



UNIVERSIDAD DE EXTREMADURA
Escuela Politécnica
Ingeniería Informática

Proyecto Fin de Carrera
Aportación a RoboComp: Diseño y Desarrollo de un Componente
de Adquisición Biométrica

Autor: Adrián Tinoco Herrero
Fdo.:

Director: José Moreno del Pozo
Fdo.:

Tribunal Calificador
Presidente: Pedro Núñez Trujillo
Fdo:

Secretario: Francisco Andrés Hernández
Fdo.:

Vocal: Antonio M. Silva Luengo
Fdo.:

CALIFICACIÓN:
FECHA:

Índice de contenido

Agradecimientos.....	7
Parte I: Presentación de la tecnología, Estado del Arte y Propuestas.....	8
1. Qué es Bitalino.....	8
2. Descripción general.....	8
3. Descripción de los componentes.....	9
3.1 MCU - Unidad Micro Controladora.....	9
3.2 PWR - Alimentación.....	9
3.3 EMG - Electromiógrafo.....	10
3.4 EDA - Actividad Electro Dérmitica.....	10
3.5 LUX - Foto diodo.....	11
3.6 ECG - Electro Cardiograma.....	12
3.7 ACC - Acelerómetro tri-axial.....	13
4. Estado del Arte.....	14
4.1 Versatilidad del cuerpo humano para el control a través de interfaces electromiográficas de bajo coste.....	14
4.2 Una valoración de los sensores EMG de un solo canal para el reconocimiento gestual.....	15
4.3 Evaluación de una Aplicación reconocedora de contexto para el control de robots móviles por medio de datos fisiológicos: El Caso de Estudio ToBITas.....	16
5. Propuestas.....	17
Parte II: Adquisición, Estudio y Análisis de Señales Biométricas.....	18
6. Electromiógrafo: Adquisición y procesado de señales EMG.....	18
6.1 Adquisición.....	18
6.1.1 Electrodos.....	18
6.1.2. Colocación.....	19
6.1.3 Función de transferencia.....	21
6.2 Estudio y Caracterización de la señal.....	22
6.2.1 Muestras de estudio.....	22
6.2.2 Análisis de las muestras en MATLAB.....	24
6.2.3 Procesado de la señal.....	32
7. Acelerómetro: adquisición y procesado de señales ACC.....	36
7.1. Adquisición.....	36
7.1.1 Sistema de referencia.....	37
7.1.2 Función de transferencia.....	37
7.1.3 Fase de calibración.....	38
7.2 Estudio y caracterización de la señal.....	39
7.2.1 Muestras objeto de estudio.....	39
7.2.2 Análisis de las muestras en MATLAB.....	40
7.2.3 Procesado de la señal.....	42
8. Electro Cardiograma: adquisición y procesado de señales ECG.....	45
8.1 Adquisición.....	45
8.1.1 Colocación de los electrodos.....	45
8.1.2 Función de transferencia.....	47
8.2 Estudio y Caracterización de la señal.....	48
8.2.1 Muestras de estudio.....	48
8.2.2 Análisis de las muestras en MATLAB.....	49
8.2.3 Procesado de la señal.....	51
Parte III: Aportación a RoboComp: Diseño y Desarrollo de un Componente de Adquisición Biométrica.....	53

9.	RoboComp: Un Framework de Desarrollo para Robótica.....	53
9.1	Componentes de Software en RoboComp, una breve introducción.....	53
9.1.1	Ingeniería de software basada en componentes.....	53
9.1.2	Modelo de componentes en RoboComp.....	54
9.2	Modelo de comunicación: Ice.....	56
9.3	Soporte de lenguajes.....	56
9.4	Herramientas.....	56
10.	Diseño de una componente de adquisición biométricas.....	57
10.1	Definición de requisitos mediante Modelo de Casos de Uso.....	57
10.1.1	Actores.....	57
10.1.2	Diagrama de Casos de Uso.....	58
10.1.3	Descripción de los Casos de Uso.....	58
10.2	Entorno de desarrollo: herramientas y librerías seleccionadas.....	65
10.2.1	RoboComp.....	65
10.2.2	Lenguaje de Programación.....	65
10.2.3	Entorno de Desarrollo Integrado (IDE): QtCreator configurado con Qt4.....	65
10.2.4	Qwt: Widgets en Qt para Aplicaciones Técnicas.....	66
10.2.5	API de bitalino.....	66
10.3	Manual del programador.....	67
10.3.1	Esquema de clases.....	67
11.	Manual del usuario.....	74
11.1	Emparejamiento del dispositivos.....	74
11.2	Pestaña de conexión.....	75
11.3	Adquisición de datos en crudo (RAW).....	76
11.4	Pestaña EMG.....	79
11.5	Pestaña ACC.....	82
11.6	Pestaña ECG.....	84
Parte IV: Caso práctico: Interacción Hombre-Máquina.....	86	
12.	Presentación del robot: LearnBot.....	86
13.	Interfaz de control.....	88
14.	Pruebas de uso.....	90
14.1	Participantes.....	90
14.2	Procedimiento.....	90
14.3	Resultados de las pruebas.....	91
15.	Resultado, Conclusiones y Futuro.....	94
Notas Finales: Bibliografía y Anexos.....	95	
16.	Bibliografía.....	95
17.	Anexo I: Instalación de RoboComp.....	97
18.	Anexo II: Generación de un componente en Robocomp usando robocompdsl.....	100
19.	Anexo III: Instalación de Qwt.....	102

Índice de ilustraciones

Ilustración 1: Placa de desarrollo bitalino.....	8
Ilustración 2: Módulo MCU.....	9
Ilustración 3: Módulo PWR.....	9
Ilustración 4: Módulo EDA Actividad Electro Dérmica.....	11
Ilustración 5: Módulo LUX.....	11
Ilustración 6: Módulo ECG.....	12
Ilustración 7: Módulo ACC.....	13
Ilustración 8: Control de un brazo robótico por medio de EMG.....	14
Ilustración 9: Porcentajes de éxito en las pruebas.....	15
Ilustración 10: Esquema del Pipeline de 5 fases.....	15
Ilustración 11: Ubicación del brazalete.....	16
Ilustración 12: Electrodo reutilizable.....	18
Ilustración 13: Electrodo desechable.....	19
Ilustración 14: Diagrama de colocación de los sensores en el bíceps.....	20
Ilustración 15: Diagrama de colocación de los sensores en el antebrazo.....	20
Ilustración 16: Gráfico comparativo - Filtros tipo Butterworth y Chebyshev.....	29
Ilustración 17: Sistema de referencia en un chip ACC típico.....	36
Ilustración 18: Sistema de referencia en el módulo ACC de bitalino.....	37
Ilustración 19: Calibración de el eje X del acelerómetro.....	38
Ilustración 20: Giramos el módulo ACC.....	38
Ilustración 21: Giro completo de 180°.....	38
Ilustración 22: Colocación de los electrodos en el pecho.....	46
Ilustración 23: Colocación de los electrodos en las manos.....	46
Ilustración 24: Electrodos colocados en las muñecas.....	46
Ilustración 25: Electrodo colocado en el tobillo derecho.....	47
Ilustración 26: Ejemplo de modelo por componentes.....	54
Ilustración 27: Diagrama de casos de uso de Bitalino BioSignals.....	58
Ilustración 28: Ejemplo: aplicaciones desarrolladas usando qwt.....	66
Ilustración 29: Diagrama de clases de Bitalino BioSignals.....	67
Ilustración 30: Búsqueda de dispositivos bluetooth en GNU/Linux.....	74
Ilustración 31: Selección de PIN en GNU/Linux.....	74
Ilustración 32: Dispositivo emparejado en GNU/Linux.....	75
Ilustración 33: Añadir nueva dirección MAC.....	75
Ilustración 34: Introducción de dirección MAC.....	76
Ilustración 35: Conexión correcta.....	76
Ilustración 36: Conexión incorrecta.....	76
Ilustración 37: Casillas de selección de canal.....	77
Ilustración 38: Numeración de puertos digitales en bitalino.....	77
Ilustración 39: Adquiriendo datos en crudo.....	78
Ilustración 40: Pestaña para activar el volcado de muestras.....	78
Ilustración 41: Pestaña EMG: Adquiriendo datos en RAW.....	79
Ilustración 42: Pestaña EMG: Filtro Paso Bajo aplicado.....	80
Ilustración 43: Pestaña EMG: Filtro Media Móvil aplicado.....	80
Ilustración 44: Pestaña EMG: Detección de picos.....	81
Ilustración 45: Pestaña EMG: LED de detección y contador de picos.....	81
Ilustración 46: Pestaña ACC: Datos en RAW.....	82
Ilustración 47: Pestaña ACC: Selección de ejes.....	82
Ilustración 48: Pestaña ACC: Filtro Media Móvil.....	83
Ilustración 49: Pestaña ACC: Selección de disparadores.....	84

Ilustración 50: Pestaña ECG: Detección de BPM.....	84
Ilustración 51: Pestaña ECG: Periodo de cálculo de BPM.....	85
Ilustración 52: Pestaña ECG: Led de detección y contador de picos.....	85
Ilustración 53: Robot LearnBot.....	86
Ilustración 54: Sistema de tracción diferencial.....	86
Ilustración 55: Placa de desarrollo ODROID-C1.....	87
Ilustración 56: Sistema de control mediante módulo ACC.....	88
Ilustración 57: Sistema de control mediante el módulo EMG.....	89
Ilustración 58: Circuito de prueba.....	90
Ilustración 59: Sujetos realizando las pruebas.....	91

Índice de gráficos

Gráfico 1: Señal EMG típica.....	10
Gráfico 2: Señal típica de tipo EDA.....	11
Gráfico 3: Señal ECG típica.....	12
Gráfico 4: Señal ACC típica. Los tres ejes se encuentras representados.....	13
Gráfico 5: Toma1: 10 contracciones de bíceps a lo largo de 12 segundos.....	22
Gráfico 6: Toma 2: 4 contracciones de bíceps a lo largo de 6 segundos.....	22
Gráfico 7: Toma 3: 6 contracciones de bíceps a lo largo de 10 segundos.....	22
Gráfico 8: Toma 4: 3 contracciones de bíceps a lo largo de 4 segundos y medio.....	23
Gráfico 9: Toma 5: Flexión completa del brazo y relajación posterior.....	23
Gráfico 10: Toma 6: 3 contracciones del bíceps a lo largo de 3 segundos.....	23
Gráfico 11: Toma 7: 7 contracciones de bíceps a lo largo de 11 segundos.....	23
Gráfico 12: Transformada de Fourier, en doble representación. Eje x en bins.....	25
Gráfico 13: Transformada de Fourier, en simple representación. Eje x en Bins.....	25
Gráfico 14: Transformada de Fourier. Simple representación. Eje x en hercios.....	26
Gráfico 15: Transformada de Fourier. Simple representación. Eje x normalizado a la freq. de Nyquist.....	26
Gráfico 16: Toma 1: Transformada de Fourier. Simple representación en hercios.....	27
Gráfico 17: Toma 2: Transformada de Fourier. Simple representación en hercios.....	27
Gráfico 18: Toma 4: Transformada de Fourier. Simple representación en hercios.....	27
Gráfico 19: Toma 5: Transformada de Fourier. Simple representación en hercios.....	27
Gráfico 20: Toma 6: Transformada de Fourier. Simple representación en hercios.....	28
Gráfico 21: Toma 7: Transformada de Fourier. Simple representación en hercios.....	28
Gráfico 22: Filtro de Butterworth Paso Bajo Orden 6, FC=50Hzs.....	30
Gráfico 23: Toma 2: Datos en crudo.....	30
Gráfico 24: Toma 2: Después de pasar por el filtro.....	30
Gráfico 25: Toma 2: Señal filtrada (en rojo) superpuesta a la señal en crudo (en azul).....	31
Gráfico 26: Toma 4: Señal filtrada (en rojo) superpuesta a la señal en crudo (en azul).....	31
Gráfico 27: Toma 7: Señal filtrada (en rojo) superpuesta a la señal en crudo (en azul).....	31
Gráfico 28: Toma 4: Señal después de pasar por el filtro paso bajo y media móvil.....	32
Gráfico 29: Toma 4: Señales superpuestas. Señal original en azul. En rojo tras atravesar el filtro media móvil.....	33
Gráfico 30: Toma 4: Reducción de muestras a razón de 1:40.....	33
Gráfico 31: Toma 4: Cálculo de envolvente. Ventana tamaño = 2. Valor mínimo= 0.1.....	34
Gráfico 32: Toma 4: Punto de activación en 0.35 mV.....	35
Gráfico 33: Ejemplo: Vibraciones en una señal ACC.....	36
Gráfico 34: Toma 1: Movimientos bruscos de 180° durante 7 segundos.....	39

Gráfico 35: Toma 2: Giros suaves de 180° durante 10 segundos.....	39
Gráfico 36: Toma 3: Giros de 180° con temblores.....	39
Gráfico 37: Toma 4: Vibraciones aleatorias a lo largo de 8 segundos.....	39
Gráfico 38: Transformada de Fourier de la Toma1, simple representación en hercios.....	40
Gráfico 39: Transformada de Fourier de la Toma3, simple representación en hercios.....	40
Gráfico 40: Transformada de Fourier de la Toma4, simple representación en hercios.....	40
Gráfico 41: Filtro Butterworth Orden 2 FCS=50Hzs.....	41
Gráfico 42: Señal antes de pasar por el filtro.....	41
Gráfico 43: Señal tras pasar por el filtro.....	41
Gráfico 44: Señal antes de pasar por el filtro.....	42
Gráfico 45: Toma1 tras atravesar un Filtro Media Móvil con ventana M=40 muestras.....	42
Gráfico 46: Toma1 tras atravesar dos veces un Filtro Media Móvil con ventana M=40 muestras....	43
Gráfico 47: Toma 1 tras atravesar tres veces un Filtro Media Móvil con ventana M = 40 muestras.	43
Gráfico 48: Reducción de muestras sobre la señal tras atravesar el Filtro Media Móvil descrito anteriormente.....	43
Gráfico 49: Disparador establecido a 0.3 durante la detección de cambios en el módulo ACC.....	44
Gráfico 50: Toma1: Latidos relajados en un intervalo de 6 segundos.....	48
Gráfico 51: Toma 2: Latidos relajados en un intervalo de 6 segundos.....	48
Gráfico 52: Toma 3: Latidos relajados en un intervalo de 8 segundos.....	48
Gráfico 53: Toma 4: Latidos levemente acelerados en un intervalo de 7 segundos.....	48
Gráfico 54: Toma 1: Transformada de Fourier. Simple representación en hercios.....	49
Gráfico 55: Toma 2: Transformada de Fourier. Simple representación en hercios.....	49
Gráfico 56: Toma 3: Transformada de Fourier. Simple representación en hercios.....	49
Gráfico 57: Toma 4: Transformada de Fourier. Simple representación en hercios.....	49
Gráfico 58: Función de transferencia. Filtro Butterworth Grado 2 FCS=50Hz.....	50
Gráfico 59: Toma 2 antes de pasar por el filtro.....	50
Gráfico 60: Toma 2 después de pasar por el Filtro.....	50
Gráfico 61: Toma 2 tras atravesar un Filtro de Media Móvil con ventana M=40 muestras.....	51
Gráfico 62: Toma 2 tras atravesar los filtros anteriores y reducir su número de muestras.....	51
Gráfico 63: Toma 2 en su proceso de detección de impulsos.....	52

Índice de fórmulas

Fórmula 1: Filtro Media Móvil (valor absoluto).....	32
Fórmula 2: Filtro Media Móvil.....	42

Agradecimientos

A Alejandro Hidalgo, autor del primer prototipo de la aplicación. Su trabajo ha sido muy útil durante el comienzo de este Proyecto de Fin de Carrera.

A Mario Haut y Mercedes Paoletti, cuya inestimable ayuda ha sido esencial en la parte final del trabajo para poder implementar el componente y conectarlo con LearnBot.

A José Moreno mi tutor, por su apoyo, paciencia y flexibilidad. Sin su ayuda no podría haber llegado hasta aquí.

A Robolab, que me han proporcionado apoyo y material. Siempre han sido amables y atentos conmigo.

A la Escuela Politécnica de Cáceres, después de todos estos años han logrado convertirme en un ingeniero competente.

A mis jefes y compañeros de Univérsitas, por ofrecerme flexibilidad y medios para llevar a cabo este fin.

A mis alumnos en la Academia. Los que fueron, los que son y los que serán, pues son un apoyo y una motivación extra.

A mi familia, amigos y novia, por su apoyo, paciencia y comprensión a lo largo de todos estos años.

A todos ellos, gracias.

Parte I: Presentación de la tecnología, Estado del Arte y Propuestas

1. Qué es Bitalino.

Bitalino es un kit de adquisición de bajo coste, basado en el estándar Arduino, especialmente indicado para el aprendizaje y desarrollo de aplicaciones que empleen el uso de señales corporales y biométricas.

Está diseñado para estudiantes, profesores, programadores, artistas, investigadores, desarrolladores I+D y no requiere de habilidades previas dentro de la electrónica.

Bitalino es capaz de adquirir información del cuerpo humano mediante sensores y transmitir estas señales al ordenador mediante el uso de distintos métodos de comunicación, destacando sobre todo el Bluetooth.

2. Descripción general

Este es el aspecto que presenta el dispositivo:

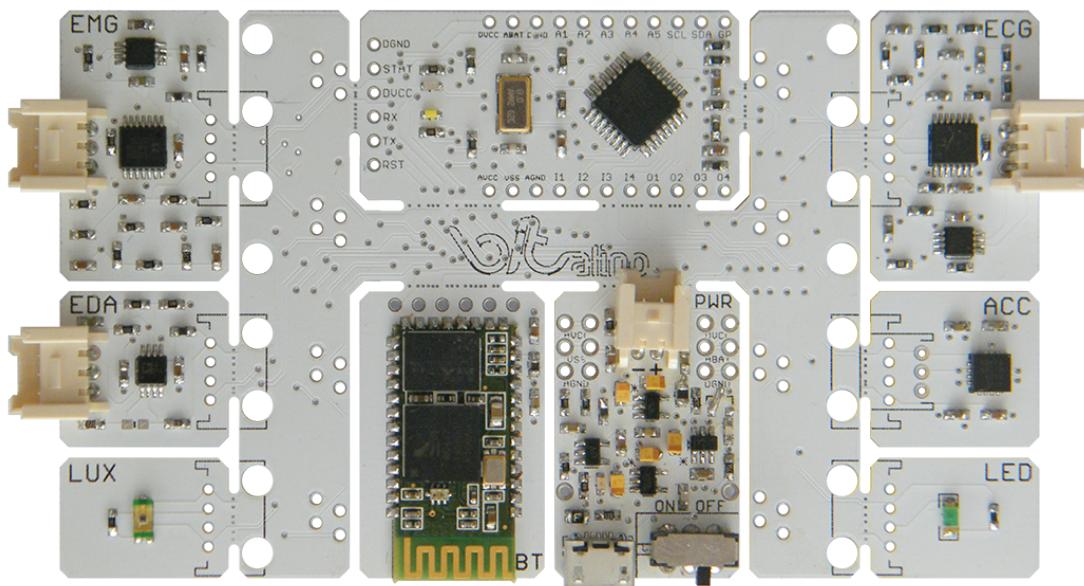


Ilustración 1: Placa de desarrollo bitalino

Especificaciones técnicas:

- Frecuencia de muestreo: Configurable a 1, 10, 100 o 1000Hz
- Puertos analógicos: 4 entradas de 10 bits y 2 entradas de 6 bits
- Puertos digitales: 4 entradas de 1 bit y 4 salidas de 1 bit.
- Enlace de datos: Bluetooth v2.0 Clase II (alcance de hasta 10 metros)
- Sensores: EMG, ECG, EDA, Acelerómetro, Luz (se describen más adelante)
- Actuadores: LED
- Peso: 30 gramos
- Tamaño: 100x60 mm

3. Descripción de los componentes

3.1 MCU - Unidad Micro Controladora

El bloque MCU convierte las señales analógicas de los sensores a formato digital, y muestrea todos los canales. Proporciona acceso a los canales digitales y analógicos, así como a los periféricos.

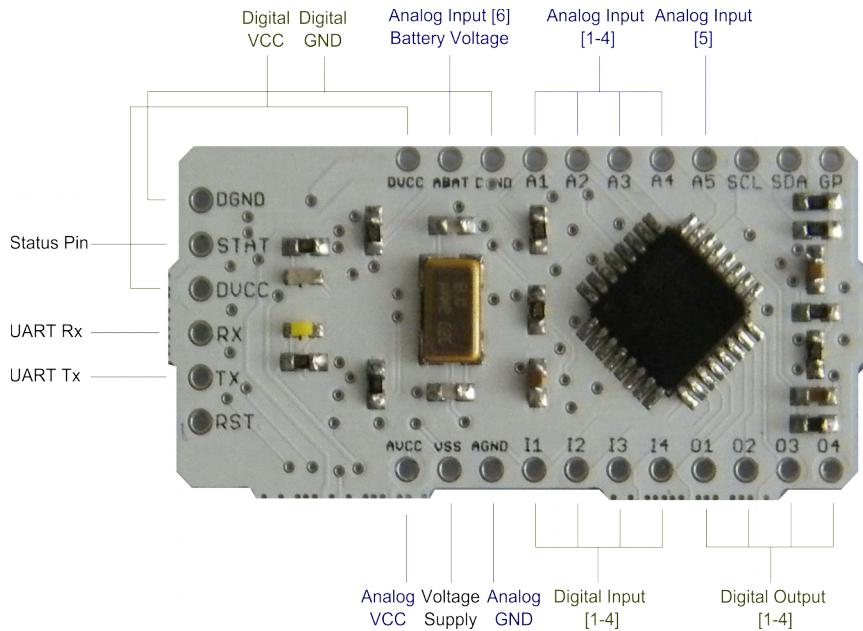


Ilustración 2: Módulo MCU

3.2 PWR - Alimentación

El bloque de control de alimentación proporciona energía a todos los bloques de BITalino. Este módulo además dispone de un cargador incorporado que controla el estado de la batería. La carga del dispositivo se realiza mediante un puerto Micro-USB cuando el dispositivo está apagado.

El bloque también proporciona acceso a señales de control, su esquemático es el siguiente:

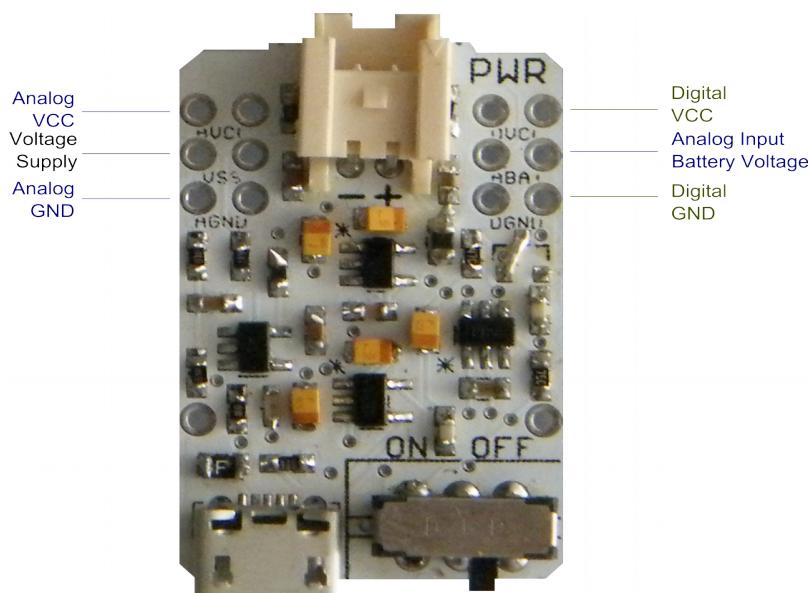


Ilustración 3: Módulo PWR

3.3 EMG - Electromiógrafo

La activación muscular involucra la acción de músculos y nervios, los cuales son accionados por corrientes eléctricas muy pequeñas. Medir la actividad eléctrica en los músculos y nervios puede ser un medio útil de interacción Hombre-Máquina.

El sensor es capaz de realizar medidas electromiográficas usando electrodos bipolares de superficie.

Una representación típica de este tipo de señales:

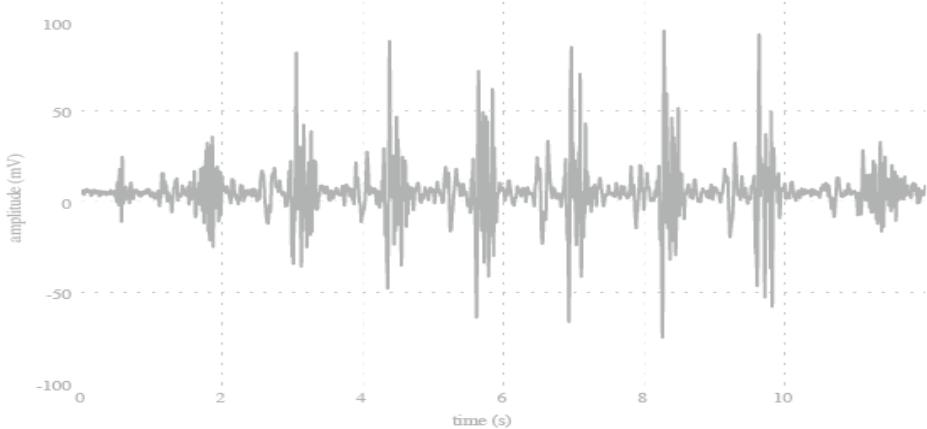


Gráfico 1: Señal EMG típica

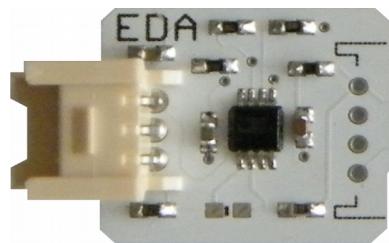
Aplicaciones

La electromiografía de superficie es una técnica que es usada en muchas aplicaciones clínicas y biomédicas, en áreas como HCI (Interacción Hombre-Máquina), neurología, rehabilitación, ortopedia, ergonomía, deportes, etc. Es ampliamente utilizada como herramienta de bio retro alimentación para detectar la fatiga muscular, desordenes de control-motor y lumbalgia.

La detección de actividad muscular isométrica, cuando no se ha producido ningún movimiento, permite una definición de clases de movimientos de gestos sutiles no motores que sirven para controlar interfaces involuntariamente y sin alterar el entorno. Estas señales pueden ser usadas para controlar dispositivos prostéticos tales como manos prostéticas, brazos y extremidades inferiores, así como para controlar dispositivos electrónicos como por ejemplo teléfonos móviles o PDA's, también puede emplearse como señales de control para algunos tipos de videojuegos interactivos.

3.4 EDA - Actividad Electro Dérmbica

La Actividad Electro-Dérmbica puede ser definida como la carga eléctrica que se transmite a través de ciertos elementos presentes en la piel, asociada con la actividad de las glándulas sudoríparas y producida por cualquier estímulo que genere alerta o una respuesta orientativa. El sensor EDA es capaz de medir la actividad en la piel con una gran capacidad sensitiva. Con un señal de bajo ruido y con un circuito de acondicionamiento y amplificación, somos capaces de proporcionar unas capacidades de sensitividad precisas y detectar incluso las más mínimas respuestas electro dermales.



*Ilustración 4: Módulo EDA
Actividad Electro Dérmitica*

Señal típica:

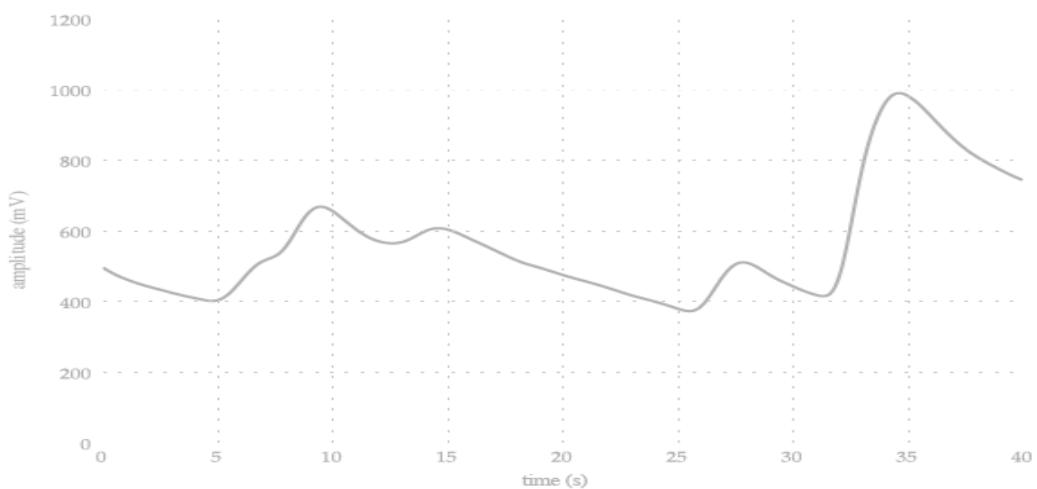


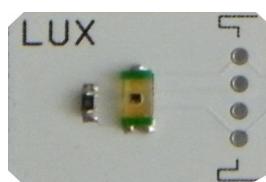
Gráfico 2: Señal típica de tipo EDA

Aplicaciones:

Algunas de las aplicaciones de este sensor incluyen la detección de cambios en la atención y en los estados cognitivos y emocionales. Los sensores EDA han sido usados para la bio adquisición del estado de relajación, detección de reacciones comprensivas del sistema nervioso central entre otras aplicaciones.

3.5 LUX - Foto diodo

Un foto diodo es un tipo de foto detector capaz de convertir luz en corriente o voltaje. En este caso la salida dada es voltaje.



*Ilustración 5:
Módulo LUX*

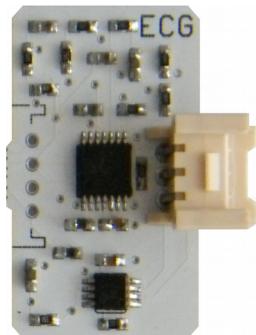
Aplicaciones

Usando una fuente de luz apropiada, este sensor puede ser utilizado para la sincronización de la adquisición de datos fisiológicos respecto a focos potenciales de luz, aunque puede usarse también para medir la luz ambiente, o usado conjuntamente con una fuente de luz para monitorizar los datos de presión sanguínea por ejemplo.

3.6 ECG - Electro Cardiograma

La conducción de acciones potenciales a través del corazón genera corrientes eléctricas que pueden ser captadas por electrodos sujetos en la piel. La monitorización de los cambios eléctricos que acompañan a los latidos del corazón es llamado electrocardiograma. Las variaciones en tamaño y duración de las ondas de un ECG son útiles para diagnosticar ritmos cardiacos anormales y patrones conductivos de interés.

Los ECG funcionan principalmente mediante la detección y amplificación de los minúsculos cambios que se producen en la piel y que son causados durante el ciclo que realizan los músculos cardiacos durante cada latido.



*Ilustración 6:
Módulo ECG*

ECG típico:

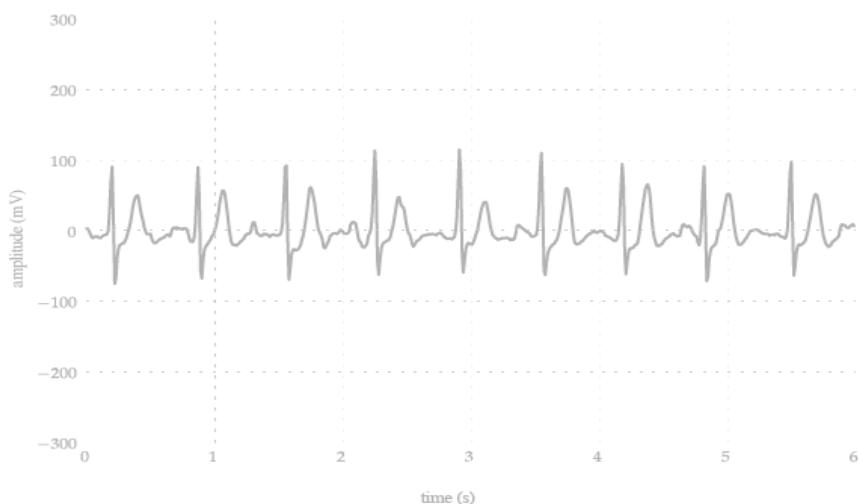


Gráfico 3: Señal ECG típica

Aplicaciones

La más importante de las aplicaciones de los sensores ECG es centrarse en el bienestar del paciente y en estudiar su ritmo cardíaco y monitorizar su estrés mediante la verificación vital y biométrica.

3.7 ACC - Acelerómetro tri-axial

El acelerómetro tri axial está basado en la tecnología MEMS (Micro Sistemas Electro-Mecánicos) y ha sido desarrollado para aplicaciones biométricas donde la cinemática y las medidas de movimiento son requeridas. Este sensor puede medir la aceleración relativa a la caída libre y el modelo disponible es capaz de detectar la magnitud y dirección de esta misma aceleración, como un vector de cantidad. Este vector resultante puede ser usado para sentir la posición, vibración, un golpe, caída, etc...

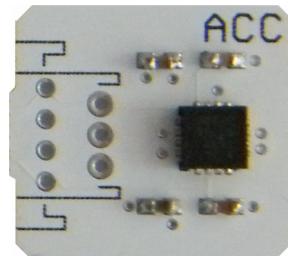


Ilustración 7:
Módulo ACC

Representación de una señal típica ACC:

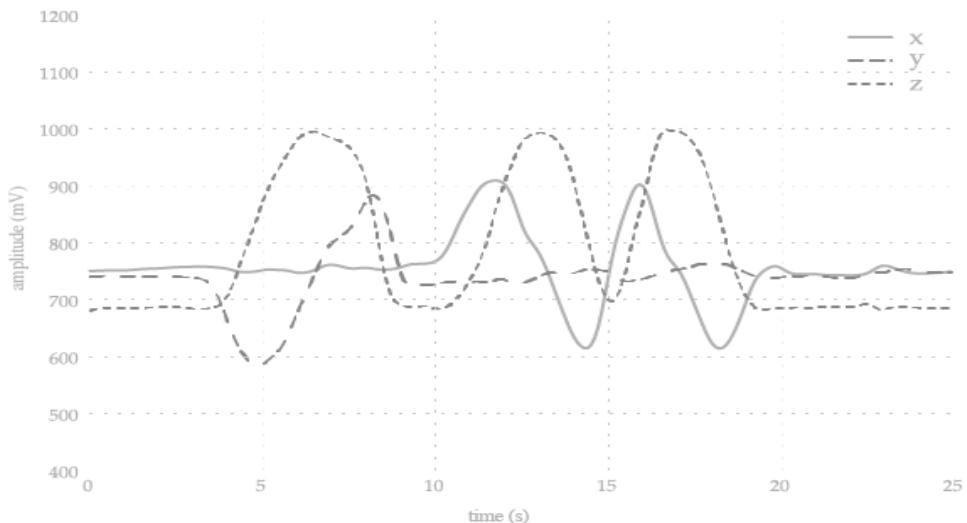


Gráfico 4: Señal ACC típica. Los tres ejes se encuentran representados

Aplicaciones

Hoy en día los acelerómetros son usados ampliamente en áreas tan diferenciadas como la ingeniería, biología, monitorización de edificios, biomecánica, etc. Incluso los teléfonos móviles y las consolas de videojuegos los llevan. La medición de vibraciones, particularmente en la investigación de aplicaciones y estudios del movimiento dinámico y el análisis biomecánico.

Estos tipos de sensores acelerómetros han sido diseñados para aplicaciones que involucren lecturas de acelerometría continuas o intermitentes en usuarios. Dependiendo de la aplicación puede requerirse un sistema de adquisición con hasta 3 puertos analógicos disponibles, en un intento de usar todas las medidas funcionales. El mismo sensor puede ser empleado para aplicaciones biaxiales y uniaxiales.

4. Estado del Arte

La plataforma objeto de este estudio lleva en el mercado desde 2013, sin embargo cuenta ya con cerca de una veintena de publicaciones, en ramas tan variadas como la monitorización del sueño, la inteligencia artificial, la neurociencia o los videojuegos.

El carácter poliédrico del dispositivo lo convierte en una herramienta atractiva dentro de ámbitos de investigación muy diversos, acotaremos el análisis al campo de la robótica y la inteligencia artificial y estudiaremos algunas de las publicaciones más interesantes que se han publicado en los últimos dos años.

4.1 Versatilidad del cuerpo humano para el control a través de interfaces electromiográficas de bajo coste

Versatility of human body control through low-cost electromyographic interface

ANTONIO BERNARDINO, YVES RYBARCZYK, JOSE BARATA

Faculdade de Ciencias e Tecnologia, Universidade Nova de Lisboa

Departamento de Engenharia Electrote

Se trata de un interesante estudio publicado en 2014 en el que se demuestra que el cuerpo humano puede servir de control para manejar dispositivos mecánicos a través de la interfaz EMG que proporciona Bitalino.

En el citado estudio se propone un método de ajuste y calibración para permitir controlar una mano robótica a través de los músculos abdominales, con resultados prometedores.

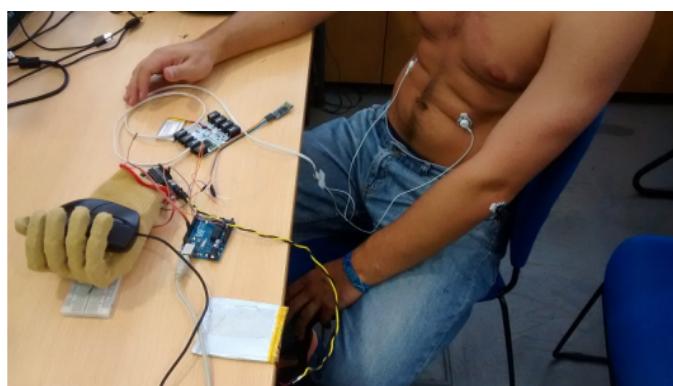


Ilustración 8: Control de un brazo robótico por medio de EMG

El sistema es capaz de ajustarse para cada individuo, aunque necesita de un proceso de calibrado previo. Los resultados demuestran que los usuarios que pasan por este proceso obtienen un rendimiento razonablemente bueno sobre el sistema, en contraposición con los usuarios que no han pasado por este proceso:

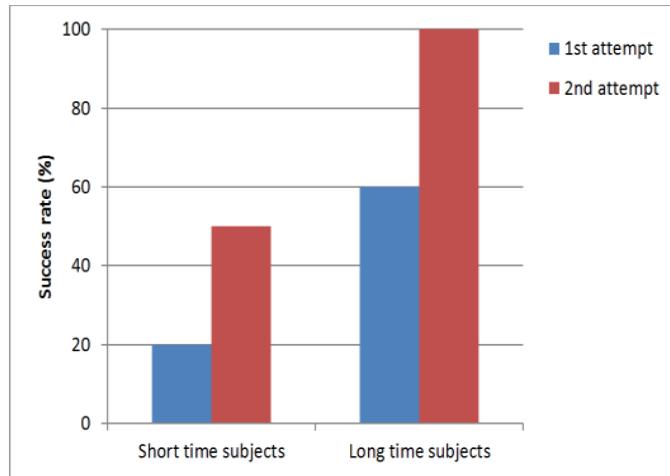


Ilustración 9: Porcentajes de éxito en las pruebas

A raíz de este trabajo se vislumbran líneas de desarrollo interesantes, pues este sistema parece idóneo para personas con una minusvalía física que tengan problemas para manejar sus extremidades superiores y precisen de una asistencia que podría ser brindada a través de la robótica.

4.2 Una valoración de los sensores EMG de un solo canal para el reconocimiento gestual

An Assessment of Single-Channel EMG Sensing for Gestural Input

Dartmouth Computer Science Technical Report TR2015-767

Travis Peters

Este trabajo valora la interfaz EMG como herramienta de adquisición de información gestual.

Se plantea un pipeline para el reconocimiento dividido en 5 fases. La adquisición se realiza mediante un brazalete de sensores, se propone también un método de entrenamiento para establecer fronteras de decisión.

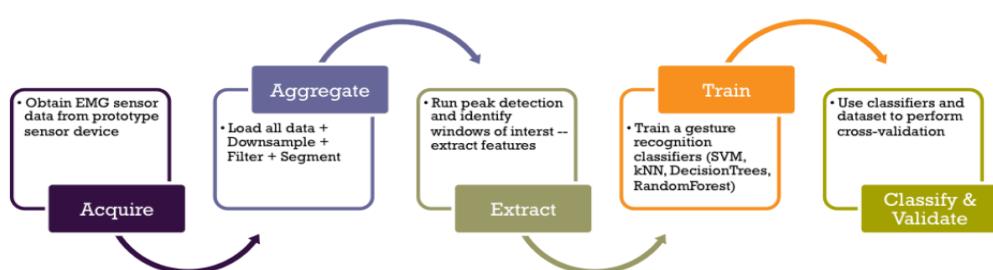


Ilustración 10: Esquema del Pipeline de 5 fases

El sistema es capaz de reconocer 5 gestos diferentes haciendo uso de la muñeca y el antebrazo.

Los resultados son aparentemente buenos, sin embargo el sistema se ve lastrado por varios inconvenientes: es fuertemente dependiente del lugar donde se coloquen los sensores, la vía de entrada es única y puede presentar ruido y por último el sistema tiene un tiempo de respuesta de al menos 2 segundos que impide el procesamiento en tiempo real.

A pesar de las limitaciones expuestas, el estudio abre la puerta a la utilización de los sensores EMG como medio de reconocimiento de emociones. Este sistema puede ser un buen discriminador en el contexto de un sistema más amplio que mejore el alfabeto gestual con la inclusión de giroscopios y acelerómetros.

4.3 Evaluación de una Aplicación reconocedora de contexto para el control de robots móviles por medio de datos fisiológicos: El Caso de Estudio ToBITas

Evaluation of a Context-Aware Application for

Mobile Robot Control Mediated by

Physiological Data: The ToBITas Case Study

Borja Gamecho, José Guerreiro, Ana Priscila Alves, André Louren, Hugo Plácido da Silva, Luis Gardeazabal, Julio Abascal and Ana Fred

Egokituz Laboratory, University of the Basque Country, 20018 Donostia, Spain

PIA Group, Instituto de Telecomunicações, 1049-001 Lisboa, Portugal

Este estudio propone una aplicación para el control de robots móviles mediante el uso de dos sensores EMG y un acelerómetro conectados al brazo derecho del sujeto en pruebas.

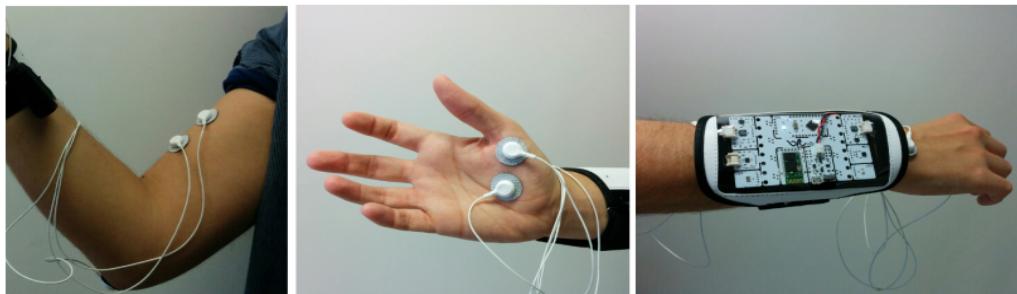


Ilustración 11: Ubicación del brazalete

El software desarrollado para el reconocimiento es ejecutado en un smartphone y es capaz de ofrecer resultados razonablemente buenos sobre una población de 30 individuos sin aprendizaje previo.

Si bien el estudio se centra en las capacidades del algoritmo desarrollado, el método de adquisición resulta interesante para el desarrollo de líneas de trabajo similares dirigidas a manipular sistemas diversos en el campo de la interacción hombre-máquina.

5. Propuestas

Del breve estudio del estado del arte que hemos desarrollado en el punto nº4 se pueden constatar dos principales vías de trabajo e investigación para la adquisición de señales con aplicación en nuestro ámbito de estudio:

- Las interfaces de Bitalino (especialmente EMG y ACC) **como medio de control** sobre dispositivos móviles en el campo de la robótica y en el del desarrollo de software con accesibilidad.
- Las interfaces de Bitalino **como medio de reconocimiento gestual y emocional** en el campo de la inteligencia artificial y la robótica.

Se propone el desarrollo de algoritmos de adquisición y evaluación enfocado a propósitos específicos y el desarrollo de librerías y APIs que permitan explotar el potencial de las mencionadas interfaces para nuestros sistemas de trabajo.

Nuestro objetivo consistirá en diseñar un componente de software integrado en nuestro framework de robótica capaz de adquirir, analizar y utilizar de forma útil las señales biométricas proporcionadas por bitalino.

El trabajo va a consistir en tres partes claramente diferenciadas:

1. Análisis y estudio de las muestras de datos proporcionadas por los diferentes componentes de bitalino, enfocado a un procesado informático de las mismas.
2. Diseño y desarrollo de un componente de software capaz de responder a nuestras necesidades, integrado en el marco de nuestro framework de trabajo en el laboratorio, Robocomp.
3. Pruebas experimentales. Adquisición y uso de señales biométricas en un entorno real o simulado. Conclusiones del trabajo.

Parte II: Adquisición, Estudio y Análisis de Señales Biométricas

6. Electromiógrafo: Adquisición y procesado de señales EMG

Como mencionamos previamente en el punto 3.3, nuestro dispositivo bitalino es capaz de proporcionarnos información biométrica procedente de sensores ubicados sobre la piel. Esta información representa el potencial eléctrico que atraviesa la piel entre los dos puntos en los que decidamos ubicar nuestros sensores.

Una diferencia de potencial eléctrico significativa significará que el músculo que se encuentra debajo de dicho sensor se ha accionado, ya sea voluntaria o involuntariamente y puede utilizarse como elemento de comunicación hombre-maquina. Aunque el potencial eléctrico que circula por un músculo puede ser muy grande, nuestros electrodos están posicionados en la superficie de la piel, y la intensidad de corriente que llega a la epidermis es muy baja, por lo que las variaciones que seremos capaces de detectar no excederán en ningún caso de 1,65mV.

El presente punto desarrollará el estudio y medición de dicha información eléctrica: su adquisición, su filtrado y procesado para obtener medidas más eficientes, la detección de picos, el establecimiento de fronteras de decisión y finalmente la implementación de nuestro prototipo de software.

6.1 Adquisición

La adquisición de información eléctrica se llevará a cabo mediante electrodos auto adhesivos. Los electrodos son intercambiables y existe una gran variedad de modelos a utilizar pero en nuestro estudio nos centrarnos en aquellos electrodos auto-adhesivos y con gel incorporado:

6.1.1 Electrodos

Nuestro dispositivo EMG se compone de tres electrodos. Dos de ellos (con cable de color blanco) serán los electrodos principales que medirán la diferencia de potencial entre los dos puntos especificados de la superficie del cuerpo. El tercer electrodo (con cable de color negro) servirá como electrodo de referencia y estará ubicado en una zona del cuerpo con conductividad neutra, como por ejemplo un codo, articulación o el extremo de un hueso.

A continuación describimos los modelos de electrodos que utilizaremos en nuestro trabajo.

6.1.1.1 Electrodo reutilizable



Ilustración 12: Electrodo reutilizable

Diámetro: 22mm

Grosor: 1mm

Núcleo: De plata/cloruro de plata y recubierto de polímero.

Tipo de gel: hidro-gel conductor y adhesivo

Las pruebas con este tipo de electrodo ofrecen lecturas útiles y razonablemente limpias, pero su valor como electrodo reutilizable queda en entredicho, poniéndose de manifiesto rápidamente su baja eficiencia al ser reutilizado, convirtiéndose de esta manera en un electrodo desechable.

6.1.1.2 Electrodo desechable



Ilustración 13: Electrodo desechable

Diámetro: 24mm

Grosor: 1mm

Núcleo: De plata/cloruro de plata y recubierto de polímero.

Tipo de gel: hidro-gel conductor y adhesivo

Si bien este tipo de electrodo solo permite ser utilizado una vez, ofrece una respuesta más fiable ya que la señal se recibe un poco más limpia y no es tan sensible a la colocación como lo es el electrodo anterior. En lo sucesivo emplearemos este electrodo como base para nuestras pruebas.

6.1.2. Colocación

Si bien nuestros electrodos son susceptibles de ser utilizados con cualquier músculo del cuerpo que esté en contacto con la piel, por motivos ergonómicos nos interesa utilizar ubicaciones pertenecientes al tronco superior del cuerpo, preferiblemente en los brazos para una activación sencilla y voluntaria de los impulsos eléctricos.

Apoyándonos en el estudio de Travis Peters (Dartmouth Computer Science Technical Report TR2015-767) seleccionaremos las siguientes dos ubicaciones para nuestras capturas de datos: bíceps y antebrazo.

Resultan posiciones de trabajo cómodas, que se puede realizar el estudio sentado y disponer de ubicaciones cercanas para colocar nuestro dispositivo.

6.1.2.1 Bíceps

Colocando los electrodos a una distancia aproximada de 10 cms sobre la superficie del bíceps, y colocando el electrodo de referencia en el codo del sujeto podremos realizar lecturas eléctricas sobre los movimientos de dicho músculo.

Esta ubicación proporciona una señal limpia, pero de baja intensidad.

El sujeto debe aprender como mover el bíceps de manera adecuada para generar señales limpias y definidas, ya que el músculo proporcionará lecturas de potencial tanto en su contracción como en su relajación, generando perfiles de señal difíciles de interpretar.

En esta posición es recomendable ubicar el brazo en una postura cómoda y realizar contracciones espasmódicas del bíceps si deseamos trabajar con una señal fácil de caracterizar.

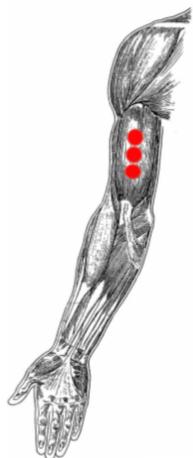


Ilustración 14: Diagrama de colocación de los sensores en el bíceps

6.1.2.2 Antebrazo

Esta ubicación es interesante, ya que nos permite leer contracciones y movimientos realizados por la mano, así como por la muñeca, ya que ambos generar corrientes en esa zona del brazo.

Proporciona señales claras y es más como de utilizar.

Los electrodos deben posicionarse en la zona que ilustra la fotografía mientras el electrodo de control se coloca en el codo al igual que hacemos al utilizar el bíceps.

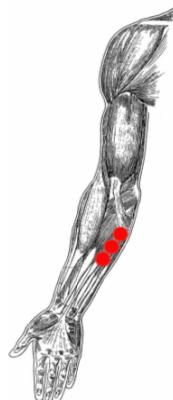


Ilustración 15: Diagrama de colocación de los sensores en el antebrazo

6.1.3 Función de transferencia

Bitalino nos ofrece valores digitales de la lectura eléctrica que proporcionan los electrodos a una frecuencia de muestreo de 1, 10, 100 o 1000 muestras por segundo, dependiendo de la configuración que le indiquemos. Nos interesa obtener una señal lo mas detallada y con la mejor resolución posible, de modo que trabajaremos con lecturas a 1000 muestras por segundo, y posteriormente podemos reducir el número de muestras si lo consideramos oportuno.

El dispositivo es capaz de proporcionarnos muestras digitales con un valor que oscila entre 0 y 1023, siendo el 0 equivalente a un potencial de -1,65 mV y el 1023 equivalente a +1,65mV. Se requiere pues realizar la transformación oportuna para adaptar este formato a una representación más fiel de la señal.

El fabricante nos ofrece una función de transferencia adecuada a las características del dispositivo:

$$EMG(mV) = [-1.65 \text{ mV}, +1.65 \text{ mV}]$$

$$EMG(V) = \frac{\left(\frac{ADC}{2^n} - \frac{1}{2}\right) \cdot VCC}{G_{EMG}}$$

$$EMG(mV) = EMG(V) \cdot 1000$$

$$VCC = 3.3 \text{ V} (\text{voltaje operativo})$$

$$G_{EMG} = 1000 (\text{ganancia del sensor})$$

$$EMG(V) = \text{Valor del EMG en voltios (V)}$$

$$EMG(mV) = \text{Valor del EMG en milivoltios (mV)}$$

$$ADC = \text{Valor sampleado del canal}$$

$$n = \text{Número de bits del canal, típicamente } n=10$$

6.2 Estudio y Caracterización de la señal

Para el estudio de nuestra señal generaremos 7 tomas mediante nuestro software prototipo del cual hablaremos más adelante, las tomas tienen una duración de entre 3 y 12 segundos y han sido generadas utilizando la posición del bíceps mediante contracciones espasmódicas y en diferente cantidad para cada toma.

6.2.1 Muestras de estudio

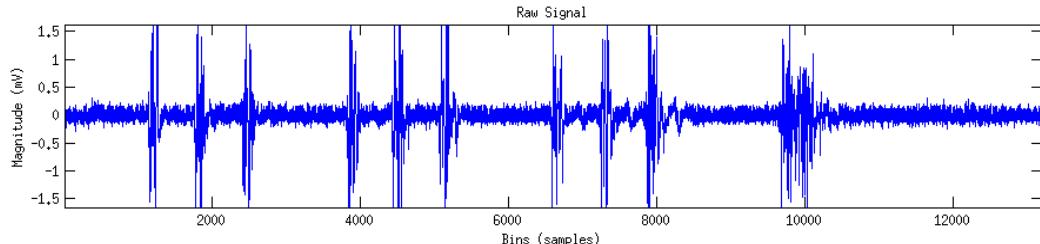


Gráfico 5: Toma 1: 10 contracciones de bíceps a lo largo de 12 segundos

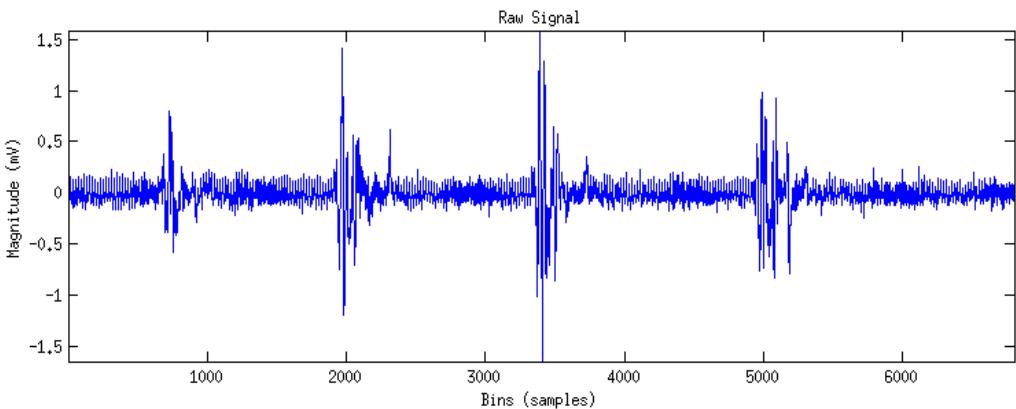


Gráfico 6: Toma 2: 4 contracciones de bíceps a lo largo de 6 segundos

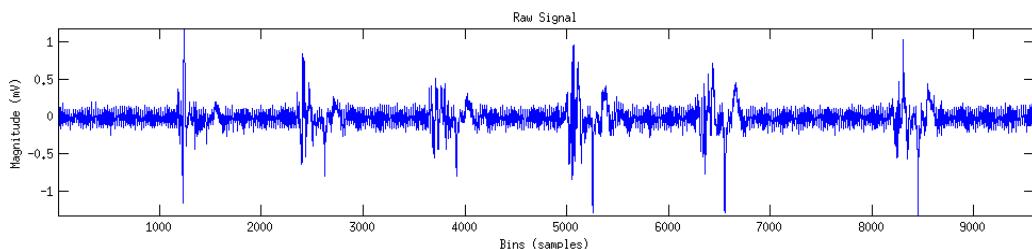


Gráfico 7: Toma 3: 6 contracciones de bíceps a lo largo de 10 segundos

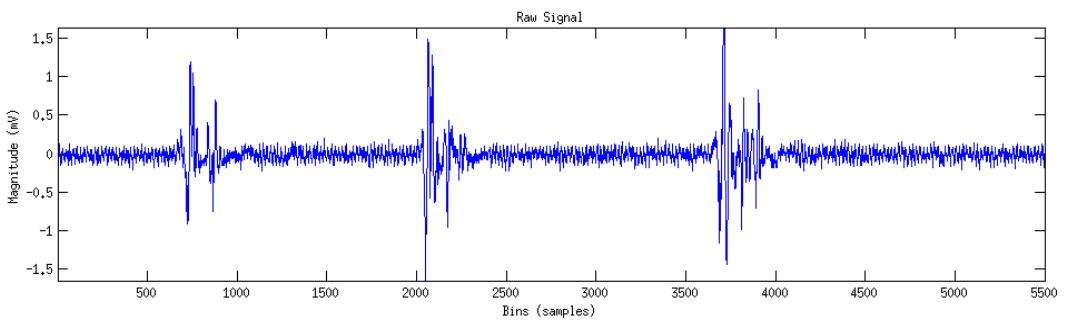


Gráfico 8: Toma 4: 3 contracciones de bíceps a lo largo de 4 segundos y medio

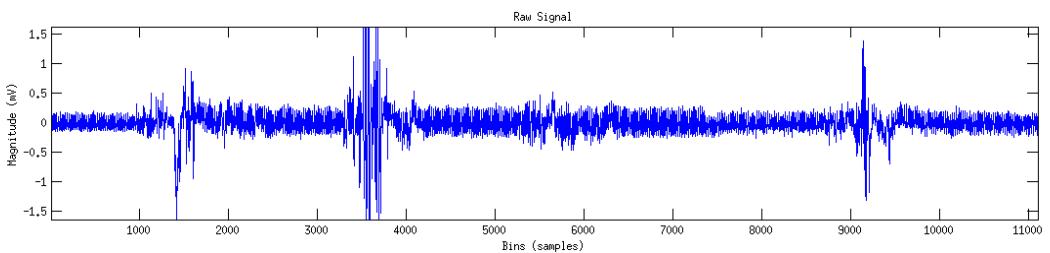


Gráfico 9: Toma 5: Flexión completa del brazo y relajación posterior

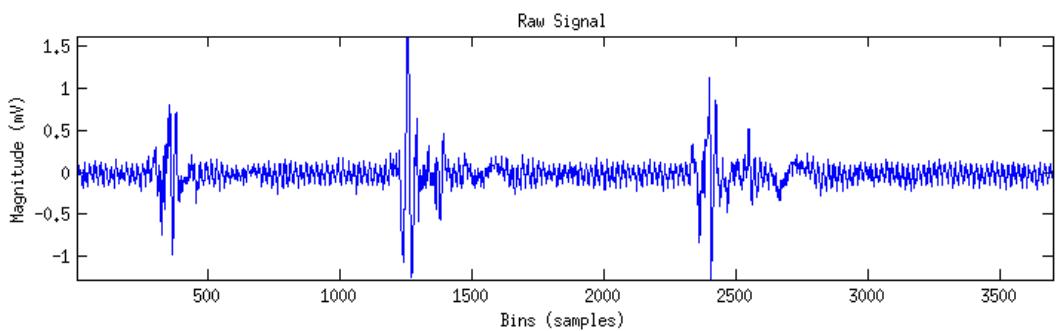


Gráfico 10: Toma 6: 3 contracciones del bíceps a lo largo de 3 segundos

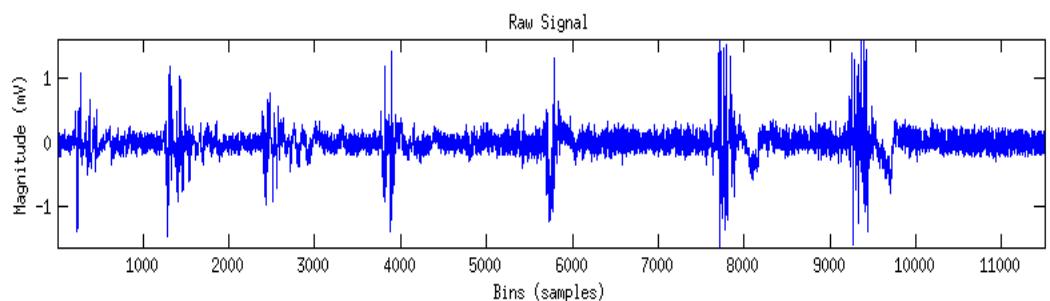


Gráfico 11: Toma 7: 7 contracciones de bíceps a lo largo de 11 segundos

Con una población de tomas significativas procedemos al análisis buscando patrones comunes y características de la señal que sean de nuestro interés.

En este punto nuestro prototipo es capaz de exportar los datos en un fichero “.txt” con el objetivo de realizar el estudio matemático de la señal en una herramienta adecuada para nuestro propósito como es MATLAB.

Para cada toma presentada con anterioridad, realizaremos su transformación al dominio de la frecuencia, mediante la transformación de Fourier oportuna, con el objeto de observar en qué zonas del espectro se encuentran sus componentes más significativas, aplicar filtros sobre la señal y comparar resultados.

A continuación se detalla el código en MATLAB empleado a tal efecto:

6.2.2 Análisis de las muestras en MATLAB

6.2.2.1 Análisis espectral de las muestras

Tomando como ejemplo la Toma 3, procedemos a analizar la señal en el dominio de la frecuencia utilizando el siguiente script:

```
%% Cargamos los datos y mostramos la señal en crudo
x = importdata('toma1b.txt');
plot(x);
xlabel('Bins (samples)');
ylabel('Magnitude (mV)');
title('Raw Signal');
num_bins_x = length(x);
fs = 1000; % 62.5 samples per second
fnyquist = fs/2; %Nyquist frequency
axis tight

%% Espectro en frecuencias doble representacion Magnitud/Bins no es correcto al 100% porque empieza
en 1, en lugar de en 0 (por eso pone almost!)
x_f = abs(fft(x));
plot(x_f);
xlabel('Frequency (Bins - almost!)')
ylabel('Magnitude');
title('Double-sided Magnitude spectrum');
axis tight

%% Espectro en frecuencias doble representacion corregido
x_f = abs(fft(x));
fax_bins = [0 : num_bins_x-1];
plot(fax_bins, x_f);
xlabel('Frequency (Bins)');
ylabel('Magnitude');
title('Double-sided Magnitude spectrum');
```

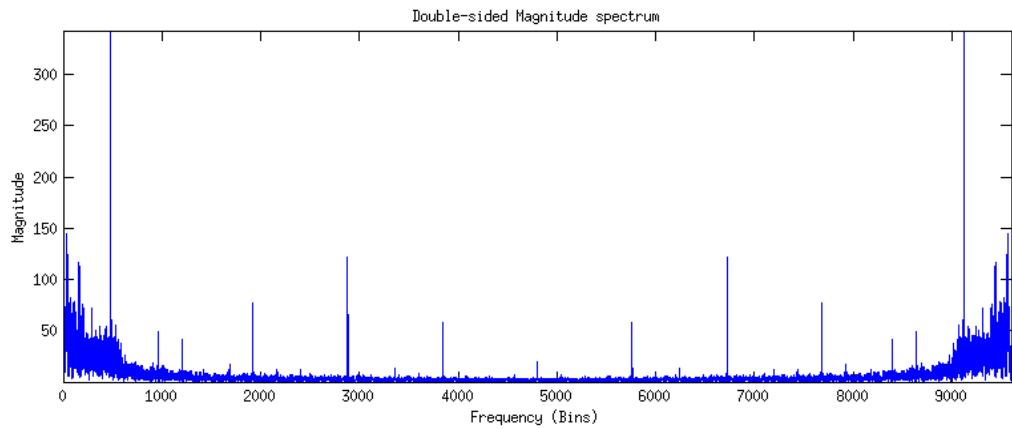


Gráfico 12: Transformada de Fourier, en doble representación. Eje x en bins

```
%% Espectro en frecuencias simple representación en bins
N_2 = ceil(num_bins_x/2);
plot(fax_bins(1:N_2), x_f(1:N_2))
xlabel('Frequency (Bins)')
ylabel('Magnitude');
title('Single-sided Magnitude spectrum (bins)');
axis tight
```

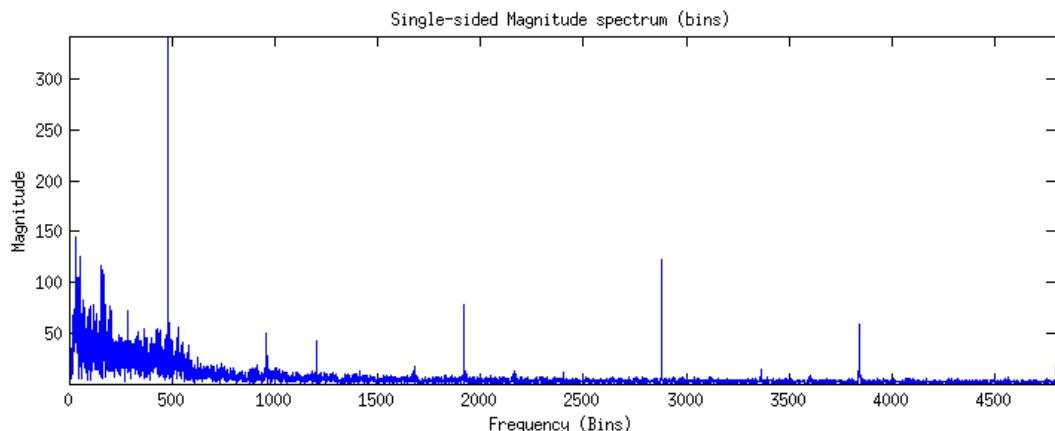


Gráfico 13: Transformada de Fourier, en simple representación. Eje x en Bins

```
%% Espectro en frecuencias simple representación en herzios
% Cada muestra esta separada por fs/num_bins_x Herzios.
bin_vals = [0 : num_bins_x-1];
fax_Hz = bin_vals*fs/num_bins_x;
N_2 = ceil(num_bins_x/2);
plot(fax_Hz(1:N_2), x_f(1:N_2))
xlabel('Frequency (Hz)')
ylabel('Magnitude');
title('Single-sided Magnitude spectrum (Hertz)');
axis tight
```

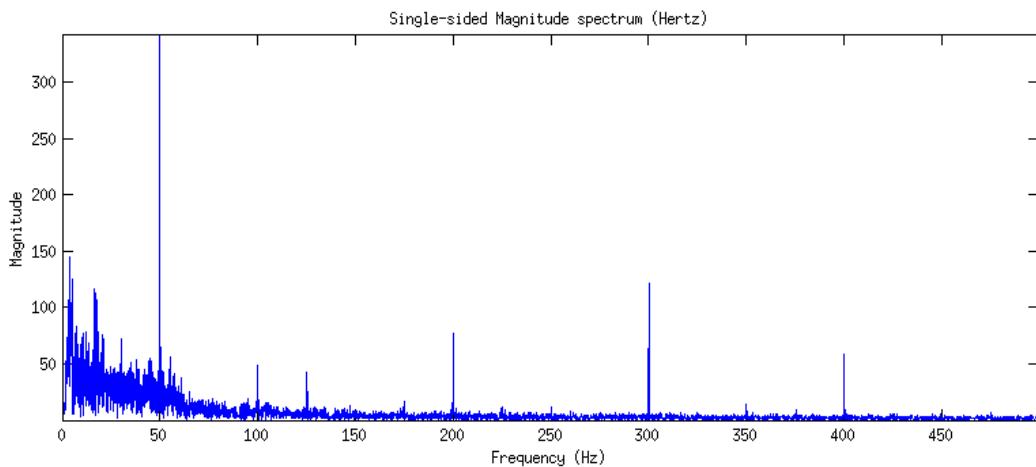


Gráfico 14: Transformada de Fourier. Simple representación. Eje x en hercios

```
%% Espectro en frecuencias simple representación con eje x normalizado
% Normalizado respecto a la frecuencia de Nyquist = fs/2
bin_vals = [0 : num_bins_x-1];
fax_norm = (bin_vals*fs/num_bins_x)/fnyquist; % same as bin_vals/(N/2)
N_2 = ceil(num_bins_x/2);
%plot(fax_norm(1:N_2), x_f(1:N_2))
% Otra forma:
plot([0:1/(num_bins_x/2 -1):1], x_f(1:num_bins_x/2));
xlabel({'Frequency (Normalised to Nyquist Frequency'...
    '1=Nyquist frequency)'});
ylabel('Magnitude');
title('Single-sided Magnitude spectrum (Normalised to Nyquist)');
axis tight
```

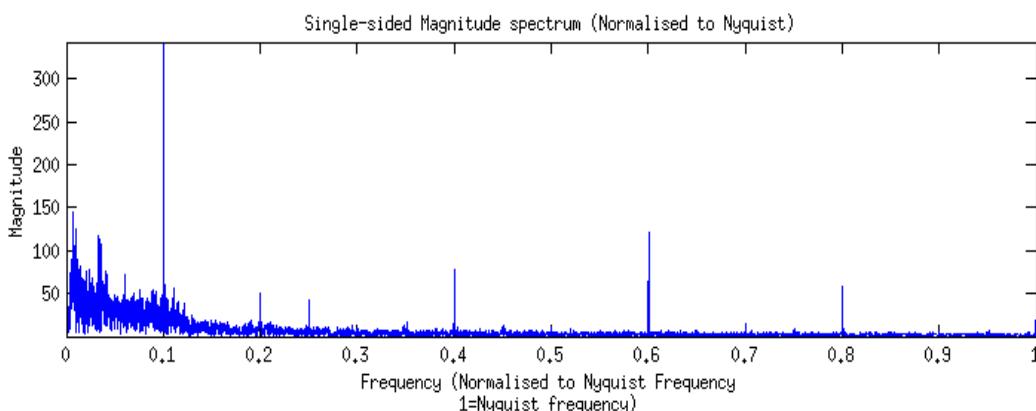


Gráfico 15: Transformada de Fourier. Simple representación. Eje x normalizado a la freq. de Nyquist

El proceso anterior arroja interesantes resultados. Podemos observar como la señal presenta sus componentes más significativas en frecuencias bajas puesto que aquellas componentes que se encuentran antes de llegar a 50 hercios tienen mayor amplitud.

La señal tiende a atenuarse a partir de 50 hercios hasta casi volverse imperceptible.

Podemos observar además ruido impulsivo a lo largo de todo el espectro de la señal, particularmente intenso en 50 hercios. Esto puede deberse a alguna señal de origen

electromagnético que interfiera con nuestro dispositivo, algún aparato cercano, como unos auriculares, una fuente de alimentación o el propio ordenador portátil.

A continuación se muestra la Transformada de Fourier para el resto de tomas estudiadas, con objeto de comparar sus perfiles y encontrar similitudes:

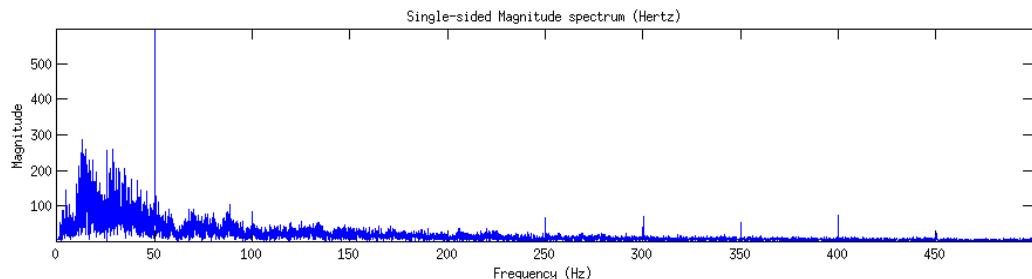


Gráfico 16: Toma 1: Transformada de Fourier. Simple representación en hercios

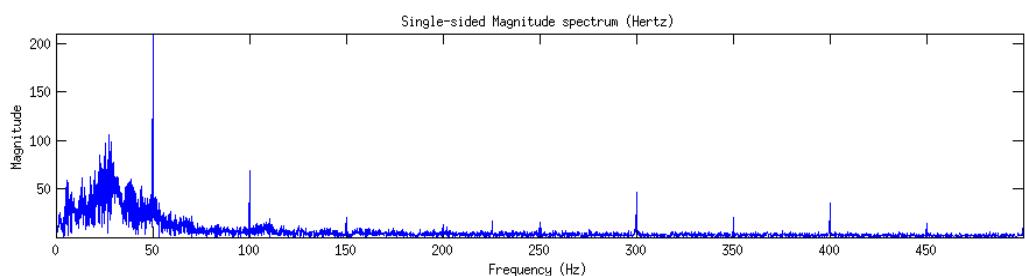


Gráfico 17: Toma 2: Transformada de Fourier. Simple representación en hercios

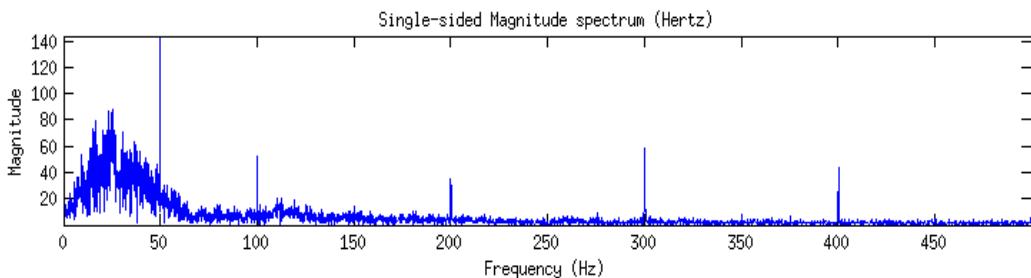


Gráfico 18: Toma 4: Transformada de Fourier. Simple representación en hercios

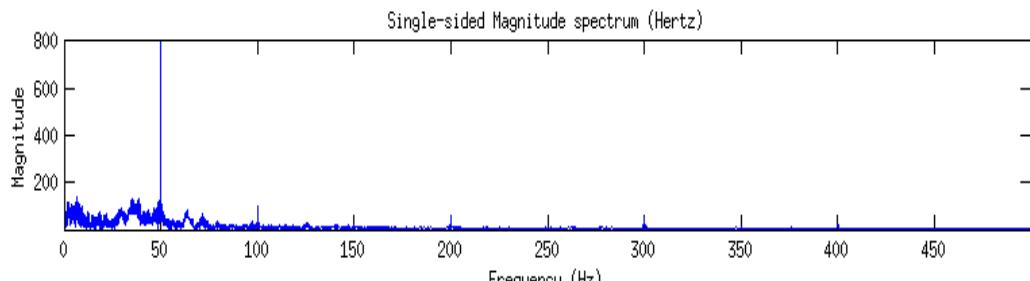


Gráfico 19: Toma 5: Transformada de Fourier. Simple representación en hercios

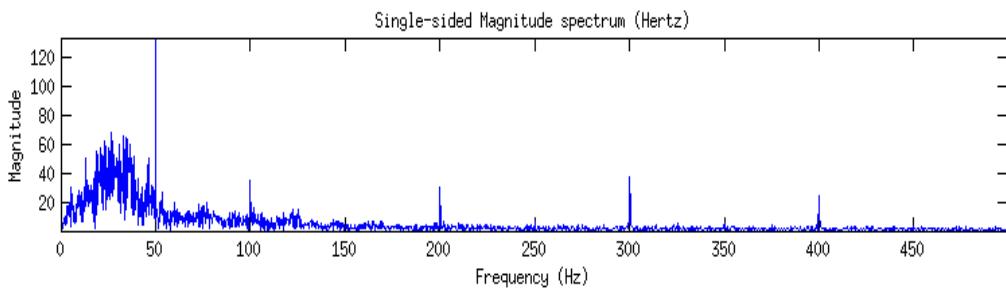


Gráfico 20: Toma 6: Transformada de Fourier. Simple representación en hercios

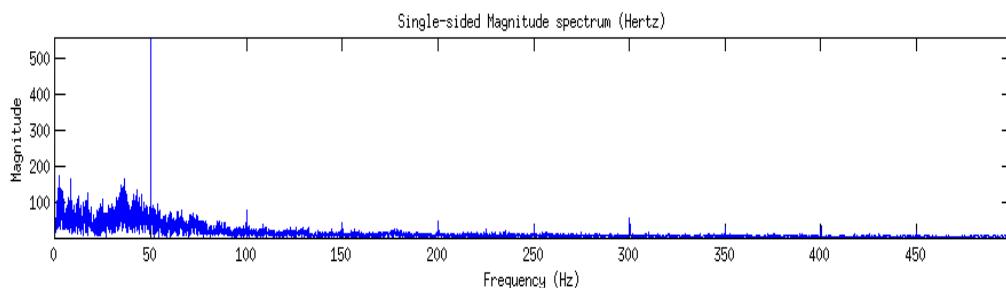


Gráfico 21: Toma 7: Transformada de Fourier. Simple representación en hercios

Puede observarse que todas las tomas reúnen características muy similares. La mayor parte de sus componentes significativas están presentes por debajo de 50 hercios.

También es interesante destacar que el ruido impulsivo se localiza en los mismos puntos para todas las tomas estudiadas, independientemente de su duración, aunque en algunas tomas el ruido es menos intenso que en otras.

Hacer pasar la señal por un filtro paso bajo con una frecuencia de corte de 50 hz parece la opción más lógica, puesto que mantendrá el perfil de la señal a cero en las zonas de menor potencia, excluyendo el ruido impulsivo que se observa a partir de ese punto.

Resultan discutibles no obstante los casos en los que la señal se encuentra muy saturada de impulsos en un corto periodo de tiempo, como por ejemplo en la Toma 5 y en la Toma 7, pues en este caso se observan componentes útiles hasta 60 o 70hzs aproximadamente. En estos casos el filtrado no dañará la señal, puesto que se mantendrán sus componentes de mayor potencia intactas y la proporción Señal/Ruido mejorará a pesar de todo, pero es cierto que podría emplearse un filtro menos restrictivo en aras de preservar mayor cantidad de información útil.

La opción más inmediata es utilizar un filtro de Butterworth o bien de Chebyshev con un orden lo suficientemente elevado para que no deteriore la señal por debajo de la frecuencia de corte, pero lo suficientemente pequeño para que no ponga en compromiso la eficiencia del sistema.

6.2.2.2 Filtrado

En el siguiente gráfico se muestra una comparativa con la respuesta en frecuencias de los dos filtros

prouestos:

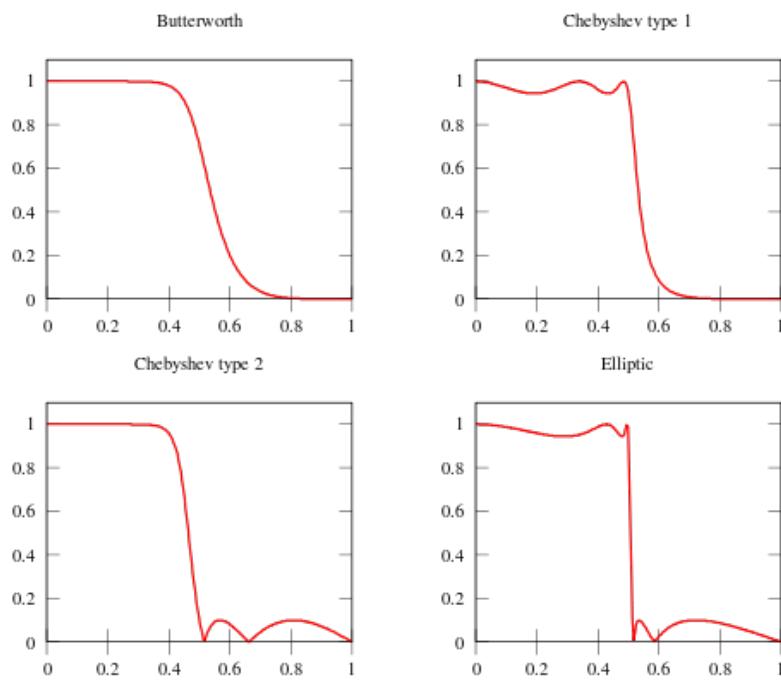


Ilustración 16: Gráfico comparativo - Filtros tipo Butterworth y Chebyshev

En el gráfico anterior podemos observar que si bien ambos tipos de filtro sirven perfectamente a nuestro propósito, el filtro de tipo Chebyshev introducirá perturbaciones en nuestra señal justo antes de llegar a la frecuencia de corte, lo cual es indeseable dado el reducido ancho de banda con el que vamos a trabajar.

El filtro de Butterworth en cambio muestra una respuesta perfectamente lineal hasta llegar a la Frecuencia de Corte, pero después su caída no es pronunciada y la señal presentará componentes no deseados en las frecuencias posteriores próximas a la frecuencia de corte.

Este aspecto del filtro de Butterworth lo podemos solucionar parcialmente si trabajamos con un filtro de orden 4 o superior, y si establecemos la frecuencia de corte en 40hz, para que la curva de descenso empiece un poco antes de llegar al punto deseado.

Dado que nuestro filtro va a ser implementado en C++, debemos ser cautos a la hora de elegir el orden de mismo: un orden elevado generará un filtro preciso, pero el número de cálculos matemáticos será elevado.

Hemos de establecer un compromiso entre eficacia y eficiencia para que nuestra aplicación se desenvuelva adecuadamente.

Después de realizar las pruebas pertinentes, nos decantamos por un filtro de Butterworth de orden 6, con una frecuencia de corte en 50 hercios, si bien desplazaremos la curva de descenso un poco antes en el espectro para lograr que no pase ninguna componente que se encuentre fuera de nuestra frecuencia de corte.

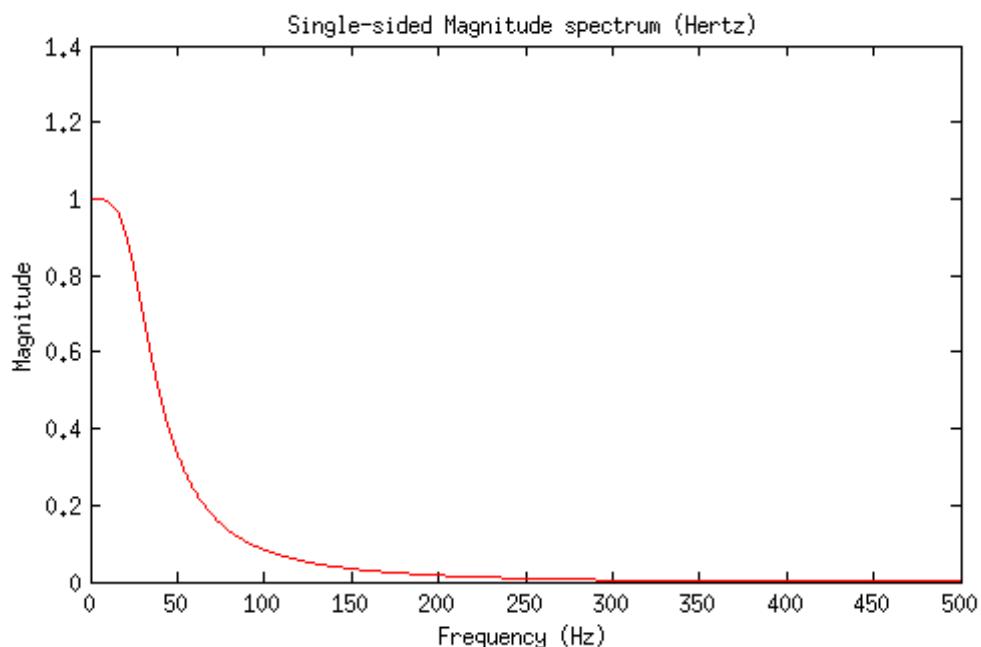


Gráfico 22: Filtro de Butterworth Paso Bajo Orden 6, FC=50Hzs

Seleccionando como ejemplo la toma nº 2, podemos observar como actúa el filtro sobre nuestra señal.

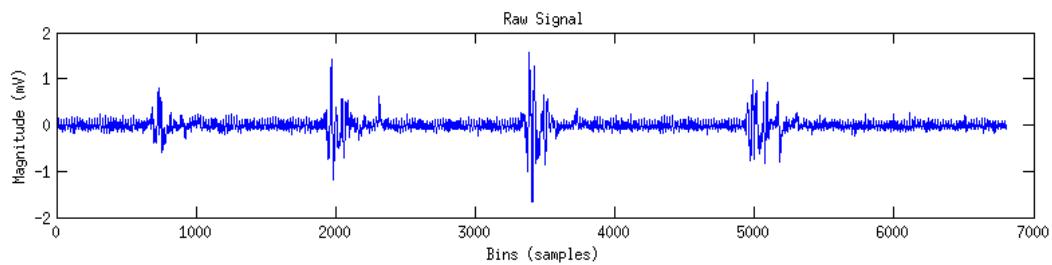


Gráfico 23: Toma 2: Datos en crudo

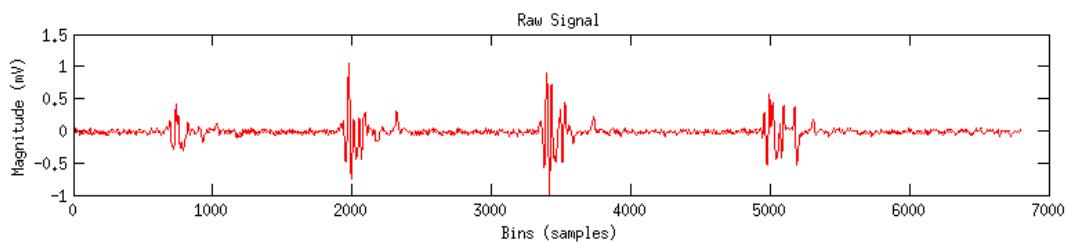


Gráfico 24: Toma 2: Despues de pasar por el filtro

La superposición de ambas gráficas es bastante reveladora:

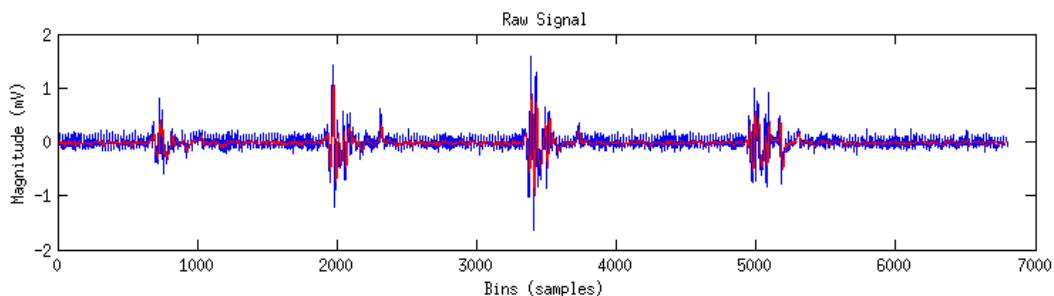


Gráfico 25: Toma 2: Señal filtrada (en rojo) superpuesta a la señal en crudo (en azul)

Al estudiar la señal en el dominio del tiempo, después de pasar por nuestro filtro, observamos que suaviza su amplitud en las zonas donde no hay impulsos, pero se mantiene prácticamente igual en las zonas que detectan alta actividad eléctrica.

Podemos considerar que el filtro es adecuado, pues mejora sustancialmente la relación Señal/Ruido.

El resto de tomas estudiadas, presentan una desenvoltura similar, como puede apreciarse en las gráficas de las tomas 4 y 7 respectivamente:

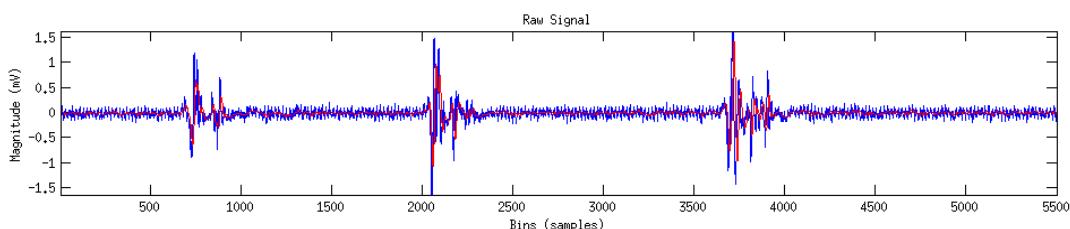


Gráfico 26: Toma 4: Señal filtrada (en rojo) superpuesta a la señal en crudo (en azul)

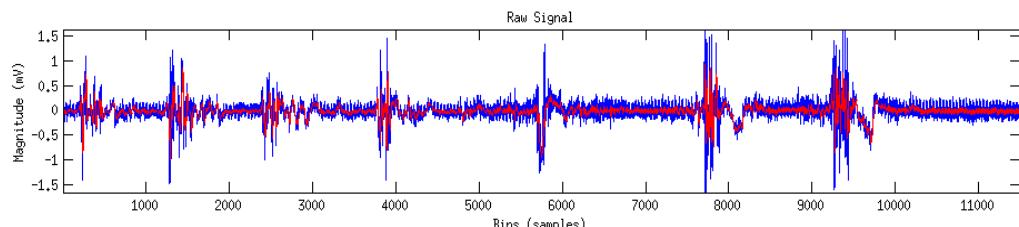


Gráfico 27: Toma 7: Señal filtrada (en rojo) superpuesta a la señal en crudo (en azul)

El resto de tomas muestran resultados similares. Se considera pues que nuestro filtro trabaja correctamente y es capaz de eliminar una cantidad importante del ruido de nuestra señal.

Nuestro prototipo incorporará varios de los filtros que hemos estudiado, en distintas variantes en función de su orden y punto de corte con la intención de hacer el software lo más funcional y versátil posible, si bien el filtro más adecuado y recomendable es el Butterworth de Orden 6 con punto de corte en 50hz.

6.2.3 Procesado de la señal

Una vez que hemos logrado obtener una representación de la señal lo más detallada y limpia posible, nuestro siguiente objetivo es ser capaces de detectar en qué punto se produce un cambio significativo de voltaje, porque esto significaría que se ha activado el músculo.

Dicho de otro modo, deseamos ser capaces de detectar cuando alguien ha movido el músculo de forma voluntaria y cuando el músculo se encuentra en reposo.

La señal actual, una vez transformada al pasar por el filtro paso bajo, presenta los siguientes inconvenientes:

- Sigue siendo una señal muy inestable, ya que sus valores oscilan constantemente entre voltajes negativos y positivos, incluso cuando la señal se encuentra en reposo.
- El número de muestras con el que trabajamos es muy elevado y dispara los tiempos de procesado en nuestro software.

Por ello vamos a realizar sucesivas transformaciones a dicha señal que resultarán en un conjunto de datos mucho más adecuados y manejables para llevar a cabo nuestro propósito.

6.2.3.1 Filtro de media móvil

Se propone la utilización de un filtro de media móvil. El propósito de dicho filtro será suavizar las muestras y establecer una buena aproximación para computar la envolvente de nuestra señal EMG.

El proceso se basa en la utilización de una ventana deslizante M que podemos establecer con un valor propuesto de 40 muestras, aunque nuestro prototipo permitirá parametrizar dicho valor.

$x[n]$ es la señal de entrada y $s[n]$ es nuestra señal de salida.

$$s[n] = \frac{1}{M} \sum_{i=0}^{M-1} |x[n-i]|$$

Fórmula 1: Filtro Media Móvil (valor absoluto)

El proceso da como resultado una señal con un perfil mucho más interesante para nuestra aplicación, ya que transforma las componentes negativas en positivas y aproxima los valores siguiendo una progresión predecible.

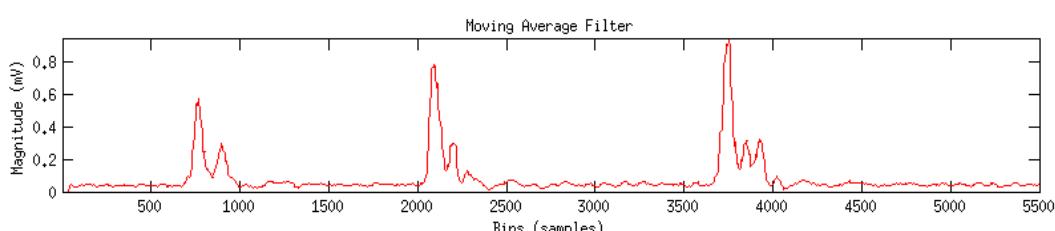


Gráfico 28: Toma 4: Señal después de pasar por el filtro paso bajo y media móvil

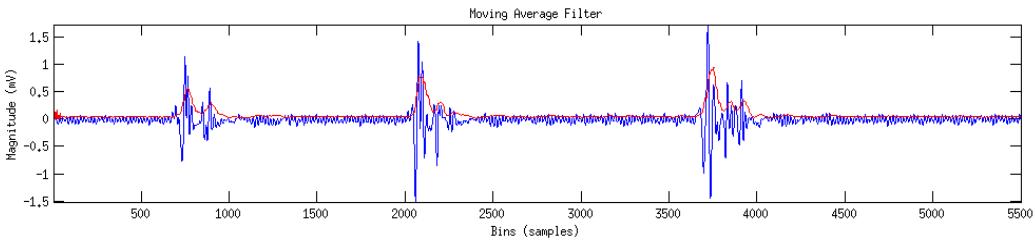


Gráfico 29: Toma 4: Señales superpuestas. Señal original en azul. En rojo tras atravesar el filtro media móvil.

Como puede observarse, el filtro media móvil es una buena herramienta para lograr una primera aproximación en nuestro objetivo de delimitar las regiones de interés.

El filtro funciona aceptablemente bien estableciendo una ventana deslizante con un tamaño de 40 muestras. Cuanto menor sea la ventana deslizante, más se asemejará el resultado a la señal original. La perdida de información será mínima, pero la señal presentará algunos de los problemas mencionados anteriormente.

Una ventana deslizante de mayor tamaño dará como resultado una señal más suave y con menos altibajos, pero la altura de nuestras regiones de interés será menor y resultará más difícil establecer las fronteras de detección.

6.2.3.2 Reducción de muestras (Downsampling)

Hemos establecido inicialmente una frecuencia de muestreo de 1000 muestras por segundo a la hora de adquirir los datos. Esta configuración responde a la necesidad de obtener una señal con la máxima resolución posible, pero una vez obtenida esta señal, puede interesarnos reducir el número de muestras para trabajar con un conjunto de datos menor que agilice los procesos y suavice aún más nuestra señal.

Una vez realizados los procesos anteriormente descritos, podemos procesar la señal resultante y tomar muestras de nuevo, a intervalos regulares, dando como resultado una señal más manejable.

En el siguiente gráfico se muestra el aspecto que tendrá nuestra señal si tomamos muestras a en una proporción de 40 a 1, es decir, una muestra por cada 40 muestras de la señal original:

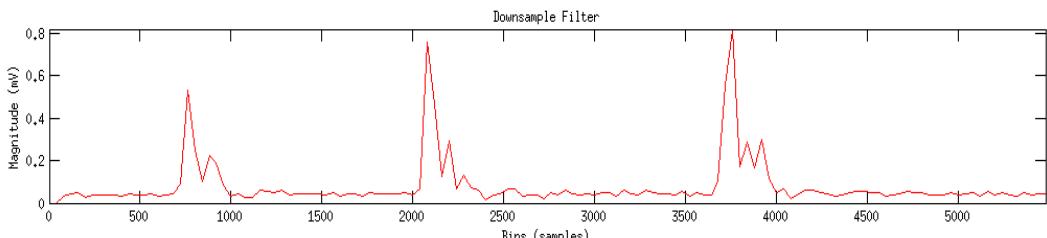


Gráfico 30: Toma 4: Reducción de muestras a razón de 1:40

Una reducción tan drástica dará como resultado una señal cuya verdadera frecuencia de muestreo es de $1000/40 = 25$ muestras por segundo. Sin embargo estas muestras son enormemente significativas y resultan más que suficientes para llevar a cabo nuestro propósito.

A pesar de todo, la señal procesada aún presenta una característica indeseable: su valor oscila rápidamente en determinados intervalos. Las oscilaciones en intervalos cortos pueden dar lugar a falsos positivos, por lo que aplicaremos un detector de envolvente que describimos a continuación.

6.2.3.3 Envolvente de la señal

Un detector de envolvente puede ser de utilidad para hacer que nuestra señal tenga un perfil todavía más sencillo.

Nuestra intención es poder detectar con facilidad en qué puntos la señal sobrepasa un cierto valor que denominaremos *disparador* o *threshold*.

Una señal con oscilaciones frecuentes en un intervalo pequeño conducirá a errores de lectura y a falsos positivos ya que la señal sobrepasará una y otra vez nuestro disparador en un corto periodo de tiempo.

Para solucionar este problema se propone capturar la envolvente de la señal, desechando cualquier punto de la misma que no constituya un máximo local.

Un máximo local será una muestra de nuestra señal que tenga una magnitud superior a la de un conjunto de muestras precedentes y un conjunto de muestras posteriores.

Por lo tanto, necesitamos una vez más una ventana deslizante, que servirá para comparar nuestra muestra actual en un entorno definido.

Si tenemos en cuenta que nuestra señal en este punto está siendo muestreada a razón de 25 mps, no debemos excedernos en el tamaño de dicha ventana, ya que bastará con comparar nuestra muestra actual con la muestra anterior y posterior en ese punto. El buen funcionamiento de este detector está condicionado al tamaño de la ventana con relación a la frecuencia de muestreo de la señal origen.

Definir esta ventana con un valor de 0, significará que todas las muestras son consideradas máximos locales y que por lo tanto nuestro filtro devolverá la misma señal que recibió de entrada.

Dado que nuestra señal tiene zonas de reposo donde no se alcanzan máximos, será importante definir un valor mínimo de magnitud a partir de la cual se realizará el estudio, aceptando como válida cualquier muestra por debajo de este valor.

El aspecto que presentará la señal estableciendo un entorno de tamaño 2 y un valor mínimo de 0.1 se muestra a continuación:

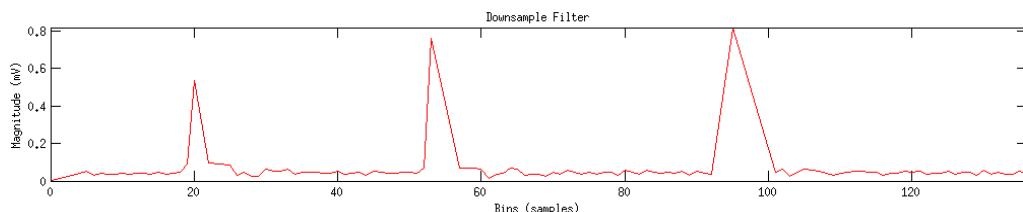


Gráfico 31: Toma 4: Cálculo de envolvente. Ventana tamaño = 2. Valor mínimo= 0.1

Como puede observarse, la señal ahora presenta picos que delimitan perfectamente las zonas de la señal en las cuales se ha producido un impulso eléctrico considerable. No presenta oscilaciones peligrosas que puedan dar lugar a falsos positivos y permanece en un valor cercano a 0 cuando no se produce ningún movimiento en el músculo.

Véase que se han perdido gran cantidad de muestras por el camino. De las 5500 muestras que conformaban la señal inicialmente, nos hemos quedado con tan solo 138 tras los sucesivos filtros aplicados.

Es importante mencionar que el tiempo de computación en el caso de este último filtro no es excesivamente grande debido a la reducción en el número de muestras que hemos establecido en el punto 6.2.3.2.

6.2.3.4 Detección de picos

Ahora que disponemos de una señal limpia, en la cual se puede discriminar claramente las regiones de interés, podemos establecer nuestro algoritmo de detección.

Estableceremos un valor *disparador* o *threshold* que servirá de punto de activación cada vez que una muestra de nuestra señal lo supere, manteniéndose activado siempre que la señal se mantenga por encima de ese valor y desactivándose en caso de que la magnitud de nuestra señal caiga.

El valor de nuestro threshold debe ser escogido cuidadosamente, y un valor propuesto razonable sera el 50% de la altura promedio que alcancen nuestros impulsos.

Es pues necesario una fase de observación o calibrado que permita establecer este valor, dado que la altura de los impulsos es variable en función del músculo utilizado o incluso de la ejecución de los movimientos del sujeto de estudio.

Si utilizamos como ejemplo la toma nº 4, podemos observar que existen 3 impulsos, que presentan valores máximos de 0.53, 0.75 y 0.82 mV respectivamente. Esto nos da un promedio de 0.7mV. Si aplicamos la función propuesta, un buen punto de activación será 0.35mV.

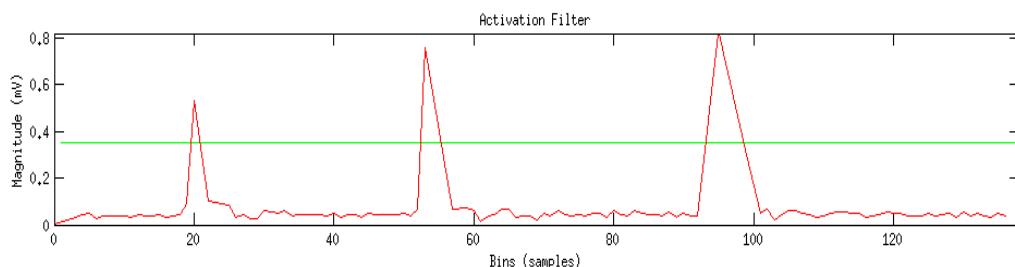


Gráfico 32: Toma 4: Punto de activación en 0.35 mV

Un filtro correctamente configurado dará lugar a una detección del impulso con éxito. En este punto la ausencia de falsos positivos o falsos negativos esta condicionada a tres factores:

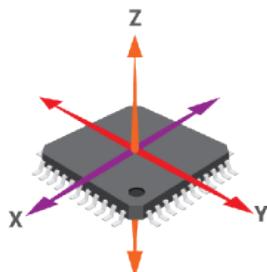
1. Electrodos en buen estado y correctamente colocados
2. El sujeto debe generar movimientos musculares lo suficientemente intensos e impulsivos en la medida de lo posible, que generen un contraste importante entre las zonas con baja y alta diferencia de potencial en nuestra señal.
3. Filtros correctamente configurados, de acuerdo al tipo de señal introducida. Ventanas de tamaño adecuado de acuerdo con el número de muestras con los que trabajemos.

Si tenemos en cuenta los 3 puntos mencionados, nuestro sistema será capaz de detectar sin problemas la activación muscular, por lo que podemos considerar que el estudio y caracterizado de la señal es un objetivo resuelto.

Nuestro esfuerzo se va a dirigir hacia la implementación de un prototipo de procesado que nos permita el estudio de la señal en tiempo real.

7. Acelerómetro: adquisición y procesado de señales ACC

En el punto 3.7 hemos hablado brevemente del acelerómetro tri-axial de bitalino. Como comentamos anteriormente, un acelerómetro de este tipo puede proporcionarnos tres vectores que nos informan de la aceleración en fuerzas G que esta sufriendo el chip con respecto a su posición de acuerdo con la figura que se muestra a continuación.



*Ilustración 17:
Sistema de referencia
en un chip ACC típico*

En el ámbito de la interacción hombre-maquina esta información puede ser útil para conocer la posición del chip con respecto a un sistema de referencia, para detectar un cambio de posición, o simplemente una vibración o perturbación en el mismo. Poniendo un ejemplo cotidiano, podemos descubrir el uso de acelerómetros en prácticamente cualquier teléfono moderno para conocer su posición relativa y descubrir cuando el usuario ha girado o cambiado de posición el dispositivo.

7.1. Adquisición

Este módulo no requiere de parches ni de contacto directo con la piel del sujeto. Sera necesario únicamente colocar el chip en la posición deseada de acuerdo con el sistema de referencia que adoptemos para poder medir los cambios de aceleración con respecto a cualquiera de sus tres ejes.

Debe tenerse muy en cuenta que el dispositivo es enormemente sensible a cualquier cambio de velocidad en sus ejes, por lo que si se desea medir dichos cambios, no nos interesa que haya vibraciones en exceso en el dispositivo, ya que de lo contrario tendremos que esperar a que dichas perturbaciones se detengan para obtener una lectura clara.



Gráfico 33: Ejemplo: Vibraciones en una señal ACC.

7.1.1 Sistema de referencia

Es importante conocer el sistema de referencia que utiliza nuestro módulo. El siguiente esquema pretende ilustrar de forma sencilla la posición de nuestros vectores de aceleración respecto a la placa de adquisición de bitalino.

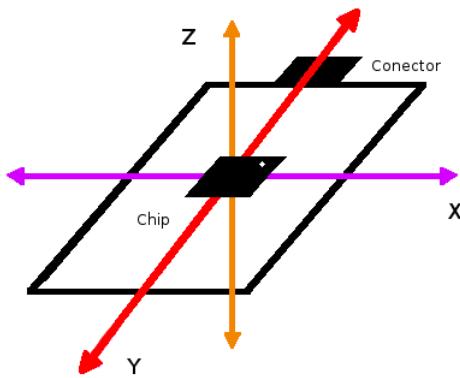


Ilustración 18: Sistema de referencia en el módulo ACC de bitalino

7.1.2 Función de transferencia

Al igual que el módulo EMG, utilizaremos una frecuencia de muestreo de 1000 muestras por segundo, por las mismas razones expuestas en el punto 6.1.3

El dispositivo es capaz de proporcionarnos los datos de aceleración en forma de muestras digitales con un valor que oscila entre 0 y 1023, siendo el 0 equivalente a una aceleración de -3G y el 1023 equivalente a +3G.

Recordemos que G es la aceleración que genera la tierra sobre cualquier objeto que se encuentre cercano a ella. Su valor es $G=9.81 \text{ m/seg}^2$

La función de transferencia proporcionada por el fabricante es la siguiente:

$$ACC(g) = [-3g, +3g]$$

$$ACC(g) = \frac{ADC - C_{min}}{C_{max} - C_{min}} \cdot 2 - 1$$

$ACC(g)$ = Valor de aceleración en fuerza g

ADC = Valor muestreado del canal

C_{min} = Valor mínimo de calibración (habitualmente $C_{min} \approx 208$)

C_{max} = Valor máximo de calibración (habitualmente $C_{max} \approx 312$)

7.1.3 Fase de calibración

Como hemos visto en el punto anterior, es necesario establecer los valores C_{min} y C_{max} mediante una etapa de calibración previa a la adquisición efectiva de los datos.

Estos valores nos indican el valor muestrado que nos devolverá el sensor a través del canal cuando el eje de nuestro acelerómetro se encuentre a -1g y a 1g respectivamente.

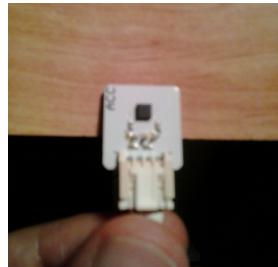
Recordemos que estos valores se encontrarán siempre dentro del intervalo [0, 1023].

Los valores recomendados por el fabricante son $C_{min} \approx 208$ y $C_{max} \approx 312$ y aunque hemos podido comprobar que son valores correctos que se aproximan fielmente a la horquilla de adquisición del módulo ACC con el que trabajamos, conviene desarrollar este punto con objeto de poder trabajar con otros componentes que no se encuentren adecuadamente calibrados.

El procedimiento de calibración debe aplicarse a cada eje del acelerómetro por separado, ya que estos valores pueden ser distintos en cada prueba. El proceso se describe a continuación:

- Para cada eje del acelerómetro:

1 . Obtener el valor de C_{min} . Para ello debemos colocar el eje de nuestro acelerómetro de forma que presente una aceleración de -1g, provocada por la propia atracción gravitatoria de la Tierra. Para ello podemos ayudarnos de una superficie horizontal, como por ejemplo una mesa. Debemos en todo caso mover el acelerómetro con cuidado y evitar que haya oscilaciones bruscas que sobrepasen el intervalo [-1g , -1g] ya que el rango de trabajo de nuestro acelerómetro es [-3g, 3g].



*Ilustración 19:
Calibración de el eje
X del acelerómetro.*

2. Una vez obtenido C_{min} debemos obtener el valor de C_{max} . Para ello debemos girar 180° sobre el eje del acelerómetro que estemos trabajando, siempre ayudándonos de una superficie perfectamente horizontal para que este valor sea fiable. Una vez más dicho movimiento debe llevarse a cabo con cuidado, ya que las oscilaciones pueden llevarnos a error.



*Ilustración 20:
Giramos el módulo
ACC*



*Ilustración 21:
Giro completo de
180°*

7.2 Estudio y caracterización de la señal

Como ya hicimos anteriormente con el módulo EMG, procederemos a capturar una población de muestras significativa con el objetivo de introducirlas en MATLAB y estudiar su forma en el dominio del tiempo y de la frecuencia.

Nuestro módulo ACC consta de tres ejes, pero su funcionamiento es idéntico, de modo que podemos limitar el estudio a un eje aislado, ya que el trabajo que hagamos es perfectamente extrapolable a los otros dos.

7.2.1 Muestras objeto de estudio

Se plantean las siguientes 4 muestras, pensadas para trabajar en distintos escenarios: Movimientos bruscos de 180° , movimientos lentos de 180° y movimientos de 180° con vibraciones y ruido. El eje utilizado para adquirir las muestras no es relevante.

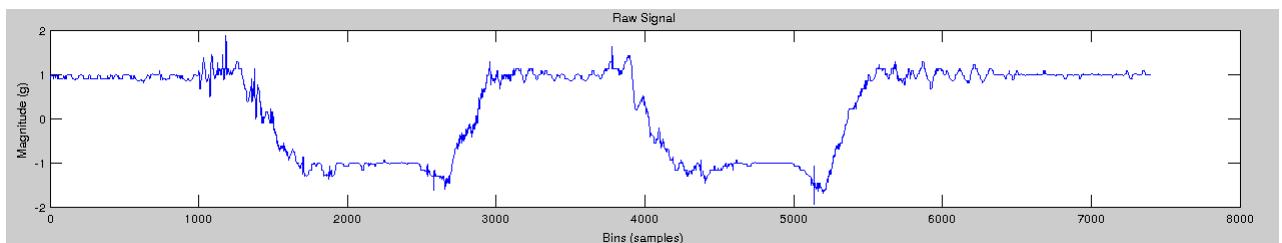


Gráfico 34: Toma 1: Movimientos bruscos de 180° durante 7 segundos.

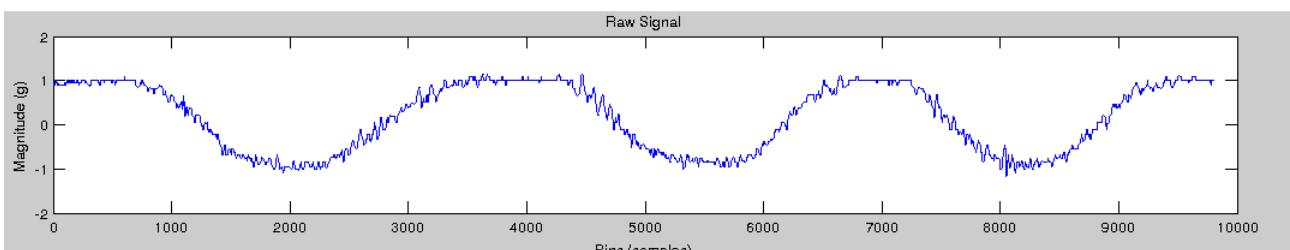


Gráfico 35: Toma 2: Giros suaves de 180° durante 10 segundos.

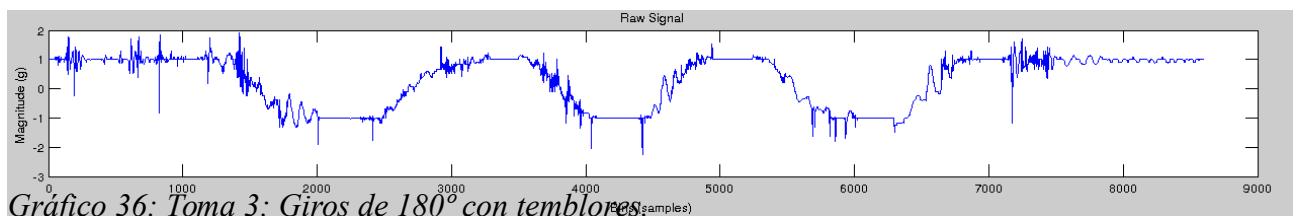


Gráfico 36: Toma 3: Giros de 180° con temblores.

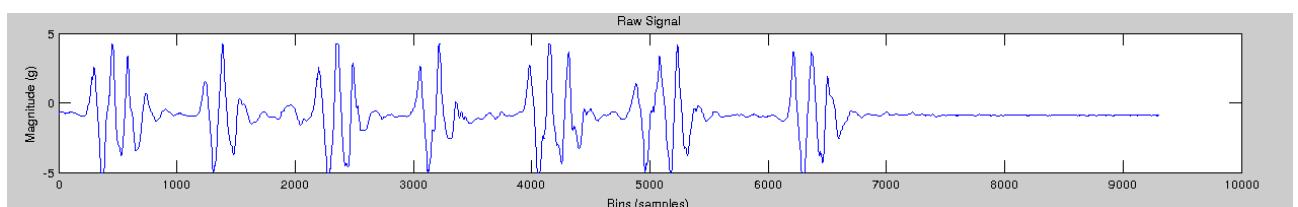


Gráfico 37: Toma 4: Vibraciones aleatorias a lo largo de 8 segundos.

Las muestras seleccionadas muestran características de nuestro interés por presentar patrones que se asemejan a los que resultarán derivados del uso que hagamos del acelerómetro en el futuro. También se incluye una muestra aleatoria que ha de servirnos de control en el análisis espectral de la señal.

7.2.2 Análisis de las muestras en MATLAB

7.2.2.1 Análisis espectral de las muestras

Siguiendo el mismo procedimiento descrito en el punto 6.2.2.1 obtenemos el perfil en el dominio de la frecuencia de las distintas tomas:

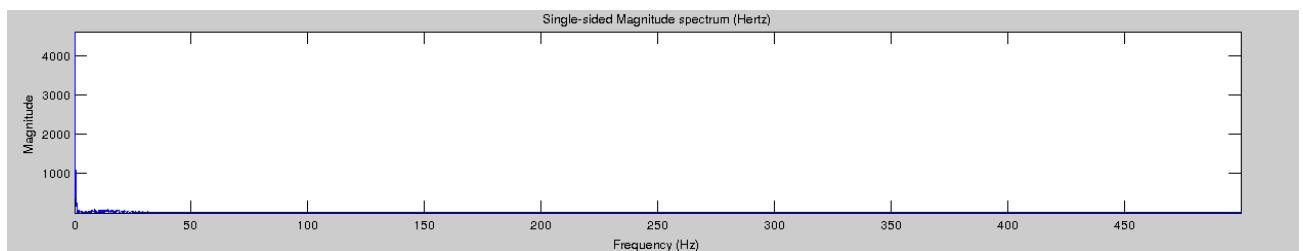


Gráfico 38: Transformada de Fourier de la Toma1, simple representación en hercios

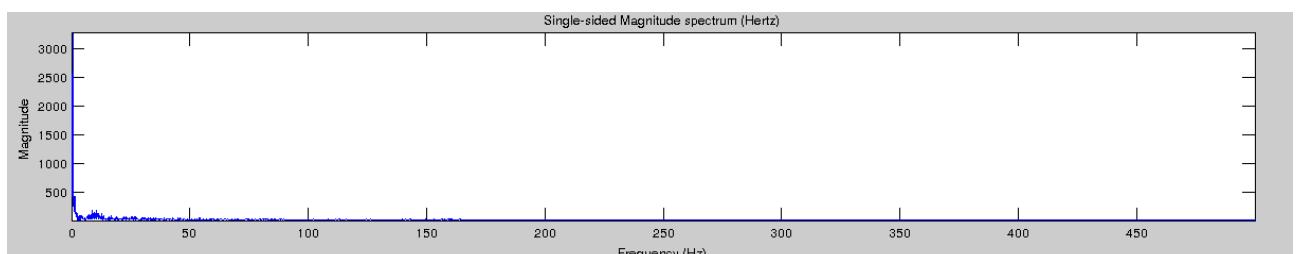


Gráfico 39: Transformada de Fourier de la Toma3, simple representación en hercios

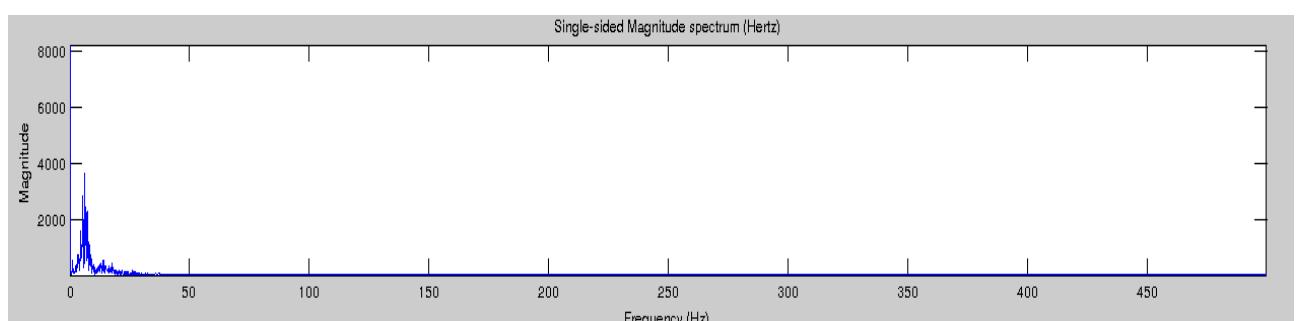


Gráfico 40: Transformada de Fourier de la Toma4, simple representación en hercios

Se puede observar que todas las componentes significativas ocupan un rango de frecuencias menor a 50hzs en todas las tomas analizadas. También puede observarse que por encima de una frecuencia de corte establecida en 50hzs, no existen componentes significativas en ninguna de las tomas, que puedan desecharse para construir una señal más limpia y mejorar la proporción Señal/Ruido. Por lo tanto el uso de un filtro de tipo Paso Bajo en este tipo de señales resultará redundante.

Para ilustrar este razonamiento tomemos como ejemplo la muestra nº1. Vamos a pasar dicha señal por un filtro Paso Bajo de tipo Butterworth de orden 2 con Frecuencia de Corte Superior en 50Hzs.

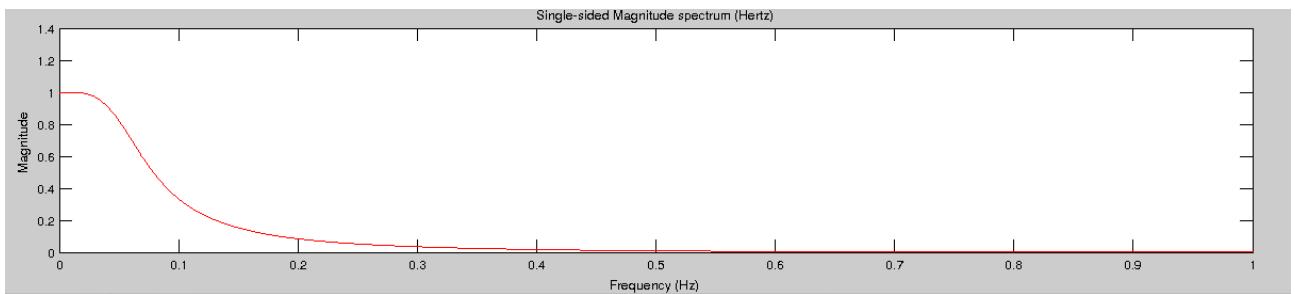


Gráfico 41: Filtro Butterworth Orden 2 FCS=50Hzs

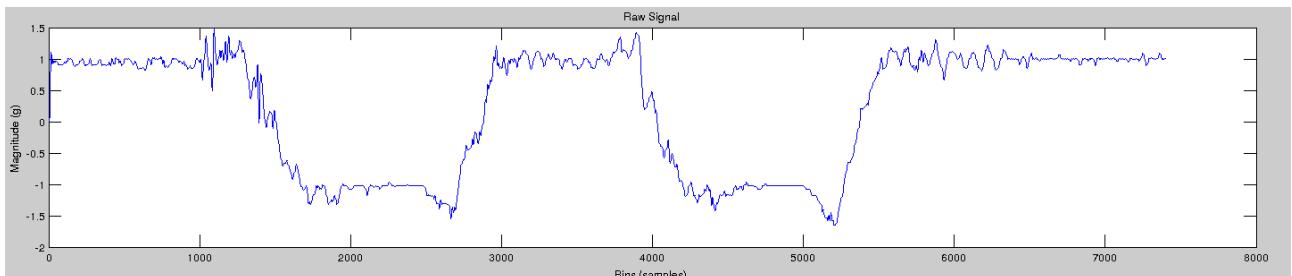


Gráfico 42: Señal antes de pasar por el filtro

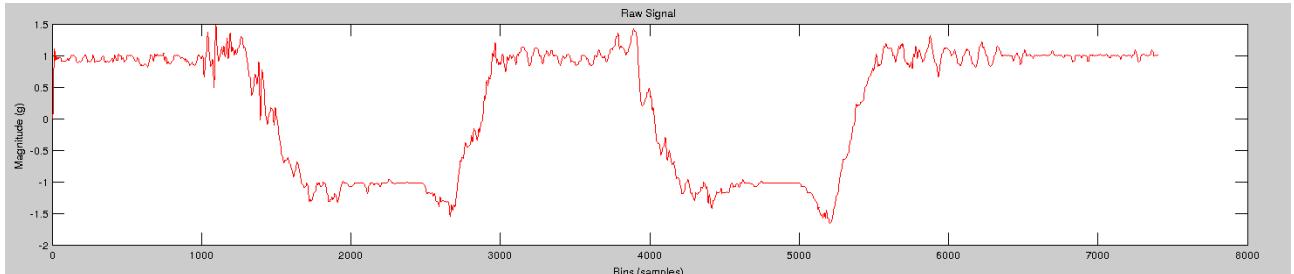


Gráfico 43: Señal tras pasar por el filtro

Como puede observarse, el filtro no realiza ninguna mejora sustancial sobre los datos muestreados, y en cambio es costoso computacionalmente hablando. Es innecesario disponer del mismo durante el procesado de la señal.

Resulta interesante suavizar la señal para reducir oscilaciones y obtener valores más estables, mediante un filtro de tipo media móvil, que estudiaremos en el siguiente punto.

7.2.3 Procesado de la señal

La señal ACC es en principio bastante sencilla de procesar. Bastará con detectar cuando la posición del eje sobrepase un nivel de aceleración preestablecido denominado disparador o threshold.

Como hemos expuesto anteriormente, la señal sin filtrar es perfectamente válida para su procesado, pero aun debemos responder a la necesidad de obtener una señal más suave y con menos oscilaciones que nos permita minimizar las falsas lecturas.

7.2.3.1 Filtro de media móvil

El filtro que utilizaremos para suavizar la señal sera el mismo filtro de media móvil que utilizamos para el módulo EMG, con una ligera modificación: evitaremos trabajar con los datos en su valor absoluto.

La señal ACC oscila en valores negativos y positivos, pero a diferencia de la señal EMG, en este caso no deseamos localizar picos, sino que pretendemos respetar la naturaleza de la señal tanto como sea posible, eliminando las oscilaciones que entorpezcan nuestro trabajo.

$$s[n] = \frac{1}{M} \sum_{i=0}^{M-1} x[n-i]$$

*Fórmula 2: Filtro
Media Móvil*

Como ya sabemos, M es una ventana deslizante cuyo valor recomendamos sea de 40 muestras.

$x[n]$ es la señal de entrada y $s[n]$ es nuestra señal de salida.

A continuación podemos comparar la señal original con la señal después de atravesar nuestro filtro de media móvil:

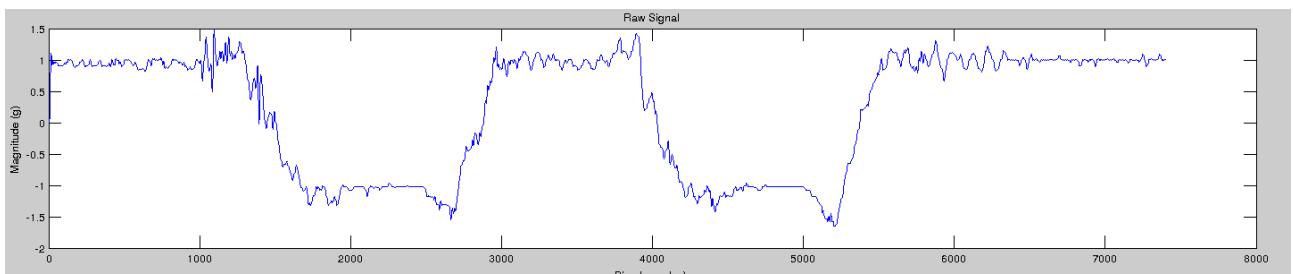


Gráfico 44: Señal antes de pasar por el filtro

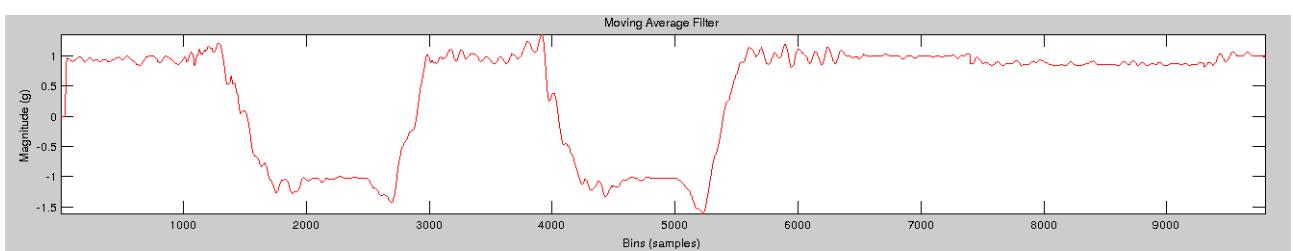


Gráfico 45: Toma1 tras atravesar un Filtro Media Móvil con ventana $M=40$ muestras

La mejora en la nitidez de la señal es claramente apreciable, aunque aún mejorable. Esto puede lograrse con sucesivas pasadas por nuestro filtro de media móvil.

Una segunda pasada por el filtro revelará una señal todavía más limpia y con muchas menos oscilaciones:

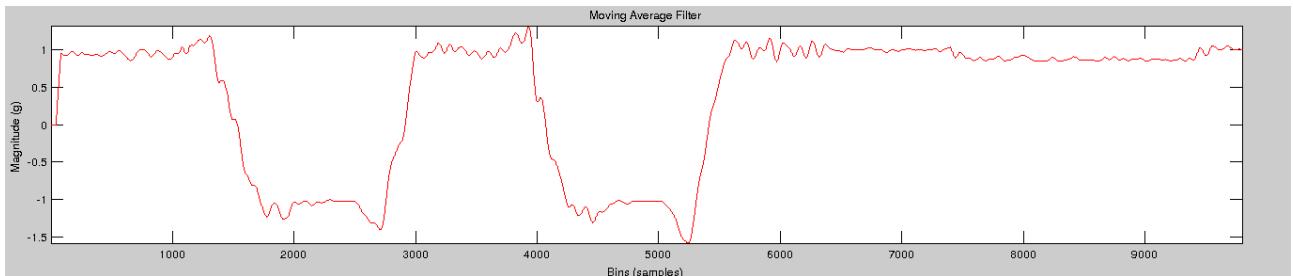


Gráfico 46: Toma1 tras atravesar dos veces un Filtro Media Móvil con ventana $M=40$ muestras

Resulta evidente que hacer pasar nuestra señal ACC por un Filtro de tipo Media Móvil de forma reiterada nos ofrece buenos resultados. Una tercera pasada muestra la señal aún más nítida si cabe:

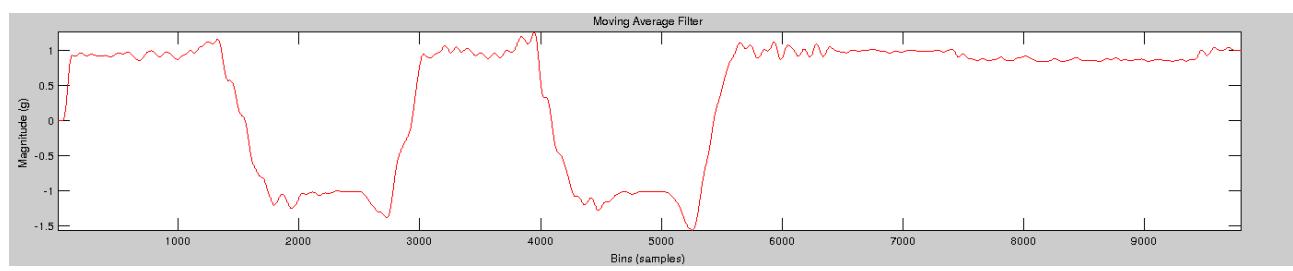


Gráfico 47: Toma 1 tras atravesar tres veces un Filtro Media Móvil con ventana $M = 40$ muestras

No obstante debemos ser cautos debido al tiempo de computación que requiere, pues cada filtrado supone un coste $n \cdot O(n^2)$ lo cual compromete sensiblemente este procesado. Debemos limitarnos a realizar como mucho dos barridos si deseamos que la futura aplicación funcione en tiempo real sin retardos poco convenientes.

7.2.3.2 Reducción de muestras (Downsampling)

Si bien este proceso no es estrictamente necesario para procesar la señal ACC, puede ser útil para lograr una señal de menor resolución que y por consiguiente reducir las tan indeseables oscilaciones que hemos mencionado anteriormente.

Como este proceso ha quedado suficientemente descrito en el punto 6.2.3.2 nos limitaremos a mostrar los efectos que una reducción de muestras a razón 1:40 provoca sobre nuestra señal:

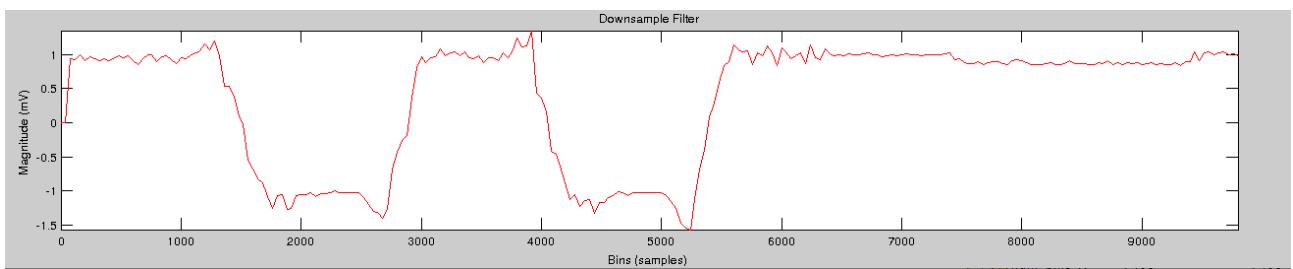


Gráfico 48: Reducción de muestras sobre la señal tras atravesar el Filtro Media Móvil descrito anteriormente

La reducción de muestras provoca una simplificación aún mayor que someter la señal a sucesivas pasadas sobre nuestro Filtro de Media Móvil, sin embargo su coste computacional es menor.

7.2.3.3 Detección de cambios

Una vez que tenemos una señal limpia con la cual trabajar, debemos establecer un método para establecer las fronteras de decisión que nos permitan identificar cuando se ha realizado un movimiento con el sensor ACC.

Como ya hicimos anteriormente, simplemente estableceremos un *disparador* o *threshold* que servirá como punto de activación cuando la señal lo atraviese encontrándose tanto por encima como por debajo de dicho valor.

El disparador será una “línea roja” que se activara cada vez que nuestra señal cruce dicho valor.

El valor de nuestro threshold debe ser establecido de forma manual, realizando el calibrado de forma empírica, pues no existe un método preestablecido para poder calcular dicho valor. Esto dependerá de la acción que deseemos detectar, que tendrá su propio ángulo de comienzo y fin para nuestro dispositivo y sus propios rangos de aceleración cambiante.

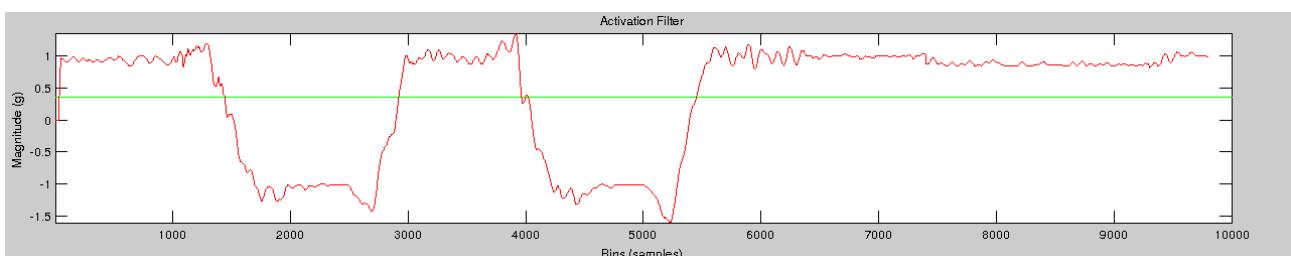


Gráfico 49: Disparador establecido a 0.3 durante la detección de cambios en el módulo ACC

En el ejemplo anterior, nuestro disparador está establecido a un valor de 0.3g. Cada vez que el eje de nuestro acelerómetro sobrepase este valor, se activará una señal. Podemos diferenciar entre un cambio ascendente o descendente, pero en todo caso este cambio se producirá cuando la señal sobrepase los 0.3g en un sentido u otro.

8. Electro Cardiograma: adquisición y procesado de señales ECG

Un módulo muy atractivo de bitalino es el módulo ECG, que nos proporciona una señal de electrocardiograma muestrado de manera digital con una resolución bastante eficaz y un precio muy asequible.

Este valor muestrado representa un potencial eléctrico proveniente de los electrodos auto adhesivos que colocaremos estratégicamente en lugares del cuerpo humano que nos permitan capturar la electricidad que genera el corazón al realizar sus contracciones y bombear la sangre. Cuando se genere potencial eléctrico en los electrodos, lógicamente significará que el corazón ha realizado una contracción.

Estos datos pueden constituir un elemento de estudio muy interesante, pues nos permite realizar análisis médicos a muy bajo coste y desarrollar aplicaciones de estudio y prevención de cardiopatías sin tener que recurrir a grandes kits profesionales.

Así mismo, es también una herramienta potencial con la que constituir un alfabeto de interacción hombre-maquina, pues una subida o bajada en el ritmo cardíaco constituye un indicador con el cual elaborar acciones o respuestas que pueden o no estar englobadas en el marco de la salud inteligente o ihealth.

8.1 Adquisición

El sistema de adquisición para nuestro módulo ECG es el mismo que el estudiado para el módulo EMG: electrodos auto adhesivos electro conductores.

Su estudio se ha desarrollado en el punto 6.1.1 y no requiere mayor análisis, si bien es necesario puntualizar que la colocación de los electrodos es muy distinta, así como su función de transferencia.

8.1.1 Colocación de los electrodos

Para realizar una buena adquisición de datos, debemos utilizar tres electrodos: un electrodo positivo, un electrodo negativo, y un electrodo de toma de tierra. Los dos primeros son típicamente blancos, y servirán para medir la diferencia de potencial que atraviese esos dos puntos de conexión en la superficie del cuerpo. El tercero servirá como diferencial, y su no colocación generará una señal más sucia, aunque igualmente válida.

8.1.1.1 Pecho

Esta ubicación es la que genera una señal más intensa y precisa, por estar muy cerca del corazón.

- El electrodo positivo debe colocarse en la parte superior derecha del pecho.
- El electrodo de tierra debe colocarse en la parte superior izquierda del pecho.
- El electrodo negativo debe colocarse en la parte inferior izquierda del pecho.

La siguiente figura ilustra con precisión la colocación de dichos electrodos:

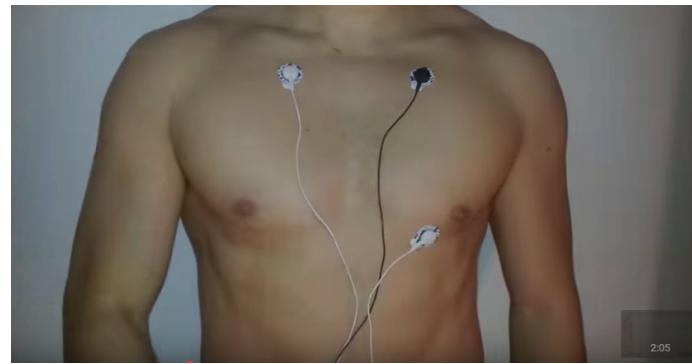


Ilustración 22: Colocación de los electrodos en el pecho

8.1.1.2 Manos

- Los electrodos negativos y de masa deben colocarse en la mano izquierda, cerca de la base del pulgar.
- El electrodo negativo debe colocarse en la mano derecha, cerca de la base del pulgar.



Ilustración 23: Colocación de los electrodos en las manos

8.1.1.3 Manos y pierna

- El electrodo negativo debe colocarse en la muñeca izquierda.
- El electrodo positivo debe colocarse en la muñeca derecha.



Ilustración 24: Electrodos colocados en las muñecas

- El electrodo de tierra debe colocarse en el tobillo de la pierna derecha.



Ilustración 25: Electrodo colocado en el tobillo derecho

8.1.2 Función de transferencia

Al igual que en los otros sensores estudiados hasta ahora, vamos a recibir de nuestro dispositivo bitalino una señal de naturaleza digital, a razón de 1000 muestras por segundo, con una resolución de 10 bits por muestra. En este caso el rango de trabajo oscila entre -1.5mV y +1.5mV, extremos que se corresponden con las muestras digitales de valor 0 y 1023 respectivamente.

El fabricante nos ofrece la siguiente función de transferencia:

$$ECG(mV) = [-1.5 \text{ mV}, +1.5 \text{ mV}]$$

$$ECG(V) = \frac{\left(\frac{ADC}{2^n} - \frac{1}{2}\right) \cdot VCC}{G_{ECG}}$$

$$ECG(mV) = ECG(V) \cdot 1000$$

$$VCC = 3.3 \text{ V} \text{ (voltaje operativo)}$$

$$G_{ECG} = 1100 \text{ (ganancia del sensor)}$$

$$ECG(V) = \text{Valor del ECG en voltios (V)}$$

$$ECG(mV) = \text{Valor del ECG en milivoltios (mV)}$$

$$ADC = \text{Valor sampleado del canal}$$

$$n = \text{Número de bits del canal, para nosotros, } n = 10$$

Nótese que la función de transferencia es la misma que la del sensor EMG, con la única diferencia de que la ganancia del sensor es ligeramente diferente. Tampoco requiere de ningún tipo de calibración previa, algo que si ocurría en el sensor ACC.

El valor de $n = 10$ requiere del canal que elijamos, ya que bitalino dispone también de canales que trabajan con una resolución de 6 bits, pero es de nuestro mayor interés trabajar con muestras tan precisas como nos sea posible.

8.2 Estudio y Caracterización de la señal

La ubicación que seleccionamos para la toma de muestras es el pecho, por generar lecturas de mayor calidad en cuanto a la intensidad de la señal y la proporción Señal/Ruido. Se trata de tres lecturas cortas de en torno a 6 u 8 segundos de duración.

8.2.1 Muestras de estudio

Se toman como muestras sujeto de análisis únicamente 3 tomas, ya que todas las señales de este tipo poseen siempre similares características, siempre y cuando se hayan adquirido en la misma zona. La única diferencia entre las tres tomas seleccionadas es su ritmo cardíaco y su duración.

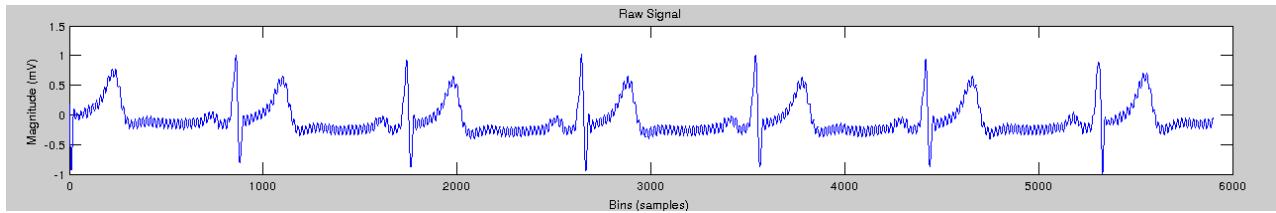


Gráfico 50: Toma 1: Latidos relajados en un intervalo de 6 segundos

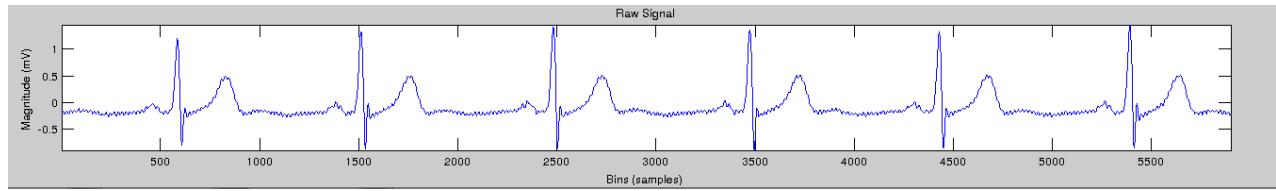


Gráfico 51: Toma 2: Latidos relajados en un intervalo de 6 segundos

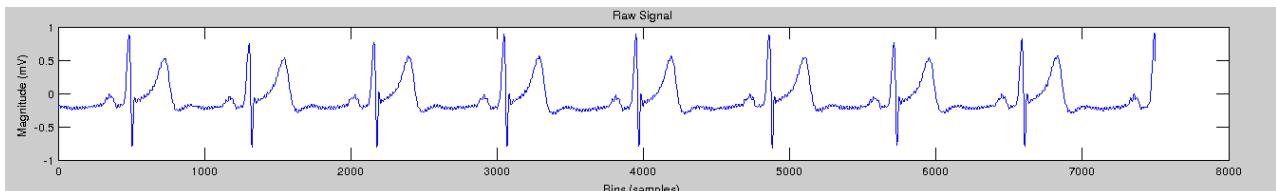


Gráfico 52: Toma 3: Latidos relajados en un intervalo de 8 segundos

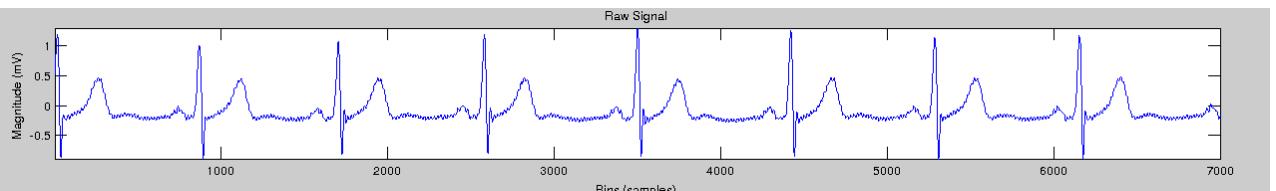


Gráfico 53: Toma 4: Latidos levemente acelerados en un intervalo de 7 segundos

Puede observarse claramente que la forma de las señales es idéntica en forma como cabría esperar.

8.2.2 Análisis de las muestras en MATLAB

8.2.2.1 Análisis espectral de las muestras

Siguiendo la misma metodología aplicada anteriormente, procedemos a estudiar las señales en el dominio de la frecuencia:

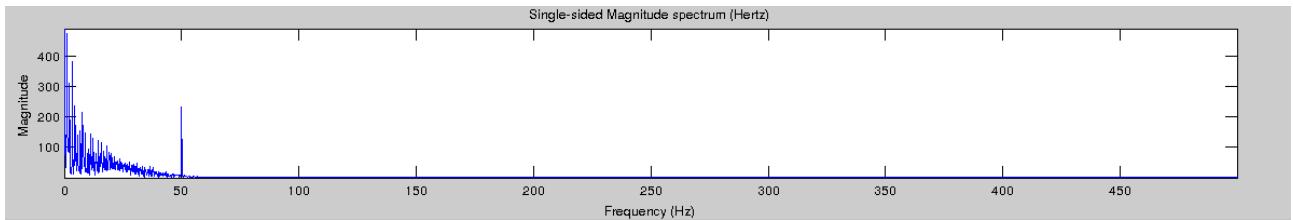


Gráfico 54: Toma 1: Transformada de Fourier. Simple representación en hercios

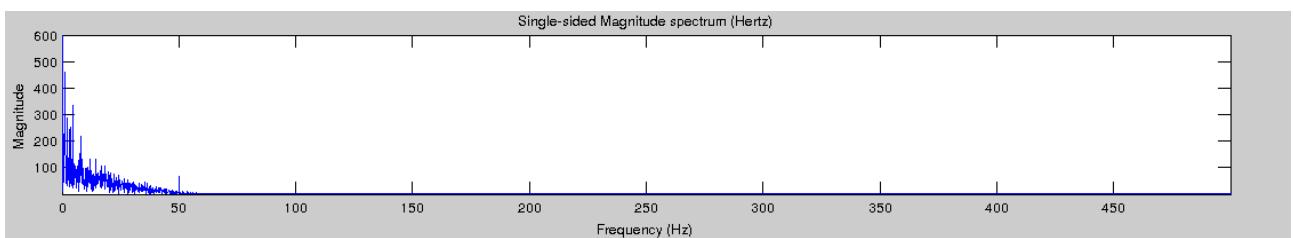


Gráfico 55: Toma 2: Transformada de Fourier. Simple representación en hercios

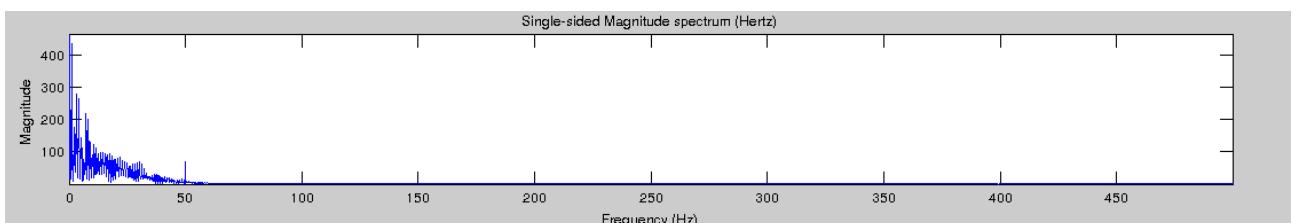


Gráfico 56: Toma 3: Transformada de Fourier. Simple representación en hercios

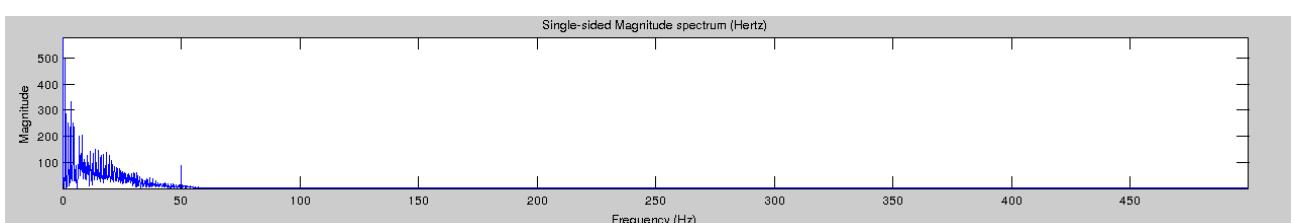


Gráfico 57: Toma 4: Transformada de Fourier. Simple representación en hercios

El análisis de Fourier de las muestras arroja un resultado muy concluyente: toda la información útil de las señales ECG se encuentra por debajo del umbral de los 50Hz.

Tal y como nos sucedió durante el análisis de las señales ACC, debemos plantearnos hasta qué punto resulta interesante aplicar un Filtro de Paso Bajo a esta señal, teniendo en cuenta que la magnitud de aquellas componentes que se encuentran por encima de 50Hz, es considerablemente baja.

Con el objeto de responder a esta pregunta aplicaremos un Filtro de Butterworth a una de las cuatro muestras escogida al azar.

Utilizaremos el mismo Filtro que estudiamos en el punto 6.2.2.2 y 7.2.2.1: un Filtro de Paso Bajo de tipo Butterworth de Grado 2, cuya función de transferencia a estas alturas nos es de sobra conocida:

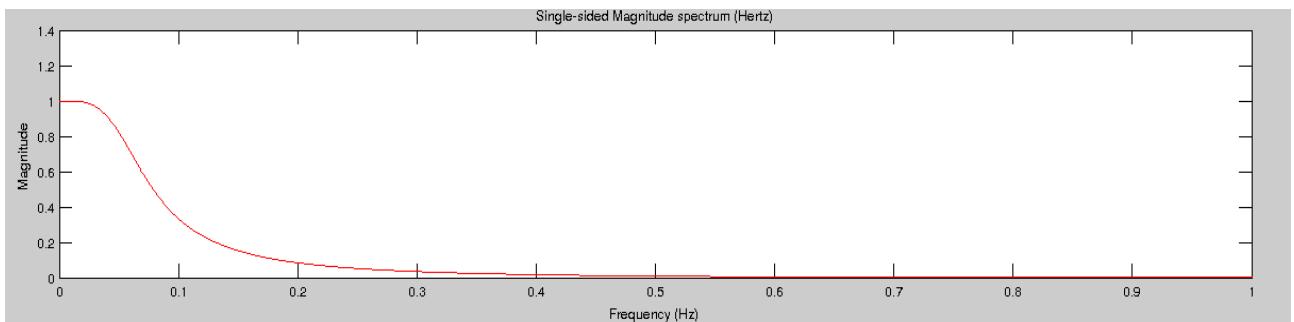


Gráfico 58: Función de transferencia. Filtro Butterworth Grado 2 FCS=50Hz

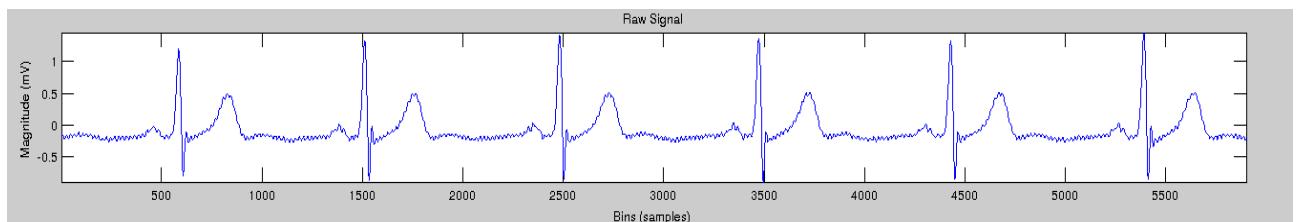


Gráfico 59: Toma 2 antes de pasar por el filtro

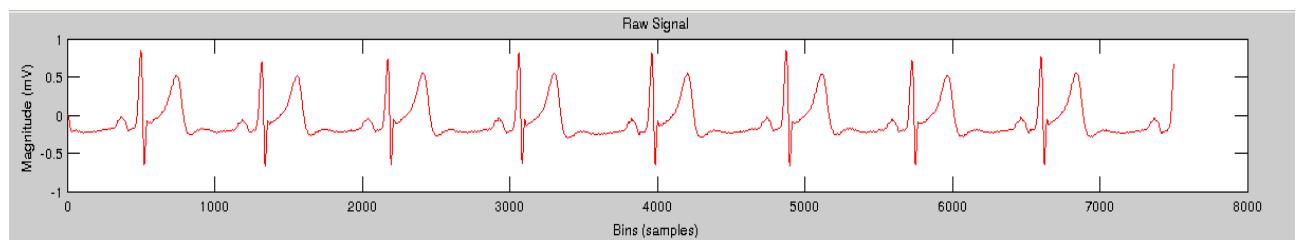


Gráfico 60: Toma 2 después de pasar por el Filtro

Se puede apreciar una mejora inmediata en la señal. Conserva su forma y toda su información, pero las líneas son mucho más definidas y suaves, resultando en una señal más limpia y más apta para su estudio.

En este caso, a diferencia del módulo ACC y al igual que nos ocurrió con el módulo EMG, resulta muy interesante filtrar la señal por medio de un Filtro de Paso Bajo, tanto más cuanto más sucia sea la señal y peor la proporción S/N de la misma.

Se deja abierta la posibilidad de estudiar señales con peor morfología y más ruido, pero se presupone que el filtro trabajará mejor si cabe en este tipo de señales.

8.2.3 Procesado de la señal

Nuestro objetivo durante el procesado de la señal ECG será lograr medir el ritmo cardíaco del sujeto. Para lograr tal efecto, debemos de contar el número de picos de voltaje generados en una sección del electrocardiograma y dividir este número entre el intervalo de observación.

Para contar el número de picos, inicialmente vamos a plantear la misma estrategia que aplicamos para el sensor EMG: aplicar un filtro de media móvil para suavizar aún mas la señal, que trabaje en valor absoluto para obtener una señal que oscile solo en términos positivos, y contar el número de veces que esta señal atraviesa un valor de umbral o threshold.

8.2.3.1 Filtro de Media Móvil

Como nuestro objetivo es calcular la envolvente de la señal y contar los picos generados, aplicaremos el mismo filtro visto en el punto 6.2.3.1.

A estas alturas este filtro nos es de sobra conocido y nos limitaremos a describir el efecto que tiene sobre la señal.

$$s[n] = \frac{1}{M} \sum_{i=0}^{M-1} |x[n-i]|$$

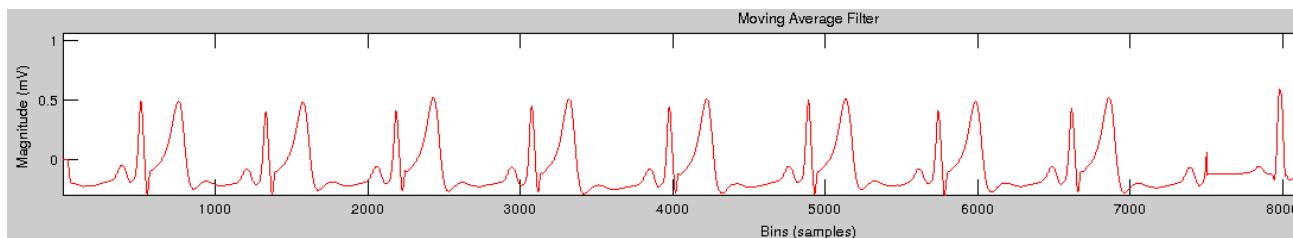


Gráfico 61: Toma 2 tras atravesar un Filtro de Media Móvil con ventana $M=40$ muestras

La señal presenta una simetría admirable y en este punto es muy apta para su medición, incluso sin necesidad de obtener una reducción de muestras o un cálculo de envolvente. Es una señal muy limpia y con una ambigüedad mínima que facilita muchísimo su estudio.

8.2.3.2 Reducción de muestras (downsampling)

Tras reducir el número de muestras de la señal anteriormente descrita, obtenemos el siguiente resultado:

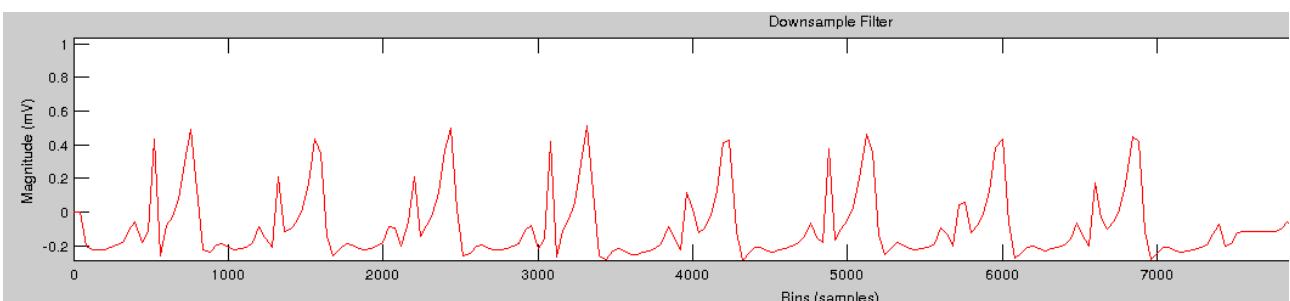


Gráfico 62: Toma 2 tras atravesar los filtros anteriores y reducir su número de muestras

Como anticipábamos, la reducción de muestras no obra ningún milagro y no nos aporta una señal más manejable en ningún sentido. Es pues un proceso opcional en este punto.

8.2.3.3 Detección de picos

Seguiremos el mismo proceso que con el módulo EMG: Estableceremos un *disparador* o *threshold* y activaremos un evento cada vez que la señal supere este valor.

El siguiente gráfico ilustra el procedimiento:

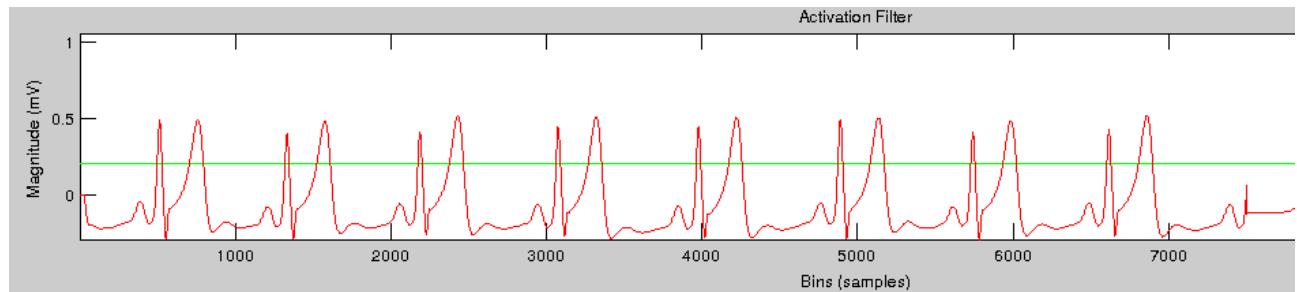


Gráfico 63: Toma 2 en su proceso de detección de impulsos

En este punto podemos observar un aspecto peculiar. La señal presenta dos picos por cada impulso cardíaco. Es muy importante tener esto en cuenta si deseamos obtener el promedio de pulsaciones por minuto al que late el corazón del sujeto sometido a las lecturas.

Esto es muy sencillo de solucionar, pues basta con dividir entre dos el número de picos leídos en el intervalo de observación. Finalmente para obtener las pulsaciones por minuto del individuo, basta con dividir la cifra resultante entre el intervalo de observación. Un simple cambio de escala convertirá esta relación en pulsaciones/minuto.

Téngase muy en cuenta que los valores de la señal dependen de múltiples factores: el estado de los electrodos así como la correcta colocación de los mismos, la preparación previa de la superficie cutánea del sujeto, el estado de nerviosismo y la frecuencia cardiaca del mismo.

Esto determina la intensidad de la señal y la separación de los impulsos, y en última instancia la amplitud de la misma. Resulta imposible predecir estos valores, más que de forma aproximada. Por esta razón el valor del disparador debe determinarse de forma manual, o mediante una calibración previa que permita medir la altura de los impulsos cardíacos obtenidos.

En futuros desarrollos se explorarán ambas posibilidades.

Parte III: Aportación a RoboComp: Diseño y Desarrollo de un Componente de Adquisición Biométrica.

9. RoboComp: Un Framework de Desarrollo para Robótica

RoboComp es un framework desarrollado por la Universidad de Extremadura. Es libre, diseñado para el desarrollo de software para robots. Emplea un paradigma de programación orientada a componentes y esta basado en Ice, un middleware ligero y fiable testado en proyectos críticos.

RoboComp proporciona un entorno de trabajo y un conjunto de herramientas destinadas a desarrollar componentes de software distribuidos, desarrollados en distintos lenguajes de programación y corriendo sobre diferentes plataformas.

9.1 Componentes de Software en RoboComp, una breve introducción

9.1.1 Ingeniería de software basada en componentes

La ingeniería de software basada en componentes (CBSE o CBD) es una rama de la ingeniería de software que se fundamenta en el desarrollo de sistemas complejos mediante el ensamblaje de elementos de software independientes, previamente elaborados denominados componentes.

Los objetivos de este paradigma son la reutilización de software, la distribución de la carga de trabajo entre los distintos elementos que componen el sistema, la independencia de las tareas respecto al lenguaje y a la plataforma en la cual se desarrollen, la abstracción y la encapsulación de la arquitectura de software respecto a cada una de sus partes.

Los estándares más conocidos que especifican este paradigma son:

- **CORBA** (Common Object Request Broker Architecture – Arquitectura Común de Intermediarios en Peticiones a Objetos). Establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.
- **DCOM** (Distributed Component Object Model – Modelo de Objetos de Componentes Distribuidos): Tecnología propietaria de Microsoft para desarrollar componentes distribuidos entre varios ordenadores que cooperan y se comunican entre sí.
- **.NET**: Framework de microsoft que enfatiza en la transparencia de las redes y de la plataforma de hardware en aras de un desarrollo de aplicaciones ágil.
- **JavaBeans**: Modelo de componentes basado en arquitectura cliente servidor. Solución multiplataforma, de fácil reutilización e integración universal, diseñado para trabajar con la maquina virtual de Java (JVM).

La idea fundamental de este paradigma consiste en la división de tareas en unidades de software independientes o componentes, que se comunican entre si mediante interfaces en el contexto de una red de computación distribuida.

De esta forma, un elemento de software o componente ofrece sus servicios a través de un punto de encuentro denominado interfaz, que es público y que sirve de nexo de unión con aquellos componentes que pretendan utilizar dichos servicios, sin necesidad de conocer cuales son las particularidades de implementación que rigen el computo de dichas tareas.

Para que entendamos mejor este paradigma, desarrollaremos un sencillo ejemplo:

Supongamos que disponemos del siguiente sistema basado en componentes:

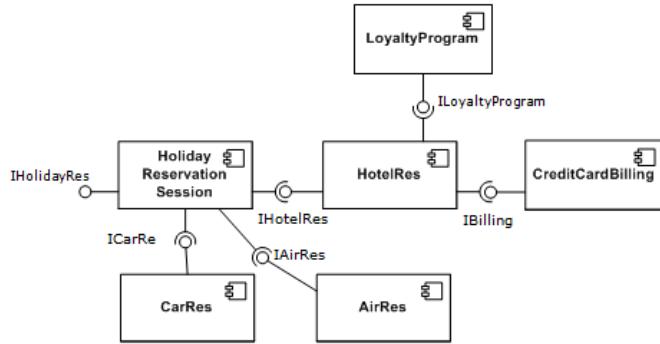


Ilustración 26: Ejemplo de modelo por componentes

El diagrama representa un sistema de reservas utilizado por una agencia de viajes, donde cada componente realiza una tarea independiente y bien diferenciada.

Las cajas rectangulares representan componentes, mientras que las líneas rectas que salen de las cajas y terminan en una bolita representan las interfaces con los servicios que cada componente publica.

Siguiendo este alfabeto gráfico resulta sencillo intuir cuando se ha establecido la comunicación entre dos componentes.

Puede observarse por ejemplo que el elemento *HotelRes* es un componente que gestiona la reserva del hotel, haciendo uso a su vez de otros dos componentes *LoyaltyProgram* y *CreditCardBilling*, que gestionan el programa de fidelización de clientes del hotel y el sistema de pago por tarjeta de crédito respectivamente.

CarRes es el componente que se encarga de reservar un coche.

AirRes es el componente encargado de reservar los billetes de avión.

HolidayReservationSession implementa la sesión de reserva y permite reservar cualquiera de los elementos descritos anteriormente.

Las características de un componente pueden ser diversas dependiendo del estándar implementado, pero en general es deseable que sean lo más independientes posible, bien documentados, robustos, que realicen un control de argumentos y testeo de errores previsor y eficiente y que sean reemplazables.

9.1.2 Modelo de componentes en RoboComp

RoboComp dispone de su propio modelo de componentes basado en el modelo ORCA y desarrollado desde 2005 para adaptarse a nuestras necesidades durante estos años.

Como hemos mencionado, utilizamos Ice como middleware de comunicaciones, aunque pueden utilizarse otras tecnologías como por ejemplo DDS.

Debido a la relación tan estrecha que tienen los componentes de RoboComp con Ice, todos ellos requieren de la inclusión de cierto código genérico y rutinario, que entorpecería el desarrollo de componentes de no estar automatizado, por lo tanto, la generación de componentes en RoboComp está automatizada y se desarrolla a partir de Lenguajes de Diseño Específico (DSLs) que comentaremos más adelante.

Cuando auto generamos un componente en RoboComp, la parte genérica del componente, aparecerá

implementada en la clase *GenericWorker*, que en ningún caso habrá de ser editada por el usuario, mientras que la parte específica que debemos de desarrollar se define en la clase *SpecificWorker*, que hereda de la anterior. Todo lo necesario para que el componente funcione correctamente se encuentra en la clase genérica.

La parte que el usuario desea implementar o desarrollar irá ubicada en la clase específica. De esta forma podemos evitar que el usuario pierda el tiempo implementando acciones que son comunes y predefinidas para todos los componentes que hacen uso de unas determinadas interfaces.

9.1.2.1 DSLs en RoboComp

El modelo de componentes de RoboComp se puede explicar a través de sus dos Lenguajes de Dominio Específico (DSLs) que se han creado para definir un componente a un nivel de abstracción muy alto. Estos DSLs son ISDL y CDSL, que se describen a continuación.

IDSL son las siglas de “Interface Definition Specific Language – Lenguaje de Definición de Interfaces Específicas”, es un subconjunto dentro de Slice, el lenguaje que utiliza Ice en la definición de sus interfaces. Con IDSL escribimos las estructuras de datos y las funciones que un componente puede implementar, requiere, suscribe o publica.

Un componente puede implementar varias interfaces, ofreciendo diferentes vistas de su funcionamiento interno. Además, la misma interface puede ser implementada por muchos componentes. Este es un ejemplo de una interface muy sencilla escrita en IDSL:

```
module RoboCompSpeech
{
    interface Speech
    {
        bool say(string text,bool overwrite);
        bool isBusy();
    };
};|
```

CDSL son las siglas de “Component Definition Specific Language – Lenguaje de Definición de Componentes Específicos” y permite al usuario especificar su nombre, interfaces accesibles, conexiones de comunicaciones, lenguaje a utilizar y otros módulos disponibles o librerías que se deseen incluir en los scripts de compilación.

```
import "/robocomp/interfaces/IDSLs/DifferentialRobot.idsl";
import "/robocomp/interfaces/IDSLs/Laser.idsl";
Component prueba
{
    Communications
    {
        requires DifferentialRobot, Laser;
    };
    language cpp;
    gui QWidget;
};|
```

Empleando estos dos DSLs, RoboComp puede generar el código fuente de un componente usando una herramienta diseñada para este propósito. El código funcional y completo del componente se crea listo para ser compilado y ejecutado.

Puesto que el generador hace diferenciación entre el código genérico y el código específico. Se puede volver a generar un componente para definir nuevos proxies manteniendo el código específico inalterado.

9.2 Modelo de comunicación: Ice

Desarrollado por ZeroC, Ice son las siglas de *Internet Communication Engine (Motor de Comunicaciones para Internet)*. Se trata de un framework RPC basado en CORBA, de código abierto. Proporciona un entorno de comunicaciones para el desarrollo de aplicaciones C++, C#, Java, JavaScript, Objective-C, PHP, Python y Ruby, y puede ejecutarse en numerosos sistemas operativos, incluyendo Linux, Windows, OS X y Android.

Ice implementa un protocolo de comunicaciones propio llamado Ice protocol, que puede correr sobre TCP, TLS, UDP y WebSocket.

Ice es por tanto un middleware orientado a objetos que permite desarrollar aplicaciones distribuidas con menos esfuerzo. Ice se hace cargo de las interacciones a bajo nivel con la red en la que deseamos trabajar y es independiente del lenguaje desarrollado.

Las características más interesantes de Ice son las siguientes:

- Soporta diferentes métodos de comunicación:
 1. Remote Method Invocation (RMI). Invocación Remota de Método.
 2. Paso de mensaje Publish/Subscribe .
 3. Asynchronous Method Invocation (AMI). Invocación Asíncrona de Método.
 4. Asynchronous Method Dispatch (AMD). Ejecución Asíncrona de Método.
- Especifica las interfaces mediante IDL, por lo tanto son fuertemente tipadas.
- Soporte de diferentes lenguajes de programación, que ya hemos mencionado anteriormente.
- Buen rendimiento y poca sobrecarga.
- Comunicaciones eficientes.

Ice proporciona APIs y librerías para soportar el paradigma cliente/servidos sobre plataformas distribuidas heterogéneas y utilizando múltiples lenguajes.

9.3 Soporte de lenguajes

RoboComp soporta los mismos lenguajes de programación que soporta Ice: C++, C#, Java, Python, Objective-C, Ruby y PHP. Si bien la mayor parte del código se encuentra escrito en C++ y Python, es posible escribir en muchos otros lenguajes manteniendo una perfecta integración en el framework gracias a Ice.

9.4 Herramientas

RoboComp extiende la funcionalidad de Ice añadiendo un conjunto de herramientas útiles para la programación de componentes robóticos distribuidos. Estas mejoran la usabilidad y hacen más sencillo el desarrollo de software complejo.

El conjunto de herramientas que se incluyen comprende componentes ya escritos y almacenados en el repositorio de robolab, que se pueden utilizar si antes los compilamos y lanzamos su ejecutable. Algunos ejemplos serían *differentialrobotComp*, *cameraComp*, *apriltagsComp*, *joystickComp* etc, que ofrecen servicios muy diversos que se pueden deducir de sus respectivos nombres. También existen herramientas precompiladas como por ejemplo la herramientas de generación de código

robocompdsl, que hemos mencionado anteriormente, o simuladores virtuales por control remoto como *rcinnerModelSimulator*.

Todo ello se integra en el árbol de repositorios del proyecto RoboComp y es fácilmente utilizable.

10. Diseño de una componente de adquisición biométricas

El objetivo ahora de nuestro estudio será diseñar un componente de software, capaz de interactuar con nuestra placa de desarrollo Bitalino, de adquirir las señales biométricas estudiadas durante la Fase II de nuestro trabajo y de emplearlas como medio de interacción hombre-máquina.

Como es lógico deseamos que este software sea modular, escalable e independiente, y deseamos también que este plenamente integrado en RoboComp, capaz de interactuar con cualquier otro componente y mantenible en el futuro.

Dado que en este punto tenemos una idea bastante clara de nuestras necesidades, una primera aproximación al problema consistirá en realizar la especificación de los requisitos que debe cumplir nuestro sistema.

Para ello emplearemos notación UML.

10.1 Definición de requisitos mediante Modelo de Casos de Uso

10.1.1 Actores

Los elementos externos al sistema que participan e interactúan con el son dos: el usuario de la aplicación y el sujeto que proporciona los valores biométricos. Solo tomaremos en consideración el primero de ellos, ya que el segundo no participa activamente del sistema y ejerce un rol pasivo que no afecta en nada a los requisitos y funcionamiento de la aplicación.

10.1.2 Diagrama de Casos de Uso

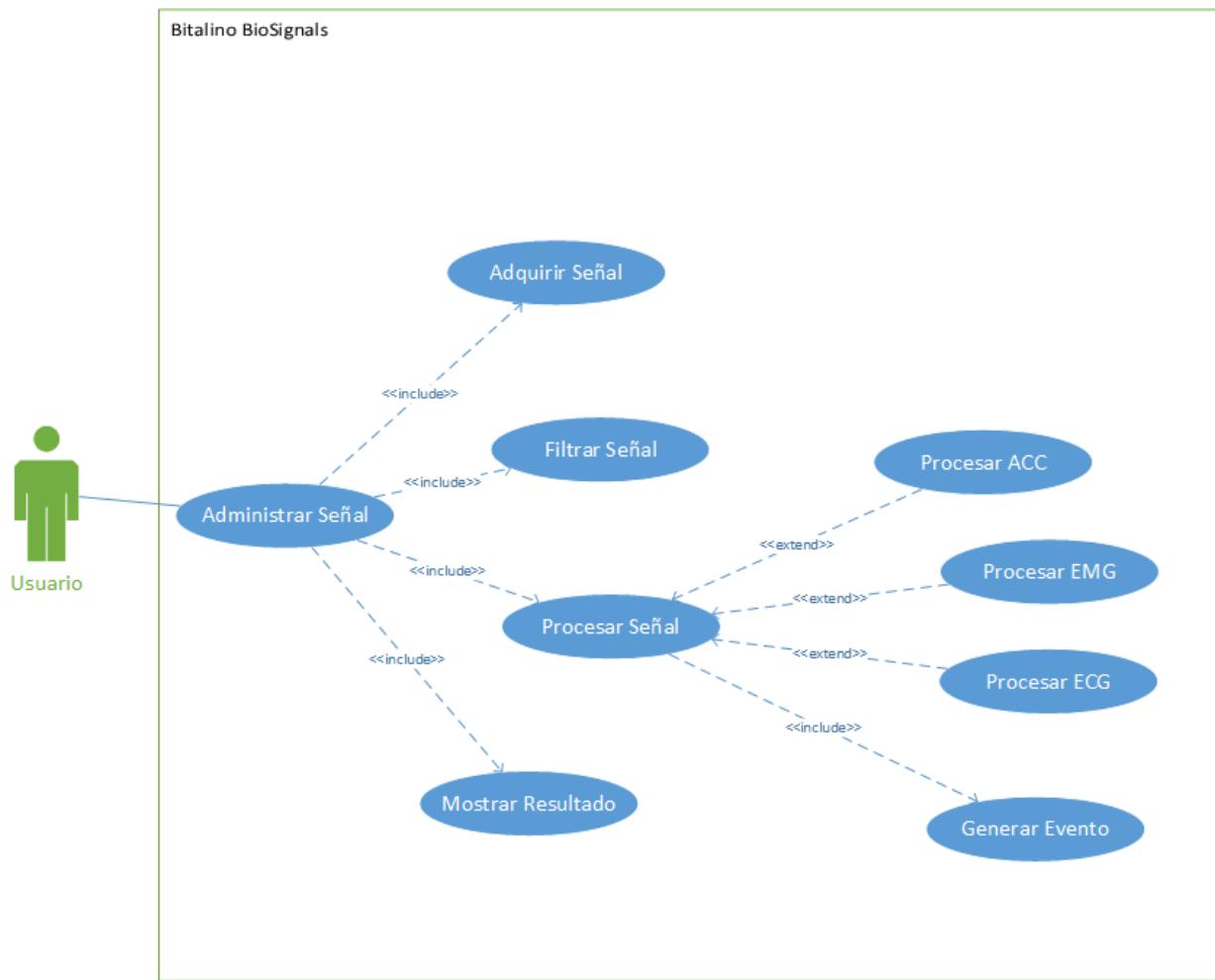


Ilustración 27: Diagrama de casos de uso de Bitalino BioSignals

10.1.3 Descripción de los Casos de Uso

Nombre del Caso de Uso:	Administrar Señal
Actores:	Usuario
Descripción:	<p>Atiende las necesidades del usuario a un nivel alto de abstracción. La tarea fundamental será la adquisición, análisis y procesado de una o varias señales, con su correspondiente generación de eventos y salida por pantalla.</p> <p>También existen otras tareas secundarias como conectar con el dispositivo o volcar los datos adquiridos en el disco duro.</p>
Disparador:	El usuario debe conectar con el dispositivo y mediante un previsible entorno gráfico de la aplicación, ordenar todo tipo de operaciones a nuestro caso de uso.

Pre-condiciones:	El dispositivo bitalino debe encontrarse encendido y correctamente emparejado con nuestro equipo mediante bluetooth.
Post-condiciones:	Dispositivo conectado y estado de ejecución variable dependiendo de las decisiones del usuario. Se mostrara por pantalla el estado de análisis y procesado de las muestras seleccionadas y se podrían generar eventos en tiempo real en función de las mismas.
Flujo Normal:	<ol style="list-style-type: none"> 1. Conectar con el dispositivo bitalino 2. Seleccionar las señales con las que trabajar: ACC (tres ejes), EMG y ECG. 3. Adquirir los datos del dispositivos 4. Procesar y analizar las muestras, ventanas de decisión y generación de eventos si los hubiere. 5 Representación por pantalla.
Flujo Alternativo:	
Excepciones:	<ul style="list-style-type: none"> - Tratar de conectar con un dispositivo mal emparejado. - Error de conexión con el dispositivo. - Error o desconexión durante la adquisición de datos.
Inclusiones:	Adquirir Señal, Filtrar Señal, Procesar Señal, Mostrar Resultado
Prioridad:	Esencial
Requisitos especiales:	Debe contemplarse la posibilidad de conectar con modelos de bitalino con direcciones físicas distintas.
Notas:	

Nombre del Caso de Uso:	Adquirir Señal
Actores:	No intervienen directamente.
Descripción:	Obtiene los datos RAW del dispositivo. Implementa el acceso al mismo mediante la API o funciones de bajo nivel suministradas por el fabricante.
Disparador:	Este caso de uso se dispara cuando el usuario desea hacer uso de esta funcionalidad a través del caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	<p>Debe haberse realizado con éxito la conexión con el dispositivo bitalino.</p> <p>La adquisición funcionará incluso aunque los módulos no estén conectados, aunque obtendrá datos inútiles.</p>
Post-condiciones:	El sistema se encontrará adquiriendo datos del dispositivo físico, con opción a posteriores procesados.
Flujo Normal:	<ol style="list-style-type: none"> 1. Comprobar la conexión con el dispositivo. 2. Abrir canal de comunicaciones haciendo uso de la API o librerías

	de bajo nivel proporcionadas por el fabricante. 3. Realizar lecturas a intervalos regulares. De acuerdo con las especificaciones técnicas, en paquetes de 100 muestras.
Flujo Alternativo:	No existe.
Excepciones:	Perdida de la conexión durante la adquisición de los datos.
Inclusiones:	No las hay.
Prioridad:	Esencial.
Requisitos especiales:	
Notas:	

Nombre del Caso de Uso:	Filtrar Señal
Actores:	No intervienen directamente.
Descripción:	Aplica los filtros estudiados durante la Fase II de nuestro estudio a las distintas señales que se deseen procesar. Se trata de una fase de pre-procesado de la señal.
Disparador:	Este caso de uso se dispara cuando el usuario desea hacer uso de esta funcionalidad a través del caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	Debe haberse realizado con éxito la conexión con el dispositivo bitalino. Debemos estar adquiriendo datos del dispositivo.
Post-condiciones:	Los datos de adquisición se encontrarán transformados tras su paso por el filtro. Serán más aptos para otros procesos.
Flujo Normal:	1. Obtener los datos adquiridos en etapas anteriores. 2. Aplicar filtro seleccionado. 3. Generar datos nuevos.
Flujo Alternativo:	No existe.
Excepciones:	No existen.
Inclusiones:	No emplea otros casos de uso.
Prioridad:	Dentro de este caso de uso, los filtros Paso Bajo tienen una prioridad Alta. Filtros media móvil tienen prioridad Esencial. Filtros de envolvente tienen prioridad Esencial. Cualquier otro filtro tiene prioridad Opcional.
Requisitos especiales:	Los datos se adquieren en ventanas discretas. Los filtros han de poder trabajar de acuerdo a este modelo de adquisición.
Notas:	Los filtros de media móvil deben de poder trabajar en valor absoluto, así como en valores positivos y negativos.

Nombre del Caso de Uso:	Procesar Señal
Actores:	No intervienen directamente.
Descripción:	Opera con los datos pre-procesados para realizar los cálculos de transformación que correspondan de acuerdo al tipo de señal adquirida, obtener sus fronteras de decisión y generar los eventos oportunos.
Disparador:	Este caso de uso se dispara cuando el usuario desea hacer uso de esta funcionalidad a través del caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	La señal objeto debe de estar en adquisición. No necesariamente debe estar filtrada.
Post-condiciones:	Se generan los datos de la señal a la salida de su fase de análisis y procesado.
Flujo Normal:	1. Se recogen los datos de la fase de filtrado . 2. Se procesan de acuerdo a su naturaleza (Ver casos de uso extendidos). 3. Se lanzan eventos y se generan resultados.
Flujo Alternativo:	1. Se recogen los datos de la fase de adquisición . 2. Se procesan de acuerdo a su naturaleza (Ver casos de uso extendidos). 3. Se lanzan eventos y se generan resultados.
Excepciones:	No existen.
Inclusiones:	No existen.
Prioridad:	Esencial.
Requisitos especiales:	Desconocidos en este punto.
Notas:	

Nombre del Caso de Uso:	Mostrar Resultado
Actores:	No intervienen directamente.
Descripción:	Opera con los datos procesados y los muestra por pantalla.
Disparador:	Este caso de uso se dispara de forma automática como consecuencia de las decisiones tomadas por el usuario en el caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	Disponer de datos procesados.
Post-condiciones:	Entorno gráfico actualizado con la información deseada.
Flujo Normal:	Representación de los datos y los procesos realizados a través del entorno gráfico de la aplicación.
Flujo Alternativo:	No existe.
Excepciones:	No especificadas en este punto.
Inclusiones:	No existen.

Prioridad:	Esencial.
Requisitos especiales:	<ul style="list-style-type: none"> - Emplear una librería gráfica adecuada a las necesidades de la aplicación que facilite las tareas de representación. - El caso de uso debe ser capaz de trabajar con paquetes de datos de tamaño predefinido, de manera que una representación de la señal en un contexto más amplio sea posible.

Nombre del Caso de Uso:	Procesar ACC
Actores:	No intervienen directamente.
Descripción:	Opera con los datos pre-procesados para realizar los cálculos de transformación necesarios para una señal de tipo ACC.
Disparador:	Este caso de uso se dispara de forma automática como consecuencia de las decisiones tomadas por el usuario en el caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	<p>La señal objeto debe de estar en adquisición.</p> <p>La señal objeto debe ser de tipo ACC.</p> <p>No necesariamente se debe trabajar con todos los ejes a la vez.</p> <p>No necesariamente debe estar filtrada.</p>
Post-condiciones:	<p>Se generan los datos de la señal a la salida de su fase de análisis y procesado.</p> <p>Se produce la detección de cambios en los ejes seleccionados de acuerdo a los estudios desarrollados en la Fase II de este trabajo.</p>
Flujo Normal:	<ol style="list-style-type: none"> 1. Se recogen los datos de la fase de filtrado. 2. Aplica suavizado a la señal por downsampling. 3. Se aplica frontera de decisión de acuerdo al eje con el que trabajemos. 4. Se lanzan eventos y se generan resultados.
Flujo Alternativo:	<ol style="list-style-type: none"> 1. Se recogen los datos de la fase de adquisición. 2. Aplica suavizado a la señal por downsampling. 3. Se aplica frontera de decisión de acuerdo al eje con el que trabajemos. 4. Se lanzan eventos y se generan resultados.
Excepciones:	No especificadas en este punto.
Inclusiones:	No existen.
Prioridad:	Esencial.
Requisitos especiales:	Debe ser capaz de trabajar con ventanas deslizantes, ya que los datos se obtienen en paquetes de 100 muestras y algunos algoritmos de proceso requieren conocer las muestras sucesivas y precedentes.

Nombre del Caso de Uso:	Procesar EMG
Actores:	No intervienen directamente.
Descripción:	Opera con los datos pre-procesados para realizar los cálculos de transformación necesarios para una señal de tipo EMG.
Disparador:	Este caso de uso se dispara de forma automática como consecuencia de las decisiones tomadas por el usuario en el caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	La señal objeto debe de estar en adquisición. La señal objeto debe ser de tipo EMG. No necesariamente debe estar filtrada aunque es lo recomendable.
Post-condiciones:	Se generan los datos de la señal a la salida de su fase de análisis y procesado. Se produce la activación de eventos cuando la señal sobrepasa la frontera de decisión establecida en la Fase II de nuestro estudio.
Flujo Normal:	<ol style="list-style-type: none"> 1. Se recogen los datos de la fase de filtrado. 2. Aplica suavizado a la señal por reducción de muestras (downsampling). 3. Aplicar calculo de envolvente a la señal anteriormente mencionada. 4. Aplicar fronteras de decisión. 5. Se lanzan eventos y se generan resultados.
Flujo Alternativo:	<ol style="list-style-type: none"> 1. Se recogen los datos de la fase de adquisición. 2. Aplica suavizado a la señal por reducción de muestras (downsampling). 3. Aplicar calculo de envolvente a la señal anteriormente mencionada. 4. Aplicar fronteras de decisión. 5. Se lanzan eventos y se generan resultados.
Excepciones:	El tamaño del buffer envolvente no debe ser superior al n.º de muestras por proceso una vez realizado el downsampling.
Inclusiones:	No existen.
Prioridad:	Esencial.
Requisitos especiales:	Debe ser capaz de trabajar con ventanas deslizantes, ya que los datos se obtienen en paquetes de 100 muestras y algunos algoritmos de proceso requieren conocer las muestras sucesivas y precedentes.

Nombre del Caso de Uso:	Procesar ECG
Actores:	No intervienen directamente.
Descripción:	Opera con los datos procesados y los muestra por pantalla.
Disparador:	Este caso de uso se dispara de forma automática como consecuencia de las decisiones tomadas por el usuario en el caso de uso <i>Administrar Señal</i> .
Pre-condiciones:	<p>La señal objeto debe de estar en adquisición.</p> <p>La señal objeto debe ser de tipo ECG.</p> <p>No necesariamente debe estar filtrada aunque es lo recomendable.</p>
Post-condiciones:	<p>Se generan los datos de la señal a la salida de su fase de análisis y procesado.</p> <p>Se produce el calculo de BPM susceptibles de disparar eventos y acciones en nuestro software.</p>
Flujo Normal:	<ol style="list-style-type: none"> 1. Se recogen los datos de la fase de filtrado. 2. Aplica suavizado a la señal por reducción de muestras (downsampling). 3. Aplicar detección y contabilizar picos. 4. Aplicar calculo de BPM a intervalos regulares. 5. Se lanzan eventos y se generan resultados.
Flujo Alternativo:	<ol style="list-style-type: none"> 1. Se recogen los datos de la fase de adquisición. 2. Aplica suavizado a la señal por reducción de muestras (downsampling). 3. Aplicar detección y contabilizar picos. 4. Aplicar calculo de BPM a intervalos regulares. 5. Se lanzan eventos y se generan resultados.
Excepciones:	Si la señal adquirida no tiene la calidad suficiente, el calculo de BPM se convierte en erróneo. Muy difícil de controlar.
Inclusiones:	No existen.
Prioridad:	Esencial.
Requisitos especiales:	Debe ser capaz de trabajar con ventanas deslizantes, ya que los datos se obtienen en paquetes de 100 muestras y algunos algoritmos de proceso requieren conocer las muestras sucesivas y precedentes.

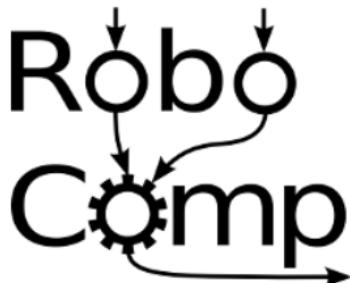
Se excluye del estudio la descripción del caso de uso *Generar Evento* pues esta parte del sistema esta abierta a la implementación orientada a un marco de trabajo específico.

10.2 Entorno de desarrollo: herramientas y librerías seleccionadas

De acuerdo a los requisitos previamente expuestos, en los siguientes puntos desarrollaremos las herramientas seleccionadas para la implementación y puesta en práctica de nuestro software.

10.2.1 RoboComp

Descrito con detalle en el punto 9 del presente documento.



RoboComp es un entorno de trabajo libre orientado a la robótica desarrollado en el Laboratorio de Robótica y Visión Artificial (ROBOLAB) de la Universidad de Extremadura.

Sus paradigmas son la eficiencia, la reusabilidad y la abstracción del hardware. RoboComp usa tecnología de componentes software para lograr sus objetivos.

Los componentes hechos con RoboComp pueden ejecutarse de manera distribuida en varios núcleos u ordenadores.

Proporciona un set de herramientas que hacen la creación y gestión de componentes una experiencia agradable.

10.2.2 Lenguaje de Programación

Con vistas a poder integrar nuestro software dentro del sistema RoboComp, el lenguaje de programación seleccionado es C++. Se trata de un Lenguaje Orientado a Objetos que proporciona un gran control al programador sobre los costes de computación y el consumo de memoria RAM, se integra con facilidad en RoboComp si se selecciona uno de los compiladores compatibles y dispone de una cantidad de librerías inmensa debido a su enorme popularidad.

10.2.3 Entorno de Desarrollo Integrado (IDE): QtCreator configurado con Qt4

10.2.3.1 Qt



Qt es una biblioteca multiplataforma empleada para el desarrollo de aplicaciones gráficas tanto en escritorio como en línea de comandos y consolas para servidores.

La principal fortaleza de Qt, además de proporcionar herramientas para el desarrollo de GUIs, es su portabilidad, ya que puede ser empleado para desarrollar aplicaciones migrables entre distintos sistemas, con un cambio mínimo en su código interno.

RoboComp integra soporte para componentes que utilicen las bibliotecas Qt4, y con reciente inclusión, también Qt5.

10.2.3.2 Qt Creator y Qt Designer

QtCreator es un Entorno Integrado de Desarrollo orientado a trabajar con las bibliotecas Qt4, o en su versión más reciente, Qt5. Dispone de un interfaz sencillo, integra herramientas de control de versiones como Git o Subversion, asistente de depuración y de depuración de memoria y herramientas asistidas de gestión de proyectos como Cmake o Autotools.

Es multiplataforma y con soporte activo.

Qt Designer es un editor gráfico de ficheros .ui (user interface). Estos ficheros sirven para definir entornos gráficos amigables de una manera muy sencilla utilizando notación XML.

10.2.4 Qwt: Widgets en Qt para Aplicaciones Técnicas

Se trata de un conjunto de componentes gráficos y librerías en C++ que son particularmente útiles para programas con un trasfondo técnico que requieran complejas representaciones gráficas.

Se integra perfectamente con Qt y con Qt Designer y proporciona una completa asistencia a la hora de representar funciones y señales de manera gráfica:

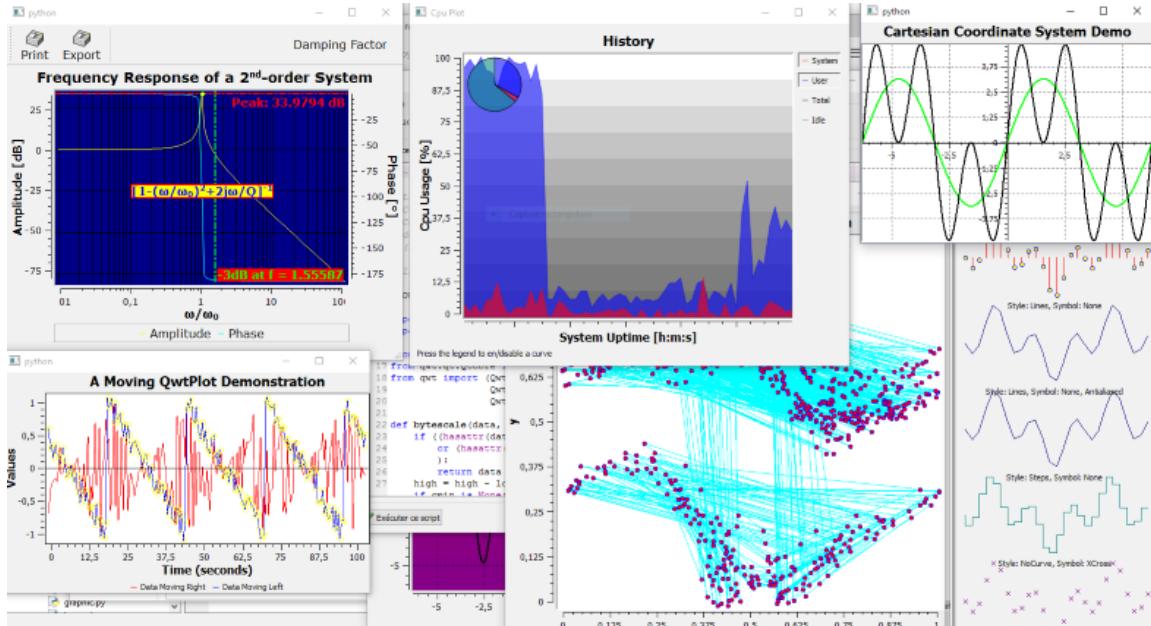


Ilustración 28: Ejemplo: aplicaciones desarrolladas usando qwt

La correcta instalación y configuración de estas librerías es imprescindible para la ejecución de nuestro proyecto, pues requiere de complejas representaciones gráficas de las señales que vamos a analizar y transformar.

10.2.5 API de bitalino



PLUX – Wireless Biosignals S.A. proporciona Interfaces de Programación de Aplicaciones (APIs) para un conjunto enorme de sistemas, incluyendo Android, Arduino, BONSAI, Objective-C para iOS o Mac OS, Matlab, C#, Python, Java y en el caso que nos ocupa, C++.

La API en C++ de bitalino esta constituida únicamente de una clase llamada BITalino implementada en sendos ficheros bitalino.h y bitalino.cpp, acompañada de un ligerísimo fichero de ejemplo implementado en el fichero test.cpp.

Esto es todo lo que necesitamos para comenzar a trabajar con el dispositivo en C++. La clase BITalino contiene un conjunto muy pequeño de funciones y estructuras diseñadas para adquirir datos e información de la placa de desarrollo bitalino.

10.3 Manual del programador

En este punto se describe la estructura interna de la aplicación, las decisiones de diseño adoptadas en función de los requisitos descritos anteriormente y los algoritmos empleados en la implementación de las soluciones a los análisis desarrollados durante la Fase II del trabajo.

El enfoque de desarrollo empleado ha sido el de desarrollar una aplicación *standalone* es decir, autónoma, para posteriormente integrarla como componente funcional en RoboComp.

10.3.1 Esquema de clases

Con el siguiente diagrama se pretende ofrecer una idea sencilla de como esta constituida la aplicación en términos de organización de software.

Los cuadros rojos representan el núcleo de la aplicación, las clases que conforman el funcionamiento de la misma.

Los cuadros azules representan los ficheros .ui, que definen las distintas ventanas que suponen el entorno gráfico de nuestra aplicación.

Los cuadros verdes son los ficheros .h generados a partir de sus respectivos ficheros .ui, que a su vez se encuentran estrechamente relacionados con algunas clases de nuestro proyecto, que heredan de objetos Qt, y por lo tanto sin dichos ficheros no pueden funcionar.

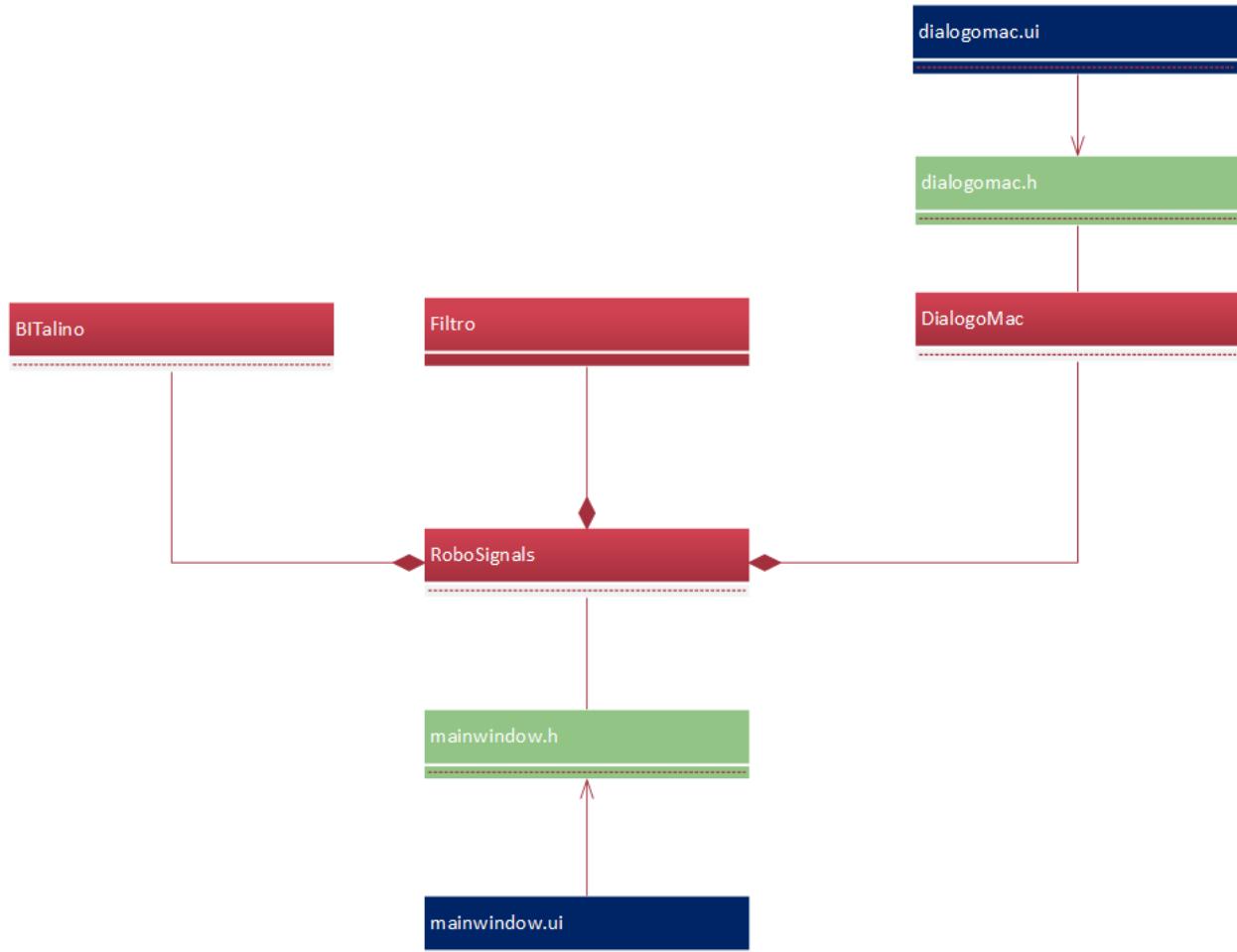


Ilustración 29: Diagrama de clases de Bitalino BioSignals

A continuación se detallan las clases que conforman el núcleo de la aplicación, dedicando tiempo a algunos algoritmos de mayor interés.

10.3.1.1 Clase BITalino

Como hemos mencionado anteriormente, se trata de la API proporcionada por el fabricante, dispone de las estructuras y funciones necesarias para conectar con el dispositivo bitalino y obtener datos de la placa de desarrollo.

Una documentación detallada puede verse en la pagina web del fabricante:

http://bitalino.com/docs/cpp-api/class_b_i_talino.html

Nos limitaremos a comentar que el constructor de la clase establece la conexión con el dispositivo (que de otro modo devolverá una excepción) y que dispone de métodos para arrancar la adquisición de datos, leer los datos mediante una estructura definida llamada Frame y detener la adquisición de datos.

Un aspecto muy importante que conviene subrayar y que determinará los desarrollos posteriores es el método de adquisición de que dispone esta API:

BITalino proporciona una lectura conjunta de todos los canales disponibles en el dispositivo, una lectura además, de 100 muestras. Nuestra aplicación trabaja en tiempo real con las lecturas de muestras que le proporciona el dispositivo. Debe comprenderse entonces, que el procesamiento de las señales viene dado en “trocitos” de 100 muestras que nos serán entregados a intervalos regulares, y que por lo tanto todos los procesos y análisis descritos en los puntos anteriores deben realizarse “de 100 en 100”, ayudándonos de ventanas deslizantes, buffers y herramientas de programación similares que complican enormemente el desarrollo.

10.3.1.2 Clase Filtro

Clase que encapsula todos los algoritmos de procesamiento descritos en la Fase II del presente trabajo. Incorpora filtros de Paso Bajo de muy diversos tipos y puntos de corte, filtros de media móvil con ventana deslizante de tamaño parametrizable, filtros de envolvente, reducción de muestras...

La mayoría de los métodos están redefinidos para trabajar con distintos tipos y estructuras de datos, pues se ha desarrollado en función de necesidades diversas a lo largo de nuestra implementación.

La clase esta diseñada específicamente para BITalino, puesto que incorpora buffers y ventanas deslizantes que nos ayudarán a procesar las señales que nos entrega de manera fragmentada la API del fabricante.

De este modo, a cada señal que deseemos procesar, le corresponde un único objeto Filtro, pues dicho objeto almacenará tantas muestras precedentes como le sean necesarias para llevar a cabo su trabajo.

Si deseamos procesar más de una señal al mismo tiempo, no hay mas que definir tantos objetos Filtro como señales deseemos procesar.

Para ilustrar el funcionamiento de esta clase, describiremos a continuación el algoritmo de media móvil.

10.3.1.2.1 Implementación del Filtro de Media Móvil

Recordemos brevemente la especificación empleada para este filtro:

$$s[n] = \frac{1}{M} \sum_{i=0}^{M-1} |x[n-i]|$$

La fórmula presentada nos dice que dada una señal de entrada x , de longitud n elementos, obtendremos una señal de salida s , de la misma longitud, cuyas muestras tendrán un valor que dependerá en cada punto de la media calculada sobre las M muestras precedentes de la señal x en ese punto.

Supongamos $M=40$. Cada elemento $s[i]$ tendrá el valor resultante de la media calculada en los 40 elementos precedentes hasta $x[i]$.

Nada más sencillo. Sin embargo debemos recordar que BITalino nos proporciona las muestras en paquetes de 100 elementos, lo que significa que a la hora de realizar el procesado en tiempo real de las muestras, no disponemos de la parte de la señal que ya habíamos recibido en el pasado, ni de la que está por venir.

Dicho de otro modo, a la hora de realizar el barrido por las 100 muestras, no podemos calcular el valor promedio sobre los 40 primeros elementos, pero sí sobre los 60 siguientes.

Este problema no tiene solución, al menos en la primera lectura que hagamos sobre el dispositivo, pero podemos evitar arrastrar este error sobre los siguientes paquetes que recibamos, declarando un buffer que almacene los 40 últimos elementos de el último paquete de muestras recibido. De este modo, cuando llegue un paquete nuevo con 100 muestras, las 40 primeras no tendrán que quedarse sin su promedio, ya que podremos calcularlo a partir de las muestras almacenadas en el buffer.

A este buffer lo vamos a llamar ventana, y vamos a definir métodos en la clase que permitan insertar y extraer elementos de dicha ventana, siguiendo una política FIFO.

Para agilizar aún más los cálculos, podemos hacer que se mantenga el valor del sumatorio actualizado en función de los últimos M elementos introducidos en el buffer, de forma que un nuevo elemento insertado en el buffer incremente el sumatorio, y un elemento eliminado del mismo lo decremente.

Si hemos comprendido todo esto, la implementación de nuestro algoritmo no entraña complicación alguna:

```
void Filtro::mediaMovil(const float src[], float dest[], int size, bool valorAbsoluto){
    for (int var = 0; var < size; ++var) {
        if (ventanaLlena){
            dest[var] = (sumatorio / tamVentanaMedia);
            this->insertarEnVentana(src[var],valorAbsoluto);
        }
        else{
            dest[var] = 0;
            this->insertarEnVentana(src[var],valorAbsoluto);
        }
    }
}
```

src = señal de entrada x

dest = señal de salida s

size = tamaño del vector de muestras src

valorAbsoluto = opción para activar la media en valor absoluto

Como puede observarse, obtener las muestras empaquetadas de 100 en 100 constituye un reto para la implementación de los algoritmos, y también para la representación de datos en pantalla.

Todos los algoritmos de la clase Filtro adolecen del mismo problema, y en algunos casos puede ser incluso más complejo de resolver, como vamos a ver a continuación.

10.3.1.2.2 Algoritmo de cálculo de envolvente

Esta implementación reviste de cierta complejidad, por ello es menester describirla cuidadosamente en el presente punto.

En primer lugar, debemos de definir nuestro algoritmo para detectar la envolvente de una señal para después centrarnos en su implementación.

En el punto 6.2.3.3 definíamos nuestra señal envolvente como el conjunto de puntos que constituyen máximos locales en un entorno de tamaño M.

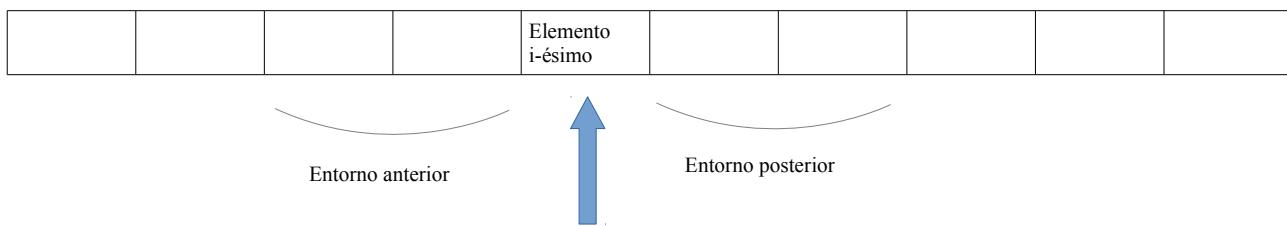
Esto quiere decir que si tenemos una señal formada por n muestras, la señal envolvente será el conjunto de aquellas muestras que sean mayores que un número M de muestras vecinas a izquierda y derecha de su posición absoluta, ignorando, o simplemente llevando a 0 a todas aquellas muestras que no supongan máximos locales.

Esta definición que en principio es muy sencilla, se complica cuando introducimos los siguientes elementos de diseño:

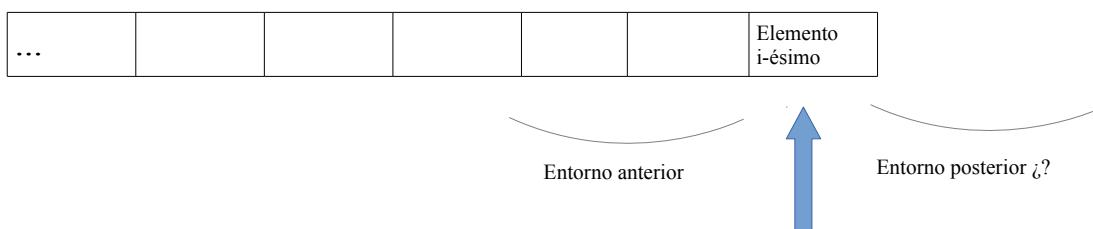
- Las muestras nos son dadas en paquetes de 100 elementos.
- La señal en este punto probablemente ha sufrido una reducción de muestras, y dichos paquetes pueden contener una población de muestras significativamente menor.

Lo aclararemos con un ejemplo práctico. Supongamos que recibimos un vector de 10 muestras que representa la señal sobre la cual deseamos calcular la envolvente.

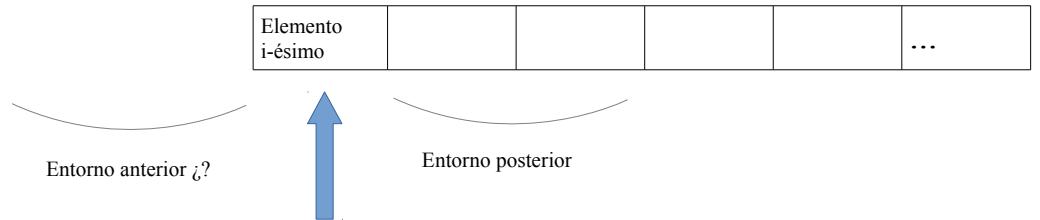
Como hemos descrito anteriormente, debemos de evaluar el entorno (por ejemplo M=2) de dicha muestra y discernir si se trata de un máximo local:



El elemento i-ésimo podrá evaluar sus cálculos sin problemas cuando se encuentre en mitad del vector, pero la situación se complica si se encuentra al final del vector, ya que no tendrá elementos sucesivos con los que comparar:



Igualmente, si el elemento que estamos evaluando se encuentra al principio del vector, nos encontraremos con un problema similar, ya que no tendrá elementos precedentes con los que comparar su valor:



Una vez más se impone la utilización de buffers. La parte del problema que corresponde a los elementos que se encuentran en la parte izquierda de nuestro vector es fácil de resolver, ya que siempre podemos disponer de las muestras de la señal que pasaron por nuestras manos en lecturas anteriores con un simple buffer de almacenamiento tal y como hicimos en la implementación del Filtro de Media Móvil.

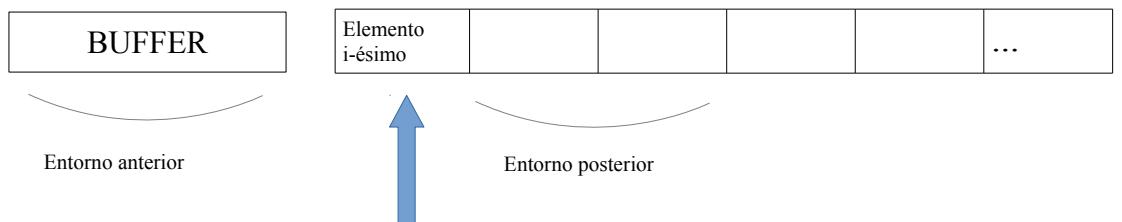
Sin embargo resolver el problema cuando estamos en la parte derecha del vector se antoja algo más complejo, pues resulta imposible disponer de unas muestras que todavía no hemos leído del dispositivo.

La solución encontrada pasa por retrasar el calculo tantas muestras como nos sean necesarias para procesar los elementos que están por venir.

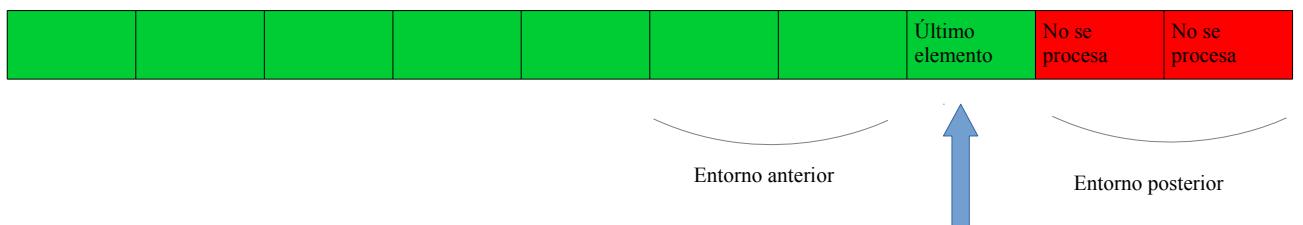
De esta forma, únicamente comenzaremos el proceso cuando el buffer de elementos precedentes se encuentre lleno, y únicamente recorreremos el vector hasta el elemento i-esimo ubicado en $i=n-M$, siendo n el tamaño del vector.

Volviendo al ejemplo anterior, realizaríamos el proceso para los elementos ubicados en las posiciones 0 a 7, y únicamente cuando el buffer de lecturas anteriores se encontrase lleno.

El primer elemento del vector podrá evaluar su entorno, puesto que el buffer almacenará elementos de lecturas anteriores:



El último elemento procesado del vector será aquel que tenga margen para evaluar el entorno posterior:



Sin embargo los elementos no procesados del vector deben tenerse en cuenta y procesarse la próxima vez que se ejecute el algoritmo, almacenándose en el buffer y tratándose como puntos nuevos a todos los efectos.

En definitiva, estamos retrasando la acción del algoritmo tantas muestras como nos son necesarias para poder evaluar los máximos locales.

Es por ello que debe buscarse un equilibrio entre la reducción de muestras que efectuamos al aplicar el Filtro de Media Móvil y el tamaño del entorno que aplicamos en el cálculo de la envolvente, ya que de configurar mal estos parámetros, nos encontraremos con una señal pobremente definida, y con un algoritmo de detección tosco y lento.

La implementación completa del algoritmo puede verse a continuación:

```

void Filtro::envolvente(QVector<QPointF> &src, QVector<QPointF> &dest, const float min){
    //std::cout<<"La fuente tiene: "<<src.length()<<" puntos" <<std::endl;
    bool anteriores;
    bool posteriores;
    int size=src.size();
    //Introducimos toda la fuente en la cola "datos" para que queden almacenados los elementos
    //no procesados al final del algoritmo
    for (int var = 0; var < size; ++var) {
        datos.push_back(src.at(var));
    }
    size = datos.size();
    QPointF dato;
    for (int var = 0; var < size-tamVentanaEnvolvente; ++var) {
        dato=datos.at(var);
        // El algoritmo se inicia una vez que la ventana envolvente esta llena
        if (this->ventanaEnvolventeLlena){
            if (dato.y()<=min){
                dest.push_back(dato);
            }
            else{
                anteriores = true;
                for (int ant = 0; ant < this->tamVentanaEnvolvente; ant++) {
                    if ((var-ant-1)>=0){
                        if (dato.y()<datos.at(var-ant-1).y()){
                            anteriores=false;
                        }
                    }
                    else{
                        if (dato.y()<this->ventanaEnvolvente.at(this->tamVentanaEnvolvente-ant-1).y()){
                            anteriores=false;
                        }
                    }
                }
                posteriores = true;
                for (int pos = 0; pos < this->tamVentanaEnvolvente; pos++) {
                    if (dato.y()<datos.at(var+pos+1).y()){
                        posteriores=false;
                    }
                }
                if (anteriores&&posteriores){
                    dest.push_back(dato);
                }
            }
        }
        this->insertarEnVentanaEnvolvente(dato);
    }
    for (int var = 0; var < size-tamVentanaEnvolvente; ++var) {
        datos.pop_front();
    }
}

```

Solo se evaluarán aquellas muestras que superen una altura mínima definida por el parámetro de entrada **min**. Esto es una forma de reducir el coste del algoritmo.

El atributo **datos** almacenará las muestras restantes (aquellas que cayeron en rojo) después de aplicar el algoritmo, de cara a próximas iteraciones.

ventanaEnvolvente es el buffer de muestras precedentes.

10.3.1.3 Clase RoboSignals

Esta clase se encarga de la gestión del entorno gráfico de la aplicación y de generar las acciones oportunas de acuerdo a las ordenes del usuario.

Debido a esto, y a que el entorno gráfico de la aplicación es muy completo, lleno de pestañas y de todo tipo de controles y parámetros, el código fuente de la clase tiene una longitud bastante extensa.

Entre sus atributos se encuentra una instancia de la clase BITalino, que nos permite realizar lecturas del dispositivo a intervalos regulares definidos por un temporizador, filtros suficientes para dar respuesta a los procesados de todas las posibles señales que el usuario puede adquirir al mismo tiempo y conexiones a todos los elementos del GUI por medio de SLOTS y métodos públicos.

Almacena un porcentaje muy elevado de código rutinario, y únicamente los métodos readFrames(), procesarEmg(...), procesarEcg(...) y procesarAcc(...) incorporan código relevante, pero en ningún caso algoritmos que merezcan una mención especial en este documento.

La clase RoboSignals es fácilmente integrable en RoboComp, ya que puede instanciarse desde un componente para desarrollar nuestro trabajo desde dentro del framework de robótica, como veremos más adelante y así comunicarnos con cualquier otro componente con más que añadir unas modificaciones mínimas a nuestro código.

11. Manual del usuario

11.1 Emparejamiento del dispositivos

Antes de empezar a trabajar con nuestro programa, es imprescindible que nuestra placa bitalino se encuentre correctamente conectada a nuestro equipo por medio de bluetooth.

Para ello debemos encender el bitalino con el pequeño interruptor que trae en la placa y establecer el emparejamiento bluetooth con el ordenador. Para ello debemos irnos a la ventana de configuración bluetooth de nuestro sistema operativo, y una vez en ella buscar los dispositivos dentro de nuestro rango de cobertura:

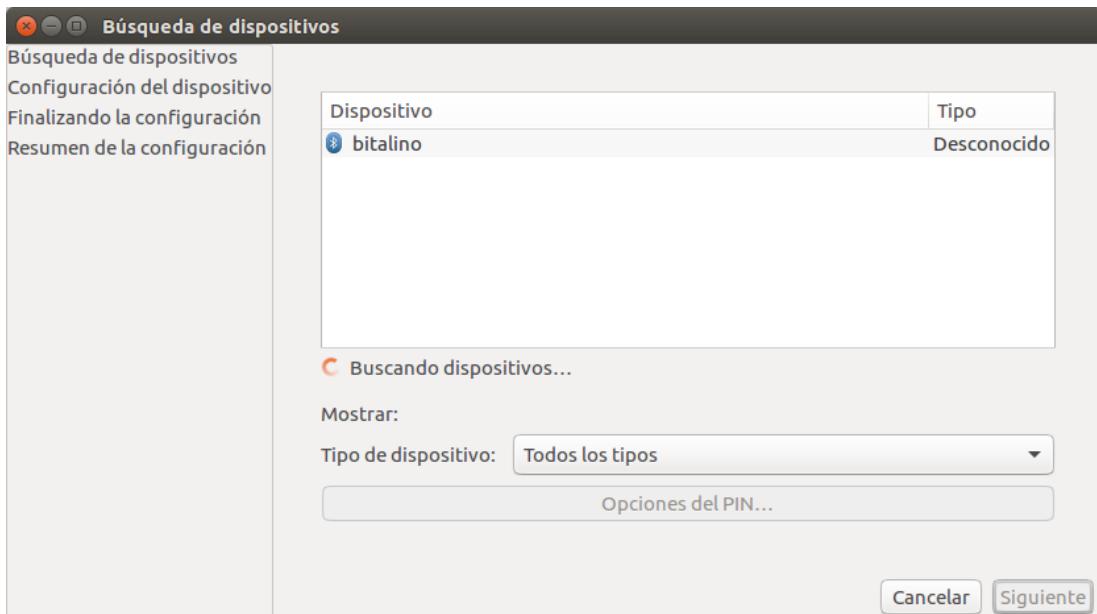


Ilustración 30: Búsqueda de dispositivos bluetooth en GNU/Linux

La placa bitalino viene configurada de serie con un código PIN de 4 dígitos: **1234**. Debemos proporcionárselo al gestor bluetooth de nuestro sistema operativo si queremos que conecte correctamente con el dispositivo:

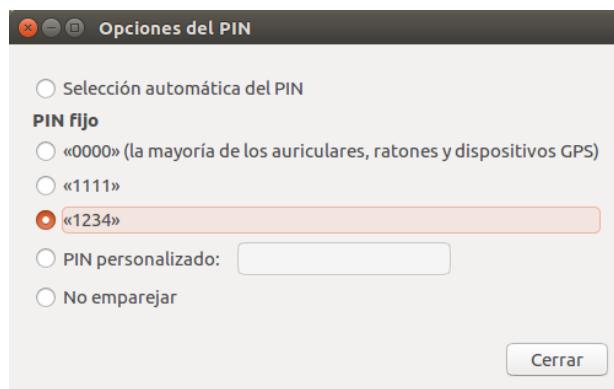


Ilustración 31: Selección de PIN en GNU/Linux

Una vez hecho esto, deberíamos de tener emparejado nuestro bitalino vía bluetooth:

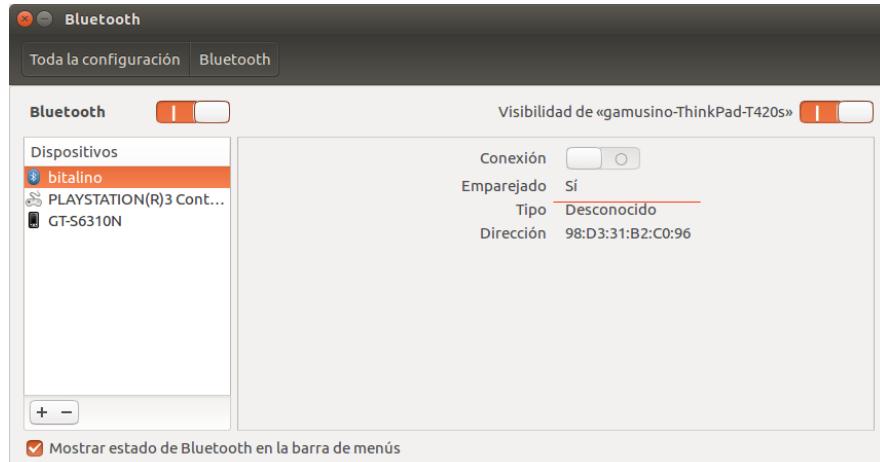


Ilustración 32: Dispositivo emparejado en GNU/Linux

Debemos poner especial atención a la dirección MAC que podemos observar en esta ventana, pues es un dato relevante que utilizaremos más adelante.

11.2 Pestaña de conexión

Como su propio nombre indica, esta parte de la aplicación se encarga de conectar con nuestro dispositivo bitalino para poder comenzar a adquirir datos.

Una vez arrancada nuestra aplicación Robolab Biosignals, nos encontraremos en la pestaña de conexión, y todas las opciones de nuestra aplicación se encontrarán desactivadas. Esto será así hasta que no comencemos a adquirir datos de los distintos sensores.

Para ello lo primero que debemos de hacer es introducir en el programa la dirección física del dispositivo bitalino con el que deseamos conectar, puesto que podría haber más de uno. Esto se puede hacer pulsando un botón que se encuentra en la parte superior izquierda de la ventana:

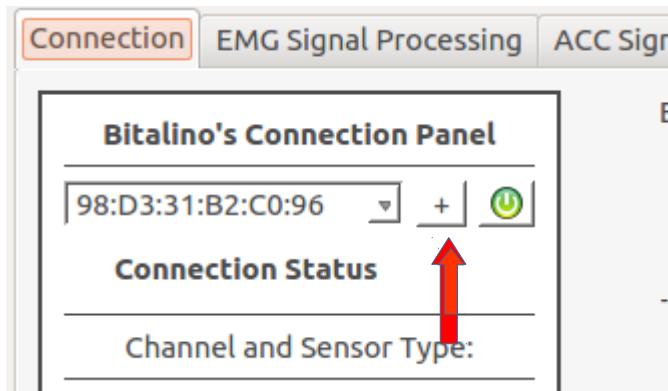


Ilustración 33: Añadir nueva dirección MAC

Esta acción desplegará una nueva ventana donde podremos introducir la dirección física de nuestro dispositivo:

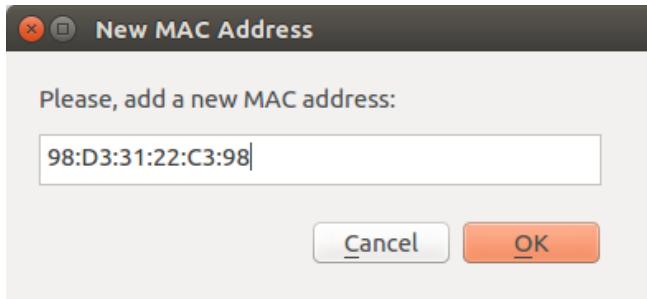


Ilustración 34: Introducción de dirección MAC



Si todos estos pasos los hemos seguido correctamente, ya podemos pulsar el botón de conexión, que presenta este ícono:

Si todo va correctamente la conexión se establecerá con éxito y veremos el siguiente mensaje:

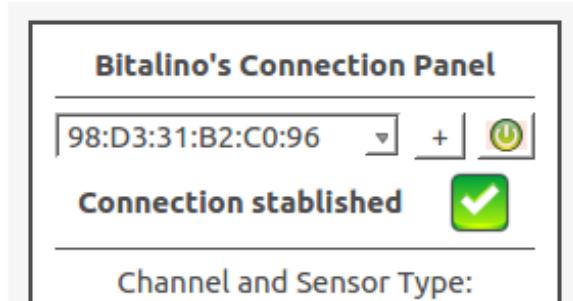


Ilustración 35: Conexión correcta

Si se ha producido algún error o la MAC es incorrecta, veremos el siguiente mensaje:

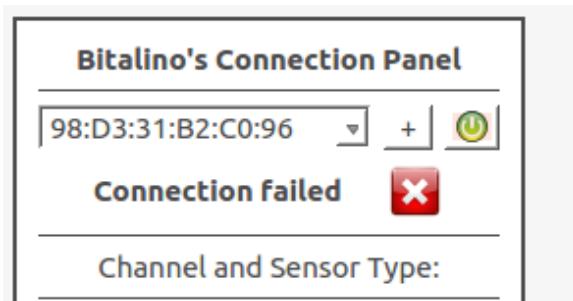


Ilustración 36: Conexión incorrecta

11.3 Adquisición de datos en crudo (RAW)

La pestaña de conexión de nuestro programa nos permite establecer conexiones a cualquiera de los siguientes tres módulos: ACC, ECG y EMG. El módulo ACC en realidad consume un canal por cada eje del acelerómetro, así que puede considerarse como tres módulos en uno.

Para comenzar a adquirir datos, solo debemos de especificar qué módulos deseamos usar marcando las casillas adecuadas:

Channel and Sensor Type:	
Sensor type	Analog Channel
<input type="checkbox"/> EMG:	1
<input checked="" type="checkbox"/> ACC X:	2
<input checked="" type="checkbox"/> ACC Y:	3
<input checked="" type="checkbox"/> ACC Z:	4
<input checked="" type="checkbox"/> ECG:	1

Ilustración 37: Casillas de selección de canal

También debemos de especificar en que canal de comunicaciones estamos conectando cada uno de los módulos. Estos no vienen numerados y los debemos conocer de acuerdo a la posición que ocupan en el módulo bitalino. Si empleamos una placa bitalino con los conectores integrados, la posición de los canales será la siguiente:

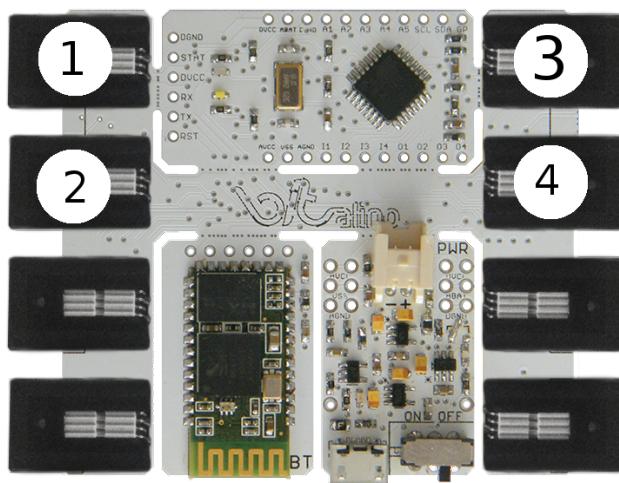


Ilustración 38: Numeración de puertos digitales en bitalino

La placa permite la adquisición simultánea de hasta 4 canales digitales con una resolución máxima de 10 bits procesadas por un conversor analógico-digital incluido en la placa. También se pueden adquirir hasta 6 señales analógicas, pero nuestra aplicación se ha desarrollado para trabajar con las primeras.

Una vez seleccionadas las señales y los canales de adquisición, debemos pulsar este botón:



y la aplicación comenzará a adquirir los datos en crudo:



Ilustración 39: Adquiriendo datos en crudo

Esta pestaña de conexión permite ver las señales que están entrando en nuestro sistema en tiempo real con una representación gráfica generada con qwt.

Si deseamos volcar los datos de alguno de los canales al disco duro para su posterior análisis o estudio lo podemos hacer si seleccionamos el módulo en la pestaña Data Capture y hacemos click en el botón que se iluminará si la selección ha sido correcta:

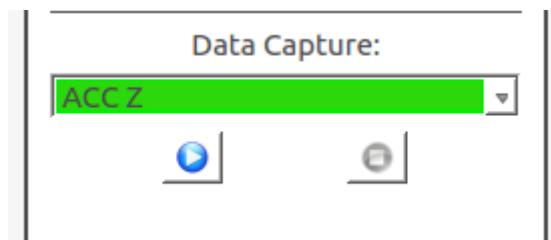


Ilustración 40: Pestaña para activar el volcado de muestras

Con los botones podemos controlar la información que se vuelca sobre el disco duro.

Dichos datos irán a parar a un fichero de texto llamado “datos.txt” ubicado en la misma carpeta

donde se encuentre el ejecutable de nuestro programa.

Cada vez que se decida volcar datos en el disco duro, se sobreescibirá dicho fichero, de modo que debemos hacer copias de seguridad si deseamos conservar de forma permanente alguna captura.

11.4 Pestaña EMG

El módulo de electro miografía es quizás el más interesante de nuestra aplicación, especialmente de cara a la robótica aplicada, pues es capaz de permitir el control remoto de dispositivos aplicando la mera activación muscular.

Para utilizarlo debemos de conectar el módulo EMG mediante la pestaña de conexión tal y como hemos descrito en el anterior punto. De ser así, podemos movernos a la pestaña EMG y veremos que algunas de las funciones ya no están desactivadas y que se visualiza la señal:

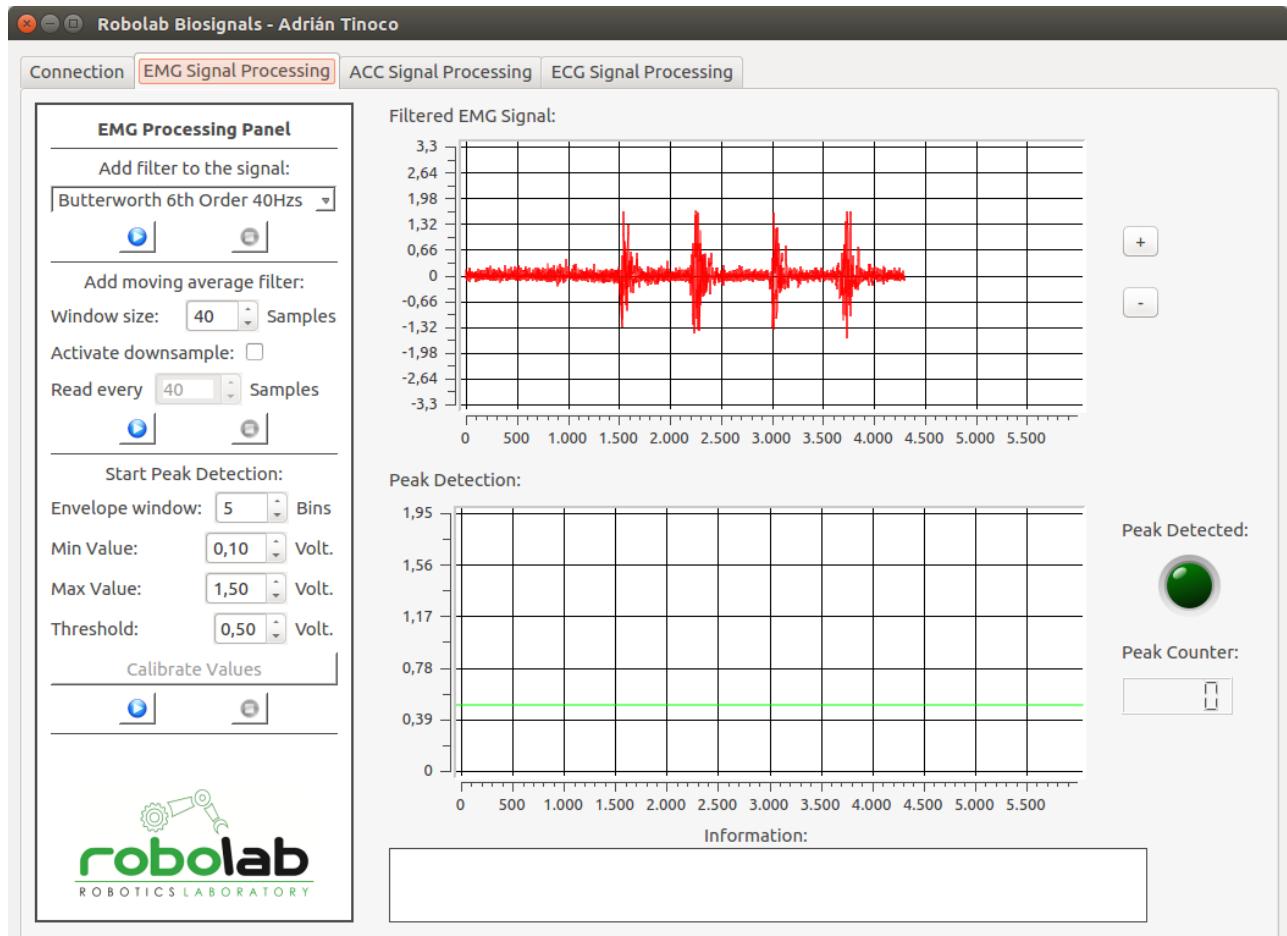


Ilustración 41: Pestaña EMG: Adquiriendo datos en RAW

Ya podemos comenzar a procesarla. Si la señal llegase excesivamente sucia, podemos ponerle un filtro de paso bajo:

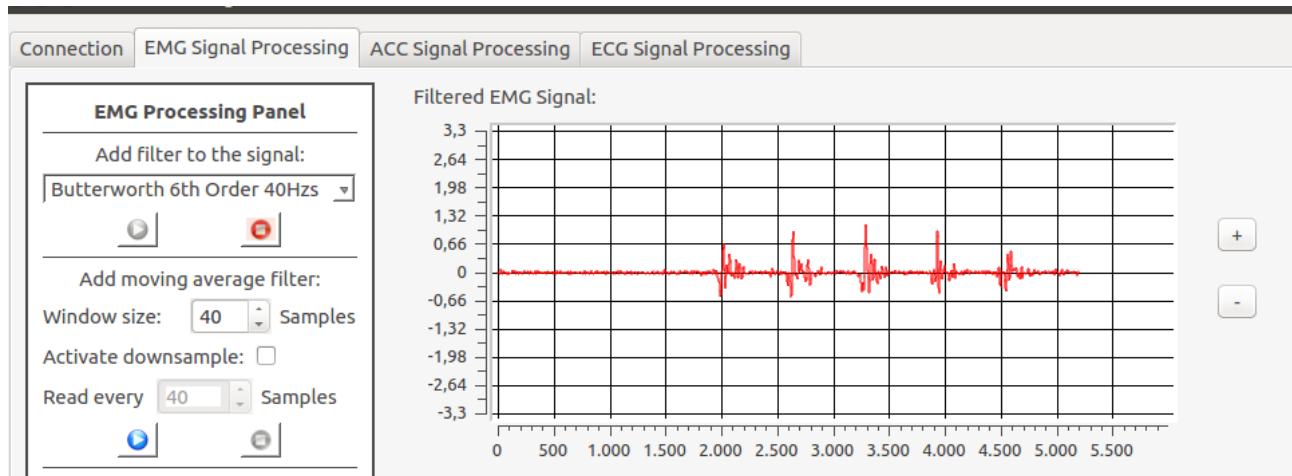


Ilustración 42: Pestaña EMG: Filtro Paso Bajo aplicado

Como puede observarse, esto da una señal más limpia, aunque perderá algo de su información original.

Una vez hecho esto, podemos aplicar un filtro de media móvil a la señal, y reducir el n.º de muestras de la misma:

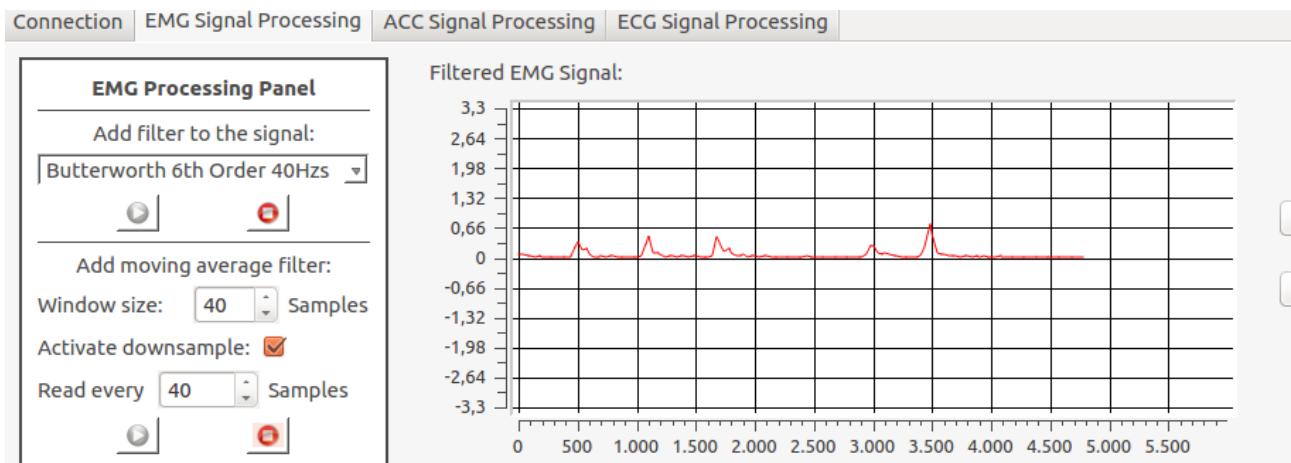


Ilustración 43: Pestaña EMG: Filtro Media Móvil aplicado

Puede verse claramente que la señal queda reducida a sus regiones de interés. Solo nos queda activar la detección de picos, algo que podemos hacer en la parte inferior del bloque de control:

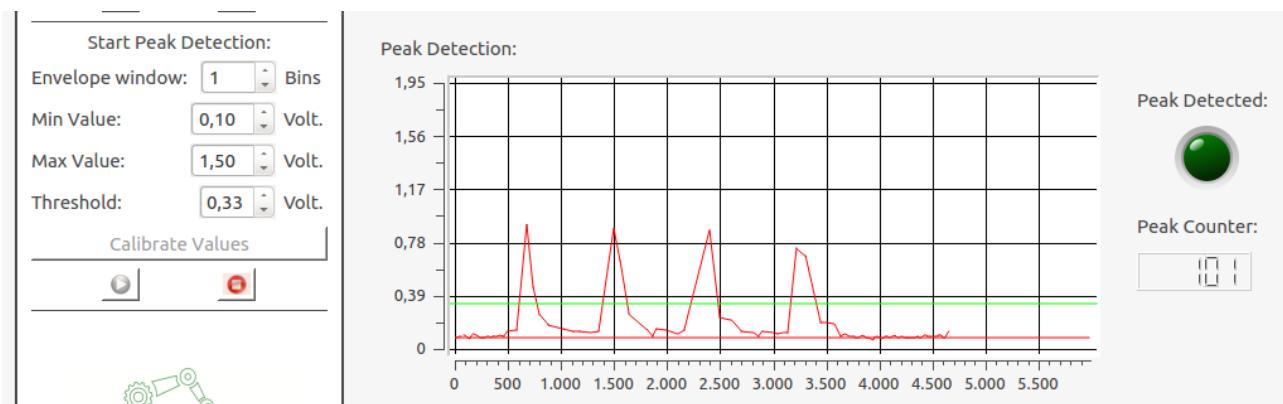
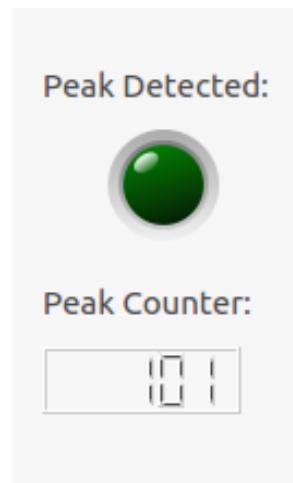


Ilustración 44: Pestaña EMG: Detección de picos

Modificando los valores del detector de envolvente y del disparador, podemos buscar un ajuste óptimo.

También disponemos de un botón de calibración que tratará de ajustar esos valores de forma automatizada.

A la derecha de la gráfica disponemos de un led y un indicador que nos informarán de el numero de picos que se han detectado durante el transcurso de la ejecución:



*Ilustración 45:
Pestaña EMG: LED
de detección y
contador de picos*

11.5 Pestaña ACC

De forma análoga al funcionamiento del módulo EMG, la pestaña ACC solo se activa cuando estamos adquiriendo señales de este tipo. Si esto es así, dicha pestaña tendrá el siguiente aspecto:

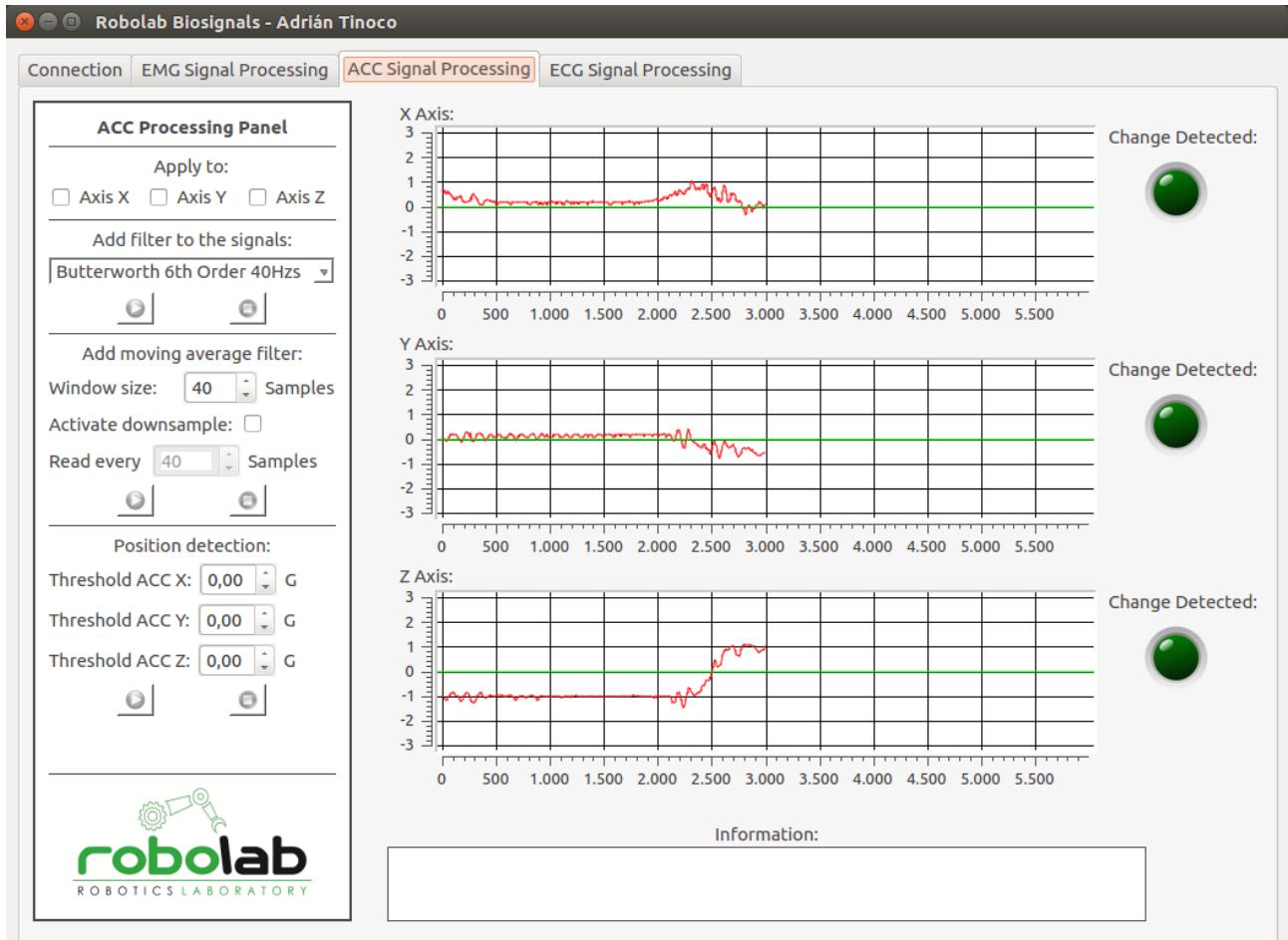


Ilustración 46: Pestaña ACC: Datos en RAW

En la imagen anterior podemos observar la adquisición de datos por medio de los tres ejes del acelerómetro tri-axial.

Los procesos de transformación que deseemos aplicar, solo se realizarán sobre los ejes que marquemos en las siguientes casillas:

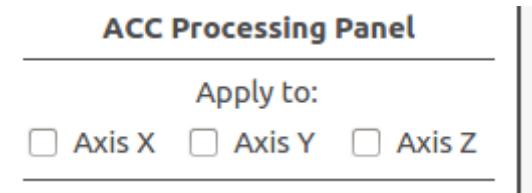


Ilustración 47: Pestaña ACC: Selección de ejes

Si tenemos marcada al menos una de ellas, entonces podremos aplicar los filtros y transformaciones que consideremos oportunos.

Disponemos de los mismos filtros de Paso Bajo que tenemos en la pestaña EMG, así como el filtro de media móvil, solo que en este caso, el filtro no trabaja en valor absoluto para respetar la naturaleza de la señal.

No se recomienda el uso de filtros Paso Bajo para este tipo de señales, como ya discutimos anteriormente.

Si decidimos aplicar alguno de estos procesos, podremos observar los cambios en las señales:

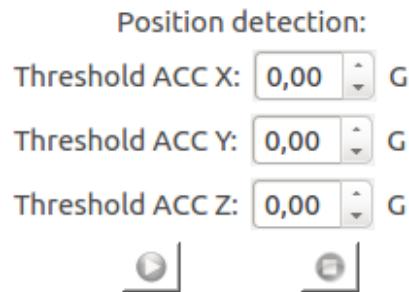


Ilustración 48: Pestaña ACC: Filtro Media Móvil

El último paso consistirá en activar la detección ACC. Que consiste simplemente en detectar cuando las señales correspondientes a los distintos ejes atraviesan una barrera preestablecida llamada threshold.

Cuando esto suceda, los leds que están a la derecha parpadearán, indicando al usuario que se ha generado un evento.

Los valores de los disparadores se pueden regular mediante los controles habilitados en el cajetín de la izquierda:



*Ilustración 49: Pestaña ACC:
Selección de disparadores*

11.6 Pestaña ECG

La pestaña del electro-cardiograma funciona de manera análoga a las dos anteriores.

Una vez que hayamos adquirido la señal en la pestaña de conexión, se activarán las opciones para realizar el filtrado y procesado de la misma en la pestaña ECG.

En este punto podemos aplicar los mismos filtros mencionados anteriormente, o si disponemos de una imagen lo suficientemente limpia, pasar directamente al procesado de la señal.

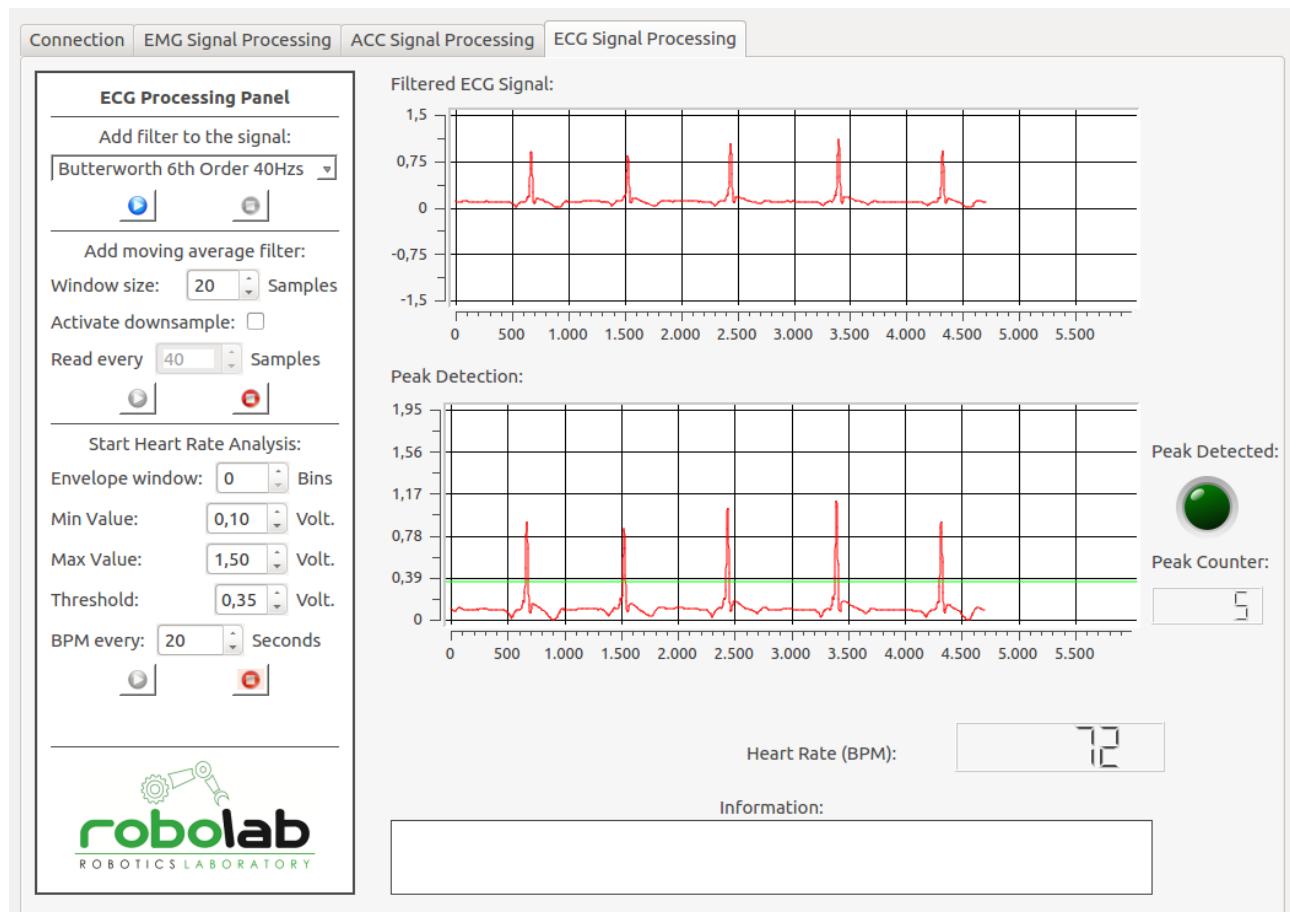


Ilustración 50: Pestaña ECG: Detección de BPM

En esta pestaña volvemos a disponer de un filtro de media móvil que trabaja en valor absoluto, y con un detector de picos, al igual que en la pestaña EMG.

El punto en que difiere es que en esta pestaña vamos a medir el ritmo cardíaco del sujeto que lleve conectados los electrodos de adquisición.

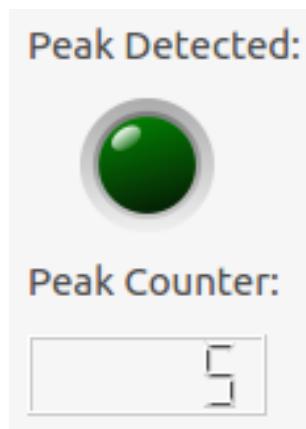
Para el cálculo de los latidos por minuto, se establece una ventana cíclica que define el intervalo de tiempo que tendremos que esperar para obtener la medición del ritmo cardíaco del “paciente”.

BPM every: Seconds

*Ilustración 51: Pestaña ECG:
Periodo de cálculo de BPM*

Cuanto más largo sea dicho intervalo, más precisas serán las mediciones, pero es fundamental que los picos del electro-cardiograma se detecten correctamente.

Para asegurarnos de que esto sucede, disponemos de un led que se iluminará cada vez que la aplicación detecte un pico en el electro-cardiograma, y de un contador de picos por intervalo que vuelve a 0 cada vez que comienza el intervalo de observación:



*Ilustración 52: Pestaña
ECG: Led de detección
y contador de picos*

Si hemos seguido todos los pasos correctamente, podremos saber con precisión a qué ritmo cardíaco late el corazón del sujeto de las pruebas.

Parte IV: Caso práctico: Interacción Hombre-Máquina

Una vez que nuestro componente es plenamente funcional, deseamos llevar a buen término nuestro propósito inicial de emplearlo como interfaz humano-robot y para ello es necesario emplear un dispositivo capaz de obedecer las órdenes que le dictemos por medio de los sensores que nos proporciona bitalino.

12. Presentación del robot: LearnBot

LearnBot (J.M. Haut et al.) es un robot que forma parte de Code2Bot, un proyecto de robótica educativa desarrollado en el Laboratorio de Robótica y Visión Artificial (ROBOLAB) de la Universidad de Extremadura.

El objetivo de dicho proyecto es emplear la robótica como herramienta multidisciplinar para el aprendizaje basado en proyectos colaborativos en entornos de Educación Primaria, Secundaria y universitaria. El proyecto busca por tanto integrar la robótica en entornos educativos y familiarizar a los docentes y alumnos en el uso de herramientas de ingeniería y visión por computador en niveles educativos más tempranos.

El dispositivo propiamente dicho es un pequeño robot modular diseñado específicamente para la introducción de la robótica en alumnos no necesariamente familiarizados con ella mediante una sencilla interfaz que permite recibir órdenes en lenguaje Python o C++.



Ilustración 53: Robot LearnBot

A pesar de su pequeño tamaño, LearnBot cuenta con 8 sensores de ultrasonidos repartidos por el perímetro del aparato, una cámara y un punto de acceso WIFI.

El movimiento del robot se lleva a cabo mediante un sistema de tracción diferencial similar al que se muestra en la siguiente ilustración:

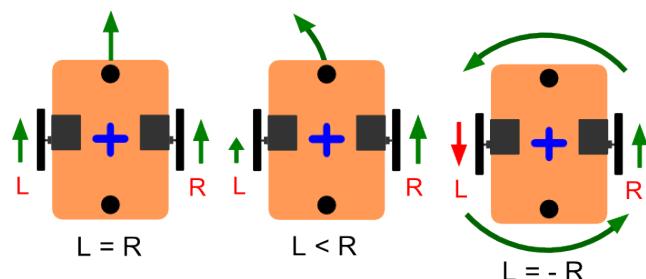


Ilustración 54: Sistema de tracción diferencial

Este sistema emplea el uso de dos motores independientes para cada rueda, permitiendo así que el robot gire en una dirección u otra, simplemente aplicando mas velocidad a una rueda u otra.

Esto permite además que nuestro robot gire sobre su propio eje.

El hardware de LearnBot esta basado en una placa Odroid-C1 con Linux y RoboComp instalados. Cuenta con baterías recargables y emplea un conector mini-usb estándar para su recarga.

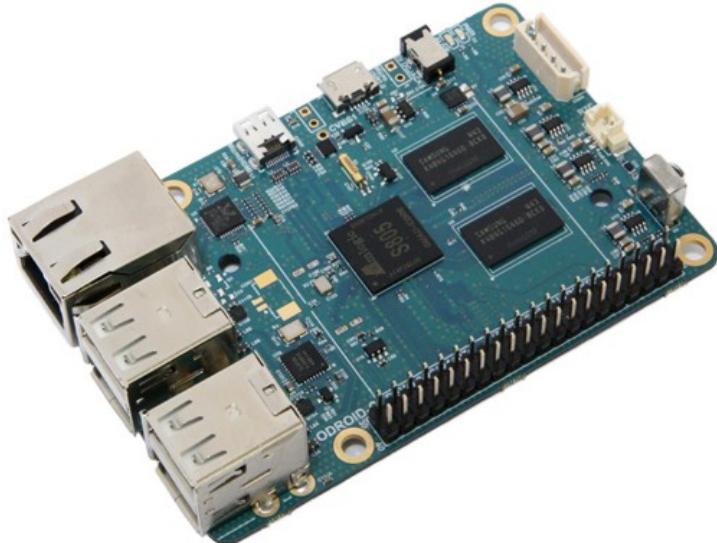


Ilustración 55: Placa de desarrollo ODROID-C1

Las fortalezas de la rebotica educativa, y de LearnBot como modelo de aprendizaje son:

- a) Integra diferentes áreas de conocimiento.
- b) Se trabaja con objetos manipulativos, permitiendo la transición de lo abstracto a lo concreto.
- c) Fuerza a los alumnos el empleo de diferentes lenguajes (gráfico, simbólico, matemático, etc.).
- d) Opera de manera síncrona con diferentes variables.
- e) Fomenta el desarrollo del pensamiento sistemático y sistémico.
- f) Contribuye a la construcción y evaluación de estrategias de adquisición de conocimiento mediante aprendizaje guiado.
- g) Crea entornos de aprendizaje por si mismo.
- h) Aplica y enseña el método científico y el modelado y representación matemáticas.
- i) Establece un entorno atractivo de aprendizaje heurístico.

13. Interfaz de control

La interfaz que vamos a definir para el control de nuestro dispositivo va a consistir en el uso del módulo EMG y de dos de los ejes proporcionados por el módulo ACC.

La señal EMG servirá como activar o desactivar nuestro robot. Forzando su avance o su detención basándonos en el movimiento muscular del antebrazo o del bíceps.

Los diferentes ejes del acelerómetro servirán para controlar la velocidad del robot y su orientación, funcionando únicamente si la señal EMG ha activado el robot previamente.

Señal	Movimiento del usuario	Información proporcionada	Comando para el robot
EMG	Usuario activa el músculo del antebrazo o bíceps.	Acción detectada: músculo contraído.	Activar robot: mover hacia adelante a una velocidad constante.
ACC: Eje Y	El usuario eleva el acelerómetro.	Ángulo de elevación positivo o negativo.	Aumento o disminución de la velocidad
ACC: Eje X	El usuario gira el acelerómetro hacia la izquierda o bien hacia la derecha.	Ángulo de giro positivo o negativo.	El robot ha de girar en un sentido o en otro un máximo de 90º

El módulo ECG se deja fuera del estudio, puesto que la información generada por el ritmo cardíaco del usuario, en principio no nos resulta útil como elemento de interacción hombre-maquina. Aunque su aplicación en el marco de tecnologías iHealth (salud inteligente) puede ser muy interesante, queda fuera del ámbito de trabajo del presente estudio.

En la siguiente ilustración se muestra la interfaz de control utilizando el módulo ACC de bitalino:

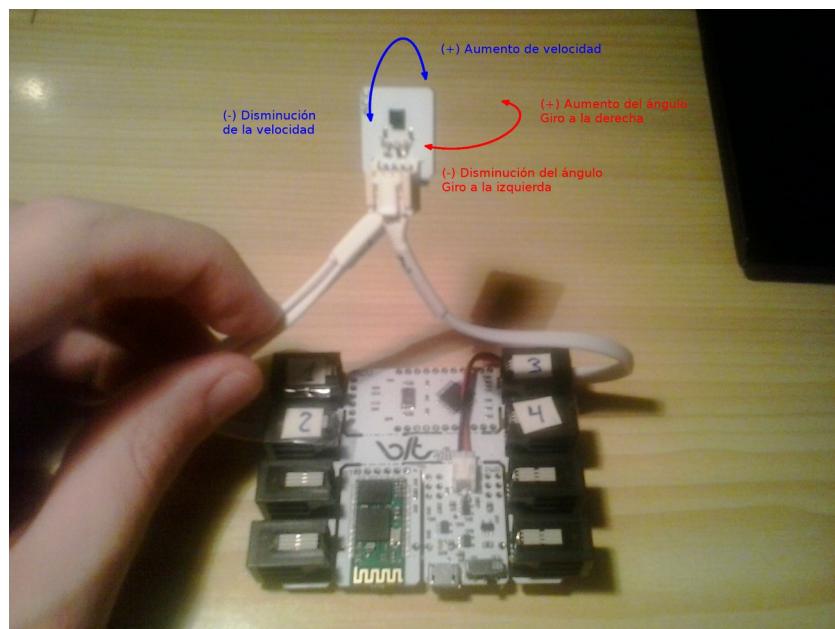


Ilustración 56: Sistema de control mediante módulo ACC

El módulo ACC colocado en posición completamente horizontal, generará una aceleración de 0g en nuestro eje Y, ordenando a nuestro robot a mantener una velocidad intermedia. Moviendo el módulo de acuerdo con las flechas azules ilustradas anteriormente, podemos lograr el aumento o disminución de la velocidad del robot.

- Si elevamos nuestro dispositivo ACC, se generará una aceleración positiva (hasta un máximo de 1g), lo cual aumentará la velocidad a la que se mueve el robot.
- Si inclinamos hacia abajo nuestra placa, se generará una aceleración negativa (hasta un mínimo de -1g), desacelerando nuestro robot hasta su completa detención de darse una inclinación suficiente.
- Si giramos el acelerómetro de acuerdo al sentido marcado por las flechas rojas en la ilustración anterior, lograremos generar fuerzas G en el eje X del acelerómetro. Un análisis análogo al anterior nos permitirá evaluar esas fuerzas para dar un sentido de giro al robot.

La colocación de los electrodos del sensor EMG tal y como se ha descrito en el apartado 6.1.2 puede verse en la siguiente figura:



Ilustración 57: Sistema de control mediante el módulo EMG

Aunque no puede verse en la ilustración, el cable de color negro proveniente del módulo de adquisición EMG, debe colocarse en un punto neutro que sirva como toma de tierra, en este caso hemos seleccionado el codo del usuario.

Es recomendable emplear distintas extremidades para el control del módulo ACC y del módulo EMG, para así no incurrir en errores y falsas lecturas, pues podría accionarse accidentalmente el músculo del antebrazo en un intento de controlar el acelerómetro.

14. Pruebas de uso

Deseamos comprobar el nivel de adaptación de un usuario promedio a los sistemas de adquisición por medio de sensores fisiológicos que hemos desarrollado a lo largo de todo este trabajo. También resulta interesante comprobar el nivel de satisfacción de los usuarios.

Para llevar a cabo este objetivo se ha puesto en marcha un estudio práctico evaluando la interacción de una serie de participantes con nuestro sistema. En nuestra prueba se pretende que dichos participantes sean capaces de controlar e interactuar con nuestro LearnBot por medio del dispositivo bitalino y llevar acabo una tarea sencilla con el robot.

La prueba debe servirnos para comprobar si es posible controlar el robot para un individuo no entrenado y si dicho participante se siente cómodo empleando dicho dispositivo.

14.1 Participantes

Hemos reunido 5 voluntarios (1 mujer y 4 hombres) de entornos alejados a la investigación y a los laboratorios Robolab en un esfuerzo por componer un conjunto de individuos no entrenados y con nula experiencia en el manejo de sistemas semejantes. De todos ellos 4 son diestros y 1 zurdo y el rango de edad oscila entre 29 y 69 años.

Existe también un monitor encargado de supervisar las pruebas y arrancar el sistema que se encargará de explicar el funcionamiento del mismo mediante una demostración, colocar los parches y calibrar el sistema de adquisición.

14.2 Procedimiento

Lo primero que vamos a hacer es recopilar los datos personales de cada usuario. Lo siguiente será realizar una serie de demostraciones para que el usuario pueda entender el sistema de control que tendrá que emplear para mover al robot. Seguidamente se colocarán los electrodos de adquisición sobre el usuario tal y como se mencionó en el punto 13. Después de esto se llevará a cabo una fase de calibración llevada a cabo de manera asistida por el investigador, y a cada usuario se le pedirá que complete una sencilla tarea que consistirá en mover el robot de un punto A a un punto B, girar y retornar al punto de partida. La distancia entre ambos puntos es de 1.35 metros.

Cada participante realizará la tarea un mínimo de dos veces, y su ejecución será grabada en vídeo siempre que sea posible.

Finalmente, se preguntará a cada participante acerca de su satisfacción respecto al sistema de control.

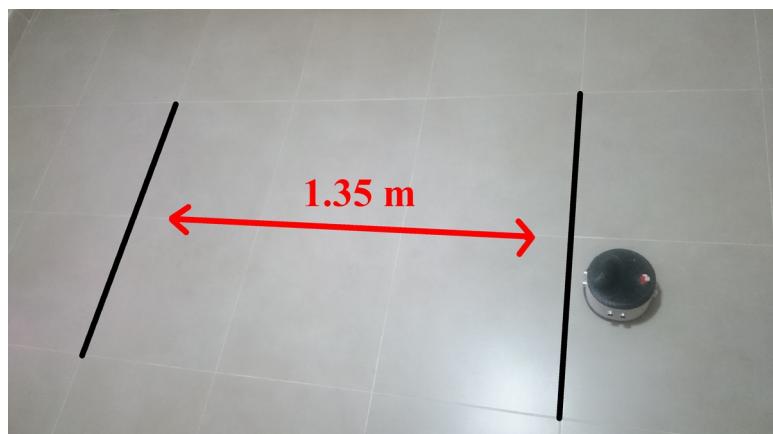


Ilustración 58: Circuito de prueba



Ilustración 59: Sujetos realizando las pruebas

14.3 Resultados de las pruebas

Los resultados de las pruebas varían mucho respecto a cada participante.

A continuación se exponen los resultados en cada caso concreto:

Usuario	1
Sexo:	Masculino.
Edad:	31
Tiempo prueba1:	1m 40s
Tiempo prueba2:	1m 27s
Observaciones:	El usuario se adapta rápidamente y sin incidencias al sistema de control, su prueba es ágil y se desenvuelve con facilidad. Su grado de satisfacción es bueno, aunque manifiesta la necesidad de crear un prototipo “wearable”, algún tipo de guante o periférico adaptativo.

Usuario	2
Sexo:	Femenino.
Edad:	67
Tiempo prueba1:	2m 50s
Tiempo prueba2:	No realizada.
Observaciones:	<p>El usuario no tiene problemas para manejar los acelerómetros y el aprendizaje se desarrolla con normalidad, sin embargo el usuario tiene problemas para adaptarse a los parches y lectura EMG. Debido a que se trata de una mujer de avanzada edad con poca masa muscular, resulta complicado encontrar un punto óptimo de lectura y la prueba resulta ardua. Además, los parches provocan reacción alérgica en el sujeto y se decide realizar una única prueba.</p> <p>Una vez solventados los problemas, el usuario logra utilizar y completar el circuito.</p> <p>Su satisfacción con el sistema es relativa.</p>

Usuario	3
Sexo:	Masculino.
Edad:	69
Tiempo prueba1:	1m 48s
Tiempo prueba2:	1m 15s
Observaciones:	<p>El usuario se adapta rápidamente al uso de los acelerómetros y también al EMG después de un corto aprendizaje. Maneja sin problemas el sistema de control y parece disfrutar de la prueba.</p> <p>A pesar de ser un hombre de edad avanzada, no parece tener problemas para asimilar nuestro sistema.</p> <p>Su grado de satisfacción es bueno y terminadas las pruebas dedica un tiempo extra a experimentar con el control y a hacer pruebas extra.</p>

Usuario	4
Sexo:	Masculino.
Edad:	42
Tiempo prueba1:	1m 02s
Tiempo prueba2:	1m 40s
Observaciones:	<p>Adaptación muy rápida al sistema de control. El usuario tiene composición atlética y un bajo índice de grasa corporal, por lo que resulta fácil tomar sus lecturas EMG, sin embargo el usuario registra un peor tiempo en la segunda prueba y no se ha podido registrar su test en vídeo.</p> <p>El usuario se pone nervioso cuando se le enfoca con la cámara y ello provoca que tense el antebrazo de manera involuntaria, activando con ello nuestro sistema EMG.</p>

Usuario	5
Sexo:	Masculino.
Edad:	29
Tiempo prueba1:	1m 26s
Tiempo prueba2:	1m 45s
Observaciones:	<p>Incidencias a la hora de capturar la señal EMG. BITalino transmite señales muy sucias con las que resulta imposible trabajar. Se cambian los parches y se aplica gel preparatorio.</p> <p>Finalmente descubrimos que una fuente de alimentación cercana esta produciendo interferencias que afectan a las lecturas de nuestro dispositivo. Despues de subsanar los problemas el usuario se adapta bien al control.</p> <p>La primera prueba se desarrolla muy rápido y sin incidentes. Durante la segunda prueba el usuario genera impulsos involuntarios con su antebrazo debido a que no lo tiene completamente relajado. Estos impulsos detienen el robot y ralentizan el circuito, dando como resultado un tiempo 19s superior al de la primera prueba.</p>

Del resultado de las pruebas podemos obtener las siguientes lecturas:

- El dispositivo es susceptible de recibir interferencias al no estar adecuadamente apantallado. Es necesario subsanar este punto en aras de obtener un prototipo más robusto.
- Los parches con electrodos que empleamos en nuestro prototipo son susceptibles de provocar reacciones alérgicas en determinados usuarios. Debemos valorar la posibilidad de emplear parches hipoalergénicos.
- Los usuarios manifiestan la necesidad de integrar nuestro sistema en un accesorio que le confiera mayor reusabilidad y comodidad, como puede ser un brazalete o guante. Este punto puede ser fácilmente subsanable si se implementa una prenda textil capaz de ubicar nuestro BITalino en el antebrazo del usuario.
- Los usuarios con poca masa muscular y con un elevado índice de grasa corporal tienen más problemas en adaptarse al sistema (debido a la dificultad de obtener lecturas positivas) que usuarios con una disposición atlética y poca grasa corporal.
- Es necesario un periodo de aprendizaje antes de poder utilizar nuestro modelo. No es necesariamente largo, pero ineludible.
- El sistema de calibración no siempre se adapta al usuario y debe emplearse una calibración manual de las fronteras de decisión en el caso de la detección EMG.
- El robot LearnBot adolece de algunos problemas de diseño que lastran su uso y condicionan sensiblemente su maniobrabilidad. La ausencia de un 4º punto de apoyo y su tendencia a moverse hacia la izquierda han influido en los tiempos de los experimentos.

15. Resultado, Conclusiones y Futuro

El sistema de interacción hombre-maquina resulta exitoso. Nuestro robot reacciona con una enorme sensibilidad tanto en entorno controlado (simulador virtual RCIS) como con el prototipo físico.

Las pruebas con nuestro LearnBot ponen de manifiesto la viabilidad y facilidad de uso del dispositivo bitalino en el ámbito de la robótica educativa.

Si bien se requiere de un cuidadoso ajuste y calibración antes de comenzar cada prueba, así como un correcto mantenimiento del equipo y de los elementos desechables (electrodos), nuestro sistema es capaz de controlar el robot con precisión, aunque requiere de cierto aprendizaje previo y una introducción al uso de los componentes.

Estos son los hitos fundamentales que podemos extraer del presente estudio:

- 1) Bitalino, y sus diferentes módulos, son una buena herramienta de adquisición biométrica debido fundamentalmente a tres motivos: bajo coste, flexibilidad y facilidad de uso.
- 2) Bitalino es una herramienta factible como elemento de interacción hombre-maquina. Si bien no todos sus módulos son útiles para este propósito, el módulo EMG y ACC proporcionan señales muy favorables.
- 3) Una buena configuración del dispositivo en términos de mantenimiento del mismo, apantallado, filtrado y procesado de la señal son absolutamente críticos para lograr buenos resultados.
- 4) El componente de software desarrollado abre la vía a futuras ampliaciones y a una mayor profundización en este campo. Nuevos ámbitos de estudio incluyen la implementación de filtros por hardware, la incorporación de nuevos módulos de adquisición, la implantación de modelos de aprendizaje, etc.
- 5) Nuestro trabajo abre una interesante vía de estudio con la inclusión de elementos de **monitorización de la salud** del usuario o ihealth dentro de robocomp. Esta vía no se ha desarrollado en este trabajo, si bien el sistema de adquisición implementado es un buen punto de inicio.
- 6) Nuestro trabajo abre una interesante vía de estudio hacia el **reconocimiento gestual** por medio de señales biométricas. Esta vía no se ha desarrollado, pero al igual que en el punto anterior, nuestro sistema de adquisición es una buena base con la que comenzar dicho estudio.

Notas Finales: Bibliografía y Anexos

16. Bibliografía

- Bernardino António, R. Y. and B. J. (2014). Versatility of human body control through low-cost electromyographic interface. In *Proc. of the Int'l Conf. on Applications of Computer Engineering (ACE)*, (OCTOBER), 87–92. <http://doi.org/10.13140/2.1.1454.3686>
- Cintas, R., Calderita, L., Manso, L., Bustos, P., & Bachiller, P. (2010). Un framework de Desarrollo para Robótica. *I Jornadas Jóvenes Investigadores Universidad de Extremadura*.
- Dorran, D. (2013). Filter Design Using Matlab Demo. Retrieved from <https://dadorran.wordpress.com/2013/10/18/filter-design-using-matlab-demo/>
- Fisher, T. (1999). Interactive Digital Filter Design. Retrieved from <https://www-users.cs.york.ac.uk/~fisher/mkfilter/>
- Gamech, B., Guerreiro, J., Alves, A. P., Lourenço, A., da Silva, H. P., Gardeazabal, L., ... Fred, A. (2014). Evaluation of a context-aware application for mobile robot control mediated by physiological data: The ToBITas case study. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8867, 147–154.
- Haut, J. M., Paoletti, M. E., Bustos, P., & García, N. (2015). Code2Bot, a social robot for the classroom. *XVI Conferencia de La Asociación Española Para La Inteligencia Artificial*, 1055–1066.
- Hidalgo Paniagua, A., Mateos Sánchez, J., Núñez Trujillo, P., Bustos, P., & Moreno del Pozo, J. (n.d.). LearnBot: A Robotic-based prototype for Educational Environments. *Workshop Hispano-Brasileiro in Autonomous Robots and Robotics Intelligence*.
- Moro Cuevas, P. (2008). Tratamiento y procesado digital de la señal con DSP. Técnicas de procesado digital de la señal aplicadas a acelerómetros. Proyecto Fin de Carrera, Escuela Universitaria de Ingeniería Técnica de Telecomunicación. Universidad Politécnica de Madrid.
- Peters, T. (2014). An Assessment of Single-Channel EMG Sensing for Gestural Input. *Dartmouth College*, 1–14.
- Rueda Cebollero, G. (2011). Procesado de señales electroencefalográficas para determinar características espectrales de episodios epilépticos. Proyecto de Fin de Carrera, Escuela Politécnica Superior. Universidad de Lleida.
- Sánchez-Tembleque Díaz-Pache, F. (2008). Tratamiento de Datos en las Técnicas Instrumentales (MATLAB). Curso, Universidad da Coruña. Master en Asistencia e Investigación Sanitaria.
- Silva, H. (BITalinoWorld). (2014). Breaking out the X and Y axis on a BITalino Plugged Accelerometer. Retrieved from https://www.youtube.com/watch?v=rh8y_NsVLI4
- Universidade Nova de Lisboa. (2015). BITalino Tutorial: How to do an ECG. Retrieved from <https://www.youtube.com/watch?v=Usiiofm-khY>

Varios. (n.d.). Especificaciones Técnicas, manuales de uso, API y documentación. PLUX – Wireless Biosignals. Retrieved from www.bitalino.com

17. Anexo I: Instalación de RoboComp

El presente anexo es una traducción al español del manual de instalación oficial de RoboComp que puede encontrarse en <https://github.com/robocomp/robocomp>. Se incluye por motivos prácticos.

Previa a la instalación de RoboComp, debemos instalar los siguientes paquetes, que podemos obtener de los repositorios de Ubuntu:

```
sudo apt-get update
```

```
sudo apt-get install git git-annex cmake g++ libgsl0-dev libopenscenegraph-dev cmake-qt-gui  
zeroc-ice35 freeglut3-dev libboost-system-dev libboost-thread-dev qt4-dev-tools yakuake openjdk-  
7-jre python-pip python-pyparsing python-numpy python-pyside pyside-tools libxt-dev pyqt4-dev-  
tools qt4-designer libboost-test-dev libboost-filesystem-dev
```

A continuación instalaremos robocomp, debemos situarnos en el directorio home utilizando el comando cd, y a continuación teclear:

```
git clone https://github.com/robocomp/robocomp.git
```

Ahora crearemos un enlace simbólico de modo que RoboComp pueda encontrar todo. Debemos introducir nuestra contraseña. Emplearemos el siguiente comando:

```
sudo ln -s /home/<tu-usuario-linux> /home/robocomp
```

Debemos editar nuestro fichero bashrc:

```
gedit ~/.bashrc
```

Añadimos las siguientes líneas al final del documento:

```
export ROBOCOMP=/home/<your-linux-user>/robocomp
```

```
export PATH=$PATH:/opt/robocomp/bin
```

Para que el bash procese el fichero modificado anteriormente debemos teclear:

```
source ~/.bashrc
```

Listo! Ahora compilaremos e instalaremos todo:

```
sudo rm -r /opt/robocomp
```

```
cd robocomp
```

```
mkdir build
```

```
cd build
```

```
cmake ..
```

```
make
```

```
sudo make install
```

Las librerías fundamentales y el simulador de RoboComp deben ahora ser compiladas e instaladas en `~/opt/robocomp`.

Le vamos a decir a Linux donde encontrar las librerías de RoboComp:

```
sudo nano /etc/ld.so.conf
```

añadimos la siguiente linea:

```
/opt/robocomp/lib/
```

guardamos los cambios y escribimos:

```
sudo ldconfig
```

Listo! Hagamos ahora algunas pruebas:

Podemos probar la instalación haciendo uso del simulador de robótica RCIS.

Primero debemos de obtener algunas texturas y elementos empleados por el simulador. Puede tardar un poco:

```
cd ~/robocomp  
git annex get .
```

Ahora podemos arrancar el simulador:

```
cd ~/robocomp/files/innermodel  
rcis simpleworld.xml
```

RCIS debería estar arrancado y funcionando con un sencillo robot equipado con un laser y una cámara RGBD, moviéndose sobre una superficie de madera. No debemos olvidar mover el entorno gráfico para poder observar el robot adecuadamente.

A continuación instalaremos algunos componentes desarrollados por RoboLab obteniéndolos a través de GitHub.

Lo que hemos instalado hasta ahora es tan solo el núcleo de RoboComp (el simulador, un generador de componentes y algunas librerías). Para disponer de otras características como por ejemplo el joystick de control, debemos ejecutar componentes de software adicional disponibles en otros repositorios, como robocomp-robolab:

```
cd ~/robocomp/components  
git clone https://github.com/robocomp/robocomp-robolab.git
```

El conjunto de componentes básicos de RoboLab se habrá descargado. Podemos encontrarlo en `~/robocomp/components/robocomp-robolab/components`

Si disponemos de un joystick USB, es el momento de probarlo. Podemos conectarlo y lanzar el componente que nos permitirá utilizarlo dentro de robocomp:

```
cd ~/robocomp/components/robocomp-robolab/components/joystickComp  
cmake .  
make  
cd bin  
sudo addgroup your-user dialout // To solve some permissions issues in Ubuntu  
./startJoyStick.sh
```

El joystick debería estar funcionando. Hará que el robot avance o gire a nuestra voluntad. Si el componente no arranca o el robot no se mueve, podemos detener el componente con el siguiente comando:

```
./forceStopJoyStickComp.sh
```

Debemos comprobar donde ha sido creado el fichero de dispositivo joystick (por ejemplo `'/dev/input/js0'`). Si dicha ruta no es `'/dev/input/js0'`, entonces debemos editar el fichero `~/robocomp/components/robocomp-robolab/components/joystickComp/etc/config` cambiándolo de acuerdo a la ruta real y reiniciar el componente.

Si no disponemos de joystick, es posible utilizar el teclado a modo de joystick. Para ello debemos instalar otro componente:

```
cd ~/robocomp/components/robocomp-robolab/components/keyboardrobotcontroller  
cmake .  
make  
src/keyboardrobotcontroller.py --Ice.Config=etc/config
```

de este modo podemos utilizar las flechas del teclado para mover el robot. La barra espaciadora para detener el robot y la tecla 'q' para salir.

18. Anexo II: Generación de un componente en Robocomp usando robocompdsl

El presente anexo es una traducción al español del manual escrito por Rajath Kumar (RNS Institute of Technology) que puede encontrarse en el siguiente enlace:

<https://github.com/rajathkumarmp/RoboComp-Tutorial/blob/master/doc/tut2.md>

Robocompdsl es una herramienta en linea de comandos que nos facilita la tarea de crear y compilar componentes. Explicamos a continuación como utilizarla.

Para empezar debemos crearnos un directorio para nuestro primer componente:

```
mkdir firstcomp  
cd firstcomp
```

Para poder generar un componente, antes debemos escribir un fichero cdsl. Los ficheros cdsl emplean un lenguaje de definición de componentes que pretende ayudarnos a generar el código completo del componente en el lenguaje deseado (habitualmente C++ o Python).

Dicho código resultaría enormemente tedioso de escribir manualmente, ya que contiene gran cantidad de código rutinario.

Para crearnos un nuevo fichero cdsl, podemos ejecutaremos el siguiente comando:

```
robocompdsl first.cdsl
```

Tras ejecutar el comando anterior se habrá generado una plantilla cdsl dentro de nuestra carpeta firstcomp.

Dicho archivo contendrá el siguiente código:

```
import "/robocomp/interfaces/IDSLs/import1.idsl";  
import "/robocomp/interfaces/IDSLs/import2.idsl";  
  
Component test  
{  
    Communications  
    {  
        implements interfaceName;  
        requires otherName;  
        subscribesTo topicToSubscribeTo;  
        publishes topicToPublish;  
    };  
    language Cpp;  
    gui Qt(QWidget);  
};
```

Este es el código esqueleto para cualquier fichero cdsl. Debe ser editado para adaptarlo al componente que deseamos generar.

Veamos como puede modificarse dicho esqueleto para generar un componente que haga uso del idsl ‘differential robot’. Debemos modificar el código anterior de la siguiente manera:

```

import "/robocomp/interfaces/IDSLs/DifferentialRobot.idsl";
Component first
{
    Communications
    {
        requires DifferentialRobot;

    };
    language Cpp;
};

```

Ahora disponemos de un cdsl que se adecúa a nuestras necesidades. El nombre del componente será ‘first’. Hará uso de DifferentialRobot y estará codificado en C++. Si además deseamos que disponga de una interfaz gráfica codificada en un archivo con extensión .ui, podemos dejar la linea *gui Qt(QWidget)* dentro del cdsl.

A partir del anterior cdsl, podemos generar un componente en C++. Para ello volveremos a emplear el comando robocompdsl de la siguiente manera:

robocompdsl first.cdsl build

En el anterior comando, build es el directorio donde el código será generado.

Después de generar el componente, debemos compilar el código mediante los siguientes comandos:

cd build
cmake .
make

En este punto, habremos logrado con éxito crear y compilar un componente haciendo uso de la herramienta robocompdsl.

19. Anexo III: Instalación de Qwt

El presente anexo es una traducción al español del manual de instalación oficial de Qwt que puede encontrarse en <http://qwt.sourceforge.net/qwtinstall.html>. Estas librerías son imprescindibles para poder ejecutar el componente.

Si disponemos de una distribución reciente de ubuntu, es posible encontrar las librerías Qwt en repositorios (versión 6.1.2 en el momento de redacción de este documento). Para ello debemos buscar los paquetes: libqwt-dev, libqwt-headers y libqwt5-qt4 (este último es tan solo un entorno de ejecución y no nos permitirá desarrollar software utilizando Qwt).

Para la instalación manual debemos seguir el siguiente tutorial:

Lo primero que debemos hacer es descargar la versión que deseemos de Qwt. Pueden encontrarse en el siguiente enlace: <https://sourceforge.net/projects/qwt/files/qwt/>. En el momento de la redacción de este documento, la última versión (en el momento de la redacción de este tutorial) de Qwt es la 6.1.3.

Instalando Qwt

Aparte de las cabeceras, las librerías y la documentación de las clases en html, una instalación apropiada de Qwt contiene un plugin para Qt Creator y un archivo de configuración para compilar aplicaciones usando Qwt.

Todos los archivos serán copiados a un directorio de instalación que es configurable editando el archivo qwtconfig.pri. Su configuración por defecto es:

/usr/local/qwt-6.1.3

Para el resto del tutorial, esta ruta será referenciada como *\${QWT_ROOT}* y ha de ser sustituida por la ruta real en los comandos que siguen.

No es imposible tener mas de una instalación de Qwt en el mismo sistema. Por ejemplo para utilizar el plugin Qwt Designer en Qt Creator es necesario que la versión de Qwt haya sido compilada con la misma combinación de Qt y compilador que se haya usado para compilar Qt Creator (ver “Help→About Qt Creator...”).

La instalación de Qwt se realiza en 3 pasos, que son muy comunes en sistemas UNIX.

1. Configuración

En este paso establecemos todos los parámetros que son necesarios para controlar como se va a compilar e instalar Qwt.

2. Compilación

En este paso los ficheros binarios son compilados a partir de los ficheros fuente.

3. Instalación

Durante la instalación se copian y organizan todos los ficheros que son necesarios para construir aplicaciones Qwt dentro del directorio destino.

La instalación no modifica el sistema más allá de copiar los archivos en un directorio de la manera apropiada. Después de borrar ese directorio el sistema queda en el mismo estado que antes de comenzar la instalación.

Configuración

La configuración de Qwt se lleva acabo editando los archivos de proyecto empleados para la compilación:

- `qwtbuild.pri`
`qwtbuild.pri` contiene los parámetros acerca de como se debe compilar Qwt. Todas las características de este archivo son únicamente para compilar Qwt y no tienen ningún impacto en como una aplicación que utilice Qwt ha de ser compilada. Normalmente sus parámetros por defecto no necesitan ser modificados.
- `qwtconfig.pri`
`qwtconfig.pri` define que módulos de Qwt serán compilados y donde se van a instalar.
`qwtconfig.pri` se instala conjuntamente con el archivo de características `qwt.prf` y todos sus parámetros son conocidos por los archivos de proyecto que compilan aplicaciones Qwt.

En el fichero `qwtconfig.pri` el significado de cada opción se encuentra explicado con detalle. Merece la pena leerlo antes que incurrir en problemas después.

Compilación e instalación

Qt Creator es una interfaz gráfica que permite realizar llamadas a `qmake/make` de modo que técnicamente podría usarse para compilar e instalar Qwt si abrimos el fichero de proyecto con Qt Creator. Sin embargo este modo requiere comprender muchos detalles, las siguientes instrucciones paso a paso son un camino mas sencillo utilizando la consola de comandos.

El primer paso antes de crear el Makefile es comprobar que estamos utilizando la versión correcta de `qmake`. Por ejemplo las versiones antiguas de distribuciones Linux a menudo utilizan un `qmake` perteneciente a Qt3.

La configuración por defecto de `qmake` normalmente genera un makefile que compila Qwt para el mismo entorno para el cual la versión de `qmake` ha sido compilada. Así que crear el makefile normalmente significa algo como esto:

```
cd qwt-6.1.3
/usr/local/Qt-5.0.1/bin/qmake qwt.pro
```

El makefile generado incluye todos los path relacionados con la versión escogida de Qt. El siguiente paso es:

```
make
```

(En sistemas multinúcleo podemos acelerar la compilación de las librerías Qwt lanzando más de un proceso simultaneo: por ejemplo “`make -j4`” en un sistema con dos núcleos).

Por último tenemos que instalar todo en los directorios que hemos especificado previamente en el archivo `qwtconfig.pri`. Normalmente será un de los directorios de sistema (`/usr/local`, `/opt`, ...) donde no disponemos de permisos de escritura, de modo que la instalación debe llevarse a cabo empleando permisos de root:

```
sudo make install
```

(En sistemas donde `sudo` no existe podemos realizar la misma acción con: `su -c “make install”`).