Universidad Carlos III de Madrid

TRABAJO FIN DE GRADO

Interfaz de control de ratón para personas con tetraplejia.

Autor:
José Adán Imedio Moreno

Tutor:
Ricardo Vergaz Benito

Grado en Ingeniería de Sistemas Audiovisuales

Escuela Politécnica Superior

27 de septiembre de 2015

UNIVERSIDAD CARLOS III DE MADRID

Resumen

Escuela Politécnica Superior

Grado en Ingeniería de Sistemas Audiovisuales

Interfaz de control de ratón para personas con tetraplejia.

por José Adán Imedio Moreno

En este TFG se propone un sistema que pretende facilitar el acceso a la informática a personas con movilidad reducida. Concretamente, permite manejar el ratón del ordenador mediante movimientos de la cabeza y soplidos. Para ello, es necesario conectar el sistema por USB al equipo y ejecutar una aplicación de escritorio.

Los componentes hardware utilizados han sido: una placa controladora basada en Arduino, un acelerómetro y un sensor de sonido. Para el desarrollo software, se han empleado los lenguajes Arduino y Java.

El resultado es un sistema funcional, que cumple con los objetivos marcados a nivel teórico. Sin embargo, aún no ha sido testeado por personas con problemas reales de movilidad, lo que hubiese permitido refinarlo y adaptarlo mejor a su finalidad.

UNIVERSITY CARLOS III OF MADRID

Abstract

School of Engineering

Bachelor's Degree in Audiovisual System Engineering

Mouse control interface for people with quadriplegy

by José Adán Imedio Moreno

The system proposed in this Bachelor Thesis is aimed at making the access to computers easier for people with severe mobility problems. It allows controlling the mouse by head movements and blows. For that, we need to connect the system by USB and to execute a desktop application.

The hardware components used for this project were: an Arduino-based controller board, an accelerometer and a sound sensor. For the software development, the programming languages Arduino and Java.

The result is a functional system, which satistfies all of the pre-defined objectives. Nevertheless, it has not been tested by people with actual mobility problems yet. This would have allowed us to improve it and better addapt it to its purpose.

Agradecimientos

Me gustaría agradecer a mi tutor, Ricardo Vergaz, su esfuerzo y su ayuda. He obtenido respuestas útiles y rápidas en todo momento por su parte, haciéndome sentir realmente amparado.

También quiero dar las gracias a mi familia. Por apoyarme en mis decisiones, y por animarme a dedicar tiempo a este TFG en una situación complicada, como es la de encontrarme trabajando a jornada completa.

A mis compañeros de carrera y de residencia, por el interés, el apoyo, e incluso la ayuda técnica que me han brindado en ciertos momentos. En especial, quiero dar las gracias a Antonio Relaño, Xavier Verguín, Enrique Pérez y Miguel Castán.

A la empresa bq, por cederme material y proporcionarme la base de conocimiento necesaria para desarrollar este proyecto.

Y, en general, a todas las personas que, con su interés y apoyo, me han animado a seguir adelante incluso en los momentos de dificultad.

Muchas gracias a todos.

Índice general

Resumen II

Abstract III

Agradecimientos IV

Índice general V

Índice de figuras VIII

Índice de cuadros X

1. Introducción 1
- 1.1. Introducción . . . 1
- 1.2. Sistemas existentes . . . 1
  - 1.2.1. Ratón de bola - Trackball . . . 2
  - 1.2.2. Joystick . . . 2
  - 1.2.3. Licornio . . . 3
  - 1.2.4. Control por la mirada - Eye-tracking . . . 4
  - 1.2.5. Control por movimiento de la cabeza . . . 6
  - 1.2.6. Control por reconocimiento de voz . . . 7
  - 1.2.7. Otros dispositivos: pulsadores externos . . . 7
  - 1.2.8. Adaptación del ratón por software . . . 8
- 1.3. Proyectos open-source . . . 9
  - 1.3.1. Ratón accesible mediante Gafas EOG . . . 9
  - 1.3.2. Coche teledirigido por movimientos de la cabeza . . . 10
- 1.4. Consideraciones para el diseño de sistemas . . . 11
  - 1.4.1. Diseño para todos . . . 11
- 1.5. Objetivos del sistema propuesto . . . 12
  - 1.5.1. Funcionamiento . . . 12

2. Diseño del sistema: Hardware 14
- 2.1. Elección del controlador . . . 14
  - 2.1.1. Raspberry Pi . . . 14
  - 2.1.2. BeagleBone . . . 15
  - 2.1.3. UDOO Neo . . . 16

2.1.4. MinnowBoard MAX . . . 17
2.1.5. Nanode . . . 18
2.1.6. Libelium Waspmote . . . 19
2.1.7. Launchpad MSP430 . . . 20
2.1.8. Pinguino PIC32 . . . 21
2.1.9. STM32 Discovery . . . 21
2.1.10. Teensy 3.1 . . . 22
2.1.11. Arduino . . . 23
2.2. Elección del sensor de movimiento . . . 25
2.2.1. Sensor de inclinación . . . 26
2.2.2. Giroscopio . . . 27
2.2.3. Sensor IMU . . . 28
2.2.4. Acelerómetro ADXL345 . . . 29
2.3. Elección del sensor auxiliar . . . 33
2.3.1. Botón . . . 34
2.3.2. Sensor de presión . . . 34
2.3.3. Sensor de flexión . . . 35
2.3.4. Sensor IR . . . 36
2.3.5. Sensor de sonido FC-04 . . . 37
2.4. Sistema físico . . . 39
2.4.1. Diseño . . . 39
2.4.2. Fabricación . . . 39

3. Diseño del sistema: Software **43**
3.1. Elección del entorno de programación . . . 43
3.1.1. Placa controladora . . . 43
3.1.2. Aplicación de escritorio . . . 44
3.2. Elección del lenguaje de programación . . . 45
3.2.1. Processing . . . 45
3.2.2. Delphi / Object Pascal . . . 46
3.2.3. C Sharp . . . 46
3.2.4. Python . . . 46
3.2.5. Visual Basic . . . 47
3.2.6. Real Basic . . . 47
3.2.7. Just Basic . . . 48
3.2.8. JavaScript . . . 48
3.2.9. Java . . . 49
3.3. Diseño del algoritmo . . . 49
3.3.1. Arduino . . . 50
3.3.2. Java . . . 54

4. Sistema propuesto **58**
4.1. El sistema en funcionamiento . . . 58
4.2. Presupuesto . . . 60
4.2.1. Total . . . 61
4.3. Pruebas en el Colegio San Rafael . . . 62
4.3.1. Colegio San Rafael . . . 62

4.3.2. Pruebas en el Colegio San Rafael 62
4.4. Conclusiones 62
4.5. Líneas futuras 63

A. Código Arduino **65**

B. Código Java **67**

C. Diagrama de conexión **70**

Bibliografía **71**

Índice de figuras

1.1. Ratón de bola o trackball[1] 2
1.2. Joystick de boca[2] 2
1.3. Joystick de barbilla[1] 3
1.4. Licornio[3] 3
1.5. Control del ratón mediante el teclado numérico[1] 4
1.6. IRISCOM - Sistema de eye-tracking[1] 5
1.7. Tablero ETRAN[5] 5
1.8. Enable Viacam - Control por movimiento de la cabeza[6] 6
1.9. Ratón accesible mediante Gafas EOG[8] 9
1.10. Coche teledirigido por movimientos de la cabeza[9] 11
1.11. Equivalencia de movimientos cabeza-ratón 13

2.1. Raspberry Pi[12] 15
2.2. BeagleBone Black[13] 15
2.3. UDOO Neo[14] 16
2.4. MinnowBoard MAX[15] 17
2.5. Nanode[16] 18
2.6. Libelium Waspmote[17] 19
2.7. Launchpad MSP430[18] 20
2.8. Pinguino PIC32[19] 21
2.9. STM32 Discovery[20] 22
2.10. Teensy 3.1[21] 23
2.11. bq ZUM BT-328[23] 25
2.12. Ejes de giro[24] 26
2.13. Sensor de inclinación C-01[25] 26
2.14. Giroscopio Grove 3-axis 27
2.15. Sensor IMU GY-951[26] 28
2.16. Acelerómetro de 3 ejes ADXL345[27] 29
2.17. Modelo de un acelerómetro en reposo[28] 30
2.18. Modelo de un acelerómetro inclinado[28] 30
2.19. Vista ampliada de un acelerómetro real[29] 31
2.20. Modelo de los condensadores[30] 31
2.21. Botón o pulsador[32] 34
2.22. Sensor de presión[33] 35
2.23. Sensor de flexión[34] 36
2.24. Sensor de infrarrojos[35] 37
2.25. Sensor de sonido FC-04[36] 38
2.26. Base de la caja principal 40

2.27. Tapa de la caja principal 40
2.28. Caja del acelerómetro 41
2.29. Caja principal 41
2.30. Caja del acelerómetro 42

3.1. CodeBender 43
3.2. Bitbloq 44
3.3. Gráficos a tiempo real con Processing y Arduino[38] 45
3.4. Logo del lenguaje Python[39] 47
3.5. Logo del lenguaje Java 49
3.6. Flujograma del bucle principal 52
3.7. Captación ante un evento sonoro prolongado 53
3.8. Captación ante dos eventos sonoros 53
3.9. Flujograma del sensor de sonido 55
3.10. Flujograma del programa Java 57

4.1. Inicialización del sistema 58
4.2. Ventana de ejecución 59
4.3. Efecto de un soplido simple 59
4.4. Efecto de un soplido doble 60
4.5. Efecto de un soplido trile 60

C.1. Diagrama de conexión 70

Índice de cuadros

1.1. Tipos de clic según repeticiones 13

4.1. Presupuesto - Electrónica 61
4.2. Presupuesto - Otros 61
4.3. Presupuesto - Total 61

Capítulo 1

Introducción

1.1. Introducción

La **tetraplejia** se define como la ausencia de movimiento y sensibilidad en las cuatro extremidades del cuerpo.

Este padecimiento es causado por algún daño en la médula espinal (daño neurológico). Se debe normalmente a un trauma severo (accidentes, heridas por arma de fuego, etc.), enfermedades degenerativas (esclerosis múltiple, esclerosis lateral amiotrófica, etc.) o problemas de nacimiento (parálisis cerebral).

Es innegable la dificultad añadida que sufren estas personas en el desempeño de sus actividades cotidianas.

La informática ha demostrado ser una herramienta muy buena para facilitarles el acceso al conocimiento y al ocio. Por eso existen diversos recursos que posibilitan su uso, que se analizan con mayor detalle en los apartados siguientes.

Este TFG pretende proponer un sistema que complemente el acceso a la informática para estas personas, utilizando tecnologías que no requieren pago de licencias y son, por tanto, muy accesibles.

1.2. Sistemas existentes

En este apartado se van a exponer los diferentes recursos que se utilizan habitualmente para facilitar el uso del ratón a personas tetrapléjicas, tal y como se define en la web del INTEF (Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado)[1].

1.2.1. Ratón de bola - Trackball

FIGURA 1.1: Ratón de bola o trackball[1]

El ratón de bola se coloca bajo la barbilla, para accionarlo con los movimientos de ésta (figura 1.1). Los botones deben ser accesibles para facilitar la pulsación y evitar grandes desplazamientos.

En algunas ocasiones, también se puede colocar en un lateral de la cara, aunque esta opción no suele aconsejarse, pues conlleva un mayor cansancio para el usuario.

1.2.2. Joystick

FIGURA 1.2: Joystick de boca[2]

Este sistema presenta dos ventajas principales con respecto al ratón de bola:

- Los objetos lejanos son más fáciles de alcanzar.

FIGURA 1.3: Joystick de barbilla[1]

- El puntero no se mueve cuando el usuario recupera la posición de reposo (varilla al centro).

Por estos motivos, se aconseja a personas que presentan problemas para controlar el ratón de bola.

Existen dos modelos, como se puede observar en las figuras 1.2 y 1.3.

1.2.3. Licornio

FIGURA 1.4: Licornio[3]

El licornio (figura 1.4) es un instrumento ampliamente utilizado por personas tetrapléjicas. Permite, mediante el movimiento de la cabeza, pulsar el teclado, pintar y, en general, realizar acciones que requieran una cierta precisión.

Podemos configurar nuestro equipo para mandar las órdenes de movimiento al ratón desde el teclado numérico. Esta opción nos permite incluso especificar la velocidad y aceleración del movimiento, para una mayor precisión.

Los controles quedarían como se indica en la figura 1.5:

FIGURA 1.5: Control del ratón mediante el teclado numérico[1]

Pantallas capacitivas

Cabe mencionar que ya existen muchos modelos de licornio que incluyen un capuchón en la punta para poder utilizarlo sobre pantallas capacitivas. De este modo, podemos usar un smartphone o tablet de un modo mucho más cómodo.

También podemos encontrar múltiples tutoriales en Internet para fabricarnos nuestro propio licornio casero[4].

1.2.4. Control por la mirada - Eye-tracking

Existen dispositivos específicos que permiten manejar el ordenador mediante el uso de la mirada. Para esto, incluyen una cámara de alta precisión que capta el movimiento ocular del usuario y realiza diferentes acciones según sea este movimiento (figura 1.6).

FIGURA 1.6: IRISCOM - Sistema de eye-tracking[1]

Para el uso de este tipo de dispositivo, es muy importante evaluar el nivel de control de la mirada que tiene el usuario. Para ello, se utiliza un tablero de señalización ETRAN, como el que se muestra en la figura 1.7.

FIGURA 1.7: Tablero ETRAN[5]

Este tipo de dispositivos precisan de una calibración muy específica para cada usuario, además de un tiempo de entrenamiento, que variará en función de las capacidades de la persona.

Permite al usuario colocar el puntero del ratón en cualquier lugar de la pantalla simplemente mirando a ese punto.

El clic puede realizarse de dos modos: parpadeando lentamente o bien manteniendo el puntero en el lugar deseado durante un tiempo predeterminado.

Para conseguir esto, se colocan dos emisores infrarrojos, uno a cada lado de la pantalla. Una cámara, situada frente al usuario, lee la desviación de las luces infrarrojas reflejadas en su pupila.

Iriscom

Este sistema ha sido desarrollado por la empresa Iriscom Sistemas, en colaboración con la Asociación de Esclerosis Lateral Amiotrófica (ADELA). El coste del sistema es de 6000€, aunque existen programas de financiación y subvenciones.

1.2.5. Control por movimiento de la cabeza

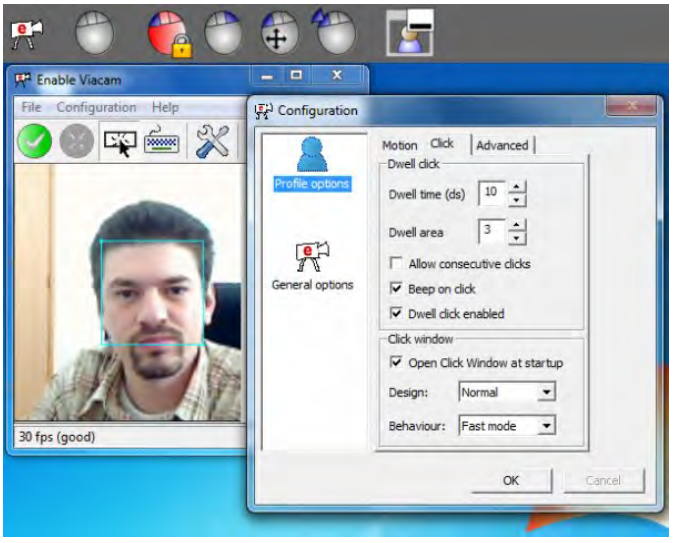

FIGURA 1.8: Enable Viacam - Control por movimiento de la cabeza[6]

Este tipo de sistemas nos permite utilizar el ratón mediante movimientos de la cabeza. El software detecta los rostros automáticamente a través de la webcam, y lee los movimientos (figura 1.8).

Podemos realizar el clic de dos formas: deteniendo el puntero en el lugar deseado durante un tiempo determinado o emitiendo un sonido a través de un micrófono.

Para poder seleccionar en cada momento el efecto que buscamos con nuestro clic, tenemos un menú siempre visible (mientras la aplicación esté activa). Aquí podemos elegir entre clic izquierdo, derecho, doble clic o arrastrar y soltar. Existe también la opción "no clic", que inhabilita el puntero. Esto es útil para momentos en que no vayamos a necesitarlo durante un tiempo prolongado (por ejemplo, durante la lectura de un pdf).

Enable Viacam

Este sistema ha sido desarrollado por la empresa CREA Sistemas Informáticos, en colaboración con el APPC (Centro de Parálisis Cerebral) de Tarragona. Es la versión de código abierto de la aplicación Ratón Facial, ya descatalogada. Podemos descargarlo gratis desde su web[7] y sólo necesitamos un ordenador y una webcam para poder usarlo.

1.2.6. Control por reconocimiento de voz

Este tipo de sistemas proporciona uno de los accesos más rápidos a la escritura de textos, por lo que son ampliamente utilizados, también por personas con lesiones diferentes a la tetraplejia.

El principal problema que presentan es que, en muchos casos, las deficiencias motoras se deben a lesiones medulares o enfermedades degenerativas. Esto suele implicar dificultades al pronunciar determinados fonemas, e incluso la pérdida progresiva del habla.

Una ventaja importante de los sistemas de reconocimiento de voz es que se pueden entrenar. Tanto para mejorar el reconocimiento de palabras que no se pronuncian correctamente como para obviar los sonidos comodín que realice el usuario.

Los principales problemas que presentan las personas discapacitadas al usar estos sistemas son:

- Incapacidad para pronunciar ciertos fonemas (como *pla* y *tra*).
- Emisión involuntaria de sonidos comodín (por ejemplo, *eh*).
- Fatiga al hablar.
- Falta de homogeneidad en el volumen, tono, ritmo o entonación.

1.2.7. Otros dispositivos: pulsadores externos

Existen los siguientes tipos:

- **Interruptor collarín:** Se activa por el movimiento o presión de la cabeza sobre el conmutador.
- **Interruptor de haz luminoso:** El movimiento de las pestañas interfiere un haz de luz.

- **Pulsador de barbilla:** Se coloca a modo de colgante. El interruptor se activa al presionar con la barbilla sobre él.
- **Sensor de humedad:** Se activa al entrar en contacto con la lengua.
- **Sensor de presión aérea:** Detecta soplidos, succiones o aspiraciones.
- **Sensor de sonido.**

1.2.8. Adaptación del ratón por software

En los propios sistemas operativos tenemos normalmente ajustes que nos permiten modificar ciertos parámetros sobre el ratón. Estos conceptos son útiles con independencia del hardware utilizado.

Regulación de velocidad

Dependerá en gran medida de las capacidades de cada persona:

- **Alta:** cuando el usuario tenga buena precisión y le suponga un gran esfuerzo realizar desplazamientos amplios del puntero.
- **Baja:** para usuarios con dificultad para situar el puntero sobre un objeto determinado.

Rechazo de doble clic indeseado

Podemos configurar la velocidad del doble clic del ratón, para evitar que éste se produzca por una falta de control del usuario.

Bloqueo de clic

Permite arrastrar el ratón sin tener que mantener pulsado el botón.

Efecto Dwell

Permite realizar un clic sin pulsar el botón, solamente posicionando el puntero durante un tiempo definido en el lugar deseado.

Pantalla curva para el puntero

Consiste en que el movimiento del ratón no tenga límites en los bordes de la pantalla. Como si de una pantalla curva se tratara, el cursor desaparece por un lado y aparece en el lado contrario. Ocurre lo mismo en la vertical.

Puntero inteligente

De manera automática, el puntero se sitúa en lugares probables de la pantalla, como por ejemplo un botón de Aceptar. De esta forma evitamos que el usuario tenga que moverse por la pantalla para su ubicación en este botón.

1.3. Proyectos open-source

En esta sección se exponen algunos proyectos que se han considerado relevantes por presentar características comunes con el sistema propuesto en el presente TFG.

Por supuesto, existen muchos más proyectos interesantes, dignos de mención. Por mero afán de sintetizar la información, se ha incluido sólo una pequeña representación de los mismos.

1.3.1. Ratón accesible mediante Gafas EOG

Figura 1.9: Ratón accesible mediante Gafas EOG[8]

Estas gafas detectan los movimientos del ojo mediante electrooculografía (EOG), una técnica biomédica basada en la lectura de las señales eléctricas emitidas por los ojos al moverse. Al mirar al frente, el voltaje medido será 0V, mientras que al mirar hacia un lado se generará una señal de entre 0,4 y 1 mV (figura 1.9).

El sistema cuenta con las siguientes etapas:

- **Captación de la señal**, a través de tres electrodos, dos de ellos colocados junto a los ojos y otro en la nariz (para marcar la tierra del circuito).
- **Amplificación**, dada la baja potencia de la señal.
- **Filtrado paso bajo**, para eliminar el ruido producido por cables y dispositivos electrónicos.
- **Eliminación del offset**.
- **Microcontrolador AT-mega328P** para enviar la señal procesada al ordendor por puerto serie.
- **Script Python** para leer los datos recibidos.
- **Aplicación C++** para proporcionar la interfaz y la integración del sistema completo.

El sistema fue desarrollado por Luis Cruz, un estudiante de 18 años, procedente de Honduras. Su idea surgió en un intento de ayudar a un compañero de clase que sufría tetraplejia [8].

1.3.2. Coche teledirigido por movimientos de la cabeza

En este sistema, los movimientos de la cabeza permiten mover un coche teledirigido. Dos placas Arduino gestionan la captación de datos y la traducción de ésta en movimiento (figura 1.10).

El sistema cuenta con:

- **Acelerómetro de 2 ejes,** para captar los movimientos de la cabeza.
- **Arduino Uno (modo slave),** para leer los valores del acelerómetro.
- **Módulo Bluetooth,** para comunicar ambas placas.

FIGURA 1.10: Coche teledirigido por movimientos de la cabeza[9]

- **Arduino Uno (modo master) + Zumo Shield,** para controlar el programa completo y mover el coche.
- **Coche.**

1.4. Consideraciones para el diseño de sistemas

Cuando diseñamos nuevos sistemas que pretenden ser accesibles, debemos seguir una serie de pautas que permiten asegurar la idoneidad del sistema creado. Estas pautas están definidas mediante el **Diseño para todos** (`http://designforall.org/`[10]), expuesto en el siguiente apartado.

1.4.1. Diseño para todos

El diseño para todos es una filosofía de diseño que tiene como objetivo conseguir que los entornos, productos, servicios y sistemas puedan ser utilizados por el mayor número posible de personas. Es un modelo de diseño basado en la diversidad humana, la inclusión social y la igualdad.

Se basa en los siguientes principios:

- **Respetuoso:** Debe evitar situaciones de riesgo para los usuarios, por lo tanto, todos los elementos que forman parte de un entorno deben estar diseñados bajo la perspectiva de la seguridad.

- **Seguro:** Considera diferentes preferencias y habilidades individuales. El producto se adapta a las características de diferentes personas al disponer de dispositivos o mecanismos que permiten la adaptación dimensional y funcional al usuario.
- **Saludable:** No debe constituir ningún riesgo para la salud ni ocasionar inconvenientes a aquellas personas que padecen alguna enfermedad o alergia.
- **Funcional:** Debe diseñarse de manera que la función para la cual ha sido creado la puedan llevar a cabo todas las personas, sin ningún problema.
- **Comprensible:** Cualquier usuario debe entender cómo usarlo sin dificultad.
- **Sostenible:** Debe garantizar que no se malgastan recursos naturales en su fabricación o uso.
- **Asequible:** Que el afán de lucro no impida que algunas personas tengan la oportunidad de disfrutarlo.
- **Atractivo:** Debe procurarse un resultado emocional y socialmente aceptable.

1.5. Objetivos del sistema propuesto

En este Trabajo Fin de Grado se pretende facilitar el uso del ordenador a personas con movilidad reducida. Para ello, se propone un sistema que permita controlar los movimientos del ratón, así como los clics de éste, mediante movimientos de la cabeza y sonidos vocales (o soplidos).

1.5.1. Funcionamiento

El cuidador debe colocar una gorra y un colgante al usuario. Cuando éste mueva la cabeza hacia adelante, el ratón del ordenador se desplazará hacia abajo; cuando la mueva hacia atrás, el ratón se moverá hacia arriba. Siguiendo la misma lógica, el movimiento de la cabeza hacia derecha e izquierda se corresponderá con los mismos movimientos del ratón, como se muestra en la figura 1.11.

Por otra parte, los clics se realizan emitiendo una serie de repeticiones de un sonido cualquiera, de menos de 0,5s de duración. También es totalmente funcional la realización de soplidos para este fin.

Concretamente, una sola repetición del sonido producirá un clic sencillo con el botón izquierdo; dos repeticiones producirán un doble clic con el botón izquierdo; y tres o más, un clic con el botón derecho (cuadro 1.1).

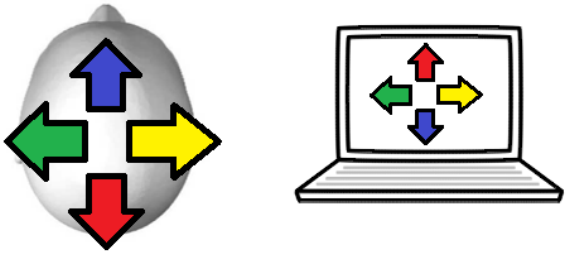

FIGURA 1.11: Equivalencia de movimientos cabeza-ratón

Repeticiones	**Efecto**
1	Clic izquierdo
2	Doble clic izquierdo
3 o más	Clic derecho

CUADRO 1.1: Tipos de clic según repeticiones

Capítulo 2

Diseño del sistema: Hardware

2.1. Elección del controlador

Para poder diseñar el sistema propuesto en el capítulo anterior, ha sido necesario seleccionar la placa controladora.

A continuación se presentan algunas opciones consideradas[11], con sus ventajas y desventajas, así como una exposición en mayor profundidad de la opción elegida: la placa bq ZUM BT-328, basada en Arduino.

Por supuesto, existen muchas más alternativas a las que se exponen aquí. Se han elegido estas muestras para la comparativa por ser las que mayor relevancia presentan con respecto al proyecto elegido. Son placas de un coste relativamente similar, y que se utilizan en aplicaciones parecidas a la que se presenta en este TFG.

2.1.1. Raspberry Pi

Ha sido desarrollada por la Raspberry Pi Foundation.

Cuenta con un procesador de 700MHz, un procesador gráfico y una memoria RAM de 512 MB. No tiene disco duro (utiliza una tarjeta SD en su lugar). Tiene dos puertos USB, un puerto Ethernet y un HDMI (figura 2.1).

Es muy apropiada para aplicaciones que requieran una interfaz gráfica.

El precio medio es de **30€**.

FIGURA 2.1: Raspberry Pi[12]

Ventajas

- Muy rápida y potente en términos de procesador y RAM.
- Gran cantidad de puertos.
- Permite instalar un sistema operativo completo.

Desventajas

- El manejo de sensores externos es mucho más complejo que con Arduino.

2.1.2. BeagleBone

FIGURA 2.2: BeagleBone Black[13]

Cuenta con un procesador de 700MHz y una memoria RAM de 256 MB. Lee tarjetas microSD y tiene puertos USB, microUSB y Ethernet (figura 2.2).

Es muy recomendable para aplicaciones que requieran a la vez lectura de sensores externos e interacción con otros dispositivos vía LAN o Internet.

El precio medio es de **50€**.

Ventajas

- Muy rápida y potente en términos de procesador y RAM.
- Gran cantidad de puertos.
- Permite instalar un sistema operativo completo.

Desventajas

- Precio elevado.
- El manejo de sensores externos es ligeramente más complejo que con Arduino.
- Más difícil de conseguir en España que una placa Arduino.

2.1.3. UDOO Neo

Figura 2.3: UDOO Neo[14]

Esta placa incluye un procesador de 1GHz y una memoria RAM de 512 MB (o incluso 1 GB, en función del modelo). Cuenta con puertos USB, microUSB, HDMI y UART. También incluye el layout de pines de Arduino UNO, con 6 pines de entrada analógica y 36 GPIO, y un sensor de 9 ejes, compuesto por acelerómetro, magnetómetro y giroscopio (figura 2.3).

Es apropiada tanto para desarrollo de aplicaciones y proyectos en entornos Linux como en Android. También para manejo de sensores externos.

El precio medio es de **85€**.

Ventajas

- Muy rápida y potente en términos de procesador y RAM.
- Gran cantidad de puertos.
- Permite instalar un sistema operativo completo.
- Es 100 % compatible con el código Arduino

Desventajas

- Precio elevado.
- Es un proyecto muy nuevo, por lo que aún es más complicada de conseguir que otras placas.

2.1.4. MinnowBoard MAX

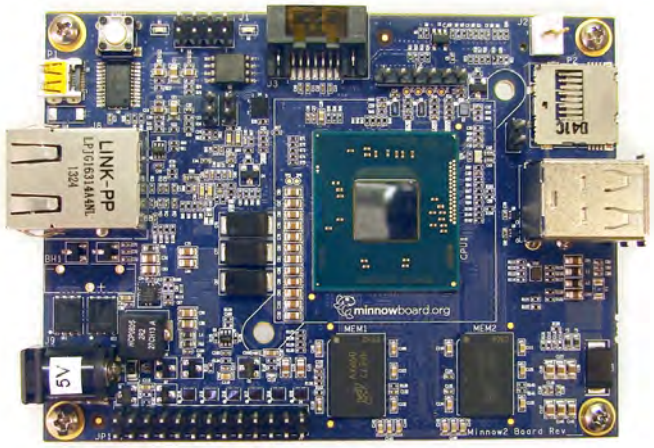

FIGURA 2.4: MinnowBoard MAX[15]

Este dispositivo incluye un procesador Intel de 1,4 GHz y una memoria RAM de 1 GB. Cuenta con puertos USB, HDMI, UART y Ethernet (figura 2.4).

Es una plataforma adecuada para aplicaciones avanzadas, como brazos robóticos.

El precio medio es de **100€**.

Ventajas

- Muy rápida y potente en términos de procesador y RAM.
- Gran cantidad de puertos.
- Permite instalar un sistema operativo completo.

Desventajas

- Precio elevado.
- El manejo de sensores externos es mucho más complejo que con Arduino.

2.1.5. Nanode

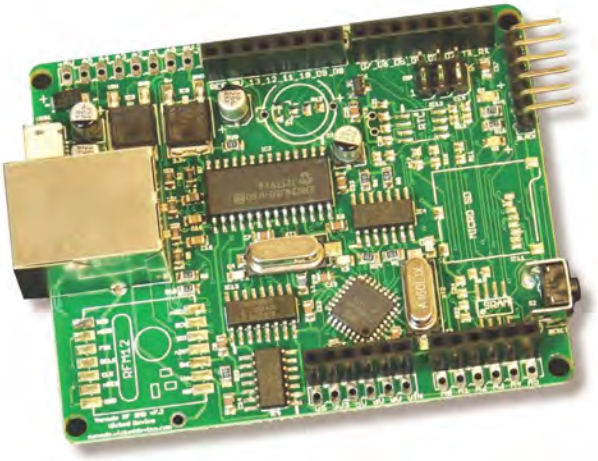

FIGURA 2.5: Nanode[16]

Esta placa cuenta con un microcontrolador de 16MHz, 2 KB de SRAM y 32 KB de memoria flash ISP. Incluye puertos microUSB y Ethernet. Tiene 6 puertos para sensores analógicos y 14 I/O digitales (figura 2.5).

Esta placa es muy apropiada para aplicaciones que impliquen el uso de hardware externo e interacción a través de Internet, haciéndola ideal para proyectos IoT.

El precio medio es de **30€**.

Ventajas

- 100 % compatible con Arduino.
- Bajo coste.

Desventajas

- La placa no viene totalmente montada.

2.1.6. Libelium Waspmote

Figura 2.6: Libelium Waspmote[17]

Esta placa cuenta con un microcontrolador de 14MHz, una RAM de 8KB, EEPROM de 4KB y FLASH de 128KB, así como 2GB de almacenamiento externo a través de una tarjeta SD. Tiene 7 entradas analógicas y 8 puertos digitales I/O. También incluye módulos de comunicación inalámbrica integrados, como RTC o XBee (figura 2.6).

Está especialmente pensada para aplicaciones reales en Smart Cities.

El precio medio es de **155€**.

Ventajas

- 100 % compatible con Arduino.
- Módulos de comunicación inalámbrica incorporados.
- Acelerómetro incorporado.

Desventajas

- Precio elevado.

2.1.7. Launchpad MSP430

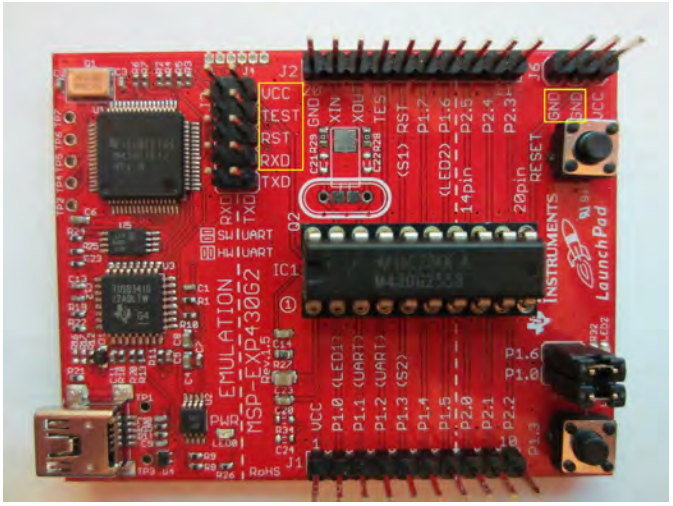

FIGURA 2.7: Launchpad MSP430[18]

Esta placa tiene un microcontrolador de 16MHz, RAM de 512B y memoria FLASH de 16KB. También incluye 8 puertos I/O digitales y 8 puertos I/O analógicos (figura 2.7).

Es especialmente apropiada para aplicaciones con sensores remotos, pues su consumo de energía es muy bajo, y tiene un modo *standby* que le permite reactivarse en tan solo 1μs.

El precio medio es de **10€**.

Ventajas

- 100 % compatible con Arduino.
- Muy bajo coste.
- Bajo consumo de energía

Desventajas

- Memorias RAM y FLASH más reducidas.
- Más difícil de conseguir en España que Arduino.

2.1.8. Pinguino PIC32

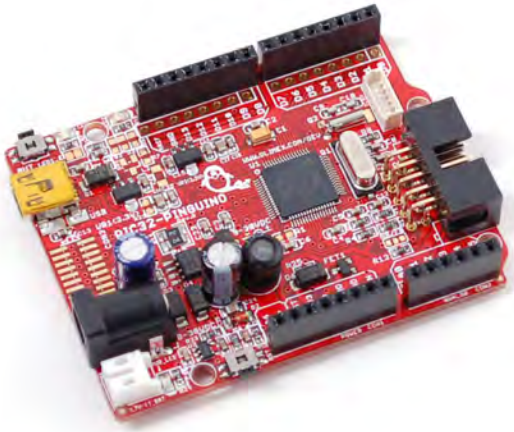

FIGURA 2.8: Pinguino PIC32[19]

Esta placa tiene un microcontrolador de 80MHz, RAM de 32KB, memoria FLASH de 256KB y ranura microSD. También incluye 8 entradas analógicas y 26 puertos I/O digitales (figura 2.8).

La funcionalidad es muy similar a la de una placa Arduino.

El precio medio es de **20€**.

Ventajas

- Mayor velocidad de reloj y memoria.
- Precio ligeramente más bajo.

Desventajas

- Comunidad aún en desarrollo.
- Pocas librerías disponibles.

2.1.9. STM32 Discovery

Esta placa cuenta con un microcontrolador de 72MHz, 20KB de memoria RAM y 128KB de memoria FLASH. Incluye 15 entradas analógicas y 43 GPIO (figura 2.9).

La funcionalidad es similar a la de una placa Arduino.

El precio medio es de **28€**.

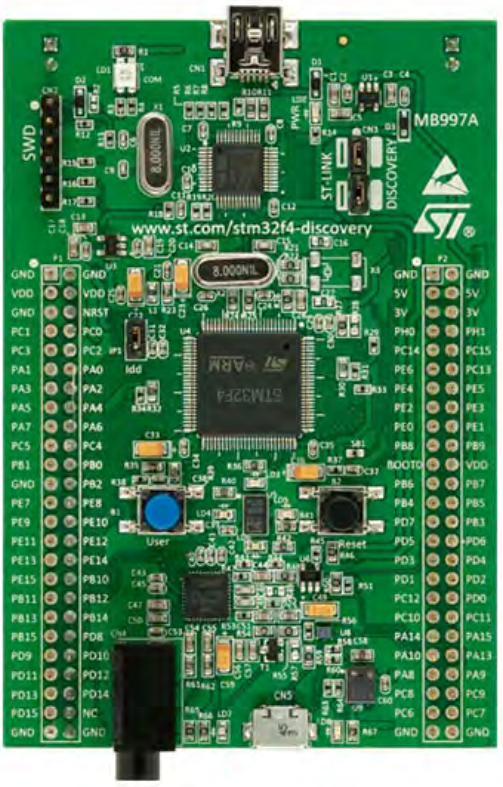

FIGURA 2.9: STM32 Discovery[20]

Ventajas

- Más velocidad de reloj y memoria.

Desventajas

- Comunidad poco desarrollada.
- Poca documentación.

2.1.10. Teensy 3.1

Esta placa cuenta con un microcontrolador de 72MHz, 256KB de memoria RAM y 2KB de memoria EEPROM. Incluye 14 entradas analógicas y 20 puertos I/O digitales (figura 2.10).

La funcionalidad es similar a la de una placa Arduino, sobre todo para proyectos que exijan una máxima optimización del espacio, debido a su reducido tamaño.

FIGURA 2.10: Teensy 3.1[21]

El precio medio es de **20€**.

Ventajas

- Más velocidad de reloj y memoria.

Desventajas

- Comunidad poco desarrollada.
- Poca documentación.

2.1.11. Arduino

¿Qué es Arduino?

Como podemos leer en la web oficial de Arduino[22]

Arduino es una plataforma de prototipado open-source basada en hardware y software fáciles de utilizar. Las placas Arduino son capaces de leer entradas (luz incidente sobre un sensor, la pulsación de un botón, o incluso un mensaje de Twitter) y convertirlo en una salida (el giro de un motor, el encendido de un LED, una publicación online...). Todo esto se define con una lista de instrucciones programadas a través del software Arduino (IDE).

Arduino ha servido para acercar la electrónica y la programación a muchas personas sin conocimientos previos. También es utilizado por mucha gente con conocimiento específico, gracias a su facilidad de uso.

Por otra parte, la ventaja principal de esta plataforma reside en la gran comunidad que se ha generado entorno a ella. Existen miles de personas en todo el mundo compartiendo conocimiento, experiencias y proyectos.

¿Por qué Arduino?

Como se ha expuesto en los apartados anteriores, existen múltiples alternativas en el mercado para poder realizar un sistema como el que se describe en este TFG. Arduino ha sido la opción elegida por los siguientes motivos:

- **Precio accesible:** Las placas Arduino son relativamente baratas (de 20 a 50 euros). Concretamente, la placa elegida tiene un precio de **34,90€.**
- **Multi-plataforma:** El software Arduino puede ser utilizado fácilmente en los SO principales: Windows, MacOSX y Linux.
- **Entorno de programación:** El IDE de Arduino es intuitivo, y muy estable.
- **Open-source:** Todos los recursos relacionados con esta plataforma son públicos y totalmente accesibles (librerías software, planos de los microcontroladores, etc.).
- **Comunidad muy desarrollada:** Existen muchos usuarios de Arduino en el mundo, por lo que los recursos disponibles son abundantes y de calidad.

bq ZUM BT-328

De la gran oferta de placas Arduino, para este TFG se ha elegido una que, entre otras características, cuenta con un módulo Bluetooth integrado. Esto resulta de gran utilidad para nuestro sistema, ya que no obliga al usuario a estar unido por un cable USB al ordenador (recordemos que la placa controladora se transporta en una caja, a modo de colgante). Esta funcionalidad no ha sido implementada, pero se considera como una posible mejora.

El hecho de que los planos de las placas estén publicados con licencia Creative Commons permite que otras empresas puedan rediseñar, mejorar y comercializar sus propias placas compatibles. Éste es el caso de la opción elegida para el presente TFG: la bq ZUM BT-328, fabricada por la empresa española bq.

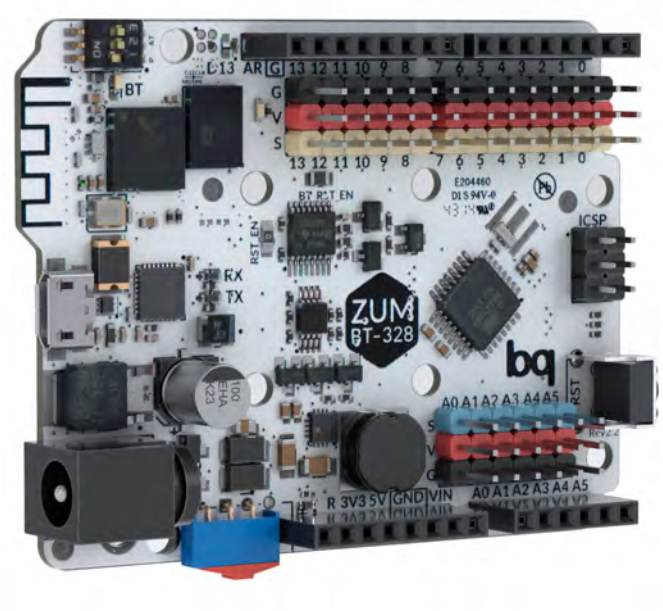

FIGURA 2.11: bq ZUM BT-328[23]

Las ventajas principales que presenta esta placa con respecto a la Arduino BT original son las siguientes:

- **Corriente de salida:** 3,2A frente a 1A.
- **Botón de encendido:** Muy útil durante las pruebas del sistema.
- **Sets de 3 pines:** Cada puerto cuenta con un set de pines S(señal), V(5V) y G(tierra). Esto nos permite usar ciertos componentes para las pruebas de un modo mucho más sencillo, sin utilizar una protoboard intermedia.

2.2. Elección del sensor de movimiento

Para poder transformar la inclinación de la cabeza del usuario en una determinada acción del sistema, es necesario utilizar un componente que permita captar ese movimiento.

Concretamente, debe ser un componente de sensibilidad relativamente alta, pues los usuarios potenciales presentan problemas importantes de movilidad. Y debe leer el movimiento en al menos dos ejes: Roll (cabeza hacia los lados) y Pitch (cabeza hacia abajo y hacia arriba), como se indica en la figura 2.12

Para realizar esta función, se barajaron diferentes opciones, que se exponen a continuación. En el último apartado, se analiza en mayor profundidad la opción elegida: el acelerómetro ADXL345.

FIGURA 2.12: Ejes de giro[24]

2.2.1. Sensor de inclinación

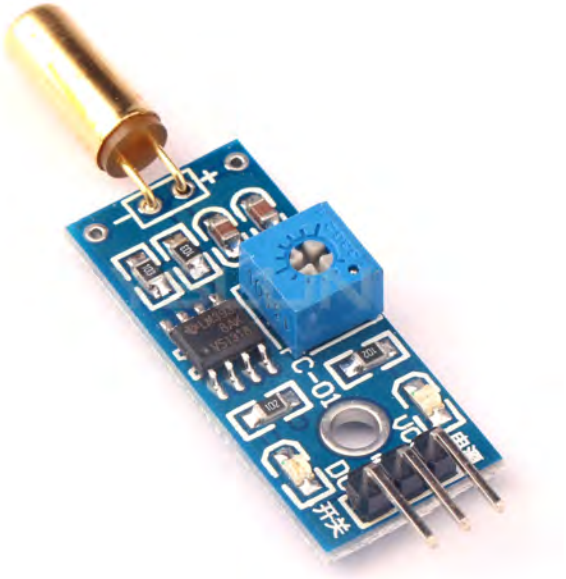

FIGURA 2.13: Sensor de inclinación C-01[25]

Estos sensores informan cuando su inclinación es mayor de un ángulo determinado. Su salida es digital, siendo su output 0 o 1 en función de si se ha superado o no el umbral. Existen modelos muy sencillos, que sólo constan de la cápsula medidora, y otros que incluyen un potenciómetro con el que variar el valor umbral (como el mostrado en la figura 2.13).

Suelen utilizarse para detectar la caída o vuelco de objetos, como coches robóticos.

Precio en Internet: **3,5€**

Ventajas

- Bajo precio
- Fácil manejo.

Desventajas

- Para no perder resolución, es imprescindible que la posición natural del sistema sea totalmente vertical.

2.2.2. Giroscopio

Figura 2.14: Giroscopio Grove 3-axis

Un giroscopio nos permite medir velocidades angulares en los ejes Yaw, Pitch y Roll. Existen modelos con diferentes rangos de medida, en función de la aplicación (figura 2.14).

Su aplicación mayoritaria es en navegación. Cada vez más, se utilizan en combinación con acelerómetros.

Precio en Internet: **47,13€**.

Ventajas

- Permite calcular la inclinación en los ejes de interés.

Desventajas

- Precio elevado.

2.2.3. Sensor IMU

FIGURA 2.15: Sensor IMU GY-951[26]

Una Unidad de Medida Inercial (en inglés: Inertial Measurement Unit - IMU) es la combinación de un acelerómetro y un giroscopio, pudiendo ofrecer hasta 6 grados de libertad (en inglés: Degrees Of Freedom - DOF). Existen modelos con hasta 9 DOF, por la inclusión de un magnetómetro (figura 2.15).

Se utilizan típicamente en dispositivos que requieren conocimiento de su posición exacta en el espacio, como brazos robóticos.

Precio en Internet: **63,33€**.

Ventajas

- Permite medir la inclinación en los ejes de interés.
- Permite obtener más información, lo cual es útil de cara a una posible mejora del sistema.

Desventajas

- Precio muy elevado.

2.2.4. Acelerómetro ADXL345

FIGURA 2.16: Acelerómetro de 3 ejes ADXL345[27]

Un acelerómetro permite medir la aceleración lineal de un cuerpo sobre un sistema de referencia cartesiano x, y, z (aceleración dinamica), así como la posición de dicho objeto con respecto a la superficie de la Tierra (aceleración estática).

Se suele usar para medir la inclinación de un objeto, gracias a que sus medidas están referenciadas al vector de la gravedad. Un ejemplo típico de este uso es el cambio de orientación en la pantalla de un smartphone o tablet ante un movimiento del dispositivo por parte del usuario (figura 2.16).

¿Por qué el acelerómetro ADXL345?

Para el presente TFG se ha elegido un acelerómetro ya que permite medir la inclinación de un elemento en los ejes de interés.

Se ha elegido concretamente el modelo ADXL345 por presentar todas las características necesarias con un precio muy asequible: **11,50€**.

Funcionamiento de un acelerómetro

Un acelerómetro es, en esencia, un dispositivo que mide diferencias entre las aceleraciones lineales que experimenta y el vector de la gravedad.

Se puede modelar el sensor como se muestra en la figura 2.17, una masa dentro de una caja que, en su estado natural de reposo, experimenta una aceleración positiva en el eje z de 9,8m/s^2, o de 1g. Para que las 3 componentes sean 0, el objeto debe encontrarse **en caída libre**.

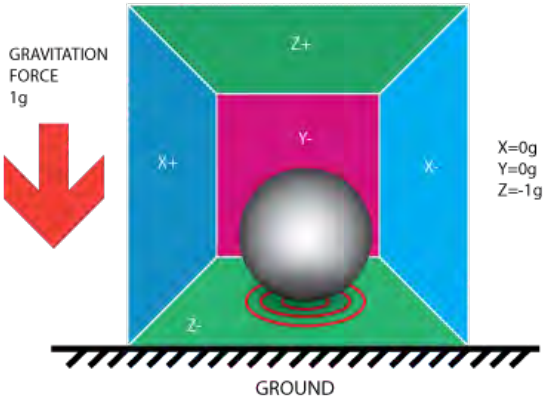

FIGURA 2.17: Modelo de un acelerómetro en reposo[28]

Cuando el sensor se inclina, sigue experimentando una aceleración de módulo 1g, aunque en esta ocasión será una combinación de componentes en los 3 ejes, como se puede ver en la figura 2.18.

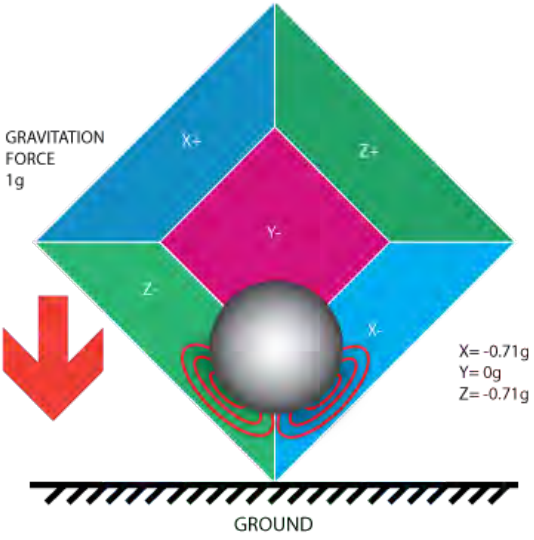

FIGURA 2.18: Modelo de un acelerómetro inclinado[28]

En la práctica, para medir estas variaciones se utilizan unas masas micrométricas (o incluso nanométricas) que quedan suspendidas y, a su vez, unidas a una estructura de varillas metálicas móviles, cargadas eléctricamente. Dichas varillas se encuentran entrelazadas, aunque ligeramente separadas de otro juego de varillas estáticas, también

cargadas, formando una hilera de condensadores (figura 2.19). Cuando el sensor es sometido a una aceleración, tanto la masa como las primeras varillas se mueven, generando diferencias de potencial en el interior de los condensadores (figura 2.20).

Por lo tanto, el output de un acelerómetro de tres ejes son 3 valores de voltaje que nos indican la aceleración en cada eje.

Sin embargo, los valores manipulados en el software no son directamente estos voltajes. El módulo ADC del controlador los mapea a valores entre 0 y 1023, por ser la resolución del acelerómetro 10 bits.

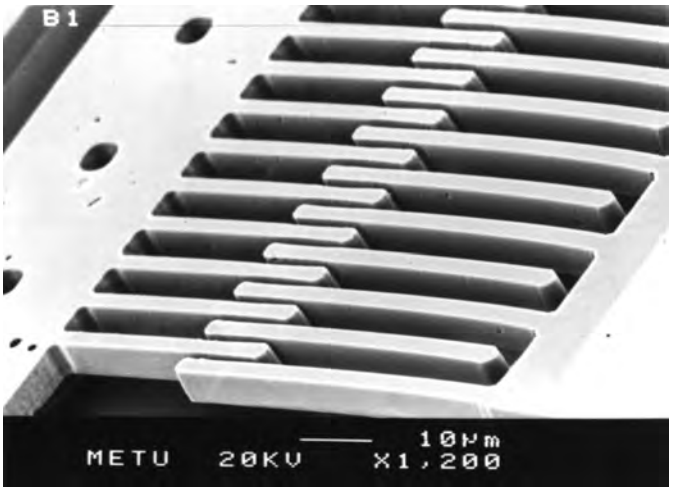

FIGURA 2.19: Vista ampliada de un acelerómetro real[29]

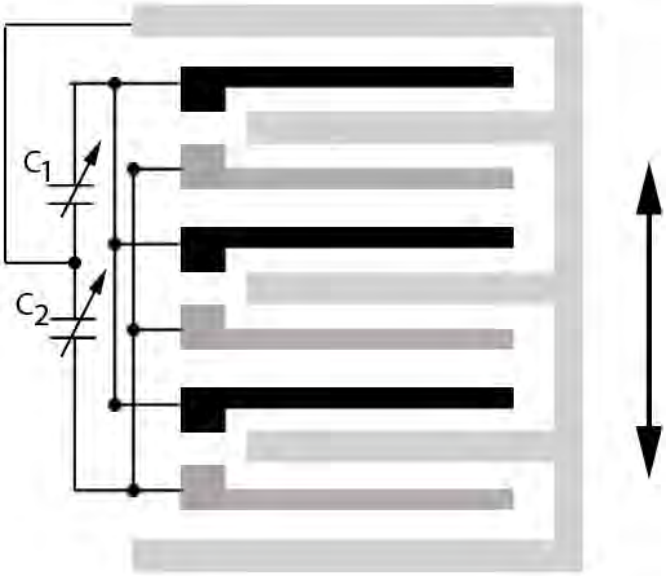

FIGURA 2.20: Modelo de los condensadores[30]

Medida de la inclinación

Si definimos las matrices de rotación entorno a los ejes x, y, z como los ángulos Roll (ϕ), Pitch (θ) y Yaw (ψ), respectivamente, obtenemos[31]

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \tag{2.1}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \tag{2.2}$$

$$R_z(\psi) = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

Podemos expresar el vector de aceleración estática experimentado por el sensor ($\boldsymbol{G}$) en función de dichos ángulos y de la aceleración lineal (a_r) como:

$$G_p = \begin{pmatrix} G_p x \\ G_p y \\ G_p z \end{pmatrix} = R(g - a_r) \tag{2.4}$$

Asumiento a_r nula y el orden xyz por convenio, la expresión queda:

$$R_{xyz} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = R_x(\phi) R_y(\theta) R_z(\psi) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.5}$$

$$= \begin{pmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\cos(\psi) & -\sin(\theta) \\ \cos(\psi)\sin(\theta)\sin(\phi) - \cos(\theta)\sin(\psi) & \cos(\phi)\cos(\psi) + \sin(\theta)\sin(\phi)\sin(\psi) & \cos(\theta)\sin(\phi) \\ \cos(\phi)\cos(\psi)\sin(\theta) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \cos(\psi)\sin(\phi) & \cos(\theta)\cos(\phi) \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.6}$$

$$= \begin{pmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\psi) \\ \cos(\theta)\cos(\phi) \end{pmatrix} \tag{2.7}$$

Así nos queda una expresión en función de los ángulos de mayor interés: Roll (ϕ) y Pitch (θ). Lógicamente, la operación será al contrario: buscaremos conocer estos ángulos a partir de los valores de aceleración medidos por el sensor. Para ello, normalizamos el vector de aceleración:

$$\frac{1}{\sqrt{G_p x^2 + G_p y^2 + G_p z^2}} \begin{pmatrix} G_p x \\ G_p y \\ G_p z \end{pmatrix} = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\psi) \\ \cos(\theta)\cos(\phi) \end{pmatrix} \tag{2.8}$$

Y, finalmente, obtenemos las ecuaciones deseadas:

$$\tan(\phi_{xyz}) = (\frac{G_p y}{G_p z}) \tag{2.9}$$

$$\tan(\theta_{xyz}) = (\frac{-G_p x}{G_p y \sin(\phi) + G_p z \cos(\phi)}) = \frac{-G_p x}{\sqrt{G_p y^2 + G_p z^2}} \tag{2.10}$$

2.3. Elección del sensor auxiliar

Para poder realizar los clics en el ordenador, era necesario seleccionar un segundo sensor.

Éste debía cumplir con los siguientes requisitos:

- Fácil de utilizar.
- Cómodo y poco intrusivo para el usuario.
- No interferir en la lectura del acelerómetro.
- Bajo coste.

A continuación, se mencionan las diferentes alternativas consideradas. Se han descartado otras muchas opciones por no cumplir los requisitos citados o por presentar una funcionalidad muy superior a la necesaria, ya que esto suele redundar en un precio excesivo. En último lugar, se analiza en mayor profundidad el sensor elegido, el sensor de sonido FC-04.

FIGURA 2.21: Botón o pulsador[32]

2.3.1. Botón

Un botón es un componente digital que devuelve un valor 0 o 1, dependiendo de si está o no pulsado (figura 2.21).

Precio en Internet: **0,38€**

Ventajas

- Muy bajo coste.
- Fácil de utilizar.

Desventajas

- Interfiere en la lectura del acelerómetro, si se pulsa con cualquier parte de la cabeza.
- Intrusivo, si se pulsa con los dientes o la lengua.

2.3.2. Sensor de presión

Este componente devuelve un valor 0 cuando no se está ejerciendo ninguna fuerza sobre él. En caso contrario, devuelve un valor analógico proporcional a dicha fuerza (figura 2.22).

FIGURA 2.22: Sensor de presión[33]

Precio en Internet: **42,25€**

Ventajas

- Fácil de manipular por su forma.

Desventajas

- Alto coste.
- Interfiere en la lectura del acelerómetro, si se pulsa con cualquier parte de la cabeza.
- Intrusivo, si se pulsa con los dientes o la lengua.
- Puede llegar a causar dolor al usuario.

2.3.3. Sensor de flexión

Un sensor de flexión devuelve un valor 0 en su posición natural. En caso contrario, devuelve un valor analógico proporcional a la flexión a la que se someta (figura 2.23).

Precio en Internet: **7,61€**

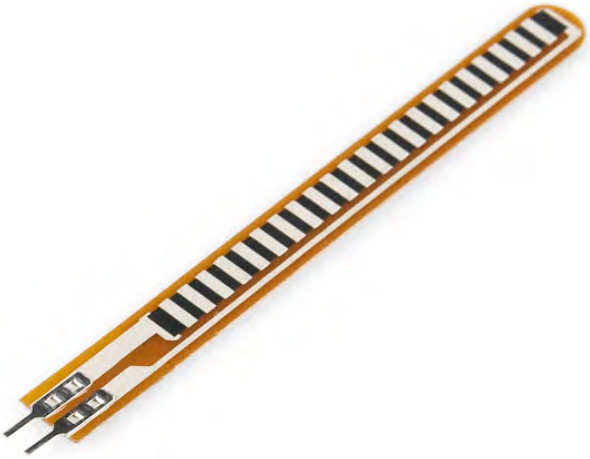

FIGURA 2.23: Sensor de flexión[34]

Ventajas

- Fácil de manipular por su forma.

Desventajas

- Interfiere en la lectura del acelerómetro, si se acciona con cualquier parte de la cabeza.
- Intrusivo, si se acciona con la boca.
- Muy inestable, si se acciona con la boca.

2.3.4. Sensor IR

Un sensor de infrarrojos permite detectar si hay un objeto cerca, mediante el envío y recepción de un haz infrarrojo. En el caso de que el objeto sea muy oscuro y mate, no lo detectará (figura 2.24).

De este modo, el clic se podría realizar acercando la lengua al sensor.

Precio en Internet: **15,16€**

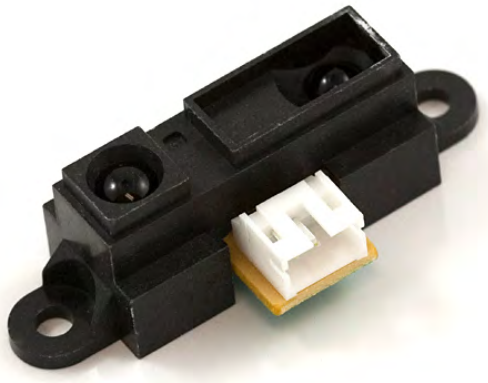

FIGURA 2.24: Sensor de infrarrojos[35]

Ventajas

- No interfiere en la lectura del acelerómetro.
- Poco intrusivo.

Desventajas

- Precio ligeramente elevado.
- Ligeramente intrusivo.

2.3.5. Sensor de sonido FC-04

Este sensor incorpora un micrófono de tipo electret, y devuelve un valor analógico en función del volumen de sonido captado. Incluye también un potenciómetro, que permite regular la sensibilidad de la respuesta (figura 2.25).

¿Por qué el sensor de sonido FC-04?

Ha sido la opción elegida por no interferir en la lectura del acelerómetro, ser muy poco intrusivo para el usuario y tener un coste reducido: **5,30€**.

FIGURA 2.25: Sensor de sonido FC-04[36]

Micrófono electret

De manera genérica, un micrófono se compone de las siguientes partes:

- **Diafragma:** es una membrana que vibra ante la incidencia de ondas de presión sonora.
- **Cápsula:** es un transductor mecánico-eléctrico, que permite transformar la onda de presión sonora en una señal eléctrica.
- **Rejilla:** es una cobertura colocada sobre el diafragma para protegerlo de agentes externos.
- **Carcasa:** es el recipiente donde se colocan el resto de componentes.

Los micrófonos se clasifican principalmente según dos parámetros: la directividad y la tecnología de transducción mecánico-eléctrica (obviamos otros conceptos como la respuesta en frecuencia, por ser poco relevantes en este contexto).

En cuanto a la **directividad**, este micrófono es unidireccional, presentando una respuesta mucho más alta ante ondas que inciden desde el frente. Esta característica es muy apropiada para el uso concreto de este proyecto. Un patrón polar más amplio aumentaría la probabilidad de falsos positivos en el clic, haciendo el sistema menos fiable.

Con respecto a la **tecnología de transducción**, se trata de un micrófono electrostático. Este tipo de dispositivos se caracterizan porque el núcleo de la cápsula y el diafragma forman un condensador. La capacidad del mismo variará ante las vibraciones del diafragma, produciendo una señal eléctrica de salida proporcional a la señal acústica de entrada. Para conseguir este efecto, es necesario que la cápsula esté cargada eléctricamente.

De entre los diferentes tipos de micrófonos electrostáticos, éste es un electret. Se caracteriza por utilizar un electrodo polarizado desde su fabricación. Por ello, no necesita alimentación externa.

2.4. Sistema físico

El sistema en bruto resulta algo difícil de colocar. Sobre todo, teniendo en cuenta que los usuarios potenciales son personas con problemas graves de movilidad.

Para resolver este problema, se han diseñado dos cajas y se han fabricado mediante impresión 3D FDM (deposición de plástico fundido).

2.4.1. Diseño

Para diseñar las cajas se ha utilizado el software open source FreeCAD [37].

Caja principal

En esta caja se alojan la placa controladora y el sensor de sonido (figura 2.26). Se ha considerado un agujero en el lateral de la caja para que sobresalga el micrófono. Para que el sensor quede totalmente fijo, la tapa de la caja incluye unos salientes verticales que lo colocan en la posición deseada (figura 2.27).

Caja del acelerómetro

En esta segunda caja se aloja el acelerómetro. Tiene un agujero circular para la entrada del cable y una tapa superior. Tiene una base ligeramente curva para adaptarse mejor a la visera de la gorra. Puede verse el modelo en la figura 2.28.

2.4.2. Fabricación

La fabricación de las cajas se ha realizado en PLA (Ácido Poliláctico) mediante impresión 3D FDM. Para ello, he utilizado una impresora bq Witbox, cedida por la propia empresa bq, debido a que actualmente me encuentro trabajando en la misma.

Puede verse el resultado final en las figuras 2.29 y 2.30.

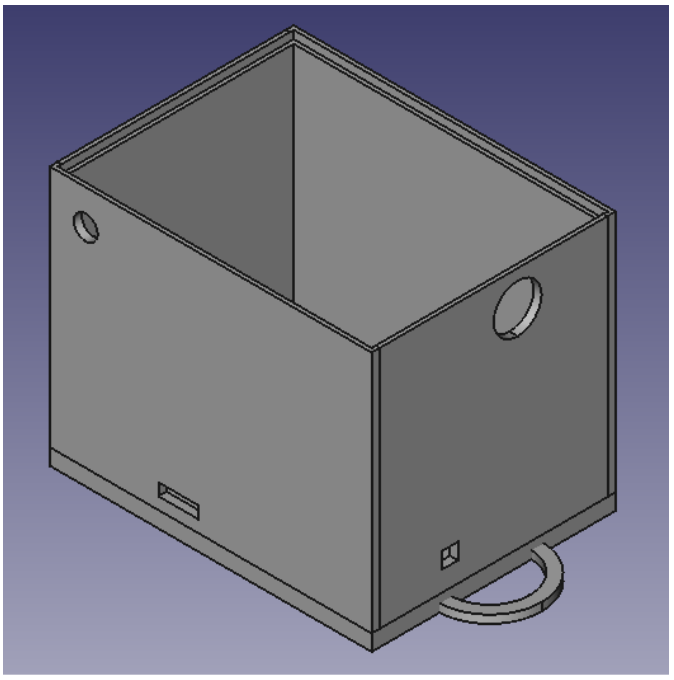

Figura 2.26: Base de la caja principal

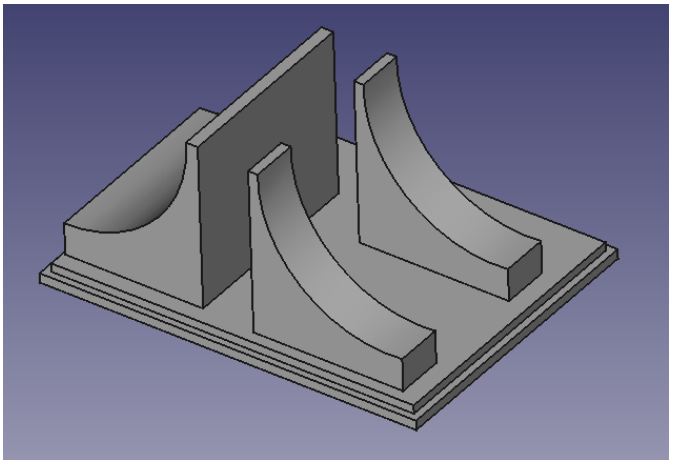

Figura 2.27: Tapa de la caja principal

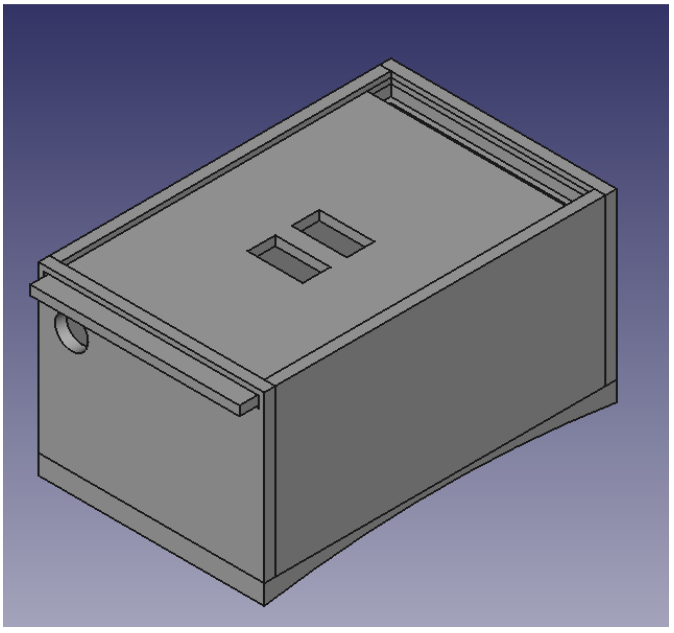

Figura 2.28: Caja del acelerómetro

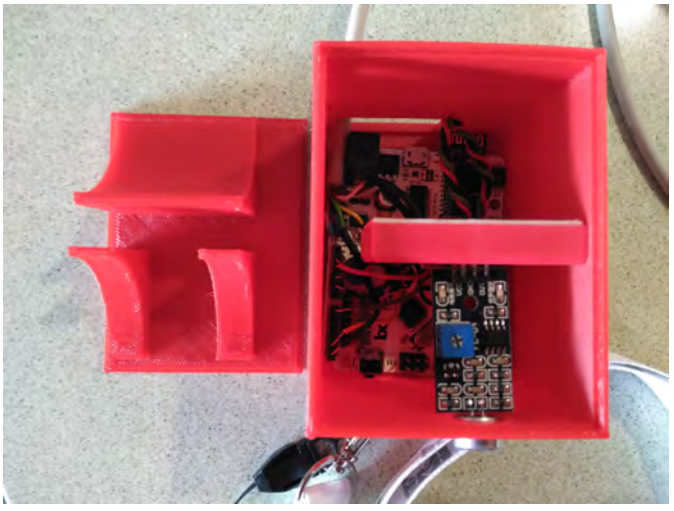

Figura 2.29: Caja principal

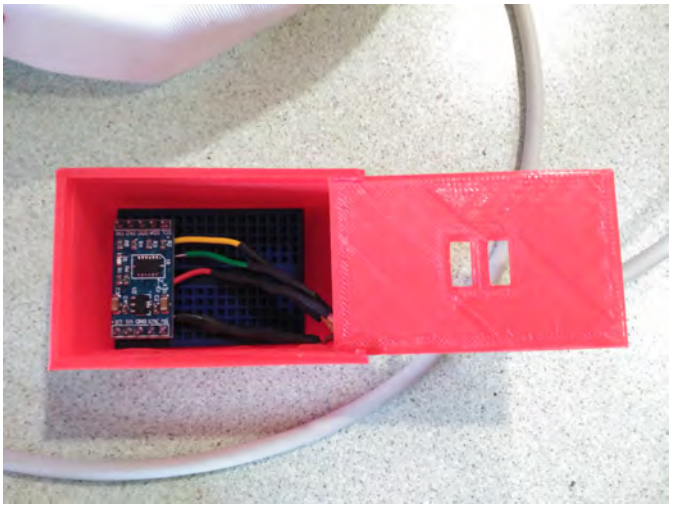

Figura 2.30: Caja del acelerómetro

Capítulo 3

Diseño del sistema: Software

3.1. Elección del entorno de programación

3.1.1. Placa controladora

La programación software de la placa controladora (bq ZUM BT-328 - Compatible con Arduino) se ha realizado con el **Arduino IDE**.

Cabe destacar que, a pesar de ser ésta la opción más ampliamente utilizada, existen alternativas. En esta ocasión, no se realiza una comparación exhaustiva de las mismas por no haberse considerado relevante para la documentación del proyecto.

A continuación, se citan dos ejemplos de entornos alternativos.

CodeBender

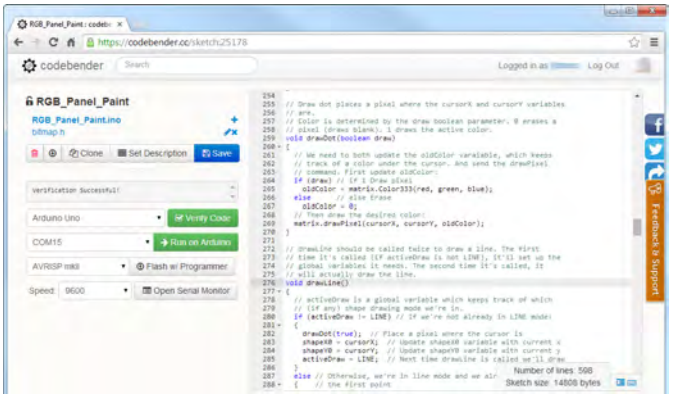

Figura 3.1: CodeBender

Es una aplicación online con todas las funcionalidades del IDE original (monitor serie, acceso a librerías, etc.) y algunas adicionales (figura 3.1).

Su principal ventaja es la existencia de una base de datos en la nube donde compartir nuestro código y acceder al de otras personas.

Bitbloq

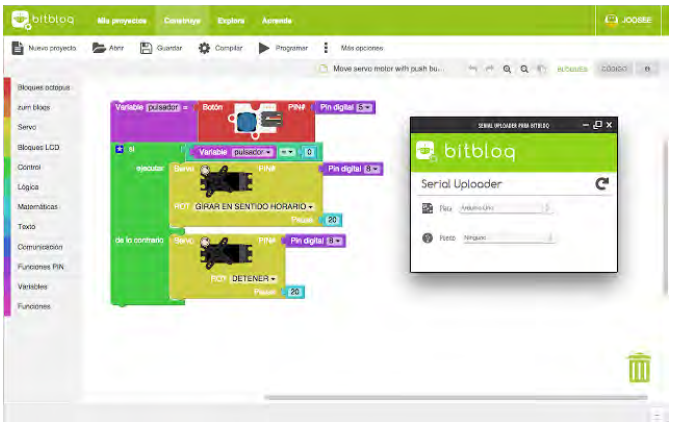

FIGURA 3.2: Bitbloq

Es una aplicación online enfocada al aprendizaje de la programación hardware. Está orientada a niños, aunque es útil para cualquier persona interesada sin conocimientos previos. Se fundamenta en el uso de bloques, en lugar de líneas de código (figura 3.2).

Existen otras aplicaciones similares, como **ArduBlock** o **Minibloq**.

3.1.2. Aplicación de escritorio

Por su parte, la programación de la aplicación de escritorio se ha realizado con **NetBeans**. Se ha elegido dicho entorno por estar enfocado principalmente al lenguaje de programación Java. El fundador y principal patrocinador del proyecto es la empresa Sun MicroSystems (actualmente administrado por Oracle Corporation), también responsable del desarrollo del lenguaje Java.

En este caso, tampoco se ha realizado una comparación exhaustiva de las diferentes opciones por no ser relevante para el desarrollo del proyecto.

La alternativa principal es **Eclipse**, desarrollado originalmente por IBM y mantenido actualmente por la Fundación Eclipse.

3.2. Elección del lenguaje de programación

Para poder realizar la lectura de la información captada por los sensores (a través del puerto serie) era necesario seleccionar un lenguaje de programación.

Debía cumplir los siguientes requisitos:

- Lectura del puerto serie de un modo sencillo.
- Posibilidad de creación de una GUI (Interfaz Gráfica de Usuario) intuitiva.
- Multi-plataforma.
- Librerías para poder manejar el ratón del ordenador.

A continuación, se exponen las diferentes alternativas consideradas, con sus ventajas e inconvenientes.

3.2.1. Processing

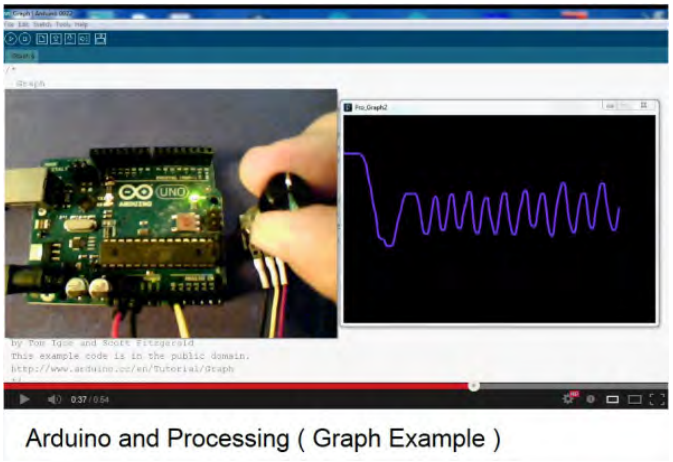

FIGURA 3.3: Gráficos a tiempo real con Processing y Arduino[38]

Es un lenguaje de programación optimizado para visualización de gráficos. Incluye un IDE propio de fácil manejo, al estilo de Arduino. Está basado en Java (figura 3.3).

Ventajas

- Cumple con los requisitos planteados.
- Fácil de utilizar.

Desventajas

- Al ser un lenguaje creado con fines didácticos, su funcionalidad no es completa (probablemente, no fuese un impedimento para este proyecto concreto).

3.2.2. Delphi / Object Pascal

Es un lenguaje orientado a objetos, derivado de Pascal. Su principal ventaja es la potencia en la gestión y administración de bases de datos.

Ventajas

- Cumple con los requisitos planteados.

Desventajas

- Ninguna destacable.

3.2.3. C Sharp

Es un lenguaje de programación orientado a objetos. Presenta un gran parecido con Java. Aunque es multiplataforma, está optimizado para SO Windows.

Ventajas

- Cumple con la mayoría de requisitos planteados.

Desventajas

- Funciona de manera óptima sólo en Sistemas Operativos Windows.

3.2.4. Python

Es un lenguaje de programación interpretado, diseñado para ser inteligible e intuitivo. Esto lo hace más sencillo de utilizar que otros lenguajes mayoritarios. Es multiparadigma.

FIGURA 3.4: Logo del lenguaje Python[39]

Ventajas

- Cumple con los requisitos planteados.
- Más intuitivo que otros lenguajes.

Desventajas

- Ninguna destacable.

3.2.5. Visual Basic

Es un lenguaje de programación dirigido por eventos (sólo se ejecutan partes del programa ante la ocurrencia de eventos). Gracias a su simplicidad sintáctica, ha sido utilizado por mucha gente a lo largo de los años como una primera introducción a la programación.

Ventajas

- Cumple con la mayoría de requisitos planteados.

Desventajas

- Sólo funciona en Sistemas Operativos Windows.

3.2.6. Real Basic

Es una herramienta de desarrollo visual de aplicaciones o RAD (Rapid Application Development). Utiliza una versión del lenguaje BASIC propia, orientada a objetos.

Ventajas

- Cumple con los requisitos planteados.

Desventajas

- No es un lenguaje de programación completo.

3.2.7. Just Basic

Es un intérprete de BASIC. No está orientado a objetos, aunque permite usar GUIs y gráficos. Fue lanzado en 2004 bajo licencia freeware. Su última actualización es del año 2005.

Ventajas

- Cumple con los requisitos planteados.

Desventajas

- No es un lenguaje de programación completo.
- Poco actualizado.
- Sólo funciona en Sistemas Operativos Windows.

3.2.8. JavaScript

Es un lenguaje de programación interpretado, orientado a objetos. Su uso principal es en aplicaciones web, del lado del cliente. Permite la creación de páginas web dinámicas, y cuenta con multitud de herramientas para el manejo de interfaces de usuario.

Ventajas

- Cumple con los requisitos planteados.
- Más intuitivo que otros lenguajes.

Desventajas

- Está principalmente enfocado a uso en web.

3.2.9. Java

FIGURA 3.5: Logo del lenguaje Java

Es un lenguaje de programación de propósito general y orientado a objetos. Está optimizado para ser multiplataforma, por lo que una aplicación compilada en una plataforma puede correr en todas las demás.

¿Por qué Java?

Observando las diferentes opciones existentes, se puede concluir que no existe una que destaque de manera clara. Esto se debe a que la lectura de puertos COM y los movimientos del ratón son funciones relativamente comunes. En todo caso, la optimización de Java para su uso multiplataforma puede considerarse una ventaja, así como la presencia de algo más de documentación en Internet que en el caso de otros lenguajes.

Finalmente, como suele ser habitual en el contexto del desarrollo software, el lenguaje elegido dependerá de la experiencia previa del desarrollador. En mi caso, al ser Java el lenguaje con el que más familiarizado estoy, consideré que era la mejor opción.

3.3. Diseño del algoritmo

En el apartado Objetivos del sistema, dentro del capítulo de Introducción, se puede ver una descripción sencilla del mismo.

A continuación, se procede a explicar el algoritmo en profundidad.

De manera global, podríamos dividirlo en los siguientes pasos:

1. Se conecta la placa controladora al ordenador por USB y se ejecuta la aplicación de escritorio.
2. El sistema toma como origen de coordenadas los valores leídos en ese instante por el acelerómetro (asumimos que, en el momento que el cuidador conecte la placa al ordenador, el usuario estará en una posición cómoda y natural, que se considerará su punto 0).
3. Entramos en el bucle principal:
   a) El programa cargado en la placa controladora revisa en bucle si se produce un movimiento o un soplido.
   b) Si esto ocurre, se envía un carácter determinado por el puerto serie.
   c) El programa Java revisa en bucle si está recibiendo información por el puerto serie.
   d) Cuando se recibe un carácter, se comprueba cuál y se ejecuta un movimiento del ratón o un clic.
4. En el ordenador, queda a la vista un cuadro de diálogo con la opción *Salir del programa*.

3.3.1. Arduino

El bucle implementado en la placa controladora se ha dividido en dos diagramas de flujo, con el fin de hacerlo más fácil de entender.

En el primer diagrama, se puede ver el bucle completo (figura 3.6). En el segundo, se desglosa el contenido del bloque *evaluación del sensor de sonido* (figura 3.9).

Bucle general

En primer lugar, se realiza la calibración del sistema, que consiste en la lectura de los valores de posición del acelerómetro. Éstos se almacenan como **origen de coordenadas**. A partir de ese origen de referencia, se considerará que el usuario ha movido la cabeza intencionadamente **si supera una distancia de 30 unidades** en cualquiera de las cuatro direcciones.

A continuación, entramos en el bucle principal:

1. Se lee el valor medido en ese instante por ambos sensores.
2. Se introduce una **espera de 50ms** para reducir la frecuencia de captación de los sensores. Así se obtendrá una cantidad de muestras por segundo mucho más manejable.
3. Comprobamos si el usuario ha desplazado la cabeza en x.
   a) En caso positivo, comprobamos el sentido del movimiento:
      1) Si el desplazamiento se ha producido en x-, se imprime un carácter **C** por el puerto serie, se evalúa el sensor de sonido y se vuelve a iniciar el bucle.
      2) Si se ha producido en x+, imprimimos un carácter **D**, se evalúa el sensor de sonido y se vuelve a iniciar el bucle.
   b) En caso negativo, comprobaremos si se ha producido algún movimiento en y.
      1) Si se ha producido un movimiento en y-, imprimimos un carácter **A**, se evalúa el sensor de sonido y se vuelve a iniciar el bucle.
      2) Si ha ocurrido en y+, imprimimos un carácter **B**, se evalúa el sensor de sonido y se vuelve a iniciar el bucle.
      3) Si no se ha producido movimiento, se evalúa el sensor de sonido y se vuelve a iniciar el bucle.

Evaluación del sensor de sonido

Al realizar las primeras pruebas con el sistema, se comprobó que, ante la ocurrencia de un evento sonoro, el sensor de sonido **no capta un valor máximo de forma constante**. Es decir, si emitimos un sonido constate cerca del sensor, veremos una respuesta como la de la figura 3.7.

Esto nos lleva a dos conclusiones importantes:

- No podemos tomar cada valor máximo aislado como un evento sonoro.
- No podemos tomar cada sucesión de valores máximos consecutivos como un evento sonoro.

Si emitimos dos sonidos constantes, separados por una breve pausa, veremos un resultado similar al reflejado en la figura 3.8.

Tras realizar varias pruebas, y obtener un resultado similar, podemos extraer una nueva conclusión:

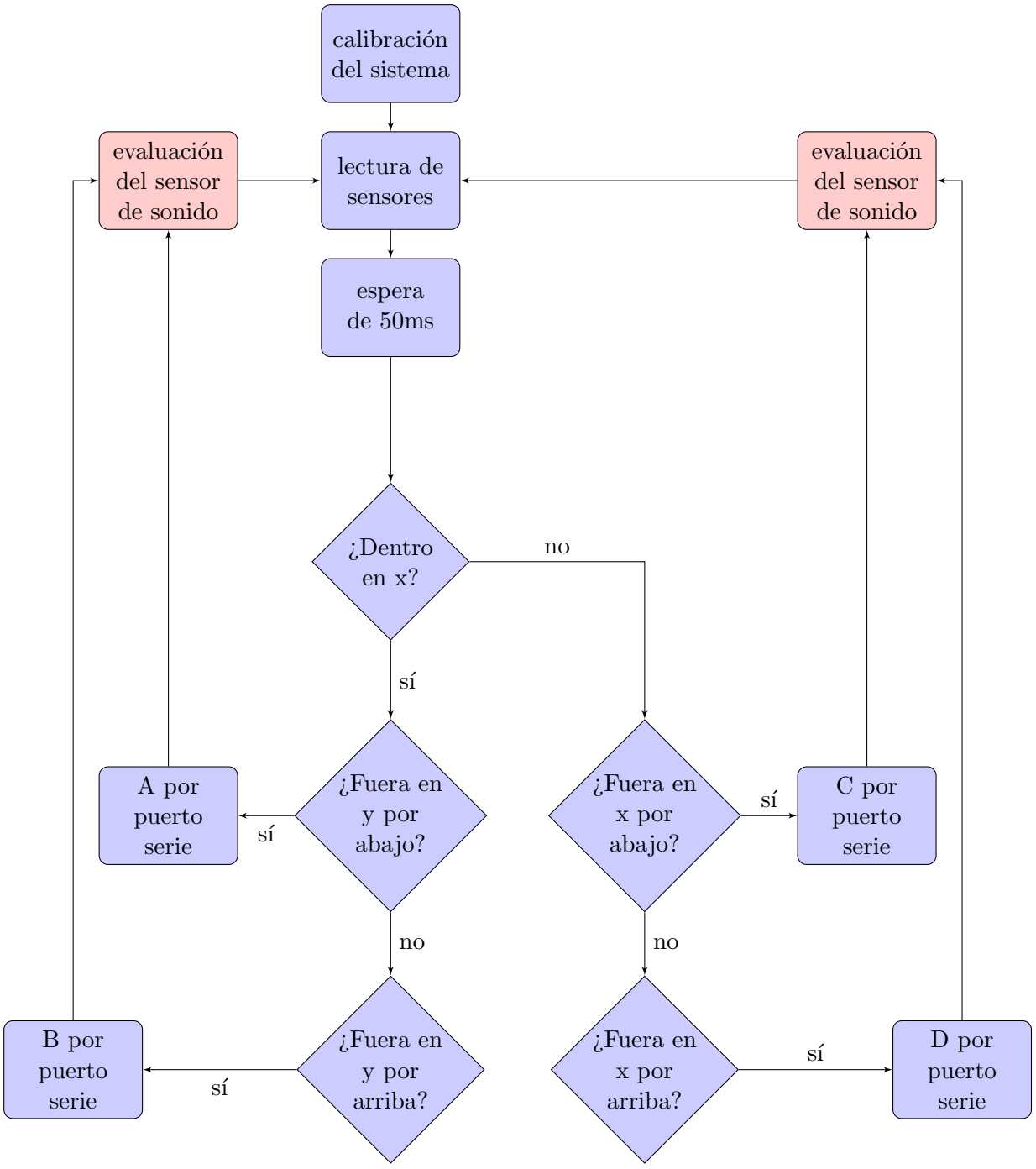

Figura 3.6: Flujograma del bucle principal

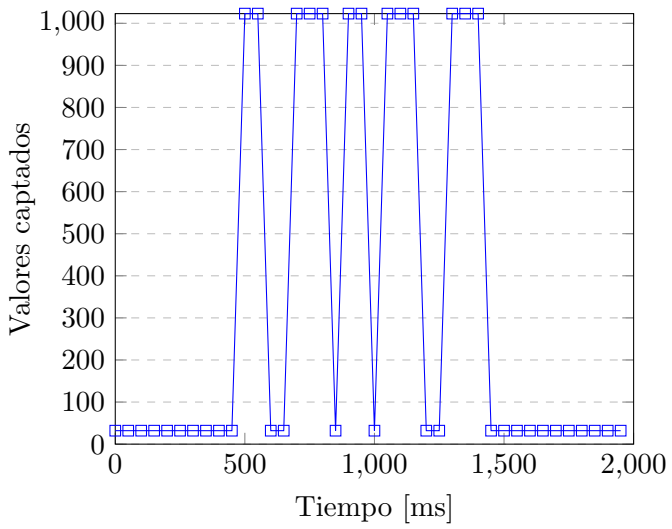

FIGURA 3.7: Captación ante un evento sonoro prolongado

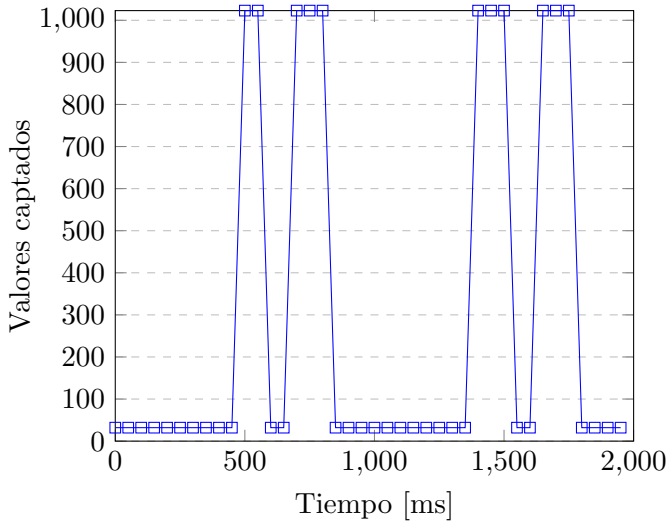

FIGURA 3.8: Captación ante dos eventos sonoros

- Al emitir varios sonidos cortos consecutivos, la duración de los mismos es casi siempre inferior a 500ms. Asumimos, por tanto, que si se registran máximos distanciados entre sí más de 500ms, corresponden a eventos independientes.

El algoritmo se puede describir con los siguientes pasos:

1. El sensor registra un máximo.
2. Durante los siguientes 500ms, se asume que todos los valores son máximos.
3. Finalizado ese intervalo, y transcurridos 1000ms más, se comprueba si era un evento aislado.

a) En caso positivo, se imprime un carácter E en por el puerto serie y continúa el flujo del programa.

b) En caso negativo, se comprueba el número de eventos consecutivos registrados:

1) Si son 2, se imprime un carácter F y continúa el flujo del programa.

2) Si son 3 o más, se imprime un carácter G y continúa el flujo del programa.

Para implementar esta lógica, en el algoritmo se incluyen tres contadores:

- **cont1:** Sirve para, una vez detectado un evento sonoro, no tener en cuenta la captación de los siguientes 500ms. Inicialmente, está desactivado. Cuando el sensor de sonido detecta un máximo, se activa, y se incrementa durante 9 iteraciones. Luego, se vuelve a desactivar.
- **since_last:** Su función es detectar si dos eventos deben ser agrupados, o son independientes. Para ello, cuenta las iteraciones transcurridas entre dos desactivaciones de cont1. Si transcurren más de 20 (1s), se consideran eventos independientes.
- **num_blows:** Se utiliza para contabilizar eventos consecutivos. Cuando since_last alcanza el valor 20, se comprueba el valor de num_blows, que será 1, 2 o 3 (un valor mayor se contabilizará como un 3).

3.3.2. Java

En la aplicación de escritorio de Java se realiza la interpretación de los caracteres recibidos y la ejecución de las acciones de ayuda para el manejo del ratón (figura 3.10).

Es importante tener en cuenta que, en ningún momento, el sistema anula el control del ratón por la vía natural. Es decir, aunque el sistema esté en marcha, siempre se puede seguir manejando el ratón con el mouse tradicional.

El algoritmo se define del siguiente modo:

1. Se abre el puerto COM para recibir los caracteres.
2. Leemos un valor recibido por dicho puerto.
3. Filtramos si el valor es relevante

   a) Si es un carácter de separación (10 o 13, en ASCII), se descarta.

   b) En caso contrario, continuamos con la ejecución

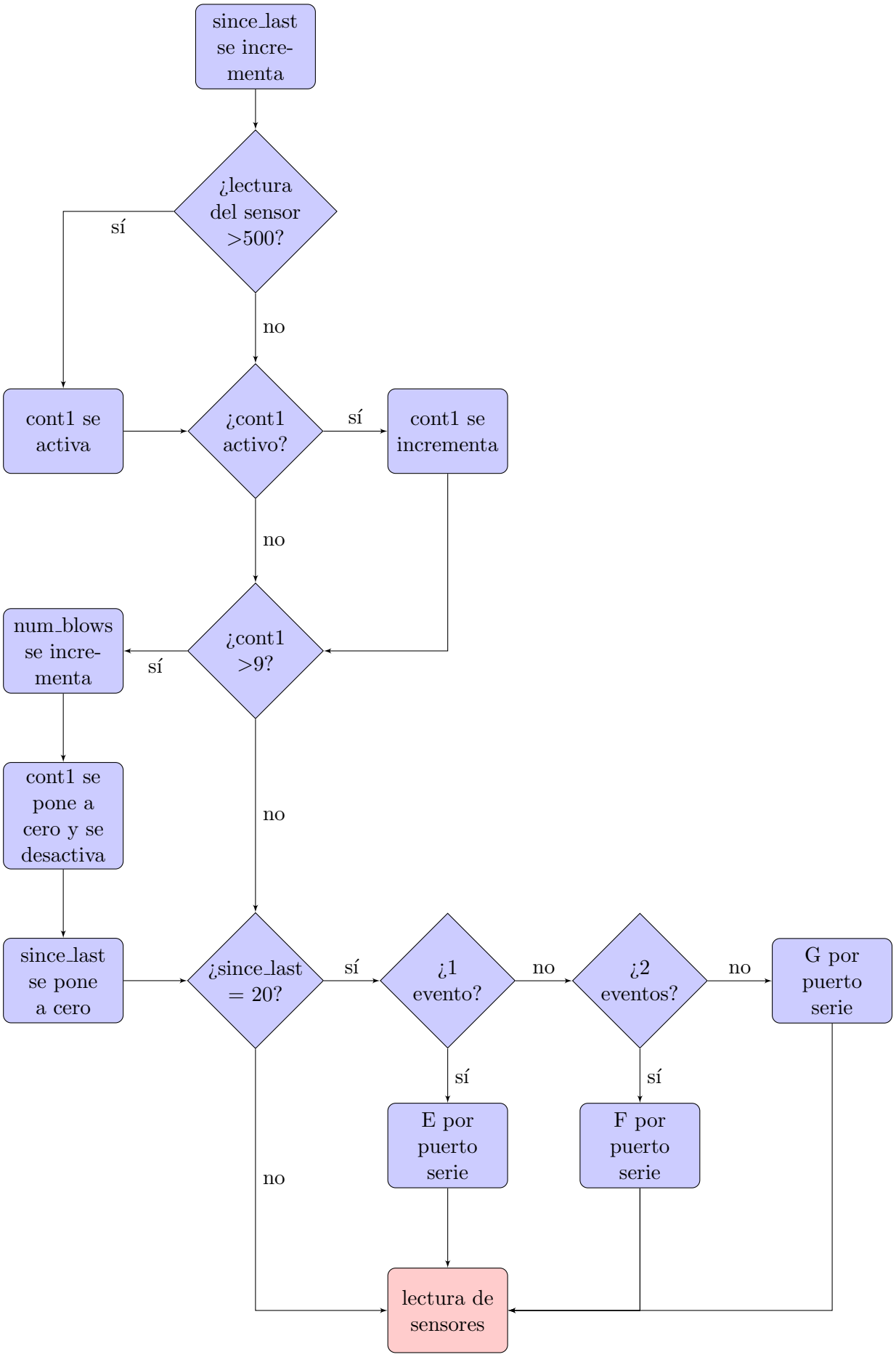

Figura 3.9: Flujograma del sensor de sonido

4. Leemos el valor actual de las coordenadas del ratón, para tomarlas como referencia.

5. En función del valor recibido, realizamos una acción:

   a) **A):** el ratón se mueve hacia y-.

   b) **B):** el ratón se mueve hacia y+.

   c) **C):** el ratón se mueve hacia x-.

   d) **D):** el ratón se mueve hacia x+.

   e) **E):** el ratón realiza un clic sencillo.

   f) **F):** el ratón realiza un doble clic.

   g) **G):** el ratón realiza un clic derecho.

6. En este punto, hay una ventana de Java a la vista que nos ofrece la opción de salir del programa en todo momento.

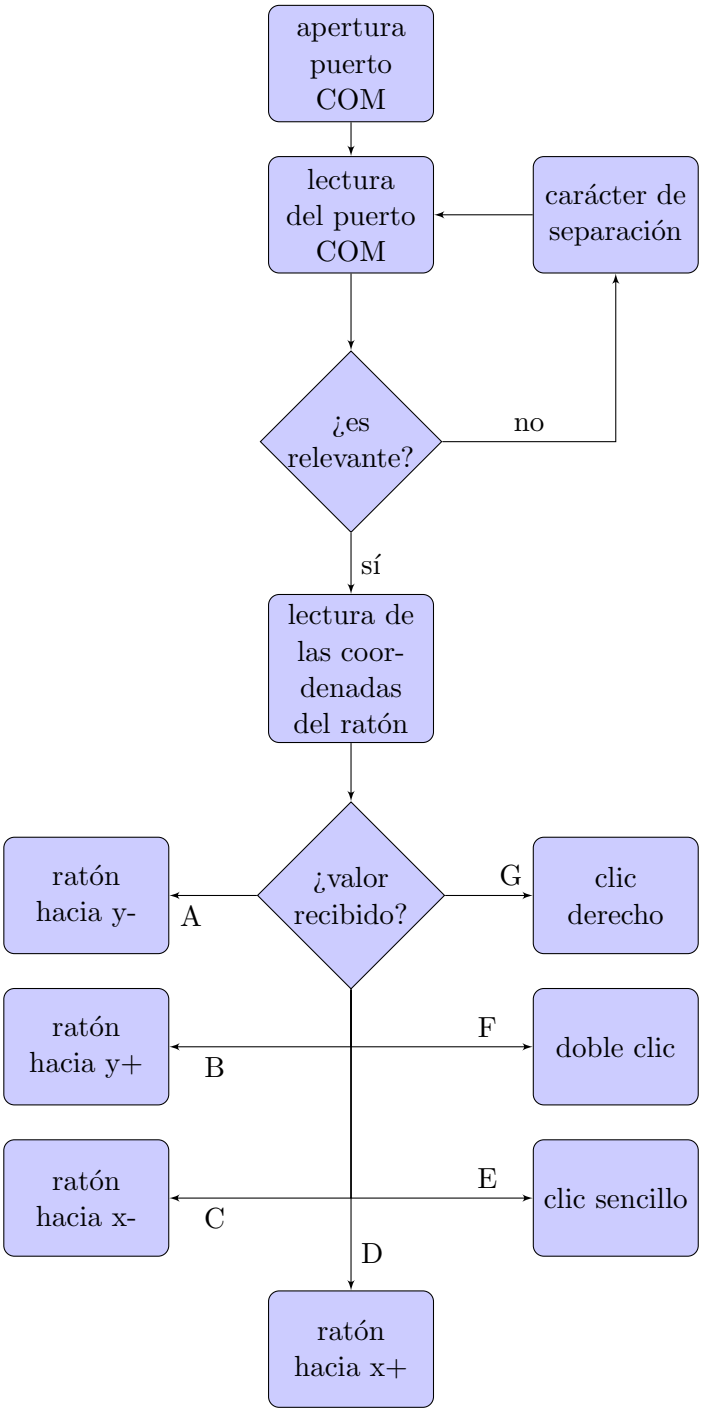

Figura 3.10: Flujograma del programa Java

Capítulo 4

Sistema propuesto

4.1. El sistema en funcionamiento

Para poder inicializar el sistema, tenemos que ejecutar el archivo *TFG.jar*. Esto implica que el usuario debe tener instalada la máquina virtual de Java [40].

Al hacer clic, aparecerá una ventana donde debemos seleccionar el puerto COM correspondiente. Generalmente, sólo habrá una opción, como se muestra en la figura 4.1.

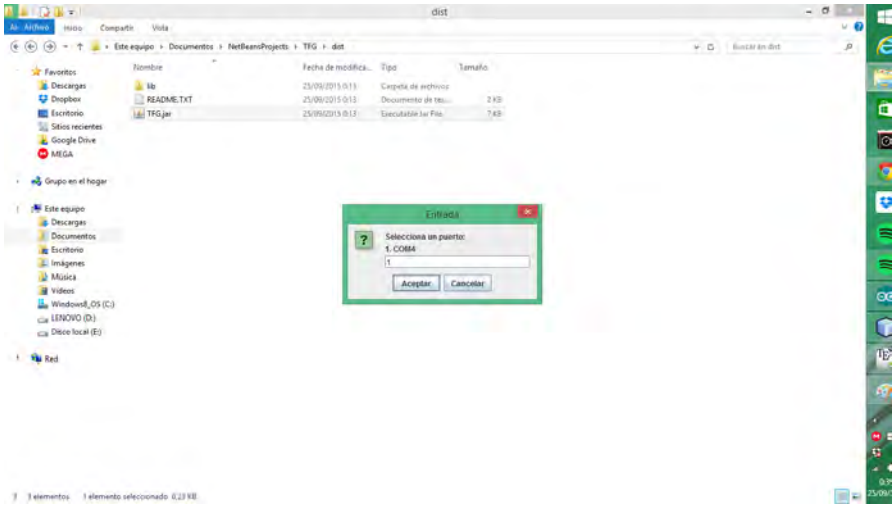

FIGURA 4.1: Inicialización del sistema

Una vez introducido el número de puerto, veremos la ventana de ejecución del programa. Ésta nos ofrece la opción de cerrarlo en todo momento (figura 4.2). Cabe destacar que **el ratón convencional permanece siempre habilitado**.

A partir de este punto, podemos controlar el ratón con movimientos de la cabeza, del modo que se explicaba en el apartado Objetivos del sistema.

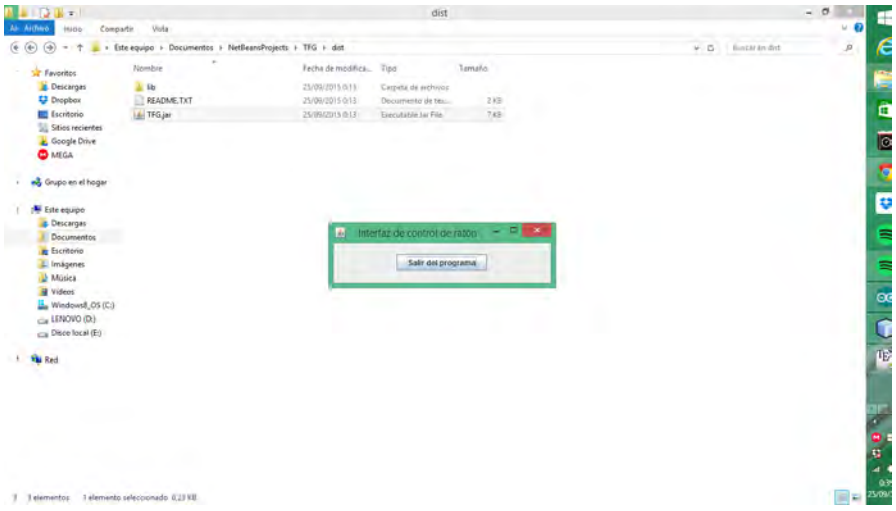

FIGURA 4.2: Ventana de ejecución

En la figura 4.3 podemos ver el efecto de un soplido simple: estaremos realizando un clic sencillo con el botón izquierdo del ratón. Por lo tanto, el cursor se situará en el lugar donde estuviese el ratón.

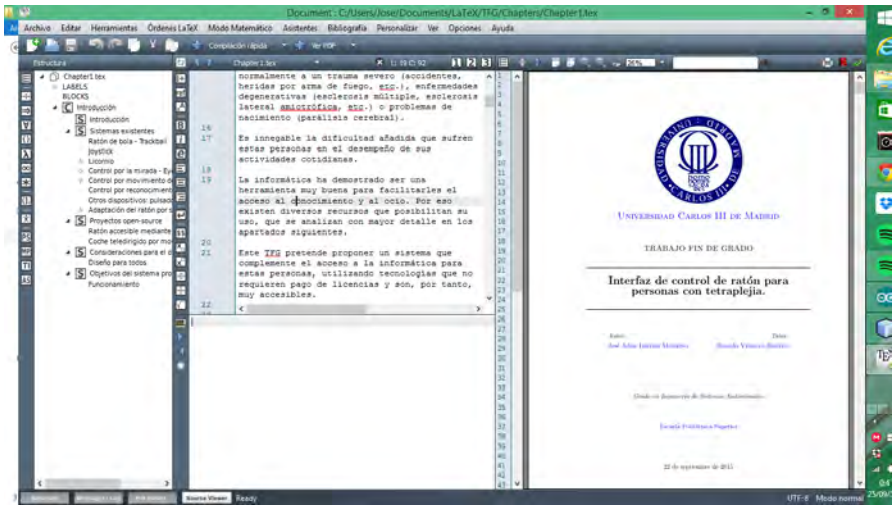

FIGURA 4.3: Efecto de un soplido simple

En la figura 4.4 vemos el efecto de un soplido doble: estaremos realizando un doble clic con el botón izquierdo del ratón. Por lo tanto, la palabra quedará seleccionada.

Por último, un triple soplido equivaldrá a un clic derecho del ratón (figura 4.5). Veremos en pantalla las diferentes opciones relativas a la palabra seleccionada.

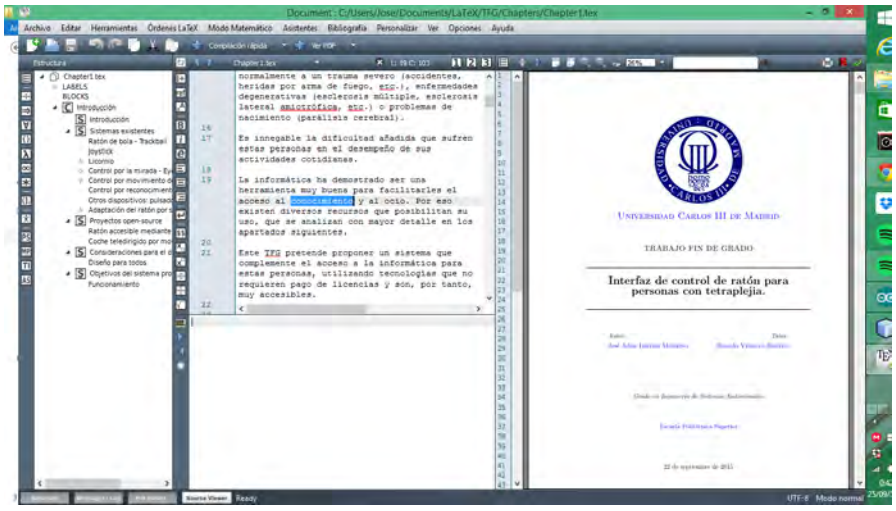

Figura 4.4: Efecto de un soplido doble

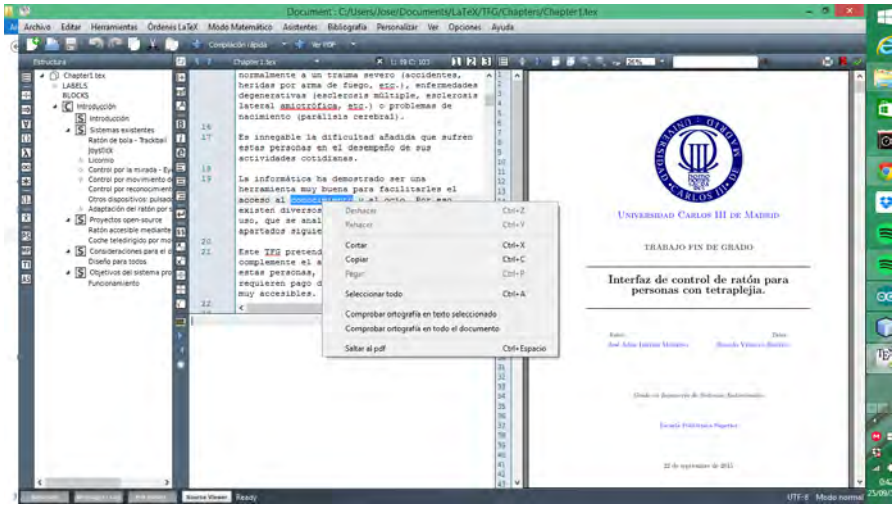

Figura 4.5: Efecto de un soplido trile

4.2. Presupuesto

A continuación, se detallan los costes del proyecto, de cara a una potencial comercialización.

En primer lugar, se desglosan los costes relativos a componentes electrónicos, en el cuadro 4.1.

En segundo lugar, se desglosan los costes del resto de piezas que componen el sistema (cuadro 4.2).

Unidades	Descripción	Precio unitario (€)	Precio total (€)
1	Placa controladora bq ZUM BT-328	34,90	34,90
1	Acelerómetro ADXL345 3 ejes	11,50	11,50
1	Sensor de Sonido FC-04	5,30	5,30
2	Módulo Board 170 Pines	3,85	7,70
1	Cable de manguera apantallada	1,07	1,07
1	Cable unipolar (varios colores)	1,65	1,65
1	Cable USB - microUSB	1,80	1,80
TOTAL			**62,12€**

Cuadro 4.1: Presupuesto - Electrónica

Unidades	Descripción	Precio unitario (€)	Precio total (€)
0,1	Filamento PLA para impresión 3D	19,90	1,99
1	Gorra básica	2,85	2,85
1	Tira de velcro	1,20	1,20
1	Cinta para el cuello	1,60	1,60
TOTAL			**7,64€**

Cuadro 4.2: Presupuesto - Otros

Unidades	Descripción	Precio unitario (€)	Precio total (€)
1	Componentes electrónicos	62,12	62,12
1	Resto de componentes	7,64	7,64
1	Honorarios (para 100uds. vendidas)	66,50	66,50
TOTAL			**136,26€**

Cuadro 4.3: Presupuesto - Total

4.2.1. Total

Para valorar el coste de las horas dedicadas por el ingeniero al desarrollo del sistema, se toma como referencia un coste bruto de 35€/h. Teniendo en cuenta que el tiempo dedicado al desarrollo del sistema ha sido de **190h**, el coste total sería de **6650€**.

Asumiendo la venta de 100 unidades del sistema, podríamos dividir los honorarios del ingeniero a partes iguales, quedando un coste por unidad como se muestra en el cuadro 4.3.

4.3. Pruebas en el Colegio San Rafael

Desde un primer momento, se pensó en la posibilidad de probar el sistema desarrollado en este TFG con potenciales usuarios reales.

4.3.1. Colegio San Rafael

El **Colegio de Educación Especial Hospital San Rafael** es un centro que acoge a alumnos con discapacidad motora, intelectual y otros trastornos asociados.

La oferta pedagógica del centro abarca las Etapas Educativas siguientes:

- Educación infantil (3-6 años).
- Educación Básica Obligatoria (6-16 años).
- Transición a la vida adulta (16-21 años).

En dichas etapas, se proporciona un tratamiento integral, que incluye aspectos pedagógicos, psicológicos, de rehabilitación, etc.

4.3.2. Pruebas en el Colegio San Rafael

Por las características del centro, se pensó que podría ser una opción ideal para poder utilizar el sistema en un entorno real.

Debido a problemas de agenda por ambas partes, no se han podido realizar estas pruebas antes del 27 de septiembre de 2015, fecha límite de entrega de la presente memoria. Por ese motivo, no se ha incluido documentación sobre la visita, aunque se intentará incluir en la sesión de defensa.

4.4. Conclusiones

Tal y como se ha detallado a lo largo de todo el documento, este sistema pretende ser un complemento de bajo coste para los recursos de ayuda ya existentes, en el ámbito del acceso a la informática. Está enfocado a personas con discapacidad motora severa (principalmente, tetraplejia).

Se han utilizado diferentes recursos de licencia abierta en el desarrollo. El objetivo de esta decisión era obtener la máxima versatilidad, así como dejar el camino abierto a posibles colaboraciones por parte de otras personas.

De cualquier modo, no deja de ser un prototipo. La realización de pruebas con usuarios reales revelaría decenas de posibles ajustes y mejoras. Una de los más evidentes sería la presencia de más opciones en la aplicación de escritorio (para regular factores como la sensibilidad del movimiento), aunque surgirían muchas más, a buen seguro.

Creo que este sistema es un perfecto ejemplo de cómo el hardware libre y la impresión 3D pueden permitirnos realizar proyectos de ayuda funcionales y de bajo coste. Éste es sólo un ejemplo de todo lo que se puede llegar a hacer. Espero que anime a otros estudiantes a mejorarlo o a idear proyectos similares que también puedan servir de ayuda a personas que lo necesiten.

4.5. Líneas futuras

En este apartado, se exponen las posibles mejoras que podrían incluirse en el sistema propuesto.

Conexión inalámbrica

En el prototipo presentado, la comunicación entre la placa controladora y el PC se realiza a través de un cable USB - microUSB. Sería relativamente sencillo hacer que esta comunicación se realizase por vía inalámbrica (por ejemplo, a través de BlueTooth).

Placa controladora más pequeña

Al ser una aplicación que no exige una gran cantidad de puertos, podría utilizarse un modelo de tamaño más reducido para ahorrar espacio en el sistema.

Aplicación de escritorio más completa

En la aplicación actual, sólo se ofrece la opción de salir del programa. Se podrían incluir otras:

- Recálculo del origen de coordenadas.

- Adaptación de los umbrales de movimiento a la movilidad del usuario en cada dirección.
- Regulación de la sensibilidad del movimiento.

Archivo ejecutable con instalador

Como se indica en el apartado El sistema en funcionamiento, el archivo ejecutable que permite iniciar el programa tiene extensión .jar. Esto quiere decir que el usuario necesita tener instalada la máquina virtual de Java (JVM) para poder usarlo. Sería ideal poder utilizar un archivo .exe con instalador incorporado, que comprobase si la JVM está instalada, y en caso negativo, buscase e instalase automáticamente la última versión.

Apéndice A

Código Arduino

```
#include <Wire.h>
#include <ADXL345.h>

//Variable adxl is an instance of the ADXL345 library
ADXL345 adxl;
int xIni,yIni,zIni;

boolean cont1_active = false;
int cont1 = 0;
int since_last = 100;
int num_blows = 0;


void setup(){
  Serial.begin(9600);

  //We need to read the position of the accelerometer in the
  //first iteration in order to use it as a reference.
  adxl.powerOn();
  adxl.readAccel(&xIni, &yIni, &zIni);
}

void loop(){

  //Now, within the main loop, we read the current position
  //of the accelerometer.
  int x,y,z;
  adxl.readAccel(&x, &y, &z);

  //And the input of the sound sensor.
  int sonido = analogRead(A0);
  since_last++;

  //If we are within the bounds in the X axis, we check the
  //position in the Y axis, and move consequently.
  if((x>(xIni-30))&&(x<(xIni+30))){
    if (y < (yIni-30)) {
      Serial.println("A");  //Up
```

```
    }else if (y > (yIni+30)){
      Serial.println("B");  //Down
    }
  //Otherwise, we move in the X axis.
  }else{
    if (x < (xIni-30)){
      Serial.println("C");  //Left
    }else if (x > (xIni+30)){
      Serial.println("D");  //Right
    }
  }

  //If a maximum is registered with the sound sensor,
  //the user has blown.
  if(sonido>500){
    cont1_active = true;
  }
  //cont1 makes the program ignore the next 9 iterations,
  //once a maximum is registered.
  if(cont1_active){
    cont1++;
  }
  if(cont1>9){
    num_blows++;
    cont1=0;
    cont1_active=false;
    since_last=0;
  }
  //since_last counts the number of iterations since last
  //blow, so we can decide if they are independent events
  //or a group of two or three.
  if(since_last==20){
    if(num_blows==1){
      Serial.println("E");  //Single left click
    }else if (num_blows==2){
      Serial.println("F");  //Double left click
    }else {
      Serial.println("G");  //Right click
    }
    num_blows=0;
  }

  delay(50);

}
```

Apéndice B

Código Java

```
package tfg;

import java.awt.Robot;
import com.fazecast.jSerialComm.*;
import java.awt.AWTException;
import java.awt.Container;
import java.awt.EventQueue;
import java.awt.GridBagLayout;
import java.awt.MouseInfo;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.InputEvent;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;

public class TFG extends JFrame{

    //Initialization of the program window and the button "Exit the program"
    public TFG() {
        initUI();
    }

    private void initUI() {
        JButton quitButton = new JButton("Salir del programa");

        quitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        createLayout(quitButton);

        setTitle("Interfaz de control de raton");
        setSize(350, 100);
```

```
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createLayout(JComponent... arg) {
        Container pane = getContentPane();
        pane.setLayout(new GridBagLayout());
        add(arg[0]);
    }


    public static void main(String[] args) throws IOException {

        //We create a robot object, which will allow us to control the mouse.
        Robot robot = null;
        try {
            robot = new Robot();
        } catch (AWTException ex) {
            Logger.getLogger(TFG.class.getName()).log(Level.SEVERE, null, ex);
        }

        //We ask the user to choose one among the COM ports
        String text = "Selecciona un puerto: ";
        SerialPort ports[] = SerialPort.getCommPorts();
        int i = 1;
        for(SerialPort port : ports) {
            text = text + "\n" + i++ + ". " + port.getSystemPortName();
        }
        int inputValue = Integer.parseInt(JOptionPane.showInputDialog(text));
        int chosenPort = inputValue;
        SerialPort port = ports[chosenPort - 1];
        if(port.openPort()) {
                System.out.println("Successfully opened the port.");
        } else {
                System.out.println("Unable to open the port.");
                return;
        }
        port.setComPortTimeouts(SerialPort.TIMEOUT_NONBLOCKING, 0, 0);

        //The program window is launched
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                TFG ex = new TFG();
                ex.setVisible(true);
            }
        });

        //The main loop begins
        boolean aux = true;
        while(aux){
            int inputArduino = port.getInputStream().read();
            //Characters 10 and 13 are skipped, as they are used as a
            //separation between two pieces of actual data.
            if((inputArduino==13)||(inputArduino==10)){
```

```
                System.out.print("\n");
            }else{
                System.out.println(inputArduino);
                //If we are receiving useful data, first we need to read the
                //current position of the mouse.
                int xIni = (int)MouseInfo.getPointerInfo().getLocation().getX();
                int yIni = (int)MouseInfo.getPointerInfo().getLocation().getY();

                //Now, a movement or a click will be done, depending on
                //the character we have received.
                if(inputArduino==65){
                    robot.mouseMove(xIni, yIni-10);
                }else if (inputArduino==66){
                    robot.mouseMove(xIni, yIni+10);
                }else if (inputArduino==67){
                    robot.mouseMove(xIni-10, yIni);
                }else if (inputArduino==68){
                    robot.mouseMove(xIni+10, yIni);
                }else if (inputArduino==69){
                    robot.mousePress(InputEvent.BUTTON1_MASK);
                    robot.mouseRelease(InputEvent.BUTTON1_MASK);
                }else if (inputArduino==70){
                    robot.mousePress(InputEvent.BUTTON1_MASK);
                    robot.mouseRelease(InputEvent.BUTTON1_MASK);
                    robot.mousePress(InputEvent.BUTTON1_MASK);
                    robot.mouseRelease(InputEvent.BUTTON1_MASK);
                }else if (inputArduino==71){
                    robot.mousePress(InputEvent.BUTTON3_MASK);
                    robot.mouseRelease(InputEvent.BUTTON3_MASK);
                }
            }
        }
        }

}
```

Apéndice C

Diagrama de conexión

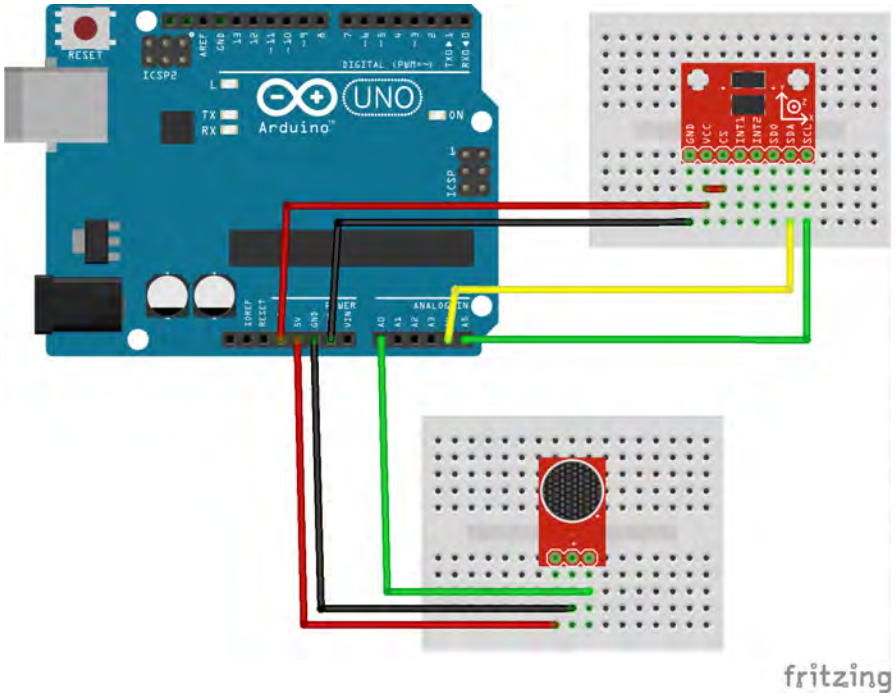

Figura C.1: Diagrama de conexión

Bibliografía

[1] INTEF. *Acceso al ordenador de alumnado con dificultad motora.* http://www.ite.educacion.es/formacion/materiales/146/cd/m4_dificultad_motora/acceso_directo_al_ratn.html/ [Última visita: 19/09/2015].

[2] Switched on Media. *Screen Readers and Assistive Technology: 5 Ways to Improve Web Accessibility.* http://switchedonagency.com/blog/screen-readers-and-assistive-technology-5-ways-to-improve-web-accessibility/ [Última visita: 19/09/2015].

[3] S.L. SINPROMI. *Licornio.* http://www.sinpromi.es/es/civat/show/licornio/ [Última visita: 19/09/2015].

[4] Francisco A. Nieto. *Licornio para pantallas táctiles capacitivas.* http://www.crmfalbacete.org/recursosbajocoste/catalogo/Licornio_para_pantallas_tactiles_capacitivas.pdf [Última visita: 20/09/2015].

[5] Víctor Ortega. *Esclerosis Lateral Amiotrófica: Compendio de información.* https://sites.google.com/site/vicortega2/etran [Última visita: 20/09/2015].

[6] EyeComTec. *Enable Viacam.* https://www.eyecomtec.com/3103-enable-viacam [Última visita: 20/09/2015].

[7] CREA Software. *Enable Viacam.* http://eviacam.sourceforge.net/ [Última visita: 20/09/2015].

[8] Goli Mohammadi. *Build the Eyeboard Open Source Eye-Tracking Project.* http://makezine.com/2012/02/23/build-the-eyeboard-open-source-eye-tracking-project/ [Última visita: 19/09/2015].

[9] Buzz Humphrey. *Quadriplegic Control of an RC Car.* http://www.instructables.com/id/Quadriplegic-Control-of-an-RC-Car/ [Última visita: 20/09/2015].

[10] Design for All Foundation. *Design for All.* http://designforall.org/ [Última visita: 20/09/2015].

[11] BBVAopen4u. *The best alternatives to Arduino: from the Do it Yourself to the Internet of Things*. http://bbvaopen4u.com/en/actualidad/best-alternatives-arduino-do-it-yourself-internet-things [Última visita: 20/09/2015].

[12] Raspberry Pi Foundation. *Raspberry Pi*. https://www.raspberrypi.org/ [Última visita: 20/09/2015].

[13] BeagleBoard.org Foundation. *BeagleBone Black*. http://beagleboard.org/black [Última visita: 20/09/2015].

[14] Aidilab SECO USA Inc. *UDOO Neo*. http://www.udoo.org/udoo-neo/ [Última visita: 20/09/2015].

[15] Minnowboard.org. *MinnowBoard MAX*. http://www.minnowboard.org/ [Última visita: 20/09/2015].

[16] Nanode Open Source. *Nanode*. http://www.nanode.eu/ [Última visita: 20/09/2015].

[17] S.L. Libellium Comunicaciones Distribuidas. *Libellium Waspmote*. http://www.libelium.com/ [Última visita: 20/09/2015].

[18] Inc. Texas Instruments. *MSP430 Launchpad*. http://www.ti.com/ww/en/launchpad/launchpad.html?DCMP=rtos&HQS=ep-sdo-rtos-pr-lp-launchpad-en [Última visita: 20/09/2015].

[19] Pinguino Project. *Pinguino PIC32*. http://www.pinguino.cc/ [Última visita: 20/09/2015].

[20] STMicroelectronics N.V. *STM32 Discovery*. http://www.st.com/web/en/catalog/tools/FM116/SC959/SS1532/PF250863?sc=stm32-discovery [Última visita: 20/09/2015].

[21] LLC. PJRC.com. *Teensy 3.1*. https://www.pjrc.com/ [Última visita: 20/09/2015].

[22] Arduino.cc. *Introduction*. https://www.arduino.cc/en/Guide/Introduction [Última visita: 20/09/2015].

[23] bq.com. *Placa bq ZUM BT-328*. http://www.bq.com/es/placa-zum-bt [Última visita: 20/09/2015].

[24] Wikipedia. *Degrees of freedom (mechanics)*. https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics) [Última visita: 20/09/2015].

[25] Deal Extreme. *Angular Transducer Tilt Slant Angle Sensor Module for Arduino)*. http://www.dx.com/p/angular-transducer-tilt-slant-angle-sensor-module-for-arduino-150783#.Vf6RMRHtmko [Última visita: 20/09/2015].

[26] Amazon. *SainSmart GY-85 Sensor Modules Accelerometer Gyroscope Module).* http://www.amazon.com/SainSmart-Modules-Accelerometer-Gyroscope-HMC5883L/dp/B00KQDIU3U [Última visita: 20/09/2015].

[27] Deal Extreme. *ADXL345 Digital Acceleration Tilt Angle Sensor Module for Arduino).* http://www.dx.com/p/adxl345-digital-acceleration-tilt-angle-sensor-module-for-arduino-148741#.Vf6UWhHtmko [Última visita: 20/09/2015].

[28] Starlino. *A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications).* http://www.starlino.com/imu_guide.html [Última visita: 20/09/2015].

[29] Microsystems. *Surface Micromachined Gyroscope.* http://www.microsystems.metu.edu.tr/gyroscope/gyroscope.html [Última visita: 20/09/2015].

[30] BYU Mechanical Engineering. *Introduction to Microelectromechanical Systems (MEMS)).* https://compliantmechanisms.byu.edu/content/introduction-microelectromechanical-systems-mems [Última visita: 20/09/2015].

[31] Freescale Semiconductor. *Tilt Sensing Using a Three-Axis Accelerometer.* http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf [Última visita: 20/09/2015].

[32] Vilros. *Big 12mm Button.* https://www.vilros.com/arduino/ard-acc/big-12mm-buttons.html [Última visita: 20/09/2015].

[33] Conrad. *Interlink FSR-400 Pressure Sensor.* http://www.conrad.com/ce/en/overview/0231110/Pressure-Sensors [Última visita: 20/09/2015].

[34] Electan. *Sharp IR Sensor - GP2Y0A21YK0F.* http://www.electan.com/sensor-flexion-sparkfun-p-3135.html [Última visita: 20/09/2015].

[35] Emartee. *Sensor de Flexión Sparkfun.* http://www.emartee.com/product/42073/Sharp%20IR%20Sensor%20%20%20GP2Y0A21YK0F [Última visita: 20/09/2015].

[36] Mercado Libre. *Módulo sensor de sonido Arduino.* http://electronica.mercadolibre.com.ar/componentes-electronicos/modulo-sensor-detector-de-sonido-con-microfono-ideal-arduino [Última visita: 20/09/2015].

[37] FreeCAD. *Download.* http://www.freecadweb.org/wiki/?title=Download [Última visita: 27/09/2015].

[38] Arduining. *Arduino and Processing.* http://arduining.com/2013/08/05/arduino-and-processing-graph-example/ [Última visita: 20/09/2015].

[39] Wikipedia. *Python*. https://es.wikipedia.org/wiki/Python [Última visita: 20/09/2015].

[40] Java. *Download Options*. https://www.java.com/en/download/help/download_options.xml [Última visita: 24/09/2015].