

UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE RECONOCIMIENTO GESTUAL

GRADO EN INGENIERÍA DE
SISTEMAS ELECTRÓNICOS

JOSÉ MIGUEL RÍOS RUBIO
MÁLAGA, 2015

Título del Trabajo Fin de Grado

Autor: José Miguel Ríos Rubio

Tutor: Francisco Javier Vizcaíno Martín

Departamento: Departamento de Tecnología Electrónica

Titulación: Grado en Ingeniería de Sistemas Electrónicos

Palabras clave: Diseño hardware, Desarrollo software, Diseño CAD 3D, Sistema basado en microcontrolador, Sistema de interacción natural, Bluetooth Low Energy, Unidad de medida inercial (IMU)

Resumen

Dentro del mundo de la electrónica existe una gran diversidad de campos, de entre todos ellos, en este trabajo fin de grado, nos centraremos en el ámbito de la interacción humano-máquina.

Se procederá pues, a realizar el prototipo de un *dispositivo de interacción natural simple* capaz de comunicarse de forma inalámbrica con un sistema externo, en este caso un PC. Para ello, se recurrirá al uso de un *sistema basado en microcontrolador* que procesará la lectura obtenida de un *acelerómetro* y determinará el gesto realizado para, posteriormente, transmitirlo a través de un enlace *bluetooth* al sistema receptor.

De este modo, comenzaremos por realizar el *diseño hardware* del sistema, diseñando el esquema eléctrico y el diseño CAD de la placa de circuito impreso (mediante el uso de la herramienta *Eagle*), para finalizar con el desarrollo de la placa y el proceso de soldadura de componentes. Una vez finalizado el hardware del sistema nos ocuparemos del *diseño software*, tanto de la parte correspondiente al prototipo desarrollado como de la parte localizada en el PC, la cual nos permitirá verificar el correcto funcionamiento del sistema.

Complementariamente al desarrollo hardware y software, crearemos el modelo 3D (gracias al software *Freecad*) de la carcasa que albergará al prototipo.

English version of the title

Author: José Miguel Ríos Rubio

Supervisor: Francisco Javier Vizcaíno Martín

Department: Departamento de Tecnología Electrónica

Degree: Grado en Ingeniería de Sistemas Electrónicos

Keywords: Hardware design, Software development, 3D CAD design, Microcontroller based system, Natural user interface system, Bluetooth Low Energy, Inertial measurement unit (IMU)

Abstract

Inside the world of electronics there is a large diversity of fields, from among all, in this Final Degree Project, we will focus in the ambit of the human-machine interaction.

We will proceed then, to perform the prototype of a *simple natural user interface* that is able to make a wireless communication with an extern system, in our case a PC. To do that, we will use a *microcontroller based system* that will process the acquired reads from an *accelerometer* and determines the gesture accomplished to transmit it later through a *Bluetooth* link to the reception system.

In that way, we will start by implementing the *hardware design* of the system, designing the electrical schematic and the CAD design of the printed circuit board (by using the Eagle software), manufacturing the board and doing the components solder process. Once completed the hardware of the system we will continue with the *software design*, at both side of the system: the prototype manufactured and the extern system (PC), that allows verify the correct functionality of the system.

Complementary to the hardware and software development, we will work out the 3D model of the prototype enclosure (by using the Freecad software).

Agradecimientos

Haber llegado al punto de mi vida en cual me encuentro no ha sido un camino fácil, no obstante, gracias a todas aquellas personas que han estado presente a lo largo de mis estudios, ese camino se vio suavizado y se hizo más ameno.

Familia, amigos, compañeros, profesores... por vuestro tiempo, ayuda, consejos y apoyo, gracias.

Contenido

Capítulo 1. Introducción	1
1.1. Contexto tecnológico	2
1.2. Objetivos del Trabajo Fin de Grado	5
1.3. Estructura de la Memoria	5
Capítulo 2. Especificaciones del sistema	7
2.1. Requisitos	12
Capítulo 3. Desarrollo del sistema	15
3.1. Selección de componentes	15
3.2. Desarrollo hardware	18
3.3. Desarrollo software	23
3.3.1. Configurando el entorno de programación	23
3.3.2. Software	26
3.4. Desarrollo del modelo 3D de la carcasa	34
Capítulo 4. Verificación y pruebas.....	39
4.1. Sistema de pruebas	39
4.2. Pruebas realizadas	45
4.2.1. Pruebas de funcionalidad	45
4.2.2. Pruebas de prestaciones	49
4.2.3. Tabla resumen de pruebas	50
Capítulo 5. Manual de instalación y uso	51
5.1. Manual de instalación	51
5.2. Manual de uso	52
5.2.1. Configuración inicial	52
5.2.2. Conexión con el sistema a controlar.....	52

5.2.3. Cambio de modo	52
5.2.4. Modo continuo.....	52
5.2.5. Modo gestos.....	52
5.2.6. Consumo	53
5.2.7. Movimientos y posiciones posibles	53
5.2.8. Gestos predeterminados	54
Capítulo 6. Conclusiones y trabajo futuro.....	57
Apéndice A. Presupuesto de elaboración	59
Referencias	61

Lista de Acrónimos

ADC	Analog-Digital Converter
BLE	Bluetooth Low Energy
CAD	Computer Aided Design
CNC	Control Numérico Computarizado
DOF	Degrees Of Freedom
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GPS	Global Positioning System
I2C	Inter-Integrated Circuit
ICSP	In Circuit Serial Programming
IDE	Integrated development environment
IMS	Inertial Measurement System
IMU	Inertial Measurement Unit
LED	Light Emitting Diode
MEMs	Microelectromechanical systems
PC	Personal Computer
PCB	Printed Circuit Board

PCI	Placa de Circuito Impreso
POO	Programación Orientada a Objetos
RF	Radio Frecuencia
RTI	Rutina de Tratamiento de Interrupción
SMD	Surface Mounted Device
SPI	Serial Peripheral Interface
SWNUI	Simple Wireless Natural User Interface
TFG	Trabajo Fin de Grado
TPV	Terminal Punto de Venta
UART	Universal Asynchronous Receiver-Transmitter
UMA	Universidad de Málaga
USB	Universal Serial Bus

Capítulo 1. Introducción

Desde el surgimiento de la especie humana y, mucho antes de la evolución del lenguaje, la comunicación no verbal nos ha acompañado hasta hoy día. No obstante, el estudio de ésta no alcanza una importancia digna de mención hasta la década de 1950, en la que en el año 1952, el antropólogo *Ray Birwhistell* escribió "*Introduction to kinesics*" [1], el cual es considerado como uno de los iniciales estudios modernos sobre comunicación no verbal. *Ray Birwhistell* acuñó el término "*kinesics*" ("*kinésica*" / "*cinésica*" en castellano), proveniente del griego *kinesis*, que significa movimiento, para referirse al estudio del movimiento humano. De este modo, la kinésica, junto con la "*proxémica*" (estudio de la forma de uso del entorno por parte de la persona a la hora de comunicarse) y la "*paralingüística*" (estudio de las cualidades no verbales de la voz) componen la base principal del lenguaje no verbal. Con lo cual, los gestos pueden clasificarse en resumidas cuentas dentro de la "*kinésica*" del lenguaje no verbal.

Se entiende como reconocimiento gestual la adquisición e interpretación de movimientos de carácter expresivo del cuerpo o de partes del mismo. De esta forma, se debe diferenciar claramente entre un movimiento, que puede o no tener significado, y un gesto, cuyo carácter significativo se encuentra intrínsecamente ligado a su acción.

Normalmente, los gestos presentan la funcionalidad de permitir la comunicación no verbal entre dos seres que comparten la misma interpretación para el conjunto de señales que conforman el lenguaje gestual, y decimos "normalmente" pues suele haber casos en los que el gesto suele ir acompañado de algún sonido característico ("paralingüismo"). Una clara distinción entre comunicación verbal y no verbal radica en el hecho de que el primero es utilizado para comunicar información mientras que el

segundo es usado para comunicar estados de ánimo y/o actitudes respecto a un acontecimiento.

1.1. Contexto tecnológico

Determinar la posición y el movimiento en el espacio de una entidad es de gran importancia para una gran variedad de aplicaciones. Si observamos hoy día el mercado actual nos encontramos en medio de una batalla comercial en la que se intenta hallar qué sistema de captura y reconocimiento de movimiento es el más eficientemente aceptado por la sociedad y, consecuentemente, el más exitoso.

Las empresas relacionadas con los videojuegos fueron las que, en primera instancia, se abrieron paso en este mundo en busca de la mejora en la experiencia de juego de los usuarios, intentando encontrar una interfaz con la que el jugador pudiera interaccionar de forma innovadora y más cercana a la realidad. Entre los dispositivos más reconocidos y que mayor impacto han tenido se encuentran, entre otros (figura 1), el *Sensor Kinect* [2] de *Microsoft*, lanzado el 4 de noviembre de 2010 [3], que entró en el “Guinness World Records” (Libro Guinness de los Records) por ser el periférico de juego vendido más rápidamente tras vender una media de 133.333 unidades por día en sus primeros 60 días [4]. Básicamente, este dispositivo es un periférico de control para la videoconsola *Xbox 360*, aunque posteriormente se adaptase para su uso en PC con sistemas operativos *Windows 7* y *Windows 8*. El *Sensor Kinect* se encuentra en competencia directa con el *Wii Remote* [5] de *Nintendo* y el *PlayStation Move* [6] de *PlayStation* de los que hablamos a continuación.



Figura 1. Dispositivos de reconocimiento gestual (Kinect, Wiimote y PlayStation Move).

El *Wii Remote* fue desarrollado como periférico de control de la videoconsola *Nintendo Wii* y, apareció en el mercado junto con esta, el 19 de noviembre de 2006 [7].

Por otro lado, el *PlayStation Move* es un sistema conformado por un dispositivo de control, *Motion Controller*, y un dispositivo basado en una cámara, *PlayStation Eye*, que se encarga de obtener la posición del dispositivo de control. El sistema salió al mercado el 17 de septiembre de 2010 [6].

Sin embargo, no solo las empresas dedicadas al entretenimiento están interesadas en el uso de este tipo de sistemas de reconocimiento de movimiento. Determinar la posición y el movimiento en el espacio de una entidad es de gran importancia para una gran variedad de aplicaciones en distintos campos. Parte de la característica más destacable de los teléfonos inteligentes (*Smartphones*) de hoy en día consiste en la detección de la orientación, inclinación y aceleración que sufren, todo esto es gracias a la incorporación de dichos sistemas de reconocimiento en estos dispositivos. Así mismo, un gran número de aplicaciones robóticas se basan en el uso de estas tecnologías sensoriales para realizar el control del movimiento de las distintas partes del robot o mecanismo asociado. Esto se extiende desde el manejo de *Drones* (vehículos aéreos no tripulados controlados remotamente) en aplicaciones militares, control de un robot de alta precisión para realizar operaciones quirúrgicas (campo de medicina y salud) como el *Sistema Quirúrgico Da Vinci* [8], dispositivos de ayuda a la navegación de vehículos (sobre todo aéreos), etc.

En la actualidad, el reconocimiento gestual se encuentra extendido entre dos tipos de tecnologías de adquisición e interpretación de gestos: las basadas en “cámara”, las cuales pueden considerarse ópticas, que requieren de un procesamiento gráfico y, por consiguiente, de visión directa del gesto (como el *Sensor Kinect* y el *PlayStation Move*), y los basados en sistemas de “unidad de medida inercial” (IMU, de sus siglas en inglés “*Inertial Measurement Unit*”), no ópticas, como el *Wii Remote*.

Los sistemas “ópticos” de reconocimiento de movimiento pueden o no servirse de algún dispositivo auxiliar al que realizar el seguimiento y determinar el movimiento realizado (como en el caso del *Motion Controller* de PlayStation) o no (*Kinect* de Microsoft), dado que estos sistemas están ampliamente ligados más al algoritmo software de reconocimiento y procesamiento de imagen. Cuanto más complejo sea dicho procesamiento mayor potencia tendrá el sistema. De esta forma, el *Sensor Kinect* es un dispositivo cuyo procesamiento de reconocimiento es tan potente que permite reconocer no solo al usuario que está interactuando, sino obtener cada uno de los movimientos realizados por este. No obstante, y como es de esperarse debido a que el sistema requiere de visión directa, si el usuario oculta sus manos tras la espalda y realiza algún movimiento con ellas esto no será detectado por el sistema.

En contraposición, las unidades de medida inercial o IMUs son dispositivos electrónicos que albergan un conjunto de sensores que permiten identificar al dispositivo en el

espacio mediante la adquisición de ciertos parámetros de aceleración, inclinación, orientación y posición. Para ello, estos sistemas pueden presentar diferentes tipos de sensores: acelerómetros, giroscopios, magnetómetros (basados en tecnologías de sistemas microelectromecánicos, MEMs), barómetros e incluso GPS (Sistema de posicionamiento global).

Los acelerómetros son sistemas capaces de medir la aceleración lineal, los giroscopios permiten determinar la velocidad angular, los magnetómetros permiten detectar campos magnéticos (como el de la Tierra), los barómetros indican la presión barométrica (en relación a la altitud) y el GPS determina la posición del sistema según las coordenadas terrestres de longitud, latitud y altitud, así como de proveer de parámetro temporal al sistema (tiempo).

Según la presencia de dichos sensores en la IMU y el número de ejes de medida de tales elementos, un sensor inercial puede ser clasificado, según los grados de libertad, por las siglas DOF (*"Degrees of freedom"*) del mismo. Así, si una IMU tiene 6 DOF, ésta suele estar compuesta por un acelerómetro y un giroscopio de 3 ejes cada uno ('X', 'Y' y 'Z'). En caso de que la IMU sea de 9 DOF, el sistema contendrá a su vez un magnetómetro.

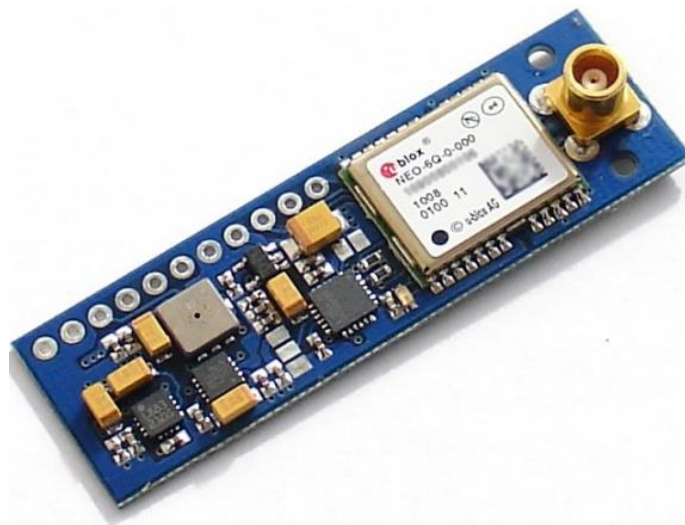


Figura 2. Ejemplo de IMU de 11 DOF (Con GPS y Barómetro incluidos).

Puede considerarse que cualquier extremidad del cuerpo humano tiene como máximo 6 grados de libertad de movimiento, que permiten determinar la posición y orientación de esa parte en el espacio.

Todo avión que sobrevuela en la actualidad nuestros cielos presenta un sistema de medición inercial (IMS), el cual cuenta con una unidad de medida inercial (IMU) para

corregir la inclinación (de forma automática) y, por tanto detectar cualquier malfuncionamiento, como avisar al piloto de cualquier peligro existente.

Conociendo estos dos tipos de tecnologías de reconocimiento de movimiento, podemos volver a analizar los productos de entretenimiento antes expuestos (*Kinect*, *Wii Remote* y *PlayStation Move*) para determinar cuál tecnología es aplicada a cada uno de ellos. De esta forma, el *Sensor Kinect* está basado exclusivamente, como ya hemos dicho, en el reconocimiento óptico, el *Wii Remote* por el contrario, es un sistema puramente basado en IMU, mientras que el *PlayStation Move* representa la fuerza de la unión de ambas tecnologías en el mismo sistema (sistema híbrido) ya que su *Motion Controller* presenta una IMU que determina aspectos de movilidad semejantes a los del *Wii Remote* mientras que, el *PlayStation Eye*, añade un reconocimiento visual de nuestros movimientos.

1.2. Objetivos del Trabajo Fin de Grado

Visto todo lo anterior, el objetivo principal de este trabajo fin de grado consiste en realizar un *sistema de interacción natural de usuario* (NUI, del inglés *Natural User Interface*) simple y de bajo coste. Para ello se procederá al desarrollo del prototipo de un elemento capaz de registrar e interpretar un conjunto de simples gestos (movimientos e inclinaciones), realizados por la mano del usuario, con el fin de controlar de forma inalámbrica diversos sistemas externos sin que exista una visión directa entre la mano y el sistema.

1.3. Estructura de la Memoria

El documento presente se encuentra dividido en varios capítulos que muestran el conjunto de contenidos desarrollados para realizar la implementación del sistema. A continuación, se procede a describir el contenido de cada uno de esos capítulos, de esta forma:

- En el primer capítulo ("*Introducción*"), el presente, se ha hablado del ámbito de este Trabajo Fin de Grado (TFG). Se ha intentado dar una pincelada muy general del TFG, comenzando por el desarrollo de una breve introducción relativa a la temática del proyecto, el contexto tecnológico actual de diversas tecnologías y productos relacionados con el sistema a desarrollar y acabando con los objetivos principales de dicho sistema y la estructura del documento.

- El capítulo segundo (*“Especificaciones del sistema”*), concierne a las especificaciones del sistema donde, analizando el objetivo que debe cumplir el mismo, se fijan las características que presentará el sistema y se establecen los requisitos que se pretenden cumplir.
- En el tercer capítulo (*“Desarrollo del sistema”*), se determina los componentes a utilizar para el diseño del sistema y se muestra todo el proceso de diseño y desarrollo y fabricación del prototipo del sistema realizado, pasando por el hardware, el software y el diseño CAD de la carcasa.
- En el capítulo cuarto (*“Verificación y pruebas”*) se define el conjunto de pruebas a realizar sobre el sistema desarrollado, el proceso de realización de tales pruebas y el resultado de las mismas, verificándose así la correcta funcionabilidad del sistema y el cumplimiento de los objetivos y especificaciones del proyecto.
- El quinto capítulo (*“Manual de instalación y uso”*), alberga el contenido relacionado con la instalación, puesta en marcha y utilización del sistema mediante la redacción de ciertos manuales de usuario.
- El capítulo sexto (*“Conclusiones y trabajo futuro”*) expone el conjunto de experiencias adquiridas en la realización del proyecto, así como un análisis de prestaciones del sistema, indicando las posibles mejoras que se podrían incorporar al prototipo realizado.

Capítulo 2. Especificaciones del sistema

Dado que nuestro sistema es un dispositivo de interacción natural (NUI) que nos va a permitir determinar gestos simples y comunicarlos de forma inalámbrica, se le dará al sistema el nombre *SWNUI* (Simple Wireless Natural User Interface) de aquí en adelante.

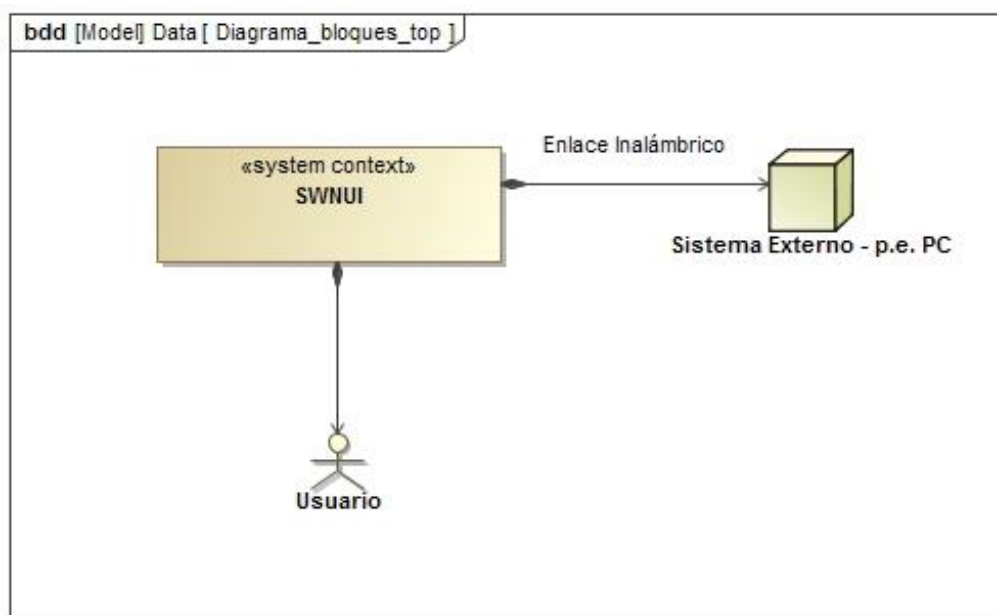


Figura 3. Contexto del sistema.

Según el objetivo de nuestro proyecto, el sistema permitirá el reconocimiento de los gestos realizados con la mano sin la necesidad de que exista visión directa entre el elemento de control y el sistema a controlar. Además, el sistema deberá ser capaz de comunicarse de forma inalámbrica con ese

elemento externo (figura 3). Lo que nos lleva a acotar ciertas características que poseerá nuestro sistema:

- Podemos determinar que, dado que no se precisa visión directa, el tipo de tecnología a utilizar será el basado en IMUs (tecnología “no óptica”).
- Ya que los gestos a registrar se realizan con la mano, sabemos que el IMU se debe situar en algún punto de la misma.
- La comunicación deberá ser inalámbrica y de un rango de alcance medio/alto. Entre ellos podría usarse RF, bluetooth, Zigbee, wifi, etc.

Por otro lado, el sistema buscará la armonía entre bajo consumo y tamaño reducido para satisfacer la tendencia actual hacia la creación de dispositivos *wearables* (“vestibles”):

- Esto implica que el sistema deberá encontrarse en un estado de bajo consumo el máximo tiempo posible y se activará para realizar el proceso de registro, procesado y envío del gesto realizado.
- La activación del sistema, por tanto, no dependerá del movimiento realizado sino que se llevará a cabo a partir de un pulsador.
- Dado que el sistema será de pequeño tamaño y *wearable*, el pulsador de activación se debe localizar en una posición confortable para la pulsación por el usuario (que se evite el tener que recurrir a la otra mano para su uso), determinamos que la posición óptima del sistema se encontrará en el dedo índice de la mano, con pulsación mediante el dedo pulgar (figura 4).
- El bajo consumo junto al bajo coste limitaría el tipo de tecnología de comunicación inalámbrica a utilizar: RF, BLE...

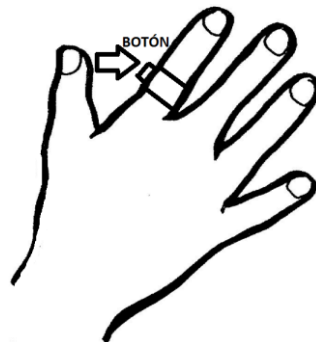


Figura 4. Pulsador en posición confortable.

Analizando el objetivo del sistema en sí, podemos apreciar que, registrar los datos de movimientos e inclinaciones realizados continuamente es un paso previo a la obtención del gesto realizado, por lo que si el sistema que queremos controlar es un sistema dinámico (requiere un flujo continuo de datos para su control, por ejemplo controlar el movimiento de un robot o un dron, o manejar el cursor de un ordenador) incluir un modo de funcionamiento continuo, conmutable con el modo de envío de gestos, no es mala idea:

- Por tanto, se implementarán dos modos de funcionamiento del sistema, el modo de envío continuo de datos y el modo de envío de gestos.
- El modo de funcionamiento debe ser señalizado para el usuario, por ejemplo mediante un LED.
- Dado que el sistema incluye un pulsador para salir del bajo de consumo, con el fin de no incrementar el coste, éste también se utilizará para realizar el cambio entre modos.

Analizando más detenidamente la tecnología necesaria tanto para el IMU, como para el enlace inalámbrico, conseguimos determinar que:

- Como los gestos a registrar son sencillos (inclinación y movimiento), el IMU se reduce a un simple acelerómetro de tres ejes.
- Ya que el sistema debe de ser de bajo consumo y la velocidad de transmisión de datos no necesita ser muy alta, la comunicación inalámbrica se puede satisfacer mediante *Bluetooth Low Energy* (BLE).

Los movimientos voluntarios de una mano suelen ser inferiores a los 7 Hz [9], por lo que la frecuencia de muestreo deberá ser superior a los 14 Hz (para cumplir con el teorema de Nyquist). De este modo, la frecuencia no será una limitación a la hora de seleccionar el elemento de procesamiento de nuestro sistema:

- Cualquier microcontrolador de gama baja/media (8 bits), poseerá la velocidad de procesamiento suficiente para cumplir con el muestreo de datos.
- Del mismo modo, el microcontrolador deberá tener los periféricos adecuados para poder comunicarse con el acelerómetro (GPIO, SPI, I2C...), así como con el módulo BLE (UART).

Partiendo del objetivo de nuestro sistema, hemos identificado las necesidades ligadas al mismo, lo que permite distinguir entre las distintas especificaciones que tendrá el sistema, delimitando las tecnologías apropiadas para su implementación. Así pues, resumiendo todo lo anterior, podemos enumerar los elementos y tecnologías que compondrán nuestro sistema:

- Núcleo de procesamiento: Microcontrolador.
- Unidad de medida inercial: Acelerómetro de tres ejes.
- Comunicación inalámbrica: Modulo *transceiver* BLE.
- Entrada de usuario: Pulsador.
- Indicación al usuario: LED.

Por tanto, el diagrama de bloques internos del sistema quedará como el representado en la figura 5.

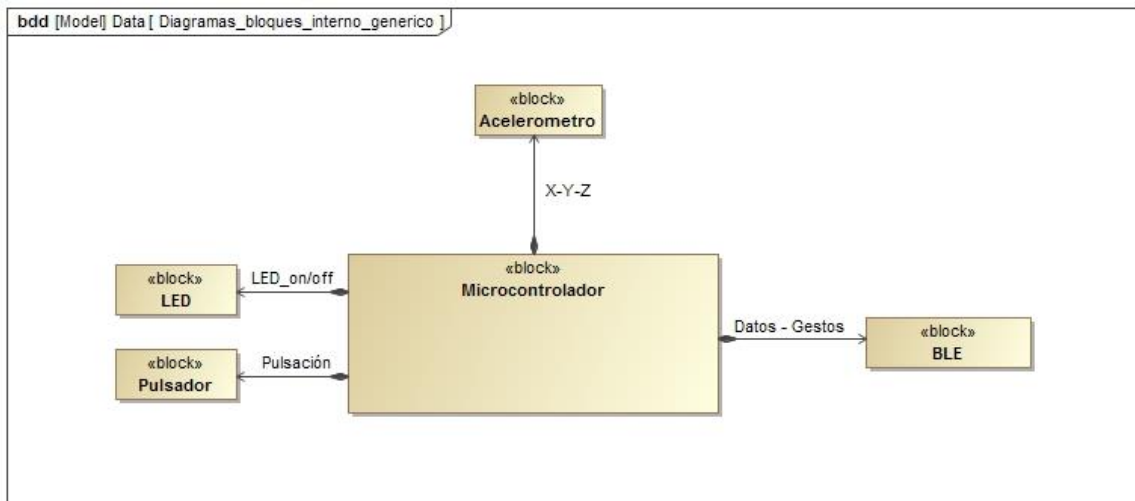


Figura 5. Pulsador en posición confortable.

Llegados a este punto, podemos determinar cuál será la funcionalidad completa de nuestro sistema. El sistema presentará dos modos de funcionamiento, llamémoslos “modo continuo” y “modo gestos”, conmutables entre sí a partir del pulsador del sistema. Por otro lado, el LED establecerá una indicación visual para el usuario dando a entender en qué modo se encuentra el sistema. El “modo continuo” del sistema enviará a través del módulo BLE, de forma continua, los datos correspondientes a la posición y movimiento de la

mano, mientras que el “modo gestos” introducirá al sistema en un estado de bajo consumo del que, al presionar y mantener pulsado el pulsador, se saldrá para muestrear y procesar los gestos realizados por la mano y, al soltar el pulsador, estos se enviarán por el módulo BLE.

Toda la funcionalidad genérica de nuestro sistema, expuesta en el párrafo anterior, se muestra en la siguiente imagen (figura 6) mediante su diagrama de actividad (diagrama de flujo).

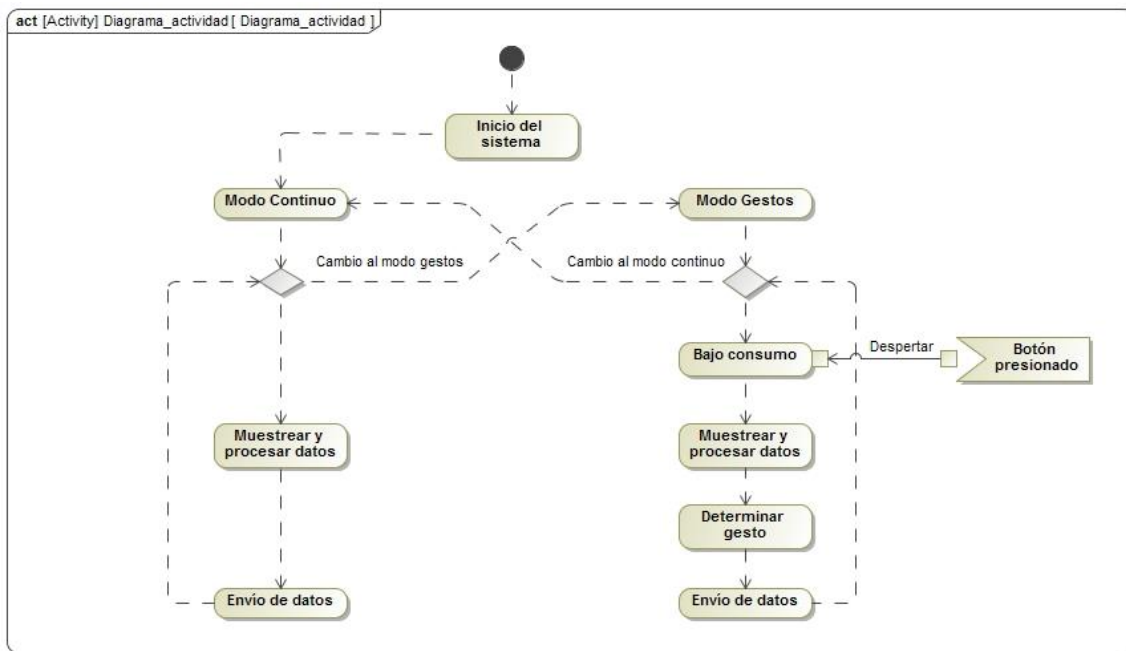


Figura 6. Pulsador en posición confortable.

La funcionalidad del sistema, así como el uso y manejo del mismo, se recogerán de forma exacta en el apartado de manuales de instalación y uso del presente documento (capítulo cinco). Además, una vez desarrollado el prototipo, los dos modos de funcionamiento del sistema se verificarán de forma apropiada en el apartado de pruebas y verificación (capítulo 4).

2.1. Requisitos

Habiendo determinado las especificaciones que deberá tener el sistema, procederemos a establecer los requisitos que permitirán el cumplimiento de esas especificaciones.

Id.	Nombre	Descripción	Tipo	Prioridad
1	Función del sistema	Procesar y determinar gestos realizados a partir del acelerómetro	Funcional	10
2	Comunicación inalámbrica	Transmitir gestos de forma inalámbrica a través de Bluetooth Low Energy	Funcional	10
3	Conexión automática	El enlace Bluetooth se establece automáticamente (en “segundo plano”)	Funcional	7
4	Modo continuo	Modo de funcionamiento continuo (Streaming de datos)	Funcional	8
5	Modo gestos	Modo de funcionamiento discreto (Envío de gestos aislados)	Funcional	9
6	Cambio de modo	Cambiar modos de forma directa por el usuario mediante el pulsador	Funcional	7
7	Bajo consumo	Porcentaje de tiempo en bajo consumo en el modo gestos superior al 85%	Funcional/Prestación	8
8	Autonomía	Tiempo de batería en modo continuo superior a 10h	Prestación	7
9	Diseño <i>Wearable</i>	Tamaño no superior a 5 cm ³ y portable en el dedo índice	Prestación	6
10	Indicación de modo	Incorporar un LED señalizador del modo actual del sistema	Funcional/Prestación	8
11	Batería ligera	Alimentación del sistema mediante baterías no superiores en tamaño a dos AAA	Prestación	7

Tabla 2.1. Tabla de descripción de requisitos

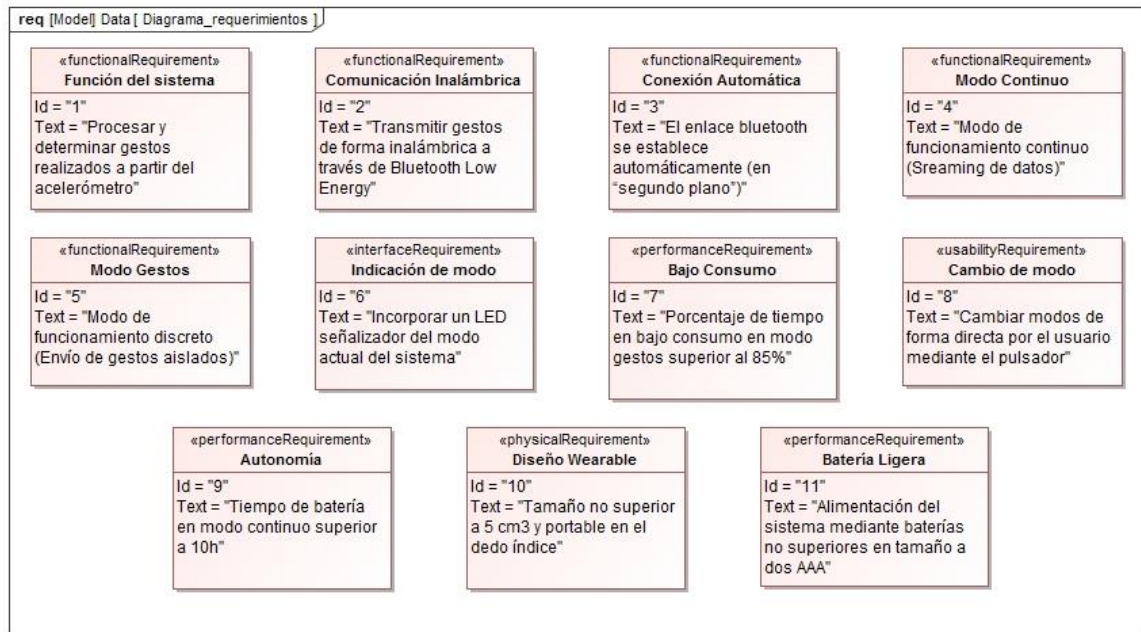


Figura 7. Diagrama SYSML de requisitos del sistema.

Capítulo 3. Desarrollo del sistema

Habiendo definido las especificaciones del sistema y los requisitos que este presentará, así como las tecnologías a utilizar, salimos del nivel del *ingeniero de sistemas* y entramos de lleno en el proceso de desarrollo del sistema. De esta forma, comenzaremos por determinar los elementos exactos, que se adecuan a las tecnologías establecidas anteriormente, a utilizar. Continuaremos por el diseño y desarrollo del hardware del sistema (esquema eléctrico y fabricación de la placa de circuito impreso) y el desarrollo software de la funcionalidad del sistema. Y, acabaremos por el proceso de creación del modelo 3D de la carcasa que almacenará al sistema.

3.1. Selección de componentes

Para satisfacer de cierta manera el bajo precio del sistema al mismo tiempo que se mantiene un tamaño reducido, buscamos ajustar los elementos del sistema para que sean directamente compatibles con la batería del mismo, sin necesidad de ningún subsistema de regulación de la tensión (evitaremos el uso de elevadores o reductores de tensión).



Figura 8. Baterías *Energizer* AAA.

Ya que fijamos como requisito que la batería del sistema tuviera un tamaño máximo de *dos pilas AAA* (figura 8), se ha decidido colocar dos pilas de este tipo en serie como alimentación del sistema. Esto podría suministrar una tensión de alimentación de 3V, una capacidad superior a 1 mAh y una corriente máxima de entrega superior a los 500 mA (capacidad y pico de corriente demandable dependientes del tipo de batería) [10].

Por tanto, teniendo en cuenta estas baterías como sistemas de alimentación, debemos buscar elementos cuyos límites de operación vayan desde un valor máximo de 3V a un valor mínimo de unos 2.5V (tensión a partir de la cual la curva de caída de tensión de las baterías se acentúa de forma considerada).



Figura 9. Módulo ADXL335. Acelerómetro de 3 ejes.

En el apartado anterior, determinamos que el papel del IMU lo tomará un único acelerómetro y, dado que buscamos un bajo coste y que la gran exactitud del sistema no es completamente necesaria, podemos recurrir a un acelerómetro analógico (el cual no presenta ningún procesamiento de las muestras que mide, cosa que aumenta el precio, como en el caso de los acelerómetros digitales). El acelerómetro seleccionado es el *ADXL335* (figura 9), definido según su datasheet [11] como un acelerómetro de tres ejes, $\pm 3g$, pequeño, de bajo consumo y salida condicionada de voltaje. Además, el voltaje de operación del mismo permite la alimentación directa de los 3V (voltaje de alimentación: 1.8 V – 3.6 V).

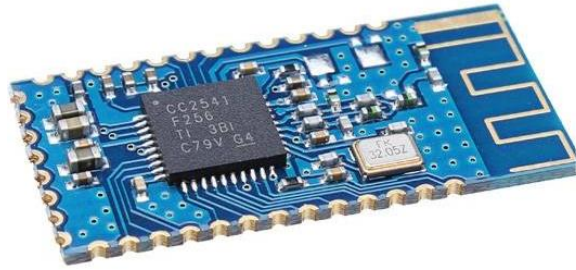


Figura 10. Módulo BLE-CC41 Bolutek. Acelerómetro de 3 ejes.

Cuando analizamos las especificaciones del sistema, concluimos que la comunicación inalámbrica sería satisfecha mediante la tecnología *Bluetooth Low Energy* (BLE). Dentro de esta tecnología, existen módulos “transceivers” conversores de comunicación serie a BLE. El módulo más barato que en la actualidad puede obtenerse es el módulo *BLE-CC41* de *Bolutek* (figura 10) [12]. Este módulo BLE, de clase II, se basa en el SOC (System On Chip) de Texas Instrument CC2541 [13], una solución de Bluetooth Low Energy y comunicaciones propietarias de 2.4 GHz. Por supuesto, el módulo tiene un rango de alimentación apto para nuestra batería (voltaje de alimentación: 2 V – 3.6 V).

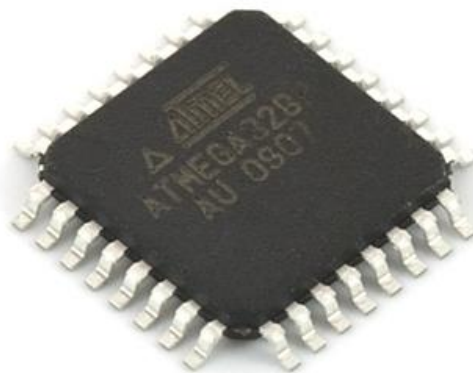


Figura 11. Microcontrolador ATMEGA328P-AU.

El microcontrolador a utilizar en nuestro sistema debe proporcionar los periféricos necesarios para poder utilizar tanto el sensor de aceleración (GPIOs) como el módulo de comunicación BLE (UART). Dado que cualquier

microcontrolador existente presenta puertos de entrada/salida, la elección del microcontrolador se reducirá a uno que presente algún puerto de comunicación serie y permita una tensión de alimentación compatible con el sistema de baterías. Dentro del abanico tan amplio de selección, optamos por el microcontrolador *ATMEGA328* de *Atmel* (figura 11) [14], el cual posee UART y una tensión de alimentación apta (tensión de alimentación: 1.8 V – 5.5 V). El *ATMEGA328* es el microcontrolador utilizado en la plataforma Arduino estándar (Arduino Uno) [15]. Este es el motivo de su elección, ya que se le puede cargar el bootloader de Arduino, que nos permitirá el uso de su capa de abstracción hardware a la hora de programar nuestro sistema.

3.2. Desarrollo hardware

Una vez determinados los componentes electrónicos a utilizar en nuestro proyecto, procedemos a la elaboración del diseño hardware del sistema (mediante el software de diseño CAD *Eagle* [16]), empezando por el esquema eléctrico.

El esquema eléctrico de nuestro sistema se puede observar en la siguiente imagen (figura 12).

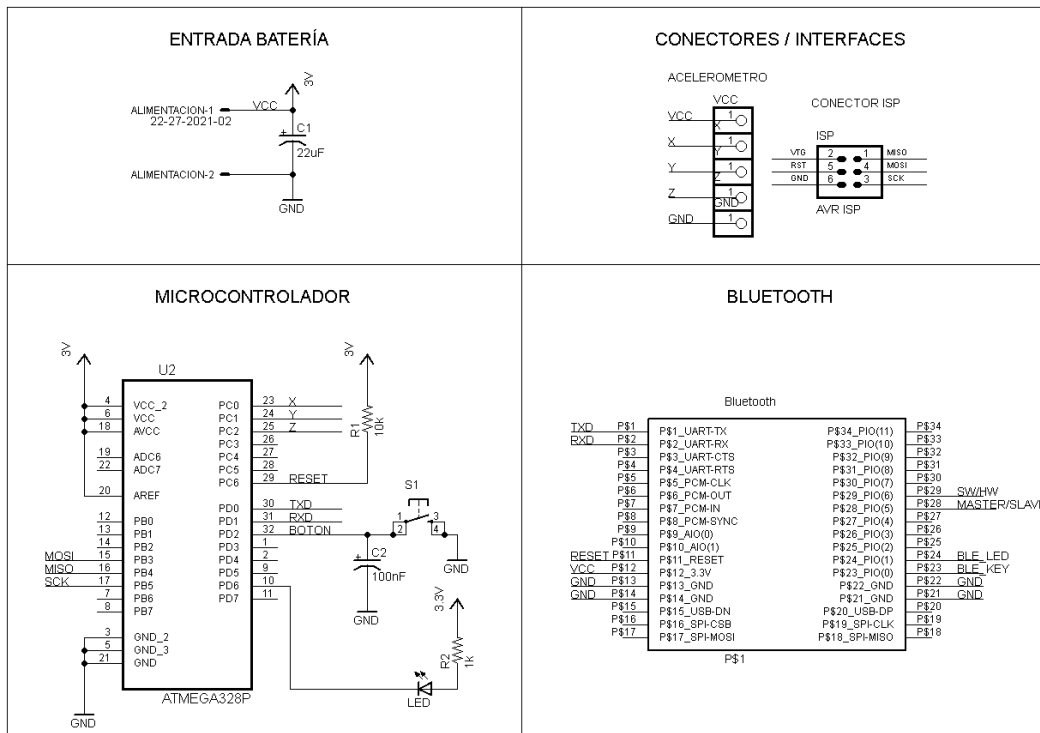


Figura 12. Esquema eléctrico del sistema.

El microcontrolador obtiene los datos del acelerómetro (pines 'X', 'Y' y 'Z' para cada eje) y los procesa (determinando el gesto realizado) para, posteriormente, enviar un comando correspondiente por el puerto serie (pines RXD y TXD) hacia el módulo BLE, el cual se encargará de transmitirlo.

El método seguido en el diseño eléctrico para permitir la programación del microcontrolador es el basado en ICSP (In Circuit Serial Programming). Este método de programación serie que utiliza el puerto SPI del microcontrolador, fue diseñado para permitir la programación de microcontroladores que se encuentran ya integrados en sistemas, sin necesidad de que estos deban ser extraídos del mismo. Por ello, en nuestro sistema eléctrico añadimos el llamado conector ICSP (figura 13) en su versión de seis pines (existe la versión de diez pines) para ahorrar tamaño.

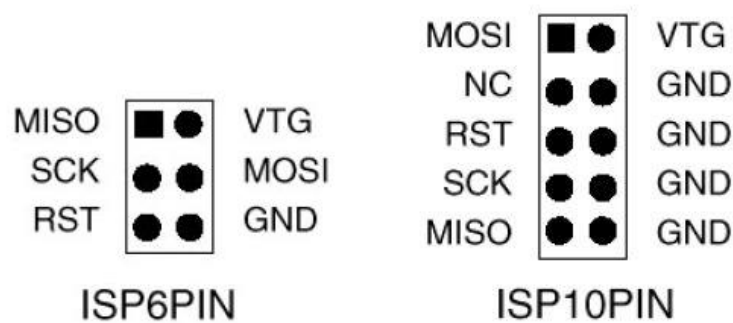


Figura 13. Conectores ICSP.

Con el fin de ahorrarnos componentes innecesarios y, ya que no se precisa de una gran velocidad de operación, el microcontrolador se programará con el Bootloader de Arduino a una frecuencia interna del sistema de 8 MHz. Esto nos evitará el uso de un resonador externo (cristal y condensadores) debido a que se configura para que el reloj del sistema se genere a partir del oscilador interno del mismo microcontrolador.

Otro aspecto a tener en cuenta en nuestro circuito, sería la conexión del pin AREF (tensión del conversor ADC) del microcontrolador a la tensión de alimentación, pues necesitaremos realizar la lectura de los valores analógicos del voltaje de cada eje del acelerómetro para determinar los datos de aceleración de forma correcta.

El pulsador del circuito es una parte fundamental del mismo ya que cumple un rol fundamental en la funcionalidad del sistema. Por este motivo, con el fin de evitarnos posibles problemas a la hora de gestionar de manera software los posibles rebotes que se producen, procedemos a incluir un condensador que filtre tales rebotes de manera hardware. Por otro lado, haciendo uso de la resistencia de “pull-up” interna del pin del microcontrolador, podremos ahorrarnos la resistencia externa que fije el nivel lógico “alto”.

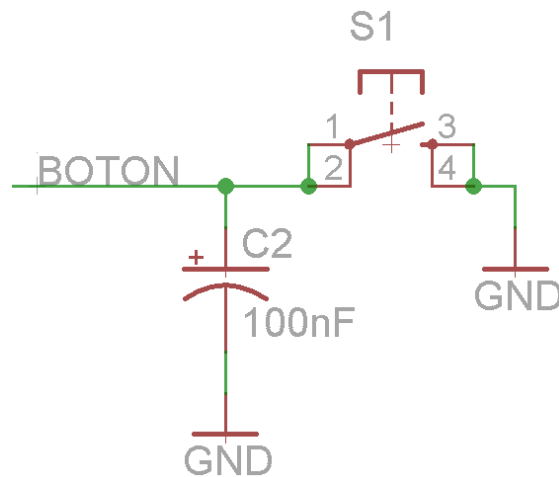


Figura 14. Pulsador con anti-rebote.

Como se puede observar, en el circuito eléctrico, se han añadido dos resistencias tanto para el pin de RESET del microcontrolador (resistencia para fijar un nivel lógico “alto”, de valor típico de 10k según el fabricante), como para el LED (resistencia que limitará el flujo de corriente). A continuación procedemos a realizar el cálculo del valor de resistencia óptimo para que el LED brille a una intensidad adecuada sin que sea un potencial elemento de consumo de corriente en nuestro sistema (2 mA):

$$\text{LED (I = 2mA)} \rightarrow R = \frac{V_{CC} - V_{LED}}{I} = \frac{3 - 1.3}{0.002} = 850 \, \Omega \approx 1 \, K\Omega$$

La tensión de 1.3 V de caída del LED se ha obtenido por medición del mismo (figura 15), ya que el LED a utilizar en el sistema debe poseer una tensión de caída en directa (V_{Forward}) muy baja, dada nuestra tensión de alimentación de 3V.



Figura 15. Medición de caída de tensión en el LED.

Una vez que tenemos diseñado el esquema eléctrico de nuestro circuito y habiendo repasado los diversos aspectos dignos de mención sobre el mismo, procedemos a diseñar la placa de circuito impreso.

Teniendo en mente el requisito del diseño *wearable*, de pequeño tamaño del sistema, determinamos que, a excepción del pulsador, el resto de componentes del circuito deberán ser de montaje superficial (SMD), además la PCI tomará el menor tamaño posible.

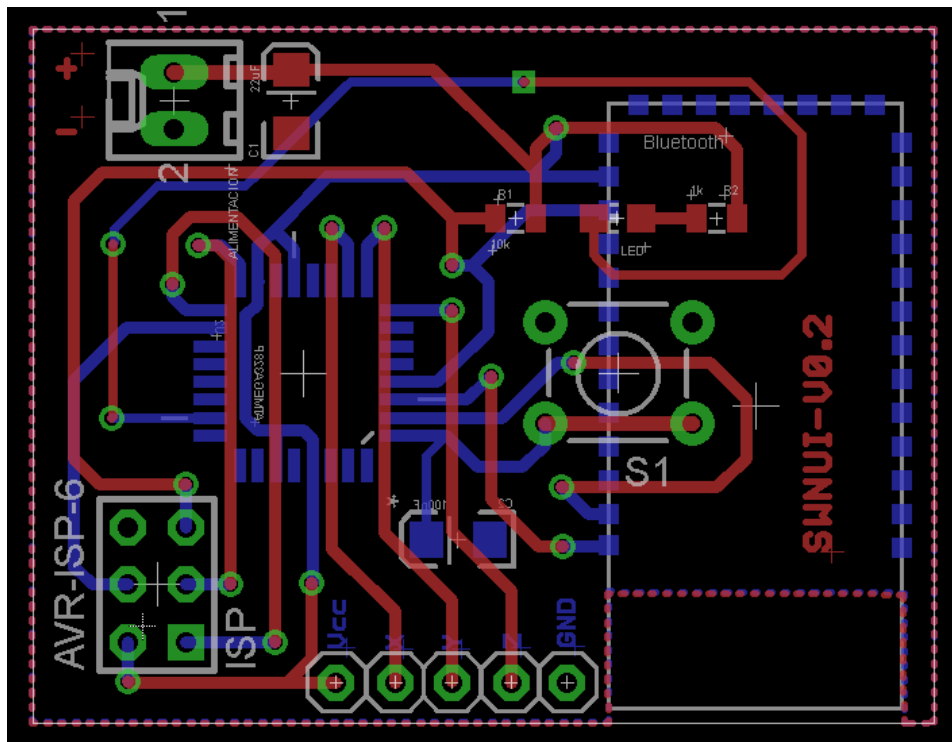


Figura 16. Diseño CAD de la placa de circuito impreso.

Conservando la idea del tamaño reducido de placa, conseguimos el resultado que se puede ver en la figura anterior (figura 16). La placa final es a doble cara, de un tamaño de 4x3 cm de lado y un ancho de pista de 0.5 mm.

Finalizado el diseño CAD de la placa de nuestro sistema, se exportaron los archivos gerbers y excellons adecuados y se enviaron para su fabricación (mediante la fresadora CNC de la UMA). El resultado del proceso de manufactura fue el siguiente (figura 17):

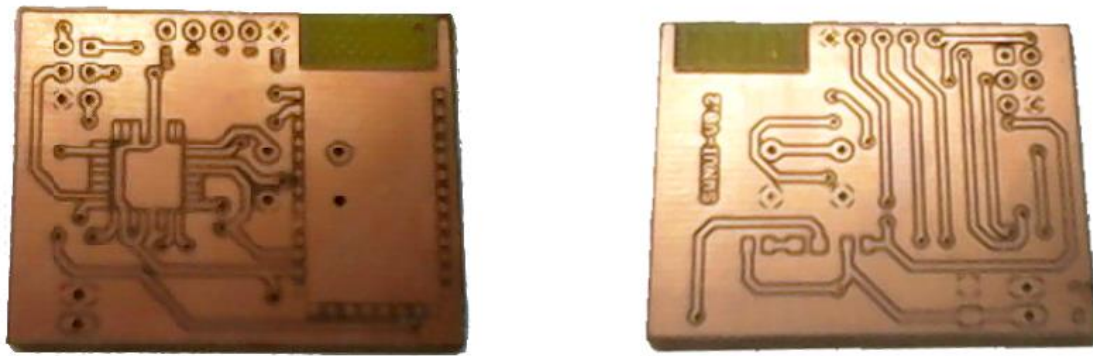


Figura 17. Placa de circuito impreso.

La fabricación de la placa mediante una fresadora, a pesar de su resultado óptimo, presenta un inconveniente, las vías no son metalizadas. Esto implica que a la hora de soldar se debe prestar mucha atención para interconectar a través de las vías las pistas correspondientes de ambas caras. Para ello, procedemos a introducir pequeños fragmentos correspondientes a terminales de componentes electrónicos discretos (por ejemplo sobrantes de resistencias) en cada vía y los soldamos. Una vez realizado esto en cada orificio pertinente, se lija las soldaduras obtenidas con el fin de eliminar el volumen vertical de las mismas, las cuales podrían entorpecer el proceso de soldadura de otros componentes (como en el caso del microcontrolador si hubiéramos diseñado vías en el interior de su emplazamiento). Estas soldaduras de unión en las vías son el primer paso en el proceso de soldadura del resto de componentes, así, lo siguiente sería las soldaduras de los componentes smd, empezando por el microcontrolador y el módulo BLE, seguido de las resistencias, el LED y los condensadores; y acabando con los componentes de mayor tamaño, es decir,

el pulsador y los pines que conforman los conectores de alimentación, programación y el acelerómetro.



Figura 18. Placa de circuito impreso tras el proceso de soldadura.

La imagen anterior (figura 18) muestra el resultado final de la placa de circuito impreso tras haber soldado todos los componentes del mismo. Esto concluye con el desarrollo hardware del sistema y nos abre la posibilidad de comenzar a programar nuestro sistema.

3.3. Desarrollo software

3.3.1. Configurando el entorno de programación

Como mencionamos anteriormente, el sistema se programará mediante el puerto ICSP a través del entorno de programación de Arduino. El dispositivo utilizado para la programación es el llamado *USBasp* [17], un programador ICSP de microcontroladores de *Atmel* de bajo precio (figura 19).

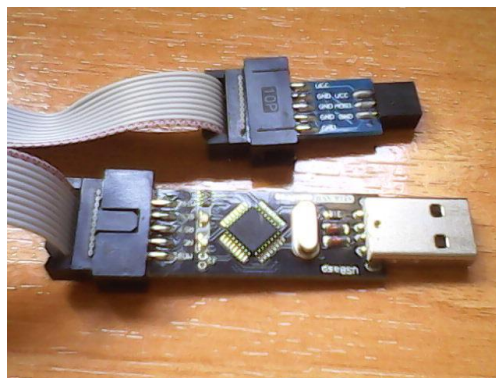


Figura 19. USBasp.

Para comenzar a programar al sistema se debe de cargar el Bootloader de Arduino. Para ello, se debe configurar el IDE Arduino con el dispositivo ATMEGA328P con reloj interno a 8 MHz (figura 20) y el USBasp como programador del sistema (figura 21).

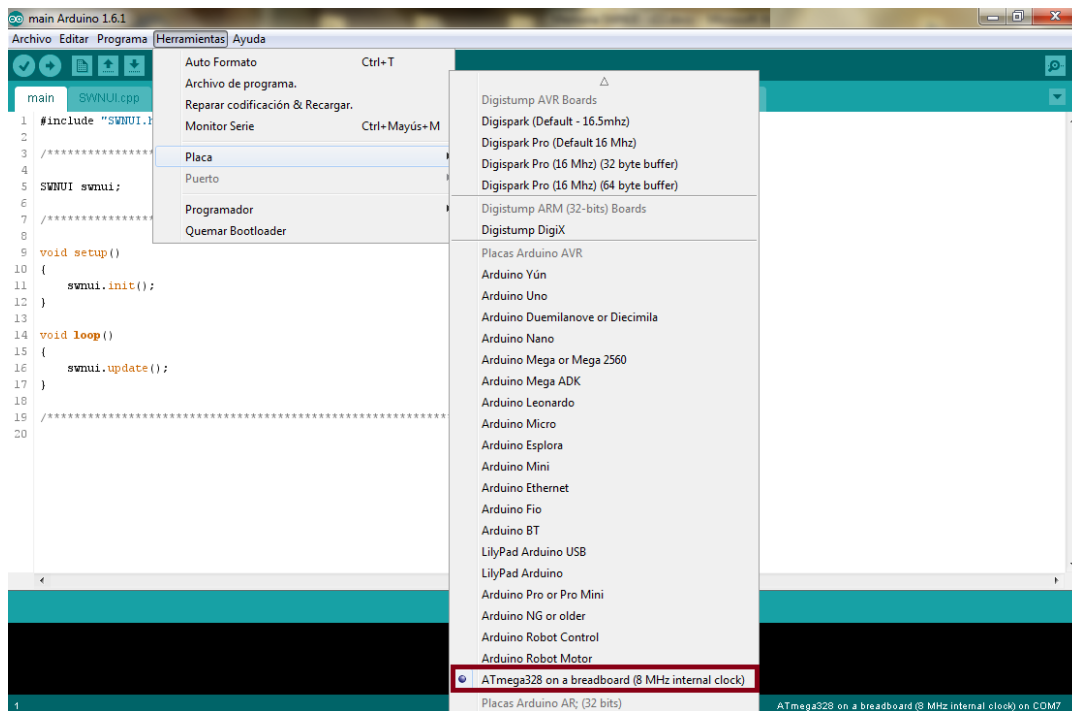


Figura 20. Configurando el dispositivo a programar.

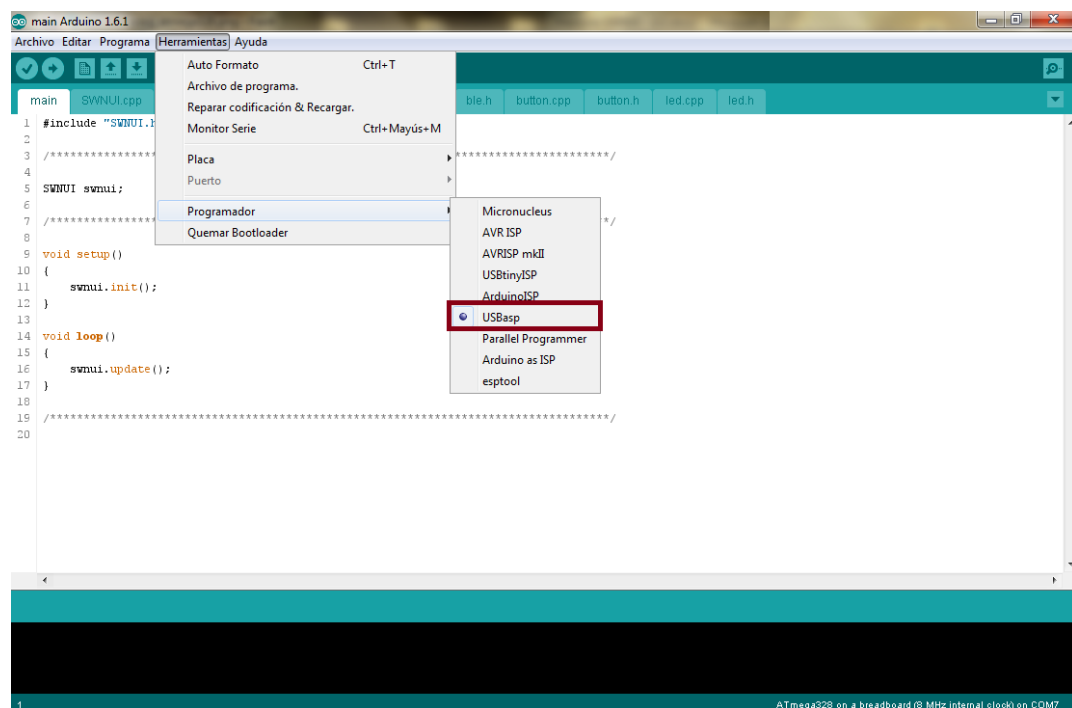


Figura 21. Configurando el programador a utilizar.

Una vez establecido tanto el microcontrolador como el programador a utilizar, se puede proceder a la programación del Bootloader (figura 22) y, una vez completado el proceso, se podrá utilizar la capa de abstracción hardware de Arduino para programar al sistema (figura 23).

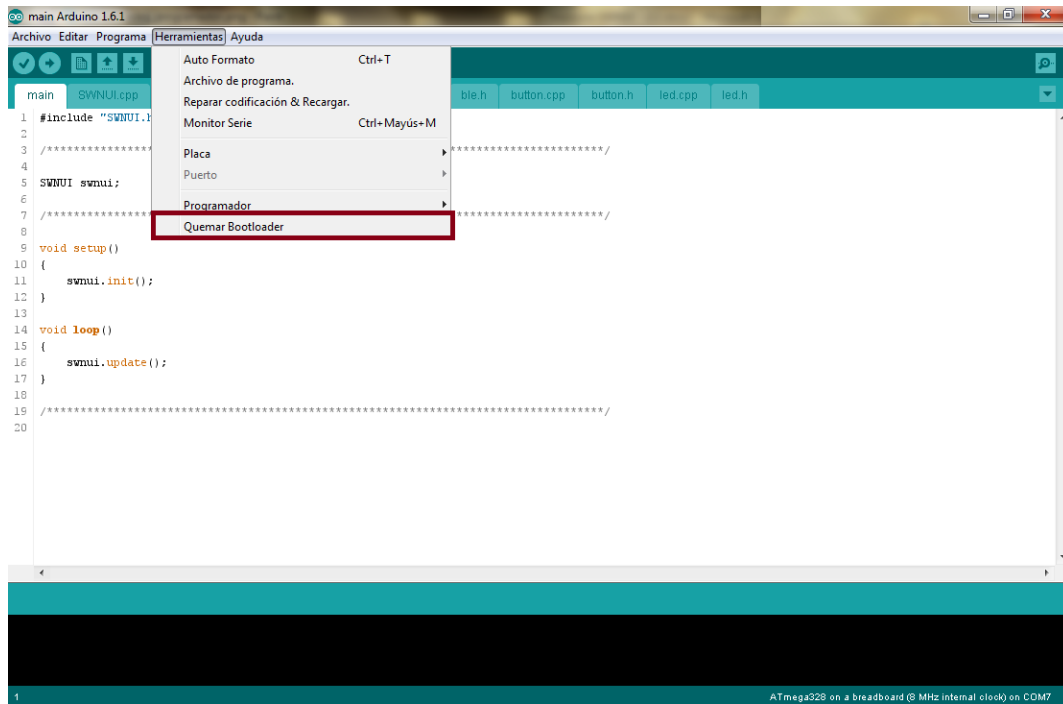


Figura 22. Quemando el Bootloader en el sistema.

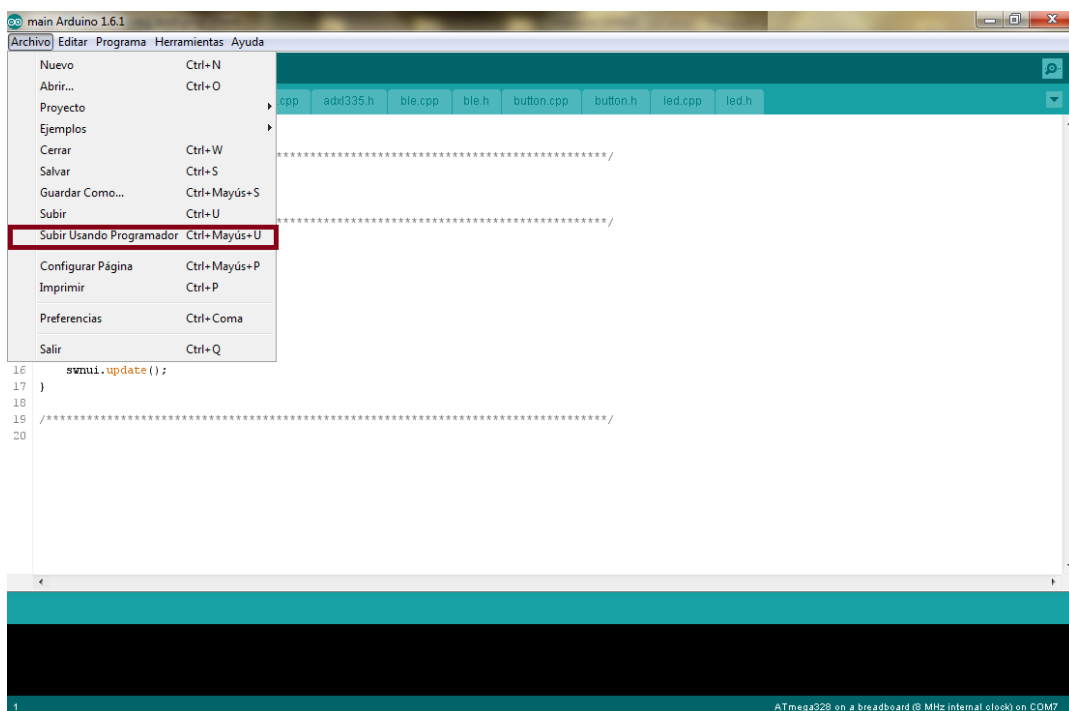


Figura 23. Carga de nuestro código en el sistema.

3.3.2. Software

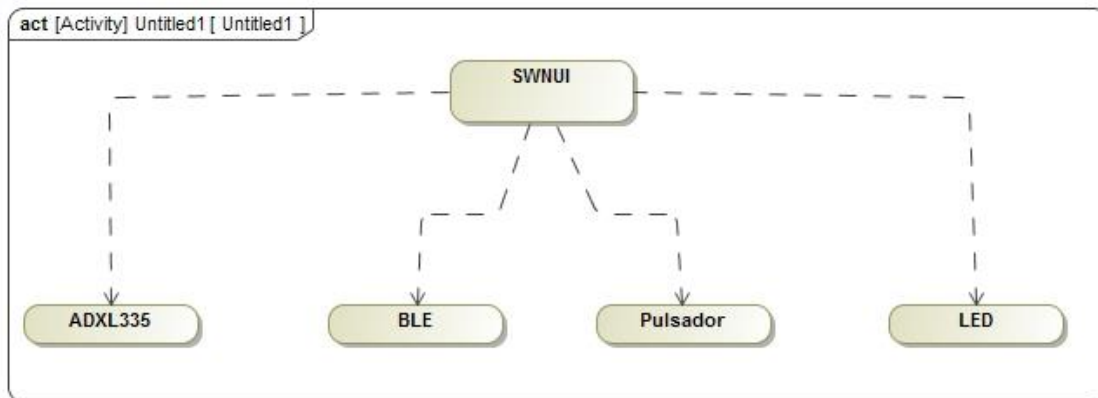


Figura 24. Clases y dependencias de nuestro sistema.

Buscando una estructura jerárquica del sistema, que nos permita ir desarrollando las distintas partes del mismo y a su vez ir verificando su funcionalidad, optamos por estructurar el código en distintas clases que controlen a cada elemento distinguible del sistema por separado (led, pulsador, módulo BLE y acelerómetro) y, una clase maestra (nuestro sistema completo, SWNUI) que controle a su vez las clases anteriores. Con esto, conseguimos un código organizado, al mismo tiempo que se enfoca al sistema como un “objeto software” único y exportable, lo que simplifica su importación y reutilización en distintos proyectos (a través de su librería). La estructura de las clases, así como la dependencia entre ellas, pueden ser observadas en la figura 24.

Con esta estructura del código basada en clases y su utilización para el desarrollo del “objeto software” diseñado que corresponde con nuestro sistema físico real, estamos haciendo uso de la “esencia” o “filosofía” de la programación orientada a objetos (POO). Este hecho, no solo especifica la organización o el tipo de metodología de programación, sino que nos lleva a un nivel un poco más alto en la programación del sistema.

La interacción entre las clases viene gobernada por la clase SWNUI, pasando por la gestión de la pulsación del pulsador mediante la clase PULSADOR, realizando la lectura de los datos del acelerómetro a través de la clase ADXL335, controlando el encendido o apagado del LED con la clase LED y enviando la información por puerto serie con la clase BLE.

Llegados a este punto, a continuación se especificará la funcionalidad de cada clase desarrollada y se explicarán brevemente los aspectos más significativos de su implementación.

Clase ADXL335:



Figura 25. Diagrama de flujo del proceso de muestreo.

Esta clase cumple la función del manejo del acelerómetro. Incluye tanto la configuración de los pines analógicos del microcontrolador como la lectura de cada eje del acelerómetro así como el filtrado de las muestras obtenidas y la determinación de los movimientos realizados y la posición del sistema.

La clase contiene un tipo de estructura (ADXL335_DATA) que almacena los distintos valores obtenidos en el proceso de muestreo de los datos (valores de lectura, aceleraciones en m/s^2 y en g, movimientos, posiciones y gestos).

La lectura de los valores de aceleración de cada eje del acelerómetro se realiza mediante el ADC (debido a la salida en tensión del acelerómetro analógico) y, para conseguir un valor más fiable, se promedian las lecturas. Según la sensibilidad del sistema a $\pm 3g$ se obtienen los valores de aceleración, en m/s^2 y en g, de cada muestra.

Las muestras atraviesan un proceso de filtrado en el que se elimina el ruido de la lectura (a través de umbrales que provocan cierta pérdida de precisión), se corrige el efecto de la aceleración de la gravedad (gravedad terrestre de 1g) y se elimina la captura consecutiva de muestras del mismo valor.

Pasado el proceso de filtrado, se procede a determinar el movimiento y la posición del sistema correspondientes a las muestras de aceleración actuales, finalizando el proceso de muestreo (figura 25) que comenzó con la lectura analógica del sensor.

Los métodos más destacables para el uso de esta clase serían los siguientes:

- `sampling()`: El método principal que realiza todo el proceso descrito anteriormente (muestreo). Correspondería con el proceso de adquisición, filtrado y determinación del par movimiento/posición. El método devuelve la estructura estándar de la clase que contiene todos los valores adquiridos durante el proceso de muestreo.
- `x()`, `y()`, `z()`, `ax()`, `ay()`, `az()`, `gx()`, `gy()`, `gz()`, `mx()`, `my()`, `mz()`, `pos()`, `mov()`: Todos ellos son métodos de consulta por separado de los valores almacenados en la estructura estándar tras el proceso de muestreo. Útil en el caso de que no se requiera extraer todos los datos de la estructura.

Clase BLE:

La clase BLE se encarga del proceso de comunicación inalámbrica del sistema a través del módulo BLE.

La clase configura la UART a utilizar y la velocidad en baudios que le corresponderá (en este caso 9600 baudios) a dicho puerto serie.

Los métodos más importantes en esta clase serían:

- `send(char c)`: El cual nos facilita el envío de bytes por el puerto serie.

- `print(String s)`: Método que permite el envío de cadenas de caracteres (adecuado sobre todo para la depuración del sistema).
- `println(String s)`: Otorga la misma funcionalidad que el método anterior pero introduce un fin de línea en la cadena de caracteres enviada.

Clase Button:

En esta clase se establece la lectura del estado del pulsador, esto es, si se encuentra presionado o no.

La clase se encarga de configurar el pin del microcontrolador correspondiente al pulsador. Esto es, pin como entrada digital con la resistencia interna de “pull-up” activada.

La rutina de tratamiento de interrupción (RTI) del pin correspondiente al pulsador no se encuentra definida dentro de ninguna clase (ya que no es posible definirla como método siendo de generación asíncrona). Dicha rutina se establece como global en el interior del fichero de implementación (archivo “.cpp”) de la clase SWNUI (a pesar de no formar parte de la misma).

Los únicos métodos de esta clase son:

- `state()`: Que nos devuelve el valor de entrada de lectura del pulsador.
- `pressed()`: Nos determina el estado del pulsador (si está presionado o no).

Clase LED:

La clase LED se encarga de gestionar el encendido o apagado del LED indicativo del sistema.

Al igual que la clase del pulsador, esta clase configura el pin del microcontrolador correspondiente. Pin como salida digital.

Los métodos más significativos de la clase son:

- `on()`: Enciende el LED.
- `off()`: Apaga el LED.
- `turn()`: Conmuta el estado del LED.
- `state()`: Determina si el LED está encendido o apagado.

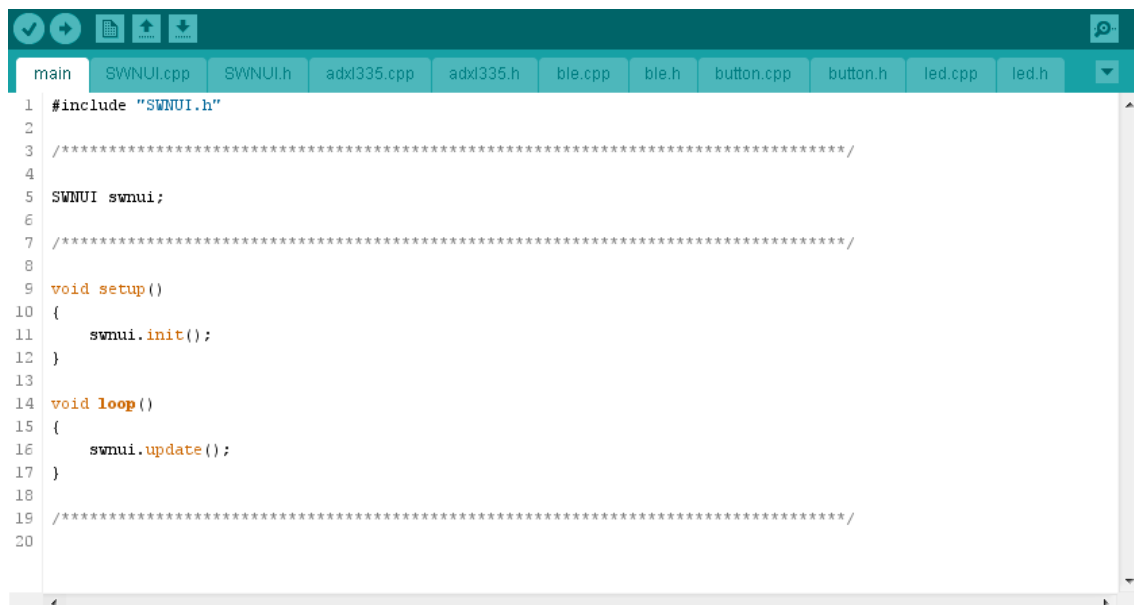
- `flash(unsigned long t)`: Hace parpadear al LED si ha transcurrido un tiempo `t` desde su última llamada.

Clase SWNUI:

Es la clase principal de nuestro sistema, contiene la referencia al resto de clases y éstas a su vez lo completan. Desde ella se tiene acceso a los elementos controlados por el resto de clases, permitiéndonos el manejo total de todos los dispositivos que conforman a nuestro sistema (microcontrolador, LED, Pulsador, Acelerómetro y BLE). De modo que se podría decir que esta clase es nuestro sistema.

Solucionados los mecanismos de muestreo de datos del acelerómetro, gestión del pulsador y control del LED en el resto de clases, solo quedaría el desarrollo del corazón de nuestro sistema, la funcionalidad del mismo.

Antes de entrar de lleno en el desarrollo de la funcionalidad, determinemos la interacción que tendrá nuestro objeto SWNUI en el programa principal. Para no perder la esencia de la programación orientada a objetos, se debe proceder a implementar todo el comportamiento del sistema dentro de su clase, así, en el programa principal exclusivamente debería existir una creación del objeto, una llamada a su configuración inicial y un método de actualización de su comportamiento (figura 26).



```
1 #include "SWNUI.h"
2
3 /*****
4
5 SWNUI swnui;
6
7 *****/
8
9 void setup()
10 {
11     swnui.init();
12 }
13
14 void loop()
15 {
16     swnui.update();
17 }
18
19 /*****
20
```

Figura 26. Programa principal.

Recordemos que el sistema deberá tener dos modos de funcionamiento, uno continuo y otro discreto, y se podrá conmutar entre ambos modos mediante la doble pulsación del pulsador. Teniendo eso en mente procedemos a programar dicha funcionalidad.

Inicialmente, correspondiente al método `init()`, la clase fija los valores iniciales de todas sus variables (entre ellas el modo inicial del sistema) e inicializa cada uno de sus objetos además de configurar el modo de bajo consumo y establecer la interrupción del pulsador (figura 27).

```
void SWNUI::init()
{
    mode = M_CONTINUOUS;
    state = INIT;

    ble = new BLE();
    led = new LED(P_LED, LOW);
    button = new BUTTON(P_BUTTON, INPUT_PULLUP);
    adxl335 = new ADXL335(P_AXIS_X, P_AXIS_Y, P_AXIS_Z);

    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    analogReference(EXTERNAL);
    button_setInterrupt();
}
```

Figura 27. Método `init()`.

Posteriormente a su inicio, en el método correspondiente a la actualización de su comportamiento, el método `update()`, el sistema entra en un bucle infinito en el que lo primero que hace es determinar el modo de funcionamiento a ejecutar según en el que se encuentre (figura 28).

```
void SWNUI::update()
{
    if(mode == M_CONTINUOUS)
        mode_continuous();
    else
        mode_gesture();
}
```

Figura 28. Método `update()`.

El “modo continuo” del sistema (figura 29) comprueba si ha existido una doble pulsación (entrada doble en la rutina de tratamiento de interrupción del pulsador en un tiempo inferior a un segundo) para el cambio de modo y en tal caso lo realiza (método `changeMode_management()`). En caso de que no se cambie de modo, se obtienen los datos correspondientes al muestreo del acelerómetro y se procede al envío de los datos correspondientes al movimiento en algún eje. Se hace parpadear al led a 1 Hz para indicarnos que se encuentra en el “modo continuo” y se espera un tiempo de muestreo (30 Hz).

```
void SWNUI::mode_continuous()
{
    changeMode_management();
    samples = adx1335->sampling();
    continuosSend();
    led_blink(1000);
    delay(T_SAMPLE);
}
```

Figura 29. Modo continuo.

El “modo gestos” del sistema es algo más complejo, dado que la funcionalidad tiene pasos bien definidos, se procede a desarrollar una máquina de estados que gestione cada parte del proceso. El diagrama de estados de dicha máquina es el mostrado en la siguiente imagen (figura 30):

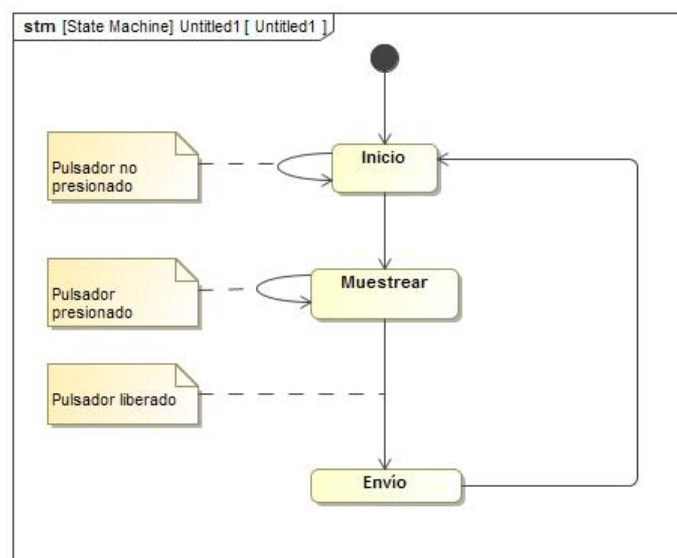


Figura 30. Máquina de estados del modo gestos.

La implementación de la máquina de estados en el código del programa se basa en la sentencia switch (figura 31). En el estado inicial, el sistema entra en bajo consumo hasta que el usuario presione el pulsador. Una vez despierto, el sistema espera un tiempo prudencial para permitir el cambio de modo (la segunda pulsación) y, al igual que en el modo continuo, se gestiona la posibilidad del cambio de modo. En caso de que no se produzca el cambio, el sistema pasará al estado de muestreo en el que, mientras se mantenga el pulsador presionado se irán almacenando en arrays los movimientos y posiciones que conformarán el gesto. Una vez soltado el pulsador, entraremos en el estado de envío en donde se determinará si el gesto corresponde a alguno predefinido y en tal caso se enviará el comando asociado.

```
void SWNUI::mode_gesture()
{
    switch(state)
    {
        default:
            state = INIT;
            break;

        case INIT:
            sleep();
            delay(CHANGE_MODE_TIME);
            changeMode_management();
            if(mode == M_GESTURE)
            {
                state = SAMPLE;
            }
            break;

        case SAMPLE:
            refreshGestureArrays();
            while(button_press())
            {
                samples = adxl_sampling();
                gestures();
                delay(T_SAMPLE);
            }
            state = SEND;
            break;

        case SEND:
            gestureRecognition();
            gestureSend();
            state = INIT;
            break;
    }
}
```

Figura 31. Modo gestos.

Con esto, tendríamos a grandes rasgos, el desarrollo del software que determina la funcionalidad que debe de tener nuestro sistema según las especificaciones del mismo. La finalización del software cubre otro de los pilares fundamentales en el proceso de desarrollo definitivo del prototipo que estamos elaborando y nos acerca un poco más hacia nuestra meta.

3.4. Desarrollo del modelo 3D de la carcasa

De forma complementaria al desarrollo hardware/software y aunque ya tenemos un prototipo funcional del sistema, parte del objetivo del sistema es ser un dispositivo “*vestible*” (portable en la mano) y, por tanto, nuestro prototipo no estaría completo hasta comprobar el correcto cumplimiento del requisito de diseño *wearable*. En este apartado, se procede a diseñar y desarrollar el modelo 3D de la carcasa del sistema (parte física/mecánica del sistema).

El software utilizado para el diseño CAD es *Freecad*, software que, como su propio nombre indica, es gratuito, además de *open source*. El proceso de diseño se basa en la creación de piezas aisladas para, posteriormente, fusionarlas entre sí hasta obtener el objeto completo.

A continuación se muestra el proceso de creación del objeto completo mediante la creación de cada pieza separada y la unión de estas (figura 32):

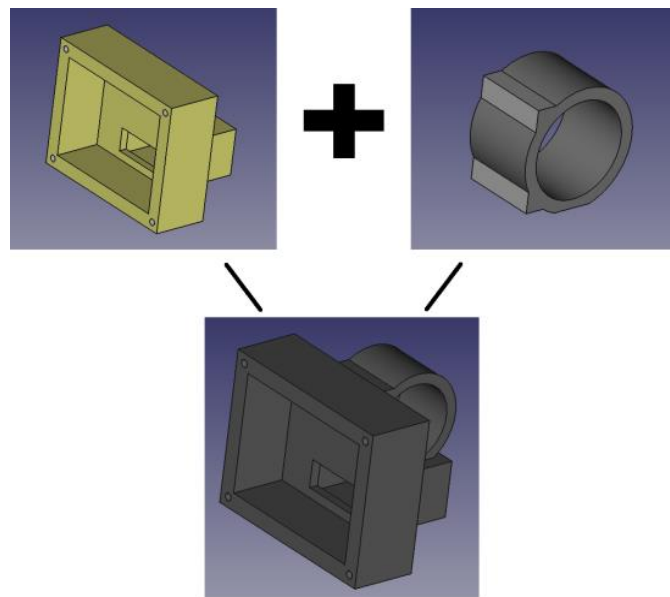


Figura 32. Piezas del SWNUI en el Freecad.

Por otro lado, se crea la tapa de la carcasa, la cual debe presentar los orificios adecuados en las posiciones requeridas para que los elementos de la placa sean accesibles. Así, los huecos corresponderán al pulsador, el LED y el conector ICSP de la PCI (figura 33).

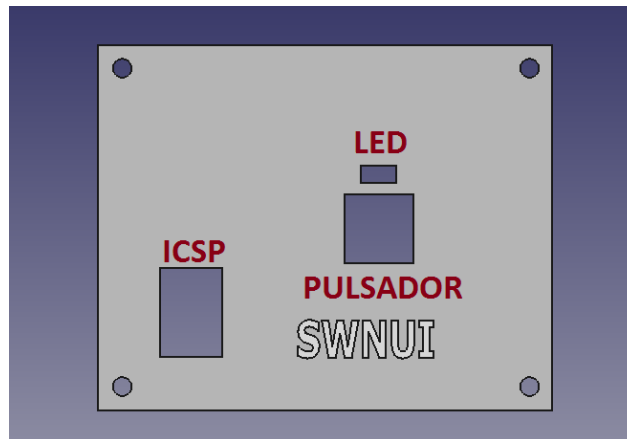


Figura 33. Tapa de la carcasa en Freecad.

Una vez que tenemos el diseño completo (figura 34), procedemos a exportar tanto la tapa como el resto de la carcasa en archivos “.obj” para su envío a fabricación (a partir de una impresora 3D).

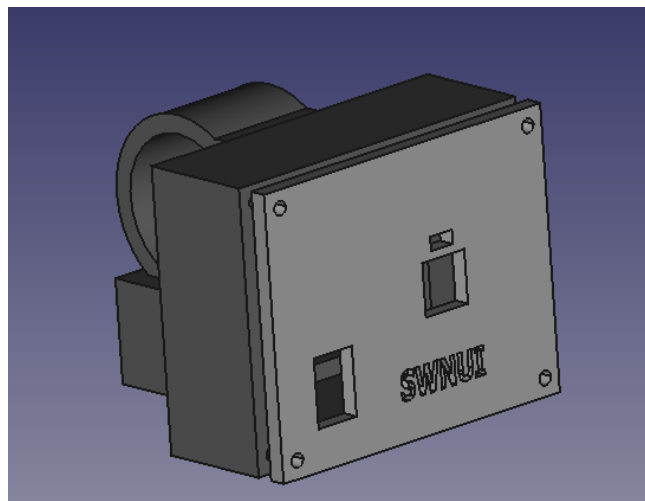


Figura 34. Diseño completo de la carcasa en Freecad.

Terminada la impresión de la pieza, obtenemos un buen resultado en el modelo realizado (figura 35). Los taladros correspondientes a los tornillos de sujeción

de la tapa tuvieron que ser repasados con una pequeña broca para eliminar impurezas que quedaron tras el proceso de fabricación.

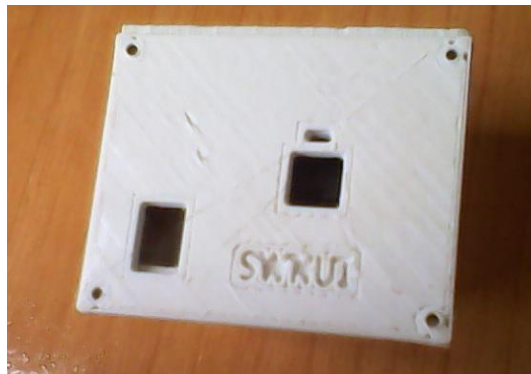
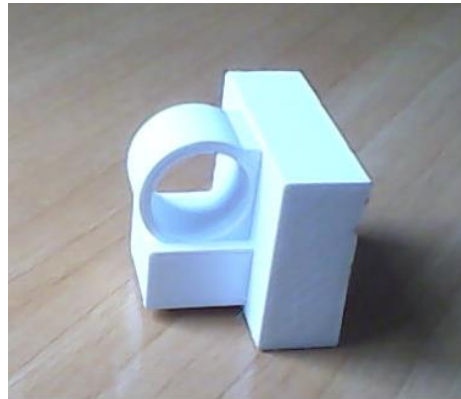
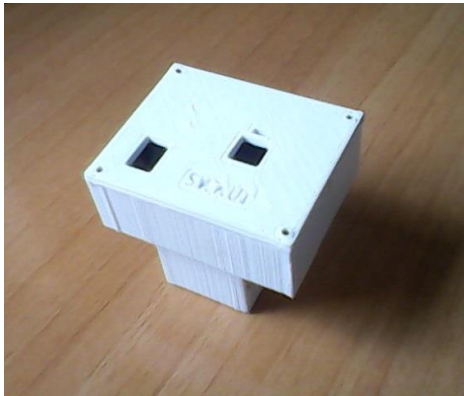


Figura 35. Carcasa impresa del SWNUI.

Una vez finalizado el desarrollo de la carcasa, solo restaría el ensamblaje del sistema dentro de la misma y su puesta en marcha.

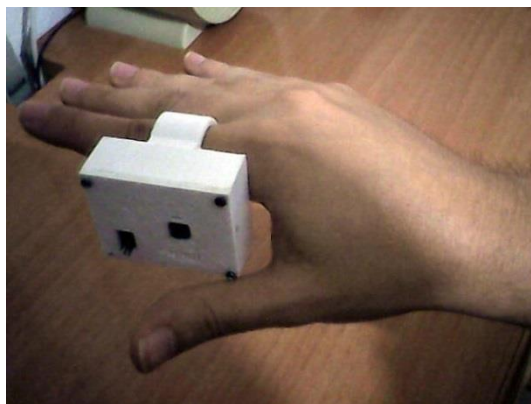


Figura 36. SWNUI completo.

Como puede observarse en la imagen anterior (figura 36), a pesar de su tosco tamaño debido a las limitaciones de las herramientas de fabricación (soldadura realizada a mano) y del hecho de que lo que se ha desarrollado es un prototipo, podemos concluir que el sistema cumple las características de un dispositivo “vestible”, que era una de las especificaciones iniciales impuestas en nuestro sistema.

Capítulo 4. Verificación y pruebas

El prototipo (SWNUI), desarrollado en el apartado anterior, tiene como finalidad permitir la interacción de un usuario con diferentes agentes o sistemas externos que presenten una interfaz compatible con el tipo de interacción que nos permite nuestro sistema. Esto implicaría la estandarización de los elementos a controlar para que sean capaces de reconocer las peticiones demandadas por la persona a través de los gestos realizados o, la modificación de estos elementos para añadirles esa compatibilidad hacia nuestro sistema.

En este apartado verificaremos el correcto funcionamiento de los dos modos de operación del SWNUI mediante el control de un PC. Para ello, se procederá a adaptar al mismo para que sea capaz de reconocer lo que el SWNUI transmite. Como se verá, esta adaptación no requiere de ninguna modificación en el hardware o software del PC ya que, todo ordenador presenta entradas USB.

4.1. Sistema de pruebas

Los ordenadores son quizás, las computadoras más genéricas existentes hasta el momento, su control se basa, sobre todo, en dos periféricos bien reconocidos: el ratón y el teclado. Ambos periféricos se conectan al PC mediante el *bus serie universal* (USB) y mediante un driver específico es reconocido por el ordenador como dispositivo de entrada.

Pues bien, nuestro sistema de prueba se basará en la creación de un elemento que actúe como periférico de entrada del ordenador a través de USB, simulando ser un ratón o un teclado. Para ello, recurriremos a un sistema desarrollado por *Digistump* denominado *Digispark* (figura 37), el cual no es más

que un microcontrolador attiny85 de *Atmel* que, a pesar de que no lleva implementado USB nativo, hace uso del firmware *V-USB* de *Atmel* para implementar vía software un controlador USB dentro de la misma unidad. Esto permite al microcontrolador ser reconocido por el ordenador como un dispositivo HID-USB (*Human Interface Device USB*) genérico. Los dispositivos HID-USB genéricos son reconocidos por la computadora gracias a un driver que viene integro en la mayoría de sistemas operativos (Windows, Unix, Android, etc.) y que permite detectar al periférico como dispositivo de interfaz humana: ratón, teclado, gamepad...

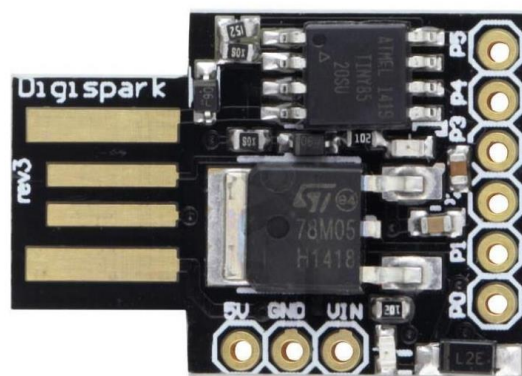


Figura 37. Digispark.

Podríamos conseguir el mismo resultado a partir de cualquier sistema que se conectará por el puerto USB y lo convirtiera a éste en un puerto serie virtual por el que enviar la información, pero esto requeriría de algún software en la computadora que se debiera ejecutar cada vez que el dispositivo se conectara, o que corriera en segundo plano ocupando memoria del sistema. El uso del *Digispark* nos permite crear un enlace que convierte al SWNUI en un auténtico periférico de ordenador, reconocible de forma automática en cuanto se conectara al mismo.

Por tanto, el sistema de prueba lo conformará el Digispark junto con un módulo BLE, que permita recibir los datos del SWNUI e interpretarlos para el control del ratón/teclado. Así, el sistema completo vendría dado como se muestra en la figura 38.

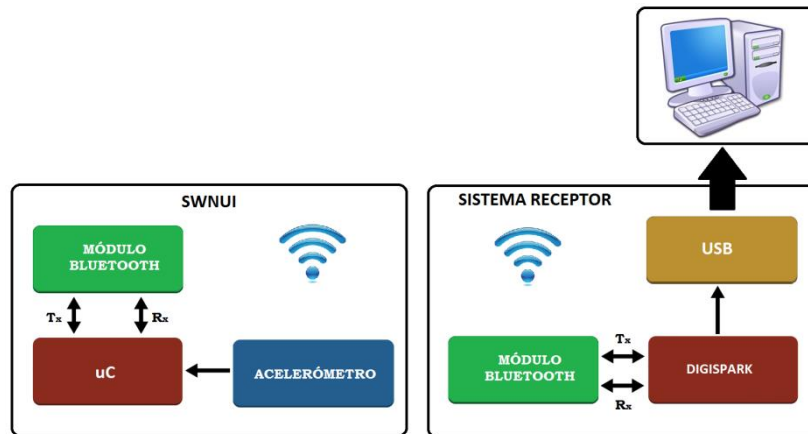


Figura 38. Sistema de verificación completo.

El Digispark es un sistema desarrollado posteriormente a la plataforma Arduino y que buscaba ser una solución de menor tamaño y precio, compatible con toda la capa de abstracción hardware creada para Arduino, y que otorgaba esa gran simplicidad a la hora de desarrollar aplicaciones. Por este motivo, el Digispark es un sistema que se programa con la misma filosofía que un Arduino y por ello, el software de programación no es otro que el IDE de Arduino, pero previamente debemos configurarlo para añadirle los componentes correspondientes a la programación del Digispark (archivos, “cores” y compilador).

El proceso de inserción de los componentes correspondientes al Digispark en el entorno de programación de Arduino se basa en la descarga de la carpeta “digistump” desde la web: <https://github.com/digistump/DigistumpArduino> y su inclusión dentro de la subcarpeta “hardware” del directorio raíz de Arduino (C:/.../Arduino.X.X.X/hardware). Esto otorga al IDE la capacidad de seleccionar como plataforma de programación las diversas placas de Digistump, entre ellas la Digispark, e inserta el compilador específico de esta plataforma (*Micronucleus*).

El microcontrolador que compone al Digispark presenta poca capacidad en memoria, pero es más que suficiente para cumplir con nuestro propósito. A pesar de ello, algo esencial que nos faltaría en el Digispark sería un puerto serie (el attiny85 no posee UART) para comunicarnos con el módulo BLE. Una

solución a esto es implementar una UART virtual dentro del microcontrolador, de modo que recurrimos al uso de la librería *SoftSerial* (Software Serial) diseñada para el Digispark. Esta librería nos permite crear un puerto serie en los pines compatibles, en nuestro caso los pines P0 (Rx) y P1 (Tx).

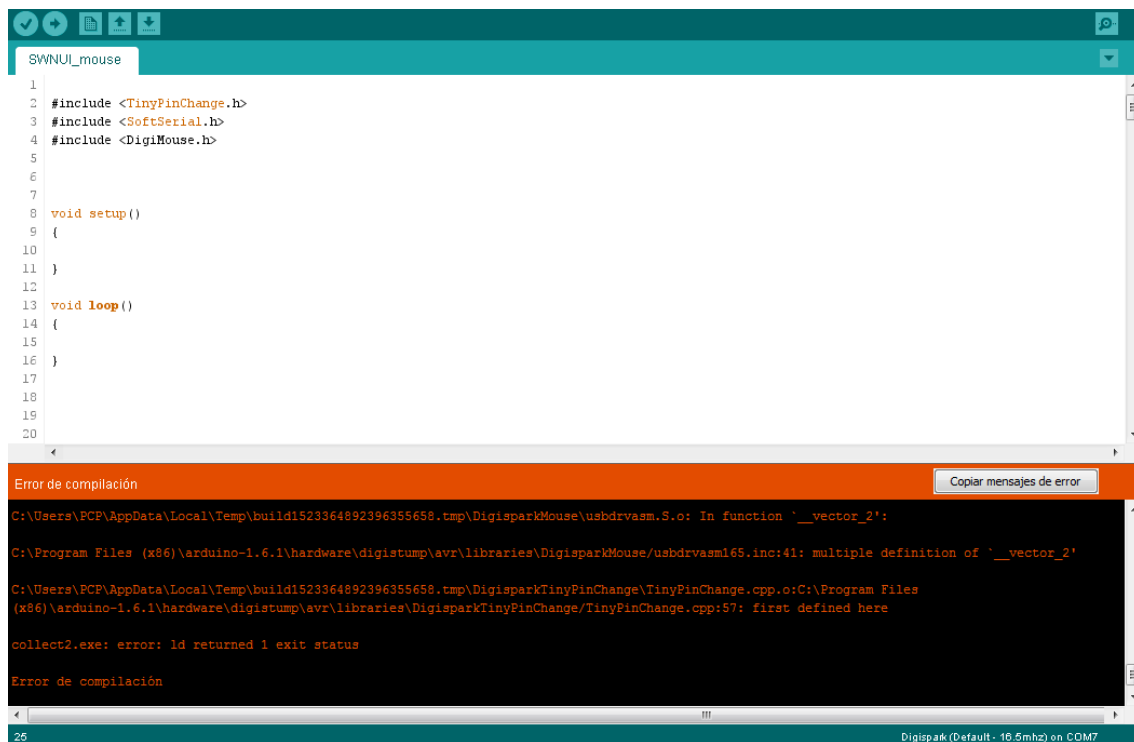
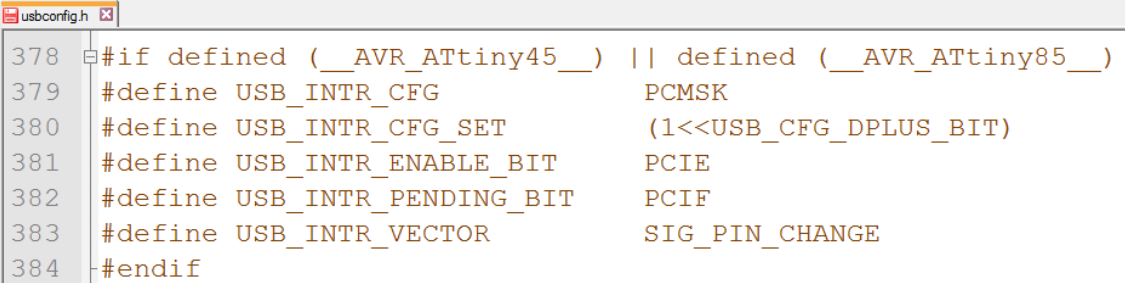


Figura 39. Error de definición múltiple del `__vector_2`.

La inserción de la nueva librería *SoftSerial* junto al uso de las librerías correspondientes a la creación y manejo del periférico USB (*DigiUSB*) plantea un nuevo problema ya que, aunque por separado las dos librerías funcionan de forma adecuada, el uso de ambas al mismo tiempo genera un error en la etapa de compilación por definición múltiple del “`__vector_2`”, correspondiente al vector de “Pin Change interrupt” del pin PCINT0 (figura 39). Esto se debe a que, por un lado, la librería *SoftSerial* hace uso del mismo para la detección de datos entrantes por el pin Rx del puerto serie creado, mientras que la librería *DigiUSB*, a pesar de que no utiliza exactamente ese pin para el puerto USB, presenta una definición genérica de las interrupciones “Pin Change” dentro del archivo de configuración del dispositivo USB “`usbconfig.h`”.

La solución a este problema pasa por identificar la definición genérica de las interrupciones de “Pin Change” para el USB en el archivo “usbconfig.h” correspondiente a cada librería de manejo del ratón y el teclado (Digimouse y Digikeyboard) y modificarla para que se utilice la interrupción hardware INT0 (correspondiente al pin P2 del Digispark) en vez de la PCINT0.

El archivo “usbconfig.h” se puede encontrar dentro de las carpetas DigisparkMouse y DigisparkKeyboard en la ruta: “.../Arduino-X.X.X/hardware/digistump/avr/libraries”. El código que nos interesa se localiza entre las líneas 378 y 384 del archivo (figura 40) y debe modificarse tal y como se muestra en la figura 41.

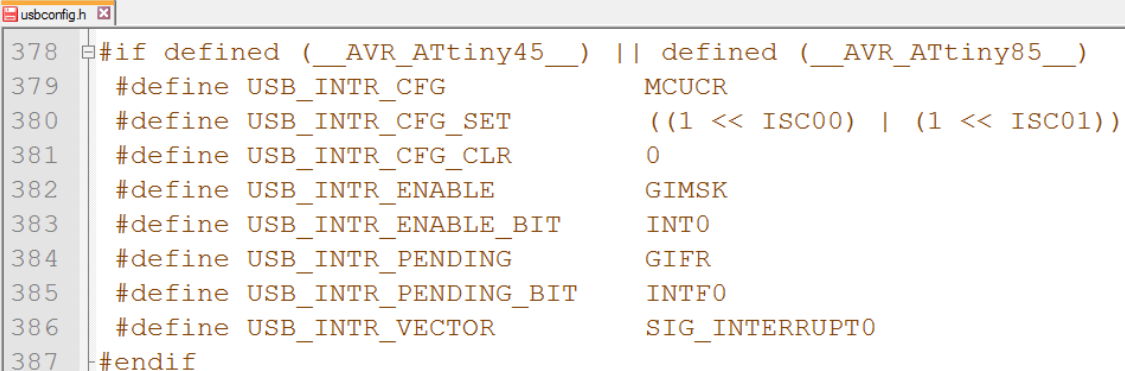


```

378 #if defined (__AVR_ATtiny45__) || defined (__AVR_ATtiny85__)
379 #define USB_INTR_CFG          PCMSK
380 #define USB_INTR_CFG_SET      (1<<USB_CFG_DPLUS_BIT)
381 #define USB_INTR_ENABLE_BIT   PCIE
382 #define USB_INTR_PENDING_BIT  PCIF
383 #define USB_INTR_VECTOR       SIG_PIN_CHANGE
384 #endif

```

Figura 40. Código de definición de las interrupciones en usbconfig.h.



```

378 #if defined (__AVR_ATtiny45__) || defined (__AVR_ATtiny85__)
379 #define USB_INTR_CFG          MCUCR
380 #define USB_INTR_CFG_SET      ((1 << ISC00) | (1 << ISC01))
381 #define USB_INTR_CFG_CLR      0
382 #define USB_INTR_ENABLE       GIMSK
383 #define USB_INTR_ENABLE_BIT   INT0
384 #define USB_INTR_PENDING      GIFR
385 #define USB_INTR_PENDING_BIT  INTF0
386 #define USB_INTR_VECTOR       SIG_INTERRUPT0
387 #endif

```

Figura 41. Código modificado de definición de la interrupción INT0 en usbconfig.h.

Modificar la librería de tal forma puede no resultar la solución más elegante, además, se pierde el pin P2 del Digispark, pues este es el correspondiente a la interrupción INT0 que hemos establecido como interrupción de la línea USB+ y, por ende, debe de cortocircuitarse con el pin correspondiente (P3).

De este modo, la conexión del Digispark con el módulo BLE vendrá dada de la forma en que se muestra en la figura 42.

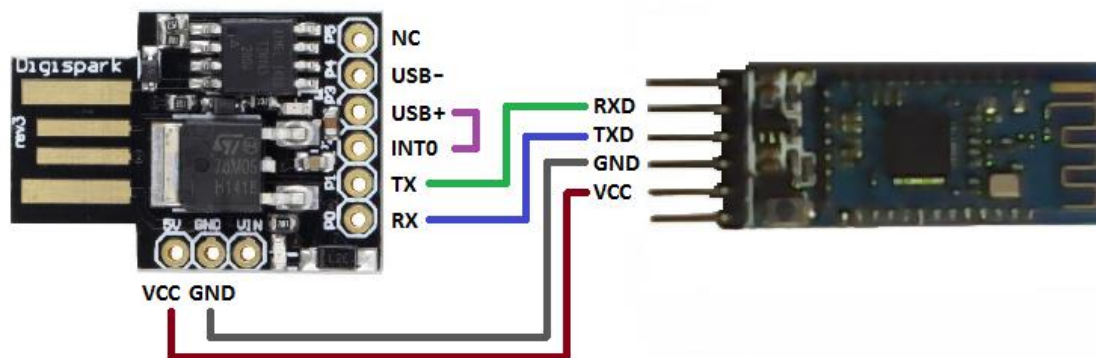


Figura 42. Código modificado de definición de la interrupción INT0 en usbconfig.h.

Para el proceso de conexión de los elementos que conforman al sistema de pruebas, se procede mediante la utilización de una placa de baquelita perforada, en la cual se suelda los conectores apropiados y se realizan las conexiones pertinentes de los dos elementos. Quedando el sistema de pruebas como se muestra en la figura 43.

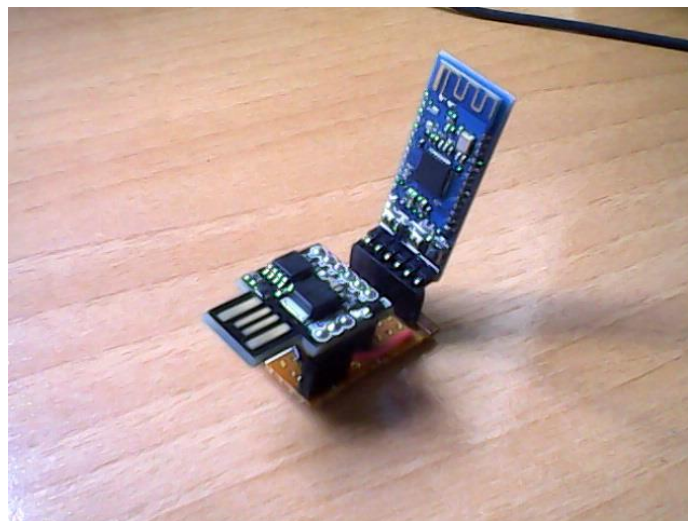


Figura 43. Sistema de pruebas. Periférico receptor del SWNUI.

Realizado todo lo anterior, tendríamos al sistema de pruebas disponible para la creación del software necesario para la recepción e interpretación de los datos comunicados por el SWNUI.

4.2. Pruebas realizadas

4.2.1. Pruebas de funcionalidad

Para comprobar la adecuada funcionalidad del SWNUI, procederemos a controlar un ordenador con sistema operativo Windows 7. Para ello, realizaremos dos tipos de controles. Por un lado, procederemos a implementar el control del cursor de la computadora con el fin de corroborar la correcta operación del “Modo continuo” del sistema, permitiendo desplazar al cursor por la pantalla y gestionar el click izquierdo. Por otra parte, implementaremos el control del programa Power Point de Microsoft Office permitiéndonos navegar entre las diapositivas y entrar y salir del modo presentación, esto lo haremos configurando al periférico como si fuera un teclado y, nos ofrecerá la oportunidad de comprobar el correcto funcionamiento del “Modo gestos” del sistema.

Prueba del “Modo continuo”: SWNUI como cursor del ordenador

En la prueba de verificación del “modo continuo”, el Digispark es conectado al ordenador. Pasado un cierto tiempo de inicialización, es reconocido como ratón HID-USB y comienza a buscar e intentar conectarse al SWNUI. Una vez conectado, se verifica que la inclinación del SWNUI en cada una de las direcciones produce el desplazamiento del cursor en pantalla, así como que se genera el Click en el momento en que se desplaza la mano en dirección al



Posición estándar

(El cursor no se
desplaza)



Inclinación derecha

(El cursor se desplaza
hacia la derecha)



Inclinación izquierda

(El cursor se desplaza
hacia la izquierda)



Inclinación arriba
(El cursor se desplaza
hacia arriba)



Inclinación abajo
(El cursor se desplaza
hacia abajo)

El código presente en el Digispark para controlar el movimiento del cursor del ordenador se corresponde al diagrama de actividad siguiente (figura 44).

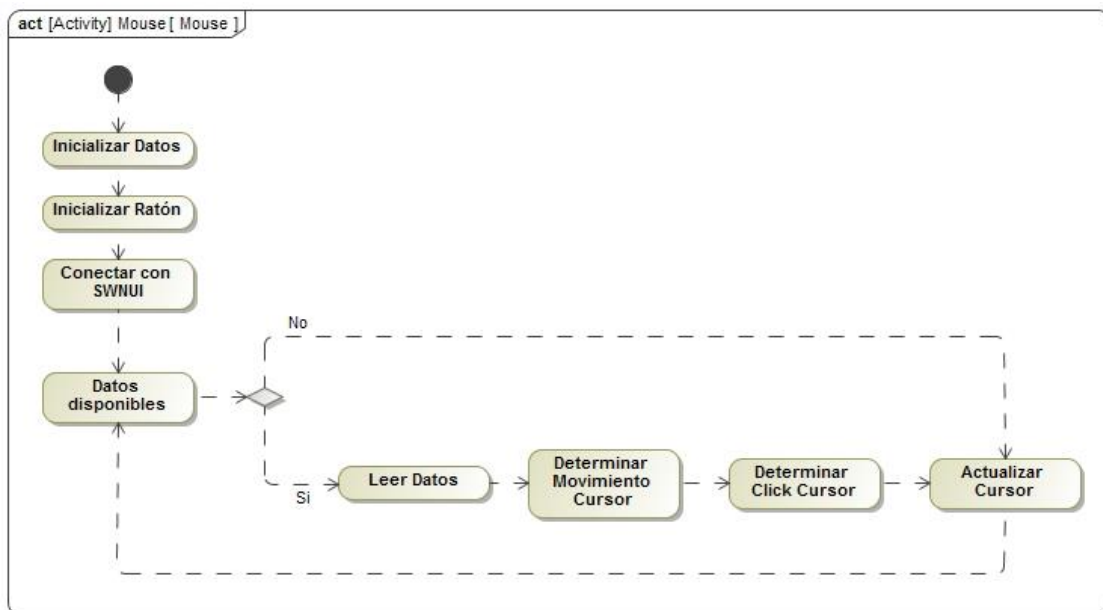


Figura 44. Diagrama de actividad del control del cursor.

A continuación, procedemos a definir la utilidad de cada bloque funcional:

- Inicializar Datos: Establece los valores iniciales de las variables del sistema.
- Inicializar Ratón: Establece la conexión USB con el ordenador como ratón HID.

- Conectar con SWNUI: Se conecta a través del módulo BLE con el SWNUI.
- Datos disponibles: Determina la existencia de bytes disponibles en el “buffer” del puerto serie (datos transmitidos por el SWNUI).
- Leer datos: Lee y almacena en variables los datos obtenidos por el puerto serie.
- Determinar movimiento cursor: Determina la dirección y velocidad de movimiento del cursor a partir de los datos obtenidos.
- Determinar click del cursor: Determina si ha de producirse un click.
- Actualizar cursor: Mantiene con vida la conexión USB y actualiza el estado (movimiento y click) del cursor.

Prueba del “Modo gestos”: SWNUI como teclado del ordenador

En la prueba de verificación del “modo gestos”, el Digispark es conectado al ordenador. Pasado un cierto tiempo de inicialización, es reconocido como teclado HID-USB y comienza a buscar e intentar conectarse al SWNUI. Una vez conectado, se verifica en principio que, ningún movimiento del SWNUI es reconocido por el ordenador mientras el pulsador no se encuentre presionado. Procediendo como se especifica en el “modo gestos”, se presiona el pulsador y se realizan los conjuntos de movimientos preestablecidos que componen a los diversos gestos, al dejar de presionar el pulsador se puede comprobar como cada uno de estos gestos es reconocido por el ordenador de forma correcta.

Cada gesto es asociado, en el código del Digispark, a una tecla del teclado adecuada para controlar la presentación de Power Point en su totalidad. De este modo, esta asociación viene a ser como sigue:

Gesto	Tecla	Función
Next	KEY_ARROW_RIGHT	Siguiente diapositiva
Prev	KEY_ARROW_LEFT	Anterior diapositiva
OK	F5	Modo presentación
CANCEL	ESC	Salir presentación

El código albergado por el Digispark que permite controlar la pulsación de teclas del ordenador se corresponde al siguiente diagrama de actividad (figura 45).

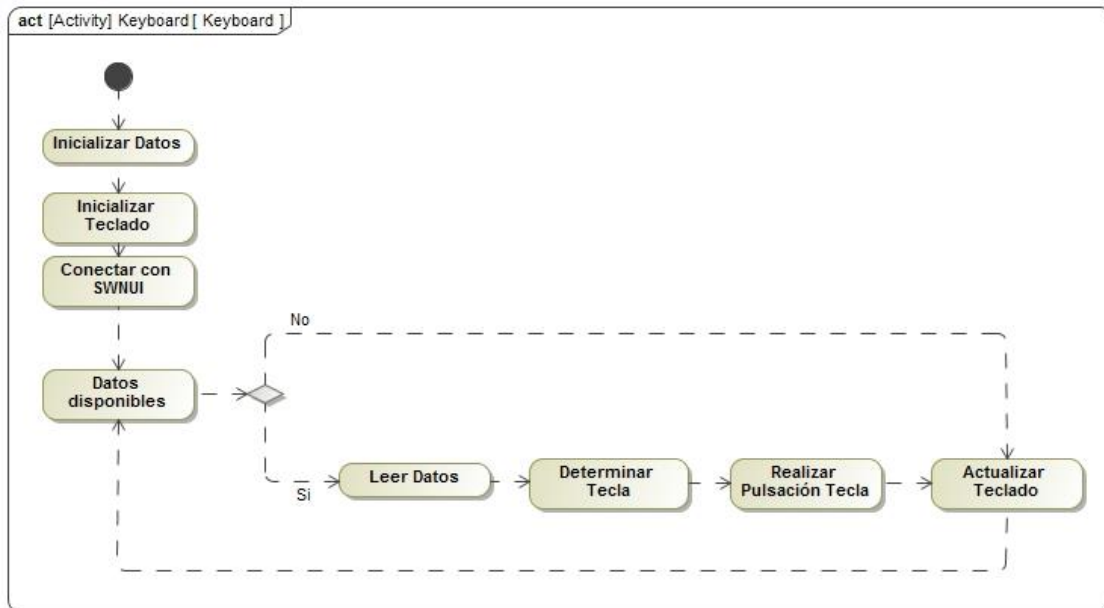


Figura 45. Diagrama de actividad del control del teclado.

De modo que la utilidad de cada bloque funcional sería:

- Inicializar Datos: Establece los valores iniciales de las variables del sistema.
- Inicializar Teclado: Establece la conexión USB con el ordenador como teclado HID.
- Conectar con SWNUI: Se conecta a través del módulo BLE con el SWNUI.
- Datos disponibles: Determina la existencia de bytes disponibles en el “buffer” del puerto serie (datos transmitidos por el SWNUI).
- Leer datos: Lee y almacena en variables los datos obtenidos por el puerto serie.
- Determinar tecla: Determina la tecla correspondiente al gesto recibido.
- Realizar pulsación tecla: Genera la pulsación de la tecla correspondiente.
- Actualizar teclado: Mantiene con vida la conexión USB.

Prueba de cambio de modo

La conmutación entre los dos modos de funcionamiento se realiza mediante la doble pulsación del pulsador del SWNUI en un periodo inferior a un segundo. Procedemos a realizar dicha operación tanto desde el “modo continuo” como desde el “modo gestos”, consiguiéndose lo deseado (la conmutación) en ambos modos. No obstante, se observa que la conmutación hacia el “modo continuo” desde el “modo gestos” genera una mayor dificultad debido al estado de bajo consumo del cual ha de despertarse el sistema previamente a la gestión del cambio de modo.

4.2.2. Pruebas de prestaciones

Prueba de diseño wearable

Con un calibre se procede a examinar las dimensiones de la carcasa diseñada y que contiene al SWNUI. El resultado dado corrobora que el diseño no supera las dimensiones de 5 cm en cualquier dirección del espacio.

A pesar de su tosco tamaño, se aprecia que el diseño “prototipado” se adecua a la portabilidad del mismo por parte del usuario, consigue no ser demasiado pesado y no suele entorpecer los movimientos del portador.

Prueba de indicación de modo

De manera visual se puede apreciar como el LED indicativo del sistema SWNUI parpadea con una frecuencia de 1 Hz mientras el modo de funcionamiento es el continuo y que, al conmutar de modo, el LED pasa a un estado apagado mientras se esté en el bajo consumo del “modo gestos”. El LED pasa a encenderse en el momento en el que se presiona el pulsador para realizar el conjunto de movimientos que determinan un gesto y, el LED vuelve a apagarse en cuanto se suelta el pulsador (entrando nuevamente en bajo consumo).

4.2.3. Tabla resumen de pruebas

Nº	Nombre o ID del requisito	Descripción de la prueba	Resultado
1	Función del sistema	Verificado mediante pruebas de funcionalidad	Favorable
2	Comunicación inalámbrica	Verificado mediante pruebas de funcionalidad	Favorable
3	Conexión automática	Verificado mediante pruebas de funcionalidad	Favorable
4	Modo continuo	Verificado por prueba de funcionalidad de modo continuo	Favorable
5	Modo gestos	Verificado por prueba de funcionalidad del modo gestos	Favorable
6	Cambio de modo	Verificado por prueba de funcionalidad de cambio de modo	Favorable
7	Diseño <i>Wearable</i>	Verificado por prueba de prestación de diseño <i>wearable</i>	Favorable
8	Indicación de modo	Verificado por prueba de prestación de indicación de modo	Favorable

Tabla 4.2.3.1 Tabla de las pruebas realizadas

Capítulo 5. Manual de instalación y uso

5.1. Manual de instalación

El SWNUI es un sistema portable diseñado para su colocación en el dedo índice del usuario (figura 46) y, el dedo pulgar permitirá la pulsación del pulsador.

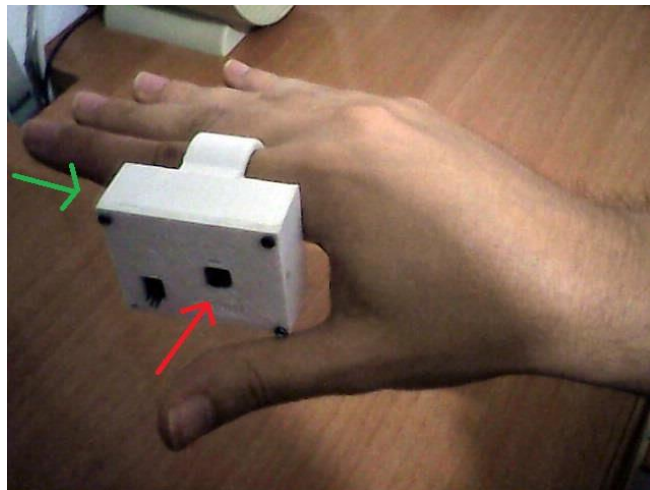


Figura 46. Instalación del SWNUI en la mano.

Para la comunicación inalámbrica el sistema utiliza *Bluetooth Low Energy*, con las limitaciones de distancia que ésta conlleva. La distancia adecuada para que el SWNUI pueda conectarse e interactuar con los sistemas externos debe de ser inferior a los 10 metros.

Respecto a los sistemas externos a los que se conectará el SWNUI, estos tendrán que presentar la interfaz de conexión e interpretación de los gestos, así como un menú, adecuado al SWNUI.

5.2. Manual de uso

5.2.1. Configuración inicial

Para comenzar, el SWNUI se deberá posicionar de forma correcta en el dedo del usuario (véase el apartado anterior, de instalación). Inicialmente, se encontrará en el modo de funcionamiento correspondiente al envío de gestos de forma aislada, el “modo gestos”, y no se requiere de ninguna configuración extra para comenzar a utilizarlo.

5.2.2. Conexión con el sistema a controlar

El SWNUI es un sistema simple por definición. Así, una vez se encuentre dentro del rango de un sistema de control, se establecerá la conexión inalámbrica de forma automática, sin que el usuario deba intervenir.

5.2.3. Cambio de modo

El modo de funcionamiento del SWNUI podrá conmutarse mediante la doble pulsación del botón en un periodo inferior a un segundo.

5.2.4. Modo continuo

El manejo continuo del sistema no es exclusivamente simple, sino que también es intuitivo. El usuario puede observar de forma directa (realimentación visual) el resultado de su interacción con el sistema a controlar.

En el caso del control del cursor de un ordenador, la inclinación de la mano determinará la dirección y velocidad de movimiento del mismo y un desplazamiento rápido en la dirección del suelo cumplirá la función del *click*.

5.2.5. Modo gestos

El “modo gestos” es el modo estándar por defecto del SWNUI, a través del cual el usuario puede interaccionar y controlar sistemas que requieran de acciones específicas independientes del tiempo, como puede ser la navegación dentro de un menú (avanzar, retroceder, aceptar, cancelar) de un sistema de consulta

de información (Pantalla Informativa) o un terminal punto de venta PTV (por ejemplo, equipos de venta de tickets en servicios de transporte, como el metro o el autobús).

Para la realización de un gesto se procede de la siguiente forma:

1. Se posiciona la mano en la posición inicial del gesto.
2. Se presiona el pulsador (se inicia el proceso de muestreo).
3. Se realizan los movimientos e inclinaciones que componen al gesto.
4. Se suelta el pulsador (se finaliza el proceso de muestreo y, se determina y envía el gesto correspondiente).

5.2.6. Consumo

El sistema entrará en un estado de bajo consumo, única y exclusivamente en el “modo gestos”, por tanto se debe evitar la utilización del “modo continuo” siempre y cuando no se requiera de su uso.

5.2.7. Movimientos y posiciones posibles

El SWNUI registra una gran variedad de posiciones y movimientos del mismo, las cuales a su vez, componen los distintos gestos preestablecidos.

A continuación se recogen las diversas posiciones reconocibles por el sistema:



Posición boca-arriba



Posición derecha



Posición izquierda



Posición boca-abajo



Posición arriba



Posición abajo

Por otro lado, los movimientos reconocibles serían, obviamente: Desplazamientos hacia la derecha, izquierda, arriba, abajo, adelante, atrás y el no desplazarse.

5.2.8. Gestos predeterminados

El conjunto de varios movimientos e inclinaciones reconocibles por el sistema (apartado anterior) pueden determinar uno de los gestos predefinidos en el sistema.

Existen cuatro gestos predeterminados que corresponden al control de interfaces del tipo: avanzar, retroceder, aceptar, cancelar.

A continuación se recopilan los movimientos y posiciones que componen a cada uno de estos gestos:

Gesto Avanzar:



1 - Posición estándar

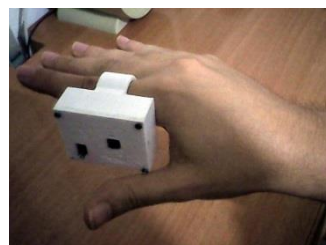


2 - Posición derecha

Gesto Retroceder:

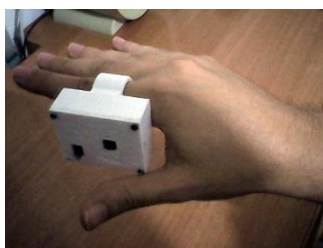


1 - Posición derecha



2 - Posición estándar

Gesto Aceptar:



1 - Posición estándar



2 - Posición derecha



3 - Posición abajo

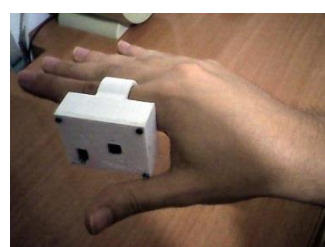
Gesto Cancelar:



1 - Posición abajo



2 - Posición derecha



3 - Posición estándar

Como puede observarse, estos gestos predeterminados son conformados exclusivamente por posiciones de la mano con el fin de simplificar al máximo el movimiento que deba realizar el usuario.

Capítulo 6. Conclusiones y trabajo futuro

Dentro del ámbito de los sistemas de interacción natural de usuario no ópticos, el prototipo del sistema desarrollado en este TFG nos ha permitido considerar diferentes posibilidades de interacción basada en gestos, con las diversas interfaces de usuario que se presentan hoy en día en la multitud de tecnologías cotidianas que nos rodean.

El objetivo del proyecto realizado nos ha llevado a ahondar en los diferentes sensores de medición inercial basados en tecnología MEM, con el fin de determinar qué sensor sería el más apropiado para nuestra aplicación según los requerimientos de nuestro sistema y el realizar el apropiado procesamiento de los datos obtenidos del mismo. Y de esta forma, acabamos determinando que exclusivamente con un acelerómetro podríamos alcanzar nuestro objetivo. Por otro lado, la necesidad de comunicación inalámbrica nos permitió profundizar en el uso de la tecnología *Bluetooth Low Energy* y aprendimos ciertas características técnicas al respecto (como es la ventaja del poco consumo, así como la limitación en velocidad con respecto al protocolo *Bluetooth* estándar).

El hecho de afrontar el proceso de creación de un prototipo para nuestro sistema, nos ha permitido abordar todos los pasos que definen la fabricación de un sistema electrónico completo, pasando por el desarrollo hardware y software, y acabando con la creación de su carcasa (desarrollo mecánico). Todo ello nos ha posibilitado aprender el manejo de los software de diseño asistido por computadora Eagle y Freecad, y hemos aprendido a exportar los archivos necesarios tanto para la fabricación de la placa de circuito impreso (ficheros Gerbers y Excellons), como para el diseño 3D de la carcasa (archivos .obj).

Evidentemente, transformar el sistema en producto desde el prototipo que es, llevaría a priorizar una mejora en la parte mecánica del mismo, buscando una

reducción mucho mayor del tamaño y el peso así como incrementar la robustez física del sistema.

El sistema elaborado cumple con su función, no obstante, presenta ciertas limitaciones a la hora de determinar los gestos (entre otros, gestos complicados pueden no ser apropiadamente reconocibles y puede producirse confusión entre gestos). Un sensor de bajas prestaciones como el acelerómetro utilizado presenta una gran cantidad de ruido que debería, más que ser filtrado (como hemos hecho), ser tratado y compensado mediante la inserción de algún otro elemento sensor de movimiento, como un giroscopio, con el fin de eliminar posibles errores en las muestras obtenidas al mismo tiempo que se aumenta la precisión y sensibilidad de las mismas, consiguiendo así, unos gestos reconocibles mucho más sofisticados. La inserción del nuevo componente hardware y su aplicación para la determinación de gestos mejorados, nos llevaría a la implementación software de ciertos algoritmos que permitan la detección, más profunda, de los nuevos gestos más complejos. Respecto a lo anterior, se ha de tener presente, que la mejora del sistema en esta dirección debe seguir la filosofía original del proyecto en cuanto a su simplicidad para el usuario.

Apéndice A. Presupuesto de elaboración

Componente	Coste unitario (€)	Unidades empleadas	Coste total (€)
ATMEGA328P	1.59 €	1	1.59 €
ADXL335	1.68 €	1	1.68 €
BLE CC2541 Bolutek Module	4.19 €	1	4.19 €
Push button	0.055 €	1	0.06 €
Led SMD	0.05 €	1	0.05 €
Resistencia SMD	0.0242€	2	0.05 €
Condensador SMD	0.05 €	2	0.1 €
Pin headers	0.0355 €	13	0.47 €
Baterías (pilas AAA)	1.125 €	2	2.25 €
PCB	1.25 €	1	1.25 €
Diseño 3D	4.78 €	1	4.78 €
Estaño	0.90 €	1	0.90 €
Total			17.37 €

Tabla en Apéndice A.1. Coste de componentes y material fungible

Equipo/software	Coste unitario (€)	Periodo de amortización	Periodo de uso	Coste total (€)
Ordenador Portatil	79.83 €	4 meses	4 meses	79.83 €
Eagle Hobbyist	23.33 €	4 meses	4 meses	23.33 €
Arduino IDE	0 €	4 meses	4 meses	0 €
Freecad	0 €	4 meses	4 meses	0 €
			Total	103.17 €

Tabla A.2

Tabla en Apéndice A.2. Amortización de equipos y licencias de software

Actividad	Horas
Coordinación del TFG	15
Análisis de requisitos	20
Desarrollo hardware/software del prototipo	150
Realización de las pruebas	20
Redacción de la memoria	49
Preparación de la presentación	25
Horas totales	279
Coste por hora	7 €
Coste total de la mano de obra	1953 €

Tabla en Apéndice A.3. Horas invertidas en la realización del proyecto (mano de obra)

Concepto	Coste (€)
Coste de componentes y material fungible	17.37 €
Amortización de equipos y licencias de software	103.17 €
Mano de obra	1953 €
Coste total	2073.54 €

Tabla en Apéndice A.3. Presupuesto total

Referencias

- [1] R. Birwhistell, Introduction to kinesics, Lousville: University of Louisville, 1952.
- [2] Microsoft, «microsoft,» Microsoft, 4 11 2010. [En línea]. Available: <http://www.microsoft.com/en-us/news/press/2010/nov10/11-04kinectlaunchpr.aspx>. [Último acceso: 21 11 2013].
- [3] Microsoft, «xbox,» Microsoft, [En línea]. Available: <http://www.xbox.com/es-ES/Kinect>. [Último acceso: 19 11 2013].
- [4] Guinness World Records, «guinnessworldrecords,» Guinness World Records, 28 11 2000. [En línea]. Available: <http://www.guinnessworldrecords.com/records-9000/fastest-selling-gaming-peripheral/>. [Último acceso: 21 11 2013].
- [5] Nintendo, «nintendo,» Nintendo, [En línea]. Available: <http://www.nintendo.com/wii/what-is-wii/#/controls>. [Último acceso: 19 11 2013].
- [6] Play Station, «playstation,» Play Station, 17 09 2010. [En línea]. Available: <http://es.playstation.com/psmove/>. [Último acceso: 2013 11 21].
- [7] El mundo, «elmundo,» El mundo, 30 11 2006. [En línea]. Available: <http://www.elmundo.es/navegante/2006/11/29/juegos/1164803253.html>. [Último acceso: 21 11 2013].
- [8] Periodico 20 minutos, «20minutos,» Periodico 20 minutos, 10 09 2007. [En línea]. Available:

- <http://www.20minutos.es/noticia/273750/0/robot/operacion/haya/>. [Último acceso: 11 21 2013].
- [9] J. M. y. J. P. F. Brunetti, «Comité Español de Automática,» 7 Septiembre 2007. [En línea]. Available: <http://www.ceautomatica.es/old/actividades/jornadas/XXVIII/documentos/2181-brunetti.pdf>. [Último acceso: 09 Junio 2015].
- [10] Energizer, «Energizer,» [En línea]. Available: <http://data.energizer.com/PDFs/E92.pdf>. [Último acceso: 08 06 2015].
- [11] Sparkfun, «Sparkfun,» 2009. [En línea]. Available: <https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf>. [Último acceso: 08 06 2015].
- [12] Bolutek, «Banggood,» [En línea]. Available: <http://img.banggood.com/file/products/20150104013145BLE-CC41-A%20Spfication.pdf>. [Último acceso: 08 06 2015].
- [13] T. Instrument, «Texas Instrument,» 2012. [En línea]. Available: <http://www.ti.com/lit/ds/symlink/cc2541.pdf>.
- [14] Atmel, «Atmel,» 2014. [En línea]. Available: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf.
- [15] Arduino, «Arduino,» [En línea]. Available: <http://www.arduino.cc/en/Main/ArduinoBoardUno>.
- [16] Eagle, «CadSoftUsa,» [En línea]. Available: <http://www.cadsoftusa.com/>. [Último acceso: 08 06 2015].
- [17] Jmoon, «Jmoon,» [En línea]. Available: <http://wiki.jmoon.co/programmers/usbasp/>. [Último acceso: 08 06 2015].

