UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ELECTRÓNICO PARA LA MONITORIZACIÓN DE LA MARCHA EN PERSONAS CON LESIÓN MEDULAR

ÁLVARO GUTIÉRREZ TENORIO

2019

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ELECTRÓNICO PARA LA MONITORIZACIÓN DE LA MARCHA EN PERSONAS CON LESIÓN MEDULAR

ÁLVARO GUTIÉRREZ TENORIO

Tutora:

BLANCA LARRAGA GARCÍA

2019

Agradecimientos

A mis tutores, Blanca y Álvaro. Gracias Blanca por tu paciencia y por haberme enseñado parte de este interesantísimo mundo que es la Ingeniería Biomédica. Gracias por tu ayuda en esas mañanas de angustia en Toledo que sin duda han sido esenciales para este proyecto. Gracias Álvaro por quitarle hierro al asunto cuando las cosas no marchaban bien y por seguir forzándome a dar más y más de mí. Gracias a ambos por vuestro apoyo y consejo para sacar adelante este TFG.

También quiero dar las gracias a todo el equipo de la Unidad de Biomecánica y Ayudas Técnicas del Hospital Nacional de Parapléjicos de Toledo, por acogerme como uno más y por vuestro apoyo. Y en especial a Antonio, por ayudarme en todo lo que he necesitado para la realización de este Trabajo.

Gracias a mis compañeros de Grado, a Sergio por compartir conmigo esos días de frustración y pocas ganas de trabajar y por ayudarme a despejarme, aunque al final acabáramos hartos uno del otro. Gracias a Jaime, Jorge, Luis y Pablo por sacarme una sonrisa con vuestras tonterías incluso cuando no tenía ganas de nada.

Gracias a mis amigos por estar siempre ahí cuando les he necesitado.

Gracias Isa, por apoyarme incondicionalmente, por hacerme ver el lado bueno de las cosas y por seguir a mi lado pese a mis agobios. Aún nos queda mucho por delante.

Gracias a mi familia, a David por ser el hermano mayor que cualquiera querría tener y a mis padres por ser tan comprensivos y un ejemplo para mí.

Resumen

Una lesión medular es cualquier alteración de la médula espinal que suponga una interrupción de las vías de conexión neurológica que van desde el cerebro hasta el resto del cuerpo. Según la localización donde se ha producido la lesión y su grado, las consecuencias pueden ir desde la pérdida de la función motora y sensitiva parcial hasta la pérdida total de las mismas.

La recuperación de la marcha es una parte fundamental de las sesiones de **rehabilitación**. Con los avances en la tecnología y, en particular, en la biomecánica, la utilización de dispositivos para el restablecimiento de la marcha es cada vez más utilizada en el ámbito clínico. Entre los dispositivos utilizados para este fin, se encuentran las **neuroprótesis**, que utilizan la estimulación eléctrica sobre ciertos músculos para mejorar y rehabilitar la función motora; y los **exoesqueletos robóticos**, estructuras robóticas con diferentes articulaciones y actuadores en cada articulación que permiten rehabilitar también la capacidad de andar.

Este Trabajo de Fin de Grado surge de la necesidad detectada en la unidad de Biomecánica y Ayudas Técnicas en el Hospital Nacional de Parapléjicos de Toledo (HNPT) para la monitorización a pacientes durante diferentes pruebas de la marcha. Por tanto, será necesario el **diseño e implementación de un sistema electrónico capaz de monitorizar la marcha**, mediante la extracción de datos del exoesqueleto **Exo-H2** con el que cuenta el hospital y, en paralelo, de los diferentes dispositivos usados en las pruebas que realizan.

En primer lugar, se han definido los requisitos que debe cumplir el sistema a diseñar, entre ellos, las características físicas y funcionales. Tras ello, se ha elegido la plataforma sobre la cual se lleva a cabo el desarrollo hardware y software del diseño. También se han explicado los fundamentos del **protocolo CAN**, ya que mediante él se produce la comunicación con el exoesqueleto.

A continuación se ha desarrollado el diseño del sistema y de los bloques por los que está formado el programa que capturará los datos. También se ha realizado un análisis de tiempos de ejecución y se han llevado a cabo una serie de mejoras del sistema.

Por último, se ha validado la funcionalidad del sistema; primero en el Laboratorio de Robótica y Control de la Universidad Politécnica de Madrid (Robolabo) y después en el HNPT. De los datos obtenidos en el ensayo realizado en el HNPT, se ha comprobado que la implementación del sistema ha sido satisfactoria y que cubre las

necesidades para las que el sistema fue diseñado.

Tras la realización de este Trabajo de Fin de Grado, se ha podido contemplar la importancia que tiene un sistema de análisis y extracción de datos en las sesiones de análisis de la marcha. Además, la utilidad de un sistema capaz de integrar diferentes dispositivos en una misma configuración favorece el análisis conjunto de los datos.

Como líneas de trabajo futuras se espera continuar con la validación en diferentes configuraciones a las ya realizadas, así como buscar posibles mejoras del sistema.

PALABRAS CLAVE: EXOESQUELETO, CAN BUS, RASPBERRY PI, REHABILITACIÓN, SISTEMA ELECTRÓNICO.

Abstract

A spinal cord injury is any alteration of the spinal cord that interrupts the neurological connection between the brain and the rest of the body. Depending on the location and the severity of the lesion, consequences can be a partial or total loss of the motor and sensitive function.

Gait recovery is an essential element in **rehabilitation** sessions. With the progress in technology and, in particular, in biomechanics, devices are more and more often used in clinical settings in order to restore gait. Among the devices used for this purpose, there are **neuroprosthesis**, which use electrical stimuli to improve and rehabilitate the motor function, and **exoskeletons**, robotic structures with different articulations and actuators in each articulation capable of restoring gait.

This Thesis arises from the detected need at the Biomechanics and Technical Aids Department of the National Hospital for Spinal Cord Injury in Toledo (NHSI) for monitoring patients in different gait tests. As a result, **the design and implementation of an electronic device for gait monitoring** is needed, extracting information from the exoskeleton **Exo-H2** that NHSI holds and the different devices used in gait tests.

First of all, the requirements the device must fulfil were highlighted, physical and functional features among others. Moreover, the development platform was chosen and the **CAN protocol** basics were explained too, as the exoskeleton communicates through it.

Once the device design process has been developed, ways of extracting data and the main program blocks were highlighted. A runtime analysis was also performed, and some improvements were carried out.

Finally, the device was validated, first of all in the laboratory and later on in the NHSI. From the data obtained during the validation phase, it was concluded that the device implementation was satisfactory and that it fulfilled the needs which the device was designed for.

After the completion of this Thesis, it was proved the importance of having a system which analyses and extracts information from gait analysis tests and the benefits that a device capable of integrating different devices in the same configuration in order to analyse all data together has. As future developments it is expected to validate the device functionality in different configurations from those already made and the search of potential improvements.

KEYWORDS: EXOSKELETON, CAN BUS, RASPBERRY PI, REHABILITATION, ELECTRONIC DEVICE .

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

ADC: Analogic Digital Converter

ARM: Advanced RISC Machines

CAN: Controller Area Network

CSV: Comma Separated Values

GPIO: General Purpose Ins and Outs

IEEE: Institute of Electrical and Electronics Engineers

I2C: Inter Integrated Circuit

LAN: Local Area Network

POSIX: Portable Operating System Interface

PGA: Programmable Gain Amplifier

RAM: Random Access Memory

RISC: Reduced Instruction Set Computer

SPI: Serial Peripheral Interface

Capítulo 1

Introducción

Este Trabajo de Fin de Grado ha sido realizado junto con el Laboratorio de Robótica y Control (Robolabo)[1] y el Hospital Nacional de Parapléjicos de Toledo[2].

En este Capítulo se expone el marco en el que se desarrolla este Trabajo de Fin de Grado. En primer lugar, se realiza una introducción al concepto de lesión medular. Posteriormente, se enuncia la situación actual de los sistemas electrónicos aplicados a la medicina. Tras ello, se explicarán las características del exoesqueleto H2 para el que se diseña el sistema electrónico, objeto este Trabajo de Fin de Grado. Y, por último, se explicarán los requisitos que debe cumplir el sistema.

1.1. Lesión Medular

Una lesión medular es cualquier alteración de la médula espinal que supone una interrupción de las vías de conexión neurológica que van desde el cerebro hasta el resto del cuerpo. El origen de esta lesión puede ser por causa congénitas, traumáticas o enfermedad. Según la localización y grado de la lesión medular, las consecuencias de la lesión pueden ir desde una pérdida parcial de la función sensitiva y motora hasta una pérdida total de las mismas.

Dependiendo de su extensión, la lesión puede ser completa cuando se interrumpen todas las conexiones medulares por debajo del lugar de la lesión; o incompleta, en la que persiste parcial o totalmente la inervación motora, sensitiva y autónoma [1].

Una de las formas más extendidas de catalogar las lesiones medulares es según su localización, mediante los términos paraplejia y tetraplejia. La paraplejia se asocia a la lesión medular en la que se ha perdido la sensibilidad y la capacidad motora del tren inferior del cuerpo. Las lesiones de este tipo se ubican en el área dorsal, lumbar o sacra. Por otro lado, la tetraplejia se asocia a la pérdida de las capacidades sensitiva y motora del tren inferior y superior y la lesión se localiza en el área cervical (ver Figura 1.1).

La asociación americana para el estudio de la lesión medular (American Spinal Injury Association, A.S.I.A) [2] establece 5 categorías en función de la gravedad de la lesión, tratándose el grado A de una lesión completa, y los grados B, C, D y E de lesiones incompletas.

[1] http://www.robolabo.etsit.upm.es

[2] https://hnparaplejicos.sescam.castillalamancha.es

En la lesión de grado A, no existe preservación motora ni sensitiva en los segmentos sacros. El grado B se corresponde con una lesión medular en la que no hay función motora pero sí sensitiva por debajo de la lesión. En el grado C, existe capacidad sensitiva y capacidad motora parciales. El grado D es diagnosticado si la capacidad sensitiva es normal, y el grado E si tanto la función sensitiva como la motora son normales [1].

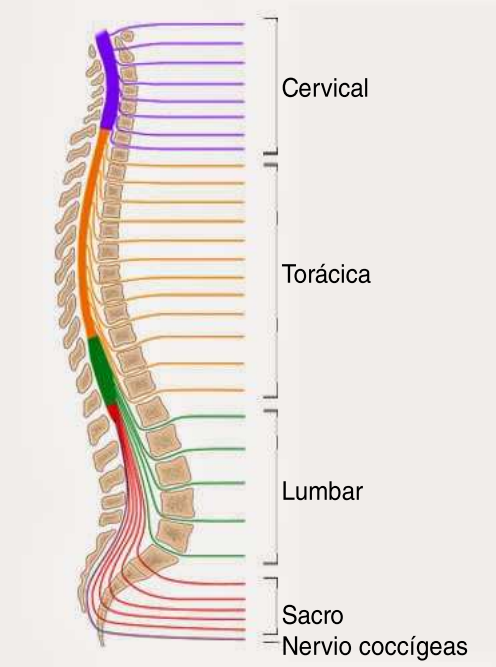


Figura 1.1: Partes de la médula espinal.

En la actualidad, según el Instituto Nacional de Estadística (INE), hay un 2.56‰de la población española afectada por una lesión medular, lo que supone alrededor de 108.200 habitantes españoles. Gracias a los avances médicos y tecnológicos de los últimos años, la esperanza de vida de estas personas ha incrementado de forma notable, llegando incluso a igualarse con la del resto de población [3, 4].

1.2. Sistemas electrónicos en la actualidad

Con los avances en la tecnología y, en particular, en la biomecánica, la utilización de dispositivos para el restablecimiento de la marcha es cada vez más empleada en el ámbito clínico.

Las aplicaciones de las nuevas tecnologías a la biomécanica van más allá de los instrumentos de medida. También forman parte de ellas sistemas expertos en clasificación y análisis de datos. Mediante estos sistemas, se pueden almacenar las bases de datos de los expedientes clínicos y generar los criterios clínicos de valoración de los datos obtenidos en las pruebas [5].

Las principales utilidades de estos sistemas son el control evolutivo de pacientes, la valoración del daño corporal y la planificación de tratamientos rehabilitadores,

quirúrgicos o farmacológicos.

La afectación de la capacidad motora de las extremidades inferiores puede derivar en la pérdida total de la capacidad del movimiento voluntario del tren inferior y, por tanto, se pierde la capacidad de andar. Esta pérdida supone para los pacientes, en la mayoría de los casos, la dependencia de una silla de ruedas para desplazarse. Es por ello que la recuperación de la marcha es una parte fundamental de las sesiones de rehabilitación [6].

En estas sesiones de rehabilitación, pueden utilizarse diferentes dispositivos, como neuroprótesis y exoesqueletos robóticos. Cada uno de estos sistemas proporciona información de manera diferente. Esto hace necesaria la unificación de los datos obtenidos de estos sistemas mediante una electrónica específica [7, 8].

1.2.1. Neuroprótesis

Una neuroprótesis es un dispositivo que, gracias a la estimulación eléctrica funcional estimula los músculos del paciente mientras, en paralelo, se realiza una actividad motora con el objeto de rehabilitar. De esta manera, no es necesario que la información llegue a través de la médula espinal sino a través de un dispositivo externo que ayuda a mantener la musculatura activa. Esto permitiría la restauración de las funciones perdidas del paciente con parálisis severa como resultado de una lesión medular [7].

Figura 1.2: Electrodos colocados sobre la pierna izquierda de un paciente [9].

Su aplicación para la rehabilitación de la marcha se realiza a través de la estimulación eléctrica de ciertos músculos de la pierna para que hagan el trabajo necesario mientras el paciente camina. Un ejemplo se muestra en la Figura 1.2. Consiste en la colocación de una banda con electrodos que se coloca alrededor de la pierna para estimular el músculo que levanta el pie al dar un paso. Estos electrodos no necesitan de una banda para ser colocados, sino que también pueden ir en pequeñas almohadillas por separado y colocados sobre la piel o implantados quirúrgicamente [10].

1.2.2. Exoesqueleto robótico

Los exoesqueletos robóticos o exoesqueletos de potencia son estructuras externas que los pacientes pueden ponerse y son controladas para alimentar un sistema de motores que tiene como función ayudar en la rehabilitación de la marcha.

Los exoesqueletos han surgido como una herramienta de rehabilitación ventajosa para individuos con lesión medular. En comparación con los paradigmas ya existentes sobre entrenamiento locomotor, los exoesqueletos pueden ofrecer tanto independencia como el establecimiento de patrones de marcha incrementando fuerzas en caso de que sea necesario para realizar ciertos movimientos. También permiten ser adaptados al paciente y sostienen su cuerpo de forma externa [11].

Figura 1.3: Sesión de rehabilitación con exoesqueleto en el Hospital Nacional de Parapléjicos de Toledo [12].

Actualmente existen varios modelos de exoesqueletos evaluados para el ámbito clínico, como el ReWalk, Vanderbilt exoskeleton, Ekso, Kinesis y Exo-H2 [13]. En el Hospital Nacional de Parapléjicos de Toledo cuentan con un Exo-H2 para realizar las sesiones de rehabilitación y análisis de la marcha, y es por ello que nos centraremos en él.

1.3. Exoesqueleto Exo-H2

El Exo-H2 es un exoesqueleto que ha sido diseñado para permitir un entrenamiento intensivo de marcha sobre el suelo. Este habilita el entrenamiento longitudinal de marcha en pacientes con lesión medular. Es un sistema seguro y robusto que abre la oportunidad a estudiar posibles formas de optimizar el proceso de rehabilitación.

Cuenta con seis articulaciones motorizadas, incluyendo cadera, rodilla y tobillo en ambas piernas. Hasta la fecha, no existe otro exoesqueleto ambulatorio utilizado para rehabilitación con el tobillo motorizado. Además de los motores, cuenta con 6 potenciómetros, 18 sensores de efecto Hall, 24 galgas extensiométricas y 4 interruptores de pie (ver Figura 1.4). Un conjunto de baterías de polímero de litio recargables alimentan el exoesqueleto.

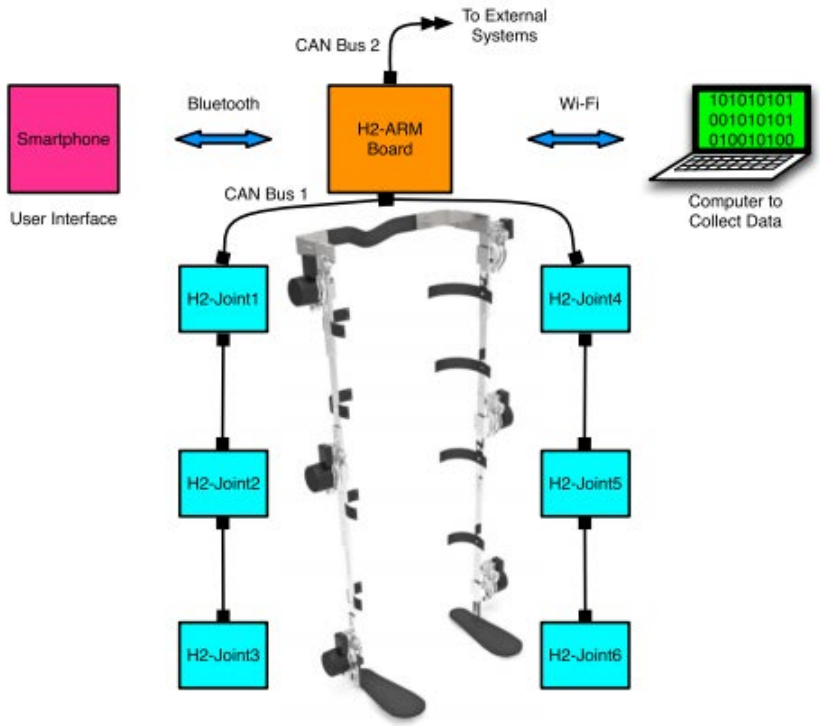


Figura 1.4: Arquitectura de red del exoesqueleto Exo-H2 [13].

El Exo-H2 presenta una arquitectura abierta que permite ser integrada con otros dispositivos o sistemas. Para este objetivo, el exoesqueleto cuenta con comunicación cableada e inalámbrica. Esta propiedad abre la posibilidad de realizar estudios combinados, como por ejemplo, con interfaces neuronales. Estos estudios combinados pueden ser usados para relacionar los aspectos de aprendizaje en la rehabilitación.

La estructura de la red de comunicación cableada es determinista y proporciona datos en tiempo real. Está basada en la tecnología Controller Area Network (CAN) que funciona a 1Mbps. La arquitectura inalámbrica permite la comunicación vía Bluetooth y vía WiFi mediante mensajes User Datagram Protocol (UDP) [13].

1.4. Requisitos del sistema

En esta Sección se detallan las características generales y los requisitos del sistema. Todos ellos surgen de las necesidades que se manifiestan en el Hospital Nacional de Parapléjicos de Toledo para monitorizar las pruebas de análisis de la marcha. El objetivo de este Trabajo de Fin de Grado es **el diseño e la implementación de un sistema electrónico capaz de monitorizar la marcha en personas con lesión medular**.

1.4.1. Características generales

El sistema a diseñar debe tener unas características específicas con respecto al tamaño, coste y consumo de energía que se detallan a continuación:

- **Compacto y ligero.** El sistema debe ser de dimensiones y peso lo más

reducidas posibles, para no ser una carga para el que lo transporte.

- **Reconfigurable.** Debe existir la posibilidad de cambiar la configuración del sistema de una manera sencilla e intuitiva. De esta forma, se podrá editar la función principal del programa del sistema.
- **Autonomía.** Es necesario que el sistema sea capaz de funcionar durante al menos una hora continua para poder llevar a cabo una sesión de pruebas completa sin necesidad de recargar la batería. Para ello, se intentará reducir el consumo medio de energía y se usará una batería que nos permita cumplir con este requisito.
- **Robusto.** Resistente a caídas, golpes y vibraciones.
- **Bajo coste.** Se intentará reducir los gastos en la fabricación del prototipo.

1.4.2. Requisitos

En esta Sección, se enumeran ciertos elementos con los que debe contar el sistema electrónico para cumplir con las necesidades mencionadas por el hospital.

- **Módulo transceptor CAN.** La principal función del sistema será recibir y decodificar los mensajes CAN enviados por el exoesqueleto a 1 Mbps. Además, debe ser capaz de proporcionar diferentes parámetros de configuración al exoesqueleto.
- **Entradas digitales.** El sistema debe incluir al menos 10 entradas digitales. De esta manera, se pueden conectar varios dispositivos de forma conjunta, como sensores digitales o neuroprótesis. También se podrían usar como señales de disparo para otros dispositivos.
- **Entradas analógicas.** También deberá contar con un mínimo de 4 entradas analógicas cuyos valores variarán entre 0 V y 5 V. Al igual que en el caso de las entradas digitales, también se podrán conectar varios dispositivos con señales analógicas y así recopilar la información de todos en un solo sistema.

Adicionalmente, se considera la posibilidad de incorporar un segundo transceptor CAN para futuras aplicaciones que incluiría un sistema sensorial paralelo.

1.4.3. Plataforma de desarrollo

Para llevar a cabo la elaboración del sistema electrónico, deberemos seleccionar una plataforma de desarrollo de las existentes en el mercado que se adecue a las características y requisitos anteriormente descritos.

Tras contemplar diferentes plataformas de Arduino y Raspberry, se ha elegido el sistema **Raspberry Pi 3 B+**, debido al previo contacto con él y a su fácil programación y uso.

Raspberry Pi 3 B+ es un ordenador de bajo coste y tamaño reducido que se puede conectar a un monitor y manejarse mediante un ratón y teclado estándar [14]. Cuenta con un procesador BCM2837B0 Cortex-A53 [15], de la familia Broadcom,

a 1.4 GHz; una memoria RAM de 1GB, un módulo Wireless LAN que cumple las especificaciones definidas por IEEE 802.11.b/g/n/ac y un módulo Bluetooth 4.2. La Raspberry Pi soporta las distribuciones propias de la arquitectura ARM como son Raspbian, Ubuntu MATE, Sarpi y RISC OS Pi entre otros. Su precio es de aproximadamente 35€, según el proveedor.

Figura 1.5: Imagen frontal de la Raspberry Pi 3 B+ [16].

Procesador	BCM2837B0 quad-core A53 (ARMv8), 1.4GHz
SDRAM	1GB LPDDR2
GPU	Broadcom Videocore-IV
Interfaces soportadas	Ethernet 4 puertos USB 2.0 SPI SDIO UART I2C I2S Bluetooth 4.1 (BLE)
Almacenamiento integrado	Micro SDHC y USB Boot Mode
Salida de video	HDMI, Composite video, DSI
Salida de audio	Jack 3.5 mm y HDMI
Alimentación	Micro USB 5V, 2.1 A
Dimensiones	85.6mm x 56.5 mm x 17 mm
Temperatura	0 a 70 ºC

Tabla 1.1: Especificaciones de la Raspberry Pi 3 B+.

1.4.3.1. Configuración de la tarjeta de desarrollo.

Para llevar a cabo el desarrollo del software necesario, antes instalaremos una distribución basada en Debian, Raspbian [17], debido a que es un sistema muy estable y con un gran soporte de su comunidad de usuarios. Este sistema operativo cuenta con un kernel de linux versión 4.14. El entorno que se va a emplear para la realización del código y las librerías es C/C++ y se compilará mediante gcc-arm-linux-gnueabihf 6.3.0 [18].

Tras la configuración del sistema operativo, procederemos a instalar las librerías necesarias para el desarrollo software del programa.

Como se ve en la Figura 1.6, la placa cuenta con 40 pines de entrada y salida digitales (GPIO). Todas ellas son de nivel lógico alto a 5V y bajo a 0V.

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figura 1.6: Pines de entrada y salida de la Raspberry Pi 3 B+ [19].

Para la utilización de los pines digitales se ha decidido usar la librería WiringPi. WiringPi es una librería basada en lenguaje C y es compatible con el BCM2835, BCM2836 y, en particular, con el BCM2837, procesador de la Raspberry Pi 3 B+ . Se ha elegido utilizar esta librería ya que permite una configuración muy sencilla de los GPIO.

Por otro lado, deberemos tener en cuenta que el sistema solo tiene entradas y salidas digitales y, por tanto, se deberá buscar un conversor analógico digital (ADC) que permita tomar muestras de valores de tensión analógicos.

1.4.4. Protocolo CAN

El Controller Area Network (CAN), es una estructura de red basada en un bus serie no síncrono, que está optimizada para tasas binarias bajas o medias. Además, asegura una comunicación fiable, robusta y segura [20]. Desde su origen, se han publicado dos estándares del bus CAN :

- Standard ISO11519: esta versión proporciona una baja tasa binaria (125 kb/s) y permite la conexión de un bus de hasta 1000 m en cable UTP (Unshielded Twisted Pair) [21, 22].
- Standard ISO11898: la tasa binaria es notablemente más alta (1 Mb/s) y permite la conexión de un bus de hasta 40 m en cable UTP o STP (Shielded Twisted Pair) [23].

La estructura de la capa física del protocolo CAN está formada por dos hilos denominados CAN-HI y CAN-LOW terminados en cada extremo por resistencias de 120 ohmios [20]. Estos dos hilos conforman una señal diferencial en la que irá codificada la información. Cada uno de los nodos del bus utilizará un conector D macho de 9 pines [24].

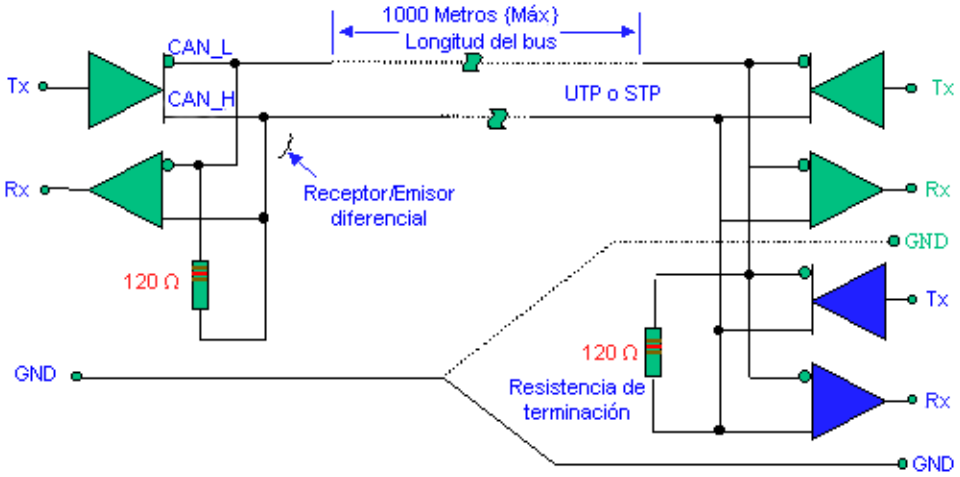


Figura 1.7: Interfaz eléctrica del Bus CAN [24].

El estándar actual del bus CAN define dos posibles formatos de trama:

- **Trama estándar.**

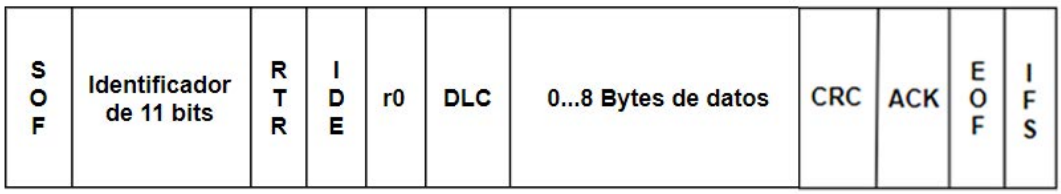


Figura 1.8: Trama estándar del protocolo CAN [25].

Una trama estándar consta de las siguientes partes:

SOF - Inicio de la trama. Un solo bit dominante marca el inicio de un nuevo mensaje.

Identificador - Establece la prioridad del mensaje. Cuanto menor sea su identificador, mayor es la prioridad.

RTR - Petición de transmisión remota. Es un bit dominante en caso de reclamar información a otro nodo.

IDE - Extensión de identificador. Bit que indica la transmisión de una trama estándar.

r0 - Bit reservado.

DLC - Código de longitud de datos. Cuatro bits que contienen el número de bytes enviados en el mensaje.

Datos - Se pueden enviar hasta 64 bits.

CRC - Confirmación cíclica redundante. Dieciséis bits que contienen la suma de control (*checksum*) para detección de errores.

ACK - Confirmación de recepción. Los nodos que reciben este mensaje sobreescriben un bit dominante en este campo, indicando que se ha enviado un mensaje correcto.

EOF - Fin de trama. 7 bits que indican el final de la trama.

IFS - Espacio entre tramas. Es necesario dejar un tiempo equivalente a 7 bits de separación entre una trama y la siguiente.

- **Trama extendida.**

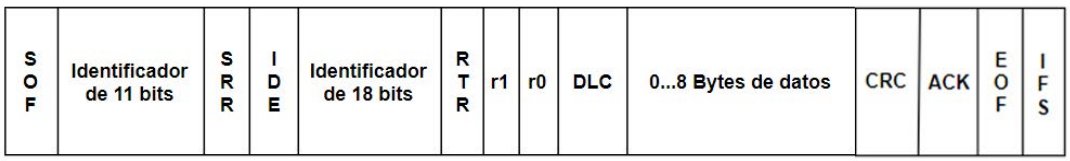


Figura 1.9: Trama extendida del protocolo CAN [25].

Como se muestra en la Figura 1.9, la trama extendida del protocolo CAN es igual que la trama estándar con la suma de:

SRR - Petición remota de sustituto. Sustituye al bit de RTR en la trama estándar.

IDE - Extensión de identificador. Un bit recesivo en este campo indica que viene seguido de 18 bits más de identificador.

r1 - Posición reservada. Se añade un bit reservado antes del bit DLC.

Arbitraje

Una de las principales características del bus CAN es el estado lógico opuesto entre el bus, la señal de entrada al transmisor y la señal de salida del receptor. Por

lo general, el nivel lógico alto es asociado con un uno, y el nivelo lógico bajo con un cero; pero no es así en el caso del bus CAN.

En este caso, se consideran los bits de valor uno como bits recesivos, y los ceros, como bits dominantes. De esta manera, siempre que haya un uno a la entrada, el bus entra en un estado recesivo en el que todas las entradas y salidas están al mismo nivel.

En el momento que dos nodos CAN intentan ocupar el bus simultáneamente, el acceso está controlado mediante un arbitraje no destructivo bit a bit. En este caso, no destructivo quiere decir que el nodo ganador del bus continúa la transmisión de su mensaje sin que sea corrompido. La prioridad de los mensajes la define el identificador, siendo el más bajo el de mayor prioridad. Un identificador constituido de ceros únicamente será el mensaje de mayor prioridad. Por lo tanto, si dos nodos comienzan a transmitir por el bus al mismo tiempo, en el momento que un nodo envíe un 0 (dominante) y el otro un 1 (recesivo) en el identificador, el que ha enviado el 1 libera la linea para que el otro nodo termine la transmisión [25].

Tipos de mensaje

- **Trama de datos.** Este tipo de mensaje es el más común en la comunicación CAN. Se caracteriza principalmente por tener un bit dominante en el campo RTR, que va seguido de 0 a 8 bytes de datos.
- **Trama remota.** Esta trama solicita la transmisión de información a otro nodo. Es similar a la trama de datos excepto porque el campo RTR contiene un bit recesivo y porque no hay campo de datos.
- **Trama de error.** Cuando un nodo detecta que ha habido un error en el bus, envía una trama de error que viola las normas del formato de trama CAN, que provocará que el resto de nodos envíen al bus otra trama de error. El nodo original responderá con una última trama de error.
- **Trama de sobrecarga.** Es igual que la trama de error y es enviado por un nodo que no es capaz de procesar la información recibida en el tiempo demandado. Envía esta trama para conseguir más tiempo entre tramas.

Capítulo 2

Diseño del sistema

En este Capítulo se expondrán todos procesos de diseño llevados a cabo para la elaboración del sistema. En primer lugar, se han estudiado los medios para la obtención de datos como son el protocolo CAN y la cuantificación de valores analógicos. Tras ello, se ha elaborado un programa que obtiene todos los datos y los almacena en el directorio elegido. Por último, se ha hecho un estudio de los tiempos de ejecución del programa y se han llevado a cabo una serie de mejoras del sistema respecto al acceso al sistema.

2.1. Protocolo CAN en Raspberry Pi

En la implementación del protocolo CAN, necesitaremos una placa que nos proporcione dos canales de CAN-Bus como se especifica en la Sección 1.4.2, debido a que la Raspberry Pi no cuenta con un módulo CAN integrado. Para ello, se ha elegido la placa **PiCAN2 Duo CAN-Bus board** para Raspberry Pi 3. La decisión de utilizar esta tarjeta se ha basado en la fácil comprensión de las librerías proporcionadas por el fabricante para su uso. La placa usa el microchip MCP2515 controlador CAN y el MCP2551 transceptor CAN [26].

Su conexión con la Raspberry Pi consistirá en un bus SPI (Serial Peripheral Interface) a 10 MHz. El funcionamiento de esta comunicación se explicará más adelante en la Sección 2.1.1.

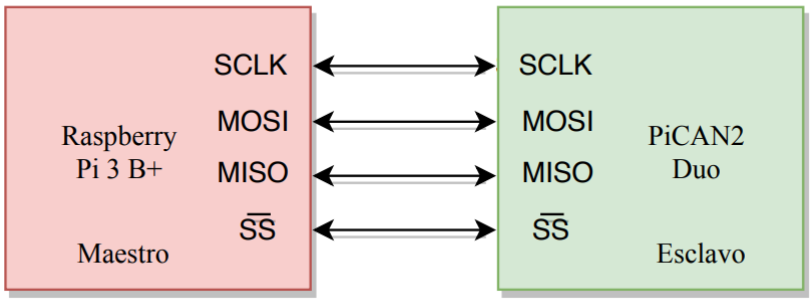


Figura 2.1: Esquema de la comunicación SPI entre Raspberry Pi y PiCAN2 Duo.

En la Figura 2.1, vemos un esquema de las líneas utilizadas para la conexión entre la Raspberry Pi y el PiCAN2 Duo, donde SCLK es el reloj (GPIO11), MOSI (GPIO10) y MISO (GPIO9) son las líneas de datos y SS (GPIO8) es el chip select activo a nivel bajo. Además de los puertos anteriores, PiCAN2 Duo utiliza los GPIO 24 y 25 para las interrupciones de transmisión de datos [26, 27].

Antes de comenzar la transmisión de datos, es necesaria la configuración de los dos transceptores con los que cuenta la placa. Para ello, debemos acceder al archivo de configuración de arranque del sistema operativo, habilitando la comunicación SPI y la interrupción de transmisión en los puertos 24 y 25.

```
$sudo nano /boot/config.txt

dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835-overlay
```

Además, siempre que se arranque el sistema, se debe configurar la velocidad a la que se producirá la transmisión de datos con los siguientes comandos:

```
$ sudo ip link set can0 type can bitrate 1000000
$ sudo ip link set up can0
```

Donde *can0* es el nombre de la interfaz que queremos levantar y *1000000* la velocidad deseada en bits/segundo.

Para el uso de periféricos CAN, Linux proporciona una librería llamada *can-utils* que cuenta con las siguientes órdenes principales:

- *Candump*: vuelca la información de los paquetes recibidos y la guarda en un archivo si es requerido.
- *Cansend*: envía el mensaje deseado por la interfaz CAN indicada. En caso de no cumplir los requisitos de formato, emitirá un error.
- *Cangen*: genera automáticamente tramas de longitud y datos aleatorios y las envía por la interfaz CAN indicada.
- *Canplayer*: reproduce el archivo guardado por *Candump*.
- *Cansniffer*: selecciona los paquetes con un identificador específico de entre todos los mensajes recibidos por la interfaz.

En el programa principal de nuestro sistema electrónico, únicamente se utilizarán *Candump* para la recepción de mensajes y *Cansend* para la posible configuración del exoesqueleto. El resto de las órdenes también serán utilizadas, pero sólo para la evaluación del comportamiento del sistema.

2.1.1. SPI - Serial Peripheral Interface

El bus SPI es un estándar de comunicaciones establecido por Motorola que consiste en un enlace serie síncrono que opera en modo dúplex [27]. Los dispositivos se conectan

usando una relación maestro/esclavo, en la que un maestro puede tener varios esclavos en su red. Es también el maestro el que decide con qué esclavo comunicarse y el que genera una señal de reloj por la que se regirá la comunicación. A partir de ese momento, se transmite información en ambas direcciones simultáneamente. Si el maestro no desea recibir información, debe descartar el byte recibido en una trama de sólo recepción.

SPI establece su comunicación mediante 4 hilos diferentes:

- **MISO** - Entrada al maestro, salida del esclavo.
- **MOSI** - Salida del maestro, entrada al esclavo.
- **SCLK** - Reloj impuesto por el maestro que regirá la comunicación. Puede tener diferentes valores según la velocidad máxima de cada esclavo.
- **SS** - Chip select. Según su valor, el esclavo escucha el mensaje que se ha enviado o no. Esta señal es activa a nivel bajo.

Existen diferentes configuraciones para una comunicación SPI. La más frecuente es la consistente en un maestro y un esclavo, pero un maestro puede controlar a la vez varios esclavos. También, existe la posibilidad de que un esclavo esté bajo el control de dos maestros diferentes.

2.2. Entradas analógicas

En esta Sección se llevará a cabo la elección de un ADC, mediante el cuál podremos cuantificar el valor de la tensión de las entradas analógicas.

2.2.1. Elección de ADC

Como se comenta en la Sección 1.4.3.1, la Raspberry Pi 3 B+ no cuenta con entradas analógicas, por lo que es necesario la implementación y montaje de un ADC.

Tras la búsqueda en diferentes proveedores, se han seleccionado 2 posibilidades para su revisión.

2.2.1.1. ADS1115-ADC para Raspberry Pi 3 B+

Este convertidor se caracteriza por contar con 4 entradas independientes que pueden utilizarse para implementar la recepción de 2 señales diferenciales. La señal de entrada puede tomar valores entre 0 y 5 voltios, y se cuantificará con 16 bits; por tanto, la precisión del valor obtenido sería del orden de centenas de microvoltios. Su comunicación con la Raspberry Pi sería a través de la interfaz serie I2C. También es interesante remarcar que es apilable en la Raspberry Pi, pero no deja un fácil acceso a las entradas y salidas.

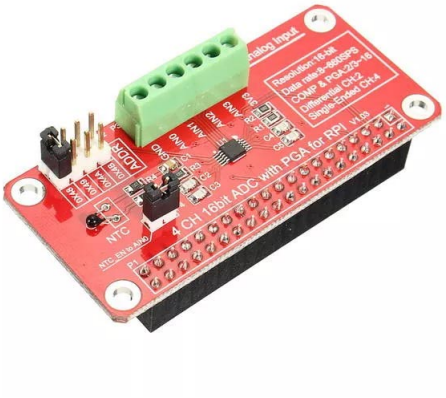


Figura 2.2: Módulo ADS1115-AD para Raspberry Pi 3 B+ [28].

2.2.1.2. ABelectronics ADC Pi

En este caso, el convertidor cuenta con 8 entradas independientes, cada una con su punto de referencia a masa. También puede tomar valores entre 0 y 5 voltios, y se puede elegir el número de bits de cuantificación siendo 17 bits lo máximo, por lo que la precisión máxima es del orden de decenas de microvoltios. Al igual que el convertidor anterior, su comunicación con la Raspberry Pi es a través de la interfaz I2C. Además, es apilable en la Raspberry Pi, y da acceso a las entradas y salidas. Por otro lado, además del convertidor, la placa cuenta con un amplificador de ganancia programable (PGA).

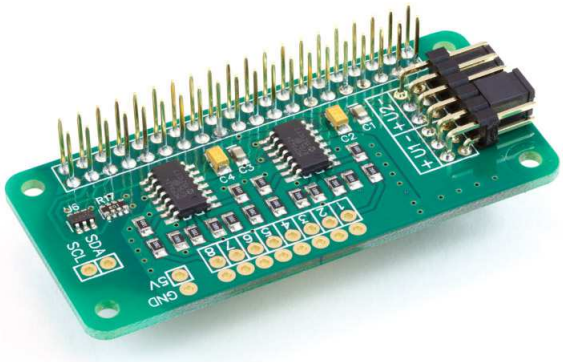


Figura 2.3: ABelectronics ADC Pi [29].

Por tanto, se puede concluir que, debido a mayores prestaciones en caso de querer ampliar las capacidades del sistema, la mejor opción para el proyecto es la placa ADC Pi de ABelectronics. En esta decisión, ha sido diferencial la posibilidad de acceder cómodamente a las entradas y salidas.

2.2.2. I2C - Inter-Integrated Circuit

El bus I2C es un estándar de comunicaciones establecido por Philips Semiconductor, conocida hoy como NXP Semiconductor, que consiste en un enlace serie síncrono, multi-maestro, multi-esclavo con terminación en colector abierto. La comunicación consiste en la conexión de diferentes nodos mediante 2 hilos: SDA y SCL.

- **SDA** - Línea de datos.
- **SCL** - Línea de reloj impuesto por el master de la comunicación.

El enlace es bidireccional y, por tanto, existen colisiones. Es por esto, que en I2C hay un arbitraje CSMA/CD. Cada nodo tendrá una dirección asignada y será referenciado por ella a la hora de enviar los mensajes a través del bus.

2.2.2.1. Arbitraje

El arbitraje en I2C se produce bit a bit. Cuando un maestro procede a enviar un mensaje, debe comprobar antes si el bus está libre. Para ello, durante cada bit, mientras SCL está a nivel alto, cada maestro comprueba si el valor de la línea es el mismo que el que ha enviado. Existe la posibilidad de que dos maestros completen una transacción sin errores, siempre y cuando, las transmisiones fueran idénticas [30].

En el momento que un maestro intenta poner poner SDA a nivel alto, pero detecta que sigue a nivel bajo, sabe que ha perdido el arbitraje e interrumpe la transmisión. Es importante remarcar que, en este proceso, no se pierde información.

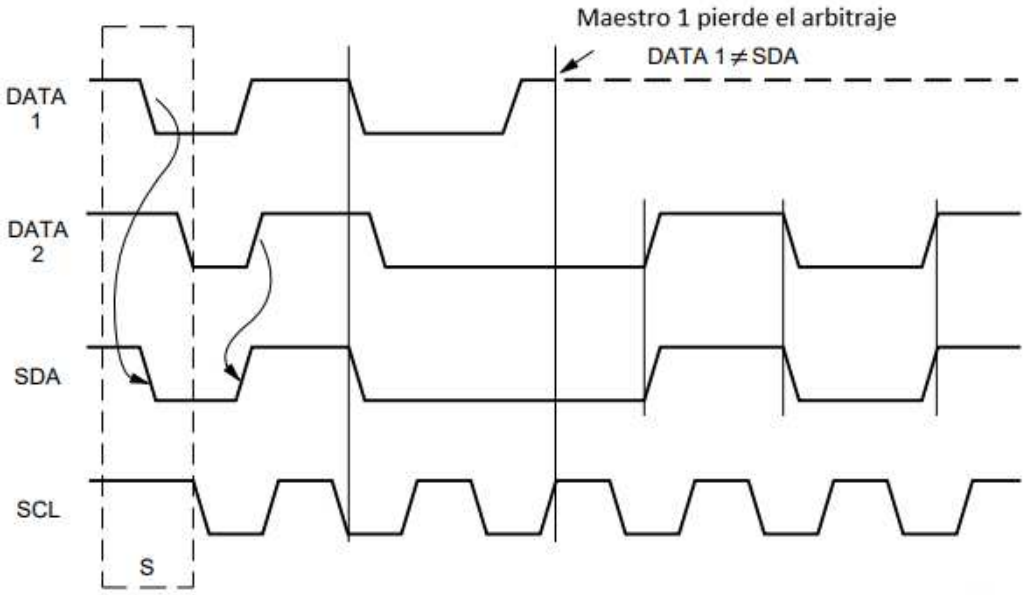


Figura 2.4: Proceso de arbitraje entre dos maestros [31].

2.3. Programa principal

En esta Sección se describe el desarrollo del código de este Trabajo de Fin de Grado. En él se consideran 5 grandes bloques: el receptor CAN, el emisor CAN, el muestreo de las entradas digitales y analógicas y la rutina de atención al teclado.

Los 5 bloques serán desarrollados por separado para analizar su funcionamiento y tras ello, se irán integrando en un programa principal basado en el uso de threads y temporizadores para el control del tiempo de ejecución.

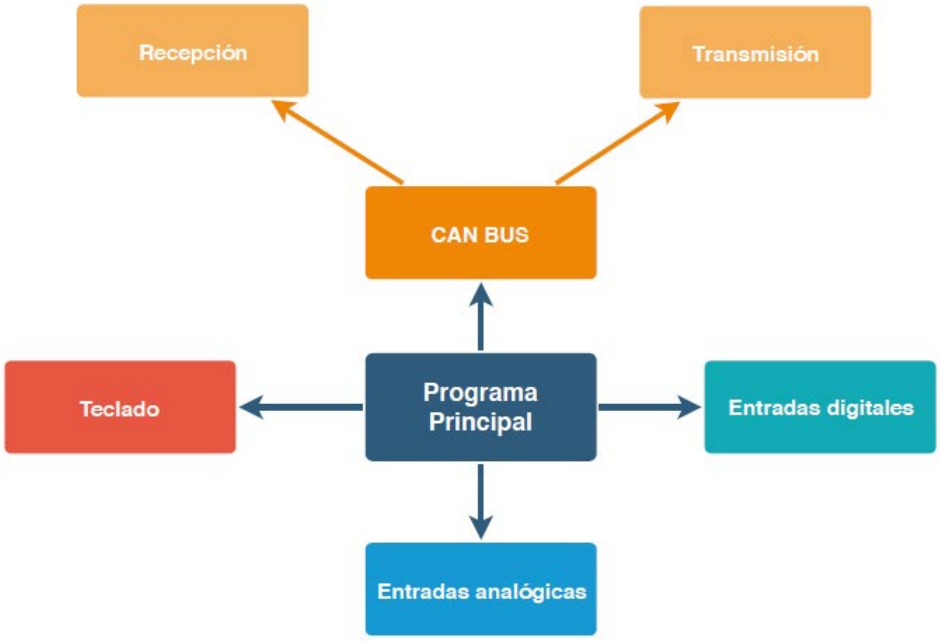


Figura 2.5: Diagrama de bloques del programa principal.

2.3.1. Recepción CAN - Candump

Para la implementación de esta parte del código, partimos de la librería linux-can-utils para lenguaje C que podemos encontrar en el repositorio *linux-can* en GitHub. De todos los programas de la librería, utilizaremos *Candump*.

Candump cuenta con diferentes opciones de configuración, entre ellas la posibilidad de guardar todos los mensajes recibidos en un archivo creado automáticamente, de transformar el mensaje hexadecimal a binario o de aplicar una máscara a los mensajes del bus y únicamente mostrar los que pasan el filtro. Además, cuenta con una opción que configura el transceptor CAN para que haga de puente entre dos buses diferentes. Un ejemplo del comando que invoca *Candump* es:

```
$ candump can0 -l
```

Donde se ha habilitado la opción de *log*.

Para este programa reutilizaremos las líneas de código correspondientes a la apertura de un socket para la recepción CAN, y el sistema de almacenamiento

automático de los datos; aunque este último, se modificará. Antes de ejecutarse, el programa está continuamente comprobando el estado de un semáforo que le indica si debe continuar con la ejecución o esperar sin hacer nada, como se puede ver en la Figura 2.6. En caso de continuar, comienza leyendo de los parámetros, la interfaz CAN por la que queremos recibir los paquetes. Ya que tenemos dos disponibles, can0 y can1, configuraremos can0 por defecto, que es el elegido para la conexión con el exoesqueleto. Tras la configuración de la interfaz, se abre un socket en esta de carácter no bloqueante. El carácter no bloqueante hace que, mientras no reciba algún paquete de datos por la línea, libere todos los recursos para que el resto de hebras del programa puedan seguir ejecutándose.

Una vez abierto el socket, como se ha comentado anteriormente, se guardarán todos los paquetes recibidos en un archivo de texto en formato CSV (Comma Separated Values). Además, según la documentación del exoesqueleto H2, sabemos que los mensajes recibidos contendrán únicamente 4 posibles identificadores (110,120,130 y 140); por tanto, también crearemos 4 archivos diferentes en los que guardaremos los paquetes según su identificador. El nombre de estos archivos será *candump* seguido de la fecha y hora del inicio de ejecución del programa. El formato de estos archivos se definirá más adelante.

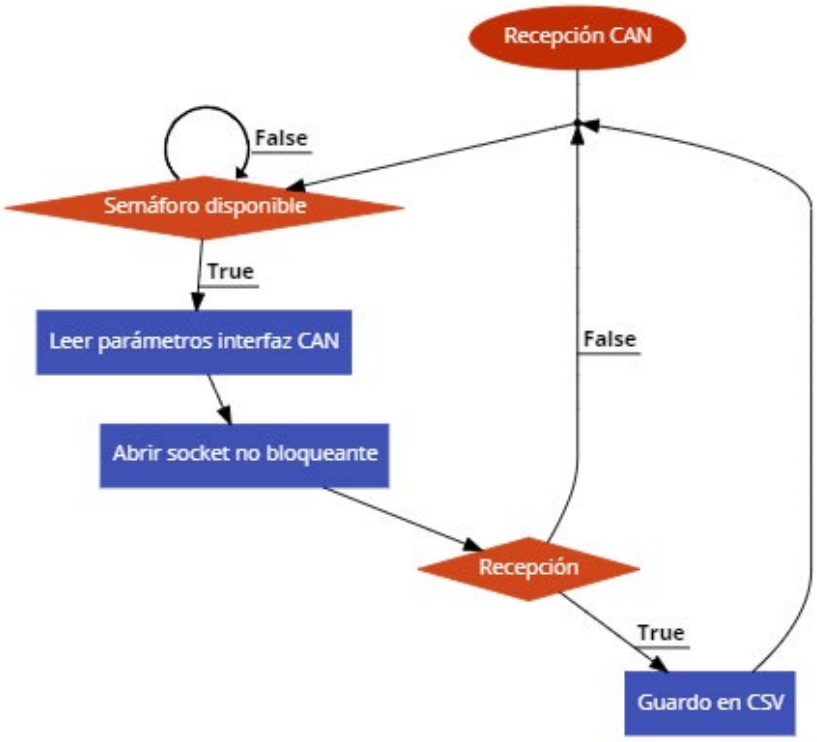


Figura 2.6: Diagrama de flujo del bloque de recepción CAN

2.3.2. Transmisión CAN - Cansend

Para la transmisión de mensajes CAN, la librería linux-can-utils nos proporciona el programa *Cansend*. La única misión de este bloque es la de enviar por el bus el mensaje indicado.

El programa lee los parámetros de entrada entre los que se encuentra la interfaz por la que se desea enviar el mensaje (can0 o can1 en este caso) y el mensaje en el sistema de numeración hexadecimal siguiendo el siguiente formato:

```
$ cansend can0 123#112233445566
```

Siendo '123' el identificador con el que se desea enviar el mensaje, '112233445566' el mensaje y '#' el separador entre ellos.

Este programa estará controlado por las normas de arbitrio definidas en la Sección 1.4.4. Por tanto, si encuentra que el bus está siendo utilizado por otro nodo CAN de un identificador menor que el suyo, esperará a que este termine su transmisión para poder empezar con la suya.

La utilidad de este bloque del programa se divide en dos: la configuración inicial del exoesqueleto según unos parámetros de entrada y la depuración de errores y la validación del sistema de recepción CAN.

- **Configuración inicial** - Se creará un archivo de texto (*reconfig.txt*) en el que se especifique, en primer lugar, con un valor lógico si se debe reconfigurar el exoesqueleto. En caso afirmativo, se especificarán los valores de todos los parámetros de configuración, que estarán en formato decimal y tendrán un valor entre 0 y 255. El formato de este fichero de texto será **OPCIÓN= VALOR**. En el momento que se arranque el programa, se leerán todos los valores y se enviarán los respectivos mensajes de reconfiguración. Se incluye un ejemplo del archivo *reconfig.txt* en el Anexo B.

- **Validación de *Candump*** - Se habilitará la opción de enviar mensajes mediante la pulsación de ciertas teclas del teclado. Estas teclas se especificarán en el apartado de lectura del teclado. Se destinarán 4 teclas, una para cada identificador que *Candump* puede interpretar.

2.3.3. Muestreo de señales digitales

La misión principal de este bloque será la de leer el valor lógico de las entradas digitales que se habilitarán como tal y de guardar dichos valores en un fichero externo en formato CSV.

Antes de realizar la lectura periódica de las entradas, debemos designar cuáles de los pines físicos con los que cuenta la Raspberry Pi serán los que utilicemos. Según las especificaciones, necesitamos un máximo de 10 entradas digitales, por lo tanto elegiremos los GPIO 18, 27, 22, 23, 5, 6, 13, 16, 20 y 21, dado a que están libres y son de fácil acceso.

Para la configuración de estos pines como entradas, debemos antes llamar a la función *wiringPiSetupGpio()* que será la que permita utilizar todos los métodos de configuración, lectura y escritura sobre estos pines. Tras ello, llamaremos a la función

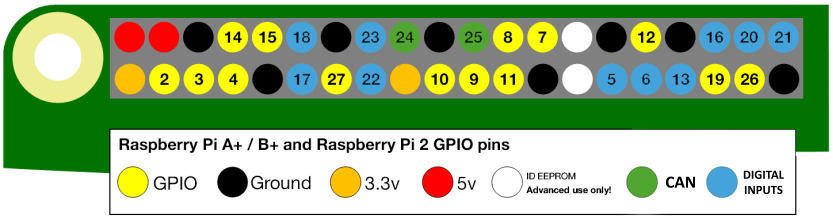


Figura 2.7: Pines utilizados para el muestreo de señales digitales [32].

pinMode(int pin, int mode) para cada uno de ellos con el valor de INPUT(0) en el modo.

Tras la configuración, procederemos al muestreo de las entradas. Al igual que en el caso de *Candump*, existirá un semáforo que le indicará a la tarea si debe ejecutarse o esperar. Debido a que la escritura en un fichero en Raspbian es una tarea que nos limita en tiempo, guardaremos los valores en un array de 10 veces el número de entradas. De esta forma, tomaremos 10 muestras de cada entrada cada un determinado tiempo y cuando se complete el array, imprimiremos todos los valores en el fichero con su correspondiente formato.

Para mejorar la eficiencia de este bloque, se ha decidido parametrizar el número de entradas. Este valor deberá configurarse en el documento de reconfiguración del apartado anterior (*reconfig.txt*) siendo 1 el valor mínimo y 10 el máximo. El apartado correspondiente a este valor será NUM_DIGITALS= XX.

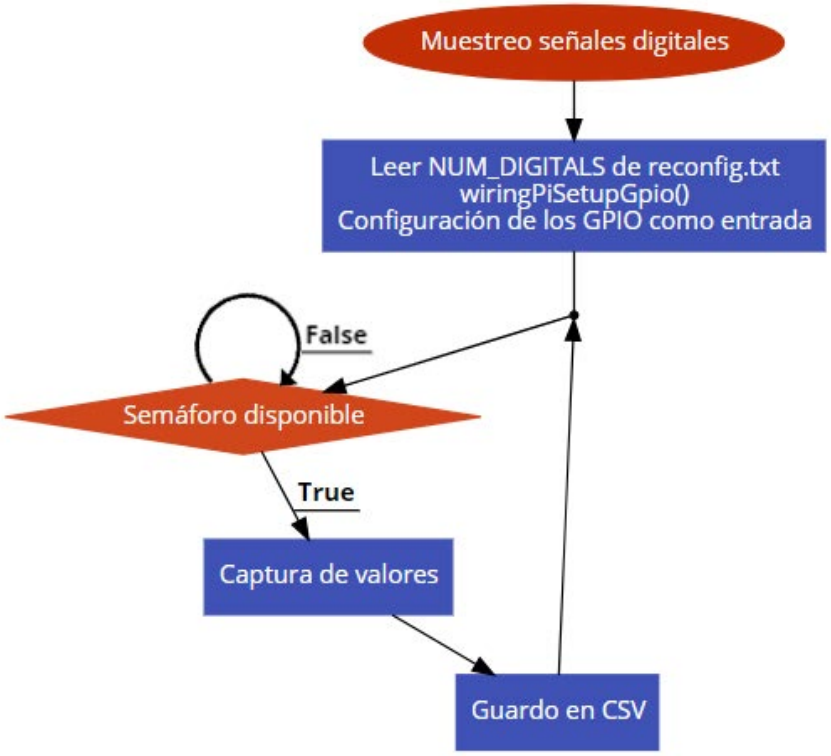


Figura 2.8: Diagrama de flujo del bloque de la captura de entradas digitales.

2.3.4. Muestreo de señales analógicas

En este bloque, leeremos los valores de las entradas analógicas del ADC de ABelectronics. Para ello, el fabricante nos proporciona una librería que controla el bus I2C y se comunica con el ADC. Esta librería se llama ADCPi es accesible desde el perfil de GitHub del fabricante (abelectronicsuk).

El ADC separa en dos grupos de 4 las 8 entradas analógicas. La placa cuenta con 12 pines en los que se selecciona la dirección I2C de cada uno de los dos grupos de entradas. Para una lectura correcta del ADC, el fabricante recomienda varias configuraciones de las direcciones de la placa para la comunicación vía I2C.

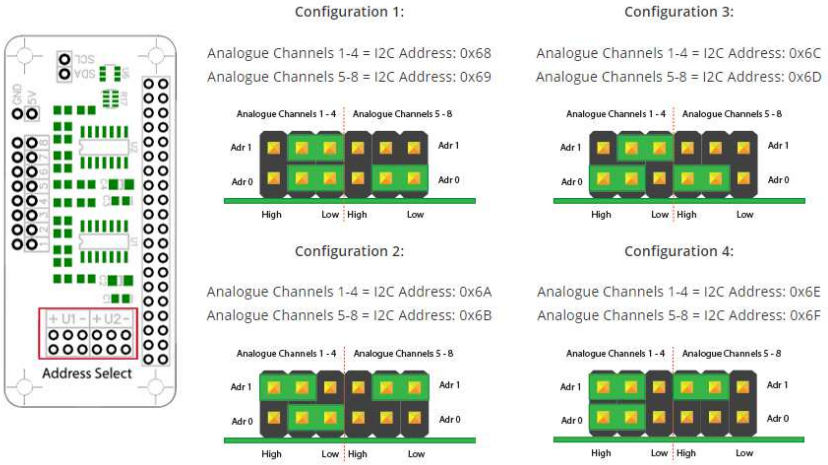


Figura 2.9: Configuraciones de las direcciones I2C del ADC ABelectronics [33].

Para la realización del sistema, se ha decidido utilizar la configuración 1. Por tanto, la dirección correspondiente a los canales del 1 al 4 es 0x68 y a los canales del 5 al 8 es 0x69. Estos valores son importantes a la hora de utilizar la librería en C del fabricante [33].

El método utilizado para leer los valores analógicos es

double read_voltage(char address, char channel, int bitrate, int pga, char conversionmode)

- *Address* es el valor de la dirección que queremos leer en hexadecimal y con el prefijo 0x.
- *Channel* es el número del canal que queremos leer. Puede tomar valores desde el 1 hasta el 4.
- *Bitrate* es el número de bits con el que queremos cuantificar el valor, eligiendo un valor entre 12, 14, 16 ó 17.
- *Pga* es el valor del amplificador nombrado en la Sección 2.3.4, *conversionmode* es el modo de conversión; que tomará el valor 0 para conversión de un disparo y 1 para la continua.

Por las mismas razones que en el caso de las entradas digitales, los valores se guardarán en un array de 10 veces el número de entradas analógicas y cuando se llene, se imprimirán dichos valores en su fichero de salida con su respectivo formato. Además, cuenta con el mismo semáforo que las entradas digitales.

De nuevo, debido a la necesidad de mejorar la eficiencia de este bloque, se ha parametrizado el número de entradas analógicas. Análogamente al apartado anterior, existirán un valor en el fichero de reconfiguración (*reconfig.txt*) siendo 1 el mínimo y 8 el máximo. El apartado correspondiente será NUM_ANALOGS= X.

2.3.5. Rutina de atención al teclado

Para la gestión de todas las tareas, es decir, la activación o desactivación de todas ellas, se ha introducido una rutina que actuará frente a todos los eventos del teclado. Esto se ha hecho ya que se espera un acceso remoto a la Raspberry Pi desde un ordenador con teclado.

La librería que nos permitirá llevar a cabo esta función es *conio* [34]. *Conio* cuenta con un método que comprueba si se ha producido un evento en el teclado. Este método se llama *int kbhit*(). Si se invoca *kbhit*() desde un bucle permanente, podremos atender continuamente los eventos que se produzcan en el teclado. Este bucle no podrá bloquear la ejecución del resto de tareas, ya que dependerá del mismo semáforo que estas.

Una vez se produce la pulsación de una tecla, se diferenciarán los siguientes casos:

- **Q** - Botón de *quit*. Una vez se pulsa, se terminan todos los procesos en ejecución y se cierran todos los archivos sobre los que se está escribiendo.
- **C** - Botón de *capture*. Este botón activa los semáforos de todas las tareas y las pone en ejecución.
- **S** - Botón de *stop*. Se desactivan los semáforos y se pausan todas las tareas.

2.3.6. Sistema de hebras

En las secciones anteriores se ha explicado cómo se comportarán cada una de las partes del programa principal por separado. Para la ejecución conjunta de todas las tareas, se ha decidido utilizar un sistema de hebras (threads) que ejecutarán de forma paralela cada uno de los métodos.

Para ello, se ha hecho uso de la librería *pthread* [35] que permite la creación y ejecución de un thread con los métodos *int pthread_create*() al que se le pasa como parámetro el objeto thread y el método a ejecutar y *int pthread_join*() que ejecuta la hebra.

Como las tareas de muestreo y *Candump* acceden a las mismas variables (los semáforos), deberemos establecer zonas de exclusión mutua (*mutex*), para que una hebra no escriba en una variable mientras otra la está leyendo. Para ello utilizaremos los métodos *int pthread_mutex_lock*(*pthread_mutex_t**) e *int pthread_mutex_unlock*(*pthread_mutex_t**). Cuando una hebra invoca *pthread_mutex_lock*() sobre una zona, la bloquea de manera que cualquier otra hebra que intente acceder a ella, se quede esperando hasta que se libere mediante *pthread_mutex_unlock*().

2.4. Temporización

El objetivo de esta Sección es el de realizar las tareas expuestas en la Sección 2.3, llevando a cabo un control del momento en el que se ejecutan. Para ello, antes debemos conocer cuáles son los tiempos de ejecución máximos de cada una de las tareas.

Se ha realizado un programa para cada una de las tareas por separado. Tomando el momento en el que se empieza a ejecutar la tarea y el momento en el que termina, podemos calcular, con precisión del orden de microsegundos, el tiempo de ejecución. Este programa, va tomando el valor máximo durante la ejecución del mismo; en caso de que un nuevo valor supere el máximo, este pasará a ser el nuevo máximo. Tras varios minutos de ejecución, se termina el programa y se imprime por pantalla el valor máximo.

2.4.1. Análisis de tiempos

Se ha realizado un análisis de los tiempos de ejecución de las tareas del programa principal, en particular, para los 4 valores de cuantificación del ADC, y se han obtenido los siguientes valores.

Número de entradas	**1**	**2**	**3**	**4**	**5**
Entradas digitales	123	142	152	164	177
Entradas analógicas (12)	4952	11984	17905	23698	25728
Entradas analógicas (14)	17488	35913	53475	73323	86371
Entradas analógicas (16)	65003	131591	196544	261654	325710
Entradas analógicas (17)	254882	510869	765798	1021896	1271616

Tabla 2.1: Tiempos máximos de ejecución en μs en función del número de entradas.

Número de entradas	**6**	**7**	**8**	**9**	**10**
Entradas digitales	191	208	240	247	296
Entradas analógicas (12)	35601	42701	49128	-	-
Entradas analógicas (14)	104732	121523	138382	-	-
Entradas analógicas (16)	389877	453769	519199	-	-
Entradas analógicas (17)	1522492	1771366	2024070	-	-

Tabla 2.2: Tiempos máximos de ejecución en μs en función del número de entradas.

2.4.1.1. Candump

Para obtener un valor real del tiempo de ejecución de la recepción de paquetes CAN, se ha conectado el sistema electrónico al exoesqueleto para el que está siendo diseñado.

Según los valores obtenidos, el exoesqueleto manda, de media, 344 mensajes CAN por segundo. Por otro lado, tras una ejecución continua de 10 minutos, el valor máximo de procesado en esta tarea es de 30 milisegundos aproximadamente. Este valor nos reduciría la frecuencia de recepción a 33 muestras por segundo; pero, debido a que se desea recibir el número máximo de datos, mantendremos la frecuencia máxima. Esto quiere decir que se recibirán los paquetes del bus CAN siempre que se pueda, ya que no son necesarias muestras periódicas.

2.4.1.2. Cansend

Debido a que *Cansend* únicamente se utilizará al principio del programa con el fin de reconfigurar el exoesqueleto, no se ha tenido en cuenta para el estudio de temporización.

2.4.1.3. Entradas digitales

Para este caso, no se ha usado un escenario de simulación real, ya que *wiringPi* tarda el mismo tiempo en leer una entrada digital, sea un 0 o un 1.

En la Figura 2.10, se puede ver el tiempo máximo de ejecución de la tarea de muestreo para diferentes números de entradas digitales. Es fácil comprobar que según aumentan las entradas, el tiempo aumenta de manera casi lineal, como era de esperar, ya que también aumenta el número de valores a muestrear y a imprimir. También cabe destacar que en todos los casos, el tiempo mostrado se corresponde con el ciclo de ejecución en el que se imprimen todos los valores en el fichero.

En vista de estos valores, poniendo el número de entradas a 10 como valor fijo, podríamos alcanzar una frecuencia de muestreo de aproximadamente 3300 muestras por segundo. Este valor es muy elevado y no se usará en ningún momento ya que la mayoría de las muestras serían idénticas.

2.4.1.4. Entradas analógicas

Para la simulación de esta tarea, se han decidido hacer 4 experimentos diferentes, uno para cada valor de bits de cuantificación posible (12, 14, 16 y 17 bits). Al igual que en el caso anterior, se tomará el tiempo máximo de ejecución para todos los valores posibles del número de entradas.

En la Figura 2.11, se ve cómo crece el tiempo de ejecución de manera lineal con el número de entradas. Además, se puede contemplar cómo la cuantificación de 17 bits es descartable debido a sus elevados valores, dándonos una frecuencia máxima, con 8 entradas, de 0.5 muestras por segundo.

Por otro lado, debido a que no se necesita gran precisión en el valor mostrado, sino el mayor número de muestras posibles, se ha decidido tomar el valor de 12 bits de cuantificación.

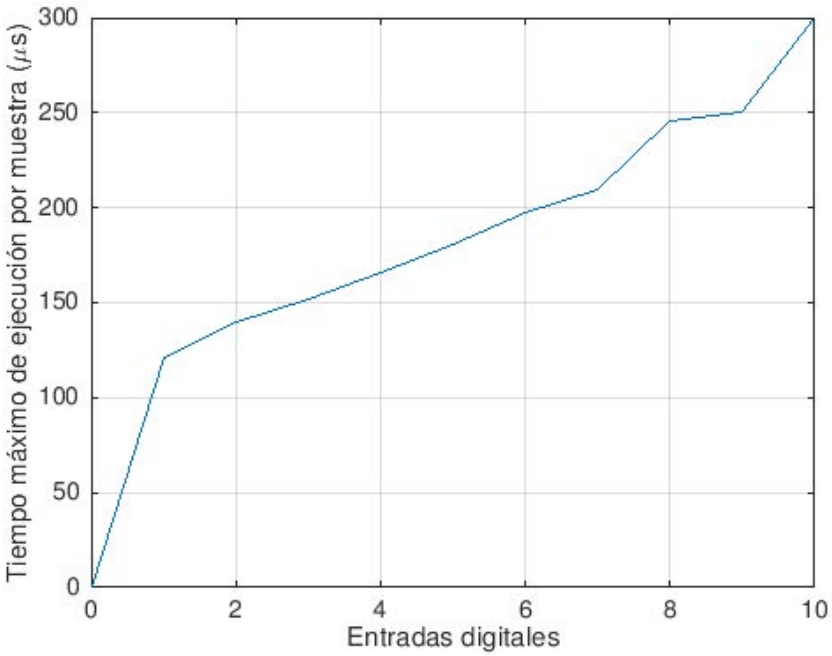


Figura 2.10: Tiempo máximo de ejecución de muestreo en función del número de entradas digitales.

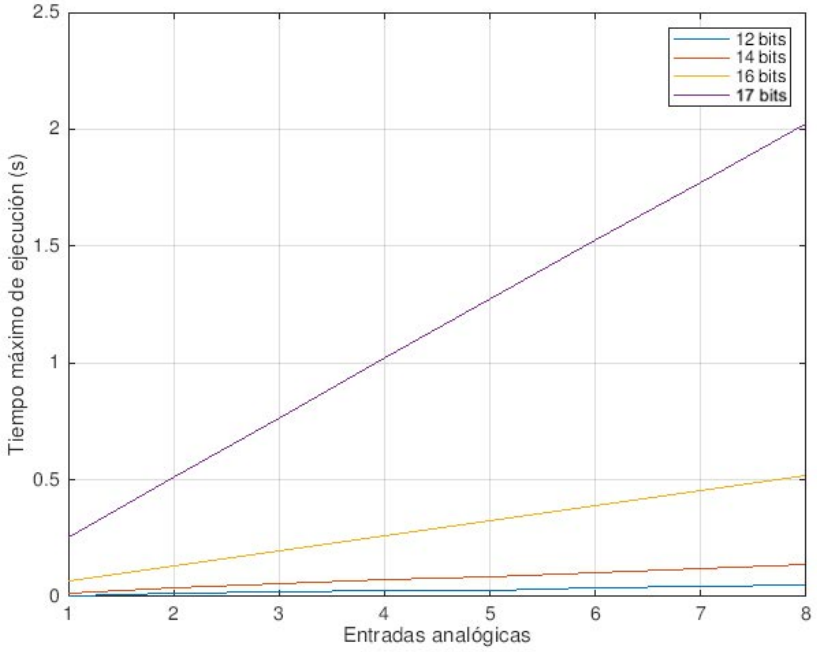


Figura 2.11: Tiempo máximo de ejecución de muestreo en función del número de entradas analógicas y bits de cuantificación.

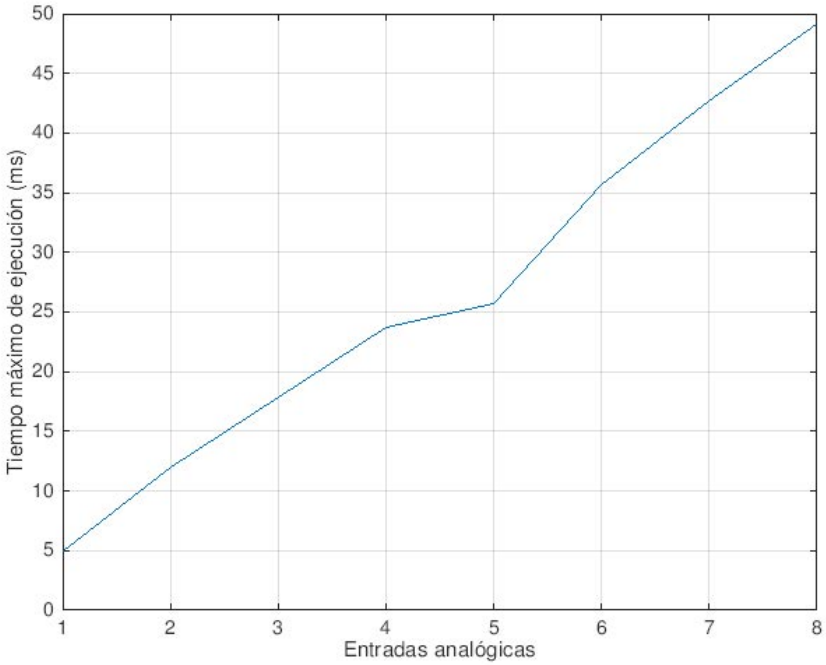


Figura 2.12: Tiempo máximo de ejecución de muestreo en función del número de entradas analógicas para 12 bits de cuantificación.

En este caso específico, en la Figura 2.12 se puede ver cómo crece el tiempo linealmente, excepto en el caso en el que cambia de dirección I2C (de la entrada 4 a la entrada 5).

En vista de que, para el valor máximo del número de entradas, el tiempo es de aproximadamente 50 ms, permitirá muestrear a una frecuencia máxima de 20 muestras por segundo. No obstante, debido a que la intención es utilizar este módulo para un máximo de 4 entradas, podremos multiplicar la frecuencia por 2, dándonos un máximo de 40 muestras por segundo.

2.4.2. Implementación de la temporización en el código

Para implementar la temporización de las tareas principales del código, se hará uso del fichero de texto *reconfig.txt* anteriormente mencionado en esta memoria y de la librería *tmr*, que permite un uso sencillo de los timers POSIX [35].

En el fichero *reconfig.txt*, se definirá un nuevo valor en el que se indicará el número de muestras por segundo que se desea, llamado *SAMPLES_PER_SECOND*. Se leerá este valor, se calculará el periodo de muestreo y se guardará el valor para la configuración del timer.

Se utilizarán los métodos *tmr_new*(*notify_func_t isr*), que crea el objeto timer, donde *isr* es la función que debe realizar; y *void tmr_startus*(*tmr_t∗ this, int us*), que inicia el timer, siendo *us* el tiempo en microsegundos que debe esperar.

En el momento en el que se ejecuta el timer, este comienza una cuenta en microsegundos y cuando llega al valor indicado en la función *tmr_startus*(), se ejecuta la función declarada como *isr* en *tmr_new*(*notify_func_t isr*).

Para nuestro caso, lo que hará la función de *isr* será activar un nuevo semáforo

del que dependerán ambas hebras de muestreo y que indicará a estas el momento en el que se debe realizar el muestreo. De esta manera, se logra la ejecución con periodo seleccionable del muestreo de entradas, tanto analógicas como digitales.

2.5. Mejoras del programa

Hasta ahora tenemos un sistema funcional, que satisface todos los requisitos explicados en la Sección 1.4.2. Aun así, se ha decidido llevar a cabo una serie de modificaciones para mejorar la experiencia del usuario y hacer más fácil el análisis de los datos extraídos.

2.5.1. Sistema de ficheros

Como se menciona en las Secciones 2.3.1, 2.3.3 y 2.3.4, los datos que se obtienen son guardados por separado en una serie de ficheros. Debido a que se va a utilizar el sistema de manera continuada y en varias sesiones, se creará un directorio según la fecha de la sesión, y otro según la hora. De esta manera, cada vez que se ejecute el programa, los ficheros de datos quedarán localizados en su respectiva fecha y hora de comienzo.

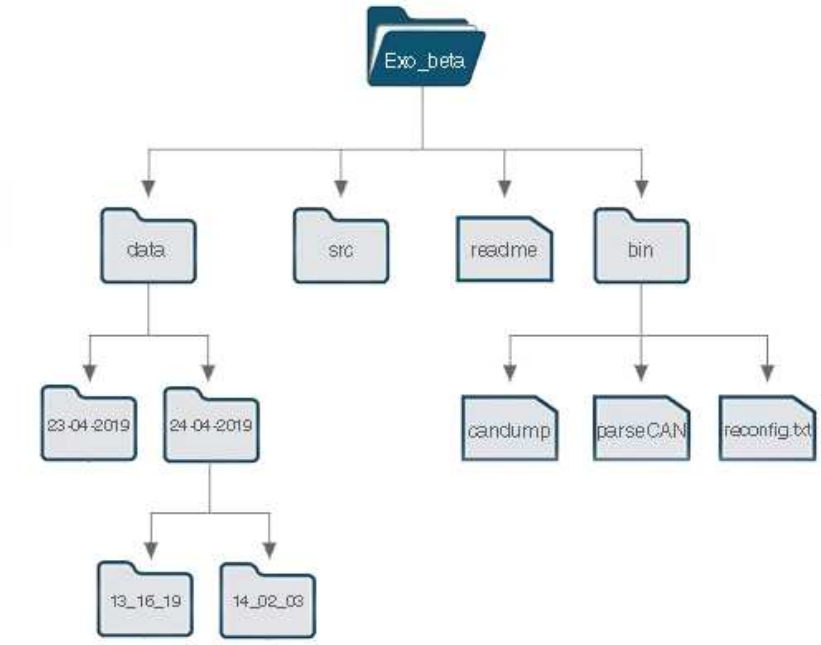


Figura 2.13: Ejemplo de sistema de ficheros en la Raspberry Pi

2.5.2. Inicio mediante disparo

Además del comienzo de captura de datos mediante el teclado, se ha decidido incluir dos de las entradas digitales como gatillo o *trigger*; uno de comienzo y

otro de final, para sincronizar la captura del sistema con el funcionamiento del resto de dispositivos. Esta opción es seleccionable en *reconfig.txt* en el valor de *KEY_OR_TRIGGER*. Si se le da el valor de 0, el programa sólo será sensible al teclado; en caso de ser 1, sólo lo será al trigger y en caso de ser 2, será sensible a ambos. Además, se ha añadido otro nuevo valor, *STOP_TRIGGER*, que habilitará el gatillo de final o no, según su valor sea 1 o 0.

En caso de habilitarse los dos gatillos, sus conectores serán los correspondientes a los GPIO 21 y 20, de comienzo y final respectivamente.

2.5.3. Análisis de mensajes CAN

Ya que los mensajes CAN se han guardado en su propio formato, extraer la información necesaria de ellos es una tarea complicada. Por ello, se ha creado un pequeño programa que, siguiendo la documentación del H2, descodifica los mensajes recibidos según sea su identificador e imprime los valores en un nuevo fichero *"parse_candump_XXX.csv*, donde XXX es el identificador recibido. De esta manera, se obtiene una tabla de valores significativos en un rango de 0 a 255, ya que la longitud máxima de estos es un byte.

Este programa, se ejecutará por defecto al terminar el programa principal. En caso de querer ejecutarse externamente, se deberá hacer de la siguiente manera:

```
$ ./parseCAN -fichero_a_traducir -directorio_destino –ID_CAN
```

De esta manera, al introducir el ID de los mensajes que están contenidos en el fichero, el programa dará un nombre a cada byte y comenzará a descodificar.

2.6. Diseño y ensamblado del encapsulado

En esta Sección, se explicará el diseño y montaje de la caja que servirá de encapsulado al sistema electrónico. Así mismo, la caja contará con los huecos necesarios para los conectores del diseño final.

2.6.1. Diseño en 3D

Para el diseño de la caja, se utilizará TinkerCAD, que es una herramienta online para el diseño de piezas en 3D [36].

En primer lugar, debemos medir las dimensiones necesarias para que quepa la Raspberry y todos los cables y componentes del conexionado. Las dimensiones del sistema actual son 85mm x 56mm x 48mm [37].

Debido a que todos los conectores elegidos se localizarán en la tapa de la caja y que sus dimensiones son fijas, las dimensiones de la caja se verán condicionadas por ellos y no por la Raspberry. Por tanto las dimensiones finales elegidas son 140mm x 90mm x 80mm.

Por otro lado, la Raspberry debe ir anclada a la caja así que son necesarios 4 taladrados que se corresponderán con los taladrados de la propia placa. También, son necesarios 4 agujeros, que se corresponden con los 2 hubs usb, el conector Ethernet de la Raspberry y el conector de alimentación elegido. Además, se habilitará un taladrado

adicional para una posible ventilación. Por último, se añadirán 6 agujeros para poder atornillar la tapa de la caja (ver Figura 2.14).

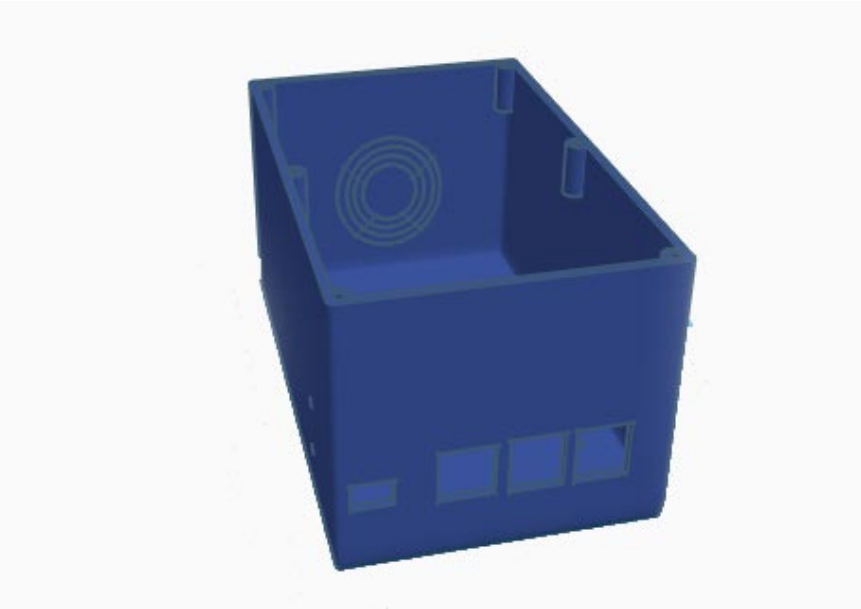


Figura 2.14: Diseño en 3D de la caja.

Para el diseño de la tapa, simplemente debemos hacer los agujeros correspondientes a los conectores y a sus orificios de atornillado como se ve en la FIgura 2.15.

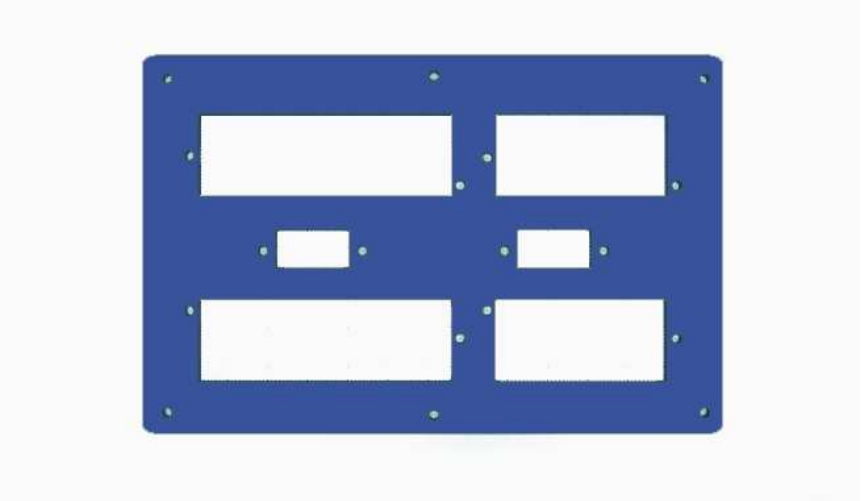


Figura 2.15: Diseño en 3D de la tapa.

2.6.2. Cables y conectores

- **Alimentación**

  Debido a que la alimentación de la Raspberry Pi es únicamente mediante micro-USB, en primer lugar pondremos un cable micro-USB [38] que se cortará, pelará y empalmará con el conector de alimentación elegido. Ambos componentes se pueden ver en la Figura 2.16. Para una fácil alimentación, como conector externo se ha escogido un coaxial hembra [39], que irá anclado a una pared de la caja y en uno de los agujeros ya diseñados.

(a) Cable micro-USB acodado.

(b) Conector de alimentación coaxial hembra.

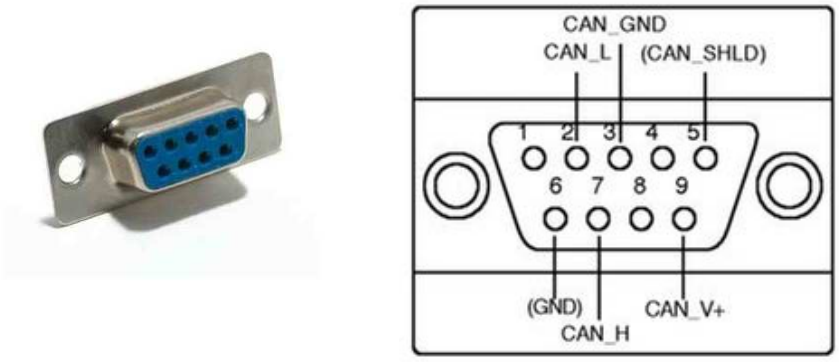
Figura 2.16: Conectores para la alimentación de Raspberry Pi.

- **Conexionado CAN**

  Para la conexión del bus CAN, es necesario un conector DB-9, mostrado en la Figura 2.17, ya que es el que se utiliza para el exoesqueleto H2. Para nuestro caso utilizaremos únicamente los puertos correspondientes a *CAN_L*,*CAN_H* y *GND*.

Pin	Señal	Descripción
1	Reservado	Para mejoras
2	CAN_L	Nivel bajo dominante
3	CAN_GND	Tierra
4	Reservado	Para mejoras
5	CAN_SHLD	Blindaje (opcional)
6	GND	Tierra (opcional)
7	CAN_H	Nivel alto dominante
8	Reservado	Para mejoras
9	CAN_V+	Alimentación (opcional)

Tabla 2.3: Leyenda del conector DB-9 para CAN bus de la Figura 2.17(b)



(a) Conector DB-9 hembra. (b) Esquema conexionado CAN.

Figura 2.17: Conector DB-9 para CAN bus.

- **Entradas analógicas y digitales**

  Contamos con 10 entradas digitales y 8 analógicas. A cada entrada, le asignaremos dos pines, uno correspondiente a la señal y otro a la masa. Por lo tanto, son necesarios 36 puertos para el conexionado. Separando las entradas analógicas y digitales, utilizaremos dos conectores de 12 entradas para las digitales (BC2EHDRS12P) y otros dos de 8 entradas del fabricante Conexcon para las analógicas (BC2EHDRS08P) [40]. En la Figura 2.18 se puede ver una foto y un esquema del conector.

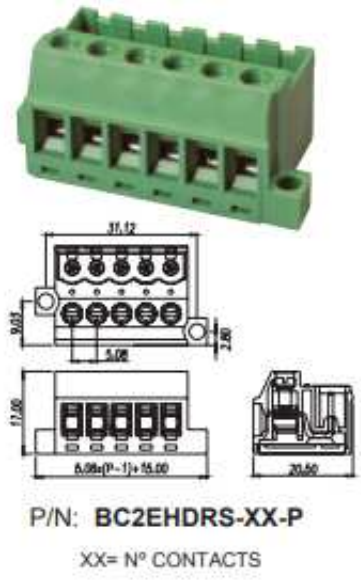


Figura 2.18: Conector hembra para las entradas analógicas y digitales.

2.6.3. Ensamblado

Para ensamblar todos los componentes anteriormente mencionados con la caja y la tapa, haremos uso de una serie de tornillos y tuercas de unas medidas específicas.

En el anclaje de la Raspberry a la caja, utilizaremos 4 tornillos M2.5x0.45 que se atornillarán a 4 separadores metálicos que atraviesan los taladrados laterales de la Raspberry.

Además, se ha soldado el conector coaxial hembra a una placa de puntos para su fijación dentro de la caja. Esta placa será taladrada por dos puntos que servirán de anclaje a la caja. La fijación se llevará a cabo mediante dos tornillos y dos tuercas M3.

Para la conexión de los GPIO con las entradas digitales, se ha buscado un cable para el puerto GPIO cuya conexión sea fiable y robusta frente a golpes. La mejor opción que se ha encontrado es el CABLE GPIO PARA RASPBERRY PI B+ 40 PIN 150MM de Electan [41]. Este cable cuenta con conexiones para los 40 pines de la placa, pero sólo se utilizarán los correspondientes a las entradas digitales.

Otra posibilidad habría sido los cables puente hembra. Estos se descartaron porque podrían soltarse debido a que no tienen ningún tipo de sujeción mecánica.

Figura 2.19: Cable GPIO de 40 entradas para Raspberry Pi 3 B+.

La conexión de las entradas analógicas se cablearán mediante cables puente. Estos cables tienen un conector macho en cada punta y nos permitirán la conexión entre el ADC y los conectores correspondientes a las entradas analógicas. Esta conexión consistirá en el atornillado de las entradas de los conectores mientras el cable se encuentra en su interior. El resultado final se puede ver en la Figura 2.20.

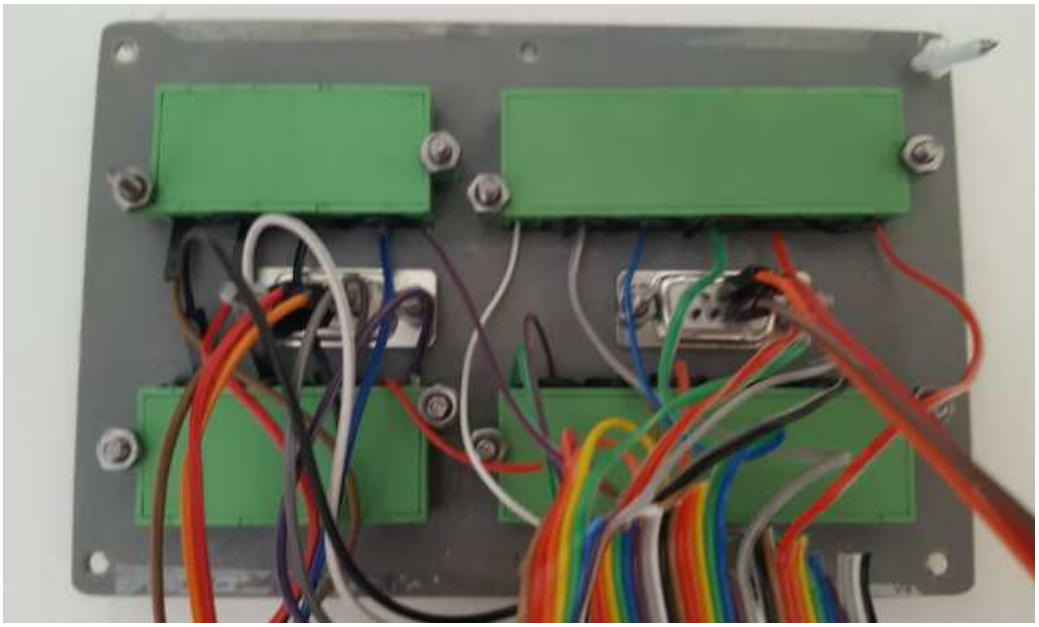


Figura 2.20: Cableado de entradas analógicas y digitales.

También, se anclarán los conectores de entradas analógicas y digitales mencionadas en la Sección 2.6.2 a la tapa de la caja. Este proceso será de la misma forma que el del conector coaxial, mediante tornillos y tuercas M3 que atravesarán los agujeros del conector destinados para su fijación.

Por último, se colocará la tapa encima de la caja y se atornillará mediante 6 tornillos M3 en los correspondientes agujeros de la caja.

2.7. Página web para acceso remoto

Tras un largo proceso de codificación y ensamblado, es necesaria la configuración de una vía de comunicación con el sistema electrónico de fácil acceso. En esta Sección, se describirá el proceso de codificación de una página web. El objetivo de esta página web es el inicio del proceso de captura de datos y el acceso a ellos. El uso de esta página web está explicado en el Anexo E.

2.7.1. Configuración del servidor

Para llevar a cabo la configuración de un servidor que ejecute los comandos necesarios para capturar los datos y por su fácil utilización para programación web, se ha decidido utilizar el lenguaje JavaScript.

En primer lugar, es necesario que la Raspberry esté conectada a una red. Con conectarla al ordenador desde el que se va a acceder mediante un cable Ethernet, sería suficiente. Una vez conectada a la red, se levantará un servidor en el puerto 80. Este servidor atenderá únicamente a peticiones GET, a las que responderá con una vista EJS (Embedded JavaScript). Una vista EJS es un código que emite un HTML basándose en código JavaScript. De esta manera, el servidor almacenará las variables del estado de la captura y el HTML que envíe al cliente dependerá de ellas.

La comunicación entre el cliente y el servidor será mediante la transferencia de comandos en los parámetros de la ruta URL. El servidor los leerá, procesará y responderá con un HTML acorde con el estado del programa.

2.7.2. Página HTML

Para un fácil control de la captura de datos y su posterior acceso, se dividirá la página en dos partes iguales. A la izquierda estarán los parámetros de configuración y los botones de comienzo y pausa de la captura de datos. En la parte derecha de la pantalla, aparecerá una lista con los ficheros de la carpeta *data* que se menciona en la Sección 2.5.1 y como se puede ver en la Figura 2.21.

Los parámetros de configuración serán los mismos que se encuentran en el fichero *reconfig.txt*. Al lado de cada parámetro, aparecerá un cuadro de texto donde se podrá poner el valor del parámetro. El valor de ciertos parámetros se podrá seleccionar de una lista de valores válidos. Una vez introducidos todos los valores deseados, se pulsará el botón de *COMENZAR*. Este botón se encargará de enviar todos los valores al servidor y este los escribirá en el fichero *reconfig.txt*. Tras ello, el servidor iniciará el programa de captura.

En caso de querer reconfigurar el exoesqueleto, se seleccionará el valor *SÍ* para la opción *RECONFIGURAR*, y se desplegarán todos los posibles parámetros a configurar del exoesqueleto.

Tras pulsar el botón *COMENZAR*, éste desaparecerá y sólo permanecerá el botón de *PARAR*, como se ve en la Figura 2.22. Este botón parará la captura del programa principal y se volverá al estado inicial del HTML.

En el lado derecho de la pantalla, se encuentra una lista de los ficheros de la carpeta *data*. De esta lista se podrá elegir una de las carpetas según su fecha. Tras ello, se podrá pulsar el botón *VER*, que se encuentra al lado de la lista. Al pulsar este botón, los ficheros mostrados en la lista pasarán a ser los que contiene la carpeta seleccionada. Al lado del botón *VER* se encuentra el botón *VOLVER*. Al presionar este botón, se visualizarán los ficheros y directorios contenidos en la carpeta anterior. De esta manera, se podrá explorar el directorio *data* con todos los datos capturados. Si se entra en un directorio con ficheros de datos, aparecerá la opción de pulsar un botón *DESCARGAR* que descargará el fichero seleccionado en el dispositivo desde el que se acceda a la web.

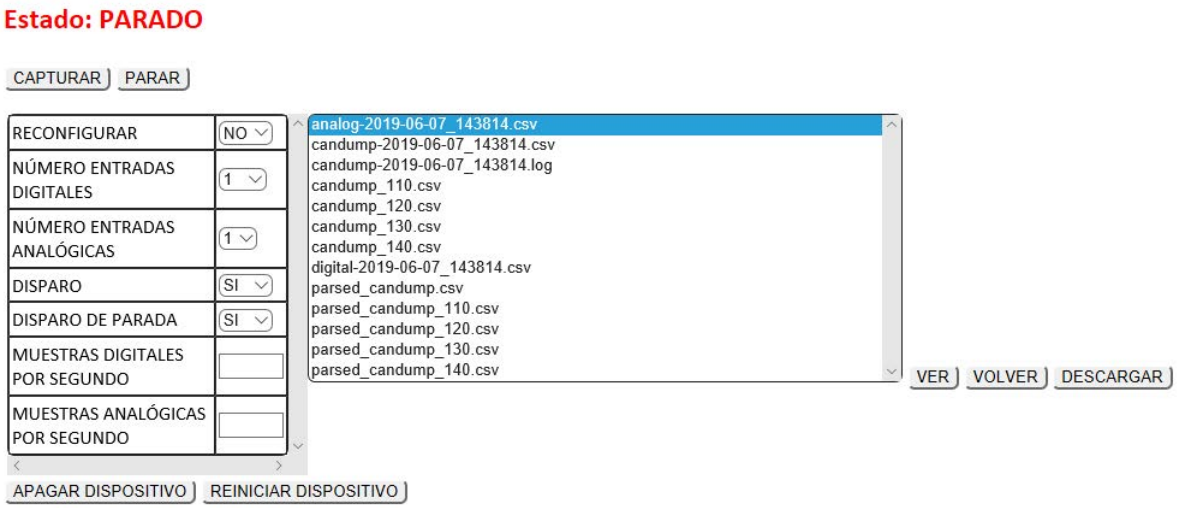


Figura 2.21: Captura de la página web con el programa parado.

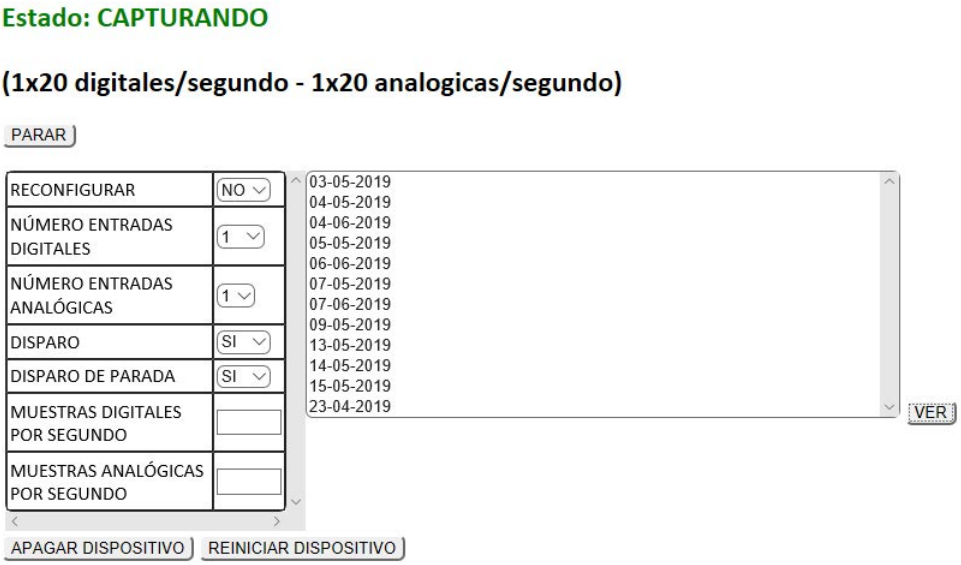


Figura 2.22: Captura de la página web con el programa capturando.

Capítulo 3

Validación del sistema

En este Capítulo se explicarán los procesos de validación usados para probar la funcionalidad del sistema electrónico diseñado. En primer lugar, se hablará del módulo USB-CAN utilizado para validar el sistema de recepción CAN. Tras ello, se explicarán los experimentos llevados a cabo en el Hospital Nacional de Parapléjicos de Toledo para validar el sistema completo.

3.1. Pruebas previas

Para una correcta validación del sistema, se ha considerado necesaria una prueba de la recepción CAN previa al simulacro en el Hospital. Esta prueba se llevará a cabo mediante el uso de un módulo USB-CAN. Este nos permite enviar paquetes CAN desde nuestro propio ordenador. De esta manera, será posible la comprobación del funcionamiento del sistema. El módulo USB-CAN elegido para este Trabajo de Fin de Grado es el Adaptador convertidor USB-CAN de Q Baihe (ver Figura 3.1) [42]. Se ha elegido este sistema por su bajo precio y su fácil transporte.

Figura 3.1: Módulo USB-CAN de Q Baihe.

Este módulo viene acompañado de unos drivers y un programa mediante el cuál se podrán enviar los mensajes CAN, cuya captura aparece en la Figura 3.2. Tras instalar los drivers, se envían dos mensajes desde la Raspberry hasta el módulo, y en la pantalla principal del programa se puede comprobar la recepción de estos dos mensajes. Para comprobar la recepción de paquetes en la Raspberry, se envían cinco mensajes desde el módulo USB-CAN al módulo can0 de la Raspberry y se comprueba su recepción mediante el comando *candump*, explicado en la Sección 2.3.1 (ver Figuras 3.2 y 3.3).

USBCAN V7.10A

Total: 5000 Display receive only Statistical model Clear Pause Continue Save Auto Save

No	Direction	Time scale	Frame Type	Frame Format	Frame ID	Data Length	Data(LDouble-click Hex->Dec)
0	Receive	20:54:19:417	Data frame	Standard frame	00000123	4	de ad be ef
1	Receive	20:54:53:242	Data frame	Standard frame	00000123	4	de ad be ef
2	Send	20:55:04:445	Data frame	Standard frame	00000000	8	11 22 33 44 55 66 77 88
3	Send	20:55:05:610	Data frame	Standard frame	00000000	8	11 22 33 44 55 66 77 88
4	Send	20:55:05:985	Data frame	Standard frame	00000000	8	11 22 33 44 55 66 77 88
5	Send	20:55:06:197	Data frame	Standard frame	00000000	8	11 22 33 44 55 66 77 88
6	Send	20:55:06:372	Data frame	Standard frame	00000000	8	11 22 33 44 55 66 77 88

Figura 3.2: Captura del programa USB-CAN durante el experimento.

```
pi@raspberrypi:~ $ cansend can0 123#deadbeef
pi@raspberrypi:~ $ cansend can0 123#deadbeef
pi@raspberrypi:~ $ candump can0
  can0  000   [8]  11 22 33 44 55 66 77 88
  can0  000   [8]  11 22 33 44 55 66 77 88
  can0  000   [8]  11 22 33 44 55 66 77 88
  can0  000   [8]  11 22 33 44 55 66 77 88
  can0  000   [8]  11 22 33 44 55 66 77 88
```

Figura 3.3: Captura del terminal de la Raspberry Pi durante el experimento.

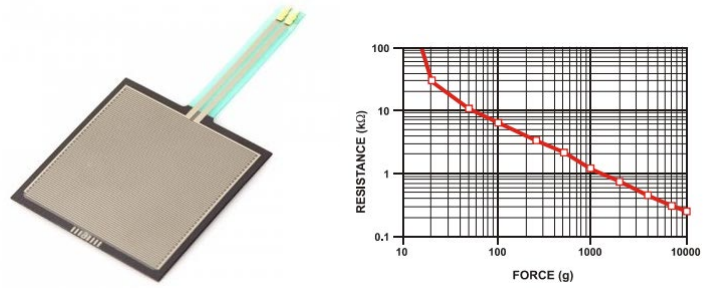
Tras este experimento podemos concluir que la recepción y el envío CAN son correctos y podemos realizar el ensayo en el Hospital Nacional de Parapléjicos de Toledo.

3.2. Ensayo y valores obtenidos

En esta Sección, se explicará la configuración realizada para la validación y las pruebas realizadas con el dispositivo. Tras ello, se mostrarán los valores obtenidos y se realizará un análisis de los mismos.

3.2.1. Configuración

La validación que se ha realizado consiste en la medición de la fuerza de varios músculos de la pierna derecha. Esto se conseguirá mediante los sensores de fuerza del exoesqueleto H2 y mediante la colocación de un sensor de fuerza resistivo (FSR) entre el músculo a medir y las sujeciones del exoesqueleto. Este sensor formará parte de un divisor de tensión con una resistencia de 100 kΩ y se conectarán sus bornas a una entrada analógica, como se puede ver en la Figura 3.5. De esta manera podremos ver pequeñas variaciones en la presión ejercida sobre él. Además, se conectará a otra entrada analógica una señal de salida del exoesqueleto que indica cuándo se está realizando un ejercicio.



(a) Sensor FSR.

(b) Características del sensor FSR.

Figura 3.4: Sensor FSR y sus características.

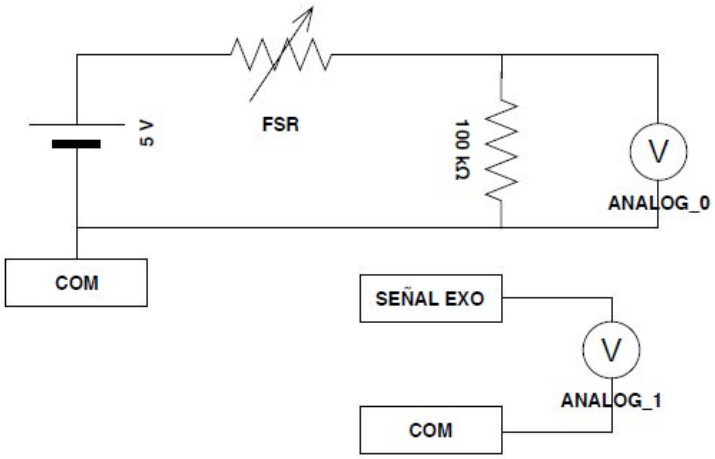


Figura 3.5: Configuración para el simulacro en el Hospital.

3.2.2. Pruebas realizadas

Para este Trabajo de Fin de Grado, se han realizado los mismas pruebas para 3 sujetos diferentes. El objetivo de estas pruebas es el estudio del movimiento de articulaciones aisladas y de la fuerza que estas pueden ejercer. De esta manera se podrá contrastar los valores obtenidos para sacar conclusiones.

En primer lugar, el sujeto que realice la prueba debe ponerse el exoesqueleto. Una vez ajustadas todas las articulaciones del exoesqueleto al paciente correspondiente, se coloca el FSR en el empeine del pie derecho como se puede ver en la Figura 3.6 y comienza la captura de datos. Para esta prueba, se ha elegido una frecuencia de muestreo de las señales analógicas de 8 muestras por segundo. Tras realizar la prueba con el primer sujeto, se cambiará la frecuencia de muestreo a 25 muestras por segundo para validar la captura a una frecuencia más alta.

La primera prueba a realizar es una contracción isométrica del tobillo derecho mientras la articulación del tobillo derecho del exoesqueleto es la única articulación que no está bloqueada por el exoesqueleto. El bloqueo del resto de articulaciones se realiza con el fin de asegurarse que sólo se mueve el tobillo y que la rodilla y cadera no aportan nada a ese movimiento. De esta manera, el sujeto debe ejercer toda la

(a) Colocación del exoesqueleto.

(b) Colocación del FSR.

Figura 3.6: Colocación del exoesqueleto y del FSR.

fuerza posible intentando realizar un movimiento dorsal del tobillo mientras se le ofrece resistencia al movimiento y así lograr la contracción voluntaria máxima. Este ejercicio se repetirá 5 veces.

Tras esta primera prueba, se realizarán 10 flexiones y 10 extensiones del tobillo derecho, siendo el sujeto ayudado por el exoesqueleto y acompañando al mismo durante el movimiento.

La tercera prueba es una contracción isométrica de la rodilla derecha. Se colocará el FSR por encima de la rodilla del sujeto y debajo de una de las sujeciones del exoesqueleto como se puede ver en la Figura 3.7. El exoesqueleto se bloquea de nuevo para permitir un movimiento aislado de la rodilla y el sujeto intenta realizar una extensión de la rodilla mientras se le ofrece resistencia a ese. De esta manera, se mide la fuerza que realiza el sujeto en esta contracción isométrica.

Por último, se realizan 10 flexiones y 10 extensiones de la rodilla derecha, siendo el sujeto, de nuevo, ayudado por el exoesqueleto para realizar el movimiento.

Tras realizar el segundo ejercicio, se termina la captura de datos, se retira el FSR y el exoesqueleto siendo la duración total de las pruebas son aproximadamente 10 minutos.

3.2.3. Valores obtenidos

Tras la realización de las pruebas a los 3 sujetos, se procede a hacer un análisis de los datos obtenidos. Se van a presentar los datos del sujeto 1 como una muestra representativa para mostrar los datos que se obtienen de las pruebas realizadas. Posteriormente, se presentará una comparativa entre los 3 sujetos analizados.

Figura 3.7: Contracción isométrica de la rodilla derecha del sujeto.

3.2.3.1. Sujeto 1

En primer lugar, representaremos los datos obtenidos de las dos entradas analógicas, la activación del exoesqueleto y el valor del FSR, junto con la fuerza realizada por el motor en el tobillo derecho obtenida del exoesqueleto. Se ha normalizado tanto esta última como la entrada analógica del FSR, para poder contrastar los valores, debido a que la fuerza realizada por el motor varía entre 0 y 255 y del FSR una tensión alrededor de los 4 voltios.

Como se puede ver en la Figura 3.8, las señales obtenidas son muy ruidosas. En primer lugar, se ve que la señal de activación del exoesqueleto es la menos ruidosa e indica el comienzo de la contracción. A partir de ese momento, tanto el FSR como la fuerza del motor del tobillo ascienden en valor. También se puede distinguir fácilmente las 5 repeticiones que se realizan del ejercicio. Por último, podemos ver cómo la forma de ambas señales, *MOTOR TOBILLO* y *FSR*, se parecen mucho, pudiendo concluir que la captura de mensajes CAN y entradas analógicas son correctas.

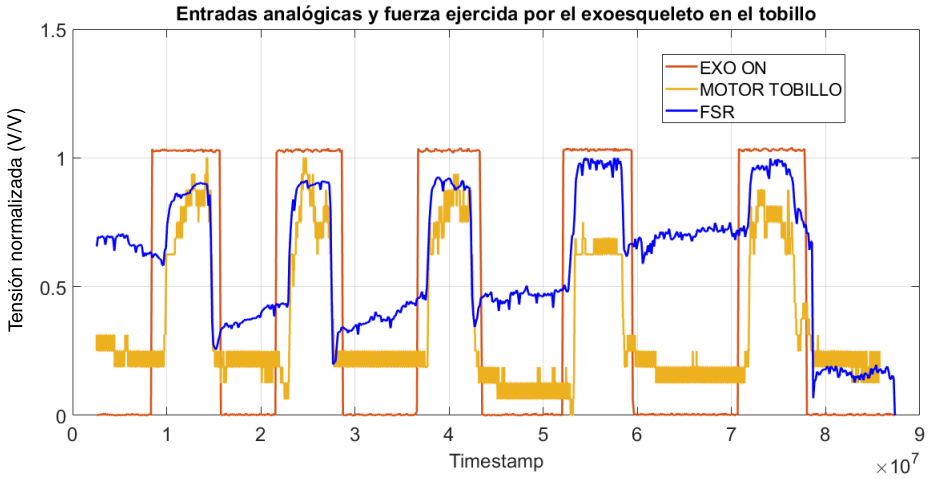


Figura 3.8: Contracción isométrica del tobillo.

Tras la contracción isométrica, se realiza la flexión y extensión del tobillo 10 veces. En este caso, en vez de representar la fuerza del motor del exoesqueleto, se representará la posición angular del tobillo. De esta manera se podrá contemplar el efecto que tiene la posición angular del tobillo sobre la presión ejercida sobre el FSR.

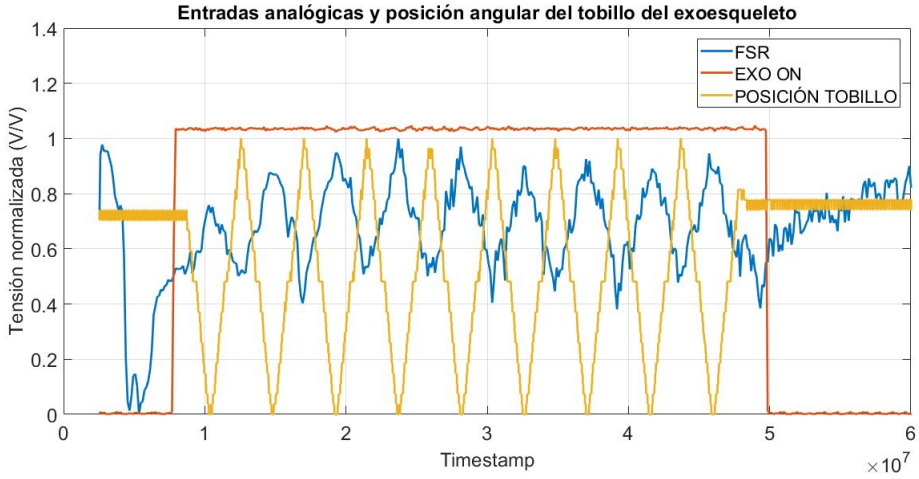


Figura 3.9: Flexión y extensión de tobillo

En la Figura 3.9, se puede ver cómo la señal *FSR* va en contrafase a la señal *POSICIÓN TOBILLO*. Cuanto mayor es el ángulo entre pie y tibia, menor es el valor de la posición del tobillo. Por lo que podemos concluir que la presión es mayor cuanto menor es la posición del tobillo. Además, anteriormente se ha comentado que el ejercicio consistía en 10 repeticiones, pero en la gráfica sólo se ven 9 oscilaciones. Esto es muestra de otra aplicación del sistema electrónico, detectando la falta de repeticiones en el tratamiento.

Concluido el segundo ejercicio, comienza la contracción isométrica de la rodilla.

Al igual que en el primer ejercicio, se representará la fuerza ejercida por el motor, en este caso, de la rodilla, junto con el FSR y la activación del exoesqueleto.

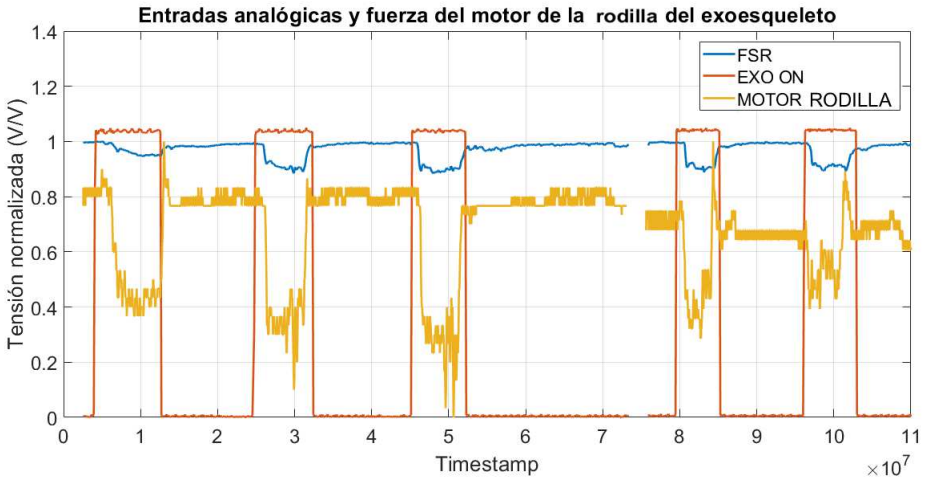


Figura 3.10: Contracción isométrica de la rodilla.

De nuevo de la Figura 3.10 se pueden sacar ciertas conclusiones. En primer lugar, que la captura fue interrumpida para comprobar que la captura de valores era correcta, como se puede ver en la falta de valores entre el timestamp $7 * 10^7$ y $8 * 10^7$. También se contempla que el ejercicio se repite 5 veces. Por último, se puede ver que de nuevo coinciden la forma de la fuerza ejercida por el motor con la del FSR.

Finalmente, representaremos la posición del sensor FSR y la activación del exoesqueleto junto con la posición angular de la rodilla.

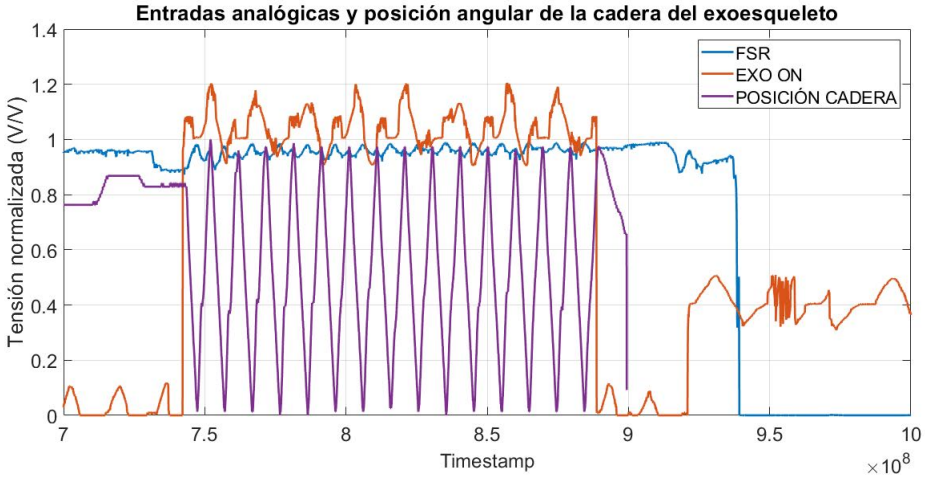


Figura 3.11: Flexión y extensión de la rodilla

En este caso, la Figura 3.11 no permite sacar demasiadas conclusiones respecto a la presión ejercida sobre el FSR. Sí lo hace la función de la posición angular de la rodilla, que nos indica que el ejercicio se repitió 15 veces.

3.2.3.2. Comparativa de los tres sujetos

Otra de las posibles aplicaciones del sistema electrónico diseñado es que permite la comparación de fuerza entre sujetos. Esta comparación permitiría valorar los parecidos

entre los datos de personas con lesión medular y los obtenidos de personas sanas y así valorar el desarrollo del paciente y realizar una terapia acorde a este. Anteriormente se han normalizado los valores obtenidos para su análisis, en este caso, no se normalizará, para poder compararlos.

Se compararán los valores obtenidos de la contracción isométrica de tobillo y rodilla de los 3 sujetos.

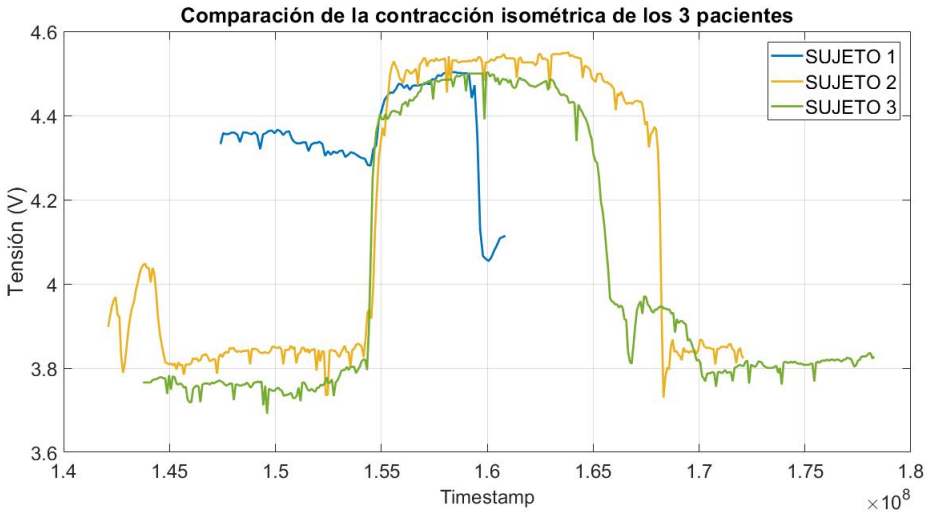


Figura 3.12: Comparación de los 3 sujetos en la contracción del tobillo.

De la Figura 3.12 se pueden sacar ciertas conclusiones. En primer lugar, que la duración de la contracción ha sido diferente en los 3 casos. Por tanto, se deberá tener en cuenta a la hora de compararlos. También se puede ver que, aunque las amplitudes máximas son parecidas, la mayor variación en el valor es la del *SUJETO 2*. Esto muestra que el *SUJETO 2* ha realizado una contracción mayor, además de más duradera.

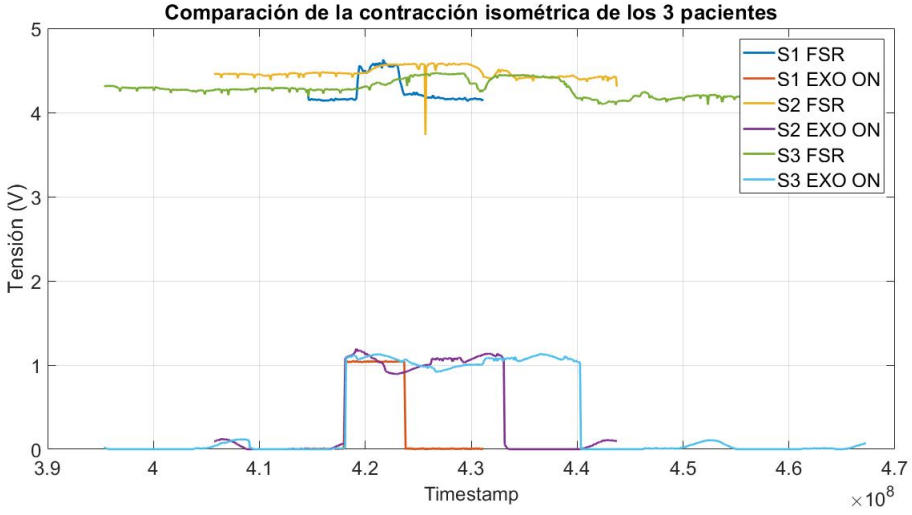


Figura 3.13: Comparación de los 3 sujetos en la contracción de la rodilla.

En el caso de la Figura 3.13, se ha representado, además del valor del FSR, la activación del exoesqueleto para cada uno de los sujetos. Respecto a la intensidad de la contracción no se pueden sacar muchas conclusiones, debido a que no hay grandes variaciones en el valor del FSR. No obstante, en la Figura 3.13 sí se puede observar que las contracciones son de diferente duración, como se comentó previamente. Por lo tanto, para poder hacer una comparación justa en personas sanas, sería necesario que las contracciones fueran de una duración aproximadamente igual. En el caso de personas lesionadas, la duración de estas contracciones puede aportar información sobre la activación muscular y se deberá tener en cuenta en el análisis de los datos.

3.3. Conclusiones

Tras haber analizado y comparado los resultados obtenidos, se puede concluir que la validación del sistema electrónico se ha realizado con éxito. La captura de los datos ha sido correcta y coherente con los valores esperados.

La configuración de los sensores para esta prueba ha sido diseñada para la validación del sistema electrónico, pero en los próximos ensayos deberán tenerse en cuenta ciertos aspectos:

- El análisis de los datos permite localizar fallos en la realización de las pruebas como un número de repeticiones insuficiente o excesivo.
- La representación de los datos permite comprobar que la prueba ha sido la misma para todos los sujetos excepto la duración de las contracciones.
- Es posible que el lugar donde se ha decidido colocar el sensor no sea el más adecuado para lograr un análisis igual para todos los sujetos y ensayos.
- Se puede detectar un mal funcionamiento del exoesqueleto o la avería de sus sensores gracias a los mensajes que envía por el bus CAN.

Se puede concluir, por tanto, que el objetivo de este Trabajo de Fin de Grado, mencionado en la Sección 1.4, ha sido cumplido.

Capítulo 4

Conclusiones y líneas futuras

4.1. Conclusiones

A continuación, se exponen las conclusiones obtenidas a lo largo de la realización de este Trabajo de Fin de Grado:

1. La utilización de electrónica de consumo de bajo coste y desarrollos software propios han permitido la implementación de un sistema electrónico capaz de monitorizar la marcha en personas con lesión medular de manera robusta, compacta y ligera.

2. El sistema electrónico es capaz de integrar diferentes dispositivos analógicos y digitales para un análisis conjunto de los datos.

3. Los experimentos realizados tanto en el laboratorio como en el Hospital Nacional de Parapléjicos de Toledo han resultado satisfactorios, cumpliendo con las especificaciones necesarias.

4. El sistema es capaz de levantar un servidor web al mismo tiempo que captura datos. Esto es muy útil para un acceso remoto a los datos desde cualquier ordenador en la misma red.

5. La impresión en 3D es idónea para lograr una estructura a medida, robusta y de bajo peso; siempre y cuando se realice un diseño correcto.

4.2. Líneas futuras

Las líneas futuras del presente Trabajo de Fin de Grado van orientadas a aumentar las funcionalidades del sistema electrónico. Cabe destacar las siguientes líneas de actuación:

- Validación del sistema electrónico en configuraciones diferentes a las ya realizadas y, en particular, en una configuración que no gire entorno al exoesqueleto H2. Un posible ejemplo de este tipo de configuración es un sistema de electromiográfica superficial e implantada con agujas. La obtención de datos de este sistema sería útil para detectar la degradación de la aguja implantada por el roce con las fibras musculares.

- Validación de la reconfiguración del exoesqueleto H2 a través del sistema electrónico diseñado en este Trabajo de Fin de Grado.
- Mejora de los datos proporcionados en la página web. Habilitar un espacio en la página web para proporcionar gráficas de los datos obtenidos en tiempo real. Tras consultar esta opción con el equipo del Hospital Nacional de Parapléjicos de Toledo, se considera conveniente que estas gráficas se actualicen cada 200 milisegundos y reinicien sus valores cada 10 segundos. Además, habilitar la opción de descargar estas gráficas de la misma manera que los ficheros de datos.
- Habilitar una señal digital de salida en el sistema electrónico como señal de disparo. Esta señal permitiría la integración de otros sistemas en la configuración y sincronizar sus ejes de tiempos en la captura.
- Desarrollo de una aplicación para dispositivos Android que permita un mejor acceso al sistema electrónico y a los datos obtenidos desde dispositivos móviles.

Bibliografía

[1] Eduardo Díaz Velázquez. Análisis sobre la lesión medular en España - Informe de resultados. Technical report, 2012.

[2] A.S.I.A. American Spinal Injury Association. https://asia-spinalinjury.org/. [online; accesed 2019-06-09].

[3] Solidaridad Digital. La esperanza de vida de los lesionados medulares se iguala a la del resto de población. 2009.

[4] Instituto Nacional de Estadística. Tasa de población con discapacidad que tiene diagnosticadas determinadas enfermedades crónicas según la enfermedad por CCAA y sexo . http://ine.es/jaxi/Datos.htm?path=/t15/p418/a2008/hogares/p02/modulo1/l0/{&}file=04028.px. [Online; accessed 2019-06-08].

[5] Instituto de Biomecánica de Valencia. Aplicaciones Biomecánicas. Technical report.

[6] Dorothy Weiss and Lisa S. Krivickas. Rehabilitation in Neuromuscular Disorders. In *Neuromuscular Disorders: Treatment and Management*, pages 115–136. Elsevier, 2011.

[7] Monzurul Alam and Yong-ping Zheng. Motor neuroprosthesis for injured spinal cord: who is an ideal candidate? *Neural Regeneration Research*, 12(11):1809, 2017.

[8] Antonio J Del-Ama, Angel Gil-Agudo, José L Pons, and Juan C Moreno. Hybrid gait training with an overground robot for people with incomplete spinal cord injury: a pilot study. *Frontiers in human neuroscience*, 8:298, 2014.

[9] Hospital La Pedrera. El Hospital La Pedrera incorpora el sistema WalkAide, una Neuroprótesis para el tratamiento del pie equino. http://lapedrera.san.gva.es/noticias/-/asset{_}publisher/sH3K/content/el-hospital-la-pedrera-incorpora-el-sistema-walkaide-una-neuroprotesis-para-el-tratam jsessionid=0476A4444C43C3EA57439DE38AB7334F.appli7{_}node2?redirect=http{%}3A{%}2F{%}2Fl. [Adaptado; accesed 2019-06-09].

[10] SCI Models Systems. Lesión de la médula espinal y rehabilitación de la marcha. Technical report, 2011.

[11] Ashraf S. Gorgey. Robotic exoskeletons: The current pros and cons. *World Journal of Orthopedics*, page 9, 2018.

[12] Revista Aquí. Felicitan al Hospital de Parapléjicos por los trabajos sobre exoesqueletos. http://revistaaqui.info/felicitan-al-hospital-de-paraplejicos-por-los-trabajos-sobre-exoesqueletos/. [Adaptado; accesed 2019-06-09].

[13] Magdo Bortole, Anusha Venkatakrishnan, Fangshi Zhu, Juan C Moreno, Gerard E Francisco, Jose L Pons, and Jose L Contreras-Vidal. The H2 robotic exoskeleton for gait rehabilitation after stroke: early findings from a clinical study. *Journal of NeuroEngineering and Rehabilitation*, 12(1):54, 2015.

[14] Raspberry Pi. What is a Raspberry Pi? https://www.raspberrypi.org/help/what-is-a-raspberry-pi/. [Online; accessed 2019-03-23].

[15] Arm Limited. ARM ® Cortex ®-A53 MPCore Processor Technical Reference Manual. Technical report, ARM, 2013.

[16] Raspberry Pi. Raspberry Pi 3 Model B+. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Online; accessed 2019-06-11].

[17] Raspbian - Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/raspbian/. [Online; accessed 2019-03-23].

[18] Debian – Details of package gcc-arm-linux-gnueabihf in stretch. https://packages.debian.org/stretch/devel/gcc-arm-linux-gnueabihf. [Online; accessed 2019-05-06].

[19] Raspberry Pi Geek. GPIO Pinout. http://www.raspberry-pi-geek.com/howto/GPIO-Pinout-Rasp-Pi-1-Model-B-Rasp-Pi-2-Model-B. [Online; accessed 2019-06-11].

[20] Pat Richards. Preliminary M AN228. Technical Report 12, Microchip Technology Inc, 2002.

[21] ISO 11519-2:1994 - Road vehicles – Low-speed serial data communication – Part 2: Low-speed controller area network (CAN). https://www.iso.org/standard/19470.html, 1994. [online; accesed 2019-06-05].

[22] CAN Bus ISO 11519-2:1994. Technical report, ISO, 1994.

[23] ISO 11898-3:2006(en), Road vehicles — Controller area network (CAN) — Part 3: Low-speed, fault-tolerant, medium-dependent interface. https://www.iso.org/obp/ui/{#}iso:std:iso:11898:-3:ed-1:v1:en. [Online; accessed 2019-03-23].

[24] CAN Bus Interface Description CANbus Pin Out, and Signal names. Controller Area Network. http://www.interfacebus.com/CAN-Bus-Description-Vendors-Canbus-Protocol.html. [Online; accessed 2019-03-29].

[25] Steve Corrigan. Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN). Technical report, Texas Instruments, 2002.

[26] Duo B Rev. PiCAN 2 DUO DATASHEET. www.skpang.co.uk, 2016. [online; accesed 2019-06-05].

[27] Inc. Motorola. SPI Block Guide V03.06. https://opencores.org/usercontent/doc/1499360489, 2003. [Online; accessed 2019–06-05].

[28] Bang good. ADS1115-ADC. https://www.banggood.com/ADS1115-ADC-Module-For-Raspberry-Pi-3-2-B-p-1115682.html. [Online; accessed 2019-06-11].

[29] ABElectronics. ADC Pi. https://www.abelectronics.co.uk/docs/stock/raspberrypi/adcpi/adcpi-1.jpg. [Online; accessed 2019-06-11].

[30] NXP Semiconductors. I2C-bus specification and user manual. Technical report, NXP, 2014.

[31] Atomic Rhubarb. I 2 C 2-Wire. Technical report, 2012.

[32] My electronics Lab. Raspberry pi 3 gpio pinout 40 pin header block connector. https://myelectronicslab.com/wp-content/uploads/2016/06/raspbery-pi-3-gpio-pinout-40-pin-header-block-connector-1-1.png, 2019. [online; accesed 2019-03-27].

[33] ADC Pi - ADC converter for the Raspberry Pi. https://www.abelectronics.co.uk/p/69/adc-pi-raspberry-pi-analogue-to-digital-converter{#}code. [Online; accessed 2019-05-06].

[34] Jaroska. CONIO Reference Manual 2.0. Technical report, 2004.

[35] IEEE Computer Society. Portable Applications Standards Committee. and England) Open Group (Reading. *Standard for information technology : portable operating system interface (POSIX) : base definitions.* Institute of Electrical and Electronics Engineers, 2004.

[36] TinkerCAD. TinkerCAD. https://www.tinkercad.com/. [Online; accessed 2019-06-11].

[37] Mechanical Drawing - Raspberry Pi 3B +. [Online; accessed 2019-03-29].

[38] Qualtek - Cable Assemblies - DigiKey. https://www.digikey.com/product-detail/en/qualtek/3021078-03/Q1226-ND/8681881. [Online; accessed 2019-05-07].

[39] PJ-037A CUI Inc. — Connectors, Interconnects — DigiKey. https://www.digikey.com/product-detail/en/cui-inc/PJ-037A/CP-037A-ND/1644545. [Online; accessed 2019-05-07].

[40] Bc2esd and Bcesdk. CATALOGUE BC2ESD-BC2EHD-BC2ESH-BCESDK. Technical report, Conexcon.

[41] Electan - Cable GPIO Raspberry PI 3 B+. https://www.electan.com/cable-gpio-para-raspberry-pin-150mm-p-6187.html?gmeltn=1&gclid=CjwKCAiAs8XiBRAGEiwAFyQ-esDU5tulisO3TMRUWHlqjKugDUjP1IWdMYSRl_2kM73UKCxZU7SgPBoCmX0QAvD_BwE. [Online; accessed 2019-06-05].

[42] USB-CAN Q Baihe. https://www.amazon.es/gp/product/B017N7YEAO?ref=em_1p_0_ti&ref_=pe_3459411_360920091. [Online; accessed 2019-06-06].

Apéndice A

Arquitectura de comunicaciones del exoesqueleto H2

H2 - HYPER Exoskeleton

I. INTRODUCTION

This document explains the electronics hardware and software related to H2 exoskeleton. It is targeted to be an intern document and is not intended to be a user's manual.

II. ELECTRONIC HARDWARE

The electronic hardware of H2 consists of three main parts:

i) A centralised HAL unit with two customized electronic boards: H2-HAL-ARM and H2-HAL-COM. H2-HAL-ARM contains an ARM (Advanced Risk Machine) microcontroller running the control algorithms, and H2-HAL-COM is intended for wireless communications with external devices.

ii) Six customized electronic drivers (H2-Joint1-6) with an ARM microcontroller that drive the DC brushless motors;

iii) A physical communication network that guarantees strict determinism, data collision avoidance and optimized data transfer for small data packets between H2-ARM and H2-Joint1-6 boards.

III. H2-HAL ARM and COM BOARDS

H2-HAL-ARM board (the H2 main board) was designed specifically for real-time control of the whole H2 exoskeleton. It interacts to H2-Joint1-6 boards acquiring sensors information and controlling the actuators. The small size of this board (only 57 x 43 mm) and very low power consumption allows it to be placed on the exoskeleton structure, reducing the bulk, complexity and difficulty of wiring, in addition to minimize connections. Moreover, it eliminates the need of a backpack been carried by the user.

H2-HAL-ARM computational power relies on a ST Microelectronics ARM microcontroller STM32F405RG running at 168 MHz. The board has three independent CAN (Control Area Network) transceiver channels of communications: two are used to connect with all the six H2-Joint boards (one for each leg), exchanging information and commanding the six joint's actuators. The other channel is intended to connect to external systems that can control H2 in real time.

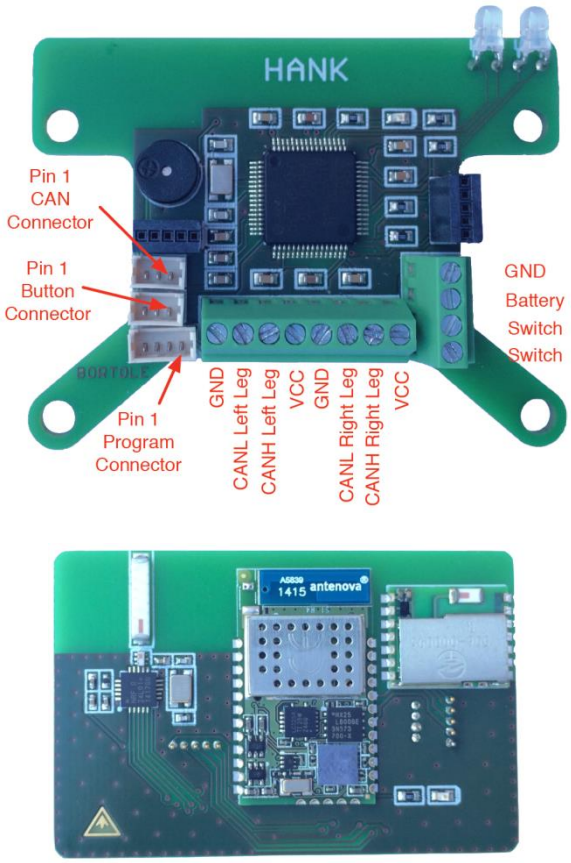


Fig. 1. H2-HAL ARM and COM boards.

H2-HAL-COM board, connected on top of H2-HAL-ARM, is designed to provide wireless communication with external devices. Board size is 53 mm x 36 mm.

The board also has three wireless communications ports: Bluetooth, Wi-Fi, and 2.4 GHz connection with proprietary protocol. Bluetooth communication can connect to a user interface. Wi-Fi link is used to connect to external systems to visualize joint data. 2.4 GHz connection is designed to interact with other hardware devices in the environment.

H2 architecture is powered by a Li-ion battery of 22.5 VDC nominal voltage. A switched regulator with high efficiency is designed to generate 3.3 VDC to power H2-HAL boards. Two LEDs are used to report H2 state. When H2 turn on LED 1 blinks red, LED 2 blinks orange and finally LED 1 turns on green. LED 2 will turn on green if all 6 joints are connected or orange otherwise. LED 1 turns red if a communication error occurs or if any joint disconnects, and the power for the joints is turned off. If this occurs, H2-ARM has to be

restarted with a power cycle. Figure 1 depicts H2-HAL boards and its connections.

H2-ARM runs an algorithm that receives commands to control each joint independently. The joints are numbered as follows: Right Hip = 1; Right Knee = 2; Right Ankle = 3; Left Hip = 4; Left Knee = 5; Left Ankle = 6. Three types of control are possible for each joint: position, stiffness and torque. The commands are received by CAN. Details about commands format are given in the specific sections of communication.

Moreover, H2-ARM runs an algorithm for walking, based on a pre-recorded gait pattern. This gait can be controlled using Bluetooth commands, explained in the Bluetooth Communication section.

A. Bluetooth Communication

Bluetooth communication is intended to be used with wireless interfaces. H2 receives commands in byte format from Bluetooth devices. Table I summarizes commands accepted and its functions. Byte values are in decimal format.

TABLE I BLUETOOTH BYTE COMMANDS.

Byte Value	Command
0	Stop gait
1	Gait at speed 1
2	Gait at speed 2
3	Gait at speed 3
4	Gait at speed 4
5	Gait at speed 5
6	Gait at speed 6
7	Gait at speed 7
8	Gait at speed 8
9	Gait at speed 9
10	Gait at speed 10
11	Disable joints mode
12	Reserved for future use
13	Not implemented
14	Not implemented
15	Start Wi-Fi data
16	Stop Wi-Fi data
17	Start CAN data
18	Stop CAN data
19	Enable sound
20	Disable sound

H2 has a pre-recorded gait pattern in its memory. Bluetooth commands from 0 to 10 can stop or start the replication of this gait at different speeds. The decimal value 11 disables all H2 joints. Values from 15 to 18 start or stop sending data via Wi-Fi and/or CAN. Values 19 and 20 enables/disables sound.

If Bluetooth sending data has been started, H2 will send 6 bytes every 200 milliseconds regarding its internal data. Table II summarizes the data sent.

TABLE II BLUETOOTH DATA FRAMES SENT BY H2.

Byte 1	Exo state
Byte 2	Battery voltage
Byte 3	Running time – hours
Byte 4	Running time – minutes
Byte 5	Running time - seconds
Byte 6	255 (End of frame)

Exo state values represent the actual gait, where 0 means H2 stopped; values from 1 to 10 represent speed; and 11 means disabled. Battery voltage value is given without decimal separator (for example, 245 means 24.5 volts).

B. User Interface

Figure 2 depicts a screen of the user interface developed to test H2 in a simple way. It was developed for an Android smartphone. With 5 main buttons the user can control the gait speed in 10 different levels. Other 2 secondary buttons are used to start or stop the data stream sent by H2 via Wi-Fi or CAN. One button enables/disables sound.

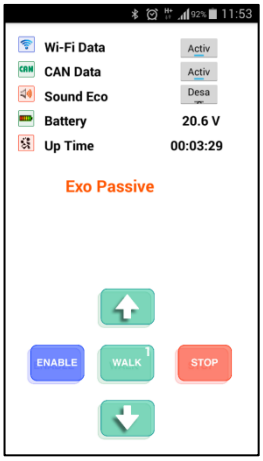


Fig. 2. H2 smartphone interface for Bluetooth communication.

Before trying to command H2 by Bluetooth, the user has to pair H2 Bluetooth interface with the smartphone (or laptop, etc.). The sequence for this should be: turn on Bluetooth on your device and look for near devices; pair with H2M; if a passcode is needed, try “1234”.

After successfully paired, you are ready for communicating with H2.

C. Wi-Fi Communication

H2 Wi-Fi port is intended to communicate with external devices like a laptop. When turned on, H2 creates an open Wi-Fi spot named H2M. Simply connect a computer in this network and you be able to receive data via UDP from H2. Your device should receive the following IP automatically: 192.168.1.2. H2 is the network host and has the following parameters: IP: 192.168.1.1; Subnet mask: 255.255.255.0; Port: 2000. H2 will send data to the following device: IP: 192.168.1.2; Subnet mask: 255.255.255.0; Port: 3000.

If Wi-Fi data has been started, H2 will send 35 bytes every 10 milliseconds regarding its internal data. Table III summarizes the data sent.

TABLE III. WI-FI DATA FRAMES SENT BY H2.

Byte 1	115 (Start of frame)
Byte 2	Right hip angle
Byte 3	Right knee angle
Byte 4	Right ankle angle
Byte 5	Left hip angle
Byte 6	Left knee angle
Byte 7	Left ankle angle
Byte 8	Right hip gauge torque
Byte 9	Right knee gauge torque
Byte 10	Right ankle gauge torque
Byte 11	Left hip gauge torque
Byte 12	Left knee gauge torque
Byte 13	Left ankle gauge torque
Byte 14	Right hip motor torque
Byte 15	Right knee motor torque
Byte 16	Right ankle motor torque
Byte 17	Left hip motor torque
Byte 18	Left knee motor torque
Byte 19	Left ankle motor torque
Byte 20	Right heel FSR
Byte 21	Right toe FSR
Byte 22	Left heel FSR
Byte 23	Left toe FSR
Byte 24	Time stamp – byte 0
Byte 25	Time stamp – byte 1
Byte 26	Time stamp – byte 2
Byte 27	Time stamp – byte 3
Byte 28	Battery voltage
Byte 29	Reserved for future use
Byte 30	Reserved for future use
Byte 31	Reserved for future use
Byte 32	Reserved for future use
Byte 33	Reserved for future use
Byte 34	Reserved for future use
Byte 35	120 (End of frame)

Angles are given in degrees. Torque are given in Nm. Foot switch information goes from 0 to 255, Battery voltage value is given without decimal separator (for example, 245 means 24.5 volts).

D. CAN Communication

H2 CAN port is intended to communicate with external devices in real time. The bus speed is set to 1 Mbps and it accepts messages in standard format with packets of 6 bytes. Table IV summarizes commands accepted and its functions. Byte values are in decimal format.

The command Joint Control can be used to control each one of the six joints independently. Motor ID values are: 1 = Right Hip; 2 = Right Knee; 3 = Right Ankle; 4 = Left Hip; 5 = Left Knee; 6 = Left Ankle. For type of control, it has the following values: 1 = Position control; 2 = Stiffness control; 3 = Torque control; 4 = Motors disabled; 5 = Motors stopped. When Position Control is used, byte 3 is the set point for that joint and bytes 4, 5 and 6 are not used. For Torque Control, byte 3 is the set point for that joint and bytes 4, 5 and 6 are not used. Stiffness Control uses byte 3 as the set point for position and byte 4 as the percentage of stiffness for that joint (where the value 0 means no stiffness and the value 100 means the maximum possible stiffness).

The commands Min Angles Accepted and Max Angles Accepted can be used to set the minimum and maximum angles accepted as set point for Position Control.

The command Start/Stop CAN Data is used to start or stop sending data via CAN (byte 1 = 1 starts data; byte 1 = 0 stops data).

If CAN data has been started, H2 will send 3 messages with 6 bytes each every 10 milliseconds, regarding its internal data. Table V summarizes the data sent.

Angles are given in degrees. Torque is given in Nm. Foot FSR information is from 0 to 255. Battery voltage value is given without decimal separator (for example, 245 means 24.5 volts).

TABLE IV. CAN MESSAGES COMMANDS.

Message ID = 70	**Joint Control**
Byte 1	Motor ID
Byte 2	Type of control
Byte 3	Position/torque set point
Byte 4	Stiffness set point
Byte 5	Reserved for future use
Byte 6	Reserved for future use
Message ID = 75	**Min Angles Accepted**
Byte 1	Right hip min angle
Byte 2	Right knee min angle
Byte 3	Right ankle min angle
Byte 4	Left hip min angle
Byte 5	Left knee min angle
Byte 6	Left ankle min angle
Message ID = 80	**Max Angles Accepted**
Byte 1	Right hip max angle
Byte 2	Right knee max angle
Byte 3	Right ankle max angle
Byte 4	Left hip max angle
Byte 5	Left knee max angle
Byte 6	Left ankle max angle
Message ID = 85	**Start/Stop CAN Data**
Byte 1	Start/Stop CAN Data
Byte 2	Reserved for future use
Byte 3	Reserved for future use
Byte 4	Reserved for future use
Byte 5	Reserved for future use
Byte 6	Reserved for future use

IV. H2-JOINT1-6 BOARDS

Each one of the six H2's joints is equipped with a H2-Joint board (numbered from 1 to 6). Each board is in charge of data acquisition of the different joint's sensors: angular position, interaction torque, motor torque, joint speed and foot-ground contact. H2-Joint1-6 contain all the circuitry of the analog filters for each joint sensor and also the amplifiers for the strain gauges. After filtering and amplifying process, the signals are digitalized by an ARM microcontroller (STM32F302K8). A small data packet of six bytes aggregates the sensor's information on each joint and these data is sent to H2-ARM controller every one millisecond.

TABLE V. CAN DATA FRAMES SENT BY H2.

Message ID = 110	**Joint Angle**
Byte 1	Right hip angle
Byte 2	Right knee angle
Byte 3	Right ankle angle
Byte 4	Left hip angle
Byte 5	Left knee angle
Byte 6	Left ankle angle
Message ID = 120	**Joint Torque**
Byte 1	Right hip gauge torque
Byte 2	Right knee gauge torque
Byte 3	Right ankle gauge torque
Byte 4	Left hip gauge torque
Byte 5	Left knee gauge torque
Byte 6	Left ankle gauge torque
Message ID = 130	**Foot Switch**
Byte 1	Right heel FSR
Byte 2	Right toe FSR
Byte 3	Left heel FSR
Byte 4	Left toe FSR
Byte 5	Battery voltage
Byte 6	Reserved for future use
Message ID = 140	**Motor Torque**
Byte 1	Right hip motor torque
Byte 2	Right knee motor torque
Byte 3	Right ankle motor torque
Byte 4	Left hip motor torque
Byte 5	Left knee motor torque
Byte 6	Left ankle motor torque

The brushless DC motor's drives are embedded directly into the H2-Joint1-6 boards. The drive, developed specifically for this application, is lightweight, compact and receives digital set points. It is small enough to be mounted directly to the motor side in the exoskeleton's frame. This approach decreases the amount of electromagnetic noise and the number of wires in the exoskeleton. Figure 3 depicts a scheme of H2-Joint1-6 board's architecture. In figure 4, H2-Joint board and its connections are shown.

V. H2 NETWORK STRUCTURE

H2-ARM and H2-Joint1-6 boards communicate via a deterministic real-time network based on CAN technology. Through this digital bus working at 1 Mbps, each H2-Joint1-6 is connected to the main controller H2-ARM. The network is flexible in terms of configuration and automatically avoids data collision and corrects errors regarding to data packets transmission.

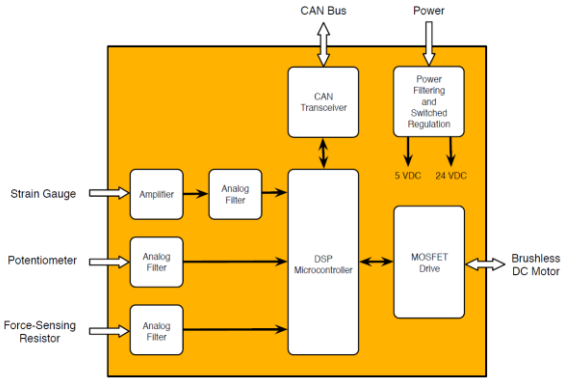


Fig. 3. H2-Joint1-6 board's architecture used in each exoskeleton joint. The objective is two folded: acquisition of the different joint's sensors (angular position, interaction torque, joint speed and foot ground contact) and drive the brushless DC motors.

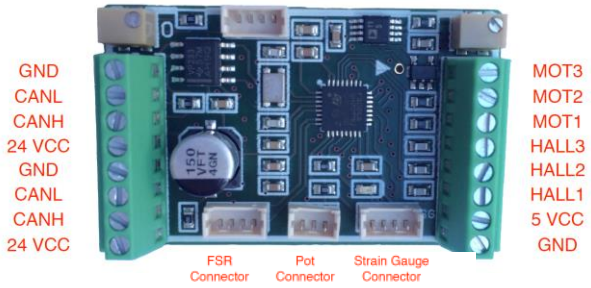


Fig. 4. H2-Joint board that powers each H2 joint and its connections.

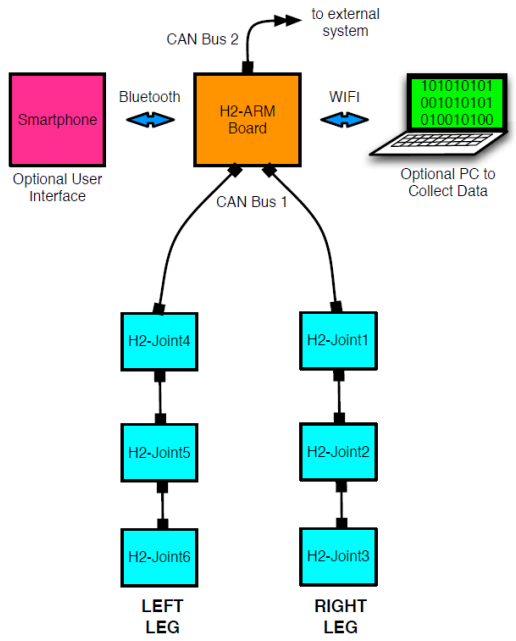


Fig. 5. H2's control architecture overview. All sensors in both legs are connected to the H2-Joint1-6 boards, that communicate to H2-ARM board through a deterministic real-time network.

Each communication cycle in the network protocol involves passing a message from the H2-ARM node to all H2-Joint1-6 nodes in the network. As the message travels through the bus, each H2-Joint1-6 reads its assigned actuator command data byte (by looking for its own ID number and message byte sequence). Then, each H2-Joint1-6 returns one message back to H2-ARM node with its locally collected sensor's data.

Because the communication cycles occur at a fixed rate (1 kHz) set by the control scheme, this protocol allows for deterministic control. Also, it provides built-in network error detection because at every message received, each H2-Joint1-6 has to return data information to H2-ARM. In this way, H2-ARM has a strong way to determine the integrity of the network and the correct operation of the joint's actuators. Figure 5 depicts an overview of H2's general control architecture.

Apéndice B

Fichero de configuración del exoesqueleto H2

```
#KEY_OR_TRIGGER= KEYBOARD(0) TRIGGER_PIN(1) BOTH(2)
#SAMPLES_PER_SECOND MUST NOT BE OVER 45 (APROXIMATION)
#RECONFIG MUST BE TRUE TO SEND NEW CAN FRAMES
#######################################################
RECONFIG= no
NUM_DIGITALS= 1
NUM_ANALOGS= 1
KEY_OR_TRIGGER= 0
STOP_TRIGGER= 0
DIGITAL_SAMPLES_PER_SECOND= 10
ANALOG_SAMPLES_PER_SECOND= 10
#######################################################
MOTOR_ID= 255
TYPE_OF_CONTROL= 255
POSITION/TORQUE_SET_POINT= 255
STIFFNESS_SET_POINT= 255
#######################################################
RIGHT_HIP_MIN_ANGLE= 255
RIGHT_KNEE_MIN_ANGLE= 255
RIGHT_ANKLE_MIN_ANGLE= 255
LEFT_HIP_MIN_ANGLE= 255
LEFT_KNEE_MIN_ANGLE= 255
LEFT_ANKLE_MIN_ANGLE= 255
#######################################################
RIGHT_HIP_MAX_ANGLE= 255
RIGHT_KNEE_MAX_ANGLE= 255
RIGHT_ANKLE_MAX_ANGLE= 255
LEFT_HIP_MAX_ANGLE= 255
LEFT_KNEE_MAX_ANGLE= 255
LEFT_ANKLE_MAX_ANGLE= 255
#######################################################
```

START/STOP_CAN_DATA= 255
##

Apéndice C

Impacto social, económico, ético y ambiental

Tras el desarrollo de este Trabajo de Fin de Grado, se ha realizado un análisis del impacto que tiene el proyecto sobre la economía, la sociedad, la ética y el medio ambiente. Todos ellos aportan un valor añadido al trabajo realizado.

- **Impacto social:** este proyecto tendrá un impacto directo sobre los pacientes con lesión medular del Hospital Nacional de Parapléjicos de Toledo. Estos pacientes podrán beneficiarse de unas sesiones de rehabilitación más personalizadas. También contarán con un análisis más profundo de los datos obtenidos en las pruebas que realizasen. El personal médico podría comparar los datos con los de otros pacientes y así poder predecir el desarrollo el paciente en cuestión. Se ha demostrado en este Trabajo Fin de Grado que se puede realizar dicha comparativa así como agilizar tanto los datos de las pruebas como el análisis de los mismos.

- **Impacto económico:** el análisis de los datos obtenidos mediante este sistema electrónico permitirá un estudio más exhaustivo y, por tanto, reduciría los costes de tratamientos posteriores mediante la detección de posibles problemas durante la rehabilitación del paciente. Además de lo comentado anteriormente, la mejora del proceso de rehabilitación tendrá como consecuencia una reducción del tiempo del mismo y, por consiguiente, menor gasto en recursos humanos.

- **Impacto ético:** este proyecto no tiene mayor impacto ético que el respeto a la integridad de los pacientes con lesión medular en el desarrollo completo de su tratamiento.

- **Impacto ambiental:** el impacto de este proyecto sobre el medio ambiente es indirecto, ya que, como se ha dicho anteriormente, los periodos de rehabilitación pueden acortarse. Esta reducción de tiempo provocará un menor número de viajes de muchos de los pacientes al hospital, reduciendo la cantidad de CO2 emitido por los coches en los que se desplazan.

Apéndice D

Presupuesto económico

Este Trabajo de Fin de Grado ha sido desarrollado durante ocho meses con el departamento de Tecnología Fotónica y Bioingeniería en el Laboratorio de Robótica y Control (Robolabo) usando sus recursos. Se ha calculado el presupuesto económico teniendo en cuenta costes en recursos humanos y en recursos materiales.

- **Costes en recursos humanos**

  Los costes en recursos humanos se han calculado a partir de un coste por hora aproximado y de las horas trabajadas por el jefe de proyecto y el estudiante de ingeniería.

Tabla D.1: Costes en recursos humanos

	Coste por hora (€)	**Horas**	**Coste total (€)**
Ingeniero - Jefe de proyecto	22	50	1,100
Estudiante de Ingeniería	15	480	7,200
TOTAL			**8,300€**

- **Costes en recursos materiales**

  En este caso, los costes han sido calculados a partir de su precio de coste y la amortización sufrida por los recursos materiales en el tiempo del desarrollo del proyecto. Para la electrónica de consumo se ha tenido en cuenta una amortización de 1 año, para los ordenadores e impresoras 5. En el caso de las piezas mecánicas y cables se ha tenido en cuenta que es un gasto puntual sin amortización.

Tabla D.2: Costes en recursos materiales

	Vida útil (años)	Unidades	Coste (€)	Amortización (€/mes)	Tiempo usado (mes)	Coste total (€)
Raspberry Pi 3 B+	1	1	35.99	3	8	24
PiCAN2 Duo	1	1	59.05	4.92	8	39.37
ADC Pi	1	1	21.51	1.79	8	14.34
USB-CAN	1	1	22.76	1.90	2	3.80
Piezas mecánicas y cables	-	-	20.24	-	3	20.24
Impresora 3D	5	1	160	2.66	8	21.28
Ordenador portátil	5	1	700	11.66	8	93.33
TOTAL						**216.36**

Teniendo en cuenta ambos costes, el coste total de este Trabajo de Fin de Grado ha sido **8,516.36€**.

Apéndice E

Manual de Usuario - Exo Beta v1.0

Manual de Usuario - Exo Beta v1.0

Álvaro Gutiérrez Tenorio

E.1. Introducción

El sistema para el que Exo Beta ha sido diseñado es un sistema Raspberry Pi 3B+. Raspberry Pi 3 B+ es un ordenador de bajo coste y tamaño reducido que se puede conectar a un monitor y manejarse mediante un ratón y teclado estándar. Cuenta con un procesador BCM2837B0 Cortex-A53, de la familia Broadcom, a 1.4 GHz; una memoria RAM de 1GB, un módulo Wireless LAN que cumple las especificaciones definidas por IEEE 802.11.b/g/n/ac y un módulo Bluetooth 4.2.

El sistema cuenta con una distribución basada en Debian, Raspbian. Este sistema operativo es muy estable y cuenta con un gran soporte de la comunidad.

En este manual se muestran las diferentes formas de ejecutar la captura de datos mediante el programa Exo Beta. Para la versión 1.0 existen dos posibles maneras de ejecutar el programa: mediante terminal o mediante acceso a la página web.

Antes de la ejecución del programa, se deben configurar algunos aspectos del sistema que se explican a continuación.

E.2. Conexión del sistema a una red

El sistema se puede conectar a una red mediante Ethernet o WiFi.

E.2.1. Conexión Ethernet

En primer lugar, se debe conectar el sistema a través de un cable Ethernet al equipo desde el que se va a efectuar el acceso. Una vez conectado el cable, el sistema se identificará con el equipo y adoptará una dirección IP automáticamente.

Figura E.1: Conector para cable Ethernet.

E.2.2. Conexión WiFi

Para conectar el sistema a una red inalámbrica será necesario haber realizado antes una conexión Ethernet (ver Sección E.2.1) o acceder al entorno gráfico del sistema mediante HDMI. Si se conecta el sistema a un monitor mediante HDMI, podremos acceder a la interfaz gráfica. En la parte superior derecha está el icono de WiFi. Simplemente tenemos que pulsarlo y seleccionar «Encender Wifi». Ahora, en la lista de redes disponibles seleccionamos la red WiFI a la que se tiene que conectar el sistema, y se pulsa sobre ella para conectar.

También se puede configurar la conexión mediante la consola de comandos. En primer lugar, se listarán las redes disponibles al alcance del sistema mediante el siguiente comando:

```
sudo iwlist wlan0 scan
```

Una vez se sepa la red y contraseña a la que se pretende conectar el sistema, se debe editar el fichero /etc/wpa_supplicant/wpa_supplicant.conf con el comando:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

En este fichero, se añadirá lo siguiente al final del archivo, cambiando los datos por los de la red WiFi correspondiente.

```
network={
ssid="nombre-de-tu-wifi"
psk="password-de-tu-wifi"
key_mgmt=WPA-PSK
}
```

Quedando finalmente el fichero de la siguiente forma:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=ES

network={
ssid="nombre-de-tu-wifi"
psk="password-de-tu-wifi"
key_mgmt=WPA-PSK
}
```

Por último, se reinicia la Raspberry mediante el comando:

```
sudo reboot
```

E.3. Identificar dirección IP

La dirección IP del sistema se puede ver conectando el sistema a un monitor mediante HDMI. En la parte superior derecha del escritorio aparecerá el símbolo correspondiente a conexión por cable. Si posamos el ratón sobre este, aparecerá una ventana con la dirección IP del sistema. Otra forma de verla es mediante la consola de comandos, ejecutando el siguiente comando:

```
ifconfig wlan0
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.168  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6ef1:23bc:9f88:6783  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:69:9a:70  txqueuelen 1000  (Ethernet)
        RX packets 5368  bytes 292280 (285.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10632  bytes 14577872 (13.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura E.2: Captura de los datos de red de la Raspberry Pi.

Como se puede ver en la Figura E.2, la dirección IP aparece en el valor correspondiente a inet.

E.4. Acceso remoto al sistema

El acceso al sistema se puede hacer principalmente por dos métodos: una conexión vía SSH o mediante la herramienta *Conexión a Escritorio remoto*.

E.4.1. Conexión vía SSH

Desde un sistema Linux: para realizar una conexión vía SSH, sólo es necesario abrir la consola de comandos y ejecutar

```
ssh pi@IP_DEL_SISTEMA_A_CONECTAR
```

Desde un sistema Windows: será necesario descargar una herramienta de conexión vía SSH como SmarTTY o PuTTY. Se configuran los datos del sistema en estas herramientas y se conectará a la Raspberry Pi.

Las credenciales del sistema son las siguientes:

```
Usuario: pi
Clave: 1234
```

En ambos sistemas operativos la conexión es a la consola de comandos y no a la interfaz gráfica.

E.4.2. Conexión a Escritorio remoto

Esta herramienta sólo se encuentra en sistemas Windows.

Se abrirá el buscador de Windows y se escribirá *Conexión a Escritorio remoto*, y se ejecutará dicha herramienta. Tras ello, en la ventana que aparece se debe introducir la dirección IP del sistema (ver Sección E.3). Posteriormente pedirá los datos de autentificación, y se deberán rellenar con las siguientes credenciales:

```
Usuario: pi
Clave: 1234
```

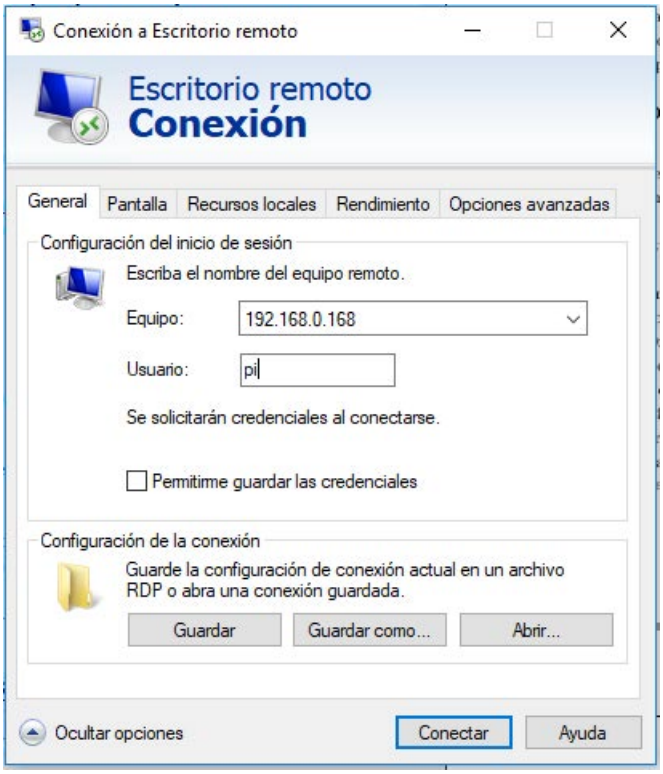


Figura E.3: Captura del programa Conexión a Escritorio remoto.

Finalmente, emergerá en una ventana el entorno gráfico del sistema.

E.5. Configuración de los parámetros de captura

Antes de comenzar la captura de datos, es necesario configurar todos los parámetros para la captura.

A continuación, se describe la utilidad de cada uno de los parámetros a configurar:

- **RECONFIG**: si se quiere reconfigurar los parámetros del exoesqueleto Exo-H2, el valor de este parámetro debe ser *TRUE* y en caso contrario, *FALSE*.
- **NUM_DIGITALS**: este es el número de entradas digitales que se quiere capturar. Su valor mínimo es 1 y el máximo 10.
- **NUM_ANALOGS**: este es el número de entradas analógicas a capturar. Siendo 1 el valor mínimo y 8 el máximo.
- **KEY_OR_TRIGGER**: esta opción define si la captura de datos se controlará desde el teclado o desde un pin de entrada, que actuará como disparador. Si el valor es 0, se activa el teclado. Si el valor es 1, se activa el disparador. Y si el valor es 2, se activarán ambos.
- **STOP_TRIGGER**: sólo se podrá activar si la opción *KEY_OR_TRIGGER* está en modo disparo o ambas. Si su valor es 1, se podrá pausar la captura de los datos mediante una entrada digital que actua como disparador. Si su valor es 0, esta opción será desactivada.

- **DIGITAL_SAMPLES_PER_SECOND**: su valor indicará el número de muestras digitales que se tomarán por segundo.
- **ANALOG_SAMPLES_PER_SECOND**: su valor indicará el número de muestras analógicas que se tomarán por segundo.

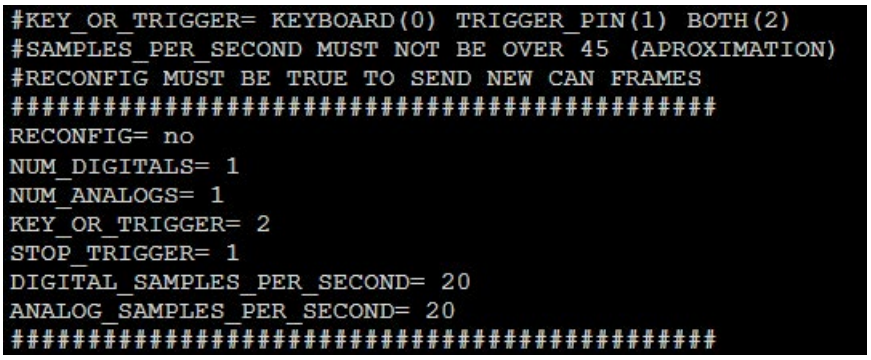


Figura E.4: Captura de los parámetros principales de configuración.

El resto de parámetros de configuración son valores entre 0 y 255. La selección de estos debe ser decidida en base a la configuración deseada del exoesqueleto H2 (ver Manual del Usuario del Exo-H2).

E.6. Ejecución mediante terminal

Para ejecutar la captura mediante comandos en el terminal de la Raspberry Pi, antes se debe acceder al sistema mediante escritorio remoto o conexión SSH. En primer lugar, será necesario estar conectado a la misma red que la Raspberry Pi (ver Sección E.2). Tras ello, hay que identificar la dirección IP que tiene el sistema, como se explica en la Sección E.3 y acceder al sistema siguiendo las instrucciones de la Sección E.4.

Una vez se ha accedido a la consola de comandos, para ejecutar el programa sólo será necesario introducir el siguiente comando:

```
/home/pi/Exo_beta/bin/candump
```

O desplazarse al directorio donde está el programa y ejecutarlo después:

```
cd /home/pi/Exo_beta/src/
./candump
```

Antes de ejecutarlo, es necesario configurar los parámetros de captura mediante el comando:

```
nano /home/pi/Exo_beta/src/reconfig.txt
```

En este archivo se configurarán todos los parámetros como se explica en la Sección E.5.

Tras la configuración de los parámetros, se ejecuta el programa y aparecerá un mensaje como el de la Figura E.5.

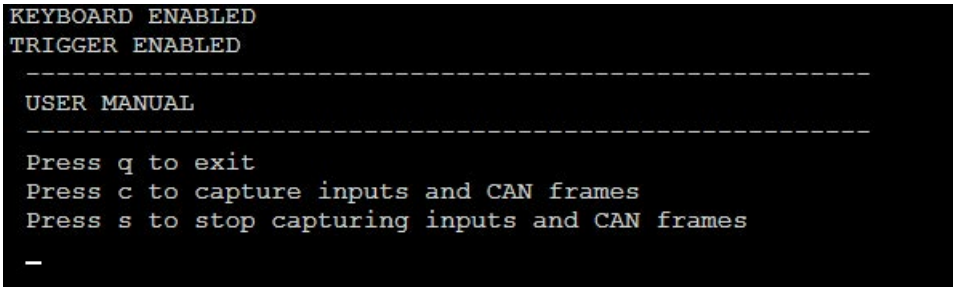


Figura E.5: Captura de bienvenida al programa.

Para iniciar la captura de datos, como se muestra en la Figura E.5, se debe pulsar la tecla c. O si se ha configurado la opción de *TRIGGER* previamente, mediante un flanco de subida de tensión en la entrada digital *TRIGGER*.

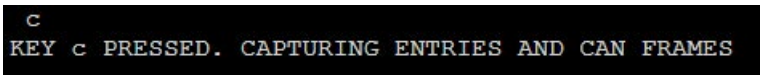


Figura E.6: Captura de datos iniciada.

Una vez se ha iniciado la captura, para pararla podemos pulsar la tecla s, también para pausarla y poder volver a empezar la captura en otro momento (ver Figura E.7), o se puede pulsar la tecla q, para salir del programa y terminar la captura definitivamente (ver Figura E.8).

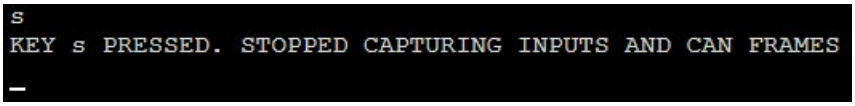


Figura E.7: Captura de datos pausada.

Una vez terminada la captura, se pueden acceder a los datos desplazándose al directorio *data*.

```
cd /home/pi/Exo_beta/data/
```

Ya en ese directorio, se deberá seleccionar la carpeta correspondiente al día en el que se ha producido la captura. Y tras ello, la hora en la que se realizó.

```
cd directorio_de_fecha/directorio_de_hora
```

En esta carpeta verán los ficheros correspondientes a todos los datos capturados, entre ellos los datos digitales, analógicos y los mensajes CAN. Todos estos ficheros se pueden ver con el comando:

```
less fichero_de_datos
```

```
q
KEY q PRESSED. EXITING...
/home/pi/Exo_beta/src/parse /home/pi/Exo_beta/data/17-06-2019/20_14_
34/candump_110.csv /home/pi/Exo_beta/data/17-06-2019/20_14_34/ -1
File /home/pi/Exo_beta/data/17-06-2019/20_14_34/candump_110.csv pars
ed
/home/pi/Exo_beta/src/parse /home/pi/Exo_beta/data/17-06-2019/20_14_
34/candump_120.csv /home/pi/Exo_beta/data/17-06-2019/20_14_34/ -2
File /home/pi/Exo_beta/data/17-06-2019/20_14_34/candump_120.csv pars
ed
/home/pi/Exo_beta/src/parse /home/pi/Exo_beta/data/17-06-2019/20_14_
34/candump_130.csv /home/pi/Exo_beta/data/17-06-2019/20_14_34/ -3
File /home/pi/Exo_beta/data/17-06-2019/20_14_34/candump_130.csv pars
ed
/home/pi/Exo_beta/src/parse /home/pi/Exo_beta/data/17-06-2019/20_14_
34/candump_140.csv /home/pi/Exo_beta/data/17-06-2019/20_14_34/ -4
File /home/pi/Exo_beta/data/17-06-2019/20_14_34/candump_140.csv pars
ed
/home/pi/Exo_beta/src/parse /home/pi/Exo_beta/data/17-06-2019/20_14_
34/candump-2019-06-17_201434.csv /home/pi/Exo_beta/data/17-06-2019/2
0_14_34/ -5
File /home/pi/Exo_beta/data/17-06-2019/20_14_34/candump-2019-06-17_2
01434.csv parsed
```

Figura E.8: Fin del programa.

E.7. Ejecución mediante acceso a la página web

Para conectarse al sistema mediante la página web, antes se deben haber seguido los pasos explicados en las Secciones E.2 y E.3.

Una vez esté el sistema conectado a una red y sepamos su dirección IP en ella, abriremos un explorador web e introduciremos la IP del sistema en la barra de direcciones. La página que aparecerá será la mostrada en la Figura E.9.

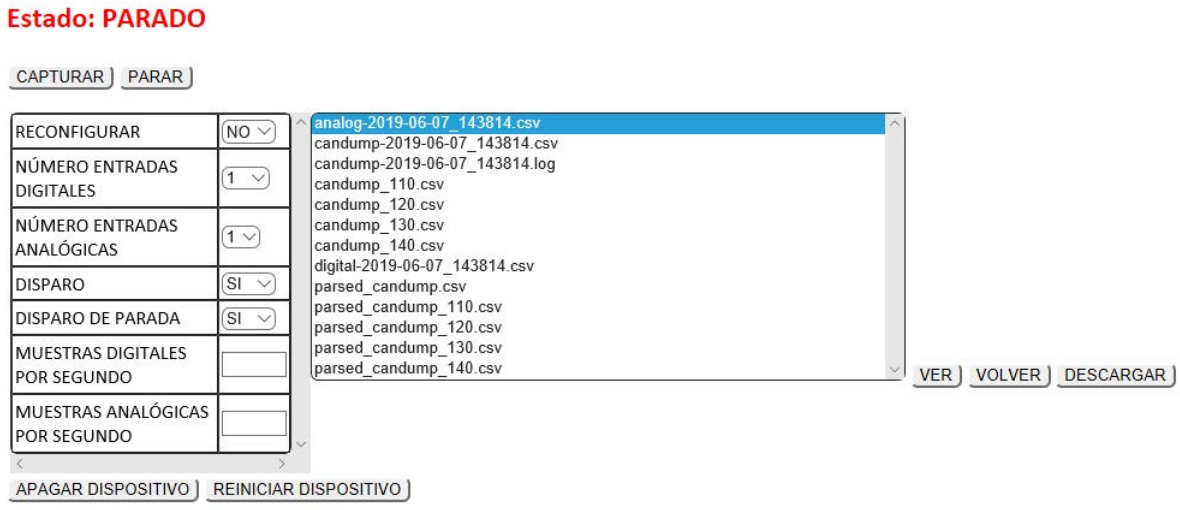


Figura E.9: Pantalla de bienvenida a la captura de datos en la página web.

La parte izquierda corresponde a los parámetros para configurar la captura que se explican en la Sección E.5. Una vez esté configurados, se pulsará el botón CAPTURAR para comenzar la captura. Tras ello la pantalla cambiará para mostrar que está capturando.

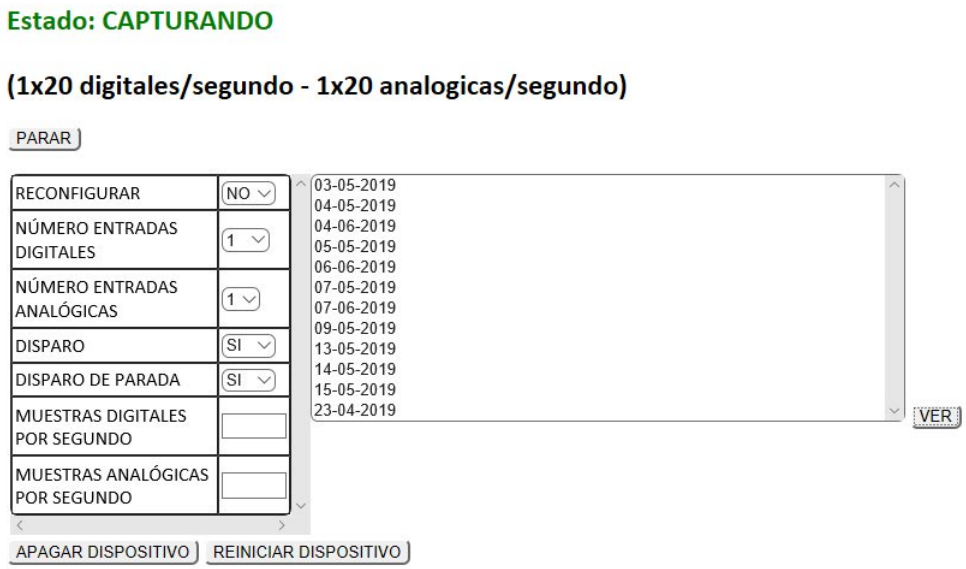


Figura E.10: Pantalla de de la captura en la página web.

En la parte superior izquierda de la Figura E.10, se puede ver que el sistema está capturando, que se muestran el número de entradas y la frecuencia a la que estas son capturadas. Además, el botón de CAPTURAR habrá desaparecido y la sólo se podrá pausar la captura mediante el botón PARAR.

Tanto si el sistema está parado como si está capturando, se puede explorar los datos obtenidos en capturas ya realizadas en la parte derecha de la pantalla. Para abrir un directorio, solo hay que seleccionar el deseado y pulsar el botón VER. Este directorio se abrirá y mostrará su contenido. Se deberá hacer lo mismo de nuevo y seleccionar la hora de los datos a visualizar. Por último, se podrá seleccionar el fichero de datos que se quiere visualizar y se mostrará.

Si se desea descargar alguno de los ficheros, se puede hacer mediante la pulsación del botón DESCARGAR, tras haber seleccionado el fichero que se desea.

Finalmente, en la parte inferior de la pantalla aparecen dos botones. Uno es el de APAGAR DISPOSITIVO, mediante el cuál se apaga el dispositivo y se dejará de tener acceso a la página web. El botón de REINICIAR DISPOSITIVO reinicia el mismo y tras un corto tiempo, se podrá volver a acceder al sistema. No se recomienda el uso de ninguno de los dos mientras se está realizando una captura de los datos.

Apéndice F

Manual del Desarrollador - Exo Beta v1.0

F.1. Introducción

El sistema de extracción de datos Exo Beta está formado por tres bloques tal y como se muestra en la Figura F.1:

- Raspberry Pi 3 B+ (con 10 entradas digitales).
- ADC Pi (con 8 entradas analógicas).
- PiCAN2 Duo (con 2 puertos Bus - CAN).

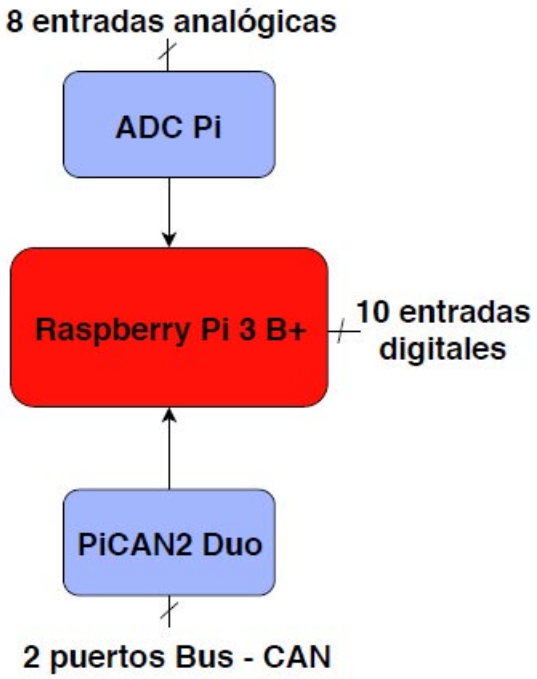


Figura F.1: Esquema de la arquitectura del sistema Exo Beta.

En este documento se proporciona información sobre el sistema Exo Beta. La Sección F.2 explica las características hardware del sistema. La Sección F.3 explica los principales bloques del software para capturar datos. A continuación, en la Sección F.5 explica el procedimiento para conectar el sistema a una red. Finalmente, se detalla el acceso al sistema de manera remota en la Sección F.6.

F.2. Hardware

Como se ha comentado en la Sección F.1, el sistema Exo Beta está formado por tres bloques. En esta Sección se explican las características de cada uno de ellos.

F.2.1. Raspberry Pi 3 B+

Raspberry Pi 3 B+ es un ordenador de tamaño reducido que cuenta con un procesador BCM2837B0 Cortex-A53, de la familia Broadcom, a 1.4 GHz. También tiene una memoria RAM de 1GB, un módulo Wireless LAN que cumple las especificaciones definidas por IEEE 802.11.b/g/n/ac, y un módulo Bluetooth 4.2 (ver Figura F.2).

Figura F.2: Imagen de la Raspberry Pi 3 B+.

El sistema cuenta con una distribución basada en Debian modificada para este sistema que se llama Raspbian. Se ha cargado este sistema operativo sobre una memoria SD y se ha introducido en la Raspberry Pi.

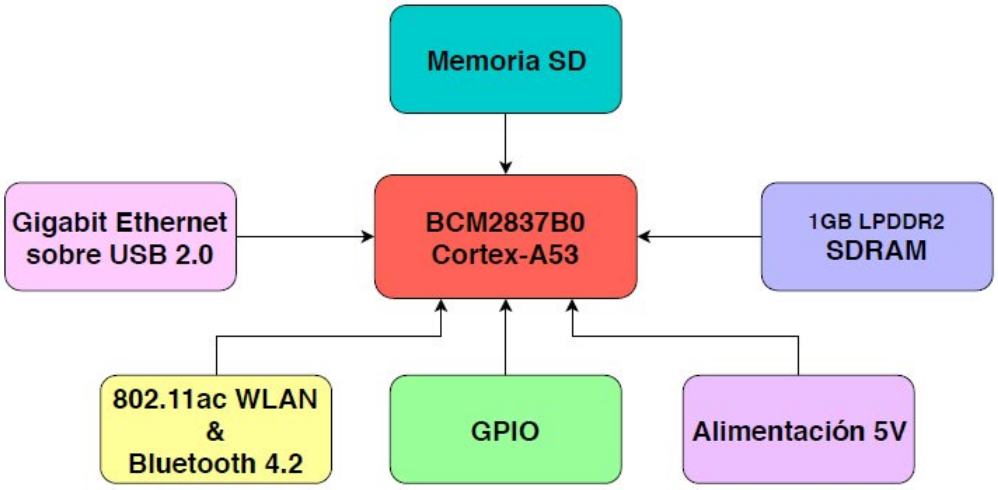


Figura F.3: Esquema de los componentes de la Raspberry Pi 3 B+.

En la Figura F.3, se muestran los seis bloques controlados por el Cortex A53 y en los que se divide la Raspberry Pi 3 B+:

- **Alimentación**: el sistema es alimentado con 5 voltios y 2.1 amperios.
- **RAM**: el sistema cuenta con una memoria RAM LPDDR2 de 1GB.
- **Memoria SD**: el sistema operativo está instalado sobre una memoria SD de 16GB.
- **GPIO**: 40 pines que se pueden utilizar como entradas y salidas.
- **Gigabit Ethernet**: máximo 300 Mbps sobre USB 2.0. También soporta Power over Ethernet.
- **802.11ac WLAN & Bluetooth 4.2**: módulos para la comunicación inalámbrica en la banda de 2.4GHz y 5GHz.

F.2.2. ADC Pi

El sistema ADC Pi es un convertidor analógico digital desarrollado por ABElectronics. Este convertidor cuenta con 8 entradas analógicas de 0 a 5 voltios que se cuantifican con 17 bits. La comunicación con la Raspberry Pi es a través del protocolo I2C. Las direcciones I2C se pueden seleccionar con unos jumpers que se encuentran en la placa del convertidor. Además, el sistema cuenta con un amplificador de ganancia programable de ganancias 1, 2, 4 u 8.

En la parte derecha de la Figura F.4 se pueden ver los pines para la selección de direcciones I2C. El fabricante proporciona 4 posibles configuraciones, que se muestran en la Figura F.5.

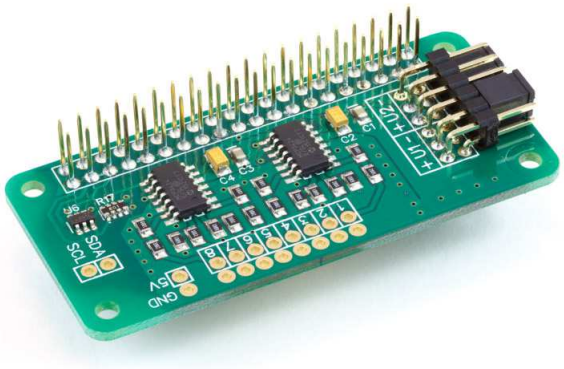


Figura F.4: ADC Pi de ABElectronics.

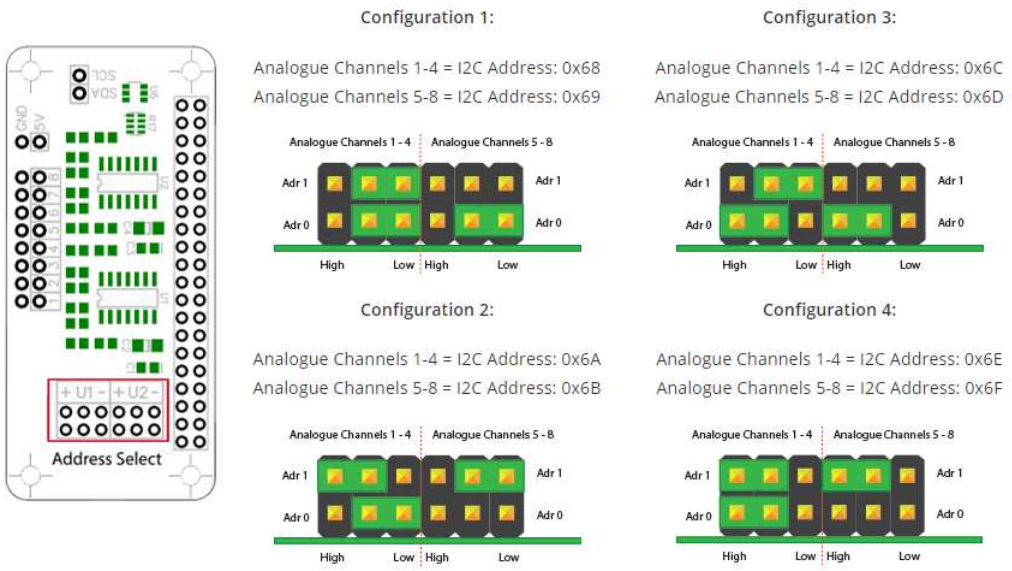


Figura F.5: Configuraciones para las direcciones I2C del ADC Pi.

F.2.3. PiCAN2 Duo

El sistema PiCAN2 Duo es un módulo de expansión para Raspberry Pi que permite al sistema comunicarse a través del protocolo CAN. Cuenta con el controlador CAN MCP2515 y el transceptor MCP2551 de Microchip.

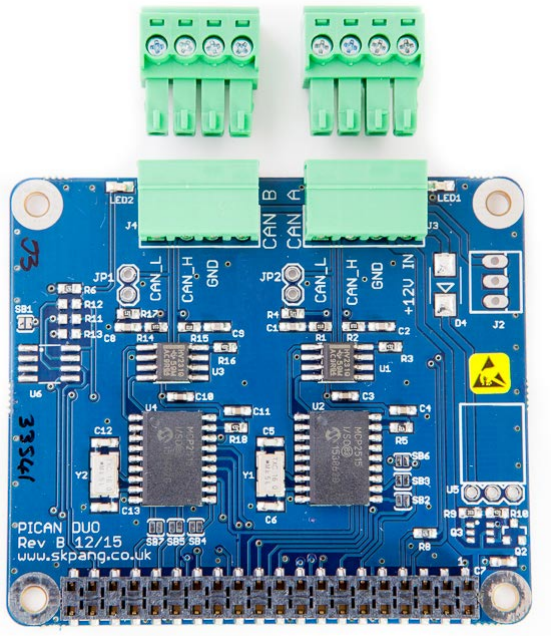


Figura F.6: Placa PiCAN2 Duo

En la Figura F.6, se puede ver que la placa cuenta con dos canales Bus-CAN que incluyen una terminación de 120 Ohmios cada una.

El sistema soporta el protocolo CAN hasta 1 Mbps. La comunicación con la Raspberry Pi es a través del protocolo SPI a 10 MHz. Esta comunicación utiliza los pines GPIO25 y GPIO24 para las interrupciones de recepción.

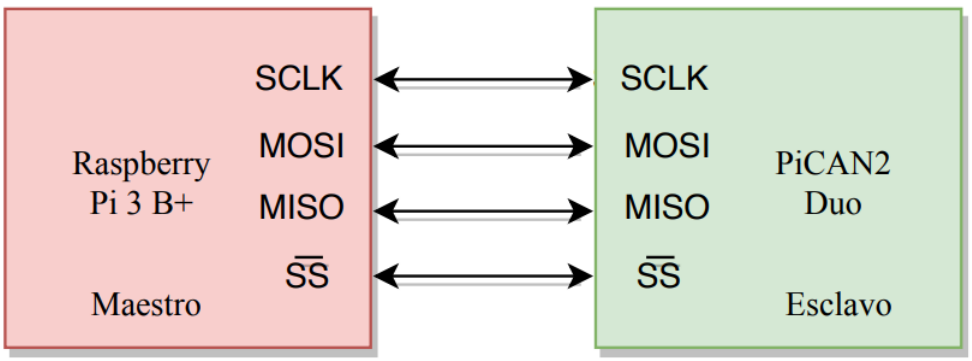


Figura F.7: Relación SPI entre Raspberry Pi y PiCAN2 Duo.

F.3. Software

El sistema cuenta con un programa principal para capturar datos desarrollado en lenguaje C. Este hace uso de diferentes librerías para el control de los módulos

externos a la Raspberry comentados en la Sección F.2. Además de para controlar dichos dispositivos, se utilizan otras librerías para controlar los tiempos de ejecución del programa y las interrupciones del teclado.

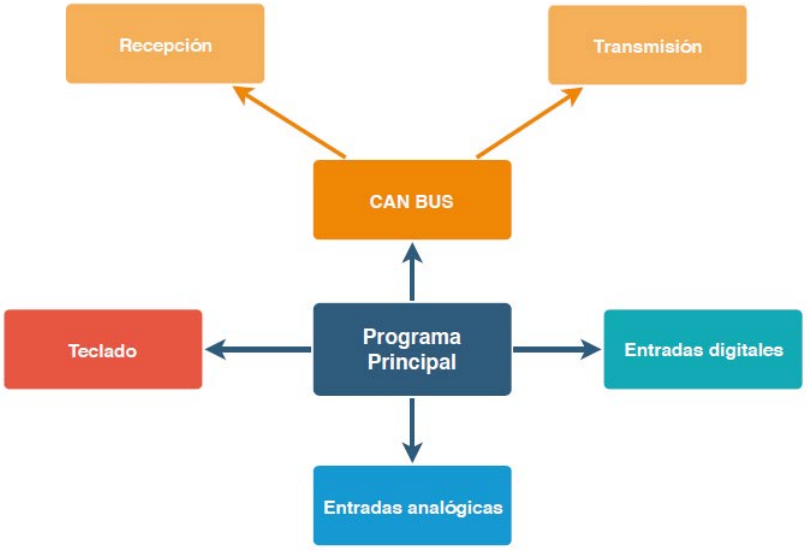


Figura F.8: Diagrama de bloques del programa principal.

El programa principal cuenta con 5 bloques principales como se ve en la Figura F.8:

- **Entradas digitales**: en primer lugar, se configuran las entradas digitales mediante la librería I y se capturan las entradas de forma periódica.
- **Entradas analógicas**: mediante la librería *ADC_Pi* proporcionada por el fabricante ABElectronics, se realiza la lectura periódica de la entrada.
- **Recepción CAN Bus**: mediante la librería *Candump* de *linux-can-utils* se captura todos los mensajes que se envían por el bus.
- **Transmisión CAN Bus**: mediante la librería *Cansend* de *linux-can-utils* se envía por el bus mensajes de configuración.
- **Atención al teclado**: mediante la librería *Conio* se atienden todas las pulsaciones que se producen en el teclado.

F.3.1. Librerías utilizadas

Las librerías que se utilizan en este programa son las siguientes:

- ***Conio***: esta librería proporciona acceso a la entrada y salida de la consola del sistema. En particular se usará el método *kbhit()*, que indica si se ha pulsado una tecla y cuál.

- ***ADC_Pi***: esta librería la proporciona el fabricante y se puede encontrar en la web de ABElectronics. De esta librería se usa en particular el método read_voltage(), que devuelve el valor de la tensión en la entrada seleccionada.

- ***Linux-can-utils***: esta librería se puede encontrar en el repositorio de GitHub de Linux.

- ***Tmr***: esta librería usa los timers POSIX del sistema para lanzar interrupciones tras un tiempo determinado. Se usarán los métodos *tmr_init()* y *tmr_start()*.

F.3.2. Programa para decodificar mensajes CAN

Ya que los mensajes CAN se han guardado en su propio formato, extraer la información necesaria de ellos es una tarea complicada. Por ello, se ha creado un pequeño programa que, siguiendo la documentación del H2, descodifica los mensajes recibidos según sea su identificador e imprime los valores en un nuevo fichero *parse_candump_XXX.csv*, donde XXX es el identificador recibido. De esta manera, se obtiene una tabla de valores significativos en un rango de 0 a 255, ya que la longitud máxima de estos es un byte.

Este programa, se ejecutará por defecto al terminar el programa principal. En caso de querer ejecutarse externamente, se deberá hacer de la siguiente manera:

```
$ ./parseCAN -fichero_a_traducir -directorio_destino -ID_CAN
```

De esta manera, al introducir el ID de los mensajes que están contenidos en el fichero, el programa dará un nombre a cada byte y comenzará a descodificar.

F.4. Sistema de ficheros

Los datos que se obtienen son guardados por separado en una serie de ficheros de formato CSV. Debido a que se va a utilizar el sistema de manera continuada y en varias sesiones, se ha creado un directorio según la fecha de la sesión, y otro según la hora. De esta manera, cada vez que se ejecute el programa, los ficheros de datos quedarán localizados en su respectiva fecha y hora de comienzo.

A continuación se describen los directorios en los que se distribuyen los ficheros del programa.

- **Src**: en este directorio se almacenan todos los ficheros correspondientes al código del programa. También se encuentra aquí el fichero *reconfig.txt* con los parámetros de configuración de la captura.

- **Bin**: en este directorio se encuentran los archivos binarios del programa y una copia del fichero de configuración *reconfig.txt*.

- **Data**: en esta carpeta se almacenan todos los archivos de los datos que se han capturado. Estos archivos están organizados en carpetas, por fecha y por hora, como se puede ver en la Figura F.9.

- **Readme**: este fichero contiene información relevante del programa principal.

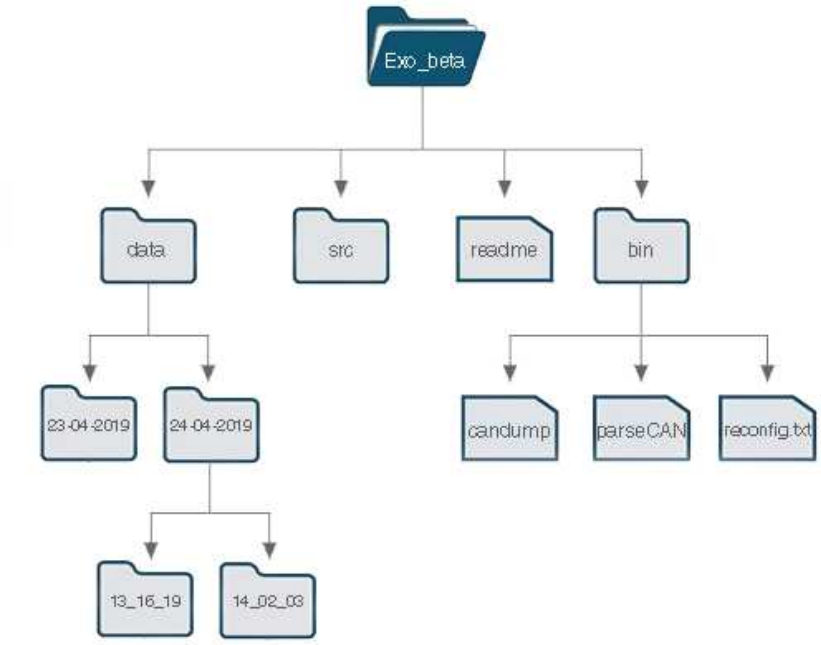


Figura F.9: Sistema de ficheros en la Raspberry Pi

F.5. Red

Como se menciona en la Sección F.2.1, el sistema cuenta con medios para la conexión en red cableada o inalámbrica.

F.5.1. Conexión cableada

Para conectar el sistema a una red mediante un cable Ethernet, basta con conectar este en la ranura Ethernet del sistema (ver Figura F.10). El sistema detectará esta conexión y se le asociará una dirección IP en la red.

Figura F.10: Conector Ethernet del sistema.

F.5.2. Conexión inalámbrica

El sistema también se puede conectar a una red de forma inalámbrica. Para ello será necesario haber realizado antes una conexión Ethernet (ver Sección F.5.1) o acceder al entorno gráfico del sistema mediante HDMI. Si se conecta el sistema a un monitor mediante HDMI, podremos acceder a la interfaz gráfica. En la parte superior derecha está el icono de WiFi. Simplemente tenemos que pulsarlo y seleccionar «Encender Wifi». Ahora, en la lista de redes disponibles seleccionamos la red WiFI a la que se tiene que conectar el sistema, y se pulsa sobre ella para conectar.

También se puede configurar la conexión mediante la consola de comandos. En primer lugar, se listarán las redes disponibles al alcance del sistema mediante el siguiente comando:

```
sudo iwlist wlan0 scan
```

Una vez se sepa la red y contraseña a la que se pretende conectar el sistema, se debe editar el fichero /etc/wpa_supplicant/wpa_supplicant.conf con el comando:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

En este fichero, se añadirá lo siguiente al final del archivo, cambiando los datos por los de la red WiFi correspondiente.

```
network={
ssid="nombre-de-tu-wifi"
psk="password-de-tu-wifi"
key_mgmt=WPA-PSK
}
```

Quedando finalmente el fichero de la siguiente forma:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
```

```
update_config=1
country=ES

network={
ssid="nombre-de-tu-wifi"
psk="password-de-tu-wifi"
key_mgmt=WPA-PSK
}
```

Por último, se reinicia la Raspberry mediante el comando:

```
sudo reboot
```

F.5.3. Acceso SSH

Para acceder al sistema vía SSH será necesaria la introducción de las credenciales del sistema.

```
Usuario: pi
Clave: 1234
```

Este acceso se puede realizar desde otro sistema Linux ejecutando en la consola de comandos:

```
ssh pi@IP_DEL_SISTEMA_A_CONECTAR
```

También se puede realizar este acceso a través de herramientas de conexión vía SSH como SmarTTY o PuTTY. Se configurarán los datos del sistema en estas herramientas y se conectará a la Raspberry Pi.

F.6. Servidor web

El sistema cuenta con un servidor web que solo responde a peticiones GET y que carga una vista HTML. Este servidor está ejecutado mediante Node.js y, por lo tanto, el servidor está escrito en JavaScript.

La vista HTML depende de las variables del estado del programa. Esto se puede hacer mediante una vista en formato EJS.

Desde la vista HTML se envían comandos al servidor en los parámetros de la ruta, como por ejemplo:

```
192.168.0.168/?cmd=xxx
```

Los comandos a los que atiende son:

- **Start**: si se recibe este comando y el sistema no está capturando, este comenzará la captura con los parámetros de configuración que le envía en el resto de parametros de la ruta.
- **Stop**: si el sistema está capturando y se recibe este comando, el sistema parará la captura. Si no se está capturando, no hace nada.

- **Date**: este comando configura la fecha del sistema con la del dispositivo desde el que se accede.
- **Shutdown**: el dispositivo se apaga cuando llega este comando.
- **Reboot**: si llega este comando, el sistema se reinicia.

El acceso a esta página web es mediante la introducción de la dirección IP del sistema en el navegador del dispositivo desde el que se pretende acceder, siempre y cuando esté en la misma red.