Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Trabajo Fin de Grado

Ingeniería en Electrónica Industrial y Automática

Prototipo instrumental de medición de presión en torniquetes

Documentos:

1 Memoria

2 Planos

3 Pliego de condiciones

4 Presupuesto

5 Anexo:

5-1: Código

Autor:

D. José Ignacio Marqués Ortega

Tutor:

D. Ángel Perles Ivars

Valencia, Septiembre de 2017

Gracias a Ángel Perles Ivars por su capacidad docente y acompañamiento durante el largo transcurso del proyecto

Al Dr. Pablo Rodríguez y mi padre por brindarme la oportunidad de acercarme a la investigación.

A mi familia y pareja por apoyarme.

A mis compañeros de fatiga, José Juan y Roberto Zazo, por soportarme y escuchar mis lamentaciones.

Resumen

En los últimos años, se han desarrollado plataformas tecnológicas *open source* como Arduino o Raspberry pi, que permiten a los usuarios resolver problemas de toda clase. Estas plataformas han demostrado su potencia en entornos docentes y de investigación.

En este trabajo se han desarrollado tres prototipos basados en el microcontrolador (μC) Atmega 328P de Atmel, integrados en tarjetas de Arduino, cuyo fin es el de facilitar la realización de un estudio médico, sobre la idoneidad de utilización de un nuevo sistema de torniquete no-neumático HemaClear, en cirugías con isquemia. En concreto, los dispositivos desarrollados son para medir la presión que ejerce el HemaClear sobre los miembros, ya que carece de mecanismos de control de presión.

Los dispositivos dan la medida en milímetros de mercurio y son capaces de almacenar los datos obtenidos en una memoria SD en forma de archivo csv, para facilitar su importación a programas de cálculo como Excel o Matlab. Se han utilizado sensores de fuerza resistivos, como el FSR 402 de interlink o el A201 de Tekscan y sensores de presión de aire digitales.

Palabras clave: *Open source*, microcontrolador, Atmega 328P, Arduino, HemaClear, isquemia, memoria SD, presión, Sensor de fuerza resistivo, sensor de presión de aire digitales, archivos csv.

SUMMARY

In recent years, open source technology platforms such as Arduino or Raspberry pi have been developed, which allow users to solve problems of all kinds. These platforms have proven their power in teaching and research environments.

In this work, three prototypes based on Atmel's Atmega 328P microcontroller (μC) have been developed, integrated in Arduino cards, whose purpose is to facilitate the accomplishment of a medical study, on the suitability of using a new tourniquet system Non-pneumatic HemaClear, in surgeries with ischemia. In particular, the devices developed are to measure the pressure exerted by HemaClear on the limbs, since it lacks pressure control mechanisms.

The devices give the measurement in millimeters of mercury and are able to store the data obtained in an SD memory in the form of csv file, to facilitate its importation to programs of calculation like Excel or Matlab. Resistive force sensors, such as Tekscan's interlink FSR 402 or A201 and digital air pressure sensors have been used.

Key words: Open source, microcontroller, Atmega 328P, Arduino, HemaClear, ischemia, SD memory, pressure, Resistive force sensor, digital air pressure sensor, csv files.

RESUM

En els últims anys, s'han desenvolupat plataformes tecnològiques de codi obert com Arduino o Raspberry pi, que permeten als usuaris resoldre problemes de tota mena. Aquestes plataformes han demostrat la seva potència en entorns docents i de recerca.

En aquest treball s'han desenvolupat tres prototips basats en el microcontrolador (μC) Atmega 328P d'Atmel, integrats en targetes d'Arduino, el fi és el de facilitar la realització d'un estudi mèdic, sobre la idoneïtat d'utilització d'un nou sistema de torniquet no-pneumàtic HemaClear, en cirurgies amb isquèmia. En concret, els dispositius desenvolupats són per mesurar la pressió que exerceix el HemaClear sobre els membres, ja que no té mecanismes de control de pressió.

Els dispositius donen la mesura en mil·límetres de mercuri i són capaços d'emmagatzemar les dades obtingudes en una memòria SD en forma d'arxiu csv, per facilitar la seva importació a programes de càlcul com Excel o Matlab. S'han utilitzat sensors de força resistius, com el FSR 402 de Interlink o el A201 d'Tekscan i sensors de pressió d'aire digitals.

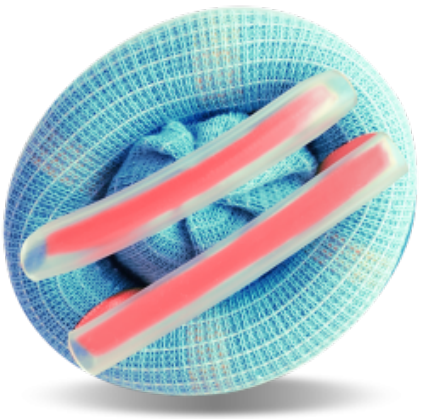
Paraules clau: codi obert, microcontrolador, Atmega 328P, Arduino, HemaClear, isquèmia, memòria SD, pressió, Sensor de força resistiu, sensor de pressió d'aire digitals, arxius csv.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Prototipo instrumental de medición de presión en torniquetes

1. Memoria



Autor:

D. José Ignacio Marqués Ortega

Tutor:

D. Ángel Perles Ivars

Índice

1 Objeto

El objeto de este trabajo es el desarrollo de un dispositivo electrónico de bajo coste, que mida la presión ejercida por un nuevo sistema de isquemia de miembros HemaClear®, que actúa también de torniquete, utilizado en operaciones de traumatología para lograr operar sin sangre.

El dispositivo se utilizará para realizar un estudio observacional de la idoneidad del funcionamiento de dicho sistema en términos de presión, comparado con la información facilitada por el fabricante, devolverá la lectura en milímetros de mercurio (mmHg) y almacenará los datos obtenidos durante el transcurso de la operación.

2 Antecedentes.

Los torniquetes arteriales fueron introducidos en cirugía ortopédica en 1873 por el Dr. Frederich von Esmarch para bloquear la llegada de sangre al campo quirúrgico tras la exanguinación del miembro a operar, mediante una banda elástica.

En la actualidad existen varios tipos de torniquete siendo el modelo más común, el manguito neumático hinchable. Los torniquetes neumáticos regulan la presión que ejercen en superficie mediante un sistema de medida analógico o digital integrado en el modelo comercial. Recientemente se ha introducido un modelo homologado de torniquete denominado HemaClear® (abreviando HC) basado en un anillo tórico elástico, que en la misma maniobra de colocación produce la exanguinación del miembro y el bloqueo del flujo sanguíneo. La presión que produce se estima mediante unas tablas que aporta el fabricante combinando tres parámetros: tamaño HC; circunferencia del miembro en el sitio de colocación y distancia entre este punto y los dedos.

El Dr. Pablo Rodríguez, anestesiólogo del Hospital General de Valencia, detectó varios casos de lesiones nerviosas (neuroapraxia) en las extremidades, tras operaciones con isquemia, coincidiendo con la utilización de la media compresiva. Una de las posibles causas es un exceso de presión, tanto en cantidad como en duración temporal, por lo que se propuso realizar un estudio midiendo la presión del HC® durante las intervenciones. El problema era medir la presión real que ejercía esta media sobre la extremidad. Se comentó con el equipo de trabajo, y tras barajar varias posibilidades, el Dr. José Ignacio Marqués sugirió el desarrollo de un dispositivo específico para la realización del estudio.

3 Estudio de las necesidades.

Dada la naturaleza del proyecto y el entorno donde debe desempeñar su actividad, es necesario cumplir ciertos requisitos de cara a la seguridad de los pacientes y hacer más fácil la obtención de datos para la realización del estudio.

Para ello, hay que analizar y conocer el entorno de las operaciones con isquemia:

3.1 Torniquetes.

El torniquete es un dispositivo utilizado para constreñir y comprimir el sistema circulatorio, tanto arterial como venoso. Se utilizan en ámbito quirúrgico y de emergencia. Actualmente existen dos tipos diferentes de torniquetes. En este apartado procederemos a explicar sus características:

3.1.1 Torniquetes quirúrgicos:

Estos se utilizan para impedir que el flujo de sangre llegue a un miembro, al cual se le va a practicar una cirugía. Este hecho facilita y reduce el tiempo de la intervención. Hay dos tipos de torniquetes quirúrgicos:

- **Torniquetes neumáticos (figura 1):**

Consta de una bolas o vejiga, una funda contenedora de la misma, y una salida/entrada de aire conectada a un dispositivo regulador de la presión. Su principal característica es que el usuario puede regular la presión que este ejerce sobre los tejidos del paciente.

- **Torniquetes no – neumáticos (figura 2):**

Consta de un toroide de gel o silicona al cual se le enrolla una media compresiva. Este dispositivo es de fácil colocación, permite reducir el tiempo de cirugía, característica muy atractiva desde el punto de vista del cirujano. Sin embargo, carece de elemento regulador en términos de presión.

Fig. 1: Torniquete neumático

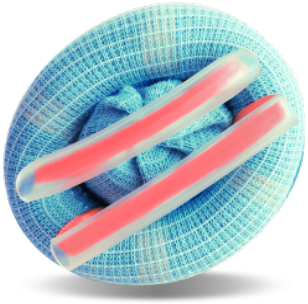


Fig. 2: Torniquete HemaClear

3.1.2 Torniquetes de emergencia:

Como su nombre indica, este tipo de torniquetes se utiliza en situaciones críticas en las que un individuo corre peligro de muerte, ocasionado por una sección arterial, herida profunda e incluso una amputación de miembros. El uso de estos dispositivos se reduce a la intervención de paramédicos en accidentes de cualquier entorno y uso militar (figura 3). Estos son cintas que se colocan a modo de cinturón alrededor del miembro afectado por encima de la herida abierta.

Fig. 3: Torniquete emergencias

La distribución de presiones que ejercen los torniquetes sobre el miembro, depende del área de aplicación de la fuerza. En la figura 4 se observa la distribución teórica de las presiones según el tipo de torniquete utilizado.

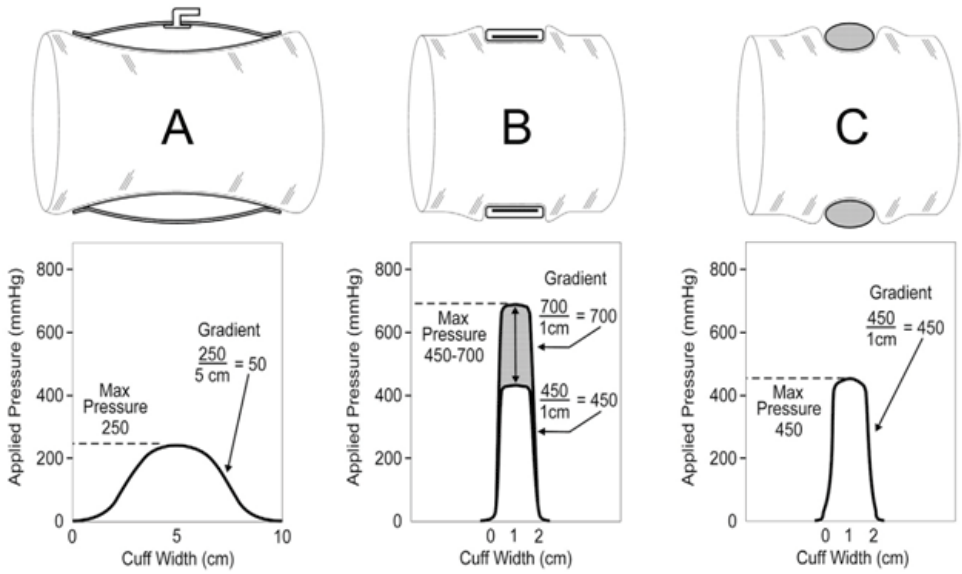


Fig. 4: Distribución de los gradientes de presión

El objeto de estudio en este caso, se centra en los torniquetes quirúrgicos no-neumáticos de la marca HemaClear®, ya que no se sabe a ciencia cierta si la presión que ejercen sobre los miembros es excesiva, para la duración de las operaciones donde se utiliza, al no disponer de mecanismos de regulación de presión

3.2 Isquemia.

Se define isquemia como: "detención o disminución de la circulación de sangre a través de las arterias de una determinada zona, que comporta un estado de sufrimiento celular por falta de oxígeno y materias nutritivas en la parte afectada." [1] Por este motivo las intervenciones han de ser como máximo de dos horas. Por otra parte, la isquemia se utiliza para lograr operar sin sangre, hecho que mejora la visibilidad de los tejidos dañados facilitando la intervención al cirujano.

3.3 Presión.

A lo largo del escrito, siempre se habla de presión. La presión se define como la fuerza aplicada sobre una superficie determinada:

$$P = \frac{F}{A}$$

Donde "P" es la presión , "F" fuerza aplicada y "A" área o superficie; siendo las unidades en el sistema internacional [N/m^2]. Otras unidades de presión representativas por su uso son: psi, atmosferas, bar, pascales, milímetro de mercurio, etc. En este caso se utilizará el [mmHg], ya que es la magnitud típica de la presión arterial y la más extendida en medicina. Existen varias magnitudes susceptibles de ser medidas en lo que a presión se refiere, como: la presión absoluta, presión manométrica y la presión diferencial o relativa.

- **Presión absoluta:**

Este tipo de presión se mide desde 0 Pa, y tiene como referencia la presión estática de forma aislada, en la medida de la presión absoluta se refleja el efecto de la presión atmosférica. Se utiliza para obtener la presión atmosférica, en altímetros o presiones al vacío.

- **Presión manométrica:**

Esta toma como referencia la presión atmosférica, es muy utiliza para obtener medidas de presión en neumáticos y presión arterial.

- **Presión diferencial:**

Este tipo de medida de presión dinámica se utiliza para mantener la presión relativa entre dos contenedores, como tanques de compresión o líneas de transmisión asociadas.

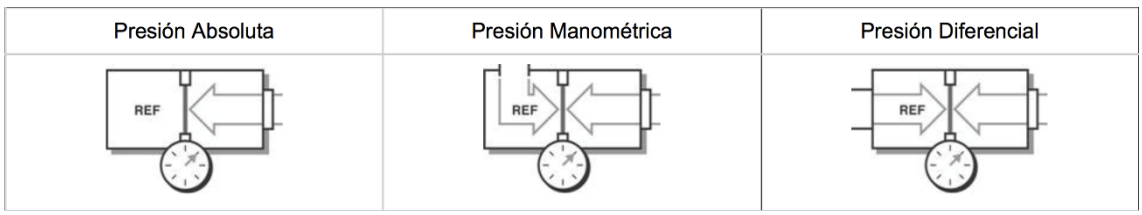


Fig. 5: Tipos de medida de presión

3.4 Magnitud a medir en el HemaClear.

HemaClear® es un dispositivo de torniquete no-neumático, por tanto, no tiene mecanismo de regulación que controle la presión real que ejerce sobre los pacientes, el dispositivo debe de ser capaz de medir la presión manométrica.

HemaClear siendo utilizado en quirófano

4 Planteamiento de las soluciones y alternativas.

En este apartado se va analizar de forma detallada las diferentes alternativas para los componentes del dispositivo. Desde la tarjeta de adquisición de datos hasta la carcasa que contiene nuestro prototipo.

4.1 Sensores.

Los sensores son elementos capaces de detectar magnitudes físicas y provocar una respuesta eléctrica, esta puede ser interpretada por un computador o en este caso un µC. Dado que se pretende diseñar un prototipo instrumental de medición de presión manométrica, se han valorado diferentes sensores que reaccionen a esta magnitud física.

4.1.1 Galgas extensiométricas:

Las galgas extensiométricas son sensores que se basan el efecto piezoresistivo (figura.6). La piezoresitividad, es una propiedad que poseen algunos semiconductores de modificar su resistencia a la corriente eléctrica, cuando son sometidos a esfuerzos mecánicos, ya sea de tracción o compresión. Estos sensores son utilizados en la industria para medir esfuerzo, deformación, presión, etc.

Las galgas son capaces de medir las deformaciones de un material, pero solo en una dirección, si se desea conocer las deformaciones en más direcciones es necesario combinar distintos sensores como se muestra en la figura 7.

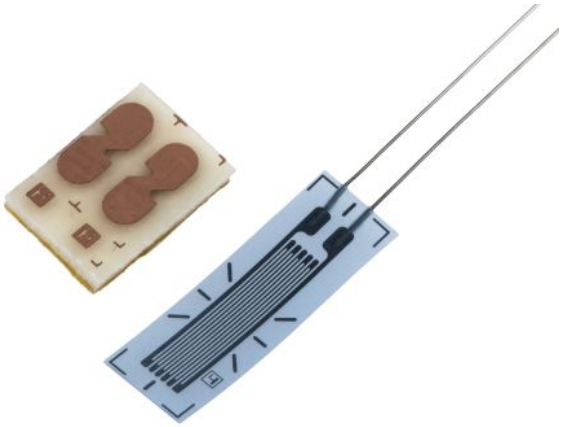


Fig. 6: Galga

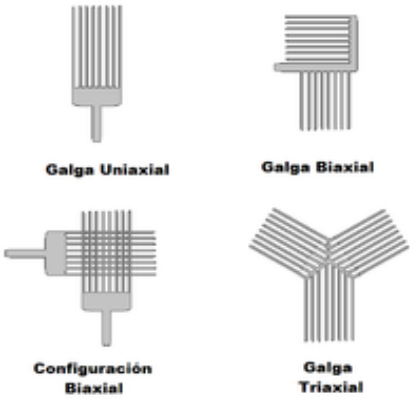


Fig. 7: Distribuciones posibles

Las galgas serán descartadas porque requieren de instalación previa en el miembro donde se desea medir la presión, situación no deseable para la aplicación del prototipo, que ha de ser de fácil utilización.

4.1.2 Sensores de fuerza resistivos:

Los sensores de fuerza resistivos son elementos que reaccionan alterando su resistencia a la corriente cuando se le aplica una fuerza sobre el área activa. La resistencia va disminuyendo conforme se aplica más presión.

Estos sensores tienen multitud de aplicaciones, entre ellas: se utilizan como sensores de movimiento, que detectan si un objeto cambia de posición; también actúan como sensores de fuerza de contacto y para medir la masa de un objeto. En las siguientes figuras 8 y 9 se pueden observar varios modelos de este tipo de sensores.

Fig. 8: Sensor A201

Fig. 9: Sensor FSR 402

Este tipo de sensores es una opción muy atractiva para el desarrollo del prototipo, ya que puede colocarse sin ningún tipo de problema debajo del torniquete.

4.1.3 Sensores de presión de fluido:

Este tipo de sensores se utilizan tanto en la industria, para controlar la presión que pueda albergar cualquier tipo de condición o depósito de fluido; como en aparatos instrumentales en medicina, automoción, robótica, etc.

Existe una amplia gama de este tipo de sensores y múltiples fabricantes, y se han valorado algunas marcas. A destacar los sensores de Honeywell serie ABP, como el de la figura 10, y Omron presente en multitud de instrumental médico, en España.

Fig. 10: Sensor ABPMANT100PG2A3

La implementación de este tipo de sensores es viable, ya que se dispone de transmisión de señal tanto analógicas, que necesita del diseño de una etapa de amplificación; como digitales, que envían la señal mediante estándares de comunicación I^2C o SPI ya calibrados de fábrica, cualidad muy atractiva que ahorra tiempo en el desarrollo del prototipo

4.2 Sistema de adquisición de datos.

El sistema de adquisición de datos se encarga de gestionar todos los procesos tanto la obtención como el procesamiento de los datos obtenidos. Un computador con una tarjeta de adquisición de datos es buena opción, que será descartada ya que es preferible un sistema autónomo y portátil. Por ello, se va optar por el uso un microcontrolador (abreviado µC).

El µC, es un circuito integrado digital programable compuesto por una unidad central de procesos (CPU), memorias (RAM y ROM) y multitud de periféricos integrados.

En los últimos años se han popularizado el uso de µC en todo tipo de productos electrónicos, como electrodomésticos, consolas, móviles, etc. También han surgido plataformas *open source*, coincidiendo con el auge del movimiento DIY (en inglés "*do it yourself*"). Estas, se han convertido en herramientas muy socorridas a la hora de diseñar prototipos digitales, por su comunidad activa y variedad de elementos compatibles. Para elegir el µC adecuado se han valorado diferentes plataformas:

4.2.1 Microcontrolador STM32:

STMicroelectronics es una empresa dedicada al diseño, fabricación y distribución de productos electrónicos y entornos de programación para los mismos. Han desarrollado una serie de placas para prototipado rápido como se muestra en la figura 11, muy versátiles y de gran potencia gracias a que incorporan un µC ARM Cortex-M de 32bits.

Fig. 11: STM32 Nucleo

Aunque es una opción viable y muy asequible, la descartaremos debido al entorno de desarrollo es de difícil utilización y dispone de información limitada para consultas que puedan surgir durante el desarrollo

4.2.2 Arduino:

Arduino es un ecosistema de *software* y *hardware* libre. Este entorno es muy atractivo a la hora de diseñar e implementar prototipos electrónicos de toda clase, ya que Arduino dispone de: una gran comunidad de desarrolladores activos, el entorno de programación es libre, simple y multiplataforma; en cuanto al hardware, las placas de prototipado son baratas, versátiles y reutilizables.

Por todo ello esta será la plataforma elegida para desarrollar el prototipo de medida de presión. En concreto se va utilizar la placa Arduino NANO como la de la figura 12.

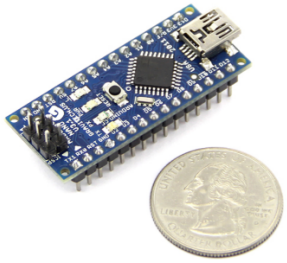


Fig. 12: Tarjeta Arduino NANO

4.3 Protocolos y estándares de comunicación.

Para trasmitir datos entre componente electrónicos existen múltiples formas, a destacar la comunicación serie, donde la información se transmite bit a bit a través de un único canal o cable. También es posible comunicarse mediante comunicación paralela, esta consiste en la transmisión simultánea de varios bits mediante diferentes canales que posteriormente se sincronizan para que la información llegue a destino.

Sin embargo, se dispone diferentes tipos de protocolos o estándares de comunicación serie, algunos de los más utilizados en la industria son I^2C (siglas de "*Inter-Integrated Circuit*" o también conocido como *TWI* siglas de *Two Wires Interface)* y SPI (siglas de *Serial Peripheal Interface*). Estos protocolos están disponibles en el µC ATMega328P del Arduino y existen librerías específicas para utilizarlos fácilmente.

4.3.1 I^2C:

El I^2C es uno de los protocolos más utilizados en la industria, normalmente para comunicar circuitos integrados entre sí. Este fue creado en 1982 por Philips. Se caracteriza principalmente por utilizar dos líneas para trasmitir datos. El "SDA" es la línea por la cual se transfieren los datos en forma de bytes y el "SCL" es por donde se transmite la señal del reloj para sincronizar los diferentes circuitos integrados.

Cada dispositivo conectado al bus I²C posee una dirección única, esta dirección en el caso del Arduino es de 7bits y suele expresarse en hexadecimal. Cada dispositivo puede ser configurado como maestro o como esclavo, el maestro es el que comienza la transmisión de datos y genera la señal de reloj; el esclavo espera las órdenes del maestro. Esta configuración puede cambiar a lo largo de la ejecución de la aplicación, si los dispositivos aceptan la condición de maestro. Existen algunos dispositivos como sensores, que únicamente se pueden configurar como esclavos.

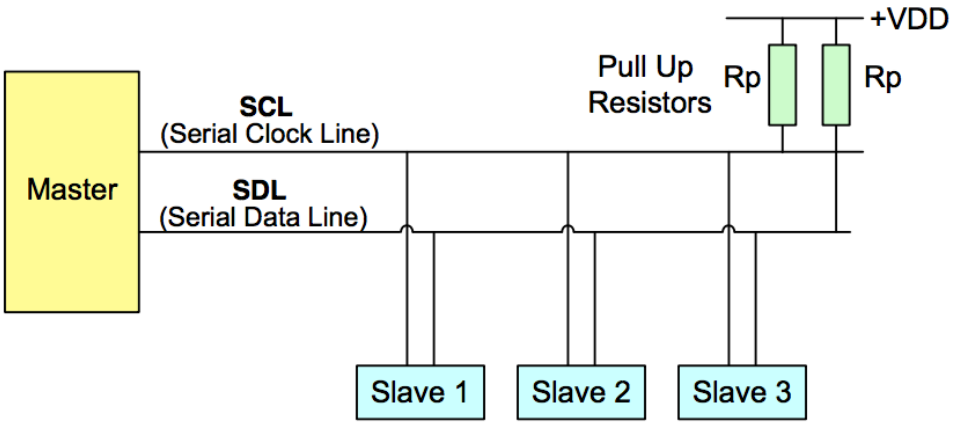


Fig. 13: Diagrama de conexión bus I2C

En cuanto a la conexión, tal y como puede observarse en la figura 13, si se quiere conectar varios integrados dispuestos a diferentes distancias del maestro, es necesario conectar dos resistencias *Pull-up* en las líneas de trasmisión de datos para que la señal llegue correctamente. Sin embargo, al utilizar una placa Arduino no será necesario, ya que al implementar la librería "*Wire.h*" se configuran las resistencias en *pull-up*, asociadas a los pines del bus de comunicaciones I²C .

La velocidad de transferencia de datos oscila entre los 100 kB/s y el modo de alta velocidad que permite velocidades de transmisión de hasta 3 MB/s. Otra característica a tener en cuenta, es que la transmisión de datos es "*half dúplex*", esto significa que el flujo de datos solo se puede trasmitir en un sentido cada vez, por lo que en el momento que uno de los dispositivos conectados comience a recibir información, este tendrá que esperar hasta que el emisor de esa información pare de trasmitir para empezar a responder.

4.3.2 SPI:

Al igual que el I²C, SPI es otro estándar de comunicaciones serie a poca distancia, que permite controlar otros dispositivos digitales sincronizados mediante una señal de reloj.

El SPI requiere de cuatro líneas de transmisión de datos: "SCK" por donde envía el maestro la señal de reloj, para mantener sincronizados los diferentes elementos; "SS"o "CS" este es la línea mediante la cual, el maestro activa el esclavo concreto con el cual necesita comunicarse; "MOSI" es línea de comunicación del maestro al esclavo y "MISO" es la línea mediante la cual el esclavo se comunica o envía datos al maestro. El esquema de conexiones para múltiples esclavos será como la que muestra la figura 14.

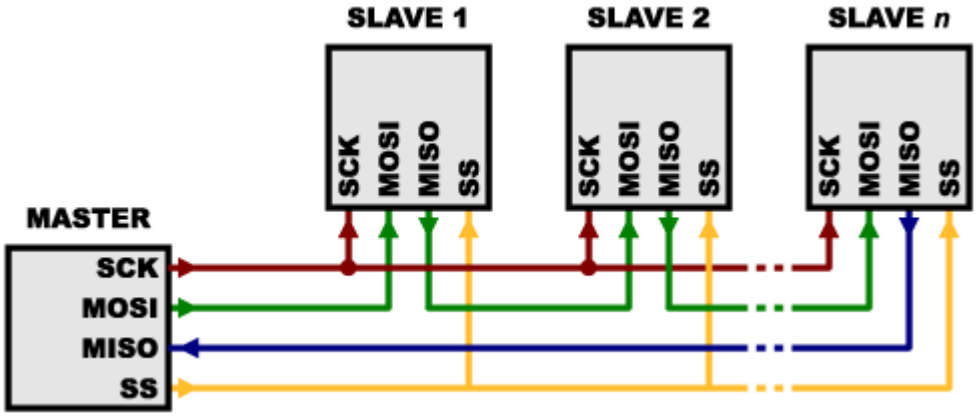


Fig. 14: Diagrama de conexión bus SPI

Estos estándares de comunicación son muy útiles a la hora de controlar diferentes sensores y actuadores digitales. En los prototipos que se van a exponer en el siguiente apartado se utilizarán.

4.4 Sistema de almacenamiento de datos.

Para facilitar el tratamiento de los datos obtenidos para la realización del estudio, se requiere que el prototipo sea capaz de almacenar los resultados de las mediciones durante las intervenciones. Debido a que la memoria FLASH de la tarjeta Arduino NANO es muy limitada, se valora la posibilidad de añadir un soporte de memoria externo como una tarjeta SD, dado que existen multitud de módulos con zócalos para diferentes tipos de tarjeta de almacenamiento de datos compatibles con Arduino. Se utilizará un módulo SD que admite comunicación SPI como el de la figura 15

Fig. 15: Módulo SD

Los datos se almacenarán en la tarjeta SD como archivos de datos separo por comas .csv, que facilitará la exportación de datos al Excel para su posterior tratamiento. Los archivos .csv son de formato abierto, utilizados para representar tablas donde las columnas se separan mediante signos de puntuación (“,” o “;”) y las filas por saltos de línea. Otra alternativa sería almacenarlos en un archivo de texto “txt” pero dificultaría la exportación a programas de cálculo como Excel o Matlab.

4.5 Sistema de visualización.

Los elementos visuales son imprescindibles para comprobar que el prototipo está funcionando correctamente además de permitir visionar la medida en tiempo real. Para ello, se ha decidido utilizar un *dispaly* LCD 16x2 como el de la figura 16, contralado vía I^2C para disminuir la cantidad de pines utilizados del µC. Otro de los motivos es que el entorno de desarrollo *software* Arduino dispone de librerías específicas que facilitan la programación para el uso de pantallas LCD.

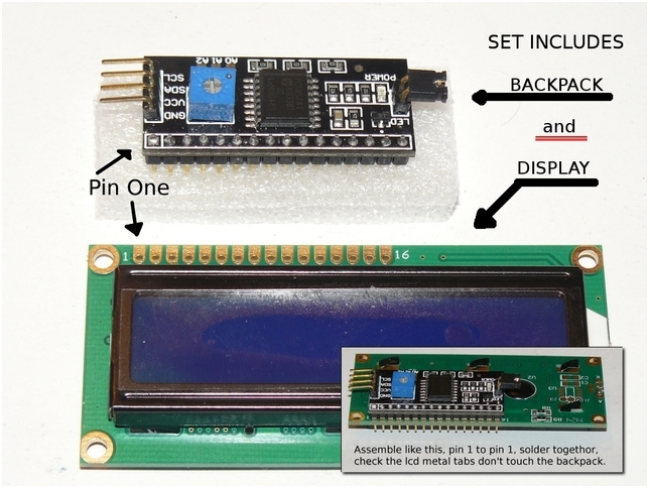


Fig. 16: LCD I2C

Como alternativa viable, cabe la posibilidad de utilizar pantallas de tecnología OLED (siglas de *Organic Light-Emmiting Diode*) que reducen el consumo de energía y tamaño. De este tipo de pantalla se valoraron los OLED *displa*y de Adafruit como el que se muestra en la figura 17. Estos permiten incluso representar gráficos, y el fabricante facilita librerías para Arduino.

Fig. 17: LCD I2C

5 Descripción detallada de la solución aportada

Para dar respuesta a estas necesidades se han desarrollado tres prototipos a lo largo del trabajo, cada uno de ellos mejora en algún aspecto al anterior. Sin embargo, son todos funcionales y se han utilizado en la realización del estudio. La diferencia principal de los diferentes dispositivos desarrollados, reside en el tipo de sensor utilizado con su consecuente tratamiento de señal característico.

Para el desarrollo de los prototipos se han utilizado casi en su totalidad herramientas de software gratuitas a excepción del programa de CAD (siglas en inglés "*computer-aided desing*", en castellano "diseño asistido por ordenador") SolidWorks, que abaratan el coste de fabricación del prototipo. El diseño, fabricación de la PCB (siglas de "placa de circuito impreso") y representación de los esquemático normalizados, se ha utilizado el EasyEda. Esta herramienta es completamente gratuita y online, por lo que solo hace falta conexión a internet. Por otra parte, se ha utilizado Fritzing para que los diagramas de conexiones resulten más visuales y fáciles de reproducir. Este entorno es muy utilizado dentro de la comunidad "*maker",* desarrolladores de Arduino, homólogos compatibles, Raspberry Pi, etc. Dado que dispone de multitud de librerías gráficas para representar las conexiones de los diferentes prototipos, además de permitir genera el código y la PCB del proyecto a desarrollar. La carcasa de los prototipos ha sido diseñada mediante el *software* de CAD SolidWorks para impresión 3D.

5.1 Prototipo 1

5.1.1 Descripción.

El primer prototipo diseñado es un dispositivo electrónico digital basado en el uC de 8 bits, ATmega328p de Atmel, integrado en una tarjeta Arduino Nano. El sensor analógico utilizado es el modelo FSR-402 de Interlik como el mostrado anteriormente en la figura 9, por otro lado, cuenta con una pantalla LCD de 16x2, donde se muestra la medida en tiempo real, además de almacenar la medida en una memoria SD cada dos segundos. En cuanto a la alimentación se ha optado por la utilización de banco de carga externo de 5000 mA, regulado a 5V, para asegurar el buen funcionamiento del dispositivo.

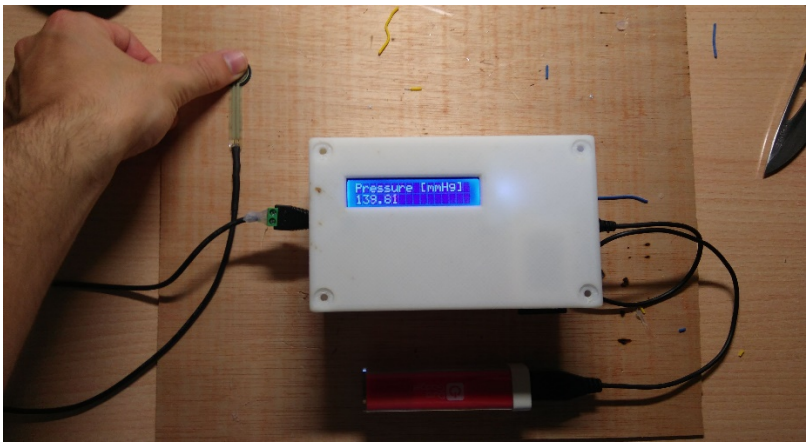


Fig. 18: Prototipo 1.0 terminado

5.1.2 Hardware desarrollado.

5.1.2.1 Arduino NANO.

El Arduino Nano 3V3 es una placa de prototipado rápido, pequeña de fácil incorporación a una PCB. Esta se basa, como se ha mencionado en varias ocasiones, en el µC ATmega 328P de 8 bits.

En cuanto a sus características cabe destacar:

- Voltaje de operación: 5 Voltios.
- Voltaje de entrada o alimentación: [5-20] Voltios.
- Entradas/salidas digitales 14, de los cuales 6 son PWM.
- Corriente de salida en pines 40 mA.
- Memoria Flash de 32 KB de los cuales 2 KB reservados para el *bootlouder*.
- Memoria SRAM de 2KB.
- Memoria EEPROM 1KB.
- Frecuencia del reloj 16 MHz
- Dispone de 7 pines analógicos asociados a un conversor analógico digital de 10bits.

Por otro lado, la tarjeta de prototipos Arduino Nano dispone de pines reservados o que pueden ser configurados para uso específico como muestra en el diagrama de la figura 19.

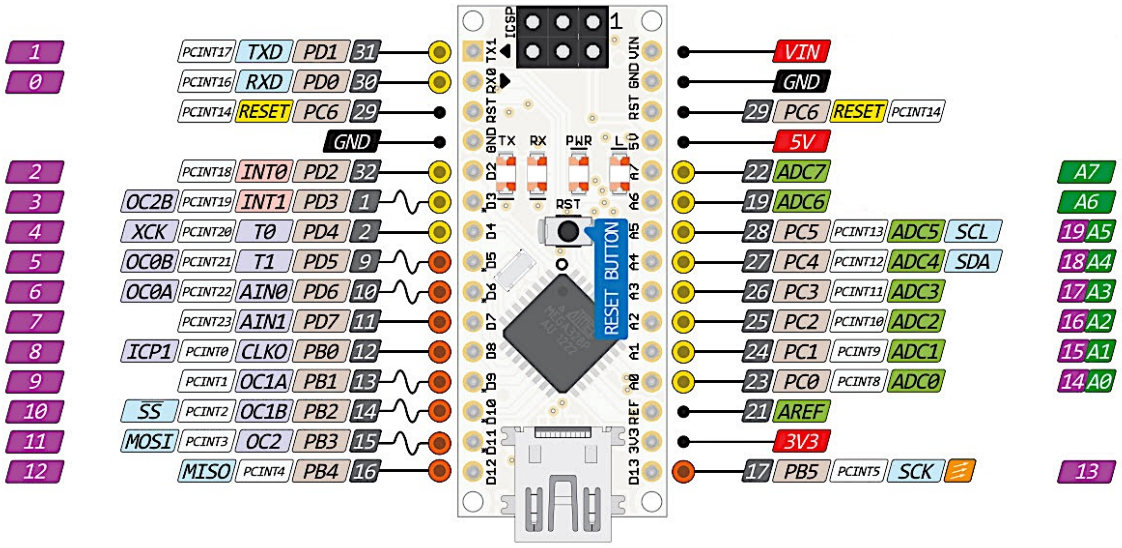


Fig. 19: Prototipo 1.0 terminado

Donde se pueden ver los pines reservados para la comunicación I²C (A4 y A5), la comunicación SPI (D10, D11, D12, D13) y el pin IOREF, que permite cambiar la referencia de tensión de la placa para obtener una resolución más adecuada al tratamiento de señal analógica, este puede soportar una tensión de entrada comprendida entre 1,2 V y 5 V.

5.1.2.2 Sensor FSR 402 y obtención de señal.

El sensor FSR 402 de Interlink, funciona básicamente como un potenciómetro que altera su resistencia eléctrica al aplicar presión sobre el área activa. El rango de resistencia que abarca, oscila entre un megaohmio y cien ohmios [1MΩ - 100 Ω], soporta fuerzas puntuales hasta once kilogramos, más allá de ese rango, satura.

El sensor consta de varias capas como se muestra en la figura 19, una primera membrana donde van impresas las conducciones y anclados los bornes del sensor, luego tiene un polímero adhesivo que hace de separador entre la primera membrana y la membrana conductora, es la que permite la conducción de corriente cuando se aplica presión. Por ultimo tiene una membrana protectora que separa del exterior la parte activa del sensor. Al estar formado por membranas, hace que el sensor sea muy fino, esta característica es idónea para introducir el sensor debajo del torniquete (el orden de las capas corresponde de abajo hacia arriba como se muestra en la figura 20)

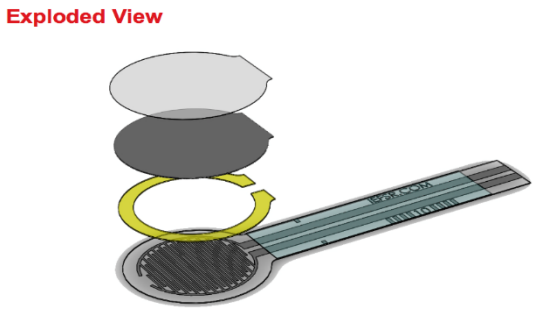


Fig. 20: Capas sensor

Para la obtención de la señal, el fabricante recomienda diferentes tipos de circuito para aplicaciones determinadas que se pueden consultar en el anexo 2. En este caso, basta con poner una resistencia de 4k7 como divisor de tensión conectada a tierra, como se muestra en el diagrama del esquema de la figura 21. Además, para garantizar que la señal no oscila inadecuadamente, se ha añadido un regulador de tensión de alta precisión que, además, alimentará el pin de referencia del AD de la tarjeta para que la resolución de la medida sea mejor y el rango del mismo se adapte de 0 a 4,1 V.

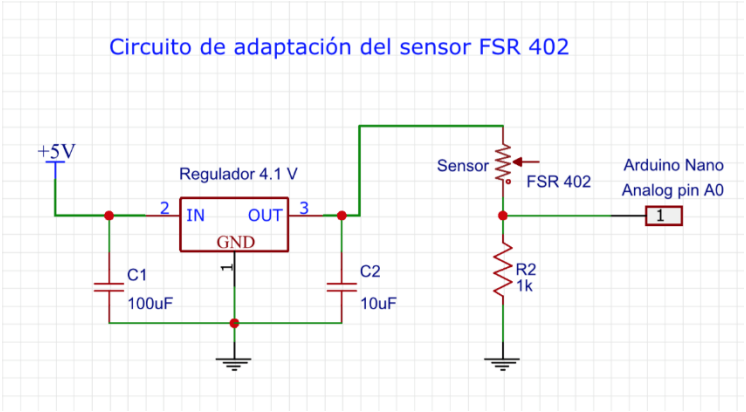
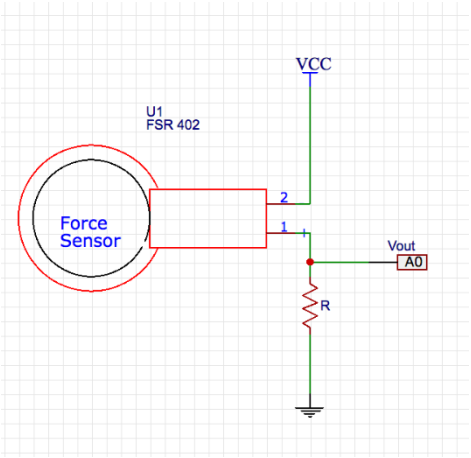


Fig. 21: Circuito de adaptación de señal

- **Cálculo del divisor de tensión:**



1) Se obtiene la función de trasferencia que relaciona la tensión de entrada y la tensión de salida, conectada a la entrada analógica AO. Mediante el teorema de Millman :

$$Vout = Vcc\left[\frac{1}{\left(1+\frac{R\ sensor}{R}\right)}\right]$$

2) Conocidas la tensión de entrada Vcc =4.1 V y el rango de valores que puede tomar el sensor Rsensor = [100Ω - 1MΩ]

Pmin = 0 mmHg → Rsensor >=1MΩ → Vout =0 V

Pmax = 7.757 mmHg → Rsensor<=100 Ω → Vout= 4,00 V

De sustituir los datos correspondientes en la fórmula anterior, resulta:

R =4166,66 Ω → Rnormalizada = 4700 Ω ~ R=4k7

El fabricante facilita, en el manual de usuario, todos los datos expuestos anteriormente, además de una gráfica (figura 22), que relaciona la masa en gramos con la resistencia del sensor, por lo que será necesario obtener una gráfica similar que relacione la presión en [mmHg].

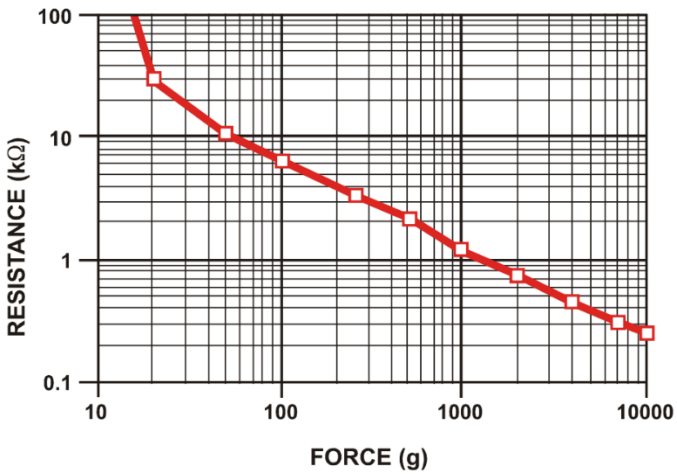


Fig. 22: FSR Respuesta

Como se observa en la gráfica de la figura 21, la escala es logarítmica en ambos ejes, con lo que se puede intuir que la respuesta en cuanto a la resistencia no va a ser lineal. Se ha procedido a obtener de manera experimental la relación que existe entre la presión [mmHg] y la resistencia del sensor.

- **Obtención experimental de las curvas de resistencia y conductancia:**

Para el diseño el experimento, se ha utilizado como el elemento regulado y calibrado, el esfigmomanómetro utilizado en las operaciones con isquemia, antes de la incorporación del HemaClear, como el de la figura 23.

Fig. 23: Manguito calibrador

Como base de medida, se ha construido un soporte que consta de una almohadilla de gel recubierta con vendas para dar consistencia a la estructura. Véase en la figura 24.

Fig. 24: Base de medida

Según la información que da el fabricante, dadas las características físicas del sensor, antes de comenzar a utilizarlo convine fatigar el mismo, sometiéndolo al 110% de la carga máxima que va soportar en la sesión de toma de medidas, durante diez minutos. Por ello, antes de comenzar la toma de medidas se someterá al sensor a una carga previa de 600 mmHg. Una vez transcurridos diez minutos, se

aliviará la presión del sensor, se conectará el polímetro digital y se comenzará a anotar los resultados de la resistencia incrementando de 10 mmHg cada vez. Se obtiene la gráfica como la que se muestra en la figura 25.

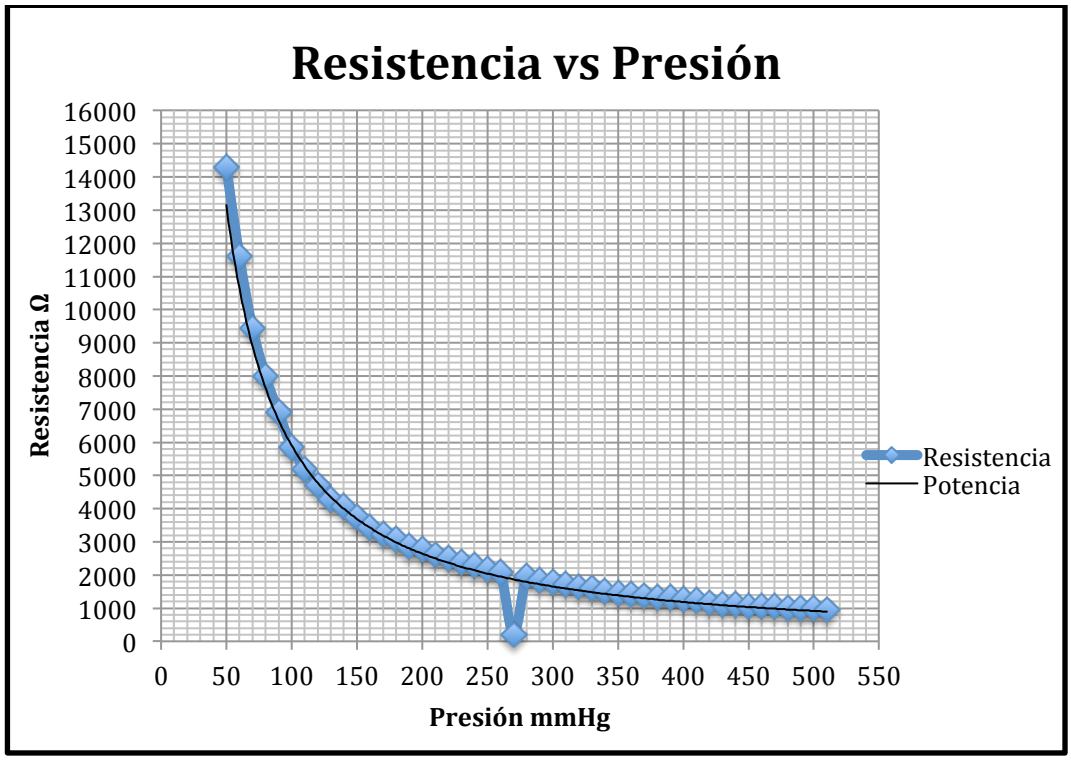


Fig. 25: Gráfica experimental Resistencia Vs Presión FSR 401

Como se puede observar, la relación de la resistencia frente a presión no es líneal. Sin embargo, la relación con respecto a la conductancia, sí lo es. La conductancia se define como la inversa de la resistencia eléctrica, y se mide en Siemens según el sistema internacional, por tanto, al realizar la inversa de los datos obtenidos resulta la figura 26. Donde se puede observar existe una relación lineal con la que poder trabar.

Para obtener la relación teórica que existe entre la conductancia frente a la presión hay que partir de la función de transferencia sin desarrollar.

$$Vout = Vcc\left[\frac{1}{\left(1+\frac{1}{R\times Y_{fsr}}\right)}\right]$$

Donde $Y_{fsr} = 1/R_{sensor}$

El orden de la R_{sensor} de KΩ implica que, el orden de Y_{Fsr} es de mS o µS. Este dato es importante de cara a la programación, ya que la capacidad de cálculo del Arduino puede que desprecie el dato de la conductancia si este opera con otra cifra de orden mayor, como unidades o decenas, por lo que necesitamos pasar el dato a µS como veremos en el apartado de programación.

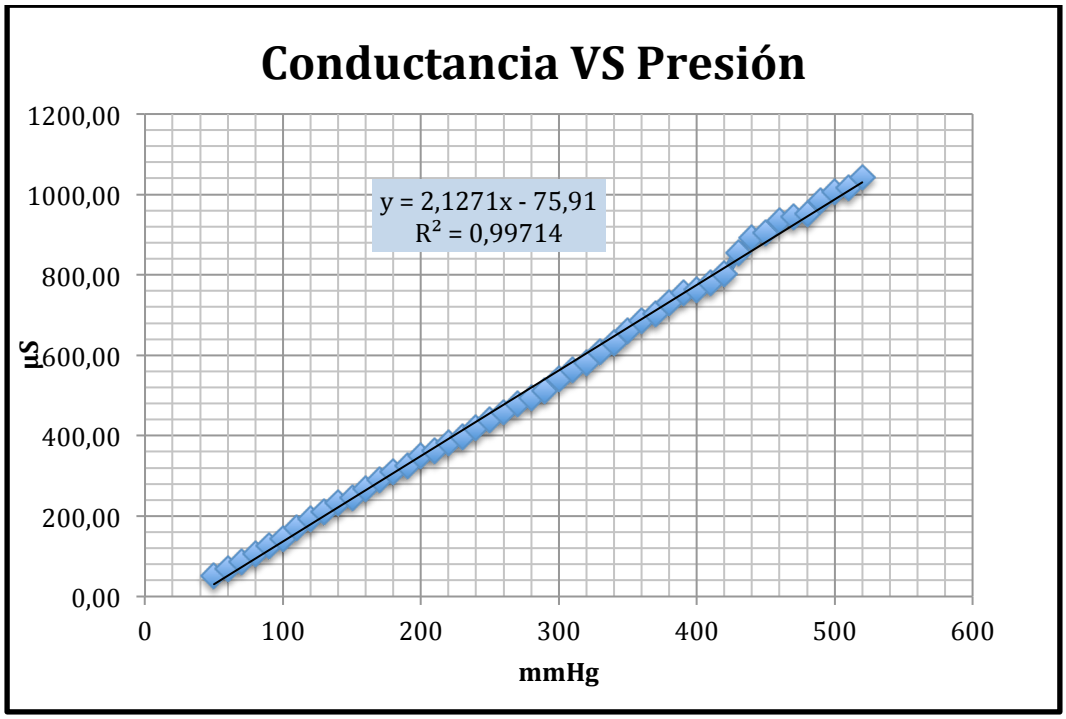


Fig. 26: Grafica experimental Conductancia Vs Presión FSR 401

Los datos obtenidos en el proceso experimental se pueden observar en la tabla 1.

Por otra parte, para la relación de presión frente a la tensión y comprobar la relación que existe entre estos parámetros con el circuito terminado, se ha procedido de la misma forma, se han realizado varios barridos de toma de medidas para comprobar la evolución de la señal mediante el lector analógico digital del Arduino NANO y el polímetro digital. El resultado se puede apreciar en la gráfica de la figura 27 y las anotaciones de medidas en la tabla 2.

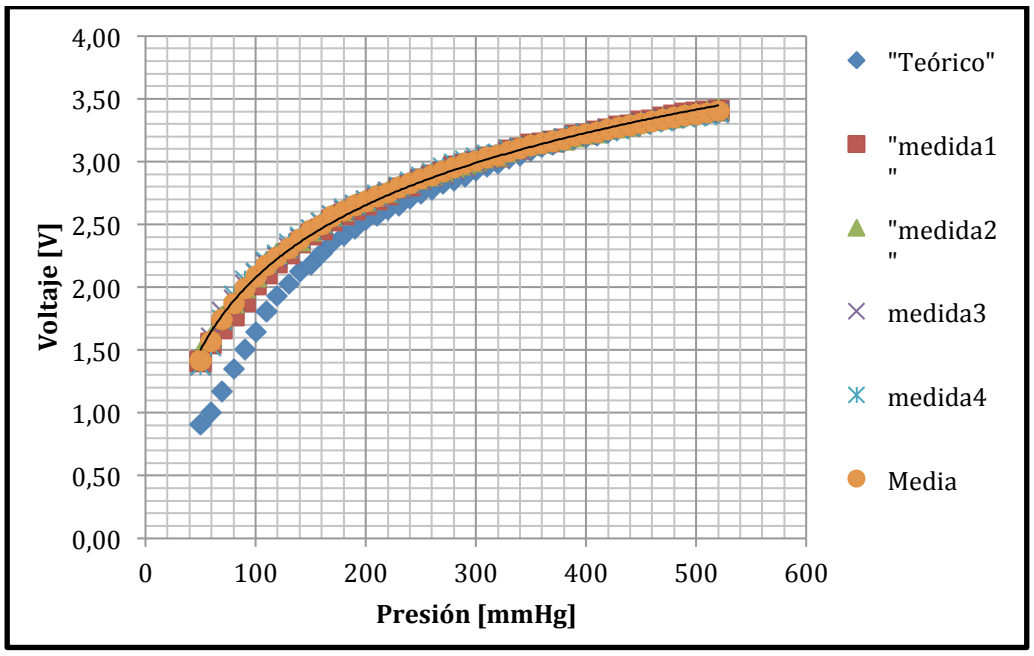


Fig. 27: Gráfica experimental Tensión Vs Presión FSR 401

Como puede observarse, después del proceso de fatiga inicial, la repetitividad del sensor aumenta. A partir de presiones superiores a 200 mmHg, las medidas difieren mínimamente con la estimación teórica, por lo que podemos concluir que está bien diseñado. Por otra parte, es necesario calibrar el dispositivo cada vez que vaya a ser usado para obtener una medida lo más precisa posible. Los datos obtenidos para la gráfica de la figura 26 se pueden observar en la tabla 2.

mmHg	kΩ	Ω	S	mS	µS
50	19.20	19200	0.0001	0.05	52.08
60	14.30	14300	0.0001	0.07	69.93
70	11.60	11600	0.0001	0.09	86.21
80	9.45	9450	0.0001	0.11	105.82
90	7.99	7990	0.0001	0.13	125.16
100	6.90	6900	0.0001	0.14	144.93
110	5.86	5860	0.0002	0.17	170.65
120	5.20	5200	0.0002	0.19	192.31
130	4.73	4725	0.0002	0.21	211.64
140	4.29	4290	0.0002	0.23	233.10
150	4.08	4075	0.0002	0.25	245.40
160	3.73	3730	0.0003	0.27	268.10
170	3.44	3440	0.0003	0.29	290.70
180	3.23	3230	0.0003	0.31	309.60
190	3.07	3070	0.0003	0.33	325.73
200	2.86	2855	0.0004	0.35	350.26
210	2.76	2760	0.0004	0.36	362.32
220	2.62	2620	0.0004	0.38	381.68
230	2.52	2515	0.0004	0.40	397.61
240	2.38	2380	0.0004	0.42	420.17
250	2.28	2275	0.0004	0.44	439.56
260	2.19	2185	0.0005	0.46	457.67
270	2.09	2090	0.0005	0.48	478.47
280	2.03	2030	0.0005	0.49	492.61
290	1.95	1950	0.0005	0.51	512.82
300	1.85	1846	0.0005	0.54	541.71
310	1.78	1777	0.0006	0.56	562.75
320	1.72	1724	0.0006	0.58	580.05
330	1.64	1640	0.0006	0.61	609.76
340	1.59	1588	0.0006	0.63	629.72
350	1.51	1513	0.0007	0.66	660.94
360	1.46	1457	0.0007	0.69	686.34
370	1.42	1423	0.0007	0.70	702.74
380	1.37	1368	0.0007	0.73	730.99
390	1.33	1326	0.0008	0.75	754.15
400	1.31	1312	0.0008	0.76	762.20
410	1.28	1282	0.0008	0.78	780.03
420	1.25	1247	0.0008	0.80	801.92
430	1.17	1171	0.0009	0.85	853.97
440	1.12	1123	0.0009	0.89	890.47
450	1.10	1104	0.0009	0.91	905.80
460	1.07	1071	0.0009	0.93	933.71
470	1.06	1060	0.0009	0.94	943.40
480	1.05	1050	0.0010	0.95	952.38
490	1.02	1017	0.0010	0.98	983.28
500	0.99	993	0.0010	1.01	1007.05
510	0.99	985	0.0010	1.02	1015.23
520	0.96	960	0.0010	1.04	1041.67

Tabla 1: obtención experimental Resistencia y conductancia de FSR 401

Fatiga del sensor 450 mmHg 20min		39 min		Vcc=4.096		Vcc=4.096		Vcc=4.096		Vcc=4.096	
		FSR402 divisor resistivo 4k7		medida1		medida 2		medida 3		medida 4	
Resistencia	Resistencia	Pressure	Vin Teorico	Vin arduino	Vin Polimetro	Vin arduino	Vin Polimetro	Vin arduino	Vin Polimetro	Vin arduino	Vin Polimetro
Ohm	Kohm	mmHg	V	V	V	V	V	V	V	V	V
16200	16,20	50	0,91	1,41	1,40	1,48	1,46	1,38	1,37	1,38	1,37
14300	14,30	60	1,00	1,55	1,50	1,60	1,60	1,60	1,59	1,53	1,52
11600	11,60	70	1,17	1,67	1,66	1,77	1,76	1,80	1,79	1,74	1,73
9450	9,45	80	1,35	1,77	1,77	1,88	1,88	1,90	1,90	1,93	1,90
7990	7,99	90	1,50	1,89	1,88	2,00	1,99	2,02	2,01	2,05	2,03
6900	6,90	100	1,64	2,02	2,02	2,09	2,08	2,11	2,09	2,12	2,11
5860	5,86	110	1,81	2,11	2,11	2,20	2,18	2,19	2,17	2,20	2,19
5200	5,20	120	1,93	2,20	2,20	2,27	2,26	2,24	2,23	2,26	2,25
4725	4,73	130	2,03	2,27	2,27	2,31	2,31	2,31	2,30	2,34	2,33
4290	4,29	140	2,13	2,35	2,34	2,37	2,36	2,39	2,38	2,40	2,39
4075	4,08	150	2,18	2,42	2,41	2,45	2,44	2,46	2,45	2,46	2,45
3730	3,73	160	2,27	2,46	2,46	2,51	2,52	2,51	2,50	2,51	2,50
3440	3,44	170	2,35	2,53	2,52	2,57	2,56	2,57	2,56	2,57	2,56
3230	3,23	180	2,41	2,58	2,58	2,61	2,60	2,61	2,60	2,63	2,62
3070	3,07	190	2,46	2,62	2,61	2,66	2,65	2,65	2,64	2,66	2,65
2855	2,86	200	2,53	2,66	2,65	2,69	2,68	2,69	2,68	2,69	2,68
2760	2,76	210	2,57	2,70	2,69	2,73	2,72	2,72	2,71	2,74	2,73
2620	2,62	220	2,62	2,74	2,73	2,76	2,75	2,76	2,75	2,75	2,74
2515	2,52	230	2,65	2,79	2,78	2,80	2,79	2,79	2,78	2,80	2,79
2380	2,38	240	2,71	2,81	2,82	2,83	2,82	2,82	2,81	2,84	2,83
2275	2,28	250	2,75	2,86	2,85	2,86	2,85	2,85	2,84	2,88	2,87
2185	2,19	260	2,78	2,89	2,88	2,89	2,88	2,87	2,86	2,91	2,90
2090	2,09	270	2,82	2,92	2,91	2,92	2,91	2,90	2,89	2,95	2,94
2030	2,03	280	2,85	2,96	2,94	2,95	2,94	2,93	2,92	2,98	2,96
1950	1,95	290	2,88	2,98	2,97	2,98	2,97	2,96	2,94	3,01	3,00
1846	1,85	300	2,93	3,00	2,99	3,00	2,99	2,99	2,98	3,03	3,01
1777	1,78	310	2,96	3,03	3,02	3,03	3,02	3,02	3,01	3,05	3,04
1724	1,72	320	2,98	3,04	3,04	3,05	3,04	3,05	3,04	3,06	3,05
1640	1,64	330	3,02	3,09	3,08	3,08	3,07	3,07	3,06	3,09	3,08
1588	1,59	340	3,05	3,10	3,09	3,11	3,09	3,09	3,08	3,11	3,09
1513	1,51	350	3,09	3,14	3,13	3,13	3,11	3,12	3,11	3,12	3,11
1457	1,46	360	3,12	3,15	3,14	3,15	3,14	3,14	3,13	3,14	3,13
1423	1,42	370	3,13	3,16	3,15	3,17	3,16	3,16	3,15	3,16	3,15
1368	1,37	380	3,16	3,18	3,17	3,18	3,17	3,18	3,17	3,18	3,17
1326	1,33	390	3,18	3,21	3,20	3,20	3,19	3,20	3,19	3,21	3,19
1312	1,31	400	3,19	3,22	3,21	3,22	3,21	3,22	3,21	3,22	3,21
1282	1,28	410	3,21	3,24	3,23	3,23	3,22	3,24	3,23	3,23	3,22
1247	1,25	420	3,23	3,26	3,25	3,26	3,25	3,26	3,24	3,24	3,23
1171	1,17	430	3,27	3,28	3,26	3,27	3,26	3,27	3,26	3,26	3,25
1123	1,12	440	3,30	3,29	3,28	3,29	3,28	3,28	3,27	3,28	3,27
1104	1,10	450	3,31	3,32	3,31	3,31	3,30	3,30	3,29	3,30	3,29
1071	1,07	460	3,33	3,33	3,32	3,33	3,32	3,32	3,31	3,31	3,30
1060	1,06	470	3,33	3,35	3,34	3,34	3,33	3,34	3,33	3,32	3,31
1050	1,05	480	3,34	3,37	3,36	3,35	3,34	3,35	3,34	3,34	3,33
1017	1,02	490	3,36	3,38	3,37	3,37	3,36	3,36	3,35	3,35	3,34
993	0,99	500	3,37	3,39	3,38	3,38	3,37	3,37	3,36	3,36	3,35
985	0,99	510	3,38	3,40	3,39	3,40	3,39	3,38	3,37	3,37	3,37
960	0,96	520	3,39	3,41	3,40	3,42	3,41	3,40	3,39	3,38	3,37

Tabla 2: Resultado medición circuito completo

5.1.2.3 Pantalla LCD.

Se ha optado por una pantalla LCD de dieciséis columnas y dos filas, ya que es espacio suficiente para poder visualizar la medida, además de la facilidad de control mediante el software de Arduino, como se ha mencionado anteriormente. Se dispone de librerías específicas, en concreto "*LiquidCrystal*.h". Sin embargo, dado que la pantalla necesita de dieciséis pines para poder funcionar, se ha optado por la inclusión de un módulo de comunicación I^2C que reduce a cuatro los pines para su control. El montaje es como se puede ver en la figura 16.

Para el control de este dispositivo, existen librerías desarrolladas por terceros muy útiles como "*LiquidCrystal_I2C*.h", pero habrá de incluir una librería especifica de Arduino, la "*Wire*.h" que configura la comunicación serie de I^2C para el correcto funcionamiento de la aplicación.

5.1.2.4 Módulo SD.

Para el almacenamiento de los datos, se ha integrado un módulo SD que función mediante comunicación serie SPI. Para el control del mismo y al igual que la pantalla, se dispone de librerías específicas incluidas en el software de Arduino, la librería "SD.h". Es necesario incluir la librería "SPI.h" para la habilitar la configuración de la comunicación SPI.

La tarjeta donde se almacenen datos es necesario que esté en formato FAT32, que es el único formato que soporta el Arduino para escribir y leer ficheros. Los datos, como se ha mencionado anteriormente han de almacenarse en ficheros CSV, para facilitar la importación de los mismo a otros programas como el Excel o Matlab.

5.1.2.5 Montaje.

En primer lugar, mostraremos las conexiones de los diferentes elementos del circuito. Estas conexiones pueden observarse en la figura 28, para la representación se ha utilizado el programa de diseño de prototipado libre, con el fin de hacer una representación clara en caso de que se quiera reproducir el proyecto. Los esquemáticos se pueden consultar en el apartado 2. Planos.

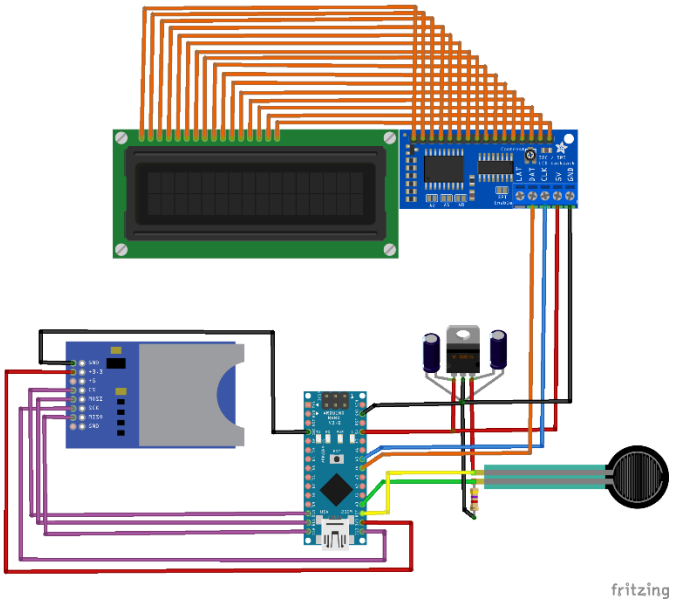


Fig. 28: Conexiones prototipo 1

Se ha diseñado una PCB específica para la aplicación mediante Easyeda, sin embargo, en la fabricación, se ha utilizado una placa taladrada de prototipos y las pistas se ha hecho con estaño. El ancho de pista de la PCB por donde se alimenta los distintos periféricos es de 0,5 mm además de contar con un plano de masa por cara, para reducir el ruido electromagnético.

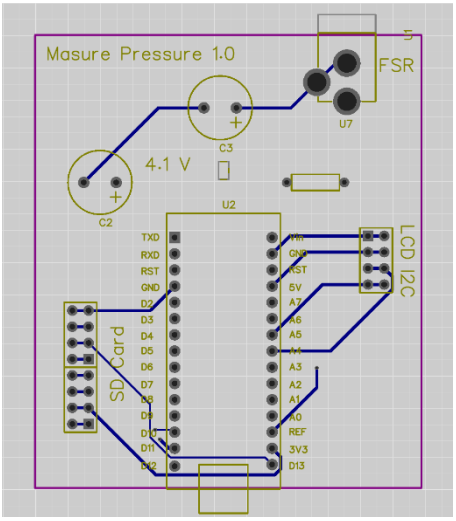


Fig. 29: PCB capa B

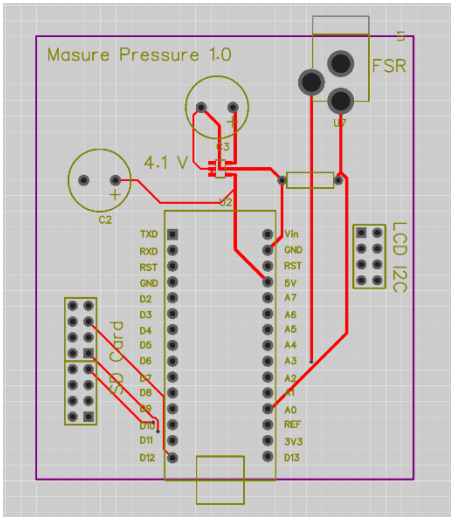


Fig. 30: PCB capa A

En las siguientes imágenes se puede observar el prototipo final funcionado, en el estudio preliminar con cadáveres.

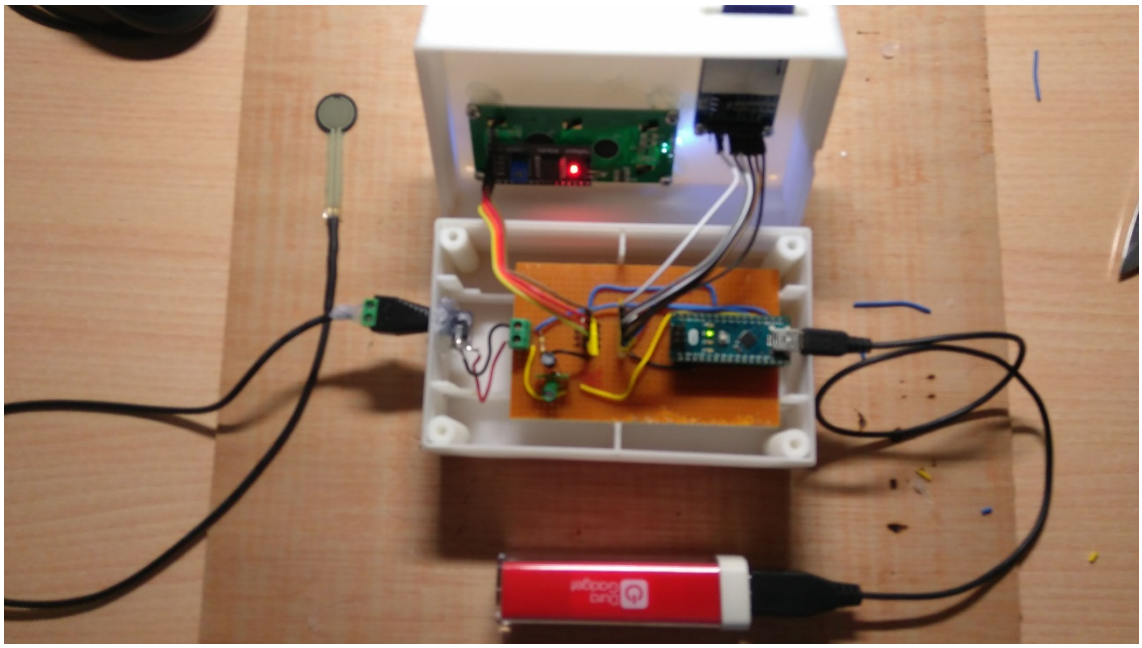


Fig. 31: prototipo 1

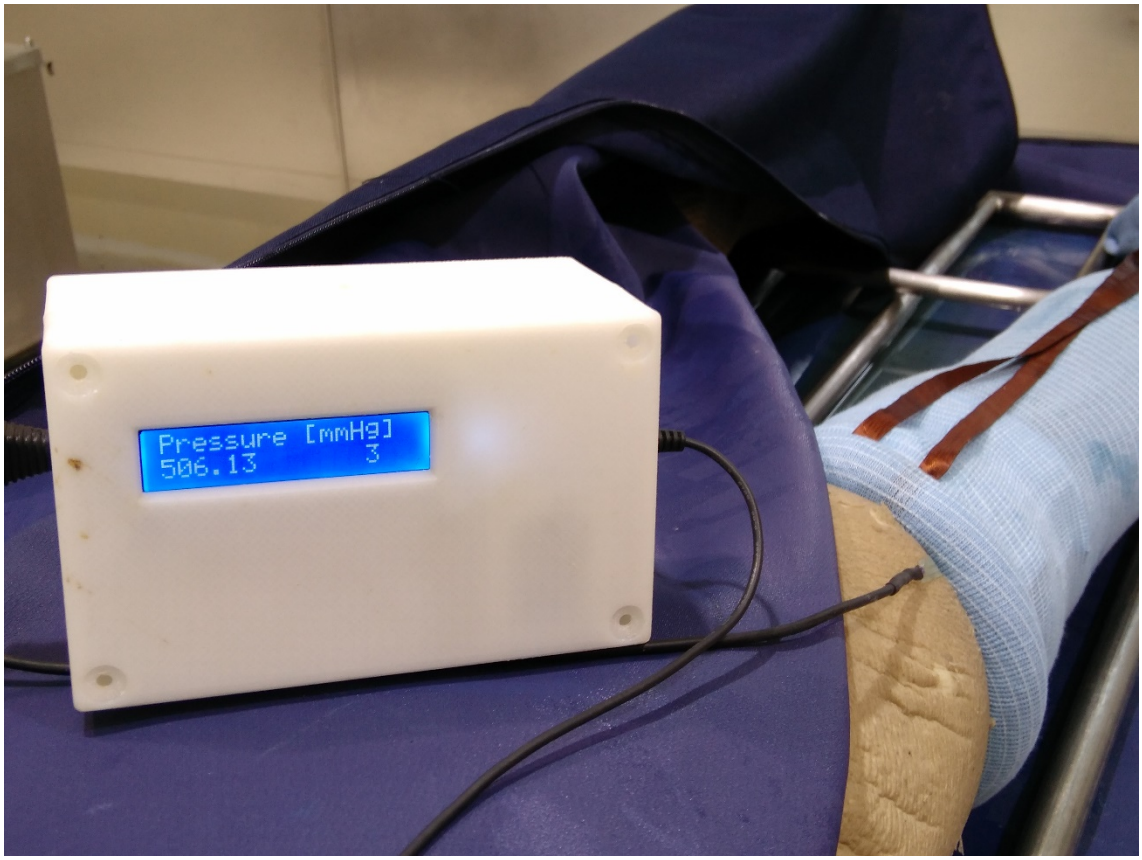


Fig. 32: prototipo 1 en estudio preliminar

5.1.3 Software desarrollado.

La estructura principal de un programa o Sketch en Arduino como se puede observar en la figura 33, consta de tres partes muy diferenciadas. En primer lugar, se ha de incluir las librería y declarar las variables globales que se van a utilizar para el programa, seguidamente está la función *viod setup*(), en esta sección, se incluyen todas las configuraciones de los periféricos y su inicialización. Es importante saber que esta parte del código solo se ejecuta una vez al principio y no se volverá a ejecutar hasta que se reinicie la tarjeta. Por último, está la sección de bucle infinito tan característica de los µC, el *void loop(),* donde se incluye todo código del programa que se desee ejecutar de forma cíclica.

Fig. 33: Esquema Sketch Arduino

5.1.3.1 *Librerías.*

```
12  /**** Libraries*****/
13
14  #include <SPI.h>
15  #include <SD.h>
16  #include <Wire.h>
17  #include <LiquidCrystal_I2C.h>
18
19  #include <avr/io.h>
20  #include <avr/interrupt.h>
21
22  #include <math.h>
```

SPI.h

Esta librería, esta incluida en el *software* de Arduino y permite utilizar el estándar de comunicación SPI.

SD.h

Librería incluida en Arduino, que se utiliza para controlar dispositivos de memoria externas como pueden ser las tarjetas SD y MicoSD

Wire.h

Librería íncluida en Arduino, habilita la comunicación I²C de la placa.

LiquidCrystal_I2C.h

Como se ha mencionado anteriormente en el escrito, esta librería sirve para el control de la pantalla LCD mediante comunicación I²C

avr/io.h y avr/interrupt.h

Estas librerías se ha de incluir cuando se desea utilizar las interrupciones en lo µC de AVR, en este caso como se va habilitar un *timer* para incluir los dato en la tarjeta SD son necesarias

5.1.3.2 *Constantes y variables globales.*

Las constantes utilizadas en este programa son las que se muestran a continuación:

```
#define AD 0.00400391 //ADC 4.10/1023.00 (10 bit AD)
#define Ra  4630
#define Vcc 4.10 //Power for FSR402 and ARef
```

En primer lugar, encontramos la constante AD, esta se utiliza para ahorra tiempo de ejecución al programa. Es el resultado de la conversión analógico-digital del Arduino, que como se ha mencionado anteriormente es de 10Bits. Esta

conversión permite obtener el valor de voltaje de la señal para poder obtener una relación de presión frente a conductancia, como se ha descrito en el apartado de acondicionamiento de señal.

La conversión analógico-digital funciona de la siguiente manera:

$$Vseñal = Lectura\_analógica \times \frac{Vcc}{1 - 2^n}$$

Esta fórmula nos permite obtener el valor real en voltios de una señal que es leída a través del conversor analógico-digital. Cuyos parámetros son:

- **Vseñal** → Valor en voltios de la señal de entrada a la tarjeta de adquisición de datos.
- **Lectura_analógica** → Valor analógico de la señal . En este caso oscila entre el 0 y 1023 correspondientes a los 10 bits del conversor.
- **Vcc** → Tensión de alimentación.
- ***n***→ número de bits del conversor, en este caso 10.

Por tanto, la constante AD, es el factor de conversión analógico-digital resultante de la operación:

$$AD = \frac{Vcc}{1 - 2^n} = \frac{4.1}{1023} \approx 0.00400391$$

Seguidamente tenemos los constantes Ra y Vcc, que se corresponden con los valores del divisor de tensión en ohmios y la tensión de alimentación del sensor en voltios respectivamente.

En cuanto a las variables utilizadas:

```
const int CS = 10; //comunicacion wiht sd, CS pin
volatile int seconds = 0; //for Timer1 interrupt (1s)
volatile byte mins = 0;
float analogin, cond, vin, rsensor, aux1 = 0, aux2 = 0, pressure;

int dato;
```

Hay que destacar que la variable CS, se corresponde con el pin de control del módulo SD, en este caso se corresponde con el pin digital D10. Por otra parte las variables tipo ***float,*** se corresponden con la obtención de la medida de presión. Por ultimo cabe destacar la variable tipo *volatile* utilizadas en el manejador de interrupciones internas del µC.

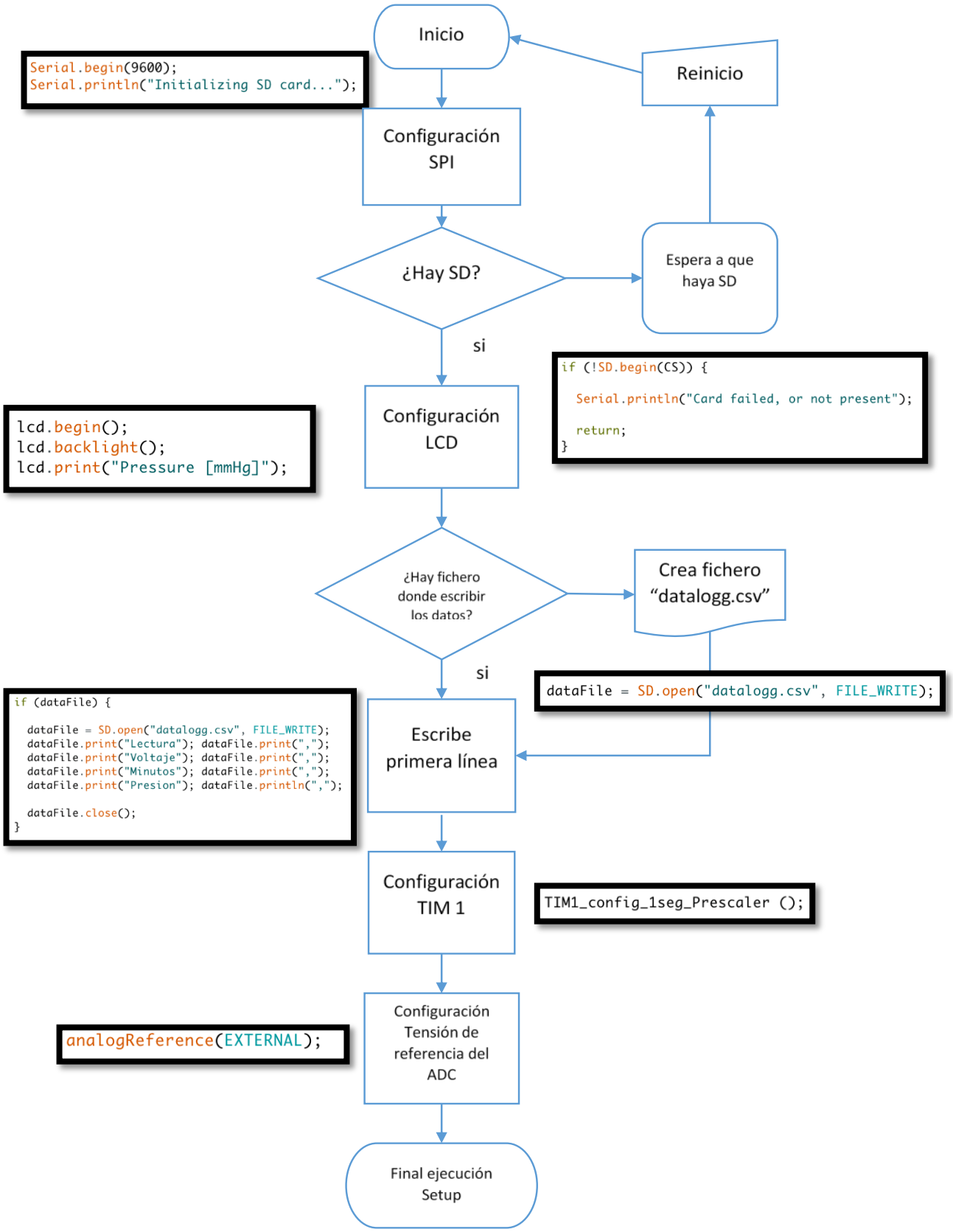
Por último, hay que comentar los objeto tipo File y Lyquid_Crystal2C, estos objetos son utilizados para la creación de ficheros mediante la librería SD y control de la pantalla LCD mediante I^2C.

```
LiquidCrystal_I2C lcd(0x27, 16, 2); //lcd object
File dataFile;//file object
```

Para crear el objeto LiquidCrystal_I2C se necesita configurarlo pasándole unos parámetros. En concreto se corresponde con la dirección de 7bits en hexadecimal de la pantalla LCD , en este caso es la 0x27 , el número de columnas y número de filas.

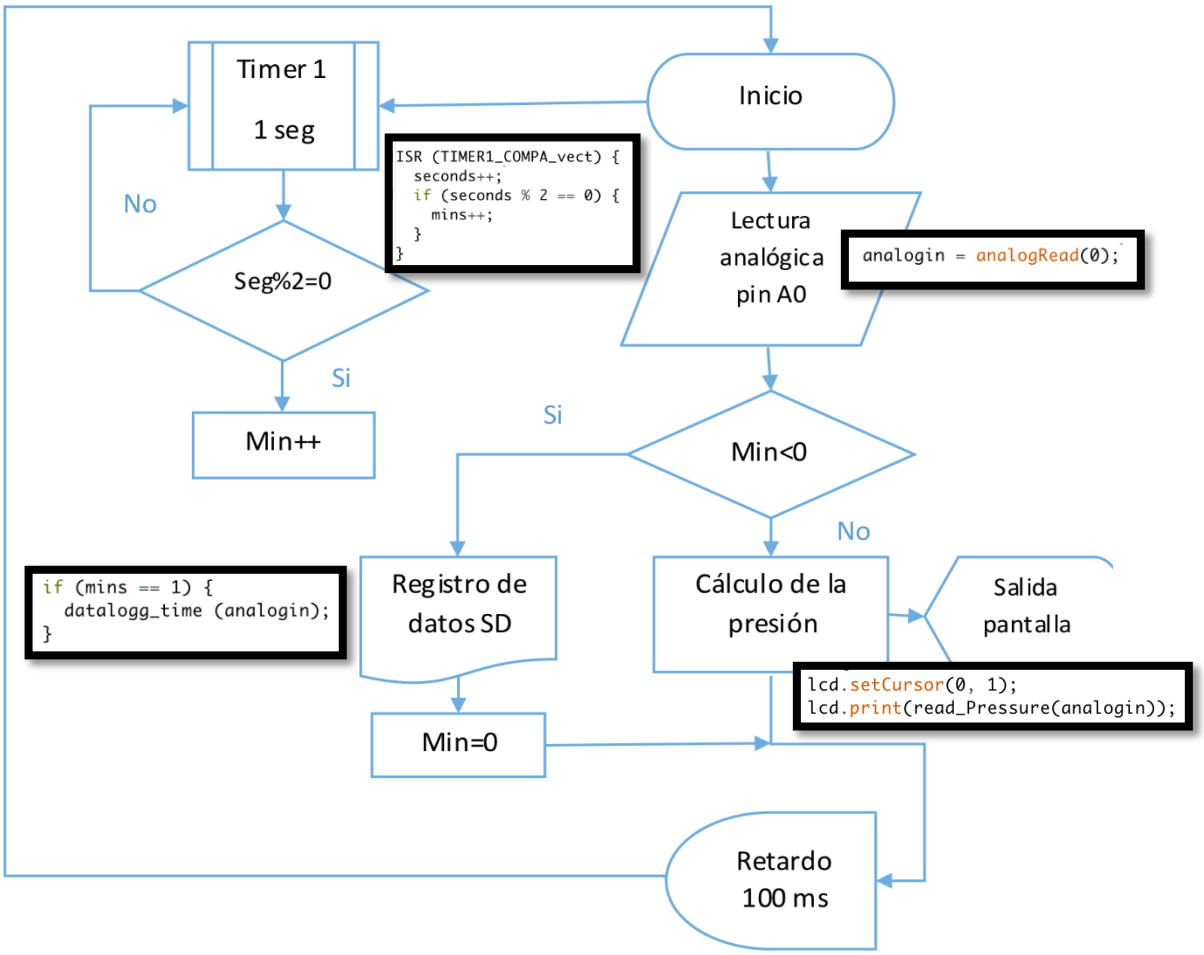
5.1.3.3 void setup()

Este es el flujograma del código para la configuración del Arduino.



5.1.3.4 *void loop()*

Flujograma del bucle infinito, con secuencia paralela del *timer 1*.



En el flujograma se puede observar que hay dos hilos de ejecución paralelos, un programa principal que toma medidas y las muestra por pantalla cada 100 milisegundos. El programa secundario se corresponde con el manejador de interrupción ISR() del *Timer 1,* que ejecuta el código cada segundo alterando una variable que cuando coincide con un número par modifica aumentado la variable min, esto provoca que en el hilo principal de ejecución se guarden las medidas en el fichero de datos en la SD.

5.1.3.5 Funciones realizadas.

Para la toma y almacenamiento de las medidas sean realizado funciones especificar

- **Para la lectura de datos:**

```
float read_Pressure (float y) {

  vin = AD * y;
  aux1 = (Vcc - vin);
  aux2 = Ra / vin;
  rsensor = aux1 * aux2;
  cond = (1 / rsensor) * 1000000;
  pressure = 0.5131 * cond;

  return (pressure);

}
```

Esta función es tipo *float,* hay que pasarle un parámetro *float,* que al caso será la lectura analógica.

Las siguientes líneas de código se encargan de ejecutar la función de transferencia calculada en el apartado para obtener la medida de presión en mmHg..

Se han utilizado parámetros auxiliares para no perder precisión en los cálculos y reducir el tiempo de ejecución de la instrucción.

El proceso de cálculo es el siguiente:

Primero se obtiene la tensión de entrada por el pin analógico A0. Después se calcula la resistencia del sensor para sacar la conductancia que se pasarán a microsiemens (µS) para no perder resolución de la medida y por último se obtendrá la presión en mmHg.

$$R_{sensor} = \frac{Ra \times (Vcc - Vin)}{Vin}$$

$$Cond[\mu S] = \frac{1}{R_{sensor}}$$

$$Pressure[mmHg] = 0{,}5131 \times Cond\ [\mu S]$$

- **Para el registro de datos en la SD**:

```
void datalogg_time (float x) {

  vin = AD * x;
  aux1 = (Vcc - vin);
  aux2 = Ra / vin;
  rsensor = aux1 * aux2;
  cond = (1 / rsensor) * 1000000;
  pressure = 0.5131 * cond;

  dato++;

  dataFile = SD.open("datalogg.csv", FILE_WRITE);

  dataFile.print(x);
  dataFile.print(",");
  dataFile.print(vin);
  dataFile.print(",");
  dataFile.print(dato);
  dataFile.print(",");
  dataFile.print(pressure);
  dataFile.println(",");

  dataFile.close();

  lcd.setCursor(13, 1);
  lcd.print(dato);
  mins = 0;
}
```

Esta función es tipo *void* ya que no devuelve ningún valor, pero al igual que la función anterior, hay que pasarle un parámetro *float.*

Como en Arduino una función no puede ejecutar otras funciones, es necesario calcular la presión antes de anotar los datos en el fichero "*datalogg.csv*".

En le fichero se almacena los parámetros de lectura analógica, tensión de entrada, numero de tomas hachas y la presión.

Por último, se cierra el fichero, se muestra por pantalla, en concreto en la esquina inferior derecha el nuero de tomas almacenadas y se pone la variable min a cero.

Cabe destacas que la función SD.open(" ",FILE_type), abre el fichero y si no existe lo crea, en caso de volver a ejecutar el programa si ya existía previamente el fichero dentro de la memoria SD de alguna sesión anterior, el fichero se continuaría escribiendo sin sobrescribir los datos previos.

- **Configuración del *timer* y manejador de interrupción**.

Un *timer* se puede definir como un contador interno del µC que cuenta a una frecuencia determinada por el reloj del sistema, este puede ser interno o externo. En el µC ATmega328P hay tres tipos de *timer* :

Timer 0: Este es un temporizador de 8bits, es decir puede registrar como máximo 256 valores. Este se es utilizado por las funciones reservadas de Arduino delay() y millis(), por lo que no es conveniente utilizarlo en esta aplicación, ya que podría intrferir en la ejecución normal el programa.

Timer 1: Temporizador de 16 bits, registra como máximo 1024 valores. Este es utilizado en la librería *servo.h.*

Timer 2*:* Temporizador es de 8 bits, utilizado por la función tone().

Se utilizara en este caso el *timer 1* , ya que el prototipo no utiliza servomotores, además de tener un capacidad de 16bits. En cuanto al funcionamiento, es muy sencillo el *timer* se ejecuta y cuando acaba de contar levanta un *flag* para que el µC lo atienda, en ese momento se reinicia.

Para configurar el *timer1* es necesario atacar a los registros *TCCR1A, TCCR1B y TIMSK*

15.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 15-4. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP

15.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 15-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

15.11.8 TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

s

- **Bit 1 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (see “Interrupts” on page 57) is executed when the OCF1A Flag, located in TIFR1, is set.

- **Bit 0 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow interrupt is enabled. The corresponding Interrupt Vector (See ”Interrupts” on page 57) is executed when the TOV1 Flag, located in TIFR1, is set.

Fig 34: Registro del timer1 AVR Atmega328P

La función de configuración del timer1 a 1 segundo es la siguiente:

```
void TIM1_config_1seg_Prescaler() {

  /* Internal interrupt timer1 16 Bits for comparing, see register on AVR datasheet */

  cli();                  //clear all interuptions.

  TCCR1A = 0;
  TCCR1B = 0;

  OCR1A = 15624;          // value to compare the 1024 prescaling to 1s
  TCCR1B |= (1 << WGM12);

  TCCR1B |= (1 << CS10);
  TCCR1B |= (1 << CS12);

  TIMSK1 |= (1 << OCIE1A);

  sei();//set the interruptions.
}
```

En primer lugar se llama a la función cli(), esta deshabilita todas la interrupciones globales para que no interfieran en la configuración del *timer* . Las siguientes líneas de código limpian, es decir, ponen a 0 los registros TCCR1 para comenzar la configuración.

Se pretende configura el *timer* a un segundo, por lo que se necesita de un preescalado ya que si no configuramos nada la frecuencia de contaje será la del propio reloj de cuarzo del Arduino 16Mhz.

$$T = \frac{1}{f} = \frac{1}{16Mhz} = 62{,}5ns$$

Esto significa que la cuenta la acabará dado que el *Timer 1* es de 16bis en:

$$T_{ovf} = (2^{16} - 1) \times 62{,}5ns \cong 4{,}1ms$$

Prescalar el *timer* alterando los bits CS10, CS12 del registro TCCR1B para obtener e preescalado de 1024.

$$T = \frac{1}{f/prescaler} = \frac{1}{16Mhz/1024} = 64\mu s$$

$$T_{ovf} = (2^{16} - 1) \times 64\mu s \cong 4{,}19s$$

Dado que este tiempo de cuenta es excesivo, es necesario calcular un valor para que el *timer* deje de contar al segundo, en vez de a los 4,19s. Ese es el parámetro OCR1A y se calcula:

$$Valor\ CTC = \frac{T_{deseado}}{T} - 1 = \frac{1s}{64\mu s} - 1 = 15624$$

Por último al acabar de configurar debidamente el *timer 1*, se habilitan otra vez las interrupciones globales mediante la función “sei()” .

Como los *timers* son interrupciones internas, es necesario habilitar un servicio de interrupción, que en los µCs AVR se declara mediante la función ISR (_interrupt). En este caso, hay que habilitar la rutina del servicio de interrupciones para el timer1, configurado anteriormente mediante la función que vemos a continuación.

```
ISR (TIMER1_COMPA_vect) {
  seconds++;
  if (seconds % 2 == 0) {
    mins++;
  }
}
```

El funcionamiento de la rutina de interrupción se representa en el siguiente flujograma, figura 35.

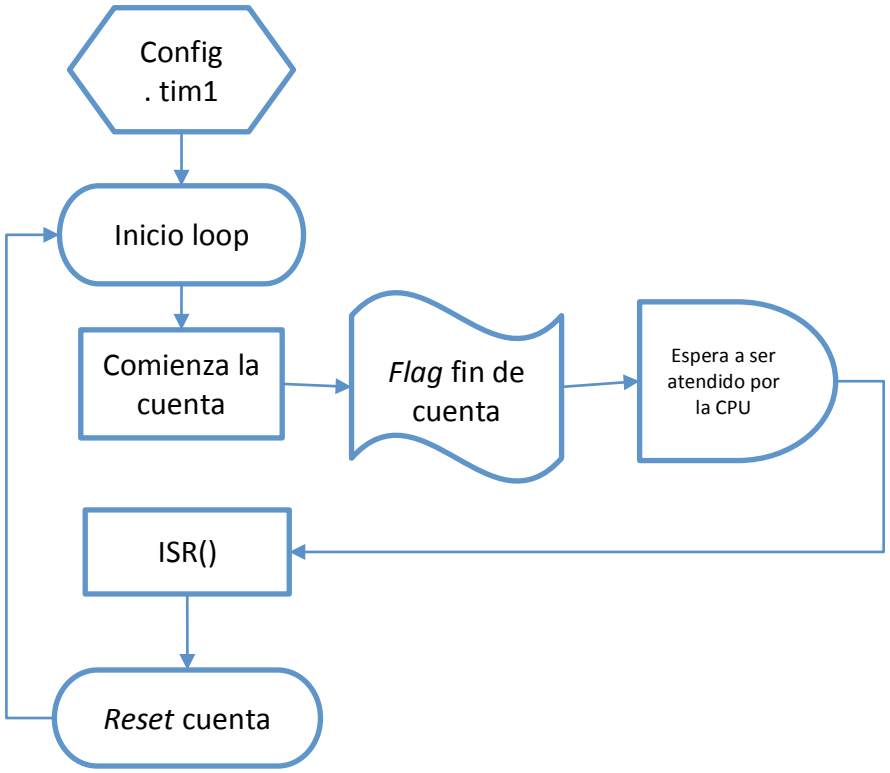


Fig 35: flujograma funcionamiento interrupciones Arduino.

El código completo puede consultarse en el Anexo.

5.2 Prototipo 2

5.2.1 Descripción.

El segundo prototipo diseñado es un dispositivo electrónico digital basado en el uC de 8 bits, ATmega328p de Atmel, integrado en una tarjeta Arduino Nano. El sensor analógico utilizado es el modelo A201 de Tekscan como el mostrado anteriormente en la figura 8, cuenta con una pantalla LCD de 16x2, donde se muestra la medida en tiempo real, además de almacenar la medida en una memoria SD cada dos segundos. la alimentación se utiliza el mismo banco de carga externo de 5000 mA, regulado a 5V, para asegurar el buen funcionamiento del dispositivo.

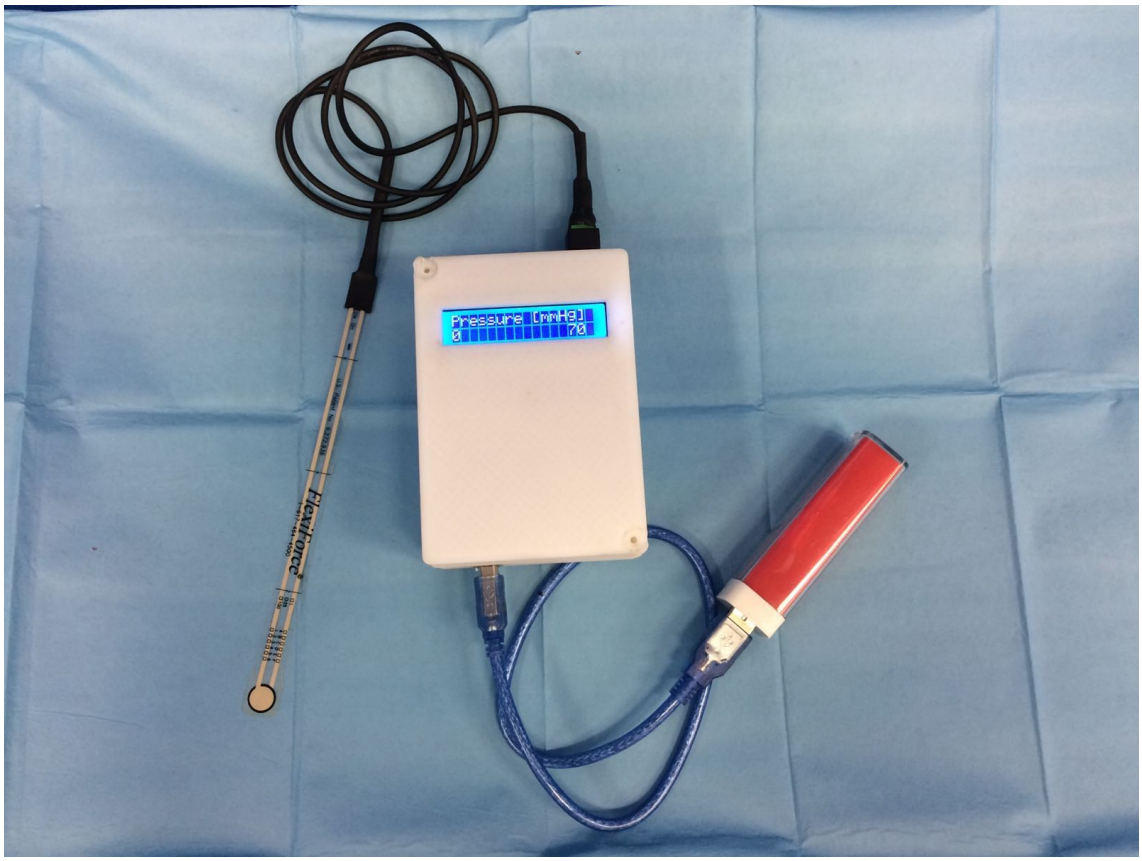


Fig. 36: Prototipo 2

El prototipo 2 es muy similar al anterior, únicamente se ha cambiado el sensor y el circuito de adaptación de señal, por lo que en los siguientes apartados se desarrollara únicamente los elemento que difieren con el prototipo anterior ya que el resto es completamente igual.

5.2.2 Hardware desarrollado.

El hardware del prototipo utiliza el mimo modulo SD y pantalla que el anterior, por ello este apartado se centrara en el sensro A201 y el circuito de adaptación de señal.

5.2.2.1 Sensor A201 y obtención de la señal.

El sensor A201 de Tekscan, funciona básicamente como el sensor FSR402, altera su resistencia eléctrica al aplicar presión sobre el área activa. El rango de resistencia que abarca, oscila entre 10 megaohmios y cien kilohmios [10MΩ - 100kΩ], soporta fuerzas puntuales hasta once kilogramos, más allá de ese rango, satura.

Las características físicas del A201, presentan unas mejoras con respecto al utilizado en el prototipo anterior. Es más fino, los elementos conductores están hecho con aleación de plata, además se de ser menos susceptible a roturas. Está formado por diferentes capas como se muestra en la figura 37

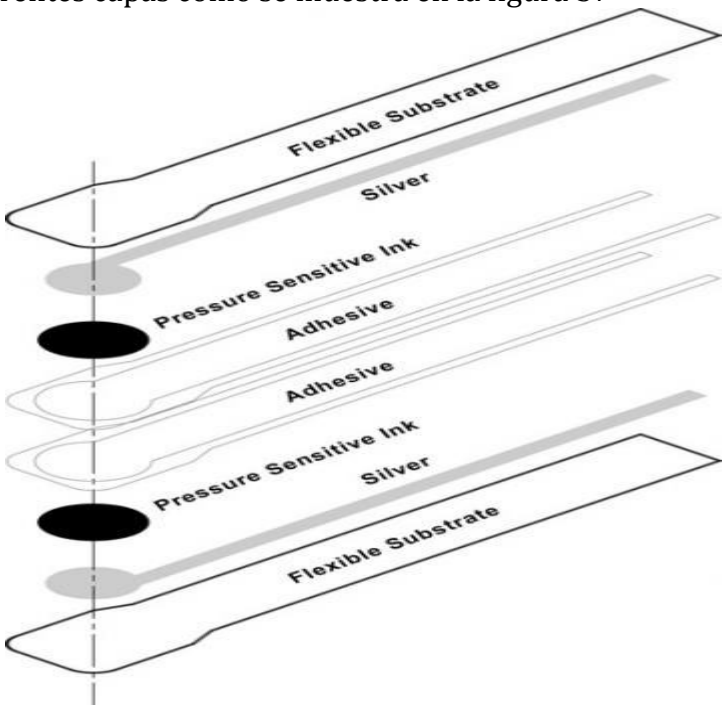


Fig. 37: Sensor A201

En cuanto a la respuesta del sensor con respecto a la presión, es muy similar al FSR402. El fabricante facilita en el manual de usuario, las gráficas con la respuesta tanto de resistencia como de conductancia, además de proporcionar algunos circuitos de adaptación de señal, como se puede observar en la figura 38.

Cabe destacar que la diferencia principal con el anterior sensor es su respuesta en tensión, ya que presenta una tendencia lineal, lo que facilita el calibrado una vez diseñado el circuito de acoplamiento.

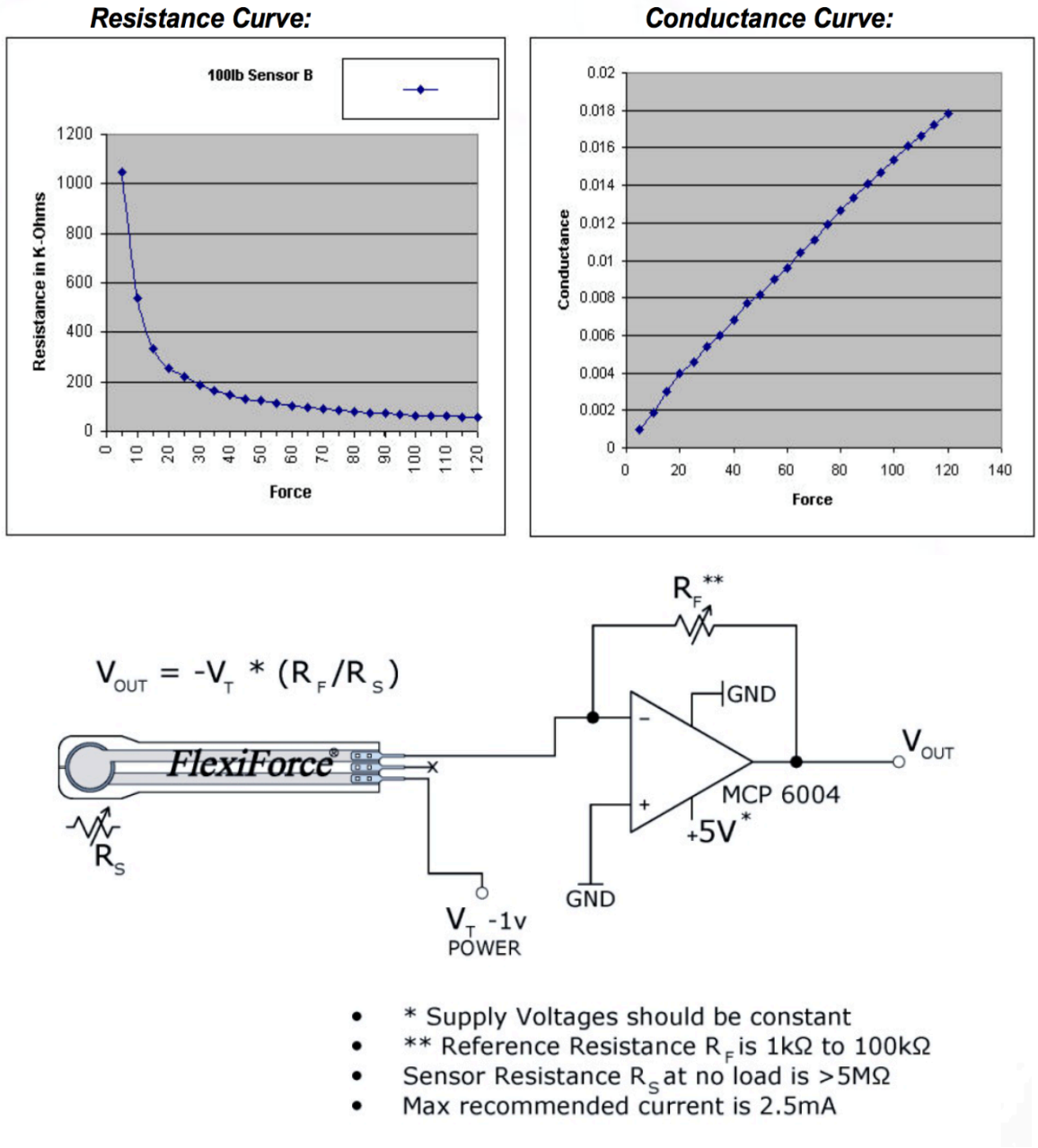
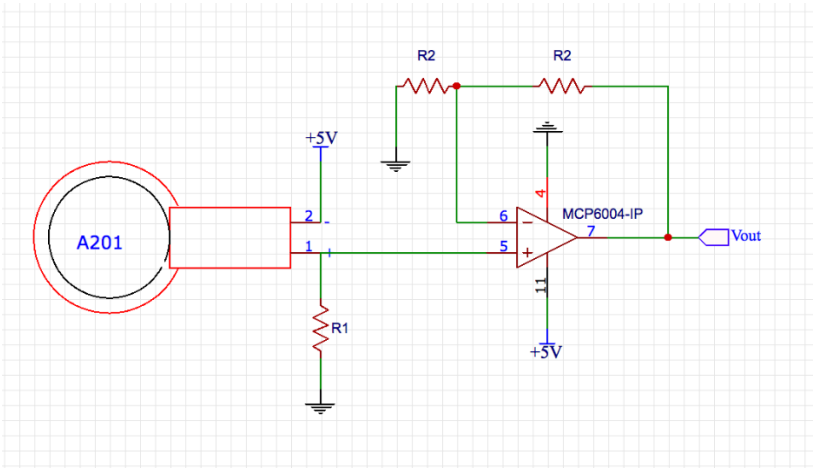


Fig 38: Curvas de respuesta y circuito de acoplamiento del sensor A201 dados por el fabricante

5.2.2.2 Calibrado.

A pesar de que el fabricante recomienda el circuito mostrado anteriormente, para el diseño de este dispositivo se ha optado por un circuito amplificador no inversor, ya que la respuesta de señal es del orden de 10 veces menor que la del sensor FSR 402.

Para el dimensionamiento del circuito se procederá de forma similar a la del dispositivo anterior:



En primer lugar, se precederá al cálculo de los diferentes elementos del circuito.

1) Aplicando el teorema de Millman en las parta del amplificador operacional se obtienen las siguientes ecuaciones:

$$V^+ = \frac{\frac{Vcc}{Rsens}}{\frac{1}{R1} + \frac{1}{Rsens}}$$

$$V^- = \frac{1}{2} * Vout$$

Al igualar los potenciales en ambas patas $V^+ \approx V^-$ se obtiene la función de transferencia :

$$Vout = 2 * Vcc \frac{R1}{Rsens + R1}$$

2) Una vez obtenida la función de transferencia del circuito se procede al cálculo de la R1, ya que el valor del R2 se fijará a 10kΩ.

P=0 → Vout=0

Pmax→ Vout = 4,9 V

De sustituir los valores en la fdt, se obtiene que :

$$\boxed{\text{R1=96,36k}\Omega \approx R1normalizada = 100k\Omega}$$

5.2.2.3 Calibrado

El fabrícate recomienda que antes de comenzar con la calibración, se ha de preacondicionar el sensor, sometiéndolo al 110% de la carga máxima que va a soportar. Para sacar su línea de tendencia para relacionar la magnitud física a medir, en este caso presión, con respecto a la tensión generada. Es necesario sacar cinco puntos y en caso de que utilizarse el circuito recomendado únicamente tres. Por lo que se han obtenido los siguientes datos con su consiguiente grafica que relación tensión y presión en diferentes sensores por si algún sensor se rompe tener varios calibrados. Véase tabla 3 y figura 39.

Proceso de calibracion sensor de furza resisitivo A201 de FlexiForce					
Paso	Descripción	Presión [mmHg]	Repeticiones	tiempo [min]	Medida [V]
1	Precalentar	580	5	30 seg	sensor 1
2	Calibrado	120	1	1	3,30
		240	1	1	3,35
		360	1	1	3,39
		480	1	1	3,42
		600	1	1	3,44
1	Precalentar	580	5	30 seg	sensor 2
2	Calibrado	120	1	1	3,35
		240	1	1	3,42
		360	1	1	3,50
		480	1	1	3,51
		600	1	1	3,56
1	Precalentar	580	5	30 seg	sensor 3
2	Calibrado	120	1	1	3,28
		240	1	1	3,30
		360	1	1	3,32
		480	1	1	3,37
		600	1	1	3,39
1	Precalentar	580	5	30 seg	Sensor 4
2	Calibrado	120	1	1	3,29
		240	1	1	3,31
		360	1	1	3,33
		480	1	1	3,34
		600	1	1	3,36

Tabla 3: Calibrado A201

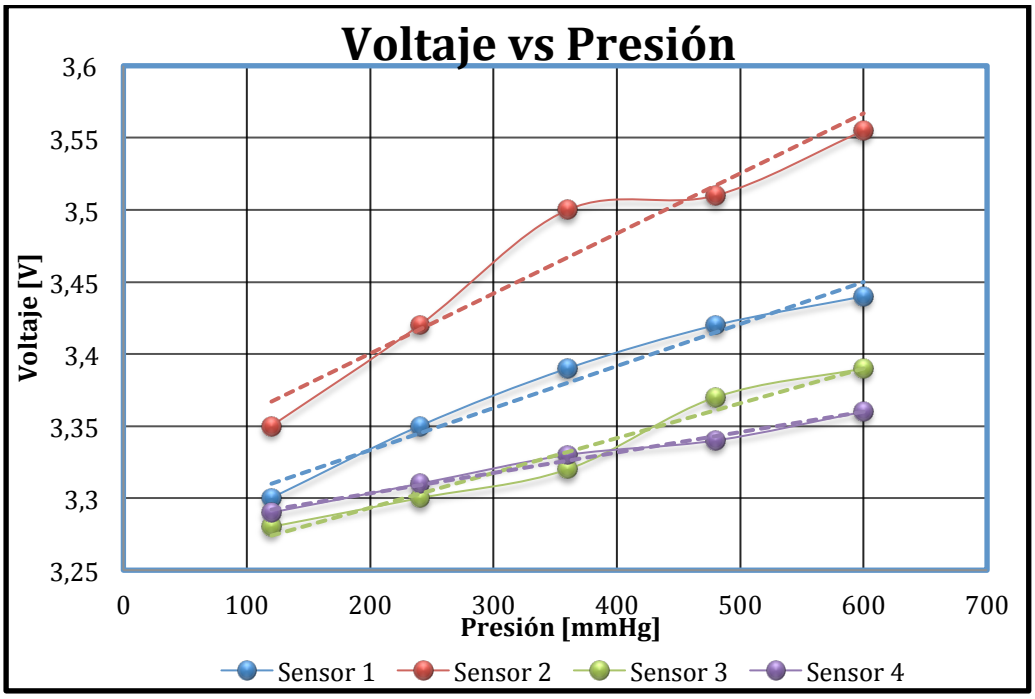


Fig. 39: Calibrado A201

Se han obtenido las siguientes funciones de transferencia para los diferentes sensores calibrados:

sensor 1	V = 0,0003P + 3,275	R^2 2=0,97222
sensor2	V=0,0004P + 3,317	R^2 =0,94056
sensor 3	V=0,0002P + 3,245	R^2 =0,97000
Sensor 4	V = 0,0001*P + 3,275	R^2 = 0,98973

5.2.3 Montaje

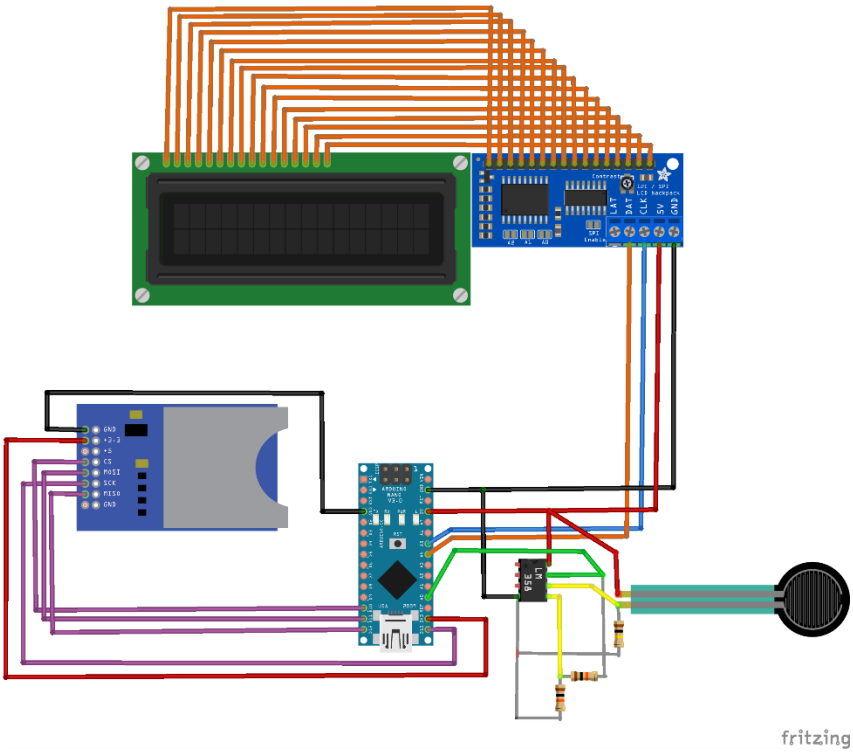


Fig. 40: Conexionado prototipo 2

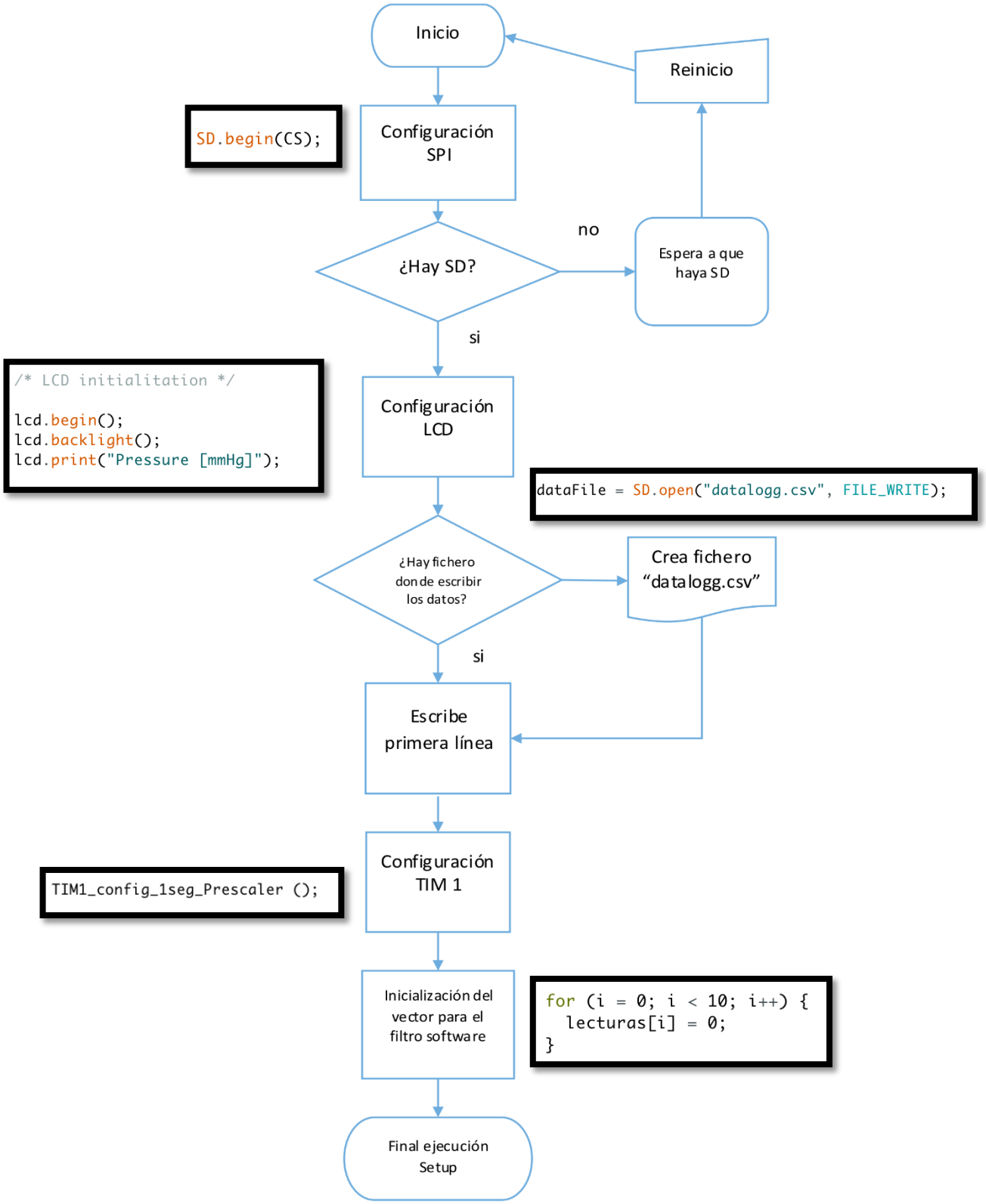
Como se puede apreciar en la figura 40, el conexionado del circuito es muy similar al del prototipo 1, a excepción del circuito da acoplamiento de señal al pin A0, que se puede distinguir con los colores marrillo y verde. Puede consultarse los esquemático en él.

5.2.4 Software desarrollado

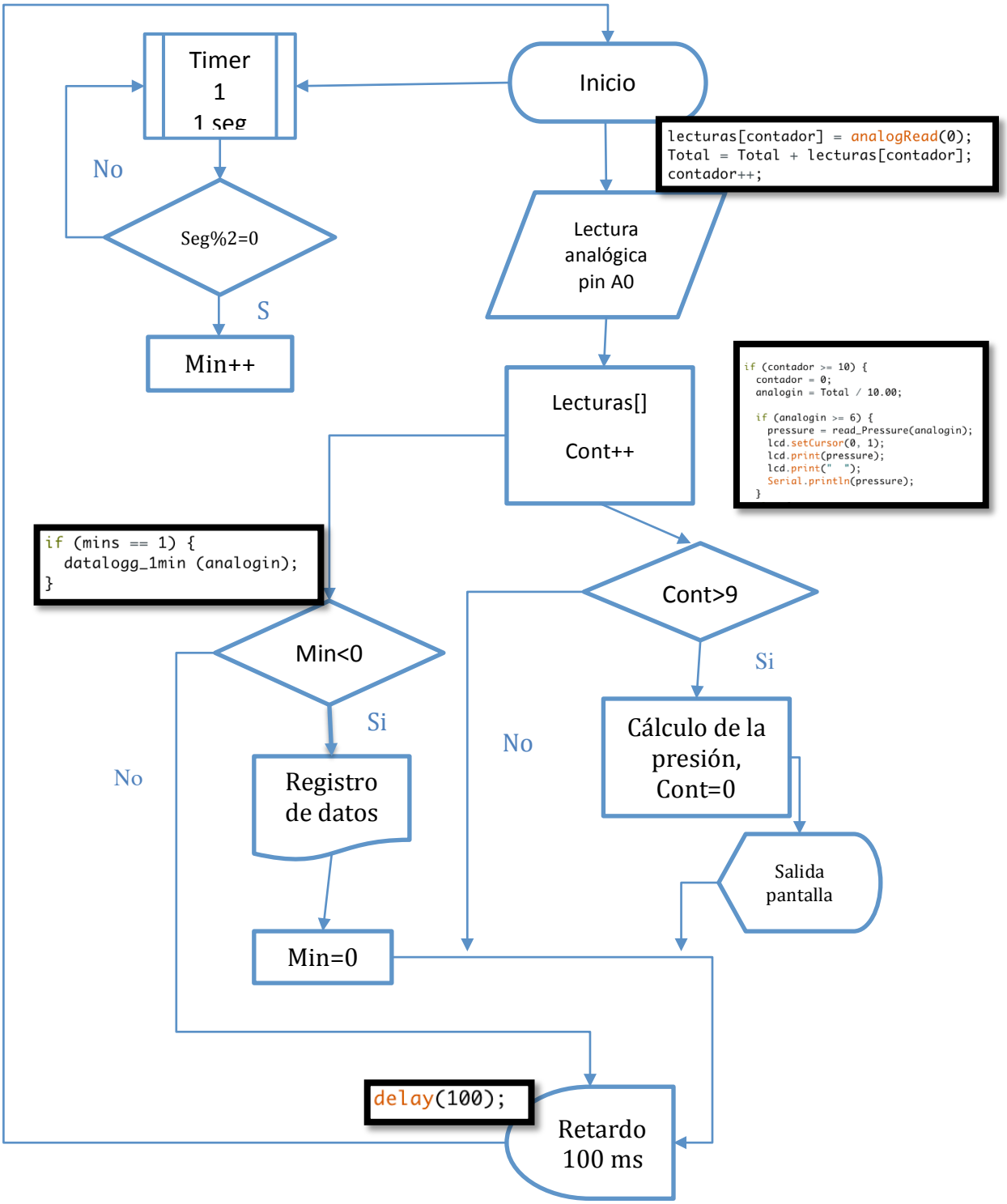
La programación de este dispositivo, al utilizar los mismos elementos, es idéntica a la del prototipo 1. Por ello, solo se va a explicar médiate flujogramas el funcionamiento del *void setup()* y *void loop()*., prescindiendo de las librerías ya que son las mismas.

Dadas las características del sensor, la lectura analógica es susceptible de captar ruido, por lo que se le ha añadido un filtro software. Este filtro toma diez medidas y saca una media, evitando así que la medida “baile”, mostrando una lectura estable.

5.2.4.1 *Void setup()*



5.2.4.2 Void loop()



Puede consultarse el código en el anexo 2

5.3 Prototipo 3

5.3.1 Descripción.

El tercer prototipo diseñado es un dispositivo electrónico digital basado en el µC de 8 bits, ATmega328p de Atmel, integrado en una tarjeta Pro Trinket de Adafruit. Se ha utilizado el sensor de presión manométrica de aire ABPMANT100PG2A3 de HoneyWell como el mostrado anteriormente en la figura10. La medida se muestra en una pantalla OLED de 1,3". En cuanto a la alimentación, se ha incorporado una batería LIPO de 3,7V y capacidad de 2000mAh, además de un módulo de carga Sparkfun, que permite la carga de la misma y alimentar el circuito.

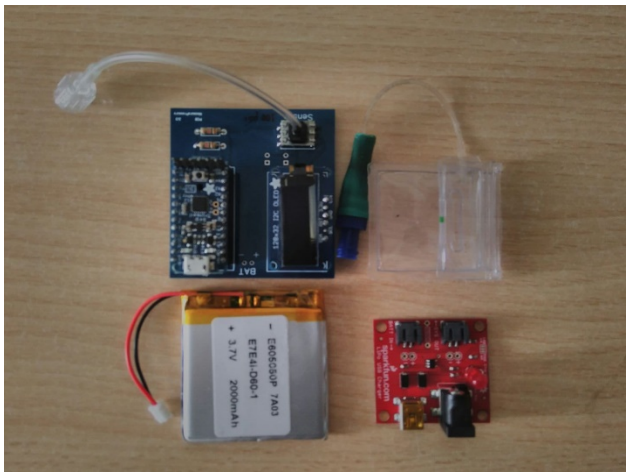


Fig.41: Partes del prototipo 3

Nótese que el dispositivo está en fase de construcción, sin embargo, se ha probado en quirófano con resultados satisfactorios, por lo que concluido este trabajo se seguirá desarrollando para su posterior utilización.

5.3.2 Hardware desarrollado.

5.3.2.1 Pro Trinket de Adafruit.

La tarjeta Pro Trinket 3.3V, es un dispositivo de prototipado rápido diseñado y fabricado por Adafruit. Lleva integrado el µC Atmega 328P de Atmel. El atractivo de esta tarjeta es su tamaño, potencia y compatibilidad con el software de Arduino. La placa presenta similitudes con las utilizadas en los prototipos anteriores a excepción de que la tensión de funcionamiento es de 3,3V , hecho que facilita la incorporación de una batería LIPO al circuito sin necesidad de añadir un módulo de adaptación de potencia; y la frecuencia del reloj interno es 12MHz. Además de proporcionar hasta 150 mA en las patillas de salida. En cuanto al patillaje se distribuye como se puede ver en la figura42.

Hay que tener en cuenta a la hora de programar la placa con la interfaz de Arduino, que la placa no posee auto-reset por lo que habrá que presionar el botón de reset cada vez que se quiera modificar el programa. Se podrá cargar un programa nuevo mientras el led rojo de la placa este parpadeado, su duración es

de 10 segundos aproximadamente, una vez trascurrido ese tiempo, si se desea modificar algo más del programa habría que repetir la operación anterior.

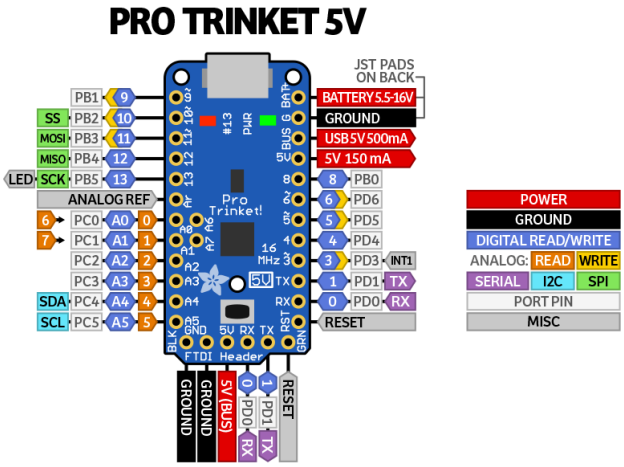


Fig 42: Patillaje Pro Tinket 5V

Como puede observarse, la Pro Tirnket dispone de los protocolos de comunicación I²C y SPI por lo que se simplificará el circuito al máximo.

5.3.2.2 Pantalla OLED.

Este tipo de pantalla son de muy reducido tamaño. En concreto, este modelo de Adafruit mide menos de una pulgada de diagonal, sin embargo, aunque se de reducido tamaño, los elementos proyectados en la misma se ven con mucha claridad. En la figura 43, puede observarse la diferencia de tamaño con respecto a la pantalla utilizada en lo prototipos anteriores.

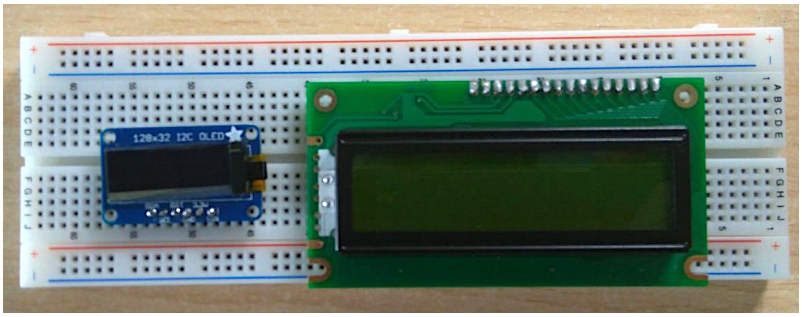


Fig 43: Diferencia de tamaño Monocrome vs LCD

Por otra parte, admite la comunicación I²C y el fabricante ha desarrollado librerías específicas para Arduino, lo que facilita aún más su control. Hay que destacar que la pantalla está preparada para funcionar a 3,3V de estado alto, pero el fabricante incorpora un pin de 5V para que se compatible con los µC que funciones a 5V.

5.3.2.3 Sistema de Alimentación.

Las baterías LIPO o de polímero de litio, son ligeras y albergan una gran capacidad de carga. Las hace ideales para aligerar el peso de algún proyecto portátil, como puede ser un cuadricóptero casero.

Hay que tener especial cuidado a la hora de utilizar este tipo de baterías, ya que pueden llegar a arder si se sobrecargan o calientan demasiado. Por ello, se ha decidido incluir un módulo especificó de carga para este tipo de baterías evitando así que puedan surgir problemas. Véase la figura 44.

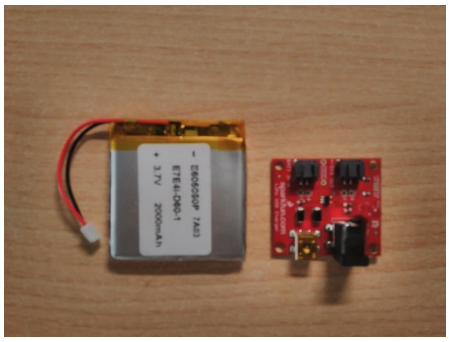


Fig 44: Batería y módulo de carga

5.3.2.4 Sensor de aire.

En este caso se ha optado por un sensor de presión de aire modelo ABPMANT100PG2A3 (figura 10) de Honeywell. Este sensor es digital por lo que viene calibrado de fábrica, presenta un error en fondo de escala de 1,5%.

Se comunica mediante I2C, por lo que para incluirlo en un circuito únicamente es necesario poner dos resistencias de 1kΩ conectadas a la fuente de alimentación, que en este caso será de 3,3V. Para obtener la información de medida del sensor, hay que proceder de la siguiente forma:

El fabricante facilita en el *datasheet* la forma en la que manda los datos, como puede verse en la figura 45.

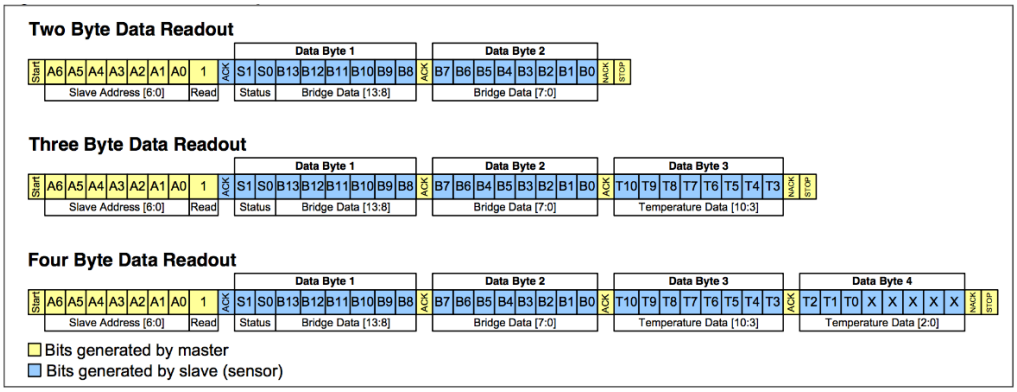


Fig 45: Datos transmitidos por el sensor vía I2C

El sensor envía información en forma de 14 bits. El modelo no dispone de compensación de temperatura. Por ello, hay que fijarse en el primer diagrama de la figura 45, donde se puede ver que el primer tramo de bits se corresponde con la dirección del sensor, que solo se puede configurar como esclavo. Los 16 bits restantes se corresponden con el valor de la lectura.

En el *Data Byte 1*, los dos primeros bits se corresponden con el estado de la medida del sensor, que son los que informan de si la medida es correcta o está habiendo algún tipo de problema en la transmisión. Existen varios estados posibles come se indica en la tabla 4.

Status Bits		Definition
S1	S0	
0	0	normal operation, valid data
0	1	device in command mode*
1	0	stale data: data that has already been fetched since the last measurement cycle, or data fetched before the first measurement has been completed
1	1	diagnostic condition

Tabla 5: Status bits

Si S1 y S2 se corresponden con el valor 00, indica que la medida se ha obtenido correctamente. El resto de valores no interesan de cara a la esta aplicación.

La medida la componen los últimos 14 bits, comprendidos entre la data *byte1* y *la data byte2,* según el *datasheet* la medida se obtiene de la siguiente ecuación, figura 46.

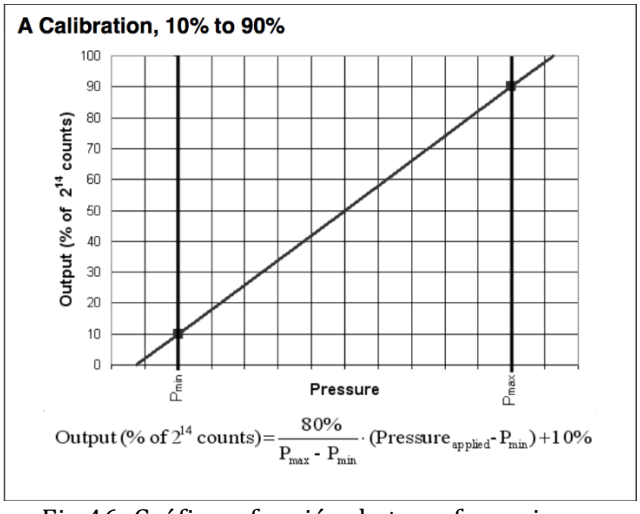


Fig 46: Gráfica y función de transferencia.

El rango de presiones del sensor es de [0 - 100]psi o [0 - 5171] mmHg, por lo que los parámetros Pmin y Pmax coincidirán con estos valores. En cuanto a 80% y 10% se corresponde con los valores decimales del porcentaje correspondiente al valor 16384 (2^{14}).

Valores:

- Pmax = 5171 mmHg
- Pmin = 0 mmHg
- 80% = 13107,2
- 10% = 1638,4
- Output(%)→ Es lo que llega al µC este valor oscila entre 0 y 16384 . Se utilizará para obtener l medida deseada (*Pressure applied)*

En el apartado de *software* desarrollado se muestra como realizar estas operaciones con µC. La gran ventaja de este tipo de sensores es que puedes obtener el valor en las unidades que se desee, únicamente hay que hacer la conversión de psi a la unidad deseada y aplicarla en la función de transferencia.

5.3.2.5 Montaje.

En cuanto al montaje, es similar a los descritos anteriormente, únicamente se utilizarán los pines de comunicación I^2C y SPI. Se puede consultar el esquemático en el apartado 2.Planos

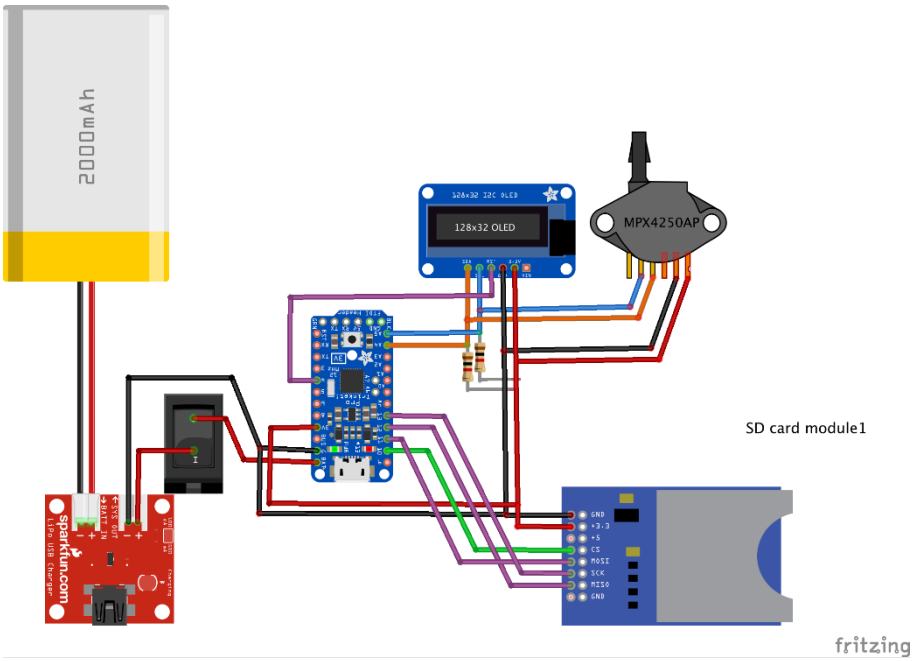


Fig. 47: Diagrama de conexiones prototipo 3

Nótese que el sensor no coincide con el modelo escogido, pero al caso nos sirve, ya que se ha conectado siguiendo el patillaje del modelo correcto y cumple la función de aclarar el montaje completo.

Se ha diseñado una PCB específica para este prototipo y se ha fabricado, con la intención de incluirlo en el dispositivo completado. El diseño de la PCB se ha realizo con la herramienta online de Easyeda con el siguiente resultado: véase figura 48.

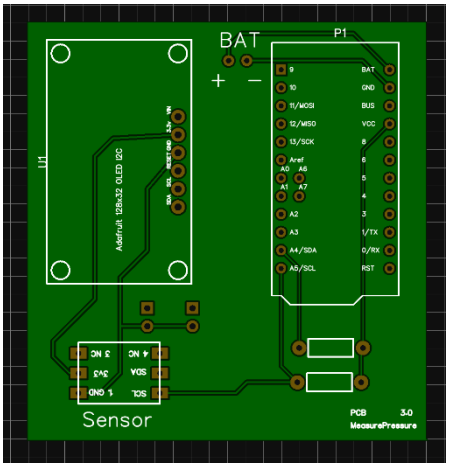


Fig 48: PCB prototipo 3

Para trasmitir la presión al sensor se ha utilizado conducciones normales de gotero, una llave de tres pasos y una bolsa recipiente que debe ponerse debajo del torniquete.

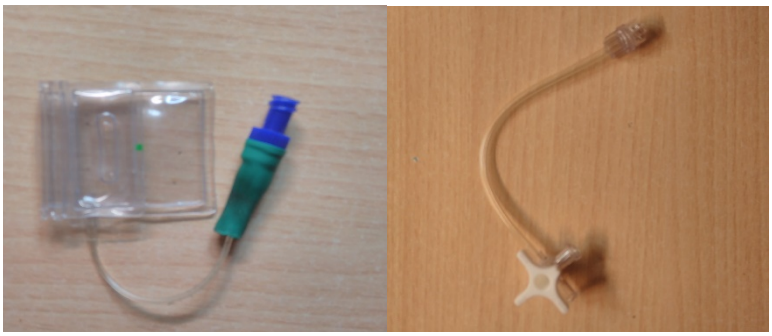


Fig. 49: Bolas contenedora y llave de tres pasos

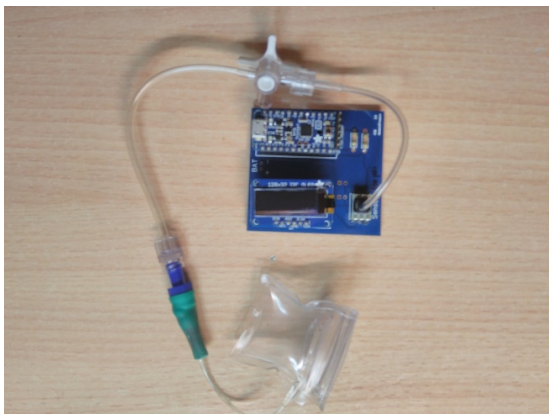


Fig 50: Trasmisión de presión al sensor completo

5.3.3 Software desarrollado

La programación de este dispositivo es similar a las anteriores, la diferencia principal reside en el proceso de obtención de la medida I²C y que las librerías incluidas son diferentes ya que se trata de otra tarjeta y pantalla.

5.3.3.1 Librerías.

Las librerías son básicamente para configurar la pantalla y la compatibilidad de programación de la tarjeta con Arduino.

```
#include <TinyWireM.h>
#include <USI_TWI_Master.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <gfxfont.h>

#include <stdint.h>
```

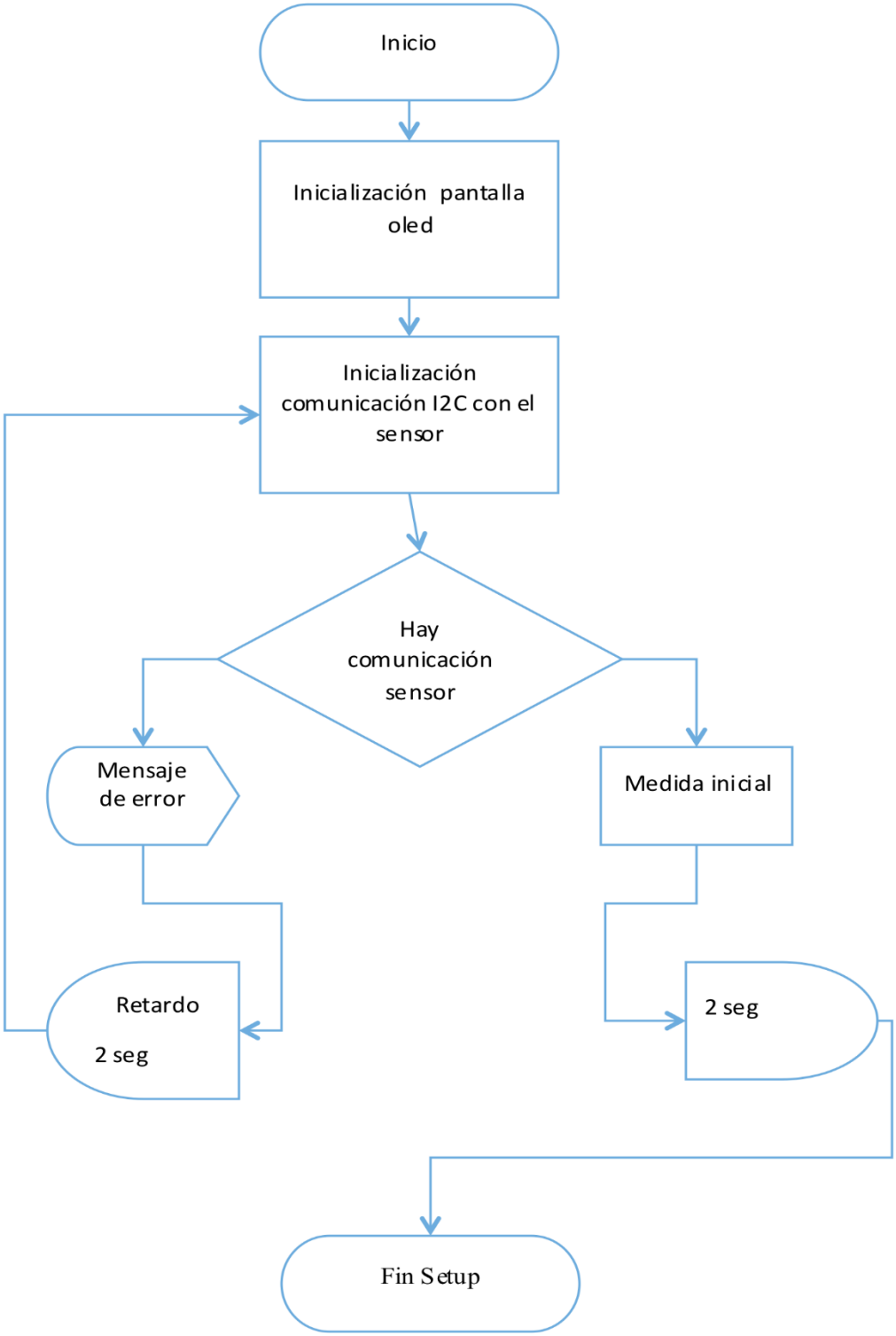
- TinyWireM.h y USI_TWI_Master.h, son librerías desarrolladas por Adafruit para compatibilizar la comunicación I²C de sus rajetas con Arduino.

- Adafruit_SSD1306.h, es una librería específica para el control de las pantallas OLDE de Adafruit.

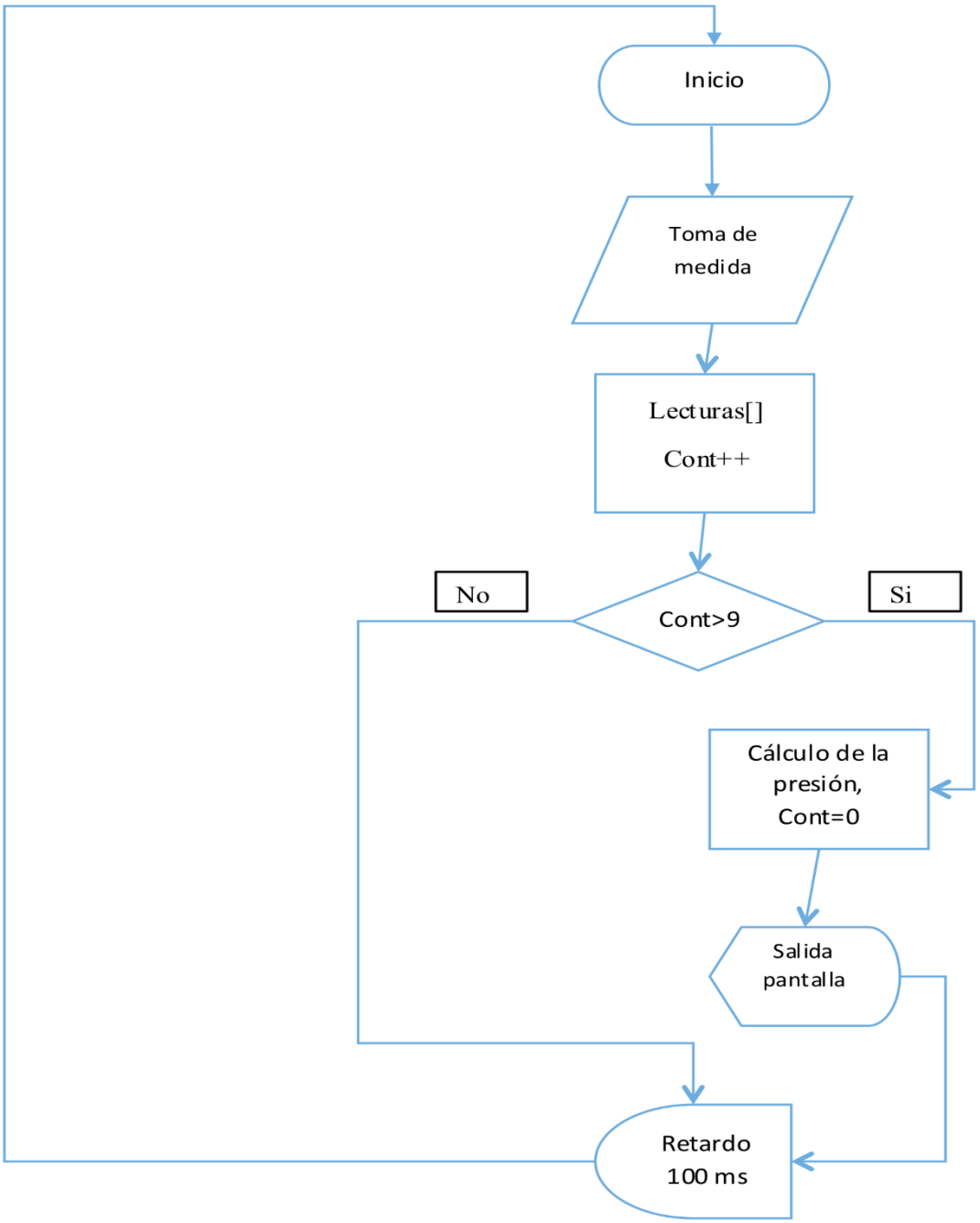
- Adafruit_GFX.h y gfxfont.h, son librerías para el control e implementación de gráficos en las pantallas de Adafruit..

- Stdint.h, es una librería estándar que amplia un tipo de variable especialmente útiles para programar µC, ya que poseen capacidad limitada, y muchas veces hay que hacer tratamiento de datos bit a bit. En este caso se utilizará para declarar la variable uint8_t y uint16_t, que son variables sin signo que ocupan 8 y 16 bits respectivamente.

5.3.3.2 void setup()



5.3.3.3 void loop()



5.3.3.4 Funciones realizadas.

En este apartado solo se explicará la función de obtención de medida del sensor vía I^2C, ya que el resto de funciones son iguales a las desarrolladas en los otros prototipos. Puede consultarse el código completo en el anexo 1.

Como se ha mencionado anteriormente, el sensor manda la información en ristras de 16bits. Para extraer la información se ha desarrollado la siguiente función:

```
uint8_t ReadPressure_mmHg(int id, float *value) {

  uint16_t dataword;
  uint8_t data[2];
  uint8_t ss_bit;

  Wire.requestFrom(id, 2);    // Le pido al sensor 2 Bytes de información.

  while (Wire.available()) {
    data[0] = Wire.read();
    data[1] = Wire.read();
  }
  ss_bit = data[0] >> 6;

  if (ss_bit == 0x00) {

    dataword = data[0] & 0x3F;

    dataword = (dataword << 8) | data[1];

    *value = (((dataword - 1638.40) * (5171.00 - 0.00)) / 13107.20) ;

    return 0;

  } else {
    return 1;
  }
}
```

Como puede verse la función es de tipo uint8_t, a la cual se le pasa un parámetro id, que coincide con la dirección del sensor y una dirección de memoria donde se almacenará el valor de la presión. La función devuelve un 0, si la medida se ha realizado correctamente o un 1 si no se ha llevado a término la medida.

Las variables utilizadas, pueden observase que tienen un tamaño concreto en bits. La variable *dataword,* se utiliza para almacenar la palabra completa de la medida, la variable data[2], es un vector que almacena los bytes de información que manda el sensor para su posterior tratamiento de bits y por último la variable ss_bit, donde se almacena si la medida es correcta o no .

El proceso de obtención de la información en mmHg,, es bit a bit por lo que requiere el enmascarar la información para obtener datos útiles al propósito. La función de la siguiente forma.

1) El maestro pide al esclavo 2 bytes, es decir, 16bits de información y deja el canal de comunicación abierto.

```
Wire.requestFrom(id, 2);
```

La función es específica de la librería Wire.h, en este caso el id del sensor se corresponde con la dirección 0x28.

2) Mientras esta el canal de información abierto, se llenan el vector data[], de esta forma separamos los bytes obtenidos del sensor. Ubicados en el data[0] y data[1]. Recodar que los primeros bits del primer Byte que manda el sensor se corresponde con el estado de la medida.

```
while (Wire.available()) {
  data[0] = Wire.read();
  data[1] = Wire.read();
}
```

El resultado es que tenemos en cada celda del vector data[], dos valores de 8 bits. Que se corresponde con el *Data Byte 1* y *Data Byte 2* mencionados en el apartado de descripción del sensor.

3) Sacamos la información de los bits s1 y s2 almacenados en data[0] para comprobar si la medida es correcta.

```
ss_bit = data[0] >> 6;
```

Los bits s1 y s2 se encuentran al principio del Data Byte 1, por lo que para obtener su valor, es necesario correr el data[0] seis posiciones a la derecha, de esta forma aislamos los bits en la variable ss_bit y se conocerá el estado de la medida.

4) Si la medida es correcta, se procede a sacar el valor de la presión en mmHg, el valor de salida del sensor se corresponde con los 6 últimos bits del *Data byte 1* y los 8 bits del *Data Byte 2*. Por ello hay que sacar y componer la variable de 16 bits dataword mediante el enmascaramiento de los bits obtenidos en data[].

```
dataword = data[0] & 0x3F;

dataword = (dataword << 8) | data[1];
```

En primer lugar se procede a obtener los primeros 6 bits de información contenidos en la variable data[0], mediante el enmascaramiento *AND.* Para

su mejor entendimiento se utilizará una representación gráfica con celdas que simbolizan los bits.

Dataword 16 bits inicio															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Vector data[]															
S1	S2	x	x	x	x	x	x	y	y	y	y	y	y	y	y
data[0]								data[1]							

Dadas las variables con su espacio asignado procedemos a sacar los primeros 8 bits de *dataword*. Para ello sacamos toda la información de data[0] enmascarándolo con el parámetro 0x3F (equivale a 00111111 en binario), la función lógica *AND* (&), se caracteriza por poner a 0 los bits por lo que tendríamos la información exacta correspondiente a los 6 últimos bit de data[0].

El resultado de la primera línea de código es la siguiente:

Dataword 16 bits															
0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x

La siguiente línea de código realiza la operación en dos etapas. La primera consiste en correr los bits de *dataword* 8 posiciones a la izquierda.

Dataword 16 bits															
0	0	x	x	x	x	x	x	0	0	0	0	0	0	0	0

La segundo incluye mediante la función lógica *OR* (|) el valor completo almacenado en data[1]. Obteniendo por fin, el valor deseado para su posterior implementación en la función de transferencia.

Dataword 16 bits completado															
0	0	x	x	x	x	x	x	y	y	y	y	y	y	y	y

5) Una vez obtenido el parámetro *dataword,* se calcula la presión mediante la función facilitada por el fabricante.

```
*value = (((dataword - 1638.40) * (5171.00 - 0.00)) / 13107.20) ;
```

6 Conclusiones

6.1 Sobre el objeto del proyecto.

El proyecto es un claro ejemplo de cómo utilizando plataformas de bajo coste y libres, se pueden resolver problemas, no solo de carácter doméstico o lúdicos, sino que también, se pueden utilizar en el ámbito técnico y científico. Poniendo a disposición del público general, unas herramientas asequibles que permitan desarrollas funciones específicas, sin la necesidad de una cuantiosa financiación.

En cuanto al objeto del proyecto, se han desarrollado tres prototipos funcionales. Capaces de mostrar la medida en milímetros de mercurio, almacenar datos y de tener la respuesta de forma no invasiva, por lo que se deduce que se han cumplido los ítems fijados.

El trabajo también ha servido para afianzar conocimientos técnicos adquiridos en la carrera, en materias como la electrónica, instrumentación e informática industrial. Desarrollando programación de microcontroladores y adquiriendo conocimiento sobre protocolos o estándares de comunicación, cada vez más presentes en la industria.

A demás, brinda la oportunidad de desarrollar un producto para un fin específico, en el ámbito de la medicina, donde se hace palpable la importancia que tiene conocer el entorno en al cual el dispositivo va a desarrolla su función, para su diseño y facilitar su utilización.

6.2 Futuras mejoras y otras aplicaciones.

El dispositivo se puede utilizar para otras tareas de ámbito médico, ya que una mejor monitorización garantiza un mejor servicio al paciente y disminuye el tiempo de postoperatorio.

El primer y segundo prototipo, se pueden utilizar para entrenar la maniobra de Selick. Esta consiste en presionar con una fuerza no superior a 4 Newtons el cartílago cricoides, parte externa del cuello, con el fin de impedir reflujo gástrico a los pulmones durante las maniobras de intubación traquear. Únicamente habría que cambiar la programación y adaptarlo a la nueva magnitud de medida. El proceso es más fácil, ya que el fabricante da la respuesta del sensor ante la fuerza aplicada en Newton y kilogramos.

El tercer prototipo, se puede utilizar en operaciones oftalmológicas, donde a los pacientes para que el ojo permanezca lo más inmóvil posible, se les aplica presión que no debe exceder los 30 milímetros de mercurio. Habría que añadir elementos visuales o sonoros para que el médico advirtiera que está haciendo la presión correcta. Por otra parte, se puede adaptar como dispositivo de localización de la zona epidural, facilitando así la labor del anestesista.

7 Bibliografía

Citas

[1] "Definición isquemia", *Google,* 2017 [En línea]. Disponible en: https://www.google.es/search?q=isquemia&oq=isquemia&aqs=chrome..69i57j0l5.2171j1j7&sourceid=chrome&ie=UTF-8 [Accedido: 6-May-2017].

Consultas

[1] Artero Óscar Torrente, *Arduino: curso práctico de formación.* Madrid: RC libros, 2013.

[2] Cortés Ferrando Reyes y J.C. Monjaraz, *Arduino: aplicaciones en robótica, mecatrónica e ingenierías*. México: Alfaomega,2015.

[3] "*tourniquets.org*", J.A. McEwen, 2017 [En línea]. Disponible en: http://tourniquets.org/ [Accedido: 4-Ene-2017].

[4] "*Arduino-Reference*", [En línea]. Disponible en: https://www.arduino.cc/en/Reference/HomePage [Accedido: 4 -May-2016].

[5] "*Fundamentos de Medidas de Presión- National Intruments*", [En línea]. Disponible en: http://www.ni.com/white-paper/13034/es/ [Accedido: 4-Ene-2017].

[6] "*Interlink Electronics FSR Force Sensing Resistors*", [En línea]. Disponible en : http://www.interlinkelectronics.com/FSR402.php [Accedido: 23-Mar-2016].

[7] "*FlexiForce User Manual| Tekscan*.", [En línea]. Disponible en: https://www.tekscan.com/resources/articles-research/flexiforce-user-manual [Accedido: 20- Jun- 2016].

[8] "*I2C Commmunication with Digital Otuput Pressure*", [En linea]. Disponible en: https://sensing.honeywell.com/index.php?ci_id=45841 [5-Feb-2017].

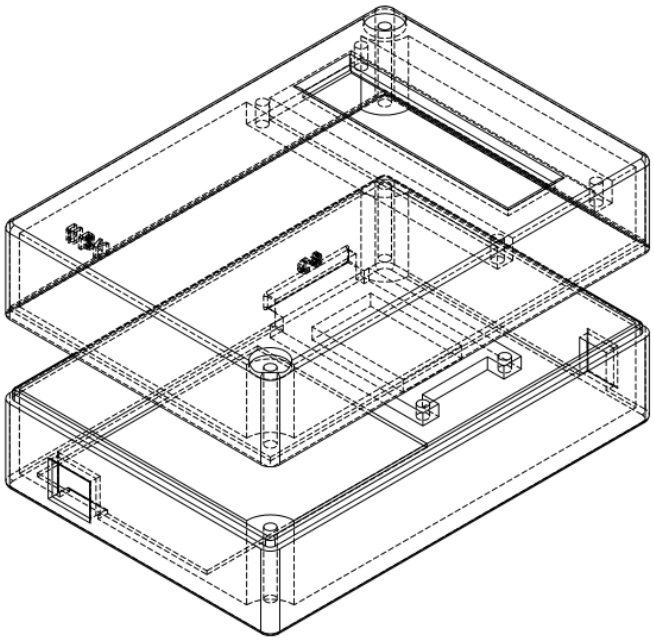
[9] Bibliografía de la Universidad Publica de Navarra. Oficina de Referencias. "Guía para citar y referenciar. IEEE Style", 2016 [En linea]. Disponible en: http://www2.unavarra.es/gesadj/servicioBiblioteca/tutoriales/Citar_referenciar_(IEEE).pdf. [Accedido: 5- Jul-2017].

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Prototipo instrumental de medición de presión en torniquetes

2. Planos



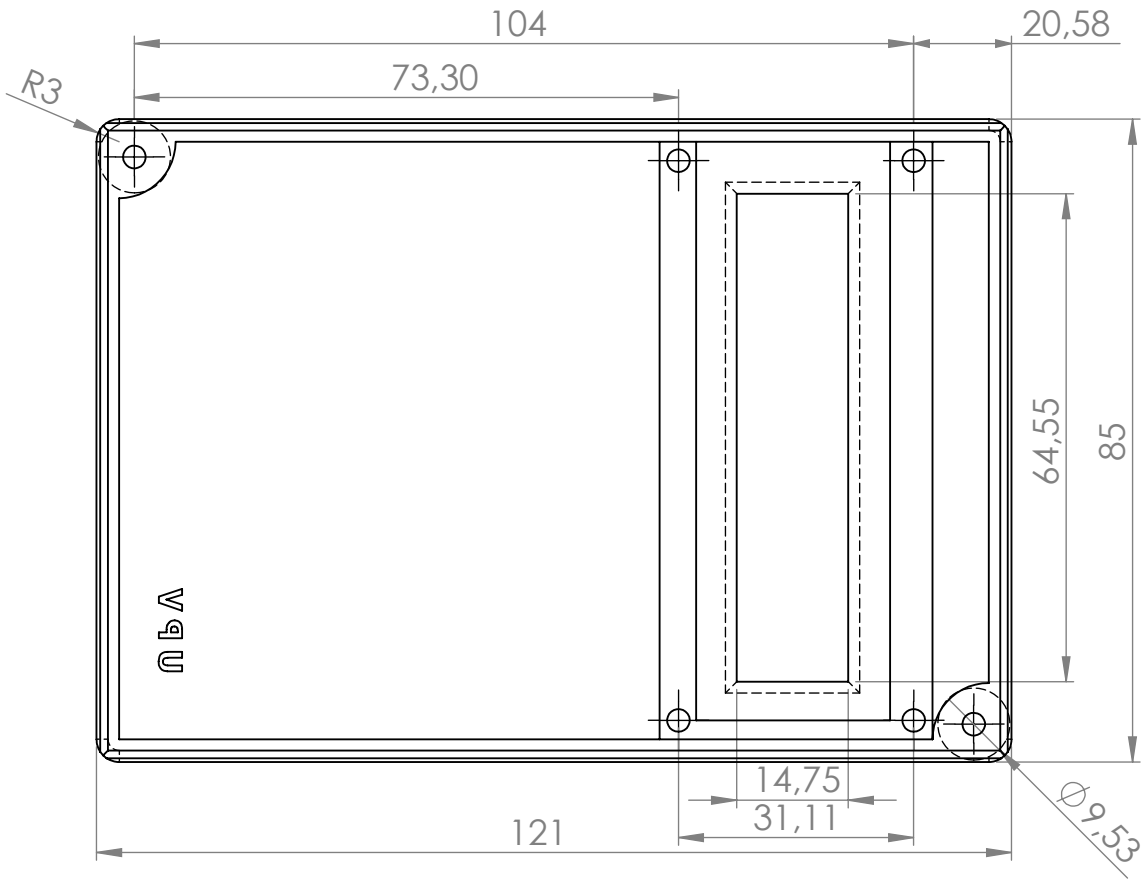
Autor:

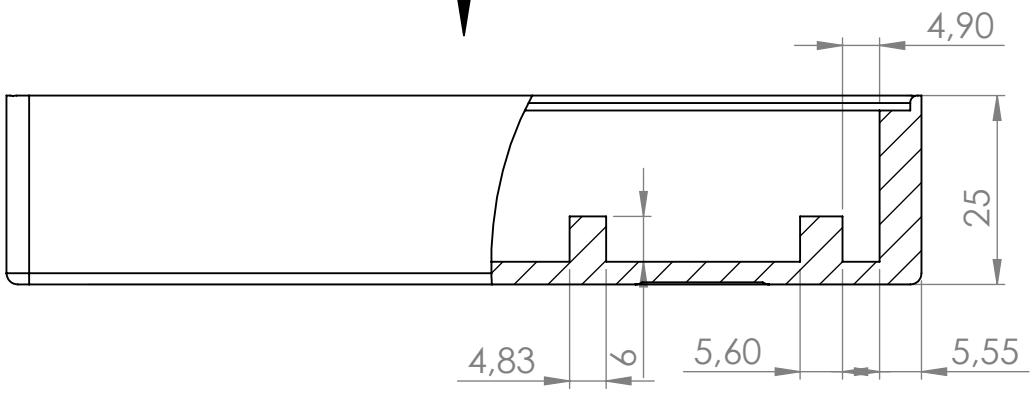
D. José Ignacio Marqués Ortega

Tutor:

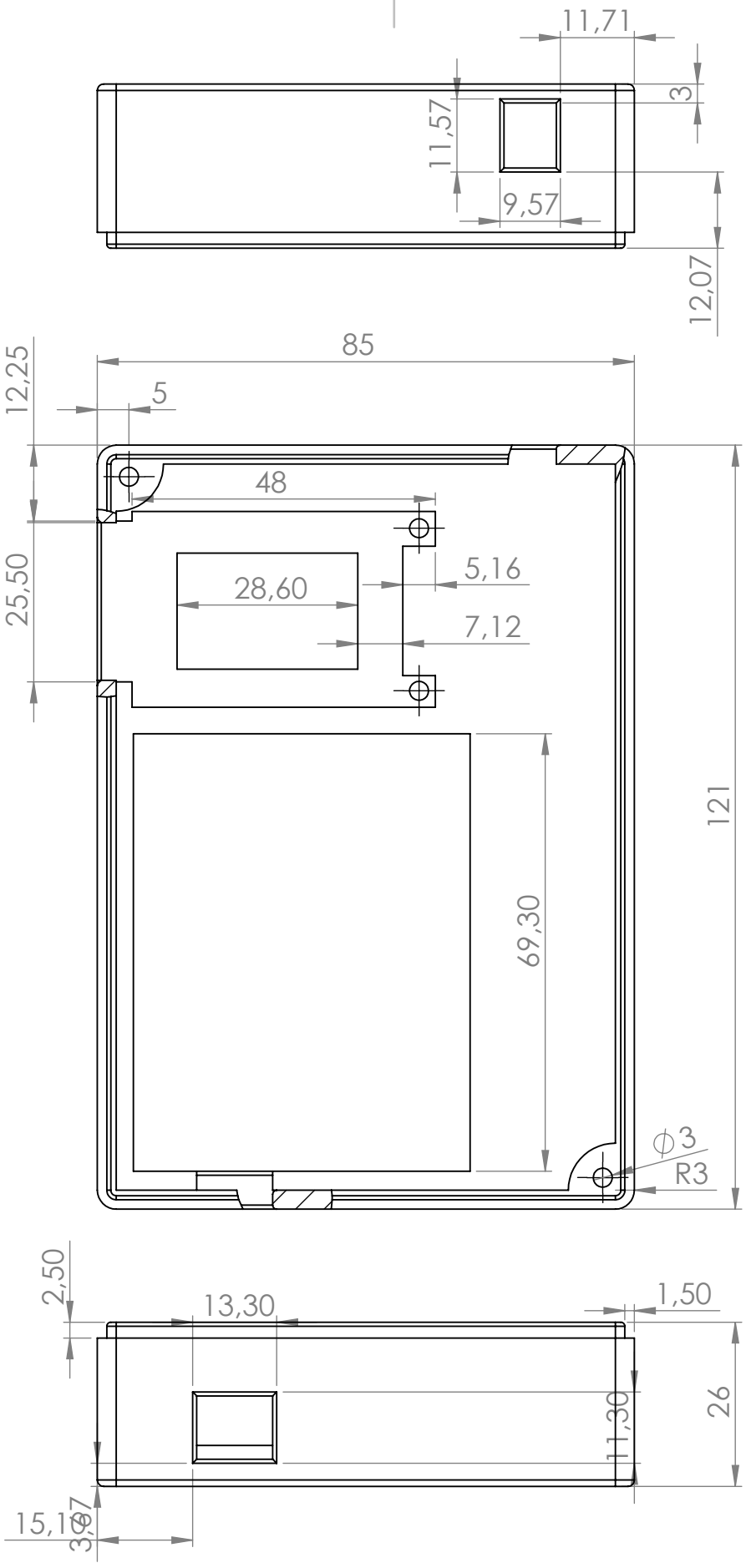
D. Ángel Perles Ivars

Índice de planos

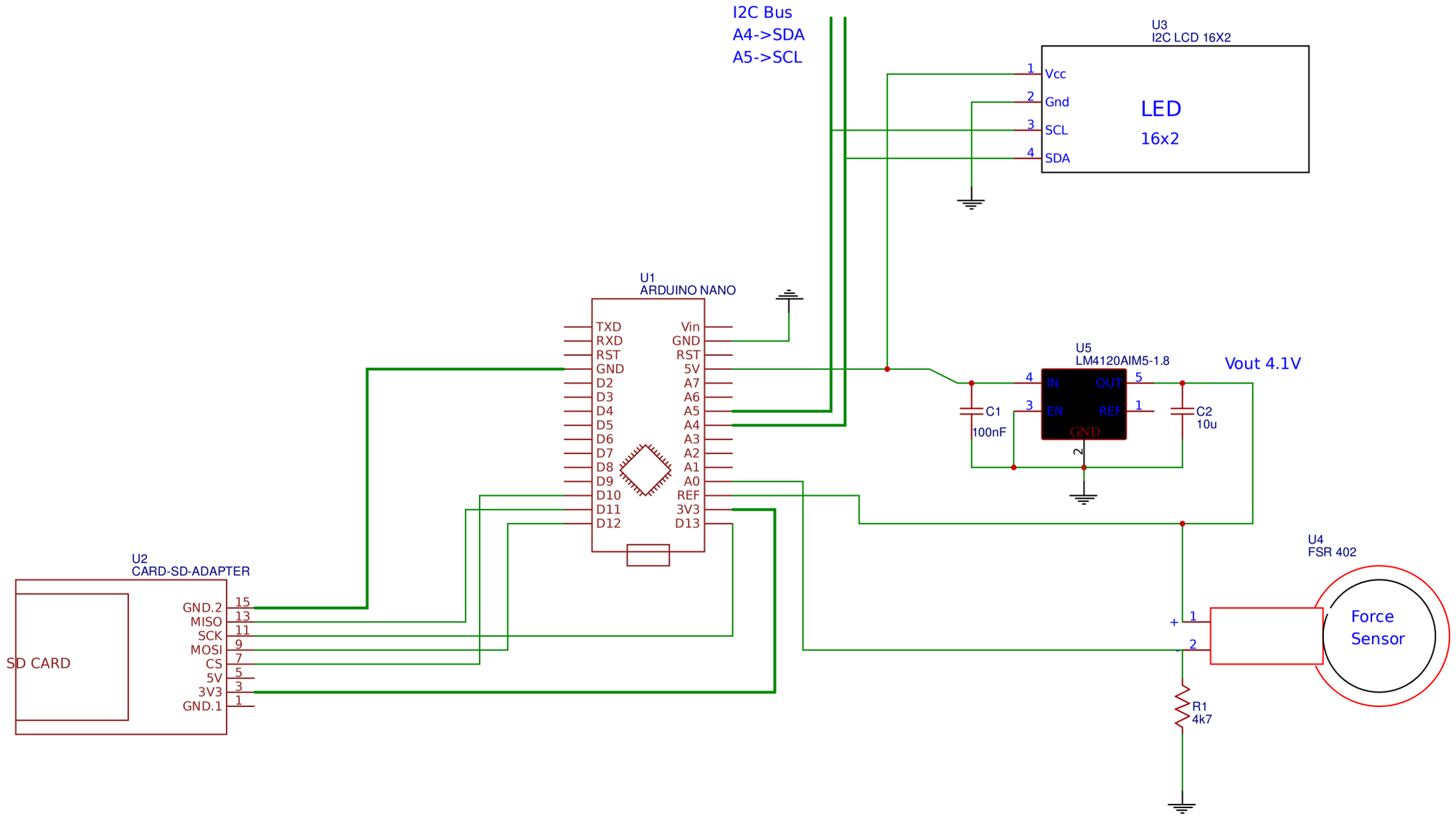




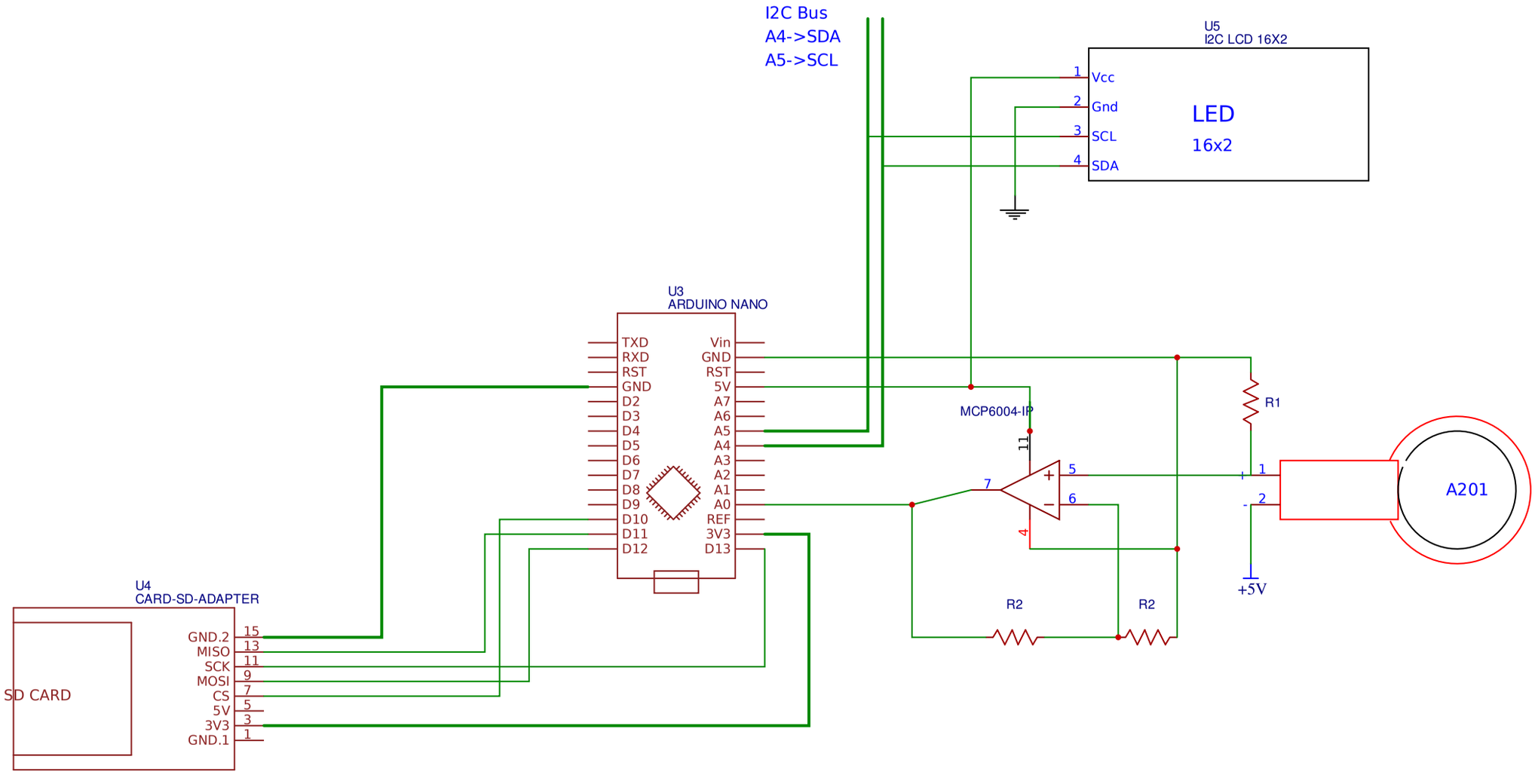
Nombre: CARCASA MACHO	Escala modelo: 1:1	Plano nº: 1	Escuela Técnica Superior de Ingeniería del Diseño
Alumno: Jose Ignacio Marqués Ortega			
Fabricante: Repro. 3D ETSID	Material:	Peso:	Fecha: Julio de 2017



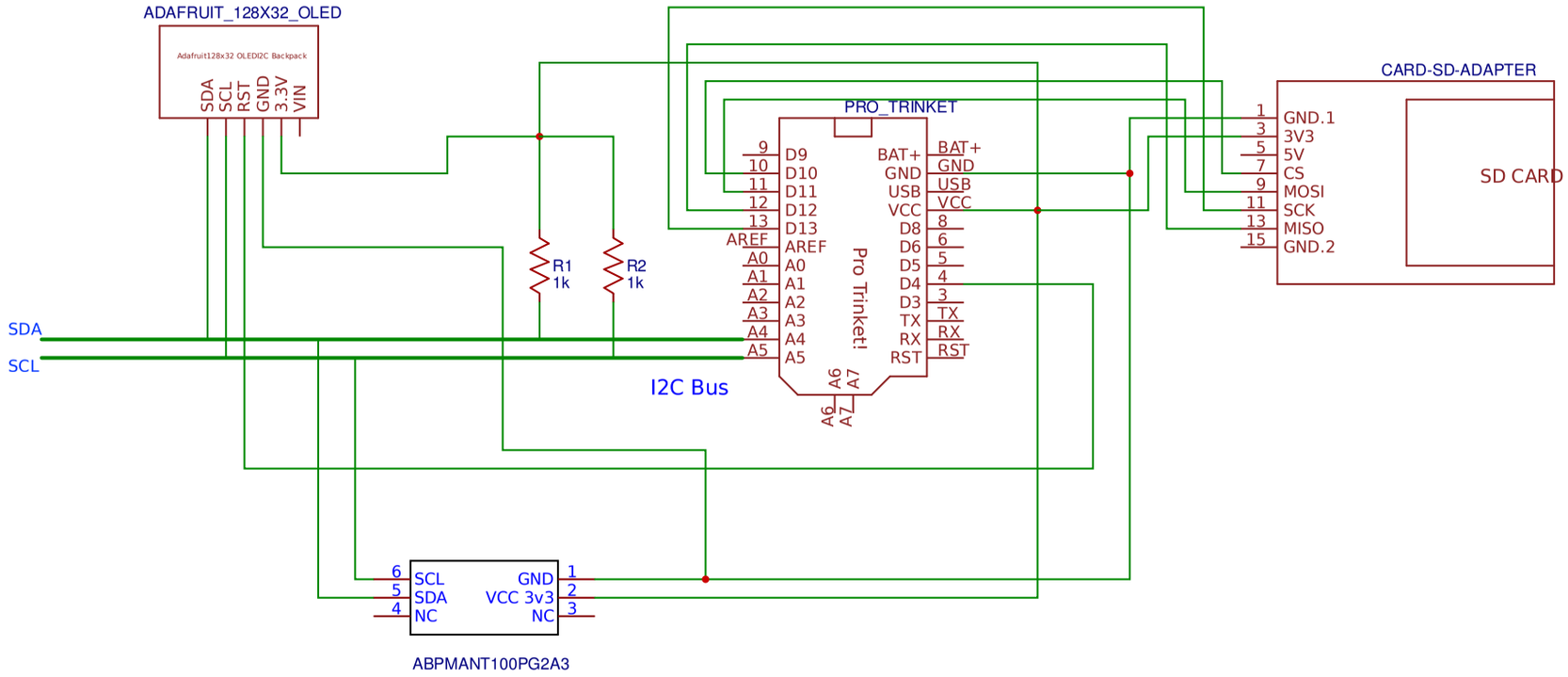
Nombre: CARCASA MACHO	Escala modelo: 1:1	Plano nº: 2	Escuela Técnica Superior de Ingeniería del Diseño
Alumno: Jose Ignacio Marqués Ortega			
Fabricante: Repro. 3D ETSID	Material:	Peso:	Fecha: Julio de 2017



TITLE: Prototipo 1.0	REV: 1.0
Date: 2017-06-05	Sheet: 1/1
EASYEDA V4.5.6	Drawn By: José Ignacio Marqués



TITLE: Prototipo 2.0		REV: 1.0
Date: 2016-12-18	Sheet: 1/1	
EasyEDA V3.12.1	Drawn By: Jose Ignacio Marqués	



TITLE: Prototipo 3		REV: 1.0
Date: 12/05/2017	Sheet: 1/1	
EasyEDA V4.4.6	Drawn By: Jose Ignacio Marqués	

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Prototipo instrumental de medición de presión en torniquetes

3. Pliego de Condiciones

Autor:

D. José Ignacio Marqués Ortega

Tutor:

D. Ángel Perles Ivars

PLIEGO DE CONDICIONES

1. Definición y alcances del pliego

El presente proyecto supone el desarrollo de un prototipo de medición de presión para torniquetes no neumáticos, utilizados en operaciones con isquemia. El prototipo se basa en el uC Atmega 328P integrado en una placa Arduino.

El microcontrolador se programa mediante el software de Arduino, basándose en el uso de las bibliotecas desarrollada para este entorno. Por ello, si se altera alguno de los componentes hay que encontrar librerías específicas para el control de dichos componentes.

El objeto de este documento es determinar las circunstancias en las cuales el prototipo va a desarrollar su función.

2. Condiciones y normas de carácter general

A continuación, se exponen una serie de condiciones o normas de cumplimiento de carácter general.

a. Utilización

- El sistema debe utilizarse para el fin que ha sido desarrollado y operar con el dispositivo dentro de los márgenes de seguridad. Cualquier uso diferente debe considerase inadecuado por tanto no garantiza el funcionamiento correcto del dispositivo.
- Hay que asegurar que antes de utilizar el dispositivo la batería está cargada, para el buen funcionamiento del mismo.
- Hay que comprobar la calibración antes de utilízalo en una toma de medidas.
- Comprobar si el sistema es seguro antes de comenzar a utilizarlo

b. Mantenimiento

- Antes de realizar cualquier labor de mantenimiento hay que asegurarse que el dispositivo no está conectado, ya que algunos elementos pueden sufrir daños.
- Una vez realizadas las maniobras de mantenimiento, comprobar que el circuito está bien conectado siguiendo os esquemáticos disponibles en [Planos]

3. Condiciones particulares.

a. Condiciones del encargo

- El dispositivo debe devolver la medida de presión en mmHg
- A de ser capaz de almacenar datos
- A de ser portátil para facilitar la su utilización.
- Debe de poder ser esterilizado en caso de que fuese necesario.

b. Condiciones legales

Licencia:

- El trabajo se publica bajo licencia GNU GPLv3 y CC que permite cualquier modificación siempre que se publique bajo las mismas condiciones y se reconozca el trabajo del autor del mismo.
- Es una licencia de *software* abierto, la cual está disponible para cualquier persona que quiera desarrolla proyectos similares.
- El autor no se hace responsable del uso fuera del carácter académico y de investigación del proyecto

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Prototipo instrumental de medición de presión en torniquetes

4. Presupuesto

Autor:

D. José Ignacio Marqués Ortega

Tutor:

D. Ángel Perles Ivars

Presupuesto:

1. Sobre el presupuesto

Para elaborar el presupuesto, los precios de los materiales se han tomado de los catálogos de distribuidores oficiales y los fabricantes. Se ha elaborado un presupuesto individual por cada prototipo desarrollado.

El coste de mano de obra se ha calculado a partir de la estimación del precio de 50€/h por el trabajo de un graduado en ingeniería industrial. Se ha estimado 35 h invertidas en el diseño y fabricación por cada prototipo.

2. Prototipo 1

2.1 Materiales

Prototipo 1				
Concepto	Unidades	Cantidad	Precio unitario	total
Materiales				
Placa Arduino Nano 3.0	ud	1	18,36 €	18,36 €
LCD 16x2	ud	1	12,52 €	12,52 €
Módulo I2C LCD	ud	1	2,50 €	2,50 €
Módulo SD	ud	1	1,44 €	1,44 €
Sensor FSR 402	ud	1	7,58 €	7,58 €
Regulador de tensión 4,1 V	ud	1	0,87 €	0,87 €
Resistencia 0,25 w	ud	1	0,28 €	0,28 €
Condensador 100nF	ud	1	0,06 €	0,06 €
Condensador 10 uF	ud	1	0,06 €	0,06 €
Cable de audio paralelo 2x 0,09mm2	M	1	0,58 €	0,58 €
Conexión jack macho	ud	1	1,13 €	1,13 €
Conexión jack hembra	ud	1	0,58 €	0,58 €
Cubre cables termo retráctil	ud	2	0,10 €	0,20 €
PCB Easyeda	ud	1	4,15 €	4,15 €
Carcasa impresión 3D	H	12	6,95 €	83,40 €
Subtotal coste de materiales				133,70 €

Tabla1: coste materiales

2.2 Mano de obra

Mano de obra				
Diseño y programación	h	35	50,00 €	1.750,00 €
Graduado en ingeniería electrónica industrial y automática				
Subtotal coste mano de obra				1.750,00 €

Tabla2: mano de obra

2.3 Coste del proyecto

Coste del proyecto		
Materiales		133,70 €
Mano de obra		1.750,00 €
Total		1.883,70 €
IVA	21%	395,58 €
Gastos adicionales	10%	188,37 €

Tabla3: Coste del proyecto1

PVP	2.467,65 €

El precio total del prototipo 1 asciende a dos mil cuatrocientos sesenta y siete euros con sesenta y cinco céntimos.

3. Prototipo 2

3.1 Materiales

Prototipo 2				
Concepto	Unidades	Cantidad	Precio unitario	total
Materiales				
Placa Arduino Nano 3.0	ud	1	18,36 €	18,36 €
LCD 16x2	ud	1	12,52 €	12,52 €
Módulo I2C LCD	ud	1	2,50 €	2,50 €
Módulo SD	ud	1	1,44 €	1,44 €
Sensor A201	ud	1	7,58 €	7,58 €
Amplificador operacional MCP-6004	ud	1	0,43 €	0,43 €
Resistencia 0,25 w	ud	3	0,28 €	0,84 €
Cable de audio paralelo 2x 0,09mm2	M	1	0,58 €	0,58 €
Conexión jack macho	ud	1	1,13 €	1,13 €
Conexión jack hembra	ud	1	0,58 €	0,58 €
Cubre cables termo retráctil	ud	2	0,10 €	0,20 €
PCB Easyeda	ud	1	4,15 €	4,15 €
Carcasa impresión 3D	H	10	6,95 €	69,50 €
Subtotal coste de materiales				119,80 €

Tabla 3: Coste de materiales prototipo 2

3.2 Mano de obra

Mano de obra				
Diseño y programación	h	35	50,00 €	1.750,00 €
Graduado en ingeniería electrónica industrial y automática				
Subtotal coste mano de obra				1.750,00 €

Tabla 4: Coste mano de obra prototipo 2

3.3 Coste del proyecto

Coste del proyecto		
Materiales		119,80 €
Mano de obra		1.750,00 €
Total		1.869,80 €
IVA	21%	392,66 €
Gastos adicionales	10%	186,98 €

Tabla 5: Coste del proyecto prototipo 2

PVP	2.449,44 €

El precio total del prototipo 2 asciende a dos mil cuatrocientos cuarenta y nueve euros con cuarenta y cuatro céntimos.

4.Prototipo 3

4.1 Materiales

Prototipo 3				
Concepto	Unidades	Cantidad	Precio unitario	total
Materiales				
Pro Trinket 3v	ud	1	12,36 €	12,36 €
Pantalla Oled	ud	1	19,54 €	19,54 €
Sensor Presión aire I2C de HoneyWell	ud	1	21,50 €	21,50 €
Módulo SD	ud	1	1,44 €	1,44 €
Resistencia 0,25 w	ud	2	0,28 €	0,56 €
Módulo carga batería	ud	1	17,55 €	17,55 €
Batería Lipo 3,7V 2A	ud	1	15,61 €	15,61 €
Interruptor	ud	1	1,20 €	1,20 €
PCB Easyeda	ud	1	4,15 €	4,15 €
Carcasa impresión 3D	h	9,2	6,95 €	63,94 €
Subtotal coste de materiales				157,85 €

Tabla 6: Coste de materiales prototipo 3

4.2 Mano de obra

Mano de obra				
Diseño y programación	h	35	50,00 €	1.750,00 €
Graduado en ingeniería electrónica industrial y automática				
Subtotal coste mano de obra				1.750,00 €

Tabla 7: Coste mano de obra prototipo 3

4.3 Coste del proyecto

Coste del proyecto		
Materiales		157,85 €
Mano de obra		1.750,00 €
Total		1.907,85 €
IVA	21%	400,65 €
Gastos adicionales	10%	190,79 €

Tabla 8: coste final proyecto prototipo 3

PVP	2.499,28 €

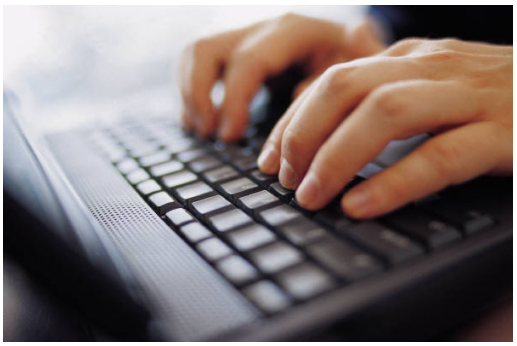
El precio total del prototipo 3 asciende a dos mil cuatrocientos noventa y nueve euros con veintiocho céntimos.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Prototipo instrumental de medición de presión en torniquetes

5.Anexos



Autor:

D. José Ignacio Marqués Ortega

Tutor:

D. Ángel Perles Ivars

Anexo

A.Código prototipo 1

```
/**
   @File "HemaClearPressure_1.ino"
   @Brief this program is to control a pressure measure machine prototype:

    You need Arduino nano board, SD module SPI comunication , LCD 16x2 whit I2C comunication ,
    an FSR402 Interlink sensor whit 4k7 resitor.

   @Author Jose Ignacio Marques
   @Date 23/06/2016
*/

/**** Libraries*****/

#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#include <avr/io.h>
#include <avr/interrupt.h>

#include <math.h>


/** Variables globales y funciones**/

#define AD 0.00400391 //ADC 4.10/1023.00 (10 bit AD)
#define Ra  4630
#define Vcc 4.10 //Power for FSR402 and ARef

LiquidCrystal_I2C lcd(0x27, 16, 2); //lcd object
File dataFile;//file object

const int CS = 10; //comunicacion wiht sd, CS pin
volatile int seconds = 0; //for Timer1 interrupt (1s)
volatile byte mins = 0;
float analogin, cond, vin, rsensor, aux1 = 0, aux2 = 0, pressure;

int dato;

void TIM1_config_1seg_Prescaler (void);
void datalogg_1min (float x);
float read_Pressure (float y);

void setup() {

  /** Beguin serial comunication with de computer for a test */
```

```
48.
49.  Serial.begin(9600);
50.  Serial.println("Initializing SD card...");
51.
52.
53.  /** Initilaitation SPI comunication wiht SD module */
54.
55.  if (!SD.begin(CS)) {
56.
57.    Serial.println("Card failed, or not present");
58.
59.    return;
60.  }
61.
62.  Serial.println("card initialized...");
63.
64.  /** LCD initialitation */
65.
66.  lcd.begin();
67.  lcd.backlight();
68.  lcd.print("Pressure [mmHg]");
69.
70.  /**Open or create a file in the SD */
71.
72.  dataFile = SD.open("datalogg.csv", FILE_WRITE);
73.
74.  if (dataFile) {
75.
76.    dataFile = SD.open("datalogg.csv", FILE_WRITE);
77.    dataFile.print("Lectura"); dataFile.print(",");
78.    dataFile.print("Voltaje"); dataFile.print(",");
79.    dataFile.print("Minutos"); dataFile.print(",");
80.    dataFile.print("Presion"); dataFile.println(",");
81.
82.    dataFile.close();
83.  }
84.
85.  TIM1_config_1seg_Prescaler ();
86.
87.  analogReference(EXTERNAL);
88.
89. }
90.
91. void loop() {
92.
93.   analogin = analogRead(0);
94.
95.   if (analogin == 0) {
96.     lcd.setCursor(0, 1);
97.     lcd.print("0       ");
98.   }
99.   else {
```

```
100.              lcd.setCursor(0, 1);
101.              lcd.print(read_Pressure(analogin));
102.             }
103.             if (mins == 1) {
104.              datalogg_time (analogin);
105.             }
106.
107.             delay(100);
108.            }
109.
110.            void TIM1_config_1seg_Prescaler() {
111.
112.             /* Internal interrupt timer1 16 Bits for comparing, see register on AVR
    datasheet */
113.
114.             cli();              //clear all interuptions.
115.
116.             TCCR1A = 0;
117.             TCCR1B = 0;
118.
119.             OCR1A = 15624;        // value to compare the 1024 prescaling to 1s
120.             TCCR1B |= (1 << WGM12);
121.
122.             TCCR1B |= (1 << CS10);
123.             TCCR1B |= (1 << CS12);
124.
125.             TIMSK1 |= (1 << OCIE1A);
126.
127.             sei();//set the interruptions.
128.            }
129.
130.            void datalogg_time (float x) {
131.
132.             vin = AD * x;
133.             aux1 = (Vcc - vin);
134.             aux2 = Ra / vin;
135.             rsensor = aux1 * aux2;
136.             cond = (1 / rsensor) * 1000000;
137.             pressure = 0.5131 * cond;
138.
139.             dato++;
140.
141.             dataFile = SD.open("datalogg.csv", FILE_WRITE);
142.
143.             dataFile.print(x);
144.             dataFile.print(",");
145.             dataFile.print(vin);
146.             dataFile.print(",");
147.             dataFile.print(dato);
148.             dataFile.print(",");
149.             dataFile.print(pressure);
150.             dataFile.println(",");
```

```
151.
152.          dataFile.close();
153.
154.          lcd.setCursor(13, 1);
155.          lcd.print(dato);
156.          mins = 0;
157.         }
158.
159.         float read_Pressure (float y) {
160.
161.          vin = AD * y;
162.          aux1 = (Vcc - vin);
163.          aux2 = Ra / vin;
164.          rsensor = aux1 * aux2;
165.          cond = (1 / rsensor) * 1000000;
166.          pressure = 0.5131 * cond;
167.
168.          return (pressure);
169.
170.         }
171.
172.         ISR (TIMER1_COMPA_vect) {
173.          seconds++;
174.          if (seconds % 2 == 0) {
175.            mins++;
176.          }
177.         }
178.
```

B.Código prototipo 2

```
/**
@File: "proto2_ok.ino"
@Review: 4.0
@Brief: this program is to control a pressure measure prototipe:

You need Arduino nano board, SD module SPI comunication , LCD 16x2 whit I2C
comunication ,
an A201 Flexiforce sensor.

@Author Jose Ignacio Marqués
@Date 11/04/2017

*/

#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
#include <avr/io.h>
#include <avr/interrupt.h>

#include <math.h>

/* Global variables & Functions */

#define AD  0.004916 //ADC 5.03/1023.00 (10 bit AD)

LiquidCrystal_I2C lcd(0x27, 16, 2); //lcd object
File dataFile; //file object

const int CS = 10; //comunicacion wiht sd
volatile int seconds = 0; //for Timer1 interrupt (1s)
volatile byte mins = 0;

/* Data variable **/

float analogin = 0, vin, pressure;
int datatime = 0;

/* Filter variable */

int lecturas[10]; // Almacena 10 medidas analógicas para realizar un promedio.[0,
1023] 10Bits
int Total = 0; // recoge el sumatorio de las medidas
int contador = 0; // cuenta las medias
int i;

void setup() {

/* Initilitation serial cmunication */

/* Initilaitation SPI comunication wiht SD module */

SD.begin(CS);

/** LCD initialitation **/

lcd.begin();
lcd.backlight();
lcd.print("Pressure [mmHg]");

/*New datafile */

dataFile = SD.open("datalogg.csv", FILE_WRITE);

if (dataFile) {

dataFile = SD.open("datalogg.csv", FILE_WRITE);
dataFile.print("Lectura"); dataFile.print(",");
```

```
dataFile.print("Segundos"); dataFile.print(",");
dataFile.print("Voltaje"); dataFile.print(",");
dataFile.print("Presion"); dataFile.println(",");

dataFile.close();
}

TIM1_config_1seg_Prescaler ();

for (i = 0; i < 10; i++) {
lecturas[i] = 0;
}

}

void loop() {

Total = Total - lecturas[contador];

lecturas[contador] = analogRead(0);
Total = Total + lecturas[contador];
contador++;


if (contador >= 10) {
contador = 0;
analogin = Total / 10.00;

if (analogin >= 6) {
pressure = read_Pressure(analogin);
lcd.setCursor(0, 1);
lcd.print(pressure);
lcd.print("  ");
Serial.println(pressure);
}
else {
lcd.setCursor(0, 1);
lcd.print(0);
lcd.print("    ");
}

if (mins == 1) {
datalogg_1min (analogin);
}
}
delay(100);
}

void TIM1_config_1seg_Prescaler() {

/* Internal interrupt timer1 16 Bits for comparing, see register on AVR datasheet */
```

```
cli();              //clear all interuptions.

TCCR1A = 0;
TCCR1B = 0;

OCR1A = 15624;        // value to compare the 1024 prescaling to 1s
TCCR1B |= (1 << WGM12);

TCCR1B |= (1 << CS10);
TCCR1B |= (1 << CS12);

TIMSK1 |= (1 << OCIE1A);

sei();              //set the new configuration of interruptions.
}

float read_Pressure (float y) {

vin = AD * y;

pressure = (vin - 3.275) / 0.0003;

return (pressure);
}

void datalogg_1min (float x) {

vin = AD * x;

pressure = (vin - 3.275) / 0.0003;

datatime = datatime + mins;

dataFile = SD.open("datalogg.csv", FILE_WRITE);

dataFile.print(x);
dataFile.print(",");
dataFile.print(datatime);
dataFile.print(",");
dataFile.print(vin);
dataFile.print(",");
dataFile.print(pressure);
dataFile.println(",");

dataFile.close();

lcd.setCursor(13, 1);
lcd.print(datatime);
mins = 0;

}
```

```
ISR (TIMER1_COMPA_vect) {

seconds++;

if (seconds % 2 == 0) {

mins++;

}
}
```

C.Código prototipo 3

```
#include <TinyWireM.h>
#include <USI_TWI_Master.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#include <gfxfont.h>
#include <stdint.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

int idSensor = 0x28, idOled = 0x3C;
float pressure, InitPressure, pressuremid;

int datalogg[10]; // Almacena 10 medidas analógicas para realizar un promedio.[0, 1023] 10Bits
int total = 0; // recoge el sumatorio de las medidas
int counter = 0; // cuenta las medias
int i;

uint8_t ReadPressure_mmHg(int id, float *value);
void print_meassure();

void setup() {

/** Configuracion de la pantalla **/

display.begin(SSD1306_SWITCHCAPVCC, idOled);

display.display();
delay(2000);

display.clearDisplay();

/** iniciamos el vector de lecturas a 0 **/
```

```
for (i = 0; i < 10; i++) {
datalogg[i] = 0;
}

/** Escribimos en la pantalla la presion inicial **/

if (ReadPressure_mmHg(idSensor, &InitPressure) == 1) {
display.println("Error 404");
display.display();
delay(2000);
} else {
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 0);
display.println("Pressure  [mmHg]");
display.display();
display.print("Pi = ");
display.print(InitPressure);
display.display();
delay(2000);
}
}

void loop() {

total = total - datalogg[counter];
ReadPressure_mmHg(idSensor, &pressure);
datalogg[counter] = pressure;
total = total + datalogg[counter];
counter++;

if (counter >= 10) {
counter = 0;
pressuremid = total / 10.00;
print_meassure();
}

delay(100);

}

/** Funciones lectura sensor de aire**/

uint8_t ReadPressure_mmHg(int id, float *value) {

uint16_t dataword;
uint8_t data[2];
uint8_t ss_bit;

Wire.requestFrom(id, 2);    // Le pido al sensor 2 Bytes de informacion.
```

```
while (Wire.available()) {
data[0] = Wire.read();
data[1] = Wire.read();
}
ss_bit = data[0] >> 6;

if (ss_bit == 0x00) {

dataword = data[0] & 0x3F;

dataword = (dataword << 8) | data[1];

*value = (((dataword - 1638) * (5171.0 - 0.0)) / (14745.0 - 1638.0)) + 0.00;

return 0;

} else {
return 1;
}
}

/*** Funciones de la pantalla oled SSD1306 ***/

void print_meassure() {

display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 0);
display.clearDisplay();
display.println("Pressure  [mmHg]");
display.display();

display.print("Pi = ");
display.print(InitPressure);
display.display();

display.setTextSize(2);
display.setTextColor(WHITE);
display.setCursor(0, 17);
pressuremid = pressuremid - InitPressure;
display.print(pressuremid);
display.display();

}
```