COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

ASISTENTE ELECTRÓNICO DE ENTRENAMIENTOS DE NATACIÓN

- Autor: Jorge Zumarraga Martínez
- Director: Álvaro Sánchez Miralles

Junio 2019, Madrid

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Jorge Zumarraga Martínez

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: ASISTENTE ELECTRÓNICO DE ENTRENAMIENTOS DE NATACIÓN,

que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 12 de Julio de 2019

ACEPTA

Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
ASISTENTE ELECTRÓNICO DE ENTRENAMIENTOS DE NATACIÓN
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018-2019 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Jorge Zumarraga Martínez Fecha: 12/ 07/ 19

Autorizada la entrega del proyecto
EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Sánchez Miralles Fecha: 12/ 07/ 19

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TECNICA SUPERIOR DE INGENIERIA-ICAI

Asistente electrónico de entrenamientos de natación

RESUMEN DEL PROYECTO

Autor: Zumarraga Martínez, Jorge
Director: Sánchez Miralles, Álvaro

1-INTRODUCCIÓN

El proyecto consiste en el diseño, programación e implantación de un sistema para la medición de tiempos y largos durante la práctica de natación. Dicho sistema cuenta con un sensor de contacto que se situará en la pared de la piscina, una caja de control desde donde se gestiona el sistema y un display desde donde se mostrarán los tiempos y número de largos recorridos. El nadador durante el uso del sistema podrá sin interrumpir su nado visualizar dichos datos.

El sistema cuenta con dos modos de uso. El primero está enfocado a entrenamientos de largas distancias donde en el display se muestran largos totales recorridos y tiempo total de entrenamiento. El segundo está enfocado al seguimiento del ritmo de entrenamiento. En este modo de uso en el display se muestran el número de largos recorridos y el tiempo empleado en recorrer el último largo.

2-OBJETIVOS

Los objetivos del sistema son los siguientes:

1- Realizar un aplicación intuitiva y sencilla en su uso. La lógica del sistema debe ser fácilmente controlada por el usuario con un manejo sencillo a través de pulsadores.

2- El sistema debe ser ligero para poder instalarlo en distintos lugares con facilidad.

3- Cómoda visualización de los datos. El display debe tener las dimensiones y calidad de visualización adecuadas para que la visualización de los datos no modifique el trascurso del entrenamiento del usuario.

3-METODOLOGÍA

Para la realización del proyecto se ha dividido en tres partes: fabricación de la placa de contacto, programación del microprocesador y configuración del display.

El microprocesador se ha programado en lenguaje C usando el editor MPLAB. Las entradas del micro son el sensor de contacto y pulsadores para la interacción con el usuario. Las salidas del micro son leds para visualizar el modo de uso, una bocina para realizar una rutina de inicio de entrenamiento y el display. Se usa el timer 1 del microprocesador para la gestión de la lógica y el control de los tiempos. También se emplea la UART para la comunicación con el display.

En cuanto a la placa de contacto, el armazón del sensor se ha fabricado con unas dimensiones de 45x30 cm. El sistema de fijación a la pared cuenta con ventosas para hacer el dispositivo portable y poder implantarlo en distintas piscinas. Contará con un sensor de fuerza resistivo y un tratado de la señal antes de llegar al microprocesador.

La visualización de los datos se hace a través del display Nextion NX8048T070 de 7 pulgadas. Para la configuración del display se ha empleado el programa Nextion Editor. En la configuración se han definido el fondo, los distintos objetos y su localización dentro del display. Una vez realizada la configuración dichos objetos, en este caso dígitos, se modifican a través del sistema de comunicación serie del microprocesador.

4-RESULTADOS

El sistema dependiendo de la opción de uso seleccionada por el usuario muestra distintas visualizaciones por el display.

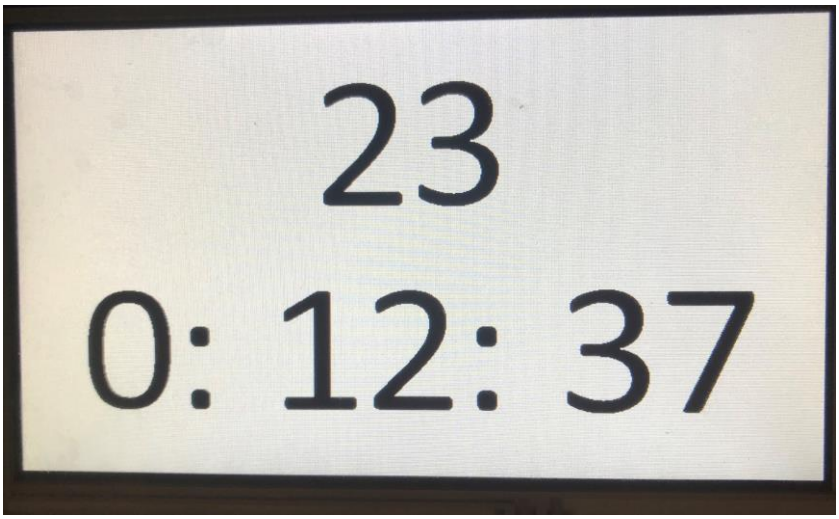


Figura 1. Display en un instante en el modo de uso 1

En la figura 1 observamos el funcionamiento del modo de uso 1. Se visualizan los datos de largos y tiempo total de entrenamiento. En el primer modo de uso se muestran los datos de largos recorridos y el tiempo con el siguiente formato: h:mm:ss. Siendo h horas, m minutos y s segundos. En este instante se observa que el usuario lleva nadados 23 largos y han transcurrido 12 minutos y 37 segundos de entrenamiento.

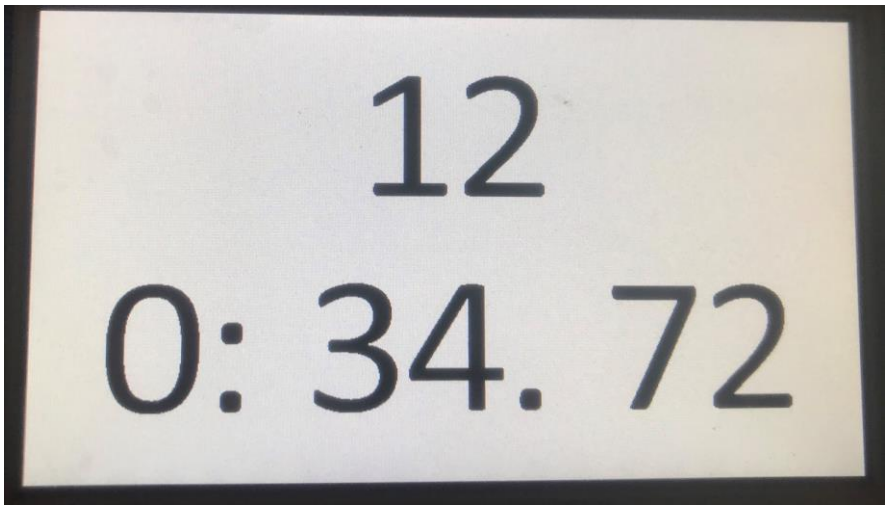


Figura 2. Display en un instante en el modo de uso 2

En la figura 2 observamos el funcionamiento del modo de uso 2. Se visualizan los datos de largos y tiempo en el que el usuario a completado el último largo. En el segundo modo de uso se muestran los datos de largos recorridos y el tiempo empleado en realizar la última vuelta con el siguiente formato: m:ss.cc. Siendo m minutos, s segundos y c centésimas. En este instante se observa que el usuario lleva nadados 12 largos y el último largo lo ha realizado en 34 segundos y 72 centésimas.

5-CONCLUSIONES

Este sistema tiene funciones similares a dispositivos ya existentes en el mercado. Con este sistema se eliminan las carencias de las distintas soluciones ya creadas. Lo que hace este sistema único es la cómoda visualización de los datos, el precio económico, el manejo sencillo de la aplicación y su portabilidad.

Las pruebas de uso realizadas han sido satisfactorias. El sistema desarrollado interpreta la señal proveniente del sensor, gestiona la lógica correspondiente en función de la opción de uso seleccionada por el usuario y muestra los datos correspondientes con total claridad en el display.

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TECNICA SUPERIOR DE INGENIERIA-ICAI

Electronic assistant for swimming trainings

SUMMARY OF THE PROYECT

Author: Zumarraga Martínez, Jorge
Director: Sánchez Miralles, Álvaro

1-INTRODUCTION

The project consists of the design, program and implementation of a system for measuring times and lengths during swimming practice. This system has a contact sensor that is located on the wall of the pool, a control box from which the system is managed and a screen showing the times and the number of the laps. The swimmer may visualize the data without interrupting the training.

The system has two uses. The first is focused on long distance training where the total training time is shown, whilst the second is focused on monitoring the training rhythm. In this second use, the screen shows the number laps and the time spent traveling the last lap.

2-OBJECTIVES

The objectives of the system are the following:

1- Make an intuitive and simple application in its use. The logic of the system must be easily controlled by the user with a simple operation through push buttons.

2- The system must be light so it can be easily installed in different places.

3- Comfortable visualization of the data. The display must have the appropriate dimensions and quality so that the visualization of the data improves the comfort of the user.

3-METHODOLOGY

The creation of the project has been divided in three phases: the manufacture of the touchpad, the microprocessor programming and the display configuration.

The microprocessor has been programmed in C language using the MPLAB editor. The micro inputs are the contact sensor and push buttons for the interaction with the user. The micro outputs are LEDs to inform about the mode of use, a horn to perform a training start routine and the display. The timer 1 of the microprocessor is used for logic management and time control. The UART is also used for the communication with the display.

In the manufacture of the touchpad, the sensor frame has been manufactured with the dimensions of 45x30 cm. The fixing system of the touchpad to the wall are suction cups to make the device portable and able to implant it in different pools. It will have a resistive force sensor and a signal processing before reaching the microprocessor.

The visualization of the data is done through the 7-inch Nextion NX8048T070 display. The Nextion Editor program has been used for the configuration of the display. In the configuration, the background, the different objects and their location within the display have been defined. Once the configuration has been carried out, said objects, in this case digits, are modified through the serial communication system of the microprocessor.

4-RESULTS

The system shows different visualizations on the display depending on the option of use selected by the user.

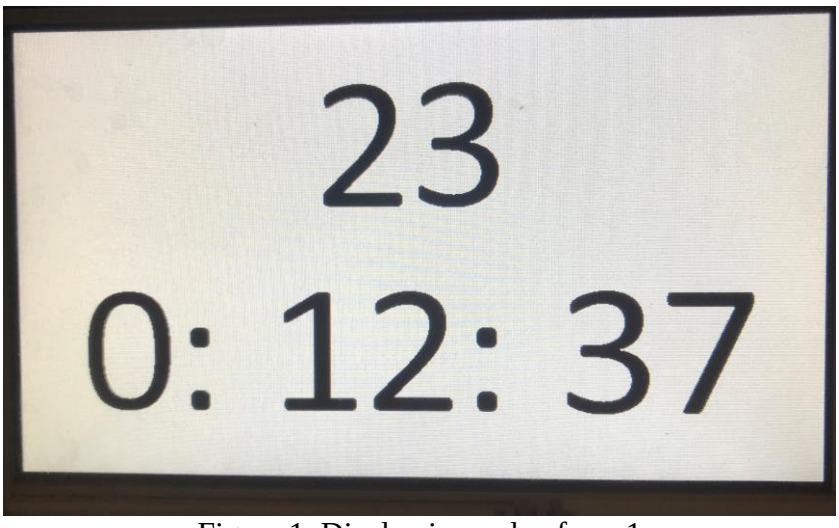


Figure 1. Display in mode of use 1

In Figure 1 we observe the operation of the use mode 1. The data of laps and total training time is displayed. In the first mode of use time is shown in the following format: h: mm: ss. Being h hours, m minutes and s seconds. At this moment the user has swum 23 laps in a time of 12 minutes 37 seconds.

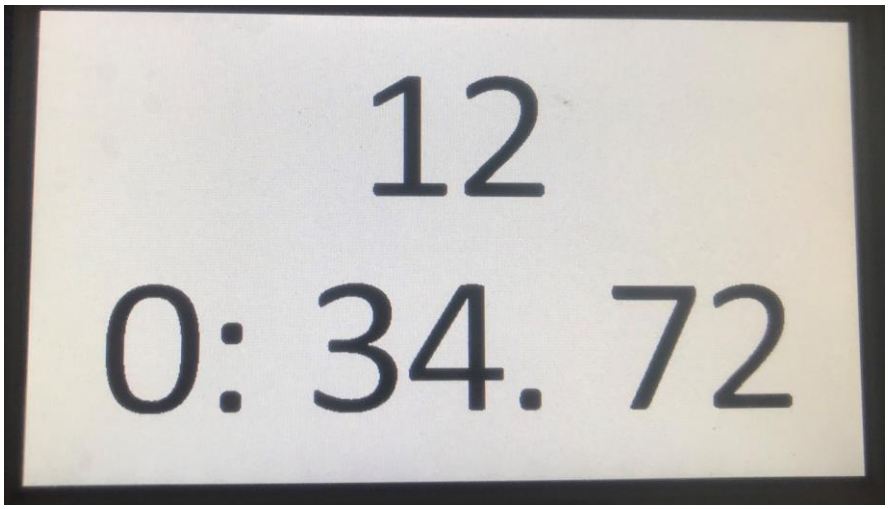


Figure 2. Display in mode of use 2

In figure 2 we observe the operation of the use mode 2. The data of laps and time in which the user has completed the last lap are displayed. In the second mode of use, the time used to carry out the last round is shown in the following format: m: ss: hh. Being m minutes, s seconds and h hundredths of a second. At this moment, it is observed that the user has swam 12 laps and the last lap was performed in 34 seconds and 72 hundredths of second.

5. CONCLUSIONS

This system has similar functions to existing devices in the market. With this system, the deficiencies of the different solutions already created are eliminated. What makes this system unique is the comfortable visualization of the data, the economical Price, the easy handling of the application and its portability.

The tests of use have been satisfactory. The developed system interprets the signal coming from the sensor, manages the corresponding logic according to the use option selected by the user and displays the corresponding data with total clarity on the display.

COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

ASISTENTE ELECTRÓNICO DE ENTRENAMIENTOS DE NATACIÓN

- Autor: Jorge Zumarraga Martínez
- Director: Álvaro Sánchez Miralles

Junio 2019, Madrid

ÍNDICE DE CONTENIDO:

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS:

PARTE I MEMORIA DESCRIPTIVA

CAPÍTULO 1 INTRODUCCIÓN

1.1 INTRODUCCIÓN

El proyecto consiste en el diseño, programación e implantación de un sistema para la medición de tiempos y largos durante la práctica de natación. Dicho sistema cuenta con un sensor de contacto que se situará en la pared de la piscina, una caja de control desde donde se gestiona el sistema y un display donde se mostrarán los tiempos y número de largos recorridos. El nadador durante el uso del sistema podrá sin interrumpir su nado visualizar dichos datos.

El sistema cuenta con dos modos de uso. El primero está enfocado a entrenamientos de largas distancias donde en el display se muestran largos totales recorridos y tiempo total de entrenamiento. El segundo está enfocado al seguimiento del ritmo de entrenamiento. En este modo de uso en el display se muestran el número de largos recorridos y el tiempo empleado en recorrer el último largo.

1.2 ESTADO DEL ARTE

En la actualidad hay varios dispositivos para un propósito similar al de este proyecto.

1.2.1-Smartwatches y pulseras de actividad.

Estos dispositivos fueron creados a comienzos de este siglo. Son relojes de pulsera computarizados, su funcionalidad va más allá que la del reloj convencional. Estos dispositivos disponen de varios sensores que son capaces de medir principalmente ritmo cardiaco, número de largos recorridos, tiempos de nado y distancia recorrida en aguas abiertas. Para la obtención de los datos del nado estos dispositivos cuentan con un acelerómetro que detecta el impulso al hacer contacto con la pared.

Estos instrumentos incorporan un microprocesador en el que procesa los datos generados por el usuario y además controla la comunicación externa, la memoria y distintos componentes periféricos. Para la visualización de los datos contienen un display de entre una y dos pulgadas. Estos relojes registran los entrenamientos y tienen la posibilidad de descargar los datos en un ordenador.

Las principales marcas de estos dispositivos son Garmin, Apple, Samsung y Fitbit Versa. Su precio oscila desde los 20€ para los más sencillos hasta los 600€ para los más dotados en tecnología y alta gama.

Figura 1. Smartwatch de actividad fabricante Garmin

1.2.2-Sistemas empleados en competiciones de alto nivel.

Estos dispositivos son los más precisos del mercado. Estos sistemas se componen de un panel de contacto, un sistema de arranque que incluye un pulsador, un flash y una bocina, un cronómetro digital, un display de LEDs donde quedan visualizados los tiempos para los espectadores y una impresora de tiempos donde quedan registrados los tiempos para el uso de los jueces.

El panel de contacto pesa aproximadamente 25 kilos, está compuesto por una carcasa de PVC tratado para mejorar sus prestaciones de agarre y varios sensores. Los sensores empleados son sensores de banda de contacto, están compuestos por dos bandas conductoras y un mecanismo elástico de separación. El nadador cuando presiona el panel de contacto ejerce una presión superior a la contrapresión ejercida por el mecanismo de separación del sensor, esto provoca el contacto entre las bandas conductoras y el sistema detecta de esta manera la llegada del nadador.

Los principales fabricantes de estos dispositivos son Omega, Alge-Timing, Daktronics y Swiss Timing. El precio de estos sistemas supera los 2000€.

Figura 2. Sistema de cronometraje de natación para alta competición de Swiss Timing.

1.2.3-Registrador electrónico para la práctica de natación en piscina:

El dispositivo consiste en un sistema para recopilar datos del entrenamiento de natación. El sistema recopila datos de tiempos parciales, tiempos totales, número de largos y distancia recorrida. El sistema consiste en 3 principales componentes: una placa de toque, una unidad de control, y una unidad remota.

La placa de toque está formada por una caja estanca, un sensor piezoeléctrico y su acondicionamiento de la señal. El sensor piezoeléctrico emite un pulso de señal cada vez que es presionado, el pulso pasa por el circuito de acondicionamiento antes de llegar al microprocesador. El circuito de acondicionamiento consiste en un comparador con histéresis para generar una onda cuadrada entre los márgenes establecidos de tensión, esto facilita la lectura de la señal por parte del microprocesador y elimina el ruido que pueda haber en la señal.

La unidad de control se encarga de gestionar la lógica del sistema, cuenta con dos pulsadores para el manejo del sistema por parte del usuario. La unidad recibe la señal de la placa de toque, gestiona la lógica del manejo de tiempos y distancias y manda la información vía bluetooth a la unidad remota.

La unidad remota consiste en una aplicación de un teléfono móvil. Desde la unidad remota se manda a la unidad de control la distancia del largo de la piscina. En la aplicación se visualizan los datos del entrenamiento: número de largos, distancia recorrida, tiempo total y tiempos parciales. También señala cual ha sido la vuelta más rápida como se observa en la figura 3.

Figura 3. Interfaz de usuario en la unidad remota

1.3 MOTIVACIÓN

El avance de la tecnología también hace mella en el ámbito deportivo. El uso de la tecnología en el deporte mejora la eficiencia, comodidad y rendimiento del deportista. El cansancio del entrenamiento y la concentración en la técnica de nado complica la memorización del número de largos recorridos en entrenamientos de largas distancias. Por otro lado, la eficiencia del entrenamiento aumenta si se conoce al ritmo al que va nadando pudiendo así fijar un ritmo objetivo y reflejar la evolución.

Los sistemas ya desarrollados con propósitos similares al de este proyecto no cumplen los requerimientos en los que se basa este trabajo. Los smartwatches son una solución

económica, portable y cómoda, pero carece de precisión e interrumpe el nado a la hora de visualizar los datos en el pequeño display. Los sistemas empleados en alta competición son una solución precisa, pero de precio elevado difícil de transportar ya que tiene unas grandes dimensiones y suele estar permanentemente fijado a una sola piscina. El registrador electrónico para la práctica de natación en piscina es una solución que registra los largos y tiempos del entrenamiento y quedan registrados en una aplicación móvil, pero este sistema carece de display para visualizar los datos durante el nado y el sensor es poco ergonómico.

Por los motivos mencionados anteriormente surge la idea de este proyecto. Se desea desarrollar un sistema económico, portable, preciso y con una cómoda visualización de los datos del entrenamiento. Es una solución que no está desarrollada y que elimina las carencias de los sistemas ya empleados.

1.4 OBJETIVOS DEL PROYECTO

Los objetivos del sistema son los siguientes:

1- Realizar un aplicación intuitiva y sencilla en su uso. La lógica del sistema debe ser fácilmente controlada por el usuario con un manejo sencillo a través de pulsadores.

2- El sistema debe ser ligero para poder instalarlo en distintos lugares con facilidad.

3- Cómoda visualización de los datos. El display debe tener las dimensiones y calidad adecuadas para que la visualización de los datos no modifique el trascurso del entrenamiento del usuario.

1.5 RECURSOS EMPLEADOS

Para el desarrollo de este sistema son necesarios los siguientes componentes de hardware y de software.

1.5.1 Hardware

1- Microcontrolador dsPIC33FJ32MC202

En el desarrollo del proyecto se ha decidido implantar el software en el dsPIC33FJ32MC202.

Figura 4. Microcontrolador dsPIC33FJ32MC202

2- Pulsadores

Se han empleado pulsadores para la interacción del usuario con el sistema.

Figura 5. Pulsador cpm-0039 de 12 mm

3- Sensor de fuerza resistivo

Este sensor varía el valor de su resistencia con la fuerza aplicada. El valor de su resistencia disminuye con el aumento de la fuerza aplicada.

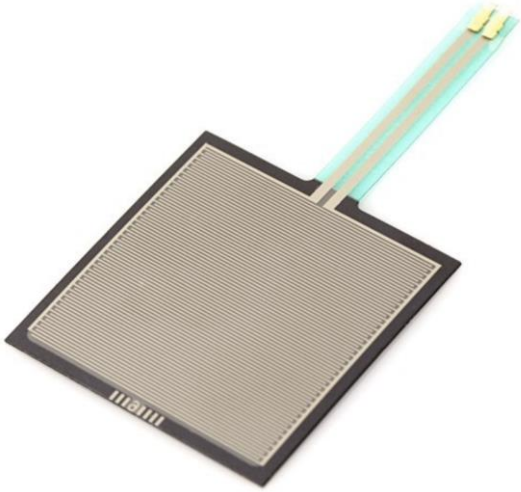


Figura 6. Sensor FSR 406.

4-Display Nextion NX8048T070

Se ha empleado el display Nextion NX8048T070 para la visualización de los datos.

Figura 7. Display Nextion NX8048T070

5- Placas de PVC

Se han empleado dos placas de PVC de 30x45 cm para la fabricación de la placa de toque.

6-Otros componentes electrónicos

Resistencias, condensadores, cableado y LEDs.

1.5.2 Software

1- MPLAB IDE v5.10

Se ha empleado el entorno de desarrollo integrado MPLAB IDE v5.10. Es un editor gratuito orientado a la programación de microprocesadores y otros productos de la marca Microchip.

Figura 8. Logo de MPLAB X IDE.

2- Nextion Editor

Se ha empleado el programa Nextion Editor para la configuración, edición y simulación del Display Nextion NX8048T070.

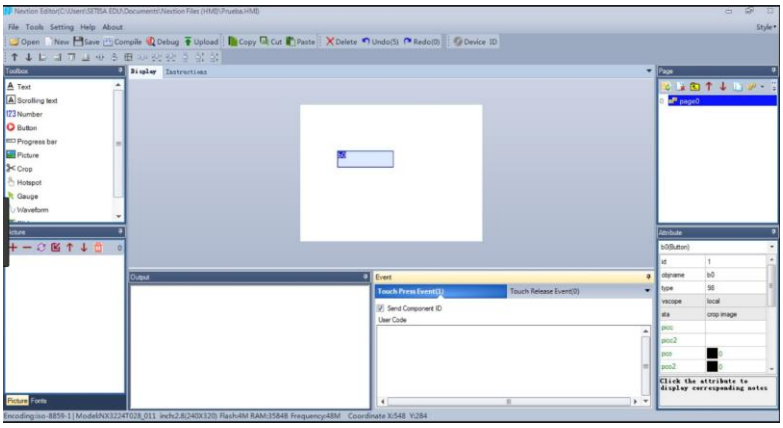


Figura 9. Entorno de desarrollo Nextion Editor.

1.6 METODOLOGÍA DE TRABAJO

Para la realización del proyecto se ha dividido en tres partes: fabricación de la placa de contacto, programación del microprocesador y configuración del display.

En cuanto a la placa de contacto el sistema tiene tres vías de ataque: la selección del sensor, diseño del armazón y sistema de fijación. Para la selección del sensor se han realizado pruebas con varios tipos de sensores y se ha escogido el que mejor resultados obtiene. Por otro lado, el diseño del armazón tiene que satisfacer las características de diseño, la placa de toque debe ser impermeable y debe tener las dimensiones adecuadas para no limitar el estilo con el que el nadador haga el contacto en ella. Por último, se ha ideado el sistema de fijación.

El siguiente paso es programar el microprocesador, se ha programado en lenguaje C usando el editor MPLAB. La programación del microprocesador se ha dividido en tres partes: programación de la lógica del sistema, programación de la entrada del sensor y

programación del display. Las entradas del micro son el sensor de contacto y pulsadores para la interacción con el usuario. Las salidas del micro son leds para visualizar el modo de uso, una bocina para realizar una rutina de inicio de entrenamiento y el display. Se usa el timer 1 del microprocesador para la gestión de la lógica y el control de los tiempos, el módulo A/D para la entrada de la señal proveniente del sensor y la UART para la comunicación con el display.

La visualización de los datos se hace a través del display Nextion NX8048T070 de 7 pulgadas. Para la configuración del display se ha empleado el programa Nextion Editor. En la configuración se han definido el fondo, los distintos objetos y su localización dentro del display. Una vez realizada la configuración dichos objetos, en este caso dígitos, se modifican a través del sistema de comunicación serie del microprocesador.

CAPÍTULO 2: PLACA DE TOQUE

En este capítulo se va a detallar la fabricación de la placa de toque. En primer lugar, se especifica y se pone en conocimiento el funcionamiento del sensor. En segundo lugar, se precisa el circuito acondicionador del sensor. En último lugar, se aclara la fabricación de la placa de toque.

2.1 SENSOR

El sensor seleccionado para la placa de toque es el sensor de fuerza resistivo FSR 406. Este sensor tiene una relación inversa entre fuerza aplicada y el valor de la resistencia. En la figura 10 mostramos la característica fuerza-resistencia del sensor. Estos sensores no son aptos para mediciones precisas de fuerza.

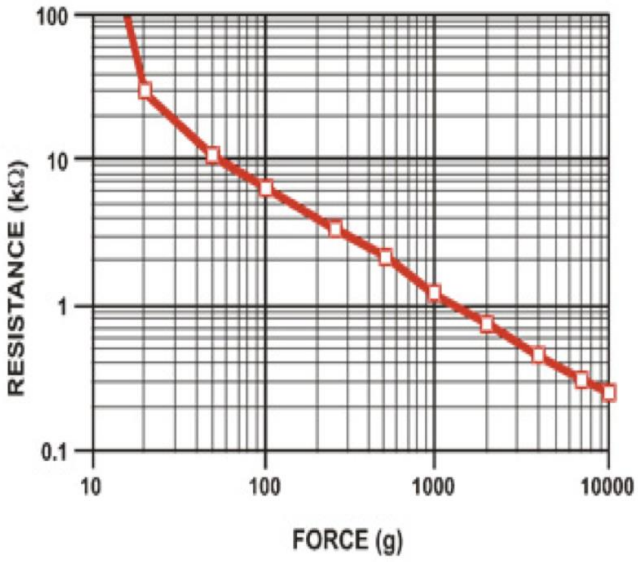


Figura 10. Característica fuerza-resistencia del sensor FSR 406

El sensor está formado por cuatro capas. La primera capa consiste en una capa activa con un circuito integrado con un patrón de forma. En segundo lugar, está la capa espaciadora, su función es fijar la separación entre la primera capa y la tercera capa mientras no se ejerza fuerza sobre el sensor. La tercera capa es una lámina conductora, en cuanto más fuerza se ejerza más contactos habrá entre la capa conductora y la capa activa produciendo un descenso del valor de la resistencia. La última capa consiste en un adhesivo para mantener las capas unidas.

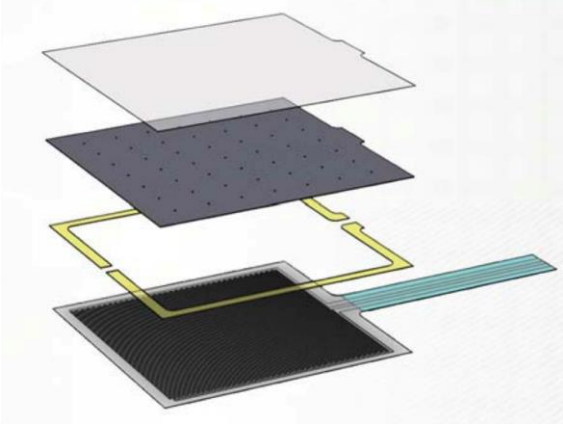


Figura 11. Esquema sensor FSR 406

2.2 ACONDICIONAMIENTO DEL SENSOR

El sensor se va a utilizar en forma de divisor de tensión como en la Figura 12, de esta forma en cuanto mayor sea la fuerza aplicada mayor será el voltaje de salida. Siendo Vout $= V1x \frac{R1}{R1+Rsensor}$.

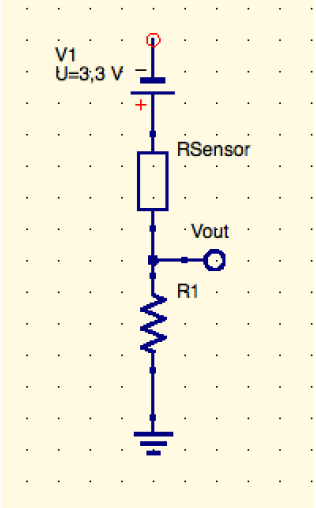


Figura 12. Esquema del circuito acondicionador del sensor

Para la selección de la resistencia R1 buscamos la característica fuerza-voltaje más conveniente, es decir, una característica que permita diferenciar bien los voltajes de salida entre las distintas fuerzas ejercidas por el nadador y las ejercidas por el impulso del agua. Experimentalmente se han obtenido los siguientes resultados: fuerza ejercida por el impulso del agua son inferiores a 200 gramos fuerza y la fuerza ejercida por el contacto del nadador son superiores a 400 gramos fuerza. Por ello se ha seleccionado R1= 10KΩ. Con R1 en el circuito acondicionador queda una característica fuerza-voltaje de salida como en la Figura 13.

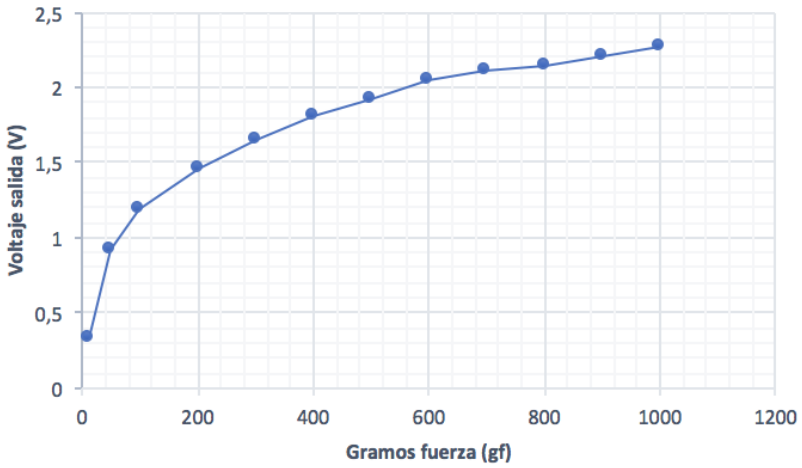


Figura 13. Característica Fuerza-Voltaje salida

2.3 FABRICACIÓN ARMAZÓN

La fabricación del armazón va sujeta a las condiciones del proyecto. La placa de contacto debe tener las dimensiones adecuadas para un como contacto con la pared y no limitar el estilo con el que el nadador haga el contacto ya sea a una mano, a dos manos o con volteo. También debe ser estanco para su correcto funcionamiento en medio acuático.

Para satisfacer las condiciones de diseño se ha ideado un sistema con 2 placas de PVC colocadas con un separador y selladas por un aislante, un prensaestopa PG 9 en la placa posterior para sellar el paso del cableado y 4 ventosas para su fijación en la pared. Las placas de PVC se diseñaron con unas dimensiones de 30x45 cm y las cuatro esquinas achaflanadas con un radio de curvatura de 10mm para evitar posibles cortes.

Figura 14. Placa posterior con prensaestopa (izq.) y placa delantera (dcha.).

CAPÍTULO 3: CAJA DE CONTROL

3.1 HARDWARE

El sistema cuenta con varias entradas y varias salidas. El microcontrolador recibe la información proveniente de las entradas, gestiona la lógica correspondiente y actúa a través de las salidas. Las entradas del micro son: pulsadores para la interacción con el usuario y el panel de contacto. Las salidas del micro son: LEDs para indicar el estado del sistema, un zumbador para realizar un sistema de comienzo del entrenamiento y un display donde se mostrarán los datos.

3.1.1 Conexionado de pines del micro

Pin del micro empleado	Descripción	Conexión
6	Rb2	Pulsador1
7	An5	Sensor
15	Rb6	Pulsador 2
24	Rb13	Zumbador
25	Rb14	LED2
26	Rb15	LED1
11	tx	Rx display
14	rx	Tx display

Tabla 1. Conexiones pines del micro.

3.1.2 Circuitos de entradas y salidas

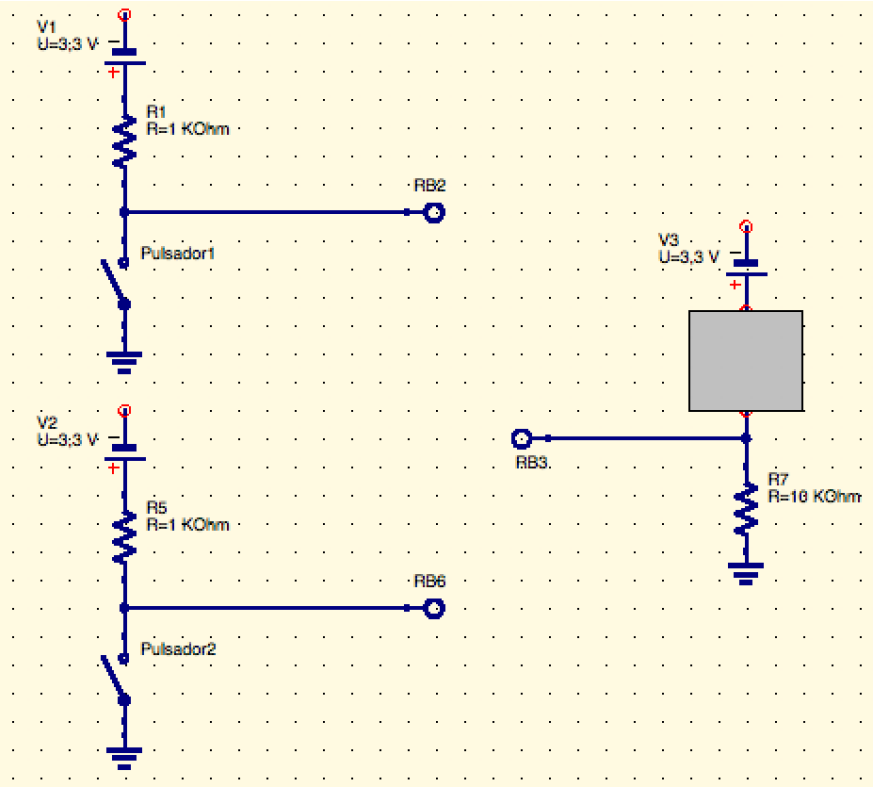


Figura 15. Esquema conexiones entradas del micro.

En la Figura 15 se observan los circuitos de entrada al micro. Las entradas constan de dos entradas digitales conectadas a los pulsadores y una entrada analógica proveniente del sensor de la placa de toque. Los pulsadores se han diseñado con esquema *pull-up*. La tensión de salida del micro es de 3,3 V. Se han diseñado las resistencias de los pulsadores para una corriente máxima de 3,3 mA, para ello se han escogido unas resistencias de 1 KΩ.

El pulsador 1 es el pulsador de manejo del sistema con él se selecciona el modo de uso, se ha conectado al pin RB2. El pulsador 2 es el pulsador de reset, inicializa el sistema de nuevo, se ha conectado al pin RB6. La señal proveniente del sensor se ha conectado al pin correspondiente a la entrada analógica AN5.

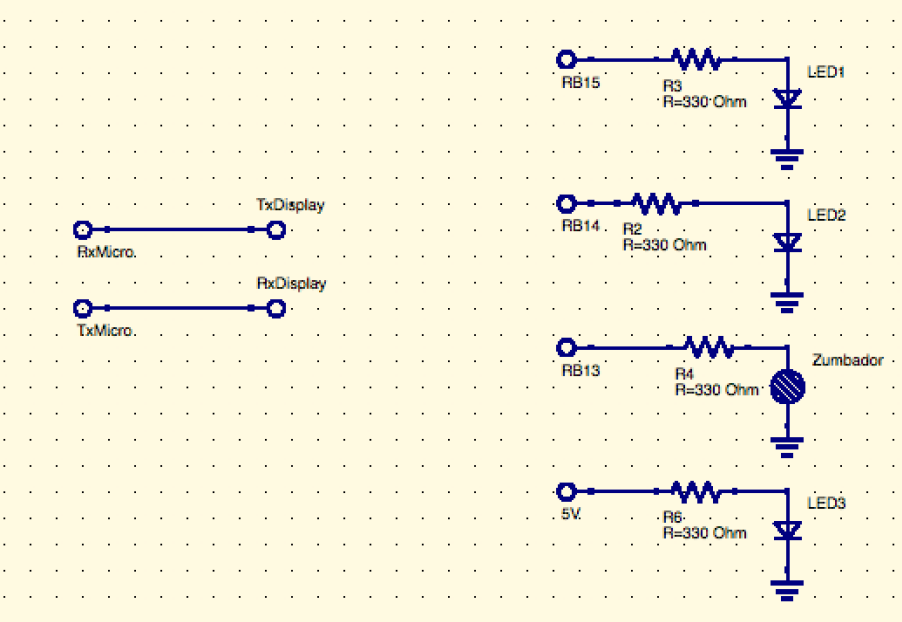


Figura 16. Esquema conexiones de salida del micro.

En la Figura 16 se observan los circuitos de salida del micro. Las salidas constan de tres LEDs y un zumbador. Las salidas del micro en modo activo son de 3,3V. Se han diseñado las resistencias para una corriente máxima de 5mA. La caída de tensión de los diodos LED es de 1,8V. Se han escogido resistencias de 330 Ω.

$$I = \frac{3{,}3 - 1{,}8}{330} = 4{,}45\ mA$$

Los LEDS 1 y 2 son los encargados de mostrar el modo de uso seleccionado por el usuario, van conectados a los pines RB15 y RB14. El LED 3 muestra el encendido del sistema. El zumbador va conectado al pin RB13, se encarga de señalizar la salida. Por último, para el manejo del display se ha conectado el terminal de transmisión del micro (TxMicro) al terminal de recepción del display (RxDisplay) y el terminal de recepción del micro (RxMicro) al terminal de transmisión del display (TxMicro).

3.1.3 Alimentación

La alimentación del microcontrolador se realiza con una batería de 9V conectada al puerto Jack de alimentación del micro. El positivo pasa por un conmutador para el encendido y apagado del sistema.

3.2 SOFTWARE

3.2.1 Timer

El Timer es un periférico del microcontrolador con la función de gestionar y controlar el tiempo sin restar tiempo de CPU. El control del tiempo se basa en el oscilador principal de frecuencia F_{cy}= 39,61MHz. Este periférico es especialmente útil a la hora de ejecutar rutinas de forma periódica. El microcontrolador empleado dsPIC33MC32FJ202 contiene 3 Timers.

Los principales registros del Timer son el TMRx, el PRx, TxCON, e IFS0.

El registro TMRx es un registro de 16 bits, su función es llevar la cuenta del timer. Contiene el valor del timer en cada instante, incrementa con cada ciclo de tiempo configurado por el pre-escalado PRx.

El registro PRx configura es el valor que debe alcanzar el Timer para resetear su cuenta. El pre-escalado indica al Timer la cantidad de ciclos de reloj que deben transcurrir para actualizar su valor en TMRx. Para obtener el valor del pre-escalado en función del tiempo se emplea la siguiente formula:

$$\text{PRx} = \frac{F_{cy}}{\text{pre-escalado}} \text{ x tiempo}$$

El control del Timer se lleva a través del registro TxCON sus bits más importantes son los mostrados en la Figura 17.

	Bit 15	-	***Bit 5***	***Bit 4***	-
TxCON	TON		TCKPS1	TCKPS0	

Figura 17. Bits registro TxCON

- La función del bit TON dentro del registro TxCON es poner en funcionamiento o en pausa el Timer. Si el bit está a 1 el Timer actualiza su cuenta, si está a 0 el Timer para de contar.

- Los bits 4 y 5 del registro TxCON correspondientes a TCKPS1 y TCKPS0 recogen el pre-escalado que puede variar de 0 a 3. Si el valor es 0 el pre-escalado es nulo, el Timer se actualiza cada ciclo del oscilador (25ns). Si el valor es 1 el pre-escalado es 1:8, el Timer se actualiza cada ocho ciclos del oscilador (200ns). Si el valor es 2 el pre-escalado es 1:64, el Timer se actualiza cada sesenta y cuatro

ciclos del oscilador (1,6µs). Si el valor es 3 el pre-escalado es 1:256, el Timer se actualiza cada doscientos cincuenta y seis ciclos del oscilador (6,4µs).

El registro IFS0 se encarga de indicarnos cuando el Timer ha terminado la cuenta, es decir, cuando el valor del registro TMRx ha alcanzado el valor configurado en el registro PRx.

Hay dos formas de emplear el Timer, por interrupciones o polling. En el caso del polling el sistema está constantemente interpretando si el Timer ha llegado al fin de la cuenta, cuando llega el momento ejecuta el código correspondiente. En el modo de uso por interrupciones por lo general cuando el Timer llega al fin de la cuenta la CPU pausa la ejecución del código y ejecuta la subrutina asociada al fin de la cuenta del Timer. Para este modo de uso es necesario habilitar la interrupción, para ello es necesario poner a 1 el bit del registro de interrupciones asociado al Timer.

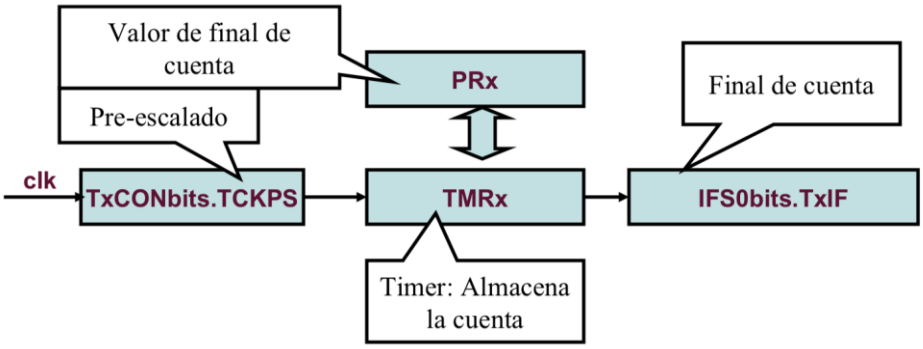


Figura 18. Esquema de manejo del Timer.

En este proyecto se ha empleado el Timer 1 con interrupciones como modo de uso para evitar desperdicios de ciclos de CPU. Para habilitar las interrupciones se ha puesto a 1 el bit del registro de interrupciones asociado al Timer 1. El Timer se ha configurado con el objetivo de que cuente hasta 1 ms, cuando la cuenta llega a 1 ms salta el flag de interrupción y se ejecuta la rutina de interrupción donde su vuelve a poner el flag de interrupción a 0 para que el Timer vuelva a contar. Para configurar el Timer a 1 ms se ha fijado el pre-escalado a 0 (el timer se actualiza cada 25ns) y PR1= 40000 (4000x25ns=1ms). En la subrutina se ha codificado la lógica del cronometro donde se actualizan las variables asociadas al tiempo.

3.2.2 Módulo de conversión A/D

La conversión analógico-digital consiste en a través de distintos procedimientos muestrear una señal analógica y convertirlo en valores digitales. El conversor analógico digital del dsPIC33FJ32MC202 sigue el esquema indicado en la Figura 19. El conversor está compuesto por dos elementos: un canal y un conversor A/D. Este módulo se emplea para convertir la señal proveniente de la placa de contacto.

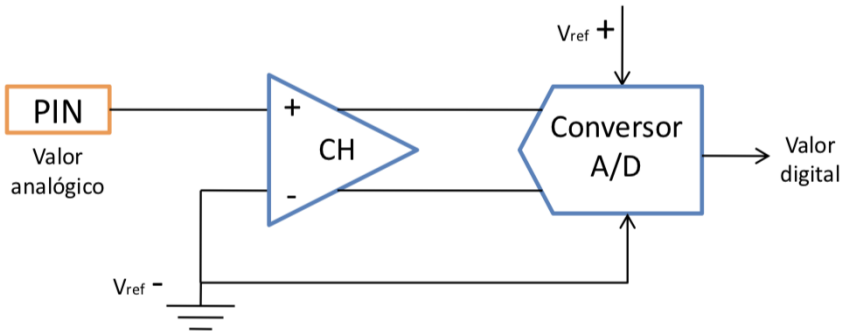


Figura 19. Esquema conversor A/D del dsPIC33FJ32MC202.

El canal, compuesto por un elemento con un condensador, actúa como muestreador-retenedor. El condensador se carga con la señal analógica que entra a través del pin al que está conectado. Una vez completada la carga del condensador será el valor de tal tensión la que se convierta en un valor digital a través del conversor.

La función del conversor es convertir el valor analógico en valor digital. Para el adecuado uso del convertidor debe tener una tensión de referencia. En este caso el valor de la tensión de referencia alta (Vref +) es de 3,3 V y el valor de la tensión de referencia baja (Vref -) es de 0 V. Un atributo importante del conversor es la resolución. En el microprocesador utilizado el conversor tiene una resolución de 10 bits. Con dicha resolución los valores convertidos variarán entre 0 para señal analógica de 0 V (Vref -) y 0x3FF (1023 en decimal) para 3,3 V (Vref +).

Un aspecto a considerar son los tiempos involucrados en la conversión. El tiempo de espera es el tiempo necesario que necesita el canal para que se cargue y se descargue completamente antes de empezar la siguiente conversión. Por otro lado, el tiempo total en una conversión es la suma de tiempo de muestro y tiempo de conversión. Los tiempos de conversión se miden en periodos. Para cada bit de resolución es necesario un periodo de muestro más dos periodos para terminar la conversión. Por lo tanto, para una conversión de 10 bits como es el caso son necesarios 12 periodos de tiempo.
El tiempo de muestreo depende de la velocidad de conversión configurada. Se ha escogido una velocidad razonable de 500 muestras por minuto (500 Ksps). Con esta velocidad de muestreo el tiempo de muestreo es de un periodo. Sumando el tiempo de muestreo y el tiempo de conversión da lugar a un tiempo total de conversión de 13 periodos.

El periodo de la conversión no coincide con el periodo de tiempo del oscilador (25 ns). Por ello será necesario introducir un pre-escalado para alcanzar el periodo deseado y conseguir conversiones adecuadas.

Los principales registros del módulo conversor A/D son: ADC1BUFx, AD1CON, AD1CSSL, AD1PCFGL, AD1CHS0 y registros para el control de interrupciones.

El registro ADC1BUFx es un registro de 16 bits donde se guarda el valor digital de la conversión.

El registro AD1CON son tres registros (AD1CON1, AD1CON2 y AD1CON3) desde donde se configuran los parámetros del conversor. Está compuesto por el esquema de bits de las figuras X, x y x.

	Bit 15	-	*Bit 7-5*	-	*Bit 2*	*Bit 1*	*Bit 0*
AD1CON1	ADON		SSRC		ASAM	SAMP	DONE

Figura 20. Esquema del registro AD1CON1.

- El bit ADON es el encargado de poner en marcha el convertidor, se pone a 1.
- Los bits SSRC configura cuando comienza la conversión teniendo en cuenta las distintas fuentes. Se ha configurado a 111, la conversión da lugar una vez realizados SAMC muestreos de adquisición.
- El bit ASAM configura el modo de conversión automática. Se configura a 0, modo no automático.
- El bit SAMP configura cuando muestrear y cuando convertir mientras SSRC esté a 0. No es nuestro caso, se configura a 0.
- El bit DONE es un bit de lectura que muestra 1 cuando la conversión a finalizado.

	-	***Bit 10***	*Bit 9-8*	-	*Bit 5-2*	
AD1CON2		CSCNA	CHPS		SMPI	

Figura 21. Esquema del registro AD1CON2.

- El bit CSCNA en caso de estar a 1 se realiza un barrido por cada pin que esté configurado en el registro AD1CSSL. Se configura a 0
  [1]
- El bit CHPS configura el canal del convertidor. Se configura a 0.

- El bit SMPI configura la cantidad de conversiones necesarios para hacer saltar la interrupción. Se configura a 0.

	-	***Bit 12-8***	*Bit 7-0*
AD1CON3		SAMC	ADCS

Figura 22. Esquema del registro AD1CON3.

- Los bits SAMC configuran la cantidad de ciclos de adquisición son necesarios antes de realizar la conversión en caso de configurar los bits SSRC a 111. Se configura a 1.

- Los bits ADCS configuran el pre-escalado mencionada anteriormente. Su valor sigue la siguiente fórmula. Se configura a 101 (5 en decimal).

$$T_{AD} = T_{CY} \, x \, (ADCS + 1)$$

El registro AD1PCFGL está formado por 6 bits que representan los canales analógicos del micro que van de AN0 a AN5. El registro configura que dicha entrada es analógica al poner su bit a 0. Se ha empleado la puerta analógica AN5, por tanto, se ha configurado el registro como 0x20 negado (bit 5 correspondiente a la entrada AN5).

El registro AD1CSSL El registro AD1PCFGL está formado por 6 bits que representan los canales analógicos del micro que van de AN0 a AN5. Se emplea para realizar escaneos. Se configura a 1 las entradas que se quieran escanear. No se emplea este uso en este proyecto.

El registro AD1CHS0 relaciona la entrada empleada con el muestreador empleado. Se emplea el muestreador 0 al que se le asocian los primeros 4 bits del registro. La entrada analógica empleada es la 5, por lo tanto, se configura el registro a 0x5.

En el proyecto se ha empleado un solo pin analógico, por ello se ha empleado el modo de conversión simple. Cada vez que se ha requerido conocer el valor de la entrada se ha dado la orden de inicio de la conversión, posteriormente se ha esperado al fin de la conversión y finalmente se ha transferido el valor al código principal.

3.2.3 UART

El módulo UART (Universal Asynchronous Receiver Transmitter) es el módulo de comunicación serie del microprocesador. La comunicación se rige por el protocolo RS-

232. A través de este módulo de comunicación es posible comunicar el micro con un servidor, otro micro o un módulo bluetooth entre otros. En este proyecto se emplea la UART para la comunicación con el display. Se detallará la comunicación en el siguiente capítulo.

La comunicación serie se ha establecido en los pines 11 y 14 del micro. Para ello se han remapeado los pines situando la transmisión de la UART (Tx) en el pin 11 y la recepción de la UART (Rx) en el pin 14.

El módulo UART tiene 5 principales registros: U1MODE, U1STA, U1TXREG, U1RXREG y U1BRG.

El registro U1MODE es un registro de 16 bits que su función principal es configurar el funcionamiento de la UART. Está compuesto por el esquema de la Figura 23.

Bits	*15*	-	*5*	-	*2-1*	*0*
	UARTEN		ABAUD		PDSEL	STSEL

Figura 23. Esquema bits registro U1MODE.

- El bit UARTEN es el encargado de poner en funcionamiento el módulo, se configura a 1.

- Los bits PDSEL son los encargados de establecer el número de bits de transmisión-recepción y la paridad. Se configura a 00, es decir, 8 bits de comunicación sin paridad.

- El bit STSEL configura el número de bits de stop que tiene la comunicación. Se configura a 0, un bit de stop.

El registro U1STA almacena los bits de estado de las comunicaciones del micro. Está compuesto por el esquema de la Figura 24.

Bits	*15*	-	*10*	*9*	*8*	*7-6*	-	*4*	*3*	*2*	*1*	*0*
	UTXISEL1		UTXEN	UTXBF	TRMT	URXISEL		RIDLE	PERR	FERR	OERR	URXDA

Figura 24. Esquema bits registros U1STA.

- El bit UTXISEL1 configura la interrupción de la transmisión. Se ha configurado a 1, en este caso salta la interrupción en el momento que los 4 buffers de transmisión están vacíos.

- El bit UTXEN activa la transmisión

- El bit UTXBF se activa en el caso de que el buffer de transmisión esté completo

- Los bits URXISEL configuran la interrupción de la recepción. Se ha configurado en 00. En este modo la interrupción se activa cuando hay un dato disponible.

- El bit RIDLE muestra el estado de la recepción.

- El bit PERR muestra error de paridad.

- El bit FERR muestra errores en los bits de start y stop.

- El bit OERR muestra error de sobrecapacidad del buffer de recepción.

- El bit URXDA muestra la existencia de algún dato en el buffer de la recepción.

El registro U1TXREG contiene el buffer de transmisión de la comunicación y el registro U1RXREG contiene el buffer de recepción de la comunicación.

El registro U1BRG establece el valor para configurar los baudios de comunicación del micro. Se ha establecido la comunicación en 9600 baudios, la frecuencia Fcy es de 39,61MHz. Su valor sigue la siguiente fórmula:

$$\text{U1BRG} = \frac{Fcy}{Baudios\ x\ 16} - 1$$

Una vez configurado el módulo el funcionamiento de la transmisión de datos es el siguiente: se crea un vector de cola de transmisión donde se almacenan los datos que se quieren enviar y una variable que indica el tamaño de dicho vector. Una vez rellenado el vector se hace saltar el flag de interrupción de la transmisión. Después comienza la subrutina de transmisión donde se transmite byte a byte lo registrado en el vector de cola de transmisión.

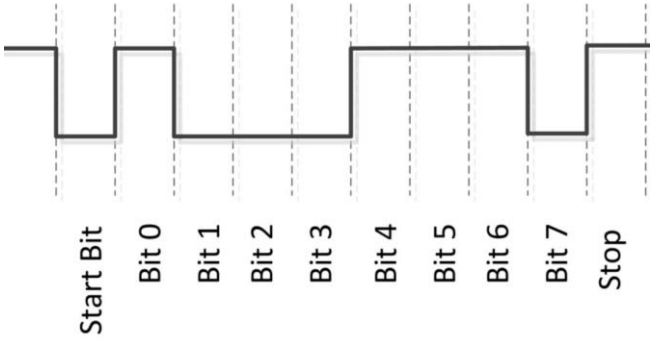


Figura 25. Ejemplo transmisión UART.

En la figura 25 observamos un ejemplo de transmisión como el empleado en el proyecto, 8 bits de transmisión, sin paridad y un solo bit de stop. En este ejemplo se está enviando el byte "10001110". La velocidad empleada es de 9600 baudios, esto significa 1/9600 segundos por cada bit transferido.

3.2.4 Programación del sistema

La programación del sistema se ha dividido en 7 partes:

1. Inicialización de los módulos (UART, Timer, convertidor A/D) y variables.
2. Programación de la selección del modo de uso del sistema por parte del usuario.
3. Rutina de arranque del ejercicio.
4. Gestión del sistema dependiendo del modo de uso.
5. Programación del Timer.
6. Programación de la gestión de la entrada analógica.
7. Programación de la gestión del display.

La inicialización de los módulos se ha realizado como se indican en los apartados 3.2.1, 3.2.2 y 3.2.3.

La programación de la selección del modo de uso se ha realizado a través de un solo pulsador. Se ha programado un detector de flanco que va sumando las veces que se presiona el pulsador, si la suma excede al número de opciones el sistema vuelve a empezar.

La rutina de arranque del ejercicio cuenta con una cuenta atrás de 9 segundos. Los 5 primeros segundos son de espera para que el usuario se posicione para comenzar el entrenamiento. Los últimos 4 segundos se realiza la cuenta atrás con el sonido del zumbador, produciente éste tres sonidos intermitentes de medio segundo y uno continuo de un segundo.

El sistema cuenta con dos modos de uso. El primero, enfocado a entrenamientos de fondo, muestra en el display el número de largos recorridos y el tiempo total de entrenamiento.

Para la gestión de es este modo el sistema detecta cuando el usuario ha realizado una vuelta a través de la entrada analógica y suma una unidad a la variable *largos*, registra el tiempo que se emplea durante el transcurso del entrenamiento y muestra por pantalla dichos datos. El segundo modo de uso, enfocado al ritmo de entrenamiento, el sistema registra del mismo modo que antes los largos recorridos y el tiempo empleado en realizar la última vuelta, resetea el cronometro en cada vuelta y muestra por pantalla dichos datos.

El Timer se ha configurado para que ejecute la rutina de interrupción cada 1ms, dentro de esta rutina se encuentra la lógica del cronometro. Se ha creado un sumatorio de milisegundos, al llegar a 1000 se resetea a 0 y suma una unidad al sumatorio de segundos. Cuando el sumatorio de segundos alcanza 60 se resetea a 0 y se suma una unidad al sumatorio de minutos. Cuando el sumatorio de minutos llega a 60 se resetea a 0 y se suma

una unidad al sumatorio de horas. También se emplea el Timer para la gestión del tiempo en la rutina de inicio de entrenamiento y para el uso de retardos.

La programación del convertidor A/D se ha llevado a cabo para interpretar la señal proveniente de la placa de toque. Para ello se han obtenido experimentalmente las siguientes medidas suponiendo linealidad en el convertidor:

Tipo de señal	Hexadecimal	Decimal	Señal
Contacto por toque	0x3C0	960	2,88V
Contacto por ruido	0x15E	350	1,02V

Tabla 2. Resultados señal placa de toque.

Para la detección de la placa de toque se ha filtrado la señal en el valor decimal de 750 que corresponde a una señal de 1,92V.

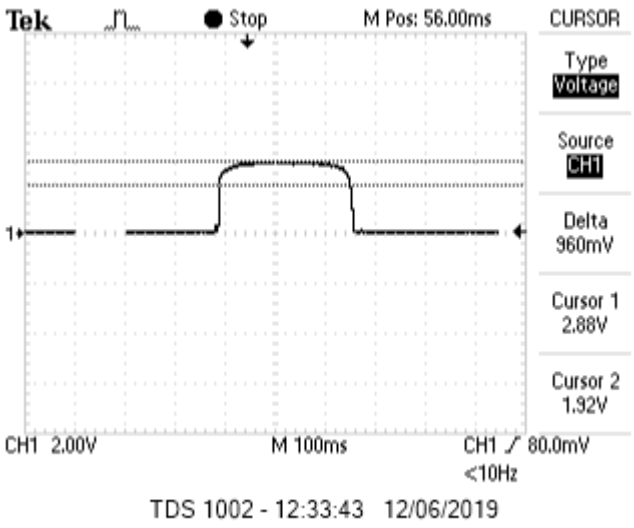


Figura 26. Señal del sensor en un contacto por toque captada por osciloscopio.

En la figura 26 se observa que la señal alcanza 2,88V, por encima del límite figado en 1,92V.

Por último, se ha programado el display. Para su programación se han creado funciones para facilitar su manejo. La gestión del display está directamente relacionada con el modo de uso seleccionado. En el primer modo de uso se muestran los datos de largos recorridos y el tiempo con el siguiente formato: h:mm:ss. Siendo h horas, m minutos y s segundos. En el segundo modo de uso se muestran los datos de largos recorridos y el tiempo empleado en realizar la última vuelta con el siguiente formato: m:ss.cc. Siendo m minutos, s segundos y c centésimas.

CAPÍTULO 4: VISUALIZACIÓN DE LOS DATOS

4.1 DISPLAY

El display seleccionado para este proyecto es el Nextion NX8048T070 de 7 pulgadas. Es una pantalla HMI (Human Machine Interface) que combina una pantalla TFT con un procesador y memoria. Se configura a través del software Nextion Editor desde donde se desarrollan objetos dentro del display (números, botones, gráficos, texto, etc.). La interacción del micro con la pantalla se realiza a través de la comunicación serie basándose en comandos codificados en ASCII.

Figura 27. Ejemplo de uso pantalla Nextion.

4.2 CONFIGURACIÓN DEL DISPLAY

La configuración de la pantalla se realiza a través del software Nextion Editor.

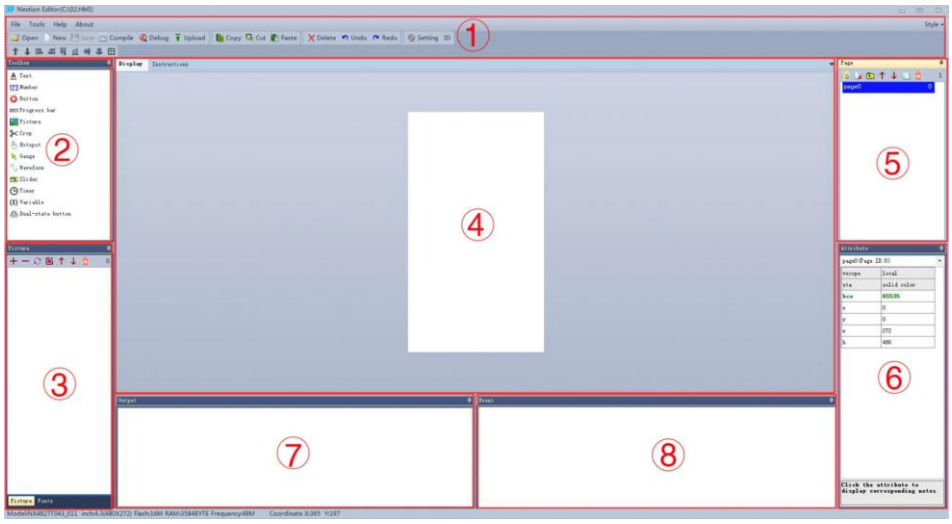


Figura 28. Interface del software Nextion Editor.

La configuración de la pantalla se realiza a través de los siguientes pasos:

1- Abrir un proyecto nuevo y seleccionar el modelo de pantalla NX8048T070.

2- Crear una fuente para los objetos donde intervengan dígitos, texto o símbolos. Se selecciona el tipo de fuente y su tamaño.

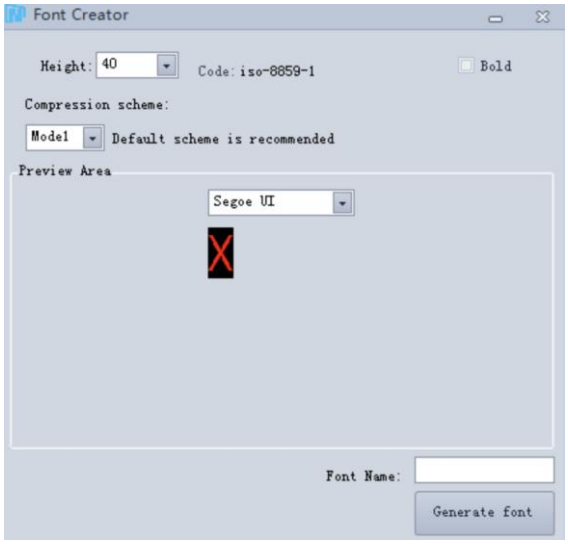


Figura 29. Generador de fuentes del software Nextion Editor.

3- Añadir objetos a la pantalla generada en la posición 4 de la Figura 28. Para ello se selecciona el tipo de objeto en la posición 2 de la misma figura y se arrastra a la pantalla dejándolo en el lugar que se desee.

4- Asignar una fuente a dichos objetos.

5- Añadir valores iniciales a dichos objetos.

6- Compilar el programa.

7- Implementarlo en el display. Al compilar el programa se genera un archivo .tft el cual se implementa en el display a través de una tarjeta de memoria micro SD.

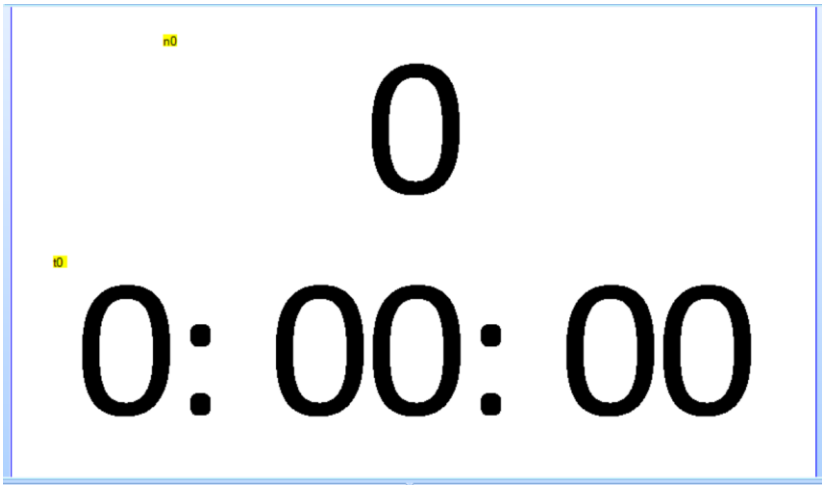


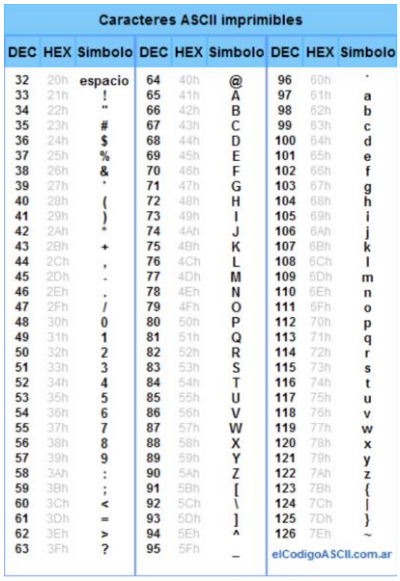
Figura 30. Configuración del display.

Se ha configurado el display como se observa en la Figura 30. Se ha creado un objeto numérico, que por defecto tiene la identidad de n0 con un valor inicial de 0. El objeto

numérico va a mostrar la cuenta de largos. Por otro lado, se ha añadido un objeto de texto, que por defecto tiene la identidad de t0. Se ha creado el objeto de texto para mostrar datos temporales. Para los datos temporales se ha escogido el objeto de texto para poder tener la estructura de tiempos con el tipo de separación deseado en cada momento.

4.3 GESTIÓN DEL DISPLAY

La gestión del display consiste en modificar el contenido de los objetos creados anteriormente en la configuración. La modificación de los objetos se realiza a partir de comandos codificados en ASCII transmitidos al display a través de la comunicación serie UART.



Caracteres ASCII imprimibles

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

elCodigoASCII.com.ar

Figura 31. Caracteres ASCII imprimibles.

La transmisión de los comandos sigue el siguiente protocolo: en un primer lugar se transmite el comando codificado en ASCII, en segundo lugar, el nuevo valor o texto que se desee modificar codificado en ASCII byte a byte y, por último, 3 bytes con el valor de 0xFF para indicar el fin de la instrucción.

Para la gestión de la pantalla se han empleado los siguientes 3 comandos:

1. Page 0. Este comando resetea los objetos del display a sus valores iniciales. La transmisión de esta instrucción es la siguiente:

Comando	p	a	g	e		0			
Transmisión	0x70	0x61	0x67	0x65	0x20	0x30	0xFF	0xFF	0xFF

2. N0.val=. Este comando modifica el valor del objeto numérico creado con la identidad n0. En caso de querer poner dicho valor a 1 la transmisión sería la siguiente:

n	0	.	v	a	l	=	1			
0x6E	0x30	0x2E	0x76	0x61	0x6C	0x3D	0x31	0xFF	0xFF	0xFF

3. T0.txt=”string”. Este comando modifica el contenido del objeto de texto creado con identidad t0. Se ha empleado de manera distinta en los modos de uso del sistema. Para el primer modo de uso el string tiene el siguiente formato: h:mm:ss, siendo h horas, m minutos y s segundos. En el segundo modo de uso el string tiene el siguiente formato: m:ss.cc. Siendo m minutos, s segundos y c centésimas. La transmisión se hace de la misma forma que los casos anteriores.

CAPÍTULO 5: RESULTADOS

Se ha probado el funcionamiento del sistema obteniendo resultados satisfactorios. El sistema de alimentación funciona correctamente, la placa de toque está correctamente calibrada, el display muestra los datos deseados. Pasamos a comprobar el uso de los modos de uso.

5.1 PLACA DE CONTACTO

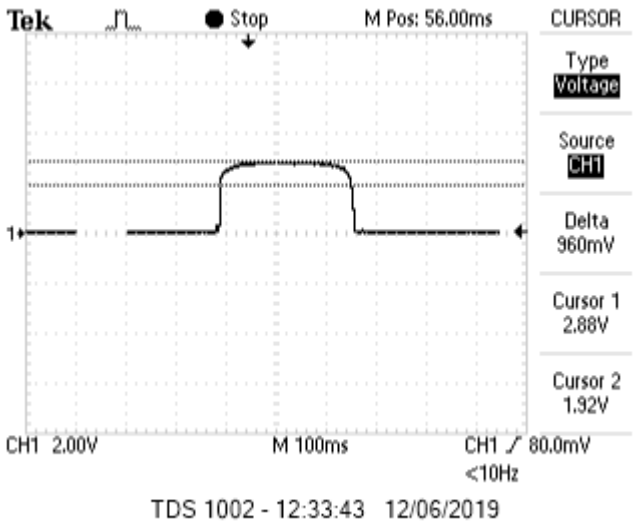


Figura 26. Señal del sensor en un contacto por toque captada por osciloscopio

En la figura 26 observamos el funcionamiento de la placa de contacto. Cuando el usuario haga contacto con la placa de toque el sensor variará su resistencia provocando una señal en la salida del circuito como la observada. La señal alcanza los 2,88V, superior a los 1,92V fijados como umbral de aceptación.

5.2 MODO DE USO 1

Se ha comprobado el funcionamiento del modo de uso 1 obteniendo los siguientes resultados.

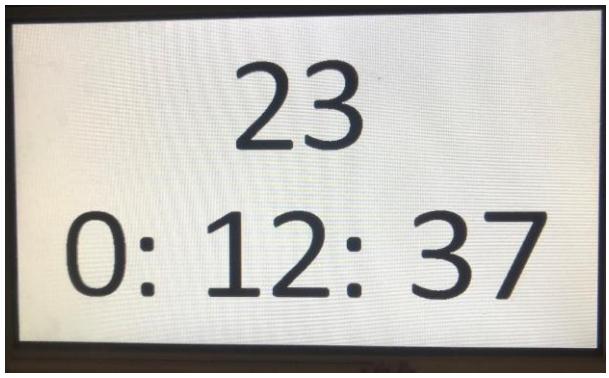


Figura 32. Display en un instante en modo de uso 1

En la figura 32 observamos el funcionamiento del modo de uso 1. Se visualizan los datos de largos y tiempo total de entrenamiento. En este instante se observa que el usuario lleva nadados 23 largos y han transcurrido 12 minutos y 37 segundos de entrenamiento.

5.3 MODO DE USO 2

Se ha comprobado el funcionamiento del modo de uso 2 obteniendo los siguientes resultados.

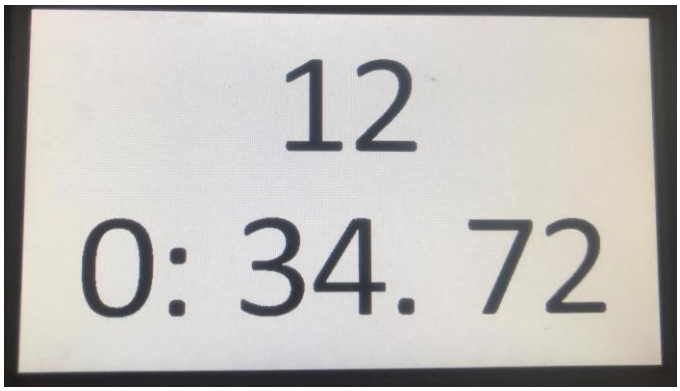


Figura 33. isplay en un instante en modo de uso 2.

En la figura 33 observamos el funcionamiento del modo de uso 2. Se visualizan los datos de largos y tiempo en el que el usuario a completado el último largo. En este instante se observa que el usuario lleva nadados 12 largos y el último largo lo ha realizado en 34 segundos y 72 centésimas.

CAPÍTULO 6: CONCLUSIONES

Este sistema permite al usuario conocer durante el transcurso del entrenamiento el número de vueltas, tiempo total de entrenamiento y tiempos parciales realizados. Conocer dichos datos ocasiona una mejora en la eficiencia del entrenamiento, así como aumentar el confort al no tener que memorizar la cuenta de largos y conocer el ritmo de nado.

Este sistema tiene funciones similares a dispositivos ya existentes en el mercado. Con este sistema se eliminan las carencias de las distintas soluciones ya creadas. Lo que hace este sistema único es la cómoda visualización de los datos, el precio económico y el manejo sencillo de la aplicación y su portabilidad.

Las pruebas de uso realizadas han sido satisfactorias. El sistema desarrollado interpreta la señal proveniente del sensor, gestiona la lógica correspondiente en función de la opción de uso seleccionada por el usuario y muestra los datos correspondientes con total claridad en el display.

Se han cumplido los principales objetivos del proyecto. Se ha realizado una aplicación intuitiva y sencilla en su uso. Se ha desarrollado un sistema ligero y portable con el fin de poder instalarlo donde se desee en cada momento. Por último, se ha desarrollado una cómoda visualización de los datos con un display de 7 pulgadas y alta calidad de visualización.

CAPÍTULO 7: REFERENCIAS BIBLIOGRÁFICAS

[1] Howard, M. M. (2017). *Sports Rec.* Recuperado de How do a swimming touch pads work?: https://www.sportsrec.com/553132-how-do-swimming-touch-pads-work.html

[2] Ibáñez, S. D. (2017). *Registrador electrónico para la práctica de natación en piscina*. Madrid: Universidad Pontificia Comillas.

[3] John. (2018). *Smartwatch Labs*. Recuperado de https://smartwatchlabs.com/waterproof-fitness-trackers-for-swimming/

[4] *Nadar bien*. (2019). Recuperado de https://nadarbien.com/wp-content/uploads/2016/12/reloj-acuatico-garmin-swim.jpg

[5] Ruiz, L. G. (2016). *Alge-Timing españa.* Recuperado de https://docplayer.es/6153429-Alge-timing-sistemas-de-natacion-v1-113.html

[6] *Variopool*. (s.f.). Recuperado de https://variopool.co.uk/products/swiss-timing-equipment/

[7] Carrion, L., Ochoa, D., Valverde, J.A. (2009). Análisis del funcionamiento del sensor de fuerza resistivo (fsr) con Labview. *Programa de Pasantías Academicas, UPS Cuenca.* Recuperada de http://www.datalights.com.ec/site2/images/stories/robotica/nap/nap_fsr.pdf

[8] Interlink Electronics, Inc. (2011) Interlink Electronics FSR Force Sensing Resistors. *Interlink electronics sensor technolgies. USA.*Recuperado de https://www.electronicoscaldas.com/datasheet/FSR Integration_Guide_Interlink.pdf

[9] Uart Basics. Ece353: introduction to microprocessor systems. Universyty of Wisconsin-Madison. Recuperado de: https://ece353.engr.wisc.edu/serial-interfaces/uart-basics/ Nextion Editor Quick Start Guide

[10] Nextion Editor Quick Start Guide. Itead. Recuperado de: https://www.itead.cc/wiki/Nextion_Editor_Quick_Start_Guide

[11] Sánchez Miralles, A. (2018). *Libro de texto: teoría y práctica.* Recuperado de https://sifo.comillas.edu/pluginfile.php/2260472/mod_resource/content/8/Libro%20SED%20v4.pdf

PARTE II MANUAL DE USO

CAPÍTULO 1 MODO DE USO

1.1 INSTALACIÓN DEL SISTEMA

1- Sitúe la caja de control en las inmediaciones de la pared de la piscina. La caja de control debe quedar en una situación que dé lugar a un fácil manejo estando el usuario en el agua.

2- Fije la placa de toque a la pared de la piscina. La placa de toque cuenta con cuatro ventosas en la parte posterior como instrumentos de fijación. Asegúrese de la correcta fijación y estabilidad.

3- Fije el display a la pared de la piscina. El display cuenta con cuatro ventosas en la parte posterior como instrumentos de fijación. Asegúrese de la correcta fijación y estabilidad.

1.2 MODOS DE USO

El sistema cuenta con 2 modos de usos:

Modo de uso 1: se visualizan los datos de largos y tiempo total de entrenamiento. Se muestran los datos de largos recorridos en la parte superior del display y el tiempo en la parte inferior del display con el siguiente formato: h:mm:ss. Siendo h horas, m minutos y s segundos.

Modo de uso 2: se visualizan los datos de largos y tiempo en el que el usuario a completado el último largo. Se muestran los largos recorridos en la parte superior del display y el tiempo empleado en realizar la última vuelta en la parte inferior del display con el siguiente formato: m:ss.cc. Siendo m minutos, s segundos y c centésimas.

1.3 USO DEL SISTEMA

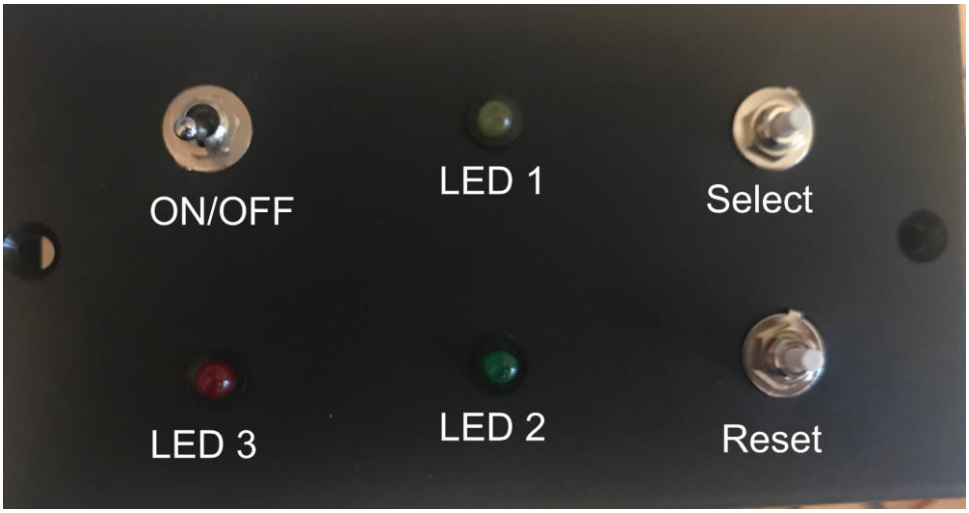


Figura 34. Caja de control.

1- Puesta en marcha: encienda el sistema accionando el interruptor (ON/OFF) en sentido ascendente, Observará que se enciende el LED 3 que indicará que el sistema está encendido.

2- Seleccione el modo de uso: presione el pulsador se selección de modo (Select) .Se iluminará el LED1 en caso de seleccionar el modo 1 o se iluminará el LED2 en caso de seleccionar el modo 2.

3- Comienzo del entrenamiento: una vez seleccionado el modo de uso dispone de 5 segundos de preparación para el inicio del entrenamiento. Una vez transcurridos los 5 segundos comenzará una cuenta atrás sonora de 3 segundos indicando la salida al finalizar.

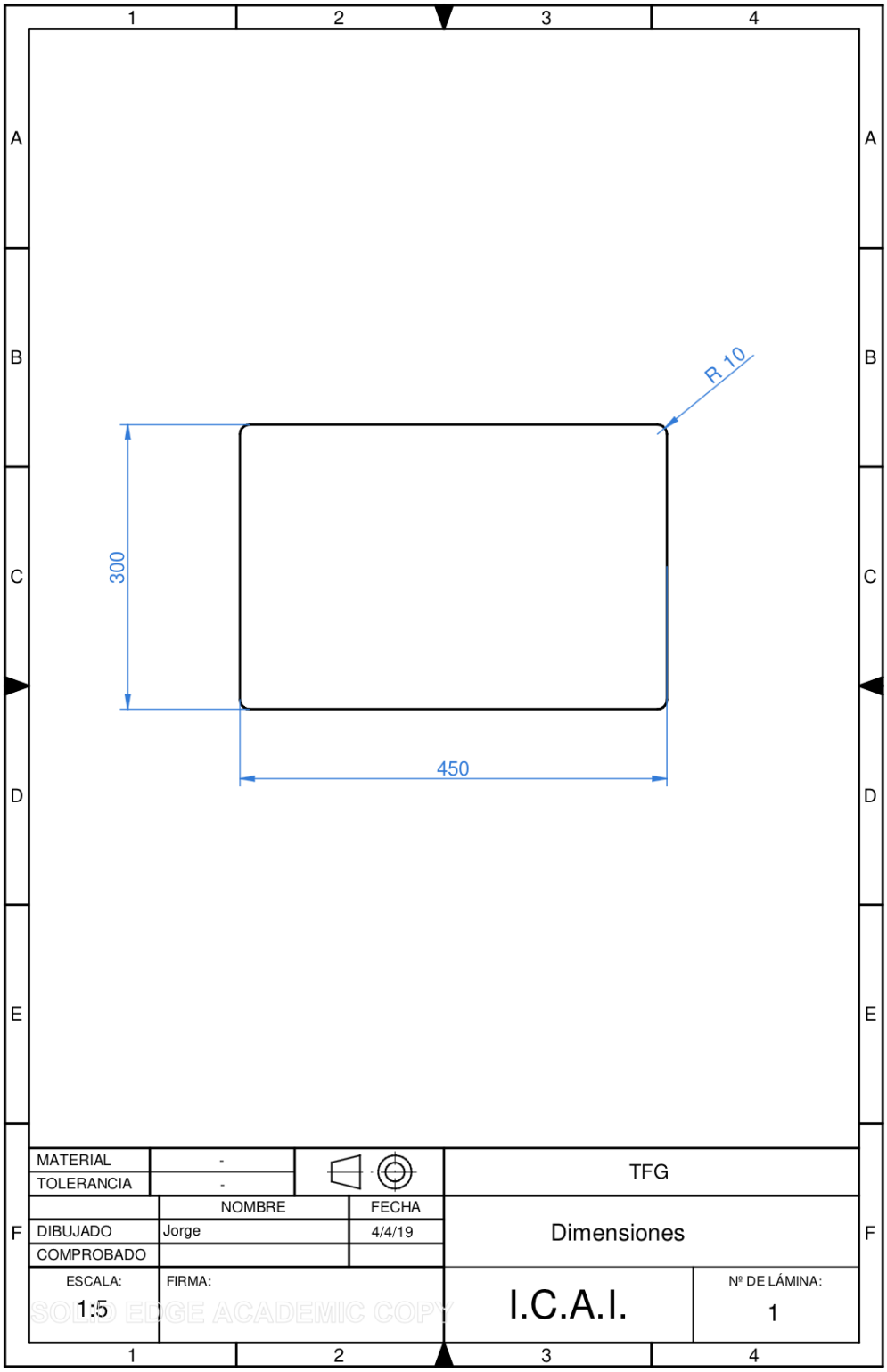
4- Cambio de modo de uso: durante la rutina de inicio de entrenamiento puede modificar la opción de modo de uso y volverá al punto 3.

5- Reset: para resetear el sistema pulse el pulsador de Reset y volverá al punto 2.

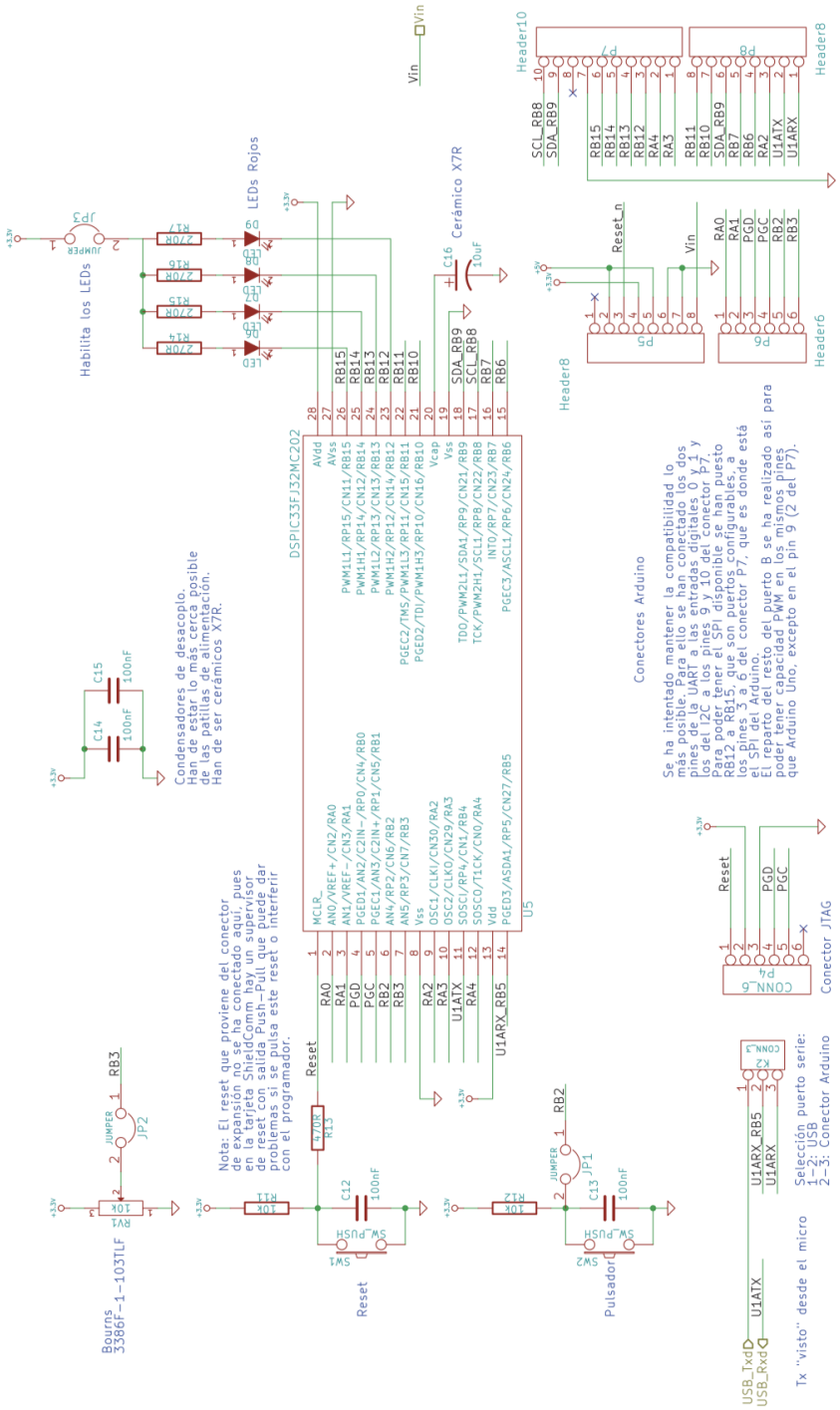
6- Apagado del sistema: para apagar el sistema accione el interruptor (ON/OFF) en sentido descendente. Observará que el LED 3 se apaga.

PARTE III PLANOS Y ESQUEMAS

CAPÍTULO 1: PLANOS FABRICACIÓN DE PLANCHAS PARA PLACA DE TOQUE



CAPÍTULO 2: ESQUEMA PLACA DE DESARROLLO DEL MICROCONTROLADOR DSPIC33FJ32MC202



PARTE IV ESTUDIO ECONÓMICO

CAPÍTULO 1. PRESUPUESTO

1.1 HARDWARE

Hardware	Ud.	Precio Ud. (€)	Precio total (€)
Pulsadores	2	0,3	0,6
Leds	3	0,02	0,06
Conmutador	1	0,8	0,8
Cableado	sn	1	1
Micro con placa	1	40	40
Placas pvc	2	5	10
Display	1	80	80
Sensor fsr	1	2,75	2,75
Pila 9v	1	3	3
Resistencias	7	0,02	0,14
Presnsaestopas	3	0,02	0,06
Caja estanca	1	3	3
Ventosas	8	1	8
TOTAL			**149,41**

Tabla 3. Desglose precios hardware.

1.2 SOFTWARE

Software	Ud.	Precio Ud. (€)	Precio total (€)
Nextion editor	1	0	0
Mplab	1	0	0
Microsoft office	1	150	150
TOTAL			**150**

Tabla 4. Desglose precios software

1.3 HERRAMIENTAS

Herramientas	Ud.	Precio ud. (€)	Precio total (€)
Ordenador	1	600	600
Soldador	1	300	300
Taladro	1	100	100
Polímetro	1	40	40
OTROS	Sn	50	50
TOTAL			**1090**

Tabla 5. Desglose precios herramientas.

1.4 MANO DE OBRA

Mano de obra	Horas	Precio hora	Precio total (€)
Desarrollo	80	12	960
Producción	20	12	240
TOTAL			**1200**

Tabla 6. Desglose precios mano de obra.

1.4 PRESUPUESTO GENERAL

Concepto	Precio total (€)
Hardware	149,41
Software	150
Herramientas	1090
Mano de obra	1200
TOTAL	**2589,41**

Tabla 7. Presupuesto general.

CAPÍTULO 2. ESTUDIO DE VIABILIDAD ECONÓMICA

El objetivo de este capítulo es llevar a cabo el estudio de la rentabilidad económica del sistema diseñado. Para ellos se estudian los costes de producción, la situación de mercado y el precio de venta.

2.1 CÁLCULO DE COSTE DE PRODUCCIÓN

El coste de producción total se ha calculado como la suma de la inversión inicial más la suma del coste por unidad.

La inversión inicial recoge los conceptos de herramientas, software y mano de obra de desarrollo. Suma un total de 2200€.

El coste por unidad recoge los conceptos de hardware y mano de obra por producción. Suma un total de 389,41 €.

El coste de producción total en función de las unidades producidas sigue la siguiente fórmula: 2200+ 389,41xUd.

2.2 PRECIO DE VENTA.

Teniendo en cuenta los costes de producción, la situación de mercado y el objetivo de diseñar una solución económica se fija un precio de venta de 430€. Con este precio de venta llegamos a los siguientes resultados:

Margen bruto del producto: $\frac{430}{389,41} x100 = 10,4\%$

Unidades vendidas necesarias para rentabilidad del proyecto:

2200+ 389,41xUd. = 430xUd. Ud.= 55 unidades

A partir de 55 unidades vendidas el VAN es positivo, siendo así viable el proyecto.

PARTE V CÓDIGO

CAPÍTULO 1: LIBRERÍAS EMPLEADAS

Para el desarrollo del sistema se han empleado las siguientes librerías:

- Config.h: recoge funciones para configurar el micro.
- Ad.h: recoge funciones para el manejo y configuración del módulo conversor A/D.
- Display.h: recoge funciones para el manejo del display.

1.1 CONFIG.C

El driver de configuración recoge las funciones necesarias para una configuración inicial del micro: InicializarReloj y RemapeaPeriféricos.

```
#include "config.h"



// ------------------------------------------------------------------------
--------------
// ----------------------------------- FUNCIONES ----------------------
--------------
// ------------------------------------------------------------------------
--------------
```

1.1.1 Función InicializarReloj

La función InicializarReloj configura la frecuencia del oscilador para que el micro funcione a 40 MIPS (Millions of Instructions per Second).

```
/* Nombre: InicializarReloj
 * Descripción: Inicializa el reloj interno FRC para que funcione con
PLL.
 * Argumentos: Ninguno
 * Valor devuelto: Ninguno
 */

void InicializarReloj(void)
 {
// Configurar frecuencia del oscilador FRC (FOSC), cuya frecuencia
nominal
// (Fin) son 7.37 MHz, para que el microprocesador opere a 40 MIPS (FCY)
// FOSC = Fin * M/(N1*N2) FCY = FOSC/2
// FOSC = 79.2275 MHz FCY = 39.61375 MHz
CLKDIVbits.PLLPRE  =  0;       // Preescalado del PLL:   N1 = 2
PLLFBD             = 41; // Multiplicador del PLL: M = PLLFBD + 2 = 43
CLKDIVbits.PLLPOST =  0; // Postescalado del PLL:  N2 = 2
```

```
OSCTUN   = 21; // Fin = 7.37 MHz + OSCTUN * 30 kHz = 8 MHz
CLKDIVbits.PLLPRE  =  0; // Preescalado del PLL:   N1 = 2
PLLFBD   = 40; // Multiplicador del PLL: M = PLLFBD + 2 = 43
CLKDIVbits.PLLPOST =  0; // Postescalado del PLL:  N2 = 2
*/

// Funciones para desbloquear la escritura del registro OSCCON
__builtin_write_OSCCONH(0x01); // Nuevo reloj: FRC w/ PLL
__builtin_write_OSCCONL(OSCCON | 0x01);       // Iniciar el cambio de
reloj

while (OSCCONbits.COSC != 1);  // Esperar a que se produzca el cambio de
reloj
while (OSCCONbits.LOCK != 1);  // Esperar a que se sincronice el PLL
}
```

1.1.2 Función RemapeaPerifericos

La función RemapeaPerifericos configura el conexionado de pines del micro con los distintos periféricos. Configura la transmisión de la UART en el pin 11 y la recepción en el pin 14.

```
/* Nombre: RemapeaPerifericos
 * Descripción: Situa en los pines adecuados los perifericos del micro
 * Argumentos: Ninguno
 * Valor devuelto: Ninguno
 */
void RemapeaPerifericos(void)
 {

// Funciones para desbloquear la escritura del registro OSCCON
__builtin_write_OSCCONL(OSCCON & 0xBF);       // Desbloquea el PPS

        RPINR18bits.U1RXR = 5; // Asigna U1RX al pin 14 que es RP5
        RPOR2bits.RP4R = 3; // Asignar U1TX al pin 11 que es RP4

        __builtin_write_OSCCONL(OSCCON | 0x040); // Bloquea los PPS

}
```

1.2 AD.C

El driver del conversor analógico-digital recoge las funciones necesarias para la configuración y manejo del mismo. Incluye las funciones: init_ad y get_ad.

1.2.1 Función init_ad

La función init_ad inicializa el convertidor analógico-digital y configura los pines de entrada analógicos del micro.

```
#include "ad.h"
void init_ad(int pines){
TRISB |= pines & 0x3F >> 2;//Configura pines de entrada
TRISA |= pines & 3;//Configura pines de entrada
AD1PCFGL = ~pines;//Configura que las entradas son analógicas
AD1CON3 = 0x105; // SAMC = 1, ADCS = 5 -> 1 ciclo de muestreo
AD1CON2 = 0;
AD1CON1 = 0x80E0; // ADON, SSRC = 111
}
```

1.2.2 Función get_ad

La función get_ad ejecuta la rutina de lectura del conversor y devuelve el valor de lectura.

```
int get_ad(int pin){
AD1CHS0 = pin; // selecciona pin de conversión
IFS0bits.AD1IF = 0;//Baja flag
AD1CON1bits.SAMP = 1; // comienza
while (!IFS0bits.AD1IF); // conversión terminada?
return ADC1BUF0; //Devuelve lectura
}
```

1.3 DISPLAY.C

El driver del display recoge las funciones necesarias para un cómodo manejo del mismo. Incluye las funciones: EnviarLargos, EnviarTiempo1, EnviarTiempo2 y ResetDisplay.

1.3.1 Función EnviarLargos

La función EnviarLargos envía al display el comando para actualizar con el valor deseado el campo destinado a la cuenta de largos. Para ello recibe el valor deseado, rellena el vector de transmisión con el comando correspondiente y activa el flag de transmisión.

```
#include "display.h"

/*Funcion: EnviarLargos
 *Autor: Jorge Zumarraga
 *Fecha: 10/05/2019
 *Descripcion: Rellena vector de transmision con el
```

```
 *numero de largos
 *Comando a enviar:n0.val=largos
 */
void EnviarLargos(int x)
{
    //Variables locales
    int cen_largos;//Centenas de largos
    int dec_largos;//decenas de largos
    int un_largos;//Unidades de largos

    //Calculo de variables
    cen_largos=x/100;
    dec_largos=(x-(cen_largos)*100)/10;
    un_largos=(x-(cen_largos)*100)-(dec_largos*10);

    //rellenas vector transmision
    ui_icola_tr=0;
    uc_cola_transmision[ui_icola_tr] = 0x6E;//n
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x30;//0
    ui icola tr++;
    uc_cola_transmision[ui_icola_tr] = 0x2E;//.
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x76;//v
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x61;//a
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x6C;//l
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x3D;//=
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+cen_largos);//centenas
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+dec_largos);//decenas
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+un_largos);//unidades
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;//3 bytes fin de transmision
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    IFS0bits.U1TXIF = 1;//Activa flag de transmision
}
```

1.3.2 Función EnviarTiempo1

La función EnviarTiempo1 envía al display el comando para modificar el campo de tiempo con el formato del modo de uso 1. Para ello accede a las variables globales de tiempo, rellena el vector de transmisión con los bytes correspondientes y activa el flag de transmisión.

```
/*Funcion: EnviarTiempo1
 *Autor: Jorge Zumarraga
```

```
 *Fecha: 17/05/2019
 *Descripcion: Rellena vector de transmision con el
 *formato de opcion1 activa el flag de transmision
 *Comando a enviar:t0.txt="h:mm:ss"
 */
void EnviarTiempo1()
{

    //Variables locales
    int dec_min;//Decenas de minutos
    int ud_min;//Unidades de minutos
    int dec_seg;//Decenas de segundos
    int ud_seg;//Unidades de segundos
    dec_seg=contador_seg/10;
    ud_seg=contador_seg-(dec_seg*10);
    dec_min=contador_min/10;
    ud_min=contador_min-(dec_min*10);

    //Rellenar vector transmision
    ui_icola_tr=0;
    uc cola transmision[ui icola tr] = 0x74;//t
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x30;//0
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x2E;//.
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x74;//t
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x78;//x
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x74;//t
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x3D;//=
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x22;//"
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+contador_hora);//horas
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x3A);//:
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+dec_min);//decenas minutos
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+ud_min);//unidades minutos
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x3A);//:
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr]     =     (0x30+dec_seg);//decenas
segundos
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr]     =     (0x30+ud_seg);//unidades
segundos
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x22;//:
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr]   =   0xFF;//3   bytes   de   fin   de
transmision
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
```

```
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    IFS0bits.U1TXIF = 1;//Activa flag transmision
}
```

1.3.3 Función EnviarTiempo2

La función EnviarTiempo2 envía al display el comando para modificar el campo de tiempo con el formato del modo de uso 2. Para ello accede a las variables globales de tiempo, rellena el vector de transmisión con los bytes correspondientes y activa el flag de transmisión.

```
/*Funcion: EnviarTiempo2
 *Autor: Jorge Zumarraga
 *Fecha: 18/05/2019
 *Descripcion: Rellena vector de transmision con el
 *formato de opcion 2 y activa el flag de transmision
 *Comando a enviar:t0.txt="m:ss.cc"
 */
void EnviarTiempo2()
{
    //Variables locales
    int dec_min;//decenas de minutos
    int ud_min;//unidades de minutos
    int decimas;//decimas de segundo
    int dec_seg;//decenas de segundos
    int ud_seg;//unidades de segundos
    int centesimas;//centesimas de segundos
    //Calculo de variables
    decimas=contador_miliseg/100;
    centesimas=(contador_miliseg-decimas*100)/10;
    dec_seg=contador_seg/10;
    ud_seg=contador_seg-(dec_seg*10);
    dec_min=contador_min/10;
    ud_min=contador_min-(dec_min*10);

    //Rellenar vector transmision
    ui_icola_tr=0;
    uc_cola_transmision[ui_icola_tr] = 0x74;//t
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x30;//0
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x2E;//.
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x74;//t
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x78;//x
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x74;//t
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x3D;//=
    ui_icola_tr++;
```

```
    uc_cola_transmision[ui_icola_tr] = 0x22;//"
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+ud_min);// unidades min
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x3A);//:
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+dec_seg);//decenas de seg
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+ud_seg);//ud de seg
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x2E);//.
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+decimas);//decimas
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = (0x30+centesimas);//centesimas
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x22;//"
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;//3 bytes fin de transmision
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    IFS0bits.U1TXIF = 1;//Activa flag transmision
}
```

1.3.4 Función ResetDisplay

La función ResetDisplay envía al display el comando correspondiente al reseteo del mismo. Para ello rellena el vector de transmisión con los bytes adecuados y activa el flag de transmisión.

```
/*Funcion: ResetDisplay
 *Autor: Jorge Zumarraga
 *Fecha: 19/05/2019
 *Descripcion: Rellena vector de transmision con el
 *comando para resetear el display
 *Comando a enviar: page 0
 */
void ResetDisplay()
{
    ui_icola_tr=0;
    uc_cola_transmision[ui_icola_tr] = 0x70;//p
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x61;//a
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x67;//g
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x65;//e
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x20;//espacio
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0x30;//0
    ui_icola_tr++;
```

```
    uc_cola_transmision[ui_icola_tr] = 0xFF;//3 bytes fin de transmision
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    uc_cola_transmision[ui_icola_tr] = 0xFF;
    ui_icola_tr++;
    IFS0bits.U1TXIF = 1;//Activa flag transmision
}
```

CAPÍTULO 2: PROGRAMA PRINCIPAL

2.1 INICIALIZACIONES

En primer lugar, se ejecutan las inicializaciones correspondientes para la configuración del micro. Después, se declaran las funciones que se van a ejecutar en el main y por último se declaran las variables globales.

```
#include "config.h"
#include "p33FJ32MC202.h"
#define BAUDRATE 9600//Define velocidad comunicacion

// ------------------------------------------------------------------------
--------------
// ----------------------------- BITS DE CONFIGURACI?N ----------------
--------------
// ------------------------------------------------------------------------
--------------

// 1. Eliminar el segmento de arranque flash
// 2. Permitir la escritura del segmento de arranque flash
_FBS(BSS_NO_BOOT_CODE & BWRP_WRPROTECT_OFF);

// 1. No proteger la memoria de programa contra escritura
// 2. No proteger el cÛdigo
_FGS(GWRP_OFF & GCP_OFF);

// 1. Utilizar el oscilador interno (FRC) en el arranque
// 2. Arrancar directamente con el oscilador seleccionado
_FOSCSEL(FNOSC_FRC & IESO_OFF);

// 1. Permitir la conmutaciÛn del reloj y deshabilitar la monitorizaciÛn
de fallos
// 2. Desactivar el oscilador primario
// 3. Utilizar el oscilador secundario como entrada y salida digital
// 4. Permitir m?ltiples remapeos de los pines
_FOSC(FCKSM_CSECMD & POSCMD_NONE & OSCIOFNC_ON & IOL1WAY_OFF);

// 1. Permitir la deshabilitaciÛn del watchdog timer
//    poniendo a 0 el bit SWDTEN del registro RCON
_FWDT(FWDTEN_OFF);

// 1. Esperar 128 ms y resetear el microcontrolador al enchufar la
alimentaciÛn
// 2. Controlar los pines de PWM desde el registro PORT al arrancar
// 3. Los pines PWM high est·n activos a nivel alto
// 4. Los pines PWM low est·n activos a nivel alto
// 5. Utilizar los pines est·ndar (SDA1 y SCL1) para el I2C
_FPOR(FPWRT_PWR128 & PWMPIN_ON & HPOL_ON & LPOL_ON & ALTI2C_OFF);

// 1. Programar y depurar a travÈs de los pines PGC1 y PGD1
// 2. Desactivar el interfaz para JTAG
_FICD(ICS_PGD1 & JTAGEN_OFF);
```

```
#include "config.h"
#include "p33FJ32MC202.h"
#define BAUDRATE 9600//Define velocidad comunicacion

///DECLARACION FUNCIONES
void Reset_crono();
void Retardo(int r);
///DECLARACION VARIABLES GLOBALES
int retardo=0;//cuenta el retardo en milisegundos
int contador_miliseg = 0;//Cuenta milisegundos
int contador_seg = 0;//Cuenta segundos
int contador_min=0;//Cuenta minutos
int contador_hora=0;//Cuenta horas
int contadorf=0;//Contador filtro pulsador
int contadorfad=0;//Contador filtro A/D
unsigned int ui_icola_tr = 0;//Situacion cola transmision
unsigned int ui_icabeza_tr=0;//Situacion cabeza transmision
char uc_cola_transmision[30];//Vector transmision
```

2.2 MAIN

En el main en un primer lugar se realizan las configuraciones del micro, timer y conversor A/D. Después se declaran las variables locales y por último se introduce la lógica del sistema en un bucle infinito.

```
int main(void) {
    InicializarReloj();//Llamada a funcion InicializarReloj
    RemapeaPerifericos();//Llamada a funcion RemapeaPerifericos
    PORTB = 0x0FFF;//Incicializacion puerto b
    TRISB = 0x0FFF;//12 primeros bits entradas, 4 ultimos salidas
    TMR1=0;//Contador timer a 0
    PR1=0x9C40; // cuenta hasta 40000 (1ms con pre-escalado 0)
    IFS0bits.T1IF=0;//Flag timer a 0
    IEC0bits.T1IE=1;//Habilita interrupcion timer
    T1CON=0x8000;//Activa modulo timer
    InicializarUART();//Llamada a funcion InicializarUART
    ResetDisplay();//Llamada a funcion ResetDisplay
    init_ad(0x20); //Configura An5 como analogico

    ///VARIABLES LOCALES
    int dato ;// Varible lectura convertidor A/D
    int op=0;// Variable opcion de uso
    int estadoant=1;//Variable 1 detector de flanco
    int estadoact=1;//Variable 2 detector de flanco
    int A=0;//mascara
    int op_ant=0; // Varable opcion anterior
    int Cuenta=0; //Variable cuenta
    int estadoant_reset=1; //Detector flanco reset
    int estadoact_reset=1; //Detector flanco reset
    int largos=0; //Variable contador largos
    int filtroad=0; //variable para filtro A/D
    int filtrod=1;// Variable para filtro pulsador
    int reset=1;//Incicializa reset a 1
    int inicio=0;//Variable inicio de entrenamiento
```

2.2.1 Bucle infinito

El bucle infinito está dividido en cuatro partes. En la primera parte está recogida la lógica del reseteado del sistema. En la segunda parte se realiza la lógica de selección del modo de uso. En la tercera parte se sitúa la rutina de inicio de entrenamiento con la cuenta atrás y, por último, se realiza la lógica del entrenamiento en función del modo de uso escogido.

```
///BUCLE INFINITO
   while (1)
   {
        //RESET: detector de flanco con pulsador en Rb6
        //reseta variables op,inicio,largos y el display
        estadoant_reset=estadoact_reset;
        estadoact_reset=(PORTB&0x40);
        if((estadoact_reset!=estadoant_reset))
        {
            if(estadoact_reset==0)
            {
                ResetDisplay();
                op=0;
                inicio=0;
                PORTB=0x0FFF;
                largos=0;
            }
        }

        dato = get_ad(5); //Lee conversor A/D en pin AN5

////////////////////////////////////////////
//LOGICA SELECCION DE OPCIONES DE USO////////
////////////////////////////////////////////
//Algoritmo de detector de flanco en el pulsador
//Cuando detecta flanco suma 1 a la variable opcion

        estadoant=estadoact;
        estadoact=(PORTB&4);
        if((estadoact!=estadoant))
        {
            if(estadoact==0)
            {
                if(filtrod==0)
                {
                    filtrod=1;
                    contadorf=0;
                    op++;
                }
            }
        }
        if(contadorf==200)
        filtrod=0;

        if(op>2)//Vuelve a 0 cuando supera 2
        {
            op=0;
            PORTB=PORTB&0x3FFF;
```

```
        }

        if(op==1)//Enciende LED1
        PORTB=PORTB|0x8000;

        if(op==2)//Enciende LED2
        {
            A=PORTB&0x3FFF;
            PORTB=A|0x4000;
        }
////////////////////////////////////////////
// Rutina INICIO ENTRENAMIENTO cuenta atras//
////////////////////////////////////////////
//Cuenta 5 segundos, luego comienza la cuenta
//atras sonora de 3 segundos y señala el inicio
////////////////////////////////////////////

        if(op!=op_ant)//Reinicia en caso de cambiar de opcion
        {
            Cuenta=0;
            Reset_crono();
            if(op!=0)
            Cuenta=1;
            op_ant=op;
        }

        if(Cuenta==1)
        {
            if(contador_seg<5)
            PORTB=PORTB&(~0x2000);

            if(contador_seg>=5)
            {
                if(contador_seg<=7)
                {
                    if(contador_miliseg==1)
                    PORTB=PORTB|0x2000;

                    if(contador_miliseg==500)
                    PORTB=PORTB&(~0x2000);
                }

            }
            if(contador_seg==8)
            PORTB=PORTB|0x2000;

            if(contador_seg==9)
            {
                    Cuenta=0;
                    PORTB=PORTB&(~0x2000);
                    inicio=1;
                    Reset_crono;
            }
        }else{
                PORTB=PORTB&(~0x2000);
             }
////////////////////////////////////////
////Logica durante entrenamiento///////
////////////////////////////////////////
```

```
//Si la opcion es 1 envia tiempo en el modo de envio 1
//Si la opcion es 2 envia tiempo en el modo de envio 2
//Los largos se envian en ambos modos de uso
//Se añaden retardos de 20 milisegundos para no saturar
//el display
        if(inicio==1)
        {
            if(op==1)
            {
                Retardo(20);
                EnviarTiempo1();
            }
            if(dato>500)//Detecta llegada nadador
            {
                if(filtroad==0)
                {
                    PORTB=0x0FFF;
                    filtroad=1;
                    contadorfad=0;
                    largos++;//suma largo
                    if(op==2)
                    {
                        EnviarTiempo2();//Envia tiempo
                        Reset_crono;//Resetea crono
                    }
                    Retardo(20);//retardo
                    EnviarLargos(largos);//envia largos
                }

            }

            if(contadorfad==1000)
            filtroad=0;
        }

    }
    return 0;
}
```

CAPÍTULO 3: INTERRUPCIÓN TIMER 1

El timer está configurado en modo interrupción con una cuenta de 1 milisegundo. En primer lugar, se baja el flag de interrupción para que el timer comience a ejecutar la siguiente cuenta. En segundo lugar, se actualizan variables relacionadas con la cuenta de milisegundos y, por último, se realiza la lógica del cronometro contando milisegundos, segundos, minutos y horas.

```
/*Rutina interrupcion timer 1
 *Sucede cada 1 milisegundos, incluye logica de cronometro
 *cuenta milisegundos, segundos, minutos y horas
 *tambien lleva cuenta de los filtros A/D y del pulsador
 */
void __attribute__((interrupt, no_auto_psv)) _T1Interrupt(void)
{
    IFS0bits.T1IF=0;
    contador_miliseg++;
    contadorf++;
    contadorfad++;
    retardo++;

    if(contador_miliseg==1000)
    {
        contador_miliseg=0;
        contador_seg++;

    }
    if(contador_seg==60)
    {
        contador_seg=0;
        contador_min++;
    }
    if(contador_min==60)
    {
        contador_min=0;
        contador_hora++;
    }

return ;
}
```

CAPÍTULO 4: MÓDULO UART

El módulo UART se configura con la función InicializarUart. La transmisión y recepción se ejecutan dentro de las rutinas de interrupción correspondientes.

4.1 FUNCIÓN INICIALIZARUART

La función InicializarUart recoge la configuración del módulo UART. El módulo se configura para un modo de uso por interrupciones con una comunicación de 8 bits con un bit de stop y sin paridad.

```
void InicializarUART(void) {


    TRISB |= 0x20; // Configura RX como entrada
    TRISB &= 0xFFEF; // Configura TX como salida
    U1BRG = (FCY / BAUDRATE) / 16 - 1;//Configura velocidad de
comunicacion
    U1MODEbits.STSEL = 0;//Configura 1 bit de stop
    U1MODEbits.PDSEL = 0;//Configura 8 bits de comunicacion sin paridad
    U1MODEbits.ABAUD = 0;
    U1MODEbits.UARTEN = 1;//Pone en funcionamiento el modulo
    U1STAbits.UTXISEL1 = 1;//Configura interrupcion transmision
    U1STAbits.URXISEL = 0;//Configura interrupcion recepcion
    U1STAbits.UTXEN = 1;//Activa transmision
    IFS0bits.U1TXIF = 0;//Flag transmision a 0
    IFS0bits.U1RXIF = 0;//Flag recepcion a 0
    IEC0bits.U1TXIE = 1;//Habilita interrupcion transmision
    IEC0bits.U1RXIE = 1;//Habilita interrupcion recepcion
}
```

4.2 RUTINA DE INTERRUPCIÓN DE TRANSMISIÓN

En la interrupción de la transmisión se baja el flag de interrupción y se rellena el registro de transmisión byte a byte con el vector a transmitir. Cada vez que se rellena el registro de transmisión vuelve a comenzar la rutina ejecutando la transmisión y liberando el buffer, de esta manera no se satura el buffer de transmisión.

```
/*Rutina interrupcion transmision UART
 */
void __attribute__((interrupt,no_auto_psv)) _U1TXInterrupt(void)
{
IFS0bits.U1TXIF = 0; // Borrar la bandera de la interrupción
if(ui_icola_tr > ui_icabeza_tr )//Comprueba si hay algo que enviar
{
    U1TXREG = uc_cola_transmision[ui_icabeza_tr];//Rellena buffer
    ui_icabeza_tr++;//Pasa al siguiente
    if (ui_icola_tr == ui_icabeza_tr)//Fin de la transmision
    {
```

```
        ui_icola_tr = 0; //Variable a 0
        ui_icabeza_tr = 0; //Variable a 0
    }
}
}
```

4.3 RUTINA INTERRUPCIÓN RECEPCIÓN

En la interrupción de la recepción únicamente se baja el flag de interrupción.

```
/*Rutina interrupcion recepcion UART
 */
void __attribute__((interrupt, no_auto_psv)) _U1RXInterrupt(void) {
    IFS0bits.U1RXIF = 0; // Borrar la bandera de la interrupcion
}
```

CAPÍTULO 5: OTRAS FUNCIONES

5.1 FUNCIÓN RETARDO

La función Retardo recibe el valor del retardo deseado en milisegundos y ejecuta un bucle con la duración de dicho retardo.

```
/*Funcion: Retardo
 *Autor: Jorge Zumarraga
 *Fecha: 17/04/2019
 *Descripcion: Realiza un retardo de los milisegundos indicados
 */
void Retardo(int r)
{
    retardo=0;
    while(retardo<r);
    return;
}
/*Funcion: Retardo
 *Autor: Jorge Zumarraga
 *Fecha: 15/04/2019
 *Descripcion: Resetea el cronometro a 0
 */
```

5.2 FUNCIÓN RESET_CRONO

La función Reset_crono resetea las variables correspondientes al registro del tiempo.

```
void Reset_crono()
{
    contador_miliseg=0;
    contador_seg=0;
    contador_min=0;
    contador_hora=0;
    return;
}
```

PARTE VI DATASHEETS

- Microcontrolador dsPIC33FJ32MC202: http://ww1.microchip.com/downloads/en/devicedoc/70283k.pdf

- Sensor FSR 406: https://www.electronicoscaldas.com/datasheet/FSR400Series_Interlink.pdf

- Nextion NX8048T070: https://www.itead.cc/wiki/NX8048T070