ANEXOS

ANEXO I. CÓDIGO

En este anexo se presenta el código fuente del Firmware de Instrumentación en C++/Processing (Arduino), el código fuente en scripts de Matlab para la identificación de gestos estáticos y dinámicos, y el código de la aplicación con interfaz NUI en Python.

- FIRMWARE DE INSTRUMENTACIÓN:

```
/* FIRMWARE DE INSTRUMENTACIÓN*/
/* SOURCE CODE BY VICTOR MALUMBRES TALLES*/
/* SDA ==> A4*/
/* SCL ==> A5*/
/* INCLUDES AND DEFINES*/

#include <Wire.h> /* Librería para manejo de comunicación I2C.*/
#include <SoftwareSerial.h> /* Librería para manejo de puerto Serial Virtual */

#define    MPU9250_ADDRESS          0x68   /*SI AD0 ==> +3.3V, MPU9250_ADDRESS 0x69*/
#define    MAG_ADDRESS              0x0C
 #define    GYRO_FULL_SCALE_250_DPS    0x00
#define    GYRO_FULL_SCALE_500_DPS    0x08
#define    GYRO_FULL_SCALE_1000_DPS   0x10
#define    GYRO_FULL_SCALE_2000_DPS   0x18
 #define    ACC_FULL_SCALE_2_G       0x00
#define    ACC_FULL_SCALE_4_G       0x08
#define    ACC_FULL_SCALE_8_G       0x10
#define    ACC_FULL_SCALE_16_G      0x18
/* ACEL CONFIGURATION REGISTER*/
#define    ACC_CONFIGURATION_1        28
#define    ACC_CONFIGURATION_2        29
#define    ACC_LOWPASSFILTER_5Hz      0x06
/* GYRO CONFIGURATION REGISTER */
#define    GYRO_CONFIGURATION_1       27
#define    GYRO_CONFIGURATION_2       26
```

```
#define    GYRO_LOWPASSFILTER_5Hz      0x06


/* OTHER REGISTERS */
#define MPU9250_ACCEL_XOUT_H         0x3B
#define MPU9250_ACCEL_XOUT_L         0x3C
#define MPU9250_ACCEL_YOUT_H         0x3D
#define MPU9250_ACCEL_YOUT_L         0x3E
#define MPU9250_ACCEL_ZOUT_H         0x3F
#define MPU9250_ACCEL_ZOUT_L         0x40
#define MPU9250_TEMP_OUT_H           0x41
#define MPU9250_TEMP_OUT_L           0x42
#define MPU9250_GYRO_XOUT_H          0x43
#define MPU9250_GYRO_XOUT_L          0x44
#define MPU9250_GYRO_YOUT_H          0x45
#define MPU9250_GYRO_YOUT_L          0x46
#define MPU9250_GYRO_ZOUT_H          0x47
#define MPU9250_GYRO_ZOUT_L          0x48
#define MPU9250_MAG_XOUT_H           0x03
#define MPU9250_MAG_XOUT_L           0x04
#define MPU9250_MAG_YOUT_H           0x05
#define MPU9250_MAG_YOUT_L           0x06
#define MPU9250_MAG_ZOUT_H           0x07
#define MPU9250_MAG_ZOUT_L           0x08
/* OTHER DEFINES */
#define pi 3.1415
#define Tms 10 /* en ms */
#define Ts 0.01 /* en s */


/* GLOBAL VARIABLES */
double ax,ay,az;
double axf,ayf,azf;
double ax_1,ay_1,az_1;
double offax,offay,offaz;
double gx,gy,gz;
double gxf,gyf,gzf;
double gx_1,gy_1,gz_1;
```

```
double offgx,offgy,offgz;
double mx,my,mz;
int offset0=0;
float flex0=0;
int offset1=0;
float flex1=0;
int offset2=0;
float flex2=0;
int offset3=0;
float flex3=0;
int offset4=0;
float flex4=0;
int k=0;

/* functions */
double media(double vector[],int s){
  double suma=0;
  for(int a=0;a<s;a++){
   suma=vector[a]+suma;
   }
  suma=suma/s;
  return suma;
 }
bool writeRegister(const uint8_t register_addr, const uint8_t value) {
   //send write call to sensor address
   //send register address to sensor
   //send value to register
   bool write_status = 0;
   Wire.beginTransmission(MAG_ADDRESS); //open communication with
   Wire.write(register_addr);
   Wire.write(value);
   Wire.endTransmission();

   return write_status; //returns whether the write succeeded or failed
}
```

```
uint8_t readRegister(const uint8_t register_addr) {
   uint8_t data = 0;
   Wire.beginTransmission(MPU9250_ADDRESS);
   Wire.write(register_addr);
   Wire.endTransmission();
   Wire.requestFrom(MPU9250_ADDRESS, 1);
   while(Wire.available()) {
      data = Wire.read();    // receive a byte as character
   }
   return data; //return the data returned from the register
}

int16_t readRegisters(const uint8_t msb_register, const uint8_t lsb_register) {
   uint8_t msb = readRegister(msb_register);
   uint8_t lsb = readRegister(lsb_register);
   uint16_t value=((((uint16_t)msb) << 8) | lsb); /*ENTERO SIN SIGNO, RANGO [0,65535]*/
   int16_t v=-(int16_t)value;
   return v;

}
int16_t readMagRegisters(const uint8_t msb_register, const uint8_t lsb_register) {
   uint8_t msb = readMagRegister(msb_register);
   uint8_t lsb = readMagRegister(lsb_register);
   uint16_t value=((((uint16_t)msb) << 8) | lsb); /*ENTERO SIN SIGNO, RANGO [0,65535]*/
   int16_t v=-(int16_t)value;
   return v;
}
int16_t get_AX_INT(void) {
    return  readRegisters(MPU9250_ACCEL_XOUT_H, MPU9250_ACCEL_XOUT_L);
}
```

```
int16_t get_AY_INT(void) {
    return  readRegisters(MPU9250_ACCEL_YOUT_H, MPU9250_ACCEL_YOUT_L);
}
int16_t get_AZ_INT(void) {
    return  readRegisters(MPU9250_ACCEL_ZOUT_H, MPU9250_ACCEL_ZOUT_L);
}
int16_t get_GX_INT(void) {
    return  readRegisters(MPU9250_GYRO_XOUT_H, MPU9250_GYRO_XOUT_L);
}
int16_t get_GY_INT(void) {
    return  readRegisters(MPU9250_GYRO_YOUT_H, MPU9250_GYRO_YOUT_L);
}
int16_t get_GZ_INT(void) {
    return  readRegisters(MPU9250_GYRO_ZOUT_H, MPU9250_GYRO_ZOUT_L);
}
int16_t get_MX_INT(void) {
    return  readMagRegisters(MPU9250_MAG_XOUT_H, MPU9250_MAG_XOUT_L);
}
int16_t get_MY_INT(void) {
    return  readMagRegisters(MPU9250_MAG_YOUT_H, MPU9250_MAG_YOUT_L);
}
int16_t get_MZ_INT(void) {
    return  readMagRegisters(MPU9250_MAG_ZOUT_H, MPU9250_MAG_ZOUT_L);
}


void acel_calibracion(double offx,double offy,double offz,int num_muestras){
   /* calibracion cuando dispositivo esta orientado en z -1g*/
   double sum_x=0;
   double sum_y=0;
   double sum_z=0;
   for(int i=0;i<num_muestras;i++){
    int16_t int_ax=get_AX_INT();
    int16_t int_ay=get_AY_INT();
    int16_t int_az=get_AZ_INT();
    double d_ax=((double)int_ax)*(1/16384.0);
    double d_ay=((double)int_ay)*(1/16384.0);
```

```
    double d_az=((double)int_az)*(1/16384.0)+1;
    sum_x+=d_ax;
    sum_y+=d_ay;
    sum_z+=d_az;
   }
   offx=sum_x/num_muestras;
   offy=sum_y/num_muestras;
   offz=sum_z/num_muestras;

 }

void gyro_calibracion(double offx,double offy,double offz,int num_muestras){
   /* calibracion cuando dispositivo esta orientado en z -1g*/
   double sum_x=0;
   double sum_y=0;
   double sum_z=0;
   for(int i=0;i<num_muestras;i++){
    int16_t int_gx=get_GX_INT();
    int16_t int_gy=get_GY_INT();
    int16_t int_gz=get_GZ_INT();
    double d_gx=-((double)int_gx/32.8);
    double d_gy=-((double)int_gy/32.8);
    double d_gz=-((double)int_gz/32.8);
    sum_x+=d_gx;
    sum_y+=d_gy;
    sum_z+=d_gz;
   }
   offx=sum_x/num_muestras;
   offy=sum_y/num_muestras;
   offz=sum_z/num_muestras;

 }

void setup() {
  pinMode(OUTPUT,13);
  pinMode(OUTPUT,8);
```

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

  /* Offset de los flexores */
  offset0=analogRead(0);
  offset1=analogRead(1);
  offset2=analogRead(2);
  offset3=analogRead(3);
  offset4=analogRead(4);

  // Configurar acelerometro
  writeRegister(ACC_CONFIGURATION_1 , ACC_FULL_SCALE_2_G);
  writeRegister(ACC_CONFIGURATION_2, ACC_LOWPASSFILTER_5Hz);
  // Configurar giroscopio
  writeRegister(GYRO_CONFIGURATION_1,GYRO_FULL_SCALE_1000_DPS);
  writeRegister(GYRO_CONFIGURATION_2,GYRO_LOWPASSFILTER_5Hz);

  /* Calibración acelererómetro */
  acel_calibracion(offax,offay,offaz,1024);
  gyro_calibracion(offgx,offgy,offgz,1024);
}
void loop() {
 /* Flex. Data*/
 int f0=analogRead(0);
 int f1=analogRead(1);
 int f2=analogRead(2);
 int f3=analogRead(3);
 //int f4=analogRead(4);

 flex0=((float)f0) - 300.0;
 flex0*=(-100.0/(offset0-300));
 flex0+=100; /* % */
 if(flex0<0) flex0=0;
 if(flex0>100) flex0=100;

 flex1=((float)f1) - 300.0;
```

```
flex1*=(-100.0/(offset1-300));
flex1+=100; /* % */
if(flex1<0) flex1=0;
if(flex1>100) flex1=100;

flex2=((float)f2) - 300.0;
flex2*=(-100.0/(offset2-300));
flex2+=100; /* % */
if(flex2<0) flex2=0;
if(flex2>100) flex2=100;

flex3=((float)f3) - 300.0;
flex3*=(-100.0/(offset3-300));
flex3+=100; /* % */
if(flex3<0) flex3=0;
if(flex3>100) flex3=100;

//flex4=((float)f4) - 300.0;
//flex4*=(-100.0/(offset4-300));
//flex4+=100; /* % */
//if(flex4<0) flex4=0;
//if(flex4>100) flex4=100;
flex4=0;



/* Accel. Data */

int16_t axd=get_AX_INT();
int16_t ayd=get_AY_INT();
int16_t azd=get_AZ_INT();
ax=axd*(1/16384.0)-offax;
ay=ayd*(1/16384.0)-offay;
az=azd*(1/16384.0)-offaz;

if((ax>-0.03)&&(ax<0.03)) ax=0;
if((ay>-0.03)&&(ay<0.03)) ay=0;
```

```
if((az>-0.03)&&(az<0.03)) az=0;


/* Gyro. Data */


int16_t gxd=get_GX_INT();
int16_t gyd=get_GY_INT();
int16_t gzd=get_GZ_INT();
gx=-((float)gxd/32.8) - offgx;
gy=-((float)gyd/32.8) - offgy;
gz=-((float)gzd/32.8) - offgz;




 Serial.print("ax");
 Serial.print(ax);
 Serial.print("ay");
 Serial.print(ay);
 Serial.print("az");
 Serial.print(az);
 Serial.print("gx");
 Serial.print(gx);
 Serial.print("gy");
 Serial.print(gy);
 Serial.print("gz");
 Serial.print(gz);
 Serial.print("f0");
 Serial.print(flex0);
 Serial.print("f1");
 Serial.print(flex1);
 Serial.print("f2");
 Serial.print(flex2);
 Serial.print("f3");
 Serial.print(flex3);
 Serial.print("f4");
 Serial.print(flex4);
 Serial.println();
```

```
 //delay(10);
}
```

- SCRIPTS DE MATLAB

1. Reconocimiento de gestos estáticos

```
% Static Gesture Recognition

clear all;
close all;
delete(instrfind);

% Serial COM Configuration
BaudRate=9600;
arduino=serial('COM3','BaudRate',BaudRate);
fopen(arduino);

% Load Hand Gesture Package
HandGestureLibrary

% Variables
    Ts=0.01; % Tsample[s]
    a=[0;0;0]; % {ax;ay,;az} with gravity effect [m/s2]
    ai=[0;0;0]; %{aix;aiy,;aiz} without gravity effect [m/s2]
    aik_1=[0;0;0];
    v=[0;0;0]; % {vx;vy;vz} [m/s]
    vk_1=[0;0;0];
    p=[0;0;0]; %[m]
    pk_1=[0;0;0];
    g=[0;0;0]; %{gx;gy;gz} [°/s]
    A=0.85; % Simple Kalman Filter Constant
    rpy=[0;0;0];%{roll(tx),pitch(ty),yaw(tz)} Hand Orientation
    rpy_a=[0;0;0]; % Acelerometer´s Orientation
    rpy_g=[0;0;0]; % Gyroscope´s Orientation
    R=eye(3); % Rotation Matrix

% Main

while(1)
tic

%% Get Serial Data
str=fgetl(arduino);
data=read_data(str);

%% Acel,Gyro & Flexors data
a=[9.8*data(1);9.8*data(2);9.8*data(3)]; %  a[m/s2]
g=[data(4);data(5);data(6)]; % g[°/s]
f=[data(7);data(8);data(9);data(10);data(11)]; % flexion[%]
gest_data=[a;f];
```

```
%% Calculate Orientation
rpy_g=rpy_g+g*Ts; % °
rpy_a=[atan2(a(2),(sqrt(a(1)*a(1)+a(3)*a(3))));atan2(a(1),(sqrt(a(2)*a
(2)+a(3)*a(3))));atan2((sqrt(a(2)*a(2)+a(1)*a(1))),a(3))]; % rad
rpy_a=rpy_a*(180/pi); % °

% Simple Kalman Filter
rpy=A*rpy_a+(1-A)*rpy_g;
rpy_g=rpy; % Update Gyro`s Orientation to reduce acumulate/integral
error.

%% Create Rotation Matrix & Eliminate Gravity Effect
R=rpy2rotmatrix(rpy(1)*(pi/180),rpy(2)*(pi/180),rpy(3)*(pi/180));
ai=R'*a+[0;0;9.8];

%% Ajustes
if similar(a(3),-9.8,0.5)
    rpy(1)=0;
    rpy(2)=0;
end
if similar(a(1),0,0.5)
    a(1)=0;
end
if similar(a(2),0,0.5)
    a(2)=0;
end
if similar(a(3),0,0.5)
    a(3)=0;
end

%% Recognition
gest_str='none';

if compare(gest_data,static_up,0.5,20)
    gest_str='up';
end
if compare(gest_data,static_down,0.5,20)
    gest_str='down';
end
if compare(gest_data,static_tick,0.5,20)
    gest_str='tick';
end
if compare(gest_data,static_left,0.5,20)
    gest_str='left';
end
if compare(gest_data,static_right,0.5,20)
    gest_str='right';
end
if compare(gest_data,static_fist,0.5,20)
    gest_str='fist';
end
if compare(gest_data,static_stop,0.5,20)
    gest_str='stop';
end

disp(gest_str);

% Wait to end of Ts
```

```
T=toc;
 while(T<Ts)
    T=toc;
 end

end

% End of Serial COM ·& Clean Workspace
fclose(arduino);
clear all
```

2. Reconocimiento de gestos dinámicos

```
% Dynamic Gesture Recognition

clear all;
close all;
delete(instrfind);

% Serial COM Configuration
BaudRate=9600;
arduino=serial('COM3','BaudRate',BaudRate);
fopen(arduino);

% Load Hand Gesture Package
HandGestureLibrary

% Variables
    Ts=0.01; % Tsample[s]
    a=[0;0;0]; % {ax;ay,;az} with gravity effect [m/s2]
    ai=[0;0;0]; %{aix;aiy,;aiz} without gravity effect [m/s2]
    aik_1=[0;0;0];
    v=[0;0;0]; % {vx;vy;vz} [m/s]
    vk_1=[0;0;0];
    p=[500;500;0]; %[m]
    pk_1=[0;0;0];
    g=[0;0;0]; %{gx;gy;gz} [°/s]
    A=0.85; % Simple Kalman Filter Constant
    rpy=[0;0;0];%{roll(tx),pitch(ty),yaw(tz)} Hand Orientation
    rpy_a=[0;0;0]; % Acelerometer´s Orientation
    rpy_g=[0;0;0]; % Gyroscope´s Orientation
    R=eye(3); % Rotation Matrix
    px=[500;500]; % {px_x;px_y}
    pdraw=[500;500]; % {px_x;px_y}
    a_buffer=[]; % acceleration buffer for hand gesture recognition
    d_vector=[]; % distance vector
    aux=0;

 % Other Variables (Maq.Estados)
    state='none';
    gesture='none';
    start_recognition=0;
    end_recognition=0;
```

```
% Main

while(1)
tic

%% Get Serial Data
str=fgetl(arduino);
data=read_data(str);

%% Acel,Gyro & Flexors data
a=[9.8*data(1);9.8*data(2);9.8*data(3)]; %  a[m/s2]
g=[data(4);data(5);data(6)]; % g[°/s]
f=[data(7);data(8);data(9);data(10);data(11)]; % flexion[%]

%% Calculate Orientation
rpy_g=rpy_g+g*Ts; % °
rpy_a=[atan2(a(2),(sqrt(a(1)*a(1)+a(3)*a(3))));atan2(a(1),(sqrt(a(2)*a
(2)+a(3)*a(3))));atan2((sqrt(a(2)*a(2)+a(1)*a(1))),a(3))]; % rad
rpy_a=rpy_a*(180/pi); % °

% Simple Kalman Filter
rpy=A*rpy_a+(1-A)*rpy_g;
rpy_g=rpy; % Update Gyro`s Orientation to reduce acumulate/integral
error.

%% Create Rotation Matrix & Eliminate Gravity Effect
R=rpy2rotmatrix(rpy(1)*(pi/180),rpy(2)*(pi/180),rpy(3)*(pi/180));
ai=R'*a+[0;0;9.8];

%% Ajustes
if similar(a(3),-9.8,0.5)
    rpy(1)=0;
    rpy(2)=0;
end
if similar(a(1),0,0.5)
    a(1)=0;
end
if similar(a(2),0,0.5)
    a(2)=0;
end
if similar(a(3),0,0.5)
    a(3)=0;
end


%% States Machine

% Machine Inputs
if similar(a(2),-9.8,0.3)
 start_recognition=1;
 end_recognition=0;
elseif similar(a(3),-9.8,0.3)
 start_recognition=0;
 end_recognition=1;
end
```

```
% States
switch state
    case 'none'
        if start_recognition==1
            state='hand_gesture';
        end
        disp('none')

    case 'hand_gesture'

        if aux==0
            pause(0.5);
            aux=1;
        end
        disp('hand_gesture')
        a_buffer=[a_buffer,a];

        if end_recognition==1
            % Acel.Buffer Processing
            mx=mean(a_buffer(1,:));
            my=mean(a_buffer(2,:));
            mz=mean(a_buffer(3,:));
            ax=a_buffer(1,:)-mx;
            ay=a_buffer(2,:)-my;
            az=a_buffer(3,:)-mz;
            a_buffer=[ax;ay;az];
            % Calculate distance to every gesture of library.
            d_up=distance(dinamic_up,a_buffer)
            d_down=distance(dinamic_down,a_buffer)
            d_right=distance(dinamic_right,a_buffer)
            d_left=distance(dinamic_left,a_buffer)
            d_circle=distance(dinamic_circle,a_buffer)
            % Extract de min. distance
            d_vector=[d_up,d_down,d_right,d_left,d_circle];
            sol=minimum(d_vector);
            dmin=sol(1);
            pos=sol(2);
            % What hand gesture is?
            gesture=pos2gesture(pos);
            disp(dmin);
            disp(gesture);
            aux=0;
            a_buffer=[0;0;0];
            d_vector=[0,0,0,0,0];
            state='none';
            pause;

        end
end


% Wait to end of Ts
T=toc;
 while(T<Ts)
    T=toc;
 end

end
```

```
% End of Serial COM ·& Clean Workspace
fclose(arduino);
clear all
```

3. Otros scripts

- HandGestureLibrary.m

```
%  Hand Gesture Library Package

%% Dynamic Gesture
% up
up=load('C:\Users\Victor\Desktop\CURSO4\TFG\TFG\Scripts_MATLAB\LIBRERI
A_DE_GESTOS\a_upz');
dinamic_up=up.sg;
% down
down=load('C:\Users\Victor\Desktop\CURSO4\TFG\TFG\Scripts_MATLAB\LIBRE
RIA_DE_GESTOS\a_downz');
dinamic_down=down.sg;
% left
left=load('C:\Users\Victor\Desktop\CURSO4\TFG\TFG\Scripts_MATLAB\LIBRE
RIA_DE_GESTOS\a_left');
dinamic_left=left.sg;
% right
right=load('C:\Users\Victor\Desktop\CURSO4\TFG\TFG\Scripts_MATLAB\LIBR
ERIA_DE_GESTOS\a_right');
dinamic_right=right.sg;
% circle
circle=load('C:\Users\Victor\Desktop\CURSO4\TFG\TFG\Scripts_MATLAB\LIB
RERIA_DE_GESTOS\a_circle');
dinamic_circle=circle.sg;
% cross

%% Static
static_tick=[0;-9.8;0;0;100;100;100;0];
static_stop=[0;-9.8;0;0;0;0;0;0];
static_up=[9.8;0;0;100;100;100;100;0];
static_down=[-9.8;0;0;100;100;100;100;0];
static_left=[0;0;-9.8;100;100;100;100;0];
static_right=[0;0;9.8;100;100;100;100;0];
static_fist=[0;-9.8;0;100;100;100;100;0];
```

- read_data.m

```
function data=read_data(st)
    ax_pos=0;
    ay_pos=0;
    az_pos=0;
    gx_pos=0;
    gy_pos=0;
    gz_pos=0;
```

```
f0_pos=0;
f1_pos=0;
f2_pos=0;
f3_pos=0;
f4_pos=0;

%  tomar tamaño del array (b)
[a b]=size(st);

% tomar las posiciones donde char es letra ('ax','ay' ...)
for i=1:1:b

    if i~=b
     n=i+1;
     st2=st(i:n);
    end

    if strcmp(st2,'ax')
        ax_pos=i;
    end
    if strcmp(st2,'ay')
        ay_pos=i;
    end
    if strcmp(st2,'az')
        az_pos=i;
    end
    if strcmp(st2,'gx')
        gx_pos=i;
    end
    if strcmp(st2,'gy')
        gy_pos=i;
    end
    if strcmp(st2,'gz')
        gz_pos=i;
    end
    if strcmp(st2,'f0')
        f0_pos=i;
    end
    if strcmp(st2,'f1')
        f1_pos=i;
    end
    if strcmp(st2,'f2')
        f2_pos=i;
    end
    if strcmp(st2,'f3')
        f3_pos=i;
    end
    if strcmp(st2,'f4')
        f4_pos=i;
    end
end

% tomar los datos numéricos
c=ax_pos+2;
d=ay_pos-1;
st1=st(c:d);
ax=str2double(st1);
```

```
    c=ay_pos+2;
    d=az_pos-1;
    st1=st(c:d);
    ay=str2double(st1);

    c=az_pos+2;
    d=gx_pos-1;
    st1=st(c:d);
    az=str2double(st1);

    c=gx_pos+2;
    d=gy_pos-1;
    st1=st(c:d);
    gx=str2double(st1);

    c=gy_pos+2;
    d=gz_pos-1;
    st1=st(c:d);
    gy=str2double(st1);

    c=gz_pos+2;
    d=f0_pos-1;
    st1=st(c:d);
    gz=str2double(st1);

    c=f0_pos+2;
    d=f1_pos-1;
    st1=st(c:d);
    f0=str2double(st1);

    c=f1_pos+2;
    d=f2_pos-1;
    st1=st(c:d);
    f1=str2double(st1);

    c=f2_pos+2;
    d=f3_pos-1;
    st1=st(c:d);
    f2=str2double(st1);

    c=f3_pos+2;
    d=f4_pos-1;
    st1=st(c:d);
    f3=str2double(st1);

    c=f4_pos+2;
    d=b-2;
    st1=st(c:d);
    f4=str2double(st1);

    % recoger datos en matriz data
    data=[ax;ay;az;gx;gy;gz;f0;f1;f2;f3;f4];

end
```

- similar.m

```
function ok=similar(value,c_value,err)
    error=value-c_value;
    abs_error=abs(error);

    if abs_error <= err
        ok=1;
    else
        ok=0;
    end
end
```

- rpy2rotmatrix.m

```
function R=rpy2rotmatrix(r,p,y)
R(1,1)=cos(y)*cos(p);
R(1,2)=cos(y)*sin(p)*sin(r)-sin(y)*cos(r);
R(1,3)=cos(y)*sin(p)*cos(r)+sin(y)*sin(r);
R(2,1)=sin(y)*cos(p);
R(2,2)=sin(y)*sin(p)*sin(r)+cos(y)*cos(r);
R(2,3)=sin(y)*sin(p)*cos(r)-cos(y)*sin(r);
R(3,1)=-sin(p);
R(3,2)=cos(p)*sin(r);
R(3,3)=cos(p)*cos(r);
end
```

- compare.m

```
function bool=compare(g1, g2, g_err, f_err)
 %% Función para comparar gestos estáticos
 % data ==> [ax;ay;az;f0;f1;f2;f3;f4]
 % g1 ==> gesto 1 data
 % g2 ==> gesto 2 data
 % g_err ==> error acción de la gravedad
 % f_err ==> error flexión articular
 %% code
 bool=0;
 if similar(g1(1),g2(1),g_err)
     b1=1;
 else
     b1=0;
 end
 if similar(g1(2),g2(2),g_err)
     b2=1;
 else
     b2=0;
 end
 if similar(g1(3),g2(3),g_err)
     b3=1;
 else
     b3=0;
 end
```

```
 if similar(g1(4),g2(4),f_err)
     b4=1;
 else
     b4=0;
 end
 if similar(g1(5),g2(5),f_err)
     b5=1;
 else
     b5=0;
 end
 if similar(g1(6),g2(6),f_err)
     b6=1;
 else
     b6=0;
 end
 if similar(g1(7),g2(7),f_err)
     b7=1;
 else
     b7=0;
 end
 if similar(g1(8),g2(8),f_err)
     b8=1;
 else
     b8=0;
 end

 if b1 && b2 && b3 && b4 && b5 && b6 && b7 && b8
     bool=1;
 else
     bool=0;
 end

end
```

- dtw.m

```
function [Dist,D,k,w]=dtw(t,r)
%Dynamic Time Warping Algorithm
%Dist is unnormalized distance between t and r
%D is the accumulated distance matrix
%k is the normalizing factor
%w is the optimal path
%t is the vector you are testing against
%r is the vector you are testing
[rows,N]=size(t);
[rows,M]=size(r);
%for n=1:N
%     for m=1:M
%         d(n,m)=(t(n)-r(m))^2;
%     end
%end
d=(repmat(t(:),1,M)-repmat(r(:)',N,1)).^2; %this replaces the nested
for loops from above Thanks Georg Schmitz

D=zeros(size(d));
D(1,1)=d(1,1);
```

```
for n=2:N
    D(n,1)=d(n,1)+D(n-1,1);
end
for m=2:M
    D(1,m)=d(1,m)+D(1,m-1);
end
for n=2:N
    for m=2:M
        D(n,m)=d(n,m)+min([D(n-1,m),D(n-1,m-1),D(n,m-1)]);
    end
end

Dist=D(N,M);
n=N;
m=M;
k=1;
w=[];
w(1,:)=[N,M];
while ((n+m)~=2)
    if (n-1)==0
        m=m-1;
    elseif (m-1)==0
        n=n-1;
    else
      [values,number]=min([D(n-1,m),D(n,m-1),D(n-1,m-1)]);
      switch number
      case 1
        n=n-1;
      case 2
        m=m-1;
      case 3
        n=n-1;
        m=m-1;
      end
  end
    k=k+1;
    w=cat(1,w,[n,m]);
end
```

- distance.m

```
function d=distance(Base,Data)
% Calculate distance between series
% Using DTW algorithm

[dx,Dx,kx,wx]=dtw(Base(1,:),Data(1,:));
[dy,Dy,ky,wy]=dtw(Base(2,:),Data(2,:));
[dz,Dz,kz,wz]=dtw(Base(3,:),Data(3,:));
d_vector=[dx;dy;dz];
d=sqrt(dx*dx + dy*dy + dz*dz);
end
```

- minimum.m

```
function solution=minimum(vector)
    solution=[0,0];
    dmin=1e6;
    pos=0;

    for i=1:1:length(vector)
        if vector(i)<dmin
            dmin=vector(i);
            pos=i;
        end
    end

    solution=[dmin,pos];

end
```

- pos2gesture.m

```
function gesture=pos2gesture(pos)
    gesture='ERR';

    switch pos
        case 1
            gesture='up';
        case 2
            gesture='down';
        case 3
            gesture='right';
        case 4
            gesture='left';
        case 5
            gesture='circle';
        otherwise
            gesture='none';
    end
end
```

- Draw_Magnitudes.m

```
% DRAWING

% acceleration
figure(1)
title('Aceleraciones Medidas (F+Fg)')
subplot(3,1,1)
plot(t_plot,a_plot(1,:))
xlabel('time')
ylabel('ax [m/s2]')
subplot(3,1,2)
plot(t_plot,a_plot(2,:))
xlabel('time')
ylabel('ay [m/s2]')
subplot(3,1,3)
plot(t_plot,a_plot(3,:))
xlabel('time')
ylabel('az [m/s2]')

%acceleration i
figure(2)
title('Aceleraciones por movimiento (F)')
subplot(3,1,1)
plot(t_plot,ai_plot(1,:))%,t_plot,sg(1,:))
xlabel('time')
ylabel('aix [m/s2]')
subplot(3,1,2)
plot(t_plot,ai_plot(2,:))%,t_plot,sg(2,:))
xlabel('time')
ylabel('aiy [m/s2]')
subplot(3,1,3)
plot(t_plot,ai_plot(3,:))%,t_plot,sg(3,:))
xlabel('time')
ylabel('aiz [m/s2]')

%velocity
figure(5)
subplot(3,1,1)
plot(t_plot,g_plot(1,:))
xlabel('time')
ylabel('gx [°/s]')
subplot(3,1,2)
plot(t_plot,g_plot(2,:))
xlabel('time')
ylabel('gy [°/s]')
subplot(3,1,3)
plot(t_plot,g_plot(3,:))
xlabel('time')
ylabel('gz [°/s]')

%rpy
figure(6)
title('Orientación')
subplot(3,1,1)
plot(t_plot,rpy_plot(1,:))
xlabel('time')
ylabel('roll [°]')
```

```
subplot(3,1,2)
plot(t_plot,rpy_plot(2,:))
xlabel('time')
ylabel('pitch [°]')
subplot(3,1,3)
plot(t_plot,rpy_plot(3,:))
xlabel('time')
ylabel('yaw [°]')

% velocity
figure(7)
title('velocity')
subplot(2,1,1)
plot(t_plot,v_plot(1,:))
xlabel('time')
ylabel('vx [m/s]')
subplot(2,1,2)
plot(t_plot,v_plot(2,:))
xlabel('time')
ylabel('vy [m/s]')

% position
figure(8)
title('position')
subplot(3,1,1)
plot(t_plot,p_plot(1,:))
xlabel('time')
ylabel('px [m]')
subplot(3,1,2)
plot(t_plot,p_plot(2,:))
xlabel('time')
ylabel('py [m]')
subplot(3,1,3)
plot(p_plot(1,:),p_plot(2,:))
xlabel('time')
ylabel('draw')
```

- SCRIPTS DE PYTHON

1. Main.py

```
import vlc
import sys
import math
import gesture
import time
import serial
import grafo
from numpy import array, zeros, argmin, inf, ndim


# Functions

# similar values
def similar(a, b, error):
    bit = False
    resta = a - b
    abs_resta = math.fabs(resta)
    if abs_resta <= error:
        bit = True
    else:
        bit = False
    return bit

# compare two strings
def str_comp(str1, str2):
 i = 0
 str1_len = len(str1)
 str2_len = len(str2)
 if str1_len == str2_len:
  while i < str1_len:
   if str1[i] == str2[i]:
    equal = True
   else:
    equal = False
   i = i + 1
 else:
  equal = False
 return equal

# gesture comp
def gesture_comp (g1, g2, error):
  Simil_bool = False

  a1_list = g1.get_a_list()
  f1_list = g1.get_f_list()
  a2_list = g2.get_a_list()
  f2_list = g2.get_f_list()
  c1 = similar(a1_list[0], a2_list[0], error)
  c2 = similar(a1_list[1], a2_list[1], error)
  c3 = similar(a1_list[2], a2_list[2], error)
  c4 = similar(f1_list[0], f2_list[0], 20)
  c5 = similar(f1_list[1], f2_list[1], 20)
  c6 = similar(f1_list[2], f2_list[2], 20)
```

```
  c7 = similar(f1_list[3], f2_list[3], 20)
  c8 = similar(f1_list[4], f2_list[4], 20)

  if c1 and c2 and c3 and c4 and c5 and c6 and c7 and c8:
   Simil_bool = True
  else:
   Simil_bool = False

  return Simil_bool



arduino = serial.Serial('COM6', 9600, parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE, bytesize=serial.EIGHTBITS, timeout=0)

# VLC Media Config.

# playlist
track01 = 'C:/Users/Victor/Desktop/PlaylistTFG/hitthelights.mp3'
track02 = 'C:/Users/Victor/Desktop/PlaylistTFG/whiplash.mp3'
track03 = 'C:/Users/Victor/Desktop/PlaylistTFG/anestesia.mp3'
track04 = 'C:/Users/Victor/Desktop/PlaylistTFG/jumpinthefire.mp3'
track05 = 'C:/Users/Victor/Desktop/PlaylistTFG/noremorse.mp3'
track06 = 'C:/Users/Victor/Desktop/PlaylistTFG/thefourhorseman.mp3'
track07 = 'C:/Users/Victor/Desktop/PlaylistTFG/seekanddestroy.mp3'
# more tracks¿?
playlist = [track01, track02, track03, track04, track05, track06,
track07]
index_playlist = 0
len_playlist = 7
volume_level = 0
max_level = 100
min_level = 0

# VLC Media Configuration
vlcInstance = vlc.Instance()
player = vlcInstance.media_player_new()
player.set_mrl(playlist[index_playlist])


# gesture_library
"""
    data:
    aceleracion = [ax, ay, az]
    flexores = [indice, corazon, anular, meñique, pulgar]
"""
# 1. Pause ==>  Gesto Stop de la Biblioteca de gestos estáticos
acel = [0, -1.0, 0]
flex = [0, 0, 0, 0, 0]
pause_gest = gesture.Gesture(acel, flex, 'pause')

# 2. Play ==>  Gesto Tick de la Biblioteca de gestos estáticos
acel = [0, -1.0, 0]
flex = [0, 100, 100, 100, 0]
play_gest = gesture.Gesture(acel, flex, 'play')

# 3. Stop

# 4. Volume up ==> Gesto Up de la Biblioteca de gestos estáticos
```

```
acel = [1.0, 0, 0]
flex = [100, 100, 100, 100, 0]
up_gest = gesture.Gesture(acel, flex, 'up')

# 5. Volume down ==> Gesto Down de la Biblioteca de gestos estáticos
acel = [-1.0, 0, 0]
flex = [100, 100, 100, 100, 0]
down_gest = gesture.Gesture(acel, flex, 'down')

# 6. Mute==> Gesto Fist de la Biblioteca de gestos estáticos
acel = [0, -1.0, 0]
flex = [100, 100, 100, 100, 0]
mute_gest = gesture.Gesture(acel, flex, 'mute')

# 7. Next track ==> Gesto Right de la Biblioteca de gestos estáticos
acel = [0, 0, 1.0]
flex = [100, 100, 100, 100, 0]
next_gest = gesture.Gesture(acel, flex, 'next')

# 8. Prev track ==> Gesto Left de la Biblioteca de gestos estáticos
acel = [0, 0, -1.0]
flex = [100, 100, 100, 100, 0]
prev_gest = gesture.Gesture(acel, flex, 'prev')

# 9. Exit
acel = [0, 0, -1.0]
flex = [0, 100, 100, 0, 0]
exit_gest = gesture.Gesture(acel, flex, 'exit')


# Machine of states
rep = grafo.Grafo('play') # rep ==> {play, pause, stop}
track = grafo.Grafo('none') # track ==> {none, next, prev}
volume = grafo.Grafo('none') # volume ==> {none, up, down, mute}

# calibration wait
# global variables
time.sleep(1)
str = ''
line = []
a_data = []
f_data = []
run = True

player.play()
volume_level = player.audio_get_volume()
print(volume_level)


while run:

 for c in arduino.read():
  # take character until '/n'
  a = chr(c)
  line.append(a)
  if c == 10:  # '/n' == 10
   str1 = ''.join(line)
   print(str1)
   # processing ==> take acceleration and flexors data
```

```
  ax_pos = str1.find('ax')
  ay_pos = str1.find('ay')
  az_pos = str1.find('az')
  gx_pos = str1.find('gx')
  f1_pos = str1.find('f0')
  f2_pos = str1.find('f1')
  f3_pos = str1.find('f2')
  f4_pos = str1.find('f3')
  f5_pos = str1.find('f4')
  # take data
  l = len(str1)
  if l>=65 and l<=85: # Filter for Erroneous Serial Lines
   if ax_pos != -1:
    ax_str = str1[ax_pos+2:ay_pos]
    ax_f = float(ax_str)
    a_data.insert(0, ax_f)
   else:
    a_data.insert(0, 0)
   if ay_pos != -1:
    ay_str = str1[ay_pos+2:az_pos]
    ay_f = float(ay_str)
    a_data.insert(1, ay_f)
   else:
    a_data.insert(1, 0)
   if az_pos != -1:
    az_str = str1[az_pos+2:gx_pos]
    az_f = float(az_str)
    a_data.insert(2, az_f)
   else:
    a_data.insert(2, 0)
   print(a_data)

   if f1_pos != -1:
    f1_str = str1[f1_pos + 2:f2_pos]
    f1_f = float(f1_str)
    f_data.insert(0, f1_f)
   else:
    f_data.insert(0, 0)
   if f2_pos != -1:
    f2_str = str1[f2_pos + 2:f3_pos]
    f2_f = float(f2_str)
    f_data.insert(1, f2_f)
   else:
    f_data.insert(1, 0)
   if f3_pos != -1:
     f3_str = str1[f3_pos + 2:f4_pos]
     f3_f = float(f3_str)
     f_data.insert(2, f3_f)
   else:
    f_data.insert(2, 0)
   if f4_pos != -1:
     f4_str = str1[f4_pos + 2:f5_pos]
     f4_f = float(f4_str)
     f_data.insert(3, f4_f)
   else:
    f_data.insert(3, 0)
   if f5_pos != -1:
     f5_str = str1[f5_pos + 2:len(str1)-1]
     f5_f = float(f5_str)
```

```
      f_data.insert(4, f5_f)
    else:
     f_data.insert(4, 0)
    print(f_data)

    # take gesture
    gt = gesture.Gesture(a_data, f_data, 'taken')

    simil = gesture_comp(gt, pause_gest, 0.5)
    if simil == True:
     pause_str = pause_gest.get_gest()
     gt.set_gest(pause_str)

    simil = gesture_comp(gt, play_gest, 0.5)
    if simil == True:
     play_str = play_gest.get_gest()
     gt.set_gest(play_str)

    simil = gesture_comp(gt, up_gest, 0.5)
    if simil == True:
        up_str = up_gest.get_gest()
        gt.set_gest(up_str)

    simil = gesture_comp(gt, down_gest, 0.5)
    if simil == True:
        down_str = down_gest.get_gest()
        gt.set_gest(down_str)

    simil = gesture_comp(gt, mute_gest, 0.5)
    if simil == True:
        mute_str = mute_gest.get_gest()
        gt.set_gest(mute_str)

    simil = gesture_comp(gt, next_gest, 0.5)
    if simil == True:
        next_str = next_gest.get_gest()
        gt.set_gest(next_str)

    simil = gesture_comp(gt, prev_gest, 0.5)
    if simil == True:
        prev_str = prev_gest.get_gest()
        gt.set_gest(prev_str)

    simil = gesture_comp(gt, exit_gest, 0.5)
    if simil == True:
            exit_str = prev_gest.get_gest()
            gt.set_gest(exit_str)


    # execute machine of states of media reproduction

    # rep
    if rep.get_state() == 'play':
     if gt.get_gest() == 'pause':
      rep.set_state('pause')
      player.pause()
     elif gt.get_gest() == 'stop':
      rep.set_state('stop')
      player.stop()
```

```
     else:
      rep.set_state('play')
    elif rep.get_state() == 'pause':
     if gt.get_gest() == 'play':
      rep.set_state('play')
      player.play()
     elif gt.get_gest() == 'stop':
      rep.set_state('stop')
      player.stop()
     else:
      rep.set_state('pause')
    elif rep.get_state() == 'stop':
     if gt.get_gest() == 'play':
      rep.set_state('play')
      player.play()
     else:
      rep.set_state('stop')

    # track
    if track.get_state() == 'none':
        if gt.get_gest() == 'next':
            track.set_state('next')
            index_playlist += 1
            if index_playlist > (len_playlist-1):
                index_playlist = (len_playlist-1)
            player.stop()
            player.set_mrl(playlist[index_playlist])
            player.play()
            #next track
        elif gt.get_gest() == 'prev':
            track.set_state('prev')
            track.set_state('prev')
            index_playlist -= 1
            if index_playlist < 0:
                index_playlist = 0
            player.stop()
            player.set_mrl(playlist[index_playlist])
            player.play()
            #rev track
    elif track.get_state() == 'next':
        if gt.get_gest() == 'prev':
            track.set_state('prev')
            index_playlist -= 1
            if index_playlist < 0:
                index_playlist = 0
            player.stop()
            player.set_mrl(playlist[index_playlist])
            player.play()

        elif gt.get_gest() != 'next':
            track.set_state('none')
    elif track.get_state() == 'prev':
        if gt.get_gest() == 'next':
            track.set_state('next')
            index_playlist += 1
            if index_playlist > (len_playlist - 1):
                index_playlist = (len_playlist - 1)
            player.stop()
            player.set_mrl(playlist[index_playlist])
```

```
            player.play()

        elif gt.get_gest() != 'prev':
            track.set_state('none')

    # volume
    if volume.get_state() == 'none':
        if gt.get_gest() == 'up':
            volume.set_state('up')
            volume_level = volume_level + 1
            if volume_level > max_level:
                volume_level = max_level
            player.audio_set_volume(volume_level)

        elif gt.get_gest() == 'down':
            volume.set_state('down')
            volume_level = volume_level - 1
            if volume_level < min_level:
                volume_level = min_level
            player.audio_set_volume(volume_level)

        elif gt.get_gest() == 'mute':
            volume.set_state('mute')
            player.audio_set_volume(0)
    elif volume.get_state() == 'up':
        volume_level = volume_level + 1
        if volume_level > max_level:
            volume_level = max_level
        player.audio_set_volume(volume_level)
        if gt.get_gest() == 'down':
            volume.set_state('down')
            volume_level = volume_level - 1
            if volume_level < min_level:
                volume_level = min_level
            player.audio_set_volume(volume_level)

        elif gt.get_gest() == 'mute':
            volume.set_state('mute')
            player.audio_set_volume(0)
        elif gt.get_gest() != 'up':
            volume.set_state('none')
    elif volume.get_state() == 'down':
        volume_level = volume_level - 1
        if volume_level < min_level:
            volume_level = min_level
        player.audio_set_volume(volume_level)
        if gt.get_gest() == 'up':
            volume.set_state('up')
            volume_level = volume_level + 1
            if volume_level > max_level:
                volume_level = max_level
            player.audio_set_volume(volume_level)

        elif gt.get_gest() == 'mute':
            volume.set_state('mute')
            player.audio_set_volume(0)
        elif gt.get_gest() != 'down':
            volume.set_state('none')
    elif volume.get_state() == 'mute':
```

```
        if gt.get_gest() == 'up':
            volume.set_state('up')
            volume_level = volume_level + 1
            if volume_level > max_level:
                volume_level = max_level
            player.audio_set_volume(volume_level)
        elif gt.get_gest() == 'down':
            volume.set_state('down')
            volume_level = volume_level - 1
            if volume_level < min_level:
                volume_level = min_level
            player.audio_set_volume(volume_level)
        elif gt.get_gest() != 'mute':
            volume.set_state('none')
            player.audio_set_volume(volume_level)

    print(rep.get_state())
    print(track.get_state())
    print(volume.get_state())
    print(volume_level)
    if gt.get_gest() == 'exit':
        run = False



    # clean variables
    str1 = ''
    line = []
    a_data = []
    f_data = []
```

2. Class Grafo.py

```
""" Machine of States Class"""

class Grafo:

    def __init__(self, str_state):
        self.state = str_state

    # get state
    def get_state(self):
        return self.state

    # change state
    def set_state(self, str_state):
        self.state = str_state
```

3. Class Gesture.py

```
""" Hand Gesture Class"""

class Gesture:

    # Gesture Class Constructor
    def __init__(self, a_data, f_data, str_gesture):
        self.strgest = str_gesture
        self.ax = a_data[0]
        self.ay = a_data[1]
        self.az = a_data[2]
        self.f1 = f_data[0]
        self.f2 = f_data[1]
        self.f3 = f_data[2]
        self.f4 = f_data[3]
        self.f5 = f_data[4]
    # Get string
    def get_gest(self):
        return self.strgest

    # Set string
    def set_gest(self, str_gest):
        self.strgest = str_gest

    # Get acceleration data
    def get_a_list(self):
        adata = []
        adata.insert(0, self.ax)
        adata.insert(1, self.ay)
        adata.insert(2, self.az)
        return adata

    # Get flexions data
    def get_f_list(self):
        fdata = []
        fdata.insert(0, self.f1)
        fdata.insert(1, self.f2)
        fdata.insert(2, self.f3)
        fdata.insert(3, self.f4)
        fdata.insert(4, self.f5)
        return fdata
```

4. Vlc.py

Debido a la gran extensión que supone el script vlc.py se presentan un conjunto de enlaces que muestran en un repositorio el código fuente del núcleo de VLC Media y la documentación relacionada para su uso.

Código:

https://pypi.org/project/python-vlc/

https://wiki.videolan.org/python_bindings

(Los dos enlaces son válidos. Únicamente se diferencian en que el primero direcciona a la web de Python y el otro a un wiki de VLC)

Documentación:

https://www.olivieraubert.net/vlc/python-ctypes/doc/

ANEXO II. DATASHEETS

En este anexo se presentan extractos de datasheets que han tenido gran importancia durante las fases de diseño electrónico del proyecto.

- **MPU-9250.**

  - Introducción

2.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-9250 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec and integrated 16-bit ADCs
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.2mA
- Sleep mode current: 8µA
- Factory calibrated sensitivity scale factor
- Self-test

2.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-9250 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g* and integrated 16-bit ADCs
- Accelerometer normal operating current: 450µA
- Low power accelerometer mode current: 8.4µA at 0.98Hz, 19.8µA at 31.25Hz
- Sleep mode current: 8µA
- User-programmable interrupts
- Wake-on-motion interrupt for low power operation of applications processor
- Self-test

Extracto 1. Features

- Características técnicas del acelerómetro.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Full-Scale Range	AFS_SEL=0		±2		*g*
	AFS_SEL=1		±4		*g*
	AFS_SEL=2		±8		*g*
	AFS_SEL=3		±16		*g*
ADC Word Length	Output in two's complement format		16		bits
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*
	AFS_SEL=1		8,192		LSB/*g*
	AFS_SEL=2		4,096		LSB/*g*
	AFS_SEL=3		2,048		LSB/*g*
Initial Tolerance	Component-Level		±3		%
Sensitivity Change vs. Temperature	-40°C to +85°C AFS_SEL=0 Component-level		±0.026		%/°C
Nonlinearity	Best Fit Straight Line		±0.5		%
Cross-Axis Sensitivity			±2		%
Zero-G Initial Calibration Tolerance	Component-level, X,Y		±60		mg
	Component-level, Z		±80		mg
Zero-G Level Change vs. Temperature	-40°C to +85°C		±1.5		mg/°C
Noise Power Spectral Density	Low noise mode		300		µg/√Hz
Total RMS Noise	DLPFCFG=2 (94Hz)			8	mg-rms
Low Pass Filter Response	Programmable Range	5		260	Hz
Intelligence Function Increment			4		mg/LSB
Accelerometer Startup Time	From Sleep mode		20		ms
	From Cold Start, 1ms V_{DD} ramp		30		ms
Output Data Rate	Low power (duty-cycled)	0.24		500	Hz
	Duty-cycled, over temp		±15		%
	Low noise (active)	4		4000	Hz

Extracto 2. Specifications Accel.

- Características técnicas del giróscopo.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Full-Scale Range	FS_SEL=0		±250		°/s
	FS_SEL=1		±500		°/s
	FS_SEL=2		±1000		°/s
	FS_SEL=3		±2000		°/s
Gyroscope ADC Word Length			16		bits
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)
	FS_SEL=1		65.5		LSB/(°/s)
	FS_SEL=2		32.8		LSB/(°/s)
	FS_SEL=3		16.4		LSB/(°/s)
Sensitivity Scale Factor Tolerance	25°C		±3		%
Sensitivity Scale Factor Variation Over Temperature	-40°C to +85°C		±4		%
Nonlinearity	Best fit straight line; 25°C		±0.1		%
Cross-Axis Sensitivity			±2		%
Initial ZRO Tolerance	25°C		±5		°/s
ZRO Variation Over Temperature	-40°C to +85°C		±30		°/s
Total RMS Noise	DLPFCFG=2 (92 Hz)		0.1		°/s-rms
Rate Noise Spectral Density			0.01		°/s/√Hz
Gyroscope Mechanical Frequencies		25	27	29	KHz
Low Pass Filter Response	Programmable Range	5		250	Hz
Gyroscope Startup Time	From Sleep mode		35		ms
Output Data Rate	Programmable, Normal mode	4		8000	Hz

Extracto 3. Specifications Gyro.

- Registros para configuración del acelerómetro.

1C	28	ACCEL_CONFIG	R/W	ax_st_en	ay_st_en	az_st_en	ACCEL_FS_SEL[1:0]	-	
1D	29	ACCEL_CONFIG 2	R/W	-				ACCEL_FCHOICE_B	A_DLPF_CFG

Extracto 4. Accel. Config.

- Registros para configuración del giróscopo.

1B	27	GYRO_CONFIG	R/W	XGYRO_Ct en	YGYRO_Ct en	ZGYRO_Ct en	GYRO_FS_SEL [1:0]	-	FCHOICE_B[1:0]

Extracto 5. Gyro. Config.

- Registros de mediciones del acelerómetro.

3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT_H[15:8]
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT_L[7:0]
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT_H[15:8]
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT_L[7:0]
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT_H[15:8]
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT_L[7:0]

Extracto 6. Accel. Outs

- Registros de mediciones del giróscopo.

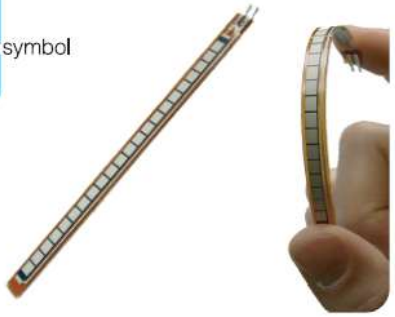
43	67	GYRO_XOUT_H	R	GYRO_XOUT_H[15:8]
44	68	GYRO_XOUT_L	R	GYRO_XOUT_L[7:0]
45	69	GYRO_YOUT_H	R	GYRO_YOUT_H[15:8]
46	70	GYRO_YOUT_L	R	GYRO_YOUT_L[7:0]
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT_H[15:8]
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT_L[7:0]

Extracto 7. Gyro. Outs

- **Flex Sensors.**

  - Características eléctricas y mecánicas.

spectrasymbol



FLEX SENSOR FS

Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
- Simple Construction
- Low Profile

Mechanical Specifications

-Life Cycle: >1 million
-Height: ≤0.43mm (0.017")
-Temperature Range: -35°C to +80°C

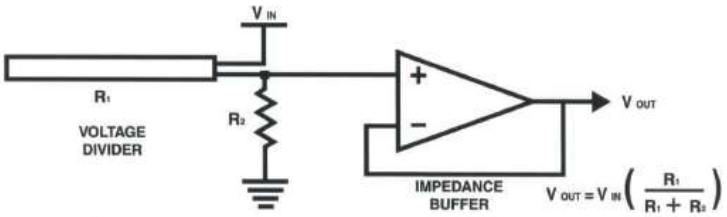
Electrical Specifications

-Flat Resistance: 10K Ohms ±30%
-Bend Resistance: minimum 2 times greater than the flat resistance at 180° pinch bend *(see "How it Works" below)*
-Power Rating : 0.5 Watts continuous; 1 Watt Peak

Extracto 8. Flex Specifications

- Ejemplo de etapa de instrumentación propuesto por el fabricante.

BASIC FLEX SENSOR CIRCUIT:



Following are notes from the ITP Flex Sensor Workshop

"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces errer due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324."

"You can also test your flex sensor using the simplest circut, and skip the op amp."

Extracto 9. Instru. Flex

- **Plataforma Arduino UNO r3**
    - Características básicas

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Extracto 10. Arduino Features

Al ser la plataforma **Arduino UNO r3** de Hardware libre, expresamos que en la página web del fabricante se encuentra el Esquemático y *Layout* de la plataforma.

A continuación, presentamos el enlace que redirecciona a estos planos:

https://store.arduino.cc/arduino-uno-rev3

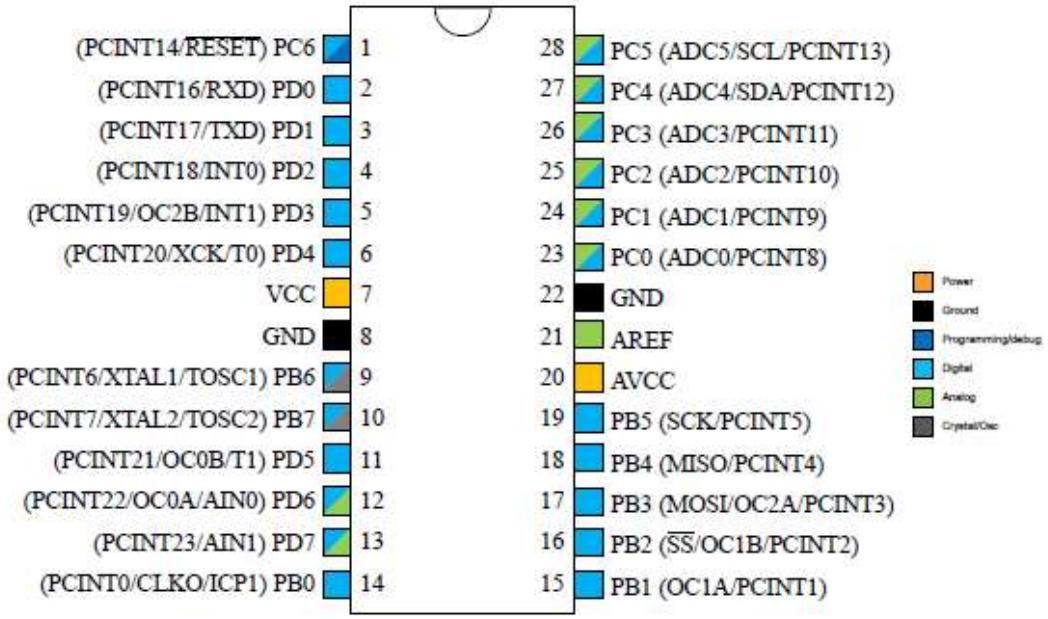
- **Atmega 328p**

    - Características básicas.

Features	ATmega328/P
Pin Count	28/32
Flash (Bytes)	32K
SRAM (Bytes)	2K
EEPROM (Bytes)	1K
Interrupt Vector Size (instruction word/vector)	1/1/2
General Purpose I/O Lines	23
SPI	2
TWI (I^2C)	1
USART	1
ADC	10-bit 15kSPS
ADC Channels	8
8-bit Timer/Counters	2
16-bit Timer/Counters	1

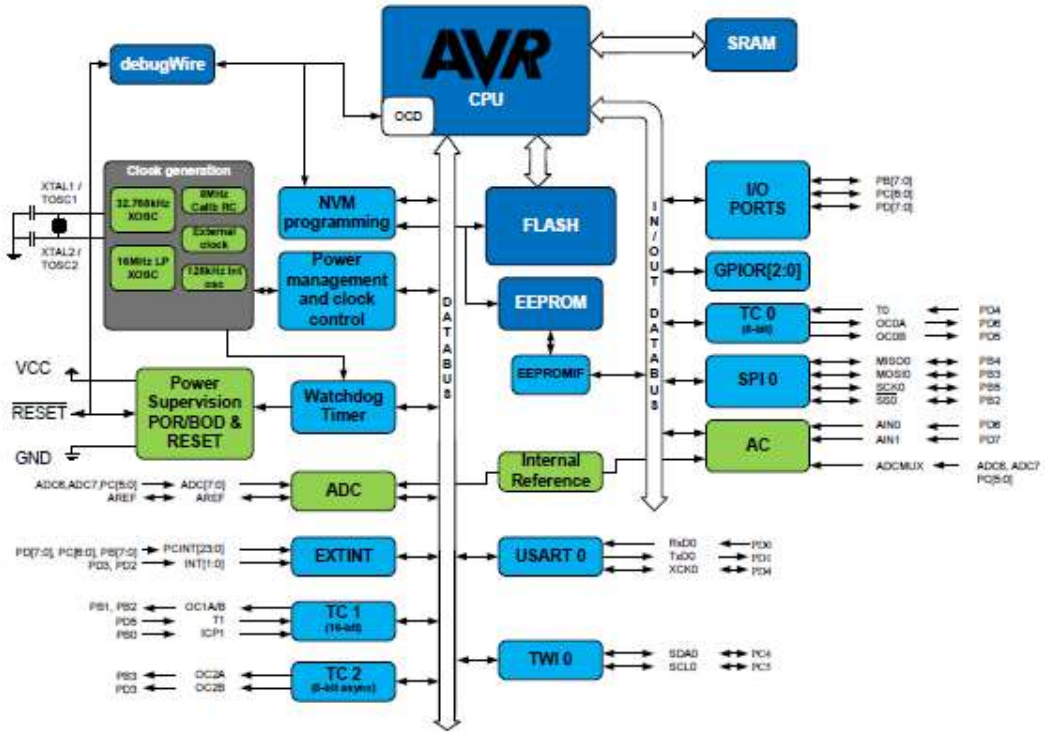
Extracto 11. uC Specifications

- Pin-out



Extracto 12. Pinout

- Diagrama de Bloques



Extracto 13. Block Diagram