PODÓMETRO CONSTRUÍDO CON ARDUINO E ACELERÓMETRO

Christian Cidre Taboada

Brais González Fernández

Alicia Rojo Vázquez

Paula Rojo Vázquez

STEMbach
bacharelato de excelencia

Táboa de Contidos

RESUMO

A actividade física está directamente relacionada coa nosa saúde. Por iso, a práctica regular de exercicio pode contribuír a mellorala, reducindo o risco de padecer enfermidades cardiovasculares ou estrés, ou axudando a controlar o peso.

Unha das principais actividades físicas practicadas polo ser humano é camiñar. Pódese dicir que unha persoa que camiñe arredor de 10000 pasos leva unha vida activa.

Cando camiñamos, non nos movemos con velocidade uniforme. Se tomamos como referencia un punto do noso corpo, por exemplo a cadeira, poderiamos comprobar que a aceleración da mesma non é constante, tanto no eixe horizontal como no vertical.

Neste traballo de investigación, e facendo uso da tecnoloxía co emprego de placas Arduino e acelerómetros, tratamos de proporcionar un instrumento que poida axudar a controlar a actividade física de camiñar, realizada por calquera persoa e en calquera lugar.

Para iso, construíuse un podómetro do seguinte xeito: tomáronse diferentes medidas da aceleración obtida nun punto do corpo ao camiñar, mediante un acelerómetro conectado á placa Arduino. A continuación, calculáronse os pasos dados a partir destas medidas, e finalmente enviouse o resultado a unha pequena pantalla.

Conséguese como produto final un podómetro sinxelo e fácil de manexar, que pode facilitar a tarefa de contar os pasos dados a calquera persoa que desexe mellorar o seu estado de saúde a través da actividade física de camiñar.

1. INTRODUCIÓN

O emprego das novas tecnoloxías está presente en case todos os ámbitos da vida e da sociedade, como a industria alimentaria [1], a educación [2], ou a medicina [3]. Os avances tecnolóxicos posibilitan que calquera persoa poida dispor de dispositivos electrónicos que lle permiten desfrutar dunha maior calidade de vida [4]. Non obstante, tamén poden repercutir negativamente na saúde: o emprego excesivo de dispositivos e tecnoloxía pode provocar adicción [5], trastornos de visión [6], ou falta de actividade física [7].

Pola contra, a práctica de exercicio aeróbico pode contribuír notablemente á mellora da nosa condición física, pois reduce o risco de padecer enfermidades cardiovasculares, diabetes ou hipertensión, entre outras doenzas [7].

Entre as posibles actividades físicas para manterse en forma destaca, pola súa sinxeleza, a de camiñar. Realizada de forma regular, axuda a manter os niveis de colesterol en valores normais, controla o peso e mellora a circulación. Para conseguir beneficios mediante este exercicio, débense dar arredor de 10000 pasos ao día [8].

Unha vez que se aprende, camiñar convértese nun acto mecánico, que se executa incontables veces de forma automática. Se analizamos con detemento esta actividade, observamos que nun punto do noso corpo, que tomamos como referencia, se produce un cambio na aceleración vertical e horizontal (Figura 1).

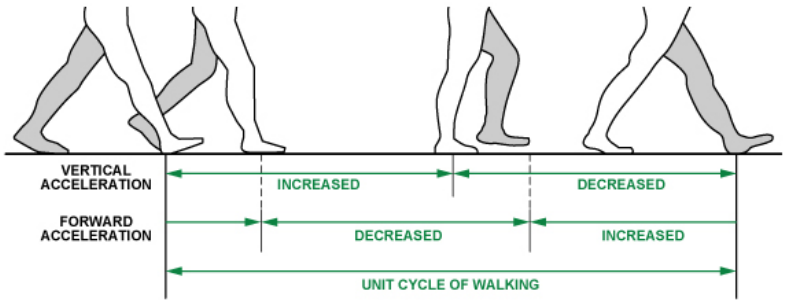

Figura 1. Representación do proceso de camiñar [9].

Ao empezar a camiñar, a aceleración no eixe X aumenta, xa que temos que desequilibrarnos para empezar a marcha; logo flexiónase o xeonllo e, ao estiralo, crece no eixe Y e decrece no X; o proceso repítese constantemente nunha sucesión de impulsos e freadas.

Os podómetros son dispositivos que permiten levar conta do número de pasos que se dan. Estes dispositivos, que poden ser electrónicos ou electromecánicos, posúen un sensor interno que detecta o balanceo de cada paso, e o rexistra [10].

O obxectivo deste proxecto é deseñar un podómetro que detecte e conte correctamente os pasos dados por unha persoa camiñando a diferente ritmo, mediante un acelerómetro conectado a unha placa Arduino. Rexistraranse os valores de aceleración nun punto do corpo, e farase unha análise dos datos obtidos, tratando de traducir esas diferenzas de aceleración en pasos, mediante a linguaxe de programación R. Unha vez adestrado e testado o algoritmo de contar pasos, impleméntase en Arduino, de xeito que nun pequeno monitor conectado se visualice, como resultado final, o número de pasos dados por cada persoa.

2. CONTEXTUALIZACIÓN E XUSTIFICACIÓN

Este proxecto de investigación levouse a cabo durante parte dos cursos 2018-19 e 2019-20, no marco do StemBach desenvolto no IES Eduardo Blanco Amor (Ourense), en colaboración coa Escola Superior de Enxeñaría Informática da Universidade de Vigo.

2.1. Xustificación

O exercicio a través da camiñada permite mellorar a condición física, necesaria para gozar dun bo estado de saúde. Para cumprir co obxectivo de 10000 pasos diarios, sería recomendable dispor dalgún dispositivo que conte os pasos que damos. Actualmente, existen no mercado numerosos dispositivos que permiten medir estes pasos e, a partir deles, calcular a distancia percorrida. A día de hoxe, é habitual atopar podómetros integrados en pulseiras ou reloxos intelixentes [11], [12], mentres que outros son simples apps que se instalan nos teléfonos móbiles [13] - [15].

Con este proxecto pretendemos construír e programar un podómetro baseado nun microcontrolador de baixo custo que nos permita demostrar o seu principio de funcionamento.

3. MATERIAIS E MÉTODOS

3.1. Materiais

O primeiro paso do proxecto foi a construción dun prototipo que permitise tomar medidas da aceleración. Escollemos para iso unha placa de hardware e software libre, Arduino, que conta cun microcontrolador e un conxunto de pins de entrada e saída, ós que conectar diferentes sensores e actuadores.

Inicialmente empregouse Arduino [16], [17] como un sistema de adquisición de datos co que obter un conxunto de medidas da aceleración, pero para a análise e o estudo estatístico traballamos co software R, moito máis adecuado para isto.

Indicamos a continuación os diferentes dispositivos que conforman o prototipo.

Arduino UNO

Tipo de placa Arduino (Figura 2) cunha memoria flash de 32 KB e unha velocidade de reloxo de 16 MHz. Dispón de 14 entradas e saídas dixitais e 6 entradas analóxicas. A programación farase a través do propio IDE de Arduino.

Figura 2.Placa Arduino.

Unidade de medición inercial MPU6050

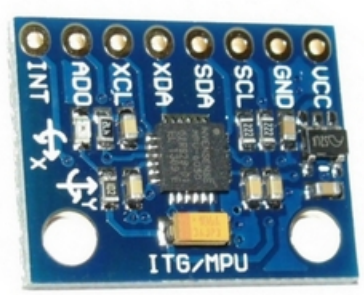

Figura 3. Unidade de medición inercial.

Esta IMU (Inertial Measurement Unit) (Figura 3) é un dispositivo que combina un acelerómetro de tres eixes con xiroscopio, tamén de tres eixes. Úsase para obter información da orientación e posicionamento dun obxecto no espazo (en teléfonos móbiles por exemplo), aínda que neste caso unicamente traballaremos coas medidas da aceleración.

Este modelo ten un convertedor A/D de 16 bits por canal, o que dá un rango de valores entre -32768 e +32768 correspondentes a -2g e +2g respectivamente: para 1g obteriamos o valor de 16384 (segundo datos do fabricante, con g a aceleración da gravidade).

A conexión a Arduino é por I2C, usando o pin A4 para datos e o A5 para reloxo.

O bus I2C é un tipo de arquitectura mestre-escravo, actuando Arduino como mestre e proporcionando un sinal de reloxo co que se sincroniza co MPU.

Para a obtención das medidas necesitamos incluír na programación de Arduino as librarías MPU6050, I2C e WIRE.

Módulo para tarxeta microSD

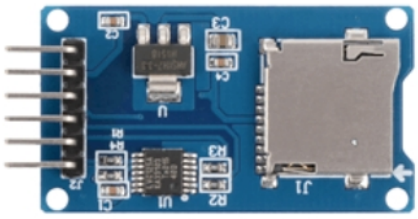

Figura 4. Módulo para tarxeta microSD.

Elemento que se acopla á placa e que permite almacenar nunha tarxeta tipo micro SD a información de aceleración que se obtén coa IMU nas distintas probas. (Figura 4).

A tarxeta cos datos extráese e conéctase posteriormente a un ordenador, no que contamos cun software apropiado, R.

Para poder usala necesitamos recorrer a librarías SD e SPI específicas. Tamén é importante ter creado un arquivo tipo texto na tarxeta, previamente á realización das probas.

Pantalla LCD

Figura 5. Pantalla LCD.

Unha vez optimizado o algoritmo, impleméntase en Arduino e visualízanse directamente os pasos nunha pantalla LCD 16x2 (Figura 5).

A libraría necesaria para a programación é a LiquidCrystal.h

Software R

R [18] é unha contorna e linguaxe de programación de software libre moi utilizada no entorno científico, especialmente indicada para a análise estatística e gráfica. Distribúese baixo a licencia GNU GPL, e pode ser empregada en diferentes sistemas operativos: UNIX, Windows e MacOSX.

3.2. Métodos

3.2.1. Deseño do experimento

O deseño do experimento comprende tres etapas:

1. Recollida de datos con Arduino: montaxe do prototipo, programación de Arduino e realización dos paseos.

2. Análise dos datos en R e optimización do algoritmo.

3. Paso do algoritmo a Arduino e prototipo definitivo con visualización en LCD.

A Figura 6 mostra o diagrama de bloques do proceso:

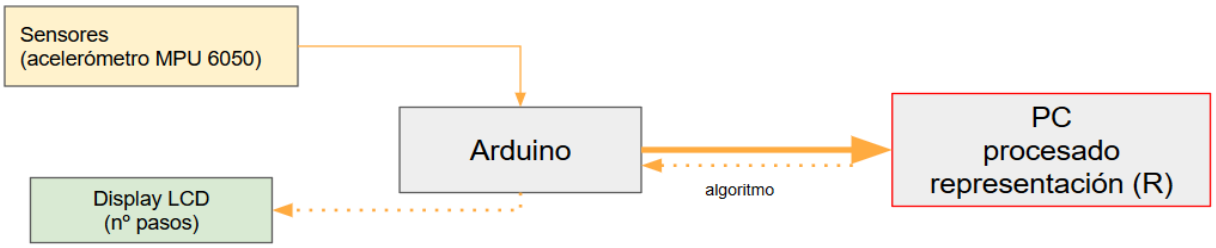

Figura 6. Diagrama de bloques do proceso desenvolvido.

A conexión de Arduino, MPU6050 e módulo micro SD para obter os datos de aceleración é a seguinte (Figura 7):

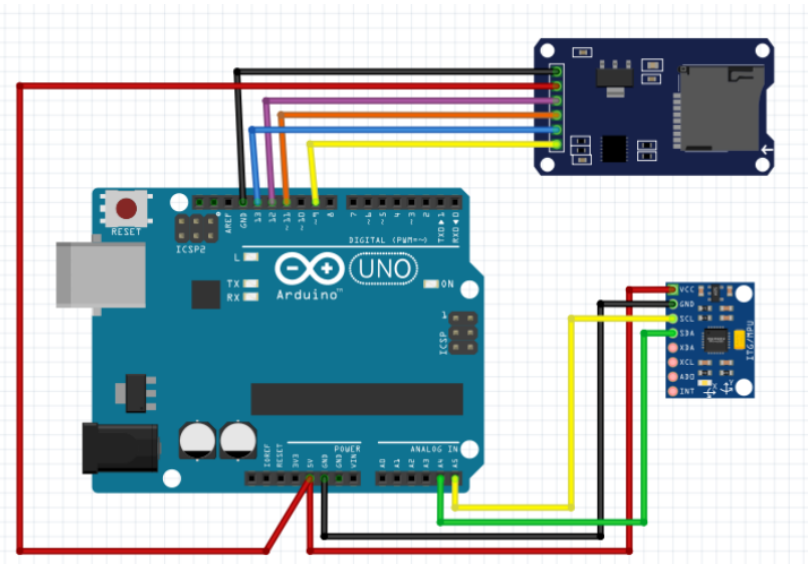

Figura 7. Conexión dos diferentes elementos.

Para traballar co acelerómetro foi preciso facer un calibrado por software do dispositivo, mantendo a unidade en repouso e horizontal durante o proceso. Inténtase conseguir que as aceleracións nos eixos X e Y se aproximen a cero para compensar calquera desviación na montaxe dos compoñentes da placa.

Na Figura 8 observamos os datos proporcionados despois do calibrado:

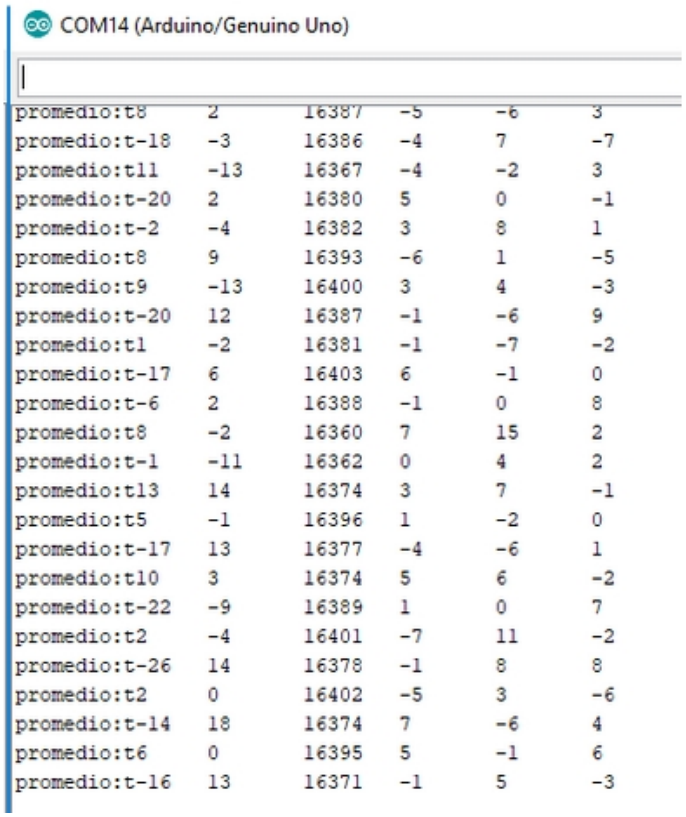

Figura 8. Datos obtidos tras o calibrado.

As tres primeiras columnas son as aceleracións en X, Y e Z. Na terceira observamos a medida da gravidade terrestre (g). As tres últimas corresponden a medicións do xiroscopio (non se usarán).

No programa para a toma de datos, implementouse unha función que se executa durante uns segundos cando se conecta Arduino (co dispositivo acoplado á cadeira do camiñante pero en repouso), que fai a media de 100 medidas de aceleración en cada eixe. Esta media restarase dos datos obtidos en cada momento para eliminar a parte correspondente á posición de traballo, que é aproximadamente vertical. Así eliminamos a compoñente da gravidade e os datos referiranse á aceleración debida o movemento.

A continuación faise un escalado dos datos para obter a aceleración en m/s^2:

$$a = (\text{medida real} - \text{media inicial}) * 9.8/16384{,}0 \text{ (para } a_x, a_y \text{ e } a_z) \quad (1)$$

Por último, calcúlase o módulo da aceleración, que se gardará na tarxeta micro SD:

$$|\vec{a}| = \sqrt{a_x^{\,2} + a_y^{\,2} + a_z^{\,2}} \qquad (2)$$

Unha vez recollidos os datos, observamos claramente as variacións que se producen na aceleración (Figura 9):

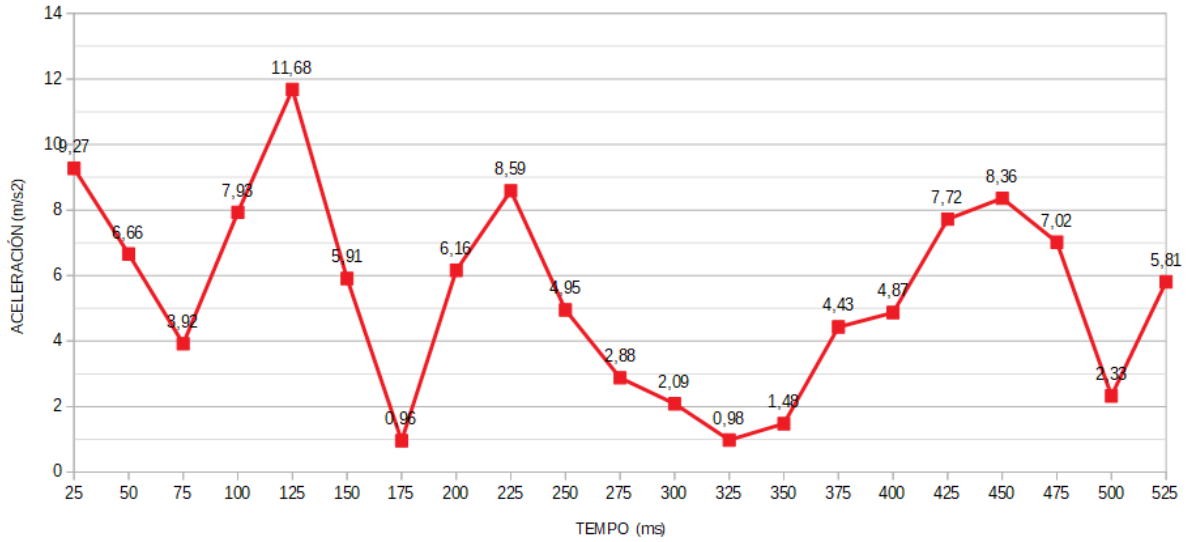

Figura 9. Representación gráfica das variacións da aceleración.

Non todos os picos da gráfica anterior se corresponden con un paso. Nalgúns casos están producidos polo ruído dos compoñentes electrónicos, normalmente de alta frecuencia. Polo tanto, tentamos deseñar un algoritmo en R que detecte os pasos reais e non teña en conta outro tipo de sinais. Para iso, consideraremos que un dato é un paso se:

• O seu valor é maior que outro catalogado como limiar (variable "limiar" no programa).

• Desde o paso anterior o dato diminuíu por debaixo do limiar (oscilación ó camiñar).

• Desde o paso anterior pasou un tempo mínimo (número determinado de datos) debido ás limitacións das persoas (variable "retardo2" no programa).

Para escoller os valores das variables que identifican ó valor limiar e o retardo entre pasos deseñouse un algoritmo que calculara tamén o erro cometido ó contabilizar os pasos totais. A combinación de valores que nos ofreza o menor erro será a que se teña en conta para programar o Arduino e obter a lectura directa dos pasos.

A Figura 10 amosa o esquema da conexión final de Arduino, IMU e pantalla.

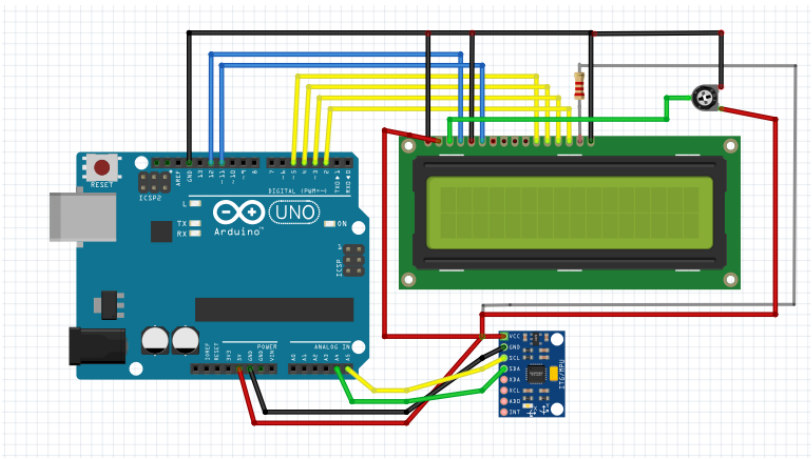

Figura 10. Esquema final do dispositivo construído.

3.2.2. Poboación mostral

Para realizar os paseos de toma de datos, e para ter mostras diferentes da poboación, escolléronse tres persoas de idades e sexo diferentes:

- Home de 17 anos.
- Muller 1 de 53 anos.
- Muller 2 de 37 anos.

3.2.3. Protocolo de recollida de datos

A recollida de datos lévase a cabo do seguinte xeito:

- Escóllese un espazo sen xente que poida interferir nos paseos (patio exterior do instituto, corredor do instituto, garaxe particular, beirarrúa despexada).
- O prototipo acóplase á cadeira do camiñante suxeito ó pantalón cunha cinta.
- Conéctase a batería e permanécese en repouso uns segundos.
- Comenza o paseo á velocidade escollida contando simultaneamente os pasos reais.
- Sempre que sexa posible un observador exterior conta tamén os pasos.
- Reinicialízase Arduino para poñer a cero o tempo de medida na tarxeta microSD e identificar así cada paseo.

Ao transferir a tarxeta ao ordenador créanse tantos arquivos de tipo csv como paseos diferentes se obtiveran, incluíndo no nome o número de pasos reais, o nome da persoa, o número de paseo e a velocidade; isto permite identificalos en todo momento.

O número de valores de aceleración varía en cada ficheiro entre os 300 e os 875 aproximadamente, segundo sexa a velocidade rápida ou lenta. Isto fai que o total de datos que temos que analizar sexa moi elevado, en torno ós 18000, e indica a necesidade de utilizar un software adecuado, como R, para o seu tratamento.

Cada persoa fai 12 camiñadas de 35 pasos a diferentes velocidades: lenta, normal e rápida, obtendo 36 rexistros tal e como se mostra na Táboa A.1 do Anexo. Ademais tamén se realizou un conxunto de paseos de 300 pasos, co fin de comprobar como funciona en camiñadas longas.

3.2.4. Análise estatística

O programa en R que detecta os pasos fai tamén a análise estatística dos valores de limiar e retardo que ofrecen menor diferencia co valor real do número de pasos que o suxeito contou directamente en cada camiñada. Os 36 rexistros obtidos divídense para o seu estudo en dous conxuntos, un de adestramento, con 27 elementos, e un de test, con 9. Como pode observarse na Táboa A.1, fíxose un reparto equitativo segundo velocidade e persoa para non condicionar os resultados. O primeiro conxunto, máis grande, servirá para estudar o paso e determinar o valor das variables e o segundo para probar o funcionamento sobre uns datos diferentes ós que se usaron para o adestramento.

Considéranse 6 posibles valores de limiar (entre 3 e 8), e 10 posibles valores de retardo (entre 3 e 12). O programa xera para cada conxunto os seguintes arquivos:

- Sesenta arquivos .csv cos resultados de pasos reais, pasos detectados, diferenza e porcentaxe de fallo, para cada combinación de variables.
- Un arquivo co erro e a desviación estándar do erro para cada combinación, que nos permite obter aquela que máis se aproxima ó valor real.
- Sesenta gráficos coa posición dos pasos detectados (en vermello) e o valor de aceleración medido.

As Figuras 11 e 12 amosan un gráfico para valores 4 e 10 nun arquivo a velocidade rápida no que coinciden pasos reais e detectados, e outro para valores 10 e 10 a velocidade rápida no que vemos o aumento do erro debido a un limiar moi alto:

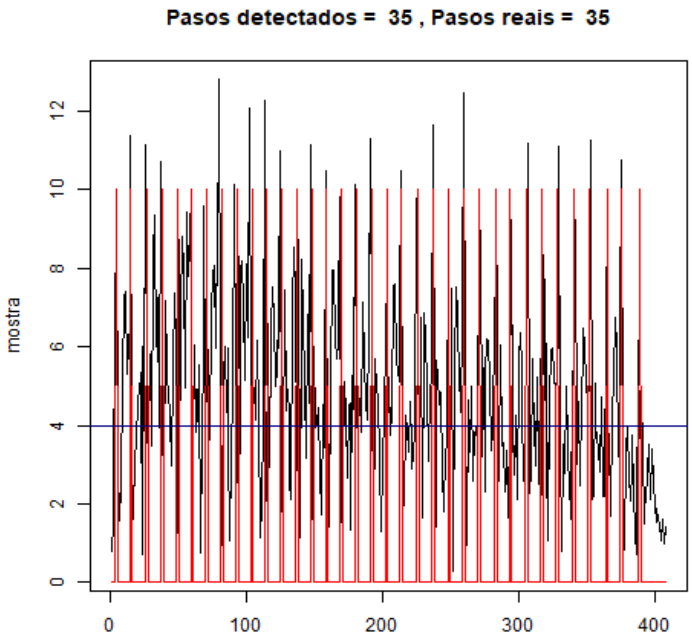

Figura 11. Coincidencia de pasos reais e detectados.

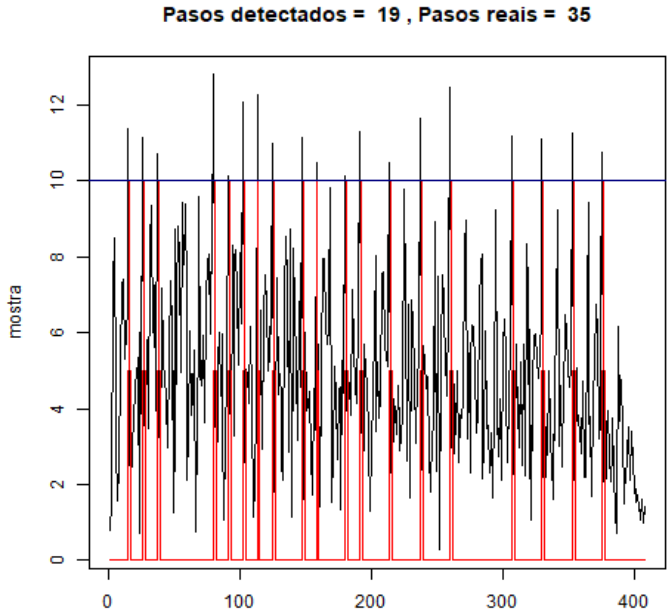

Figura 12. Aumento de pasos erróneos detectados.

3.2.5. Implementación do algoritmo de detección en Arduino

Unha vez optimizado o algoritmo en R, impleméntase na linguaxe Arduino, prescindindo xa da tarxeta SD e incorporando unha pantalla LCD na que se amosan directamente os pasos que se detectan (Figura 13).

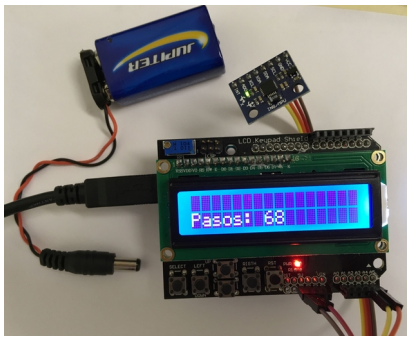

Figura 13. Prototipo final do podómetro.

4. RESULTADOS

Na Táboa A.2 do Anexo mostramos os erros e correspondente desviación estándar para as posibles combinacións de limiar entre 3 e 8, e retardo entre 3 e 12, co conxunto de adestramento.

Os valores que producen menos erros son limiar 4 e retardo 10 ou 11.

Para o conxunto de test faise a mesma análise, e obtéñense os resultados da Táboa A.3 (Anexo).

Á vista dos resultados obtidos, escóllense os valores de limiar 4 e retardo 10 para o programa definitivo, cos seguintes erros (Táboa 1):

Táboa 1. Resultados para un limiar = 4 e un retardo = 10.

	Erro (%)	Desviación estándar do erro (%)
Adestramento	13.4	13.6
Test	10.7	12.0
Media	12.1	12.8

Dado que os valores nos erros son bastante elevados, faise unha análise dos arquivos para as mellores combinacións do conxunto de adestramento (Táboa A.4 do Anexo) e de test (Táboa A.5 do Anexo).

Obsérvase que os valores cos máximos erros sempre coinciden nos mesmos arquivos. Analizado o número de datos que conteñen eses arquivos (para o grupo de adestramento) e comparados cos feitos pola mesma persoa a unha velocidade similar, vese que son significativamente distintos, ó que nos leva a pensar na posibilidade de que houbese un erro humano na conta dos pasos reais.

A análise dos arquivos de 300 pasos feitos polo home confirma o valor das variables como limiar 4 e retardo 10, e ademais mostra un erro moito menor (Táboa A.6 do Anexo).

5. CONCLUSIÓN

Os resultados do traballo mostran que é posible obter datos da variación de aceleración que se produce ó camiñar e que, cun algoritmo adecuado, eses datos tradúcense nos pasos que dá unha persoa. Ademais é posible facelo con hardware de baixo custo como é o IMU 6050 e a tarxeta Arduino.

Por outra parte, empregouse o método científico, formulando e posteriormente comprobando unha hipótese, sendo de gran utilidade esta experiencia por poder confirmar de xeito propio a utilidade e veracidade do mesmo. Tamén nos permitiu traballar en equipo nun proxecto científico, colaborando e aprendendo de catedráticos universitarios, sendo unha valiosa oportunidade poder facelo antes incluso de rematar o Bacharelato.

Ademais adquirimos novos coñecementos, físicos (en canto á aceleración), tecnolóxicos (Arduino, o acelerómetro, LCD...) e informáticos (o software R), que nos servirán para a nosa futura formación científica.

6. PROPOSTA DE CONTINUIDADE

Para a obtención de resultados máis fiables proponse aumentar a poboación mostral así como o número de paseos e a distancia dos mesmos.

Tamén se propón cambiar o sistema de toma de datos substituíndo a tarxeta SD, que deu moitos problemas, por un microprocesador con conexión sen fíos (ESP8266 por exemplo).

Por último sería interesante mellorar o deseño do conxunto cun sistema de suxeición máis eficaz que evitase movementos de rebote que poidan ser fonte de erros.

7. BIBLIOGRAFÍA

[1] Parzanese M. (2016). Tecnologías para la industria alimentaria. Ultrasonidos. Ficha, 19, 1-9.

[2] García, C.M.A, & Maldonado, L.J.P. (2005). Aplicaciones educativas de las tecnologías de la información y la comunicación. Ministerio de Educación.

[3] Pérez, M. A. A., Morales, M. B. P., & Pérez, M. C. A. (2019). Utilización de apps móviles en el control de la salud. La Ciencia al Servicio de la Salud, 10(2), 22-29.

[4] Atkinson, R. D., & Castro, D. (2008). Digital quality of life: Understanding the personal and social benefits of the information technology revolution. Available at SSRN 1278185.

[5] Echeburúa, E., & De Corral, P. (2010). Adicción a las nuevas tecnologías ya las redes sociales en jóvenes: un nuevo reto. Adicciones, 22(2), 91-96.

[6] Aparna, J., Devi, R. G., & Jyothipriya, A. (2019). Eye complications in children due to excessive use of electronic gadgets. Drug Invention Today, 12(6), 1182-1184.

[7] Márquez Rosa, S., Rodríguez Ordax, J., & De Abajo Olea, S. (2006). Sedentarismo y salud: efectos beneficiosos de la actividad física. Apuntes, 83, 12-24.

[8] Wattanapisit, A., & Thanamee, S. (2017). Evidence behind 10,000 steps walking. *Journal of Health Research*, *31*(3), 241-248.

[9] Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer: https://www.analog.com/media/en/analog-dialogue/volume-44/number-2/articles/pedometer-design-3-axis-digital-acceler.pdf (última consulta: febreiro 2020).

[10] http://tapasdeciencia.blogspot.com/2016/03/como-funciona-un-podometro.html (última consulta: decembro 2019).

[11] https://www.fitbit.com/es/charge3 (última consulta: febreiro 2020).

[12] https://www.apple.com/es/watch (última consulta: febreiro 2020).

[13] https://accupedo.softonic.com/android (última consulta: febreiro 2020).

[14] https://play.google.com/store/apps/details?id=cc.pacer.androidapp (última consulta: febreiro 2020).

[15] https://play.google.com/store/apps/details?id=com.runtastic.android.me.lite (última consulta: febreiro 2020).

[16] Gallego, E. (2014). Empezando con Arduino UNO. Complubot.

[17] Torrente, O. (2013). Arduino, curso práctico de formación. Alfaomega.

[18] R: https://cran.r-project.org (última consulta: febreiro 2020).

ANEXOS

Táboa A.1. Rexistros obtidos para as diferentes camiñadas.

Nº paseo	Suxeito	Nº pasos reais	Velocidade	Adestramento	Test
1	Home	35	Lenta	X	
2	Home	35	Lenta	X	
3	Home	35	Lenta	X	
4	Home	35	Lenta		X
5	Home	35	Normal	X	
6	Home	35	Normal	X	
7	Home	35	Normal	X	
8	Home	35	Normal		X
9	Home	35	Rápida	X	
10	Home	35	Rápida	X	
11	Home	35	Rápida	X	
12	Home	35	Rápida		X
1	Muller 1	35	Lenta	X	
2	Muller 1	35	Lenta	X	
3	Muller 1	35	Lenta	X	
4	Muller 1	35	Lenta		X
5	Muller 1	35	Normal	X	
6	Muller 1	35	Normal	X	
7	Muller 1	35	Normal	X	
8	Muller 1	35	Normal		X
9	Muller 1	35	Rápida	X	
10	Muller 1	35	Rápida	X	
11	Muller 1	35	Rápida	X	
12	Muller 1	35	Rápida		X
1	Muller 2	35	Lenta	X	
2	Muller 2	35	Lenta	X	
3	Muller 2	35	Lenta	X	
4	Muller 2	35	Lenta		X
5	Muller 2	35	Normal	X	
6	Muller 2	35	Normal	X	
7	Muller 2	35	Normal	X	

8	Muller 2	35	Normal		X
9	Muller 2	35	Rápida	X	
10	Muller 2	35	Rápida	X	
11	Muller 2	35	Rápida	X	
12	Muller 2	35	Rápida		X

Táboa A.2. Resultados obtidos en termos de erro para os diferentes limiares e retardos para o conxunto de adestramento. En vermello aparecen as combinacións limiar-retardo con menor erro.

LIMIAR	RETARDO	ERRO (%)	DESVIACIÓN (%)
3	3	64.0	39.3
3	4	56.8	35.3
3	5	49.6	31.0
3	6	42.7	26.6
3	7	35.8	23.1
3	8	29.6	21.8
3	9	23.4	20.8
3	10	19.2	19.8
3	11	17.9	18.4
3	12	18.3	17.4
4	3	60.6	47.0
4	4	51.9	41.1
4	5	41.7	36.3
4	6	32.2	31.4
4	7	23.0	24.9
4	8	18.1	21.1
4	9	13.5	16.4
4	**10**	**10.7**	**12.0**
4	**11**	**11.2**	**10.0**
4	12	12.7	10.1
5	3	53.3	40.4
5	4	44.7	32.5
5	5	34.9	28.3
5	6	28.3	23.6
5	7	24.4	21.1
5	8	19.4	19.1
5	9	15.7	18.7
5	10	14.5	18.8
5	11	16.7	18.0

5	12	20.0	18.1
6	3	44.4	29.5
6	4	41.5	29.3
6	5	36.7	28.9
6	6	33.1	28.7
6	7	32.3	28.7
6	8	30.5	29.5
6	9	30.6	29.2
6	10	30.5	29.4
6	11	34.6	26.9
6	12	39.0	24.9
7	3	51.7	35.3
7	4	50.6	34.5
7	5	50.6	34.4
7	6	49.4	35.2
7	7	49.2	35.2
7	8	48.6	36.1
7	9	48.8	35.9
7	10	49.5	35.3
7	11	54.7	30.4
7	12	58.9	26.9
8	3	62.3	35.2
8	4	62.1	35.4
8	5	62.3	36.2
8	6	61.9	36.7
8	7	62.0	36.6
8	8	62.1	36.4
8	9	62.1	36.4
8	10	63.2	34.9
8	11	68.9	27.2
8	12	73.2	23.4

Táboa A.3. Resultados obtidos en termos de erro para os diferentes limiares e retardos para o conxunto de test. En vermello aparece a combinación limiar-retardo con menor erro.

LIMIAR	RETARDO	ERRO (%)	DESVIACIÓN (%)
3	3	70.6	56.6
3	4	56.8	46.7
3	5	42.7	37.6
3	6	35.0	32.1
3	7	24.1	27.4
3	8	18.4	17.8
3	9	16.3	12.2
3	10	12.1	7.4
3	11	10.2	7.2
3	12	10.6	10.5
4	3	54.8	48.2
4	4	44.7	36.5
4	5	35.4	27.3
4	6	26.2	16.8
4	7	22.3	15.0
4	8	16.9	13.7
4	9	14.0	14.2
4	**10**	**13.4**	**13.6**
4	11	15.2	13.3
4	12	17.4	14.9
5	3	48.0	30.5
5	4	43.9	24.1
5	5	36.6	21.3
5	6	31.4	21.6
5	7	26.1	24.2
5	8	24.2	25.4
5	9	24.6	25.1
5	10	24.9	24.8
5	11	28.0	23.1

5	12	31.6	21.9
6	3	50.1	34.7
6	4	46.3	33.0
6	5	44.3	33.5
6	6	40.0	36.5
6	7	38.1	37.7
6	8	38.8	37.0
6	9	39.0	36.8
6	10	39.7	36.3
6	11	42.9	33.9
6	12	48.6	31.0
7	3	50.4	37.3
7	4	48.8	38.8
7	5	47.0	40.6
7	6	46.3	41.4
7	7	47.6	40.1
7	8	48.2	39.5
7	9	48.2	39.5
7	10	48.9	39.0
7	11	54.6	34.9
7	12	57.7	33.0

Táboa A.4. Resultados para as mellores combinacións para o conxunto de adestramento. En vermello móstranse as porcentaxes de fallo maiores dun 25% (coinciden no mesmo paseo).

Limiar/Retardo							
	4/10		**4/11**		**4/12**		
Pasos reais	Detectados	% fallo	Detectados	% fallo	Detectados	% fallo	**Nº medidas en arquivo**
35	34	3	34	3	34	3	
35	34	3	34	3	33	6	
35	41	17	38	9	37	6	
35	34	3	34	3	34	3	
35	32	9	32	9	32	9	
35	33	6	33	6	33	6	
35	22	**37**	22	**37**	22	**37**	**696**
35	40	14	39	11	38	9	780
35	38	9	38	9	35	0	843
35	42	20	36	3	36	3	
35	43	23	39	11	37	6	
35	34	3	33	6	31	11	
35	53	**51**	50	**43**	47	**34**	**704**
35	30	14	30	14	30	14	490
35	31	11	31	11	31	11	480
35	33	6	33	6	33	6	
35	35	0	35	0	35	0	
35	34	3	34	3	33	6	
35	36	3	32	9	29	17	430
35	35	0	32	9	29	17	408
35	26	**26**	25	**29**	25	**29**	**378**
35	34	3	32	9	29	17	
35	34	3	31	11	29	17	
35	32	9	29	17	27	23	
35	34	3	32	9	29	17	
35	34	3	32	9	29	17	
35	33	6	30	14	28	20	

Táboa A.5. Resultados para as mellores combinacións para o conxunto de test. En vermello móstranse as maiores porcentaxes de fallo.

Limiar/Retardo						
	4/10		**4/11**		**4/12**	
Pasos reais	Detectados	% fallo	Detectados	% fallo	Detectados	% fallo
35	33	6	33	6	33	6
35	29	17	29	17	29	17
35	19	46	19	46	19	46
35	36	3	36	3	35	0
35	31	11	31	11	31	11
35	34	3	34	3	34	3
35	33	6	30	14	29	20
35	28	20	27	23	23	34
35	32	9	30	14	28	20

Táboa A.6. Erros obtidos cos parámetros finais considerados. En vermello o erro e a desviación para o limiar e retardo definitivos.

CAMIÑADA 300 PASOS			
LIMIAR	**RETARDO**	**ERRO (%)**	**DESVIACIÓN (%)**
4	**10**	**1.3**	**1.5**
4	11	3.7	6.3
4	12	10.3	9.6
5	10	2.0	2.0
5	11	6.3	7.8
5	12	14.7	9.7

Programa Arduino que permite obter as aceleracións en bruto polo porto serie

```
// Obtención de aceleracións en bruto
// Librerías I2C para controlar o mpu6050
// A librería MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h

#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

MPU6050 sensor;

// Valores RAW (sen procesar) do acelerómetro nos eixes x,y,z
int16_t ax, ay, az;

void setup() {
  Serial.begin(57600);    //Iniciando porto serial
  Wire.begin();           //Iniciando I2C
  sensor.initialize();    //Iniciando o sensor

  if(sensor.testConnection()) Serial.println("Sensor iniciado
correctamente");
  else Serial.println("Erro");
}

void loop() {
  // Ler as aceleracións
  sensor.getAcceleration(&ax, &ay, &az);

  // Mostrar as lecturas separadas por un [tab]
  Serial.print("a[x y z]:\t");
  Serial.print(ax); Serial.print("\t");
  Serial.print(ay); Serial.print("\t");
  Serial.println(az);
  delay(100);
}
```

Programa Arduino que permite obter e gardar na micro SD o módulo da aceleración

```
// Ó conectar manter o acelerómetro quieto uns segundos para calibrar
// Calibramos e escalamos datos brutos con información do sensor
// Escribimos datos en memoria microSD


#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include <SPI.h>
#include <SD.h>


MPU6050 sensor;
File meusDatos;
// Valores brutos da aceleración
int16_t  ax, ay, az;


// Variables para a calibración
float xMedia;
float yMedia;
float zMedia;

void setup() {

    Wire.begin();
    Serial.begin(38400);
    sensor.initialize();
    calibrate();

    Serial.print("Iniciando SD ...");
    if (!SD.begin(9)) {
        Serial.println("Error ao iniciar");
    return;
    }
    Serial.println("Iniciado correctamente");

}

void loop() {

  // Lemos o acelerómetro
  sensor.getAcceleration(&ax, &ay, &az);

  // Restamos offset de calibración e escalamos valores brutos
  float ax_m_s2 = (ax-xMedia) * (9.81/16384.0);
  float ay_m_s2 = (ay-yMedia) * (9.81/16384.0);
  float az_m_s2 = (az-zMedia) * (9.81/16384.0);

  // Calculamos módulo da aceleración
  float totAcel= sqrt(pow(ax_m_s2,2)+pow(ay_m_s2,2)+pow(az_m_s2,2));
```

```
 // Abrimos arquivo en SD e escribimos valores
  meusDatos=SD.open("acel.txt", FILE_WRITE);

  if (meusDatos) {
        meusDatos.print(String(millis()/1000));
        meusDatos.print(",");
        meusDatos.println(String(totAcel));
        meusDatos.close();

  }
  else {
    Serial.println("Error abrindo arquivo");
  }
  delay(25);
}

// Función que fai a media de 100 medidas en cada eixe
void calibrate() {
  float xval[100]={0};
  float yval[100]={0};
  float zval[100]={0};

  float sum=0;
  float sum1=0;
  float sum2=0;

  for (int i=0;i<100;i++) {
    sensor.getAcceleration(&ax, &ay, &az);
    xval[i]=float(ax);
    yval[i]=float(ay);
    zval[i]=float(az);
    sum=xval[i]+sum;
    sum1=yval[i]+sum1;
    sum2=zval[i]+sum2;
  }
  xMedia=(sum/100.0);
  yMedia=(sum1/100.0);
  zMedia=(sum2/100.0);
  delay(200);

}
```

Programa Arduino que permite mostra os pasos na LCD

```
// Ó conectar manter o acelerómetro quieto uns segundos para calibrar
// Calibramos e escalamos datos brutos con información do sensor
// Visualízanse pasos na LCD


#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);


MPU6050 sensor;

// Valores brutos da aceleración
int16_t  ax, ay, az;


// Variables para a calibración

float xMedia;
float yMedia;
float zMedia;

// Variables para contar pasos

int bandeira=0; // Para indicar se seguimos contando ou non
int contadorPasos=0;
int retardo=11;
int retardo2=10; // Retardo ata que contamos un novo paso
int limiar=4; // Valor a partir do que consideramos a mostra como paso


void setup() {

    Wire.begin();
    Serial.begin(38400);
    // Número de filas e columnas do display
    lcd.begin(16,2);
    // Posición cursor columna 0, liña 0
    lcd.setCursor(0,0);
    lcd.print("Pasos: ");
    sensor.initialize();
    calibrate();

}

void loop() {
  // Lemos o acelerómetro
  sensor.getAcceleration(&ax, &ay, &az);

  // Restamos offset de calibración e escalamos valores brutos
  float ax_m_s2 = (ax-xMedia) * (9.81/16384.0);
```

```
  float ay_m_s2 = (ay-yMedia) * (9.81/16384.0);
  float az_m_s2 = (az-zMedia) * (9.81/16384.0);

  // Calculamos módulo da aceleración
  float totAcel= sqrt(pow(ax_m_s2,2)+pow(ay_m_s2,2)+pow(az_m_s2,2));
  delay(25);

  // Calculamos pasos
   retardo = retardo + 1;
     if (totAcel > limiar && bandeira == 0 && retardo > retardo2){ //
Condición para considerar que hai un paso
      contadorPasos = contadorPasos + 1;
      bandeira = 1;
      retardo = 0; // Evitamos contar pasos ata que pasa un tempo
    }
    if (totAcel < limiar && bandeira == 1){ // Se o valor da aceleración é
menor que o limiar, ponse a bandeira a 0 para poder medir
      bandeira = 0;
    }
  lcd.setCursor(0,1);
  lcd.print(contadorPasos);

}

// Función que fai a media de 100 medidas en cada eixe
void calibrate() {
  float xval[100]={0};
  float yval[100]={0};
  float zval[100]={0};

  float sum=0;
  float sum1=0;
  float sum2=0;

  for (int i=0;i<100;i++) {
    sensor.getAcceleration(&ax, &ay, &az);
    xval[i]=float(ax);
    yval[i]=float(ay);
    zval[i]=float(az);
    sum=xval[i]+sum;
    sum1=yval[i]+sum1;
    sum2=zval[i]+sum2;
  }
  xMedia=(sum/100.0);
  yMedia=(sum1/100.0);
  zMedia=(sum2/100.0);
  delay(200);
 }
```

Programa R para detectar pasos, xerar gráficas e facer a análise estatística dos erros

```
# Selección de todos os arquivos do directorio
medidas = list.files(pattern = paste("acel.*csv",sep=""))


# limiar: Valor a partir do que se considera a mostra analizada un paso.
Por defecto limiar = 5
# retardo2: Retardo permitido. Por defecto retardo2 = 5

resultados <- matrix(nrow = length(medidas), ncol = 5, dimnames =
list(     c(1:length(medidas)),     c("Nome",     "Pasos     Reais","Pasos
Detectados","Diferencia", "Porcentaxe Erro")))
er <- matrix(nrow = length(medidas), ncol = 1)

percorrido_limiar =s eq(3,6) # Fixamos máximo e mínimo para percorrer o
parámetro limiar
percorrido_retardo2 = seq(8,12) # Facemos o mesmo para o outro parámetro,
retardo2
n_probas = length(percorrido_limiar)*length(percorrido_retardo2) # Número
de combinacións de valores que se van probar
error <- matrix(nrow = n_probas, ncol = 4, dimnames = list( c(1:n_probas),
c("Limiar", "Retardo","Error","desv_error")))

ii<-0 # Índice para o vector de errores medios

for (limiar in percorrido_limiar) {
  for (retardo2 in percorrido_retardo2 ) {

    # Xeración de táboa para gardar os resultados en .csv"

     # Procesamos todos os ficheiros

    j <- 0 # Variable para almacenar o resultado na matriz resultados

    for (ficheiro in medidas) {

      j <- j + 1
      # Lectura de datos do ficheiro
      datosOrixinais = read.csv(ficheiro, header = FALSE)
      print(ficheiro)

      # Extracción de pasos reais, a partir do nome do ficheiro
      # Suponse a mesma estrutura de nome para todos os ficheiros
      pasosReais = as.integer(substr(ficheiro, 11, 12))

      # Xeramos vector con dimensiónn igual ó número de mostras existentes
      # mostra <- as.numeric(datosOrixinais[!is.na(datosOrixinais$V1),])
      mostra <- datosOrixinais[!is.na(datosOrixinais$V1),]

     # Inicializamos as variables para contar o número de pasos existentes
na mostra

      bandeira = 0  # Bandeira que indica se seguimos contando pasos ou non
```

```
      contadorPasos = 0   # contador de pasos

      retardo = 12  # Tempo que agardamos para contar o seguinte paso

         pasosDetectados = replicate(length(muestra), 0)   # Vector para
almacenar as posicións dos pasos detectados. Poñemos valor de 10 (só para
pintar na gráfica en vermello)

        posicion = 1   # Variable que contabiliza as posiciós do vector
pasosDetectados nas que se detecta un paso

      # Percorremos todo o vector coas medidas realizadas, e verificamos se
se cumplen as condicións para ser un paso

      for (i in muestra){
        retardo = retardo + 1
        posicion = posicion + 1
          if (i > limiar && bandeira == 0 && retardo > retardo2){  # Se o
valor da medida es maior ca o limiar, e estamos dentro do retardo
permitido, detéctase un paso
           contadorPasos = contadorPasos + 1 # Increméntase nunha unidade o
número de pasos detectados
           bandeira = 1  # Ponse a bandeira a 1
            retardo = 0 # Ponse o retardo a 0 para evitar contar outro nun
periodo de tempo moi seguido
             pasosDetectados [posicion] = 10 # A efectos de representacion
gráfica unicamente, a posicion do paso detectado no vector pasosDetectados
ponse a 10
        }

        if (i < limiar && bandeira == 1){ # Se o valor da mostra é menor ca
o do limiar, poñemos novamente a bandeira a 0, para poder comparar o
seguinte valor
          bandeira = 0
        }
      }

      # Representamos gráficamente os resultados

         figura <- paste(gsub(".csv","", ficheiro), limiar, "_", retardo2,
".tiff",sep="")
       print(figura)
       tiff(filename = figura)
       plot (mostra, type = "l")
        title(main = paste("Pasos detectados = ", contadorPasos, ", Pasos
reais = ", pasosReais))
         lines(seq(1, length(muestra)), pasosDetectados[1:length(muestra)],
col = "red") # Debúxanse as liñas correspondentes ós pasos detectados,
almancenados en pasosDetectados
       abline(limiar, 0, col = "navy") # Debúxase liña horizontal no valor
limiar a partir do cal unha mostra é considerada un paso
       dev.off()

      # Almacenamos resultados
      er[j]=round(abs((pasosReais-contadorPasos))*100/pasosReais)
         resultados[j,]=c(fichero, pasosReais, contadorPasos, pasosReais-
contadorPasos, er[j])
```

```
    }

    ii <- ii+1
    mean_er=round(mean(er),digits=1) # Error medio para esta combinación de
parámetros de limiar e retardo2
     std_er=round(sd(er),digits=2) # Desviación estándar do error para esta
combinación de parámetros de limiar e retardo2

      error[ii,]=c(limiar,retardo2,mean_er,std_er) # Gardámolo nun vector
global

    outfile = paste("Resultados_", limiar, "_", retardo2, ".csv",sep="")
    write.csv2(resultados, outfile) # Almacenamos os resultados obtidos nun
.csv
  }
}

# Finalizado o proceso, gardamos en "errores,csv" todos os valores de error
medio, para todas as combinacións probadas
write.csv(error,"errores.csv")
```