Universidad Zaragoza
1542

Trabajo Fin de Grado

Anexo Técnico

Adrián Rosales Serrano

Director/es

José Luis Villarroel Salcedo

ESCUELA DE INGENIERIA Y ARQUITECTURA

2017

Tabla de contenido

1. Diseño Hardware. Sensor de aceleración.

Como ya se ha comentado, para resolver el problema de la cadencia de la embarcación, se ha decidido utilizar un acelerómetro. Debe registrar la aceleración a la que es sometida a la altura de las pedalinas.

El centro de masas se encuentra, aproximadamente, en el punto medio del bote, pero al tratarse de un movimiento longitudinal y suponiendo que la aceleración es homogénea en toda la embarcación, se ha decidido registrar la aceleración en las pedalinas.

De este modo evitará instalaciones innecesarias en el bote y hará que la pantalla sea visible para el deportista.

Los requerimientos del proyecto dicen que la función del acelerómetro ha de ser detectar la cadencia del bote y nada más.

Las altas frecuencias son despreciables ya que se quiere obtener el periodo de una señal pseudoperiódica cuya frecuencia fundamental oscila entre los 0,25Hz y los 0,6Hz.

Para obtener una señal relativamente fiel a la real y que no la convierta en una sinusoide se tendrá que tener en cuenta la inclusión de las componentes frecuenciales con una frecuencia superior.

Por este motivo en el acelerómetro que se necesita el ancho de banda de éste, no será un problema, en todo caso será algo a solucionar debido al posible ruido de alta frecuencia que pueda introducir.

La sensibilidad del acelerómetro que se necesita será relativamente baja. Al tratarse de un dispositivo que se desliza en el agua, no vamos a encontrarnos con cambios excesivamente bruscos debido a la naturaleza del movimiento a medir, al contrario de lo que podría pasar en un movimiento en el que hubiese detenciones en seco o impactos y rebotes.

Los valores de aceleración rara vez superan el rango de ± 1g, solamente en casos de mala ejecución de la técnica empleada en la recuperación, hará obtener valores menores de -1g, pero será del todo irrelevante conocer la magnitud exacta.

La aplicación trata más de un reconocimiento de forma de onda, mediante la determinación del periodo que de una reconstrucción o una obtención de información de dicha señal. Por este motivo se tendrá que elegir un acelerómetro que mida aceleraciones dentro del rango de ±1g.

El método de adquisición de datos tiene que permitir elegir la frecuencia de muestreo fácilmente, ya que, al tratarse de una señal tan lenta, no tendría sentido utilizar las frecuencias de muestreo típicas de las aplicaciones que leen aceleraciones, es decir, tendremos que determinar una frecuencia de muestreo óptima para nuestro propósito, que es calcular de la manera más eficientemente posible la cadencia de la embarcación.

Los valores de alimentación del acelerómetro son poco relevantes, aunque a la hora de realizar pruebas y por compatibilidades con el resto del proyecto se valorará la posibilidad de que sea conectable a 5V.

El gasto energético será un elemento a tener en cuenta debido al alto consumo del *GPS* y la pantalla *LCD* en comparación con el resto de elementos del proyecto.

De las tecnologías disponibles en sensores de aceleración, se vio que los más comunes eran los de:

- Tecnología capacitiva
- Tecnología piezoeléctrica.

Estos últimos se descartaron debido a que están diseñados para medir valores de aceleración elevados y a unas frecuencias muy altas.

Por el contrario, los acelerómetros basados en tecnología capacitiva están preparados para medir aceleraciones de menor frecuencia y de rangos muchísimo menores.

El método por el cual vamos a leer esos valores de aceleración es la principal decisión de diseño para el proyecto. Las distintas opciones eran:

- Acelerómetro ADXL202 que muestra unos valores de aceleración codificados en una señal *PWM*, de tal manera que nosotros tendríamos que contar los tiempos entre flancos de subida y bajada (saber el tiempo en alto) y calcular el valor de aceleración.
- Acelerómetro ADXL345 que genera interrupciones y transmite la información a través de una interfaz de comunicación serie.
- Acelerómetro MMA7361 que muestra una tensión proporcional a la aceleración y se leerá en un ADC externo o del propio microcontrolador.

Primero se estudió el uso del acelerómetro ADXL202 que medía la aceleración en 2 ejes y permitía añadir unos condensadores para el filtrado de la entrada.

Una vez filtrada la tensión que mostraba, digitalizaba los valores obtenidos y a través de "dutty cycle modulator", codificaba la información con una señal *PWM* de periodo variable mediante una resistencia conectada a la patilla Rset.

Inicialmente se utilizó este acelerómetro. Para su gestión se implementaron interrupciones en el microcontrolador, éstas estaban asociadas a los flancos de subida y bajada, para posteriormente matemáticamente obtener el "dutty" de la señal. Para obtener el valor de aceleración se utilizó la formula proporcionada por el fabricante.

Este acelerómetro se descartó porque para la posible modificación del periodo de muestreo, sería necesario modificar el hardware de la etapa, además no disponía de modo de bajo consumo. También existía la limitación de que había que realizar la medición a través de los flancos, por ello se pensó que sería mejor utilizar otro interfaz de lectura que me permitiese medir la aceleración en el momento que quisiera, sin hacer estas mediciones rigurosamente (sin saltar ninguna).

El segundo acelerómetro que se barajó fue el ADXL345, que a priori parecía el óptimo, ya que tenía integrada una interfaz de comunicación serie para pasar los valores de aceleración periódicamente. Disponía de dos patillas de interrupción que saltaban con la detección de máximos o mínimos, pero dada la relevancia que quería que tuviese el proyecto, se descartó ya que prefería utilizar un acelerómetro más barato y que me permitiese el desarrollo del método óptimo de detección de cadencia en base a muestras tomadas en la embarcación. Llegamos a la conclusión de que dicho acelerómetro eliminaría una parte muy importante del trabajo a realizar, que estaba muy interesado en desarrollar.

El acelerómetro elegido finalmente es el módulo MMA7361 que muestra

- Tensión proporcional a la aceleración
- Dispone de un filtro de orden 1 con frecuencia de corte de 1,5Khz para filtrar el ruido de CLK
- Mide en los 3 ejes espaciales
- Sensor de caída libre
- Modo de bajo consumo.

Se ha elegido por su simplicidad su modularidad y su precio.

2. Diseño Hardware. Unidad lógica

Para la elección del microcontrolador, se necesitaba un micro que permitiese testear cómodamente en placa blanca y de una gran comunidad trabajando con él para facilitar lo máximo posible la programación del prototipo. La idea principal era trabajar con microcontroladores con los que se había trabajado anteriormente de forma que ya estuviésemos familiarizados con ellos.

Dada la naturaleza del problema y estimando la complejidad de las tareas a realizar, se llegó a la conclusión de que la arquitectura del microcontrolador debería de ser de 8 bits. Ya que un microcontrolador de una arquitectura superior sería del todo innecesario.

Inicialmente se barajó la posibilidad de usar el microcontrolador utilizado en las prácticas de la asignatura de Sistemas Electrónicos programables.

Una de las dificultades que encontré fue que el espacio entre patillas no era el que tienen las placas de montaje rápido. Dada la necesidad de hacer pruebas con el microcontrolador montado fuera de su placa de programación decidí recurrir al microcontrolador **atmega328p**, principalmente porque era un microcontrolador que ya había utilizado previamente y encajaba en las "placas de montaje rápido".

Gracias a que dispone de una amplia comunidad, que había creado y hacía uso de librerías me facilitó enormemente la tarea de programación.

Destacando las siguientes ventajas:

- Es un micro que se puede alimentar a 5V que dado la modularidad es muy conveniente en mi proyecto.
- Dispone de patillas con *ADC* que me permiten leer los valores de tensión que muestra mi acelerómetro.
- Tiene suficientes puertos digitales para controlar los botones y el display que mostrará las variables necesarias para el entrenamiento.
- Dispone de patillas de interrupción digital para los pulsadores.
- Tiene un *UART* que se utilizará para leer las tramas del módulo *GPS*.

3. Diseño Hardware. GNSS

Para la elección del *GPS* se necesitaba un módulo que facilitara la lectura de las tramas obtenidas, puesto que la aplicación tiene unos tiempos de respuesta lentos, se optó por la simplicidad.

Los errores de precisión no inciden de una manera relevante en este trabajo debido a que nos interesa medir un desplazamiento "lineal" a lo largo del tiempo y no necesitábamos precisión milimétrica para el desarrollo de la aplicación, es decir, sabiendo que describo una trayectoria lineal, nos es indiferente que dicha trayectoria esté desplazada en cualquiera de los ejes; ya que el dato relevante es la longitud de esa trayectoria y no los puntos exactos por los que transcurre.

En cuanto a la dispersión de nuestro error, tampoco nos afecta notablemente ya que los valores de velocidad en los que nos movemos hacen que porcentualmente hablando el error no suponga más de +- 2,5m.

Inicialmente se utilizó un módulo *GPS* (Fastrax IT500) de un proyecto anterior con el que él había trabajado, pero al examinarlo, nos dimos cuenta de que su tamaño (el módulo tenía una antena externa muy aparatosa que habría sido difícilmente integrable en mi dispositivo) y su coste (70€) no se ajustaba a las necesidades del proyecto. y teniendo en cuenta que uno de los principales requisitos de mi proyecto era el precio del producto final, decidimos continuar la búsqueda de un módulo que se ajustase más a mis necesidades.

Se buscó módulos *GPS* con una interfaz serie sencilla, que tuvieran antena integrada, debido a que el uso del dispositivo se iba a dar siempre en entornos al aire libre y que además me permitiese la rápida conexión con un ordenador y en una placa de montaje rápido para facilitarme la etapa de configuración y puesta a punto.

Tras la búsqueda, el dispositivo elegido fue un módulo *GPS* de U-Block que integra el *GPS* NEO-6M-000-1.

- Se trata de un módulo que dispone de 4 patillas: *VCC*, *GND* Rx y Tx.
- Su consumo es sólo de 50 *mA*.
- Para su conexión al ordenador únicamente se necesita un cable de TTL *UART* 5V.
    - descargar desde su página web un programa llamado u-centre que permite la configuración del módulo mediante una interfaz visual y te permite seleccionar todos los parámetros necesarios como, por ejemplo:
        - Las tramas *GPS* que envía.
        - El tipo de arranque que usa el *GPS*.
        - La frecuencia de las tramas.
        - El modo de funcionamiento.

Por todos estos motivos y principalmente por su precio, ha sido el módulo *GPS* elegido para el proyecto.

4. Diseño detallado. Sensor de aceleración. Toma de datos.

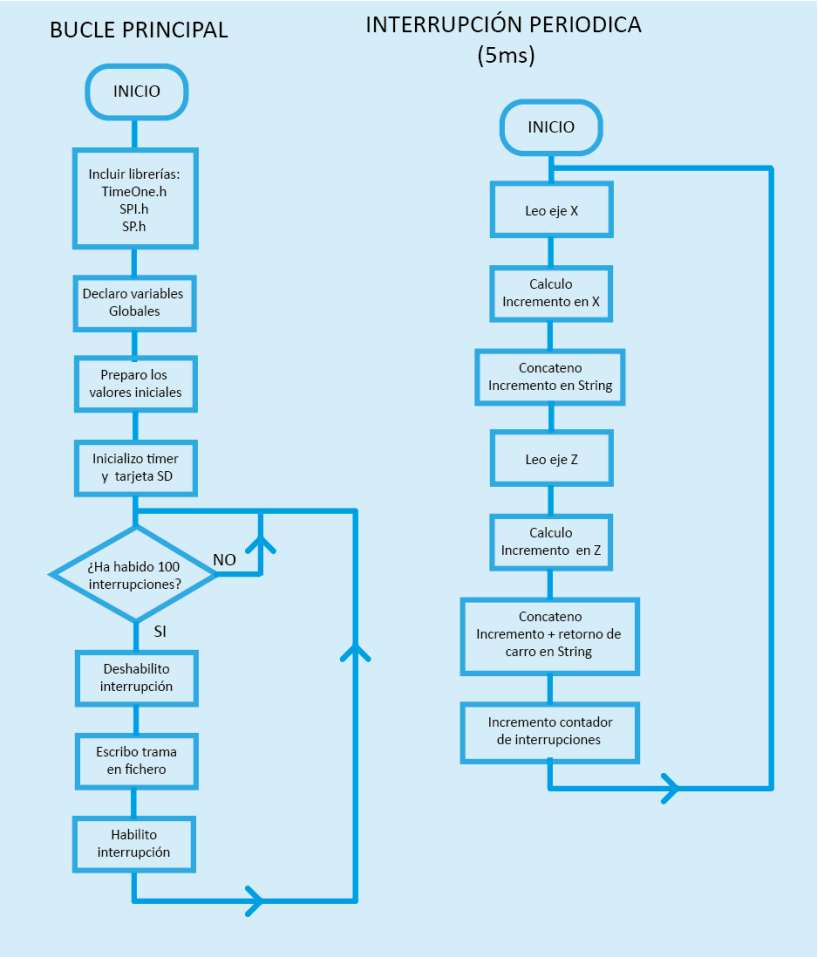
Respecto a las limitaciones de la tarjeta SD. su principal inconveniente es que para realizar un acceso a un fichero y escribir los datos almacenados en el buffer, requiere de demasiado tiempo (unos 14 milisegundos). Este problema podría ser resuelto guardando una gran cantidad de datos en un buffer y realizar el volcado cada mucho tiempo, de forma que existiese ese retraso de 14 milisegundos cada muchas muestras tomadas, pero el tamaño del buffer es de 512 bytes, por lo que me alcanza para almacenar unas 60 muestras.

Debido a esto se ha barajado la posibilidad de codificar las muestras tomadas de forma que escribamos en la SD únicamente el incremento de la aceleración en vez del valor de la muestra, pero como he comentado antes, el problema de la tarjeta SD es de tiempo de acceso a fichero, no de tiempo de escritura de la trama, por esto, codificando la información lo único que conseguiríamos sería aumentar el número de valores tomados entre cada desfase de escritura de forma que cada muestra en lugar de ocupar 12 bytes debido a que la trama es de esta forma: xxx/tyyy/r/t y de esta forma codificada pasa a x/ty/r/t. de esta forma aumentaríamos el tamaño de la ristra de datos tomados un 50%, a pesar de que el problema seguiría estando pero menos frecuentemente. De esta forma tomaríamos 100 muestras a 200 Hz y sufriríamos el retraso de escritura de 14 ms cada 500 ms.

Por tanto, se optó por tomar las muestras de forma codificada siendo conscientes de que cada vez que se escribe la trama almacenada se pierden 2 tomas de datos.

Quedando de esta forma el diagrama de flujo de la tarea encargada de esto:

DIAGRAMA DE FLUJO
TOMA DE DATOS



5. Diseño detallado. Sensor de aceleración. Tratado de datos.

Para el diseño del filtro paso bajo:

En **el filtro paso bajo** de primer orden que aplicamos a la muestra, se partió haciendo un diseño del filtro continuo:

$$H(s) = \frac{1}{1+\tau s} \; y \; sabiendo \; que \; s = j2\pi F \; \rightarrow \quad H(F) = \frac{Y(F)}{X(F)} = \frac{1}{1+\tau j2\pi F}$$

Al tratarse de una función compleja, hallaremos su módulo

$$|H(F)| = \frac{1}{\sqrt{1+(\tau j2\pi F)^2}} \; donde \; podemos \; comprobar:$$

$$Si \; F = 0 \; \rightarrow H(F) = 1 \; y \; Si \; F = \infty \; \rightarrow H(F) = 0$$

Que efectivamente se trata de un **filtro paso bajo**. Por tanto, igualando al valor que tomará en su frecuencia de corte y despejando τ.

$$\frac{1}{\sqrt{2}} = \frac{1}{\sqrt{1+(\tau j2\pi F_c)^2}} \; \rightarrow \; \tau = \frac{1}{2\pi F_c}$$

Como ya se explicado antes, al realizar el filtro con F_c=0.6Hz, el ruido de la señal seguía estando presente, y teniendo en cuenta que los armónicos de una frecuencia superior a esa frecuencia nunca son armónicos relevantes y tienen una amplitud (potencia) mayor, puedo permitirme ajustar más la frecuencia de corte de mi filtro y aunque no se encuentren en la banda de paso (se van a ver atenuadas) me servirá para:

- Igualar la potencia con las frecuencias más bajas.
- Quitar más ruido sin recurrir a un filtro de orden superior.

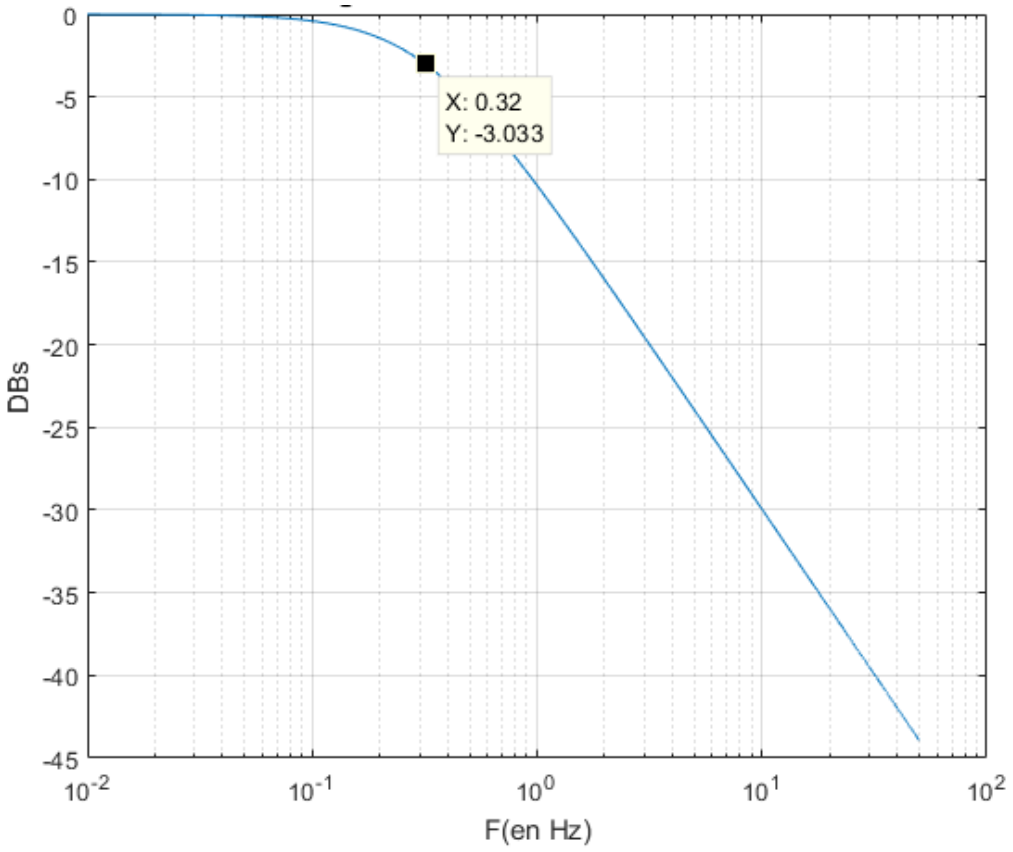
Probando distintos filtros el que daba una señal más limpia, ha sido el de una F_c=0.3Hz, que sustituyendo en la fórmula (), me daba una $\tau \approx 0.5$. De este modo, discretizando el filtro, se calculó la función de transferencia del filtro en tiempo discreto:

$$G(s) = \frac{1}{1 + 0.5s} \;\rightarrow\; con\, T = 10ms \;\rightarrow\; G(z) = \frac{0.2}{z - 0.98}$$

Del que despejando Y_k:

$$Y_k = 0.02X_{k-1} + 0.98Y_{k-1}$$

En la siguiente figura vamos a ver la respuesta en frecuencia del filtro implementado, en la que he marcado aproximadamente la frecuencia de corte:



Tras este desarrollo, surge otro problema, el coste computacional del filtrado. Para cada uno de los valores (100 por segundo), se deben realizar 2 multiplicaciones, una por 0.02 y otra por 0.98. además, una suma de los dos valores obtenidos.

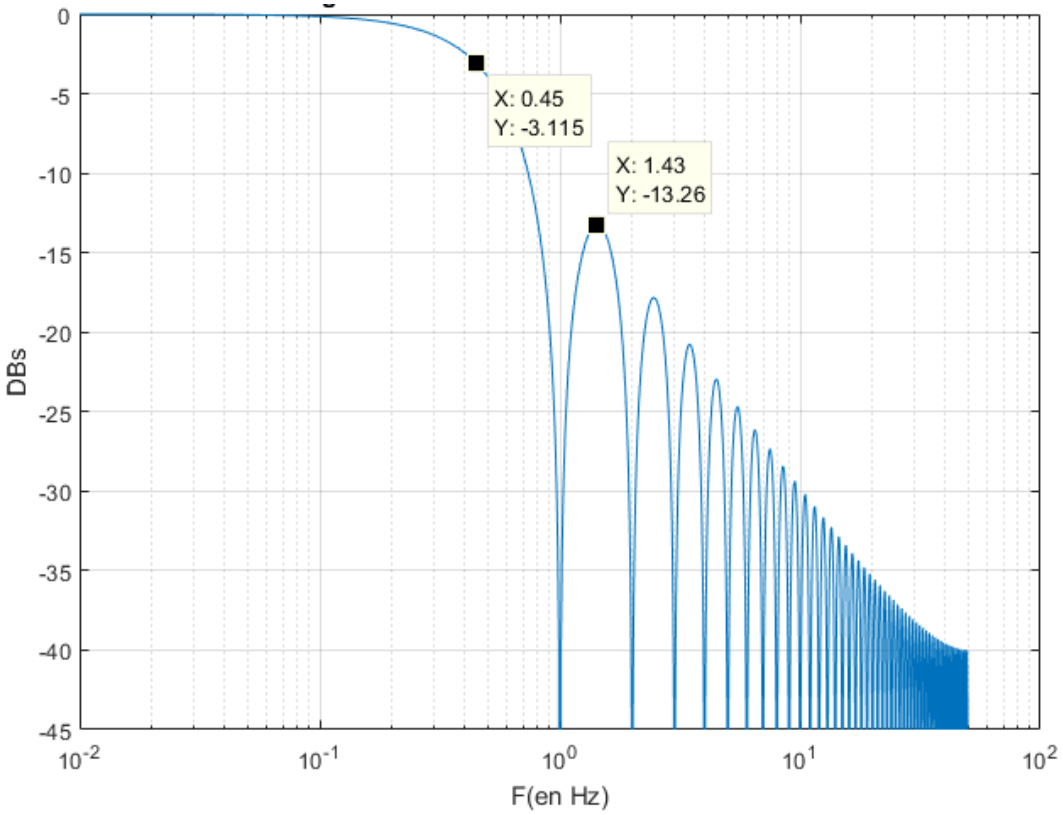
Para el diseño del filtro de media móvil:

La implementación de este filtro consiste en que cada uno de los valores, es la suma ponderada de los últimos n valores de forma que cuanto mayor sea n, menor es la frecuencia de corte.

Probando distintos valores de media, se determinó que con el valor que más parecida quedaba la muestra al filtro paso bajo, era con 100. De esta forma, para cada valor nuevo, el filtro hacía la media ponderada junto con los 99 valores anteriores.

Esta forma de ejecución tiene un problema de exceso de operaciones. Pero deduje que, para cada valor, en lugar de cada iteración del filtro calcular el valor medio, variaremos el valor de la suma, utilizando una variable que almacene el sumatorio de los últimos 100 valores, de forma que quitando el primer elemento del sumatorio y añadiendo en su lugar el nuevo valor, se evitará realizar el sumatorio constantemente. Al tratarse de números en base diez y siendo el valor a ponderar 0.01, conseguiremos el valor medio corriendo la coma a la izquierda dos posiciones de la variable suma de forma fácil.

En la siguiente figura veremos el diagrama de bode de la respuesta en frecuencia del filtro de media móvil:



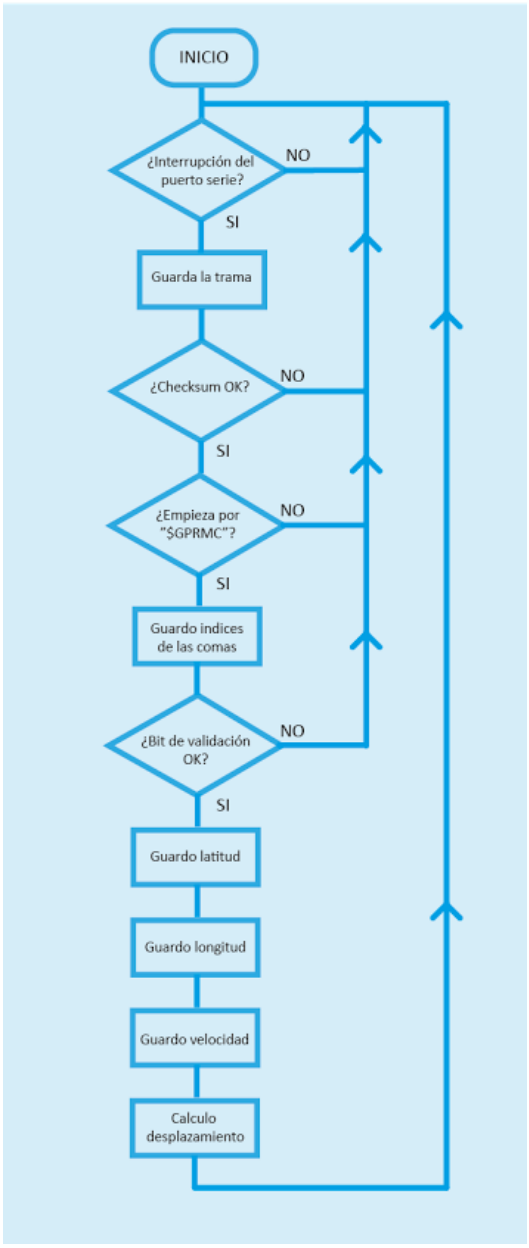
D

Como podemos observar, su frecuencia de corte queda más cercano al punto teórico establecido anteriormente en este apartado. Inicialmente, el tamaño de los lóbulos que se ven en su bode puede parecer excesivo, por ello, comprobé que una posibilidad de este filtro consiste en aplicarlo consecutivamente varias veces, de forma que la banda de paso se estrechaba y el tamaño de los lóbulos disminuía considerablemente con cada iteración.

6. Diseño detallado. GPS. Experimentación previa

En este apartado se puede ver el diagrama de bloques de la tarea del GPS.

DIAGRAMA DE FLUJO
GPS



7. Futuro del prototipo. QFD para garantizar la calidad.

Existen actualmente algunos modelos de cuenta paladas en el mercado que van desde los muy sofisticados y precisos a otros que incluso pueden ser descargados en un Smartphone. El producto presentado en este proyecto pretende satisfacer a un consumidor que disfrute de la actividad del remo a modo de pasatiempo y no de una forma profesional: un aparato que contabilice las variables necesarias para definir un entrenamiento (número de paladas efectuadas, tiempo a los 500 m etc.), suficientemente preciso y con un precio asequible. Con estas especificaciones se procedió a elaborar un prototipo que las solucionara lo mejor posible.

Tras su fabricación, se quiso estudiar la aceptación de éste entre los clientes objetivo consultándoles qué características valorarían más en un dispositivo de este tipo y qué les parecía el propuesto. También se les preguntó por otros dispositivos del mercado que clasificaremos como “Gama alta” y “Gama baja” atendiendo a características como su resolución de palada, la duración de su batería o la precisión de su GPS.

Con vistas en una futura comercialización y para asegurar la calidad del diseño y del desarrollo de nuevos productos mejorados derivados de éste, realizamos un QFD (despliegue funcional de la calidad) a partir de las opiniones recogidas entre el equipo de remo del Club Deportivo Helios.

El QFD es una técnica que se emplea en el ámbito industrial cuya finalidad es el aseguramiento de la calidad del nuevo producto permitiendo al cliente tomar parte en este proceso proporcionando su opinión. Al final de su realización podrá determinarse qué características de este son más valoradas por el cliente y, por tanto, las que deberán mejorarse (si el producto no las tenía de antemano).

El QFD comienza con la elaboración de la matriz “Qués frente a Cómos” en la que nos plantearemos qué características son relevantes en un cuentapaladas y cómo se cuantifican. La matriz de la derecha determina la importancia que el mercado da a cada “Qué”, la de debajo la evaluación técnica del producto y de la competencia y la de arriba (el tejado) confronta las características según su relación entre ellas.

La matriz QFD se presenta a continuación:

	1	2	3	4	5	6	7	8	9	10	11
11											
10											
9											3
8											
7											
6								-3	3		
5						-9					
4										3	
3					9	9		9	-3		9
2			9					9	-3		
1			9					3	-9	9	-3

¿QUÉ's?	¿CÓMO's?	ACCESORIOS REQUERIDOS	CAPACIDAD DE MEMORIA	PRECIO	PALADAS POR MINUTO	SEGUNDOS	TIEMPO DE DURACIÓN	METROS	PLATAFORMAS DISPONIBLES	SENCILLEZ INTERFAZ	TIEMPO DE INSTALACIÓN	TECNOLOGÍA ADAPTABLE	Importancia	Valoracion del mercado	Nuestro producto	Competencia X (gama alta)	Competencia Y (gama baja)		Objetivo	Ratio de mejora	Argumento de venta	Ponderacion absoluta	Ponderacion relativa	Orden de importancia
		1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	4	8	9	10	11	12
INTERCAMBIABLE ENTRE BOTES	1	9		9			1				9	3	5	3	5	5	1		5	1	1,5	7,5	0,1	2
MEMORIA PARA ALMACENAR DATOS	2		9	3			1		9	9			1	4	1	5	1		1	1	1	1	0	6
ECONOMICO	3	9	9	9	3		9	9	9	9	9	9	5	5	5	1	3		5	1	1,5	7,5	0,1	2
RESOLUCIÓN DE PALADA	4			3	9					1			3	3	3	5	4		3	1	1	3	0	5
RESOLUCIÓN DEL CRONOMETRO	5					9	1						3	3	3	5	3		4	1,3	1	4	0,1	4
DURACIÓN BATERÍAS	6	1	3	9		3	9		3	1			4	5	2	5	5		5	2,5	1,5	15	0,2	1
PRECISION DEL GPS	7			9			3	9				3	3	4	2	5	1		3	1,5	1	4,5	0,1	3
COMPATIBILIDAD CON DISPOSITIVOS	8	1	3	9			1		9	3			1	1	1	5	1		1	1	1	1	0	6
FACILIDAD DE USO	9	3	9	1					3	9	3		5	4	5	2	5		5	1	1,5	8	0,1	2
FACILIDAD DE INSTALACIÓN	10	9		3							9		5	4	5	3	1		5	1	1,5	7,5	0,1	2
ADAPTABILIDAD PARA MINUSVALIDOS	11	1		9							1	9	5	1	5	5	1		5	1	1,5	7,5	0,1	2
Orientación deseada	1	**D**	**D**	**D**	**U**	**U**	**U**	**U**	**D**	**U**	**D**	**U**												
Ponderación absoluta	2	3,8	2,9	6,5	0,8	1,2	3,5	1,6	2,3	2,5	3,5	2,6												
Ponderación relativa	3	0,1	0,1	0,2	0	0	0,1	0,1	0,1	0,1	0,1	0,1												
Orden de importancia	4	2	5	1	11	10	4	9	8	7	3	6												
Valoración técnica	5	3	4	5	3	3	5	4	1	4	4	1												
Nuestro producto	6	5	1	5	3	3	2	2	1	5	5	5												
Competencia X	7	5	5	1	5	5	5	5	5	2	3	5												
Competencia Y	8	1	1	3	4	3	5	1	1	5	1	1												
Objetivo técnico	9	5	1	5	3	4	5	3	1	5	5	5												

U = Flecha hacia arriva

D= Flecha hacia abajo

CONCLUSIONES:

- El punto que más tenemos que mejorar es la duración de las baterías ya que es nuestro punto más débil y dista bastante de la calidad disponible en la competencia. Se es consciente de que al tratarse de un prototipo modular y haberse utilizado componentes incrustados en módulos que tienen leds o componentes innecesarios, ha aumentado el consumo significativamente.
- Aunque no son de las características a mejorar que más prioritarias, convendría en un futuro contemplar la mejora de la resolución del cronómetro y el GPS para equipararnos más a los de nuestra competencia, pero sin aumentar el precio, que es nuestro punto distintivo frente a los demás modelos.
- La evaluación técnica realizada del producto coincide con la evaluación del mercado por lo que no ha supuesto diferencia en el resultado final.

Universidad Zaragoza
1542

Trabajo Fin de Grado

Anexo de Código

Adrián Rosales Serrano

Director/es

José Luis Villarroel Salcedo

ESCUELA DE INGENIERIA Y ARQUITECTURA

2017

Tabla de contenido

1. Código de la toma de datos

```
//*****************************
//INCLUIMOS LIBRERÍAS NECESRIAS
//*****************************
#include <TimerOne.h>
#include <SPI.h>
#include <SD.h>
//************************
//DECLARAMOS LAS VARIABLES
//************************
const int chipSelect = 4;
long counter = 1;
String dataString = "";
int sensorX = 0;
int sensorZ = 0;
int sensorXant = 0;
int sensorZant = 0;
int incrementoX = 0;
int incrementoZ = 0;

//**************
//INICIALIZACIÓN
//**************
void setup(void)
{
  sensorX = analogRead(0);                                  //Leo valor inicial del eje X
  dataString += String(sensorX);                            //Lo concateno al string
  dataString += "\t";                                       //Concateno un tabulador al string
  sensorZ = analogRead(1);                                  //Leo valor inicial del eje X
  dataString += String(sensorZ);                            //Lo concateno al string
  dataString += "\r\n";                                     //Concateno un retorno de carro al
string

  if (!SD.begin(chipSelect)) {                              //Inicializo tarjeta SD
    return;
  }

  Timer1.initialize(5000);                                  // configuro interrupción periodica
cada 5 ms (200Hz)
  Timer1.attachInterrupt(ISR_Data);                         // Activa la interrupcion y la asocia
a ISR_Data
}

//*************************
//INTERRUPCIÓN ACELERÓMETRO
//*************************
void ISR_Data()
{
  sensorXant = sensorX;                                      //Asigno valor anterior del eje X
  sensorX = analogRead(0);                                   //Leo el eje X
  incrementoX = sensorX - sensorXant;                        //Calculo el incremento del eje X
  dataString += String(incrementoX);                         //Lo concateno al string
  dataString += "\t";                                        //Concateno un tabulador al string

  sensorZant = sensorZ;                                      //Asigno valor anterior del eje Z
  sensorZ = analogRead(1);                                   //Leo el eje Z
  incrementoZ = sensorZ - sensorZant;                        //Calculo el incremento del eje X
  dataString += String(incrementoZ);                         //Lo concateno al string

  dataString += "\r\n";                                      //Concateno un retorno de carro al
string
  counter++;                                                 //Incremento contador
}

//***************
//BUCLE PRINCIPAL
//***************
void loop(void){
  if (counter > 100 ) {                                      //Si ha habido 100 interrupciones
    Timer1.stop();                                           //Detengo la interrupción periódica
    File dataFile = SD.open("datalog.txt", FILE_WRITE);      //Accedo al fichero
    if (dataFile) {
      dataFile.print(dataString);                            //Escribo el String
      dataFile.close();
    }
    else {

    }
    dataString = "";                                         //Reinicializo la interrupción
    counter = 1;
    Timer1.resume();
  }
}
```

2. Código del filtrado de la señal.

```
%Declaro variables muestra, frecuencia de muestreo, periodo, tamaño y vector t.
%A = A';
Fs = 100;
T = 1/Fs;
L = 105000;
t = 0:0.01:1049.99;

%FILTRO VENTANA
windowSize = 100;
b = (1/windowSize)*ones(1,windowSize);
a = 1;
B = filter(b,a,A);

%FIGURA 1 FILTRO VENTANA
%Muestro la señal y la señal filtrada por filtro de ventana
figure(1);
subplot(1,2,1);
plot(t,A); ylim([450 1000]);xlim([0 1050]);
hold on
plot(t,B)
title('Filtro de ventana 100')
xlabel('t (segundos)')
ylabel('Valor acelerometro)')
legend('Input Data','Filtered Data')
%Calculo FFT de la señal y la señal filtrada.
Y = fft(A);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;

figure(1);
subplot(1,2,2);
title('Especro muestra')
xlabel('f (Hz)')
ylabel('|P1(f)|')

Ys = fft(B);
P2s = abs(Ys/L);
P1s = P2s(1:L/2+1);
P1s(2:end-1) = 2*P1s(2:end-1);
fs = Fs*(0:(L/2))/L;
plot(f,P1);ylim([0 8]);xlim([0 50]);
hold on
plot(fs,P1s);ylim([0 8]);xlim([0 50]);legend('Input Data','Filtered Data');
title('Espectro de la muestra')
xlabel('f (Hz)')
ylabel('|P1(f)|')


%FIGURA 2 ESPECTRO DE LA MUESTRA
% figure(2);
% plot(f,P1);ylim([0 8]);xlim([0 50]);
% title('Espectro de la muestra')
% xlabel('f (Hz)')
% ylabel('|P1(f)|')


%APLICO FILTRO PASO BAJO A LA SEÑAL
A1= [0];
tic;
for c = 2:1:105000;

    i=0.02*A(1,c-1)+0.98*A1(1,c-1);
    A1 = [A1 i];
    % t1 = [t1 t(1,c)];
    %L1 =L1+1;
end
toc;

%FIGURA 3
%Muestro la señal y la señal filtrada por filtro paso bajo
figure(3);
subplot(1,2,1);
plot(t,A); ylim([450 1000]);xlim([0 1050]);
hold on
plot(t,A1)
title('FILTRO PASO BAJO fc=0.5Hz')
xlabel('t (segundos)')
ylabel('Valor acelerometro)')
legend('Input Data','Filtered Data')
%junto con sus FFTs
subplot(1,2,2);
```

```
Ys = fft(A1);
P2s = abs(Ys/L);
P1s = P2s(1:L/2+1);
P1s(2:end-1) = 2*P1s(2:end-1);
fs = Fs*(0:(L/2))/L;
plot(f,P1);ylim([0 8]);xlim([0 50]);
hold on
plot(fs,P1s);ylim([0 8]);xlim([0 50]);legend('Input Data','Filtered Data');
title('Espectro de la muestra')
xlabel('f (Hz)')
ylabel('|P1(f)|')

%FILTRO DE MEDIA MÓVIL
tic;
asd = zeros(1,100);
contador = 1;
suma = 0;
AFO = [0];
for c = 2:1:105000;
    suma = suma + A(1,c) - asd(1,contador);
    asd(1,contador) = A(1,c);
    contador = contador + 1;
    if contador == 101
        contador =1;
    end
    AFO = [AFO suma/100];
end
AFO = round(AFO,0);
toc;
figure(2);
plot(t,AFO);ylim([450 1000]);xlim([0 1050]);
```

3. Código de la representación del diagrama de Bode de los filtros.

```
%diagrama de bode de filtro analogico paso bajo FC 0,32
%la respues en fase no me interesa por que es lineal en la banda de paso y
%puesto que las frecuencis mayores que son relevantes estan cercanas a la
%fc, su desfase será minimo.
tau = 0.5;
F=0:0.01:50;
H_F_1= 1./(1+tau*2*pi*F*j);
dB_1 = mag2db(abs(H_F_1));
%FIGURA 1
%Diagrama de bode del filtro paso bajo.
figure(1);
semilogx(F,dB_1),grid on;
title('Diagrama de bode del filtro Fc=0.3Hz')
xlabel('F(en Hz)')
ylabel('DBs')
%diagrama de bode de filtro media móvil, muestras a 100 HZ y tamaño ventana = 100
F_m=100;
M=100;
f=0:0.0001:0.5;
H_f=(sin(pi*f*M))./(M*sin(pi*f));
F=f*F_m

% figure(2);
% plot(F,abs(H_f));

% Transformación a frecuencias analógicas
dB_2 = mag2db(abs(H_f));
%FIGURA 3
%Diagrama de bode del filtro de media móvil
figure(3);
semilogx(F,dB_2),grid on,ylim([-45 0])
title('Diagrama de bode del filtro de media móvil M=100')
xlabel('F(en Hz)')
ylabel('DBs')

%FIGURA 4
%Comparación de los diagramas de bode de los dos filtros.
figure(4);
semilogx(F,dB_1),grid on;
hold on;
semilogx(F,dB_2),grid on,ylim([-45 0])
title('Comparación diagramas bode')
xlabel('F(en Hz)')
ylabel('DBs')
legend('low pass filter','window average filter')
```

4. Código del algoritmo de detección de máximos y mínimos.

```
%AFO = AFO';
t = 0:0.01:1049.99;
L=0;
AFO1 = 0;
t1 = 0;
max = 0;
min = 0;
tmax = 0;
tmin = 0;
tic;
%EMPECEMOS POR SUBMUESTREAR LA SEÑAL.
for c = 1:20:105000;
    AFO1 = [AFO1 AFO(1,c)];
    t1 = [t1 t(1,c)];
    L=L+1;
end
%busquemos máximos y mínimos.
ultimoesmaximo = false;
valmax=0;
valmin=0;

for c = 7:1:L-6;
    %DETECTO MAXIMOS
    if ultimoesmaximo == false %Si lo último detectado es un mínimo.
        if AFO1(1,c+4)<AFO1(1,c-1)%Si la señal vale menos, 5 muestras más tarde
            if AFO1(1,c-2) < AFO1(1,c-1)               %Si tiene esta forma   .
                if AFO1(1,c-1) > AFO1(1,c)             %                    . .
                    if AFO1(1,c-1)-valmin >4 %Dista más de 4 unidades del mínimo anterior
                        valmax = AFO1(1,c-1);%ES UN MAXIMO
                        max = [max AFO1(1,c-1)];
                        tmax = [tmax t1(1,c-1)];

                        ultimoesmaximo = true;
                    end
                end
            end
            if AFO1(1,c-3)< AFO1(1,c-2)                %Si tiene esta forma   ..
                if AFO1(1,c-2)== AFO1(1,c-1)           %                     . .
                    if AFO1(1,c-1)> AFO1(1,c)
                        if AFO1(1,c-1)-valmin >4 %Dista más de 4 unidades del mínimo anterior
                            valmax = AFO1(1,c-1); %ES UN MAXIMO
                            max = [max AFO1(1,c-1)];
                            tmax = [tmax t1(1,c-1)];

                            ultimoesmaximo = true;
                        end
                    end
                end
            end
            if AFO1(1,c-4)< AFO1(1,c-3)                %Si tiene esta forma   ...
                if AFO1(1,c-3)== AFO1(1,c-2)           %                     .   .
                    if AFO1(1,c-2)== AFO1(1,c-1)
                        if AFO1(1,c-1)> AFO1(1,c)
                            if AFO1(1,c-1)-valmin >4%Dista más de 4 unidades del mínimo anterior
                                valmax = AFO1(1,c-1); %ES UN MAXIMO
                                max = [max AFO1(1,c-1)];
                                tmax = [tmax t1(1,c-1)];

                                ultimoesmaximo = true;
                            end
                        end
                    end
                end
            end
        end
    end

    %DETECTO MINIMOS
    %Lógica inversa a los máximos para detección de mínimos.
    if ultimoesmaximo == true
        if AFO1(1,c-6)>AFO1(1,c-1)
            if AFO1(1,c-2) > AFO1(1,c-1)
                if AFO1(1,c-1) < AFO1(1,c)
                    if valmax-AFO1(1,c-1)>4
                        valmin = AFO1(1,c-1);
                        min = [min AFO1(1,c-1)];
                        tmin = [tmin t1(1,c-1)];
                        ultimoesmaximo = false;
                    end
                end
            end
            if AFO1(1,c-3)> AFO1(1,c-2)
                if AFO1(1,c-2)== AFO1(1,c-1)
                    if AFO1(1,c-1)< AFO1(1,c)
```

```
                        if valmax-AFO1(1,c-1)>4
                            valmin = AFO1(1,c-1);
                            min = [min AFO1(1,c-1)];
                            tmin = [tmin t1(1,c-1)];
                            ultimoesmaximo = false;
                        end
                    end
                end
            end
            if AFO1(1,c-4)> AFO1(1,c-3)
                if AFO1(1,c-3)== AFO1(1,c-2)
                    if AFO1(1,c-2)== AFO1(1,c-1)
                        if AFO1(1,c-1)< AFO1(1,c)
                            if valmax-AFO1(1,c-1)>4
                                valmin = AFO1(1,c-1);
                                min = [min AFO1(1,c-1)];
                                tmin = [tmin t1(1,c-1)];
                                ultimoesmaximo = false;
                            end
                        end
                    end
                end
            end
        end
    end


end
cadencia =0;
tcadencia=0;
for c = 5:1:402;
    if 60/(tmax(1,c)-tmax(1,c-1))<15;
        comprobacion = 0;
    else
    comprobacion = 60/((tmax(1,c)-tmax(1,c-2))/2)

    if comprobacion < 15
        comprobacion = 0;
    elseif comprobacion > 40
        comprobacion = 40;
    end
    end
    comprobacion = round(comprobacion,0);
    comprobacion = comprobacion*10;
    cadencia = [cadencia, comprobacion];
    tcadencia = [tcadencia, tmax(1,c)];


end
toc;
figure(1);
plot(t,AFO)
title('Algoritmo de detección de máximos y mínimos')
xlabel('t (segundos)')
ylabel('Valor acelerometro)')

hold on;
plot(t1,AFO1)
hold on;
plot(tmax,max,'o','MarkerEdgeColor','red')
hold on;
plot(tmin,min,'o','MarkerEdgeColor','blue')
hold on;
plot(tcadencia,cadencia)
```

5. Código del algoritmo integrador.

```
%AFO = AFO';
t = 0:0.01:1049.99;
%PRIMERO SUBMUESTREAMOS LA SEÑAL
AFO1=0;
t1=0;
L=0;
SSM=0;
SSSM=0;
t2=0;
max = 0;
tmax = 0;
tic;
for c = 1000:70:105000;
    AFO1 = [AFO1 AFO(1,c)];
    t1 = [t1 t(1,c)];
    L=L+1;
end

%EMPECEMOS QUITANDOLE LA COMPONENTE CONTINUA A LA SEÑAL.
media = mean(AFO1)
SSM=AFO1-media;
for c = 2:1:L-1;
    SSSM=[SSSM SSSM(1,c-1)+(SSM(1,c)*0.1)];
    t2 = [t2 t1(1,c)];
end

%DETECTAMOS MÁXIMOS EN LA INTEGRAL
for c = 3:1:L;
    if AFO1(1,c-2) < AFO1(1,c-1)
        if AFO1(1,c-1) > AFO1(1,c)
            max = [max AFO1(1,c-1)-660.2430];
                        tmax = [tmax t1(1,c-1)];
        end
    end
end
toc;
figure(1);
plot(t1,SSM)
title('Algoritmo integrador')
xlabel('t (segundos)')
ylabel('Valor acelerometro)')
hold on;
plot(t2,SSSM)
hold on;
plot(tmax,max,'o','MarkerEdgeColor','red')
```

6. Código del algoritmo seguidor de tensión.

```
AFO = AFO';
t = 0:0.01:1049.99;
L=0;
AFO1 = 0;
t1 = 0;
max = 0;
tmax = 0;
vseg=0;
valcomp=0;
contador =0;
t_1=0;
t_2=0;
t_3=0;
cadencia=0;
valor =0;
%EMPECEMOS POR SUBMUESTREAR LA MUESTRA.
for c = 1:15:105000;
    AFO1 = [AFO1 AFO(1,c)];
    t1 = [t1 t(1,c)];
    L=L+1;
end
tic;
for c = 1:1:L;
    if AFO1(1,c)>=valcomp  %Si la señal vale más que el valor del comparador
        valcomp=AFO1(1,c);  %le asigno su valor
        vseg=[vseg valcomp];
        contador =0;
    else
        contador = contador +1;
        valcomp = valcomp-1;
        vseg=[vseg valcomp];
        if contador == 4    %si han pasado 5 muestras en las que su valor es inferior
            valmax = AFO1(1,c-contador);    %Asigno el máximo
            max = [max AFO1(1,c-contador)];
            tmax = [tmax t1(1,c-contador)];
            %Calculo cadencia
            valor = round((60/((t1(1,c-contador)-t_3)/3)),0);
            if valor<15
                valor =0;
            end
            cadencia = [cadencia valor];
            t_3=t_2;
            t_2=t_1;
            t_1=t1(1,c-contador);
        end
    end

end
toc;

figure(1);
%plot(t,AFO)
%hold on;
plot(t1,AFO1)
title('Algoritmo seguidor de tensión')
xlabel('t (segundos)')
ylabel('Valor acelerometro)')
hold on;
plot(tmax,max,'o','MarkerEdgeColor','red')
hold on;
plot(t1,vseg)
hold on;
plot(tmax,cadencia*10)
```

7. Código del algoritmo seguidor de tensión en tiempo real.

```
%Algoritmo seguidor de tensión transladado al tiempo real con cálculo de
%cadencia.
%declaramos las variables.
%A = A';
Fs = 100;
T = 1/Fs;
L = 105000;
t = 0:0.01:1049.99;
vseg=0;
suma=0;
valoresVentana = zeros(1,100);
contadorVentana=1;
filteredSignal=0;
contadorAlgoritmo=0;
valcomp=0;
contadorDecrecimiento=0;
cadencia = 0;
tmax_3=0;
tmax_2=0;
tmax_1=0;
x=0;
y=0;
tic;
for c = 1:1:105000;  %para cada interrupción del acelerómetro.
    %Calculamos el sumatorio
    suma = suma + A(1,c) - valoresVentana(1,contadorVentana);
    valoresVentana(1,contadorVentana) = A(1,c);
    contadorVentana = contadorVentana + 1;
    if contadorVentana == 101
        contadorVentana =1;
    end

    contadorAlgoritmo = contadorAlgoritmo+1;


    if contadorAlgoritmo == 15 %cada 15 interrupciones
        %Calculamos valor de la señal filtrada
        filteredSignal = round(suma/100);
        %si la señal vale más que el comparador
        if filteredSignal>=valcomp
            %Le asignamos su valor
            valcomp=filteredSignal
            vseg=[vseg valcomp];
            contadorDecrecimiento =0;
            %Anotamos tiempo de posible máximo
            posibleMaximo=t(1,c);
        else %si no
            %"descargamos" el comparador
            contadorDecrecimiento = contadorDecrecimiento +1;
            valcomp = valcomp-1;
            %si se descarga 4 veces
            if contadorDecrecimiento == 4
                %el posible máximo, era un máximo.
                tmax = posibleMaximo;
                %Calculamos la cadencia con los tiempos de máximos
                %anteriores.
                cadencia = (60/((tmax-tmax_3)/3));
                if cadencia<15
                    cadencia =0;
                end
                x=[x t(1,c-4)];
                y=[y cadencia];
                tmax_3=tmax_2;
                tmax_2=tmax_1;
                tmax_1=tmax;
            end
        end
        contadorAlgoritmo = 0;
    end
end
toc;
%mostramos la cadencia.
plot (x,y);
```

8. Código final del prototipo.

```
//*****************************
//INCLUIMOS LIBRERÍAS NECESRIAS
//*****************************
#include <Tim
#include <LiquidCrystal.h>
//************************
//DECLARAMOS LAS VARIABLES
//************************
int valoresVentana[100];
unsigned long suma = 0;
int contadorVentana  = 0;
int filteredSignal = 0;
int contadorDiezmado = 0;
int valcomp = 0;
int contadorDecrecimiento = 0;
int cadencia = 0;
int dato = 0;
unsigned long posibleMaximo = 0;
unsigned long tmax = 0;
unsigned long tmax_1 = 0;
unsigned long tmax_2 = 0;
unsigned long tmax_3 = 0;



String tiempoGPS;
byte horaTimestamp;
int minutoTimestamp;
int segundoTimestamp;
int hora;
int minuto;
int segundo;
int horaEntrenamiento;
int minutoEntrenamiento;
int segundoEntrenamiento;
long int duracionEntrenamiento;
long int offsetDuracionEntrenamiento;
byte horaMostrar;
byte minutoMostrar;
byte segundoMostrar;

float latitud;
float longitud;
float latitudAnt;
float longitudAnt;
float DLatitud;
float DLongitud;
float DAngulo;
```

```
float DPosicion;
const float pi = 3.14159;
const float RTierra = 3671000;
float distanciaRecorrida;
int distanciaRecorridaEntero;
int t500mNudos;
int tiempo500m[2];

String tiempoDeEntrenamientoLCD;
String paladasPorMinutoLCD;
String tiempoALos500MetrosLCD;
String distanciaRecorridaLCD;

bool primeraEjecucion = true;
bool flagGPS = true;
bool funcionando = false;
bool GPS = false;
LiquidCrystal lcd(10, 8, 7, 6, 5, 4);

//**************
//INICIALIZACIÓN
//**************
void setup(void) {
 pinMode(2, INPUT_PULLUP);
 pinMode(3, INPUT_PULLUP);
 lcd.begin(16, 2);
 lcd.setCursor(0, 0);
      lcd.print("  Pulse boton   ");
      lcd.setCursor(0, 1);
      lcd.print(" para comenzar  ");
 attachInterrupt(digitalPinToInterrupt(2), pulsadorReset, FALLING);          //Interrupción pin digital pulsador Reset
 attachInterrupt(digitalPinToInterrupt(3), pulsadorPausa, FALLING);          //Interrupción pin digital pulsador Reset
 Serial.begin(57600);
}

//*****************************************************************************************************************
//BUCLE PRINCIPAL
//*****************************************************************************************************************
void loop(void) {
 //_______1__________2_3_________4_5__________6_7____8_____9___10_11_12
 //"$GPRMC,045103.000,A,3014.1984,N,09749.2872,W,0.67,161.46,030913,,,A*7C\r\n"   Trama GPS
 //"$GPRMC,150734.00,A,4139.49389,N,00051.88581,W,0.256,,210617,,,A*6C"\r\n"   Trama GPS

 if (Serial.available() > 0) {                                  //Si recibo algo por el puerto serie
  //long int tiempo = millis();
  String trama  = Serial.readStringUntil('\n');                          //Leo trama completa
  //Serial.println(millis()-tiempo);
  Serial.println (trama);
```

```
int indexComas [] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
if (trama.substring(0, 6) == "$GPRMC") {                    //Compruebo si empieza por "$GPRMC"
 for (int i = 1; i < 13; i++) {
  indexComas[i] = trama.indexOf(',', indexComas[i - 1] + 1);          //Guardo los indices del array en los que
  //Serial.println(indexComas[i]);                    //hay una coma
 }
 if (trama.substring(indexComas[2] + 1, indexComas[3]) == "V") {          // bit de validación ok?
  //Compruebo si el caracter de trama está OK
  GPS = false;
  Serial.println("Conectando con GPS");
 } else if (trama.substring(indexComas[2] + 1, indexComas[3]) == "A") {
  GPS = true;
  tiempoGPS = trama.substring(indexComas[1] + 1, indexComas[2]);          //Guardo Hora
  //Serial.println(hora);

  if (funcionando == true) {
   //*************************************************************************************************************
   //CÁLCULO DEL TIEMPO DE ENTRENAMIENTO
   //*************************************************************************************************************
   hora = tiempoGPS.substring(0, 2).toInt();
   //Serial.print(hora);
   minuto = tiempoGPS.substring(2, 4).toInt();
   // Serial.print(minuto);
   segundo = tiempoGPS.substring(4, 6).toInt();
   // Serial.print(segundo);

   horaEntrenamiento = hora - horaTimestamp;
   //Serial.println(horaEntrenamiento);
   minutoEntrenamiento = minuto - minutoTimestamp;
   //Serial.println(minutoEntrenamiento);
   segundoEntrenamiento = segundo - segundoTimestamp;
   //Serial.println(segundoEntrenamiento);

   duracionEntrenamiento = horaEntrenamiento * 3600 + minutoEntrenamiento * 60 + segundoEntrenamiento + offsetDuracionEntrenamiento;
   if (duracionEntrenamiento < 0) {
    horaEntrenamiento += 24;
    duracionEntrenamiento = horaEntrenamiento * 3600 + minutoEntrenamiento * 60 + segundoEntrenamiento + offsetDuracionEntrenamiento;
   }

   horaMostrar = duracionEntrenamiento / 3600;

   minutoMostrar = (duracionEntrenamiento % 3600) / 60;

   segundoMostrar = ((duracionEntrenamiento % 3600) % 60);

   tiempoDeEntrenamientoLCD = horaMostrar;
```

```
 tiempoDeEntrenamientoLCD +=":";
 if(minutoMostrar < 10){
  tiempoDeEntrenamientoLCD += "0";
  tiempoDeEntrenamientoLCD += minutoMostrar;
  tiempoDeEntrenamientoLCD +=":";
 }else{
  tiempoDeEntrenamientoLCD += minutoMostrar;
  tiempoDeEntrenamientoLCD +=":";
 }
 if(segundoMostrar < 10){
  tiempoDeEntrenamientoLCD += "0";
  tiempoDeEntrenamientoLCD += segundoMostrar;
 }else{
  tiempoDeEntrenamientoLCD += segundoMostrar;
 }
 Serial.println(tiempoDeEntrenamientoLCD);



 //*************************************************************************************************************
 //CÁLCULO DE LA DISTANCIA RECORRIDA
 //*************************************************************************************************************
 latitud = trama.substring(indexComas[3] + 1, indexComas[3] + 3).toInt() +
         (trama.substring(indexComas[3] + 3, indexComas[4]).toFloat()) / 60.0;
 longitud = trama.substring(indexComas[5] + 1, indexComas[5] + 4).toInt() +
          (trama.substring(indexComas[5] + 4, indexComas[6]).toFloat()) / 60.0;

 if (flagGPS) {                         //Si es la primera ejecución del GPS
   latitudAnt = latitud;
   longitudAnt = longitud;

   horaTimestamp = tiempoGPS.substring(0, 2).toInt();
Serial.println(horaTimestamp);
minutoTimestamp = tiempoGPS.substring(2, 4).toInt();
Serial.println(minutoTimestamp);
segundoTimestamp = tiempoGPS.substring(4, 6).toInt();
Serial.println(segundoTimestamp);
   flagGPS = false;
 }

 DLatitud = (latitud - latitudAnt) * pi / 180;
 DLongitud = (longitud - longitudAnt) * pi / 180;
 DAngulo = sqrt(DLatitud * DLatitud + DLongitud * DLongitud);
 DPosicion = DAngulo * RTierra;
 Serial.println(DPosicion);
 if (DPosicion < 0.5) {
  DPosicion = 0;
 }
 latitudAnt = latitud;
```

```
longitudAnt = longitud;

distanciaRecorrida += DPosicion;
distanciaRecorridaEntero = distanciaRecorrida;
if(distanciaRecorridaEntero > 9999){                    //xxxxx
  distanciaRecorridaLCD = distanciaRecorridaEntero;
}else if (distanciaRecorridaEntero > 999){               //0xxxx
  distanciaRecorridaLCD = "0";
  distanciaRecorridaEntero += distanciaRecorridaEntero;
}else if (distanciaRecorridaEntero > 99){                //00xxx
  distanciaRecorridaLCD = "00";
  distanciaRecorridaLCD += distanciaRecorridaEntero;
}else if(distanciaRecorridaEntero > 9 ){                 //000xx
  distanciaRecorridaLCD = "000";
  distanciaRecorridaLCD += distanciaRecorridaEntero;
}else if (distanciaRecorridaEntero < 10){                //0000x
  distanciaRecorridaLCD = "0000";
  distanciaRecorridaLCD += distanciaRecorridaEntero;
}

Serial.println(distanciaRecorrida);

//*************************************************************************************************************
//CÁLCULO DEL TIEMPO A LOS 500 METROS
//*************************************************************************************************************
t500mNudos = 971.93 / trama.substring(indexComas[7] + 1, indexComas[8]).toFloat(); //comvierto velocidad en nudos a tiempo/500m
//tiempo500m[]={minutos,segundos}
tiempo500m[1] = t500mNudos / 60.0;
tiempo500m[2] = (t500mNudos % 60 );

if (tiempo500m[1] > 15 || tiempo500m[1] < 1) {
  tiempoALos500MetrosLCD = "-:-- ";
} else if(tiempo500m[2] < 10){
  tiempoALos500MetrosLCD = tiempo500m[1];
  tiempoALos500MetrosLCD += ":0";
  tiempoALos500MetrosLCD += tiempo500m[2];
  tiempoALos500MetrosLCD += " ";
}else if(tiempo500m[2] >10 ){
  tiempoALos500MetrosLCD = tiempo500m[1];
  tiempoALos500MetrosLCD += ":";
  tiempoALos500MetrosLCD += tiempo500m[2];
  tiempoALos500MetrosLCD += " ";
}
Serial.println(tiempoALos500MetrosLCD);

Timer1.detachInterrupt();
//*************************************************************************************************************
//MUESTRO DATOS CALCULADOS POR LA PANTALLA
```

```
      //*************************************************************************************************************

      ////////////////////////////////////////////////////////////////////
      //tiempoDeEntrenamientoLCD        paladasPorMinutoLCD        //
      //                                          //
      //tiempoALos500MetrosLCD          distanciaRecorridaLCD      //
      ////////////////////////////////////////////////////////////////////

      lcd.setCursor(0, 0);
      lcd.print(tiempoDeEntrenamientoLCD);

      lcd.setCursor(10, 0);
      lcd.print(paladasPorMinutoLCD);

      lcd.setCursor(0, 1);
      lcd.print(tiempoALos500MetrosLCD);

      lcd.setCursor(10, 1);
      lcd.print(distanciaRecorridaLCD);


      Timer1.attachInterrupt(ISR_Data);
     }
    }
   }
  }
}

//*************************************************************************************************************
//CÁLCULO DE LA CADENCIA. Interrupción periódica cada 10 ms
//*************************************************************************************************************
void ISR_Data() {
 dato = analogRead(5);                          //Leemos valor acelerómetro
 suma += dato;                                  //Lo añadimos al sumatorio de los 100 últimos valores
 suma -= valoresVentana[contadorVentana];       //Quitamos el valor más antiguo
 valoresVentana[contadorVentana] = dato;        //Guardamos el valor recien leído
 contadorVentana += 1;

 if (contadorVentana == 100) {                  //Si se han leido 100 valores
  contadorVentana = 0;                          //asignamos de nuevo el índice del array.
 }

 contadorDiezmado += 1;
 if (contadorDiezmado == 15) {                  //Si hemos realizado esta tarea 15 veces (diezmado)
  filteredSignal = suma / 100;                  //Calculamos el valor medio de la señal.

  if (filteredSignal > valcomp) {               //Si el valor de la señal es mayor que el valor del comparador
   valcomp = filteredSignal;                    //Le asignamos su valor.
```

```
    contadorDecrecimiento = 0;                          //Rinicializamos el contador de decrecimiento.
    posibleMaximo = millis();                           //Guardamos el instante de este acontecimiento como posible máximo
   }
   else {                                               //Si el valor de la señal es menor que el valor del comparador
    contadorDecrecimiento += 1;                         // Incrementamos contador de decrecimiento
    //Serial.println(contadorDecrecimiento);
    valcomp -= 1;                                       //Disminuimos valor del comparador
    if (contadorDecrecimiento == 4) {                   //Si el contador de decrecimiento llega a 4
     tmax = posibleMaximo;                              //Confirmamos que se trataba de un máximo
     cadencia = (60.0 / ((tmax - tmax_3) / 1000 / 3));  //Calculamos cadencia con las 3 últimas paladas
     if (cadencia < 15) {                               //Si es inferior a 15
      cadencia = 0;                                     //Le asignamos valor 0
     }
     paladasPorMinutoLCD = cadencia;
     tmax_3 = tmax_2;                                   //Reasignamos tiempos de paladas anteriores
     tmax_2 = tmax_1;
     tmax_1 = tmax;


    }
   }
   contadorDiezmado = 0;                                //Reinicializamos contador para el diezmado
  }
}

//*************************************************************************************************************
//INTERRUPCIÓN PULSADOR RESET
//*************************************************************************************************************
void pulsadorReset() {
delay(250);
 if (digitalRead(2) == LOW) {
  Serial.println("reset");
 }
 offsetDuracionEntrenamiento = 0;
 duracionEntrenamiento = 0;
}

//*************************************************************************************************************
//INTERRUPCIÓN PULSADOR PAUSA
//*************************************************************************************************************
void pulsadorPausa() {
 delay(250);
 if (digitalRead(3) == LOW) {

  Serial.println("Pausa");
  if (GPS == true) {
   if (primeraEjecucion == true){
   lcd.setCursor(0, 0);
```

```
        lcd.print("              ");
        lcd.setCursor(0, 1);
        lcd.print("              ");
        primeraEjecucion = false;
    }
     if (funcionando == false) {
      funcionando = true;
      horaTimestamp = tiempoGPS.substring(0, 2).toInt();
      //Serial.println(horaTimestamp);
      minutoTimestamp = tiempoGPS.substring(2, 4).toInt();
      //Serial.println(minutoTimestamp);
      segundoTimestamp = tiempoGPS.substring(4, 6).toInt();
      //Serial.println(segundoTimestamp);
      flagGPS = true;
      Timer1.initialize(10000);
      Timer1.attachInterrupt(ISR_Data);
     }else  if (funcionando == true) {
      funcionando = false;
      offsetDuracionEntrenamiento = duracionEntrenamiento;
      Timer1.detachInterrupt();
     }
    }
   }
}
```

Universidad Zaragoza
1542

Trabajo Fin de Grado

Anexo de Planos

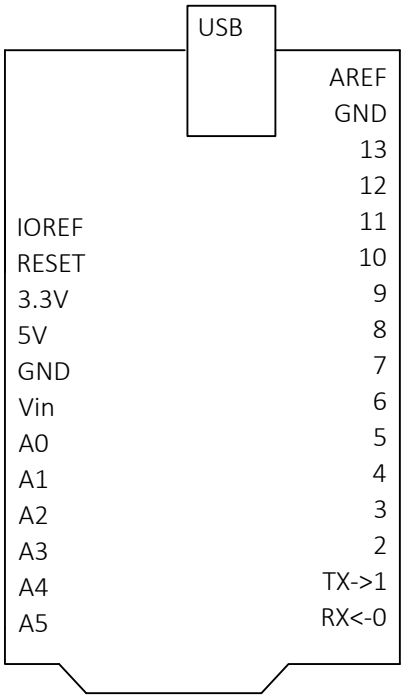
Adrián Rosales Serrano

Director/es

José Luis Villarroel Salcedo

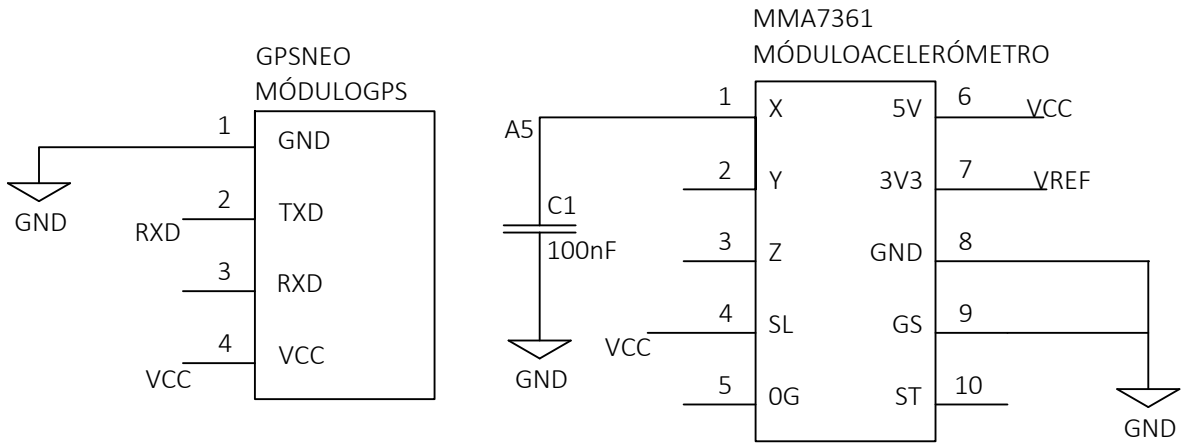
ESCUELA DE INGENIERIA Y ARQUITECTURA

2017



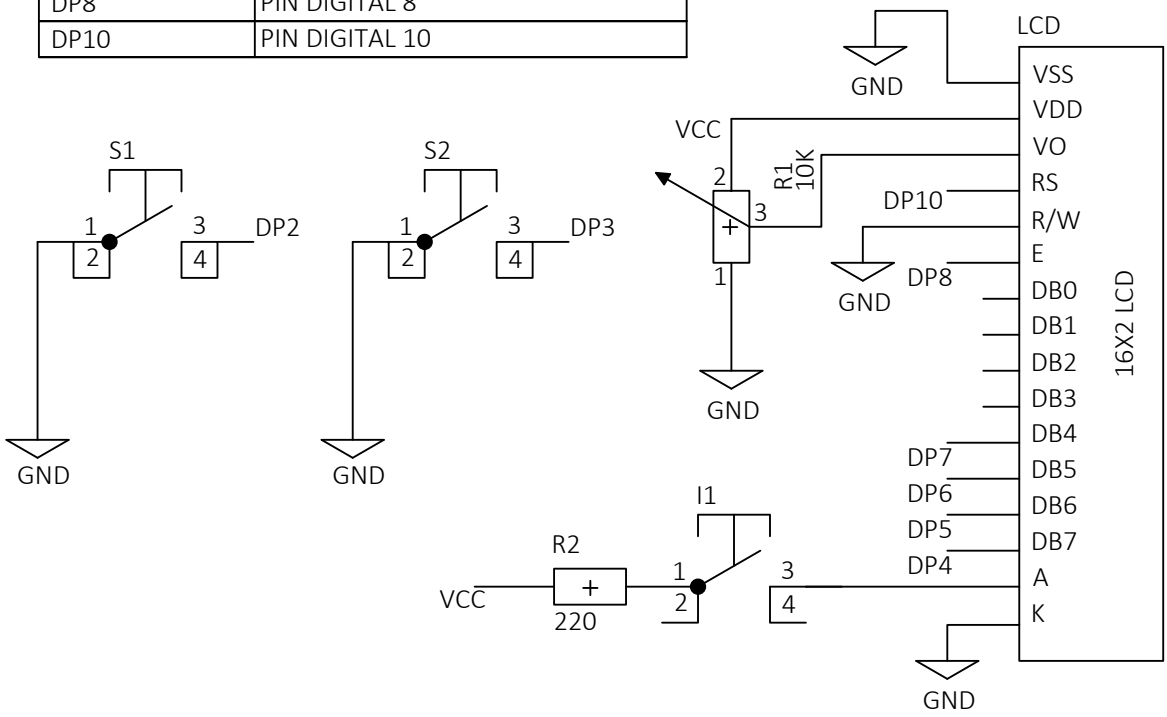
PROYECTO: DETECTOR DE CADENCIA PARA EMBARCACION DE REMO BASADA EN GPS Y ACELEROMETRO	FECHA: 18706/17
PLANO: PLANO ESQUEMÁTICO DE PLACA ARDUINO UNO REV3	ESCALA: NA
AUTOR: ARS	N° 01 /01

PIN	DESCRIPCIÓN
GND	PIN TOMA TIERRA
VCC	PIN TENSION ALIMENTACIÓN 5V
V REF	PIN REFERENCIA TENSIÓN ADC
A5	PIN ANALÓGICO 5
RXD	PIN LECTURA PUERTO SERIE



PROYECTO: DETECTOR DE CADENCIA PARA EMBARCACION DE REMO BASADA EN GPS Y ACELEROMETRO		FECHA: 30/05/17
PLANO:	PLANO ESQUEMÁTICO DE CONEXIONES PROTOBOARD INFERIOR	ESCALA: NA
AUTOR:	ARS	N° 01 /02

PIN	DESCRIPCIÓN
VCC	PIN TENSION ALIMENTACIÓN 5V
GND	PIN TOMA TIERRA
DP2	PIN DIGITAL 2
DP3	PIN DIGITAL 3
DP4	PIN DIGITAL 4
DP5	PIN DIGITAL 5
DP6	PIN DIGITAL 6
DP7	PIN DIGITAL 7
DP8	PIN DIGITAL 8
DP10	PIN DIGITAL 10



PROYECTO: DETECTOR DE CADENCIA PARA EMBARCACION DE REMO BASADA EN GPS Y ACELEROMETRO		FECHA: 30/05/17
PLANO:	PLANO ESQUEMÁTICO DE CONEXIONES PROTOBOARD SUPERIOR	ESCALA: NA
AUTOR:	ARS	Nº 02 /02