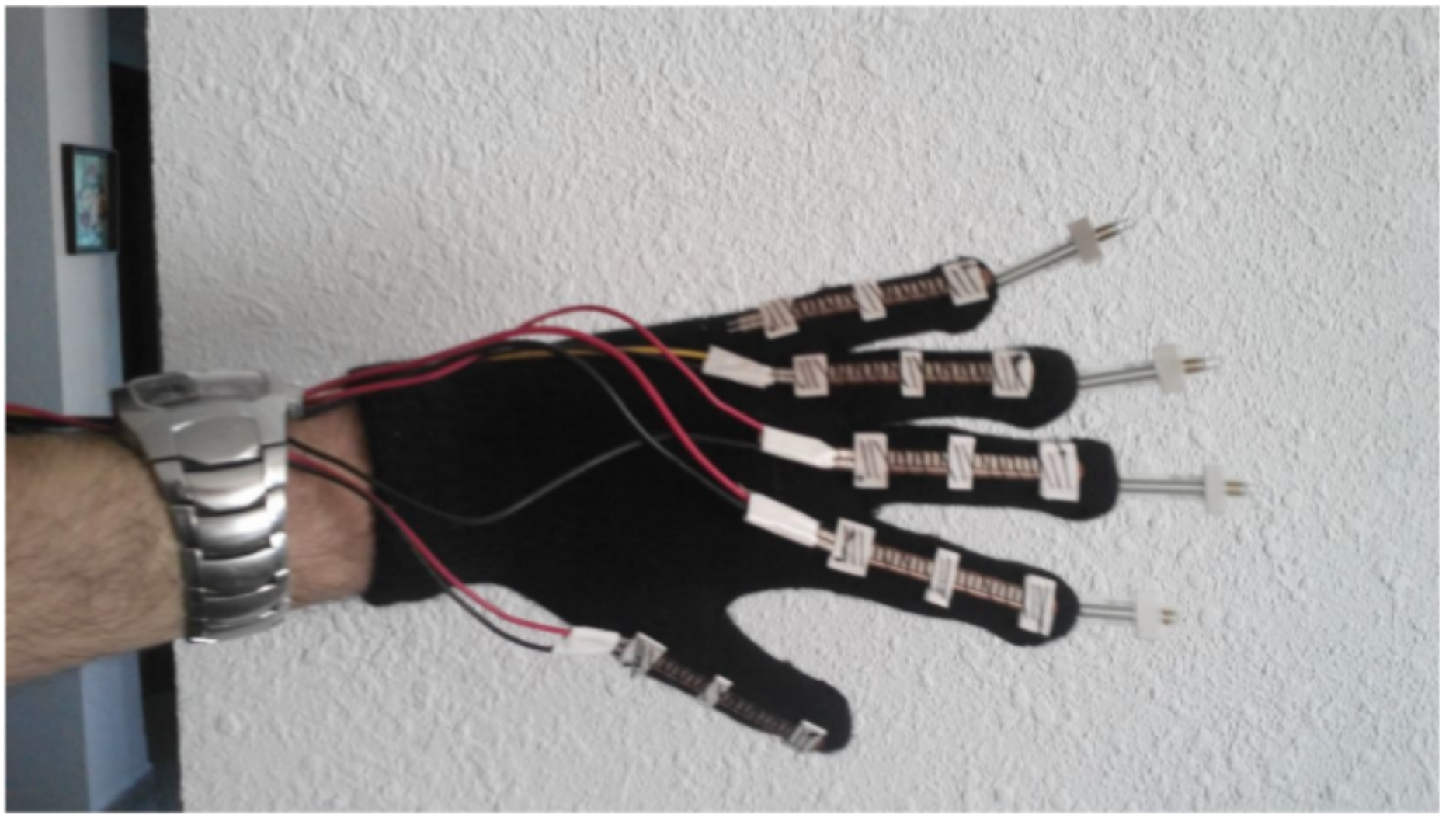
## Segunda fase: Quemarse y pincharse los dedos

Pillamos unos guantes normales de lana y el primer problema a resolver era sujetar los sensores al guante, de forma que se quedaran fijos y detectaran los movimientos cuando nos pusiéramos los guantes. Los sensores de flexión no tienen forma de ser cosidos o pegados al guante, por lo que improvisamos con un poco de cinta aislante para poder tener una superficie donde coser y fijar mientras que el resto aunque sin coser, quedaba sujetado con el hilo. Procedimiento similar hicimos con los sensores de presión aunque estos disponían de una superficie que se pegaba.

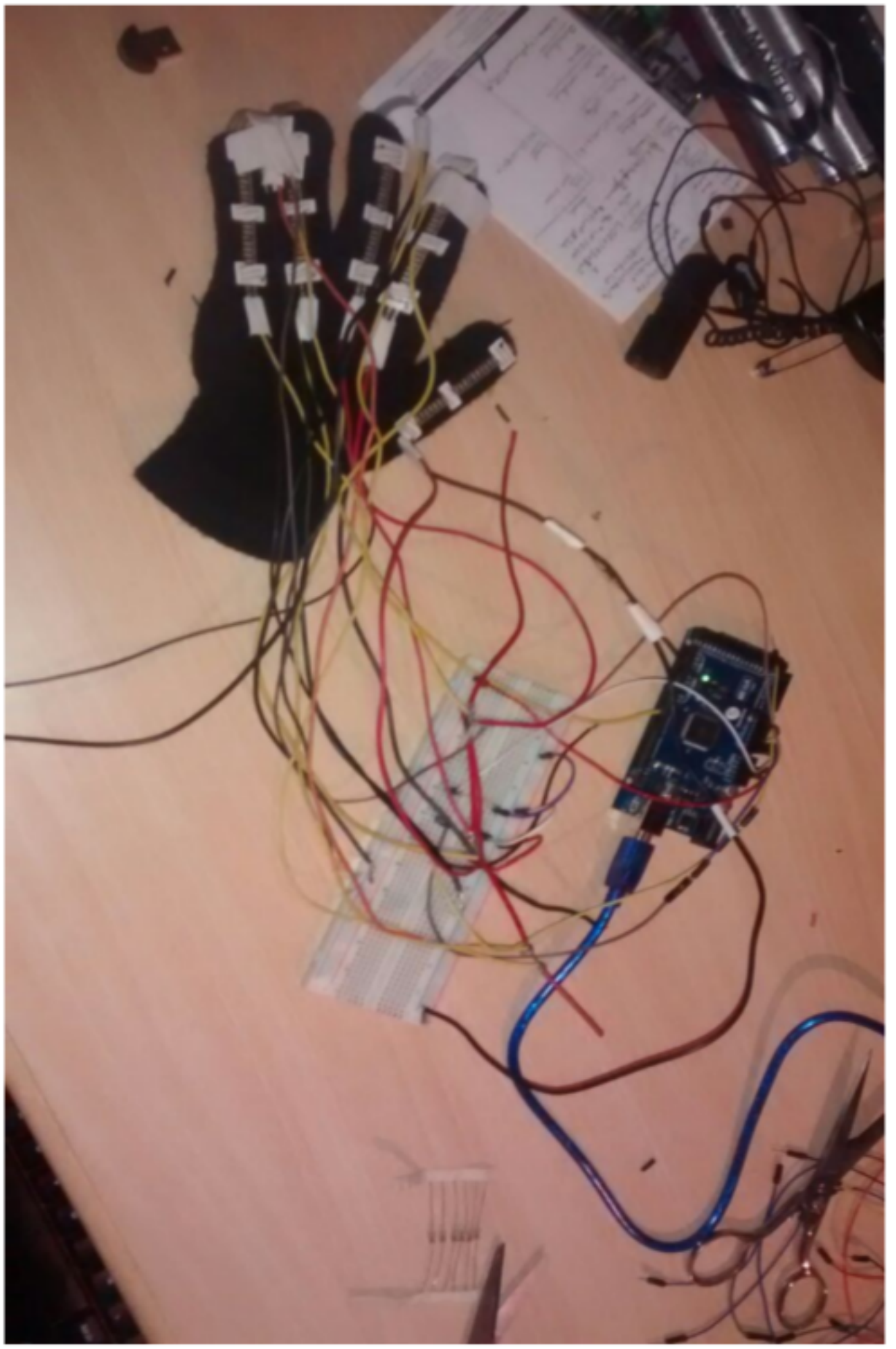
Varias veces hubo que coser y descoser, ya sea porque cosíamos lo que no era, porque quedaba mal colocado o porque cosíamos más de la cuenta, pillando tela de debajo del dedo. Aprovechamos cuando quedaron  cosidos los flexores para comprobar que tal quedaban en la mano. Inicialmente usamos la mano derecha y colocamos los flexores por debajo del dedo, pero viendo que no se doblaba bien y se clavaba los extremos, decidimos ponerlos por encima, por lo que el guante pasó a la mano izquierda.

Igualmente vimos que los pillamos ligeramente cortos, probablemente hubiera sido mejor los flexores de 11cm. De la misma forma, no queríamos pillar unos sensores de presión muy pequeños, por lo que escogimos los de 2cm de diámetro y vimos que eran un poco grandes para ser cómodos. Pero para el caso sirve igualmente.

La fase de soldar fue con diferencia la más larga y trabajosa del proyecto. Al principio, con nuestra inexperiencia y miedo de estropear los sensores con el calor, soldamos con el estaño y con cuidado los extremos de los sensores con cables reciclados de una fuente de alimentación, teniendo buenos resultados.



Inicialmente soldamos a todos los sensores un cable por cada extremo y los probamos para comprobar que seguían funcionando.



Posteriormente, siguiendo el esquema de conexión, de uno de los cables practicamos una incisión por el que soldar la resistencia y de ahí el cable que iría conectado al arduino.

Como todos los sensores tenían en común dos cables, el de masa y el de 5V, decidimos juntarlos todos en el mismo cable para evitar una gran cableado que estorbara mucho. A estas alturas las soldaduras ya no nos salían tan bien debido a la inexperiencia y a que nos quedamos sin cables finos y hubo que reutilizar otros más gruesos que no se prestaban tan bien a una buena soldadura. No quisimos perder mucho tiempo perfeccionándolas, por lo que quedaban churros y pegotes de estaño que disimulamos con cinta aislante usado para evitar contactos entre soldaduras y aunar los cables.

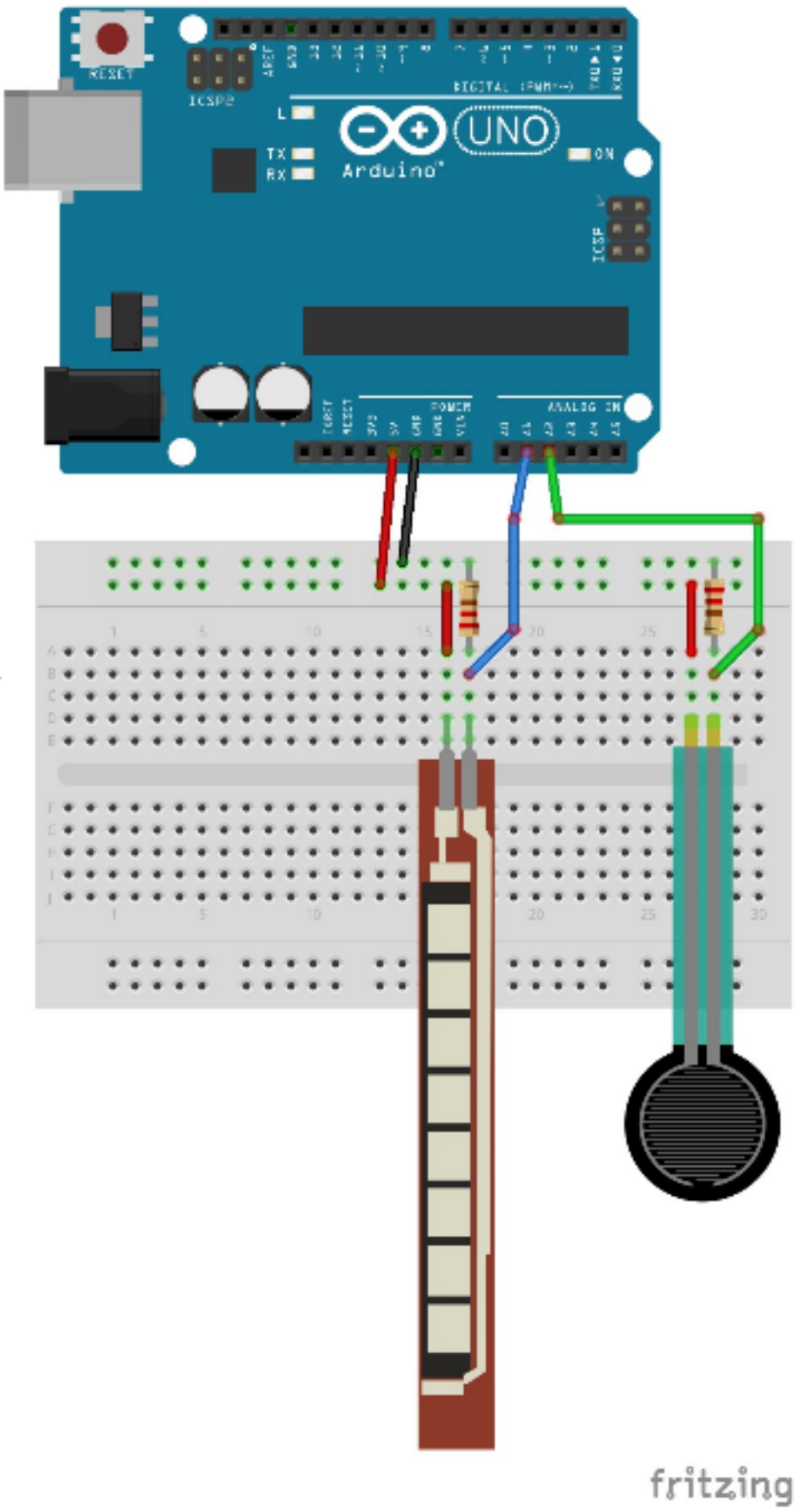

Al terminar las soldaduras, era momento de probarlos en el arduino como hicimos anteriormente con los sensores sin coser ni soldar. Tras unos momentos de estupor ante resultados sin sentido y sin encontrar explicación, revisamos de nuevo el esquema de los sensores y caímos en la cuenta de un detalle.

De uno de los conectores salen dos cables, uno con la resistencia para conectar a masa, y el otro para conectar al arduino. Los cables que iban a masa decidimos soldarlos en un cable común.

El error cometido fue confundirnos de cable y soldar los que iban al arduino, por lo que se trató de arreglar de la siguiente manera.

- El cable con la resistencia que quedó suelto se desoldó
- Los cables que quedaron soldados para ir conectados a masa se cortaron a la mitad y se volvieron a empalmar y soldar con la resistencia por el medio
- El cable desoldado del principio se volvía a soldar al inicio de la resistencia.

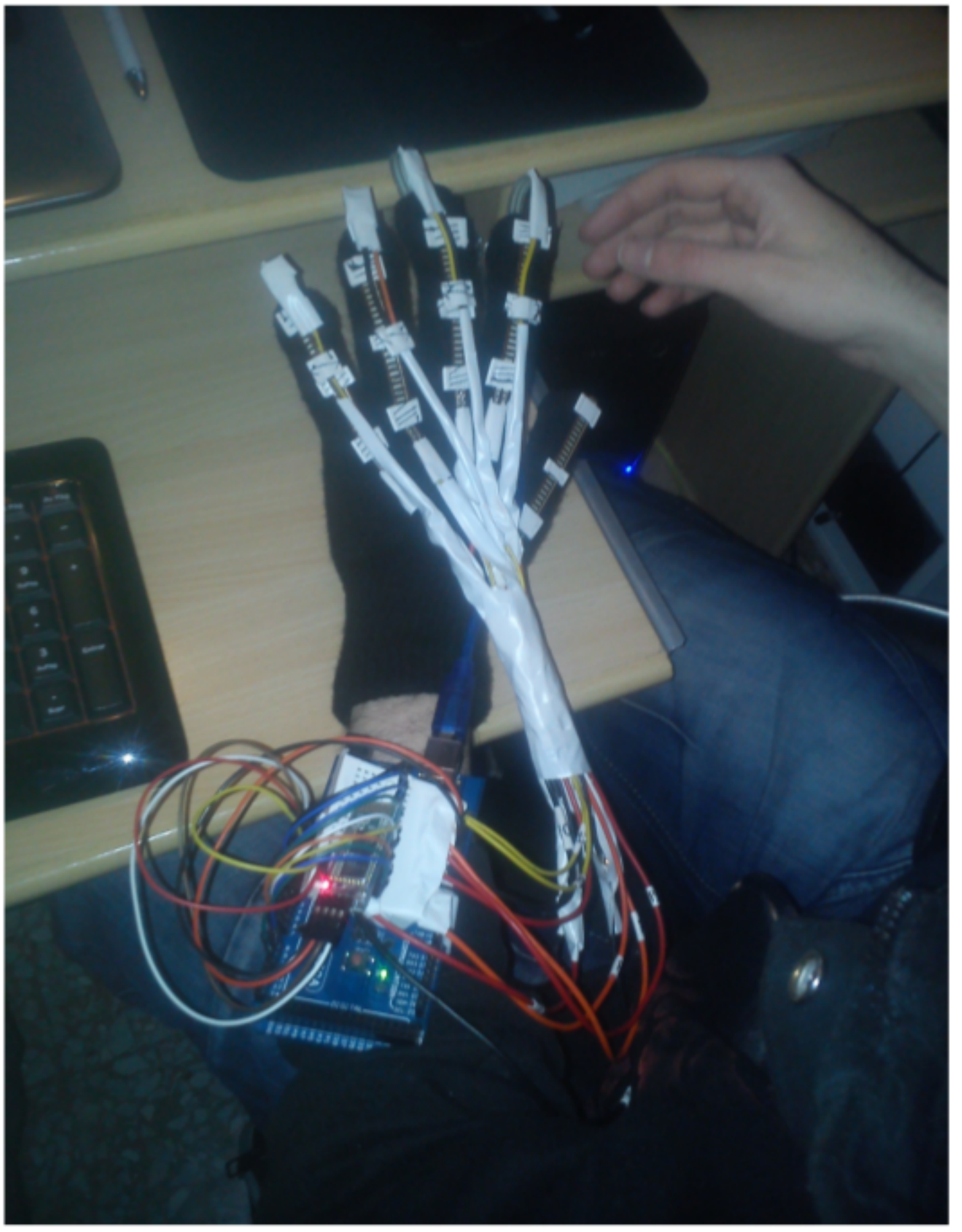
Con ello el problema se pudo resolver sin perder cables ni rehacer todo el trabajo, con el inconveniente de que las medidas calculadas en los cables ya no sirvieron y las longitudes eran dispares y no se ajustaban.



## Tercera fase: La programación de la mano

Una vez el guante preparado y funcional para poder ser programado, nos ayudamos de una protoboard para conectar los cables al arduino. En total nos salían 8 cables a conectar a los pines analógicos y dos cables comunes, uno para los 5V y otro para la masa o GND. Inicialmente en vez de 8 analógicos eran diez, pero pronto vimos que los sensores del pulgar carecían de mucho sentido por lo que decidimos retirarlos.

En nuestro caso, como usamos un arduino personal (Arduino Atmega 2560) que disponía de los pines suficientes, no hubo problemas. De haber sido la Arduino Uno de la que se disponía en clase, hubiera sido necesario echar mano de los multiplexadores para poder conectar todas las entradas.



La programación es sencilla, se adjunta un anexo con la programación en la memoria. En líneas generales, comenzamos con la simple lectura de los valores para determinar el rango en el que se movían y con ello establecer algunos valores a partir de los cuales se consideraba que se había dado un toque flojo o uno fuerte y si se estaba flexionando los dedos.

En el caso de los flexores, como en cada dedo se tenía unos valores distintos por el hecho de tener distintas flexiones, era necesario hacer una lectura inicial para establecer cual eran los valores de reposo y en base a ello detectar las flexiones cuando la diferencia del valor actual con el inicial superaba un rango determinado.

```
const int DEDOS = 4;
//state of the hand
int flexs[DEDOS];
int pressures[DEDOS];

const int FLEXOR = 1;
const int PRESSURE = 2;

//values of the flex sensor
const int minimumSensorFlex = 0;
const int maximumSensorFlex = 45;
//values which will use with the flex sensor
const int minimumFlex = 0;
const int maximumFlex = 100;
//dead zone flexion
const int deadFlexion=25;
int initialPosition[DEDOS];

//values of the pressure sensor
const int minimumSensorPressure = 10;
const int maximumSensorPressure = 40;
//dead zone pressure
const int deadPressure = 50;
//strong pressure
const int strongPressure = 300;

//pins of sensors
int flexSensorPin[DEDOS];
```

```
int pressureSensorPin[FLEXORS];


/**
 * Read a concrete sensor given the type of sensor and the number
 */
int valueOfSensor(int typeSensor, int sensor){
    int valueSensor = 0;
    int valueFinal = 0;

    //segun si estoy mirando flexor o presion
    if(typeSensor == FLEXOR){
        int sensorValue=sensor;
        valueSensor = analogRead(flexSensorPin[sensorValue]);
        valueFinal = map(valueSensor, minimumSensorFlex, maximumSensorFlex, minimumFlex, maximumFlex);

    } else if(typeSensor == PRESSURE){
        int sensorButton=sensor+1;
        valueFinal = analogRead(pressureSensorPin[sensorButton]);
    }

    return valueFinal;
}

/**
 * Read the whole state of the hand (flexion and contact)
 */
void readState(){
    //read the values of the flex's sensors
    for (int i=0; i<DEDOS; i++){
        flexs[i] = valueOfSensor(FLEXOR, i);
    }

    //read the values of the pressure's sensors
    for (int i=0; i<DEDOS; i++){
        pressures[i]=valueOfSensor(PRESSURE, i);
    }
}
```

```
void setup(){
    Serial.begin(9600);
    pressureSensorPin[0]=0;
    pressureSensorPin[1]=1;
    pressureSensorPin[2]=2;
    pressureSensorPin[3]=3;
    pressureSensorPin[4]=4;

    //pins of sensors
    flexSensorPin[0]=6;
    flexSensorPin[1]=7;
    flexSensorPin[2]=8;
    flexSensorPin[3]=9;

    Serial.println("Segundo para setear los flexores");
    delay(1000);
    for (int i=0; i<DEDOS; i++){
        flexs[i] = 0;
        pressures[i] = 0;
        initialPosition[i]=map(analogRead(flexSensorPin[i]),
                        minimumSensorFlex, maximumSensorFlex,
                        minimumFlex, maximumFlex);
    }
    Serial.println("Flexores seteados");
}
```

Una vez teniendo el estado de la mano en cada momento, tocaba  asociar flexiones y toques a acciones. Un problema que surgió es que un sólo toque se detectaba en varias pasadas del bucle principal y por tanto detectaba varios toques. Para poder detectarlo se guardó no solo el estado actual, sino también el estado previo.

Viendo la cantidad de información que se necesitaba para detectar las acciones, se llegó a la decisión de que todo el procesamiento de la información se realizaría en la arduino para así evitar una comunicación excesiva con el bluetooth y evitar que posibles pérdidas o retrasos en la información terminara con un estado de la mano “corrupto”.

Además de controlar el estado previo de la mano, se vigilaba el tiempo, de forma que si un dedo se mantenía pulsado mucho tiempo, se consideraba un toque e igualmente, si no se había presionado y

posteriormente levantado, no se consideraría como toque a no ser que pasara un tiempo determinado por medio de error y prueba.

Por último, el resultado de todo ese procesamiento quedaba guardado en arrays aparte en el que quedaba establecido si había toque o flexión y cuánta flexión.

```
/**
 * Dado el delay de la ultima vez que se proceso y el boton que se quiere mirar
 * se comprueba si ha habido pulsacion, pulsacion fuerte/larga o nada
 * devuelve un int que corresponde a la enumeracion
 */
ClickButton processButton(int delay, int button){
    /* Estados: 0 -> estados anterior y actual no pulsados
     * 1 -> anterior estado pulsado
     * 2 -> estado actual pulsado
     * 3 -> estado anterior y actual pulsado
     */
    int compareStates= ((previousPressures[button]-deadPressure) > 0) ? 1:0;
    compareStates= ((pressures[button]-deadPressure) > 0) ? compareStates+2:compareStates;
    boolean touch=false;
    boolean strong=false;

    //si no ha pasado suficiente tiempo de la anterior pulsacion y no coincide con que haya pulsado y despulsado
    //sumamos el tiempo de espera entre toque y toque y salimos
    if(timeWait[button]<0 && compareStates != 1){
        //sumamos el tiempo transcurrido
        timeWait[button]+=delay;
        return no_click;
    }

    //si hay suficiente fuerza...
    if(pressures[button]>=(strongPressure+deadPressure)){
        strong=true;
        maxPressures[button]=true;
    }

    //boton pulsado antes y despulsado, cuenta como toque
    if( compareStates == 1 ){
```

```
        touch=true;
    }

    //boton se mantiene pulsado
    else if(compareStates == 3 ){
        //sumamos el tiempo transcurrido
        timePressure[button]+=delay;
        //si ha superado el tiempo, cuenta como toque
        if(timePressure[button]>timeResponse){
            touch=true;
        }
    }

    else if(compareStates == 2){
        //acaba de pulsar, no hacemos nada
    }

    //procesamos resultado
    if(touch){
        //seteamos el actual valor como no pulsado para que no se vuelva a tener en cuenta el siguiente frame
        pressures[button]=0;
        //igualmente reseteamos su tiempo transcurrido y le ponemos un tiempo negativo para que no haga varias pulsaciones
        //del mismo toque
        timePressure[button]=0;
        //establecemos el tiempo de espera antes de considerar otro toque del mismo boton
        timeWait[button]=(timeWaitResponse*(-1));
        //miramos si hubo toque fuerte
        strong=maxPressures[button];
        //proceso fuerza y accion
        if(strong){
            //reseteamos la presion fuerte para el siguiente toque
            maxPressures[button]=false;
           return long_click;
        }else{
           return normal_click;
        }
    }
```

```
    return no_click;
}

/**
 * Proceso el estado al completo de la mano y actuo en consecuencia
 */
int processState(int delay){
    //ante el procesado, escribir por pantalla que accion ha realizado, despues
    //refinamos haciendo los envios por bluetooth

    //comprobamos...
    for (int i=0; i<DEDOS; i++){
        //comprobamos botones
        buttons[i]=processButton(delay, i);
        switch(buttons[i]){
          case normal_click:
              Serial.print("Click normal de ");
              Serial.println(i);
              break;
        case long_click:
              Serial.print("Click fuerte de ");
              Serial.println(i);
              break;
        case no_click:
              break;
    }

    //comprobamos flexores
    if((flexs[i]<=(initialPosition[i]-deadFlexion))
          || (flexs[i]>=(initialPosition[i]+deadFlexion))  ){
        flexorActived[i]=true;
        //Serial.print("Flexor ");
        //Serial.print(i);
        //Serial.print(" activado: ");
       //Serial.println(flexs[i]);
    }else{
        flexorActived[i]=false;
```

```
            //Serial.print("Flexor ");
            //Serial.print(i);
            //Serial.print(" desactivado: ");
            //Serial.println(flexs[i]);
        }
    }
}
```

Finalmente, con el estado ya procesado de la mano, establecemos la acción que se ejecuta para mandarlo por bluetooth. Para ello, le damos preferencia a los toques de los dedos. Para no andar mirando una y otra vez el array de los toques para saber si se ha pulsado o no un dedo, lo leemos una vez y convertimos el estado en un sólo numero, en el que cada dedo tiene asignado dos bits.

```
int processButton=0;
int result=0;
//variable con 8 bits, cada dos bits es un dedo
for(int i=0; i<DEDOS; i++){
    processButton=(int)buttons[i]; //enumeracion (click-fuerte, click, no click) convertida a int
    int desplazamiento=2*i;
    int resultDesplazamiento=(processButton<<desplazamiento);
    result= result | resultDesplazamiento;
}
```

Con ello, basta un simple switch y el número correspondiente para establecer los comandos, lo que facilita la tarea y ahorra muchas líneas más adelante.

Nuestra intención era controlar el ordenador con el guante, y decidimos que lo que haría sería controlar el ratón con sus clicks y el teclado, con lo que establecimos unos estados para el guante:

- Modo ratón, en el que los flexores controlarían el movimiento del ratón, algunos toques simularían los clicks y otros cambiarían a otros estados.
- Modo teclado, en el que según el número de toques sencillos determina la letra o símbolo a escribir, al estilo de los móviles antiguos, y los toques fuertes cambiarían de teclado (numérico, mayúsculas, minúsculas y símbolos) o volverían al modo de ratón.
- Modo patrones, donde se establecen más funciones simples y avanzadas como abrir un navegador u otras acciones que se nos pudiera ocurrir. Igualmente se establecería un toque para volver al modo ratón.

Traducido al código, resultaba en una variable donde se guardaba el estado actual, y según el estado, se entraba a un switch que decidía el comando a enviar por bluetooth y si era necesario actualizar el estado o no. Para el caso del modo teclado se añadía además variables para controlar el número de toques de según que dedo y el tiempo transcurrido.

El comando constaba de un simple string decidido arbitrariamente, en el que establecíamos el nombre de la acción, y separados por dos puntos, los parámetros que pudiera necesitar. Por ejemplo:

```
raton:20:30
comando:firefox
tecla:ctrl+c
tecla:d
```

Y similares. Dichos comandos eran enviados directamente por bluetooth hasta el ordenador donde disponíamos de un script en python que iba leyendo del bluetooth y según lo recibido, ejecutaba un comando u otro. Python precisa de instalar la siguiente librería para poder comunicarse con el bluetooth.

```
sudo apt-get install python-serial
```

Para ejecutar y simular el ratón y el teclado, nos ayudamos de un comando de linux, xdotool, que se usaba precisamente para simular ratón y teclado.

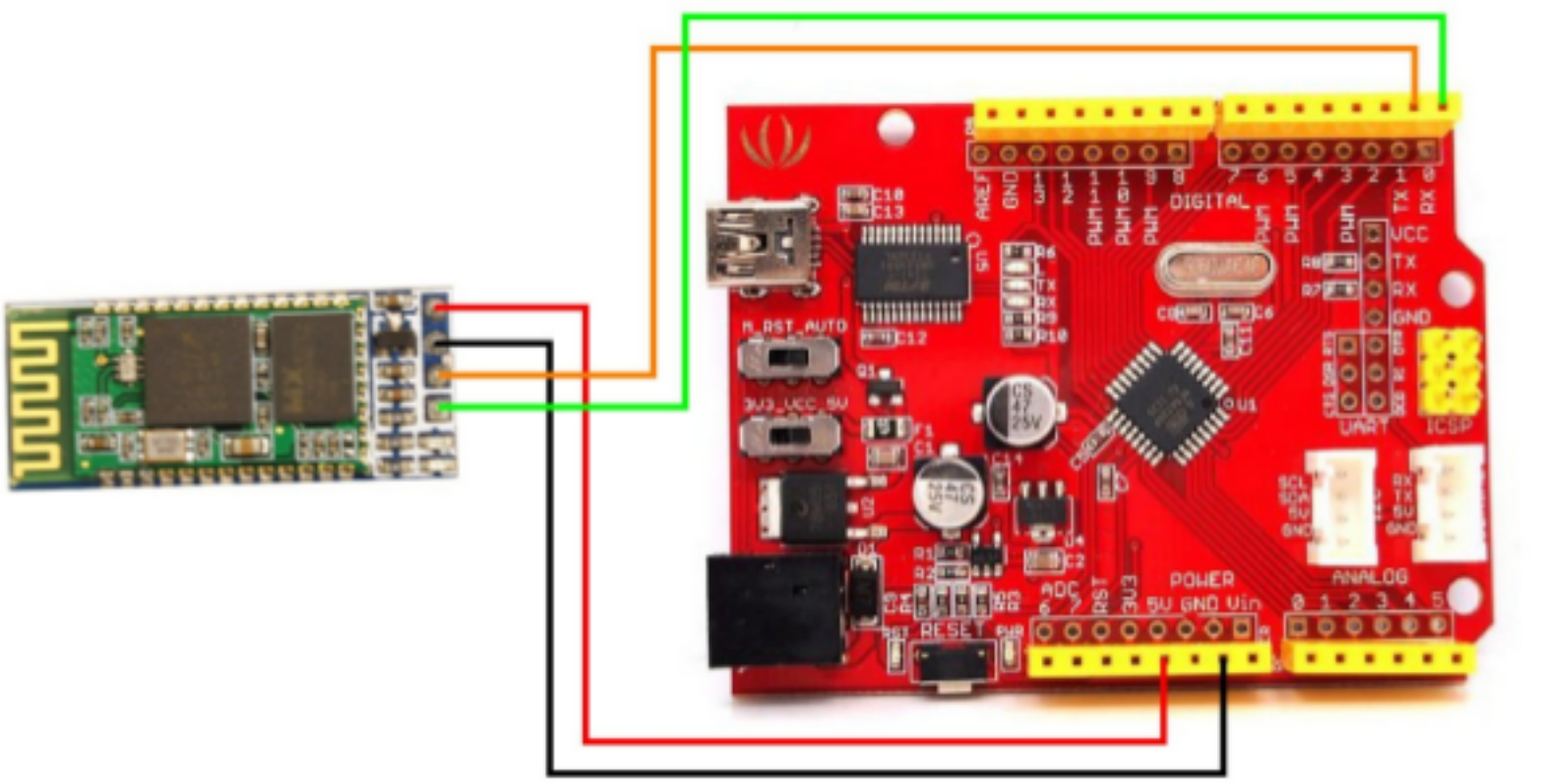
Con ello, podíamos simular prácticamente cualquier comportamiento que se nos ocurriera con el guante.

```
import subprocess, os, string, serial
def mousemove(x, y):
	subprocess.call(["xdotool", "mousemove_relative", "--", str(x), str(y)])


def processCommand(cmd):
	if cmd[0] == "click":
		subprocess.call(["xdotool", "click", "1"])
	elif cmd[0] == "clickderecho":
		subprocess.call(["xdotool", "click", "2"])
	elif cmd[0] == "raton":
		mousemove(cmd[1], cmd[2])
	elif cmd[0] == "tecla":
		subprocess.call(["xdotool", "key", cmd[1]])
```

```
        elif cmd[0] == "apretar":
                subprocess.call(["xdotool", "mousedown"])
        elif cmd[0] == "soltar":
                subprocess.call(["xdotool", "mouseup"])
        elif cmd[0] == "comando":
                subprocess.call([cmd[1]])


bluetoothSerial = serial.Serial("/dev/rfcomm1", baudrate=9600)
bluetoothSerial.write("Conectando...")
while(1):
        rcv = bluetoothSerial.readline()
        rcv=rcv[:-1]
        cmd = rcv.split(":")
        processCommand(cmd)
```

Curiosamente, lo que menos ocupa en el código fuente, es lo que más ha costado programar. El bluetooth usa muy pocas lineas para establecerse y mandar los comandos. Sin embargo, era imprescindible que el ordenador se hubiera pareado con el dispositivo previamente y además, ante cualquier error, se hacía muy difícil debuggear sin saber si era por parte del ordenador/raspberry o por parte del dispositivo o del código en sí.

La conexión del dispositivo bluetooth JY-MCU que teníamos, con el arduino es el siguiente:




Especial atención a que los pines TXD y RXD del dispositivo no tienen que ir al TX ni RX de la arduino, aunque en el esquema anterior está así establecido. Sino que deben ir a otros pines analógicos ya que esos del arduino están reservados para la comunicación con el ordenador.

Una vez establecidos, considerando que nosotros manejábamos la raspberry con raspbian instalado y con un adaptador usb-bluetooth, los siguientes pasos eran necesarios para parear el dispositivo.

Se necesita instalar los siguientes programas:

```
sudo apt-get install bluetooth bluez-utils
```

Para ver nuestro dispositivo bluetooth, el cual devuelve un nombre similar a hci0

```
hciconfig
```

Escaneamos en busca del bluetooth del arduino

```
hcitool scan
```

Si lo encuentra, se verá una dirección mac que usaremos en el siguiente comando

```
sudo bluez-simple-agent hci0 xx:xx:xx:xx:xx:xx
```

Con lo que pregunta por el pin para llevar a cabo el pareado. Dicho pin, por defecto es 1234 en el caso de nuestro dispositivo bluetooth.

Con esto, los dispositivos ya están pareados. Para facilitar todo el proceso de comunicación, editamos el fichero /etc/bluetooth/rfcomm.conf e incluimos las siguientes líneas:

```
rfcomm1 {
        bind yes;
        device xx:xx:xx:xx:xx:xx;
        channel 1;
        comment “Connection to Bluetooth serial module”
}
```

Finalmente, para conectar o reiniciar la comunicación con el dispositivo, usamos el siguiente script que libera el canal de comunicación si estuviera ocupada y establece la comunicación.

```
sudo rfcomm release 1
sudo rfcomm bind rfcomm1
```

## Cuarta fase: Pruebas en la raspberry

Como ya se ha mencionado anteriormente, el ordenador que vamos a controlar para las pruebas es una raspberry, al que le hemos instalado raspbian y que dispone de un adaptador bluetooth por usb. Además de instalarle lo necesario para controlar el bluetooth y las librerías para python además del propio python, instalamos el programa xdotool necesario para el script creado.

No hay grandes menciones mas que puntualizar lo poco adecuado de la raspberry para su uso como ordenador, ya que aunque reaccionara bien, era lento y a veces requería medidas extremas como reiniciar a la fuerza, lo que llevó que en alguna ocasión el sistema se corrompiera y hubiera que reinstalar el sistema operativo.

Por otro lado, algunos de los comandos de xdotool, usado para simular el ratón y teclado, no iban exactamente igual que en otros linux. Se tenía un array con caracteres especiales pero al no ser reconocidos en la raspberry, a modo de parche se tiene un if else que traduce dicho caracteres a los códigos numéricos correspondientes. Ni está completo ni consideramos que sea buena práctica, por lo que es un apartado que tenemos pendiente mejorar.

# Anexo: Instrucciones de la mano

Consideramos que existen dos tipos de toques, el normal y el fuerte, y el siguiente mapeo de dedos y letras que se usará posteriormente:

- Índice → A
- Corazón → B
- Anular → C
- Meñique → D

El guante dispone de tres estados, ratón, tecla y patrones.

## Estado ratón

Al arrancar se arranca en estado de ratón, en el cual los flexores son funcionales y mueven el ratón de la siguiente forma:

- A → movimiento hacia abajo
- B → movimiento hacia arriba
- C → movimiento hacia la izquierda
- D → movimiento hacia la derecha

Los toques seteados son:

- A normal → click
- A fuerte → mantener pulsado
- B normal → click derecho
- B fuerte → soltar click
- C normal → Ctrl+C
- C fuerte → cambio a estado de patrones
- D normal → Ctrl+V
- D fuerte → cambio a estado de teclado

## Estado patrones

Los flexores en este estado no están operativos. Los toques son como sigue:

- A normal → Ctrl+z
- B normal → navegador (el epiphany, que es el instalado en raspbian)
- C fuerte → volver al estado de ratón

## Estado teclado

Los flexores en este estado no están operativos. Los toques fuertes cambian el estado del teclado como sigue:

- A → Teclado de símbolos
- B → Desde otro estado, teclado de minúsculas, desde el teclado de minúsculas, se alterna entre mayúsculas y minúsculas
- C → Teclado de números
- D → Regresa al estado del ratón

Para cada uno de esos estados, los caracteres quedan mapeados como sigue en la siguiente tabla

| A normal | B normal | C normal | D normal |
|---|---|---|---|
| S | O | A | E |
| D | I | N | R |
| U | T | C | L |
| G | B | P | M |
| H | Q | Y | V |
| Ñ | J | Z | F |
| | K | W | X |
| A normal | B normal | C normal | D normal |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 0 | | |
| A normal | B normal | C normal | D normal |
| Espacio | . | , | Enter |
| ! | ? | ( | Borrar |
| ) | ¡ | ¿ | “ |
| : | ; | ' | |

# Anexo: Código fuente del servidor python

```
import subprocess, os, string, serial


def mousemove(x, y):
	subprocess.call(["xdotool", "mousemove_relative", "--", str(x), str(y)])


def processCommand(cmd):
	if cmd[0] == "click":
		subprocess.call(["xdotool", "click", "1"])
	elif cmd[0] == "clickderecho":
		subprocess.call(["xdotool", "click", "2"])
	elif cmd[0] == "raton":
		mousemove(cmd[1], cmd[2])
	elif cmd[0] == "tecla":
		subprocess.call(["xdotool", "key", cmd[1]])
	elif cmd[0] == "apretar":
		subprocess.call(["xdotool", "mousedown"])
	elif cmd[0] == "soltar":
		subprocess.call(["xdotool", "mouseup"])
	elif cmd[0] == "comando":
		subprocess.call([cmd[1]])


bluetoothSerial = serial.Serial("/dev/rfcomm1", baudrate=9600)
bluetoothSerial.write("Conectando...")


while(1):
	rcv = bluetoothSerial.readline()
	rcv=rcv[:-1]
	cmd = rcv.split(":")
	processCommand(cmd)
```

# Anexo: Código fuente del arduino

## Betaglo.ino

```
#include "Enumerations.h"
#include "ProcessCommands.h"
#include <SoftwareSerial.h>

const int DEDOS = 4;
const int FLEXORS = 5;

//state of the hand
int flexs[DEDOS];
int pressures[DEDOS];
// variable que guarda si el dedo presion paso del maximo (toque fuerte)
boolean maxPressures[DEDOS];

//previous state of the hand
int previousFlexs[DEDOS];
int previousPressures[DEDOS];

//processed state of hand
ClickButton buttons[DEDOS];
boolean flexorActived[DEDOS]; //the value of them is stored in flexs (no need to process them)


//values of the flex sensor
const int minimumSensorFlex = 0;
const int maximumSensorFlex = 45;
//values which will use with the flex sensor
const int minimumFlex = 0;
const int maximumFlex = 100;
//dead zone flexion
const int deadFlexion=25;
int initialPosition[DEDOS];

//values of the pressure sensor
const int minimumSensorPressure = 10;
```

```
const int maximumSensorPressure = 40;
//dead zone pressure
const int deadPressure = 50;
//strong pressure
const int strongPressure = 300;

//time of pressing sensor
unsigned long timePressure[DEDOS];
unsigned long timeWait[DEDOS];
unsigned long timer=0;
unsigned long timeResponse=1000;
unsigned long timeWaitResponse=1000;

const int FLEXOR = 1;
const int PRESSURE = 2;

//pins of sensors
int flexSensorPin[DEDOS];
int pressureSensorPin[FLEXORS];


/**
 * Read a concrete sensor given the type of sensor and the number
 */
int valueOfSensor(int typeSensor, int sensor){
    int valueSensor = 0;
    int valueFinal = 0;

    //segun si estoy mirando flexor o presion
    if(typeSensor == FLEXOR){
        int sensorValue=sensor;
        valueSensor = analogRead(flexSensorPin[sensorValue]);
        valueFinal = map(valueSensor, minimumSensorFlex, maximumSensorFlex, minimumFlex, maximumFlex);
    }else if(typeSensor == PRESSURE){
        int sensorButton=sensor+1;
        valueFinal = analogRead(pressureSensorPin[sensorButton]);
    }
    return valueFinal;
```

```
}


/**
 * Read the whole state of the hand (flexion and contact)
 */
void readState(){
    //read the values of the flex's sensors
    for (int i=0; i<DEDOS; i++){
        flexs[i] = valueOfSensor(FLEXOR, i);
    }
  //read the values of the pressure's sensors
    for (int i=0; i<DEDOS; i++){
        pressures[i]=valueOfSensor(PRESSURE, i);
    }
}


/**
 * Dado el delay de la ultima vez que se proceso y el boton que se quiere mirar
 * se comprueba si ha habido pulsacion, pulsacion fuerte/larga o nada
 * devuelve un int que corresponde a la enumeracion
 */
ClickButton processButton(int delay, int button){
    /* Estados: 0 -> estados anterior y actual no pulsados
     * 1 -> anterior estado pulsado
     * 2 -> estado actual pulsado
     * 3 -> estado anterior y actual pulsado
     */
    int compareStates= ((previousPressures[button]-deadPressure) > 0) ? 1:0;
    compareStates= ((pressures[button]-deadPressure) > 0) ? compareStates+2:compareStates;
    boolean touch=false;
    boolean strong=false;


    //si no ha pasado suficiente tiempo de la anterior pulsacion y no coincide con que haya pulsado y despulsado
    //sumamos el tiempo de espera entre toque y toque y salimos
    if(timeWait[button]<0 && compareStates != 1){
        //sumamos el tiempo transcurrido
        timeWait[button]+=delay;
```

```
        return no_click;
    }

    //si hay suficiente fuerza...
    if(pressures[button]>=(strongPressure+deadPressure)){
        strong=true;
        maxPressures[button]=true;
    }

    //boton pulsado antes y despulsado, cuenta como toque
    if( compareStates == 1 ){
        touch=true;
    }

    //boton se mantiene pulsado
    else if(compareStates == 3 ){
        //sumamos el tiempo transcurrido
        timePressure[button]+=delay;
        //si ha superado el tiempo, cuenta como toque
        if(timePressure[button]>timeResponse){
            touch=true;
        }
    }

    else if(compareStates == 2){
            //acaba de pulsar, no hacemos nada
    }

    //procesamos resultado
    if(touch){
        //seteamos el actual valor como no pulsado para que no se vuelva a tener en cuenta el siguiente frame
        pressures[button]=0;
        //igualmente reseteamos su tiempo transcurrido y le ponemos un tiempo negativo para que no haga varias pulsaciones
        //del mismo toque
       timePressure[button]=0;
       //establecemos el tiempo de espera antes de considerar otro toque del mismo boton
       timeWait[button]=(timeWaitResponse*(-1));
```

```
        //miramos si hubo toque fuerte
        strong=maxPressures[button];
        //proceso fuerza y accion
        if(strong){
            //reseteamos la presion fuerte para el siguiente toque
             maxPressures[button]=false;
             return long_click;
         }else{
            return normal_click;
        }
    }
    return no_click;
}

/**
 * Proceso el estado al completo de la mano y actuo en consecuencia
 */
int processState(int delay){
    //ante el procesado, escribir por pantalla que accion ha realizado, despues
  //refinamos haciendo los envios por bluetooth

  //comprobamos...
  for (int i=0; i<DEDOS; i++){
      //comprobamos botones
      buttons[i]=processButton(delay, i);
     switch(buttons[i]){
        case normal_click:
          Serial.print("Click normal de ");
          Serial.println(i);
          break;
        case long_click:
          Serial.print("Click fuerte de ");
          Serial.println(i);
          break;
      case no_click:
         break;
   }
```

```
    //comprobamos flexores
    if((flexs[i]<=(initialPosition[i]-deadFlexion))
        || (flexs[i]>=(initialPosition[i]+deadFlexion))  ){
         flexorActived[i]=true;
        //Serial.print("Flexor ");
        //Serial.print(i);
        //Serial.print(" activado: ");
        //Serial.println(flexs[i]);
    }else{
        flexorActived[i]=false;
        //Serial.print("Flexor ");
        //Serial.print(i);
        //Serial.print(" desactivado: ");
        //Serial.println(flexs[i]);
     }
   }
}

/**
 * Mueve el estado actual a un estado previo para poder leer el siguiente estado
 */
void nextState(){
    for (int i=0; i<DEDOS; i++){
        previousFlexs[i] = flexs[i];
        previousPressures[i] = pressures[i];
    }
}

void setup(){
    Serial.begin(9600);
    setupCommand();

    pressureSensorPin[0]=0;
    pressureSensorPin[1]=1;
    pressureSensorPin[2]=2;
    pressureSensorPin[3]=3;
```

```
  pressureSensorPin[4]=4;


  //pins of sensors
  flexSensorPin[0]=6;
  flexSensorPin[1]=7;
  flexSensorPin[2]=8;
  flexSensorPin[3]=9;


  Serial.println("Segundo para setear los flexores");
  delay(1000);
  for (int i=0; i<DEDOS; i++){
      timePressure[i]=0;
      timeWait[i]=0;
     flexorActived[i]=false;
     flexs[i] = 0;
     maxPressures[i]=false;
     pressures[i] = 0;
     initialPosition[i]=map(analogRead(flexSensorPin[i]),
                         minimumSensorFlex, maximumSensorFlex,
                         minimumFlex, maximumFlex);
     }
     Serial.println("Flexores seteados");


     //copio los valores al previous
     readState();


     timer=millis();
}


void loop(){
     readState();
     processState(millis()-timer);
     processCommands(buttons, flexorActived, flexs, DEDOS);
     timer=millis();
     nextState();
     //delay(300);
}
```

## Enumerations.h

```
#include "Arduino.h"


#ifndef ENUMERATION
#define ENUMERATION
//enum state button/click
enum ClickButton{
    no_click = 0,
    normal_click,
    long_click
};


enum States {
    mouse,
    keyboard,
    patterns
};


enum StatesKeyboard {
    minusculas,
    mayusculas,
    numeros,
    simbolos
};
#endif
```

## ProcessCommands.cpp

```
#include "Arduino.h"
#include "ProcessCommands.h"
#include "Enumerations.h"


SoftwareSerial mySerial(13,12); // RX, TX
States stateHand;
unsigned long timerMouse;
StatesKeyboard stateKeyboard;
unsigned long timerKeys;
int finger;
```

```
int timesFinger;
unsigned long previousTimerKeys;
int limitTimeFinger=500;
char keyboardLettersMayus[][7]={
    { 'S','D','U','G','H','Ñ' },
    {'O','I','T','B','Q','J','K' },
    {'A','N','C','P','Y','Z','W' },
    {'E','R','L','M','V','F','X' }
};

char numbers[][7]={
    { '1','5','9' },
    { '2','6','0' },
    { '3','7' },
    { '4','8' }
};

char keyboardLettersMinus[][7]={
    { 's','d','u','g','h','ñ' },
    { 'o','i','t','b','q','j','k' },
    { 'a','n','c','p','y','z','w' },
    { 'e','r','l','m','v','f','x' }
};

char symbols[][7]={
    { ' ', '!', ')', ':' },
    { '.', '?', '¡', ';' },
    { ',', '(', '¿', '\" },
  { '\n','\b',"'" }
};

//por cada dedo(4), el numero de caracteres que tiene para saber que modulo hacer,
//ordenados segun la enumeracion
int numberChars[][4]={
    { 6,7,7,7 },
    { 6,7,7,7 },
    { 3,3,2,2 },
```

```
    { 4,4,4,3 }
};

void setupCommand(){
    mySerial.begin(9600);
    stateHand=mouse;
    timerMouse=0;
    finger=-1;
    timesFinger=-1;
    timerKeys=-1;
    stateKeyboard=minusculas;
}

/*
 * Condicion imprescindible que la cadena termine con \n
 */
void sendBluetooth(String value){
    Serial.println(value);
    // Length (with one extra character for the null terminator and \n)
    int strLen = value.length() + 2;

    // Prepare the character array (the buffer)
    char charArray[strLen];

    // Copy it over
    value.toCharArray(charArray, strLen);
    charArray[strLen-2]='\n';
    charArray[strLen-1]='\0';

    if (mySerial.available()) {
      mySerial.write(charArray);
    }
}

void sendKey(){
    String key="tecla:";
    char charToAppend;
```

```
if(timerKeys!=-1 && timesFinger!=-1){
    switch(stateKeyboard){
      case minusculas:
          charToAppend=keyboardLettersMinus[finger][timesFinger];
          break;
      case mayusculas:
          charToAppend=keyboardLettersMayus[finger][timesFinger];
          break;
      case simbolos:
          charToAppend=symbols[finger][timesFinger];
          //vigilo caracteres especiales dificiles
          if(charToAppend=='\n'){
              key=key+"36";
           }else if(charToAppend=='\b'){ //borrar
              key=key+"22";
          }else if(charToAppend==' '){
             key=key+"space";
          }else if(charToAppend=='.'){
            key=key+"60";
          }else if(charToAppend=='!'){
            key=key+"50+10";
          }else if(charToAppend=='!'){
            key=key+"50+61";
          }else{
            key=key+charToAppend;
          }
          break;
      case numeros:
          charToAppend=numbers[finger][timesFinger];
          break;
    }
    if(stateKeyboard!=simbolos){
       key=key+charToAppend;
    }
}
timerKeys=-1;
timesFinger=-1;
```

```
    sendBluetooth(key);
}

void processButtonMouse(int value){
    switch(value){
      //click normal del indice
      case 64:
           sendBluetooth("click");
           break;

      //click fuerte del indice
      case 128:
         sendBluetooth("apretar");
         break;

      //click normal corazon
    case 16:
         sendBluetooth("clickderecho");
         break;

     //click fuerte corazon
     case 32:
          sendBluetooth("soltar");
          break;

    //anular normal
    case 4:
        //control C
        sendBluetooth("tecla:ctrl+c");
        break;

     //click fuerte anular
   case 8:
       Serial.println("changed state patterns");
      stateHand=patterns;
      break;
```

```
      //click normal meñique
      case 1:
           //control v
           sendBluetooth("tecla:ctrl+v");
          break;

      //click fuerte meñique
    case 2:
        Serial.println("changed state keyboard");
        stateHand=keyboard;
        stateKeyboard=minusculas;
         break;
    }
}

void processButtonKeyboard(int value){
    //lo primero mirar si queda por escribir alguna letra por espera del tiempo
    if(timerKeys!=-1 && (millis()-timerKeys)>limitTimeFinger){
        sendKey();
    }

int numberFinger=-1;
switch(value){
   //click fuerte meñique
   case 2:
       Serial.println("changed state mouse");
       stateHand=mouse;
       break;
   //anular fuerte
   case 8:
       Serial.println("changed state numbers");
       stateKeyboard=numeros;
       break;
   //dedo corazon fuerte
   case 32:
       if(stateKeyboard==minusculas){
           Serial.println("changed state mayusculas");
```

```
            stateKeyboard=mayusculas;
        }else{
            Serial.println("changed state minusculas");
            stateKeyboard=minusculas;
        }
        break;
  //click fuerte indice
 case 128:
     Serial.println("changed state simbolos");
     stateKeyboard=simbolos;
     break;

//indice
case 64:
     numberFinger=0;
    break;
 //corazon
case 16:
     numberFinger=1;
     break;
   //anular
   case 4:
      numberFinger=2;
      break;
    //meñique
   case 1:
       numberFinger=3;
       break;
}

if(numberFinger!=-1){
    if(finger!=numberFinger){
        sendKey();
        finger=numberFinger;
    }
  timesFinger=(timesFinger+1)%(numberChars[0][finger]);
  timerKeys=millis();
```

```
    }
}

void processButtonPatterns(int value){
    switch(value){
       //click normal del indice
       case 64:
            sendBluetooth("tecla:ctrl+z");
           break;

       //click fuerte del indice
      case 128:
            break;

       //click normal corazon
       case 16:
            sendBluetooth("comando:epiphany");
            break;

//click fuerte corazon
case 32:
    break;

//anular normal
case 4:
    break;

//click fuerte anular
case 8:
    Serial.println("changed state initial");
    stateHand=mouse;
    break;

//click normal meñique
case 1:
    break;
```

```
      //click fuerte meñique
      case 2:
          break;
    }
}


//guardo una lista de variables que me permite saber el estado actual del guante
//es decir, por ejemplo estoy actualmente controlando el raton. Un toque largo del meñique cambiaria al estado de teclado
//por lo que los botones pasarian a hacer otras acciones
void processCommands(ClickButton buttons[],boolean flexorActived[], int flexor[], int DEDOS){
    int processButton=0;
    int result=0;
    //variable con 8 bits, cada dos bits es un dedo
    for(int i=0; i<DEDOS; i++){
        processButton=(int)buttons[i]; //enumeracion convertida a int
        int desplazamiento=2*i;
         int resultDesplazamiento=(processButton<<desplazamiento);
         result= result | resultDesplazamiento;
    }


//preferencia a los botones para procesar el estado
if(stateHand==mouse){
     processButtonMouse(result);
} else if(stateHand==keyboard){
    processButtonKeyboard(result);
}  else if(pattems){
     processButtonPattems(result);
}



if(stateHand==mouse){
     //procesamos los flexores cuando estamos en modo raton
       for(int i=0; i<DEDOS; i++){
           int delay=millis()-timerMouse;
           if(flexorActived[i]==true && delay>50){
            int value=flexor[i];
           value = map(value, 0, 100, 1, 20);
```

```
String command="raton:";
switch(i){
  case 0:
    command=command+value;
    command=command+":0";
    sendBluetooth(command);
    break;
 case 1:
    value=value*(-1);
    command=command+value;
    command=command+":0";
   sendBluetooth(command);
   break;
case 2:
   value=value*(-1);
   command=command+"0:";
   command=command+value;
   sendBluetooth(command);
   break;
            case 3:
               command=command+"0:";
               command=command+value;
               sendBluetooth(command);
              break;
            }
            timerMouse=millis();
        }
      }
    }
}
```

## ProcessCommands.h

```
#include "Arduino.h"
#include "Enumerations.h"
#include <SoftwareSerial.h>


void setupCommand();
void sendBluetooth(String value);


//lista de comandos a realizar segun las activaciones realizadas
//tengo que pasarle el array de botones, el array de flexores y el array de los valores de los flexores
void processCommands(ClickButton buttons[],boolean flexorActived[], int flexor[],  int DEDOS);
```

# Bibliografía


http://www.instructables.com/id/DIY-Robotic-Hand-Controlled-by-a-Glove-and-Arduino/
http://www.instructables.com/id/Arduino-Wireless-Animatronic-Hand/
http://bildr.org/2012/11/flex-sensor-arduino/
http://en.wikipedia.org/wiki/Voltage_divider
http://stackoverflow.com/questions/89228/calling-an-external-command-in-python
http://blog.dawnrobotics.co.uk/2013/11/talking-to-a-bluetooth-serial-module-with-a-raspberry-pi/
http://www.correlatedcontent.com/blog/bluetooth-keyboard-on-the-raspberry-pi/
http://sourceforge.net/p/bluez/mailman/message/18444725/
http://www.heatxsink.com/entry/how-to-pair-a-bluetooth-device-from-command-line-on-linux
http://www.raspberrypi.org/forums/viewtopic.php?p=521067