Anexo 1 Descripción del estado del arte

En este Anexo se va a explicar el estado del arte de las diferentes tecnologías implicadas en el presente PFC:

1. Sistemas de bio-feedback para niños hiperactivos.
2. Dispositivos comerciales de captación de movimiento.
3. Kits de desarrollo de bajo coste.
4. Sistemas de comunicación por RF de sensores inalámbricos.

Dispositivos de bio-feedback para hiperactivos

Hoy en día existen algunas herramientas para el tratamiento de niños hiperactivos en el mercado. Sin embargo, acostumbran a ser sistemas invasivos. El más utilizado es la electroencefalografía, que se basa en el registro de la actividad bioeléctrica cerebral en condiciones basales de reposo, en vigilia o sueño, y durante diversas activaciones (habitualmente hiperpnea y estimulación luminosa intermitente) mediante un equipo de electroencefalografía (producto sanitario). Se han realizado varios estudios en niños hiperactivos usando esa técnica[8]. Estos sistemas, alteran el comportamiento del niño TDAH sin permitir que se le pueda ayudar en su estado habitual.

Dispositivos comerciales de detección de movimiento

Existen diversos dispositivos en el mercado para la detección del movimiento. A continuación se explica resumidamente los principales.

Xsense

Xsense es una de las empresas líder en el desarrollo de sensores de movimientos inerciales. Sus campos de trabajo han sido muy variados: caracterización de animaciones en 3D, rehabilitación y la ciencia del deporte, robots y estabilización de cámaras, etc.

Figura 21 Xbus Kit

Dispone de muchos dispositivos con sensores inerciales. Entre ellos el que más llama la atención es el Xbus Kit (12). Es un sistema de medida del movimiento humano portátil. El sistema irá muestreando el movimiento y almacenando los datos en una base portátil que llevará el usuario en la cintura. Tiene el hándicap de no ser inalámbrico. Los sensores envían la información a la base a través de cables. Para un niño TDAH no sería una buena aplicación ya que pasaría a ser un sistema muy invasivo, y no permitiría que el niño actuara como en condiciones normales. Además, el precio del kit es muy superior al del ECO Kit, o cualquier otro sensor inalámbrico.

InterSense

Intersense (13) es líder del mercado en el diseño y desarrollo de tecnología de precisión de movimiento. Trabaja en muchos campos, aunque uno de los primordiales es el ámbito de armas militares.

[8]En el apartado de bibliografía se incluyen algunas referencias a este respecto

Los sensores inalámbricos de movimiento que dispone no están pensados para la aplicación que queremos y son muy caras.

Ascension

Ascension es una empresa estadounidense con muchos años de experiencia en diversos campos: militar, médico, simulación en 3D, etc.

Figura 22 Imagen 3D producida por MotionSTAR Wireless LITE

Una de sus líneas de desarrollo de sistemas está orientada en la animación en 3D con sensores de movimiento.

Una de las herramientas que comercializa es el MotionSTAR Wireless LITE (14), un sistema de seguimiento del movimiento que permite capturar los movimientos de todo el cuerpo humano de hasta 5 personas a la vez en tiempo real. Es una herramienta orientada a la simulación en 3D aunque podría ser usada para un sistema equivalente al proyecto final de carrera. Tiene los mismos hándicaps que el Xbus Kit: alto coste (30000$) y muy invasivo (los sensores están conectados mediante cables).

Figura 23 MotionSTAR Wireless LITE

Microstrain

Es una empresa especializada en sensores inerciales y tecnología wireless. Dispone de un dispositivo con características parecidas a los ECO Nodes. Es el **G-Link® -mXRS™ Wireless Accelerometer Node** (15)**.** Este nodo dispone de un acelerómetro triaxial con capacidad de medir aceleraciones de ±2g. Funciona a la misma banda de frecuencia que el ECO Node, 2,4 GHz. Memoria interna de 2 Megabytes. Etc. Dispone de un único hándicap: tiene un tamaño demasiado grande. Como se puede ver en la figura, su tamaño supera los 40 mm. Por tanto, a priori es descartable.

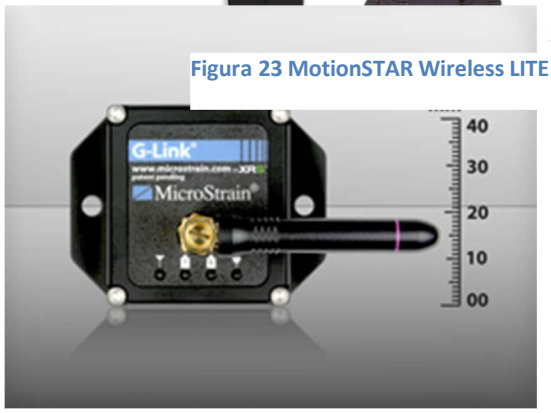


Figura 24 G-Link -mXRS Wireless Accelerometer Node

Nec-tokin

Es una empresa japonesa (16) que produce todo tipo de dispositivos electrónicos: condensadores, dispositivos piezoeléctricos, medidores de interferencias electromagnéticas, inductores, transformadores, etc.

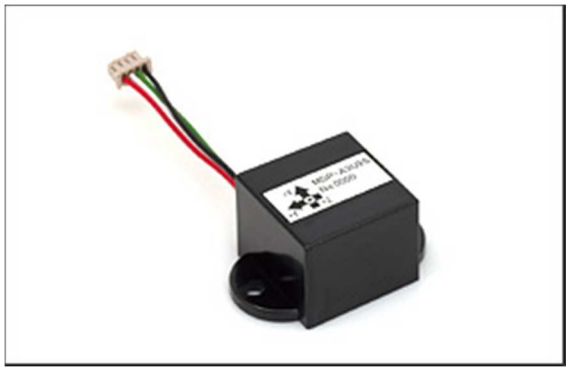


Figura 25 Dispositivo de Nec-tokin

Esta empresa ha creado un dispositivo inercial triaxial. Está muy orientado al mundo de los juegos de ordenador en 3D (en la página web

del fabricante se pone de ejemplo el Quake III). Simula el movimiento del ratón del ordenador pero en las 3 dimensiones del espacio. La comunicación con la base se realiza a través de un cable USB.

Kits de desarrollo de bajo coste con RF y sensores inerciales

Hay pocos kits de desarrollo que incluyan sensores inerciales inalámbricos y que sean de bajo coste. A continuación se muestran algunos:

eZ430-Chronos

El eZ430 (17) es un sistema integrado inalámbrico que viene incorporado en un reloj deportivo. Puede ser usado como plataforma para sistemas de seguimiento, como un display personal para redes de área personal, como un sensor inalámbrico o simplemente como un reloj.

Figura 26 eZ430 Chronos

Basado en el CC430F6137 <1 GHz RF SoC, el eZ430-Chronos es un sistema de desarrollo basado en el CC430F6137 <1 GHz RF SoC. Está provisto de un display de 96 segmentos, de un sensor de presión, y de un acelerómetro triaxial. El dispositivo inalámbrico permite al Chronos ser usado como nodo central de otros sensores inalámbricos cercanos como podómetros y medidores del electrocardiograma. El eZ430 Chronos ofrece medición de temperatura y voltaje de la batería. Está completada con una interfaz inalámbrica basada en USB CC1111.

El reloj puede ser desmontado para ser reprogramado con una aplicación personalizada. Incluye un interfaz de programación.

G-Link®mXRS™Wireless Accelerometer Node

Ya se ha explicado en el apartado anterior el funcionamiento de los nodos. Solo añadir que la empresa Microstrain vende junto con los nodos un kit de desarrollo para los sensores.

Comunicaciones

Existen diversas tecnologías para comunicar sensores inerciales con sus estaciones base. A continuación se describen cada una de estas tecnologías.

Zigbee

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área (*wireless personal area network*, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

El ámbito donde esta tecnología cobra más fuerza es en domótica. La razón de ello son diversas características que lo diferencian de otras tecnologías:

- Su bajo consumo
- Su topología de red en malla
- Su fácil integración (se pueden fabricar nodos con muy poca electrónica).

Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:

1. Facilitar las comunicaciones entre equipos móviles y fijos.
2. Eliminar cables y conectores entre éstos.
3. Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, teléfonos móviles, computadoras portátiles, ordenadores personales, impresoras o cámaras digitales.

Wi-Fi

Nokia y Symbol Technologies crearon en 1999 una asociación conocida como WECA (Wireless Ethernet Compatibility Alliance, Alianza de Compatibilidad Ethernet Inalámbrica). Esta asociación pasó a denominarse Wi-Fi Alliance en 2003. El objetivo de la misma fue crear una marca que permitiese fomentar más fácilmente la tecnología inalámbrica y asegurar la compatibilidad de equipos.

De esta forma, en abril de 2000 WECA certifica la interoperabilidad de equipos según la norma IEEE 802.11b, bajo la marca Wi-Fi. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos. Se puede obtener un listado completo de equipos que tienen la certificación Wi-Fi en Alliance - Certified Products.

En el año 2002 la asociación WECA estaba formada ya por casi 150 miembros en su totalidad.

La norma IEEE 802.11 fue diseñada para sustituir el equivalente a las capas físicas y MAC de la norma 802.3 (Ethernet). Esto quiere decir que en lo único que se diferencia una red Wi-Fi de una red Ethernet es en cómo se transmiten las tramas o paquetes de datos; el resto es idéntico. Por tanto, una red local inalámbrica 802.11 es completamente compatible con todos los servicios de las redes locales (LAN) de cable 802.3 (Ethernet).

Existen diversos tipos de Wi-Fi, basado cada uno de ellos en un estándar IEEE 802.11 aprobado. Son los siguientes:

Los estándares IEEE 802.11b, IEEE 802.11g e IEEE 802.11n disfrutan de una aceptación internacional debido a que la banda de 2.4 GHz está disponible casi universalmente, con una velocidad de hasta 11 Mbps, 54 Mbps y 300 Mbps, respectivamente.

En la actualidad ya se maneja también el estándar IEEE 802.11a, conocido como WIFI 5, que opera en la banda de 5 GHz y que disfruta de una operatividad con canales relativamente limpios. La banda de 5 GHz ha sido recientemente habilitada y, además, no existen otras tecnologías (Bluetooth, microondas, ZigBee, WUSB) que la estén utilizando, por lo tanto existen muy pocas interferencias. Su alcance es algo menor que el de los estándares que trabajan a 2.4 GHz (aproximadamente un 10%), debido a que la frecuencia es mayor (a mayor frecuencia, menor alcance).

Un primer borrador del estándar IEEE 802.11n que trabaja a 2.4 GHz y a una velocidad de 108 Mbps. Sin embargo, el estándar 802.11g es capaz de alcanzar ya transferencias a 108 Mbps, gracias a diversas técnicas de aceleramiento. Actualmente existen ciertos dispositivos que permiten utilizar esta tecnología, denominados Pre-N.

Existen otras tecnologías inalámbricas como Bluetooth que también funcionan a una frecuencia de 2.4 GHz, por lo que puede presentar interferencias con Wi-Fi. Debido a esto, en la versión 1.2 del estándar Bluetooth por ejemplo se actualizó su especificación para que no existieran interferencias con la utilización simultánea de ambas tecnologías, además se necesita tener 40.000 k de velocidad.

RF del ECO Kit

La radiofrecuencia que usa el ECO Kit opera en la misma banda frecuencial que el wifi y el bluetooth: 2,4 GHz. Dispone de un transceptor GFSK. La tasa de transmisión puede configurarse hasta 1 Mbps. Puede operar en 125 canales diferentes con un tiempo entre paso de un canal a otro de menos de 200us. Soporta saltos de frecuencia.

Anexo 2 Puesta en marcha del ECO Kit

Introducción

Para el desarrollo del Proyecto Fin de Carrera, se ha contado con el ECO Kit, un hardware desarrollado por EPL (Embedded Platform Laboratories), una empresa creada en el seno del Departamento de Ciencias de la Computación de la National Tsing Hua University (Taiwán), y la Universidad de California.

Durante todo el desarrollo del proyecto, se ha contado con la ayuda del profesor Pai H. Chou (phchou@uci.edu) y su colaboradora la Srta. Jessie Tu (corila.tu@gmail.com).

Figura 27 Prof. Pai H. Chou

Durante el aprendizaje del uso del ECO Kit, se han encontrado varios problemas que han dificultado mucho el avance. Algunos de ellos por la incoherencia en la documentación que ha ofrecido el fabricante. En otras ocasiones se ha paralizado el avance en el proyecto por errores en las librerías. Entre los ECO Nodes que se recibieron con el ECO Kit, algunos de ellos llegaron estropeados. Todos estos problemas se han ido resolviendo poco a poco con la ayuda de los fabricantes, que han ido proporcionando toda la información y material adicional que ha hecho falta con mucha diligencia.

El Kit dispone de los siguientes elementos:

b) 7 ECO nodes, con sus adaptadores para la carga eléctrica, y para la conexión con la placa de desarrollo.

Figura 28 ECO Nodes

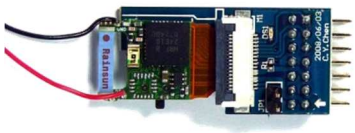


Figura 29 ECO Node con adaptador

c) Development Board, que se usa para cargar los programas en los ECO Nodes. También se usa para transmitir caracteres por el puerto serie del ordenador.

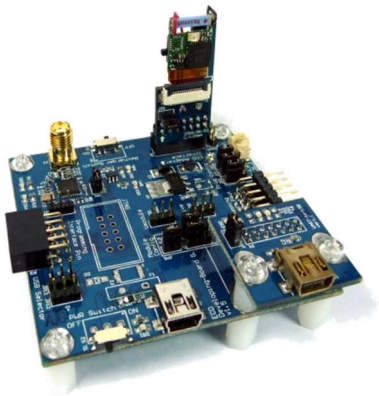


Figura 30 Development Board

d) Ethernet Base Station DEMO9S12NE64, que se usa para recibir datos de los sensores y enviarlos al ordenador. Se va a usar para la transmisión al ordenador el puerto serie.

Figura 31 Ethernet Base Statio (Estación Base)

e) Multi-Eco Battery Charging Board, que se usa para cargar simultáneamente varios ECO Nodes.

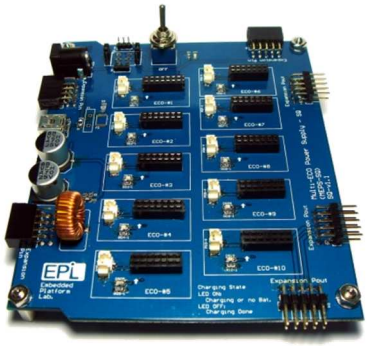


Figura 32 Multi-ECO Battery Charging Board

f) Adaptador de puerto serie, necesario para conectar la placa de desarrollo con el ordenador.

Figura 33 Adaptador de puerto serie

El ECO Node, tiene un acelerómetro (Hitachi Metal's H34C) que capta los movimientos en las tres dimensiones del espacio y mide temperatura de 0 a 75 °C. Dispone de un módulo RF que se utilizará para enviar y recibir información del movimiento y de la temperatura. Cada ECO Node se comunicará con la Ethernet Base Station proporcionándole información de los acelerómetros cada vez que tenga datos nuevos en función de la frecuencia de muestreo. A su vez, la Estación Base transmitirá los datos que vaya recibiendo hacia el ordenador, que se encargará de procesar la información en tiempo real. Cuando detecte un cambio en el estado de ánimo del niño con TDAH, actuará sobre el niño. De esta forma conseguimos un feedback que servirá para el tratamiento.

Puesta en marcha paso a paso

Se explica a continuación como se ha puesto en funcionamiento todo el sistema, los programas y drivers que se han necesitado, y los problemas que se han encontrado. Para ello, primero de todo, se describe la problemática que se ha encontrado con cada una de las diferentes partes para después abordar la problemática general de la puesta en marcha de todo el Kit.

El ECO Node

Los ECO Nodes[9] son unos dispositivos de tamaño extremadamente pequeño, apenas ocupan 1 cm^3. Pesan solo 2 gramos incluidos la batería y la antena. El módulo de RF tiene un alcance de 10 metros a una velocidad de 1Mbps. La batería tiene una duración de 4 horas.

Los bloques más importantes que componen el ECO Node:

1. Un bloque de RF que nos permitirá comunicarnos a gran velocidad con la Estación Base. Funciona a 2,4GHz.
2. Un acelerómetro de 3 ejes. Mide hasta de -3g a +3g.
3. Un LED.
4. Un Switch para encender el ECO Node.
5. Un puerto de expansión para la programación del ECO Node, y para la carga y descarga.

En el Kit se encuentra la documentación necesaria para guiar paso a paso para la programación de los ECO Nodes[10]. Se tiene a disposición unas librerías en C que proporcionan los fabricantes del hardware. Esas librerías sirven de base para la programación de los ECO Nodes. Sin embargo, para el desarrollo del PFC no son suficientes, ya que se ha necesitado editar una

[9] Podemos encontrar una descripción más completa en los documentos adjuntos en el CD del PFC, en la carpeta Documentacion, en los documentos ECO_Node.ppt y ECO_tutorial_v1.pdf (paginas 1-3).

[10] En la carpeta Documentacion, está el documento ECO_tutorial_v1.0.pdf donde a partir de la página 5 se explica cómo poner a punto el ordenador para conseguir programar el ECO Node. Además da una explicación completa de las funciones principales disponibles en las librerías que tenemos por defecto.

librería propia. El programa que se ha empleado ha sido Eclipse, versión para C/C++. La carga del programa compilado se realiza a través de un programa: el Eco Manager. Este programa está diseñado a propósito para cargar los programas a la Development Board. Junto con las librerías, se encuentran varios programas en la web del fabricante[11] que sirven para hacer diversas pruebas.

Development Board

La Development Board es la placa que permite la programación y carga de los ECO Nodes, así como su uso como módulo de expansión por parte del ECO Node.

La Development Board tiene tres modos de funcionamiento:

a) Eco Uploader, para cargar los programas en el Eco.
b) Eco Debugger, para que el Eco conectado en la placa pueda usar el módulo de expansión de interfaz del Development Board. Este modo se usa para recibir datos de otros Eco nodes y enviarlo a través del puerto USB que conecta la placa con el ordenador.
c) Eco Battery Charger, para cargar el Eco Node conectado en ese momento a la Development Board.

Para utilizar la Development Board en modo Uploader se colocan los jumpers tal como se muestra en la figura:

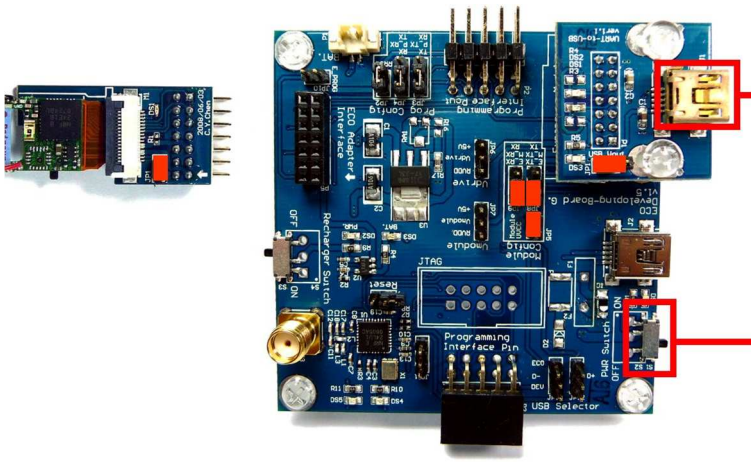


Figura 34 Development Board en modo Uploader

Por otro lado, para usarla en modo Debugger, se usa la configuración siguiente:

[11] http://epl.cs.nthu.edu.tw/EcoKit/

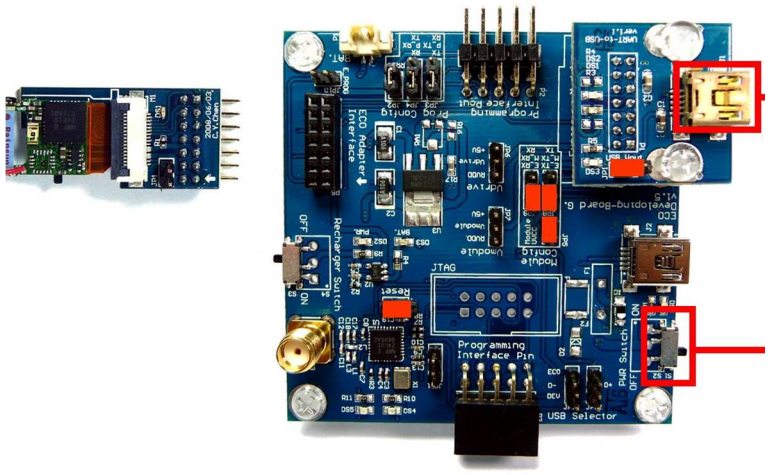


Figura 35 Development Board en modo Debugger

Para usar la placa en modo Battery Charger se colocan los jumpers como en el modo Debugger. Además, el ECO Node deberá estar con el interruptor de apagado, y el jumper colocado. En cuanto la placa Development Board, además de tener los jumpers colocados como en el modo Debugger, se debe activar el interruptor Recharger Switch.

Para la carga de los programas al ECO Node se han necesitado los siguientes componentes:

1. Un ECO Node
2. La Development Board
3. El adaptador de puerto serie
4. El adaptador USB-miniUSB que conecta el adaptador de puerto serie con el ordenador

Se necesitan también los siguientes programas y drivers:

1. Python-2.5.4
2. Pygtk-setup
3. Gtk-2.12.9-win32-1
4. Eco_Installer
5. f5u109_xp
6. Eclipse IDE for C/C++ Developers, lo hemos descargado de la web del fabricante

Todos estos programas se encuentran en la web del fabricante[12] del ECO Kit.

Para cargar los ECO Nodes la mejor solución es usar la Multi-Eco Battery Charging Board, ya que de esta forma se pueden cargar varios ECO Nodes a la vez. Para ello, se colocan los ECO Nodes con sus adaptadores en la placa. Se ha de comprobar que los ECO Nodes estén apagados y el jumper de los adaptadores colocado. Se disponen los jumpers en la placa tal como se muestra en la figura:

[12] http://epl.cs.nthu.edu.tw/EcoKit/

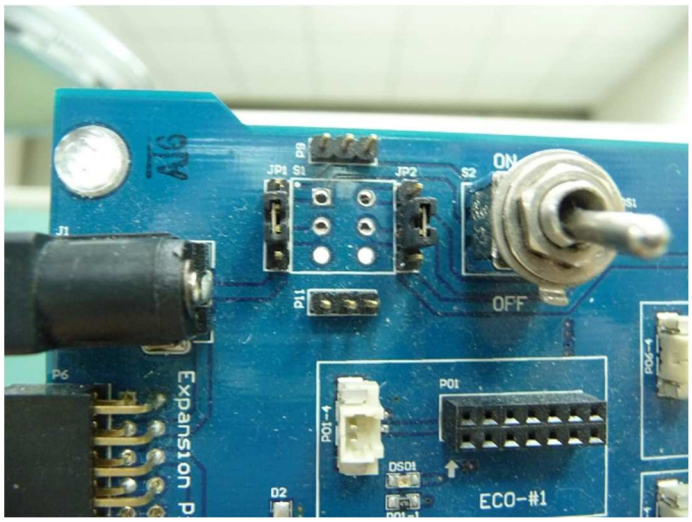


Figura 36 Configuración jumpers para la carga de nodos

Anexo 3 Firmware del ECO Node

Librerías, variables globales y funciones

Podemos ver debajo el código donde incluimos las librerías, declaramos las variables globales y funciones.

```
/* Programa principal eco_client.c para la emisión de muestras
 * del acelerómetro por Radiofrecuencia de un ECO Node.
 *
 * Primavera 2010
 * Lucas Muley Vilamú
 * Estudiante de Ingeniería de Telecomunicaciones
 * Universidad de Zaragoza
*/

#include <eco/reg24e1.h>
#include <eco/eco_sys.h>
#include <adc/adc.h>
#include <isr/isr_timer.h>
#include <serial/serial.h>
#include <rf/rf.h>

/*Configuración de los parámetros de la rf*/
struct rf_config rf_data_rx = { {0x00}, /* data2 width */
        {0xd8}, /* data1 width *///27 bytes
        {0x00, 0x00, 0x00, 0x00, 0x00}, /* addr2 */
        {0x00, 0x00, 0xb5, 0xb5, 0xb5}, /* addr1, host addr */
        {0x63}, /* 24-bit address, 8-bit CRC */
        {0x6f, 0xed} };// canal de frecuencia 118
struct rf_config rf_data_tx = { {0x00}, /* data2 width */
        {0x88}, /* data1 width *///17 bytes
        {0x00, 0x00, 0x00, 0x00, 0x00}, /* addr2 */
        {0x00, 0x00, 0xb7, 0xb7, 0xb7}, /* addr1, host addr */
        {0x63}, /* 24-bit address, 8-bit CRC */
        {0x6f, 0xec} };// canal de frecuencia 118

extern char msg[10];
unsigned int cont;

int accx,accy,accz;
int n_eco;

void store_cpu_rate(unsigned long hz);
```

El código introducido en el ECO Node utiliza diversas librerías[13] del fabricante del ECO Kit que permiten gestionar…comunicación, acceso a periféricos, etc…. Durante el proceso de desarrollo del firmware se han corregido varios errores presentes en las mismas, llegando

[13] Estas librerías se pueden encontrar en el CD del PFC y también en el Anexo 5, junto con los demás códigos.

incluso a reprogramar completamente la librería isr_timer.h. Brevemente se comentan las funcionalidades de las principales librerías:

```
#include <eco/reg24e1.h>
#include <eco/eco_sys.h>
#include <adc/adc.h>
#include <isr/isr_timer.h>
#include <serial/serial.h>
#include <rf/rf.h>
```

Una vez incluidas las librerías se declaran las variables globales que se van a usar en el programa. En primer lugar, se inicializa las variables de configuración de la transmisión por RF. A continuación podemos ver la configuración de recepción y envío.

```
/*Configuración de los parámetros de la rf*/
struct rf_config rf_data_rx = { {0x00}, /* data2 width */
        {0xd8}, /* data1 width *///27 bytes
        {0x00, 0x00, 0x00, 0x00, 0x00}, /* addr2 */
        {0x00, 0x00, 0xb5, 0xb5, 0xb5}, /* addr1, host addr */
        {0x63}, /* 24-bit address, 8-bit CRC */
        {0x6f, 0xed} };// canal de frecuencia 118
struct rf_config rf_data_tx = { {0x00}, /* data2 width */
        {0x88}, /* data1 width *///17 bytes
        {0x00, 0x00, 0x00, 0x00, 0x00}, /* addr2 */
        {0x00, 0x00, 0xb7, 0xb7, 0xb7}, /* addr1, host addr */
        {0x63}, /* 24-bit address, 8-bit CRC */
        {0x6f, 0xec} };// canal de frecuencia 118
```

La estructura de datos que se muestra arriba está definida en la librería rf.h.

Al disponer los ECO Nodes de dos antenas para la transmisión y recepción se puede dar una configuración diferente para cada antena. Para mayor simplicidad, solo hemos usado una de las antenas. Por tanto el campo de longitud de datos del canal 2 (*data2_width*) está a 0 para ambas transmisiones (transmisión y recepción). En el campo de longitud de datos del canal 1(*data1_width*) tenemos un valor diferente para la transmisión y la recepción. En recepción se ha dejado la longitud de campo que había por defecto (27 bytes), en cambio en la transmisión se ha disminuido (17 bytes) para disminuir el tiempo de emisión del ECO Node. Por tanto el valor de longitud queda en 0xd8 para la recepción y en 0x88 para la transmisión. Los valores están en formato hexadecimal y representan la longitud en bits del mensaje que se transmitirá:

$$0\text{xd8 (hex)} = 216\ bits = 27\ bytes$$

$$0\text{x88 (hex)} = 136\ bits = 17\ bytes$$

En los bytes addr1[5] y addr[5] tenemos las direcciones de envío o recepción para cada uno de los canales. Podemos comprobar que para el canal 2, las direcciones están a cero. En el caso de la recepción, la dirección que aparece es la del ECO Node (0x00, 0x00, 0xb5, 0xb5, 0xb5). El ECO Node capturará los datos que tengan solo la dirección indicada, que es la suya. En cambio, para la transmisión, la dirección que aparece es la dirección de la Estación Base (0x00, 0x00, 0xb7, 0xb7, 0xb7).

Después de los bytes de las direcciones, la estructura tiene un byte para indicar la longitud de la dirección. Los 7 primeros bits del byte contienen la longitud. El último bit contiene un código de redundancia cíclica de control.

$$0\text{x}63\ (hex) = 0110\ 0011\ (bits)$$

$$Longitud = 0110\ 001 = 24\ (bits)$$

$$CRC = 1 = 1$$

La longitud del campo de dirección va a ser de 24 bits = 3 bytes.

Los dos últimos bytes, nos servirán para configurar la RF. En concreto, el último byte nos permite elegir el canal de RF y determinar si es envío o recepción. En el caso de la recepción:

$$rf\_prog = 0xed\ (hex) = 1110\ 1101$$

$$canal = 1110\ 110\ (bits) = 118$$

$$envío/recepción\ = 1 = recepción$$

Una vez se ha configurado las estructuras de los parámetros de envío y recepción de RF se declaran las demás variables globales.

```
extern char msg[10];
unsigned int cont;

int accx,accy,accz;
int n_eco;

void store_cpu_rate(unsigned long hz);
```

extern char msg será la variable donde se almacenará el mensaje a enviar. *unsigned int cont* es el contador que se irá incrementando por cada muestra capturada de los acelerómetros. *int accx, accy, accz* contendrán el último valor muestreado de las aceleraciones en los tres en los tres ejes del espacio. *int n_eco* va a guardar el valor del ECO Node que se está utilizando. Por último, la función *store_cpu_rate(unsigned long hz)* es la función que define la frecuencia de la CPU. Necesita ser declarada antes para ser utilizada.

Inicialización de variables y configuraciones iniciales

Una vez en el *main*, declaramos las variables que se van a usar y que todavía no habían sido declaradas (no son globales). También inicializamos las variables que necesitan tener un valor inicial. *p_contador* va a ser actualizado cada vez que se envíe una muestra por RF con el valor de cont. De esta forma, podremos saber si una muestra nueva ha sido enviada ya o no por RF. La variable *seleccion*, indica en qué estado se encuentra el ECO Node. Si el valor es 'o' entonces el nodo está esperando a que la Base Station ordene el envío de muestras. Si el valor es 'u' quiere decir que hemos recibido la orden de empezar el envío de muestras. *envios_o* es un contador que se usará para contabilizar el número de veces que se envía a la Estación Base la confirmación de una orden de detención de envío.

```
void main()
{
        unsigned int p_contador;

        char seleccion='o';

        char envios_o=0;

        int i=0;

        store_cpu_rate(16);   //configuramos la frecuencia del
procesador

        P0_DIR &= ~0x28;      //configuración del led
        P0_ALT &= ~0x28;

        adc_init(ADC_CLK_D8, ADC_RES_12, EXTREF);    //configuración
del ADC del acelerómetro

        rf_init();      //encendemos la RF

        /*hacemos dos parpadeos en los leds para poder ver visualmente
que se está inicializando el ECO Node*/
        blink_led();
        mdelay(200);
        blink_led();
        mdelay(200);
        blink_led();
        mdelay(200);
        blink_led();
        mdelay(200);

        p_contador=0;
        cont=0;

        n_eco=1;
        msg[7]=0;
```

store_cpu_rate(16) configura la frecuencia del reloj a 16 Mhz.

```
        P0_DIR &= ~0x28;      //configuración del led
        P0_ALT &= ~0x28;
```

La configuración de los registros *P0_DIR* y *P0_ALT* configura el led del nodo. *adc_init(ADC_CLK_D8, ADC_RES_12, EXTREF)* configura el conversor analógico digital. *EXTREF* pondrá el ADC en estado de parada. *ADC_CLK_D8* lo configura el conversor a la frecuencia del reloj, dividido entre 8. *ADC_RES_12* configura la resolución del conversor a 12 bits, que es la máxima permitida por el ADC. *rf_init()* incializa los parámetros de RF. A continuación encendemos y apagamos los leds del nodo para comprobar que se ha encendido correctamente el ECO Node. Inicializamos las restantes variables.

Bucle principal

Para explicar fácilmente el funcionamiento del bucle principal, debajo hay un diagrama de flujo que describe su comportamiento.

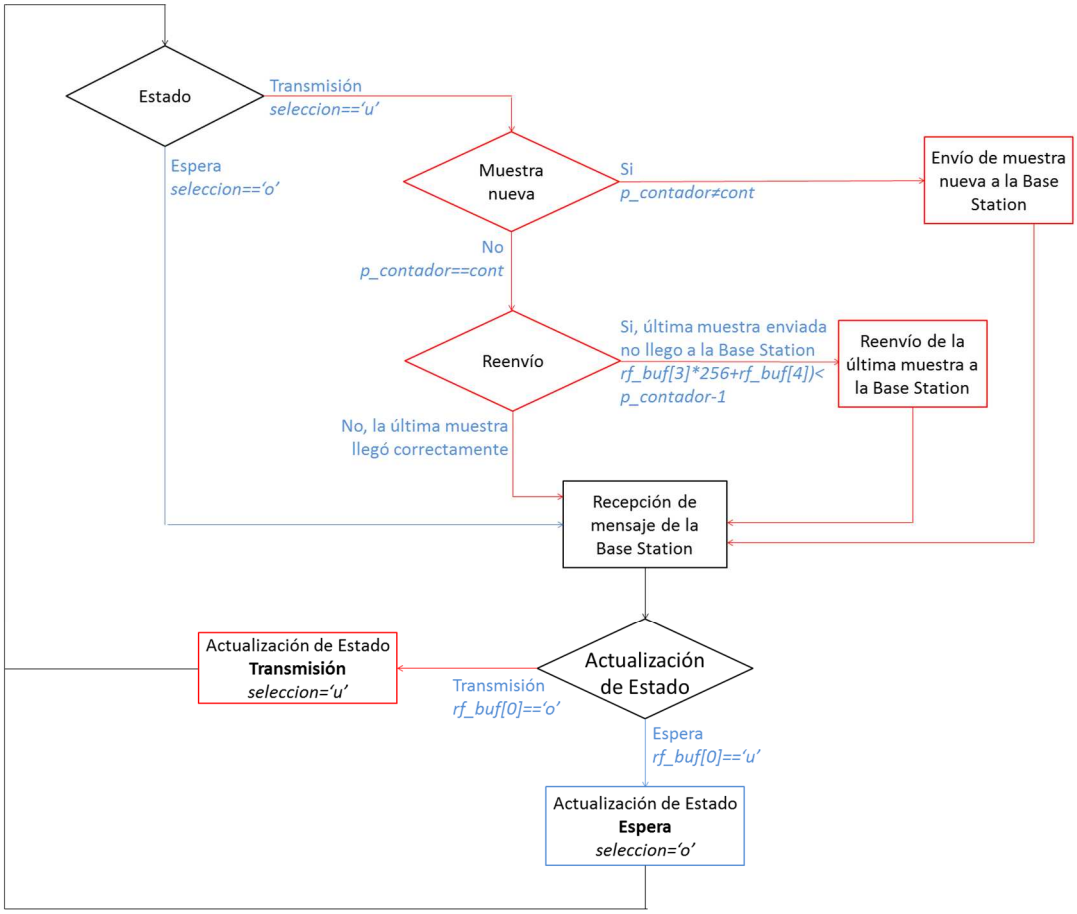


Figura 37 Diagrama de flujo de ECO Node

El bucle principal empieza diferenciando entre los dos estados en que puede estar el ECO Node. Si está en estado de Espera, lo único que hace el bucle es esperar la recepción de un mensaje de la Base Station y una vez recibido el mensaje mirar si contiene una orden de empezar a transmitir muestras del acelerómetro.

```
. . .
switch(selección){
      case 'o':
            cont=0;
            /*realizamos un parpadeo en el Led del ECO Node para poder
ver visualmente que está encendido y con batería. Con este parpadeo
también detectamos que estamos recibiendo muestras de la Base
Station*/
            mdelay(2*n_eco-1);
            for(i=0;i<4;i++){
                  blink_led();
                  mdelay(10);
```

```
                  blink_led();
                  mdelay(300);
            }

            break;
. . .
```

Como se puede ver arriba, el Estado Espera, lo único que producirá será un parpadeo en el led del ECO Node. Sirve para detectar que existe comunicación entre los ECO Nodes y la Base Station.

En el caso del Estado de Transmisión podemos ver el código a continuación:

```
. . .
switch(selección){
      case 'o':
            . . .
      case 'u':
            /* Se detecta si hay una muestra nueva*/
            if(p_contador!=cont){
                  p_contador=cont;
                  blink_led();
                  rf_configure(cfg_tx);
                  rf_send(dst_addr, 3, msg, 17);
                  blink_led();
            }
            else{
                  /*Si no hay una muestra nueva, se analiza la trama
recibida por la Base Station para ver si la última muestra que recibió
la base fue la última que envió en ECO Node, si no es así, entonces se
reenvía la última muestra otra vez*/
                  if((rf_buf[3]*256+rf_buf[4])<p_contador-1){
                        blink_led();
                        rf_configure(cfg_tx);
                        rf_send(dst_addr, 3, msg, 17);
                        blink_led();
                  }
            }

            break;
      default:
      break;
}
. . .
```

Cuando la variable *seleccion* tiene el valor 'u' significa que estamos en el estado de Transmisión. Lo primero que hará el ECO Node será comprobar si ha habido una interrupción del Timer y si por tanto tenemos una muestra nueva. En el caso de tener una muestra nueva, la transmitirá por RF. En el caso de no tener una muestra nueva, analizará la trama recibida por la Base Station para comprobar si la última muestra recibida por la base es la misma que la última que envío. Si no es el caso, quiere decir que la última muestra se ha perdido. Por tanto volverá a enviarla.

El código de recepción de paquete de la Estación Base es el siguiente:

```
/*Recepción de paquete de la Base Station*/
            rf_configure(cfg_rx);
            rf_wait_msg();
```

Una vez hemos recibido un paquete de la Base Station hemos de actualizar el estado del Nodo. El valor del estado del ECO Node lo encontraremos en el primer byte de la trama que ha transmitido la Base Station, *rf_buf[0]*. Si se tiene *rf_buf[0]='u'* y *seleccion='o'* quiere decir que el nodo estaba en estado de Espera, y que ahora va a pasar al estado de Transmisión. A continuación tenemos el código del caso *rf_buf[0]=='u'*:

```
switch(rf_buf[0]){
      case 'u':
            if(seleccion=='o'){
                  timer0_initialize();    /*Se inicializa el Timer*/
                  msg[8]='u';
            }
            seleccion='u';
            break;
. . .
```

Cuando el nodo detecta la orden de la Base Station de pasar al estado Transmisión, inicializará el Timer y pondrá el valor de *msg[8]='u'*. Esto sirve para que la base pueda confirmar que el nodo se ha puesto en funcionamiento y está transmitiendo muestras de la aceleración. Por último, una vez inicializado el tiemer, el ECO Node cambiará el valor de *seleccion* a 'u' para que el estado se actualice a Transmisión.

Cuando la Base Station quiera parar el envío de muestras del ECO Node, enviará un paquete con *rf_msg[0]=='o'*. El nodo hará lo que se indica en el código siguiente:

```
switch(rf_buf[0]){
      case 'u':
            . . .
      case 'o':
            cont=0;     /*Inicialización del contador*/
            /*Detención del Timer 0*/
            EA = 0;
            TR0 = 0;
            ET0 = 0;
            seleccion='o';    /*Cambio del Estado*/
            /*Reset en el valor de los campos del mensaje de
transmisión*/
            msg[0] = 0;
            msg[1] = 0;
            msg[2] = 0;
            msg[3] = 0;
            msg[4] = 0;
            msg[5] = 0;
            msg[6] = 0;
            msg[7] = 0;
            msg[8] = 'o';
            msg[9] = '0';
```

```
            for(envios_o=0;envios_o++;envios<6){
                  mdelay(100);
                  rf_configure(cfg_tx);
                  rf_send(dst_addr, 3, msg, 23);
            }
            break;

      default:
            break;
}
```

Una vez recibido la orden de parar de enviar muestras, el nodo detendrá el Timer, inicializará el contador y los valores de los campos del mensaje, cambiará el estado del nodo (*seleccion='o'*) y por último hará 5 transmisiones a la Base Station para que la base tenga confirmación de que el ECO Node ha actualizado su estado.

Configuración del Timer 0

Para generar las interrupciones a la frecuencia de 40 Hz se ha tenido que programar una librería pensada expresamente para ello. Las librerías proporcionadas por el fabricante no han funcionado correctamente. Se ha utilizado uno de los 3 timers disponibles en el microprocesador nRF24E1. Se han programado 2 funciones: *void timer0_initialize (void)* y *static void timer0_isr(void) interrupt 1 using 1*. La primera de ellas es la encargada de configurar el timer con los parámetros adecuados. La segunda es la función de interrupción del timer0. Se ejecutará cada vez que haya una interrupción del timer.

El funcionamiento del timer0 se describe detalladamente en el datasheet del nRF24E1. Resumidamente, consiste en un contador que se puede configurar en tres modos distintos. El nodo lo usará en el modo 1 y por tanto dispondrá de una longitud de 16 bits. Inicialmente se configura el valor inicial desde el que empieza a contar, y cuando llega al valor de 0xFFFF genera una interrupción. Esa interrupción ejecutará la subrutina de interrupción del timer0. El contador se puede configurar para que vaya sumando a dos frecuencias diferentes: Freloj/4, o Freloj/12. El nodo va a usar la Freloj/12 ya que si no el contador no será capaz de contar el número de veces suficientes como para sumar 25ms. Necesitamos una periodo mayor, y por tanto una frecuencia menor. El valor del número de ciclos que debe contar es:

$$T = 25\ ms = X * T_{contador} = X * Freq_{contador}$$

$$Freq_{contador} = \frac{Freloj}{12} = \frac{16\ Mhz}{12} = 1,\hat{3}\ MHz$$

$$X = 25\ ms * Freq_{contador} = 25 \cdot 10^{-3} * 1{,}3 \cdot 10^{6} = 33333\ ciclos\ del\ contador$$

Por tanto, el contador debe contar 33333 ciclos. El valor inicial que debe tener el contador es uno que al llegar a 0xFFFF habrá contado 33333 ciclos. Por tanto:

$$Vinicial_{contador} = 0xFFFF - 33333 = \ 65535 - 33333 = 32202 = 0x7DCA$$

Por último, al $Vinicial_{contador}$ se le va a restar 20 ciclos. Estos 20 ciclos es el tiempo que tarda el microprocesador en parar el contador, obtener el $Vinicial_{contador}'$, actualizar los valores de

los registros del contador, y por último, volver a activar el contador. El valor de 20 ciclos es una aproximación.

$$Vinicial_{contador}' = 32202 - 20 = 0x7DB6$$

A continuación se puede ver el código de la librería del Timer0:

```
/*
 * Author(s): Lucas Muley Vilamú (CPS, Universidad de Zaragoza)
 *
 * Permission to copy, modify, and distribute this program is granted
 * for noncommercial purposes, provided the author(s) and copyright
 * notice are retained. All other uses require explicit written
 * permission from Lucas Muley Vilamú.
 *
 * A simple timer ISR for time 0
 *
 * Lucas Muley Vilamú <lucasmuley@yahoo.es>
 * 2011/01/12
 */
#include <eco/reg24e1.h>
#include <serial/serial.h>
#include <eco/eco_sys.h>
#include <rf/rf.h>

void int_print(int val);
void mdelay(unsigned int msec);

struct rf_config *cfg_rx = &rf_data_rx;
struct rf_config *cfg_tx = &rf_data_tx;


extern unsigned int cont;

extern unsigned char rf_buf[RF_BUF_LEN];
char dst_addr[3] = { 0x65, 0x65, 0x65 };

extern int n_eco;
extern int accx,accy,accz;

char msg[10] = {0x00};


//#define TIMER0_COUNT 0xBEE5 /* 80 Hz 33.333/2 = 16666 = 0x411A. Por
tanto 0xFFFF-0x411A=0xBEE5. Se le restan 20 cliclos.*/
#define TIMER0_COUNT 0x7DB6 /* 40 Hz 33.333 = 0x8235. Por tanto
0xFFFF-0x8235=0x7DCA. Se le restan 20 ciclos. Por tanto 0x7DB6*/

/*La señal de reloj funciona a Freloj/12=16MHz/12=1,3333Mhz. Por tanto
T = 0,75us. Si queremos muestrear a 25ms, necesitamos 25ms/0,75us =
33.333 ciclos del timer para conseguirlo*/
static void timer0_isr(void)  interrupt 1 using 1
{
```

```
	unsigned i;

	TR0 = 0; /* stop timer 0 */
	/*Durante este tiempo el timer0 no estará contando el tiempo, ya
que estará actualizando el registro del valor inicial del timer.
	 * Este tiempo perdido equivale aproximadamente a 20 ciclos de
reloj. Se han medido experimentalmente*/
	i = TIMER0_COUNT + TL0 + (TH0<<8);
	TL0 = i;
	TH0 = i >> 8;
	TR0 = 1; /* start timer 0 */

	accx = adc_read(X_AXIS);
	accy = adc_read(Y_AXIS);
	accz = adc_read(Z_AXIS);

	msg[0] = (accx >> 8);
	msg[1] =  accx & 0xff;
	msg[2] =  (accy >> 8);
	msg[3] = accy & 0xff;
	msg[4] = (accz >> 8);
	msg[5] = accz & 0xff;
	msg[6] = ((cont & 0xff00) >> 8);
	msg[7] = cont & 0xff;
	cont++;

	switch(n_eco){
	case 1:
		msg[9] = '1';
		break;
	case 2:
		msg[9] = '2';
		break;
	case 3:
		msg[9] = '3';
		break;
	case 4:
		msg[9] = '4';
		break;
	case 5:
		msg[9] = '5';
		break;
	default:
		break;
	}
}


void timer0_initialize (void)
{
EA = 0; /* disable interrupts */
//timer0_tick = 0;
TR0 = 0; /* stop timer 0 */
```

```
TMOD &= ~0x0F; /* clear timer 0 mode bits */
TMOD |= 0x01; /* put timer 0 into 16-bit no prescale */
TL0 = (TIMER0_COUNT & 0x00FF);
TH0 = (TIMER0_COUNT >> 8);
PT0 = 0; /* set low priority for timer 0 */
ET0 = 1; /* enable timer 0 interrupt */
TR0 = 1; /* start timer 0 */
EA = 1; /* enable interrupts */
}
```

La rutina de interrupción del timer0 se ejecutará cada vez que el contador del timer0 llegue a 0xFFFF. Lo primero que hace la rutina es parar el contador y reiniciarlo a su valor inicial. Después se reiniciarlo, vuelve a encenderlo. El tiempo que tarda en realizar estas operaciones se ha aproximado a 20 ciclos. A continuación, la subrutina de interrupción capturará los datos del acelerómetro y actualizará los valores del mensaje a transmitir (*msg*). Una vez actualizado *msg*, saldrá de la subrutina de interrupción y se seguirá ejecutando el código.

Anexo 4 Firmware de la Estación Base.

Para el desarrollo del Firmware de la Estación Base, se ha usado el proyecto Connector App que ofrece el proveedor de la Estación Base, como punto de partida para el desarrollo. Para la implementación del sistema se ha modificado el bucle principal y se han programado dos funciones principales.

A continuación se muestra el bucle principal:

```
while(1){

        tkh_send_to_eco_unizar();
        tkh_recv_from_eco_unizar();

      }
```

A continuación se puede ver la función *tkh_send_to_eco_unizar(void)*, una función desarrollada durante el proyecto para el envío de datos a los ECO Nodes.

```
//funcion de envio de datos al ECO Nodedisable i
void tkh_send_to_eco_unizar(void) {
  char buf_recepcion[20];
  INT16 len=0;
  int z=0,j=0;

  switch(caso){


    case 0:
      //estado incial, a la espera de recibir la orden de inicio de
muestreo
      len=SCIrecv(buf_recepcion);
      if(buf_recepcion[0]=='u'){
        caso=1;
      } else{
        status[0]='o';
        fast_init_rf24g_send();
        rf24g_send(status,183,27);
        rf24g_send(status,180,27);
        rf24g_send(status,181,27);
      }
      break;
    case 1:
    //empezamos a recibir datos alternando un nodo y otro
      status[0]='u';
      status[1]='1';
      fast_init_rf24g_send();
      variable=variable+1;
      status[1]=variable>>8;
      status[2]=variable;
      len=SCIrecv(buf_recepcion);
      if(buf_recepcion[0]=='o') {
        caso=3;
        status[0]='o';
```

```
        crc=0; //usamos esta variable de forma provisional
      }


      if(condicion==1){
        status[3]=seq_nodo[0];
        status[4]=seq_nodo[1];
        rf24g_send(status,183,27); //direccion 170=0xaa. ECO Node 4.
Canal 118



      }else if(condicion==2){
        status[3]=seq_nodo2[0];
        status[4]=seq_nodo2[1];
        rf24g_send(status,180,27); //direccion 180=0xb4. ECO Node 5.
Canal 118


      } else{
        status[3]=seq_nodo3[0];
        status[4]=seq_nodo3[1];
        rf24g_send(status,181,27); //direccion 181=0xb5. ECO Node 1.
Canal 118
      }

      break;
    case 2:
      //este estado no se utiliza
      break;
    case 3:
      //en este estado solo se va a enviar varios mensajes a los ECO
Nodes para que finalicen su transmision
      crc=crc+1; //usamos esta variable de forma provisional
      SCIsend("caso 3");
      status[0]='o';
      fast_init_rf24g_send();
      rf24g_send(status,183,27);
      rf24g_send(status,180,27);
      rf24g_send(status,181,27);

      if (crc==30){

          caso=0;
          crc=0;
      }

      break;
    default:
      break;
  }
}
```

A continuación se muestra la función *thh_recv_from_eco_unizar(void)*. Esta función será la responsable de recibir los datos que le llegan de los ECO Nodes.

```
void tkh_recv_from_eco_unizar(void) {
  int i = 1, j = 0;
  UINT8* buf;
  char x[5];
  c=0;

  init_rf24g_recv();
  c = rf24g_recv(tempbuf[j]);  //definida como UINT8 tempbuf[20][32]

  if(c==0){

    SCIsendC('F');
    crc=0;

    for(i=1;i<9;i++){
      for(j=0;j<1;j++){
        crc=crc+tempbuf[j][i];
        SCIsendC(tempbuf[j][i]);
      }
    }

    SCIsendC(tempbuf[0][10]);

    SCIsendC(tempbuf[0][9]);

    crc=crc+tempbuf[0][10];
    crc=crc+tempbuf[0][9];
    /*Envio del CRC*/
    SCIsendC(crc&0x00FF);

    SCIsendC('F');
    SCIsend("\r");

    if(condicion==1){
      //acabo de recibir del nodo 2
      seq_nodo[0]=tempbuf[0][7];
      seq_nodo[1]=tempbuf[0][8];

    }else if(condicion==2){
      seq_nodo2[0]=tempbuf[0][7];
      seq_nodo2[1]=tempbuf[0][8];

    }else{
      seq_nodo3[0]=tempbuf[0][7];
      seq_nodo3[1]=tempbuf[0][8];

    }
  }

    if(condicion==1){
```

```
      //acabo de recibir del nodo 2
      condicion=3;
      /*condicion=2; //modificamos para solo usar dos nodos*/
    }else if(condicion==2){

      condicion=3;
    }else{

      condicion=1;
    }
  init_rf24g_send();

}
```

Anexo 5 Scripts en Matlab y software del PC

A continuación se van a mostrar los diferentes scripts que se han utilizado para la programación de la aplicación.

Script para la toma de datos de usuarios

A continuación se muestra el script utilizado para la carga de datos en personas. La función de este script es tomar datos de una persona durante 5 minutos y almacenarlas en un archivo[14].

```
function main()
clear all;
close all;

%variables globales
global s;           % puerto serie
global filename;    % fichero de log
global i;
global token_last;
global errores;
global seq_ant;
global muestras;
global muestras_captured;
global n_data;
global inicio;
global seq_0;
global e;


%_____________________________________%

%inicialización de variables
seq_ant=[0 0 0 0 0];
errores=[0 0 0 0 0];
muestras=[0 0 0 0 0];
muestras_captured=[0 0 0 0 0];
n_data=[0 0 0 0 0];
inicio=0;
seq_0=[0 0 0 0 0];
e=0;
token_last=[];
indice=1;
visualizar=[0 0 0 0 0];
n_min=1;



buffer1=[];
buffer2=[];


error_nodo_base=0;
error_base_pc=0;


%_____________________________________%

%pedimos los datos de la persona, el estado de actividad y el numero de
%minutos que durará el experimento
n_persona=input('Introduce el número de la persona\n');
actividad=input('Introduce el estado de actividad\nEstado 1 : Actividad nula
-> ECO Node encima de la mesa sin movimiento\nEstado 2 : Actividad baja  ->
estado de calma, por ejemplo estudiando, viendo la televisión, navegando por
internet, etc.\nEstado 3 : Actividad media -> actividades que tengan incluido
```

[14] Este script lleva el nombre de pserial2.m

```
un desplazamiento físico: mover una caja, caminar de un sitio a otro, hablar
de pié en público\nEstado 4 : Actividad alta  -> actividad ajetreada, como por
ejemplo realizando ejercicio físico, movimientos nerviosos, saltos, etc\n');
n_min=input('¿Cuantos minutos durará el experimento?\n');
n_min_str=int2str(n_min);   %número de minutos en formato texto

%creamos el nombre del archivo donde vamos a guardar los datos nuevos
cabecera=(strcat('Date_',datestr(now, 29),'_Hour_',datestr(now, 'HH-MM-
SS'),'_Person_',int2str(n_persona),'_Activity_',int2str(actividad)));
archivo=(strcat(cabecera,'log.dat'));

%abrimos el archivo donde guardaremos los datos
filename = fopen(archivo, 'w');

%escribimos los nombres de los datos que vamos a almacenar
fprintf(filename, 'timestamp,time,nodo,ax,ay,az,sec\r\n');

%_____________________________%

%inicialización del Freescale
disp('Inicializamos el Freescale.');
OpenPort('COM3');

%enviamos una u para activar los nodos
fprintf(s, 'u');

%se espera n_min*60-4 segundos. Se pone -4 para que la base Freescale
%reciba la orden de parar el ECO Node 4 segundos antes de que el contador
%del ECO Nodo llegue a la cifra de n_min minutos.
pause(60*n_min-4)

%envío la orden al Freescale para que empieze a mandar mensajes a los nodos
%para que paren de enviar
fprintf(s, 'o');
pause(4.5)
fprintf(s, 'o');
fprintf(s, 'o');
fprintf(s, 'o');
fprintf(s, 'o');

%cerramos el puerto
ClosePort();

%cerramos el archivo de texto
fclose(filename)




%ejecutamos una funcion que eliminará las redundancia de datos que pueda
%haber para el caso que queramos enviar varias veces el mismo dato desde
%los ECO Nodes para el caso de transmisiones con interferencias
delete_redundance




function OpenPort(ComPort)

s=serial(ComPort);

set(s,'BaudRate',115200);
%set(s,'Terminator','CR');
set(s,'Timeout',3);
```

```
set(s, 'ReadAsyncMode', 'continuous');
%s.BytesAvailableFcnMode = 'terminator';

s.BytesAvailableFcnCount = 50;
s.BytesAvailableFcnMode = 'byte';
s.BytesAvailableFcn = {@mydispcallback_file};
% s.BytesAvailableFcn = {@pepito};

fopen(s);
disp('puerto abierto');

end

function ClosePort()

%Close ComPort
fclose(s);
disp('puerto cerrado')
end

%esta función se ejecuta automaticamente cada vez que recibimos un string
%de datos por el puerto serie
function mydispcallback_file(obj, event)
    char line;
    %si tenemos datos nuevos...

    if (obj.BytesAvailable >  0)

        nData = obj.BytesAvailable; %contamos con que sea aleatorio
        line = fread(obj,nData);    %linea de datos nuevos que hemos obtenido
        line=line';
        count=nData;
        %inicializacion de variables
        i=0;
        verificacion=0;
        index=1;
        %token_last son los datos que nos han sobrado la vez anterior
        %concatenamos los datos sobrantes con los datos nuevos
        line=cat(2,token_last,line);

        while verificacion==0

            token=[];
            remain=[];
            for i=1:1:length(line)-13

                if (line(i)=='F')&&(line(12+i)=='F')&&(line(i+13)==13)
%                     disp('vamos bien')
                    line;
                    token=line(i:i+12);
                    remain=line(i+13:end);

                    break;
                end
                if i==length(line)-13
%                     disp('contador superado')
                    line;
                    token=[];
                    remain=line(length(line)-13+1:end);
                    break;

                end
            end
```

```
            line=token;
            %en el caso de que el numero de datos de "remain" sea inferior a
            %15, quiere decir que en "remain" no hay datos suficientes como
            %para formar un string de datos completo. Por tanto, para
            %completar lo que falta, hemos de esperar a la siguiente trama
            %de datos. "remain" pasará a ser token_last
            if numel(remain)<15
                %marcamos que se ha cumplido la condicion, para no volver a
                %entrar en el bucle
                verificacion=1;
            end

            %estos 3 "if" comprueban que la trama de datos es una trama
            %correcta de datos del acelerómetro del ECO Node. Para eso mira
            %que tenga la longitud adecuanda, y que el primer y untimo dato
            %sea una "F"
            if ((numel(line)==13)&&(line(1)=='F')&&(line(13)=='F'))

                %primero de todo se verifica el CRC de base-PC
                suma=0;
                for i=2:1:11
                    suma=suma+line(i);
                end
                while(suma>255)
                    suma=suma-256;
                end

                if suma==line(12)*1  %si se cumple esta condicion no hay error
base-PC
                    %ahora se verifica el CRC de nodo-base
                    suma=0;
                    for i=2:1:9
                        suma=suma+line(i);
                    end
                    while(suma>255)
                        suma=suma-256;
                    end
%                     suma=suma-bytes(2);
%                     if suma<0
%                         suma=suma+256;
%                     end
%
                    if suma==line(11)*1-1
                        for i=1:1:13
                            if line(i)==14
                                line(i)=13;
                            end
                        end
                        suma=line(11)*1;
                    elseif suma==line(11)*1
                %                         line
                    %valores del acelerometro
                    muestra_x = 256*line(2) + line(3);
                    muestra_y = 256*line(4) + line(5);
                    muestra_z = 256*line(6) + line(7);
                    %numero de secuencia en formato texto
                    seq= int2str(256*line(8) + line(9));
                    %numero de secuencia en formato decimal
                    seq2=256*line(8) + line(9);
                    %obtenemos el número de nodo. Restamos 48 ya que el
                    %valor 0 en la tabla ASCII es 48 en decimal.
                    nodo=line(10)-48;
                    %guardamos el valor de la primera sequencia en seq_0.
                    %Es importante ya que no siempre el primer valor de
                    %todos va a ser la sequencia 0, sino que puede variar
                    if((nodo>0)&&(nodo<6))
                        if(inicio==0)
                            inicio=1;
```

```
                            seq_0(nodo)=seq2;
                        end
                    end
                    %verificamos antes de empezar que hemos recibido un
                    %valor de numero de nodo que está entre 1 y 5
                    if (nodo>0)&&(nodo<6)&&(seq2>0)
                        %errores es el numero de de secuencia recibido
                        %menos el valor de la primera secuencia menos
                        %numero de muestras que hemos recibido. Esto para
                        %cada uno de los nodos
                        errores(nodo)=seq2-seq_0(nodo)-n_data(nodo);
                        muestras(nodo)=seq2;
                        muestras_captured(nodo)=seq2-seq_0(nodo);
                        if(muestras_captured(nodo)~=0)

errorrel(nodo)=errores(nodo)/muestras_captured(nodo)*100;
                        end
                        %una vez cada 5 muestra recibidas mostramos por
                        %pantalla como va el muestreo. No lo mostramos
                        %siempre para no enlentecer el proceso

                        visualizar(indice,:)=[seq2 muestra_x muestra_y
muestra_z nodo];
                        indice=indice+1;


                        if nodo==1
                            buffer1=cat(1,[muestra_x muestra_y muestra_z
seq2],buffer1);
                        end
                        if nodo==4
                            buffer2=cat(1,[muestra_x muestra_y muestra_z
seq2],buffer2);
                        end
                        if(e==100)
%                             indice
                            errores
                            n_data
                            muestras_captured
                            errorrel
                            e=0;
                            if(length(buffer1)<=300)
                                axisx1=1:1:length(buffer1)-1;
%                                 numel(visualizar(:,2))
                            else
                                axisx1=length(buffer1)-300:1:length(buffer1)-
1;
                            end

                            if(length(buffer2)<=300)
                                axisx2=1:1:length(buffer2)-1;
%                                 numel(visualizar(:,2))
                            else
                                axisx2=length(buffer2)-300:1:length(buffer2)-
1;
                            end

                            sizee=size(buffer1);
                            if (sizee(2)==4)&&(sizee(1)>200)
                            figure(1)
                            plot(buffer1(1:200,1))
                            hold on
                            plot(buffer1(1:200,2),'r')
                            plot(buffer1(1:200,3),'g')
                            plot(buffer1(1:200,4),'k')
%                             legend('x','y','z','seq2');
                            hold off
                            end
```

```
                            sizee=size(buffer2);
                            if (sizee(2)==4)&&(sizee(1)>200)
                            figure(2)
                            plot(buffer2(1:200,1))
                            hold on
                            plot(buffer2(1:200,2),'r')
                            plot(buffer2(1:200,3),'g')
                            plot(buffer2(1:200,4),'k')
%                             legend('x','y','z','seq2');
                            hold off
                            end


                            figure(nodo)
                            plot(axisx,visualizar(axisx,2))
                            hold on
                            plot(axisx,visualizar(axisx,3),'r')
                            plot(axisx,visualizar(axisx,4),'g')
                            legend('x','y','z');
                            hold off


                        end
                        e=e+1;
                    end
                    %escribimos en el documento de texto un string con los
                    %parámetros temporales y los datos
                    if((nodo>0)&&(nodo<6))
                        fprintf(filename, datestr(now, 'yyyymmddHHMMSS'));
                        fprintf(filename, ',');
                        fprintf(filename, datestr(now, 'MMSS'));
                        fprintf(filename, ',');
                        fprintf(filename, char(line(10)));
                        fprintf(filename, ',');
                        fprintf(filename, int2str(muestra_x));
                        fprintf(filename, ',');
                        fprintf(filename, int2str(muestra_y));
                        fprintf(filename, ',');
                        fprintf(filename, int2str(muestra_z));
                        fprintf(filename, ',');
                        fprintf(filename, seq);
                        fprintf(filename, '\r\n');
                    %incrementamos el numero de datos que hemos recibido
                    %correctamente

                        n_data(nodo)=n_data(nodo)+1;
                    end

                     else
                        error_nodo_base=error_nodo_base+1;
                        fprintf('Error de CRC tx nodo-base = %d tx base-pc =
%d\r',error_nodo_base, error_base_pc)
% % % %                           suma
% % % %                           bytes(11)*1
% % % %                           bytes*1

                    end
                else
                    error_base_pc=error_base_pc+1;
                    fprintf('Error de CRC tx nodo-base = %d tx base-pc =
%d\r',error_nodo_base, error_base_pc)
                end
            end
            %asignamos a "line" el resto que hemos obtenimos al dividir la
            %vez anterior con la funcion "strtok". Es un proceso que se
            %repetirá iterativamente hasta que el nuevo remain sea menos
            %que 17
```

```
            line=remain;

        end
        %asignamos el remanente a "token_last" para concatenarlo con la nueva
        %trama de datos que recibamos por el puerto serie
        token_last=remain;
    end
end

function delete_redundance
    %leemos el archivo de datos donde hemos guardado los datos anteriormente
    M = csvread(archivo,1,1)

    [rows,col]=size(M)
    %creamos una matriz con los siguientes datos: nº de ECO Node(nº
    %de columna), la sequencia(del ECO Node) de la última muestra adquirida, y
la
    %posicion de la sequencia en new_M. Es para 5 ECO Nodes
    Seq=[0 0 0 0 0;     %nº de seq
         1 1 1 1 1];    %posicion en new_M de la sequencia
    i=2;
    %asignamos un valor inicial a new_M
    new_M=[0 0 0 0 0 1];
    while i<=rows   %mientras no hagamos procesado todos los datos de "M"
        if (M(i,2)>0)&&(M(i,2)<6)   %si el nº de ECO es entre 0 y 5
            n_eco = M(i,2);         %nº de eco del dato que tratamos
            n_seq_eco = M(i,6);     %nº de seq de ese dato
            i_new_M = Seq(2,n_eco); %i q ocupa en new_M
            if(n_seq_eco == Seq(1,n_eco))     %redundancia de datos!
                new_M(i_new_M,:) = M(i,:);    %sobreescribimos en new_M
            else                              %si no hay redundancia
                %concatenamos el dato nuevo con los datos que ya teníamos
                new_M = cat(1, new_M, M(i,:));
                [rows2,col]=size(new_M);
                %actualizamos datos de Seq
                Seq(1,n_eco) = n_seq_eco;
                Seq(2,n_eco) = rows2;
            end
        end
        i=i+1;
    end
    %esta parte de la funcion guardará en diferentes archivos ".mat" las
    %muestras de cada ECO Node
    [rows2,col]=size(new_M)
    archivo_mat=(strcat(cabecera,'log.mat'));
    %guardamos un archivo ".mat" con todos los datos. Posteriormente
    %guardaremos en archivos separados los datos de cada nodo
    save(archivo_mat,'new_M');
    for i=1:1:5
        %creamos un vector binario, que nos indica donde hay un dato del
        %nodo "i"
        sel=new_M(:,2)==i;
        %creamos el nombre del archivo donde guardaremos los datos de cada
        %nodo
        archivo_sep=strcat(cabecera,'_Nodo_',int2str(i),'.mat');
        %seleccionamos los datos de nuestro vector
        new_M_separate=new_M(sel,:)
        %guardamos los datos obtenidos del nodo "i"
        save(archivo_sep,'new_M_separate');
        [r,c]=size(new_M_separate)
        %sacamos por pantalla los datos de numero de datos emitido por los
        %nodos, el numero de datos recibido, y el tanto por ciento de error
        if r>1
            n_sequencias=new_M_separate(end,6)-new_M_separate(2,6)
            muestras_received=(numel(new_M_separate(:,6))-1)
            error=(n_sequencias-(numel(new_M_separate(:,6))-
1))/n_sequencias*100
        end
```

```
    end


end


end
```

Script en Matlab para validar los detectores de actividad

Este script tiene el objetivo de imitar el proceso del sistema final para validar los detectores de actividad. Para ello se introducen dentro del sistema las muestras obtenidas en el script anterior para medir la tasa de error de cada algoritmo de detección y sus umbrales.[15]

```
function main
clear all;
close all;

%variables globales
global filename;    % fichero de log
global s;           % puerto serie

global buffer_byte;      % buffer de los bytes del puerto série
global p_one_byte;       % puntero que indica el inico de los bytes en el
buffer
global p_last_byte;      % puntero que indica el final de los bytes en el
buffer
global NBUFFERBYTE;

global buffer_sample;
global p_one_sample;
global p_last_sample;
global NBUFFERSAMPLE;
global token_last;

global Last_sample;

global NPACKET;
global new_packet;

global datos;

global varianza;
global i_varianza;

global opcion;

global errors;
global ERRORS_PERMISED;
global MAX_STRING_ERROR_PERMISED;

global muneca_colgante;
global activity;
%_____________________________________%

%inicialización de variables
n_min=9;
NBUFFERBYTE=512;
buffer_byte=zeros(1,NBUFFERBYTE);
p_one_byte=1;
p_last_byte=1;
NBUFFERSAMPLE=256;
```

[15] Este script lleva el nombre de pfc.m

```
buffer_sample=zeros(NBUFFERSAMPLE,6,10);
p_one_sample=ones(1,10);
p_last_sample=ones(1,10);
token_last=[];

Last_sample=zeros(1,10);

NPACKET=100;     %tamaño de datos con el que empaquetaremos las muestras para
filtrar y posteriormente procesar
new_packet=zeros(1,10);     %vector con 10 campos correspondientes a cada uno
de los nodos. Indican si hay en el buffer de cada nodo un número suficiente de
muestras (NPACKET)
datos=[];

varianza=[];
i_varianza=ones(1,10);

opcion=1;

errors=zeros(2,10);
ERRORS_PERMISED = 30;
MAX_STRING_ERROR_PERMISED = 10;

muneca_colgante=1;
activity=1;
opcion=1;

%_____________________________%

%pedimos los datos de la persona, el estado de actividad y el numero de
%minutos que durará el experimento
n_persona='';
actividad='';
% n_min=input('¿Cuantos minutos durará el experimento?\n');
n_min_str=int2str(n_min);   %número de minutos en formato texto

%creamos el nombre del archivo donde vamos a guardar los datos nuevos
cabecera=(strcat('Date_',datestr(now, 29),'_Hour_',datestr(now, 'HH-MM-
SS'),'_Person_',int2str(n_persona),'_Activity_',int2str(actividad)));
archivo=(strcat(cabecera,'log.dat'));

%abrimos el archivo donde guardaremos los datos
filename = fopen(archivo, 'w');

%escribimos los nombres de los datos que vamos a almacenar
fprintf(filename, 'timestamp,time,nodo,ax,ay,az,sec\r\n');

%_____________________________%

%inicialización del Freescale
disp('Inicializamos el Freescale.');
% % % OpenPort('COM8');

%enviamos una u para activar los nodos
% % % fprintf(s, 'u');

%_____________________________%

%seleccion de opciones

opcion=2;

M_1=150000;
M_2=6;
C_1=150000;
C_2=6;

samples_muneca_total=[];
```

```
samples_colgante_total=[];
for ind=1:1:4
ind
    [s_m_t,s_c_t]=findsamples(ind);
    s_m_t(M_1,M_2,1)=0;
    s_c_t(C_1,C_2,1)=0;

    samples_muneca_total=cat(3, samples_muneca_total,s_m_t);
    samples_colgante_total=cat(3, samples_colgante_total,s_c_t);
    size(samples_muneca_total)
    size(samples_colgante_total)

end
% plot(samples_muneca_total(:,:,4))
%     pause
disp('Tenemos un archivo con todas las muestras para su procesado para el
nivel de actividad seleccionado')
size(samples_muneca_total)

%_____________________________%
%este codigo se debera quitar!!!!!!!
for activity=1:1:4
    for muneca_colgante=1:1:2
        if muneca_colgante==1
            datos=samples_muneca_total(:,:,activity);
        else
            datos=samples_colgante_total(:,:,activity);
        end

%_____________________________%

Hd2=load('Hd2')
Hd(1:1:10)=Hd2.Hd2     %creamos un objeto cualquiera para que matlab entienda
que Hd va a ser de tipo objeto

switch opcion
    case 1
        NVARIANCE=400;
        NVARIANCE_MEDIA=5
        v_m=zeros(1,NVARIANCE);
        v_m_m=zeros(1,NVARIANCE_MEDIA);
        v_m_variances=[];
        buffer_variances=zeros(1,NVARIANCE_MEDIA);


    case 2
        N_MEDIA=400;
        v_m=zeros(1,N_MEDIA);
        v_m_variances=[];
    case 3
    case 4
    otherwise
end


%_____________________________%
    [F,C]=size(datos)
    NBUFFERSAMPLE=F+1;
    buffer_sample=zeros(NBUFFERSAMPLE,6,10);
    p_last_sample=NBUFFERSAMPLE;   %en el caso de querer evaluar todo
%     p_last_sample=50000;    %si solo queremos evaluar un cacho de los datos
    size(buffer_sample(:,1,1))
    size(datos(:,1))
    buffer_sample(1:end-1,1,1)=datos(:,1);
    buffer_sample(1:end-1,2,1)=datos(:,2);
    buffer_sample(1:end-1,3,1)=datos(:,3);
    buffer_sample(1:end-1,4,1)=datos(:,4);
    buffer_sample(1:end-1,5,1)=datos(:,5);
```

```
    buffer_sample(1:end-1,6,1)=datos(:,6);
    disp('este plot que viene a continuacion son los datos que vamos a
procesar ahora mismo')
%     plot(buffer_sample(:,:,1))

%_____________________________%
    %esto se tiene que quitar en condiciones normales,ya que indica que hay
    %un paquete nuevo y esto solo lo pudee decir la funcion
    %packet_generator
    new_packet(1,1)=1;

%aquí va el programa principal
while(1)
%     comentamos esta linea ya que solo se va a utilizar para simulaciones
%     donde cojamos los archivos directamente de los .log. Si lo único que
%     queremos es medir la precisión de los detectores, no necesitamos esto
%     ya que insertaremos lo datos directnte en el buffer.
%     datos = simulation_serial_port(datos);
%     packet_generator;


    % en el bucle se hace nodo a nodo todo el tratamiento de la señal,
    % desde el filtro preliminar paso bajo, hasta el detector de movimiento
    for nodo=1:1:10
        if(new_packet(1,nodo)==1)
            %primero de todo, analizamos el número de errores

            %filtrado paso bajo
            [nodo, Hd(nodo), x_f, y_f, z_f]=low_pass_filter(nodo,Hd);
%             plot(x_f)


            %realizamos el método de predicción
            switch opcion
                case 1
                    disp('Método del Cálculo de la Varianza')
                    %v=detector1(nodo, x_f, y_f, z_f);
                    [v, v_m, v_m_m, v_m_variances]=detector1_(nodo, x_f, y_f,
z_f, v_m, v_m_m, v_m_variances, NVARIANCE, NVARIANCE_MEDIA);



                case 2
                    disp('Método de la Diferencia muestra a muestra')
                    [v_m_variances,v_m]=detector2(nodo, x_f, y_f, z_f,
v_m_variances, v_m, N_MEDIA);

                case 3
                    disp('Método de Redes neuronales')
                    v=detector3(nodo, x_f, y_f, z_f);
                otherwise
            end


%             %almacenados los datos obtenidos del cálculo del movimiento
            new_packet(1,nodo)=0;


        end
    end



 %esto se deberá quitar en condiciones normales, ya que no nos sirve!!!!
 break;


end
```

```
%%Hasta los 2 end's de aqui debajo se debera quitar cuando estemos enc
ondiciones
%%normales!!!!
%aqui recogemos los datos para el estudio estadistico de los metodos de
%deteccion de movimiento

    datosestadisticos(1,muneca_colgante,activity,opcion)=mean(v_m_variances)
    datosestadisticos(2,muneca_colgante,activity,opcion)=var(v_m_variances)


n_archivo_muestras=sprintf('datosestadisticos\\p6_option%d_activity%d_muncolg%
d',opcion,activity,muneca_colgante);
    save(n_archivo_muestras,'v_m_variances');

    end
end
save('datosestadisticos\dastosestadisticosdedeteccion','datosestadisticos')


%fclose(FID);


%_____________________________%
fclose(filename)



function OpenPort(ComPort)

    s=serial(ComPort);

    set(s,'BaudRate',115200);
    %set(s,'Terminator','CR');
    set(s,'Timeout',3);

    set(s, 'ReadAsyncMode', 'continuous');
    %s.BytesAvailableFcnMode = 'terminator';

    s.BytesAvailableFcnCount = 50;
    s.BytesAvailableFcnMode = 'byte';
    s.BytesAvailableFcn = {@mydispcallback_file};
    % s.BytesAvailableFcn = {@pepito};

    fopen(s);
    disp('puerto abierto');

end


function ClosePort()

%Close ComPort
fclose(s);
disp('puerto cerrado')
end

%esta función se ejecuta automaticamente cada vez que recibimos un string
%de datos por el puerto serie
function mydispcallback_file(obj, event)
    char line;

    %si tenemos datos nuevos...
    if (obj.BytesAvailable >  0)
        nData = obj.BytesAvailable;     %contamos con que sea aleatorio
        line = fread(obj,nData)';        %linea de datos nuevos que hemos
obtenido
```

```
        if(p_last_byte + nData > NBUFFERBYTE)
            buffer_byte( p_last_byte : NBUFFERBYTE ) = line( 1 : NBUFFERBYTE -
p_last_byte + 1 );
            buffer_byte( 1 : nData - (NBUFFER - p_last_byte + 1)) = line(
NBUFFERBYTE - p_last_byte + 2 : end);
            p_last_byte = p_last_byte + nData - NBUFFERBYTE;
        else
            buffer_byte( p_last_byte : p_last_byte + nData - 1) = line;
            p_last_byte = p_last_byte + nData;
        end
    end
end



%funcion de simulación de recepción de datos del puerto serie
function datos=simulation_serial_port(datos)

    if numel(datos)>16
        [token,datos] = strtok(datos,10);
        %aquí sacamos los datos que nos interesan
        [basura,resto] = strtok(token,',');
        [time,resto] = strtok(resto,',');
        [nodo,resto] = strtok(resto,',');
        [ax,resto] = strtok(resto,',');
        [ay,resto] = strtok(resto,',');
        [az,resto] = strtok(resto,',');
        [sec,resto] = strtok(resto,',');


        str_bytes(1:18)=zeros(1,17);
        str_bytes(1)='F';
        str_bytes(17)='F';
        str_bytes(18)='\n'

        str_bytes(7)=bitshift(bitand(str2num(ax),65280),-8);
        str_bytes(8)=bitand(str2num(ax),255);

        str_bytes(9)=bitshift(bitand(str2num(ay),65280),-8);
        str_bytes(10)=bitand(str2num(ay),255);

        str_bytes(11)=bitshift(bitand(str2num(az),65280),-8);
        str_bytes(12)=bitand(str2num(az),255);

        str_bytes(3:6)=char(dec2hex(str2num(sec),4));

        str_bytes(16)=nodo;
        %esta parte de código cargará las muestras al buffer_byte
        nData = 18;     %contamos con que sea aleatorio
        line = str_bytes;        %linea de datos nuevos que hemos obtenido

        if(p_last_byte + nData > NBUFFERBYTE)
            buffer_byte( p_last_byte : NBUFFERBYTE ) = line( 1 : NBUFFERBYTE -
p_last_byte + 1 );
            buffer_byte( 1 : nData - (NBUFFERBYTE - p_last_byte + 1)) = line(
NBUFFERBYTE - p_last_byte + 2 : end);
            p_last_byte = p_last_byte + nData - NBUFFERBYTE;
        else
            buffer_byte( p_last_byte : p_last_byte + nData - 1) = line;
            p_last_byte = p_last_byte + nData;
        end
    end
end

function packet_generator

    if(p_one_byte~=p_last_byte)     %comprobamos que tenemos bytes en el
buffer circular
```

```
        if(p_last_byte>p_one_byte)
            bytes=buffer_byte(p_one_byte:p_last_byte-1);
        else
            bytes=cat(2, buffer_byte(p_one_byte:NBUFFERBYTE),
buffer_byte(1:p_last_byte-1));
        end
        bytes=cat(2, token_last, bytes);  %completamos el string de datos de
bytes con los bytes sobrantes de la iteración pasada
        verificacion=0;
        while verificacion==0

            token=[];
            remain=[];
            for i=1:1:length(bytes)-18

                if (bytes(i)=='F')&&(bytes(17+i)=='F')&&(bytes(i+18)==13)
%                     disp('vamos bien')
                    bytes;
                    token=bytes(i:i+17);
                    remain=bytes(i+18:end);

                    break;
                end
                if i==length(bytes)-18
%                     disp('contador superado')
                    bytes;
                    token=[];
                    remain=bytes(length(bytes)-18+1:end);
                    break;

                end
            end

            bytes=token;

            if numel(remain)<18
                %marcamos que se ha cumplido la condicion, para no volver a
                %entrar en el bucle
                verificacion=1;
            end


            %estos 3 "if" comprueban que la trama de datos es una trama
            %correcta de datos del acelerómetro del ECO Node. Para eso mira
            %que tenga la longitud adecuanda, y que el primer y ultimo dato
            %sea una "F"
            if ((numel(bytes)==17)&&(bytes(1)=='F')&&(bytes(17)=='F'))
                    %valores del acelerometro
                    muestra_x = 256*bytes(7) + bytes(8);
                    muestra_y = 256*bytes(9) + bytes(10);
                    muestra_z = 256*bytes(11) + bytes(12);
                    %numero de secuencia en formato texto
                    seq=int2str(hex2dec(char(bytes(3:6))));
                    %numero de secuencia en formato decimal
                    seq2=hex2dec(char(bytes(3:6)));
                    %obtenemos el número de nodo. Restamos 48 ya que el
                    %valor 0 en la tabla ASCII es 48 en decimal.
                    nodo=bytes(16)-48;

                    %como ya hemos cogido los datos de la muestra podemos
                    %eliminar esta muestra del buffer circular
                    if(p_one_byte>NBUFFERBYTE-17+1)
                        p_one_byte=17-NBUFFERBYTE+p_one_byte;
                    else
                        p_one_byte=p_one_byte+17;
                    end

                    %una vez tenemos todos los datos del nodo, comprobamos
```

```
                    %que este dato no venga repetido. En el caso de venir
                    %repetido deberíamos descartarlo
                    [delete_r,
Last_sample]=delete_redundance(nodo,seq2,Last_sample);


                if(delete_r)

                    %escribimos en el documento de texto un string con los
                    %parámetros temporales y los datos

                    fprintf(filename, datestr(now, 'yyyymmddHHMMSS'));
                    fprintf(filename, ',');
                    fprintf(filename, datestr(now, 'MMSS'));
                    fprintf(filename, ',');
                    fprintf(filename, char(bytes(16)));
                    fprintf(filename, ',');
                    fprintf(filename, int2str(muestra_x));
                    fprintf(filename, ',');
                    fprintf(filename, int2str(muestra_y));
                    fprintf(filename, ',');
                    fprintf(filename, int2str(muestra_z));
                    fprintf(filename, ',');
                    fprintf(filename, seq);
                    fprintf(filename, '\r\n');


                    %escribimos el dato en el buffer circular de muestras
                    if(p_last_sample(1,nodo)==NBUFFERSAMPLE)
                        p_last_sample(1,nodo)=1;
                    else
                        p_last_sample(1,nodo)=p_last_sample(1,nodo)+1;
                    end

buffer_sample(p_last_sample(1,nodo),1,nodo)=str2num(datestr(now, 'MMSS'));
                    buffer_sample(p_last_sample(1,nodo),2,nodo)=nodo;
                    buffer_sample(p_last_sample(1,nodo),3,nodo)=muestra_x;
                    buffer_sample(p_last_sample(1,nodo),4,nodo)=muestra_y;
                    buffer_sample(p_last_sample(1,nodo),5,nodo)=muestra_z;
                    buffer_sample(p_last_sample(1,nodo),6,nodo)=seq2;
                end

            end
            %asignamos a "bytes" el resto que hemos obtenimos al dividir la
            %vez anterior con la funcion "strtok". Es un proceso que se
            %repetirá iterativamente hasta que el nuevo remain sea menos
            %que 17
            bytes=remain;

        end
        %asignamos el remanente a "token_last" para concatenarlo con la nueva
        %trama de datos que recibamos por el puerto serie
        token_last=remain;


    %comprobamos si tenemos un numero suficiente de samples en el
    %buffer_samples para generar un nuevo packete de datos para procesar.
    %no solo nos basta tener un numero de packetes determinados en el
    %buffer de cada nodo
    if(p_last_sample(1,nodo)<p_one_sample(1,nodo))
        if((NBUFFERSAMPLE-p_one_sample(1,nodo)+1)+(p_last_sample(1,nodo)-
1)>=NPACKET)
            new_packet(1,nodo)=1;
        else
            new_packet(1,nodo)=0;
        end
    else
        if((p_last_sample(1,nodo)-p_one_sample(1,nodo))>=NPACKET)
```

```
            new_packet(1,nodo)=1;
        else
            new_packet(1,nodo)=0;
        end
    end
    end
end


function [delete_r, Last_sample] = delete_redundance(nodo, seq, Last_sample)
    if(Last_sample(nodo)==seq)
        %muestra duplicada
        delete_r = 0;
    else
        Last_sample(nodo)=seq;
        delete_r = 1;
    end
end

%esta funcion va a encargarse de realizar un filtrado paso bajo de la señal
%de aceleración de los diferentes nodos. El filtrado se va a realizar
function [nodo, H, x_f, y_f, z_f]=low_pass_filter(nodo,Hd)

    if Hd(nodo).PersistentMemory==0
        Hd2=load('Hd2');
        Hd2=Hd2.Hd2;
        Hd(nodo)=Hd2;
        Hd(nodo).PersistentMemory=1;
        disp('Nuevo filtro actualizado')
    end

    H=Hd(nodo);


    %se cargan los datos de entrada del filtro pb
    if(p_one_sample(1,nodo)>NBUFFERSAMPLE-NPACKET+1);

data=cat(1,buffer_sample(p_one_sample(1,nodo):NBUFFERSAMPLE,:,nodo),buffer_sam
ple(1:p_one_sample(1,nodo)-NBUFFERSAMPLE+NPACKET-1,:,nodo));
        p_one_sample(1,nodo)=p_one_sample(1,nodo)-NBUFFERSAMPLE+NPACKET;
    else
        data=buffer_sample(p_one_sample(1,nodo):p_last_sample(1,nodo)-
1,:,nodo);
        p_one_sample(1,nodo)=p_one_sample(1,nodo)+NPACKET;
    end

    %en este punto se va a analizar los errores que se obtienen en recepción.
    %En caso de tener un error muy elevado, se debe avisar al usuario para
    %que tome las medidas oportunas
    [errors,max_error,total_errors]=error_detection(nodo,errors,data(:,6));
    if max_error>MAX_STRING_ERROR_PERMISED
        disp('Ráfaga de errores no permitida. Comprobar cobertura
inalámbrica')
    end
    if total_errors>ERRORS_PERMISED
        disp('Demasiadas pérdidas en recepción')
    end



    %se filtran los 3 ejes de coordenadas del acelerómetro
    x_f = filter(Hd(nodo),data(:,3));
    y_f = filter(Hd(nodo),data(:,4));
    z_f = filter(Hd(nodo),data(:,5));


    x_f = data(:,3);
    y_f = data(:,4);
```

```
    z_f = data(:,5);


end


%esta función tiene el objetivo de detectar el número de muestras que se
%está perdiendo en recepción para avisar al usuario del problema.
function [errors,max_error,total_errors]=error_detection(nodo,errors,data)
    vector=cat(2,data(1)-errors(2,nodo),(data(2:end)-data(1:end-1))');
    errors(2,nodo)=data(end);
    max_error=max(vector)-1;
    total_errors=sum(vector-1);
end


function [v, v_m, v_m_m,v_m_variances]=detector1_(nodo, x_f, y_f, z_f, v_m,
v_m_m, v_m_variances, NVARIANCE, NVARIANCE_MEDIA)

    disp('Estamos dentro de detector1_')
    figure(1)

    figure(3)


    s=(sqrt(x_f.^2+y_f.^2+z_f.^2))';
    for i=length(s):-1:1
        i
        if s(i)<1000
        else
            i
            s(i)
            s=s(1:i);
            break;
        end
        if i==1
            s=s(1:1)
        end
    end

    plot(s)

    %modificacion de prueba se debe quitar!!!!!!!!!!!
%     s=s(1:500);

    size(s)
    size(v_m)
    v=cat(2, s, v_m)
    figure(1)
    plot(v)
    disp('Estos son los datos que vamos a empezar a procesar')


    for i=1:1:length(s)
        v=v(1:end-1);
        variance=var(v(end-NVARIANCE+1:end));
        v_m_m=cat(2,variance,v_m_m(1:end-1));
        v_m_variances=cat(2,v_m_variances,mean(v_m_m));

        frase=sprintf('Procesados %d datos de %d. Grado de actividad %d de 4
estados. Opcion %d. Procesando datos de la ',i,length(s),activity,opcion);
        if muneca_colgante==1
            frase=cat(2,frase,'muñeca. Queda aun el colgante para este grado
de actividad');
        else
            frase=cat(2,frase,'colgante.')
        end
        disp(frase)
```

```
        if i>NVARIANCE+NVARIANCE_MEDIA

        end
    end


end

%este modo de detección de movimiento, se va a fijar en la diferencia
%muestra a muestra
function [v_m_variances, v_m]=detector2(nodo, x_f, y_f, z_f, v_m_variances,
v_m, N_MEDIA)

% % % %     x=(abs(x_f(1:end-1)-x_f(2:end)));
% % % %     y=(abs(y_f(1:end-1)-y_f(2:end)));
% % % %     z=(abs(z_f(1:end-1)-z_f(2:end)));
    v=(sqrt(x_f.^2+y_f.^2+z_f.^2))';
    for i=length(v):-1:1
        i
        if v(i)<1000
        else
            i
            v(i)
            v=v(1:i);
            break;
        end
        if i==1
            v=v(1:1)
        end
    end

    v=v(1:end-1)-v(2:end);
    v=abs(v);
    v_m=cat(2,v,v_m);
    if (length(v_m)>N_MEDIA)
        for i=1:1:length(v_m)-N_MEDIA+1
            v_m_variances=cat(2, mean(v_m( end - i + 2 - N_MEDIA : end - i + 1
)), v_m_variances );
            frase=sprintf('Procesados %d datos de %d. Grado de actividad %d de
4 estados. Opcion %d. Procesando datos de la ',i,length(v),activity,opcion);
            if muneca_colgante==1
                frase=cat(2,frase,'muñeca. Queda aun el colgante para este
grado de actividad');
            else
                frase=cat(2,frase,'colgante.')
            end
            disp(frase)


        end
    end

end


function [samples_muneca_total,samples_colgante_total]=findsamples(activity)
    listamuestras=dir('bateriademuestras');
    j=1;
    list_ok_muneca=[];
    list_ok_colgante=[];


    samples_muneca_total=[];
    samples_colgante_total=[];
    %matrix_pbody_nodes(:,1)=muñeca
    %matrix_pbody_nodes(:,2)=colgante
    %matrix_pbody_nodes(x,:) x número de persona
```

```
    matrix_pbody_nodes=[
        5   4;
        5   4;
        5   4;
        5   4;
        1   4;
        4   1;
        4   1;
        1   4;
        1   4;
        1   4;
        1   4;
        1   4;
        1   4;
        1   4;
        0   0;
        0   0;
        1   4;
        1   4;
        1   4
        ];
    ntotal_person=19;

    %recopilamos la lista de todos los archivos de las muñecas
    for i=1:1:ntotal_person
        %en cada iteración debemos crar un string que contenga la información
        %para localizar el archivo de muestras


s=cat(2,'_Person_',int2str(i),'_Activity_',int2str(activity),'_Nodo_',int2str(
matrix_pbody_nodes(i,1)));

        for z=1:1:size(listamuestras)
            compare=strfind(listamuestras(z).name,s);

            if compare==compare
                list_ok_muneca=strvcat(list_ok_muneca,listamuestras(z).name);
                j=j+1;
                break
            end

        end

    end


    for i=1:1:ntotal_person
        %en cada iteración debemos crar un string que contenga la información
        %para localizar el archivo de muestras


s=cat(2,'_Person_',int2str(i),'_Activity_',int2str(activity),'_Nodo_',int2str(
matrix_pbody_nodes(i,2)));

        for z=1:1:size(listamuestras)
            compare=strfind(listamuestras(z).name,s);

            if compare==compare

list_ok_colgante=strvcat(list_ok_colgante,listamuestras(z).name);
                j=j+1;
                break
            end

        end

    end
```

```
    list_ok_muneca
    list_ok_colgante



    %generamos los ficheros finales
    if list_ok_muneca==list_ok_muneca
        for i=1:1:size(list_ok_muneca(:,1))
            s=list_ok_muneca(i,:);
            while s(end)~='t'
                s=s(1:end-1)
            end
            s=cat(2, 'bateriademuestras\',s);
            open(s);

samples_muneca_total=cat(1,samples_muneca_total,ans.new_M_separate);
        end
    end
    if list_ok_colgante==list_ok_colgante
        for i=1:1:size(list_ok_colgante(:,1))
            s=list_ok_colgante(i,:);
            while s(end)~='t'
                s=s(1:end-1)
            end
            s=cat(2, 'bateriademuestras\',s);
            open(s);

samples_colgante_total=cat(1,samples_colgante_total,ans.new_M_separate);

        end

    else
        samples_muneca_total=0;
        samples_colgante_total=0;
    end

    sizeofmuneca=size(samples_muneca_total)

    sizeofcolgante=size(samples_colgante_total)
end


end
```

El código del script pfc.m da como resultado el valor de la actividad de 3 bloques de muestras. Muestras de actividad baja, media y alta. Una vez obtenidas los valores de actividad el script siguiente analizará esos valores para obtener para cada algoritmo el valor de los umbrales óptimos y la tasa de error resultante.

A continuación se muestra el script prova.m, que será el encargado de medir esos valores y sacar la tasa de error.

```
close all
clear all

v_m_variances_tasa=[];
colores=['c' 'r' 'g' 'b' 'c' 'r' 'g' 'b']
directorios=dir('datosestadisticos')
for opcion=1:1:2
    for muneca_colgante=1:1:2
        for activity=1:1:4


n_archivo_muestras=sprintf('datosestadisticos\\p6_option%d_activity%d_muncolg%
d.mat',opcion,activity,muneca_colgante);
```

```
            for i=1:1:length(directorios)

compare=strfind(directorios(i).name,n_archivo_muestras(19:end));
                if compare==compare
                    load(n_archivo_muestras, 'v_m_variances');
                    if length(v_m_variances)>500
                    v_m_variances=v_m_variances(1,500:end); %eliminamos las
primeras muestras, por la distorsión del filtro paso bajo y el procesamiento.
                    end
                    activity
                    [A,B]=size(v_m_variances_tasa)

v_m_variances_tasa(1:length(v_m_variances),activity)=v_m_variances'

long_muestras_activity(opcion,muneca_colgante,activity)=length(v_m_variances);
%
v_m_variances_tasa=cat(2,v_m_variances_tasa,v_m_variances')
                    size(v_m_variances_tasa)
                    disp('v_m_variances_tasa')
                    size(v_m_variances)

                    figure(1)
                    hold on
                    plot(v_m_variances,colores(activity))
                    hold off


                    variance=var(v_m_variances)
                    media=mean(v_m_variances)

                    disp('funcion de gauss')
                    x=-1000000:1:1500000;
                    gauss=(1/(sqrt(variance)*sqrt(2*pi)))*exp((-1/2)*((x-
media)/sqrt(variance)).^2);
                    figure(2)
                    hold on
                    plot(x,gauss,colores(activity))
                    hold off
%                     pause
                    maximo=6*10^6;
                    if(activity==4)
                        maximo=max(v_m_variances)

                    end
                    x=0:1000:maximo;



                    h=hist(v_m_variances,x);
                    figure(4)
                    hold on
                    size(x)
                    size(h)
                    if size(x)==size(h)
                    plot(x,h,colores(activity));
                    end
%                     if activity==4
%                         v=axis;
%                     end
%                     axis(v)
                    axis([0 1000000 0 5000])

                    hold off

%                     pause
                    break;
                end
            end
```

```
        end

        %codigo que calcula la tasa de error y también los niveles
        %óptimos.
        if (v_m_variances_tasa)==(v_m_variances_tasa)
            for activity=2:1:4

n_muestras_activity(activity)=length(v_m_variances_tasa(:,activity))

            end

%         long_muestras_activity
%         pause
  v_m_variances_tasa(1:long_muestras_activity(opcion, muneca_colgante, 4),4)

        length( v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 4),4))
%   pause
        v_m_variances_tasa(:,4)
        length(v_m_variances_tasa(:,4))
%         pause

        cadacuanto=2
        for i=1:1:2000/cadacuanto
% % % % %         cadacuanto=100
% % % % %         for i=1:1:1000000/cadacuanto


nivel1_errores(i,2)=sum(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 2),2)>(i*cadacuanto))/long_muestras_activity(opcion,
muneca_colgante, 2);



nivel1_errores(i,3)=sum(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 3),3)<(i*cadacuanto))/long_muestras_activity(opcion,
muneca_colgante, 3);



nivel1_errores(i,4)=sum(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 4),4)<(i*cadacuanto))/long_muestras_activity(opcion,
muneca_colgante, 4);

nivel2_errores(i,2)=sum(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 2),2)>(i*cadacuanto))/long_muestras_activity(opcion,
muneca_colgante, 2);

nivel2_errores(i,3)=sum(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 3),3)>(i*cadacuanto))/long_muestras_activity(opcion,
muneca_colgante, 3);

nivel2_errores(i,4)=sum(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 4),4)<(i*cadacuanto))/long_muestras_activity(opcion,
muneca_colgante, 4);

        end
%             nivel1_errores

            figure(8)

plot((1:1:length(nivel1_errores(:,2)'))*cadacuanto,nivel1_errores(:,2)',colore
s(1))
            hold on
```

```
plot((1:1:length(nivel1_errores(:,3)'))*cadacuanto,nivel1_errores(:,3)',colore
s(2))

plot((1:1:length(nivel1_errores(:,4)'))*cadacuanto,nivel1_errores(:,4)',colore
s(3))
            hold off

            figure(9)

plot((1:1:length(nivel2_errores(:,2)'))*cadacuanto,nivel2_errores(:,2)',colore
s(1))
            hold on

plot((1:1:length(nivel2_errores(:,3)'))*cadacuanto,nivel2_errores(:,3)',colore
s(2))

plot((1:1:length(nivel2_errores(:,4)'))*cadacuanto,nivel2_errores(:,4)',colore
s(3))
            hold off


        for i=1:1:length(nivel1_errores(:,4))
            nivel1_errores(i,5)=sum(nivel1_errores(i,2:4));
        end

        for i=1:1:length(nivel2_errores(:,4))
            nivel2_errores(i,5)=sum(nivel2_errores(i,2:4));
        end
        figure(8)
        hold on
        plot((1:1:length(nivel1_errores(:,5)))*cadacuanto,nivel1_errores(:,5))
        hold off
        disp('En la figura 8 podemos ver el error medio que habría en función
de el valor que le pongamos al nivel1')



        figure(9)
        hold on
        plot((1:1:length(nivel2_errores(:,5)))*cadacuanto,nivel2_errores(:,5))
        hold off
        disp('En la figura 9 podemos ver el error medio que habría en función
de el valor que le pongamos al nivel2')

        [Y1,I1]=min(nivel1_errores(:,5));
        [Y2,I2]=min(nivel2_errores(:,5));
        s=sprintf('El nivel 1 óptimo estará a la altura de %d. El nivel 2
óptimo estará a %d. La probabilidad de error es: %d para nivel 1 y %d para
nivel 2.',I1*cadacuanto, I2*cadacuanto, (Y1)*100, Y2*100);
        disp(s)

        figure(1)
        hold on
        axis
        I1=ones(1,ans(2))*I1*cadacuanto;
        I2=ones(1,ans(2))*I2*cadacuanto;
        plot(I1,'k')
        plot(I2,'k')
        hold off
        axis([ans(1) ans(2) ans(3) I2(1)*2]);


        %calculo de las diferentes probabilidades de error en funcion del
        %grado de actividad
```

```
        tasa_de_error(1:long_muestras_activity(opcion, muneca_colgante,
2),2,muneca_colgante)=v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 2),2)>I1(1);
        tasa_de_error(1:long_muestras_activity(opcion, muneca_colgante,
2),2,muneca_colgante)=tasa_de_error(1:long_muestras_activity(opcion,
muneca_colgante,
2),2,muneca_colgante)+(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 2),2)>I2(1));


        tasa_de_error(1:long_muestras_activity(opcion, muneca_colgante,
3),3,muneca_colgante)=v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 3),3)<I1(1);
        tasa_de_error(1:long_muestras_activity(opcion, muneca_colgante,
3),3,muneca_colgante)=tasa_de_error(1:long_muestras_activity(opcion,
muneca_colgante,
3),3,muneca_colgante)+(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 3),3)>I2(1));


        tasa_de_error(1:long_muestras_activity(opcion, muneca_colgante,
4),4,muneca_colgante)=v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 4),4)<I1(1);
        tasa_de_error(1:long_muestras_activity(opcion, muneca_colgante,
4),4,muneca_colgante)=tasa_de_error(1:long_muestras_activity(opcion,
muneca_colgante,
4),4,muneca_colgante)+(v_m_variances_tasa(1:long_muestras_activity(opcion,
muneca_colgante, 4),4)<I2(1));

        close(figure(10))
        figure(10)

        hold on
        for i=1:1:4
        plot(tasa_de_error(:,i,muneca_colgante),colores(i))
%         pause
        end
        hold off


error(2,muneca_colgante)=sum(tasa_de_error(1:long_muestras_activity(opcion,
muneca_colgante, 2),2,muneca_colgante))/long_muestras_activity(opcion,
muneca_colgante, 2)*100;

error(3,muneca_colgante)=sum(tasa_de_error(1:long_muestras_activity(opcion,
muneca_colgante, 3),3,muneca_colgante))/long_muestras_activity(opcion,
muneca_colgante, 3)*100;

error(4,muneca_colgante)=sum(tasa_de_error(1:long_muestras_activity(opcion,
muneca_colgante, 4),4,muneca_colgante))/long_muestras_activity(opcion,
muneca_colgante, 4)*100  ;
%         error=error*cadacuanto
        s1=sprintf('El error cometido cada nivel de actividad por separado es
para niveles 1,2,3,4 respectivamente %d,%d,%d,%d. Para muneca_colgante=%d',
error(1), error(2),error(3),error(4),muneca_colgante);
        s2=sprintf('La media por tanto es: %d',
mean(error(:,muneca_colgante)));
        disp(s1)
        disp(s2)
        end

        pause
        close(figure(1))
        close(figure(2))
        close(figure(4))
    end
```

```
    for activity=2:1:4

        longitud(activity)=min(long_muestras_activity(opcion,:,activity))

error_producto(1:longitud(activity),activity)=tasa_de_error(1:longitud(activit
y),activity,1).*tasa_de_error(1:longitud(activity),activity,2);

        p_error(activity)=sum(error_producto(activity))/longitud(activity)
    end
    close(figure(11))
    figure(11)
    hold on
    for activity=1:1:4
    plot(error_producto(1:longitud(activity),activity), colores(activity))
    end
    hold off
    p_error
end
```

Script en Matlab del software final

A continuación el script final de la aplicación[16].

```
function main

close all
clear all
%variables globales
global filename;    % fichero de log
global serialport;           % puerto serie

global buffer_byte;      % buffer de los bytes del puerto série
global p_one_byte;       % puntero que indica el inico de los bytes en el
buffer
global p_last_byte;      % puntero que indica el final de los bytes en el
buffer
global NBUFFERBYTE;

global buffer_sample;
global p_one_sample;
global p_last_sample;
global NBUFFERSAMPLE;
global token_last;

global Last_sample;

global NPACKET;
global new_packet;

global datos;

global varianza;
global i_varianza;



global errors;
global ERRORS_PERMISED;
global MAX_STRING_ERROR_PERMISED;
global error_global;
global error_nodo_base;
```

[16] Script llamado pfc_final.m

```
global error_base_pc;


global muneca_colgante;
global activity;
global opcion;

global buffer_results;
global n_buffer_results;

global data_handles;

global nodo_muneca;
global nodo_colgante;
global opcion_n_nodo;

global muneca_activity;
global colgante_activity;
global cont_activity;
global CONT_ACTIVITY;
global estado_feedback;

global semaforoverde;
global semafororojo;
global semaforoamarillo;
global semaforoblanco;
global data_handles_semaforo;
global f_semaforo;



%_____________________________%

%inicialización de variables
n_min=9;
NBUFFERBYTE=512;
buffer_byte=zeros(1,NBUFFERBYTE);
p_one_byte=1;
p_last_byte=1;
NBUFFERSAMPLE=256;
buffer_sample=zeros(NBUFFERSAMPLE,6,10);
p_one_sample=ones(1,10);
p_last_sample=ones(1,10);
token_last=[];

Last_sample=zeros(1,10);

NPACKET=100;     %tamaño de datos con el que empaquetaremos las muestras para
filtrar y posteriormente procesar
new_packet=zeros(1,10);     %vector con 10 campos correspondientes a cada uno
de los nodos. Indican si hay en el buffer de cada nodo un número suficiente de
muestras (NPACKET)
datos=[];

varianza=[];
i_varianza=ones(1,10);

opcion=1;

errors=zeros(2,10);
ERRORS_PERMISED = 30;
MAX_STRING_ERROR_PERMISED = 10;
error_global(1:2,1:10)=zeros(2,10);
error_nodo_base=0;
error_base_pc=0;

muneca_colgante=1;
activity=1;
```

```
opcion=1;
activity_past=1;

buffer_results=[];
n_buffer_results=zeros(1,10);


muneca_activity=1;
colgante_activity=1;
cont_activity(1:4)=0;
CONT_ACTIVITY=3;
nodo_muneca=1;
nodo_colgante=4;
estado_feedback=1;

semaforo=0;
sonido=0;

reiniciando=0;


%_____________________________%
%creamos el nombre del archivo donde vamos a guardar los datos nuevos
n_persona=0;
actividad=0;
cabecera=(strcat('Date_',datestr(now, 29),'_Hour_',datestr(now, 'HH-MM-
SS'),'_Person_',int2str(n_persona),'_Activity_',int2str(actividad)));
archivo=(strcat(cabecera,'log.dat'));

%abrimos el archivo donde guardaremos los datos
filename = fopen(archivo, 'w');

%escribimos los nombres de los datos que vamos a almacenar
fprintf(filename, 'timestamp,time,nodo,ax,ay,az,sec\r\n');

%_____________________________%

%seleccion de opciones
%opciones predeterminadas

f=openfig('menuprincipal11.fig')
data_handles=guihandles(f)

disp('Seleccione las opciones que desee, y pulse el botón Aceptar')
while get(data_handles.pushbutton1,'BackgroundColor')~=[1 1 1]
    pause(3)
    disp('Seleccione las opciones que desee, y pulse el botón Aceptar')
    if get(data_handles.pushbutton7,'BackgroundColor')==[1 1 1]
        break;
    end
end




nodo_muneca_activate=(get(data_handles.checkbox1,'Value'))
nodo_colgante_activate=(get(data_handles.checkbox2,'Value'))


var1=get(data_handles.radiobutton2,'Value')
var2=get(data_handles.radiobutton3,'Value')
if var1==1
    opcion=1
else opcion=2
end

nivel1_muneca=str2double(get(data_handles.edit1,'String'))
nivel2_muneca=str2double(get(data_handles.edit2,'String'))
```

```
nivel1_colgante=str2double(get(data_handles.edit3,'String'))
nivel2_colgante=str2double(get(data_handles.edit4,'String'))
nivel0_muneca=100;
nivel0_colgante=100;


set(data_handles.checkbox1,'Enable','off');
set(data_handles.checkbox2,'Enable','off');
set(data_handles.radiobutton2,'Enable','off');
set(data_handles.radiobutton3,'Enable','off');


if opcion==2
    set(data_handles.edit1,'String',30);
    set(data_handles.edit2,'String',234);
    set(data_handles.edit3,'String',10);
    set(data_handles.edit4,'String',116);
    nivel1_muneca=str2double(get(data_handles.edit1,'String'))
    nivel2_muneca=str2double(get(data_handles.edit2,'String'))
    nivel1_colgante=str2double(get(data_handles.edit3,'String'))
    nivel2_colgante=str2double(get(data_handles.edit4,'String'))
    nivel0_muneca=4.5;
    nivel0_colgante=4.5;

end

CONT_ACTIVITY=160;
CONT_ACTIVITY=str2double(get(data_handles.edit9,'String'))

%_____________________________%

%inicialización del Freescale
disp('Inicializamos el Freescale.');
OpenPort('COM3');

%enviamos una u para activar los nodos
fprintf(serialport, 'u');

%_____________________________%
%inicializacion de variables de cada opcion de procesado
indiceq=0;

Hd2=load('Hd2');
Hd(1:1:10)=Hd2.Hd2;     %creamos un objeto cualquiera para que matlab entienda
que Hd va a ser de tipo objeto


puntos_v=400;
switch opcion
    case 1
        NVARIANCE=50;
        NVARIANCE_MEDIA=20;
        v_m=ones(NVARIANCE,10)*2500;
        v_m_m=zeros(NVARIANCE_MEDIA,10);
        v_m_variances=[0];

    case 2
        N_MEDIA=400;
        v_m=zeros(N_MEDIA-1,10);
        v_m_variances=[];

    case 3
    case 4
    otherwise
end


%_____________________________%
```

```
%aquí va el programa principal
while(1)

    %comprobacion inicial de reiniciar
    if get(data_handles.pushbutton6,'BackgroundColor')==[1 1 1]
        reiniciando=1;
        break;
    end
    if get(data_handles.pushbutton7,'BackgroundColor')==[1 1 1]
        break;
    end

    pause(0.001)
    packet_generator;
    % comprobacion de si se ha presionado el boton "Mostrar semaforo" o el
    % botón "Activar sonido"
    feedback_audiovisual;

    % en el bucle se hace nodo a nodo todo el tratamiento de la señal,
    % desde el filtro preliminar paso bajo, hasta el detector de movimiento
    for nodo=1:1:10

        if
(new_packet(1,nodo)==1)&(((nodo==nodo_muneca)&(nodo_muneca_activate==1))|((nod
o==nodo_colgante)&(nodo_colgante_activate==1)))

            %verificacion de que se están recibiendo los datos correctamente
            %en condiciones normales este codigo no se debe ejecutar
            figure(nodo)
            plot(buffer_sample(:,3,nodo), 'k')
            hold on
            plot(buffer_sample(:,4,nodo),'g')
            plot(buffer_sample(:,5,nodo),'r')
            hold off
            pause(0.01)

            %filtrado paso bajo
            [nodo, Hd(nodo), x_f, y_f, z_f]=low_pass_filter(nodo,Hd);

            %realizamos el método de predicción
            switch opcion
                case 1
                    [v_m, v_m_m]=detector1(nodo, x_f, y_f, z_f, v_m, v_m_m,
NVARIANCE, NVARIANCE_MEDIA);


                case 2

                    v_m=detector2(nodo, x_f, y_f, z_f, v_m, N_MEDIA);

%                     v_m_variances=detector2(nodo, x_f, y_f, z_f);

                case 3
                    disp('Método de Redes neuronales')
                    v=detector3(nodo, x_f, y_f, z_f);

                otherwise

            end


            %a continuación, se visualiza por pantalla como va
            %evolucionando la deteccion
%                 s=sprintf('Datos nodo 1: %d. Datos nodo 4: %d',
n_buffer_results(1,1), n_buffer_results(1,4));
%                 disp(s)
                if n_buffer_results(1,nodo)>puntos_v*2
                    n_muestras_nodo=n_buffer_results(1,nodo);
```

```
buffer_results(1:puntos_v+1,nodo)=buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo);
                    n_buffer_results(1,nodo)=puntos_v+1;
                end
                if n_buffer_results(1,nodo)>puntos_v+100


                    if nodo==nodo_muneca
                        n_muestras_nodo=n_buffer_results(1,nodo);

semilogy(1:1:puntos_v+1,(buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo))','Parent',data_handles.axes1);

vector_n1=nivel1_muneca*ones(size(buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo)))';

vector_n2=nivel2_muneca*ones(size(buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo)))';

line((1:1:puntos_v+1),vector_n1,'Parent',data_handles.axes1,'Color','r');

line((1:1:puntos_v+1),vector_n2,'Parent',data_handles.axes1,'Color',[0 0.498
0]);
                        switch opcion
                            case 1
                                set(data_handles.axes1,'XLim',[0
puntos_v],'YLim',[10 1000000]);
                                break;
                            case 2
                                set(data_handles.axes1,'XLim',[0
puntos_v],'YLim',[1 10000]);
                                break;
                            otherwise

                        end

                    end
                    pause(0.002)
                    if nodo==nodo_colgante
                        n_muestras_nodo=n_buffer_results(1,nodo);

semilogy(1:1:puntos_v+1,buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo)','Parent', data_handles.axes2);

vector_n1=nivel1_colgante*ones(size(buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo)))';

vector_n2=nivel2_colgante*ones(size(buffer_results(n_muestras_nodo-
puntos_v:n_muestras_nodo,nodo)))';

line(1:1:puntos_v+1,vector_n1,'Parent',data_handles.axes2,'Color','r');

line(1:1:puntos_v+1,vector_n2,'Parent',data_handles.axes2,'Color',[0 0.498
0]);
                        switch opcion
                            case 1
                                set(data_handles.axes2,'XLim',[0
puntos_v],'YLim',[10 1000000]);
                                break;
                            case 2
                                set(data_handles.axes2,'XLim',[0
puntos_v],'YLim',[1 10000]);
                                break;
                            otherwise
                        end

                    end
```

```
                end
%             %almacenados los datos obtenidos del cálculo del movimiento
            new_packet(1,nodo)=0;

        end
    end
end


%_____________________________%

%envío la orden al Freescale para que empieze a mandar mensajes a los nodos
%para que paren de enviar
for i=1:1:20
    fprintf(serialport, 'o');
    pause(0.01*i)
end


%cerramos el puerto
ClosePort();


close(f)
f_semaforo
if f_semaforo==f_semaforo
    close(f_semaforo)
end




%cerramos el archivo de texto
fclose(filename)
if reiniciando==1
    close all
    clear all
    pfc_final
end







function OpenPort(ComPort)

    serialport=serial(ComPort);

    set(serialport,'BaudRate',115200);
    %set(s,'Terminator','CR');
    set(serialport,'Timeout',3);

    set(serialport, 'ReadAsyncMode', 'continuous');
    %s.BytesAvailableFcnMode = 'terminator';

    serialport.BytesAvailableFcnCount = 50;
    serialport.BytesAvailableFcnMode = 'byte';
    serialport.BytesAvailableFcn = {@mydispcallback_file};
    % s.BytesAvailableFcn = {@pepito};

    fopen(serialport);
    disp('puerto abierto');

end
```

```
function ClosePort()

%Close ComPort
instrfind
fclose(ans);
disp('puerto cerrado')
end


%esta función se ejecuta automaticamente cada vez que recibimos un string
%de datos por el puerto serie
function mydispcallback_file(obj, event)
    char line;

    %si tenemos datos nuevos...
    if (obj.BytesAvailable >  0)

%         disp('datos nuevos')

        nData = obj.BytesAvailable;     %contamos con que sea aleatorio
        line = fread(obj,nData)';        %linea de datos nuevos que hemos
obtenido
%         [p_one_byte, p_last_byte]
%         nData
%         NBUFFERBYTE
        if(p_last_byte + nData > NBUFFERBYTE)
            buffer_byte( p_last_byte : NBUFFERBYTE ) = line( 1 : NBUFFERBYTE -
p_last_byte + 1 );
            buffer_byte( 1 : nData - (NBUFFERBYTE - p_last_byte + 1)) = line(
NBUFFERBYTE - p_last_byte + 2 : end);
            p_last_byte = p_last_byte + nData - NBUFFERBYTE;
        else
            buffer_byte( p_last_byte : p_last_byte + nData - 1) = line;
            p_last_byte = p_last_byte + nData;
        end
    end
end


function packet_generator
    %     disp('estamos en packet generator')
% % % %      [p_one_byte, p_last_byte]
    if(p_one_byte~=p_last_byte)     %comprobamos que tenemos bytes en el
buffer circular
% % % %                 [p_one_byte, p_last_byte]
        if(p_last_byte>p_one_byte)
            bytes=buffer_byte(p_one_byte:p_last_byte-1);
            p_one_byte=p_last_byte-1;
        else
            bytes=cat(2, buffer_byte(p_one_byte:NBUFFERBYTE),
buffer_byte(1:p_last_byte-1));
            p_one_byte=p_last_byte-1;
            if p_last_byte==1
                p_one_byte=NBUFFERBYTE;
            end
        end
        bytes=cat(2, token_last, bytes);  %completamos el string de datos de
bytes con los bytes sobrantes de la iteración pasada
% % % %                 disp('2');
% % % %                 [p_one_byte, p_last_byte];
        verificacion=0;
        if length(bytes)<15
            verificacion=1;
            remain=bytes;
        end
        if (length(token_last)>0)&&(length(bytes)>0)
            while bytes(1)*1==token_last(end)*1
```

```
%                 disp('Hola')
                bytes;
                bytes=bytes(2:end);
                if length(bytes)==1
                    break;
                end
            end
        end
        while verificacion==0
            token=[];
            remain=[];
            for i=1:1:length(bytes)-13

                if (bytes(i)=='F')&&(bytes(12+i)=='F')&&(bytes(i+13)==13)
%                     disp('vamos bien')
                    bytes;
                    token=bytes(i:i+12);
                    remain=bytes(i+13:end);

                    break;
                end
                if i==length(bytes)-13
%                     disp('contador superado')
                    bytes;
                    token=[];
                    remain=bytes(length(bytes)-13+1:end);
                    break;

                end
            end

            bytes=token;

            if numel(remain)<15
                %marcamos que se ha cumplido la condicion, para no volver a
                %entrar en el bucle
                verificacion=1;
            end


            %estos 3 "if" comprueban que la trama de datos es una trama
            %correcta de datos del acelerómetro del ECO Node. Para eso mira
            %que tenga la longitud adecuanda, y que el primer y ultimo dato
            %sea una "F"
% % % %             bytes*1
            if ((numel(bytes)==13)&&(bytes(1)=='F')&&(bytes(13)=='F'))

% % % %                 %primero de todo se verifica el CRC de base-PC
                suma=0;
                for i=2:1:11
                    suma=suma+bytes(i);
                end
                while(suma>255)
                    suma=suma-256;
                end
                if suma~=bytes(12)*1
                if (suma==bytes(12)*1+1)    %si se cumple esta condicion
                                            %quiere decir que se ha
                                            %modificado un byte de 13 a 14
                    for i=1:1:13
                        if bytes(i)==14
                            bytes(i)=13;
                        end
                    end
                elseif (suma==bytes(12)*1+2)
                    for i=1:1:13
                        if bytes(i)==14
                            bytes(i)=13;
```

```
                            break;
                        end
                    end
                elseif (suma==bytes(12)*1+3)


                    suma=0;
                    for i=2:1:9
                        suma=suma+bytes(i);
                    end
                    while(suma>255)
                        suma=suma-256;
                    end


                      if suma==bytes(11)*1-1
                        for i=1:1:13
                            if bytes(i)==14
                                bytes(i)=13;
                            end
                        end
                      end
                    if suma==bytes(11)*1

                        muestra_x = 256*bytes(2) + bytes(3);
                        muestra_y = 256*bytes(4) + bytes(5);
                        muestra_z = 256*bytes(6) + bytes(7);
                        %numero de secuencia en formato texto
                        %
seq=int2str(hex2dec(char(bytes(3:6))))
                        seq= int2str(256*bytes(8) + bytes(9)); %forma buena

                        %numero de secuencia en formato decimal
                        seq2=256*bytes(8) + bytes(9);          %forma buena
                        %                 seq2=hex2dec(char(bytes(3:6)))

                        %obtenemos el número de nodo. Restamos 48 ya que el
                        %valor 0 en la tabla ASCII es 48 en decimal.
                        nodo=bytes(10)-48;


                        %una vez tenemos todos los datos del nodo, comprobamos
                        %que este dato no venga repetido. En el caso de venir
                        %repetido deberíamos descartarlo
                        [delete_r,
Last_sample]=delete_redundance(nodo,seq2,Last_sample);


                        if(delete_r)

                            %escribimos en el documento de texto un string con
los
                            %parámetros temporales y los datos

                            fprintf(filename, datestr(now, 'yyyymmddHHMMSS'));
                            fprintf(filename, ',');
                            fprintf(filename, datestr(now, 'MMSS'));
                            fprintf(filename, ',');
                            fprintf(filename, char(bytes(10)));
                            fprintf(filename, ',');
                            fprintf(filename, int2str(muestra_x));
                            fprintf(filename, ',');
                            fprintf(filename, int2str(muestra_y));
                            fprintf(filename, ',');
                            fprintf(filename, int2str(muestra_z));
                            fprintf(filename, ',');
                            fprintf(filename, seq);
                            fprintf(filename, '\r\n');
```

```
                            %escribimos el dato en el buffer circular de
muestras
                            if(p_last_sample(1,nodo)==NBUFFERSAMPLE)
                                p_last_sample(1,nodo)=1;
                            else
                                p_last_sample(1,nodo)=p_last_sample(1,nodo)+1;
                            end

buffer_sample(p_last_sample(1,nodo),1,nodo)=str2num(datestr(now, 'MMSS'));
                            buffer_sample(p_last_sample(1,nodo),2,nodo)=nodo;

buffer_sample(p_last_sample(1,nodo),3,nodo)=muestra_x;

buffer_sample(p_last_sample(1,nodo),4,nodo)=muestra_y;

buffer_sample(p_last_sample(1,nodo),5,nodo)=muestra_z;
                            buffer_sample(p_last_sample(1,nodo),6,nodo)=seq2;

                            if(p_last_sample(1,nodo)<p_one_sample(1,nodo))
                                if((NBUFFERSAMPLE-
p_one_sample(1,nodo)+1)+(p_last_sample(1,nodo)-1)>=NPACKET)
                                    new_packet(1,nodo)=1;
                                else
                                    new_packet(1,nodo)=0;
                                end
                            else
                                if((p_last_sample(1,nodo)-
p_one_sample(1,nodo))>=NPACKET)
                                    new_packet(1,nodo)=1;
                                else
                                    new_packet(1,nodo)=0;
                                end
                            end



                        end
                    else
                        error_nodo_base=error_nodo_base+1;
                        fprintf('Error de CRC tx nodo-base = %d tx base-pc =
%d\r',error_nodo_base, error_base_pc)
% % % %                           suma
% % % %                           bytes(11)*1
% % % %                           bytes*1

                    end
                else
                    error_base_pc=error_base_pc+1;
                    fprintf('Error de CRC tx nodo-base = %d tx base-pc =
%d\r',error_nodo_base, error_base_pc)
                end
            end
            %asignamos a "bytes" el resto que hemos obtenimos al dividir la
            %vez anterior con la funcion "strtok". Es un proceso que se
            %repetirá iterativamente hasta que el nuevo remain sea menos
            %que 17
            bytes=remain;


        end
        %asignamos el remanente a "token_last" para concatenarlo con la nueva
        %trama de datos que recibamos por el puerto serie
        token_last=remain;
```

```
    end
end


    function [delete_r, Last_sample] = delete_redundance(nodo, seq,
Last_sample)
        if (nodo>0)&&(nodo<11)
            if(Last_sample(nodo)==seq)
                %             muestra duplicada
                delete_r = 0;
            else
                Last_sample(nodo)=seq;
                delete_r = 1;
            end
        else
            delete_r=0;
        end
    end


%esta funcion va a encargarse de realizar un filtrado paso bajo de la señal
%de aceleración de los diferentes nodos. El filtrado se va a realizar
function [nodo, H, x_f, y_f, z_f]=low_pass_filter(nodo,Hd)
    %el filtro que vamos a usar va a ser un filtro paso bajo de bessel, ya
    %que queremos que la fase sea lo más plana posible.
%     [A, B] = besself(5,0.5);
%     [A,B]=cheby1(10,1,0.5);

%     [B,A]=butter(5,.5);



    if Hd(nodo).PersistentMemory==0
        Hd2=load('Hd2');
        Hd2=Hd2.Hd2;
        Hd(nodo)=Hd2;
        Hd(nodo).PersistentMemory=1;
        disp('Nuevo filtro actualizado')
    end

    H=Hd(nodo);


%     figure(2)
%     freqz(B,A,500)
%     figure(1)

    %se cargan los datos de entrada del filtro pb
    if(p_one_sample(1,nodo)>NBUFFERSAMPLE-NPACKET+1);

data=cat(1,buffer_sample(p_one_sample(1,nodo):NBUFFERSAMPLE,:,nodo),buffer_sam
ple(1:(p_one_sample(1,nodo)-NBUFFERSAMPLE+NPACKET-1),:,nodo));
        p_one_sample(1,nodo)=p_one_sample(1,nodo)-NBUFFERSAMPLE+NPACKET;
    else
        data=buffer_sample(p_one_sample(1,nodo):p_one_sample(1,nodo)+NPACKET-
1,:,nodo);
        p_one_sample(1,nodo)=p_one_sample(1,nodo)+NPACKET;
    end

    %en este punto se va a analizar los errores que se obtienen en recepción.
    %En caso de tener un error muy elevado, se debe avisar al usuario para
    %que tome las medidas oportunas
    [errors,max_error,total_errors]=error_detection(nodo,errors,data(:,6));
    s=sprintf('Nodo %d. Err máx = %d. Nº errores= %d/%d.', nodo, max_error,
total_errors, data(end,6)-data(1,6));
    if total_errors<0
```

```
        disp('Se ha reseteado el contador del ECO Node. No contabilizamos este
error.')
    else
        error_global(1,nodo)=error_global(1,nodo)+total_errors;
        error_global(2,nodo)=error_global(2,nodo)+NPACKET;
        s=cat(2,s,sprintf('Err global %0.2g.',
error_global(1,nodo)/error_global(2,nodo)*100));
    end
    %da total_errors algunas veces -1
    if max_error>MAX_STRING_ERROR_PERMISED
        disp('Ráfaga de errores no permitida. Comprobar cobertura
inalámbrica.')
%         set(data_handles.text14,'String','Rafaga de errores')
%         set(data_handles.text14,'ForegroundColor',[1 0 0])
    elseif total_errors>ERRORS_PERMISED
        disp('Demasiadas pérdidas en recepción.')
%         set(data_handles.text14,'String','Muchas perdidas en recepción')
%         set(data_handles.text14,'ForegroundColor',[1 0 0])
    else
%         set(data_handles.text14,'String','Recepción de muestras fluida')
%         set(data_handles.text14,'ForegroundColor',[0 0 0])
    end
    disp(s)



    %se filtran los 3 ejes de coordenadas del acelerómetro
    x_f = filter(Hd(nodo),data(:,3));
    y_f = filter(Hd(nodo),data(:,4));
    z_f = filter(Hd(nodo),data(:,5));


    x_f = data(:,3);
    y_f = data(:,4);
    z_f = data(:,5);


    %representamos la señal filtrada
%     plot(data(:,3))
%     hold on
%     plot(x_f,'r')
%     hold off

end




%esta función tiene el objetivo de detectar el número de muestras que se
%está perdiendo en recepción para avisar al usuario del problema.
function [errors,max_error,total_errors]=error_detection(nodo,errors,data)
%     data
%     errors(2,nodo)
    vector=cat(2,data(1)-errors(2,nodo),(data(2:end)-data(1:end-1))');
    errors(2,nodo)=data(end);
    max_error=max(vector)-1;
    total_errors=sum(vector-1);
end



%detector de errores bueno
function [v_m, v_m_m]=detector1(nodo, x_f, y_f, z_f, v_m, v_m_m, NVARIANCE,
NVARIANCE_MEDIA)
    %calculo del modulo de la aceleración
    s=(sqrt(x_f.^2+y_f.^2+z_f.^2))';
```

```
    % el bucle for que tenemos a continuación nos servirá para eliminar las
    % muestras que no vamos a procesar
    for i=length(s):-1:1

        if s(i)<1000
        else
            s=s(1:i);
            break;
        end

        if i==1
            s=s(1:1);
        end

    end

    v=cat(2, s, v_m(:,nodo)');

    for i=1:1:length(s)
        v=v(1:end-1);
        variance=var(v(end-NVARIANCE+1:end));
        v_m_m_=cat(1,variance,v_m_m(1:end-1,nodo));
        v_m_m(1:NVARIANCE_MEDIA,nodo)=v_m_m_;

        n_buffer_results(1,nodo)=n_buffer_results(1,nodo)+1;
        buffer_results(n_buffer_results(1,nodo),nodo)=mean(v_m_m(:,nodo));


        actualizar_feedback


    end
    v_m(1:length(v),nodo)=v';

end




%este modo de detección de movimiento, se va a fijar en la diferencia
%muestra a muestra
function v_m=detector2(nodo, x_f, y_f, z_f, v_m, N_MEDIA)
    s=(sqrt(x_f.^2+y_f.^2+z_f.^2))';
    s=abs((s(1:end-1)-s(2:end)));



    v=cat(2,s,v_m(:,nodo)');
    if (length(v)>N_MEDIA)
        for i=1:1:length(v)-N_MEDIA+1

            n_buffer_results(1,nodo)=n_buffer_results(1,nodo)+1;
            buffer_results(n_buffer_results(1,nodo),nodo)=mean(v( end - i + 2
- N_MEDIA : end - i + 1 ));

            actualizar_feedback;
        end
        v_m(1:N_MEDIA-1,nodo)=v(1:N_MEDIA-1);
    end

end


function feedback_audiovisual
    if semaforo==0
        if (get(data_handles.pushbutton4,'BackgroundColor')==[1 1 1])
            disp('Configuramos en semaforo')
% % % %             f_semaforo=openfig('semaforo.fig')
```

```
            f_semaforo=openfig('semaforo2.fig')
            data_handles_semaforo=guihandles(f_semaforo)
            semaforo=1;


            semaforoblanco  = imread('semblanco','bmp');
            semaforoverde   = imread('semverde', 'bmp');
            semaforoamarillo= imread('semamarillo','bmp');
            semafororojo    = imread('semrojo','bmp');
            semaforoblanco  = uint8(semaforoblanco);
            semaforoverde   = uint8(semaforoverde);
            semaforoamarillo= uint8(semaforoamarillo);
            semafororojo    = uint8(semafororojo);

            switch activity
                case 1

Img=image(semaforoblanco,'Parent',data_handles_semaforo.axes1);
                case 2

Img=image(semaforoverde,'Parent',data_handles_semaforo.axes1);
                case 3

Img=image(semaforoamarillo,'Parent',data_handles_semaforo.axes1);
                case 4

Img=image(semafororojo,'Parent',data_handles_semaforo.axes1);
                otherwise
            end

set(data_handles_semaforo.axes1,'Visible','off','YDir','reverse','XLim',get(Im
g,'XData'),'YLim',get(Img,'YData'));
            Img=image(semaforoblanco,'Parent',data_handles_semaforo.axes1);

set(data_handles_semaforo.axes1,'Visible','off','YDir','reverse','XLim',get(Im
g,'XData'),'YLim',get(Img,'YData'));
        end
    end
    if sonido==0
        if (get(data_handles.pushbutton5,'BackgroundColor')==[1 1 1])
            sonido=1;
        end
    end

end


function actualizar_feedback

    %vamos a sacar por pantalla el estado de actividad en el que se
    %encuentra el usuarios
    CONT_ACTIVITY=str2double(get(data_handles.edit9,'String'));
    if nodo==nodo_muneca
        nivel1_muneca=str2double(get(data_handles.edit1,'String'));
        nivel2_muneca=str2double(get(data_handles.edit2,'String'));
        if  buffer_results(n_buffer_results(1,nodo),nodo)<nivel0_muneca
            muneca_activity=1;
        elseif buffer_results(n_buffer_results(1,nodo),nodo)<nivel1_muneca
            muneca_activity=2;
        elseif buffer_results(n_buffer_results(1,nodo),nodo)<nivel2_muneca
            muneca_activity=3;
        else
            muneca_activity=4;
        end
    end
    if nodo==nodo_colgante
        nivel1_colgante=str2double(get(data_handles.edit3,'String'));
        nivel2_colgante=str2double(get(data_handles.edit4,'String'));
```

```
        if buffer_results(n_buffer_results(1,nodo),nodo)<nivel0_colgante
            colgante_activity=1;
        elseif buffer_results(n_buffer_results(1,nodo),nodo)<nivel1_colgante
            colgante_activity=2;
        elseif buffer_results(n_buffer_results(1,nodo),nodo)<nivel2_colgante
            colgante_activity=3;
        else
            colgante_activity=4;
        end
    end



    if (nodo_muneca_activate==1)&&(nodo_colgante_activate==0)
        activity=muneca_activity;
    end
    if (nodo_muneca_activate==0)&&(nodo_colgante_activate==1)
        activity=colgante_activity;
    end
    if (nodo_muneca_activate==1)&&(nodo_colgante_activate==1)
        activity_suma=colgante_activity+muneca_activity;
        if activity_suma<=2
            activity=1;
        elseif activity_suma<=5
            activity=2;
        elseif activity_suma<=7
            activity=3;
        else
            activity=4;
        end
    end


        activity_past=estado_feedback;
        cont_activity
        estado_feedback
        CONT_ACTIVITY
        if estado_feedback == 1

            %actualización de la variable estado_feedback
            for i=1:1:4
                if cont_activity(i)>=CONT_ACTIVITY
                    estado_feedback=i;
                end
            end
            %actualización de los contadores
            if activity ==1
                cont_activity(1:4)=0;
            elseif activity == 2
                cont_activity(1)=0;
                cont_activity(2)=cont_activity(2)+1;
                cont_activity(3:4)=0;
            elseif activity == 3
                cont_activity(2:3)=cont_activity(2:3)+1;
                cont_activity(4)=0;
            elseif activity == 4
                cont_activity(2:4)=cont_activity(2:4)+1;
            end

        elseif estado_feedback == 2

            %actualización de la variable estado_feedback
            for i=1:1:4
                if cont_activity(i)>=CONT_ACTIVITY
                    estado_feedback=i;
                end
            end
            %actualización de los contadores
```

```
            if activity ==1
                cont_activity(1)=cont_activity(1)+1;
                cont_activity(2:4)=0;
            elseif activity == 2
                cont_activity(1:4)=0;
            elseif activity == 3
                cont_activity(1:2)=0;
                cont_activity(3)=cont_activity(3)+1;
                cont_activity(4)=0;
            elseif activity == 4
                cont_activity(1:2)=0;
                cont_activity(3:4)=cont_activity(3:4)+1;
            end

        elseif estado_feedback == 3

            %actualización de la variable estado_feedback
            for i=4:-1:1
                if cont_activity(i)>=CONT_ACTIVITY
                    estado_feedback=i;
                end
            end
            %actualización de los contadores
            if activity ==1
                cont_activity(1:2)=cont_activity(1:2)+1;
                cont_activity(3:4)=0;
            elseif activity == 2
                cont_activity(1)=0;
                cont_activity(2)=cont_activity(2)+1;
                cont_activity(3:4)=0;
            elseif activity == 3
                cont_activity(1:4)=0;
            elseif activity == 4
                cont_activity(1:3)=0;
                cont_activity(4)=cont_activity(4)+1;
            end

        elseif estado_feedback == 4

            %actualización de la variable estado_feedback
            for i=4:-1:1
                if cont_activity(i)>=CONT_ACTIVITY
                    estado_feedback=i;
                end
            end
            %actualización de los contadores
            if activity ==1
                cont_activity(1:3)=cont_activity(1:3)+1;
                cont_activity(4)=0;
            elseif activity == 2
                cont_activity(1)=0;
                cont_activity(2:3)=cont_activity(2:3)+1;
                cont_activity(4)=0;
            elseif activity == 3
                cont_activity(1:2)=0;
                cont_activity(3)=cont_activity(3)+1
                cont_activity(4)=0;
            elseif activity == 4
                cont_activity(1:4)=0;
            end

        end

%             if n_buffer_results(1,nodo)>puntos_v*2
            switch estado_feedback
                case 4,

%                     if cont_activity(activity)>=CONT_ACTIVITY
```

```
                        if activity_past~=4
                            estado_feedback=4;
                            set(data_handles.edit5,'BackgroundColor', [1 0 0])
                            set(data_handles.edit5,'String','Alto')
                            set(data_handles.edit6,'BackgroundColor', [1 1 1])
                            set(data_handles.edit6,'String','')
                            set(data_handles.edit7,'BackgroundColor', [1 1 1])
                            set(data_handles.edit7,'String','')
                            set(data_handles.edit8,'BackgroundColor', [1 1 1])
                            set(data_handles.edit8,'String','')
                            if semaforo==1

Img=image(semafororojo,'Parent',data_handles_semaforo.axes1);

set(data_handles_semaforo.axes1,'Visible','off','YDir','reverse','XLim',get(Im
g,'XData'),'YLim',get(Img,'YData'));
                                    disp('semafororojo')
                            end
                            if sonido==1

                            end
                        end

%                     end
                case 3,
%                     if cont_activity(activity)>=CONT_ACTIVITY

                        if activity_past~=3
                            estado_feedback=3;
                            set(data_handles.edit5,'BackgroundColor', [1 1 1])
                            set(data_handles.edit5,'String','')
                            set(data_handles.edit6,'BackgroundColor', [1 1 0])
                            set(data_handles.edit6,'String','Medio')
                            set(data_handles.edit7,'BackgroundColor', [1 1 1])
                            set(data_handles.edit7,'String','')
                            set(data_handles.edit8,'BackgroundColor', [1 1 1])
                            set(data_handles.edit8,'String','')
                            if semaforo==1

Img=image(semaforoamarillo,'Parent',data_handles_semaforo.axes1);

set(data_handles_semaforo.axes1,'Visible','off','YDir','reverse','XLim',get(Im
g,'XData'),'YLim',get(Img,'YData'));
                                    disp('semafoamarillo')
                            end
                        end
%                     end
                case 2,
%                     if cont_activity(activity)>=CONT_ACTIVITY

                        if activity_past~=2
                            estado_feedback=2;
                            set(data_handles.edit5,'BackgroundColor', [1 1 1])
                            set(data_handles.edit5,'String','')
                            set(data_handles.edit6,'BackgroundColor', [1 1 1])
                            set(data_handles.edit6,'String','')
                            set(data_handles.edit7,'BackgroundColor', [0 1 0])
                            set(data_handles.edit7,'String','Bajo')
                            set(data_handles.edit8,'BackgroundColor', [1 1 1])
                            set(data_handles.edit8,'String','')
                            if semaforo==1

Img=image(semaforoverde,'Parent',data_handles_semaforo.axes1);

set(data_handles_semaforo.axes1,'Visible','off','YDir','reverse','XLim',get(Im
g,'XData'),'YLim',get(Img,'YData'));
                                    disp('semaforoverde')
```

```
                            end
                        end
%                     end
                case 1,
%                     if cont_activity(activity)>=CONT_ACTIVITY

                        if activity_past~=1
                            estado_feedback=1;
                            set(data_handles.edit5,'BackgroundColor', [1 1 1])
                            set(data_handles.edit5,'String','')
                            set(data_handles.edit6,'BackgroundColor', [1 1 1])
                            set(data_handles.edit6,'String','')
                            set(data_handles.edit7,'BackgroundColor', [1 1 1])
                            set(data_handles.edit7,'String','')
                            set(data_handles.edit8,'BackgroundColor', [1 1 1])
                            set(data_handles.edit8,'String','Nulo')
                            if semaforo==1

Img=image(semaforoblanco,'Parent',data_handles_semaforo.axes1);

set(data_handles_semaforo.axes1,'Visible','off','YDir','reverse','XLim',get(Im
g,'XData'),'YLim',get(Img,'YData'));
                                    disp('semaforoblanco')
                            end
                        end
%                     end
                otherwise
            end
%             end
        end
%         activity_past=activity;

end
```

Anexo 6 Estudio de errores de los ECO Nodes

Para medir los diferentes estados de actividad de un niño con TDAH, es necesario disponer de un flujo de datos fiable, es decir, un flujo de datos sin pérdidas de muestras. Dado que el sistema actual no implementa un sistema de acceso al medio, se han buscado las mejores condiciones para que el flujo de datos desde los ECO Nodes hasta el ordenador sea lo más estable posible. Los experimentos que se han realizado han buscado las mejores condiciones variando: frecuencia de muestreo, número de ECO Nodes transmitiendo a la vez, número de canales usados, distancia y comunicación uni/bidireccional.

A continuación se enumeran los diferentes experimentos que se han realizado, y los resultados obtenidos.

Transmisión de 1 ECO Node

El primer experimento que se realizado ha sido transmitiendo solo un ECO Node a la vez. Se ha medido la tasa de error en función de dos variables: la distancia, y la frecuencia.

Para la realización del experimento se ha puesto a muestrear el sensor de movimiento del ECO Node y a enviar un flujo de datos por RF hasta la Estación Base durante 5 minutos. La altura respecto al suelo de la Estación Base y del ECO Node ha sido de 22cm. El canal RF usado ha sido el 118. La distancia entre la Estación Base y el ECO Node ha ido variando en función del experimento, al igual que la frecuencia. Para una frecuencia de 40 Hz, se han obtenido los siguientes resultados:

Tabla 2 Pérdidas relativas en función de la distancia para 1 ECO Node a 40Hz

	Frecuencia	nº de ECO's	distancia(m)	tª(min.)	envíos	nodo	dist entre nodos	muestras	seq	errores	error relativo(%)	Media experimento
1	40	1	1	5	1	4		11952	11967	15	0,13	**0,06**
2	40	1	1	5	1	4		11963	11968	5	0,04	
3	40	1	1	5	1	4		11951	11968	17	0,14	
4	40	1	1	5	1	4		11955	11969	14	0,12	
5	40	1	1	5	1	4		11959	11970	11	0,09	
6	40	1	1	5	1	4		11965	11970	5	0,04	
7	40	1	1	5	1	4		11965	11968	3	0,03	
8	40	1	1	5	1	4		11966	11968	2	0,02	
9	40	1	1	5	1	4		11966	11967	1	0,01	
10	40	1	1	5	1	4		11968	11969	1	0,01	
1	40	1	2	5	1	4		11965	11969	4	0,03	**0,02**
2	40	1	2	5	1	4		11970	11971	1	0,01	
3	40	1	2	5	1	4		11971	11973	2	0,02	
4	40	1	2	5	1	4		11969	11971	2	0,02	
5	40	1	2	5	1	4		11971	11972	1	0,01	
1	40	1	3	5	1	4		11854	11970	116	0,97	**0,39**
2	40	1	3	5	1	4		11812	11970	158	1,32	
3	40	1	3	5	1	4		11929	11971	42	0,35	
4	40	1	3	5	1	4		11923	11975	52	0,43	
5	40	1	3	5	1	4		11890	11969	79	0,66	
6	40	1	3	5	1	4		11964	11968	4	0,03	
7	40	1	3	5	1	4		11966	11969	3	0,03	
8	40	1	3	5	1	4		11963	11966	3	0,03	
9	40	1	3	5	1	4		11965	11967	2	0,02	
10	40	1	3	5	1	4		11966	11969	3	0,03	
1	40	1	4	5	1	4		11922	11968	46	0,38	**0,29**
2	40	1	4	5	1	4		11937	11969	32	0,27	
3	40	1	4	5	1	4		11937	11968	31	0,26	
4	40	1	4	5	1	4		11938	11969	31	0,26	
5	40	1	4	5	1	4		11933	11967	34	0,28	
1	40	1	5	5	1	4		4616	11965	7349	61,42	**69,06**
2	40	1	5	5	1	4		3864	11954	8090	67,68	
3	40	1	5	5	1	4		3563	11965	8402	70,22	
4	40	1	5	5	1	4		4523	11964	7441	62,19	

5	40	1	5	5	1	4		1942	11961	10019	83,76	

Se obtienen por tanto unas buenas medidas para 40 Hz hasta 5 metros de distancia, donde el error asciende a un 70% de error aproximadamente.

Para una frecuencia de 20 Hz se obtienen los siguientes resultados:

Tabla 3 Pérdidas relativas en función de la distancia para 1 ECO Node 20Hz

	Frecuencia	nº de ECO's	distancia(m)	tº(min.)	envíos	Nodo	dist entre nodos	Muestras	seq	errores	error relativo(%)	Media experimento
1	20	1	1	5	1	4		5980	5981	1	0,02	0,02
2	20	1	1	5	1	4		5980	5982	2	0,03	
3	20	1	1	5	1	4		5980	5981	1	0,02	
4	20	1	1	5	1	4		5980	5981	1	0,02	
5	20	1	1	5	1	4		5980	5981	1	0,02	
1	20	1	2	5	1	4		5983	5984	1	0,02	0,03
2	20	1	2	5	1	4		5983	5984	1	0,02	
3	20	1	2	5	1	4		5981	5984	3	0,05	
4	20	1	2	5	1	4		5982	5984	2	0,03	
5	20	1	2	5	1	4		5980	5982	2	0,03	
1	20	1	3	5	1	4		5982	5984	2	0,03	0,03
2	20	1	3	5	1	4		5981	5982	1	0,02	
3	20	1	3	5	1	4		5980	5981	1	0,02	
4	20	1	3	5	1	4		5980	5981	1	0,02	
5	20	1	3	5	1	4		5979	5983	4	0,07	
1	20	1	4	5	1	4		5951	5984	33	0,55	1,14
2	20	1	4	5	1	4		5947	5984	37	0,62	
3	20	1	4	5	1	4		5940	5982	42	0,70	
4	20	1	4	5	1	4		5901	5984	83	1,39	
5	20	1	4	5	1	4		5836	5981	145	2,42	
1	20	1	5	5	1	4		2311	5971	3660	61,30	71,54
2	20	1	5	5	1	4		2308	5979	3671	61,40	
3	20	1	5	5	1	4		2392	5976	3584	59,97	
4	20	1	5	5	1	4		1312	5973	4661	78,03	
5	20	1	5	5	1	4		178	5950	5772	97,01	

Por tanto se llega a la misma conclusión, que hasta 5 metros de distancia se puede enviar datos sin ningún problema.

Para una frecuencia de 10 Hz se obtienen resultados similares:

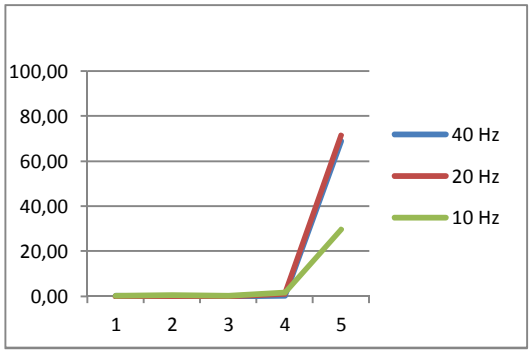
Tabla 4 Pérdidas relativas en función de la distancia para 1 ECO Node 10Hz

	Frecuencia	nº de ECO's	distancia(m)	tº(min.)	envíos	nodo	dist entre nodos	muestras	seq	errores	error relativo(%)	Media experimento
1	10	1	1	5	1	4		2989	2990	1	0,03	0,07
2	10	1	1	5	1	4		2987	2991	4	0,13	
3	10	1	1	5	1	4		2989	2990	1	0,03	
4	10	1	1	5	1	4		2990	2991	1	0,03	
5	10	1	1	5	1	4		2986	2989	3	0,10	
1	10	1	2	5	1	4		2971	2990	19	0,64	0,43
2	10	1	2	5	1	4		2974	2990	16	0,54	
3	10	1	2	5	1	4		2977	2991	14	0,47	
4	10	1	2	5	1	4		2980	2991	11	0,37	
5	10	1	2	5	1	4		2971	2990	19	0,64	
6	10	1	2	5	1	4		2976	2990	14	0,47	
7	10	1	2	5	1	4		2979	2989	10	0,33	
8	10	1	2	5	1	4		2983	2989	6	0,20	
9	10	1	2	5	1	4		2980	2990	10	0,33	
10	10	1	2	5	1	4		2982	2992	10	0,33	
1	10	1	3	5	1	4		2989	2990	1	0,03	0,06
2	10	1	3	5	1	4		2989	2991	2	0,07	
3	10	1	3	5	1	4		2989	2990	1	0,03	
4	10	1	3	5	1	4		2989	2990	1	0,03	
5	10	1	3	5	1	4		2986	2990	4	0,13	

1	10	1	4	5	1	4		2950	2990	40	1,34	1,57
2	10	1	4	5	1	4		2959	2992	33	1,10	
3	10	1	4	5	1	4		2924	2994	70	2,34	
4	10	1	4	5	1	4		2948	2992	44	1,47	
5	10	1	4	5	1	4		2944	2992	48	1,60	
1	10	1	5	5	1	4		2043	2989	946	31,65	29,78
2	10	1	5	5	1	4		1834	2991	1157	38,68	
3	10	1	5	5	1	4		1830	2990	1160	38,80	
4	10	1	5	5	1	4		2385	2991	606	20,26	
5	10	1	5	5	1	4		2406	2989	583	19,50	

Como con las demás frecuencias, se tiene un alcance de hasta 5 metros.

Se ve a continuación la gráfica del número de muestras perdidas en función de la distancia para diferentes frecuencias.



Gráfica 4 Pérdidas relativas en función de la distancia para 1 ECO Node

De la gráfica se concluye que podemos obtener un flujo fiable de datos con tasa de error por debajo de un 2% hasta los 5 metros, donde la tasa de error sube mucho.

Aunque en este experimento, la máxima distancia permitida son los 5 metros, no descartamos que sea mayor la distancia real. No se han contemplado posibles reflexiones de la señal con el suelo que harían que se produjeran un mínimo a la distancia exacta de 5 metros. Sin embargo, esto nos sirve para verificar que cumplimos las especificaciones que nos proponíamos: un alcance de 3 metros aproximadamente.

Transmisión 2 ECO Nodes a 40 Hz

El objetivo de los experimentos descritos a continuación es estudiar cuantos ECO Nodes se pueden utilizar a la vez, garantizando una calidad en el número de muestras perdidas. Este primer experimento se va ha realizado con 2 ECO Nodes transmitiendo a la vez.

El módulo de RF de los nodos no tiene control de acceso al medio. Esto provoca que al ponerse a emitir dos nodos aumente mucho la tasa de error debido a las interferencias que hay entre los dos nodos.

Una muestra de ese aumento de la tasa de error la tenemos en las siguientes tablas.

Tabla 5 Pérdidas relativas en transmisión simultánea de 2 ECO Nodes sin control de acceso al medio. Nodo 4

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11969	11971	2	118	1	0,02	**0,08**

11966	11968	2	118	1	0,02
11967	11968	1	118	1	0,01
11967	11968	1	118	1	0,01
11929	11969	40	118	1	0,33

En la tabla de arriba se ve los datos recibidos por el ECO Node número 4 en 5 experimentos de 5 minutos. La columna "muestras" da el valor del número de muestras que se han recibido, "seq" es el número de muestras que ha enviado el ECO Node. Haciendo la resta podemos ver cuántos "errores" ha habido. Con este dato podemos sacar el error relativo. Se ve que la tasa de error media del experimento es bastante buena, ya que un 0,08% es una pérdida de 8 muestras cada 10000.

No pasa lo mismo con el ECO Node 5, que es el otro nodo que se ha puesto a transmitir simultáneamente. Las interferencias hacen que se "imponga" el nodo 4 en algunos de los experimentos "machacando" las transmisiones del nodo 5. Vemos los datos a continuación.

Tabla 6 Pérdidas relativas en transmisión simultánea de 2 ECO Nodes sin control de acceso al medio. Nodo 5

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11841	11969	128	118	1	1,07	**60,59**
1	11969	11968	118	1	99,99	
2	11969	11967	118	1	99,98	
0	11969	11969	118	1	100,00	
11738	11965	227	118	1	1,90	

Se ve que en el experimento 1 y 5, el nodo 5 consigue enviar correctamente la mayoría de los datos. En cambio en los experimento 2, 3 y 4, prácticamente no se recibe ningún dato. Esta gran diferencia entre experimentos se debe también a la sincronización entre los nodos. Si los nodos se sincronizan, y mientras uno envía, el otro está esperando, se consigue una recepción sin errores. En cambio, si los nodos envían a la vez, uno de ellos se impone sobre el otro consiguiendo uno una tasa de error bajísima, mientras el otro no consigue enviar con éxito ningún dato.

Como se puede comprobar, hay un problema de interferencias entre los dos nodos. Esto es debido a que no existe control de acceso al medio.

Al encontrarse con esta problemática, se ha tenido que programar un sistema de comunicaciones nodos-Estación Base que garantizara una tasa de error lo suficientemente baja como para tener una comunicación fiable.

La primera prueba que se ha hecho ha consistido en usar una canal RF para cada nodo. La EB envía una única orden a los ECO Nodes donde se ordena que empiecen a emitir y el tiempo durante el que lo tienen que hacer. Cada nodo emite datos por un canal de comunicaciones diferente, de manera que se consigue que no existan interferencias entre los nodos. El problema que se tiene con este sistema proviene de la Estación Base. La EB, para recibir de ambos nodos tiene que ponerse a recibir datos primero por un canal y luego por el otro. A priori no tiene ningún problema si los dos nodos se "sincronizan" con la EB. Si así lo hacen,

cuando emita el nodo A por el canal X, la EB estará esperando dato nuevo por el canal X. Recibirá el dato correctamente. A continuación, el nodo B emitirá un dato por el canal Y, pero la EB ya habrá cambiado al canal Y, y por tanto también se recibirá correctamente el dato del nodo B. Y así sucesivamente.

El problema viene cuando los dos nodos no se sincronizan. Entonces cuando emite el nodo A por el canal X, la EB está esperando dato del nodo B por el canal Y. Esto provoca, que el dato del nodo A se pierda, ya que fue enviado por el canal X. Este hecho ha provocado que se tenga una tasa de error de un 50% en muchos casos, cosa que no es factible.

Se han hecho algunos experimentos para comprobar este hecho. Se han puesto a emitir el nodo 4 y el nodo 5, a una frecuencia de 40 Hz, cada uno por un canal diferente, con una distancia entre la EB y los nodos de 2 metros. Entre los dos ECO Nodes se ha calculado una distancia de 22 cm. La Estación Base y los nodos han sido colocados sobre cajas a una altura de 22 cm del suelo. Se han hecho 5 experimentos de 5 minutos. El nodo 4 ha transmitido por el canal 118, mientras que el nodo 5 lo ha hecho por el canal 119. Para el nodo 4 se ha obtenido los siguientes resultados:

Tabla 7 Pérdidas relativas en transmisión no simultánea de 2 ECO Nodes con control de acceso al medio en diferente canal. Nodo 4

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
5981	11967	5986	118	1	50,02	**50,01**
5984	11968	5984	118	1	50,00	
5984	11969	5985	118	1	50,00	
5986	11974	5988	118	1	50,01	
5982	11966	5984	118	1	50,01	

Se comprueba que se tiene una tasa de error de un 50% en el nodo 4.

Para el nodo 5 se tienen los siguientes datos:

Tabla 8 Pérdidas relativas en transmisión no simultánea de 2 ECO Nodes con control de acceso al medio en diferente canal. Nodo 5

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
5981	11967	5986	119	1	50,02	**50,02**
5986	11973	5987	119	1	50,00	
5982	11968	5986	119	1	50,02	
5981	11967	5986	119	1	50,02	
5981	11966	5985	119	1	50,02	

Se comprueba que la tasa de error es prácticamente la misma.

Al mirarse tramas de datos recibidos, se puede comprobar que se van recibiendo alternativamente un dato de un nodo, y otro de otro:

```
timestamp,time,nodo,ax,ay,az,sec
20100504210130,0130,5,2728,2720,2088,1
```

```
20100504210130,0130,4,2706,2828,2247,2
20100504210130,0130,5,2717,2720,2092,3
20100504210130,0130,4,2713,2828,2251,4
20100504210130,0130,5,2729,2721,2092,5
20100504210130,0130,4,2711,2831,2244,6
20100504210131,0131,5,2722,2717,2092,7
20100504210131,0131,4,2708,2830,2247,8
20100504210131,0131,5,2720,2722,2090,9
20100504210131,0131,4,2716,2827,2242,10
20100504210131,0131,5,2721,2729,2093,11
20100504210131,0131,4,2712,2834,2250,12
20100504210131,0131,5,2719,2724,2088,13
20100504210131,0131,4,2713,2837,2248,14
20100504210131,0131,5,2720,2722,2090,15
20100504210131,0131,4,2709,2842,2246,16
20100504210131,0131,5,2721,2720,2091,17
20100504210131,0131,4,2711,2830,2249,18
20100504210131,0131,5,2720,2713,2088,19
20100504210131,0131,4,2712,2830,2252,20
20100504210131,0131,5,2735,2715,2096,21
20100504210131,0131,4,2711,2833,2246,22
20100504210131,0131,5,2733,2730,2092,23
20100504210131,0131,4,2721,2837,2247,24
20100504210131,0131,5,2732,2717,2094,25
20100504210131,0131,4,2721,2841,2250,26
20100504210131,0131,5,2723,2722,2085,27
20100504210132,0132,4,2719,2833,2244,28
20100504210132,0132,5,2714,2719,2095,29
20100504210132,0132,4,2712,2836,2248,30
20100504210132,0132,5,2722,2719,2089,31
20100504210132,0132,4,2719,2831,2241,32
20100504210132,0132,5,2721,2716,2095,33
20100504210132,0132,4,2709,2837,2248,34
```

Como se ha podido comprobar hasta ahora, el mayor problema que se tiene es la falta de sincronización entre los dos ECO Nodes. Para solucionar este problema, se ha pensado en implementar un protocolo de comunicación bidireccional, donde la EB es la encargada de pedir a cada nodo que le envíe información. De esta manera, cada nodo enviará un dato nuevo muestreado cuando la EB se lo pida. Con esto se tiene el problema que el tráfico se duplicará, ya que cada muestra que envíe un nodo deberá venir precedida de una orden desde la EB..

El primer experimento realizado en esta línea ha sido utilizando 3 canales RF. El primero de todos (canal 118) es el utilizado para enviar órdenes desde la EB a los nodos. El segundo canal (119) lo usa el nodo 4 para comunicarse con la Estación Base. El canal 120 lo usa el nodo 5. La Estación Base usa diferentes direcciones para comunicarse con cada uno de los nodos. En el caso del nodo 4 usará la dirección 183. Para el nodo 5 la 180. De esta forma cada nodo sabe diferenciar si una orden es para él, o para el otro nodo. El experimento se ha realizado a una altura respecto al suelo de 22 cm. Una separación de 2 metros entre los nodos y la EB. Una separación entre nodos de 22 cm. Cada experimento ha durado 5 minutos a una frecuencia de muestreo de 40 Hz. Para el nodo 4 se tiene la siguiente tabla de resultados:

Tabla 9 Pérdidas relativas en transmisión no simultánea de 2 ECO Nodes con control de acceso al medio en tres canales diferentes. Nodo 4

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11802	11811	9	119	X	0,08	**0,94**

11805	11824	19	119	X	0,16
11786	11829	43	119	X	0,36
11796	11820	24	119	X	0,20
10682	11789	1107	119	X	9,39
11744	11800	56	119	X	0,47
11777	11793	16	119	X	0,14
11807	11810	3	119	X	0,03
11776	11799	23	119	X	0,19
11805	11815	10	119	X	0,08
11800	11801	1	119	X	0,01
11782	11796	14	119	X	0,12

Por tanto se obtiene una tasa de error de menos de 1%. Parece una buena tasa de error para una frecuencia de 40 Hz.

En este experimento se ha aprovechado el envío bidireccional, para programar un código de detección y corrección de errores. Cada vez que la Estación Base envía la orden a un nodo para que envíe un nuevo dato si lo tiene, incluye en la trama el último dato que ha recibido. En el caso de que no coincida con el último que envió el ECO Node, el nodo volverá a enviar el dato. Esto da más robustez al sistema. Y hace que se corrijan los errores automáticamente.

Para el nodo 5 se obtienen resultados similares:

Tabla 10 Pérdidas relativas en transmisión no simultánea de 2 ECO Nodes con control de acceso al medio en tres canales diferentes. Nodo 5

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11701	11810	109	120	X	0,92	**1,14**
11743	11823	80	120	X	0,68	
11696	11825	129	120	X	1,09	
11680	11815	135	120	X	1,14	
11700	11831	131	120	X	1,11	
11006	11796	790	120	X	6,70	
11795	11820	25	120	X	0,21	
11802	11811	9	120	X	0,08	
11734	11799	65	120	X	0,55	
11754	11813	59	120	X	0,50	
11751	11798	47	120	X	0,40	
11782	11820	38	120	X	0,32	

Este protocolo de comunicaciones tiene un problema de eficiencia. Este problema es que usamos 3 canales de comunicaciones en vez de 1. Para solventar este problema, se ha probado implementar el mismo protocolo de comunicaciones pero usando un único canal. Se ha podido comprobar que la tasa de error no aumenta. Esto es debido a que en este caso no se tienen interferencias entre nodos ya que ahora emiten de forma ordenada esperando a que la EB les dé permiso para emitir. A continuación los datos obtenidos en el nodo 4:

Tabla 11 Pérdidas relativas en transmisión no simultánea de 2 ECO Nodes con control de acceso al medio en un canal. Nodo 4

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11782	11809	27	118	X	0,23	**0,12**
11825	11828	3	118	X	0,03	
11821	11825	4	118	X	0,03	
11780	11803	23	118	X	0,19	
11817	11831	14	118	X	0,12	
11815	11827	12	118	X	0,10	
11820	11822	2	118	X	0,02	
11804	11807	3	118	X	0,03	
11802	11814	12	118	X	0,10	
11826	11831	5	118	X	0,04	
11755	11804	49	118	X	0,42	
11796	11830	34	118	X	0,29	
11794	11802	8	118	X	0,07	
11821	11826	5	118	X	0,04	
11787	11794	7	118	X	0,06	

Se ve que la tasa de error no aumenta con respecto al uso de tres canales.

Para el nodo 5 se obtiene un resultado similar:

Tabla 12 Pérdidas relativas en transmisión no simultánea de 2 ECO Nodes con control de acceso al medio en un canal. Nodo 5

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11768	11809	41	118	X	0,35	**1,12**
11800	11823	23	118	X	0,19	
11814	11822	8	118	X	0,07	
11752	11798	46	118	X	0,39	
11755	11827	72	118	X	0,61	
11803	11822	19	118	X	0,16	
11810	11817	7	118	X	0,06	
11783	11803	20	118	X	0,17	
10226	11811	1585	118	X	13,42	
11782	11826	44	118	X	0,37	
11793	11804	11	118	X	0,09	
11765	11826	61	118	X	0,52	
11784	11797	13	118	X	0,11	
11818	11821	3	118	X	0,03	
11769	11792	23	118	X	0,20	

La conclusión es que se puede usar indistintamente 1 o 3 canales, que los resultados en cuanto a tasa de error se refiere serán los mismos. La mejor solución es usar un solo canal, ya que así se optimiza el uso del espectro radioeléctrico.

Transmisión 3 ECO Nodes a 40 Hz

El último experimento que se ha realizado, ha sido ver la posibilidad de usar 3 ECO Nodes a la vez. Usando las mismas características que en los experimentos anteriores (2 metros de distancia nodos-EB, 40 Hz, 22 cm del suelo, etc.) se obtiene los siguientes resultados para el nodo 1:

Tabla 13 Pérdidas relativas en transmisión no simultánea de 3 ECO Nodes con control de acceso al medio en un canal. Nodo 1

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11467	11811	344	118	X	2,91	**3,36**
11255	11823	568	118	X	4,80	
11257	11831	574	118	X	4,85	
11411	11795	384	118	X	3,26	
11515	11816	301	118	X	2,55	
11518	11797	279	118	X	2,37	
11500	11817	317	118	X	2,68	
11424	11829	405	118	X	3,42	
11244	11815	571	118	X	4,83	
11571	11793	222	118	X	1,88	

Para el nodo 4 se obtienen los siguientes datos:

Tabla 14 Pérdidas relativas en transmisión no simultánea de 3 ECO Nodes con control de acceso al medio en un canal. Nodo 4

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11458	11812	354	118	X	3,00	**3,55**
11176	11819	643	118	X	5,44	
11231	11820	589	118	X	4,98	
11404	11828	424	118	X	3,58	
11523	11805	282	118	X	2,39	
11457	11830	373	118	X	3,15	
11565	11809	244	118	X	2,07	
11328	11816	488	118	X	4,13	
11268	11806	538	118	X	4,56	
11555	11816	261	118	X	2,21	

Por último, para el nodo 5:

Tabla 15 Pérdidas relativas en transmisión no simultánea de 3 ECO Nodes con control de acceso al medio en un canal. Nodo 5

muestras	seq	errores	canal	Envíos	error relativo(%)	Media experimento
11490	11812	322	118	X	2,73	**3,44**
11216	11823	607	118	X	5,13	
11199	11835	636	118	X	5,37	

11414	11800	386	118	X	3,27
11553	11816	263	118	X	2,23
11533	11816	283	118	X	2,40
11564	11817	253	118	X	2,14
11375	11816	441	118	X	3,73
11248	11815	567	118	X	4,80
11498	11801	303	118	X	2,57

Como se puede ver, se obtiene un error relativo de un 3,5% aproximadamente.

Anexo 7 Toma de datos

Introducción

Una vez funcionando de forma fiable los ECO Nodes, el siguiente paso es caracterizar los diferentes estados de excitación de un TDAH. Al estar los niños protegidos por la ley, y haber muchos trámites administrativos para poder hacer experimentos con ellos, se ha realizado un estudio con personas adultas. Estas personas se han ofrecido voluntariamente a simular diferentes estados de actividad para poder diferenciar los diferentes niveles de excitación que se puedan llevar a cabo en un niño con TDAH.

Se disponen de 17 voluntarios con los que se van a realizar los experimentos. Se ha realizado un estudio para caracterizar como son esas personas con las siguientes variables: edad, altura, peso e índice de masa corporal. Los resultados del estudio se pueden ver en la siguiente tabla:

Tabla 16 Datos de los voluntarios

	Edad	Altura	Peso	IMC
Persona 1	24	1,89	91	25,4752107
Persona 2	30	1,79	80	24,9680097
Persona 3	21	1,77	80	25,5354464
Persona 4	20	1,9	87	24,099723
Persona 5	21	1,83	79	23,5898355
Persona 6	26	1,7	75	25,9515571
Persona 7	20	1,74	80	26,4235698
Persona 8	24	1,86	97	28,0379235
Persona 9	23	1,87	89	25,4511138
Persona 10	27	1,7	75	25,9515571
Persona 11	21	1,8	95	29,3209877
Persona 12	24	1,71	68	23,2550186
Persona 13	24	1,68	70	24,8015873
Persona 14	24	1,73	82	27,3981757
Persona 15	34	1,91	97	26,5891834
Persona 16	25	1,87	88	25,1651463
Persona 17	33	1,76	87	28,0862603

Se pueden dividir los niveles de actividad de los voluntarios en 4 tipos:

1. Actividad nula: la caracterización consiste en dejar los ECO Nodes en reposo.
2. Actividad baja: la caracterización consiste en poner a los voluntarios en un estado de calma, por ejemplo estudiando, viendo la televisión, navegando por internet, etc. Todas estas actividades no necesitan de ningún movimiento corporal apreciable, no necesitan desplazarse hasta otro sitio.
3. Actividad media: la caracterización se realiza con actividades que tengan incluido un desplazamiento físico: mover una caja, caminar de un sitio a otro, hablar de pié en público.
4. Actividad alta: la caracterización se realiza con actividad ajetreada, como por ejemplo realizando ejercicio físico, movimientos nerviosos, saltos, etc.

Según la Biblioteca Nacional de Medicina de los Estados Unidos (18) los síntomas de hiperactividad en un niño son:

1. Juega con las manos o los pies o se retuerce en su asiento.
2. Abandona su asiento cuando lo que se espera es que se quede sentado.
3. Corre y trepa excesivamente en situaciones inapropiadas.
4. Tiene dificultad para jugar en forma silenciosa.
5. A menudo habla excesivamente, está "en movimiento" o actúa como si fuera "impulsado por un motor".

Esto lleva a pensar que unos de los sitios del cuerpo del niño que tiene más movimiento son las extremidades (punto 1). Por tanto se puede colocar un ECO Node en las manos y los pies. Colocarlo en la mano no tiene más problema ya que se puede colocar en una muñequera. En cambio, colocar un ECO Node en los pies tiene más complicaciones de cara a que pueda sufrir algún tipo de percance. Por ello se ha pensado en colocar un ECO Node en la muñeca y otro ECO Node en un colgante en el cuello. Ese colgante será muy sensible al movimiento del cuerpo de niño. Por ejemplo, si se levanta, corre, abandona el sitio donde está sentado, ese colgante reflejará esos movimientos. Todas estas actitudes del niño se ven reflejadas en los puntos 2, 3 y 5. El punto 4 (*Tiene dificultad para jugar en forma silenciosa*) solo se podría valorar si dispusiéramos de un micrófono conectado al niño.

Objetivos

El objetivo de este experimento es poder hacer una aproximación inicial de los diferentes niveles de excitación que se pueden dar en un niño hiperactivo. Este experimento tiene la dificultad de no disponer de niños con TDAH ya que están protegidos por la ley, y se necesitan permisos que alargarían el PFC mucho tiempo. Por este motivo, las muestras que se han tomado no son de auténticos niños hiperactivos, sino de “actores”. Por tanto, uno de los objetivos de este experimento es aproximarse el máximo posible al caso real de los niños. Para ello se cuenta con la ayuda de los expertos de la asociación Atenciona, que han asesorado en todo el proceso para poder reflejar al máximo un caso auténtico.

Por tanto, el objetivo principal de este experimento es sacar unos niveles de actividad de los sensores que nos sirvan de umbrales para distinguir los distintos niveles de actividad de un TDAH. Esos niveles se podrán nivelar a la hora de poner en funcionamiento los dispositivos, pero de entrada esto nos servirá para tener unos valores de referencia.

Metodología

La metodología que se ha seguido ha consistido en varios pasos:

1. Selección de candidatos que sean lo más diferentes entre sí posibles.
2. Toma de sus datos personales (edad, altura, peso), y entrega de formularios.
3. Realización de los experimentos.
4. Análisis de los datos y cálculo de los umbrales.

En la realización de los experimentos, cada persona ha simulado un estado de excitación de un niño. Se ha hecho un experimento de 5 minutos por persona para cada estado de excitación excepto para el estado de excitación 1 (*Actividad nula*), ya que con un experimento basta para todas las personas. Es decir, cada persona ha realizado un total de 3 experimentos. En total 15 minutos. Durante este tiempo, las personas han estado a una distancia de la estación base no superior a 2 metros. En cada experimento, la estación base ha recogido información de los dos

nodos de “actor”. Uno ha estado colocado en la muñeca, y otro en un colgante situado en el cuello. Los datos recogidos se han guardado en una carpeta donde el nombre del archivo de cada experimento donde figura la persona, el día, la hora, el número de ECO Node, y el estado de excitación. El nombre del archivo tiene el siguiente formato: *Date_YYYY-MM-DD_Hour_hh-mm-ss_Person_P_Activity_A_Nodo_N.mat*. El tipo de dato es *.mat*. Los datos se han guardado en 6 columnas. La primera columna marca los minutos y segundos a los que se ha tomado el dato. La segunda columna el número de nodo. La tercera indica el valor de la aceleración en el eje “x”. La cuarta en el eje “y”, y la quinta en el eje “z”. La sexta columna indica el número de secuencia. Este número indica el número de dato que ha enviado el nodo. Si se pierde algún dato lo podemos ver porque hay un salto en el número de secuencia.

Tabla 17 Datos de los acelerómetros

	1	2	3	4	5	6	7
4	647	4	2720	2837	2249	3	
5	647	4	2709	2834	2245	4	
6	647	4	2715	2834	2252	5	
7	647	4	2712	2835	2238	6	
8	647	4	2713	2834	2245	7	
9	647	4	2708	2831	2247	8	
10	647	4	2718	2837	2249	9	
11	647	4	2712	2832	2248	10	
12	647	4	2711	2824	2242	11	
13	647	4	2719	2832	2249	12	
14	647	4	2708	2835	2245	13	
15	647	4	2719	2831	2250	14	
16	647	4	2717	2828	2243	15	
17	647	4	2706	2835	2246	16	
18	647	4	2706	2826	2245	17	
19	647	4	2711	2833	2249	18	
20	647	4	2717	2827	2243	19	
21	648	4	2708	2825	2249	20	
22	648	4	2716	2840	2245	21	
23	648	4	2708	2839	2244	22	
24	648	4	2713	2835	2244	23	
25	648	4	2714	2833	2250	24	
26	648	4	2707	2830	2247	25	
27	648	4	2718	2830	2247	26	
28	648	4	2709	2836	2248	27	
29	648	4	2712	2837	2242	28	
30	648	4	2713	2830	2254	29	
31	648	4	2710	2843	2251	30	
32	648	4	2712	2830	2250	31	
33	648	4	2707	2833	2249	32	
34	648	4	2713	2832	2250	33	
35	648	4	2710	2832	2246	34	
36	648	4	2722	2838	2252	35	
37	648	4	2713	2825	2242	36	
38	648	4	2717	2841	2250	37	

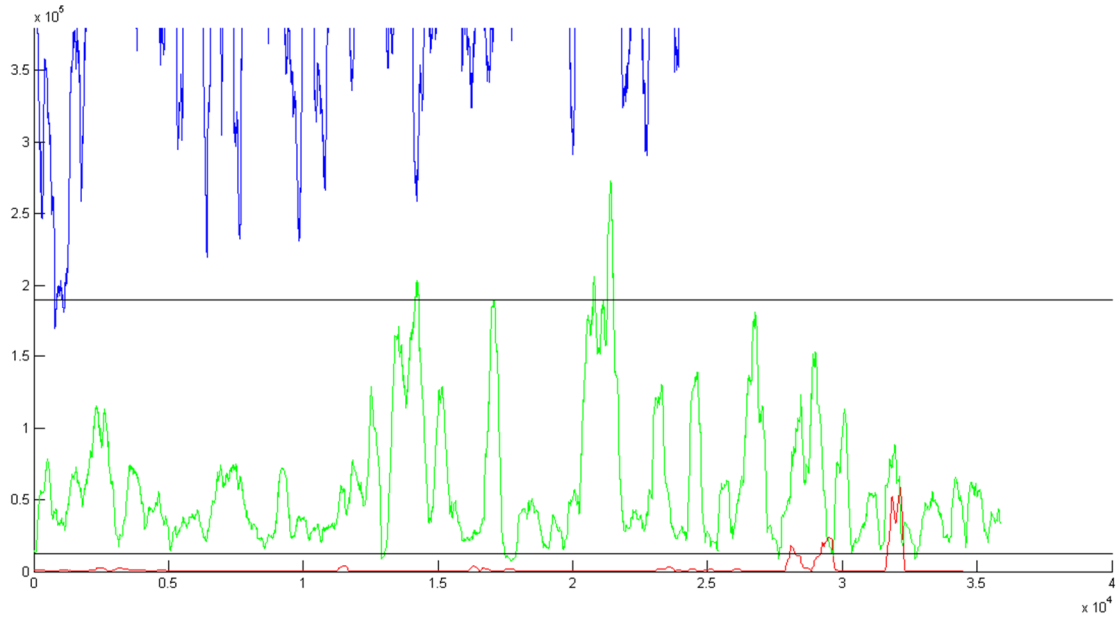
Análisis de resultados y cálculo de los umbrales

Una vez obtenida la base de datos de muestras de la actividad de personas se ha procedido al análisis de los resultados y el cálculo de los umbrales óptimos junto con la tasa de error.

Disponemos de una base de datos de muestras donde figuran las aceleraciones en los tres ejes del espacio para una actividad determinada de diez personas diferentes. Se han juntado todas las muestras por actividad obteniendo al final **tres bloques de muestras**: un bloque para actividad baja, otro para actividad media, y el último para actividad alta.

Cada uno de esos bloques se ha introducido en el sistema descrito en el apartado 4.5. Este sistema produce los valores del nivel de actividad para cada uno de los nodos. Por tanto, al final del sistema se ha obtenido una base de datos con valores calculados por el detector Esas muestras son las que se han usado para calcular los umbrales óptimos.

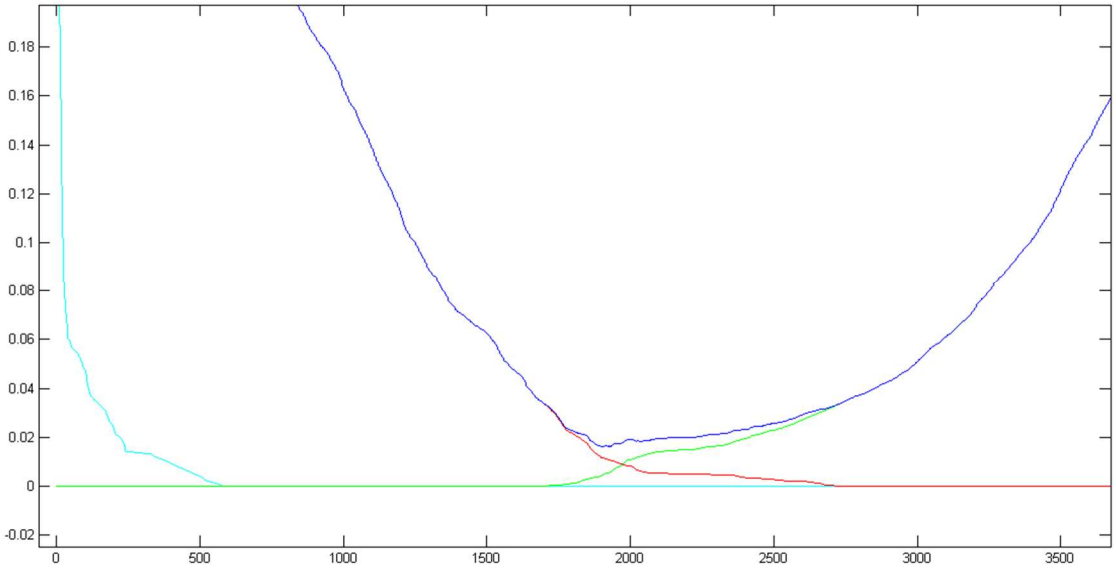
En la figura siguiente se puede ver una gráfica con los niveles de actividad de cada uno de los bloques:



Gráfica 5 Grafica del nivel de actividad

La línea de color azul representa los valores del nivel de actividad cuando los usuarios han simulado estar en un nivel de actividad alto. Como puede verse, la salida del detector da un valor que en la mayor parte de las veces supera 200000. Debajo de la línea azul encontramos la línea verde que marca el nivel de actividad a la salida del detector para muestras de la simulación a nivel medio. Por último, la línea roja marca el nivel de actividad cuando los actores han simulado estar en un nivel de actividad bajo. Las dos líneas horizontales marcan los umbrales óptimos obtenidos.

Para calcular cada uno de los umbrales óptimos se ha calculado la tasa de error resultante al ir variando el umbral desde 0 hasta 100000. La gráfica siguiente muestra como se ha realizado el cálculo:



Gráfica 6 Tasa de error en función del umbral

En el eje x representa los valores del umbral. El eje y representa la probabilidad de error para un valor x de umbral. Esta gráfica muestra el cálculo del umbral 2 para el nodo de la muñeca. Por tanto, este umbral diferenciará el nivel de actividad medio, del nivel de actividad alto.

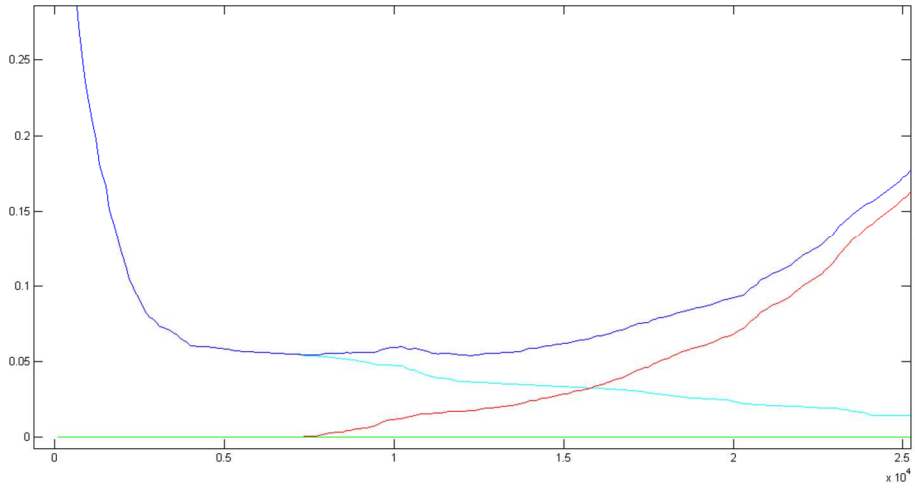
Las 4 líneas representan el error que habría para cada uno de los niveles de actividad si hubiera un umbral de actividad x. La línea azul claro, muestra el error para la actividad baja. La línea roja muestra el error para un nivel de actividad medio. La línea verde para un nivel de actividad alto. Y por último, la línea azul oscuro muestra la suma de todas las tasas de error.

Como era de esperar, en el caso de que el umbral estuviera a un nivel 0, el error sería de un 100% para las muestras a actividades bajas y medias. Mientras que sería 0% para muestras a actividades altas.

El umbral óptimo será el valor mínimo de la línea azul.

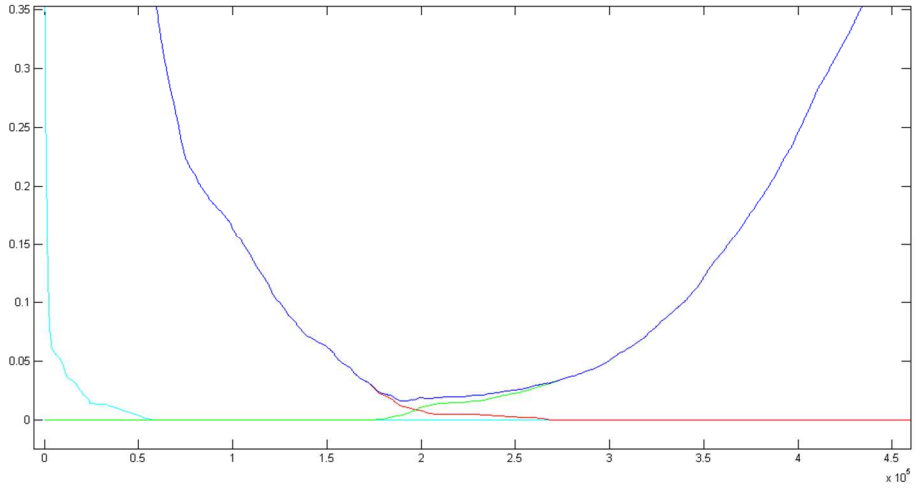
Este cálculo se ha realizado para los dos detectores que se han implementado: cálculo de la varianza y cálculo de la pendiente.

A continuación se van a mostrar las gráficas obtenidas para cada umbral usando el detector de cálculo de la varianza:



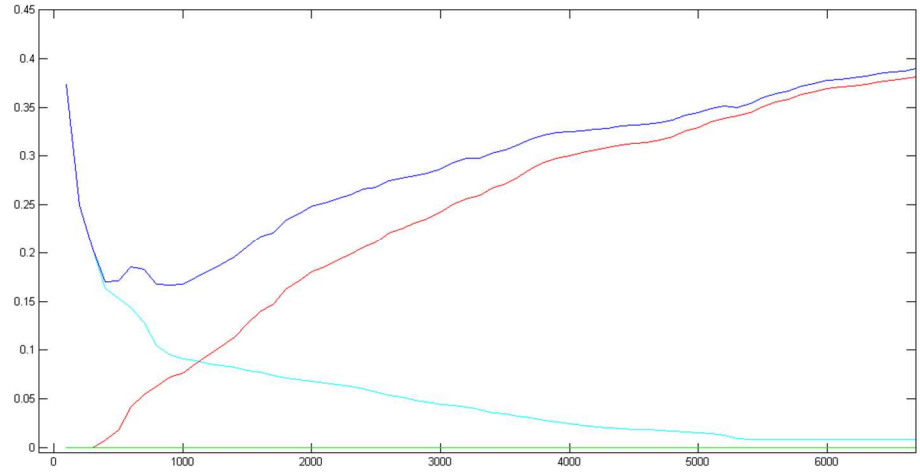
Gráfica 7 Detección por cálculo de la varianza. Muñeca umbral 1

Para la detección del umbral 1 de la muñeca se ha obtenido una tasa de error de 5,4% para un umbral óptimo de 12300.



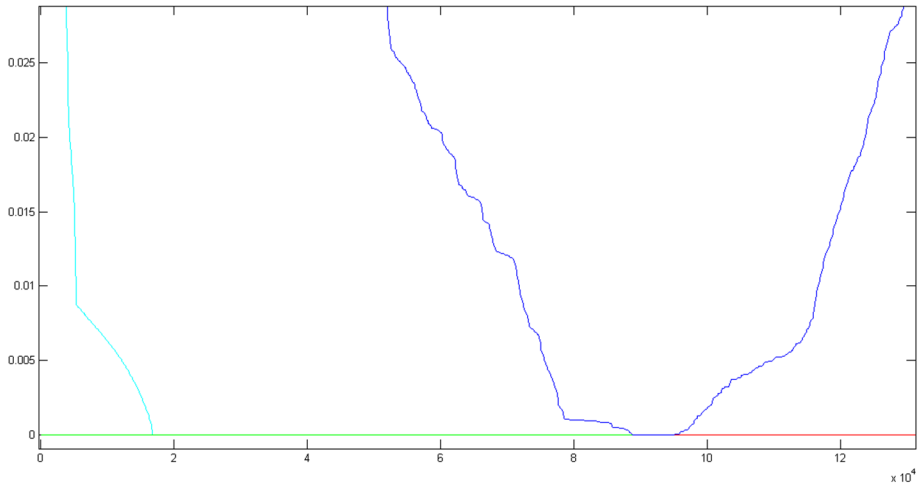
Gráfica 8 Detección por cálculo de la varianza. Muñeca umbral 2

Para la detección del umbral 2 de la muñeca se ha obtenido una tasa de error de 1,6% para un umbral óptimo de189900.



Gráfica 9 Detección por cálculo de la varianza. Colgante umbral 1

Para la detección del umbral 1 del colgante se ha obtenido una tasa de error de 1,6% para un umbral óptimo de 900.

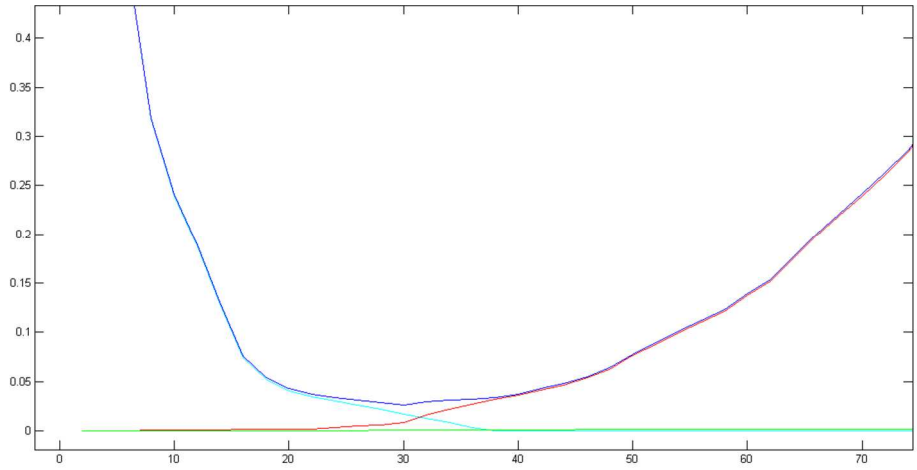


Gráfica 10 Detección por cálculo de la varianza. Colgante umbral 2

Para la detección del umbral 2 del colgante se ha obtenido una tasa de error de 0% para un umbral óptimo de 89000.

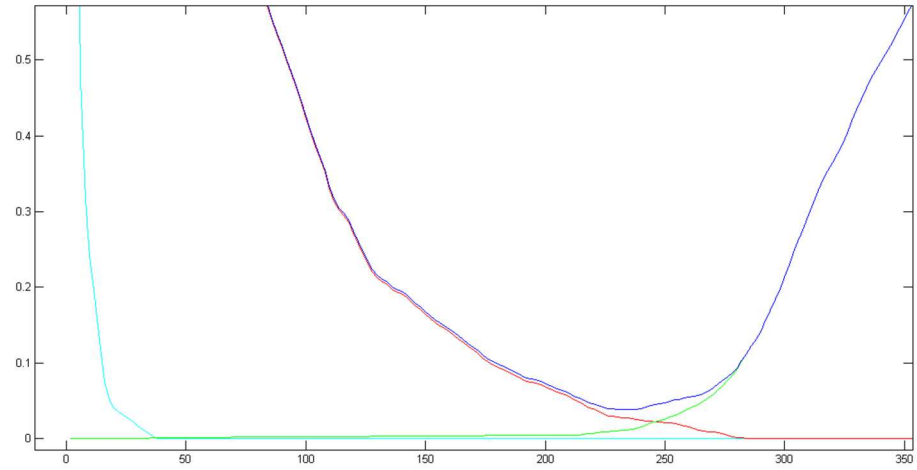
Por tanto, tenemos que de media obtendremos una tasa de error de 2,15%.

A continuación se van a mostrar las gráficas obtenidas para cada umbral usando el detector de cálculo de la pendiente:



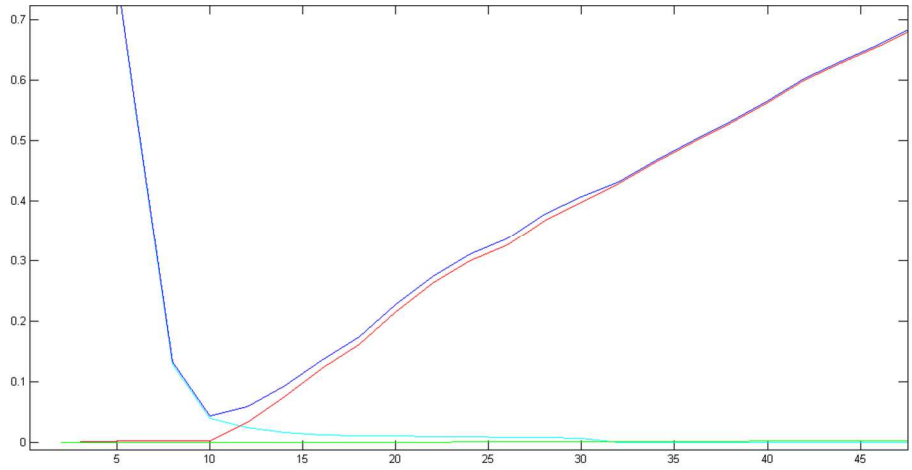
Gráfica 11 Detección por cálculo de la pendiente. Muñeca umbral 1

Para la detección del umbral 1 de la muñeca se ha obtenido una tasa de error de 2,58 % para un umbral óptimo de 30.



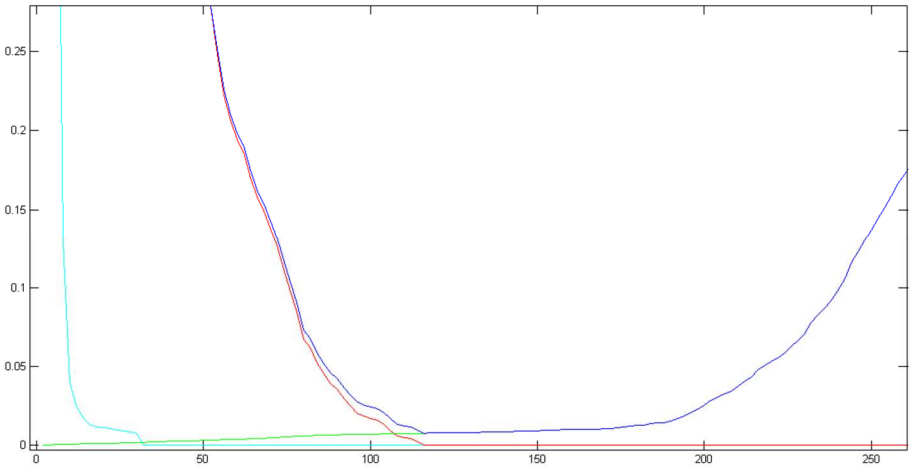
Gráfica 12 Detección por cálculo de la pendiente. Muñeca umbral 2

Para la detección del umbral 2 de la muñeca se ha obtenido una tasa de error de 3,85 % para un umbral óptimo de 234.



Gráfica 13 Detección por cálculo de pendiente. Colgante umbral 1

Para la detección del umbral 1 del colgante se ha obtenido una tasa de error de 4,43 % para un umbral óptimo de 10.



Gráfica 14 Detección por cálculo de pendiente. Colgante umbral 2

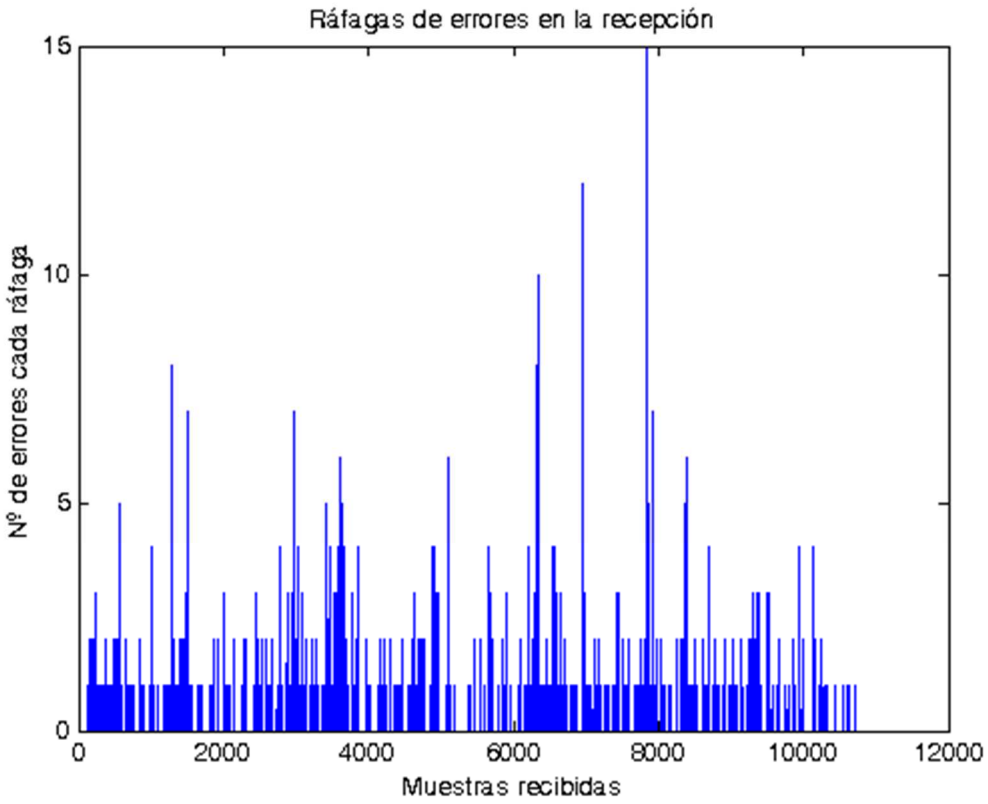
Para la detección del umbral 2 del colgante se ha obtenido una tasa de error de 0,7 % para un umbral óptimo de 116.

Por tanto, tenemos que de media obtendremos una tasa de error de 3,25%.

Comparando los dos sistemas de detección de movimiento podemos concluir que son muy parecidos en cuanto a tasa de error aunque el sistema del cálculo de la varianza es ligeramente mejor.

Anexo 8 Predicción y reconstrucción de la señal

Como ya se ha explicado en otras secciones, la comunicación entre los ECO Nodes y la Estación Base sufre algunas pérdidas. Esas pérdidas son causadas por varios factores: bloqueo de la señal por parte de algún objeto (la mano, el cuerpo), distancia entre el Nodo y la Estación Base, movimientos bruscos del usuario, etc. Como se puede observar a continuación, la mayoría de las pérdidas son de ráfagas de pocas muestras, eso hace que la señal pueda ser reconstruida con mucha facilidad.



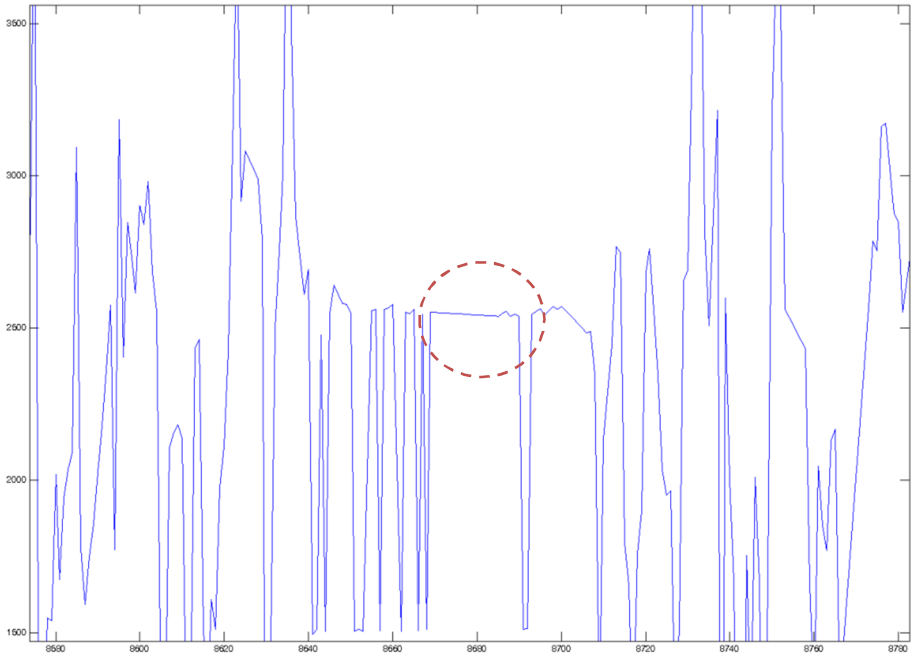
Gráfica 15 Nº de errores en ráfagas de errores en recepción

En este anexo, se va a explicar los diferentes sistemas que se han probado para la reconstrucción de la señal.

Interpolación lineal

Es el sistema más sencillo que se ha probado. Consiste en hacer una interpolación lineal en la zona de la señal donde se pierden las muestras, añadiendo tantas muestras como muestras perdidas. Tiene fundamentalmente una gran ventaja, y es que no consume muchos recursos. El principal problema que tiene es que este sistema no tiene en cuenta la "historia" de la señal. Para ráfagas de error grandes (que se dan mucho en movimientos bruscos, es decir, con una evolución de la señal caótica y con altas frecuencias) este sistema no funciona, ya que aplana la señal, dando la sensación de que el movimiento se ha estabilizado radicalmente. A continuación se puede ver una gráfica donde se ve que una ráfaga de 15 muestras ha estabilizado una señal que tenía un historial de cambios bruscos continuos. Este problema puede provocar que un usuario en un estado de actividad alto, al perder el receptor la señal, el sistema deduzca que ha cambiado a un estado de actividad medio, y mientras el niño sigue

estando en ese estado. Se puede ver gráficamente un ejemplo en la figura siguiente. Vemos en la figura como la historia de la aceleración está compuesta por cambios muy bruscos. Al producirse un error (redondeado de rojo en la gráfica), el sistema de reconstrucción de interpolación lineal produciría una señal continua durante toda la duración del error. Esto haría que el detector pudiera interpretar que el usuario está reduciendo su actividad, cosa que no es cierta. Lo que de verdad pasa es que se han perdido una serie de muestras y esto provoca un nivel de señal continuo.



Gráfica 16 Ejemplo de reconstrucción de señal con interpolación lineal

En resumidas cuentas, necesitamos un sistema que recupere la señal teniendo en cuenta el pasado de la señal, y sobre todo, su pasado frecuencial, ya que lo que nos interesa es la varianza de la señal, los cambios bruscos de la señal, y eso se ve reflejado en el espectro frecuencial. Para conseguir este, vamos a implementar un filtro adaptativo LMS.

Filtrado adaptativo

Un filtro adaptativo es un filtro cuyos pesos se adaptan dinámicamente a una señal deseada concreta. El error entre la señal deseada y la señal de salida del filtro, forma una de las variables de entrada del filtro que corrige los pesos del filtro adaptándolo a la señal deseada. Un ejemplo sencillo de filtro adaptativo lo podemos ver en la figura siguiente:

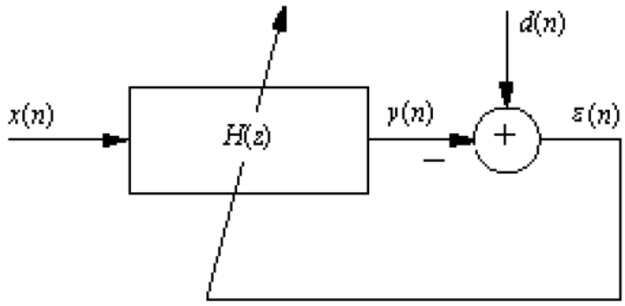


Figura 38 Filtro adaptativo

Haciendo una pequeña modificación podemos implementar un filtro predictivo, es decir, un filtro que va cambiando dinámicamente con el fin de predecir la muestra siguiente de la señal de entrada. Con este sistema, podremos ir ajustando dinámicamente el filtro a la señal que recibimos del receptor para que cuando tengamos una pérdida, el filtro prediga que muestra tenía que salir en función de la "historia" reciente de la señal. En la figura 45 vemos cómo quedará el filtro adaptativo preparado para la predicción de señal en el caso de pérdidas de señal. Este sistema tiene la ventaja de tener memoria, y se queda con la evolución pasada de la señal para predecir el futuro en caso de necesidad.

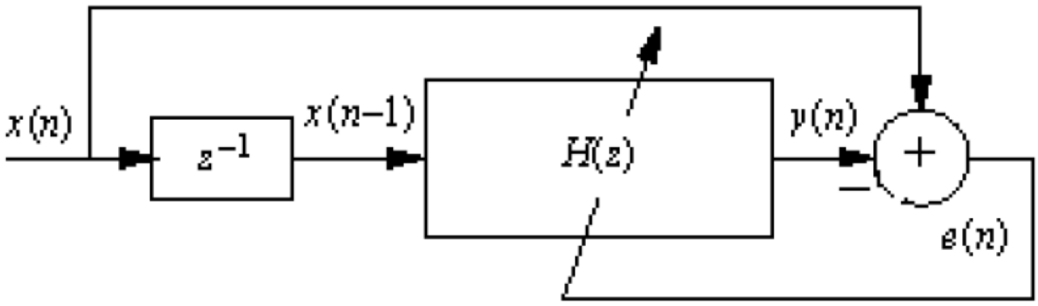
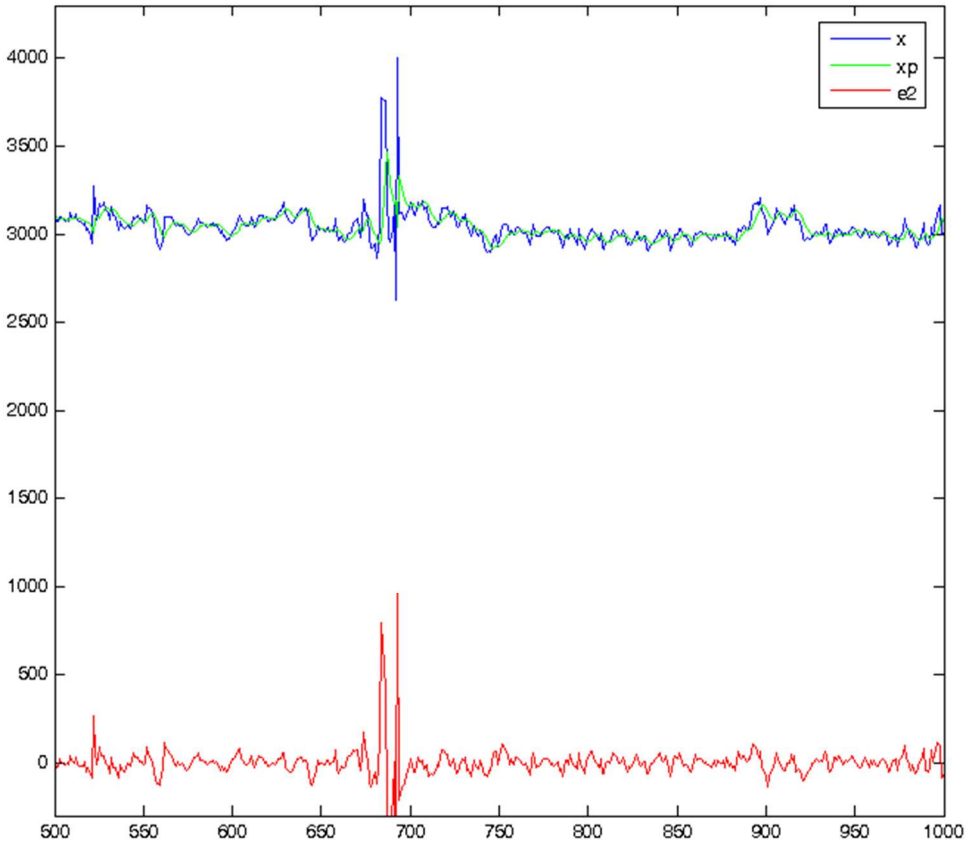


Figura 39 Filtro predictivo

Podemos ver un ejemplo de predicción continua de señal en la figura siguiente:



Gráfica 17 Ejemplo filtro predictivo para una actividad baja

Se puede ver como la señal de salida del filtro adaptativo (xp) se va ajustando dinámicamente a las variaciones de la señal de entrada (x). El error entre las dos señales (e2) tiene una media de 64,94, muy bajo para señales que oscilan entre 0 y 4000.

El filtro tiene unos parámetros que se ajustarán con el fin de minimizar la diferencia entre la señal original y la señal predicha. Para obtenerlos, se hará una media ponderada de los coeficientes óptimos de señales de ECO Nodes con actividad alta.

Conclusiones

El sistema descrito arriba, sería un buen sistema de predicción de señal si no produjera mucho costo computacional. Es por ello que finalmente se ha descartado el uso del filtrado adaptativo.

La solución más práctica para solucionar el problema que dan los errores ha sido sencillamente no incluir la reconstrucción de la señal en el diagrama de bloques final. Es decir, si perdemos muestras de señal, sencillamente se dejará de procesar el grado de actividad de la persona que lleva los nodos.

Esto puede producir que aumente la varianza de las muestras ya que la interpolación de una señal siempre expande el espectro frecuencial. Podemos ver a continuación una imagen (19) donde se ilustra este efecto.

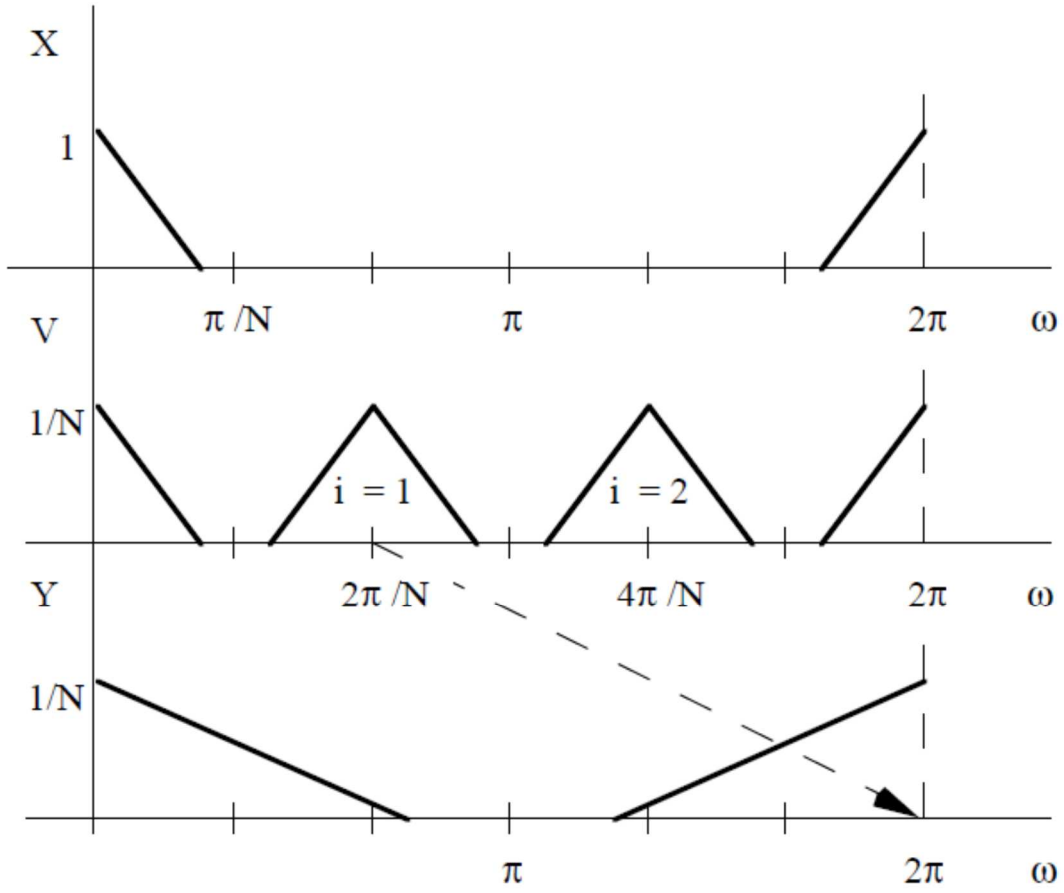


Figura 40 Expansión frecuencial en el muestreo

Esta expansión frecuencial aumentará un tanto por ciento la varianza de la señal final. A pesar de tener este hándicap, la suma del ahorro computacional que conlleva no usar filtrado adaptativo, más el ahorro de problemas que conlleva evitar la predicción lineal ha hecho que se decida no usar un bloque de reconstrucción de la señal y por tanto asumir el riesgo de un aumento de la varianza en el bloque.