UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Trabajo final de grado

Grado en ingeniería electrónica industrial y automática

# Sistema de monitorización del prototipo Endoworm 3.0

Departamento de ingeniería electrónica

Curso 2015/2016

Autora:
Beatriz Almendros Luis

Tutor:
Carlos Sánchez Díaz

Valencia, septiembre 2016

*Al equipo de Endoworm, especialmente a Carlos Sánchez por darme la oportunidad de pertenecer a este grupo de investigación del que tanto he aprendido.*

# Resumen

En la actualidad la exploración del intestino delgado ha sido una ardua tarea para el gastroenterólogo. Este problema es debido a su localización (en el punto medio de cualquier entrada natural), a su peculiar anatomía (una gran cantidad de curvas) y a su gran longitud (de unos 7 metros) que lo convierten en una estructura anatómica de difícil alcance mediante un endoscopio convencional.

Como solución a esta situación, nace en 2005 el grupo de investigación formado por gastroenterólogos del Hospital Universitari i Politécnic La Fe de Valencia, Físicos e Ingenieros en distintas especialidades de la Universidad Politécnica de Valencia.
Fruto de esta colaboración nace la idea del prototipo Endoworm, se basa en un dispositivo adaptable a cualquier endoscopio, está basado en cavidades hinchables que permiten el replegamiento del intestino sobre el mismo y por tanto una mejor y más profunda visualización.

Las etapas anteriores del proyecto se han centrado en la mejora del prototipo, donde la versión actual3.0ha implicado un avance en el software y hardware para un óptimo control del mismo. Se ha comprobado su correcto funcionamiento en cuanto a las propiedades mecánicas del prototipo, pero no se han podido detectar posibles fugas de aire en los conductos entre la bomba y el sistema. En la actualidad las presiones de cada cavidad están siendo medidas por sensores en la cabecera del hardware, debido a la distancia existente entre las cavidades y los sensores (de unos 2 metros) se presenta la duda de si los resultados obtenidos son reales o si por el contrario están distorsionados por la distancia.

De esta duda nacen los objetivos del presente trabajo fin de grado, que se centra en el diseño y desarrollo de un sistema que permita la monitorización del sistema Endoworm 3.0. Como punto de partida se comienza con el diseño de un sistema de instrumentación que permita, mediante la implementación de sensores de presión, la medida de la presión existente en cada cavidad. Al mismo tiempo se desarrollará un sistema de envío de datos inalámbrico que permita eliminar los máximos cables posibles. Una vez recibida toda la información se procesarán los datos para comprobar el correcto funcionamiento de los sensores ya existentes, además de la posibilidad de detectar fugas de aire en el sistema.

# Abreviaturas y términos empleados

cm : centímetro
mm : milímetro
m : metro
V : voltio
mV : milivoltio
A : amperio
mA : miliamperio
µA : microamperio
kPa : kilopascales
psi : libra-fuerza por pulgada cuadrada
Ω : ohmio
% : porcentaje
ºC : grados
Mbps : megabits por segundo
mL : mililitro
CER : cavidad de expansión radial
CEA : cavidad de expansión axial (fuelle)

# Índice

# 1. Memoria

## 1.1 Justificación del proyecto

### *Introducción histórica*

El prototipo Endoworm 3.0. consiste en un sistema adaptable a los endoscopios actuales obteniendo así una mejora en su funcionamiento, el sistema está compuesto por varias cavidades de presión que permiten un avance autónomo y más rápido.

La palabra endoscopio proviene de los vocablos *endo*, "dentro" y *scopia*, "observar". Como se intuye, esta práctica consiste en observar el interior del cuerpo humano a través de herramientas médicas.

El origen de este procedimiento se remonta a la civilización greco-latina, cuna de occidente donde se sabe que tanto griegos como romanos empleaban ya algunos instrumentos cuya función era precisamente la de observar el organismo de personas vivas desde el interior.

Hacia el primer milenio de nuestra era, el médico Albuskasim fue el primero en utilizar el reflejo de la luz para examinar cavidades internas. No fue hasta finales del siglo XVI cuando Wilhelm Fabry, conocido como "el padre de la cirugía alemana", inventó el *speculum auris*, el cual sigue en uso actualmente conocido como otoscopio que permite la visualización de las cavidades internas del oído.

Llegado el siglo XIX otro médico alemán, Philip Bozzini, inventó el percusor del endoscopio moderno (Figura 1), refiriéndose así al tubo *lichtleiter* que significa conductor de luz. Este instrumento tenía una cámara de doble luz que utilizaba como fuente de iluminación una vela común y un espejo. Con este invento Bozzini logró observar el recto, la faringe, la uretra, la vejiga y realizar hallazgos como cálculo[1] y neoplasias[2] en el aparato urinario.

Años más tarde, en 1853, Antoine Jean Desomeaux, médico francés, desarrolló y presentó el Uteroscopio. Este aparato cuya función principal era visualizar y examinar el tracto urinario. Posteriormente este invento fue mejorado (Figura 2), conformado por dos secciones: la primera constituida por sondas cuya forma y calibre dependían de la región a explorar; más un sistema de iluminación, formado por lentes condensadoras, el cual reflejaba ondas de luz y las proyectaba en el aparato explorador en un punto en concreto, para la creación de la luz se sustituyo la vela que usaba Bozzini por una mezcla de alcohol con turpentina (fluido obtenido de la destilación de la resina de los pinos) que aumentaba la iluminación.

[1]Los cálculos pueden formarse cuando la orina tiene un alto contenido de ciertas sustancias que forman cristales que se almacenan en el riñón.

[2]Neoplasia es el término que se utiliza en medicina para designar una masa anormal de tejido. Se produce porque las células que lo constituyen se multiplican a un ritmo superior a lo normal.

Figura 1 - Endoscopio de Philip Bozzini

Figura 2 - Uteroscopio de Antoine Jean Desomeaux

A finales del siglo XIX se desarrollaron diferentes instrumentos como el cistouretroscopio por los médicos Max y Josef Leiter. También fue creado el primer gastroscopio rígido por Johann von Mikulicz, este instrumento no era para nada flexible y su inserción en el organismo era peligroso para los pacientes.

En 1932 Rudolph Schindler en conjunto con George Wolfry inventaron un gastroscopio flexible (Figura 3). Este era una versión corregida y aumentada de la mencionada anteriormente, el cual permitía llevar a cabo los estudios incluso con el tubo doblado. Dicho tubo medía 75 centímetros de largo y 11 milímetros de diámetro. Con este endoscopio se logró examinar el interior de un estómago a través de numerosas lentes ubicadas en todo el tubo con una lámpara eléctrica en miniatura.

Seis años más tarde Norbert Henning, gracias al invento de Schindler y Wolfry, publicó fotografías a color y películas de endoscopias. No fue hasta 1945 cuando la compañía Eder Instrument fabricó un modelo con mejoras significativas: mucha más flexibilidad, así como un sistema óptico que proporcionaba imágenes más nítidas. Más tarde, en 1952, Fuji y la Corporación Olympus crean la gastrocámara.

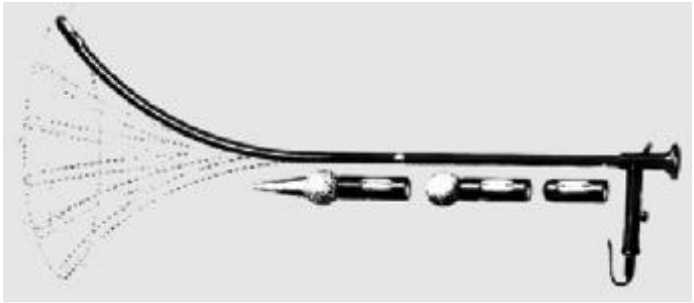


Figura 3 - Gastroscopio flexible gastrocámara

Figura 4 - Endoscopio alta definición

El siguiente avance se obtuvo en el año 1960 cuando Curtis, Hirschowitz y Peters construyen el primer fibroscopio. Este invento se trataba de un conjunto de fibras de vidrio muy finas las cuales transmiten los haces de luz en un tubo completamente flexible, el cual además estaba provisto de una lente óptica lateral que a través de una lámpara eléctrica colocada detrás de un prisma transmitía la imagen. El problema que encontraron en este invento fue la incapacidad de obtener fotografías. Este adelanto dio paso a la creación de lo que hoy conocemos como fibra óptica. En 1964 se inventó la primera gastrocámara con fibroscopio que permitía la obtención de fotografías.

A finales del siglo XX, en el 1983, hace su aparición el videoendoscopio y el endoscopio de ultrasonido. El primero creado por Sivak y Fleishner tiene como característica principal el uso de un chip para generar imágenes, consiguiendo la proyección del interior en un monitor. Además, cabe destacar, el uso de luz fría para evitar el contacto del paciente con un conductor eléctrico de baja tensión. Por su parte, el endoscopio de ultrasonido habilito el estudio de la superficie del tracto gastrointestinal así como también de las capas que se encuentran debajo, lo cual permite a los médicos determinar hasta qué punto una lesión o ulcera ha penetrado en las paredes del estomago o del intestino, y si se ha producido una metástasis en un nódulo linfático.

Por último, en 2002 se presentó el primer sistema de endoscopia del mundo basado en la tecnología HDTV (Figura 4), también llamada de alta definición.
En la actualidad se emplea el enteroscopio de doble balón, su predecesor fue el endoscopio monobalón (Figura 5). El sistema monobalón permitía la fijación del sistema en el punto deseado para su exploración. El enteroscopio de doble balón fue inventado por el doctor japonés Hironori Yamamoto en el año 1998, no fue hasta 2004 cuando se introdujo en el campo médico. Este endoscopio de pulsión-tracción avanza por el intestino hinchando y deshinchando los dos balones. Este sistema pretende replegar parte del intestino delgado para acceder a su zona media, con un endoscopio de pulsión sería imposible explorar esa zona debido a su extensa longitud y a los pliegues existentes.

**Figura 5 - Enteroscopio de doble balón y Enteroscopio monobalón**

El enteroscopio de doble balón fue un gran avance que mejoro la detección de lesiones vasculares y tumores en el intestino medio, pero posee una gran desventaja y es esta es que no se trata de un sistema autónomo, es decir, el facultativo que está realizando la exploración tiene que ir hinchando y deshinchando los balones en cada secuencia.

Este hecho hace que el procedimiento se haga más largo y tedioso, tanto para el médico como para el paciente. Para la mejora de esta situación se está desarrollando el prototipo Endoworm 3.0.

### *Funcionamiento del prototipo Endoworm 3.0*

Desde el año 2015, el grupo de investigación formado por gastroenteroĺógos endoscopistas del Hospital Universitari i Politécnic La Fe de Valencia, Físicos e Ingenieros en distintas especialidades de la Universidad Politécnica de Valencia, han centrado su trabajo en el proyecto de investigación "Desarrollo de un prototipo para la realización de la enteroscopia diagnóstica y terapéutica" orientado a buscar soluciones en este sentido. Fruto de este trabajo es la creación de un prototipo, denominado EndoWorm, que consiste en un sistema de traslación basado en cavidades hinchables adaptables a un endoscopio convencional que, con un movimiento coordinado, consiguen el objetivo propuesto: avanzar y replegar el intestino delgado sobre el propio endoscopio. El control del movimiento se realiza empleando un sistema neumático dirigido por un sistema electrónico basado en un microcontrolador. El sistema consta de los siguientes elementos:

- Medios de traslación neumáticos: comprendido por tres cavidades de silicona que al expandirse y contraerse de forma coordinada consiguen un movimiento lineal. Sus dimensiones se adaptan al calibre del endoscopio donde va situado, y tiene en cuenta el diámetro y la textura del intestino que deben replegar.
  En la Figura 6 se puede observar el motor neumático de la versión 2.0 debido a que el modelo actual no puede ser mostrado por derechos de autor. Siguiendo la imagen de izquierda a derecha la primera cavidad se trata de la cavidad de expansión axial (CEA), esta parte permite un avance axial, seguido se encuentra la cavidad de expansión radial móvil (CER móvil) que se trata de una cavidad de fijación que varía su situación en el endoscopio según la fase en la que se encuentre, por último la cavidad más a la derecha se trata de la cavidad de expansión radial fija (CER fijo) que al igual que la anterior permite la fijación del sistema.

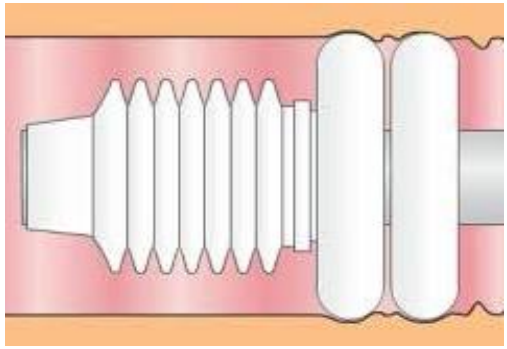


**Figura 6 - Endoworm 2.0**

- Medios electrónicos de control: permiten el movimiento automatizado y coordinado de las diferentes cavidades en el prototipo. Está integrado por: medios lógicos y programables de control (microcontrolador), medios actuadores sobre los medios de traslación (bomba de aire a presión y válvulas de llenado y vaciado), medios captadores de presión (sensores de presión diferencial en la cabecera de control) y medios de interfaz prototipo-usuario (pantalla táctil).

Para entender el funcionamiento del dispositivo se va a definir las diferentes fases presentes en un ciclo o secuencia de avance. Estas fases quedan determinadas estableciendo momentos de hinchado y deshinchado de cada cavidad. Se parte del reposo, es decir, con todas las cavidades sin volumen de aire en su interior, un ciclo presenta cinco fases:

- Fase 1 (Figura 7.a): Inflado del CER móvil. Si esta fase parte del reposo solo insufla aire a la cavidad mencionada, mientras si parte de la fase anterior (fase 5), el CER fijo se deshincha mientras se insufla aire al CER móvil.
- Fase 2 (figura 7.b): Hinchado del CEA. En esta fase es donde realmente se produce el avance, determinado por la longitud del mismo.
- Fase 3 (figura 7.c): Hinchado del CER fijo.
- Fase 4 (figura 7.d): Deshinchado CER móvil.
- Fase 5 (figura 7.e): Deshinchado del CEA.

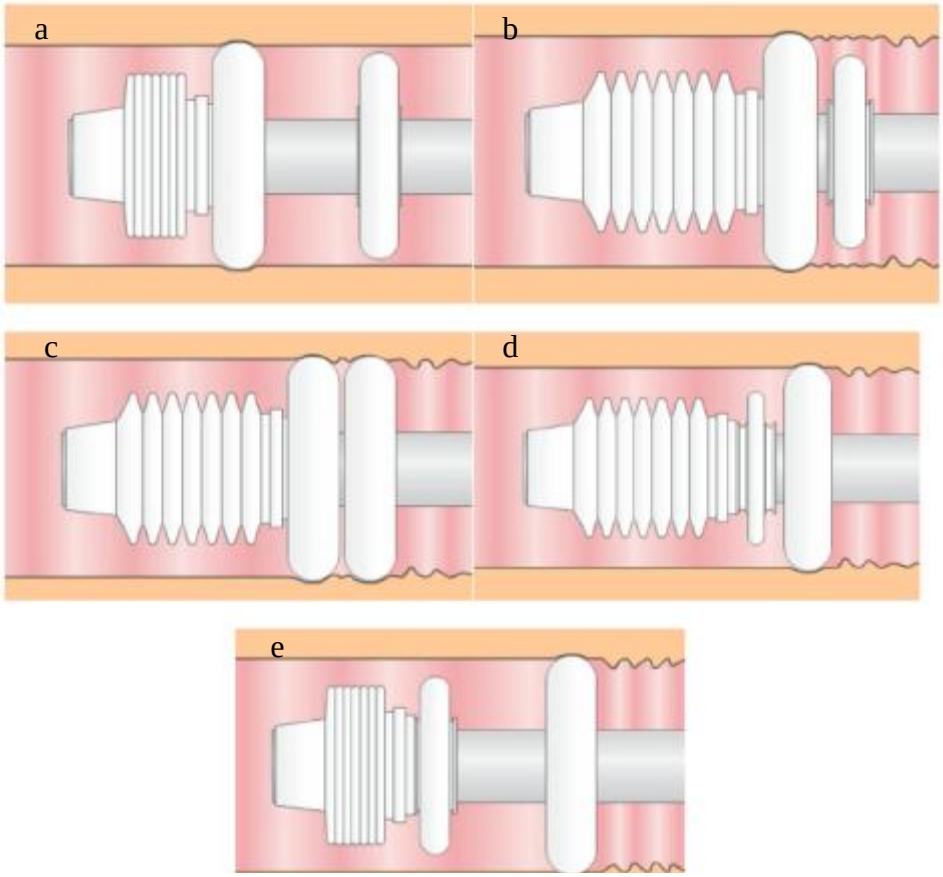



**Figura 7 - Fases en un ciclo de avance del prototipo Endoworm. Fase 1 (a), fase 2 (b), fase 3 (c), fase 4 (d) y fase 5 (e)**

El proyecto de investigación del prototipo Endoworm 3.0 está dividido en tres fases de trabajo. La primera se encarga de la fabricación de las diferentes cavidades del sistema mecánico. La segunda se trata de la mejora del software de control así como del interfaz prototipo-usuario. Por último la tercera fase está dirigida al diseño e implementación de un sistema de instrumentación que permita la monitorización del prototipo.

En la actualidad el proyecto se encuentra en una fase de mejora y de prueba en el laboratorio. Tras estos avances se pretende comenzar con los ensayos *in vivo* en animales para comprobar el correcto funcionamiento del prototipo.

## 1.2 Objeto del proyecto

El objeto del presente trabajo fin de grado consiste en la monitorización del prototipo Endoworm 3.0. Como se ha podido observar en el apartado anterior el prototipo posee tres cavidades a las cuales se les introduce aire a presión. Estas presiones en la actualidad son medidas desde la salida de la bomba en la cabecera del sistema, por lo que no se conoce si la presión al inicio del tubo será la misma que la existente dentro de la cavidad. Por este motivo se pretende diseñar e implementar un sistema de instrumentación que mida la presión existente en cada cavidad, para ello se colocarán los sensores lo más cerca posible a cada una de estas cavidades, de este modo se obtendrán resultados fiables que servirán para realizar una comparación con los valores ofrecidos por los sensores ya insertados.

La necesidad de conocer la presión real existente en cada cavidad reside en la máxima presión que se puede introducir en un paciente, de modo que si por software se configura una presión pero esta es mayor en la cavidad podría ocasionar una rotura del globo y una posible perforación en el intestino del paciente. Por tanto el conocimiento de este dato permite explorar los límites elásticos que posee los materiales de las diferentes partes y conocer la máxima presión que son capaces de soportar. Así mismo se podrá hacer una relación de la presión en cabecera con la presión real dentro de la cavidad.
También se pretende obtener gráficas de la evolución de las diferentes etapas, esto permitirá la detección de fugas y permitirá conocer el comportamiento de las cavidades en cada fase del proceso.

Los sistemas de monitorización irán sujetos al endoscopio, por ello el tamaño tiene que ser lo más reducido posible para no obstaculizar el avance del mismo. Otra causa del rozamiento son los cables de conexión del sistema a diseñar. Para ello se van a eliminar los máximos posibles añadiendo un sistema de envío inalámbrico que envíe los datos obtenidos a un ordenador externo, de este modo se podrán almacenar para su posterior procesado.
Además, en el mismo sistema, se van a introducir 2 acelerómetros para conocer la distancia recorrida por el endoscopio en cada ciclo del proceso. El software para el funcionamiento de estos sensores ya está implementado, solamente se necesitará la colocación y el envío de los valores obtenidos.

Los objetivos de este proyecto se pueden resumir en los siguientes puntos:

- Diseño e implementación de un sistema de instrumentación que permita la caracterización del prototipo Endoworm 3.0.
- Medir la presión en cada una de las cavidades (CEA, CER móvil, fija).
- Envío inalámbrico de los resultados obtenidos.
- Recepción de los datos y almacenamiento de los mismos.
- Procesado de los resultados obtenidos.

## 1.3 Soluciones alternativas

### *Alimentación*

La alimentación del sistema debe ser una tensión continua de 5 V con la suficiente corriente para que todos los componentes funcionen correctamente, para ello se han contemplado entre las siguientes opciones:

- Fuente de tensión: se trata de una fuente de alimentación variable conectada a la red eléctrica. Este tipo de alimentación suelen ser de un gran tamaño y tener un peso considerable, por lo que quedaría fuera del sistema, y se te tendría que conectar mediantes cables que tuvieran una longitud igual a la del endoscopio. Esto ofrece una corriente hasta de 2,6 A que sería suficiente para alimentar a todos el sistema.
- Pilas: el uso de pilas permitiría al sistema de monitorización ser independiente y no necesitar ningún tipo de conexión con el exterior. Existe una gran gama de pilas en el mercado, en este caso se necesitaría pilas de botón debido a su reducido tamaño. La máxima tensión y corriente en este tipo de pila son de 3 V y 35 mA respectivamente.

Debido a la demanda de la corriente para el optimo funcionamiento del sistema, añadido al inexistente espacio para la localización de las pilas hacen que el tipo de alimentación seleccionado sea mediante una fuente de tensión. Aunque el hecho de conectarlo mediante un cable de una longitud superior a los 2 m ralentizará el avance del prototipo, esto asegurará que el aporte de corriente y de tensión es la adecuada en todo momento.
Una vez que el sistema de monitorización este en funcionamiento se pretende mejorar este apartado añadiendo una alimentación interna e independiente de la red eléctrica.

### *Lectura de datos*

El método para la obtención de datos que se va a emplear es un microcontrolador que permita la lectura de estos mediante puertos analógicos. Existen dos opciones para tal fin:

- Implementación de un solo microcontrolador en la cabecera del sistema que se comunique con cada uno de los sensores mediante cable.
- Conexión entre dos microcontroladores. El primero situado en el sistema de monitorización que se encargue de la lectura y del procesado inicial de los datos. Y el segundo situado fuera del sistema que sea el receptor de estos y realice las operaciones necesarias para la obtención de los valores deseados.

En el primer caso la adquisición de los datos seria sencilla y con una frecuencia mayor al segundo. El hecho de añadir un exceso de cables que permitan la unión del microcontrolador con los sensores provocaría una ralentización del sistema, al mismo tiempo se podrían producir bloqueos en el avance del prototipo. Por ello, la mejor opción se trata de la implementación de dos microcontroladores que permitan obtener los datos mediante la comunicación inalámbrica entre ellos.

### *Transmisión de datos*

Tras la selección del medio de lectura con dos microcontroladores que se comuniquen, la transmisión de datos debe de realizarse mediante algún método que permita el envío de datos de forma inalámbrica. Los medios estudiados son los siguientes:

- Módulo Bluetooth: se trata de una placa que integra todos los componentes necesarios para el envío de datos mediante esta tecnología. Existen dos tipos de módulos según su configuración; el esclavo, que solamente permite la recepción de los datos, y el maestro, que se encarga únicamente de la emisión. La conexión entre ellos se realiza mediante una dirección IP que se configura en ambos dispositivos por igual, quedando así restringido el acceso de cualquier otro dispositivo. Este tipo de módulo es idóneo para proyectos sencillos en los que se realizan emisiones de código ASCII o de valores numéricos enteros y sin signo.
- Módulo Radiofrecuencia: de igual modo que la anterior son placas ya implementadas en las cuales se integra un módulo de radiofrecuencia que permite el envió y la recepción mediante esta tecnología. En este caso, no existen diferentes tipos según su configuración ya que tanto el emisor como el receptor puede configurarse según los comandos empleados. Existe una gran variedad en los paquetes que se pueden transmitir, ya sean valores enteros y sin signo a valores con decimales y negativos. Permite el envío de cadenas de datos hasta una longitud de 5 bytes. La conexión entre ambos módulos se realiza emparejándolos mediante una dirección IP determinada, al igual que en el caso anterior. Para su configuración se necesitan librerías propias del módulo en las que están definidos los comandos a emplear.

Dado que los datos que van a transmitirse son con decimales, para no perder precisión en los valores, se emplearan módulos de radiofrecuencia que se configurarán como emisor, en el caso del introducido en el sistema de monitorización, y como receptor, el situado en el microcontrolador externo.

### *Almacenamiento de datos*

Todos los valores obtenidos durante la toma de datos del sistema tendrán que ser almacenados para su posterior procesado. El hecho de que el microcontrolador posee una memoria insuficiente para el almacenamiento de paquetes tan grandes, hace que se necesite emplear otro medio que realice esa función.

Las alternativas contempladas son las siguientes:

- Módulo MicroSD: este módulo consiste en un adaptador de una tarjeta MicroSD a los pines del microcontrolador. Permite mediante una librería propia el almacenamiento de todos los valores que se reciban. Para su correcto funcionamiento se debe iniciar la tarjeta antes de comenzar su uso y para que los datos recibidos se guarden correctamente se debe de emplear la función que finaliza el uso del mismo.
- Clear Terminal: se trata de un programa para Windows en el cual se puede configurar el puerto y la frecuencia de entrada de los datos, esto permite la visualización de los valores recibidos en la pantalla del programa. Otra de las funciones se trata de añadir la fecha y la hora en la cual se ha obtenido cada paquete de datos. Una vez se ha registrado toda la información te permite guardarla en diferentes formatos, el más común en formato de texto (.txt).

La opción escogida es el programa Clear Terminal debido a que se trata de un software externo que no necesita ser implementado en el código del microcontrolador. Al evitar introducir más líneas de código se agiliza la ejecución del programa. Además evita añadir más hardware al sistema que pueda producir fallos ajenos a la programación.

## 1.4 Descripción de la solución adoptada

### 1.4.1 Materiales empleados

#### *Sensor de presión*

En el mercado existe una gran variedad de sensores de presión, una de las grandes diferencias se trata del tipo de medida que realiza ya que se puede tratar de un sensor de presión absoluta o de presión diferencial. En el caso de los sensores de presión absoluta el valor ofrecido se trata de la suma de la presión existente en la cavidad a medir más la presión atmosférica en ese momento, por el contrario el sensor de presión diferencial te ofrece el resultado de la resta entre la presión medida dentro de la cavidad menos la presión atmosférica.
La ubicación del sensor en el sistema estará en un cavidad cerrada, por lo que una medición diferencial no sería posible ya que la variación en un espacio cerrado es 0. El tipo de sensor a emplear será de presión absoluta. En este momento otra característica que se ha de contemplar, a parte del tipo de medida, es el rango de presiones que es capaz de medir. En este caso debido a que la presión máxima que se puede usar en el sistema, para evitar daños internos en el paciente, es de 150 kPa el rango superior de medición tendrá que estar por encima de este valor.

El sensor escogido es el modelo SCC30ASMT (Figura 8) de la marca Honeywell. Se trata de un sensor que mide presiones absolutas y posee un tamaño muy reducido (7,6 x 7,6 x 3,7 mm). Dentro del mismo modelo existen diferentes rangos de presiones, en este caso se selecciono el de mayor valor, de 0 a 30 psi (0 a 206,84 kPa).

Como se puede observar el valor superior es mucho más elevado que el máximo existente en el sistema, esto se debe a que el modelo anterior alcanza como máximo 15 psi (103,42 kPa) que se trata de un valor muy por debajo del máximo estipulado.

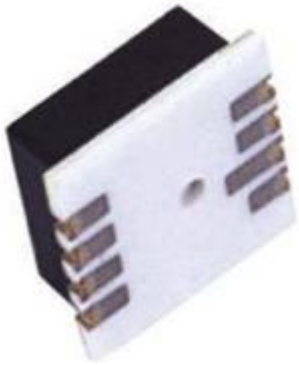


**Figura 8 - Sensor presión absoluta SCC30ASMT**

Se trata de un sensor alimentado por corriente que ofrece como salida una tensión diferencial, para transformar este valor a unidades de presión será necesario conocer la sensibilidad exacta de este. Como se puede observar en el datasheet (Anexo I) la sensibilidad ofrecida por el fabricante es de 2,90 ±0,57 mV/mA/psi. Debida a la desviación estándar existente la sensibilidad de cada uno de los sensores puede oscilar entre [2,33 3,47] mV/mA/psi.
Esta gran variación hace necesario la calibración de cada uno de los sensores para así obtener la sensibilidad real de cada uno de ellos.

Para comenzar con la calibración de los sensores se ha de definir una corriente de alimentación. El máximo valor posible es de 1,5 mA, por seguridad se escoge un valor de 0,5 mA. Una vez seleccionado este valor los elementos necesarios para comenzar con el proceso son los siguientes:

(a) Encapsular los sensores en una zona estanca.
(b) Una fuente de corriente que ofrezca la alimentación necesaria para el funcionamiento de los sensores.
(c) Un manómetro digital con una precisión de décimas.
(d) Una de las cavidades del prototipo Endoworm que nos permita simular un hinchado real.
(e) Tubos conectores entre los diferentes elementos.
(f) Un sistema manual de generación de aire a presión controlable.

Antes de comenzar el proceso de encapsulado es necesaria la soldadura de los cables que saldrán del sensor. Como se puede observar en la Figura 8 hay 8 pines de soldadura de los cuales solamente son necesarios 4, estos son los pines de la derecha, fácil de reconocer por tener uno de los pines más largo que el resto. Por motivos de seguridad se afianza la soldadura con tubo termoretráctil, como se puede observar en la Figura 9, debido al pequeño tamaño del sensor la soldadura realizada es muy débil y ante cualquier pequeño movimiento del cable correría el riesgo de soltarse. Como los sensores van encapsulados el hecho de tener que volver a soldar produciría grandes retrasos en el montaje del sistema.

Figura 9 -Sensor de presión soldado y reforzado con tubo termoretráctil

Figura 10 -Molde encapsulado inicial forma circular

Una vez se han soldado los cables a la distancia necesaria se prosigue con el encapsulado de los sensores que se trata del punto más importante. Al final de este proceso se necesita que los sensores estén perfectamente sellados ya que cualquier fuga produciría una mala medición y por tanto un dato falso.

Como se ha comentado anteriormente, este sistema será introducido en un intestino animal por lo que se ha de tener en cuenta tanto el tamaño como el material. Por ello no se pueden introducir elementos con bordes puntiagudos que entorpezcan el avance del dispositivo ni hieran el intestino. Por este motivo el material seleccionado para el encapsulamiento es silicona Sylgard. Se fabrica en forma circular de 4 cm de diámetro (en la Figura 10 se puede observar el molde utilizado para la realización del encapsulado, se trata de una capsula de Petri) para posteriormente doblarse e introducir el sensor y los tubos de salida y entrada en él, una vez colocado se sellaría el borde con silicona transparente que cure al aire (proceso terminado en Figura 11).

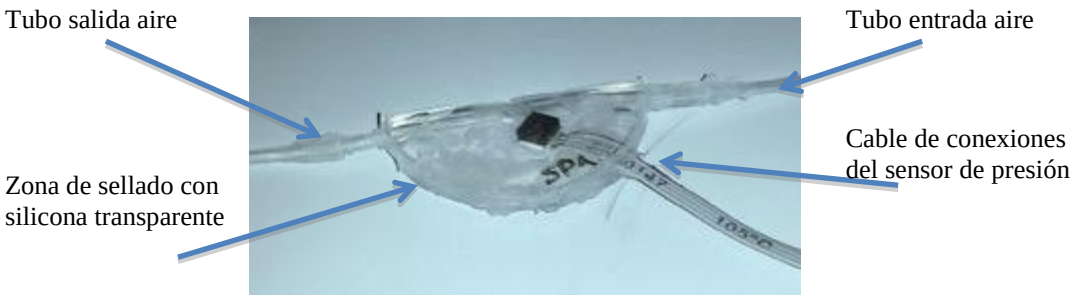



Figura 11 - Sensor de presión encapsulado

Pero este método no es efectivo, ya que la zona de sellado del encapsulado no posee la suficiente fuerza para aguantar la presión cuando se introduce el aire y se generan fugas.

Por este motivo se cambió la forma del encapsulado por una forma cilíndrica, de diámetro de 1 cm, donde al mismo tiempo se modifico el tipo de silicona por la Silastic 4820.

Este formato de encapsulado solamente requerían el sellado en los extremos, donde además del sellante se colocarán dos bridas, del menor tamaño posible, para afianzar la unión y darle más fuerza (en la Figura 12 se puede observar el encapsulado de forma cilíndrica y en la Figura 13 el sistema encapsulado abajo y sin encapsular con las bridas de apoyo arriba). También se ha recubierto el encapsulado con cinta adhesiva de alta fijación para evitar que la silicona pueda expandirse e impedir el paso del aire al conducto de salida.

Figura 12 - Encapsulado cilíndrico

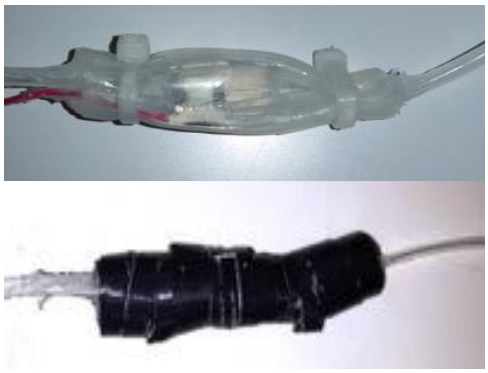


Figura 13 - Sensor presión sin cinta con forma cilíndrica arriba y sin cinta abajo

Una vez terminado el encapsulamiento de los sensores se procede a realizar la calibración de los mismos. La fuente de corriente seleccionada es la LM334 (se comentará más profundamente en el siguiente apartado y se aportarán los cálculos necesarios) regulada con una resistencia de 120 Ω para ofrecer a la salida una corriente de 0,56 mA. El esquema de conexión para el calibrado es el siguiente:

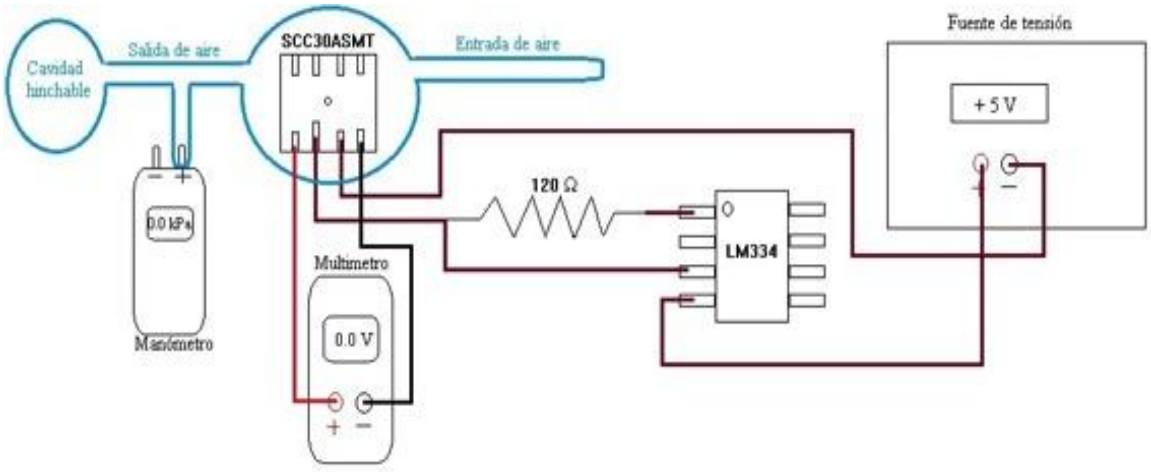



Figura 14 - Esquema de conexiones entre los diferentes elementos para el calibrado del sensor de presión

Como se puede observar en la Figura 14 el sensor de presión encapsulado está conectado a una cavidad hinchable que nos permite medir diferentes presiones. En la unión de la cavidad con el sensor se coloca el manómetro digital que nos dará la presión existente en kPa. Al mismo tiempo se conecta al sensor la fuente de corriente regulado con la resistencia para alimentarlo con la corriente deseada.

La fuente de corriente está alimentada a +5 V no teniendo ningún pin conectado a masa. Por último se conecta al sensor el milímetro para medir la tensión diferencial que este nos ofrece según la presión interna.
Una vez realizado el montaje se comenzaría con la calibración del sensor. Como los cambios de presión son muy rápidos se van a tomar fotografías para así tener valores instantáneos que nos permitan obtener la sensibilidad con mayor precisión. Las muestras tomadas para cada sensor son 16.
Un ejemplo de las fotografías tomadas es la Figura 15, donde se puede observar a la izquierda el milímetro que mide la tensión diferencial a la salida del sensor de presión, y a la derecha el manómetro midiendo la presión diferencial existente en la cavidad.

.


**Figura 15 - Fotografía tomada en el proceso de calibración del sensor de presión. A la izquierda el multímetro y a la derecha el manómetro digital**

Volviendo a la Figura 14 se puede observar que el manómetro solo tiene conectado la boquilla positiva, esto se debe a que hace mediciones diferenciales, es decir, el valor mostrado por el display es la resta entre la presión interna de la cavidad menos la presión atmosférica.
Una vez obtenido todos los valores de tensión para cada presión estos se colocan en tablas para poder obtener las gráficas que nos faciliten la recta de la pendiente y así obtener la sensibilidad del sensor. A continuación se muestra un ejemplo de las gráficas obtenidas y Tabla1 que se trata del resumen de las sensibilidades obtenidas para los 12 sensores encapsulados. Las demás gráficas así como las tablas completas se encuentran en el Anexo II de la presente memoria. Estos sensores van enumerados con las siglas SP.30.XX, siendo XX el numero del sensor desde el 00 al 12. Cabe destacar que el sensor SP.30.08 ha sido descartado por problemas del mismo.

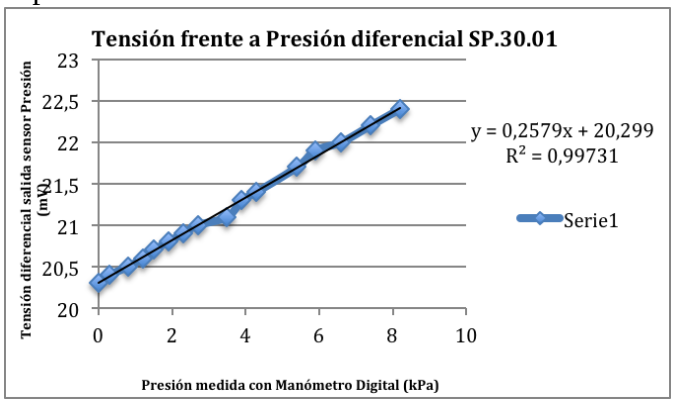



**Figura 16 - Relación de la presión medida con el manómetro con la tensión diferencial ofrecida por el sensor de presión**

**Tabla1 - resumen sensibilidades sensores SP.30.XX**

| SP.30.XX | Sensibilidad (mV/kPa) | Sensibilidad (mV/kPa*uA) |
|---|---|---|
| 0 | 0,2096 | 4,2776E-04 |
| 1 | 0,2579 | 4,5727E-04 |
| 2 | 0,2495 | 4,4316E-04 |
| 3 | 0,2157 | 4,3488E-04 |
| 4 | 0,266 | 4,7585E-04 |
| 5 | 0,2822 | 5,0393E-04 |
| 6 | 0,2619 | 4,5947E-04 |
| 7 | 0,2349 | 4,1283E-04 |
| 9 | 0,2425 | 4,2544E-04 |
| 10 | 0,2594 | 4,5429E-04 |
| 11 | 0,2698 | 4,7417E-04 |
| 12 | 0,2546 | 4,5062E-04 |

Como se puede observar en Figura 16 la linealidad de las muestras (representado con el símbolo $R^2$) es 0,99731, como el valor se aproxima a 1 se puede decir que la gráfica es lineal y por tanto se toma la muestra como válida. En el caso de tener un valor de linealidad muy por debajo de 1 se tendría que volver a realizar la toma.

El valor de la sensibilidad de la muestra se obtiene de la pendiente de la recta, la ecuación de esta es y=mx+b donde m es el dato de interés. En el caso de la Figura 16 la sensibilidad es de 0,2579 mV/kPa.

En la Tabla1 se tienen todos los valores de sensibilidad de cada sensor calibrado. En la segunda columna se puede observar que se trata de la sensibilidad en mV/kPa, mientras que en la tercera se trata de la sensibilidad en función de la corriente.

Esto se debe a que los sensores de presión varían su salida dependiendo de la corriente de alimentación. No habría problema si las fuentes de corriente fuesen invariantes en el tiempo, pero la realidad es que varían en función de la temperatura (se comenta con más datos en el siguiente apartado) por tanto la alimentación de los sensores puede variar además de variar su valor de salida. Para ello se halla la sensibilidad en mV/kPa·µA y así calcular el valor real en cada iteración.

### *Fuente de corriente*

La fuente de corriente es el elemento que permite el funcionamiento de los sensores de presión.

El modelo seleccionado es el LM334 de la marca Texas Instruments, se escoge este modelo debido a su pequeño tamaño, a que ofrece una corriente de salida suficiente para el fin destinado y a que permite un amplio rango de corriente de salida solamente regulándola con un potenciómetro. El número de fuentes que se utilizarán serán 12, 9 de las cuales serán necesarias para el montaje de los 3 sistemas y otras 3 de reserva por si fallase alguna.

Como anteriormente se ha mencionado la corriente deseada es de 0,5 mA, para que la fuente ofrezca este valor se ha calcular la resistencia de carga que la limite, para ello se utiliza la ecuación ofrecida por el datasheet (incluido en el Anexo I de esta memoria) que es la siguiente:

$$R_{set} = \frac{V_R}{I_{set}} \cdot \frac{n}{n-1} = \frac{64}{0{,}5} \cdot \frac{18}{18-1} = 135{,}53\ \Omega$$

donde $V_R$ y n vienen dadas como dato típico e $I_{set}$ es la corriente deseada.

Para el montaje de las placas no existen resistencias de montaje superficial de 135,53 Ω, y para evitar cualquier tipo de variación del valor de la resistencia no se puede usar un potenciómetro. La solución adoptada fue colocar una resistencia de 137 Ω, se trata de un valor un poco por encima del ideal pero nos ofrece una corriente similar a la planteada. Cabe recordar que no hay ningún problema con que la corriente sea más elevada a la fijada ya que los sensores son capaces de soportar hasta 1,5 mA. La alimentación de la fuente de corriente se trata de tensión continua entre un rango de 1 y 40 V, en este caso se va a fijar la tensión en 5 V y así establecer una tensión de alimentación estándar para todos los elementos del sistema.
En la Figura17 se puede observar la conexión de la fuente de corriente.

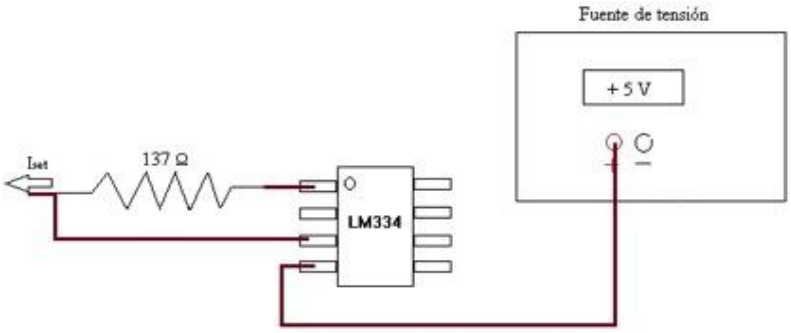



**Figura 17 - Conexión fuente de corriente**

En el apartado anterior se ha comentado que la fuente de corriente varía con respecto a la temperatura, según el datasheet el porcentaje de variación es de 0,336%/°C, es decir que por cada grado que se aumente la temperatura, la corriente aumentará un 0,336 %, sabiendo que la corriente es de 0,5 mA , por cada grado que se aumente en el ambiente la corriente aumentará 1,68 μA. Puede que este incremento no sea muy elevado, pero como se precisa la máxima exactitud en los cálculos se procede al calibrado de las fuentes de corriente según la temperatura para estimar la sensibilidad real en cada una de ellas.

Para ello es necesario introducir un sensor de temperatura en el sistema que nos indique en cada iteración la temperatura ambiente existente y permita realizar los cálculos en función de la misma. El sensor de temperatura seleccionado es el LM35DZ (en el Anexo I se puede consultar el datasheet) de la misma casa que la fuente de corriente.

Se trata de un sensor muy usado en la actualidad debido a su sencillez, ya que para su implementación solamente es necesaria una resistencia de carga que limite la corriente de entrada. No se necesita ningún elemento más para obtener el valor de temperatura real, ya que este ofrece una salida en mV que se corresponde a la temperatura en grados centígrados existente. En este caso se utiliza una resistencia de carga de 200 Ω.

Una vez se tiene el sensor de temperatura implementado se continua con la calibración de las fuentes de corriente. Para ello se van a implementar 2 sensores de temperatura colocados alrededor de la fuente para realizar una estimación de la temperatura envolvente. El hecho de que esta no sea uniforme se debe al método empleado, un tanto rudimentario, para la calibración. Este consiste en una bolsa térmica con una abertura en la parte superior por la cual se introducirá la boca de un secador que ofrezca aire caliente y aumente la temperatura del recipiente. Una vez se haya alcanzado la temperatura máxima, en este caso es de 50ºC, se irán tomando las medidas durante el descenso de esta.

Los valores son anotados en la bajada de temperatura en vez de en la subida, esto es debido a que en la bajada la temperatura es más uniforme en todo el contenedor, cosa que hace que la toma de datos sea más lenta, de este modo la gráfica obtenida tiene mayor linealidad y ofrece una sensibilidad más exacta.
En la Figura 18 se puede observar el elemento empleado para la calibración de las fuentes, se trata de una bolsa térmica que mantiene durante un largo periodo de tiempo la temperatura interna. En la Figura 19 se puede observar el circuito implementado para el calibrado, la placa será introducida dentro de la bolsa para comenzar con el proceso.

**Figura 18 - Bolsa térmica**

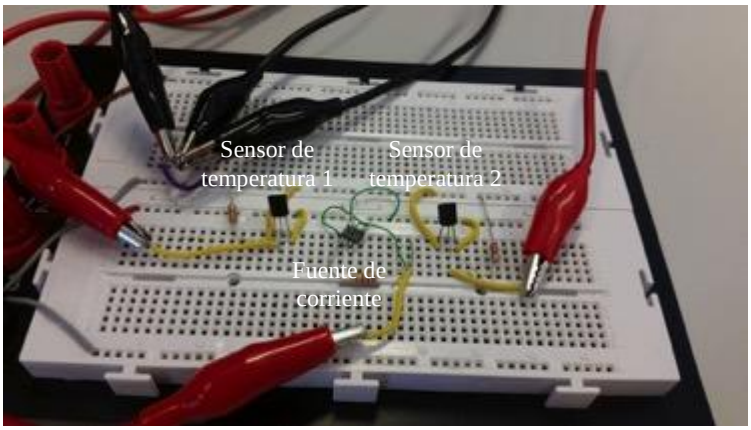



**Figura 19 - Placa board con circuito de calibración conectado**

Al igual que en el caso de los sensores de presión los datos han sido obtenidos mediante la realización de fotografías. En estas se visualizan la temperatura ofrecida por cada una de los sensores medido mediante 2 multímetros diferentes y un tercer multímetro con la corriente ofrecida por la fuente. En ambos casos los sensores son alimentados por +5 V.

Una vez obtenidos todos los valores, estos han sido trasladados a tablas para posteriormente realizar las gráficas que nos ofrezcan la sensibilidad. En la Figura 20 se puede observar un ejemplo de las graficas obtenidas en el procesado de los datos y en la Tabla 2 el resumen de la sensibilidad de cada una de las fuentes de corriente.

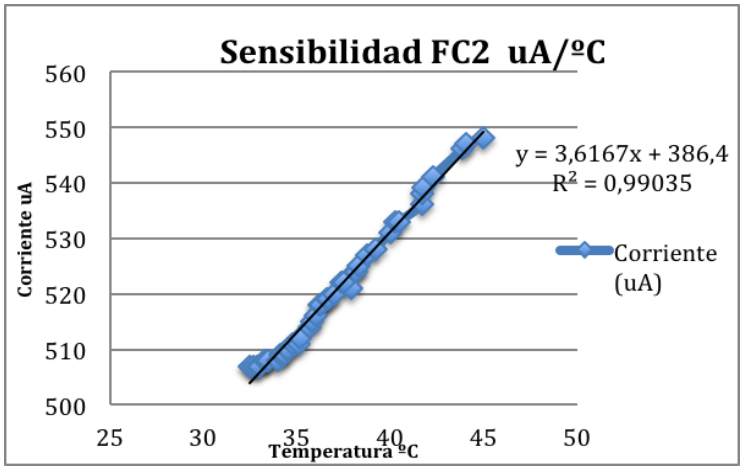



**Figura 20 - Gráfica obtenida de la corriente ofrecida por la fuente frente a la temperatura ambiente**

**Tabla 2 -Resumen sensibilidades fuentes de corriente**

| Sensibilidad Final de cada FC | |
|---|---|
| **Número FC** | **Sensibilidad (uA/°C)** |
| 0 | 2,6813 |
| 1 | 3,1327 |
| 2 | 3,6167 |
| 3 | 2,6288 |
| 4 | 2,2819 |
| 5 | 3,6753 |
| 6 | 3,2374 |
| 7 | 4,0119 |
| 8 | 3,4792 |
| 9 | 2,6448 |
| 10 | 4,2893 |
| 11 | 3,5432 |

Del mismo modo que se comentó en el apartado anterior, la sensibilidad de la fuente es ofrecida por la pendiente existente en la línea de tendencia del gráfico. Este valor viene determinado en la ecuación de la recta que se puede observar n la Figura 20, y =3,6167x+386,4 donde 3,6167 es la pendiente y por tanto la sensibilidad del sensor.

Como el gráfico muestra corriente frente a temperatura las unidades de esta sensibilidad será de µA/°C. En el Anexo II de la presente memoria se pueden observar las tablas obtenidas para cada fuente y sus correspondientes gráficas.

En la Tabla 2 se puede observar el resumen de las 12 fuentes calibradas, viendo los resultados obtenidos se comprueba que la calibración de las mismas era necesaria, si recordamos el valor de variación con respecto a la temperatura que ofrecía el datasheet, 1,68 μA/ºC, ninguna de las fuentes se acerca a ese valor.
Una vez terminado todo el proceso de calibración ya se puede proceder a su colación en las placas del sistema, estas serán comentadas y explicadas en el punto *1.3.2.* de la presente memoria.

### *Módulo de radiofrecuencia*

Uno de los principales problemas en este proyecto es el mínimo espacio existente para la colocación del sistema de monitorización, tal como se ha mencionado anteriormente, el hecho de que tenga que introducirse junto con el endoscopio por cavidades estrechas hace la obtención de datos un tema complicado.
La primera posibilidad planteada fue recibir los datos obtenidos de los sensores de presión, de los acelerómetros y del sensor de temperatura mediante cables que llegaran hasta el equipo de medida fuera del intestino.
Esta opción ralentizaría mucho el proceso de avance del endoscopio debido a que este tendría que ir tirando de una longitud de cable muy elevada.

La mejor opción es el envío inalámbrico de la información y para ello se va a utilizar un módulo de radiofrecuencia por su gran versatilidad en la programación del formato de envío.

El modelo escogido es el NRF2401, se trata de un modelo comercial muy económico. Aunque su tamaño no sea tan reducido como el deseado es de los más pequeños existentes en la actualidad. Estos módulos son transceptores, es decir mediante programación los podemos poner como emisores o como receptores o configurarlos para que hagan ambas cosas. Puede llegar a transmitir datos a 2 Mbps en distancias menores a 40 metros al aire libre y a 1Mbps a 8 metros de distancia. Se alimentan con una tensión de 3,3 V por tanto tienen un bajo consumo.

**Figura 21 - Módulo de radiofrecuencia NRF2401 con antena externa**

**Figura 22 - Módulo de radiofrecuencia NRF2401 con antena integrada**

Existen varios formatos de este módulo, el primero (Figura 21) se trata del modelo sencillo que lleva la antena acoplada en su placa, mientras el otro modelo (Figura 22) se trata de una versión más alargada de placa con un acople para una antena externa de mayor alcance. Se va a utilizar el modelo con la antena integrada como emisor de la información y el de antena externa como receptor.

Estos módulos están especialmente diseñados para su programación con Arduino. Poseen librerías especificas para dicho programa, como la RF24.h y nRF24L01.h, que permiten una configuración más sencilla e intuitiva (librerías incluidas en el Anexo III de la presente memoria). Además para el correcto funcionamiento es necesario añadir la librería SPI.h propia de Arduino que realiza la configuración de los puertos. A continuación se va a comentar algunas de las funciones básicas que han sido necesarias en su configuración para este proyecto.

Pero antes es importante saber cuál es la conexión correcta del módulo, este posee 8 pines de los cuales se van a conectar 7. Los dos primeros se tratan de la alimentación y masa los cuales permiten el funcionamiento del módulo, seguidos se encuentran los pines Cs y Ce que son los pines SPI (*Serial Peripheral Interface* o en castellano interfaz de periféricos serie) que se tratan de los pines necesarios para el intercambio de información entre diferentes dispositivos.

Por último están los pines ISCP (siglas en inglés de *In-Circuit Serial Programmingo* en castellano Programación serie en el circuito*)* estos son los encargados de permitir la programación desde el Arduino al chip del módulo, este puerto realiza la programación directa del dispositivo sin tener que emplear un programador inicial que lo configure. En la Figura 23 se puede observar la colocación de los pines.


**Figura 23 - Pines de conexión módulo de radiofrecuencia NRF2401**

El primer comando que se ha de escribir es el que define los puertos Cs y Ce, en este caso son los puertos digitales 7 y 8 del Arduino, este comando quedaría de la siguiente forma:

```
>>RF24 radio(7, 8);
```

Seguidamente se procede a la identificación del canal por el cual se van a comunicar los módulos mediante la definición de una variable constante, esta dirección está constituida por 5 dígitos que pueden variar entre 0 y 1, no importa el valor que tenga cada digito siempre que en los módulos que se vayan a interconectar sean los mismos. El comando quedaría del siguiente modo:

```
>> const byte rxAddr[6] = "00010";
```

En el caso del módulo emisor los comandos a escribir son los siguientes:

```
>>radio.begin();

>>radio.setRetries(15, 15);

>>radio.openWritingPipe(rxAddr);

>>radio.setPALevel(RF24_PA_HIGH);

>>radio.stopListening();
```

El primer comando comienza la comunicación con el módulo receptor, el siguiente es una función que configura el tiempo de espera que se quiere entre cada intento de envío y el número de intentos que se ha de realizar en el caso de que el paquete no sea recibido. Continua con un comando que realiza la escritura de los paquetes. Seguido de la configuración del amplificador de potencia, este puede ser mayor o menor en función de la potencia deseada para el envío de los paquetes, a mayor potencia mayor será el alcance de la señal. Por último se corta la escucha del módulo para evitar la recepción de cualquier paquete entrante.

Una vez se tiene configurado el módulo se procede a la escritura de los comandos necesarios para el envío de los paquetes deseados. En este caso los datos han sido unidos en cadenas de valores que permite que en un solo paquete se envíen todos los datos obtenidos en esa iteración. La cadena es declarada como un valor de coma flotante, float, debido a que los sensores ofrecen los resultados con decimales y por tanto con una variable entera se perdería información. El comando utilizado para el envío de la cadena es:

```
>> radio.write(&cadena, sizeof(cadena));
```

Se trata de un comando sencillo en el que solo es necesario la introducción de la variable a enviar y el tamaño de la misma.

Para la correcta configuración del módulo receptor los comandos necesarios no son tantos como los del emisor, esto se debe a que este solamente va a recibir y no a mandar. Los comandos para la configuración inicial del módulo serían:

```
>> radio.begin();

>> radio.openReadingPipe(0, rxAddr);
```

```
>> radio.startListening();
```

Estos comandos son muy parecidos a los utilizados para el módulo emisor. El primero se trata del comando que inicia la comunicación entre las dos partes. Luego viene el comando que pone al módulo en modo de lectura, los valores que se le introducen son: el canal por el cual se quiere leer y la dirección por la cual se transmite la información. Por último se introduce el comando que permite al módulo receptor detectar los paquetes entrantes.
Para terminar con la programación de los módulos se necesita programar una condición para que solamente ofrezca los valores leídos en el caso de haberlos recibido y no permita la reiteración de un mismo valor, para ello se implementará el siguiente condicional:

```
>> if (radio.available()>0)
{
     radio.read(&valor, sizeof(valor));
}
```

El condicional if solamente leerá el paquete entrante siempre que el comando del módulo que determina si está conectado dé un valor superior a 0, es decir un valor high. Una vez dentro del condicional el comando empleado es sencillo, ya que solamente hay que introducirle el nombre de la cadena que hemos declarado para la recepción del paquete al igual que se ha realizado para el envío y el tamaño de la misma. Para evitar problemas de tamaño es recomendable utilizar la longitud usada en la declaración del paquete de envío.

Una vez terminada la configuración los módulos están listos para su uso. Todo el código anteriormente explicado será incluido en el programa final implementado para la lectura y el guardado explicado y comentado en el apartado siguiente denominado microcontrolador, además el código completo está incluido en el Anexo IV de la presente memoria.

### *Acelerómetro*

Otro de los datos que se desean obtener en la monitorización del prototipo es la distancia que este recorre en cada ciclo del sistema. Para ello se emplean dos acelerómetros que permiten la medición de este parámetro. El modelo del acelerómetro es el ADXL345.
La situación de estos componentes en el sistema está en primer lugar en el CER móvil del prototipo, este será el encargado de detectar la variación de la distancia que sufre la cavidad en cada iteración, en uno de los carros del sistema de instrumentación se colocará el otro acelerómetro, en este caso estará fijo y servirá de referencia. El cálculo se realiza restando a los datos obtenidos por el acelerómetro fijo los valores del móvil. Cabe destacar que estos componentes miden aceleraciones, por lo que para hallar el desplazamiento se tendrá que integrar dos veces.

Desafortunadamente, los acelerómetros presentan una variación importante en sus resultados debido al ruido blanco, se trata de una señal que se produce aleatoriamente sin seguir ningún patrón y que añade valores de tensión adicionales. Estas perturbaciones están producidas por los componentes que conforman la placa.

Uno de los métodos más exactos para la eliminación de esta perturbación son los filtros Kalman, se trata de un algoritmo que sirve para poder identificar el estado oculto (no medible) de un sistema dinámico lineal y que está sometido a ruido blanco aditivo.
Para la implementación de este filtro es necesario tener una ecuación que caracterice el sistema, esto se debe a la necesidad de conocer el comportamiento lineal del mismo. Para ello se necesitará saber la presión del sistema en cada iteración, el valor ofrecido por los acelerómetros y conocer la distancia recorrida exacta por el sistema en cada momento.

La programación de estos componentes no se va a desarrollar en el presente trabajo debido a la existencia previa de ello. En el Anexo IV se puede observar la inclusión en el código de la configuración de los acelerómetros.

### *Regulador de tensión*

Anteriormente se ha comentado el deseo de establecer una tensión estándar para todos los elementos del sistema fijada en +5 V. Este hecho ha sido imposible de realizar debido a que dos de los componentes tienen una alimentación de+3,3 V, estos son los acelerómetros y los módulos de radiofrecuencia.
Para solucionar este hecho la opción escogida es la introducción de un regulador de tensión de +5 V a +3,3 V. Uno de los factores a tener en cuenta en la elección del modelo es la corriente que este es capaz de ofrecer, sabiendo que un acelerómetro consume 140 μA y el módulo de radiofrecuencia cuando está en modo emisor es de 115 mA, el regulador de tensión debe ofrecer al menos 140mA de corriente. El primer modelo seleccionado no ofrecía esta corriente por lo que fue eliminado a pesar de su reducido tamaño.

El modelo final escogido es el LP2985-33DBVT de la marca Texas Instruments, se trata de un modelo con un tamaño mayor pero capaz de alimentar a todos los componentes presentes en el sistema ya que la corriente que es capaz de ofrecer es de 150 mA.
Su conexión es sencilla, tan solo hay que conectarle la alimentación y masa para su funcionamiento, además posee un pin que se trata de la salida de +3,3 V (se puede observar en el datasheet adjuntado en el anexo I).
Para una correcta conexión del regulador este tiene que estar conectado a dos condensadores, uno en la entrada de la alimentación y otro a la salida. Esto permite la eliminación de ruido de la corriente eléctrica. El condensador a la entrada será de 470 μF y el de la salida de 47 μF, en el apartado de diseño de placas se puede observar la conexión del mismo en la Figura 33.

### *Control del sistema*

En la actualidad el uso de placas prediseñadas que incorporan un microcontrolador y un entorno de desarrollo propio está muy extendido en el mercado de la electrónica. Existen muchas compañías de hardware libre que ofrecen este tipo de producto, como ejemplo Arduino o Raspberry Pi. Ambas poseen una gran variedad de complementos para poder desarrollar una amplia selección de pequeños proyectos que ayuden al aprendizaje de las nuevas tecnologías. El hecho de que todo el hardware necesario para el desarrollo de estos proyectos sea muy económico, añadido a un software libre, hace que todo el mundo tenga a su alcance la opción de hacer de la electrónica una afición.

Para el desarrollo de este proyecto es necesario un microcontrolador que permita obtener los valores ofrecidos por los sensores y al mismo tiempo permitiera el procesado de los mismos. La mejor opción es un Arduino debido a la sencillez de su programación y a la gran variedad de módulos complementarios que estas placas presentan. Dentro de esta compañía existen muchos modelos dependiendo del número de puertos necesarios y de la capacidad de operación del microcontrolador insertado. En este caso se seleccionó el Arduino Leonardo debido a la capacidad que poseía el microcontrolador de realizar cálculos en coma flotante.
El problema de este modelo era el tamaño para introducirlo dentro del sistema que va acoplado al endoscopio, por lo que se va a utilizar como el soporte para el módulo de radiofrecuencia que reciba en el exterior los datos emitidos por el módulo interno. Además será el encargado de realizar los cálculos que permitan depurar los valores obtenidos.
En el caso del Arduino que va interno en el sistema se necesita un modelo, que manteniendo los puertos necesarios para la conexión de todos los sensores, tenga un tamaño lo más reducido posible. Con estas características se ha escogido el Arduino pro mini, este tiene 12 pines digitales, en los cuales vienen integrados los pines de ICSP y6 pines analógicos, en los cuales están incorporados los dos pines necesarios para el bus serie de datos ($I^2C$), necesarios para los acelerómetros.

El entorno necesario para la programación de estas placas es el software de Arduino, este está disponible para descargar de forma gratuita en la página del mismo. El lenguaje de programación es C añadiendo algún comando para la configuración de los puertos de las placas o de la inicialización de las comunicaciones, todo ello viene explicado detenidamente y con ejemplos en su web, además de tener un foro con una gran variedad de líneas abierta como ayuda a problemas que han surgido a los usuarios.
En el Anexo V de la presente memoria se adjunta un manual de usuario como guía de iniciación al software de Arduino.

Apoyándonos en el manual anterior, como se puede observar en el apartado de carga del programa, una vez se tiene seleccionado el tipo de placa, el puerto y el modo de programar, solamente dándole al botón de cargar el documento se graba en el chip de la placa.

Esto sucede si la placa lleva interna el bootloader que le permite una carga directa desde el programa, la mayoría de las placas llevan esta opción, como es el caso del Arduino Leonardo, mientras que en el caso de las placas de menor tamaño, como es el Arduino pro mini, no incorporan esta opción por lo que es necesario el uso de una placa extra para su programación. Para ello se usará un Arduino Uno, se trata de una placa muy parecida a la Leonardo diferenciándose únicamente en la colocación de los puertos y en la potencia del microcontrolador.

Para cargar el programa al Arduino pro mini los pasos a seguir son los siguientes:

1. Seleccionar la placa Arduino Uno, el puerto donde está conectado y el programador "AVRISP mkll" en la barra de herramientas.
2. Cargar en el Uno el programa Arduino ISP, este programa se encuentra en la pestaña de programas ejemplo del software.
3. Conectar los dos Arduinos como se observa en la Figura 24.
4. Seleccionar en la barra de herramientas de nuevo el tipo de placa, en este caso Arduino pro mini 5V, el puerto y el programador en este caso el "Arduino as ISP".
5. Una vez se tenga el programa a grabar, se selecciona en la barra te tareas la opción programa y se selecciona subir usando programador.

Estos serían los pasos necesarios para la carga del Arduino pro mini.

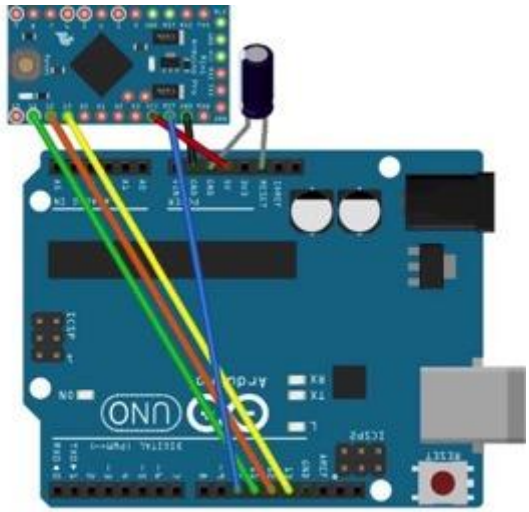


**Figura 24 - Conexión Arduino pro mini y Arduino Uno**

Para el control de los sensores se han realizado dos programas, el que irá con el pro mini en el sistema, este se encargará de la toma de datos y del envío de los mismos, y el otro será el Leonardo colocado en el exterior que recibirá los paquetes y se encargará del procesado de los mismos.
En el caso del primero se trata de un programa sencillo ya que no requiere ningún cálculo de alto nivel.

La parte inicial del programa se trata de la configuración del módulo de radiofrecuencia, explicado en el apartado anterior, seguido en el bucle de repetición por comandos del tipo:

>>cadena[0]=(5.0*analogRead(sensor1)/1023.0);

Donde cadena es la variable declarada para el almacenamiento de los valores obtenidos, en este caso la variable del sensor1 se va a guardar en la posición 0 de la cadena. La fórmula que se realiza es la que convierte los valores digitales detectados por el puerto analógico a valores analógicos, una entrada analógica proporciona una medición codificada en forma de un valor digital con un número N de bits, en este caso el máximo valor de esos bits son 1023 que equivalen a 5 V, por lo que se hace una regla de tres para hallar el valor medido. En el caso de los demás sensores sería del mismo modo cambiando el puerto de lectura. Cabe destacar que la variable sensor1 ha sido declarada previamente asociándola a una de las entradas analógicas de la placa. En el código también se ha introducido la programación de los acelerómetros proporcionada. Para la completa visualización del programa este se adjunta en el Anexo IV de la presente memoria.

En el caso del programa externo o receptor se ha realizado en primer lugar la configuración del módulo de radiofrecuencia receptor comentado en el apartado anterior. En este caso sí que ha sido necesaria la implementación de más cálculos para la obtención de los resultados reales, esto se debe a las sensibilidades presentes anteriormente expuestas y a la posible variación de temperatura.

El primer paso que se realiza es la declaración de todas las variables que van a ser necesarias para la programación, en este caso van a ser todas de tipo coma flotante, float, debido a la existencia de decimales en los resultados obtenidos. Seguido de la declaración de variables constantes como la sensibilidad de los sensores de presión y los de la fuente de corriente, la corriente suministrada por las mismas y por último la ganancia de los amplificadores existentes en las placas, estos se explican en el apartado siguiente diseño e implementación de placas. Como en el caso anterior el código está adjuntado en el Anexo IV de la presente memoria.

Un detalle importante a tener en cuenta se trata de las unidades de estas constantes, ya que una operación con distintas unidades producirá un resultado erróneo. En el caso de la sensibilidad de los sensores de presión las unidades escogidas son μA/ºC, en la sensibilidad de las fuentes de corriente son mV/kPa*μA, la corriente de las mismas en μA y por último la ganancia de los amplificadores en V/V. Como se puede observar en el caso de las unidades de tensión estas son diferentes en la sensibilidad y en la ganancia de los amplificadores, por ello se ha de tener especial cuidado en las operaciones realizadas posteriormente.

Como anteriormente se ha mencionado los resultados van a llegar en una cadena, para poder operar con ellos se puede llamar a la cadena seleccionando la posición adoptada por el valor deseado dentro de la misma, pero en este caso se va a usar una variable distinta para evitar confusiones. Estas variables son las mencionadas al inicio de la página, como ejemplo se puede observar el código necesario para almacenar en la variable s0 el primer valor de la cadena:

```
>> s0=valor[0];
```

En este caso la variable ha sido declarada con anterioridad como *float s0* y en este momento es usada para almacenar el valor existente en la cadena entrante en la posición 0, cabe recordar que en las cadenas el primer valor siempre es 0, es decir si se necesita una cadena de longitud 8 bits se tendrá que definir *float cadena[7]* ya que el número 0 ocupará un lugar en la misma.
Una vez se han almacenado los diferentes valores en las variables declaradas con anterioridad se procede a la realización de los cálculos necesarios para la obtención de las presiones en kPa.
El código empleado para el primer sensor es el siguiente:

```
>>float SP0 ()
{
  float CorrienteReal0=I0+(SensibilidadFC0*(temp-25.0));
  float Vdifs0=(s0/GananciaS0)*1000.0;
  float Sensibilidadfinal0=SensibilidadS0*CorrienteReal0
  float Presion0=Vdifs0/Sensibilidadfinal0
  return Presion0;
}
```

El primer paso para la programación de esta parte del código es la declaración de una función que contenga los cálculos y devuelva el valor de interés, para ello se declara *float SP0()*.
La primera línea devuelve el valor, en la variable CorrienteReal0, de la corriente real que la fuente está ofreciendo en cada iteración dependiendo de la temperatura ambiente, el cálculo es sencillo ya que a la corriente medida a 25 ºC se le suma o resta la variación de corriente por cada grado, esto es posible gracias a la sensibilidad de la fuente obtenida, como esta tiene unas unidades de μA/ºC si le multiplicamos la variación de temperatura existente, para ello se le resta a la temperatura actual la temperatura de 25 ºC, dará como resultado los μA que han aumentado o disminuido.

Una vez se ha obtenido la corriente real que la fuente ofrece se continua con el cálculo de la tensión diferencial que está ofreciendo realmente el sensor de presión, esto se debe a que la tensión es amplificada para que el Arduino la detecte, aunque se ha calculado una ganancia de 100 V/V, como los componentes no son ideales, esta puede variar por lo que se tiene que dar un paso atrás, y sabiendo la ganancia real del amplificador que ha sido medida con anterioridad, se obtiene el valor de la tensión diferencial ofrecida por el sensor de presión antes de la amplificación. Como este valor está en voltios se multiplica por 1000 para pasar las unidades a mV, que es una de las unidades en las que se encuentra su sensibilidad como anteriormente se ha mencionado.

En este momento se necesita calcular la nueva sensibilidad del sensor de presión en función de la corriente calculada, esto se debe a que la sensibilidad obtenida depende de esta (mV/kPa*µA). La sensibilidad según la corriente actual será el producto entre la sensibilidad del sensor conocida por la corriente real aportada por la fuente. En este momento las unidades de la sensibilidad es de mV/kPa.
Una vez hallados la sensibilidad del sensor y la tensión diferencial ofrecida por el mismo se procede al cálculo de la presión medida en kPa. Esta operación es sencilla ya que solamente se tiene que dividir la tensión diferencial entre la sensibilidad.
Cuando se llame a la función *SP0()* desde cualquier punto del programa esta devolverá el valor de presión en kPa.

Para finalizar con este apartado, la única función que faltaría es la visualización de los resultados obtenidos. Para ello se utiliza el comando:

```
>> Serial.print(SP0(),6);
```

El cual muestra el resultado por el monitor serie del programa con una longitud de 6 decimales.
Este proceso se tendría que realizar con los tres sensores existentes en el sistema, modificando las variables de cada uno.

Tras la programación de los dos microcontroladores el control del sistema está completamente realizado, ya se puede visualizar los resultados transmitidos y se pueden almacenar para su posterior procesado.

En este caso se van a almacenar los datos con un programa que guarda en cualquier formato los valores mostrados por el monitor serie. El programa es el Clear Terminal, se trata de un software libre que te puedes descargar gratuitamente desde su página web.

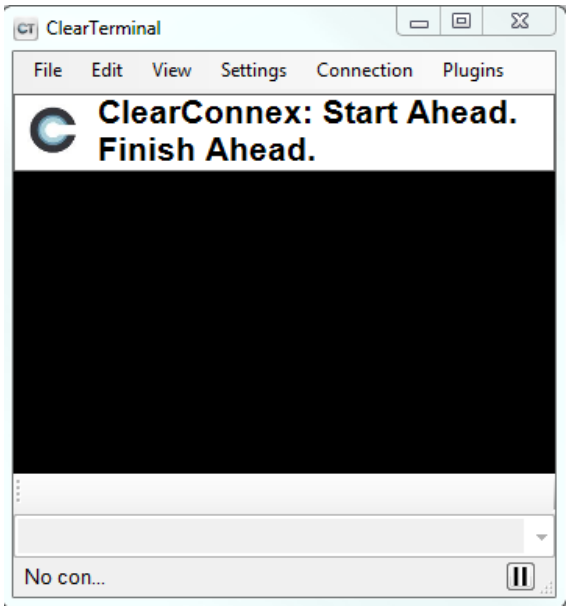



**Figura 25 - Pantalla inicial Clear Terminal**

Es un programa sencillo, en la Figura 25 se puede observar su interfaz de usuario, solamente hay que configurar el puerto por el cual se está transmitiendo la información (el mismo puerto en el que está conectado la placa de Arduino) y a la velocidad a la cual está configurado, que por defecto en todo el programa es de 9600 bauds.
Esta configuración se realiza mediante el botón en la barra de herramientas connection y new connection. Una vez se visualicen los datos por la pantalla, en el momento que se quieran guardar esto se realizará pulsando en la barra de herramientas File y posteriormente Save buffer to file, una vez se despliegue la pantalla de guardado se podrá escoger el formato de archivo en el cual se quiere guardar. En este caso el fichero es un .txt que permite una posterior apertura con Excel para el procesado de datos.

### 1.4.2 Diseño e implementación de placas

Para una optimización del espacio la mejor opción se trata de la fabricación de unas placas PCB (*Printed Circuit Board* o placa de circuito impreso)en las que colocar los componentes anteriormente comentados, se necesita que estén fijos y bien colocados. Para el diseño de cada una de las placas se ha de saber que componentes irán en cada una de ellas, para ello se van a diferenciar dos placas, una será la encargada de situar cerca de los sensores de presión la fuente de corriente y un amplificador operacional que permita amplificar los mV ofrecidos por el sensor en V que el Arduino sea capaz de leer. Por el otro lado la segunda placa que se va a diseñar albergará el regulador de tensión, el sensor de temperatura y un amplificador operacional para este, al igual que en el caso del sensor de presión el de temperatura ofrece su resultado en mV por lo que es necesario amplificarlo hasta V para ser detectado correctamente por el Arduino. A la primera se le ha puesto el nombre de placa del sensor de presión y a la segunda placa del regulador de tensión.

#### *Placa del sensor de presión*

Como anteriormente se ha mencionado, esta placa estará constituida por la fuente de corriente, los pines que conectarán con los cables del sensor de presión y un amplificador que aumente la tensión de mV a V.
El amplificador escogido es el modelo AD623ARZ de la marca Analog Devices, se trata de un amplificador single supply, se puede alimentar a una tensión y masa, no necesita una alimentación simétrica, este hecho permite mantener la tensión estándar estipulada y alimentar el amplificador entre +5 V y GND.
Debido a que el sensor de presión ofrece en su salida diferencial unas decenas de mV el amplificador tendrá que amplificarlo con una ganancia de 100 V/V, de este modo el Arduino detectará voltios y así se aumentará la precisión en la lectura de los datos. Como dato de interés el puerto analógico del Arduino puede leer como máximo 1023 bits equivalentes a 5 V, esto hace que cada bit tenga una sensibilidad de 4,88 mV.

Debido a la pequeña sensibilidad de los puertos analógicos del Arduino se necesita el aumento de la tensión ya que si no fuese así los datos tendrían entre ellos más de 5 mV de diferencia y eso no permite una buena precisión en la medida.
El diseño del amplificador es sencillo, solamente se necesita calcular la resistencia de carga (Rg) que permite la variación de la ganancia. Para ello se ha realizado la siguiente operación. Al igual que en casos anteriores el datasheet del componente se encuentra adjuntado en el Anexo I de la presente memoria.
Para el cálculo de Rg el fabricante ofrece la siguiente ecuación:

$$R_G = \frac{100 \cdot 10^3}{(G-1)} = \frac{100 \cdot 10^3}{(100-1)} = 1010{,}10\ \Omega \approx 1k10\ \Omega$$

$$Donde\ G\ es\ la\ ganancia\ deseada$$

El valor necesario de resistencia de carga para ajustar la ganancia a la desea es de 1k10 Ω, este valor de resistencia no se encuentra en mercado con este valor nominal, por lo que será necesario la colocación de dos resistencias en serie que sumen este valor, para ello se ha escogido una resistencia de 1 kΩ con una tolerancia de ±1% y una de 10 Ω.
Al diseño del amplificador se ha añadido dos elementos más ajenos a este, el primero se trata de un condensador de 0,1 µF colocado en la patilla de alimentación que elimina el posible ruido provocado por la misma. Además de un filtro paso bajo que elimina las frecuencias superiores a 6,45 Hz que puedan interferir en el proceso, este filtro es de primer orden formado por una resistencia de 330 kΩ y un condensador de 470 nF, en la Figura 26 se puede observar su conexión.
Una vez se ha diseñado esta última fase se procede al diseño de la placa, para ello se va a usar el programa Proteus para la realización del esquema y posteriormente se usará el programa Ares para el diseño de la placa.

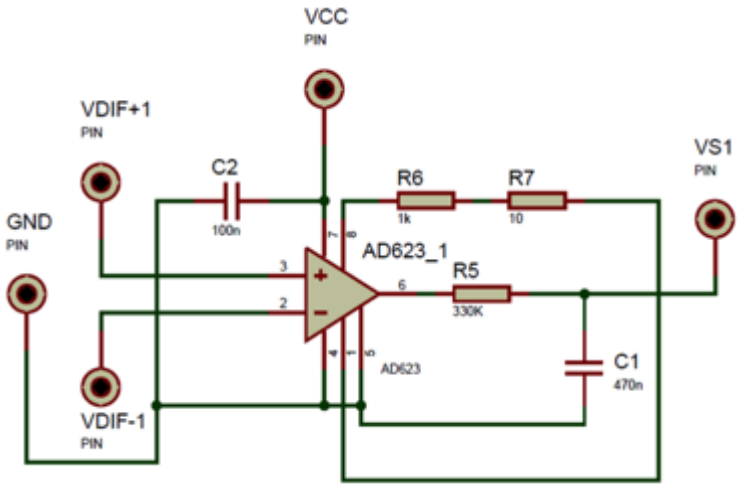



**Figura 26 - Circuito del amplificador AD623ARZ**

Tras realizar los esquemas, estos formarían una placa de un tamaño superior al deseado por lo que se ha procedido a realizar la placa a doble cara y así aprovechar el espacio por ambos lados, En la Figura 27 se puede observar la diferencia de tamaño entre la placa de doble cara a la izquierda y las otras dos opciones diseñadas solo a una cara a la derecha.

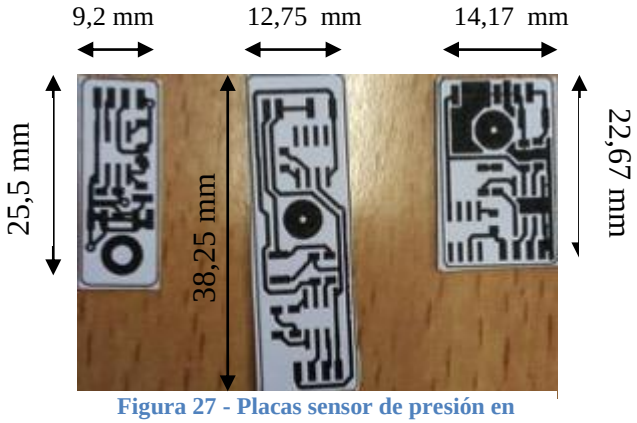



**Figura 27 - Placas sensor de presión en diferentes configuraciones**

En la Figura 26 se puede observar el esquema de la parte superior de la placa compuesta por el amplificador operacional diseñado anteriormente. En el caso de la parte inferior de la placa, esta está compuesta por la fuente de corriente de corriente, se puede observar en el Figura 28.

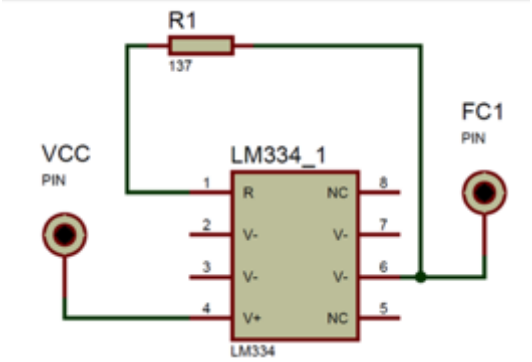



**Figura 28 - Esquema conexión Fuente de corriente**

Los planos de las placas realizadas en Ares se pueden encontrar en el apartado *2. Planos* del presente proyecto. En estos se puede visualizar la colocación de los componentes y las pistas de unión entre las patillas.
En la Figura 29 se puede observar la colocación de los componentes y los pines en la parte inferior de la placa.

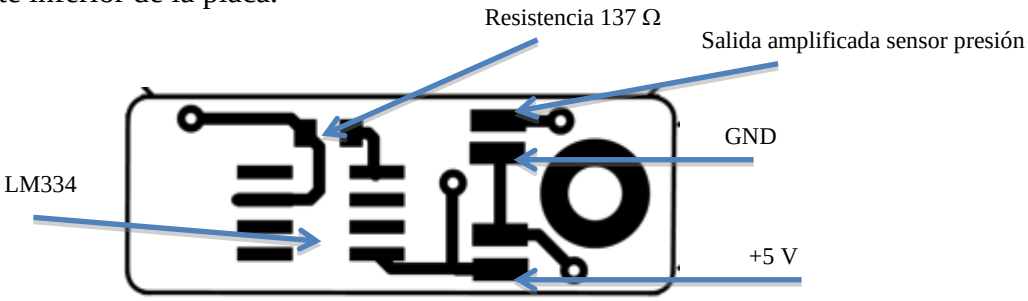



**Figura 29 - Parte inferior placa sensor de presión**

En la Figura 30 se puede observar la parte superior de la placa del sensor de presión.

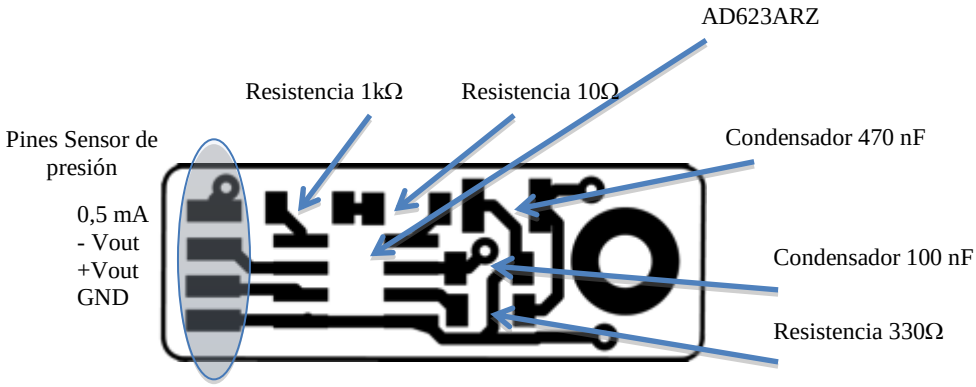



**Figura 30 - Vista superior placa sensor de presión**

La fabricación de las placas se comentará más adelante.

### *Placa del regulador del tensión*

Los últimos componentes que quedan por situar son; el regulador de tensión, el sensor de temperatura y su pertinente amplificador. En este caso el amplificador a utilizar es el modelo MAX4250EUK-T de la empresa Maxim, se trata de un amplificador de single supply, al igual que en el caso anterior, con un tamaño muy reducido y rail to rail que significa que es capaz de ofrecer a la salida una tensión muy próxima a la de alimentación (datasheet incluido en el Anexo I de la presente memoria).
El amplificador tiene una ganancia de 10 V/V, esto se debe a que la tensión ofrecida por el sensor de temperatura es de centenas de mV, para aumentar este valor a V solamente es necesaria una ganancia pequeña. En este caso el fabricante no ofrece ninguna ecuación para calcular la ganancia, por lo que se va a diseñar un amplificador operacional no inversor que mantenga el signo de la tensión de entrada en la de salida. El esquema de esta configuración se puede observar en la Figura 31.

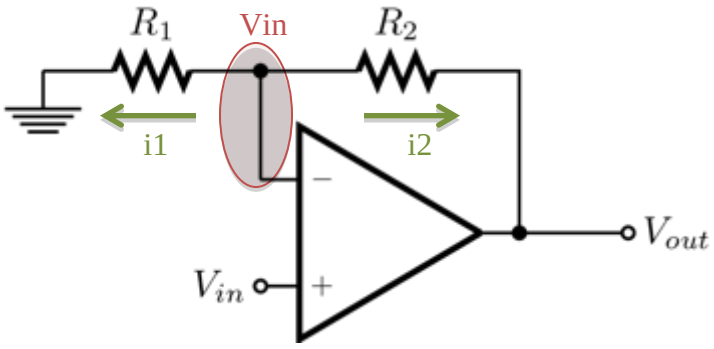



**Figura 31 - Amplificador operacional en configuración no inversor**

Como se puede observar en la Figura 31, al tratarse de un amplificador con realimentación negativa, la tensión existente en la entrada no inversora es la misma que la existente en la entrada inversora por ello se ha especificado en la imagen con el círculo rojo, las dos flechas indican las dos corrientes existente en este circuito, la primera i1 que circula desde la entrada no inversora hasta masa y la segunda i2 que realiza su recorrido por el camino contrario, de la entrada no inversora a la salida del amplificador.
Aplicando la Ley de las corrientes de Kirchhoff al nudo situado en la entrada inversora la ecuación quedaría de la siguiente manera:

$$i_1 + i_2 = 0 \rightarrow \frac{V_{in} - 0}{R_1} + \frac{V_{in} - V_{out}}{R_2} = 0 \rightarrow G = \frac{V_{out}}{V_{in}} = 1 + \frac{R_2}{R_1}$$

Las corrientes se suman ya que ambas salen del mismo nodo, en este caso la entrada inversora.
Una vez obtenida la ecuación que determina la ganancia (G) solamente se necesita encontrar los valores deseados para la ganancia deseada. La ganancia a fijar es de 10 V/V, para ello se fija el valor de una de las resistencias y se calcula el valor de la otra, en este caso se fija el valor de R2 en 100 Ω obteniéndose así un valor de R1 de 900 Ω.
Debido a que no existe ninguna resistencia de montaje superficial con un valor nominal de 900 Ω se va a realizar un montaje con dos resistencias de 1k8 Ω en paralelo que ofrecen el valor buscado.
Al igual que en el caso anterior se coloca un condensador en la alimentación de 100 nF para eliminar cualquier posible ruido debido a esta. El esquema del amplificador junto con el sensor de temperatura se puede observar en la Figura 32.

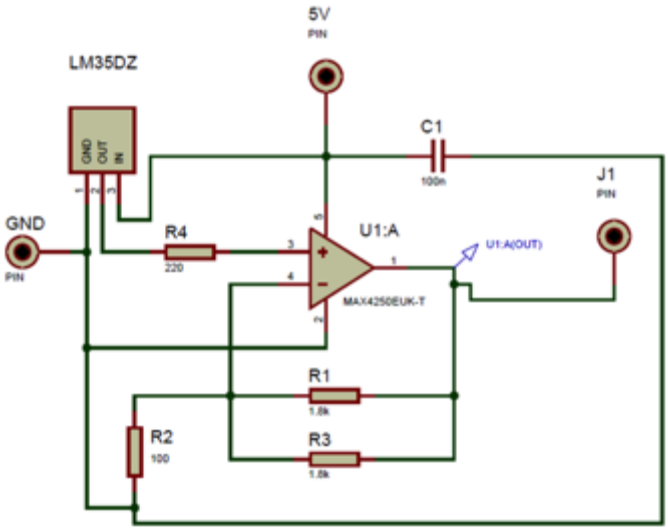



**Figura 32 - Circuito Sensor de temperatura LM35DZ y amplificador operacional MAX4250EUK-T**

En el diseño de esta placa sucede lo mismo que en el caso anterior, el tamaño de esta sería demasiado grande si se colocasen todos los componentes en la misma cara, por lo que se va a diseñar la placa a doble cara, en la parte superior se encuentra la Figura 32 y en la cara posterior se encuentra el regulador de tensión, se puede observar su conexión en la Figura 33.

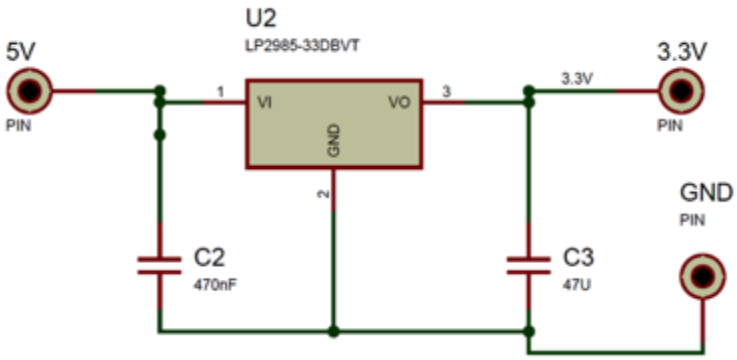



Figura 33 - Esquema conexión Regulador de tensión LP2985-33DBVT

Los esquemas de las placas se pueden observar en el apartado *2. Planos* del presente proyecto. En la Figura 34 se puede observar la colocación de los componentes y de los pines de la parte superior de la placa.

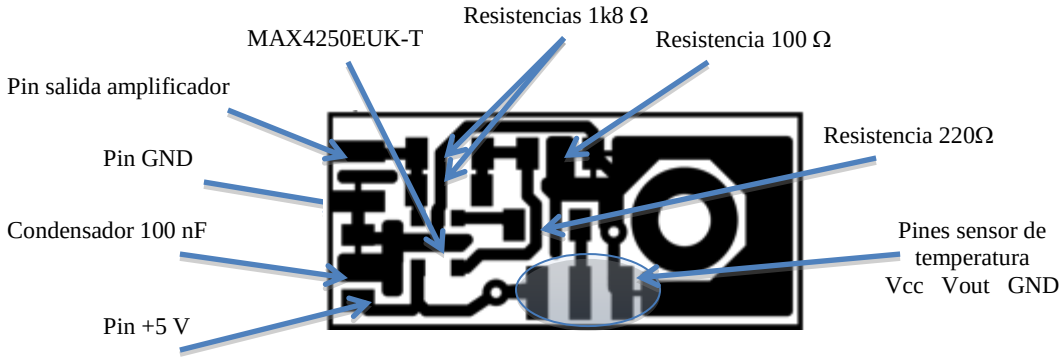



Figura 34 - Cara superior placa de pistas del regulador de tensión

La parte inferior de la placa se puede observar en la Figura 35.

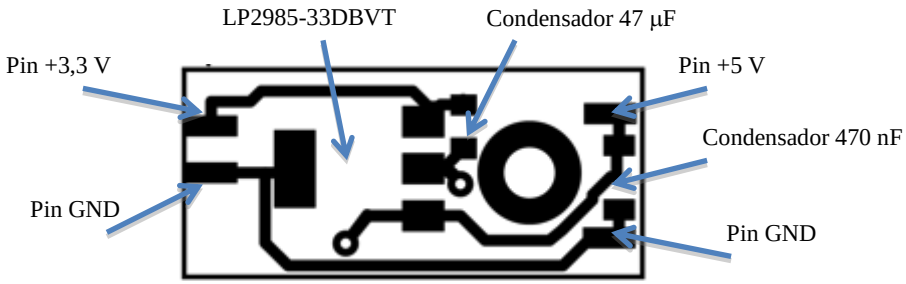



Figura 35 - Cara inferior placa de pistas del regulador de tensión

### *Fabricación de placas*

La fabricación de placas PCB es un procedimiento delicado pero sencillo. En primer lugar los materiales necesarios para este proceso son los siguientes:

1. Placa de baquelita o fibra de vidrio con emulsión positiva a doble cara.
2. Cutter.
3. Lija.
4. Impresora láser con tóner negro.
5. Transparencias para la impresora.
6. Insoladora.
7. 2 Cubetas de plástico.
8. Medidor de líquidos.
9. Agua.
10. Sosa caustica.
11. Ácido clorhídrico.
12. Peróxido de hidrógeno de 110 volúmenes.
13. Alcohol.
14. Papel de cocina.
15. Taladro con brocas de diferente tamaño.

El primer paso es el preparado de la placa. Para ello se cortará el tamaño de la placa deseada, en este caso para una mayor rapidez, debido a que se van a fabricar 12 placas del sensor de presión y 9 del regulador de tensión, se colocarán todas las placas iguales en un mismo bloque y se cortarán una vez se termine el proceso.

Seguido de la preparación del fotolito, esto es la transparencia con el esquema eléctrico impreso que permite imprimir las zonas necesarias y dejar el resto transparente, de este modo la luz UV puede traspasarlo. Recordar que para esta impresión se necesita una lamina trasparente adecuada para la impresión y una impresora láser con tóner negro.
En este caso se tendrá que imprimir las dos partes de la placa, teniendo especial cuidado con la parte inferior, ya que se debe seleccionar la impresión desde una vista inferior para su correcta colocación.

Una vez se han impreso los fotolitos y se han colocado correctamente las dos partes a imprimir una contra la otra, se coloca la placa entre las dos láminas y se fijan con cinta transparente para evitar posibles movimientos.
Cuando se termine esta parte se comienza el proceso de insolación. Para ello se necesitará una insoladora de doble cara, ya que si no se tendría que repetir el proceso dos veces, una por cada cara. En este caso debido a que el laboratorio posee una insoladora comercial de doble cara se realizo allí, pero también existen muchos tutoriales que te permite realizar la tuya propia. En este caso la duración del proceso fue de 200 segundos.
Tras finalizar el tiempo se extraen las placas y se comienza el paso de revelado. Este se compone de un 1 litro de agua templada, con agua fría no sería posible, y 12 gramos de sosa caustica. Una vez disuelta toda la sosa se introducen las placas, el tiempo de revelado oscila entre 30 y 60 segundos dependiendo del tipo de placa y de la temperatura del agua. En el momento que se observen todas las pistas se sacarán de la solución.

Ahora es el momento del ataque, este se encarga de corroer el cobre que está desprotegido. Para realizar la solución de este paso se necesitará 50 mL de agua, 25 mL de ácido clorhídrico y 25 mL de dióxido de hidrógeno. Todos los productos que se están empleando se tratan de productos muy corrosivos por lo que se tendrá que llevar un especial cuidado.
Con la mezcla lista se introduce la placa, para acelerar el proceso de ataque se puede ir moviendo el recipiente creando olas que permitan un leve movimiento a la placa. Una vez se desprenda todo el cobre y queden las vistas bien definidas se sacaran las placas y se enjuagarán con agua, de este modo se frenará el proceso de ataque.

Para finalizar se limpiarán las placas con alcohol para eliminar cualquier resto. El resultado de las placas al final del proceso se puede observar en la Figura 36.

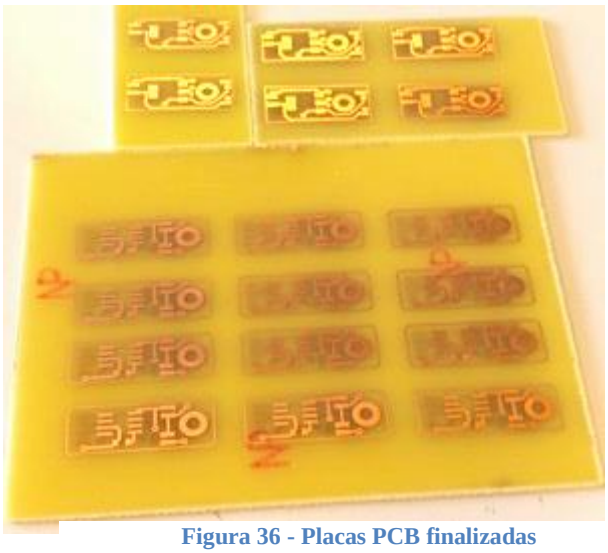


**Figura 36 - Placas PCB finalizadas**

En la parte superior de la Figura 36 se encuentran las placas del regulador de tensión y la parte inferior las placas del sensor de presión.

Una vez terminado el proceso de fabricación se comienza con la preparación de las placas para su soldadura.

El primer paso consiste en taladrar todos los puntos necesarios, en este caso serán los orificios que permitan la conexión de los componentes entre las dos caras de la placa. Estos vienen marcados con círculos como se puede observar en los Esquemas del apartado anterior, además de los taladros para la conexión entre caras también se va a realizar un taladro de un diámetro mayor para la fijación de la placa.

Tras realizar todos los taladros se prosigue con el cortado y lijado de las placas, para ellos será necesaria una cuchilla que realice un corte limpio y evite dañar las pistas. Una vez cortada se lijan los bordes hasta dejar la placa del menor tamaño posible y con las esquinas redondeadas.

En este momento las placas ya están listas para ser soldadas.

### 1.4.3 Montaje del sistema

El montaje del sistema es sencillo, se trata de la unión de las diferentes partes que se han ido fabricando o programando, estos son; los sensores de presión que se han encapsulado, las placas diseñadas en el apartado anterior, el módulo de radiofrecuencia, los acelerómetros y el Arduino pro mini.

Antes de comenzar con la unión de las diferentes partes del sistema, cabe especificar sobre qué se va a colocar y su posición exacta dentro del prototipo Endoworm 3.0.
El sistema irá colocado sobre unos carros, estos se tratan de unos cilindros huecos por el centro por donde se introduce el endoscopio. En la parte superior se fijarán los diferentes elementos mediante tornillos de silicona y bandas elásticas del mismo material.
En la Figura 37 se puede observar los tornillos de silicona, se tratan de cilindros de silicona pegados a cuadrados del mismo material que se introducirá por el taladro realizado en la placa y se pegará al carro.

Figura 37 - Tornillos de silicona

El esquema de situación de los diferentes componentes se puede observar en la Figura 38.

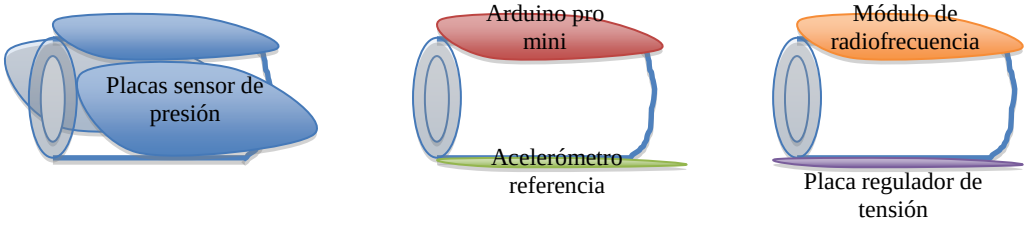



Figura 38 - Esquema de posición de los componentes del sistema

El sistema de instrumentación irá colocado al final del prototipo Endoworm 3.0, pero falta por situar un acelerómetro debido a que este va situado en el carro móvil en el inicio del endoscopio, esto hará que con la medida estática que ofrezca el acelerómetro de referencia al final menos la medida del acelerómetro móvil en cabecera se establezca la distancia recorrida en cada iteración.

La unión entre las diferentes partes se realizará con cable plano multifilar no apantallado de 10 vías. Para ello se empleará estaño de un diámetro de 0,5 mm para mayor precisión. La punta empleada en el soldador tiene un diámetro de 0,5 mm.

El primer paso consiste en soldar los cables de alimentación a los pines existentes en la placa del regulador de tensión, estos se pueden ver en la parte derecha de la placa en la Figura 36 del apartado anterior, cabe destacar que la circunferencia que se realiza a las placas para su fijación siempre quedará hacia los cables de alimentación, es decir hacia el final del sistema. Esta placa se colocará con la cara del regulador de tensión pegada al carro, hacia arriba, y el sensor de temperatura que irá fuera de la placa se doblará y se colocará hacia arriba rodeando al carro, se ha colocado un tubo termoretráctil en las patillas para evitar contactos indeseados.

En la parte superior del carro se colocará el módulo de radiofrecuencia con la antena hacia el final para tenerla lo más cerca del exterior. En la Figura 39 se puede observar la colocación de estos componentes en el primer carro.

**Figura 39 - Primer carro sistema con módulo radiofrecuencia y placa regulador de tensión**

Una vez soldados los cables de alimentación se comprobará el correcto funcionamiento de la placa, en caso afirmativo se continuará con la soldadura del módulo de radiofrecuencia. Para la soldadura de este componente habrá que llevar especial cuidado con los pines que lleva de serie el módulo, para su correcta eliminación la mejor forma es aplicándoles calor con la pistola de aire de caliente de la estación de soldadura e ir moviéndolos con unas pinzas poco a poco, cabe recordar no excederse con el tiempo de calentado ya que se podría quemar el módulo.

Una vez se han eliminado estos pines se soldarán los cables, se distinguen 3 grupos de cables; los 2 de alimentación que irán dirigidos a la placa del regulador de tensión, otros 2 cables que son los de comunicación con el microcontrolador que irán dirigidos al Arduino hacia los pines digitales 7 y 8 y por último 3 cables que corresponde con los necesarios para el envío de información (MISO, MOSI, CLK) que irán dirigidos a los pines digitales 11, 12 y 13 del Arduino.

En el siguiente carro, como se puede observar en la Figura 38, se colocará el Arduino pro mini en su parte superior y uno de los acelerómetros en su parte inferior.
En estos momentos los cables a soldar en el Arduino son los cables de alimentación procedentes de la placa del regulador de tensión, y además el cable desde el pin de salida del amplificador de temperatura que se conectará al pin analógico A4. También se soldarán los 5 cables procedentes del módulo de radiofrecuencia anteriormente mencionado.

En el caso del acelerómetro este posee, como se puede observar en la Figura 40, 8 pines de los cuales solamente se van a conectar 6, los pines Vcc, Cs y SD0 irán conectados a +5 V, el pin GND a masa y los pines SDA y SCL a los pines correspondientes en el Arduino (en el caso del Arduino pro mini se tratan de los pines analógicos A6 y A7).


**Figura 40 - Acelerómetro ADXL345**

Se continua conectando el sistema a la alimentación para comprobar que todas las soldaduras se han realizado correctamente verificando que se recibe los datos del acelerómetro y del sensor de temperatura con el módulo de radiofrecuencia conectado en el exterior. En caso afirmativo se procede a la soldadura de las placas de los sensores de presión.
En todo momento la longitud de los cables se ha realizado a una distancia suficiente para una correcta colocación y cortándolos de medidas distintas según su posición.

En el caso del tercer carro, este se situará más próximo a las cavidades posible, estarán situadas las tres placas del sensor de presión, los pines necesarios a conectar aquí son por un lado los dos cables de alimentación, masa y tensión, y la salida del amplificador de la tensión diferencial medida en el sensor. Estos pines serán los situados la parte final de la placa que se dirigirán al inicio de la alimentación y en el caso de la salida del amplificador a la entrada analógica A0, A1 y A2 del Arduino.
Por el otro lado de la placa saldrán 4 cables los cuales son la corriente de alimentación del sensor, masa y los dos pines de la tensión diferencia (+Vout y –Vout).
Cada sensor estará conectado a una cavidad distinta, por lo que cabe llevar un especial cuidado con que placa se conecta a que puerto del Arduino, la relación de conexión es la siguiente:

Sensor del CEA → pin A0
Sensor del CER móvil → pin A1
Sensor del CER fijo → pin A2

De igual modo se tendrá que anotar el número de cada sensor y en qué número de placa está conectado. Esto se debe a las sensibilidades de los componentes, se ha de tener en cuenta la sensibilidad del sensor de presión y la sensibilidad de la fuente de corriente.
El esquema de conexión debe quedar tal como se muestra en la Figura 41. Cabe destacar que la parte del sensor de presión y de su placa debe estar triplicado, siendo su conexión en el Arduino en los pines mencionados anteriormente.

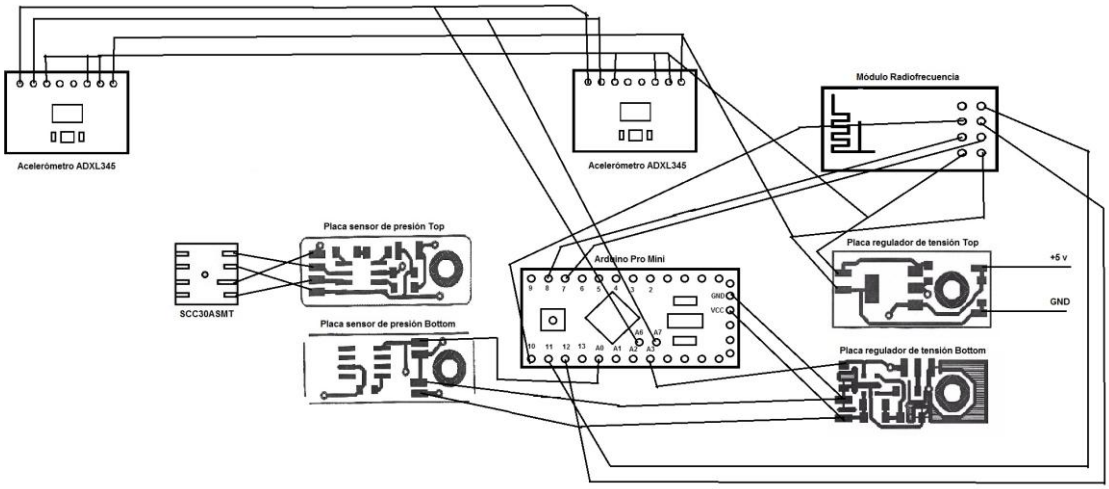



**Figura 41 - Esquema de conexión del sistema de monitorización completo**

Una vez se ha soldado todas las partes y se han colocado se comprueba de nueva su correcto funcionamiento. En caso afirmativo se finaliza soldando el acelerómetro móvil en la cabecera del sistema del mismo modo que el soldado al comienzo.
El resultado final del sistema soldado sin estar colocado sobre los carros se puede observar en la Figura 42.

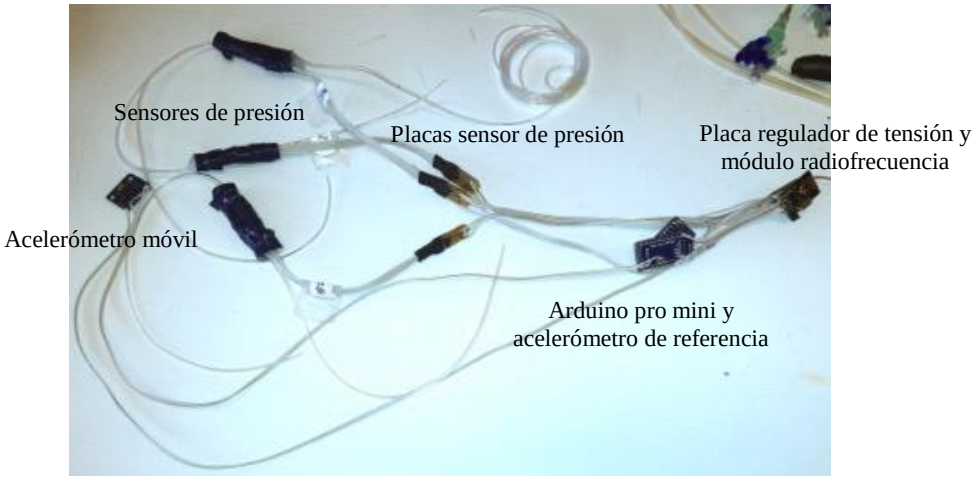



**Figura 42 - Sistema completo soldado**

Una vez montado sobre el endoscopio el sistema quedaría como se puede observar en la Figura 43.

**Figura 43 - Sistema de monitorización colocado sobre el Endoworm 3.0.**

### 1.4.4 Realización de ensayos y procesado de datos

Tras el montaje del sistema se comienza con la realización de los ensayos para comprobar el correcto funcionamiento del sistema y para la toma de datos del comportamiento del Endoworm 3.0.
Se van a realizar 3 ensayos por cada modo de velocidad ( slow, normal y fast) de la máquina.

Estos ensayos se van a realizar con el sistema de monitorización colocado sobre unos soportes que mantengan elevado el endoscopio y sin ningún tipo de rozamiento externo. Los datos obtenidos en cada toma serán almacenados en un documento .txt y posteriormente convertidos en un archivo Excel para su procesado.
Al mismo tiempo se grabará un video de la pantalla incluida en el hardware del Endoworm 3.0. para una posterior comparación de los valores mínimos y máximos medidos tanto por los sensores en cabecera cómo por los sensores del sistema de monitorización.
En el Anexo VI se incluyen todas las gráficas obtenidas tras el procesado de los datos. Se tratan de 3 gráficas por cada toma, la primera muestra la evolución de la presión en el CEA, la segunda en el CER móvil y la tercera en el CER fijo. En la Figura 44, Figura 45 y Figura 46 se muestra un ejemplo de las gráficas obtenidas en la primera toma a ritmo normal.

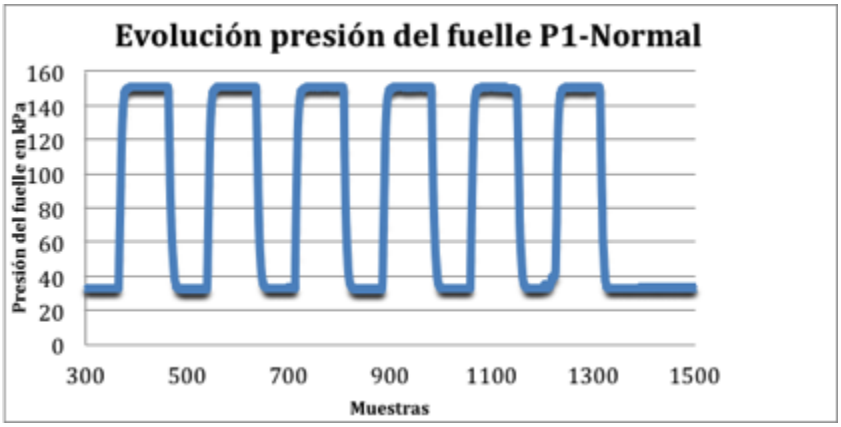



Figura 44 - Evolución de la presión en el CEA toma 1 a velocidad normal

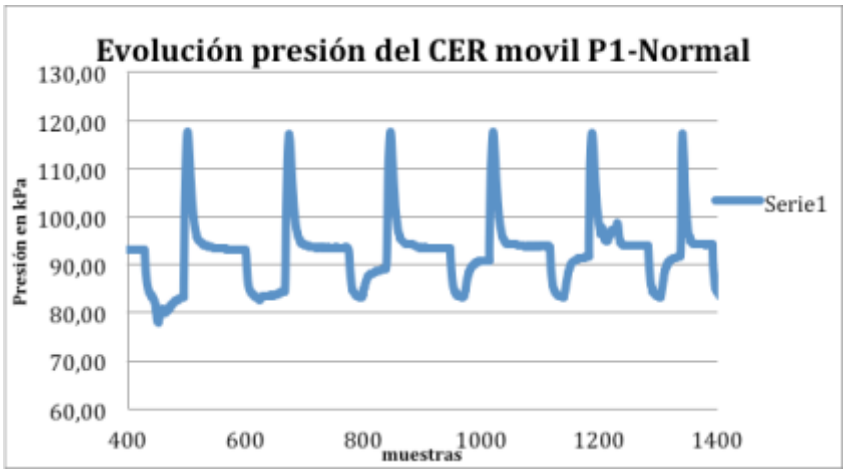



Figura 45 - Evolución de la presión en el CER móvil toma 1 a velocidad normal

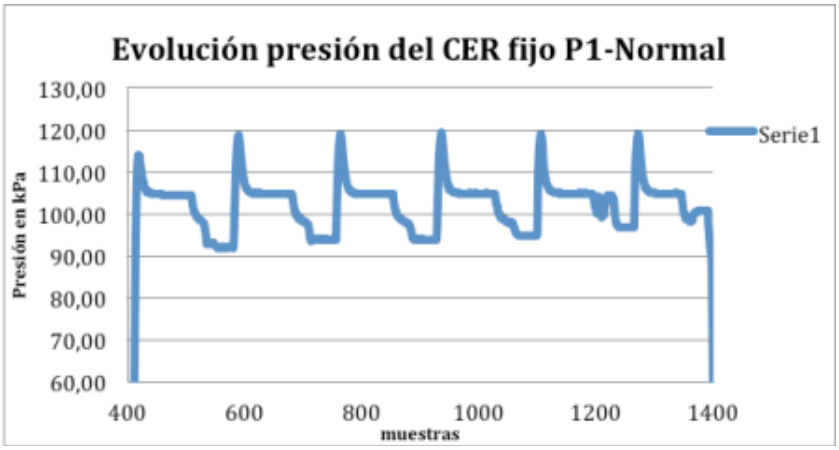



**Figura 46 - Evolución de la presión en el CER fijo toma 1 a velocidad normal**

Como se puede observar en la Figura 44 la presión en el CEA oscila entre 32 y 150 kPa, la presión mínima debería ser de 0 kPa ya que se trata del momento en el que se genera vacío para la completa extracción del aire, esto es debido a que las válvulas de llenado no tienen la suficiente fuerza para mantenerse cerradas y dejan escapar un poco de aire que es el que se muestra como mínimo en la gráfica. Este hecho ha dado lugar a una mejora en el hardware del prototipo Endoworm, modificando la conexión de las válvulas y cambiándolas por otras de mayor fuerza.

En el caso de la Figura 45 y la Figura 46 se puede observar un pico cuando se produce el aumento de la presión, esto se debe a que la silicona de los globos es viscoelástica y presenta una respuesta elástica que es instantánea mientras que la respuesta viscosa va con un poco de retardo. Una vez se compensan las dos respuestas estas se estabilizan en una presión de llenado.
En el caso de la Figura 45 esta se estabiliza a una presión de 84 kPa y cuando se produce el vacío en 84 kPa. En este caso sucede lo mismo que con el CEA, existen pérdidas en la válvula por lo que se introduce aire que provoca un aumento de la presión en la cavidad.
En la Figura 46 sucede lo mismo esta se estabiliza en el llenado en una presión de 105 kPa y en el vaciado en 94 kPa, De igual modo se comprueba que la válvula no está manteniendo estanca la cavidad y por tanto se está introduciendo una presión no deseada.

Cabe recordar que las presiones medidas se tratan de presiones absolutas, es decir llevan incluidas la presión atmosférica existente en ese momento.

### *Caracterización del Endoworm 3.0.*

Como se ha comentado en el apartado *1.3.1 Materiales empleados en el subapartado de acelerómetro*, estos presentan un ruido blanco que modifican el valor obtenido por el sensor. Para contrarrestar esta perturbación se emplean filtros Kalman que se tratan de un algoritmo que identifica el estado oculto del sistema dinámico lineal y que está sometido al ruido blanco aditivo.

Para la implementación de este filtro es necesario tener un ecuación que caracterice el sistema, esto se debe a la necesidad de conocer el comportamiento lineal del mismo. Para ello se necesitará saber la presión del sistema en cada iteración, el valor ofrecido por los acelerómetros y conocer la distancia recorrida exacta por el sistema en cada momento.
Esta distancia se había estado midiendo con los acelerómetros, pero como antes se ha mencionado estos poseen perturbaciones que modifican el valor de los mismos, por lo que se necesita un dato de referencia preciso para su comparación.

El sensor empleado para tal fin es un sensor de posición de rotación analógico, el modelo es el AMS22S de la marca Bourns (datasheet incluido en el Anexo I de la presente memoria). Se trata de un sensor que te ofrece una tensión según el ángulo de giro, en este caso no se desea conocer el ángulo de giro sino el desplazamiento lineal por lo que se va a transformar el sensor en un sistema de medición lineal.
Para esta transformación serán necesarios los siguientes elementos:

- Barra metálica que sirva de alargador del sensor.
- Conector para las dos barras.
- Rueda dentada.
- Barra dentada.
- Pieza de soporte del sensor.
- Pieza de enganche del barra dentada al endoscopio.

En primer lugar se comienza con el diseño de la pieza que será el soporte para el sensor de desplazamiento. El soporte diseñado se muestra en la Figura 47.

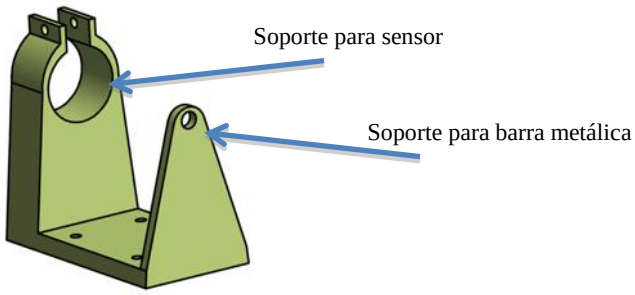



**Figura 47 - Diseño de soporte sensor de desplazamiento**

El plano de la pieza se encuentra incluido en el apartado *2. Planos* del presente proyecto.
La pieza se diseñó con el programa Inventor, en el Anexo V de la presente memoria se adjunta un manual de usuario para poder diseñar la pieza. Una vez diseñada esta se guardará en un formato .STL, este formato es el aceptado por la impresora 3D que se va a emplear.
El modelo de impresora es el Cube pro duo de la compañía 3Dsystems. Esta impresora posee un software especifico de la marca que adapta el diseño a la impresora, permite colocar la posición de la pieza en la cual se desea imprimir, escoger el material a emplear, el color del mismo y el espesor de cada. En este caso a menor espesor mayor será la precisión de la pieza.

Cabe destacar que el propio programa realiza partes secundarias en las piezas en caso de necesitar un apoyo en su impresión.

Tras la impresión de la pieza se insertará el sensor en el soporte orientado hacia el soporte de la barra. Después se colocará la barra metálica, atravesando su soporte, unida con el conector al extremo del sensor.
Para evitar posibles movimientos de la rueda dentada en la guía del barra dentada, se ha cortado de un aro de PVC laminado, también conocido por el nombre de su marca Forex, de diámetro exterior de 54,51 mm y con un diámetro interior de 48,68 mm, este aro se pegará a la rueda dentada en su parte más ancha, sirve de ampliación del espacio para evitar rozamientos con la guía. La guía se trata de una circunferencia de mayor tamaño a la de la rueda dentada y del mismo material que el aro anterior, será de un tamaño de 63 mm de diámetro con un agujero central de 3,5 mm de diámetro. Una vez recortado (véase Figura 48) se pega al aro anteriormente colocado. Se tendrán que hacer dos piezas de cada para colocar una a cada lado de la rueda dentada.

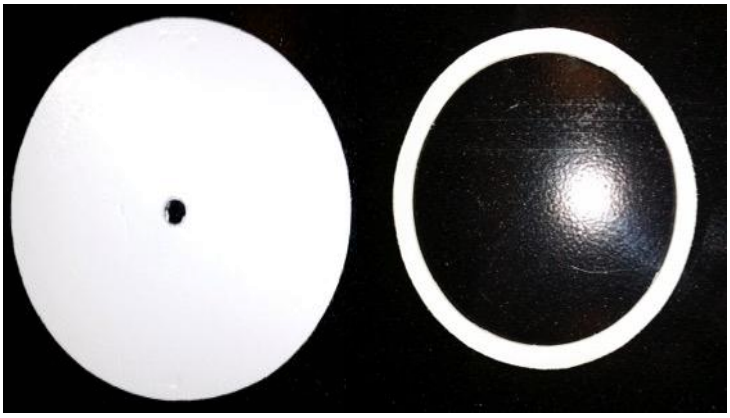


**Figura 48 - Piezas guía de la rueda dentada, a la izquierda guía y a la derecha aro de ampliación.**

Una vez pegadas las dos partes se coloca la rueda sobre la barra, el conjunto debe quedar como en la Figura 49.

**Figura 49 - Conjunto de sensor, soporte y rueda dentada modificada**

**Figura 50 - Horquilla de sujeción barra dentada**

Ahora es el turno del barra dentada, este tendrá que ir sujeto al carro del CER móvil para ello se tiene que diseñar otra pieza que cumpla con este propósito. El diseño escogido se muestra en la Figura 50. A esta pieza se le ha asignado el nombre de horquilla, por la función de sujeción que desempeña.
Como se puede observar la horquilla tiene una zona circular con una apertura en la parte superior para poder colocarlo sobre el carro, y posee 4 patas para el enganche del barra dentada. Al igual que la pieza anterior se procede al diseño con el programa Inventor y se continua con su impresión en 3D.

Una vez obtenida la pieza se procede al taladro de los orificios en el barra dentada para su colocación. Debido a que las patas de la horquilla están separadas entre sí desde el centro ellas una distancia de 10,13 mm esta será la distancia a la cual se situarán los taladros.
La barra dentada se sujetara por la parte central para equilibrar el peso del mismo y para que la horquilla ejerza la máxima fuerza posible sobre él. El sistema final se puede observar en la Figura 51.

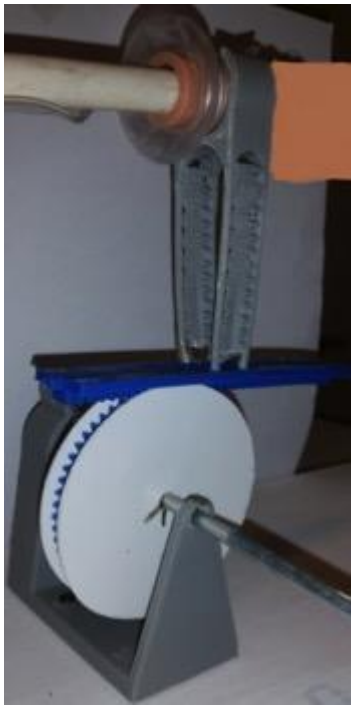


**Figura 51 - Montaje final sistema medición de distancia con el sensor de posición**

Como se ha mencionado el sensor es lineal pero se conoce su variación para grados pero no para milímetros, por ello se va a realizar una prueba de calibrado que permita obtener una sensibilidad en forma de mV/mm.
Para ello se ha impreso una regla con separación de milímetros que permita una mayor precisión. En primer lugar se coloca la regla impresa en una funda de plástico transparente, esto es para que sea más sencillo su desplazamiento, y se sitúa a la misma altura a la que la barra dentada encaja con la rueda. El sistema debe quedar tal como se muestra en la Figura 52.

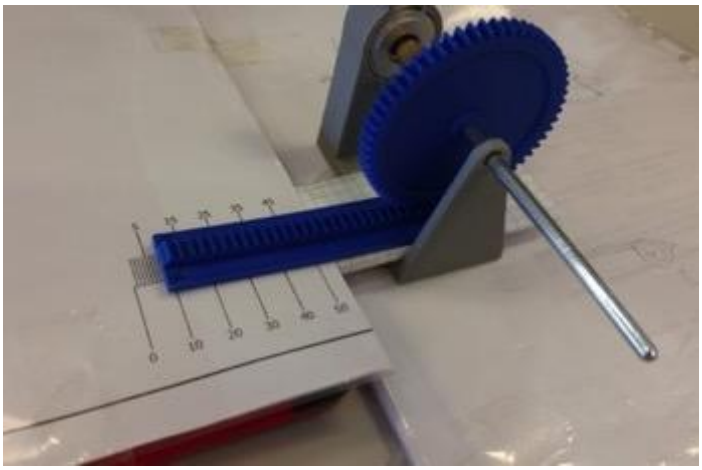


Figura 52 -Sistema de calibrado sensor de desplazamiento

Se conecta el sensor a la alimentación, este necesita una tensión de +5 V en su pata positiva y masa en la negativa, también tiene una tercera pata que se trata de la tensión de salida, esta oscilará entre 0,261 a 4,73 V. Como se trata de un sensor multivuelta habrá que tener especial cuidado la zona muerta existente entre los valores 0 y 0,261 V y 4,73 y 5 V ya que puede ofrecer valores falsos.

Al igual que en el calibrado de los demás sensores se irá anotando los milímetros movidos en cada momento y la tensión ofrecida a la salida. La distancia será medida con la regla impresa mientras que la tensión se medirá con un multímetro. Se han realizado 6 tomas de datos, 3 de ellas en sentido ascendente y otras 3 en sentido descendente en la distancia, tanto la tabla obtenida como las gráficas están incluidas en el Anexo II de la presente memoria.

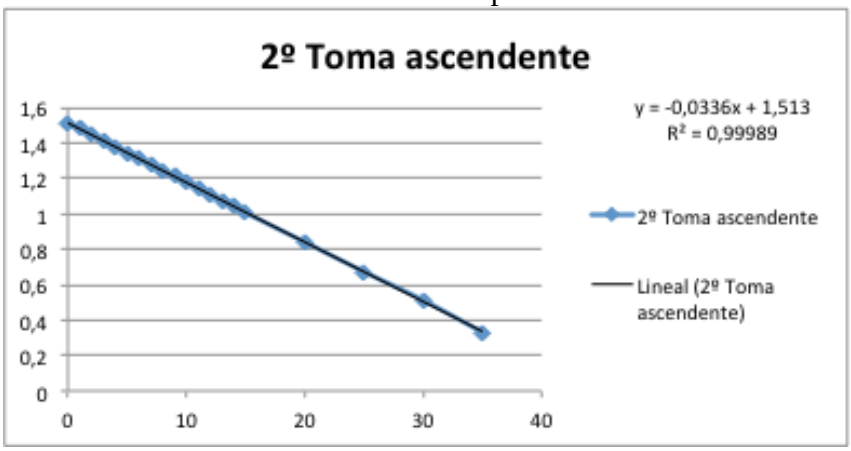



Figura 53 - Gráfica de calibración sensor de posición 2º toma ascendente

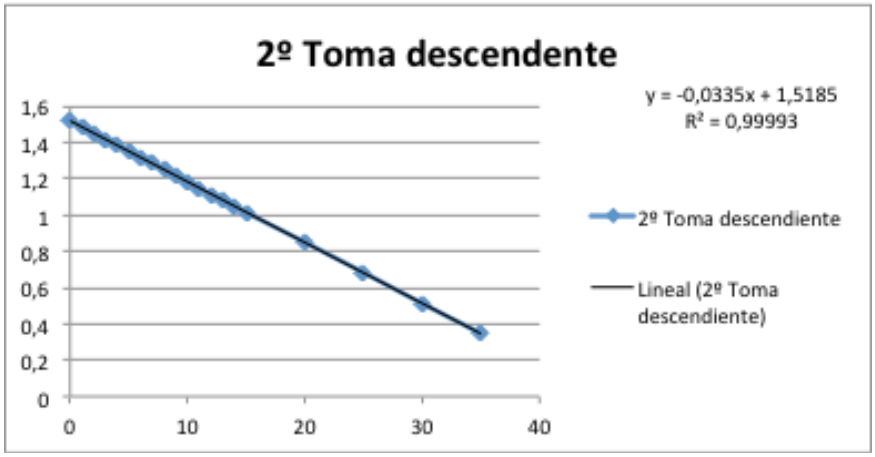



**Figura 54 - Gráfica de calibración sensor de posición 2º toma descendente**

Como se puede observar en la Figura 53 y en la Figura 54 la sensibilidad obtenida es de 33,5 mV/mm, es decir por cada 33,5 mV que varíe la tensión la barra dentada se ha desplazado un milímetro.

Una vez se tiene todo montado el programa a emplear sería el mismo que el empleado para la monitorización añadiendo una variable más que se mediría a través de un puerto analógico. Una vez se obtenga la entrada analógica se dividiría por la sensibilidad y se obtendría el desplazamiento obtenido, el código a añadir sería el siguiente:

>> float sensordesplazamiento=AX

>>cadena[y]=((5.0*analogRead(sensordesplazamiento)/1023.0)/0,0335);

Donde X es el número del puerto analógico e y es la posición en la cadena en la cual se desea colocar el resultado.

Esta caracterización del sistema no se ha podido realizar debido a los problemas existentes en la válvulas, cualquier dato obtenido por el sistema hubiese sido irreal y por tanto no se implementaría un filtro eficaz.

### 1.4.5 Resultados

Tras la realización de los ensayos y el procesado de los datos se ha obtenido la Tabla 3 que muestra los valores máximos y mínimos alcanzados en cada una de las tres velocidades y para cada cavidad.

**Tabla 3 - Tabla de presiones máximas y mínimas de los 9 ensayos**

| | | Velocidad Normal | | | Velocidad Lenta (Low) | | | Velocidad Rápida (Fast) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 N | 2 N | 3 N | 1 L | 2 L | 3 L | 1 F | 2 F | 3 F |
| CEA | P. Máx | 150 | 150 | 150 | 150 | 150 | 150 | 150 | 150 | 150 |
| | P. Min | 33 | 33 | 33 | 33 | 33 | 33 | 34 | 34 | 34 |
| CER móvil | P. Máx de pico | 117 | 117 | 111 | 116 | 116 | 115 | 108 | 103 | 103 |
| | P. Máx estable | 94 | 94 | 84 | 93 | 94 | 93 | 94 | 93 | 93 |
| | P. Min | 84 | 84 | 0 | 83 | 83 | 84 | 44 | 40 | 40 |
| CER fijo | P. Máx de pico | 119 | 118 | 118 | 117 | 118 | 117 | 118 | 118 | 118 |
| | P. Máx estable | 105 | 104 | 104 | 104 | 104 | 104 | 104 | 104 | 105 |
| | P. Min | 94 | 94 | 94 | 94 | 95 | 84 | 98 | 98 | 98 |

En la Tabla 3 se puede observar las presiones más significativas en cada una de las tomas y según la cavidad. En el caso del CEA las presiones son la máxima (P. Máx) y la mínima (P. Min), para las cavidades de expansión radial se cambia la P. Máx por la presión máxima estable (P. Máx estable) y se añade una tercera que es la presión máxima de pico (P. Máx de pico) que se trata de la presión inicial que se alcanza debido a la respuesta elástica de la silicona.
Como se observa las presiones en cada ensayo toman valores iguales o muy parecidos, excepto en el caso de las segunda toma de la velocidad normal donde en el CER móvil se produce una variación de las presiones, esto puede ser debido a un funcionamiento atópico de las válvulas o del sensor.

A continuación, en la Tabla 4, se puede observar las presión medidas por los sensores de presión en cabecera del hardware del Endoworm 3.0, estas presiones son diferenciales, por lo que para una correcta comparación se tendría que tener en cuenta la presión ambiente existente en el momento del ensayo.

**Tabla 4 - Tabla de presiones de las cavidades medidas desde los sensores de presión de cabecera**

| | | Velocidad Normal | Velocidad Lenta (Low) | Velocidad Rápida (Fast) |
|---|---|---|---|---|
| CEA | P. Máx | 129 | 129 | 129 |
| | P. Min | 0 | 0 | 0 |
| CER móvil | P. Máx de pico | 121 | 125 | 101 |
| | P. Máx estable | 30 | 30 | 30 |
| | P. Min | 0 | 0 | 0 |
| CER fijo | P. Máx de pico | 104 | 92 | 101 |
| | P. Máx estable | 10 | 10 | 10 |
| | P. Min | 0 | 0 | 0 |

Como se puedo observar en las dos tablas anteriores, los valores máximos en cada una de las cavidades no son en ninguno de los casos similares, para establecer una relación mejor se han obtenido 3 gráficas que muestran la evolución de los 3 primeros ciclos de cada cavidad y al mismo la tiempo la presión que debería haber tomado según la programación del prototipo.

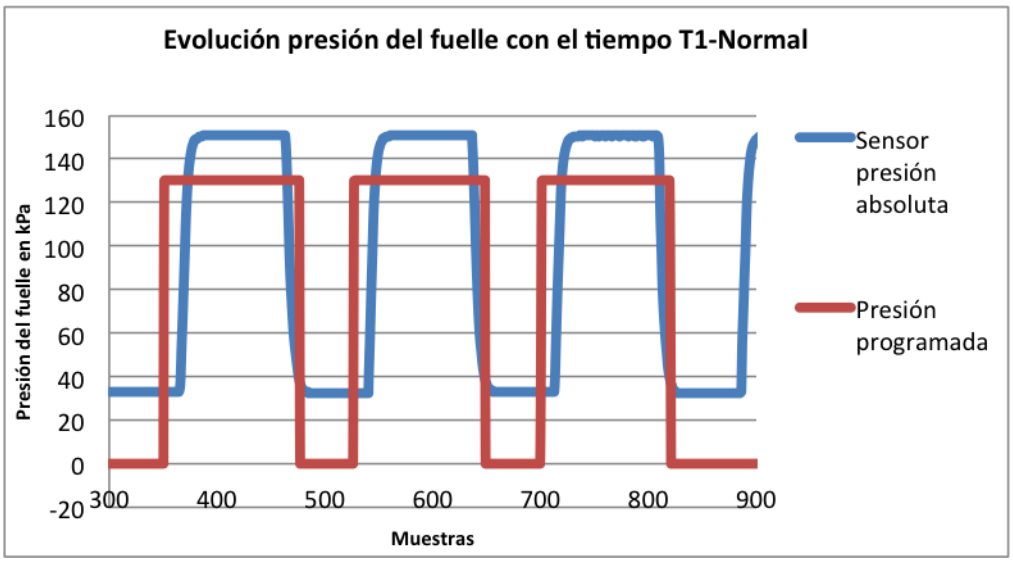



**Figura 55- Gráfica comparativa de la presión medida y la programada del CEA**

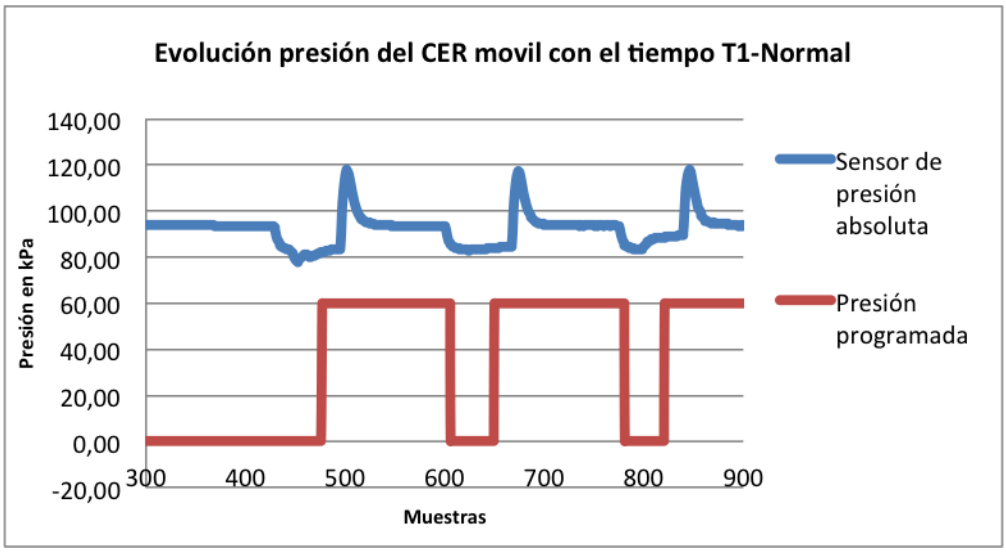



Figura 56 - Gráfica comparativa de la presión medida y la programada del CER móvil

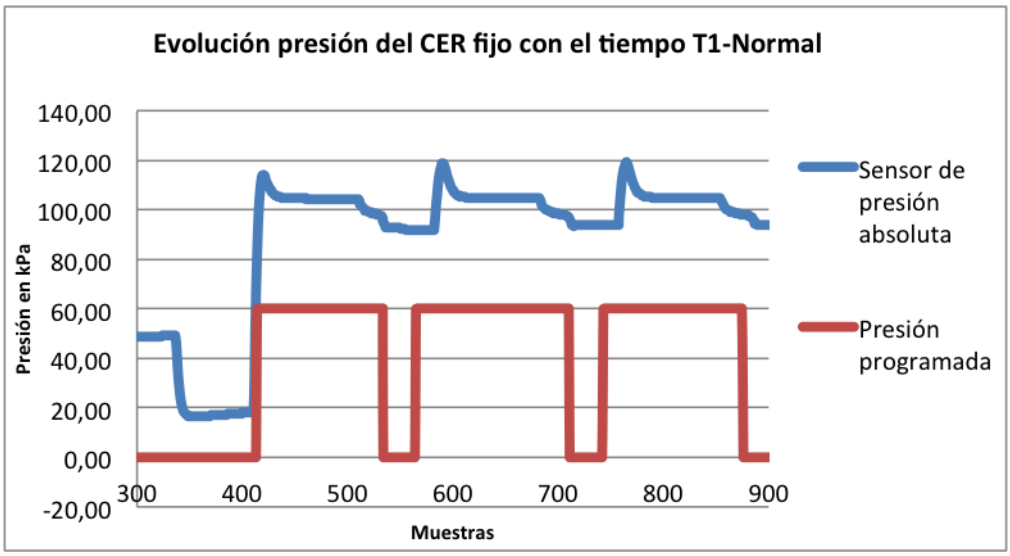



Figura 57 - Gráfica comparativa de la presión medida y la programada del CER fijo

Comenzando por la gráfica de la Figura 51 de la evolución de la presión en el CEA, se puede apreciar que los tiempos de llenado y de vaciado medidos se adaptan a los programados, mientras que la presión máxima obtenida en la cavidad es mucho mayor a la original. Habiendo sido programada una presión de 130 kPa, se está midiendo una presión máxima de 150 kPa.

El en caso de la gráfica de la Figura 52 de la evolución de la presión en el CER móvil, sucede algo parecido al caso anterior. Se tiene que los tiempos de hinchado y deshinchado se adaptan bien, pero el comportamiento del sistema es completamente distinto al programado. La presión máxima medida cuando se estabiliza es de 9 0 kPa, cuando el prototipo ha sido programado para que alcanzase los 60 kPa. También, cabe destacar, el pico existente cuando se produce el hinchado, esto es debido a las propiedades elásticas de la silicona de la cavidad.

De igual forma que en el caso anterior, la gráfica de la Figura 53 de la evolución de la presión en el CER fijo muestra el mismo comportamiento que en CER móvil. Siendo en este caso la presión máxima cuando se estabiliza la cavidad en el hinchado de 110 kPa, una presión aún más alta a la máxima de la cavidad anterior.

Estos datos revelan un comportamiento del prototipo completamente diferente al programado. Esto puede ser debido, entre otras cosas, al mal funcionamiento del hardware del sistema, como se ha podido comprobar con las válvulas.

El sistema de monitorización permite obtener la presión real a la cual están sometidas las cavidades, teniendo así un método de comprobación que permita la detección y mejora de las partes que estén funcionando de forma incorrecta.

## 1.5 Conclusiones

El principal objetivo de este proyecto era la constatación de que las presiones medidas por los sensores de presión conectados en cabecera estaban realizando medidas correctas de las presiones existentes en cada una de las 3 cavidades.

Este hecho se ha podido comprobar y se ha obtenido, como bien se explica en el apartado de resultados, que la presión máxima programada por software no es la misma que la existente en la cavidad. Esto debe en primer lugar al modo de medición de cada sensor y la distancia existente entre la cavidad y la cabecera del sistema.

Otra de las finalidades de esta instrumentación era la detección de fugas, cosa que también se ha detectado gracias a las gráficas de las presiones de la cavidades que mostraban que las presiones mínimas de los sensores no llegaban a 0 kPa, valor que deberían de alcanzar al aplicarse vacío sobre ellas.

El comportamiento del sistema de monitorización ha sido correcto, la transmisión inalámbrica ha funcionado perfectamente transmitiendo todas las muestras tomadas. Debido a la falta de tiempo y a la aparición de problemas no se ha podido realizar el encapsulamiento del sistema para comprobar el funcionamiento del módulo de radiofrecuencia con elementos que obstruyan la línea de comunicación. Este es un punto en el que trabajar ya que en el momento en el que el sistema se encuentre dentro del intestino los obstáculos a salvar serán mayores y podrían producir una pérdida de comunicación con el exterior.

Las posibles mejoras a realizar en el sistema comienzan con la implementación de un sistema de comunicación más fiable que sea capaz de transmitir aún estando encapsulado y dentro de un intestino.

Otra mejora consiste en la independización del sistema de los cables de alimentación, esto se podría realizar añadiendo una batería de 5 V capaz de alimentar a todos los componentes del sistema.

## 1.6 Bibliografía


- EldeM (Julio 2013). Endoscopía Internacional de México: Historia del endoscopio. México. Recuperado de htttp://endoscopiainternacional.com.mx/historia-del-endoscopio/

- Samper, Esther (Enero 2010). MedTempus Blog de medicina y salud: La asombrosa evolución del endoscopio: Del "guía de luz" de Bozzini a la píldora araña I. Alemania. Recuperado de http://medtempus.com/archives/la-asombrosa-evolucion-del-endoscopio-del-guia-de-luz-de-bozzini-a-la-pildora-arana-i/

- Centro médico Teknon. Enteroscopia: ¿Qué es?. Barcelona. España. Recuperado de http://www.teknon.es/servicio-de-diagnosticos/endoscopia/enteroscopia

- Ruiz Gutiérrez, José Manuel (Agosto 2007). SlideShare: Manual de Programación de Arduino. España. Recuperado de http://es.slideshare.net/kacarot201/manual-programacion-arduino-9276333

- Prometec: Comunicación dúplex con NRF2401. Bilbao. España. Recuperado de http://www.prometec.net/duplex-nrf2401/

- Al Williams (Agosto 2015). Hackaday: Hacking a NRF2401 radio for longer range. EEUU. Recuperado de https://hackaday.com/2015/08/15/hacking-a-nrf24l01-radio-for-longer-range/#more-166032

- Orlando (Abril 2014). Hetpro: módulos de comunicación NRF2401. España. Recuperado de http://hetpro-store.com/TUTORIALES/modulos-de-comunicacion-nrf24l01/

- Dubinsky, Uria. Instructables: Uploading scketch to Arduino Pro Mini using Arduino Uno. Israel. Recuperado de http://www.instructables.com/id/Uploading-sketch-to-Arduino-Pro-Mini-using-Arduino/

- H. Shih, Randy. Tools for Design Using AutoCAD 2014 and Autodesk Inventor 2014. Recuperado de http://www.sdcpublications.com/pdfsample/978-1-58503-806-0-7.pdf

- Lopez Aguilar, Fernando (Agosto 2014). michelletorres.mx: Que es una variable y como se declaran en C++. Recuperado de http://michelletorres.mx/que-es-una-variable-y-como-se-declaran-en-c/

- Parra, Ronny (Enero 2009). SlideShare: Capitulo 4, funciones. Recuperado de http://es.slideshare.net/javi2401/funciones-en-c-presentation

- Manual de programación de Arduino. Recuperado de http://dfists.ua.es/~jpomares/arduino/page_02.htm

- Ruiz Gutiérrez, José Manuel. Arduinobot: Manual de programación. Recuperado de https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf

## 1.7 Anexo I – Datasheet

Honeywell

# SCC SMT Series

## Microstructure Pressure Sensors 0 psi to 5 psi through 0 psi to 100 psi

The SCC SMT Series offers an extremely low-cost sensor element with a temperature-stable output when driven with a constant current source. These integrated circuit sensors were designed for extremely cost-sensitive applications where precise accuracy over a wide temperature range is not required.

The standard surface mount package includes an optional ported lid to fit in a variety of applications.

The absolute devices have an internal vacuum reference and an output voltage proportional to absolute pressure. The differential devices allow application of pressure to either side of the sensing diaphragm and can be used for gage or differential measurements.

The 4-pin closed bridge configuration allows electrical connection with additional pads provided for mechanical support. Pulsed power is recommended to achieve maximum accuracy and conserve battery power in portable applications.

### FEATURES

- Low cost
- Small size
- Internal temperature compensation
- Absolute or gage pressures
- High-impedance bridge
- Low power consumption

### POTENTIAL APPLICATIONS

- Pneumatic controls
- Automotive diagnostics
- Medical equipment/instrumentation
- Dental equipment
- Environmental controls
- Barometric pressure measurement
- Altimeters
- Pneumatic controls
- Battery powered equipment

# SCC SMT Series

## PRESSURE SENSOR SPECIFICATIONS [1]

| Characteristic | Maximum Rating |
|---|---|
| Supply current, $I_s$ | 1.5 mA |
| Compensated temperature | 0 °C to 50 °C [32 °F to 122 °F] |
| Operating temperature | -40 °C to 125 °C [-40 °F to 257 °F] |
| Storage temperature | -55 °C to 125 °C [-67 °F to 257 °F] |
| Humidity | 0% to 100% RH |
| Lead temperature (soldering 2 s to 4 s) | 250 °C [482 °F] |

## STANDARD PRESSURE RANGES [1]

| Operating Pressure | Maximum [2] Pressure | Sensitivity [3] | | Unit |
|---|---|---|---|---|
| | | Nominal | Std. Dev. | |
| 0 psi to 5 psi | 20 psi | 7.50 | ±0.68 | mV/mA/psi |
| 0 psi to 15 psi | 30 psi | 4.30 | ±0.37 | mV/mA/psi |
| 0 psi to 30 psi | 60 psi | 2.90 | ±0.57 | mV/mA/psi |
| 0 psi to 100 psi | 150 psi | 1.30 | ±0.20 | mV/mA/psi |

## PERFORMANCE SPECIFICATIONS [1]

| Characteristic | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|
| Zero pressure offset ($T_A$ = 25 °C) | -30.0 | -10.0 | 20.0 | mV |
| Linearity, hysteresis, repeatability [4] | -1.0 | 0.2 | 1.0 | % FSS |
| Temperature effect on span [5] | -1.5 | 0.25 | 1.5 | % FSS |
| Temperature effect on offset [5] | -2.0 | .5 | 2.0 | % FSS |
| Long-term stability of offset and span [6] | – | 0.1 | – | % FSS |
| Response time (10% to 90%) [7] | – | 0.1 | – | ms |
| Input resistance ($T_A$ = 25 °C) | 4.00 | 5.00 | 6.50 | kΩ |
| Output impedance | 4.00 | 5.00 | 6.50 | kΩ |

## NOTES

Note 1: Reference conditions: Supply current, $I_s$ = 1.0 mA, $T_A$ = 25 °C to 70 °C [32 °F to 158 °F], common-mode Line pressure = 0 psig, pressure applied to P1 unless otherwise noted.

Note 2: If the maximum pressure is exceeded, even momentarily, the package may leak or burst, or the pressure sensing die may fracture.

Note 3: Sensitivity is the ratio of the output signal voltage change to the corresponding input pressure change. The sensitivity is characterized by design and periodic production testing. This parameter in not 100 % tested in production.

Note 4: Linearity is based on best straight line fit. Hysteresis is the maximum output difference at any point within the operating pressure range for increasing and decreasing pressure.

Note 5: Maximum error band of the offset voltage and the error of the band of the span over the compensated temperature range, relative to the 25 °C reading. Typical temperature coefficients for span and resistance are -2200 ppm/°C and 2200 ppm/°C, respectively. Temperature effects on offset and span are guaranteed by design. These parameters are not 100 % tested in production.

Note 6: Long term stability over a one year period.

Note 7: Response time for 0 psi to full scale span pressure step change.

# LM134/LM234/LM334 3-Terminal Adjustable Current Sources

Check for Samples: LM134, LM234, LM334

## FEATURES

- Operates From 1V to 40V
- 0.02%/V Current Regulation
- Programmable From 1µA to 10mA
- True 2-Terminal Operation
- Available as Fully Specified Temperature Sensor
- ±3% Initial Accuracy

## DESCRIPTION

The LM134/LM234/LM334 are 3-terminal adjustable current sources featuring 10,000:1 range in operating current, excellent current regulation and a wide dynamic voltage range of 1V to 40V. Current is established with one external resistor and no other parts are required. Initial current accuracy is ±3%. The LM134/LM234/LM334 are true floating current sources with no separate power supply connections. In addition, reverse applied voltages of up to 20V will draw only a few dozen microamperes of current, allowing the devices to act as both a rectifier and current source in AC applications.

The sense voltage used to establish operating current in the LM134 is 64mV at 25°C and is directly proportional to absolute temperature (°K). The simplest one external resistor connection, then, generates a current with ≈+0.33%/°C temperature dependence. Zero drift operation can be obtained by adding one extra resistor and a diode.

Applications for the current sources include bias networks, surge protection, low power reference, ramp generation, LED driver, and temperature sensing. The LM234-3 and LM234-6 are specified as true temperature sensors with ensured initial accuracy of ±3°C and ±6°C, respectively. These devices are ideal in remote sense applications because series resistance in long wire runs does not affect accuracy. In addition, only 2 wires are required.

The LM134 is specified over a temperature range of −55°C to +125°C, the LM234 from −25°C to +100°C and the LM334 from 0°C to +70°C. These devices are available in TO hermetic, TO-92 and SOIC-8 plastic packages.

### Connection Diagrams

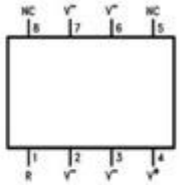



**Figure 1. SOIC-8 Surface Mount Package (LM334M; LM334M/NOPB; LM334MX; LM334MX/NOPB) See Package Number D**

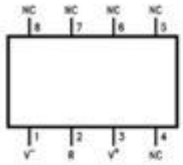



**Figure 2. SOIC-8 Alternative Pinout Surface Mount Package (LM334SM; LM334SM/NOPB; LM334SMX; LM334SMX/NOPB) See Package Number D**

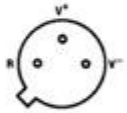



**Figure 3. TO Metal Can Package (Bottom View) See Package Number NDV**


**Figure 4. TO-92 Plastic Package (Bottom View) See Package Number LP**

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

## Absolute Maximum Ratings(1)(2)

| | | | |
|---|---|---|---|
| V⁺ to V⁻ Forward Voltage | | LM134/LM234/LM334 | 40V |
| | | LM234-3/LM234-6 | 30V |
| V⁺ to V⁻ Reverse Voltage | | | 20V |
| R Pin to V⁻ Voltage | | | 5V |
| Set Current | | | 10 mA |
| Power Dissipation | | | 400 mW |
| ESD Susceptibility(3) | | | 2000V |
| Operating Temperature Range(4) | | LM134 | −55°C to +125°C |
| | | LM234/LM234-3/LM234-6 | −25°C to +100°C |
| | | LM334 | 0°C to +70°C |
| Soldering Information | TO-92 Package (10 sec.) | | 260°C |
| | TO Package (10 sec.) | | 300°C |
| | SOIC Package | Vapor Phase (60 sec.) | 215°C |
| | | Infrared (15 sec.) | 220°C |

(1) "Absolute Maximum Ratings" indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not ensure specific performance limits.
(2) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/Distributors for availability and specifications.
(3) Human body model, 100pF discharged through a 1.5kΩ resistor.
(4) For elevated temperature operation, $T_J$ max is:

| | |
|---|---|
| LM134 | 150°C |
| LM234 | 125°C |
| LM334 | 100°C |

See Thermal Characteristics.

## Thermal Characteristics

over operating free-air temperature range (unless otherwise noted)

| Thermal Resistance | TO-92 | TO | SOIC-8 |
|---|---|---|---|
| $\theta_{JA}$ (Junction to Ambient) | 180°C/W (0.4" leads) | 440°C/W | 165°C/W |
| | 160°C/W (0.125" leads) | | |
| $\theta_{JC}$ (Junction to Case) | N/A | 32°C/W | 80°C/W |

## Electrical Characteristics[1]

| Parameter | Conditions | | LM134/LM234 | | | LM334 | | | Units |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Typ | Max | Min | Typ | Max | |
| Set Current Error, V⁺=2.5V[2] | 10µA ≤ $I_{SET}$ ≤ 1mA | | | | 3 | | | 6 | % |
| | 1mA < $I_{SET}$ ≤ 5mA | | | | 5 | | | 8 | % |
| | 2µA ≤ $I_{SET}$ < 10µA | | | | 8 | | | 12 | % |
| Ratio of Set Current to Bias Current | 100µA ≤ $I_{SET}$ ≤ 1mA | | 14 | 18 | 23 | 14 | 18 | 26 | |
| | 1mA ≤ $I_{SET}$ ≤ 5mA | | | 14 | | | 14 | | |
| | 2 µA≤$I_{SET}$≤100 µA | | | 18 | 23 | | 18 | 26 | |
| Minimum Operating Voltage | 2µA ≤ $I_{SET}$ ≤ 100µA | | | 0.8 | | | 0.8 | | V |
| | 100µA < $I_{SET}$ ≤ 1mA | | | 0.9 | | | 0.9 | | V |
| | 1mA < $I_{SET}$ ≤ 5mA | | | 1.0 | | | 1.0 | | V |
| Average Change in Set Current with Input Voltage | 2µA ≤ $I_{SET}$ ≤ 1mA | 1.5 ≤ V⁺ ≤ 5V | | 0.02 | 0.05 | | 0.02 | 0.1 | %/V |
| | | 5V ≤ V⁺ ≤ 40V | | 0.01 | 0.03 | | 0.01 | 0.05 | %/V |
| | 1mA < $I_{SET}$ ≤ 5mA | 1.5V ≤ V ≤ 5V | | 0.03 | | | 0.03 | | %/V |
| | | 5V ≤ V ≤ 40V | | 0.02 | | | 0.02 | | %/V |
| Temperature Dependence of Set Current[3] | 25µA ≤ $I_{SET}$ ≤ 1mA | | 0.96T | T | 1.04T | 0.96T | T | 1.04T | |
| Effective Shunt Capacitance | | | | 15 | | | 15 | | pF |

(1) Unless otherwise specified, tests are performed at $T_j$ = 25°C with pulse testing so that junction temperature does not change during test
(2) Set current is the current flowing into the V⁺ pin. For the Basic 2-Terminal Current Source circuit shown in Figure 13. $I_{SET}$ is determined by the following formula: $I_{SET}$ = 67.7 mV/$R_{SET}$ (@ 25°C). Set current error is expressed as a percent deviation from this amount. $I_{SET}$ increases at 0.336%/°C @ $T_j$ = 25°C (227 µV/°C).
(3) $I_{SET}$ is directly proportional to absolute temperature (°K). $I_{SET}$ at any temperature can be calculated from: $I_{SET}$ = $I_o$ ($T/T_o$) where $I_o$ is $I_{SET}$ measured at $T_o$ (°K).

## Electrical Characteristics[1]

| Parameter | Conditions | | LM234-3 | | | LM234-6 | | | Units |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Typ | Max | Min | Typ | Max | |
| Set Current Error, V⁺=2.5V [2] | 100µA ≤ $I_{SET}$ ≤ 1mA | | | | ±1 | | | ±2 | % |
| | $T_J$ = 25° | | | | | | | | |
| Equivalent Temperature Error | | | | | ±3 | | | ±6 | °C |
| Ratio of Set Current to Bias Current | 100µA ≤ $I_{SET}$ ≤ 1mA | | 14 | 18 | 26 | 14 | 18 | 26 | |
| Minimum Operating Voltage | 100µA $I_{SET}$ ≤ 1mA | | | 0.9 | | | 0.9 | | V |
| Average Change in Set Current with Input Voltage | 100µA ≤ $I_{SET}$ ≤ 1mA | 1.5 ≤ V⁺ ≤ 5V | | 0.02 | 0.05 | | 0.02 | 0.01 | %/V |
| | | 5V ≤ V⁺ ≤ 30V | | 0.01 | 0.03 | | 0.01 | 0.05 | %/V |
| Temperature Dependence of Set Current[3] | 100µA ≤ $I_{SET}$ ≤ 1mA | | 0.98T | T | 1.02T | 0.97T | T | 1.03T | |
| Equivalent Slope Error | | | | | ±2 | | | ±3 | % |
| Effective Shunt Capacitance | | | | 15 | | | 15 | | pF |

(1) Unless otherwise specified, tests are performed at $T_j$ = 25°C with pulse testing so that junction temperature does not change during test
(2) Set current is the current flowing into the V⁺ pin. For the Basic 2-Terminal Current Source circuit shown in Figure 13. $I_{SET}$ is determined by the following formula: $I_{SET}$ = 67.7 mV/$R_{SET}$ (@ 25°C). Set current error is expressed as a percent deviation from this amount. $I_{SET}$ increases at 0.336%/°C @ $T_j$ = 25°C (227 µV/°C).
(3) $I_{SET}$ is directly proportional to absolute temperature (°K). $I_{SET}$ at any temperature can be calculated from: $I_{SET}$ = $I_o$ ($T/T_o$) where $I_o$ is $I_{SET}$ measured at $T_o$ (°K).

## APPLICATION HINTS

The LM134 has been designed for ease of application, but a general discussion of design features is presented here to familiarize the designer with device characteristics which may not be immediately obvious. These include the effects of slewing, power dissipation, capacitance, noise, and contact resistance.

### Calculating $R_{SET}$

The total current through the LM134 ($I_{SET}$) is the sum of the current going through the SET resistor ($I_R$) and the LM134's bias current ($I_{BIAS}$), as shown in Figure 13.

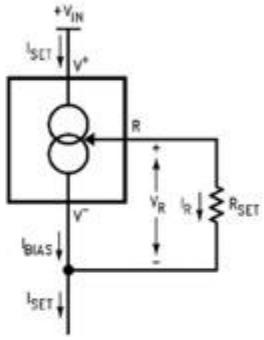



**Figure 13. Basic Current Source**

A graph showing the ratio of these two currents is supplied under **Ratio of $I_{SET}$ to $I_{BIAS}$** in Typical Performance Characteristics. The current flowing through $R_{SET}$ is determined by $V_R$, which is approximately 214µV/°K (64 mV/298°K ~ 214µV/°K).

$$I_{SET} = I_R + I_{BIAS} = \frac{V_R}{R_{SET}} + I_{BIAS} \quad (1)$$

Since (for a given set current) $I_{BIAS}$ is simply a percentage of $I_{SET}$, the equation can be rewritten

$$I_{SET} = \left(\frac{V_R}{R_{SET}}\right)\left(\frac{n}{n-1}\right)$$

where

- n is the ratio of $I_{SET}$ to $I_{BIAS}$ as specified in Electrical Characteristics and shown in the graph (2)

Since n is typically 18 for 2µA ≤ $I_{SET}$ ≤ 1mA, the equation can be further simplified to

$$I_{SET} = \left(\frac{V_R}{R_{SET}}\right)(1.059) = \frac{227\ \mu V/°K}{R_{SET}} \quad (3)$$

for most set currents.

### Slew Rate

At slew rates above a given threshold (see curve), the LM134 may exhibit non-linear current shifts. The slewing rate at which this occurs is directly proportional to $I_{SET}$. At $I_{SET}$ = 10µA, maximum dV/dt is 0.01V/µs; at $I_{SET}$ = 1mA, the limit is 1V/µs. Slew rates above the limit do not harm the LM134, or cause large currents to flow.

### Thermal Effects

Internal heating can have a significant effect on current regulation for $I_{SET}$ greater than 100µA. For example, each 1V increase across the LM134 at $I_{SET}$ = 1 mA will increase junction temperature by ≈0.4°C in still air. Output current ($I_{SET}$) has a temperature coefficient of ≈0.33%/°C, so the change in current due to temperature rise will be (0.4) (0.33) = 0.132%. This is a 10:1 degradation in regulation compared to true electrical effects. Thermal effects, therefore, must be taken into account when DC regulation is critical and $I_{SET}$ exceeds 100µA. Heat sinking of the TO package or the TO-92 leads can reduce this effect by more than 3:1.

TEXAS INSTRUMENTS

LM35

SNIS159G – AUGUST 1999 – REVISED AUGUST 2016

# LM35 Precision Centigrade Temperature Sensors

## 1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-µA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

## 2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

## 3 Description

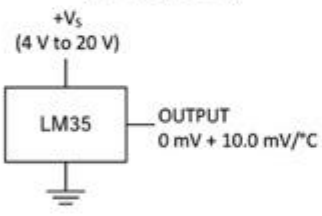
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 µA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

**Device Information**(1)

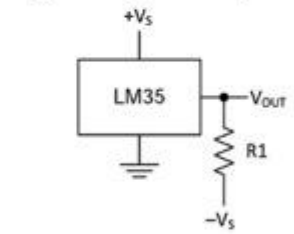
| PART NUMBER | PACKAGE | BODY SIZE (NOM) |
|---|---|---|
| LM35 | TO-CAN (3) | 4.699 mm × 4.699 mm |
| | TO-92 (3) | 4.30 mm × 4.30 mm |
| | SOIC (8) | 4.90 mm × 3.91 mm |
| | TO-220 (3) | 14.986 mm × 10.16 mm |

(1) For all available packages, see the orderable addendum at the end of the datasheet.

**Basic Centigrade Temperature Sensor (2°C to 150°C)**




**Full-Range Centigrade Temperature Sensor**




Choose $R_1 = -V_S / 50\ \mu A$
$V_{OUT} = 1500$ mV at 150°C
$V_{OUT} = 250$ mV at 25°C
$V_{OUT} = -550$ mV at −55°C

## Table of Contents



## 4 Revision History

**Changes from Revision F (January 2016) to Revision G** — **Page**

- Equation 1, changed From: 10 mV/°F To: 10mv/°C ... 13
- *Power Supply Recommendations*, changed From: "4-V to 5.5-V power supply" To: "4-V to 30-V power supply: ... 19

**Changes from Revision E (January 2015) to Revision F** — **Page**

- Changed NDV Package (TO-CAN) pinout from Top View to Bottom View ... 3

**Changes from Revision D (October 2013) to Revision E** — **Page**

- Added *Pin Configuration and Functions* section, *ESD Ratings* table, *Feature Description* section, *Device Functional Modes*, *Application and Implementation* section, *Power Supply Recommendations* section, *Layout* section, *Device and Documentation Support* section, and *Mechanical, Packaging, and Orderable Information* section ... 1

**Changes from Revision C (July 2013) to Revision D** — **Page**

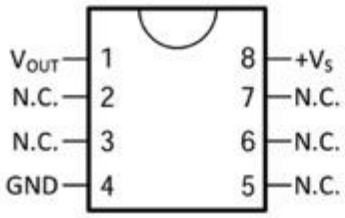
- Changed *W* to Ω ... 1
- Changed *W* to Ω in *Abs Max* tablenote. ... 4

## 5 Pin Configuration and Functions

**NDV Package**
**3-Pin TO-CAN**
**(Bottom View)**

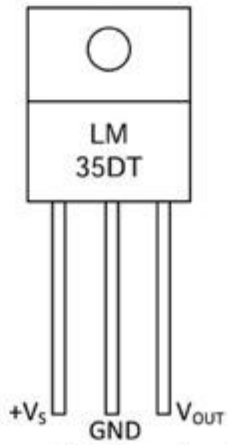

Case is connected to negative pin (GND)

**D Package**
**8-PIN SOIC**
**(Top View)**




N.C. = No connection

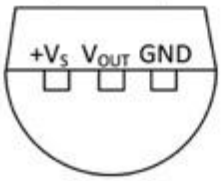
**NEB Package**
**3-Pin TO-220**
**(Top View)**




Tab is connected to the negative pin (GND).

**NOTE:** The LM35DT pinout is different than the discontinued LM35DP

**LP Package**
**3-Pin TO-92**
**(Bottom View)**




**Pin Functions**

| PIN | | | | | TYPE | DESCRIPTION |
|---|---|---|---|---|---|---|
| NAME | TO46 | TO92 | TO220 | SO8 | | |
| $V_{OUT}$ | — | — | — | 1 | O | Temperature Sensor Analog Output |
| N.C. | — | — | — | 2 | — | No Connection |
| | — | — | — | 3 | | |
| GND | — | — | — | 4 | GROUND | Device ground pin, connect to power supply negative terminal |
| N.C. | — | — | — | 5 | — | No Connection |
| | — | — | — | 6 | | |
| | — | — | — | 7 | | |
| $+V_S$ | — | — | — | 8 | POWER | Positive power supply pin |

# 6 Specifications

## 6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)[1][2]

| | | MIN | MAX | UNIT |
|---|---|---|---|---|
| Supply voltage | | –0.2 | 35 | V |
| Output voltage | | –1 | 6 | V |
| Output current | | | 10 | mA |
| Maximum Junction Temperature, $T_J$max | | | 150 | °C |
| Storage Temperature, $T_{stg}$ | TO-CAN, TO-92 Package | –60 | 150 | °C |
| | TO-220, SOIC Package | –65 | 150 | |

(1) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(2) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

## 6.2 ESD Ratings

| | | | VALUE | UNIT |
|---|---|---|---|---|
| $V_{(ESD)}$ | Electrostatic discharge | Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001[1] | ±2500 | V |

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

## 6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

| | | MIN | MAX | UNIT |
|---|---|---|---|---|
| Specified operating temperature: $T_{MIN}$ to $T_{MAX}$ | LM35, LM35A | –55 | 150 | °C |
| | LM35C, LM35CA | –40 | 110 | |
| | LM35D | 0 | 100 | |
| Supply Voltage (+$V_S$) | | 4 | 30 | V |

## 6.4 Thermal Information

| THERMAL METRIC[1][2] | | LM35 | | | | UNIT |
|---|---|---|---|---|---|---|
| | | NDV | LP | D | NEB | |
| | | 3 PINS | | 8 PINS | 3 PINS | |
| $R_{\theta JA}$ | Junction-to-ambient thermal resistance | 400 | 180 | 220 | 90 | °C/W |
| $R_{\theta JC(top)}$ | Junction-to-case (top) thermal resistance | 24 | — | — | — | |

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.
(2) For additional thermal resistance information, see *Typical Application*.

## 6.5 Electrical Characteristics: LM35A, LM35CA Limits

Unless otherwise noted, these specifications apply: $-55°C \le T_J \le 150°C$ for the LM35 and LM35A; $-40°C \le T_J \le 110°C$ for the LM35C and LM35CA; and $0°C \le T_J \le 100°C$ for the LM35D. $V_S$ = 5 Vdc and $I_{LOAD}$ = 50 µA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to $T_{MAX}$ in the circuit of Figure 14.

| PARAMETER | TEST CONDITIONS | LM35A | | | LM35CA | | | UNIT |
|---|---|---|---|---|---|---|---|---|
| | | TYP | TESTED LIMIT(1) | DESIGN LIMIT(2) | TYP | TESTED LIMIT(1) | DESIGN LIMIT(2) | |
| Accuracy(3) | $T_A$ = 25°C | ±0.2 | ±0.5 | | ±0.2 | ±0.5 | | °C |
| | $T_A$ = –10°C | ±0.3 | | | ±0.3 | | ±1 | |
| | $T_A$ = $T_{MAX}$ | ±0.4 | ±1 | | ±0.4 | ±1 | | |
| | $T_A$ = $T_{MIN}$ | ±0.4 | ±1 | | ±0.4 | | ±1.5 | |
| Nonlinearity(4) | $T_{MIN} \le T_A \le T_{MAX}$, $-40°C \le T_J \le 125°C$ | ±0.18 | | ±0.35 | ±0.15 | | ±0.3 | °C |
| Sensor gain (average slope) | $T_{MIN} \le T_A \le T_{MAX}$ | 10 | 9.9 | | 10 | | 9.9 | mV/°C |
| | $-40°C \le T_J \le 125°C$ | 10 | 10.1 | | 10 | | 10.1 | |
| Load regulation(5) $0 \le I_L \le 1$ mA | $T_A$ = 25°C | ±0.4 | ±1 | | ±0.4 | ±1 | | mV/mA |
| | $T_{MIN} \le T_A \le T_{MAX}$, $-40°C \le T_J \le 125°C$ | ±0.5 | | ±3 | ±0.5 | | ±3 | |
| Line regulation(5) | $T_A$ = 25°C | ±0.01 | ±0.05 | | ±0.01 | ±0.05 | | mV/V |
| | 4 V ≤ $V_S$ ≤ 30 V, $-40°C \le T_J \le 125°C$ | ±0.02 | | ±0.1 | ±0.02 | | ±0.1 | |
| Quiescent current(6) | $V_S$ = 5 V, 25°C | 56 | 67 | | 56 | 67 | | µA |
| | $V_S$ = 5 V, $-40°C \le T_J \le 125°C$ | 105 | | 131 | 91 | | 114 | |
| | $V_S$ = 30 V, 25°C | 56.2 | 68 | | 56.2 | 68 | | |
| | $V_S$ = 30 V, $-40°C \le T_J \le 125°C$ | 105.5 | | 133 | 91.5 | | 116 | |
| Change of quiescent current(5) | 4 V ≤ $V_S$ ≤ 30 V, 25°C | 0.2 | 1 | | 0.2 | 1 | | µA |
| | 4 V ≤ $V_S$ ≤ 30 V, $-40°C \le T_J \le 125°C$ | 0.5 | | 2 | 0.5 | | 2 | |
| Temperature coefficient of quiescent current | $-40°C \le T_J \le 125°C$ | 0.39 | | 0.5 | 0.39 | | 0.5 | µA/°C |
| Minimum temperature for rate accuracy | In circuit of Figure 14, $I_L$ = 0 | 1.5 | | 2 | 1.5 | | 2 | °C |
| Long term stability | $T_J$ = $T_{MAX}$, for 1000 hours | ±0.08 | | | ±0.08 | | | °C |

(1) Tested Limits are ensured and 100% tested in production.
(2) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(3) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(4) Non-linearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(5) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.
(6) Quiescent current is defined in the circuit of Figure 14.

## 6.6 Electrical Characteristics: LM35A, LM35CA

Unless otherwise noted, these specifications apply: −55°C ≤ $T_J$ ≤ 150°C for the LM35 and LM35A; −40°C ≤ $T_J$ ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ $T_J$ ≤ 100°C for the LM35D. $V_S$ = 5 Vdc and $I_{LOAD}$ = 50 μA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to $T_{MAX}$ in the circuit of Figure 14.

| PARAMETER | TEST CONDITIONS | | LM35A | | | LM35CA | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|
| | | | MIN | TYP | MAX | TYP | TYP | MAX | |
| Accuracy(1) | $T_A$ = 25°C | | | ±0.2 | | | ±0.2 | | °C |
| | | Tested Limit(2) | | | ±0.5 | | | ±0.5 | |
| | | Design Limit(3) | | | | | | | |
| | $T_A$ = −10°C | | | ±0.3 | | | ±0.3 | | |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | | | | ±1 | |
| | $T_A$ = $T_{MAX}$ | | | ±0.4 | | | ±0.4 | | |
| | | Tested Limit(2) | | | ±1 | | | ±1 | |
| | | Design Limit(3) | | | | | | | |
| | $T_A$ = $T_{MIN}$ | | | ±0.4 | | | ±0.4 | | |
| | | Tested Limit(2) | | | ±1 | | | | |
| | | Design Limit(3) | | | | | | ±1.5 | |
| Nonlinearity(4) | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | | | ±0.18 | | | ±0.15 | | °C |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | ±0.35 | | | ±0.3 | |
| Sensor gain (average slope) | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$ | | | 10 | | | 10 | | mV/°C |
| | | Tested Limit(2) | | | 9.9 | | | | |
| | | Design Limit(3) | | | | | | 9.9 | |
| | −40°C ≤ $T_J$ ≤ 125°C | | | 10 | | | 10 | | |
| | | Tested Limit(2) | | | 10.1 | | | | |
| | | Design Limit(3) | | | | | | 10.1 | |
| Load regulation(5) 0 ≤ $I_L$ ≤ 1 mA | $T_A$ = 25°C | | | ±0.4 | | | ±0.4 | | mV/mA |
| | | Tested Limit(2) | | | ±1 | | | ±1 | |
| | | Design Limit(3) | | | | | | | |
| | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | | | ±0.5 | | | ±0.5 | | |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | ±3 | | | ±3 | |
| Line regulation(5) | $T_A$ = 25°C | | | ±0.01 | | | ±0.01 | | mV/V |
| | | Tested Limit(2) | | | ±0.05 | | | ±0.05 | |
| | | Design Limit(3) | | | | | | | |
| | 4 V ≤ $V_S$ ≤ 30 V, −40°C ≤ $T_J$ ≤ 125°C | | | ±0.02 | | | ±0.02 | | |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | ±0.1 | | | ±0.1 | |

(1) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(2) Tested Limits are ensured and 100% tested in production.
(3) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(4) Non-linearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(5) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

## Electrical Characteristics: LM35A, LM35CA (continued)

Unless otherwise noted, these specifications apply: −55°C ≤ $T_J$ ≤ 150°C for the LM35 and LM35A; −40°C ≤ $T_J$ ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ $T_J$ ≤ 100°C for the LM35D. $V_S$ = 5 Vdc and $I_{LOAD}$ = 50 µA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to $T_{MAX}$ in the circuit of Figure 14.

| PARAMETER | TEST CONDITIONS | | LM35A | | | LM35CA | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|
| | | | MIN | TYP | MAX | TYP | TYP | MAX | |
| Quiescent current(6) | $V_S$ = 5 V, 25°C | | | 56 | | | 56 | | µA |
| | | Tested Limit(2) | | | 67 | | | 67 | |
| | | Design Limit(3) | | | | | | | |
| | $V_S$ = 5 V, −40°C ≤ $T_J$ ≤ 125°C | | | 105 | | | 91 | | |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | 131 | | | 114 | |
| | $V_S$ = 30 V, 25°C | | | 56.2 | | | 56.2 | | |
| | | Tested Limit(2) | | | 68 | | | 68 | |
| | | Design Limit(3) | | | | | | | |
| | $V_S$ = 30 V, −40°C ≤ $T_J$ ≤ 125°C | | | 105.5 | | | 91.5 | | |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | 133 | | | 116 | |
| Change of quiescent current(5) | 4 V ≤ $V_S$ ≤ 30 V, 25°C | | | 0.2 | | | 0.2 | | µA |
| | | Tested Limit(2) | | | 1 | | | 1 | |
| | | Design Limit(3) | | | | | | | |
| | 4 V ≤ $V_S$ ≤ 30 V, −40°C ≤ $T_J$ ≤ 125°C | | | 0.5 | | | 0.5 | | |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | 2 | | | 2 | |
| Temperature coefficient of quiescent current | −40°C ≤ $T_J$ ≤ 125°C | | | 0.39 | | | 0.39 | | µA/°C |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | 0.5 | | | 0.5 | |
| Minimum temperature for rate accuracy | In circuit of Figure 14, $I_L$ = 0 | | | 1.5 | | | 1.5 | | °C |
| | | Tested Limit(2) | | | | | | | |
| | | Design Limit(3) | | | 2 | | | 2 | |
| Long term stability | $T_J$ = $T_{MAX}$, for 1000 hours | | | ±0.08 | | | ±0.08 | | °C |

(6) Quiescent current is defined in the circuit of Figure 14.

## 6.7 Electrical Characteristics: LM35, LM35C, LM35D Limits

Unless otherwise noted, these specifications apply: −55°C ≤ $T_J$ ≤ 150°C for the LM35 and LM35A; −40°C ≤ $T_J$ ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ $T_J$ ≤ 100°C for the LM35D. $V_S$ = 5 Vdc and $I_{LOAD}$ = 50 µA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to $T_{MAX}$ in the circuit of Figure 14.

| PARAMETER | TEST CONDITIONS | LM35 | | | LM35C, LM35D | | | UNIT |
|---|---|---|---|---|---|---|---|---|
| | | TYP | TESTED LIMIT(1) | DESIGN LIMIT(2) | TYP | TESTED LIMIT(1) | DESIGN LIMIT(2) | |
| Accuracy, LM35, LM35C(3) | $T_A$ = 25°C | ±0.4 | ±1 | | ±0.4 | ±1 | | °C |
| | $T_A$ = −10°C | ±0.5 | | | ±0.5 | | ±1.5 | |
| | $T_A$ = $T_{MAX}$ | ±0.8 | ±1.5 | | ±0.8 | | ±1.5 | |
| | $T_A$ = $T_{MIN}$ | ±0.8 | | ±1.5 | ±0.8 | | ±2 | |
| Accuracy, LM35D(3) | $T_A$ = 25°C | | | | ±0.6 | ±1.5 | | °C |
| | $T_A$ = $T_{MAX}$ | | | | ±0.9 | | ±2 | |
| | $T_A$ = $T_{MIN}$ | | | | ±0.9 | | ±2 | |
| Nonlinearity(4) | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | ±0.3 | | ±0.5 | ±0.2 | | ±0.5 | °C |
| Sensor gain (average slope) | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | 10 | 9.8 | | 10 | | 9.8 | mV/°C |
| | | 10 | 10.2 | | 10 | | 10.2 | |
| Load regulation(5) 0 ≤ $I_L$ ≤ 1 mA | $T_A$ = 25°C | ±0.4 | ±2 | | ±0.4 | ±2 | | mV/mA |
| | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | ±0.5 | | ±5 | ±0.5 | | ±5 | |
| Line regulation(5) | $T_A$ = 25°C | ±0.01 | ±0.1 | | ±0.01 | ±0.1 | | mV/V |
| | 4 V ≤ $V_S$ ≤ 30 V, −40°C ≤ $T_J$ ≤ 125°C | ±0.02 | | ±0.2 | ±0.02 | | ±0.2 | |
| Quiescent current(6) | $V_S$ = 5 V, 25°C | 56 | 80 | | 56 | 80 | | µA |
| | $V_S$ = 5 V, −40°C ≤ $T_J$ ≤ 125°C | 105 | | 158 | 91 | | 138 | |
| | $V_S$ = 30 V, 25°C | 56.2 | 82 | | 56.2 | 82 | | |
| | $V_S$ = 30 V, −40°C ≤ $T_J$ ≤ 125°C | 105.5 | | 161 | 91.5 | | 141 | |
| Change of quiescent current(5) | 4 V ≤ $V_S$ ≤ 30 V, 25°C | 0.2 | 2 | | 0.2 | 2 | | µA |
| | 4 V ≤ $V_S$ ≤ 30 V, −40°C ≤ $T_J$ ≤ 125°C | 0.5 | | 3 | 0.5 | | 3 | |
| Temperature coefficient of quiescent current | −40°C ≤ $T_J$ ≤ 125°C | 0.39 | | 0.7 | 0.39 | | 0.7 | µA/°C |
| Minimum temperature for rate accuracy | In circuit of Figure 14, $I_L$ = 0 | 1.5 | | 2 | 1.5 | | 2 | °C |
| Long term stability | $T_J$ = $T_{MAX}$, for 1000 hours | ±0.08 | | | ±0.08 | | | °C |

(1) Tested Limits are ensured and 100% tested in production.
(2) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(3) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(4) Non-linearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(5) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.
(6) Quiescent current is defined in the circuit of Figure 14.

## 6.8 Electrical Characteristics: LM35, LM35C, LM35D

Unless otherwise noted, these specifications apply: −55°C ≤ $T_J$ ≤ 150°C for the LM35 and LM35A; −40°C ≤ $T_J$ ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ $T_J$ ≤ 100°C for the LM35D. $V_S$ = 5 Vdc and $I_{LOAD}$ = 50 µA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to $T_{MAX}$ in the circuit of Figure 14.

| PARAMETER | TEST CONDITIONS | | LM35 | | | LM35C, LM35D | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|
| | | | MIN | TYP | MAX | MIN | TYP | MAX | |
| Accuracy, LM35, LM35C[1] | $T_A$ = 25°C | | | ±0.4 | | | ±0.4 | | °C |
| | | Tested Limit[2] | | | ±1 | | | ±1 | |
| | | Design Limit[3] | | | | | | | |
| | $T_A$ = −10°C | | | ±0.5 | | | ±0.5 | | |
| | | Tested Limit[2] | | | | | | | |
| | | Design Limit[3] | | | | | | ±1.5 | |
| | $T_A$ = $T_{MAX}$ | | | ±0.8 | | | ±0.8 | | |
| | | Tested Limit[2] | | | ±1.5 | | | | |
| | | Design Limit[3] | | | | | | ±1.5 | |
| | $T_A$ = $T_{MIN}$ | | | ±0.8 | | | ±0.8 | | |
| | | Tested Limit[2] | | | | | | | |
| | | Design Limit[3] | | | ±1.5 | | | ±2 | |
| Accuracy, LM35D[1] | $T_A$ = 25°C | | | | | | ±0.6 | | °C |
| | | Tested Limit[2] | | | | | | ±1.5 | |
| | | Design Limit[3] | | | | | | | |
| | $T_A$ = $T_{MAX}$ | | | | | | ±0.9 | | |
| | | Tested Limit[2] | | | | | | | |
| | | Design Limit[3] | | | | | | ±2 | |
| | $T_A$ = $T_{MIN}$ | | | | | | ±0.9 | | |
| | | Tested Limit[2] | | | | | | | |
| | | Design Limit[3] | | | | | | ±2 | |
| Nonlinearity[4] | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | | | ±0.3 | | | ±0.2 | | °C |
| | | Tested Limit[2] | | | | | | | |
| | | Design Limit[3] | | | ±0.5 | | | ±0.5 | |
| Sensor gain (average slope) | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | | | 10 | | | 10 | | mV/°C |
| | | Tested Limit[2] | | | 9.8 | | | | |
| | | Design Limit[3] | | | | | | 9.8 | |
| | | | | 10 | | | 10 | | |
| | | Tested Limit[2] | | | 10.2 | | | | |
| | | Design Limit[3] | | | | | | 10.2 | |
| Load regulation[5] 0 ≤ $I_L$ ≤ 1 mA | $T_A$ = 25°C | | | ±0.4 | | | ±0.4 | | mV/mA |
| | | Tested Limit[2] | | | ±2 | | | ±2 | |
| | | Design Limit[3] | | | | | | | |
| | $T_{MIN}$ ≤ $T_A$ ≤ $T_{MAX}$, −40°C ≤ $T_J$ ≤ 125°C | | | ±0.5 | | | ±0.5 | | |
| | | Tested Limit[2] | | | | | | | |
| | | Design Limit[3] | | | ±5 | | | ±5 | |

(1) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(2) Tested Limits are ensured and 100% tested in production.
(3) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(4) Non-linearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(5) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

## Electrical Characteristics: LM35, LM35C, LM35D (continued)

Unless otherwise noted, these specifications apply: −55°C ≤ $T_J$ ≤ 150°C for the LM35 and LM35A; −40°C ≤ $T_J$ ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ $T_J$ ≤ 100°C for the LM35D. $V_S$ = 5 Vdc and $I_{LOAD}$ = 50 µA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to $T_{MAX}$ in the circuit of Figure 14.

| PARAMETER | TEST CONDITIONS | | LM35 | | | LM35C, LM35D | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|
| | | | MIN | TYP | MAX | MIN | TYP | MAX | |
| Line regulation (5) | $T_A$ = 25°C | | | ±0.01 | | | ±0.01 | | mV/V |
| | | Tested Limit (2) | | | ±0.1 | | | | |
| | | Design Limit (3) | | | | | | ±0.1 | |
| | 4 V ≤ $V_S$ ≤ 30 V, −40°C ≤ $T_J$ ≤ 125°C | | | ±0.02 | | | ±0.02 | | |
| | | Tested Limit (2) | | | | | | | |
| | | Design Limit (3) | | | ±0.2 | | | ±0.2 | |
| Quiescent current (6) | $V_S$ = 5 V, 25°C | | | 56 | | | 56 | | µA |
| | | Tested Limit (2) | | | 80 | | | 80 | |
| | | Design Limit (3) | | | | | | | |
| | $V_S$ = 5 V, −40°C ≤ $T_J$ ≤ 125°C | | | 105 | | | 91 | | |
| | | Tested Limit (2) | | | | | | | |
| | | Design Limit (3) | | | 158 | | | 138 | |
| | $V_S$ = 30 V, 25°C | | | 56.2 | | | 56.2 | | |
| | | Tested Limit (2) | | | 82 | | | 82 | |
| | | Design Limit (3) | | | | | | | |
| | $V_S$ = 30 V, −40°C ≤ $T_J$ ≤ 125°C | | | 105.5 | | | 91.5 | | |
| | | Tested Limit (2) | | | | | | | |
| | | Design Limit (3) | | | 161 | | | 141 | |
| Change of quiescent current (5) | 4 V ≤ $V_S$ ≤ 30 V, 25°C | | | 0.2 | | | 0.2 | | µA |
| | | Tested Limit (2) | | | | | | 2 | |
| | | Design Limit (3) | | | 2 | | | | |
| | 4 V ≤ $V_S$ ≤ 30 V, −40°C ≤ $T_J$ ≤ 125°C | | | 0.5 | | | 0.5 | | |
| | | Tested Limit (2) | | | | | | | |
| | | Design Limit (3) | | | 3 | | | 3 | |
| Temperature coefficient of quiescent current | −40°C ≤ $T_J$ ≤ 125°C | | | 0.39 | | | 0.39 | | µA/°C |
| | | Tested Limit (2) | | | | | | | |
| | | Design Limit (3) | | | 0.7 | | | 0.7 | |
| Minimum temperature for rate accuracy | In circuit of Figure 14, $I_L$ = 0 | | | 1.5 | | | 1.5 | | °C |
| | | Tested Limit (2) | | | | | | | |
| | | Design Limit (3) | | | 2 | | | 2 | |
| Long term stability | $T_J$ = $T_{MAX}$, for 1000 hours | | | ±0.08 | | | ±0.08 | | °C |

(6) Quiescent current is defined in the circuit of Figure 14.

# LP2985 150-mA Low-noise Low-dropout Regulator With Shutdown

## 1 Features

- Output Tolerance of
  - 1% (A Grade)
  - 1.5% (Standard Grade)
- Ultra-Low Dropout, Typically
  - 280 mV at Full Load of 150 mA
  - 7 mV at 1 mA
- Wide $V_{IN}$ Range: 16 V Max
- Low $I_Q$: 850 µA at Full Load at 150 mA
- Shutdown Current: 0.01 µA Typ
- Low Noise: 30 $\mu V_{RMS}$ With 10-nF Bypass Capacitor
- Stable With Low-ESR Capacitors, Including Ceramic
- Overcurrent and Thermal Protection
- High Peak-Current Capability
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
  - 200-V Machine Model (A115-A)

## 2 Applications

- Portable Devices
- Digital Cameras and Camcorders
- CD Players
- MP3 Players

## 3 Description

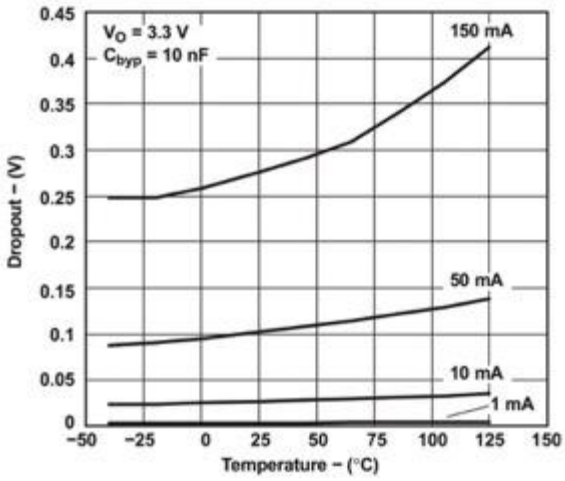
The LP2985 family of fixed-output, low-dropout regulators offers exceptional, cost-effective performance for both portable and nonportable applications. Available in voltages of 1.8 V, 2.5 V, 2.8 V, 2.9 V, 3 V, 3.1 V, 3.3 V, 5 V, and 10 V, the family has an output tolerance of 1% for the A version (1.5% for the non-A version) and is capable of delivering 150-mA continuous load current. Standard regulator features, such as overcurrent and overtemperature protection, are included.

**Device Information**(1)

| PART NUMBER | PACKAGE | BODY SIZE (NOM) |
|---|---|---|
| LP2985 | SOT-23 (5) | 2.90 mm x 1.60 mm |

(1) For all available packages, see the orderable addendum at the end of the data sheet.

**Dropout Voltage vs Temperature**



## Table of Contents



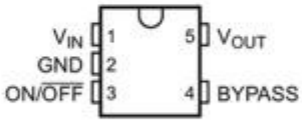
## 4 Revision History

**Changes from Revision N (June 2011) to Revision O** **Page**

- Added *Applications*, *Device Information* table, *Pin Functions* table, *ESD Ratings* table, *Thermal Information* table, *Feature Description* section, *Device Functional Modes*, *Application and Implementation* section, *Power Supply Recommendations* section, *Layout* section, *Device and Documentation Support* section, and *Mechanical, Packaging, and Orderable Information* section. ..... 1
- Deleted *Ordering Information* table. ..... 1

## 5 Pin Configuration and Functions

DBV (SOT-23) PACKAGE
(TOP VIEW)




**Pin Functions**

| PIN | | TYPE | DESCRIPTION |
|---|---|---|---|
| NAME | NO. | | |
| BYPASS | 4 | I/O | Attach a 10-nF capacitor to improve low-noise performance. |
| GND | 2 | — | Ground |
| ON/$\overline{OFF}$ | 3 | I | Active-low shutdown pin. Tie to $V_{IN}$ if unused. |
| $V_{IN}$ | 1 | I | Supply input |
| $V_{OUT}$ | 5 | O | Voltage output |

# 6 Specifications

## 6.1 Absolute Maximum Ratings

over virtual junction temperature range (unless otherwise noted)(1)

| | | MIN | MAX | UNIT |
|---|---|---|---|---|
| $V_{IN}$ | Continuous input voltage range(2) | –0.3 | 16 | V |
| $V_{ON/\overline{OFF}}$ | ON/$\overline{OFF}$ input voltage range | –0.3 | 16 | V |
| | Output voltage range(3) | –0.3 | 9 | V |
| $I_O$ | Output current(4) | Internally limited (short-circuit protected) | | — |
| $\theta_{JA}$ | Package thermal impedance(4) (5) | | 206 | °C/W |
| $T_J$ | Operating virtual junction temperature | | 150 | °C |
| $T_{stg}$ | Storage temperature range | –65 | 150 | °C |

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions* is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The PNP pass transistor has a parasitic diode connected between the input and output. This diode normally is reverse biased ($V_{IN} > V_{OUT}$), but will be forward biased if the output voltage exceeds the input voltage by a diode drop (see *Application Information* for more details).
(3) If load is returned to a negative power supply in a dual-supply system, the output must be diode clamped to GND.
(4) Maximum power dissipation is a function of $T_J(max)$, $\theta_{JA}$, and $T_A$. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) – T_A)/\theta_{JA}$. Operating at the absolute maximum $T_J$ of 150°C can affect reliability.
(5) The package thermal impedance is calculated in accordance with JESD 51-7.

## 6.2 ESD Ratings

| | | | VALUE | UNIT |
|---|---|---|---|---|
| $V_{(ESD)}$ | Electrostatic discharge | Human body model (HBM), per ANSI/ESDA/JEDEC JS-001, all pins(1) | 2000 | V |
| | | Charged device model (CDM), per JEDEC specification JESD22-C101, all pins(2) | 1000 | |

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

## 6.3 Recommended Operating Conditions

| | | MIN | MAX | UNIT |
|---|---|---|---|---|
| $V_{IN}$ | Supply input voltage | 2.2(1) | 16 | V |
| $V_{ON/\overline{OFF}}$ | ON/$\overline{OFF}$ input voltage | 0 | $V_{IN}$ | V |
| $I_{OUT}$ | Output current | | 150 | mA |
| $T_J$ | Virtual junction temperature | –40 | 125 | °C |

(1) Recommended minimum $V_{IN}$ is the greater of 2.5 V or $V_{OUT(max)}$ + rated dropout voltage (max) for operating $I_L$.

## 6.4 Thermal Information

| THERMAL METRIC(1) | | LP2985 DBV 5 PINS | UNIT |
|---|---|---|---|
| $R_{\theta JA}$ | Junction-to-ambient thermal resistance | 206 | °C/W |

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.

## 6.5 Electrical Characteristics

at specified virtual junction temperature range, $V_{IN} = V_{OUT(NOM)} + 1$ V, $V_{ON/\overline{OFF}} = 2$ V, $C_{IN} = 1$ µF, $I_L = 1$ mA, $C_{OUT} = 4.7$ µF (unless otherwise noted)

| PARAMETER | | TEST CONDITIONS | $T_J$ | LP2985A-xx | | | LP2985-xx | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | MIN | TYP | MAX | MIN | TYP | MAX | |
| $\Delta V_{OUT}$ | Output voltage tolerance | $I_L$ = 1 mA | 25°C | –1 | | 1 | –1.5 | | 1.5 | %$V_{NOM}$ |
| | | 1 mA ≤ $I_L$ ≤ 50 mA | 25°C | –1.5 | | 1.5 | –2.5 | | 2.5 | |
| | | | –40°C to 125°C | –2.5 | | 2.5 | –3.5 | | 3.5 | |
| | | 1 mA ≤ $I_L$ ≤ 150 mA | 25°C | –2.5 | | 2.5 | –3 | | 3 | |
| | | | –40°C to 125°C | –3.5 | | 3.5 | –4 | | 4 | |
| | Line regulation | $V_{IN}$ = [$V_{OUT(NOM)}$ + 1 V] to 16 V | 25°C | | 0.007 | 0.014 | | 0.007 | 0.014 | %/V |
| | | | –40°C to 125°C | | | 0.032 | | | 0.032 | |
| $V_{IN} - V_{OUT}$ | Dropout voltage[1] | $I_L$ = 0 | 25°C | | 1 | 3 | | 1 | 3 | mV |
| | | | –40°C to 125°C | | | 5 | | | 5 | |
| | | $I_L$ = 1 mA | 25°C | | 7 | 10 | | 7 | 10 | |
| | | | –40°C to 125°C | | | 15 | | | 15 | |
| | | $I_L$ = 10 mA | 25°C | | 40 | 60 | | 40 | 60 | |
| | | | –40°C to 125°C | | | 90 | | | 90 | |
| | | $I_L$ = 50 mA | 25°C | | 120 | 150 | | 120 | 150 | |
| | | | –40°C to 125°C | | | 225 | | | 225 | |
| | | $I_L$ = 150 mA | 25°C | | 280 | 350 | | 280 | 350 | |
| | | | –40°C to 125°C | | | 575 | | | 575 | |
| $I_{GND}$ | GND pin current | $I_L$ = 0 | 25°C | | 65 | 95 | | 65 | 95 | µA |
| | | | 25°C (LP2985-10) | | | 125 | | | 125 | |
| | | | –40°C to 125°C | | | 125 | | | 125 | |
| | | | –40°C to 125°C (LP2985-10) | | | 160 | | | 160 | |
| | | $I_L$ = 1 mA | 25°C | | 75 | 110 | | 75 | 110 | |
| | | | 25°C (LP2985-10) | | | 140 | | | 140 | |
| | | | –40°C to 125°C | | | 170 | | | 170 | |
| | | $I_L$ = 10 mA | 25°C | | 120 | 220 | | 120 | 220 | |
| | | | 25°C (LP2985-10) | | | 250 | | | 250 | |
| | | | –40°C to 125°C | | | 400 | | | 400 | |
| | | $I_L$ = 50 mA | 25°C | | 350 | 600 | | 350 | 600 | |
| | | | 25°C (LP2985-10) | | | 650 | | | 650 | |
| | | | –40°C to 125°C | | | 1000 | | | 1000 | |
| | | $I_L$ = 150 mA | 25°C | | 850 | 1500 | | 850 | 1500 | |
| | | | 25°C (LP2985-10) | | | 1800 | | | 1800 | |
| | | | –40°C to 125°C | | | 2500 | | | 2500 | |
| | | $V_{ON/\overline{OFF}}$ < 0.3 V (OFF) | 25°C | | 0.01 | 0.8 | | 0.01 | 0.8 | |
| | | $V_{ON/\overline{OFF}}$ < 0.15 V (OFF) | –40°C to 105°C | | 0.05 | 2 | | 0.05 | 2 | |
| | | | –40°C to 125°C | | | 5 | | | 5 | |
| $V_{ON/\overline{OFF}}$ | ON/$\overline{OFF}$ input voltage[2] | $V_{ON/\overline{OFF}}$ = HIGH → O/P ON | 25°C | | 1.4 | | | 1.4 | | V |
| | | | –40°C to 125°C | 1.6 | | | 1.6 | | | |
| | | $V_{ON/\overline{OFF}}$ = LOW → O/P OFF | 25°C | | 0.55 | | | 0.55 | | |
| | | | –40°C to 125°C | | | 0.15 | | | 0.15 | |
| $I_{ON/\overline{OFF}}$ | ON/$\overline{OFF}$ input current | $V_{ON/\overline{OFF}}$ = 0 | 25°C | | 0.01 | | | 0.01 | | µA |
| | | | –40°C to 125°C | | | –2 | | | –2 | |
| | | $V_{ON/\overline{OFF}}$ = 5 V | 25°C | | 5 | | | 5 | | |
| | | | –40°C to 125°C | | | 15 | | | 15 | |

(1) Dropout voltage is defined as the input-to-output differential at which the output voltage drops 100 mV below the value measured with a 1-V differential.
(2) The ON/$\overline{OFF}$ input must be driven properly for reliable operation (see *Application Information*).

## Electrical Characteristics (continued)

at specified virtual junction temperature range, $V_{IN}$ = $V_{OUT(NOM)}$ + 1 V, $V_{ON/\overline{OFF}}$ = 2 V, $C_{IN}$ = 1 µF, $I_L$ = 1 mA, $C_{OUT}$ = 4.7 µF (unless otherwise noted)

| PARAMETER | | TEST CONDITIONS | $T_J$ | LP2985A-xx | | | LP2985-xx | | | UNIT |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | MIN | TYP | MAX | MIN | TYP | MAX | |
| $V_n$ | Output noise (RMS) | BW = 300 Hz to 50 kHz, $C_{OUT}$ = 10 µF, $C_{BYPASS}$ = 10 nF | 25°C | | 30 | | | 30 | | µV |
| $\frac{\Delta V_{OUT}}{\Delta V_{IN}}$ | Ripple rejection | f = 1kHz, $C_{OUT}$ = 10 µF, $C_{BYPASS}$ = 10 nF | 25°C | | 45 | | | 45 | | dB |
| $I_{OUT(PK)}$ | Peak output current | $V_{OUT} \geq V_{O(NOM)}$ – 5% | 25°C | | 350 | | | 350 | | mA |
| $I_{OUT(SC)}$ | Short-circuit current | $R_L$ = 0 (steady state) (3) | 25°C | | 400 | | | 400 | | mA |

(3) See Figure 6 in *Typical Performance Characteristics*.

ANALOG DEVICES

# Single and Dual-Supply, Rail-to-Rail, Low Cost Instrumentation Amplifier

Data Sheet **AD623**

## FEATURES

- Easy to use
- Rail-to-rail output swing
- Input voltage range extends 150 mV below ground (single supply)
- Low power, 550 µA maximum supply current
- Gain set with one external resistor
  - Gain range: 1 to 1000
- High accuracy dc performance
  - 0.10% gain accuracy (G = 1)
  - 0.35% gain accuracy (G > 1)
- Noise: 35 nV/√Hz RTI noise at 1 kHz
- Excellent dynamic specifications
  - 800 kHz bandwidth (G = 1)
  - 20 µs settling time to 0.01% (G = 10)

## APPLICATIONS

- Low power medical instrumentation
- Transducer interfaces
- Thermocouple amplifiers
- Industrial process controls
- Difference amplifiers
- Low power data acquisition

## GENERAL DESCRIPTION

The AD623 is an integrated, single- or dual-supply instrumentation amplifier that delivers rail-to-rail output swing using supply voltages from 3 V to 12 V. The AD623 offers superior user flexibility by allowing single gain set resistor programming and by conforming to the 8-lead industry standard pinout configuration. With no external resistor, the AD623 is configured for unity gain (G = 1), and with an external resistor, the AD623 can be programmed for gains of up to 1000.

The superior accuracy of the AD623 is the result of increasing ac common-mode rejection ratio (CMRR) coincident with increasing gain; line noise harmonics are rejected due to constant CMRR up to 200 Hz. The AD623 has a wide input common-mode range and amplifies signals with common-mode voltages as low as 150 mV below ground. The AD623 maintains superior performance with dual and single polarity power supplies.

Table 1. Low Power Upgrades for the AD623

| Part No. | Total $V_S$ (V dc) | Typical $I_Q$ (µA) |
|---|---|---|
| AD8235 | 5.5 | 30 |
| AD8236 | 5.5 | 33 |
| AD8237 | 5.5 | 33 |
| AD8226 | 36 | 350 |
| AD8227 | 36 | 325 |
| AD8420 | 36 | 85 |
| AD8422 | 36 | 300 |
| AD8426 | 36 | 325 (per channel) |

## FUNCTIONAL BLOCK DIAGRAM

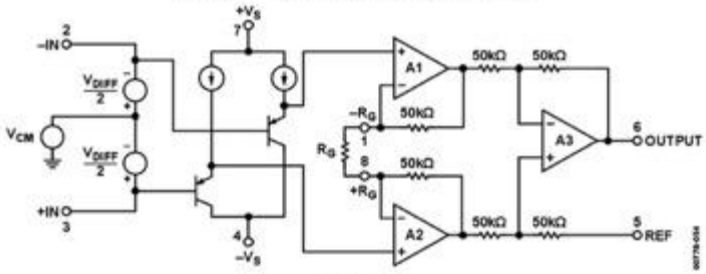



Figure 1.

# SPECIFICATIONS

## SINGLE SUPPLY

Typical at 25°C, single supply, $+V_S$ = 5 V, $-V_S$ = 0 V, and $R_L$ = 10 kΩ, unless otherwise noted.

**Table 2.**

| Parameter | Test Conditions/ Comments | AD623A | | | AD623ARM | | | AD623B | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Typ | Max | Min | Typ | Max | Min | Typ | Max | |
| GAIN | G = 1 + (100 k/$R_G$) | | | | | | | | | | |
| Gain Range | | 1 | | 1000 | 1 | | 1000 | 1 | | 1000 | |
| Gain Error[1] | G1 $V_{OUT}$ = 0.05 V to 3.5 V<br>G > 1 $V_{OUT}$ = 0.05 V to 4.5 V | | | | | | | | | | |
| G = 1 | | | 0.03 | 0.10 | | 0.03 | 0.10 | | 0.03 | 0.05 | % |
| G = 10 | | | 0.10 | 0.35 | | 0.10 | 0.35 | | 0.10 | 0.35 | % |
| G = 100 | | | 0.10 | 0.35 | | 0.10 | 0.35 | | 0.10 | 0.35 | % |
| G = 1000 | | | 0.10 | 0.35 | | 0.10 | 0.35 | | 0.10 | 0.35 | % |
| Nonlinearity | G1 $V_{OUT}$ = 0.05 V to 3.5 V<br>G > 1 $V_{OUT}$ = 0.05 V to 4.5 V | | | | | | | | | | |
| G = 1 to 1000 | | | 50 | | | 50 | | | 50 | | ppm |
| Gain vs. Temperature | | | | | | | | | | | |
| G = 1 | | | 5 | 10 | | 5 | 10 | | 5 | 10 | ppm/°C |
| G > 1[1] | | | 50 | | | 50 | | | 50 | | ppm/°C |
| VOLTAGE OFFSET | Total RTI error = $V_{OSI}$ + $V_{OSO}$/G | | | | | | | | | | |
| Input Offset, $V_{OSI}$ | | | 25 | 200 | | 200 | 500 | | 25 | 100 | µV |
| Over Temperature | | | | 350 | | | 650 | | | 160 | µV |
| Average Temperature Coefficient (Tempco) | | | 0.1 | 2 | | 0.1 | 2 | | 0.1 | 1 | µV/°C |
| Output Offset, $V_{OSO}$ | | | 200 | 1000 | | 500 | 2000 | | 200 | 500 | µV |
| Over Temperature | | | | 1500 | | | 2600 | | | 1100 | µV |
| Average Tempco | | | 2.5 | 10 | | 2.5 | 10 | | 2.5 | 10 | µV/°C |
| Offset Referred to the Input vs. Supply (PSR) | | | | | | | | | | | |
| G = 1 | | 80 | 100 | | 80 | 100 | | 80 | 100 | | dB |
| G = 10 | | 100 | 120 | | 100 | 120 | | 100 | 120 | | dB |
| G = 100 | | 120 | 140 | | 120 | 140 | | 120 | 140 | | dB |
| G = 1000 | | 120 | 140 | | 120 | 140 | | 120 | 140 | | dB |
| INPUT CURRENT | | | | | | | | | | | |
| Input Bias Current | | | 17 | 25 | | 17 | 25 | | 17 | 25 | nA |
| Over Temperature | | | | 27.5 | | | 27.5 | | | 27.5 | nA |
| Average Tempco | | | 25 | | | 25 | | | 25 | | pA/°C |
| Input Offset Current | | | 0.25 | 2 | | 0.25 | 2 | | 0.25 | 2 | nA |
| Over Temperature | | | | 2.5 | | | 2.5 | | | 2.5 | nA |
| Average Tempco | | | 5 | | | 5 | | | 5 | | pA/°C |
| INPUT | | | | | | | | | | | |
| Input Impedance | | | | | | | | | | | |
| Differential | | | 2\|\|2 | | | 2\|\|2 | | | 2\|\|2 | | GΩ\|\|pF |
| Common-Mode | | | 2\|\|2 | | | 2\|\|2 | | | 2\|\|2 | | GΩ\|\|pF |
| Input Voltage Range[2] | $V_S$ = 3 V to 12 V | ($-V_S$) – 0.15 | | ($+V_S$) – 1.5 | ($-V_S$) – 0.15 | | ($+V_S$) – 1.5 | ($-V_S$) – 0.15 | | ($+V_S$) – 1.5 | V |

| Parameter | Test Conditions/ Comments | AD623A Min | AD623A Typ | AD623A Max | AD623ARM Min | AD623ARM Typ | AD623ARM Max | AD623B Min | AD623B Typ | AD623B Max | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Common-Mode Rejection at 60 Hz with 1 kΩ Source Imbalance | | | | | | | | | | | |
| G = 1 | $V_{CM}$ = 0 V to 3 V | 70 | 80 | | 70 | 80 | | 77 | 86 | | dB |
| G = 10 | $V_{CM}$ = 0 V to 3 V | 90 | 100 | | 90 | 100 | | 94 | 100 | | dB |
| G = 100 | $V_{CM}$ = 0 V to 3 V | 105 | 110 | | 105 | 110 | | 105 | 110 | | dB |
| G = 1000 | $V_{CM}$ = 0 V to 3 V | 105 | 110 | | 105 | 110 | | 105 | 110 | | dB |
| OUTPUT | | | | | | | | | | | |
| Output Swing | $R_L$ = 10 kΩ | 0.01 | | (+$V_S$) – 0.5 | 0.01 | | (+$V_S$) – 0.5 | 0.01 | | (+$V_S$) – 0.5 | V |
| | $R_L$ = 100 kΩ | 0.01 | | (+$V_S$) – 0.15 | 0.01 | | (+$V_S$) – 0.15 | 0.01 | | (+$V_S$) – 0.15 | V |
| DYNAMIC RESPONSE | | | | | | | | | | | |
| Small Signal –3 dB BW | | | | | | | | | | | |
| G = 1 | | | 800 | | | 800 | | | 800 | | kHz |
| G = 10 | | | 100 | | | 100 | | | 100 | | kHz |
| G = 100 | | | 10 | | | 10 | | | 10 | | kHz |
| G = 1000 | | | 2 | | | 2 | | | 2 | | kHz |
| Slew Rate | | | 0.3 | | | 0.3 | | | 0.3 | | V/µs |
| Settling Time to 0.01% | $V_S$ = 5 V | | | | | | | | | | |
| G = 1 | Step size: 3.5 V | | 30 | | | 30 | | | 30 | | µs |
| G = 10 | Step size: 4 V, $V_{CM}$ = 1.8 V | | 20 | | | 20 | | | 20 | | µs |

[1] Does not include effects of external resistor, $R_G$.
[2] One input grounded. G = 1.

## DUAL SUPPLIES

Typical at 25°C dual supply, $V_S$ = ±5 V, and $R_L$ = 10 kΩ, unless otherwise noted.

**Table 3.**

| Parameter | Test Conditions/ Comments | AD623A Min | AD623A Typ | AD623A Max | AD623ARM Min | AD623ARM Typ | AD623ARM Max | AD623B Min | AD623B Typ | AD623B Max | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| GAIN | G = 1 + (100 k/$R_G$) | | | | | | | | | | |
| Gain Range | | 1 | | 1000 | 1 | | 1000 | 1 | | 1000 | |
| Gain Error[1] | G1 $V_{OUT}$ = −4.8 V to +3.5 V<br>G > 1 $V_{OUT}$ = 0.05 V to 4.5 V | | | | | | | | | | |
| G = 1 | | | 0.03 | 0.10 | | 0.03 | 0.10 | | 0.03 | 0.05 | % |
| G = 10 | | | 0.10 | 0.35 | | 0.10 | 0.35 | | 0.10 | 0.35 | % |
| G = 100 | | | 0.10 | 0.35 | | 0.10 | 0.35 | | 0.10 | 0.35 | % |
| G = 1000 | | | 0.10 | 0.35 | | 0.10 | 0.35 | | 0.10 | 0.35 | % |
| Nonlinearity | G1 $V_{OUT}$ = −4.8 V to +3.5 V<br>G > 1 $V_{OUT}$ = −4.8 V to +4.5 V | | | | | | | | | | |
| G = 1 to 1000 | | | 50 | | | 50 | | | 50 | | ppm |
| Gain vs. Temperature | | | | | | | | | | | |
| G = 1 | | | 5 | 10 | | 5 | 10 | | 5 | 10 | ppm/°C |
| G > 1[1] | | | 50 | | | 50 | | | 50 | | ppm/°C |
| VOLTAGE OFFSET | Total RTI error = $V_{OSI}$ + $V_{OSO}$/G | | | | | | | | | | |
| Input Offset, $V_{OSI}$ | | | 25 | 200 | | 200 | 500 | | 25 | 100 | µV |
| Over Temperature | | | | 350 | | | 650 | | | 160 | µV |
| Average Tempco | | | 0.1 | 2 | | 0.1 | 2 | | 0.1 | 1 | µV/°C |
| Output Offset, $V_{OSO}$ | | | 200 | 1000 | | 500 | 2000 | | 200 | 500 | µV |
| Over Temperature | | | | 1500 | | | 2600 | | | 1100 | µV |
| Average Tempco | | | 2.5 | 10 | | 2.5 | 10 | | 2.5 | 10 | µV/°C |
| Offset Referred to the Input vs. Supply (PSR) | | | | | | | | | | | |
| G = 1 | | 80 | 100 | | 80 | 100 | | 80 | 100 | | dB |
| G = 10 | | 100 | 120 | | 100 | 120 | | 100 | 120 | | dB |
| G = 100 | | 120 | 140 | | 120 | 140 | | 120 | 140 | | dB |
| G = 1000 | | 120 | 140 | | 120 | 140 | | 120 | 140 | | dB |
| INPUT CURRENT | | | | | | | | | | | |
| Input Bias Current | | | 17 | 25 | | 17 | 25 | | 17 | 25 | nA |
| Over Temperature | | | | 27.5 | | | 27.5 | | | 27.5 | nA |
| Average Tempco | | | 25 | | | 25 | | | 25 | | pA/°C |
| Input Offset Current | | | 0.25 | 2 | | 0.25 | 2 | | 0.25 | 2 | nA |
| Over Temperature | | | | 2.5 | | | 2.5 | | | 2.5 | nA |
| Average Tempco | | | 5 | | | 5 | | | 5 | | pA/°C |
| INPUT | | | | | | | | | | | |
| Input Impedance | | | | | | | | | | | |
| Differential | | | 2\|\|2 | | | 2\|\|2 | | | 2\|\|2 | | GΩ\|\|pF |
| Common-Mode | | | 2\|\|2 | | | 2\|\|2 | | | 2\|\|2 | | GΩ\|\|pF |
| Input Voltage Range[2] | $V_S$ = +2.5 V to ±6 V | (−$V_S$) − 0.15 | | (+$V_S$) − 1.5 | (−$V_S$) − 0.15 | | (+$V_S$) − 1.5 | (−$V_S$) − 0.15 | | (+$V_S$) − 1.5 | V |

| Parameter | Test Conditions/ Comments | AD623A | | | AD623ARM | | | AD623B | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Typ | Max | Min | Typ | Max | Min | Typ | Max | |
| Common-Mode Rejection at 60 Hz with 1 kΩ Source Imbalance | | | | | | | | | | | |
| G = 1 | $V_{CM}$ = +3.5 V to −5.15 V | 70 | 80 | | 70 | 80 | | 77 | 86 | | dB |
| G = 10 | $V_{CM}$ = +3.5 V to −5.15 V | 90 | 100 | | 90 | 100 | | 94 | 100 | | dB |
| G = 100 | $V_{CM}$ = +3.5 V to −5.15 V | 105 | 110 | | 105 | 110 | | 105 | 110 | | dB |
| G = 1000 | $V_{CM}$ = +3.5 V to −5.15 V | 105 | 110 | | 105 | 110 | | 105 | 110 | | dB |
| OUTPUT | | | | | | | | | | | |
| Output Swing | $R_L$ = 10 kΩ, $V_S$ = ±5 V | (−$V_S$) + 0.2 | | (+$V_S$) − 0.5 | (−$V_S$) + 0.2 | | (+$V_S$) − 0.5 | (−$V_S$) + 0.2 | | (+$V_S$) − 0.5 | V |
| | $R_L$ = 100 kΩ | (−$V_S$) + 0.05 | | (+$V_S$) − 0.15 | (−$V_S$) + 0.05 | | (+$V_S$) − 0.15 | (−$V_S$) + 0.05 | | (+$V_S$) − 0.15 | V |
| DYNAMIC RESPONSE | | | | | | | | | | | |
| Small Signal −3 dB Bandwidth | | | | | | | | | | | |
| G = 1 | | | 800 | | | 800 | | | 800 | | kHz |
| G = 10 | | | 100 | | | 100 | | | 100 | | kHz |
| G = 100 | | | 10 | | | 10 | | | 10 | | kHz |
| G = 1000 | | | 2 | | | 2 | | | 2 | | kHz |
| Slew Rate | | | 0.3 | | | 0.3 | | | 0.3 | | V/µs |
| Settling Time to 0.01% | $V_S$ = ±5 V, 5 V step | | | | | | | | | | |
| G = 1 | | | 30 | | | 30 | | | 30 | | µs |
| G = 10 | | | 20 | | | 20 | | | 20 | | µs |

[1] Does not include effects of external resistor, $R_G$.
[2] One input grounded. G = 1.

## SPECIFICATIONS COMMON TO DUAL AND SINGLE SUPPLIES

**Table 4.**

| Parameter | Test Conditions/ Comments | AD623A | | | AD623ARM | | | AD623B | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Typ | Max | Min | Typ | Max | Min | Typ | Max | |
| NOISE | | | | | | | | | | | |
| Voltage Noise, 1 kHz | Total RTI noise = $\sqrt{(e_{ni})^2 + (2e_{no}/G)^2}$ | | | | | | | | | | |
| Input, Voltage Noise, $e_N$ | | | 35 | | | 35 | | | 35 | | nV/√Hz |
| Output, Voltage Noise, $e_{no}$ | | | 50 | | | 50 | | | 50 | | nV/√Hz |
| RTI, 0.1 Hz to 10 Hz | | | | | | | | | | | |
| G = 1 | | | 3.0 | | | 3.0 | | | 3.0 | | µV p-p |
| G = 1000 | | | 1.5 | | | 1.5 | | | 1.5 | | µV p-p |
| Current Noise | f = 1 kHz | | 100 | | | 100 | | | 100 | | fA/√Hz |
| 0.1 Hz to 10 Hz | | | 1.5 | | | 1.5 | | | 1.5 | | pA p-p |
| REFERENCE INPUT | | | | | | | | | | | |
| $R_{IN}$ | | | 100 ± 20% | | | 100 ± 20% | | | 100 ± 20% | | kΩ |
| $I_{IN}$ | $V_{IN}$+, $V_{REF}$ = 0 V | | 50 | 60 | | 50 | 60 | | 50 | 60 | µA |
| Voltage Range | | $-V_S$ | | $+V_S$ | $-V_S$ | | $+V_S$ | $-V_S$ | | $+V_S$ | V |
| Gain to Output | | | 1 ± 0.0002 | | | 1 ± 0.0002 | | | 1 ± 0.0002 | | V |
| POWER SUPPLY | | | | | | | | | | | |
| Operating Range | Dual supply | ±2.5 | | ±6 | ±2.5 | | ±6 | ±2.5 | | ±6 | V |
| | Single supply | 2.7 | | 12 | 2.7 | | 12 | 2.7 | | 12 | V |
| Quiescent Current | Dual supply | | 375 | 550 | | 375 | 550 | | 375 | 550 | µA |
| | Single supply | | 305 | 480 | | 305 | 480 | | 305 | 480 | µA |
| Over Temperature | | | | 625 | | | 625 | | | 625 | µA |
| TEMPERATURE RANGE | | | | | | | | | | | |
| For Specified Performance | | –40 | | +85 | –40 | | +85 | –40 | | +85 | °C |

19-1542; Rev 0; 10/99

MAXIM

# Rail-to-Rail, Fault-Protected, SPST Analog Switches



## General Description

The MAX4510/MAX4520 single-pole/single-throw (SPST), fault-protected analog switches feature a fault-protected input and Rail-to-Rail® signal-handling capability. The normally open (NO) and normally closed (NC) terminals are protected from overvoltage faults up to 36V during power-on and 44V with power off. During a fault condition, the switch input terminal (NO or NC) becomes an open circuit; only nanoamperes of leakage current flow from the fault source, and the switch output (COM) furnishes up to 13mA of the appropriate polarity supply voltage to the load. This ensures unambiguous rail-to-rail outputs when a fault begins and ends.

On-resistance is 160Ω max. The off-leakage current is only 0.5nA at +25°C and 10nA at +85°C. The MAX4510 is a normally closed switch, while the MAX4520 is a normally open switch. These CMOS switches operate with dual power supplies ranging from ±4.5V to ±20V or a single supply between +9V and +36V.

The digital input has +0.8V and +2.4V logic thresholds, ensuring both TTL- and CMOS-logic compatibility when using ±15V or a single +12V supply. The MAX4510/MAX4520 are available in 6-pin SOT23 and 8-pin µMAX packages.

## Applications

Data Acquisition
Industrial and Process-Control Systems
Avionics
ATE Equipment
Redundant/Backup Systems

## Ordering Information

| PART | TEMP. RANGE | PIN-PACKAGE | TOP MARK |
|---|---|---|---|
| **MAX4510**EUT-T | -40°C to +85°C | 6 SOT23-6 | AABZ |
| MAX4510EUA | -40°C to +85°C | 8 µMAX | — |
| **MAX4520**EUT-T | -40°C to +85°C | 6 SOT23-6 | AADK |
| MAX4520EUA | -40°C to +85°C | 8 µMAX | — |

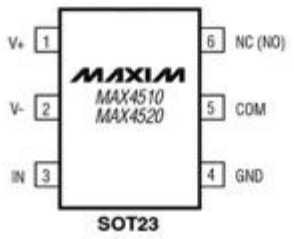
*Rail-to-Rail is a registered trademark of Nippon Motorola, Ltd.*

## Features

- ♦ **±40V Fault Protection with Power Off**
  **±36V Fault Protection with ±15V Supplies**
- ♦ **Switch is Off with Power Removed**
- ♦ **Rail-to-Rail Signal Handling**
- ♦ **160Ω max Signal Paths with ±15V Supplies**
- ♦ **On-Switch Turns Off with Overvoltage**
- ♦ **0.5nA Off-Leakage Current**
- ♦ **Output Clamped to Appropriate Supply Voltage During Fault Condition; No Transition Glitch**
- ♦ **No Power-Supply Sequencing Required**
- ♦ **±4.5V to ±20V Dual Supplies**
  **+9V to +36V Single Supply**
- ♦ **Low Power Consumption: <2mW**
- ♦ **TTL- and CMOS-Compatible Logic Inputs with Single +9V to +15V or ±15V Supplies**

## Pin Configurations/Truth Tables

TOP VIEW




SOT23

( ) ARE FOR MAX4520.

| IN | MAX4510 | MAX4520 |
|---|---|---|
| 0 | ON | OFF |
| 1 | OFF | ON |

SWITCHES SHOWN FOR LOGIC "0" INPUT.
ALL SWITCHES ARE OFF WITH POWER REMOVED.

*Pin Configurations continued at end of data sheet.*

# Rail-to-Rail, Fault-Protected, SPST Analog Switches

## ABSOLUTE MAXIMUM RATINGS

(Voltages Referenced to GND)
V+ .....-0.3V to +44.0V
V- .....-44.0V to +0.3V
V+ to V- .....-0.3V to +44.0V
COM, IN (Note 1) .....(V- - 0.3V) to (V+ + 0.3V)
NC, NO (Note 2) .....(V+ - 36V) to (V- + 36V)
NC, NO to COM .....-36V to +36V
Continuous Current into Any Terminal .....±30mA
Peak Current into Any Terminal
(pulsed at 1ms, 10% duty cycle) .....±50mA

Continuous Power Dissipation ($T_A$ = +70°C) (Note 2)
6-Pin SOT23-6 (derate 7.1mW/°C above +70°C) .....696mW
8-Pin µMAX (derate 4.10mW/°C above +70°C) .....330mW
Operating Temperature Ranges
MAX45_0EUT .....-40°C to +85°C
MAX45_0EUA .....-40°C to +85°C
Storage Temperature Range .....-65°C to +150°C
Lead Temperature (soldering, 10sec) .....+300°C

**Note 1:** COM and IN pins are not fault protected. Signals on COM or IN exceeding V+ or V- are clamped by internal diodes. Limit forward diode current to maximum current rating.
**Note 2:** NC and NO pins are fault protected. Signals on NC or NO exceeding -36V to +36V may damage the device. These limits apply with power applied to V+ or V-, or ±40V with V+ = V- = 0.

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

## ELECTRICAL CHARACTERISTICS—Dual Supplies

(V+ = +15V, V- = -15V, GND = 0, $V_{IH}$ = 2.4V, $V_{IL}$ = 0.8V, $T_A$ = $T_{MIN}$ to $T_{MAX}$, unless otherwise noted. Typical values are at $T_A$ = +25°C.) (Note 3)

| PARAMETER | SYMBOL | CONDITIONS | $T_A$ | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|---|
| **ANALOG SWITCH** | | | | | | | |
| Fault-Free Analog Signal Range | $V_{NO}$, $V_{NC}$ | Applies with power on or off | E | V- | | V+ | V |
| On-Resistance | $R_{ON}$ | $V_{COM}$ = ±10V, $I_{COM}$ = 1mA | +25°C | | 125 | 160 | Ω |
| | | | E | | | 225 | |
| NO or NC Off-Leakage Current (Notes 4, 5) | $I_{NO(OFF)}$, $I_{NC(OFF)}$ | $V_{COM}$ = ±14V; $V_{NO}$, $V_{COM}$ = ∓14V | +25°C | -0.5 | 0.01 | 0.5 | nA |
| | | | E | -10 | | 10 | |
| COM Off-Leakage Current (Notes 4, 5) | $I_{COM(OFF)}$ | $V_{COM}$ = ±14V; $V_{NO}$, $V_{NC}$ = ∓14V | +25°C | -0.5 | 0.01 | 0.5 | nA |
| | | | E | -10 | | 10 | |
| COM On-Leakage Current (Notes 4, 5) | $I_{COM(ON)}$ | $V_{COM}$ = ±14V; $V_{NO}$, $V_{NC}$ = ±14V or unconnected | +25°C | -0.5 | 0.01 | 0.5 | nA |
| | | | E | -20 | | 20 | |
| **FAULT** (V+ = +15V, V- = -15V, unless otherwise noted.) | | | | | | | |
| Fault-Protected Analog Signal Range | $V_{NO}$, $V_{NC}$ | Applies with power on (Note 6) | E | -36 | | 36 | V |
| | | Applies with power off (Note 6) | | -40 | | 40 | |
| COM Off-Leakage Current, Supplies On | $I_{COM(OFF)}$ | $V_{NO}$ or $V_{NC}$ = ±36V | +25°C | -10 | | 10 | nA |
| | | | E | -200 | | 200 | |
| NO or NC Input Leakage Current, Supplies On | $I_{NO}$, $I_{NC}$ | $V_{NO}$ or $V_{NC}$ = ±36V, $V_{COM}$ = ∓10V | +25°C | -20 | | 20 | nA |
| | | | E | -200 | | 200 | |
| NO or NC Input Leakage Current, Supplies Off | $I_{NO}$, $I_{NC}$ | $V_{NO}$ or $V_{NC}$ = ±40V, V+ = 0, V- = 0 | +25°C | -20 | 0.1 | 20 | nA |
| | | | E | -200 | | 200 | |
| Clamp Output Current, Supplies On | $I_{COM}$ | $V_{NO}$ or $V_{NC}$ = 36V | +25°C | 8 | 11 | 13 | mA |
| | | $V_{NO}$ or $V_{NC}$ = -36V | | -12 | -10 | -7 | |

## ELECTRICAL CHARACTERISTICS—Dual Supplies (continued)

(V+ = +15V, V- = -15V, $V_{IH}$ = 2.4V, $V_{IL}$ = 0.8V, GND = 0, $T_A$ = $T_{MIN}$ to $T_{MAX}$, unless otherwise noted. Typical values are at $T_A$ = +25°C.) (Note 3)

| PARAMETER | SYMBOL | CONDITIONS | $T_A$ | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|---|
| Clamp Output Resistance, Supplies On | $R_{COM}$ | $V_{NO}$ or $V_{NC}$ = ±36V | +25°C | | 1 | 2.5 | kΩ |
| | | | E | | | 3 | |
| Fault Trip Threshold | | | +25°C | V- - 0.4 | | V+ + 0.4 | V |
| Fault Output Turn-On Delay Time | | $V_{IN}$ = ±25V, $R_L$ = 10kΩ | +25°C | | 10 | | ns |
| Fault Recovery Time | | $V_{IN}$ = ±25V, $R_L$ = 10kΩ | +25°C | | 2.5 | | µs |
| **LOGIC INPUT** | | | | | | | |
| IN Input Logic High | $V_{INH}$ | | E | 2.4 | | | V |
| IN Input Logic Low | $V_{INL}$ | | E | | | 0.8 | V |
| IN Input Current | $I_{INH}$, $I_{INL}$ | $V_{IN}$ = 0.8V or 2.4V | +25°C | -1 | 0.03 | 1 | µA |
| | | | E | -5 | | 5 | |
| **SWITCH DYNAMIC CHARACTERISTICS** | | | | | | | |
| Turn-On Time | $t_{ON}$ | $V_{NO}$ or $V_{NC}$ = ±10V, $R_L$ = 2kΩ, $C_L$ = 35pF, Figure 2 | +25°C | | 350 | 500 | ns |
| | | | E | | | 600 | |
| Turn-Off Time | $t_{OFF}$ | $V_{NO}$ or $V_{NC}$ = ±10V, $R_L$ = 2kΩ, $C_L$ = 35pF, Figure 2 | +25°C | | 60 | 175 | ns |
| | | | E | | | 250 | |
| Charge Injection (Note 7) | Q | $C_L$ = 1nF, $V_{NO}$ = 0, $R_S$ = 0Ω, Figure 3 | +25°C | | 1.5 | 5 | pC |
| NO or NC Off-Capacitance | $C_{N(OFF)}$ | f = 1MHz, Figure 4 | +25°C | | 10 | | pF |
| COM Off-Capacitance | $C_{COM(OFF)}$ | f = 1MHz, Figure 4 | +25°C | | 5 | | pF |
| COM On-Capacitance | $C_{COM(ON)}$ | f = 1MHz, Figure 4 | +25°C | | 10 | | pF |
| Off-Isolation (Note 8) | $VC_{ISO}$ | $R_L$ = 50Ω, $C_L$ = 15pF, $V_{N\_}$ = $1V_{RMS}$, f = 1MHz, Figure 5 | +25°C | | -62 | | dB |
| **POWER SUPPLY** | | | | | | | |
| Power-Supply Range | V+, V- | | E | ±4.5 | | ±20 | V |
| V+ Supply Current | I+ | $V_{IN}$ = 0 or 5V | +25°C | | 100 | 175 | µA |
| | | | E | | | 250 | |
| V- Supply Current | I- | $V_{IN}$ = 0 or 5V | +25°C | | 50 | 100 | µA |
| | | | E | | | 175 | |
| GND Supply Current | $I_{GND}$ | $V_{IN}$ = 0 or 15V | +25°C | -1 | 0.01 | 1 | µA |
| | | | E | | | 10 | |
| | | $V_{IN}$ = 5V | +25°C | | 50 | 100 | |
| | | | E | | | 175 | |

# Rail-to-Rail, Fault-Protected, SPST Analog Switches

## ELECTRICAL CHARACTERISTICS—Single +12V Supply

(V+ = +12V, V- = 0, GND = 0, $V_{IH}$ = 2.4V, $V_{IL}$ = 0.8V, $T_A$ = $T_{MIN}$ to $T_{MAX}$, unless otherwise noted. Typical values are at $T_A$ = +25°C.) (Note 3)

| PARAMETER | SYMBOL | CONDITIONS | $T_A$ | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|---|
| **ANALOG SWITCH** | | | | | | | |
| Fault-Free Analog Signal Range | $V_{NO}$, $V_{NC}$ | Applies with power on or off | E | 0 | | V+ | V |
| On-Resistance | $R_{ON}$ | $V_{COM}$ = 10V, $I_{COM}$ = 1mA | +25°C | | 260 | 390 | Ω |
| | | | E | | | 500 | |
| NO or NC Off-Leakage Current (Notes 4, 5, 9) | $I_{NO(OFF)}$, $I_{NC(OFF)}$ | $V_{COM}$ = 10V, 1V; $V_{NO}$, $V_{NC}$ = 1V, 10V | +25°C | -0.5 | 0.01 | 0.5 | nA |
| | | | E | -10 | | 10 | |
| COM Off-Leakage Current (Notes 4, 5, 9) | $I_{COM(OFF)}$ | $V_{COM}$ = 1V, 10V; $V_{NO}$, $V_{NC}$ = 10V, 1V | +25°C | -0.5 | 0.01 | 0.5 | nA |
| | | | E | -10 | | 10 | |
| COM On-Leakage Current (Notes 4, 5, 9) | $I_{COM(ON)}$ | $V_{COM}$ = 1V, 10V; $V_{NO}$, $V_{NC}$ = 1V, 10V, or unconnected | +25°C | -0.5 | 0.01 | 0.5 | nA |
| | | | E | -20 | | 20 | |
| **FAULT** | | | | | | | |
| Fault-Protected Analog Signal Range | $V_{NO}$, $V_{NC}$ | Applies with power on (Note 6) | E | -36 | | 36 | V |
| | | Applies with power off (Note 6) | | -40 | | 40 | |
| COM Off-Leakage Current, Supply On | $I_{COM}$ | $V_{NO}$ or $V_{NC}$ = ±36V | +25°C | -10 | | 10 | nA |
| | | | E | -200 | | 200 | |
| NO or NC Input Leakage Current, Supply On | $I_{NO}$, $I_{NC}$ | $V_{NO}$ or $V_{NC}$ = ±36V, $V_{COM}$ = 0 | +25°C | -20 | | 20 | nA |
| | | | E | -200 | | 200 | |
| NO or NC Input Leakage Current, Supply Off | $I_{NO}$, $I_{NC}$ | $V_{NO}$ or $V_{NC}$ = ±40V, V+ = 0, V- = 0 | +25°C | -20 | 0.1 | 20 | nA |
| | | | E | -200 | | 200 | |
| Clamp Output Current, Supply On | $I_{COM}$ | $V_{NO}$ or $V_{NC}$ = 36V | +25°C | 2 | 3 | 5 | mA |
| Clamp Output Resistance, Supply On | $R_{COM}$ | $V_{NO}$ or $V_{NC}$ = 36V | +25°C | | 2.4 | 5 | kΩ |
| **LOGIC INPUT** | | | | | | | |
| IN Input Logic High | $V_{INH}$ | | E | 2.4 | | | V |
| IN Input Logic Low | $V_{INL}$ | | E | | | 0.8 | V |
| IN Input Current | $I_{INH}$, $I_{INL}$ | $V_{IN}$ = 0.8V or 2.4V | +25°C | -1 | 0.03 | 1 | µA |
| | | | E | -5 | | 5 | |

## ELECTRICAL CHARACTERISTICS—Single +12V Supply (continued)

(V+ = +12V, V- = 0, GND = 0, $V_{IH}$ = 2.4V, $V_{IL}$ = 0.8V, $T_A$ = $T_{MIN}$ to $T_{MAX}$, unless otherwise noted. Typical values are at $T_A$ = +25°C.) (Note 3)

| PARAMETER | SYMBOL | CONDITIONS | $T_A$ | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|---|
| **SWITCH DYNAMIC CHARACTERISTICS** | | | | | | | |
| Turn-On Time | $t_{ON}$ | $V_{NO}$ or $V_{NC}$ = 7V, $R_L$ = 2kΩ, $C_L$ = 35pF, Figure 2 | +25°C | | 500 | 750 | ns |
| | | | E | | | 1000 | |
| Turn-Off Time | $t_{OFF}$ | $V_{NO}$ or $V_{NC}$ = 7V, $R_L$ = 2kΩ, $C_L$ = 35pF, Figure 2 | +25°C | | 60 | 200 | ns |
| | | | E | | | 300 | |
| Charge Injection (Note 7) | Q | $C_L$ = 1nF, $V_{NO}$ = 0, $R_S$ = 0Ω, Figure 3 | +25°C | | 1 | 5 | pC |
| NO or NC Off-Capacitance | $C_{NO(OFF)}$, $C_{NC(OFF)}$ | f = 1MHz, Figure 4 | +25°C | | 9 | | pF |
| COM Off-Capacitance | $C_{COM(OFF)}$ | $V_{COM}$ = 0, f = 1MHz, Figure 4 | +25°C | | 9 | | pF |
| COM On-Capacitance | $C_{COM(ON)}$ | $V_{COM}$ = $V_{NO}$ = 0, f = 1MHz, Figure 4 | +25°C | | 22 | | pF |
| Off-Isolation (Note 8) | $V_{ISO}$ | $R_L$ = 50Ω, $C_L$ = 15pF, $V_{IN}$ = $1V_{RMS}$, f = 1MHz, Figure 5 | +25°C | | -62 | | dB |
| **POWER SUPPLY** | | | | | | | |
| Power-Supply Range | V+ | | E | 9 | | 36 | V |
| V+ Supply Current | I+ | $V_{IN}$ = 0 or 5V | +25°C | | 50 | 125 | µA |
| | | | E | | | 175 | |
| V- and GND Supply Current | $I_{GND}$ | $V_{IN}$ = 0 or 12V | +25°C | | 25 | 75 | µA |
| | | | E | | | 125 | |
| | | $V_{IN}$ = 0 or 5V | +25°C | | 50 | 125 | µA |
| | | | E | | | 175 | |

**Note 3:** Algebraic convention is used in this data sheet; the most negative value is shown in the minimum column.
**Note 4:** Leakage parameters are 100% tested at maximum-rated hot temperature and guaranteed by correlation at $T_A$ = +25°C.
**Note 5:** SOT packages are 100% tested at +25°C. Limits at the maximum-rated temperature are guaranteed by design and correlation limits at +25°C. Leakage tests for the SOT package are typical only.
**Note 6:** NC and NO pins are fault protected. Signals on NC or NO exceeding -36V to +36V may damage the device. These limits apply with power applied to V+ or V-, or ±40V with V+ = V- = 0.
**Note 7:** Guaranteed by design.
**Note 8:** Off isolation = $20\log_{10}$ [ $V_{COM}$ / ($V_{NC}$ or $V_{NO}$) ], $V_{COM}$ = output, $V_{NC}$ or $V_{NO}$ = input to off switch.
**Note 9:** Leakage testing for single-supply operation is guaranteed by testing with dual supplies.


## Features

- Non-contacting magnetic technology
- Highly resistant to vibration/shock
- Highly resistant to fluid/dust ingress
- Programmable at factory for zero position
- Robust design for industrial applications
- Highly repeatable
- RoHS compliant*

# AMS22S Non-Contacting Analog Rotary Position Sensor

### Electrical Characteristics[1] (@ 25 °C)

| | |
|---|---|
| VDD Supply Voltage | 5 V ± 10 % |
| Supply Current [2] | |
| For Low Speed Processing (Code L) | 10 mA max. |
| For High Speed Processing (Code H) | 20 mA max. |
| Output Signal (Single) | Analog |
| Independent Linearity | ±0.5 % |
| | (±0.3 % available on request) |
| Hysteresis | 0.2 % VDD max. |
| Effective Electrical Angle [3] | 340 ° |
| Programmable Electrical Angle | 10 ° to 360 ° (10 ° increments) |
| Voltage Output (Programmable) | 1 to 99 % VDD ±1 % |
| Output Resolution | 12 bit @ 360 ° |
| Load Resistance Recommended | 10K ohms to ∞ |
| Overvoltage Protection | +20 VDC |
| Reverse Voltage Protection | -10 VDC |

### Environmental Characteristics

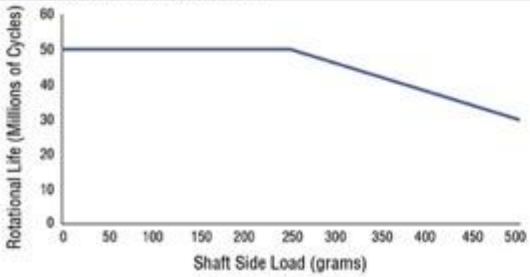
| | |
|---|---|
| Operating Temperature | -40 ° to +125 °C |
| Moisture Resistance | MIL-STD-202, Method 106 |
| Insulation Resistance @ 500 VAC | 500 MΩ min. |
| Rotational Life (Shaft Revolutions) | 50 million |
| Vibration | 15 G |
| Shock | 50 G |
| IP Rating | IP50 |
| ESD Rating | 2 kV max. |

### Mechanical Characteristics (@ 25 °C)

| | |
|---|---|
| Mechanical Angle | Continuous |
| Shaft/RPM | 200 RPM max. |
| Torque (Starting & Running) | 1.06 N-cm. (1.5 oz-in.) max. |
| Shaft Material | Stainless steel |
| Terminals | Brass / 100 % matte tin over Ni Strike (e3) |
| Bearing | Bronze sleeve |
| Soldering Condition | |
| Manual Soldering | 96.5Sn/3.0Ag/0.5Cu solid wire or no-clean rosin cored wire; 370 °C (700 °F) max. for 3 seconds |
| Wave Soldering | 96.5Sn/3.0Ag/0.5Cu solder with no-clean fl ux; 260 °C (500 °F) max. for 5 seconds |
| Wash processes | Not recommended |

[1] At room ambient: +25 °C nominal and 50 % relative humidity nominal, except as noted.
[2] See "Processing Speed" in How to Order selection guide.
[3] Other Effective Electrical Angles available. See How to Order selection guide.

### Rotational Life vs. Shaft Side Load




### Product Dimensions

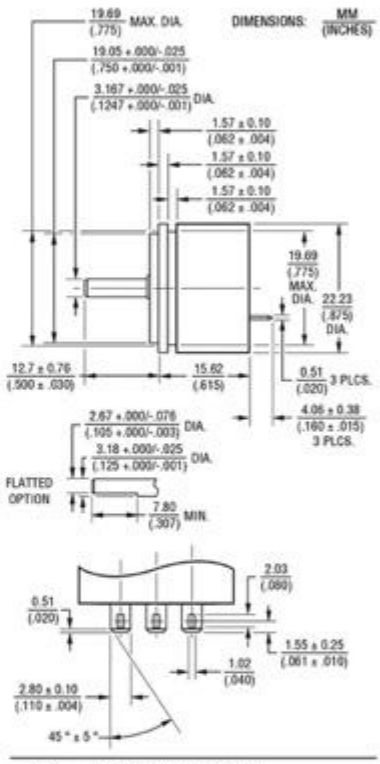
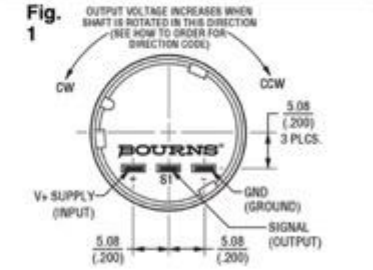
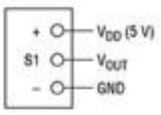



Fig. 1




### Schematic




*RoHS Directive 2002/95/EC Jan. 27, 2003 including annex and RoHS Recast 2011/65/EU June 8, 2011.
Specifications are subject to change without notice.
The device characteristics and parameters in this data sheet can and do vary in different applications and actual device performance may vary over time.
Users should verify actual device performance in their specific applications.

# AMS22S Non-Contacting Analog Rotary Position Sensor

BOURNS®

## How To Order

| A | M | S | 2 | 2 | S | 5 | A | 1 | B | L | A | F | L | 3 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**MODEL DESIGNATOR**

| Code | Description |
|---|---|
| AM | Analog Magnetic |

**CONFIGURATION**

| Code | Description |
|---|---|
| S | Single-turn |

**SIZE**

| Code | Description |
|---|---|
| 22 | 22 mm |

**MOUNTING CONFIGURATION**

| Code | Description |
|---|---|
| S | Servo Mount, 1/2 " Shaft FMS, 1/8 " Shaft Dia. |

**SUPPLY VOLTAGE, $V_{CC}$**

| Code | Description |
|---|---|
| 5 | 5 VDC |

**OUTPUT TYPE**

| Code | Type | Signal | Resolution |
|---|---|---|---|
| A1 | Analog | Single | 12 Bit |

**INDEPENDENT LINEARITY**

| Code | Description |
|---|---|
| B | 0.5 % |
| C | 0.3 % |

**PROCESSING SPEED**

| Code | Output Refresh Rate |
|---|---|
| H | High (200 µs Typ.) |
| L | Low (600 µs Typ.) |

**VOLTAGE RATIO ±1 %**

| Code | Description |
|---|---|
| A | 1-99 % of VCC |
| B | 5-95 % of VCC |
| C | 10-90 % of VCC |

**SHAFT STYLE**

| Code | Description |
|---|---|
| F | Flatted |
| R | Round |

**TERMINAL CONFIGURATION**

| Code | Description |
|---|---|
| L | Axial |

**DIRECTION***

| Code | Description |
|---|---|
| 1 | CCW (Increasing Voltage) |
| 3 | CW (Increasing Voltage) |

* *See Figure 1 for clarification.*

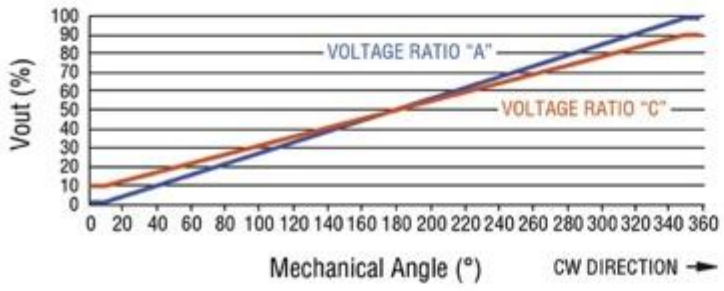
**EFFECTIVE ELECTRICAL ANGLE 10 ° Increments**

| Code | Description |
|---|---|
| 01 | 10 ° |
| 02 | 20 ° |
| 03 | 30 ° |
| 34 | 340 ° |
| 35 | 350 ° |
| 36 | 360 ° |

*Notes:*
- *Effective Electrical Angle available in 10 ° increments from a minimum of 10 ° to 360 ° maximum.*
- *Use Code "BB" for 45°.*

*Shaded areas represent most common features.*

## Standard Output: 1-Turn CW Increasing (Code 334 Shown)




BOURNS®

**Asia-Pacific:**
Tel: +886-2 2562-4117
Fax: +886-2 2562-4116

**EMEA:**
Tel: +36 88 520 390
Fax: +36 88 520 211

**The Americas:**
Tel: +1-951 781-5500
Fax: +1-951 781-5700

**www.bourns.com**

REV. 10/14

## 1.8 Anexo II – Calibrado de sensores, tablas y gráficas

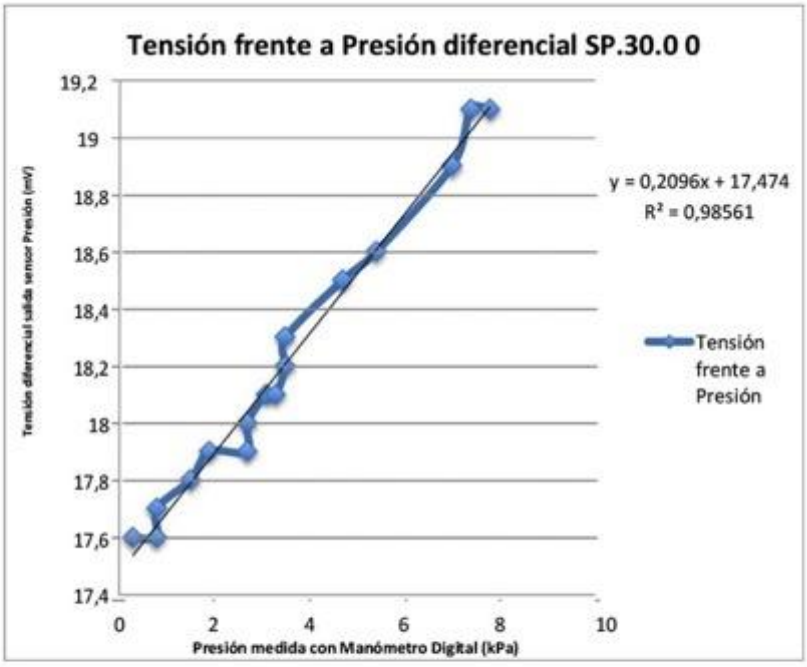
### *Resultados calibración sensores de presión*

Tabla1 - resumen sensiblidades sensores SP.30.XX

| SP.30.XX | Sensibilidad (mV/kPa) | Sensibilidad (mV/kPa*uA) |
|---|---|---|
| 0 | 0,2096 | 4,2776E-04 |
| 1 | 0,2579 | 4,5727E-04 |
| 2 | 0,2495 | 4,4316E-04 |
| 3 | 0,2157 | 4,3488E-04 |
| 4 | 0,266 | 4,7585E-04 |
| 5 | 0,2822 | 5,0393E-04 |
| 6 | 0,2619 | 4,5947E-04 |
| 7 | 0,2349 | 4,1283E-04 |
| 9 | 0,2425 | 4,2544E-04 |
| 10 | 0,2594 | 4,5429E-04 |
| 11 | 0,2698 | 4,7417E-04 |
| 12 | 0,2546 | 4,5062E-04 |

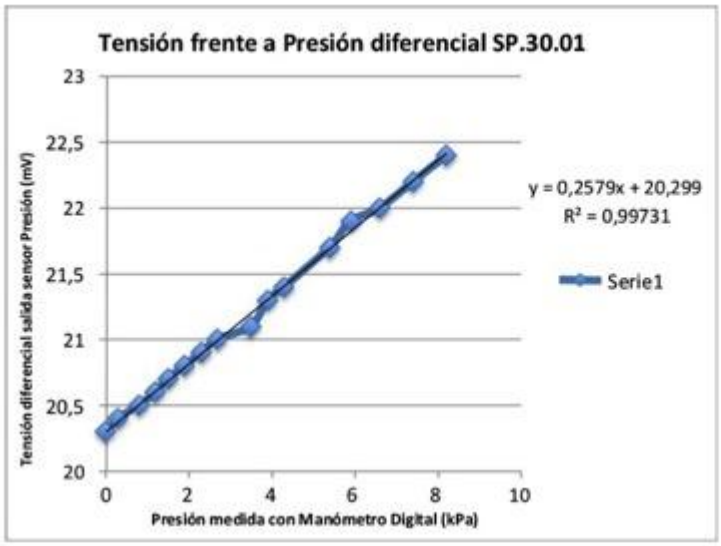
SP.30.00

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 490 | 17,6 | 0,3 | 101,51 |
| 490 | 17,6 | 0,8 | 102,01 |
| 490 | 17,7 | 0,8 | 102,01 |
| 490 | 17,8 | 1,5 | 102,71 |
| 490 | 17,9 | 1,9 | 103,11 |
| 490 | 17,9 | 2,7 | 103,91 |
| 490 | 18 | 2,7 | 103,91 |
| 490 | 18,1 | 3,1 | 104,31 |
| 490 | 18,1 | 3,3 | 104,51 |
| 490 | 18,2 | 3,5 | 104,71 |
| 490 | 18,3 | 3,5 | 104,71 |
| 490 | 18,5 | 4,7 | 105,91 |
| 490 | 18,6 | 5,4 | 106,61 |
| 490 | 18,9 | 7 | 108,21 |
| 490 | 19,1 | 7,4 | 108,61 |
| 490 | 19,1 | 7,8 | 109,01 |



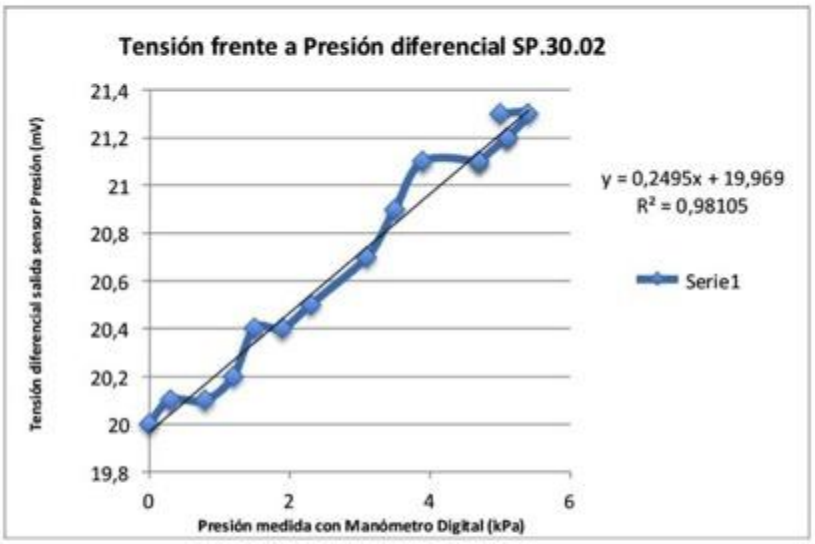
SP.30.01

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 564 | 20,3 | 0 | 101,21 |
| 564 | 20,4 | 0,3 | 101,51 |
| 564 | 20,5 | 0,8 | 102,01 |
| 564 | 20,6 | 1,2 | 102,41 |
| 564 | 20,7 | 1,5 | 102,71 |
| 564 | 20,8 | 1,9 | 103,11 |
| 564 | 20,9 | 2,3 | 103,51 |
| 564 | 21 | 2,7 | 103,91 |
| 564 | 21,1 | 3,5 | 104,71 |
| 564 | 21,3 | 3,9 | 105,11 |
| 564 | 21,4 | 4,3 | 105,51 |
| 564 | 21,7 | 5,4 | 106,61 |
| 564 | 21,9 | 5,9 | 107,11 |
| 564 | 22 | 6,6 | 107,81 |
| 564 | 22,2 | 7,4 | 108,61 |
| 564 | 22,4 | 8,2 | 109,41 |



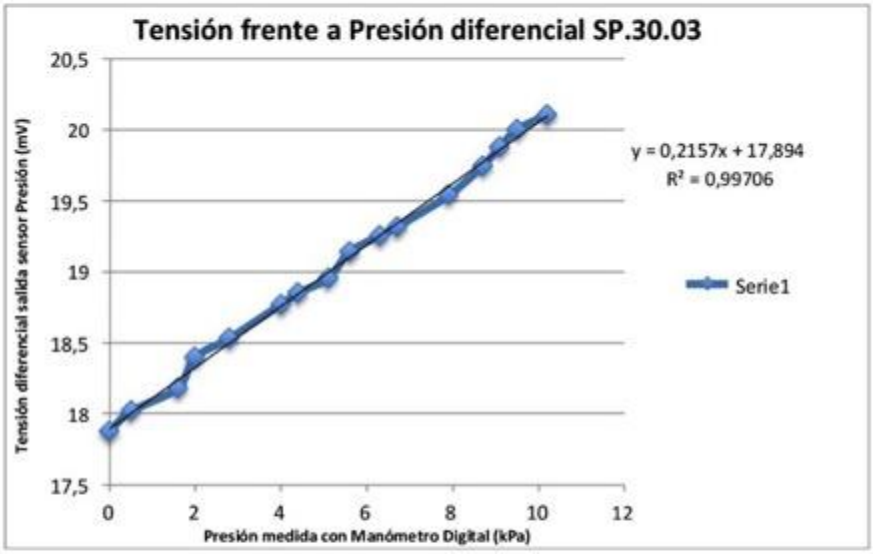
SP.30.02

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 563 | 20 | 0 | 101,21 |
| 563 | 20,1 | 0,3 | 101,51 |
| 563 | 20,1 | 0,8 | 102,01 |
| 563 | 20,2 | 1,2 | 102,41 |
| 563 | 20,4 | 1,5 | 102,71 |
| 563 | 20,4 | 1,9 | 103,11 |
| 563 | 20,5 | 2,3 | 103,51 |
| 563 | 20,7 | 3,1 | 104,31 |
| 563 | 20,9 | 3,5 | 104,71 |
| 563 | 21,1 | 3,9 | 105,11 |
| 563 | 21,1 | 4,7 | 105,91 |
| 563 | 21,1 | 4,7 | 105,91 |
| 563 | 21,2 | 5,1 | 106,31 |
| 563 | 21,2 | 5,1 | 106,31 |
| 563 | 21,3 | 5,4 | 106,61 |
| 563 | 21,3 | 5 | 106,21 |



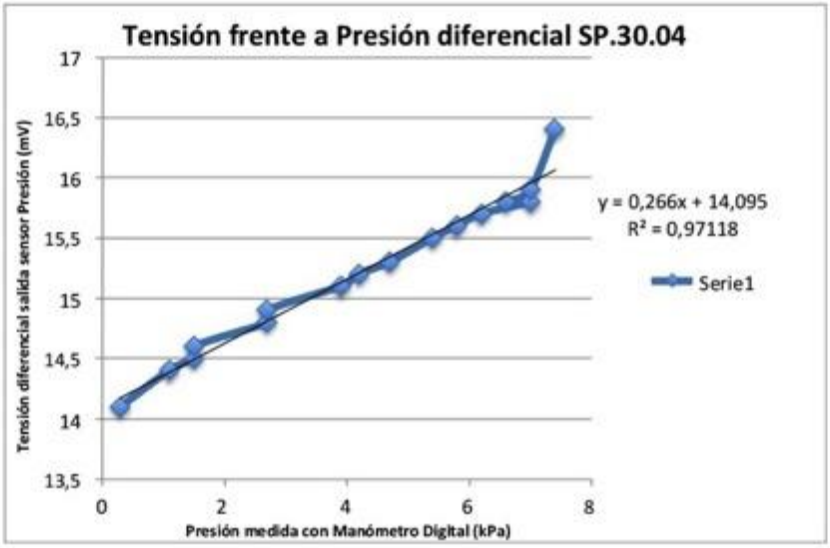
SP.30.03

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 496 | 20,10215512 | 10,2 | 111,41 |
| 496 | 19,99938337 | 9,5 | 110,71 |
| 496 | 19,87605726 | 9,1 | 110,31 |
| 496 | 19,74245398 | 8,7 | 109,91 |
| 496 | 19,53691048 | 7,9 | 109,11 |
| 496 | 19,31081262 | 6,7 | 107,91 |
| 496 | 19,24914956 | 6,3 | 107,51 |
| 496 | 19,13610063 | 5,6 | 106,81 |
| 496 | 18,95111148 | 5,1 | 106,31 |
| 496 | 18,84833972 | 4,4 | 105,61 |
| 496 | 18,76612232 | 4 | 105,21 |
| 496 | 18,52974728 | 2,8 | 104,01 |
| 496 | 18,396144 | 2 | 103,21 |
| 496 | 18,18032332 | 1,6 | 102,81 |
| 496 | 18,01588851 | 0,5 | 101,71 |
| 496 | 17,87200806 | 0 | 101,21 |



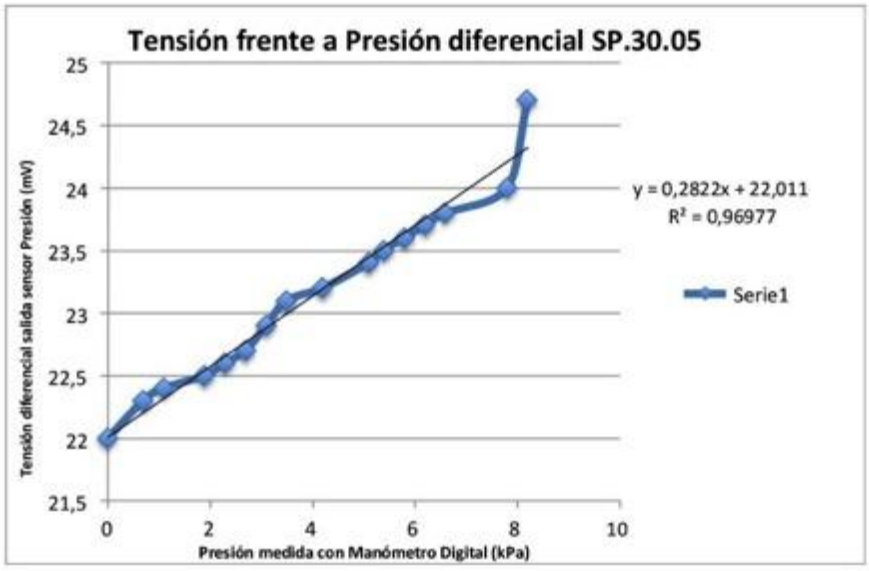
SP.30.04

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 559 | 14,1 | 0,3 | 101,51 |
| 559 | 14,4 | 1,1 | 102,31 |
| 559 | 14,5 | 1,5 | 102,71 |
| 559 | 14,6 | 1,5 | 102,71 |
| 559 | 14,8 | 2,7 | 103,91 |
| 559 | 14,9 | 2,7 | 103,91 |
| 559 | 15,1 | 3,9 | 105,11 |
| 559 | 15,2 | 4,2 | 105,41 |
| 559 | 15,3 | 4,7 | 105,91 |
| 559 | 15,5 | 5,4 | 106,61 |
| 559 | 15,6 | 5,8 | 107,01 |
| 559 | 15,7 | 6,2 | 107,41 |
| 559 | 15,8 | 7 | 108,21 |
| 559 | 15,8 | 6,6 | 107,81 |
| 559 | 15,9 | 7 | 108,21 |
| 559 | 16,4 | 7,4 | 108,61 |



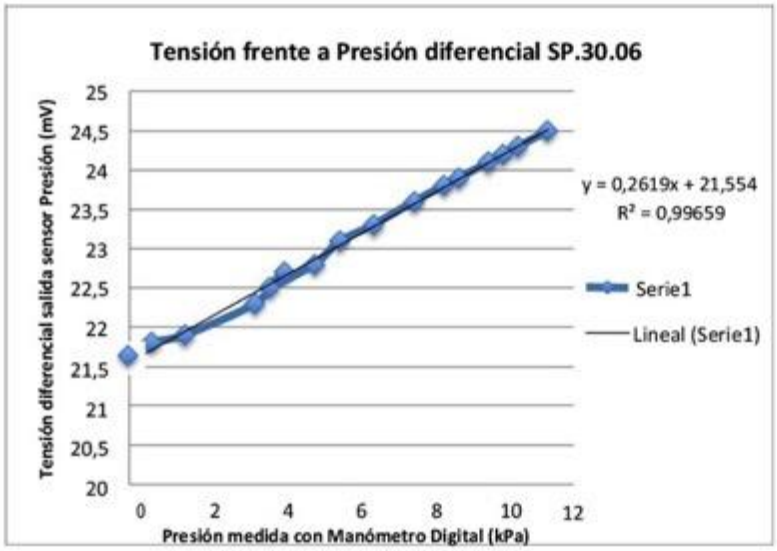
SP.30.05

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 560 | 22 | 0 | 101,21 |
| 560 | 22,3 | 0,7 | 101,91 |
| 560 | 22,4 | 1,1 | 102,31 |
| 560 | 22,5 | 1,9 | 103,11 |
| 560 | 22,6 | 2,3 | 103,51 |
| 560 | 22,7 | 2,7 | 103,91 |
| 560 | 22,9 | 3,1 | 104,31 |
| 560 | 23,1 | 3,5 | 104,71 |
| 560 | 23,2 | 4,2 | 105,41 |
| 560 | 23,4 | 5,1 | 106,31 |
| 560 | 23,5 | 5,4 | 106,61 |
| 560 | 23,6 | 5,8 | 107,01 |
| 560 | 23,7 | 6,2 | 107,41 |
| 560 | 23,8 | 6,6 | 107,81 |
| 560 | 24 | 7,8 | 109,01 |
| 560 | 24,7 | 8,2 | 109,41 |



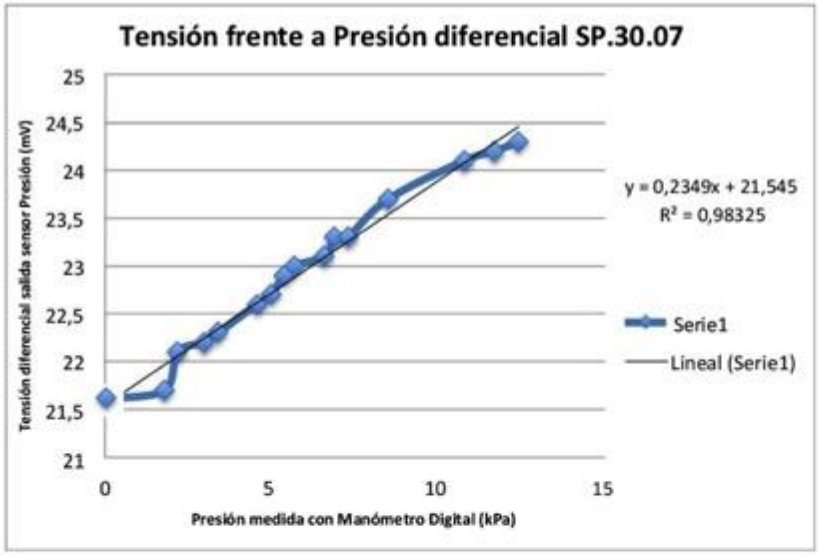
SP.30.06

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 570 | 21,6 | 0 | 101,21 |
| 570 | 21,8 | 0,6 | 101,81 |
| 570 | 21,9 | 1,5 | 102,71 |
| 570 | 22,3 | 3,4 | 104,61 |
| 570 | 22,7 | 4,2 | 105,41 |
| 570 | 22,5 | 3,8 | 105,01 |
| 570 | 22,8 | 5 | 106,21 |
| 570 | 23,1 | 5,7 | 106,91 |
| 570 | 23,3 | 6,6 | 107,81 |
| 570 | 23,6 | 7,7 | 108,91 |
| 570 | 23,8 | 8,5 | 109,71 |
| 570 | 23,9 | 8,9 | 110,11 |
| 570 | 24,1 | 9,7 | 110,91 |
| 570 | 24,2 | 10,1 | 111,31 |
| 570 | 24,3 | 10,5 | 111,71 |
| 570 | 24,5 | 11,3 | 112,51 |



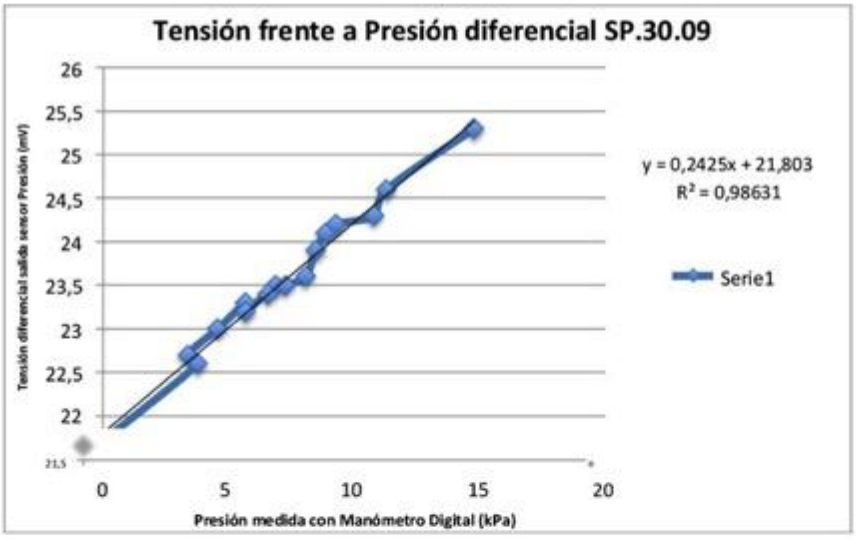
SP.30.07

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 569 | 21,6 | 0 | 101,21 |
| 569 | 21,7 | 1,8 | 103,01 |
| 569 | 22,1 | 2,2 | 103,41 |
| 569 | 22,2 | 3 | 104,21 |
| 569 | 22,3 | 3,4 | 104,61 |
| 569 | 22,6 | 4,6 | 105,81 |
| 569 | 22,7 | 5 | 106,21 |
| 569 | 22,9 | 5,4 | 106,61 |
| 569 | 23 | 5,7 | 106,91 |
| 569 | 23,1 | 6,6 | 107,81 |
| 569 | 23,3 | 6,9 | 108,11 |
| 569 | 23,3 | 7,3 | 108,51 |
| 569 | 23,7 | 8,5 | 109,71 |
| 569 | 24,1 | 10,8 | 112,01 |
| 569 | 24,2 | 11,7 | 112,91 |
| 569 | 24,3 | 12,4 | 113,61 |



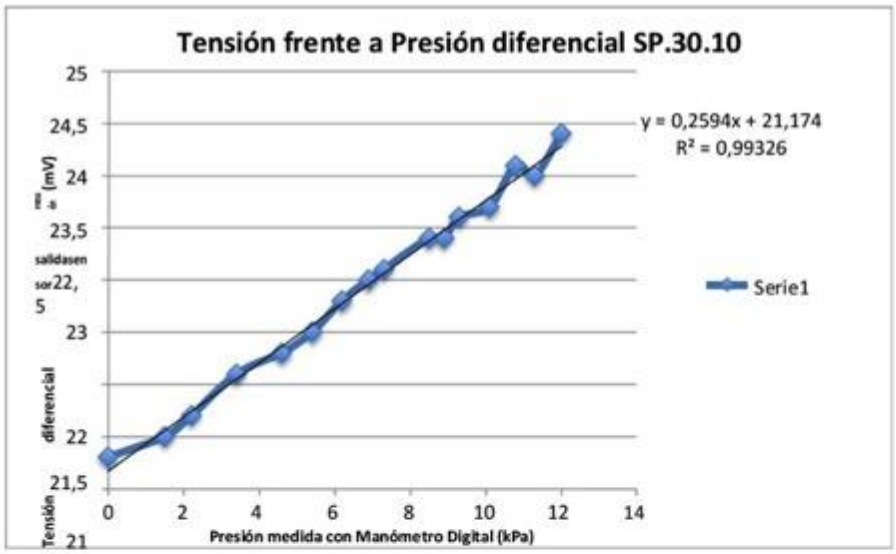
SP.30.09

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 570 | 21,7 | 0 | 101,21 |
| 570 | 22,6 | 3,8 | 105,01 |
| 570 | 22,7 | 3,4 | 104,61 |
| 570 | 23 | 4,6 | 105,81 |
| 570 | 23,3 | 5,7 | 106,91 |
| 570 | 23,2 | 5,7 | 106,91 |
| 570 | 23,5 | 6,9 | 108,11 |
| 570 | 23,4 | 6,6 | 107,81 |
| 570 | 23,5 | 7,3 | 108,51 |
| 570 | 23,6 | 8,1 | 109,31 |
| 570 | 23,9 | 8,5 | 109,71 |
| 570 | 24,1 | 8,9 | 110,11 |
| 570 | 24,2 | 9,3 | 110,51 |
| 570 | 24,3 | 10,8 | 112,01 |
| 570 | 24,6 | 11,3 | 112,51 |
| 570 | 25,3 | 14,8 | 116,01 |



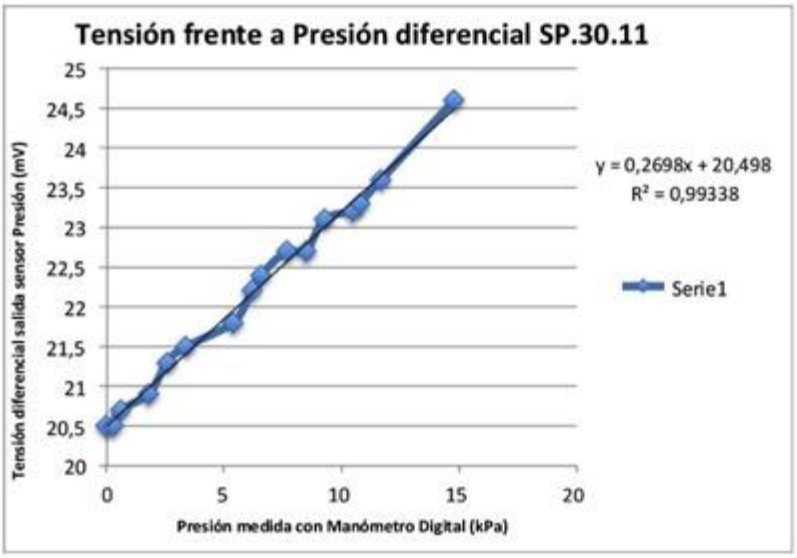
SP.30.10

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 571 | 21,3 | 0 | 101,21 |
| 571 | 21,5 | 1,5 | 102,71 |
| 571 | 21,7 | 2,2 | 103,41 |
| 571 | 22,1 | 3,4 | 104,61 |
| 571 | 22,3 | 4,6 | 105,81 |
| 571 | 22,5 | 5,4 | 106,61 |
| 571 | 22,8 | 6,2 | 107,41 |
| 571 | 23 | 6,9 | 108,11 |
| 571 | 23,1 | 7,3 | 108,51 |
| 571 | 23,4 | 8,5 | 109,71 |
| 571 | 23,4 | 8,9 | 110,11 |
| 571 | 23,6 | 9,3 | 110,51 |
| 571 | 23,7 | 10,1 | 111,31 |
| 571 | 24,1 | 10,8 | 112,01 |
| 571 | 24 | 11,3 | 112,51 |
| 571 | 24,4 | 12 | 113,21 |



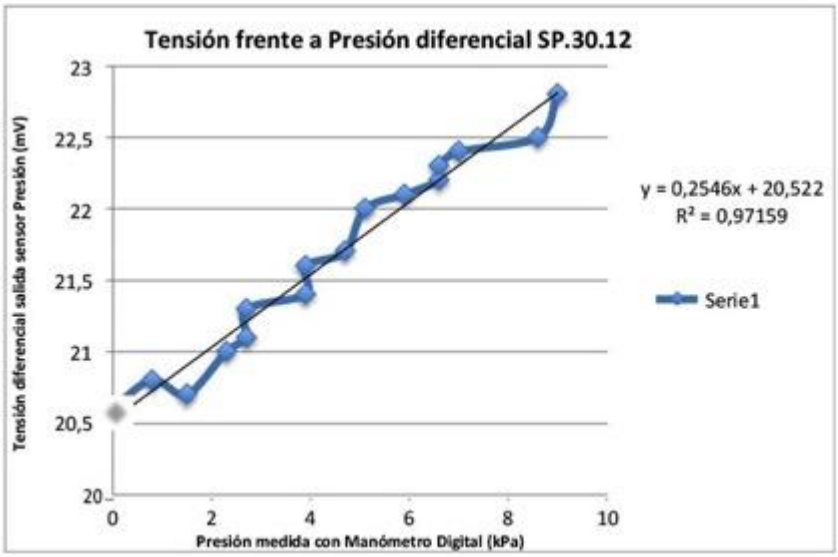
SP.30.11

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 569 | 20,5 | 0 | 101,21 |
| 569 | 20,5 | 0,3 | 101,51 |
| 569 | 20,7 | 0,6 | 101,81 |
| 569 | 20,9 | 1,8 | 103,01 |
| 569 | 21,3 | 2,6 | 103,81 |
| 569 | 21,5 | 3,4 | 104,61 |
| 569 | 21,8 | 5,4 | 106,61 |
| 569 | 22,2 | 6,2 | 107,41 |
| 569 | 22,4 | 6,6 | 107,81 |
| 569 | 22,7 | 7,7 | 108,91 |
| 569 | 22,7 | 8,5 | 109,71 |
| 569 | 23,1 | 9,3 | 110,51 |
| 569 | 23,2 | 10,5 | 111,71 |
| 569 | 23,3 | 10,8 | 112,01 |
| 569 | 23,6 | 11,7 | 112,91 |
| 569 | 24,6 | 14,8 | 116,01 |



SP.30.12

| Corriente alimentación (µA) | Tensión Diferencial salida sensor P (mV) | Presión diferencial Manometro (kPa) | Presión Absoluta Manometro (kPa) |
|---|---|---|---|
| 565 | 20,6 | 0 | 101,21 |
| 565 | 20,8 | 0,8 | 102,01 |
| 565 | 20,7 | 1,5 | 102,71 |
| 565 | 21 | 2,3 | 103,51 |
| 565 | 21,1 | 2,7 | 103,91 |
| 565 | 21,3 | 2,7 | 103,91 |
| 565 | 21,4 | 3,9 | 105,11 |
| 565 | 21,6 | 3,9 | 105,11 |
| 565 | 21,7 | 4,7 | 105,91 |
| 565 | 22 | 5,1 | 106,31 |
| 565 | 22,1 | 5,9 | 107,11 |
| 565 | 22,2 | 6,6 | 107,81 |
| 565 | 22,3 | 6,6 | 107,81 |
| 565 | 22,4 | 7 | 108,21 |
| 565 | 22,5 | 8,6 | 109,81 |
| 565 | 22,8 | 9 | 110,21 |



### *Resultados calibración fuentes de corriente*

| Sensibilidad Final de cada FC | |
|---|---|
| **Número FC** | **Sensibilidad (uA/°C)** |
| 0 | 2,6813 |
| 1 | 3,1327 |
| 2 | 3,6167 |
| 3 | 2,6288 |
| 4 | 2,2819 |
| 5 | 3,6753 |
| 6 | 3,2374 |
| 7 | 4,0119 |
| 8 | 3,4792 |
| 9 | 2,6448 |
| 10 | 4,2893 |
| 11 | 3,5432 |

| Calibrado Fuente Corriente FC0 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 46 | 47 | 46,5 | 549 |
| 44,6 | 47,4 | 46 | 544 |
| 44,1 | 45,1 | 44,6 | 541 |
| 44 | 44,2 | 44,1 | 540 |
| 43 | 45 | 44 | 539 |
| 41,8 | 44,2 | 43 | 535 |
| 41 | 42,6 | 41,8 | 531 |
| 40,7 | 41,3 | 41 | 529 |
| 40,8 | 40,6 | 40,7 | 528 |
| 39,9 | 41,7 | 40,8 | 528 |
| 40,4 | 39,4 | 39,9 | 527 |
| 40,6 | 40,2 | 40,4 | 527 |
| 39,7 | 41,5 | 40,6 | 527 |
| 39,5 | 39,9 | 39,7 | 525 |
| 39,4 | 39,6 | 39,5 | 524 |
| 38,8 | 40 | 39,4 | 523 |
| 38,7 | 38,9 | 38,8 | 522 |
| 38,8 | 38,6 | 38,7 | 521 |
| 38,2 | 39,4 | 38,8 | 521 |
| 38,4 | 38 | 38,2 | 519 |
| 37,8 | 39 | 38,4 | 519 |
| 37,9 | 37,7 | 37,8 | 517 |
| 37 | 38,8 | 37,9 | 517 |
| 36,6 | 37,4 | 37 | 515 |
| 36,7 | 36,5 | 36,6 | 514 |
| 36,4 | 37 | 36,7 | 514 |
| 35,9 | 36,9 | 36,4 | 513 |
| 35,4 | 36,4 | 35,9 | 512 |
| 35,6 | 35,2 | 35,4 | 511 |
| 34,5 | 36,7 | 35,6 | 511 |
| 33,1 | 35,9 | 34,5 | 510 |
| 33,6 | 32,6 | 33,1 | 509 |
| 32,9 | 34,3 | 33,6 | 509 |
| 30,2 | 35,6 | 32,9 | 507 |
| 32,7 | 27,7 | 30,2 | 506 |
| 29,7 | 35,7 | 32,7 | 506 |
| 30,7 | 28,7 | 29,7 | 505 |
| 31,5 | 29,9 | 30,7 | 505 |
| 30,9 | 32,1 | 31,5 | 504 |
| 41 | 20,8 | 30,9 | 503 |

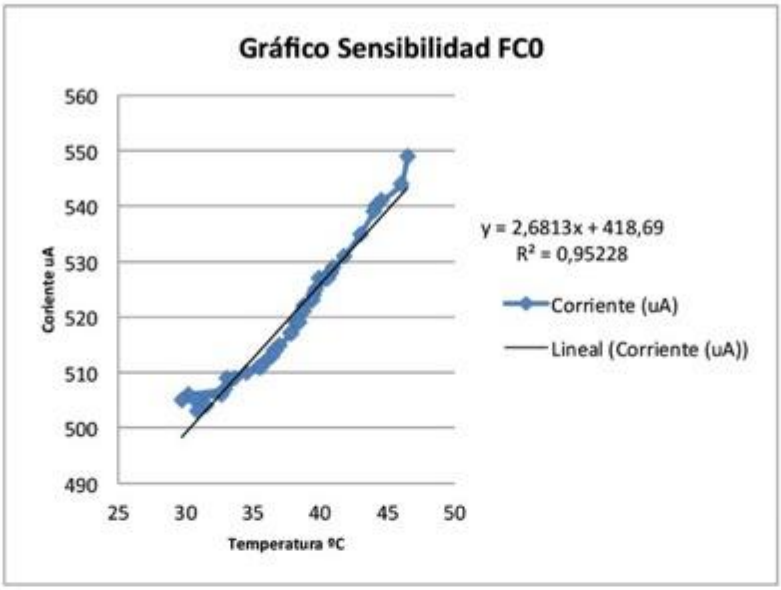
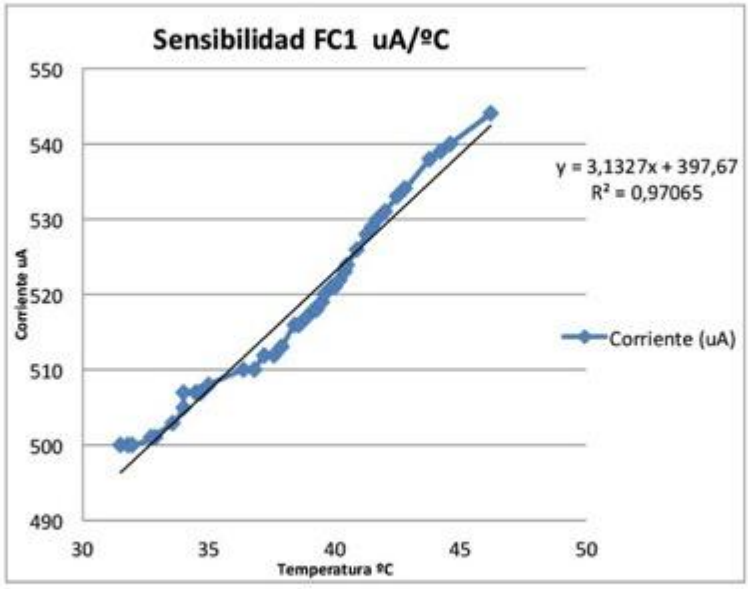
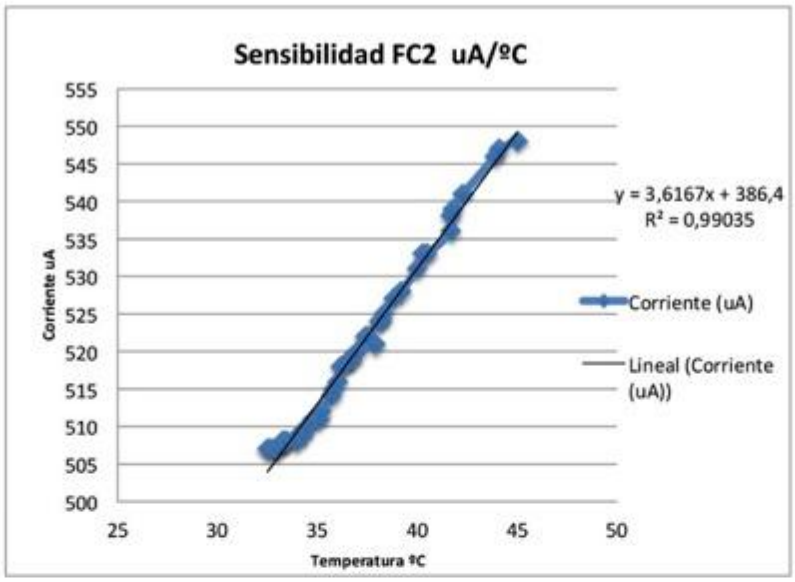


Gráfico Sensibilidad FC0
560
550
540
530
520
510
500
490
Coriente uA
25
30
35
40
45
50
Temperatura ºC
y = 2,6813x + 418,69
R² = 0,95228
Corriente (uA)
Lineal (Corriente (uA))

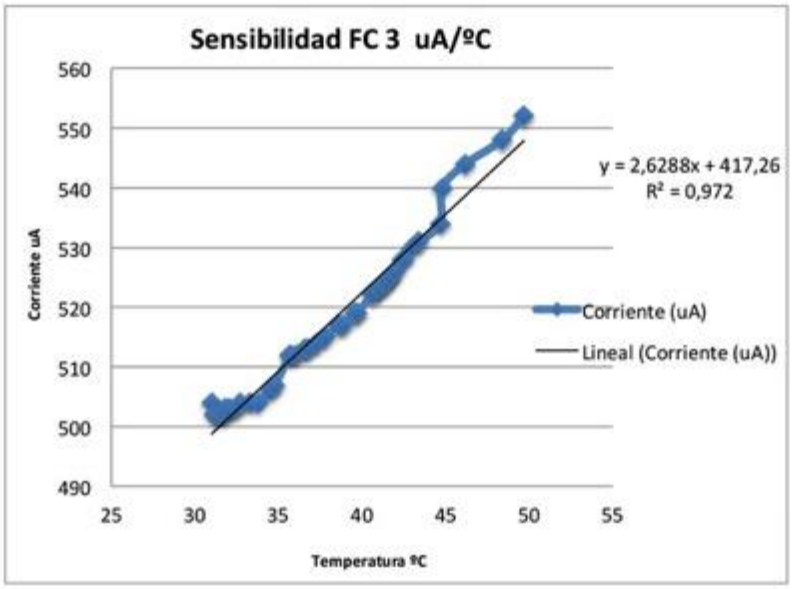
| Calibrado Fuente Corriente FC1 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 31,8 | 31,2 | 31,5 | 500 |
| 32 | 31,6 | 31,8 | 500 |
| 32,9 | 31,1 | 32 | 500 |
| 32,7 | 33,1 | 32,9 | 501 |
| 33,6 | 31,8 | 32,7 | 501 |
| 34 | 33,2 | 33,6 | 503 |
| 34 | 34 | 34 | 505 |
| 34,5 | 33,5 | 34 | 507 |
| 34,6 | 34,4 | 34,5 | 507 |
| 34,7 | 34,5 | 34,6 | 507 |
| 35 | 34,4 | 34,7 | 507 |
| 36,4 | 33,6 | 35 | 508 |
| 36,8 | 36 | 36,4 | 510 |
| 37,2 | 36,4 | 36,8 | 510 |
| 37,6 | 36,8 | 37,2 | 512 |
| 37,9 | 37,3 | 37,6 | 512 |
| 38,4 | 37,4 | 37,9 | 513 |
| 38,6 | 38,2 | 38,4 | 516 |
| 38,9 | 38,3 | 38,6 | 516 |
| 39,2 | 38,6 | 38,9 | 517 |
| 39,3 | 39,1 | 39,2 | 518 |
| 39,5 | 39,1 | 39,3 | 518 |
| 39,6 | 39,4 | 39,5 | 519 |
| 39,9 | 39,3 | 39,6 | 520 |
| 40 | 39,8 | 39,9 | 521 |
| 40,2 | 39,8 | 40 | 521 |
| 40,4 | 40 | 40,2 | 522 |
| 40,5 | 40,3 | 40,4 | 523 |
| 40,9 | 40,1 | 40,5 | 524 |
| 41,3 | 40,5 | 40,9 | 526 |
| 41,5 | 41,1 | 41,3 | 528 |
| 41,7 | 41,3 | 41,5 | 529 |
| 42 | 41,4 | 41,7 | 530 |
| 42,5 | 41,5 | 42 | 531 |
| 42,8 | 42,2 | 42,5 | 533 |
| 43,8 | 41,8 | 42,8 | 534 |
| 44,2 | 43,4 | 43,8 | 538 |
| 44,6 | 43,8 | 44,2 | 539 |
| 46,2 | 43 | 44,6 | 540 |
| 46,2 | 46,2 | 46,2 | 544 |


Sensibilidad FC1 uA/ºC
550
540
530
520
510
500
490
Corriente uA
30
35
40
45
50
Temperatura ºC
y = 3,1327x + 397,67
R² = 0,97065
Corriente (uA)

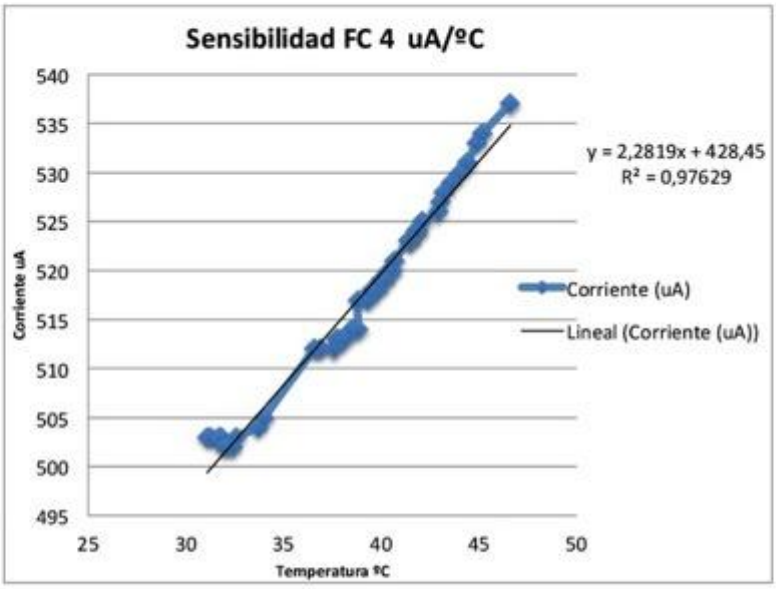
| Calibrado Fuente Corriente FC2 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 32,7 | 32,3 | 32,5 | 507 |
| 32,9 | 32,5 | 32,7 | 507 |
| 33 | 32,8 | 32,9 | 507 |
| 33,3 | 32,7 | 33 | 507 |
| 33,4 | 33,2 | 33,3 | 508 |
| 33,5 | 33,3 | 33,4 | 508 |
| 34 | 33 | 33,5 | 508 |
| 34,1 | 33,9 | 34 | 508 |
| 34,3 | 33,9 | 34,1 | 509 |
| 34,5 | 34,1 | 34,3 | 509 |
| 34,8 | 34,2 | 34,5 | 510 |
| 35,1 | 34,5 | 34,8 | 511 |
| 35,2 | 35 | 35,1 | 511 |
| 35,7 | 34,7 | 35,2 | 512 |
| 35,8 | 35,6 | 35,7 | 514 |
| 36 | 35,6 | 35,8 | 515 |
| 36,2 | 35,8 | 36 | 516 |
| 36,3 | 36,1 | 36,2 | 518 |
| 36,6 | 36 | 36,3 | 518 |
| 36,7 | 36,5 | 36,6 | 519 |
| 37 | 36,4 | 36,7 | 519 |
| 37,9 | 36,1 | 37 | 520 |
| 37,4 | 38,4 | 37,9 | 521 |
| 37,5 | 37,3 | 37,4 | 522 |
| 38,1 | 36,9 | 37,5 | 522 |
| 38,2 | 38 | 38,1 | 524 |
| 38,3 | 38,1 | 38,2 | 524 |
| 38,8 | 37,8 | 38,3 | 525 |
| 39,2 | 38,4 | 38,8 | 527 |
| 40 | 38,4 | 39,2 | 528 |
| 40,3 | 39,7 | 40 | 531 |
| 40,5 | 40,1 | 40,3 | 533 |
| 41,7 | 39,3 | 40,5 | 533 |
| 41,7 | 41,7 | 41,7 | 536 |
| 41,8 | 41,6 | 41,7 | 538 |
| 42,3 | 41,3 | 41,8 | 539 |
| 43,9 | 40,7 | 42,3 | 541 |
| 44,1 | 43,7 | 43,9 | 546 |
| 45 | 43,2 | 44,1 | 547 |
| 44,1 | 45,9 | 45 | 548 |


Sensibilidad FC2 uA/ºC
555
550
545
540
535
530
525
520
515
510
505
500
Corriente uA
25
30
35
40
45
50
Temperatura ºC
y = 3,6167x + 386,4
R² = 0,99035
Corriente (uA)
Lineal (Corriente (uA))

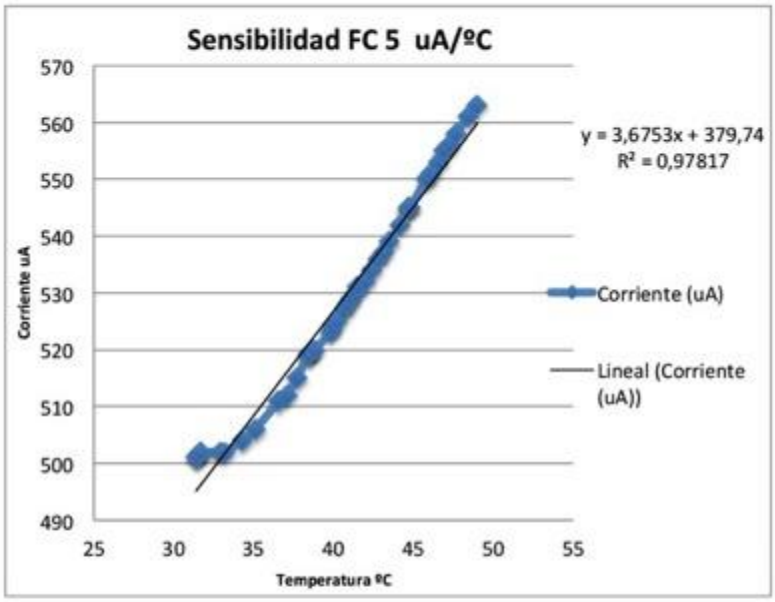
| Calibrado Fuente Corriente FC3 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 31,2 | 30,8 | 31 | 504 |
| 31,4 | 31 | 31,2 | 502 |
| 31,6 | 31,2 | 31,4 | 502 |
| 31,8 | 31,4 | 31,6 | 502 |
| 31,8 | 31,8 | 31,8 | 502 |
| 32,1 | 31,5 | 31,8 | 503 |
| 32,5 | 31,7 | 32,1 | 503 |
| 32,7 | 32,3 | 32,5 | 503 |
| 33,3 | 32,1 | 32,7 | 504 |
| 33,8 | 32,8 | 33,3 | 504 |
| 34,5 | 33,1 | 33,8 | 504 |
| 34,8 | 34,2 | 34,5 | 506 |
| 35,7 | 33,9 | 34,8 | 507 |
| 35,8 | 35,6 | 35,7 | 512 |
| 36,1 | 35,5 | 35,8 | 512 |
| 36,6 | 35,6 | 36,1 | 512 |
| 36,8 | 36,4 | 36,6 | 513 |
| 37,3 | 36,3 | 36,8 | 513 |
| 37,8 | 36,8 | 37,3 | 514 |
| 38,6 | 37 | 37,8 | 515 |
| 38,9 | 38,3 | 38,6 | 517 |
| 39,6 | 38,2 | 38,9 | 517 |
| 39,7 | 39,5 | 39,6 | 519 |
| 40,6 | 38,8 | 39,7 | 519 |
| 40,9 | 40,3 | 40,6 | 522 |
| 41 | 40,8 | 40,9 | 523 |
| 41,4 | 40,6 | 41 | 523 |
| 41,7 | 41,1 | 41,4 | 524 |
| 41,7 | 41,7 | 41,7 | 525 |
| 42 | 41,4 | 41,7 | 525 |
| 42,4 | 41,6 | 42 | 526 |
| 42,5 | 42,3 | 42,4 | 528 |
| 43 | 42 | 42,5 | 528 |
| 43,4 | 42,6 | 43 | 530 |
| 44,7 | 42,1 | 43,4 | 531 |
| 44,8 | 44,6 | 44,7 | 534 |
| 46,2 | 43,4 | 44,8 | 540 |
| 48,4 | 44 | 46,2 | 544 |
| 49,7 | 47,1 | 48,4 | 548 |
| 49,9 | 49,5 | 49,7 | 552 |


Sensibilidad FC 3 uA/ºC
560
550
540
530
520
510
500
490
Corriente uA
25
30
35
40
45
50
55
Temperatura ºC
y = 2,6288x + 417,26
R² = 0,972
Corriente (uA)
Lineal (Corriente (uA))

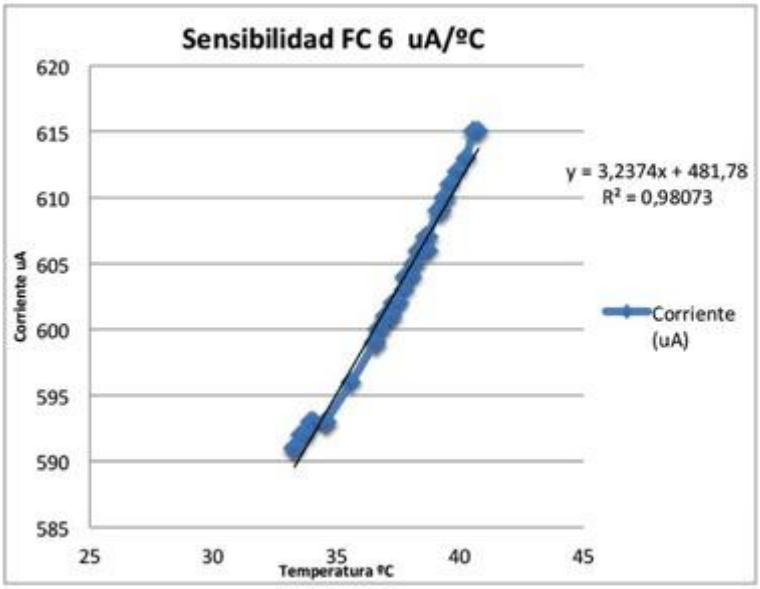
| Calibrado Fuente Corriente FC4 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 31,3 | 30,9 | 31,1 | 503 |
| 31,7 | 30,9 | 31,3 | 503 |
| 31,8 | 31,6 | 31,7 | 503 |
| 32 | 31,6 | 31,8 | 503 |
| 32,4 | 31,6 | 32 | 502 |
| 32,6 | 32,2 | 32,4 | 502 |
| 33,7 | 31,5 | 32,6 | 503 |
| 34 | 33,4 | 33,7 | 504 |
| 36,6 | 31,4 | 34 | 505 |
| 36,9 | 36,3 | 36,6 | 512 |
| 37,6 | 36,2 | 36,9 | 512 |
| 37,7 | 37,5 | 37,6 | 512 |
| 37,8 | 37,6 | 37,7 | 513 |
| 38,1 | 37,5 | 37,8 | 513 |
| 38,5 | 37,7 | 38,1 | 513 |
| 38,8 | 38,2 | 38,5 | 514 |
| 38,8 | 38,8 | 38,8 | 514 |
| 39,3 | 38,3 | 38,8 | 517 |
| 39,5 | 39,1 | 39,3 | 517 |
| 39,8 | 39,2 | 39,5 | 518 |
| 40 | 39,6 | 39,8 | 518 |
| 40,2 | 39,8 | 40 | 519 |
| 40,4 | 40 | 40,2 | 519 |
| 40,6 | 40,2 | 40,4 | 520 |
| 40,7 | 40,5 | 40,6 | 520 |
| 41,4 | 40 | 40,7 | 521 |
| 41,5 | 41,3 | 41,4 | 523 |
| 41,8 | 41,2 | 41,5 | 523 |
| 41,9 | 41,7 | 41,8 | 524 |
| 42,1 | 41,7 | 41,9 | 524 |
| 42,9 | 41,3 | 42,1 | 525 |
| 43 | 42,8 | 42,9 | 526 |
| 43,2 | 42,8 | 43 | 527 |
| 43,6 | 42,8 | 43,2 | 528 |
| 44 | 43,2 | 43,6 | 529 |
| 44,4 | 43,6 | 44 | 530 |
| 44,9 | 43,9 | 44,4 | 531 |
| 45,2 | 44,6 | 44,9 | 533 |
| 46,6 | 43,8 | 45,2 | 534 |
| 46,6 | 46,6 | 46,6 | 537 |


Sensibilidad FC 4 uA/ºC
540
535
530
525
520
515
510
505
500
495
Corriente uA
25
30
35
40
45
50
Temperatura ºC
y = 2,2819x + 428,45
R² = 0,97629
Corriente (uA)
Lineal (Corriente (uA))

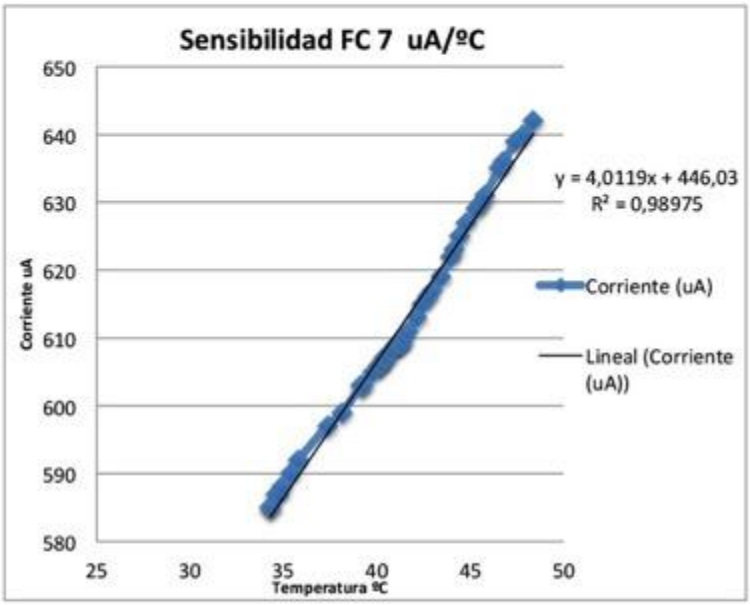
| Calibrado Fuente Corriente FC5 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 31,2 | 31,6 | 31,4 | 501 |
| 31,6 | 31,4 | 31,5 | 501 |
| 31,8 | 31,6 | 31,7 | 502 |
| 31,8 | 31,6 | 31,7 | 502 |
| 32,8 | 33 | 32,9 | 502 |
| 33,1 | 33,3 | 33,2 | 502 |
| 34,2 | 34,4 | 34,3 | 504 |
| 35 | 35,2 | 35,1 | 506 |
| 36,8 | 36,4 | 36,6 | 511 |
| 37 | 37,2 | 37,1 | 512 |
| 37,5 | 37,9 | 37,7 | 515 |
| 38,5 | 38,3 | 38,4 | 519 |
| 38,7 | 38,3 | 38,5 | 519 |
| 38,9 | 38,5 | 38,7 | 520 |
| 39 | 38,6 | 38,8 | 520 |
| 40 | 39,6 | 39,8 | 523 |
| 40 | 39,8 | 39,9 | 524 |
| 40,7 | 39,7 | 40,2 | 525 |
| 40,9 | 40,5 | 40,7 | 527 |
| 41,1 | 40,7 | 40,9 | 528 |
| 41,4 | 40,8 | 41,1 | 529 |
| 41,5 | 41,3 | 41,4 | 530 |
| 42 | 41 | 41,5 | 531 |
| 42,4 | 41,6 | 42 | 532 |
| 42,8 | 42 | 42,4 | 534 |
| 43,1 | 42,5 | 42,8 | 536 |
| 43,5 | 42,7 | 43,1 | 537 |
| 44,2 | 42,8 | 43,5 | 539 |
| 44,7 | 43,7 | 44,2 | 542 |
| 44,7 | 44,7 | 44,7 | 545 |
| 44,8 | 44,6 | 44,7 | 545 |
| 45,8 | 43,8 | 44,8 | 545 |
| 46,1 | 45,5 | 45,8 | 550 |
| 46,6 | 45,6 | 46,1 | 551 |
| 46,9 | 46,3 | 46,6 | 553 |
| 47,3 | 46,5 | 46,9 | 555 |
| 47,7 | 46,9 | 47,3 | 556 |
| 48,4 | 47 | 47,7 | 558 |
| 49 | 47,8 | 48,4 | 561 |
| 49,1 | 48,9 | 49 | 563 |


Sensibilidad FC 5 uA/ºC
570
560
550
540
530
520
510
500
490
Corriente uA
25
30
35
40
45
50
55
Temperatura ºC
y = 3,6753x + 379,74
R² = 0,97817
Corriente (uA)
Lineal (Corriente (uA))

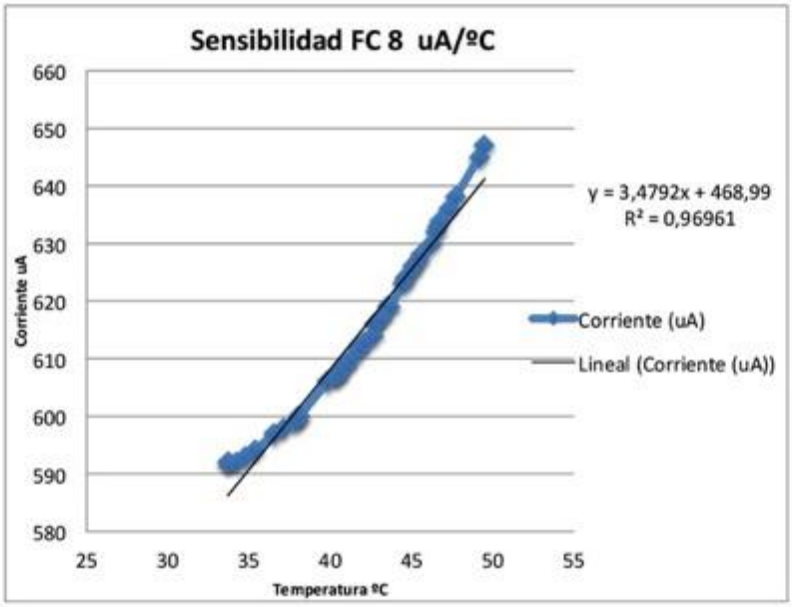
| Calibrado Fuente Corriente FC6 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 33,4 | 33,2 | 33,3 | 591 |
| 33,4 | 33,2 | 33,3 | 591 |
| 33,7 | 33,4 | 33,55 | 592 |
| 33,9 | 33,6 | 33,75 | 592 |
| 34,2 | 33,7 | 33,95 | 593 |
| 34,4 | 33,8 | 34,1 | 593 |
| 34,9 | 34,2 | 34,55 | 593 |
| 34,9 | 34,2 | 34,55 | 593 |
| 36,1 | 35,1 | 35,6 | 596 |
| 37,1 | 36 | 36,55 | 599 |
| 37,2 | 36 | 36,6 | 599 |
| 37,3 | 36,1 | 36,7 | 600 |
| 37,4 | 36,1 | 36,75 | 600 |
| 37,7 | 36,3 | 37 | 601 |
| 37,7 | 36,3 | 37 | 601 |
| 37,8 | 36,4 | 37,1 | 601 |
| 38 | 36,5 | 37,25 | 601 |
| 38,1 | 36,5 | 37,3 | 602 |
| 38,2 | 36,6 | 37,4 | 602 |
| 38,3 | 36,7 | 37,5 | 602 |
| 38,6 | 36,9 | 37,75 | 603 |
| 38,7 | 36,9 | 37,8 | 604 |
| 38,8 | 37 | 37,9 | 604 |
| 39 | 37,1 | 38,05 | 604 |
| 39,1 | 37,2 | 38,15 | 605 |
| 39,2 | 37,2 | 38,2 | 605 |
| 39,3 | 37,3 | 38,3 | 606 |
| 39,4 | 37,4 | 38,4 | 606 |
| 39,5 | 37,9 | 38,7 | 606 |
| 39,6 | 37,6 | 38,6 | 607 |
| 39,8 | 37,6 | 38,7 | 607 |
| 40,2 | 38 | 39,1 | 609 |
| 40,4 | 38,1 | 39,25 | 609 |
| 40,5 | 38,2 | 39,35 | 610 |
| 40,6 | 38,2 | 39,4 | 610 |
| 40,8 | 38,4 | 39,6 | 611 |
| 41,2 | 38,6 | 39,9 | 612 |
| 41,6 | 38,9 | 40,25 | 613 |
| 42 | 39,1 | 40,55 | 615 |
| 42,3 | 39,2 | 40,75 | 615 |


Sensibilidad FC 6 uA/ºC
620
615
610
605
600
595
590
585
Corriente uA
25
30
35
40
45
Temperatura ºC
y = 3,2374x + 481,78
R² = 0,98073
Corriente (uA)

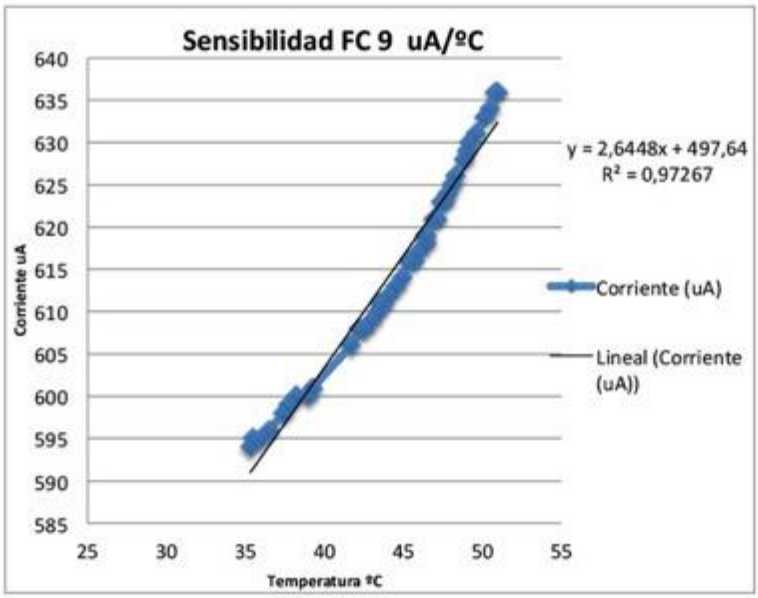
| Calibrado Fuente Corriente FC7 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 34,3 | 34,3 | 34,3 | 585 |
| 34,4 | 34,3 | 34,35 | 585 |
| 34,6 | 34,6 | 34,6 | 587 |
| 34,7 | 34,8 | 34,75 | 587 |
| 34,9 | 35 | 34,95 | 588 |
| 35,3 | 35,5 | 35,4 | 590 |
| 35,7 | 35,9 | 35,8 | 592 |
| 37,5 | 37,2 | 37,35 | 597 |
| 38,3 | 38 | 38,15 | 599 |
| 39,5 | 38,8 | 39,15 | 603 |
| 39,6 | 38,9 | 39,25 | 603 |
| 40,2 | 39,4 | 39,8 | 605 |
| 40,6 | 39,7 | 40,15 | 606 |
| 40,7 | 39,8 | 40,25 | 606 |
| 40,9 | 39,9 | 40,4 | 607 |
| 41 | 40 | 40,5 | 607 |
| 41,3 | 40,2 | 40,75 | 608 |
| 41,6 | 40,4 | 41 | 609 |
| 41,8 | 40,5 | 41,15 | 609 |
| 41,8 | 40,5 | 41,15 | 609 |
| 42 | 40,6 | 41,3 | 609 |
| 42,2 | 40,7 | 41,45 | 610 |
| 42,4 | 40,9 | 41,65 | 611 |
| 42,9 | 41,4 | 42,15 | 613 |
| 43,2 | 41,6 | 42,4 | 615 |
| 43,5 | 42 | 42,75 | 616 |
| 43,7 | 42,2 | 42,95 | 617 |
| 44,2 | 42,7 | 43,45 | 619 |
| 44,7 | 43,3 | 44 | 622 |
| 44,8 | 43,5 | 44,15 | 623 |
| 45,1 | 43,8 | 44,45 | 625 |
| 45,4 | 44,2 | 44,8 | 627 |
| 45,9 | 44,8 | 45,35 | 629 |
| 46,1 | 45,1 | 45,6 | 630 |
| 46,3 | 45,3 | 45,8 | 631 |
| 47 | 46,1 | 46,55 | 635 |
| 47,2 | 46,4 | 46,8 | 636 |
| 47,8 | 47,1 | 47,45 | 639 |
| 48,1 | 47,6 | 47,85 | 640 |
| 48,7 | 48,1 | 48,4 | 642 |


Sensibilidad FC 7 uA/ºC
650
640
630
620
610
600
590
580
Corriente uA
25
30
35
40
45
50
Temperatura ºC
y = 4,0119x + 446,03
R² = 0,98975
Corriente (uA)
Lineal (Corriente (uA))

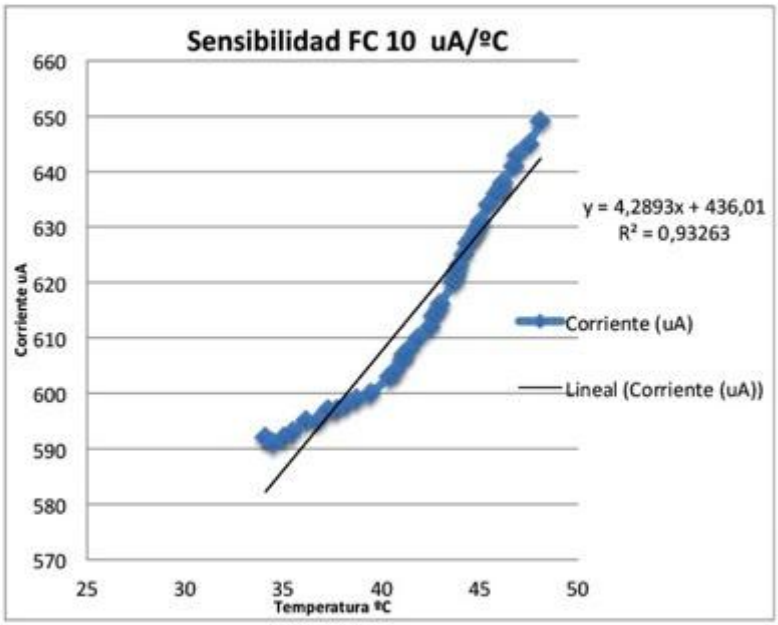
| Calibrado Fuente Corriente FC8 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 33,6 | 33,8 | 33,7 | 592 |
| 33,7 | 33,8 | 33,75 | 592 |
| 34,1 | 34,4 | 34,25 | 592 |
| 34,7 | 34,9 | 34,8 | 593 |
| 35,3 | 35,4 | 35,35 | 594 |
| 36,4 | 36,7 | 36,55 | 597 |
| 37,1 | 37,2 | 37,15 | 598 |
| 37,9 | 37,9 | 37,9 | 599 |
| 38 | 38,2 | 38,1 | 600 |
| 38 | 38,2 | 38,1 | 600 |
| 39,8 | 39,9 | 39,85 | 606 |
| 40,3 | 40,3 | 40,3 | 607 |
| 40,5 | 40,5 | 40,5 | 607 |
| 40,7 | 40,6 | 40,65 | 608 |
| 40,9 | 40,9 | 40,9 | 609 |
| 41 | 41 | 41 | 609 |
| 41,3 | 41,3 | 41,3 | 610 |
| 41,7 | 41,7 | 41,7 | 612 |
| 41,9 | 41,9 | 41,9 | 612 |
| 42,1 | 42,1 | 42,1 | 613 |
| 42,4 | 42,9 | 42,65 | 614 |
| 42,7 | 42,8 | 42,75 | 616 |
| 43 | 43,1 | 43,05 | 617 |
| 43,1 | 43,2 | 43,15 | 617 |
| 43,5 | 43,6 | 43,55 | 619 |
| 43,6 | 43,8 | 43,7 | 619 |
| 44,4 | 44,6 | 44,5 | 623 |
| 44,6 | 44,7 | 44,65 | 624 |
| 44,8 | 45,1 | 44,95 | 625 |
| 45 | 45,2 | 45,1 | 626 |
| 45,3 | 45,6 | 45,45 | 627 |
| 45,4 | 45,7 | 45,55 | 628 |
| 45,8 | 46,7 | 46,25 | 630 |
| 46,3 | 46,7 | 46,5 | 632 |
| 46,4 | 46,8 | 46,6 | 633 |
| 46,6 | 46,9 | 46,75 | 634 |
| 47,1 | 47,5 | 47,3 | 636 |
| 47,5 | 47,9 | 47,7 | 638 |
| 48,9 | 49,4 | 49,15 | 645 |
| 49,3 | 49,7 | 49,5 | 647 |


Sensibilidad FC 8 uA/ºC
660
650
640
630
620
610
600
590
580
Corriente uA
25
30
35
40
45
50
55
Temperatura ºC
y = 3,4792x + 468,99
R² = 0,96961
Corriente (uA)
Lineal (Corriente (uA))

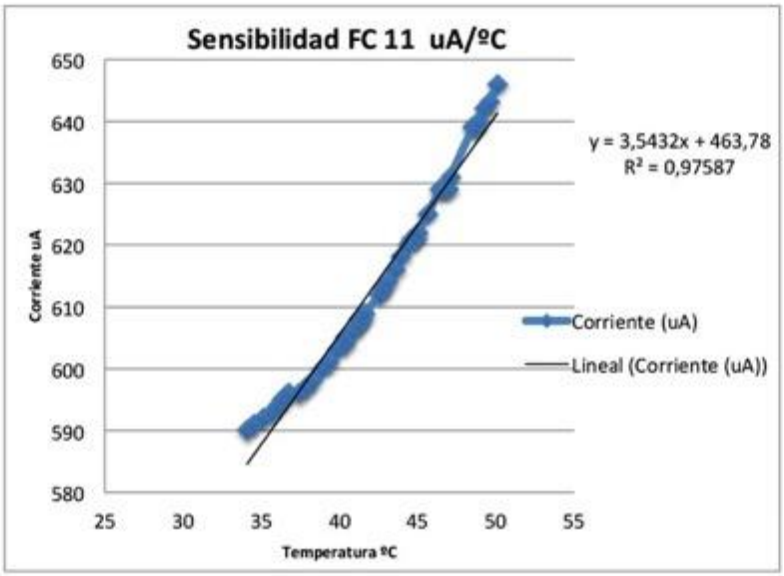
| Calibrado Fuente Corriente FC9 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 35,1 | 35,5 | 35,3 | 594 |
| 35,2 | 35,8 | 35,5 | 595 |
| 35,7 | 36,3 | 36 | 595 |
| 36,2 | 36,8 | 36,5 | 596 |
| 37 | 37,8 | 37,4 | 598 |
| 37,3 | 38,1 | 37,7 | 599 |
| 37,7 | 38,7 | 38,2 | 600 |
| 38,2 | 39,7 | 38,95 | 600 |
| 38,5 | 40 | 39,25 | 601 |
| 40,9 | 42,4 | 41,65 | 606 |
| 41,4 | 43 | 42,2 | 608 |
| 41,7 | 43,4 | 42,55 | 608 |
| 42,1 | 43,9 | 43 | 609 |
| 42,5 | 44,3 | 43,4 | 610 |
| 42,6 | 44,4 | 43,5 | 611 |
| 42,8 | 44,7 | 43,75 | 611 |
| 43,2 | 45 | 44,1 | 612 |
| 43,5 | 45,5 | 44,5 | 613 |
| 43,9 | 45,9 | 44,9 | 614 |
| 44,3 | 46,4 | 45,35 | 616 |
| 44,5 | 46,8 | 45,65 | 616 |
| 44,8 | 47 | 45,9 | 617 |
| 45 | 47,9 | 46,45 | 618 |
| 45,2 | 47,5 | 46,35 | 619 |
| 45,2 | 47,7 | 46,45 | 619 |
| 45,7 | 48,2 | 46,95 | 621 |
| 45,8 | 48,4 | 47,1 | 621 |
| 46,1 | 48,7 | 47,4 | 623 |
| 46,3 | 48,9 | 47,6 | 623 |
| 46,5 | 49,2 | 47,85 | 624 |
| 46,7 | 49,4 | 48,05 | 625 |
| 46,9 | 49,7 | 48,3 | 626 |
| 47,4 | 50,3 | 48,85 | 628 |
| 47,6 | 50,5 | 49,05 | 629 |
| 47,7 | 50,7 | 49,2 | 630 |
| 48,2 | 51 | 49,6 | 631 |
| 48,5 | 51,8 | 50,15 | 633 |
| 48,9 | 52,1 | 50,5 | 634 |
| 49,2 | 52,4 | 50,8 | 636 |
| 49,4 | 52,5 | 50,95 | 636 |


Sensibilidad FC 9 uA/ºC
640
635
630
625
620
615
610
605
600
595
590
585
Corriente uA
25
30
35
40
45
50
55
Temperatura ºC
y = 2,6448x + 497,64
R² = 0,97267
Corriente (uA)
Lineal (Corriente (uA))

| Calibrado Fuente Corriente FC10 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 34,1 | 34,1 | 34,1 | 592 |
| 34,5 | 34,5 | 34,5 | 591 |
| 35 | 35,1 | 35,05 | 592 |
| 35,6 | 35,4 | 35,5 | 593 |
| 36,4 | 35,9 | 36,15 | 595 |
| 37,1 | 36,4 | 36,75 | 595 |
| 37,7 | 36,9 | 37,3 | 597 |
| 38,1 | 37,3 | 37,7 | 597 |
| 38,6 | 37,8 | 38,2 | 598 |
| 39,2 | 38,3 | 38,75 | 599 |
| 40,1 | 38,8 | 39,45 | 600 |
| 40,9 | 39,9 | 40,4 | 603 |
| 41,1 | 40,3 | 40,7 | 604 |
| 41,4 | 40,8 | 41,1 | 606 |
| 41,4 | 40,9 | 41,15 | 607 |
| 41,6 | 41,2 | 41,4 | 608 |
| 42,1 | 41,7 | 41,9 | 610 |
| 42,9 | 42,1 | 42,5 | 612 |
| 42,8 | 42,5 | 42,65 | 614 |
| 43 | 42,7 | 42,85 | 615 |
| 43,1 | 42,9 | 43 | 616 |
| 43,7 | 43,6 | 43,65 | 620 |
| 43,8 | 43,7 | 43,75 | 621 |
| 43,9 | 43,8 | 43,85 | 622 |
| 44 | 43,9 | 43,95 | 623 |
| 44,2 | 44,2 | 44,2 | 625 |
| 44,4 | 44,4 | 44,4 | 627 |
| 44,6 | 44,6 | 44,6 | 628 |
| 44,8 | 44,8 | 44,8 | 629 |
| 44,99 | 44,9 | 44,945 | 630 |
| 45 | 45,1 | 45,05 | 631 |
| 45,4 | 45,5 | 45,45 | 634 |
| 45,8 | 45,9 | 45,85 | 636 |
| 46 | 46,1 | 46,05 | 637 |
| 46 | 46,2 | 46,1 | 637 |
| 46,1 | 46,3 | 46,2 | 638 |
| 46,7 | 46,8 | 46,75 | 641 |
| 46,8 | 47 | 46,9 | 643 |
| 47,4 | 47,6 | 47,5 | 645 |
| 48 | 48,2 | 48,1 | 649 |

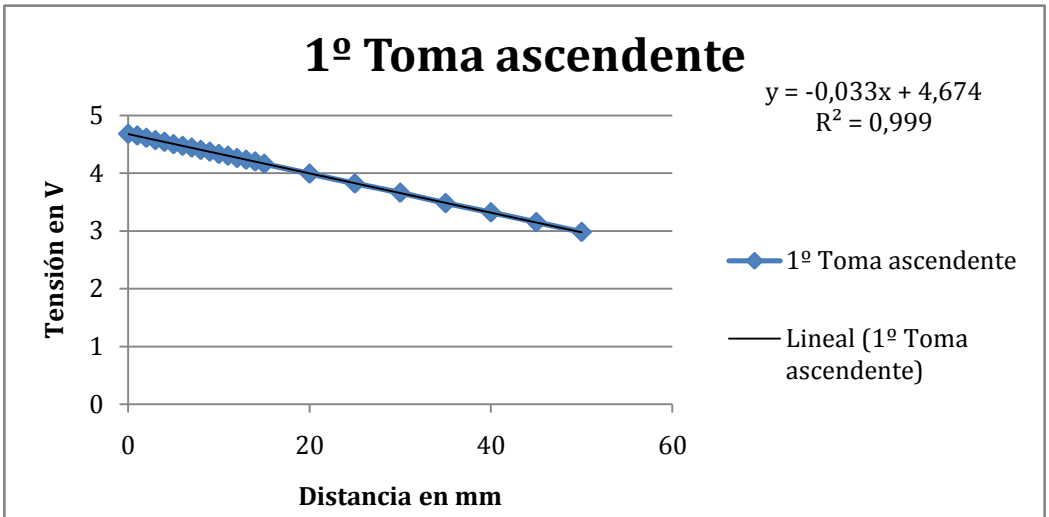

Sensibilidad FC 10 uA/ºC
660
650
640
630
620
610
600
590
580
570
Corriente uA
25
30
35
40
45
50
Temperatura ºC
y = 4,2893x + 436,01
R² = 0,93263
Corriente (uA)
Lineal (Corriente (uA))

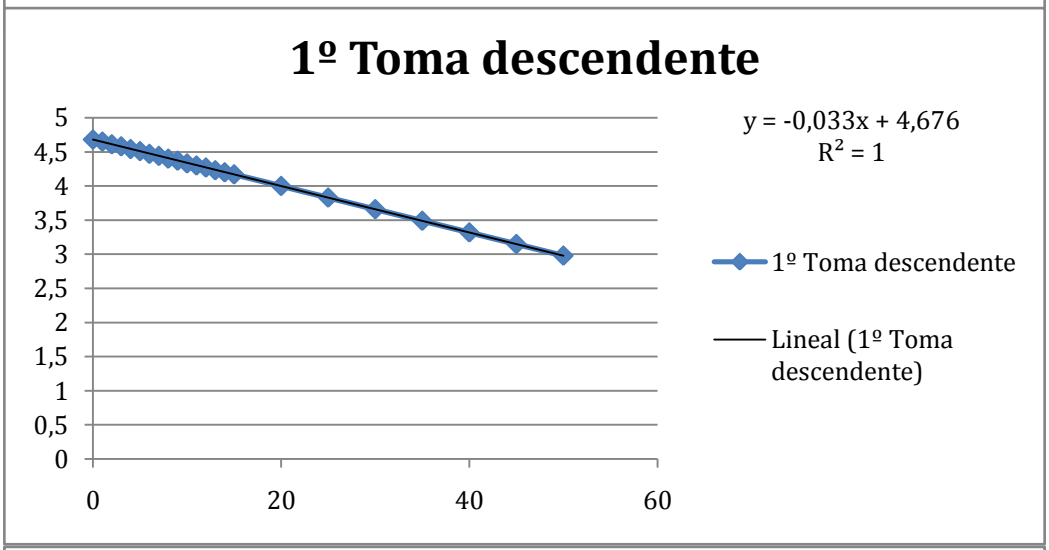
| Calibrado Fuente Corriente FC11 | | | |
|---|---|---|---|
| Temperatura 1 (ºC) | Temperatura 2 (ºC) | Media Temperatura (ºC) | Corriente (uA) |
| 34 | 34,2 | 34,1 | 590 |
| 34,4 | 34,7 | 34,55 | 591 |
| 34,9 | 35,4 | 35,15 | 592 |
| 35,4 | 36 | 35,7 | 593 |
| 35,8 | 36,3 | 36,05 | 594 |
| 36,1 | 36,6 | 36,35 | 595 |
| 36,6 | 36,9 | 36,75 | 596 |
| 37,1 | 37,8 | 37,45 | 596 |
| 37,6 | 38,2 | 37,9 | 597 |
| 37,9 | 38,6 | 38,25 | 598 |
| 38,5 | 39,3 | 38,9 | 600 |
| 38,9 | 39,8 | 39,35 | 601 |
| 39,4 | 40,3 | 39,85 | 603 |
| 39,9 | 40,8 | 40,35 | 604 |
| 40 | 40,8 | 40,4 | 605 |
| 40,5 | 41,3 | 40,9 | 606 |
| 40,8 | 41,6 | 41,2 | 607 |
| 40,9 | 41,7 | 41,3 | 608 |
| 41,1 | 41,9 | 41,5 | 608 |
| 41,2 | 42,1 | 41,65 | 609 |
| 42,1 | 43,1 | 42,6 | 612 |
| 42,4 | 43,3 | 42,85 | 613 |
| 42,6 | 43,6 | 43,1 | 614 |
| 43,1 | 44,1 | 43,6 | 616 |
| 43,4 | 44,5 | 43,95 | 618 |
| 43,9 | 45 | 44,45 | 620 |
| 44,1 | 45,2 | 44,65 | 621 |
| 44,3 | 45,5 | 44,9 | 621 |
| 44,5 | 45,6 | 45,05 | 622 |
| 45,1 | 46,2 | 45,65 | 625 |
| 45,8 | 47,1 | 46,45 | 629 |
| 46 | 47,3 | 46,65 | 629 |
| 46,1 | 47,9 | 47 | 629 |
| 46,5 | 47,7 | 47,1 | 631 |
| 47,9 | 49,2 | 48,55 | 639 |
| 48 | 49,4 | 48,7 | 639 |
| 48,2 | 49,6 | 48,9 | 640 |
| 48,5 | 50,1 | 49,3 | 642 |
| 48,8 | 50,4 | 49,6 | 643 |
| 49,2 | 51 | 50,1 | 646 |

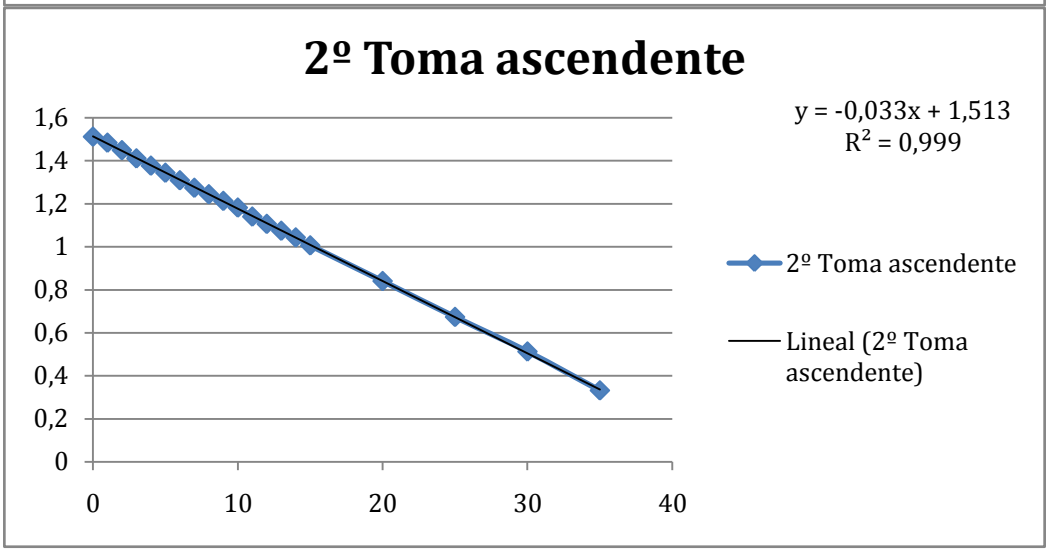

Sensibilidad FC 11 uA/ºC
650
640
630
620
610
600
590
580
Corriente uA
25
30
35
40
45
50
55
Temperatura ºC
y = 3,5432x + 463,78
R² = 0,97587
Corriente (uA)
Lineal (Corriente (uA))

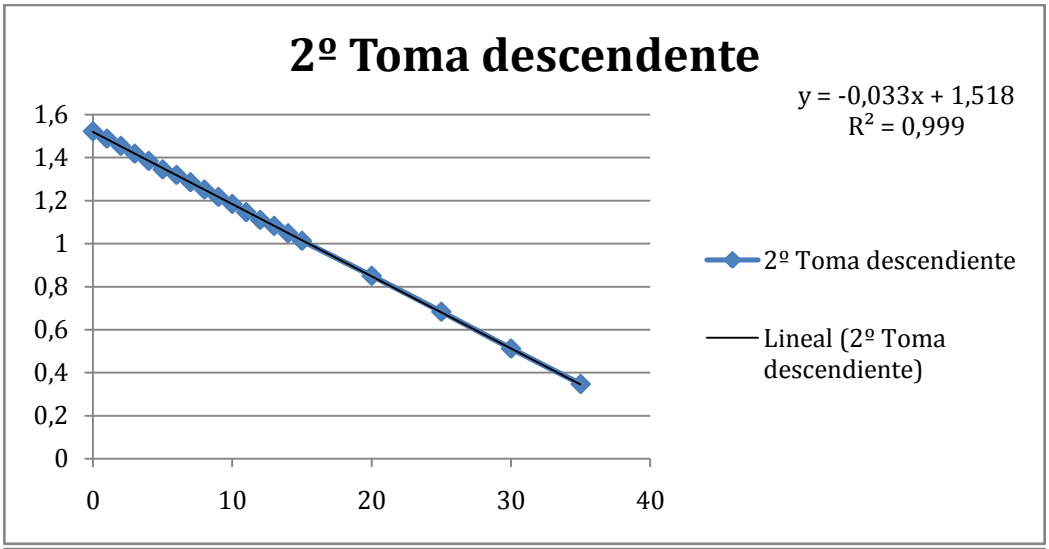
*Resultados calibración sensor de posición*

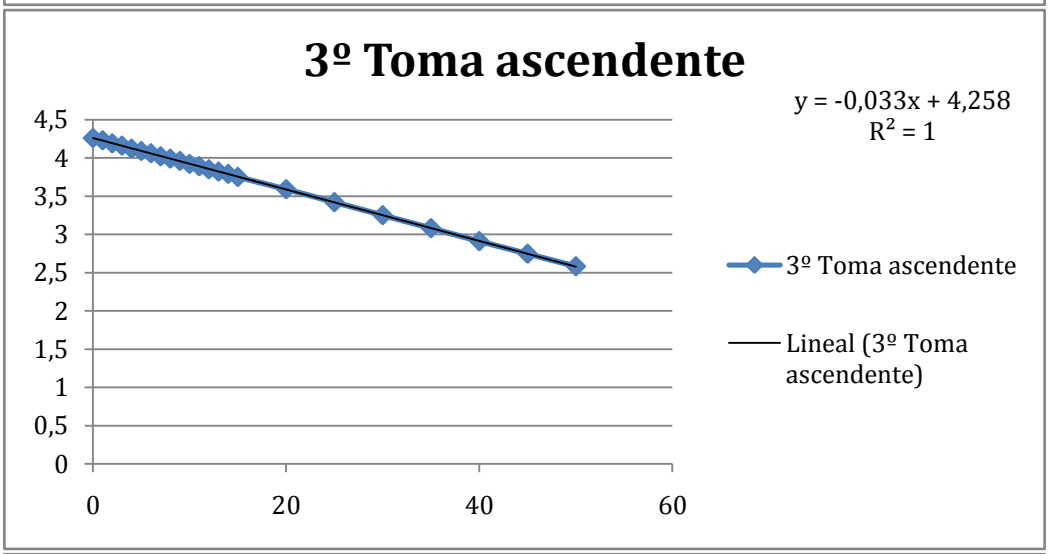
| Distancia (mm) | 1º Toma (V) | 2º Toma (V) | 3º Toma (V) | |
|---|---|---|---|---|
| 0 | 4,68 | 1,512 | 4,26 | |
| 1 | 4,65 | 1,484 | 4,23 | |
| 2 | 4,61 | 1,449 | 4,19 | |
| 3 | 4,57 | 1,411 | 4,16 | |
| 4 | 4,54 | 1,377 | 4,12 | |
| 5 | 4,5 | 1,344 | 4,09 | |
| 6 | 4,47 | 1,309 | 4,06 | |
| 7 | 4,44 | 1,274 | 4,02 | |
| 8 | 4,4 | 1,244 | 3,99 | |
| 9 | 4,37 | 1,213 | 3,96 | |
| 10 | 4,33 | 1,182 | 3,92 | |
| 11 | 4,3 | 1,14 | 3,89 | Ascendente |
| 12 | 4,26 | 1,106 | 3,85 | |
| 13 | 4,23 | 1,074 | 3,82 | |
| 14 | 4,2 | 1,043 | 3,79 | |
| 15 | 4,16 | 1,006 | 3,75 | |
| 20 | 3,99 | 0,84 | 3,59 | |
| 25 | 3,82 | 0,673 | 3,42 | |
| 30 | 3,66 | 0,512 | 3,25 | |
| 35 | 3,48 | 0,331 | 3,08 | |
| 40 | 3,32 | 0,261 | 2,91 | |
| 45 | 3,15 | 0,261 | 2,745 | |
| 50 | 2,98 | 0,261 | 2,581 | |
| 50 | 2,98 | 0,261 | 2,581 | |
| 45 | 3,15 | 0,261 | 2,746 | |
| 40 | 3,32 | 0,261 | 2,918 | |
| 35 | 3,49 | 0,347 | 3,086 | |
| 30 | 3,66 | 0,512 | 3,256 | |
| 25 | 3,83 | 0,683 | 3,42 | |
| 20 | 4 | 0,85 | 3,59 | |
| 15 | 4,17 | 1,013 | 3,75 | |
| 14 | 4,2 | 1,048 | 3,79 | |
| 13 | 4,23 | 1,082 | 3,82 | |
| 12 | 4,27 | 1,11 | 3,85 | |
| 11 | 4,3 | 1,146 | 3,89 | Descendente |
| 10 | 4,33 | 1,184 | 3,93 | |
| 9 | 4,37 | 1,217 | 3,96 | |
| 8 | 4,4 | 1,251 | 3,99 | |
| 7 | 4,44 | 1,285 | 4,02 | |
| 6 | 4,47 | 1,319 | 4,06 | |
| 5 | 4,51 | 1,345 | 4,09 | |
| 4 | 4,54 | 1,384 | 4,12 | |
| 3 | 4,58 | 1,418 | 4,16 | |
| 2 | 4,61 | 1,454 | 4,19 | |
| 1 | 4,65 | 1,488 | 4,22 | |
| 0 | 4,68 | 1,522 | 4,26 | |

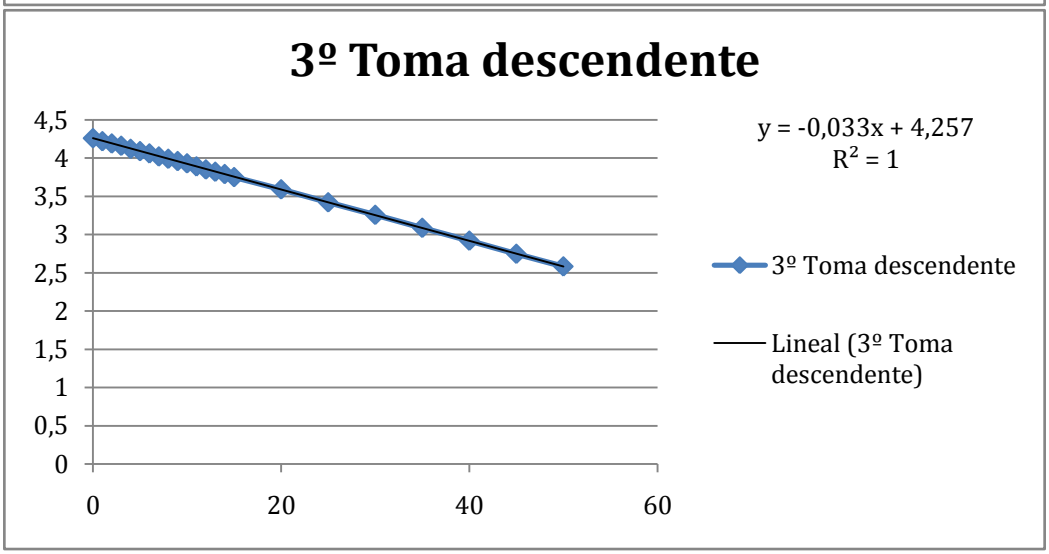

1º Toma ascendente
y = -0,033x + 4,674
R² = 0,999
Tensión en V
Distancia en mm
1º Toma ascendente
Lineal (1º Toma ascendente)


1º Toma descendente
y = -0,033x + 4,676
R² = 1
1º Toma descendente
Lineal (1º Toma descendente)


2º Toma ascendente
y = -0,033x + 1,513
R² = 0,999
2º Toma ascendente
Lineal (2º Toma ascendente)


2º Toma descendente
y = -0,033x + 1,518
R² = 0,999
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
0
10
20
30
40
2º Toma descendiente
Lineal (2º Toma descendiente)


3º Toma ascendente
y = -0,033x + 4,258
R² = 1
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0
20
40
60
3º Toma ascendente
Lineal (3º Toma ascendente)


3º Toma descendente
y = -0,033x + 4,257
R² = 1
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0
20
40
60
3º Toma descendente
Lineal (3º Toma descendente)

## 1.9 Anexo III – Librerías módulo de radiofrecuencia

*Librería NRF2401*

```
/*
 *  Nrf2401.h
 *  A simplistic interface for using Sparkfun's Nrf2401A breakout boards with Arduino
 *  Original code for http://labs.ideo.com by Jesse Tane March 2009
 *
 *  License:
 *  --------
 *  This is free software. You can redistribute it and/or modify it under
 *  the terms of Creative Commons Attribution 3.0 United States License.
 *  To view a copy of this license, visit http://creativecommons.org/licenses/by/3.0/us/
 *  or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
 *
 *  Notes:
 *  ------
 *  For documentation on how to use this library, please visit http://www.arduino.cc/playground/Main/InterfacingWithHardware
 *  Pin connections should be as follows for Arduino:
*
 *  DR1 = 2  (digital pin 2)
 *  CE  = 3
 *  CS  = 4
*  CLK = 5
 *  DAT = 6
 *
 */

#include <avr/io.h>
#include <util/delay.h>

#define NRF2401_BUFFER_SIZE 25

#define DR1 4
#define CE  8
#define CS  16
#define CLK 32
#define DAT 64
```

```
#define SELECT_CHIP   PORTD |= CS
#define DESELECT_CHIP PORTD &= ~CS
#define ENABLE_CHIP   PORTD |= CE
#define DISABLE_CHIP  PORTD &= ~CE
#define CYCLE_CLOCK   PORTD |= CLK, _delay_us(1), PORTD &=
~CLK
#define TX_DATA_HI    PORTD |= DAT
#define TX_DATA_LO    PORTD &= ~DAT
#define RX_DATA_HI    (PIND & DAT)
#define DATA_READY    (PIND & DR1)
class Nrf2401
{
 public:
 // properties
 volatile unsigned char data[NRF2401_BUFFER_SIZE];
 volatile unsigned int remoteAddress;
 volatile unsigned int localAddress;
 volatile unsigned char dataRate;
 volatile unsigned char channel;
 volatile unsigned char power;
 volatile unsigned char mode;

 // methods

 Nrf2401(void);
 void rxMode(unsigned char messageSize=0);
 void txMode(unsigned char messageSize=0);
 void write(unsigned char dataByte);
 void write(unsigned char* dataBuffer=0);
 void read(unsigned char* dataBuffer=0);
 bool available(void);

 // you shouldn't need to use anything below this point..

 volatile unsigned char payloadSize;
 volatile unsigned char configuration[15];
 void configure(void);
 void loadConfiguration(bool modeSwitchOnly=false);
 void loadByte(unsigned char byte);
};
```

### *Librería RF24*

```
/*
 Copyright (C) 2011 J. Coliz <maniacbug@ymail.com>

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 version 2 as published by the Free Software Foundation.
 */

/**
 * @file RF24.h
 *
 * Class declaration for RF24 and helper enums
 */

#ifndef __RF24_H__
#define __RF24_H__

#include "RF24_config.h"

#if defined (RF24_LINUX)
  #include "utility/includes.h"
#elif LITTLEWIRE
  #include <LittleWireSPI/LittleWireSPI.h>
#elif defined SOFTSPI
  #include <DigitalIO.h>
#endif

/**
 * Power Amplifier level.
 *
 * For use with setPALevel()
 */
typedef enum { RF24_PA_MIN = 0,RF24_PA_LOW, RF24_PA_HIGH,
RF24_PA_MAX, RF24_PA_ERROR } rf24_pa_dbm_e ;

/**
 * Data rate.  How fast data moves through the air.
 *
 * For use with setDataRate()
 */
typedef enum { RF24_1MBPS = 0, RF24_2MBPS, RF24_250KBPS } rf24_datarate_e;

/**
 * CRC Length.  How big (if any) of a CRC is included.
 *
 * For use with setCRCLength()
 */
```

```
typedef enum { RF24_CRC_DISABLED = 0, RF24_CRC_8, RF24_CRC_16 }
rf24_crclength_e;

/**
 * Driver for nRF24L01(+) 2.4GHz Wireless Transceiver
 */

class RF24
{
private:
#ifdef SOFTSPI
  SoftSPI<SOFT_SPI_MISO_PIN, SOFT_SPI_MOSI_PIN, SOFT_SPI_SCK_PIN,
SPI_MODE> spi;
#elif defined (SPI_UART)
  SPIUARTClass uspi;
#endif

#if defined (RF24_LINUX)
  SPI spi;
#endif
#if defined (MRAA)
  GPIO gpio;
#endif

  uint8_t ce_pin; /**< "Chip Enable" pin, activates the RX or TX role */
  uint8_t csn_pin; /**< SPI Chip select */
#if defined (RF24_LINUX)
  uint16_t spi_speed; /**< SPI Bus Speed */
  uint8_t spi_rxbuff[32+1] ; //SPI receive buffer (payload max 32 bytes)
  uint8_t spi_txbuff[32+1] ; //SPI transmit buffer (payload max 32 bytes + 1 byte for the
command)
#endif
  bool p_variant; /* False for RF24L01 and true for RF24L01P */
  uint8_t payload_size; /**< Fixed size of payloads */
  bool dynamic_payloads_enabled; /**< Whether dynamic payloads are enabled. */
  uint8_t pipe0_reading_address[5]; /**< Last address set on pipe 0 for reading. */
  uint8_t addr_width; /**< The address width to use - 3,4 or 5 bytes. */
  uint32_t txRxDelay; /**< Var for adjusting delays depending on datarate */


protected:
  /**
   * SPI transactions
   *
   * Common code for SPI transactions including CSN toggle
   *
   */
  inline void beginTransaction();

  inline void endTransaction();
```

```
public:

  /**
   * @name Primary public interface
   *
   *  These are the main methods you need to operate the chip
   */
  /**@{*/

  /**
   * Arduino Constructor
   *
   * Creates a new instance of this driver.  Before using, you create an instance
   * and send in the unique pins that this chip is connected to.
   *
   * @param _cepin The pin attached to Chip Enable on the RF module
   * @param _cspin The pin attached to Chip Select
   */
  RF24(uint8_t _cepin, uint8_t _cspin);
  //#if defined (RF24_LINUX)

    /**
   * Optional Raspberry Pi Constructor
   *
   * Creates a new instance of this driver.  Before using, you create an instance
   * and send in the unique pins that this chip is connected to.
   *
   * @param _cepin The pin attached to Chip Enable on the RF module
   * @param _cspin The pin attached to Chip Select
   * @param spispeed For RPi, the SPI speed in MHZ ie:
BCM2835_SPI_SPEED_8MHZ
   */

  RF24(uint8_t _cepin, uint8_t _cspin, uint32_t spispeed );
  //#endif

  #if defined (RF24_LINUX)
  virtual ~RF24() {};
  #endif

  /**
   * Begin operation of the chip
   *
   * Call this in setup(), before calling any other methods.
   * @code radio.begin() @endcode
   */
  bool begin(void);

  /**
```

```
 * Start listening on the pipes opened for reading.
 *
 * 1. Be sure to call openReadingPipe() first.
 * 2. Do not call write() while in this mode, without first calling stopListening().
 * 3. Call available() to check for incoming traffic, and read() to get it.
 *
 * @code
 * Open reading pipe 1 using address CCCECCCECC
 *
 * byte address[] = { 0xCC,0xCE,0xCC,0xCE,0xCC };
 * radio.openReadingPipe(1,address);
 * radio.startListening();
 * @endcode
 */
void startListening(void);

/**
 * Stop listening for incoming messages, and switch to transmit mode.
 *
 * Do this before calling write().
 * @code
 * radio.stopListening();
 * radio.write(&data,sizeof(data));
 * @endcode
 */
void stopListening(void);

/**
 * Check whether there are bytes available to be read
 * @code
 * if(radio.available()){
 *   radio.read(&data,sizeof(data));
 * }
 * @endcode
 * @return True if there is a payload available, false if none is
 */
bool available(void);

/**
 * Read the available payload
 *
 * The size of data read is the fixed payload size, see getPayloadSize()
 *
 * @note I specifically chose 'void*' as a data type to make it easier
 * for beginners to use.  No casting needed.
 *
 * @note No longer boolean. Use available to determine if packets are
 * available. Interrupt flags are now cleared during reads instead of
 * when calling available().
 *
```

```
   * @param buf Pointer to a buffer where the data should be written
   * @param len Maximum number of bytes to read into the buffer
   *
   * @code
   * if(radio.available()){
   *   radio.read(&data,sizeof(data));
   * }
   * @endcode
   * @return No return value. Use available().
   */
  void read( void* buf, uint8_t len );

  /**
   * Be sure to call openWritingPipe() first to set the destination
   * of where to write to.
   *
   * This blocks until the message is successfully acknowledged by
   * the receiver or the timeout/retransmit maxima are reached.  In
   * the current configuration, the max delay here is 60-70ms.
   *
   * The maximum size of data written is the fixed payload size, see
   * getPayloadSize().  However, you can write less, and the remainder
   * will just be filled with zeroes.
   *
   * TX/RX/RT interrupt flags will be cleared every time write is called
   *
   * @param buf Pointer to the data to be sent
   * @param len Number of bytes to be sent
   *
   * @code
   * radio.stopListening();
   * radio.write(&data,sizeof(data));
   * @endcode
   * @return True if the payload was delivered successfully false if not
   */
  bool write( const void* buf, uint8_t len );

  /**
   * New: Open a pipe for writing via byte array. Old addressing format retained
   * for compatibility.
   *
   * Only one writing pipe can be open at once, but you can change the address
   * you'll write to. Call stopListening() first.
   *
   * Addresses are assigned via a byte array, default is 5 byte address length
s   *
   * @code
   *   uint8_t addresses[][6] = {"1Node","2Node"};
   *   radio.openWritingPipe(addresses[0]);
   * @endcode
```

```
 * @code
 *  uint8_t address[] = { 0xCC,0xCE,0xCC,0xCE,0xCC };
 *  radio.openWritingPipe(address);
 *  address[0] = 0x33;
 *  radio.openReadingPipe(1,address);
 * @endcode
 * @see setAddressWidth
 *
 * @param address The address of the pipe to open. Coordinate these pipe
 * addresses amongst nodes on the network.
 */

void openWritingPipe(const uint8_t *address);

/**
 * Open a pipe for reading
 *
 * Up to 6 pipes can be open for reading at once.  Open all the required
 * reading pipes, and then call startListening().
 *
 * @see openWritingPipe
 * @see setAddressWidth
 *
 * @note Pipes 0 and 1 will store a full 5-byte address. Pipes 2-5 will technically
 * only store a single byte, borrowing up to 4 additional bytes from pipe #1 per the
 * assigned address width.
 * @warning Pipes 1-5 should share the same address, except the first byte.
 * Only the first byte in the array should be unique, e.g.
 * @code
 *   uint8_t addresses[][6] = {"1Node","2Node"};
 *   openReadingPipe(1,addresses[0]);
 *   openReadingPipe(2,addresses[1]);
 * @endcode
 *
 * @warning Pipe 0 is also used by the writing pipe.  So if you open
 * pipe 0 for reading, and then startListening(), it will overwrite the
 * writing pipe.  Ergo, do an openWritingPipe() again before write().
 *
 * @param number Which pipe# to open, 0-5.
 * @param address The 24, 32 or 40 bit address of the pipe to open.
 */

void openReadingPipe(uint8_t number, const uint8_t *address);

 /**@}*/
/**
 * @name Advanced Operation
 *
 *  Methods you can use to drive the chip in more advanced ways
 */
```

```
/**@{*/

/**
 * Print a giant block of debugging information to stdout
 *
 * @warning Does nothing if stdout is not defined.  See fdevopen in stdio.h
 * The printf.h file is included with the library for Arduino.
 * @code
 * #include <printf.h>
 * setup(){
 *  Serial.begin(115200);
 *  printf_begin();
 *  ...
 * }
 * @endcode
 */
void printDetails(void);

/**
 * Test whether there are bytes available to be read in the
 * FIFO buffers.
 *
 * @param[out] pipe_num Which pipe has the payload available
 *
 * @code
 * uint8_t pipeNum;
 * if(radio.available(&pipeNum)){
 *   radio.read(&data,sizeof(data));
 *   Serial.print("Got data on pipe");
 *   Serial.println(pipeNum);
 * }
 * @endcode
 * @return True if there is a payload available, false if none is
 */
bool available(uint8_t* pipe_num);

/**
 * Check if the radio needs to be read. Can be used to prevent data loss
 * @return True if all three 32-byte radio buffers are full
 */
bool rxFifoFull();

/**
 * Enter low-power mode
 *
 * To return to normal power mode, call powerUp().
 *
 * @note After calling startListening(), a basic radio will consume about 13.5mA
 * at max PA level.
 * During active transmission, the radio will consume about 11.5mA, but this will
```

```
   * be reduced to 26uA (.026mA) between sending.
   * In full powerDown mode, the radio will consume approximately 900nA (.0009mA)
   *
   * @code
   * radio.powerDown();
   * avr_enter_sleep_mode(); // Custom function to sleep the device
   * radio.powerUp();
   * @endcode
   */
  void powerDown(void);

  /**
   * Leave low-power mode - required for normal radio operation after calling powerDown()
   *
   * To return to low power mode, call powerDown().
   * @note This will take up to 5ms for maximum compatibility
   */
  void powerUp(void) ;

  /**
  * Write for single NOACK writes. Optionally disables acknowledgements/autoretries for a single write.
  *
  * @note enableDynamicAck() must be called to enable this feature
  *
  * Can be used with enableAckPayload() to request a response
  * @see enableDynamicAck()
  * @see setAutoAck()
  * @see write()
  *
  * @param buf Pointer to the data to be sent
  * @param len Number of bytes to be sent
  * @param multicast Request ACK (0), NOACK (1)
  */
  bool write( const void* buf, uint8_t len, const bool multicast );

  /**
   * This will not block until the 3 FIFO buffers are filled with data.
   * Once the FIFOs are full, writeFast will simply wait for success or
   * timeout, and return 1 or 0 respectively. From a user perspective, just
   * keep trying to send the same data. The library will keep auto retrying
   * the current payload using the built in functionality.
   * @warning It is important to never keep the nRF24L01 in TX mode and FIFO full for more than 4ms at a time. If the auto
   * retransmit is enabled, the nRF24L01 is never in TX mode long enough to disobey this rule. Allow the FIFO
   * to clear by issuing txStandBy() or ensure appropriate time between transmissions.
   *
   * @code
```

```
   * Example (Partial blocking):
   *
   *                 radio.writeFast(&buf,32);  // Writes 1 payload to the buffers
   *                 txStandBy();               // Returns 0 if failed. 1 if success.
Blocks only until MAX_RT timeout or success. Data flushed on fail.
   *
   *                 radio.writeFast(&buf,32);  // Writes 1 payload to the buffers
   *                 txStandBy(1000);           // Using extended timeouts,
returns 1 if success. Retries failed payloads for 1 seconds before returning 0.
   * @endcode
   *
   * @see txStandBy()
   * @see write()
   * @see writeBlocking()
   *
   * @param buf Pointer to the data to be sent
   * @param len Number of bytes to be sent
   * @return True if the payload was delivered successfully false if not
   */
  bool writeFast( const void* buf, uint8_t len );

  /**
   * WriteFast for single NOACK writes. Disables acknowledgements/autoretries for a
single write.
   *
   * @note enableDynamicAck() must be called to enable this feature
   * @see enableDynamicAck()
   * @see setAutoAck()
   *
   * @param buf Pointer to the data to be sent
   * @param len Number of bytes to be sent
   * @param multicast Request ACK (0) or NOACK (1)
   */
  bool writeFast( const void* buf, uint8_t len, const bool multicast );

  /**
    * This function extends the auto-retry mechanism to any specified duration.
    * It will not block until the 3 FIFO buffers are filled with data.
    * If so the library will auto retry until a new payload is written
    * or the user specified timeout period is reached.
    * @warning It is important to never keep the nRF24L01 in TX mode and FIFO full
for more than 4ms at a time. If the auto
    * retransmit is enabled, the nRF24L01 is never in TX mode long enough to disobey
this rule. Allow the FIFO
    * to clear by issuing txStandBy() or ensure appropriate time between transmissions.
    *
    * @code
    * Example (Full blocking):
    *
```

```
  *                     radio.writeBlocking(&buf,32,1000); //Wait up to 1 second to
write 1 payload to the buffers
  *                     txStandBy(1000);                   //Wait up to 1 second for
the payload to send. Return 1 if ok, 0 if failed.
  *
//Blocks only until user timeout or success. Data flushed on fail.
  * @endcode
  * @note If used from within an interrupt, the interrupt should be disabled until
completion, and sei(); called to enable millis().
  * @see txStandBy()
  * @see write()
  * @see writeFast()
  *
  * @param buf Pointer to the data to be sent
  * @param len Number of bytes to be sent
  * @param timeout User defined timeout in milliseconds.
  * @return True if the payload was loaded into the buffer successfully false if not
  */
 bool writeBlocking( const void* buf, uint8_t len, uint32_t timeout );

 /**
  * This function should be called as soon as transmission is finished to
  * drop the radio back to STANDBY-I mode. If not issued, the radio will
  * remain in STANDBY-II mode which, per the data sheet, is not a recommended
  * operating mode.
  *
  * @note When transmitting data in rapid succession, it is still recommended by
  * the manufacturer to drop the radio out of TX or STANDBY-II mode if there is
  * time enough between sends for the FIFOs to empty. This is not required if auto-ack
  * is enabled.
  *
  * Relies on built-in auto retry functionality.
  *
  * @code
  * Example (Partial blocking):
  *
  *                     radio.writeFast(&buf,32);
  *                     radio.writeFast(&buf,32);
  *                     radio.writeFast(&buf,32);  //Fills the FIFO buffers up
  *                     bool ok = txStandBy();     //Returns 0 if failed. 1 if success.
  *                                                //Blocks only until
MAX_RT timeout or success. Data flushed on fail.
  * @endcode
  * @see txStandBy(unsigned long timeout)
  * @return True if transmission is successful
  *
  */
 bool txStandBy();

 /**
```

```
  * This function allows extended blocking and auto-retries per a user defined timeout
  * @code
  *      Fully Blocking Example:
  *
  *                       radio.writeFast(&buf,32);
  *                       radio.writeFast(&buf,32);
  *                       radio.writeFast(&buf,32);   //Fills the FIFO buffers up
  *                       bool ok = txStandBy(1000);  //Returns 0 if failed after 1 second of
retries. 1 if success.
  *                                                                  //Blocks only until
user defined timeout or success. Data flushed on fail.
  * @endcode
  * @note If used from within an interrupt, the interrupt should be disabled until
completion, and sei(); called to enable millis().
  * @param timeout Number of milliseconds to retry failed payloads
  * @return True if transmission is successful
  *
  */
  bool txStandBy(uint32_t timeout, bool startTx = 0);

 /**
  * Write an ack payload for the specified pipe
  *
  * The next time a message is received on @p pipe, the data in @p buf will
  * be sent back in the acknowledgement.
  * @see enableAckPayload()
  * @see enableDynamicPayloads()
  * @warning Only three of these can be pending at any time as there are only 3 FIFO
buffers.<br> Dynamic payloads must be enabled.
  * @note Ack payloads are handled automatically by the radio chip when a payload is
received. Users should generally
  * write an ack payload as soon as startListening() is called, so one is available when a
regular payload is received.
  * @note Ack payloads are dynamic payloads. This only works on pipes 0&1 by
default. Call
  * enableDynamicPayloads() to enable on all pipes.
  *
  * @param pipe Which pipe# (typically 1-5) will get this response.
  * @param buf Pointer to data that is sent
  * @param len Length of the data to send, up to 32 bytes max.  Not affected
  * by the static payload set by setPayloadSize().
  */
 void writeAckPayload(uint8_t pipe, const void* buf, uint8_t len);

 /**
  * Determine if an ack payload was received in the most recent call to
  * write(). The regular available() can also be used.
  *
  * Call read() to retrieve the ack payload.
  *
```

```
   * @return True if an ack payload is available.
   */
  bool isAckPayloadAvailable(void);

  /**
   * Call this when you get an interrupt to find out why
   *
   * Tells you what caused the interrupt, and clears the state of
   * interrupts.
   *
   * @param[out] tx_ok The send was successful (TX_DS)
   * @param[out] tx_fail The send failed, too many retries (MAX_RT)
   * @param[out] rx_ready There is a message waiting to be read (RX_DS)
   */
  void whatHappened(bool& tx_ok,bool& tx_fail,bool& rx_ready);

  /**
   * Non-blocking write to the open writing pipe used for buffered writes
   *
   * @note Optimization: This function now leaves the CE pin high, so the radio
   * will remain in TX or STANDBY-II Mode until a txStandBy() command is issued. Can be used as an alternative to startWrite()
   * if writing multiple payloads at once.
   * @warning It is important to never keep the nRF24L01 in TX mode with FIFO full for more than 4ms at a time. If the auto
   * retransmit/autoAck is enabled, the nRF24L01 is never in TX mode long enough to disobey this rule. Allow the FIFO
   * to clear by issuing txStandBy() or ensure appropriate time between transmissions.
   *
   * @see write()
   * @see writeFast()
   * @see startWrite()
   * @see writeBlocking()
   *
   * For single noAck writes see:
   * @see enableDynamicAck()
   * @see setAutoAck()
   *
   * @param buf Pointer to the data to be sent
   * @param len Number of bytes to be sent
   * @param multicast Request ACK (0) or NOACK (1)
   * @return True if the payload was delivered successfully false if not
   */
  void startFastWrite( const void* buf, uint8_t len, const bool multicast, bool startTx = 1 );

  /**
   * Non-blocking write to the open writing pipe
   *
   * Just like write(), but it returns immediately. To find out what happened
```

```
   * to the send, catch the IRQ and then call whatHappened().
   *
   * @see write()
   * @see writeFast()
   * @see startFastWrite()
   * @see whatHappened()
   *
   * For single noAck writes see:
   * @see enableDynamicAck()
   * @see setAutoAck()
   *
   * @param buf Pointer to the data to be sent
   * @param len Number of bytes to be sent
   * @param multicast Request ACK (0) or NOACK (1)
   *
   */
  void startWrite( const void* buf, uint8_t len, const bool multicast );

  /**
   * This function is mainly used internally to take advantage of the auto payload
   * re-use functionality of the chip, but can be beneficial to users as well.
   *
   * The function will instruct the radio to re-use the data in the FIFO buffers,
   * and instructs the radio to re-send once the timeout limit has been reached.
   * Used by writeFast and writeBlocking to initiate retries when a TX failure
   * occurs. Retries are automatically initiated except with the standard write().
   * This way, data is not flushed from the buffer until switching between modes.
   *
   * @note This is to be used AFTER auto-retry fails if wanting to resend
   * using the built-in payload reuse features.
   * After issuing reUseTX(), it will keep reending the same payload forever or until
   * a payload is written to the FIFO, or a flush_tx command is given.
   */
   void reUseTX();

  /**
    * Empty the transmit buffer. This is generally not required in standard operation.
    * May be required in specific cases after stopListening() , if operating at 250KBPS
data rate.
    *
    * @return Current value of status register
    */
  uint8_t flush_tx(void);

  /**
    * Test whether there was a carrier on the line for the
    * previous listening period.
    *
    * Useful to check for interference on the current channel.
    *
```

```
  * @return true if was carrier, false if not
  */
 bool testCarrier(void);

 /**
  * Test whether a signal (carrier or otherwise) greater than
  * or equal to -64dBm is present on the channel. Valid only
  * on nRF24L01P (+) hardware. On nRF24L01, use testCarrier().
  *
  * Useful to check for interference on the current channel and
  * channel hopping strategies.
  *
  * @code
  * bool goodSignal = radio.testRPD();
  * if(radio.available()){
  *    Serial.println(goodSignal ? "Strong signal > 64dBm" : "Weak signal < 64dBm" );
  *    radio.read(0,0);
  * }
  * @endcode
  * @return true if signal => -64dBm, false if not
  */
 bool testRPD(void) ;

 /**
  * Test whether this is a real radio, or a mock shim for
  * debugging.  Setting either pin to 0xff is the way to
  * indicate that this is not a real radio.
  *
  * @return true if this is a legitimate radio
  */
 bool isValid() { return ce_pin != 0xff && csn_pin != 0xff; }

  /**
  * Close a pipe after it has been previously opened.
  * Can be safely called without having previously opened a pipe.
  * @param pipe Which pipe # to close, 0-5.
  */
 void closeReadingPipe( uint8_t pipe ) ;

  /**
   * Enable error detection by un-commenting #define FAILURE_HANDLING in
RF24_config.h
   * If a failure has been detected, it usually indicates a hardware issue. By default the
library
   * will cease operation when a failure is detected.
   * This should allow advanced users to detect and resolve intermittent hardware issues.
   *
   * In most cases, the radio must be re-enabled via radio.begin(); and the appropriate
settings
   * applied after a failure occurs, if wanting to re-enable the device immediately.
```

```
 *
 * Usage: (Failure handling must be enabled per above)
 *  @code
 *  if(radio.failureDetected){
 *    radio.begin();                       // Attempt to re-configure the radio with defaults
 *    radio.failureDetected = 0;           // Reset the detection value
 *      radio.openWritingPipe(addresses[1]); // Re-configure pipe addresses
 *    radio.openReadingPipe(1,addresses[0]);
 *    report_failure();                    // Blink leds, send a message, etc. to indicate failure
 *  }
 * @endcode
*/
//#if defined (FAILURE_HANDLING)
  bool failureDetected;
//#endif

/**@}*/

/**@}*/
/**
 * @name Optional Configurators
 *
 *  Methods you can use to get or set the configuration of the chip.
 *  None are required.  Calling begin() sets up a reasonable set of
 *  defaults.
 */
/**@{*/

/**
* Set the address width from 3 to 5 bytes (24, 32 or 40 bit)
*
* @param a_width The address width to use: 3,4 or 5
*/

void setAddressWidth(uint8_t a_width);

/**
 * Set the number and delay of retries upon failed submit
 *
 * @param delay How long to wait between each retry, in multiples of 250us,
 * max is 15.  0 means 250us, 15 means 4000us.
 * @param count How many retries before giving up, max 15
 */
void setRetries(uint8_t delay, uint8_t count);

/**
 * Set RF communication channel
 *
 * @param channel Which RF channel to communicate on, 0-127
 */
```

```
void setChannel(uint8_t channel);

  /**
 * Get RF communication channel
 *
 * @return The currently configured RF Channel
 */
uint8_t getChannel(void);

/**
 * Set Static Payload Size
 *
 * This implementation uses a pre-stablished fixed payload size for all
 * transmissions.  If this method is never called, the driver will always
 * transmit the maximum payload size (32 bytes), no matter how much
 * was sent to write().
 *
 * @todo Implement variable-sized payloads feature
 *
 * @param size The number of bytes in the payload
 */
void setPayloadSize(uint8_t size);

/**
 * Get Static Payload Size
 *
 * @see setPayloadSize()
 *
 * @return The number of bytes in the payload
 */
uint8_t getPayloadSize(void);

/**
 * Get Dynamic Payload Size
 *
 * For dynamic payloads, this pulls the size of the payload off
 * the chip
 *
 * @note Corrupt packets are now detected and flushed per the
 * manufacturer.
 * @code
 * if(radio.available()){
 *   if(radio.getDynamicPayloadSize() < 1){
 *     // Corrupt payload has been flushed
 *     return;
 *   }
 *   radio.read(&data,sizeof(data));
 * }
 * @endcode
 *
```

```
  * @return Payload length of last-received dynamic payload
  */
 uint8_t getDynamicPayloadSize(void);

 /**
  * Enable custom payloads on the acknowledge packets
  *
  * Ack payloads are a handy way to return data back to senders without
  * manually changing the radio modes on both units.
  *
  * @note Ack payloads are dynamic payloads. This only works on pipes 0&1 by
default. Call
  * enableDynamicPayloads() to enable on all pipes.
  */
 void enableAckPayload(void);

 /**
  * Enable dynamically-sized payloads
  *
  * This way you don't always have to send large packets just to send them
  * once in a while.  This enables dynamic payloads on ALL pipes.
  *
  */
 void enableDynamicPayloads(void);

 /**
  * Enable dynamic ACKs (single write multicast or unicast) for chosen messages
  *
  * @note To enable full multicast or per-pipe multicast, use setAutoAck()
  *
  * @warning This MUST be called prior to attempting single write NOACK calls
  * @code
  * radio.enableDynamicAck();
  * radio.write(&data,32,1);  // Sends a payload with no acknowledgement requested
  * radio.write(&data,32,0);  // Sends a payload using auto-retry/autoACK
  * @endcode
  */
 void enableDynamicAck();

 /**
  * Determine whether the hardware is an nRF24L01+ or not.
  *
  * @return true if the hardware is nRF24L01+ (or compatible) and false
  * if its not.
  */
 bool isPVariant(void) ;

 /**
  * Enable or disable auto-acknowlede packets
  *
```

```
  * This is enabled by default, so it's only needed if you want to turn
  * it off for some reason.
  *
  * @param enable Whether to enable (true) or disable (false) auto-acks
  */
 void setAutoAck(bool enable);

 /**
  * Enable or disable auto-acknowlede packets on a per pipeline basis.
  *
  * AA is enabled by default, so it's only needed if you want to turn
  * it off/on for some reason on a per pipeline basis.
  *
  * @param pipe Which pipeline to modify
  * @param enable Whether to enable (true) or disable (false) auto-acks
  */
 void setAutoAck( uint8_t pipe, bool enable ) ;

 /**
  * Set Power Amplifier (PA) level to one of four levels:
  * RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX
  *
  * The power levels correspond to the following output levels respectively:
  * NRF24L01: -18dBm, -12dBm,-6dBM, and 0dBm
  *
  * SI24R1: -6dBm, 0dBm, 3dBM, and 7dBm.
  *
  * @param level Desired PA level.
  */
 void setPALevel ( uint8_t level );

 /**
  * Fetches the current PA level.
  *
  * NRF24L01: -18dBm, -12dBm, -6dBm and 0dBm
  * SI24R1:   -6dBm, 0dBm, 3dBm, 7dBm
  *
  * @return Returns values 0 to 3 representing the PA Level.
  */
  uint8_t getPALevel( void );

 /**
  * Set the transmission data rate
  *
  * @warning setting RF24_250KBPS will fail for non-plus units
  *
  * @param speed RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or
RF24_2MBPS for 2Mbps
  * @return true if the change was successful
  */
```

```
  bool setDataRate(rf24_datarate_e speed);

  /**
   * Fetches the transmission data rate
   *
   * @return Returns the hardware's currently configured datarate. The value
   * is one of 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS, as defined in the
* rf24_datarate_e enum.
   */
  rf24_datarate_e getDataRate( void ) ;

/**
   * Set the CRC length
   * <br>CRC checking cannot be disabled if auto-ack is enabled
   * @param length RF24_CRC_8 for 8-bit or RF24_CRC_16 for 16-bit
   */
  void setCRCLength(rf24_crclength_e length);

  /**
   * Get the CRC length
   * <br>CRC checking cannot be disabled if auto-ack is enabled
   * @return RF24_DISABLED if disabled or RF24_CRC_8 for 8-bit or RF24_CRC_16
for 16-bit
   */
  rf24_crclength_e getCRCLength(void);

  /**
   * Disable CRC validation
   *
   * @warning CRC cannot be disabled if auto-ack/ESB is enabled.
   */
  void disableCRC( void ) ;

  /**
  * The radio will generate interrupt signals when a transmission is complete,
  * a transmission fails, or a payload is received. This allows users to mask
  * those interrupts to prevent them from generating a signal on the interrupt
  * pin. Interrupts are enabled on the radio chip by default.
  *
  * @code
  *       Mask all interrupts except the receive interrupt:
  *
  *                radio.maskIRQ(1,1,0);
  * @endcode
  *
  * @param tx_ok  Mask transmission complete interrupts
  * @param tx_fail  Mask transmit failure interrupts
  * @param rx_ready Mask payload received interrupts
  */
  void maskIRQ(bool tx_ok,bool tx_fail,bool rx_ready);
```

```
 /**@}*/
 /**
  * @name Deprecated
  *
  *  Methods provided for backwards compabibility.
  */
 /**@{*/


 /**
  * Open a pipe for reading
  * @note For compatibility with old code only, see new function
  *
  * @warning Pipes 1-5 should share the first 32 bits.
  * Only the least significant byte should be unique, e.g.
  * @code
  *   openReadingPipe(1,0xF0F0F0F0AA);
  *   openReadingPipe(2,0xF0F0F0F066);
  * @endcode
  *
  * @warning Pipe 0 is also used by the writing pipe.  So if you open
  * pipe 0 for reading, and then startListening(), it will overwrite the
  * writing pipe.  Ergo, do an openWritingPipe() again before write().
  *
  * @param number Which pipe# to open, 0-5.
  * @param address The 40-bit address of the pipe to open.
  */
 void openReadingPipe(uint8_t number, uint64_t address);

 /**
  * Open a pipe for writing
  * @note For compatibility with old code only, see new function
  *
  * Addresses are 40-bit hex values, e.g.:
  *
  * @code
  *   openWritingPipe(0xF0F0F0F0F0);
  * @endcode
  *
  * @param address The 40-bit address of the pipe to open.
  */
 void openWritingPipe(uint64_t address);

private:

 /**
  * @name Low-level internal interface.
  *
  *  Protected methods that address the chip directly.  Regular users cannot
```

```
   *  ever call these.  They are documented for completeness and for developers who
   *  may want to extend this class.
   */
  /**@{*/

  /**
   * Set chip select pin
   *
   * Running SPI bus at PI_CLOCK_DIV2 so we don't waste time transferring data
   * and best of all, we make use of the radio's FIFO buffers. A lower speed
   * means we're less likely to effectively leverage our FIFOs and pay a higher
   * AVR runtime cost as toll.
   *
   * @param mode HIGH to take this unit off the SPI bus, LOW to put it on
   */
  void csn(bool mode);

  /**
   * Set chip enable
   *
   * @param level HIGH to actively begin transmission or LOW to put in standby.
Please see data sheet
   * for a much more detailed description of this pin.
   */
  void ce(bool level);

  /**
   * Read a chunk of data in from a register
   *
   * @param reg Which register. Use constants from nRF24L01.h
   * @param buf Where to put the data
   * @param len How many bytes of data to transfer
   * @return Current value of status register
   */
  uint8_t read_register(uint8_t reg, uint8_t* buf, uint8_t len);

  /**
   * Read single byte from a register
   *
   * @param reg Which register. Use constants from nRF24L01.h
   * @return Current value of register @p reg
   */
  uint8_t read_register(uint8_t reg);

  /**
   * Write a chunk of data to a register
   *
   * @param reg Which register. Use constants from nRF24L01.h
   * @param buf Where to get the data
   * @param len How many bytes of data to transfer
```

```
 * @return Current value of status register
 */
uint8_t write_register(uint8_t reg, const uint8_t* buf, uint8_t len);

/**
 * Write a single byte to a register
 *
 * @param reg Which register. Use constants from nRF24L01.h
 * @param value The new value to write
 * @return Current value of status register
 */
uint8_t write_register(uint8_t reg, uint8_t value);

/**
 * Write the transmit payload
 *
 * The size of data written is the fixed payload size, see getPayloadSize()
 *
 * @param buf Where to get the data
 * @param len Number of bytes to be sent
 * @return Current value of status register
 */
uint8_t write_payload(const void* buf, uint8_t len, const uint8_t writeType);

/**
 * Read the receive payload
 *
 * The size of data read is the fixed payload size, see getPayloadSize()
 *
 * @param buf Where to put the data
 * @param len Maximum number of bytes to read
 * @return Current value of status register
 */
uint8_t read_payload(void* buf, uint8_t len);

/**
 * Empty the receive buffer
 *
 * @return Current value of status register
 */
uint8_t flush_rx(void);

/**
 * Retrieve the current status of the chip
 *
 * @return Current value of status register
 */
uint8_t get_status(void);

#if !defined (MINIMAL)
```

```
  /**
   * Decode and print the given status to stdout
   *
   * @param status Status value to print
   *
   * @warning Does nothing if stdout is not defined.  See fdevopen in stdio.h
   */
  void print_status(uint8_t status);

  /**
   * Decode and print the given 'observe_tx' value to stdout
   *
   * @param value The observe_tx value to print
   *
   * @warning Does nothing if stdout is not defined.  See fdevopen in stdio.h
   */
  void print_observe_tx(uint8_t value);

  /**
   * Print the name and value of an 8-bit register to stdout
   *
   * Optionally it can print some quantity of successive
   * registers on the same line.  This is useful for printing a group
   * of related registers on one line.
   *
   * @param name Name of the register
   * @param reg Which register. Use constants from nRF24L01.h
   * @param qty How many successive registers to print
   */
  void print_byte_register(const char* name, uint8_t reg, uint8_t qty = 1);

  /**
   * Print the name and value of a 40-bit address register to stdout
   *
   * Optionally it can print some quantity of successive
   * registers on the same line.  This is useful for printing a group
   * of related registers on one line.
   *
   * @param name Name of the register
   * @param reg Which register. Use constants from nRF24L01.h
   * @param qty How many successive registers to print
   */
  void print_address_register(const char* name, uint8_t reg, uint8_t qty = 1);
#endif
  /**
   * Turn on or off the special features of the chip
   *
   * The chip has certain 'features' which are only available when the 'features'
   * are enabled.  See the datasheet for details.
   */
```

```
  void toggle_features(void);

  /**
   * Built in spi transfer function to simplify repeating code repeating code
   */

  uint8_t spiTrans(uint8_t cmd);

  #if defined (FAILURE_HANDLING) || defined (RF24_LINUX)
          void errNotify(void);
  #endif

  /**@}*/

};


/**
 * @example GettingStarted.ino
 * <b>For Arduino</b><br>
 * <b>Updated: TMRh20 2014 </b><br>
 *
 * This is an example of how to use the RF24 class to communicate on a basic level.
Configure and write this sketch to two
 * different nodes. Put one of the nodes into 'transmit' mode by connecting with the
serial monitor and <br>
 * sending a 'T'. The ping node sends the current time to the pong node, which responds
by sending the value
 * back. The ping node can then see how long the whole cycle took. <br>
 * @note For a more efficient call-response scenario see the
GettingStarted_CallResponse.ino example.
 * @note When switching between sketches, the radio may need to be powered down to
clear settings that are not "un-set" otherwise
 */

 /**
 * @example GettingStarted.cpp
 * <b>For Raspberry Pi</b><br>
 * <b>Updated: TMRh20 2014 </b><br>
 *
 * This is an example of how to use the RF24 class to communicate on a basic level.
Configure and write this sketch to two
 * different nodes. Put one of the nodes into 'transmit' mode by connecting with the
serial monitor and <br>
 * sending a 'T'. The ping node sends the current time to the pong node, which responds
by sending the value
 * back. The ping node can then see how long the whole cycle took. <br>
 * @note For a more efficient call-response scenario see the
GettingStarted_CallResponse.ino example.
 */
```

```
/**
 * @example GettingStarted_CallResponse.ino
 * <b>For Arduino</b><br>
 * <b>New: TMRh20 2014</b><br>
 *
 * This example continues to make use of all the normal functionality of the radios
including
 * the auto-ack and auto-retry features, but allows ack-payloads to be written optionlly
as well. <br>
 * This allows very fast call-response communication, with the responding radio never
having to
 * switch out of Primary Receiver mode to send back a payload, but having the option to
switch to <br>
 * primary transmitter if wanting to initiate communication instead of respond to a
commmunication.
 */

 /**
 * @example GettingStarted_Call_Response.cpp
 * <b>For Raspberry Pi</b><br>
 * <b>New: TMRh20 2014</b><br>
 *
 * This example continues to make use of all the normal functionality of the radios
including
 * the auto-ack and auto-retry features, but allows ack-payloads to be written optionlly
as well. <br>
 * This allows very fast call-response communication, with the responding radio never
having to
 * switch out of Primary Receiver mode to send back a payload, but having the option to
switch to <br>
 * primary transmitter if wanting to initiate communication instead of respond to a
commmunication.
 */

 /**
 * @example GettingStarted_HandlingData.ino
 * <b>Dec 2014 - TMRh20</b><br>
 *
 * This example demonstrates how to send multiple variables in a single payload and
work with data. As usual, it is
 * generally important to include an incrementing value like millis() in the payloads to
prevent errors.
 */

/**
 * @example Transfer.ino
 * <b>For Arduino</b><br>
 * This example demonstrates half-rate transfer using the FIFO buffers<br>
 *
```

```
 * It is an example of how to use the RF24 class.  Write this sketch to two
 * different nodes.  Put one of the nodes into 'transmit' mode by connecting <br>
 * with the serial monitor and sending a 'T'.  The data transfer will begin,
 * with the receiver displaying the payload count. (32Byte Payloads) <br>
 */

 /**
 * @example Transfer.cpp
 * <b>For Raspberry Pi</b><br>
 * This example demonstrates half-rate transfer using the FIFO buffers<br>
 *
 * It is an example of how to use the RF24 class.  Write this sketch to two
 * different nodes.  Put one of the nodes into 'transmit' mode by connecting <br>
 * with the serial monitor and sending a 'T'.  The data transfer will begin,
 * with the receiver displaying the payload count. (32Byte Payloads) <br>
 */

/**
 * @example TransferTimeouts.ino
 * <b>New: TMRh20 </b><br>
 * This example demonstrates the use of and extended timeout period and
 * auto-retries/auto-reUse to increase reliability in noisy or low signal scenarios. <br>
 *
 * Write this sketch to two different nodes.  Put one of the nodes into 'transmit'
 * mode by connecting with the serial monitor and sending a 'T'.  The data <br>
 * transfer will begin, with the receiver displaying the payload count and the
 * data transfer rate.
 */

/**
 * @example starping.pde
 *
 * This sketch is a more complex example of using the RF24 library for Arduino.
 * Deploy this on up to six nodes.  Set one as the 'pong receiver' by tying the
 * role_pin low, and the others will be 'ping transmit' units.  The ping units
 * unit will send out the value of millis() once a second.  The pong unit will
 * respond back with a copy of the value.  Each ping unit can get that response
 * back, and determine how long the whole cycle took.
 *
 * This example requires a bit more complexity to determine which unit is which.
 * The pong receiver is identified by having its role_pin tied to ground.
 * The ping senders are further differentiated by a byte in eeprom.
 */

/**
 * @example pingpair_ack.ino
 * <b>Update: TMRh20</b><br>
 * This example continues to make use of all the normal functionality of the radios
including
```

```
 * the auto-ack and auto-retry features, but allows ack-payloads to be written optionlly as well.<br>
 * This allows very fast call-response communication, with the responding radio never having to
 * switch out of Primary Receiver mode to send back a payload, but having the option to if wanting<br>
 * to initiate communication instead of respond to a commmunication.
 */

/**
 * @example pingpair_irq.ino
 * <b>Update: TMRh20</b><br>
 * This is an example of how to user interrupts to interact with the radio, and a demonstration
 * of how to use them to sleep when receiving, and not miss any payloads.<br>
 * The pingpair_sleepy example expands on sleep functionality with a timed sleep option for the transmitter.
 * Sleep functionality is built directly into my fork of the RF24Network library<br>
 */

 /**
 * @example pingpair_irq_simple.ino
 * <b>Dec 2014 - TMRh20</b><br>
 * This is an example of how to user interrupts to interact with the radio, with bidirectional communication.
 */

/**
 * @example pingpair_sleepy.ino
 * <b>Update: TMRh20</b><br>
 * This is an example of how to use the RF24 class to create a battery-
 * efficient system.  It is just like the GettingStarted_CallResponse example, but the<br>
 * ping node powers down the radio and sleeps the MCU after every
 * ping/pong cycle, and the receiver sleeps between payloads. <br>
 */

 /**
 * @example rf24ping85.ino
 * <b>New: Contributed by https://github.com/tong67</b><br>
 * This is an example of how to use the RF24 class to communicate with ATtiny85 and other node. <br>
 */

 /**
 * @example timingSearch3pin.ino
 * <b>New: Contributed by https://github.com/tong67</b><br>
 * This is an example of how to determine the correct timing for ATtiny when using only 3-pins
 */
```

```
/**
 * @example pingpair_dyn.ino
 *
 * This is an example of how to use payloads of a varying (dynamic) size on Arduino.
 */

 /**
 * @example pingpair_dyn.cpp
 *
 * This is an example of how to use payloads of a varying (dynamic) size on Raspberry
Pi.
 */

/**
 * @example pingpair_dyn.py
 *
 * This is a python example for RPi of how to use payloads of a varying (dynamic) size.
 */

/**
 * @example pingpair_dyn.ino
 *
 * This is an example of how to use payloads of a varying (dynamic) size.
 */

 /**
 * @example pingpair_dyn.ino
 *
 * This is an example of how to use payloads of a varying (dynamic) size.
 */

/**
 * @example scanner.ino
 *
 * Example to detect interference on the various channels available.
 * This is a good diagnostic tool to check whether you're picking a
 * good channel for your application.
 *
 * Inspired by cpixip.
 * See http://arduino.cc/forum/index.php/topic,54795.0.html
 */

/**
 * @mainpage Optimized High Speed Driver for nRF24L01(+) 2.4GHz Wireless
Transceiver
 *
 * @section Goals Design Goals
 *
 * This library fork is designed to be...
```

```
 * @li More compliant with the manufacturer specified operation of the chip, while allowing advanced users
 * to work outside the recommended operation.
 * @li Utilize the capabilities of the radio to their full potential via Arduino
 * @li More reliable, responsive, bug-free and feature rich
 * @li Easy for beginners to use, with well documented examples and features
 * @li Consumed with a public interface that's similar to other Arduino standard libraries
 *
 * @section News News
 *
 * **Dec 2015**<br>
 * - ESP8266 support via Arduino IDE
 * - <a href="https://github.com/stewarthou/Particle-RF24">Particle Photon/Core</a> fork available
 * - ATTiny2313/4313 support added
 * - Python 3 support added
 * - RF24 added to Arduino library manager
 * - RF24 added to PlatformIO library manager
 *
 * **March 2015**<br>
 * - Uses SPI transactions on Arduino
 * - New layout for <a href="Portability.html">easier portability:</a> Break out defines & includes for individual platforms to RF24/utility
 * - <a href="MRAA.html">MRAA</a> support added ( Galileo, Edison, etc)
 * - <a href="BBB.html">BBB/Generic Linux </a> support via spidev & MRAA
 * - Support for RPi 2 added
 * - Major Documentation cleanup & update (Move all docs to github.io)
 *
 *
 * If issues are discovered with the documentation, please report them <a href="https://github.com/TMRh20/tmrh20.github.io/issues"> here</a>
 *
 * <br>
 * @section Useful Useful References
 *
 *
 * @li <a href="http://tmrh20.github.io/RF24/classRF24.html"><b>RF24</b> Class Documentation</a>
 * @li <a href="https://github.com/TMRh20/RF24/archive/master.zip"><b>Download</b></a>
 * @li <a href="https://github.com/tmrh20/RF24/"><b>Source Code</b></a>
 * @li <a href="http://tmrh20.blogspot.com/2014/03/high-speed-data-transfers-and-wireless.html"><b>My Blog:</b> RF24 Optimization Overview</a>
 * @li <a href="http://www.nordicsemi.com/files/Product/data_sheet/nRF24L01_Product_Specification_v2_0.pdf">Chip Datasheet</a>
 *
 * **Additional Information and Add-ons**
 *
```

```
 * @li <a href="http://tmrh20.github.io/RF24Network"><b>RF24Network:</b> OSI Network Layer for multi-device communication. Create a home sensor network.</a>
 * @li <a href="http://tmrh20.github.io/RF24Mesh"><b>RF24Mesh:</b> Dynamic Mesh Layer for RF24Network</a>
 * @li <a href="http://tmrh20.github.io/RF24Ethernet"><b>RF24Ethernet:</b> TCP/IP Radio Mesh Networking (shares Arduino Ethernet API)</a>
 * @li <a href="http://tmrh20.github.io/RF24Audio"><b>RF24Audio:</b> Realtime Wireless Audio streaming</a>
 * @li <a href="http://tmrh20.github.io/">All TMRh20 Documentation Main Page</a>
 *
 * **More Information and RF24 Based Projects**
 *
 * @li <a href="http://TMRh20.blogspot.com"> Project Blog: TMRh20.blogspot.com </a>
 * @li <a href="http://maniacalbits.blogspot.ca/"> Maniacal Bits Blog</a>
 * @li <a href="http://www.mysensors.org/">MySensors.org (User friendly sensor networks/IoT)</a>
 * @li <a href="https://github.com/mannkind/RF24Node_MsgProto"> RF24Node_MsgProto (MQTT)</a>
 * @li <a href="https://bitbucket.org/pjhardy/rf24sensornet/"> RF24SensorNet </a>
 * @li <a href="http://www.homeautomationforgeeks.com/rf24software.shtml">Home Automation for Geeks</a>
 * @li <a href="https://maniacbug.wordpress.com/2012/03/30/rf24network/"> Original Maniacbug RF24Network Blog Post</a>
 * @li <a href="https://github.com/maniacbug/RF24"> ManiacBug on GitHub (Original Library Author)</a>
 *
 *
 * <br>
 *
 * @section Platform_Support Platform Support Pages
*
 * @li <a href="Arduino.html"><b>Arduino</b></a> (Uno, Nano, Mega, Due, Galileo, etc)
* @li <a href="ATTiny.html"><b>ATTiny</b></a>
 * @li Linux ( <a href="RPi.html"><b>RPi</b></a> , <a href="BBB.html"><b>BBB</b></a>, <a href="MRAA.html"><b>MRAA</b></a> supported boards ( Galileo, Edison, etc))
 * @li <a href="Python.html"><b>Python</b></a> wrapper available for RPi
 *
 * <br>
 * **General µC Pin layout** (See the individual board support pages for more info)
 *
 * The table below shows how to connect the the pins of the NRF24L01(+) to different boards.
 * CE and CSN are configurable.
 *
 * | PIN | NRF24L01 | Arduino UNO | ATtiny25/45/85 [0] | ATtiny44/84 [1] | LittleWire [2]          |    RPI     | RPi -P1 Connector |
```

```
 * |-----|----------|-------------|--------------------|-----------------|------------------------|-----------
---|-------------------|
 * | 1 |  GND   |  GND      |    pin 4        |    pin 14      | GND                  | rpi-gnd
|    (25)        |
 * | 2 |  VCC   |  3.3V     |    pin 8        |    pin  1      | regulator 3.3V required | rpi-
3v3   |    (17)        |
 * | 3 |  CE    |  digIO 7  |    pin 2        |    pin 12      | pin to 3.3V            | rpi-gpio22
|    (15)        |
 * | 4 |  CSN   |  digIO 8  |    pin 3        |    pin 11      | RESET                  | rpi-gpio8
|    (24)        |
 * | 5 |  SCK   |  digIO 13 |    pin 7        |    pin  9      | SCK                    | rpi-sckl   |
(23)        |
 * | 6 |  MOSI  |  digIO 11 |    pin 6        |    pin  7      | MOSI                   | rpi-mosi
|    (19)        |
 * | 7 |  MISO  |  digIO 12 |    pin 5        |    pin  8      | MISO                   | rpi-miso
|    (21)        |
 * | 8 |  IRQ   |    -      |      -          |      -         | -                      |  -      |      -         |
 *
 * @li [0] https://learn.sparkfun.com/tutorials/tiny-avr-programmer-hookup-
guide/attiny85-use-hints
 * @li [1] http://highlowtech.org/?p=1695
* @li [2] http://littlewire.cc/
 * <br><br><br>
 *
 *
 *
 *
 * @page Arduino Arduino
*
 * RF24 is fully compatible with Arduino boards <br>
 * See <b> http://www.arduino.cc/en/Reference/Board </b> and <b>
http://arduino.cc/en/Reference/SPI </b> for more information
 *
 * RF24 makes use of the standard hardware SPI pins (MISO,MOSI,SCK) and requires
two additional pins, to control
 * the chip-select and chip-enable functions.<br>
 * These pins must be chosen and designated by the user, in RF24 radio(ce_pin,cs_pin);
and can use any
 * available pins.
 *
 * <br>
 * @section ARD_DUE Arduino Due
 *
 * RF24 makes use of the extended SPI functionality available on the Arduino Due, and
requires one of the
 * defined hardware SS/CS pins to be designated in RF24 radio(ce_pin,cs_pin);<br>
 * See http://arduino.cc/en/Reference/DueExtendedSPI for more information
 *
 * Initial Due support taken from https://github.com/mcrosson/RF24/tree/due
 *
```

```
 * <br>
 * @section Alternate_SPI Alternate SPI Support
 *
 * RF24 supports alternate SPI methods, in case the standard hardware SPI pins are otherwise unavailable.
 *
 * <br>
 * **Software Driven SPI**
 *
 * Software driven SPI is provided by the <a href=https://github.com/greiman/DigitalIO>DigitalIO</a> library
 *
 * Setup:<br>
 * 1. Install the digitalIO library<br>
 * 2. Open RF24_config.h in a text editor. Uncomment the line #define SOFTSPI<br>
 * 3. In your sketch, add #include DigitalIO.h
 *
 * @note Note: Pins are listed as follows and can be modified by editing the RF24_config.h file<br>
 *
 *     const uint8_t SOFT_SPI_MISO_PIN = 16;
 *     const uint8_t SOFT_SPI_MOSI_PIN = 15;
 *     const uint8_t SOFT_SPI_SCK_PIN = 14;
 *
 * <br>
 * **Alternate Hardware (UART) Driven  SPI**
 *
 * The Serial Port (UART) on Arduino can also function in SPI mode, and can double-buffer data, while the
 * default SPI hardware cannot.
 *
 * The SPI_UART library is available at https://github.com/TMRh20/Sketches/tree/master/SPI_UART
 *
 * Enabling:
 * 1. Install the SPI_UART library
 * 2. Edit RF24_config.h and uncomment #define SPI_UART
 * 3. In your sketch, add @code #include <SPI_UART.h> @endcode
 *
 * SPI_UART SPI Pin Connections:
 * | NRF |Arduino Uno Pin|
 * |-----|---------------|
 * | MOSI| TX(0)         |
 * | MISO| RX(1)         |
 * | SCK | XCK(4)        |
 * | CE  | User Specified|
 * | CSN | User Specified|
 *
 *
```

```
 * @note SPI_UART on Mega boards requires soldering to an unused pin on the chip.
<br>See
 * https://github.com/TMRh20/RF24/issues/24 for more information on SPI_UART.
 *
 * @page ATTiny ATTiny
 *
 * ATTiny support is built into the library, so users are not required to include SPI.h in
their sketches<br>
 * See the included rf24ping85 example for pin info and usage
 *
 * Some versions of Arduino IDE may require a patch to allow use of the full program
space on ATTiny<br>
 * See
https://github.com/TCWORLD/ATTinyCore/tree/master/PCREL%20Patch%20for%20
GCC for ATTiny patch
 *
 * ATTiny board support initially added from https://github.com/jscrane/RF24
 *
 * @section Hardware Hardware Configuration
 * By tong67 ( https://github.com/tong67 )
 *
 *    **ATtiny25/45/85 Pin map with CE_PIN 3 and CSN_PIN 4**
 * @code
 *                           +-\/-+
 *                NC      PB5  1|o   |8  Vcc --- nRF24L01  VCC, pin2 --- LED --- 5V
 *    nRF24L01  CE, pin3 --- PB3  2|    |7  PB2 --- nRF24L01  SCK, pin5
 *    nRF24L01 CSN, pin4 --- PB4  3|    |6  PB1 --- nRF24L01 MOSI, pin6
 *    nRF24L01 GND, pin1 --- GND  4|    |5  PB0 --- nRF24L01 MISO, pin7
 *                           +----+
 * @endcode
 *
 * <br>
 *    **ATtiny25/45/85 Pin map with CE_PIN 3 and CSN_PIN 3** => PB3 and PB4 are
free to use for application <br>
 *    Circuit idea from http://nerdralph.blogspot.ca/2014/01/nrf24l01-control-with-3-
attiny85-pins.html <br>
 *   Original RC combination was 1K/100nF. 22K/10nF combination worked better.
<br>
 *        For best settletime delay value in RF24::csn() the timingSearch3pin.ino sketch
can be used.         <br>
 *    This configuration is enabled when CE_PIN and CSN_PIN are equal, e.g. both 3
<br>
 *    Because CE is always high the power consumption is higher than for 5 pins
solution              <br>
 * @code
 *                                                          ^^
 *                           +-\/-+          nRF24L01   CE, pin3 ------|            //
 *                     PB5  1|o   |8  Vcc --- nRF24L01  VCC, pin2 ------x----------x--|<|--
5V
 *                NC      PB3  2|    |7  PB2 --- nRF24L01  SCK, pin5 --|<|---x-[22k]--|  LED
```

```
 *                NC      PB4  3|    |6  PB1 --- nRF24L01 MOSI, pin6  1n4148 |
 *    nRF24L01 GND, pin1 -x- GND  4|    |5  PB0 --- nRF24L01 MISO, pin7         |
 *                     |       +----+                                |
 *                     |-----------------------------------------------||----x-- nRF24L01 CSN, pin4
 *                                                                    10nF
 * @endcode
 *
 * <br>
 *    **ATtiny24/44/84 Pin map with CE_PIN 8 and CSN_PIN 7** <br>
 *        Schematic provided and successfully tested by Carmine Pastore
(https://github.com/Carminepz) <br>
 * @code
 *                           +-\/-+
 *    nRF24L01  VCC, pin2 --- VCC  1|o   |14 GND --- nRF24L01  GND, pin1
 *                         PB0  2|    |13 AREF
 *                         PB1  3|    |12 PA1
 *                         PB3  4|    |11 PA2 --- nRF24L01   CE, pin3
 *                         PB2  5|    |10 PA3 --- nRF24L01  CSN, pin4
 *                         PA7  6|    |9  PA4 --- nRF24L01  SCK, pin5
*    nRF24L01 MISO, pin7 --- PA6  7|    |8  PA5 --- nRF24L01 MOSI, pin6
*                           +----+
 *        @endcode
 *
 * <br>
 *    **ATtiny2313/4313 Pin map with CE_PIN 12 and CSN_PIN 13** <br>
 * @code
 *                           +-\/-+
 *                         PA2  1|o   |20 VCC --- nRF24L01  VCC, pin2
 *                         PD0  2|    |19 PB7 --- nRF24L01  SCK, pin5
 *                         PD1  3|    |18 PB6 --- nRF24L01 MOSI, pin6
 *                         PA1  4|    |17 PB5 --- nRF24L01 MISO, pin7
 *                         PA0  5|    |16 PB4 --- nRF24L01  CSN, pin4
 *                         PD2  6|    |15 PB3 --- nRF24L01   CE, pin3
 *                         PD3  7|    |14 PB2
 *                         PD4  8|    |13 PB1
 *                         PD5  9|    |12 PB0
 *    nRF24L01  GND, pin1 --- GND 10|    |11 PD6
 *                           +----+
 *        @endcode
 *
 * <br><br><br>
 *
 *
 *
 *
 *
 *
 * @page BBB BeagleBone Black
 *
 * BeagleBone Black is supported via MRAA or SPIDEV.
```

```
 *
 *  @note The SPIDEV option should work with most Linux systems supporting
SPIDEV. <br>
 *  Users may need to edit the RF24/utility/BBB/spi.cpp file to configure the spi device.
(Defaults: "/dev/spidev1.0";  or  "/dev/spidev1.1"; )
 *
 * <br>
 * @section AutoInstall Automated Install
 *(**Designed & Tested on RPi** - Defaults to SPIDEV on BBB)
 *
 *
 * 1. Download the install.sh file from
http://tmrh20.github.io/RF24Installer/RPi/install.sh
 * @code wget http://tmrh20.github.io/RF24Installer/RPi/install.sh @endcode
 * 2. Make it executable:
 * @code chmod +x install.sh @endcode
 * 3. Run it and choose your options
 * @code ./install.sh @endcode
 * 4. Run an example from one of the libraries
 * @code
 * cd rf24libs/RF24/examples_RPi
 * @endcode
 * Edit the gettingstarted example, to set your pin configuration
 * @code nano gettingstarted.cpp
 * make
 * sudo ./gettingstarted
 * @endcode
 *
 * <br>
 * @section ManInstall Manual Install
 * 1. Make a directory to contain the RF24 and possibly RF24Network lib and enter it:
 * @code
 *  mkdir ~/rf24libs
 *  cd ~/rf24libs
*  @endcode
 * 2. Clone the RF24 repo:
 *    @code git clone https://github.com/tmrh20/RF24.git RF24 @endcode
 * 3. Change to the new RF24 directory
 *    @code cd RF24 @endcode
 * 4. Build the library, and run an example file:
 * **Note:** See the <a
href="http://iotdk.intel.com/docs/master/mraa/index.html">MRAA </a> documentation
for more info on installing MRAA
 *    @code sudo make install  OR  sudo make install RF24_MRAA=1 @endcode
 * @code
 * cd examples_RPi
 * @endcode
 * Edit the gettingstarted example, to set your pin configuration
 * @code nano gettingstarted.cpp
 * make
```

```
 * sudo ./gettingstarted
 * @endcode
 *
 * <br><br>
 *
 * @page MRAA MRAA
 *
 * MRAA is a Low Level Skeleton Library for Communication on GNU/Linux
platforms <br>
 * See http://iotdk.intel.com/docs/master/mraa/index.html for more information
 *
 * RF24 supports all MRAA supported platforms, but might not be tested on each
individual platform due to the wide range of hardware support:<br>
 * <a href="https://github.com/TMRh20/RF24/issues">Report an RF24 bug or issue
</a>
 *
 * @section Setup Setup
 * 1. Install the MRAA lib
 * 2. As per your device, SPI may need to be enabled
 *
 * @section MRAA_Install Install
 *
 * 1. Make a directory to contain the RF24 and possibly RF24Network lib and enter it:
 * @code
 *  mkdir ~/rf24libs
 *  cd ~/rf24libs
*  @endcode
 * 2. Clone the RF24 repo:
 *    @code git clone https://github.com/tmrh20/RF24.git RF24 @endcode
 * 3. Change to the new RF24 directory
 *    @code cd RF24 @endcode
 * 4. Build the library:
 *    @code sudo make install -B RF24_MRAA=1 @endcode
 * 5. Configure the correct pins in gettingstarted.cpp (See
http://iotdk.intel.com/docs/master/mraa/index.html )
 *    @code
 *    cd examples_RPi
 *    nano gettingstarted.cpp
 *    @endcode
 * 6. Build an example
 *    @code
 *    make
 *    sudo ./gettingstarted
 *    @endcode
 *
 * <br><br><br>
 *
 *
 *
 *
```

```
 * @page RPi Raspberry Pi
 *
 * RF24 supports a variety of Linux based devices via various drivers. Some boards like RPi can utilize multiple methods
 * to drive the GPIO and SPI functionality.
 *
 * <br>
 * @section PreConfig Potential PreConfiguration
 *
 * If SPI is not already enabled, load it on boot:
 * @code sudo raspi-config  @endcode
 * A. Update the tool via the menu as required<br>
 * B. Select **Advanced** and **enable the SPI kernel module** <br>
 * C. Update other software and libraries:
 * @code sudo apt-get update @endcode
 * @code sudo apt-get upgrade @endcode
 * <br>
 * @section AutoInstall Automated Install
 *
 * 1. Download the install.sh file from http://tmrh20.github.io/RF24Installer/RPi/install.sh
 * @code wget http://tmrh20.github.io/RF24Installer/RPi/install.sh @endcode
 * 2. Make it executable:
 * @code chmod +x install.sh @endcode
 * 3. Run it and choose your options
 * @code ./install.sh @endcode
 * 4. Run an example from one of the libraries
 * @code
 * cd rf24libs/RF24/examples_RPi
 * make
 * sudo ./gettingstarted
 * @endcode
 * <br><br>
 * @section ManInstall Manual Install
 * 1. Make a directory to contain the RF24 and possibly RF24Network lib and enter it:
 * @code
 *  mkdir ~/rf24libs
 *  cd ~/rf24libs
*  @endcode
 * 2. Clone the RF24 repo:
 *    @code git clone https://github.com/tmrh20/RF24.git RF24 @endcode
 * 3. Change to the new RF24 directory
 *    @code cd RF24 @endcode
 * 4. Build the library, and run an example file:
 * @code sudo make install
 * cd examples_RPi
 * make
 * sudo ./gettingstarted
 * @endcode
 *
```

```
 * <br><br>
 * @section Build Build Options
 * The default build on Raspberry Pi utilizes the included **BCM2835** driver from
http://www.airspayce.com/mikem/bcm2835
 * 1. @code sudo make install -B @endcode
 *
 * Build using the **MRAA** library from
http://iotdk.intel.com/docs/master/mraa/index.html <br>
 * MRAA is not included. See the <a href="MRAA.html">MRAA</a> platform page
for more information.
 *
 * 1. Install, and build MRAA:
 * @code
 * git clone https://github.com/intel-iot-devkit/mraa.git
 * cd mraa
 * mkdir build
 * cd build
 * cmake .. -DBUILDSWIGNODE=OFF
 * sudo make install
 * @endcode
 *
 * 2. Complete the install <br>
* @code nano /etc/ld.so.conf @endcode
* Add the line @code /usr/local/lib/arm-linux-gnueabihf @endcode
 * Run @code sudo ldconfig @endcode
 *
 * 3. Install RF24, using MRAA
 * @code sudo make install -B RF24_MRAA=1 @endcode
 * See the gettingstarted example for an example of pin configuration
 *
 * Build using **spidev**:
 *
 * 1. Edit the RF24/utility/BBB/spi.cpp file
 * 2. Change the default device definition to @code this->device = "/dev/spidev0.0";;
@endcode
 * 3. Run @code sudo make install -B RF24_SPIDEV=1 @endcode
 * 4. See the gettingstarted example for an example of pin configuration
 *
 * <br>
 * @section Pins Connections and Pin Configuration
 *
 *
 * Using pin 15/GPIO 22 for CE, pin 24/GPIO8 (CE0) for CSN
 *
 * Can use either RPi CE0 or CE1 pins for radio CSN.<br>
 * Choose any RPi output pin for radio CE pin.
 *
 * **BCM2835 Constructor:**
 * @code
```

```
 *  RF24 radio(RPI_V2_GPIO_P1_15,BCM2835_SPI_CS0,
BCM2835_SPI_SPEED_8MHZ);
 *   or
 *  RF24 radio(RPI_V2_GPIO_P1_15,BCM2835_SPI_CS1,
BCM2835_SPI_SPEED_8MHZ);
 *
 *  RPi B+:
 *  RF24 radio(RPI_BPLUS_GPIO_J8_15,RPI_BPLUS_GPIO_J8_24,
BCM2835_SPI_SPEED_8MHZ);
 *  or
 *  RF24 radio(RPI_BPLUS_GPIO_J8_15,RPI_BPLUS_GPIO_J8_26,
BCM2835_SPI_SPEED_8MHZ);
 *
 *  General:
 *  RF24 radio(22,0);
 *  or
 *  RF24 radio(22,1);
 *
 * @endcode
 * See the gettingstarted example for an example of pin configuration
 *
 * See http://www.airspayce.com/mikem/bcm2835/index.html for BCM2835 class
documentation.
 * <br><br>
 * **MRAA Constructor:**
 *
 * @code RF24 radio(15,0); @endcode
 *
 * See http://iotdk.intel.com/docs/master/mraa/rasppi.html
 * <br><br>
 * **SPI_DEV Constructor**
 *
 * @code RF24 radio(22,0); @endcode
 *
 * See http://pi.gadgetoid.com/pinout
 *
 * **Pins:**
 *
 * | PIN | NRF24L01 |    RPI     | RPi -P1 Connector |
 * |-----|----------|------------|-------------------|
 * |  1  |   GND    | rpi-gnd    |     (25)          |
 * |  2  |   VCC    | rpi-3v3    |     (17)          |
 * |  3  |   CE     | rpi-gpio22 |     (15)          |
 * |  4  |   CSN    | rpi-gpio8  |     (24)          |
 * |  5  |   SCK    | rpi-sckl   |     (23)          |
 * |  6  |   MOSI   | rpi-mosi   |     (19)          |
 * |  7  |   MISO   | rpi-miso   |     (21)          |
 * |  8  |   IRQ    |    -       |      -            |
 *
 *
```

```
 *
 *
 * <br><br>
 ****************
 *
 * Based on the arduino lib from J. Coliz <maniacbug@ymail.com><br>
 * the library was berryfied by Purinda Gunasekara <purinda@gmail.com><br>
 * then forked from github stanleyseow/RF24 to https://github.com/jscrane/RF24-rpi
<br>
 * Network lib also based on https://github.com/farconada/RF24Network
 *
 *
 *
 *
 * <br><br><br>
 *
 *
 *
 * @page Python Python Wrapper (by https://github.com/mz-fuzzy)
 *
 * @section Install Installation:
 *
 * Install the boost libraries:  (Note: Only the python libraries should be needed, this is
just for simplicity)
 *
 * @code sudo apt-get install libboost1.50-all @endcode
 *
 * Build the library:
 *
 * @code ./setup.py build   @endcode
 *
 * Install the library
 *
 * @code sudo ./setup.py install  @endcode
 *
 *
 * See the additional <a href="pages.html">Platform Support</a> pages for information
on connecting your hardware  <br>
 * See the included <a href="pingpair_dyn_8py-example.html">example </a> for usage
information.
 *
 * Running the Example:
 *
 * Edit the pingpair_dyn.py example to configure the appropriate pins per the above
documentation:
 *
 * @code nano pingpair_dyn.py   @endcode
 *
 * Configure another device, Arduino or RPi with the <a href="pingpair_dyn_8py-
example.html">pingpair_dyn</a> example
```

```
 *
 * Run the example
 *
 * @code sudo ./pingpair_dyn.py  @endcode
 *
 * <br><br><br>
 *
 *
 * @page Portability RF24 Portability
 *
 * The RF24 radio driver mainly utilizes the <a
href="http://arduino.cc/en/reference/homePage">Arduino API</a> for GPIO, SPI, and
timing functions, which are easily replicated
 * on various platforms. <br>Support files for these platforms are stored under
RF24/utility, and can be modified to provide
 * the required functionality.
 *
 * <br>
 * @section Hardware_Templates Basic Hardware Template
 *
 * **RF24/utility**
 *
 * The RF24 library now includes a basic hardware template to assist in porting to
various platforms. <br> The following files can be included
 * to replicate standard Arduino functions as needed, allowing devices from ATTiny to
Raspberry Pi to utilize the same core RF24 driver.
 *
 * | File              |                  Purpose                                                                    |
 * |-------------------|-----------------------------------------------------------------------------|
 * | RF24_arch_config.h | Basic Arduino/AVR compatibility, includes for remaining
support files, etc   |
 * | includes.h        | Linux only. Defines specific platform, include correct
RF24_arch_config file |
 * | spi.h             | Provides standardized SPI ( transfer() ) methods                        |
 * | gpio.h            | Provides standardized GPIO ( digitalWrite() ) methods
|
 * | compatibility.h    | Provides standardized timing (millis(), delay()) methods
|
 * | your_custom_file.h | Provides access to custom drivers for spi,gpio, etc
|
 *
 * <br>
 * Examples are provided via the included hardware support templates in
**RF24/utility** <br>
 * See the <a href="modules.html">modules</a> page for examples of class
declarations
 *
 *<br>
 * @section Device_Detection Device Detection
 *
```

```
 * 1. The main detection for Linux devices is done in the Makefile, with the includes.h from the proper hardware directory copied to RF24/utility/includes.h <br>
 * 2. Secondary detection is completed in RF24_config.h, causing the include.h file to be included for all supported Linux devices <br>
 * 3. RF24.h contains the declaration for SPI and GPIO objects 'spi' and 'gpio' to be used for porting-in related functions.
 *
 * <br>
 * @section Ported_Code Code
 * To have your ported code included in this library, or for assistance in porting, create a pull request or open an issue at https://github.com/TMRh20/RF24
 *
 *
 *<br><br><br>
 */

#endif // __RF24_H__
```

## 1.10 Anexo IV – Software del sistema de monitorización

### *Código emisor*

```
/* Conexión entre el módulo de radiofrecuencia NRF24L01 y el arduino Leonardo
*      NRF24L01          Arduino
 *        Vcc             3.3 V
 *        GND              GND
 *        CSN             Pin 8
 *        CE              Pin 7
 *        MOSI             Pin 4 ICSP
 *        SCK             Pin 3 ICSP
*        IRQ             no conectado
 *        MISO             Pin 1 ICSP
 */

/*La conexión de los acelerómetros debe hacerse en los siguientes puertos, dependiendo
de la placa de Arduino que estemos utilizando (ver el doc de Wire.h en arduino.cc):
    -Arduino Uno y Ethernet: 4->SDA y 5->SCL (puertos analógicos)
    -Arduino Mega2560: 20->SDA y 21->SCL (puertos digitales)
    -Arduino Leonardo: 2->SDA y 3->SCL (puertos digitales)
    -Arduino Due: 20->SDA y 21->SCL (puertos digitales)
*/

//Librerias para correcto funcionamiento de buses de comunicación
#include <SPI.h>                                    // Comunicación mediante puerto serie
#include <Wire.h>                                    //Transmisión de bus I2C
   // Control librerias internas módulo radiofrecuencia
#include <nRF24L01.h>
#include <RF24.h>

#define ACEL_A 0x1D                                  //dirección del acelerómetro A con
SDO a +3V3
#define ACEL_B 0x53                                 //dirección del acelerómetro B con
SDO a GND
#define NUM_BYTES 6                                  //número de bytes a leer en cada
momento (2 bytes para cada eje)



RF24 radio(7, 8); //Definición puertos de control del módulo radiofrecuencia

float sensor1= A0,sensor2= A1,sensor3= A2,temp=A3;          //Asignación a variables el
puerto de entrada de sensores de presión y temperatura
float cadena[10];                                    //Array para el envio en bloque de los datos
obtenidos

const byte rxAddr[6] = "00010";                          //Dirección a la cual se envia el
paquete, sirve para restringir posibles interferencias
```

```
void setup()
{
  radio.begin();                              //Inicio de la comunicación del módulo
  radio.setRetries(15, 15);                   //Delay y número de reintentos tras cada
envio
  radio.openWritingPipe(rxAddr);              //Abre el puerto oir donde se va a
realizar la comunicación
  radio.setPALevel(RF24_PA_HIGH);             //Ajuste el nivel de amplificador
de potencia ( PA ) a uno de cuatro niveles
  radio.stopListening();                      //Corta la escucha del módulo
  Wire.begin();                               //ponemos en marcha el bus I2C, Arduino es
maestro de este bus
  Serial.begin(9600);                         //ponemos en marcha el puerto serie para
enviar los datos de los acelerómetros
  escribir(ACEL_A,0x2D,24);                   //ponemos en marcha el ADXL345 A
(ver datasheet de ADXL345, se pone Link=1 y Auto_sleep=1)
  escribir(ACEL_B,0x2D,24);                   //ponemos en marcha el ADXL345 B
igual que el anterior
}


void loop()
{
  byte buff[NUM_BYTES];                       //un vector de 6 bytes para guardar
datos leídos del dispositivo
  char cad[512];                              //un vector para transformar los datos leídos
del acelerómetro antes de enviarlos al puerto serie
  int dirReg=0x32;                            //primer registro de datos del eje X del
ADXL345 (los registros Y y Z van a continuación, ver el datasheet)
  int xa=0,ya=0,za=0;                         //variables que guardan los valores de los
ejes del acelerómetro A
  int xb=0,yb=0,zb=0;                         //variables que guardan los valores de los
ejes del acelerómetro B
  double xaD=0, yaD=0, zaD=0, xbD=0, ybD=0, zbD=0;      //declaramos variables tipo
double para poder operar en punto flotante con las variables que anteriormente eran int
double Xag=0, Yag=0, Zag=0, Xbg=0, Ybg=0, Zbg=0;      // """""""
leer(ACEL_A,dirReg,buff);                     //leemos los datos del primer ADXL345
(A, 0x1D). Cada eje lee un valor de 10 bits (2bytes)
                                    //cuidado porque primero van los bits menos
significativos (al revés)
  xa=(((int)buff[1])<<8) | buff[0];           //convertimos los 6 bytes en tres números
enteros, uno para cada eje del acelerómetro
  ya=(((int)buff[3])<<8) | buff[2];           //esto se hace colocando delante los 8 bits
más significativos y detrás los menos significativos
  za=(((int)buff[5])<<8) | buff[4];           //lecturas del acelerómetro A en LSB

  leer(ACEL_B,dirReg,buff);                   //ahora hacemos lo mismo pero con el
segundo ADXL345 (B, 0x53)
  xb=(((int)buff[1])<<8) | buff[0];           //lecturas del acelerómetro B en LSB
yb=(((int)buff[3])<<8) | buff[2];
```

```
  zb=(((int)buff[5])<<8) | buff[4];

xaD = xa;                                   //conversión de las variables int a double del
acelerómetro A para poder operar en punto flotante
  yaD = ya;
  zaD = za;

  xbD = xb;                                  //conversión de las variables int a double del
acelerómetro B para poder operar en punto flotante
  ybD = yb;
  zbD = zb;

  //Creamos la cadena con todos los datos a enviar
  cadena[0]=(5.0*analogRead(sensor1)/1023.0);
  cadena[1]=(5.0*analogRead(sensor2)/1023.0);
  cadena[2]=(5.0*analogRead(sensor3)/1023.0);
  cadena[3]=(5.0*analogRead(temp)/1023.0);
  cadena[4]=xaD;
  cadena[5]=yaD;
  cadena[6]=zaD;
  cadena[7]=xbD;
  cadena[8]=ybD;
  cadena[9]=zbD;

  radio.write(&cadena, sizeof(cadena));             //Función que envia el contenido del
array cadena mediante el módulo de radiofrecuencia
  delay(50);                                 // Espera de 50 ms
}
void escribir(int disposit,byte direcc,byte val) {
   Wire.beginTransmission(disposit);              //iniciamos la comunicación con el
acelerómetro
   Wire.write(direcc);                          //indicamos la dirección donde se va a escribir
   Wire.write(val);                            //indicamos el valor que se va a escribir
   Wire.endTransmission();                        //y finalizamos la transmisión
}

//Función para leer del acelerómetro:
// -disposit: acelerómetro (0x1D ó 0x53)
// -direcc: dirección del registro del acelerómetro a leer
// -buff: vector para guardar los valores leidos
void leer(int disposit,byte direcc,byte buff[]) {
int i=0;
  Wire.beginTransmission(disposit);              //iniciamos la comunicación con el
acelerómetro
  Wire.write(direcc);                          //indicamos la dirección desde donde se va a leer
  Wire.endTransmission();                       //finalizamos la transmisión
  Wire.beginTransmission(disposit);             //iniciamos de nuevo la comunicación con
el acelerómetro
  Wire.requestFrom(disposit,NUM_BYTES);         //solicitamos la lectura de los datos
del acelerómetro
```

```
  while(Wire.available()) {                  //tenemos que leer datos mientras haya
disponibles (el número de bytes que hemos pedido)
    buff[i]=Wire.read();                     //Aquí leemos un byte
    i++;
  }
  Wire.endTransmission();                    //y finalizamos la transmisión
}
```

### *Código receptor*

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Variables módulo radiofrecuencia
RF24 radio(7, 8);
const byte rxAddr[6] = "00010";

//Variables captación y almacenamiento de datos
float s0, s1, s2,valor[10],temp,xaD,yaD,zaD,xbD,ybD,zbD;

//Constantes del sistema
const float I0=494,I1=496,I2=501; //Corriente ofrecida de cada fuente de corriente (FC)
a 25 ºC en µA
const float SensibilidadFC0=2.6416,SensibilidadFC1=3.21,SensibilidadFC2=3.6253;
//Sensibilidad de cada FC en µA/ºC
const float SensibilidadS0=4.7585E-4,SensibilidadS1=5.0393E-
4,SensibilidadS2=3.9649E-4; //Sensibilidad del sensor de presión en mV/kPa*µA
const float GananciaS0=97.303, GananciaS1=97.191, GananciaS2=97.5; //Ganancia de
los amplificadores AD23 en V/V

void setup()
{
  Serial.begin(9600);
  Serial1.begin(9600);
  while (!Serial);
  radio.begin();
  radio.openReadingPipe(0, rxAddr);
  radio.startListening();
  Serial.println(" CEA      CER Móvil      CER Fijo      Temperatura ");
}

void loop()
{
  if (radio.available()>0)
{
    radio.read(&valor, sizeof(valor));
    s0=valor[0];
    s1=valor[1];
```

```
    s2=valor[2];
    temp=valor[3]*10;
    xaD=valor[4];
    yaD=valor[5];
    zaD=valor[6];
    xbD=valor[7];
    ybD=valor[8];
    zbD=valor[9];

Serial.print(SP0(),6);
    Serial.print(" , ");
    Serial.print(SP1(),6);
    Serial.print(" , ");
    Serial.print(SP2(),6);
    Serial.print(" , ");
    Serial.print(temp);
    Serial.print(" , ");
    Serial.print(xaD,6);
    Serial.print(" , ");
    Serial.print(yaD,6);
    Serial.print(" , ");
    Serial.print(zaD,6);
    Serial.print(" , ");
    Serial.print(xbD,6);
    Serial.print(" , ");
    Serial.print(ybD,6);
    Serial.print(" , ");
    Serial.println(zbD,6);
}
}

float SP0 () //Función para el calculo de la presión según temperatura y tensión de salida
{
  float CorrienteReal0=I0+(SensibilidadFC0*(temp-25.0)); //Calculo de la corriente real
en función de la temperatura
  float Vdifs0=(s0/GananciaS0)*1000.0; //Tensión diferencial de la salida del sensor de
presión en mV
  float Sensibilidadfinal0=SensibilidadS0*CorrienteReal0;//Sensibilidad del sensor de
presión en mV/kPa
  float Presion0=Vdifs0/Sensibilidadfinal0;//Conversión a presión de la tensión
diferencial a kPa
return Presion0;
 }

float SP1 ()
{
  float CorrienteReal1=I1+(SensibilidadFC1*(temp-25.0));
float Vdifs1=(s1/GananciaS1)*1000.0;
  float Sensibilidadfinal1=SensibilidadS1*CorrienteReal1;
float Presion1=Vdifs1/Sensibilidadfinal1;
```

```
  return Presion1;
 }

float SP2 ()
{
  float CorrienteReal2=I2+(SensibilidadFC2*(temp-25.0));
float Vdifs2=(s2/GananciaS2)*1000.0;
  float Sensibilidadfinal2=SensibilidadS2*CorrienteReal2;
float Presion2=Vdifs2/Sensibilidadfinal2;
  return Presion2;
}
```

## 1.11 Anexo V – Manuales de usuario

### *Manual de usuario software Arduino 1.6.5*

***Estructura del programa***

La estructura básica del del programa en lenguaje de Arduino se compone de dos partes; la primera se trata del bucle setup() que se encarga de recoger la configuración, la siguiente y última se trata del bucle loop() que recoje el programa a ejecutar, este bucle se repetirá infinitamente. La estructura quedaría de la siguiente forma.

```
void setup()
    {
        configuración;
    }

void loop()
    {
        programa;
    }
```

Cabe mencionar, que esta estructura es necesaria para el funcionamiento del programa.

La función de configuración debe contener la declaración de las variables globales a emplear, se trata de las primerasfunciones que se ejecutaránal inicio del programa, se ejecuta sólo una vez, y se utiliza para configurar oinicializar,entre otras cosas, los puertos de entrada y salida.
La función de programa, contenida en el bucle loo(), contiene el código que se ejecutaráreiteradamente. Es aquí donde se definirán todos los pasos a seguir en el programa, ya sean lecturas de puertos, activación de salidas, cálculos intermedios, etc.

***Definición de variables y funciones***

Una variable es un espacio en la memoria, en el cual el programador asigna un valor determinado por el tipo de dato que el lenguaje de programación va soportar, para declararla es necesario saber que tipo de dato la contiene.

Es representada por un nombre que es asignado por el programador, y se nombra justo después de su tipo de dato, los siguientes son los tipos de datos esenciales:

**int;** es numérico sirve para números enteros sin punto decimal (1,2,3,4.. )

**float;** es numérico y sirve para números con parte fraccionaria, es decir con punto decimal ( 1.2, 1.3, 1.4, ….)

**char;** es de tipo carácter y cadena de caracteres, es decir que puedes asignar desde una letra, símbolo o numero hasta una palabra o serie de caracteres ( a, 2, !, &, hola, etc.)

**bool;** es de tipo verdadero o falso, este tipo de dato mostrara en pantalla un "1" en caso de ser ***true*** o "0" en caso de ser ***false.***

**Array;** es un conjunto de valores a los que se accede con un número índice. Cualquier valor puede ser recogido haciendo uso del nombre de la matriz y el número del índice. El primer valor de la matriz es el que está indicado con el índice 0. También se puede declarar el tamaño de la matriz y posteriormente seleccionar la posición deseada en la misma.

```
>> int entradaVariable=0;
>> float miArray[]={valor0, valor 1, valor2…};
```

Una función esta compuesta por un grupo de sentencias bajo el mismo nombre que realizan una tarea específica. Cuando una función es llamada esta realiza todas las operaciones existentes en ella, una vez ofrecido el valor regresa al mismo punto donde fue llamada.
Las funciones se declaran de forma similiar a las variables, la estructura básica es:

```
>> tipodedato  función(parámetros);
```

El ; es un elemente esencial después de cada sentencia ya que marca el final de la misma, en el caso no usarlo en una frase saltará error en la compilación.
Si la función tiene que devolver un valor se deberá usar la sentencia "return nombredelavariable;", de este modo al llamar a la función en cualquier punto del programa esta devolverá el valor de la variable situada con el comando return.

***Comentarios***

Los bloques de comentarios, o multi-línea de comentarios, son áreas de texto ignoradospor el programa que se utilizan para las descripciones del código o comentarios queayudan a comprender el programa. Comienzan con / * y terminan con * / y puedenabarcar varias líneas.

```
/* esto es un bloque de comentario
 no se debe olvidar cerrar los comentarios
 estos deben estar equilibrados
*/
```

Debido a que los comentarios son ignorados por el programa y no ocupan espacio en lamemoria de Arduino pueden ser utilizados con generosidad y también pueden utilizarsepara "comentar" bloques de código con el propósito de anotar informaciones paradepuración.

Una línea de comentario empieza con / / y terminan con la siguiente línea de código. Aligual que los comentarios de bloque, los de línea son ignoradas por el programa y noocupan espacio en la memoria.

```
// esto es un comentario
```

Una línea de comentario se utiliza a menudo después de una instrucción, para proporcionar más información acerca de lo que hace esta o para recordarla más adelante.

***Aritmética***

Los operadores aritméticos que se incluyen en el entorno de programación son suma,resta, multiplicación y división. Estos devuelven la suma, diferencia, producto, ocociente (respectivamente) de dos operandos.

```
y = y + 3;
x = x - 7;
i = j * 6;
r = r / 5;
```

La operaciones se efectúan teniendo en cuenta el tipo de datos que se han definido para los operandos (int, float, etc..), por lo que, por ejemplo, si definimos 9 y 4 comoenteros “int”, 9 / 4 devuelve de resultado 2 en lugar de 2,25 ya que el 9 y 4 se valoresde tipo entero “int” (enteros) y no se reconocen los decimales con este tipo de datos.

***Constantes***

El lenguaje de programación de Arduino tiene unos valores predeterminados, que sonllamados constantes. Se utilizan para hacer los programas más fáciles de leer. Lasconstantes se clasifican en grupos:

- Constantes booleanas:definen los niveles HIGH (alto) y LOW (bajo) cuando estos se refieren al estado de las salidas digitales. FALSE se asocia con 0 (cero), mientras que TRUE se asocia con 1.

- Input/Output: son utilizadas para definir, al comienzo del programa, el modo de funcionamiento de los pines digitales mediante la instrucción pinMode de tal manera que el pin puede ser una entrada INPUT o una salida OUTPUT.

  ```
  >> pinMode(13, OUTPUT); // designamos que el PIN 13 es una salida
  ```

- Bucles condicionales: se tratan de bucles que impiden el paso a ciertos comandos mientras no se cumpla una condición. El bucle más sencillo es el empleado con el estamento “if”, si no se cumple la condición estipulada en la sentencia salta las sentencias del bucle y continúa con el programa. A este estamento se le puede añadir el comando “else” el cual es ejecutado si la condición anterior no se cumple.

```
>> if (inputPin == HIGH) // si el valor de la entrada inputPin es alto
        {
        instruccionesA; //ejecuta si se cumple la condición
        }
else
        {
        instruccionesB; //ejecuta si no se cumple la condición
        }
```

- Lectura y escritura de puertos digitales: para definir estas acciones se emplean unos comando sencillos. Para la lectura se necesitará definir una variable que almacene el valor y el comando quedaría de la siguiente forma:

  ```
  >> valor = digitalRead(Pin);
  ```

  En el caso de la escritura, como se trata de un puerto digital, solamente se podrán definir el nivel alto o bajo a la salida del puerto el comando sería:

  ```
  >> digitalWrite(nºPin, HIGH/LOW);
  ```

- Lectura y escritura de puertos analógicos: en la lectura de un determinado pin como analógica con una resolución de 10 bits, es decir el rango de valores que se pueden leer oscila entre 0 y 1023 bits, equivalente a 0 y 5 V. Estos pines no necesitan ser declarados como entradas o salidas.

  ```
  >> variable = analogRead(nºPin);
  ```

  La lectura es ofrecida en número de bits, para su conversión a voltios se tendrá que modificar el comando al siguiente:

  ```
  >> variable = (5.0*analogRead(nºPin)/1023.0);
  ```

  En el caso de la escritura, esta instrucción sirve para escribir un pseudo-valor analógico utilizando elprocedimiento de modulación por ancho de pulso (PWM) a uno de los pin´s de Arduinomarcados como “pin PWM”.El valor que se puedeenviar a estos pines de salida analógica puede darse en forma de variable o constante,pero siempre con un margen de 0-255.

  ```
  >> analogWrite(nºPin, valor);
  ```

- Delay: este comando detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción, de tal manera que 1000 equivale a 1seg.

  ```
  >> delay(1000);
  ```

- Millis: Devuelve el número de milisegundos transcurrido desde el inicio del programa enArduino hasta el momento actual. Normalmente será un valor grande (dependiendo deltiempo que este en marcha la aplicación después de cargada o después de la última vez que se pulsó el botón "reset" de la tarjeta)

```
>> variable = millis();
```

- Inicialización de puertos:el comando "Serial.begin(nºfrecuencia)" Abre el puerto serie y fija la velocidad en baudios para la transmisión de datos en serie. El valor típico de velocidad para comunicarse con el ordenador es 9600, aunque otras velocidades pueden ser soportadas.

```
>> void setup()
   {
        Serial.begin(9600);
   }
```

- Visualización de valores: Se trata de la impresión de las variables deseadas a través del puerto serie, su visualización se realiza por el Monitor Serie incluido en el programa. Estos valores se pueden ver seguidos, usando el comando "Serial.print(variable)" , o con un salto de línea, "Serial.println(variable)".
  Al mismo tiempo que se define la función para ver la variable en pantalla se puede determinar el lenguaje deseado, si no se indica nada se muestra en lenguaje ASCII, en el caso de querer modificarlo en comando quedaría "Serial.print(variable, lenguaje)", donde los lenguajes pueden ser decimal (DEC), hexadecimal (HEX), octal (OCT) o binario (BIN).
  Otra opción de visualización en el caso de valores con decimales es el número de los mismos que se desean ver por pantalla. Se define igual que con el lenguaje "Serial.print(variable, nºDecimales)".

***Entorno de desarrollo Arduino***

Para programar en Arduino es necesario la descarga del software de su página web, este es completamente gratuito y presenta varias versiones dependiendo del sistema operativo.
Una vez abierto el programa lo primero antes de comenzar a trabajar con el entorno es realizar la configuración de las comunicaciones entre la placa Arduino y el ordenador.Para ello se abrirá la pestaña "Herramientas", situada en la barra superior, una vez en allí se tendrán que configurar dos opciones, la primera "Placa" donde se tendrá que seleccionar el modelo de Arduino que se va a emplear, y en la segunda "Puerto" se seleccionará el puerto del ordenador en el cual se ha conectado la placa. En el caso de que esta no apareciese se tendrá que instalar el controlador de la misma a través del administrador de dispositivos.
El primer paso para comprobar que todo lo que hemos hecho hasta ahora está bien y familiarizarnos con el interfaz de desarrollo, es abrir uno de los ejemplos. Se recomienda abrir el ejemplo "Blink". Para ello debemos acceder a través del menú Archivo → Ejemplos → Basics → Blink .

Con este ejemplo se conseguirá que parpadee un led de la placa.
Una vez se ha seleccionado la placa y el puerto, y se tiene un programa falta compilarlo y cargarlo. Para ello se seleccionarán el primer lugar la V en el menú de iconos de la parte superior del código y una vez verificado el programa se seleccionará la flecha de su derecha, se puede observar en la Imagen 1.

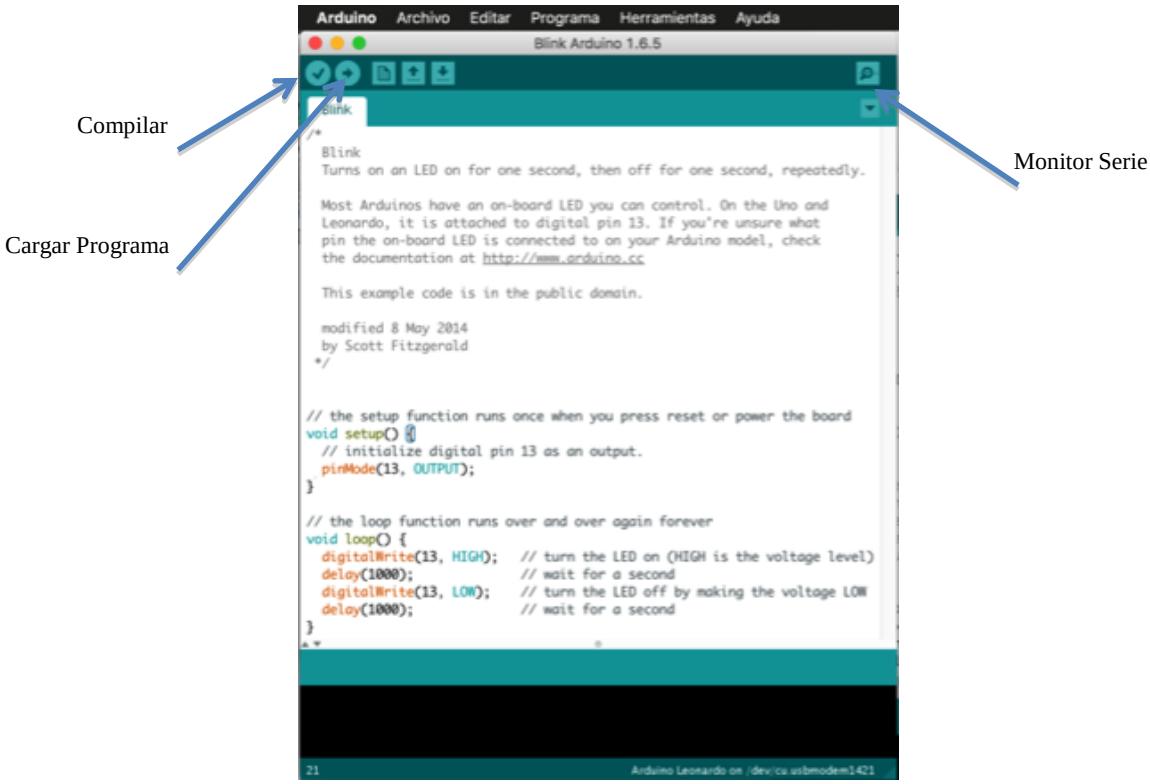



**Imagen 1 - Entorno de desarrollo Arduino**

Durante la carga del programa, en la placa USB, se encenderán los LED que indican que se están enviando y recibiendo información por el puerto serie: TX/RX. Si todo se ha realizado correctamente debe aparecer el mensaje "Subido correctamente". Ahora tan sólo queda esperar unos 8 segundos aproximadamente para comprobar que todo ha salido bien. Si el led colocado en el pin 13 de la placa se enciende y se apaga cada segundo entonces todo ha ido bien. Por fin tenemos todo listo para empezar a trabajar con la placa Arduino.

En el caso de haber programado la visualización por el Monitor Serie de alguna variable solamente se tendrá que clickar el icono de lupa existente en la barra superior a la derecha para que se abra la ventana donde ser irán mostrando los valores seleccionados.

## *Manual de usuario software Autodesk Inventor 2014*

El programa Autodesk Inventor esta enfocado al diseño en 3D de piezas. Posee un entorno de desarrollo muy intuitivo y fácil de manejar.

Tras el inicio del programa se despliega una ventana (Imagen 2) que te permite seleccionar un nuevo tipo de archivo o abrir uno ya existente, para comenzar con un proyecto nuevo se selecciona nuevo.

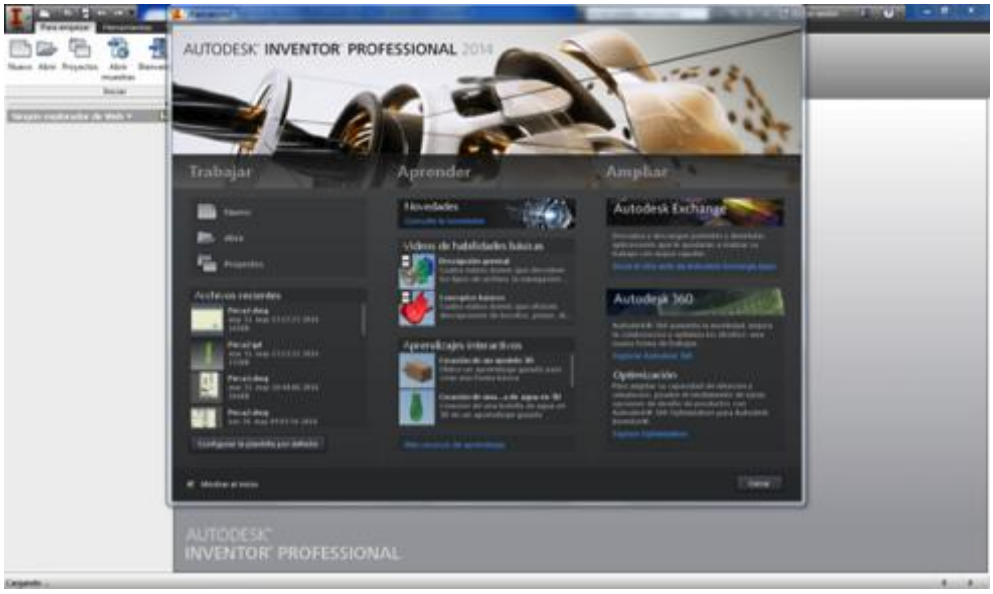



**Imagen 2 - Ventana de inicio Autodesk Inventor 2014**

El siguiente paso es seleccionar en la nueva ventana emergente (Imagen 3) el tipo de proyecto que se desea, situado en la carpeta “Métrica” del desplegable de la izquierda, se selecciona la opción “Normal (mm).ipt” y “Crear”.

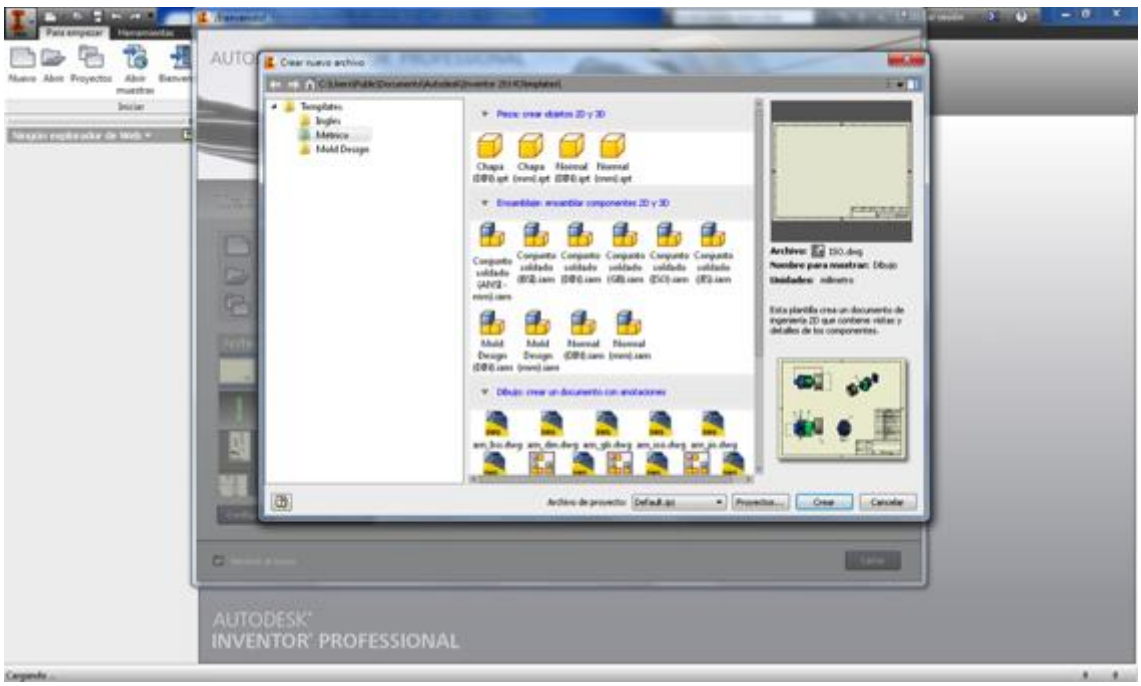



**Imagen 3 - Venta emergente selección tipo de proyecto**

En este momento se abrirá la pantalla inicial que contiene; el área de dibujo, los menús desplegables en la parte superior, la barra estándar de herramientas y las opciones de archivo, en la Imagen 4 se puede observar la situación de todos estos elementos.

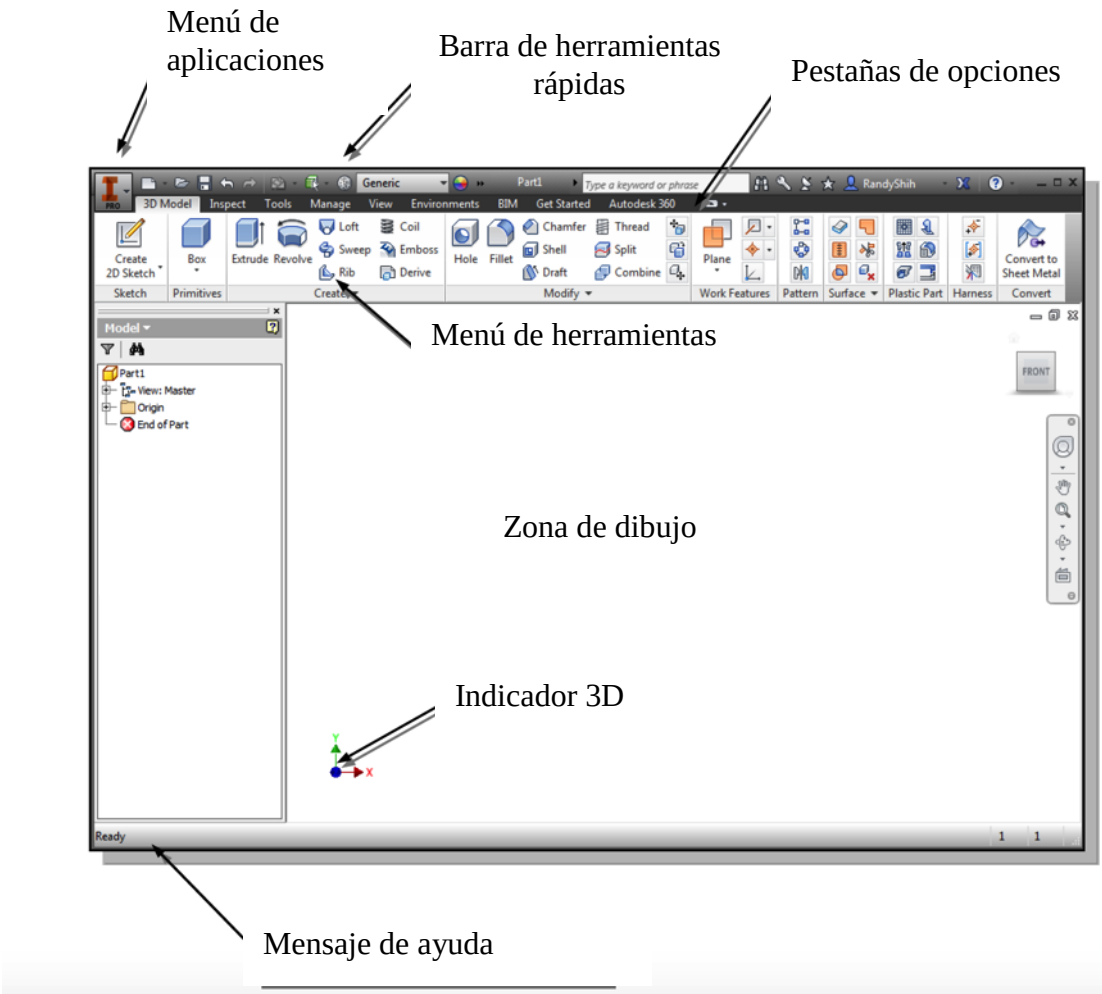



Imagen 4 - Situación elementos pantalla principal

El menú de la aplicación permite abrir, guardar, exportar en diferentes formatos el proyecto. Posee también la opción de deshacer y rehacer, así como un acceso rápido a una nueva hoja.

El primer paso para comenzar con el dibujo es seleccionar la opción “Crear boceto 2D”, una vez seleccionado aparecerá un plano en 3D en el cual se tiene que seleccionar el plano 2D donde se va a realizar el diseño.

Una vez dentro las herramientas del menú rápido se cambian de ser en 3D a 2D, en la Imagen 5 se puede observar todas las herramientas.
Existen todo tipos de utensilios, desde líneas rectas a líneas curvas, círculos, elipses. Con todas estas opciones se dibuja la silueta inicial del boceto en 2D.

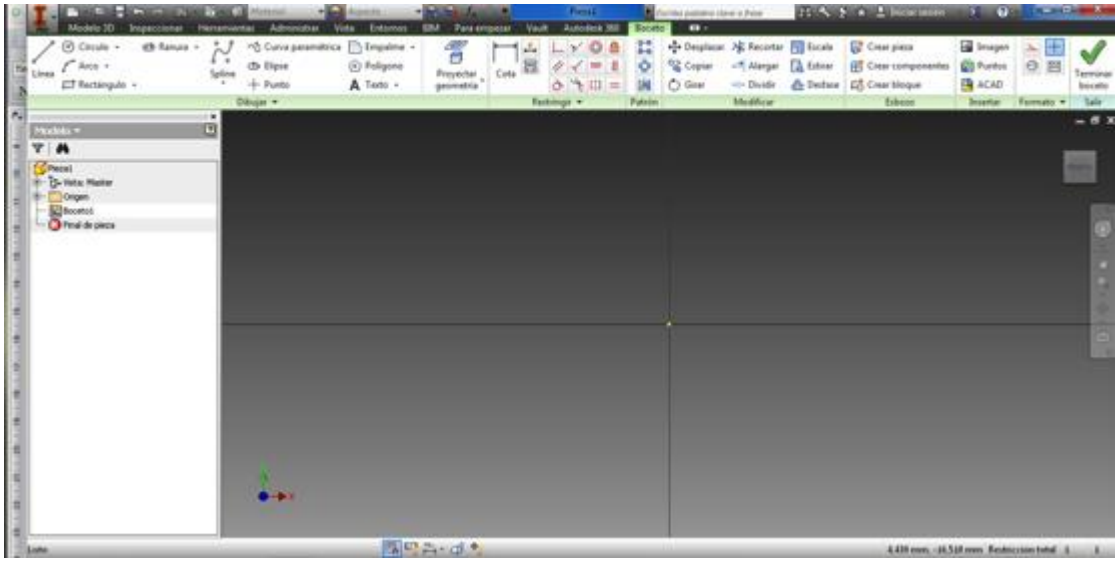


Imagen 5 - Menú boceto 2D

Una vez terminado se selecciona la opción “Terminar Boceta” en la parte superior derecha el cual volverá al menú inicial de diseño en 3D.
En este momento se comienza con el proceso de dar volumen a la pieza, para ello se empleará la herramienta “Extrusión”. Esta opción permite estirar el boceto creado generando una pieza solida, en otras palabras y como ejemplo ilustrativo convierte un circulo en un cilindro. En el menú desplegado al seleccionarlo, se puede observar en la Imagen 6, se puede definir la longitud deseada para la extrusión además de poder seleccionar el plano hacia el cual se desea hacerla. Una vez se tenga introducida la longitud deseada y el plano se selecciona el botón de “Aceptar”.

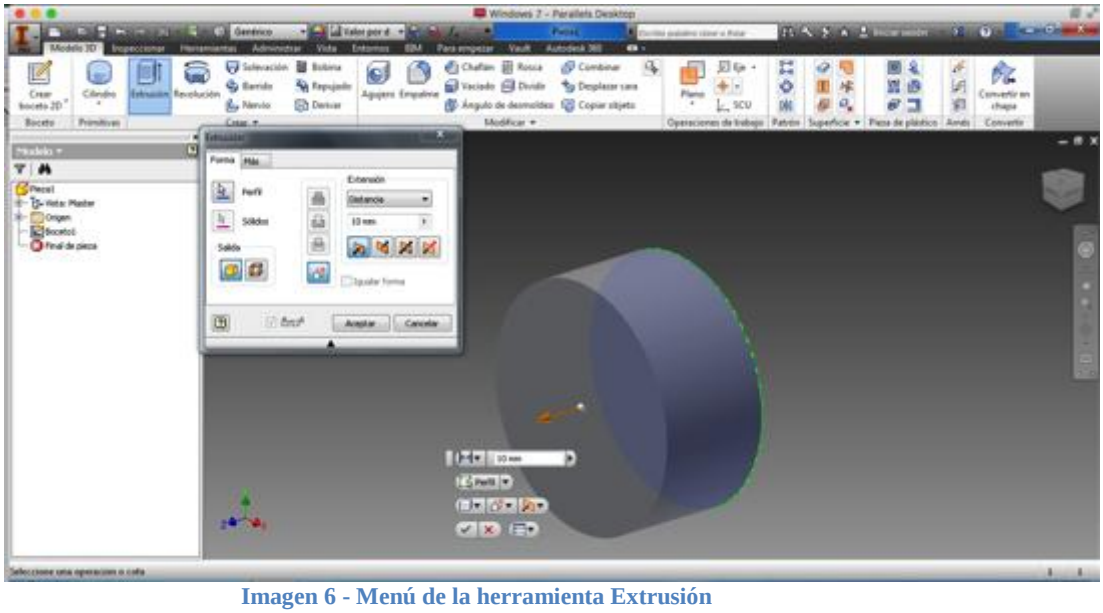


Imagen 6 - Menú de la herramienta Extrusión

Ya se tiene la pieza sólida en 3D, para moverla y poder visualizar todas sus caras se puede hacer mediante el cubo existente en la parte superior derecha, ahí se puede seleccionar cualquier vista o girarlo para ver la pieza en ángulo.

Si se desea se puede escoger el material para la apariencia de la pieza, para ello se selecciona en el menú desplegable de la parte superior izquierda el material y el color deseado.

Una vez se tenga el diseño de la pieza en 3D terminado se continúa con la creación de los planos acotados. Para ello se selecciona la hoja en blanco que se encuentra en la barra de menú a la izquierda. Aparecerá de nuevo la ventana la Imagen 3 donde se seleccionará, en este caso, la opción ISO.dwg. La ventana que aparece se puede observar en la Imagen 7. En este caso se selecciona la opción "Base" donde aparecerá una nueva ventana que elija escoger el plano desde el cual se quiere ver la imagen para crear las vistas necesarias, en la Imagen 8 se puede observar dicho menú.

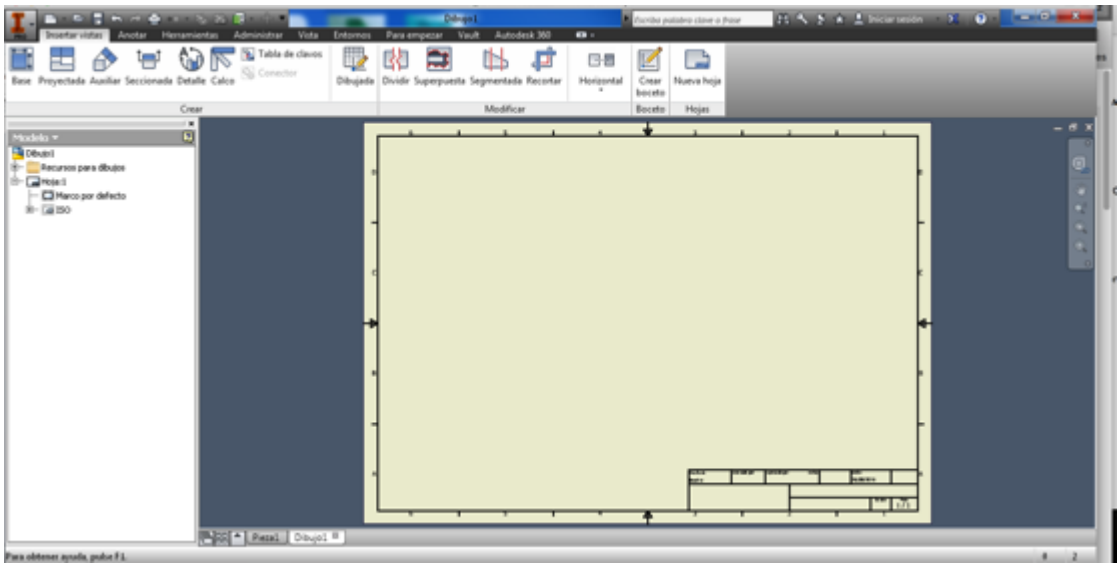


**Imagen 7 - Ventana para el diseño de planos**

En dicho menú se puede escoger la escala necesaria o si se desea realizar las vistas en formato transparente o por el contrario se puede realizar una vista en 3D de la pieza con el formato sólido.

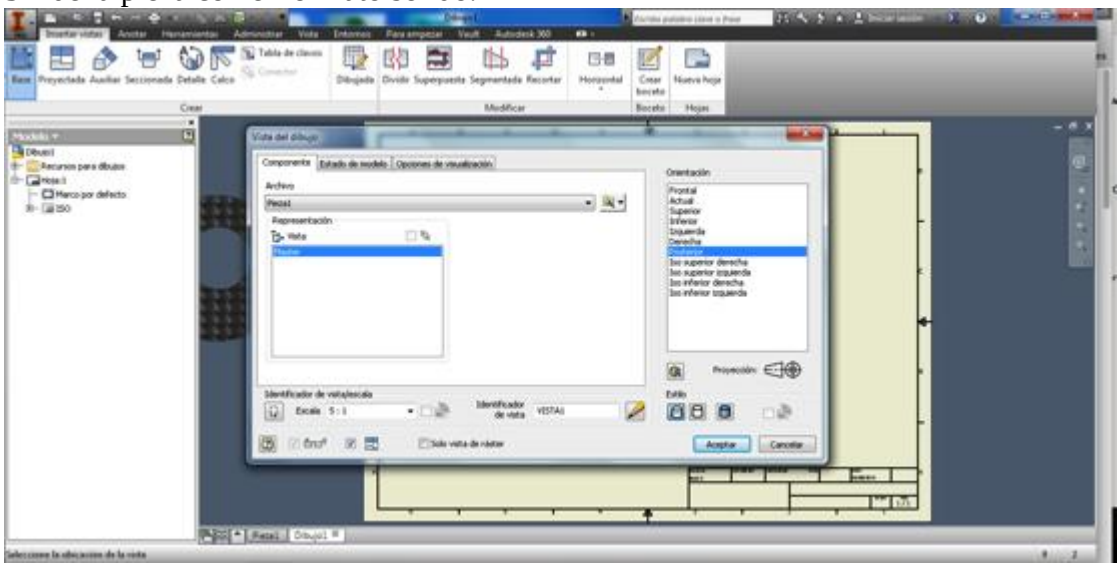
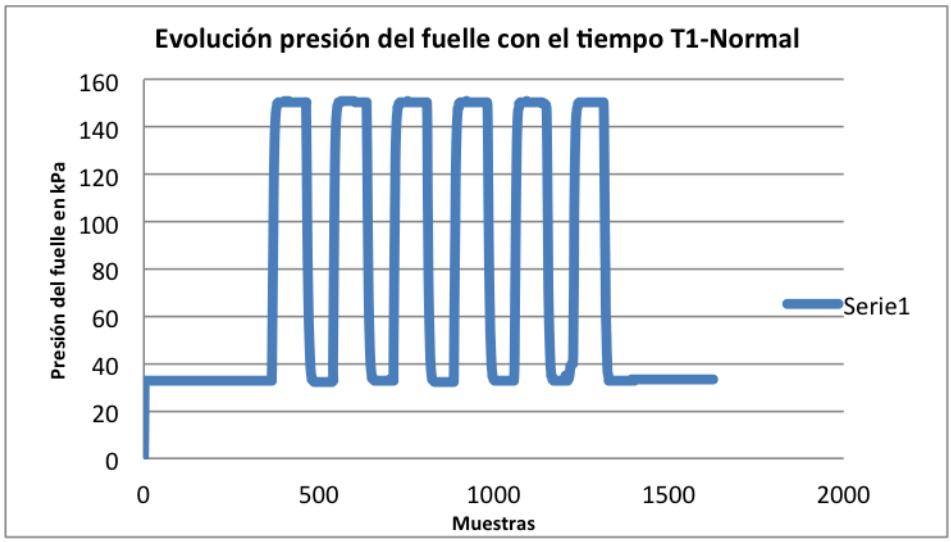


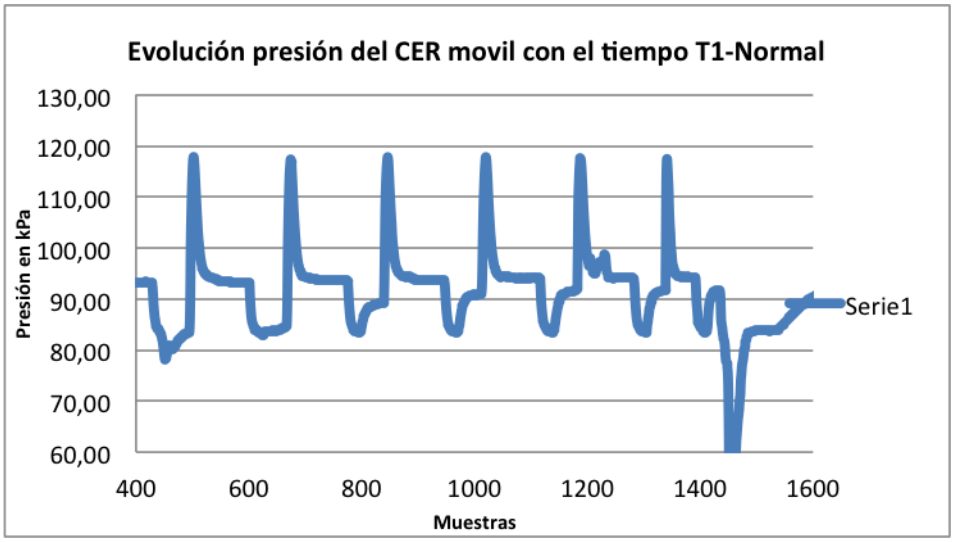
**Imagen 8 - Menú de vistas**

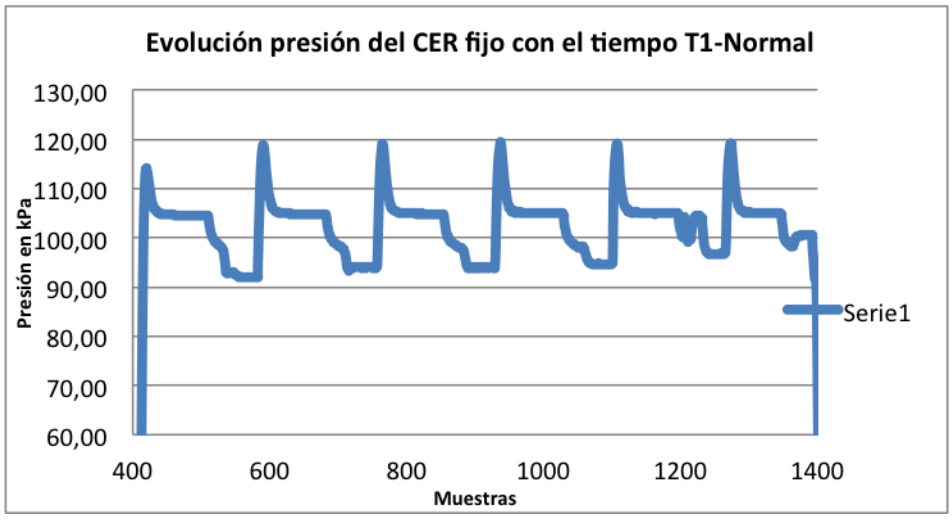
Una vez colocadas las diferentes vistas que definan la pieza se continúa con la acotación de las mismas. Para ello se cambiara de pestaña de menú por "Anotar", aquí se pueden realizar cotas lineales, del radio de una circunferencia o del diámetro.

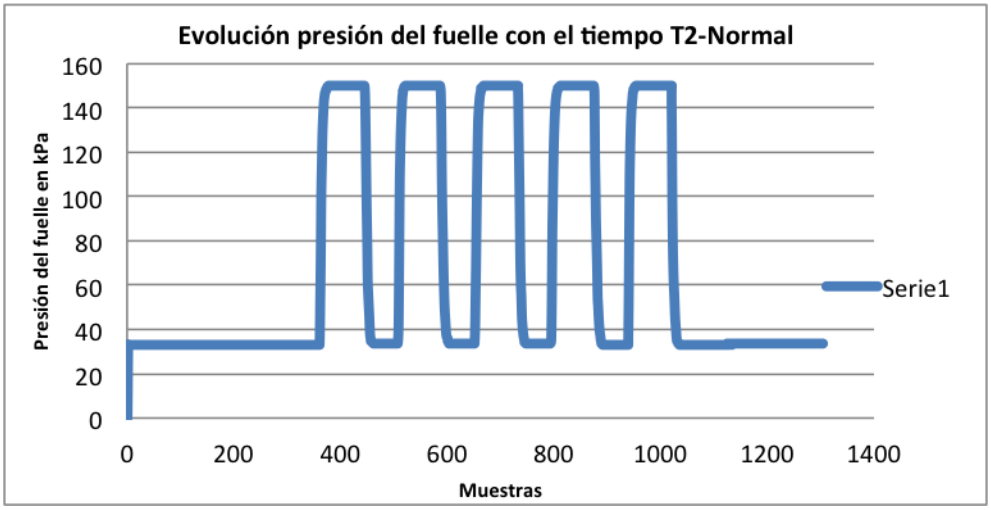
Para finalizar en la opción "I pro" situada en la parte superior izquierda se guardará el proyecto y todos los planos realizados.

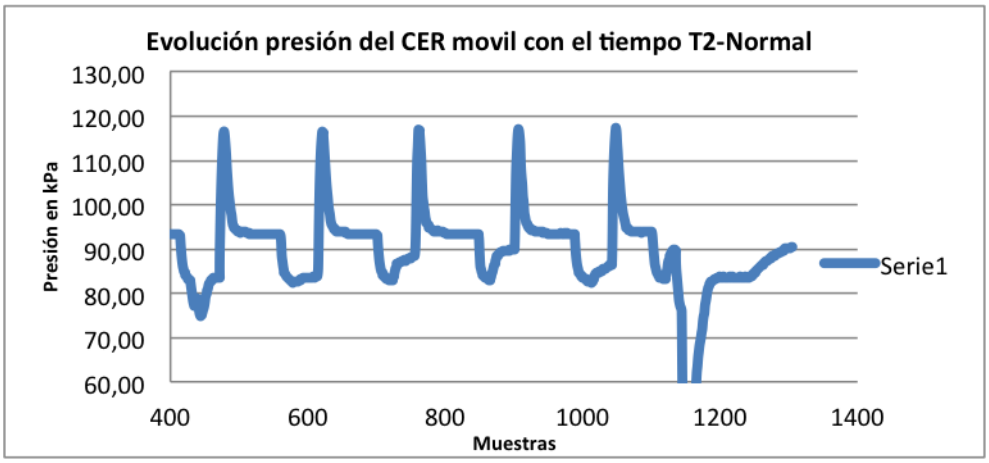
## 1.12 Anexo VI – Gráficas de la toma de datos

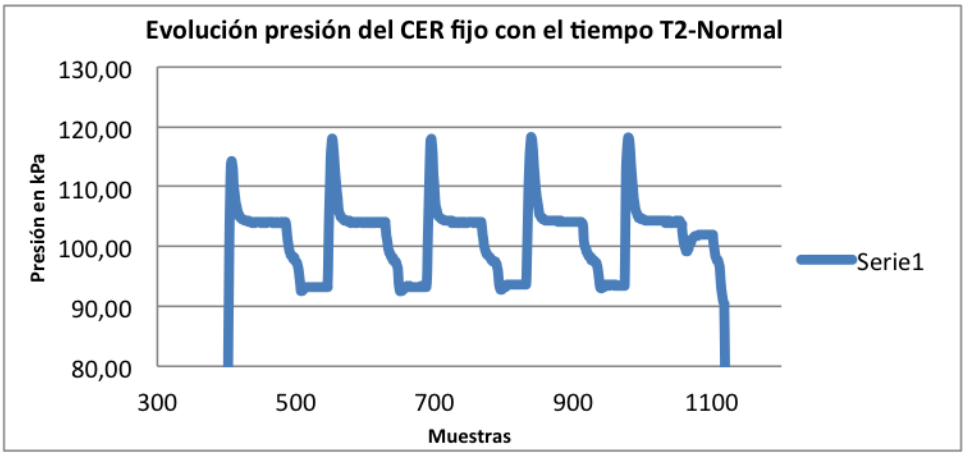


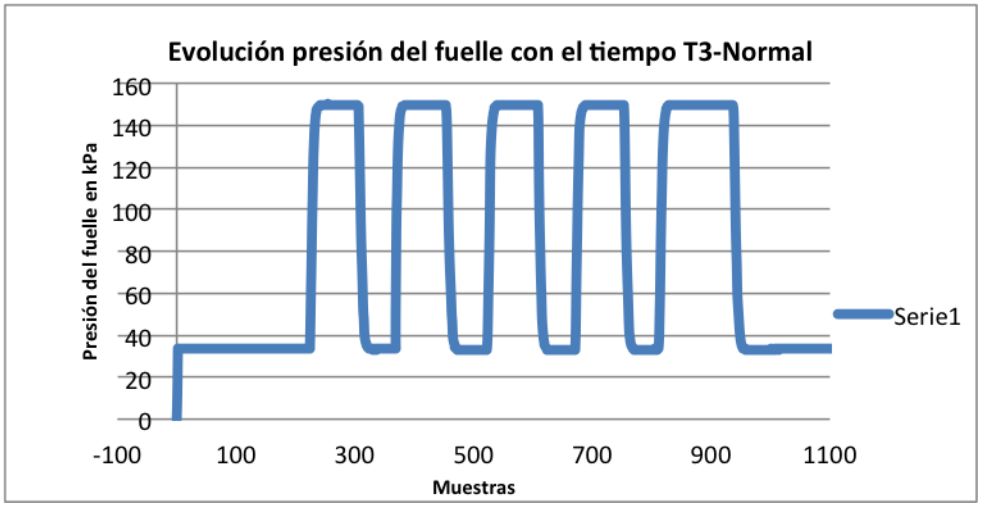



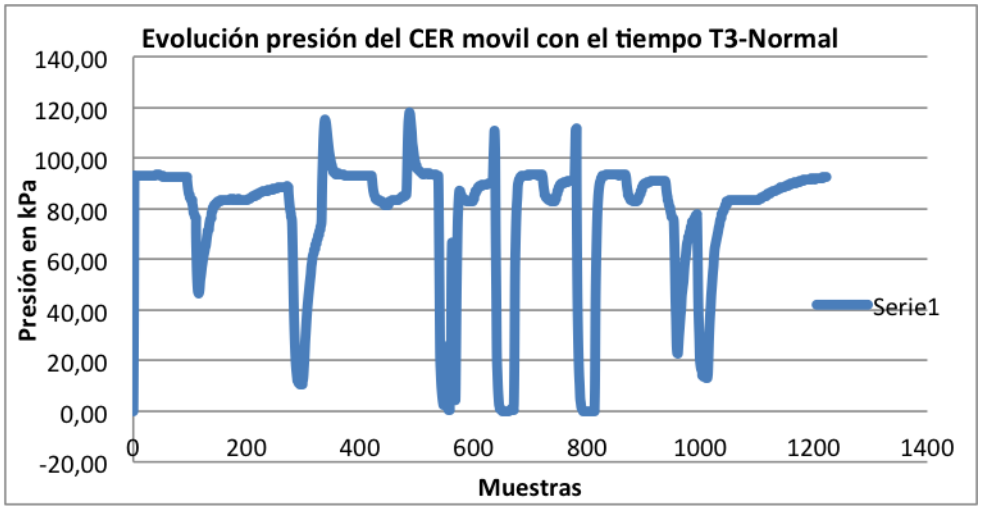



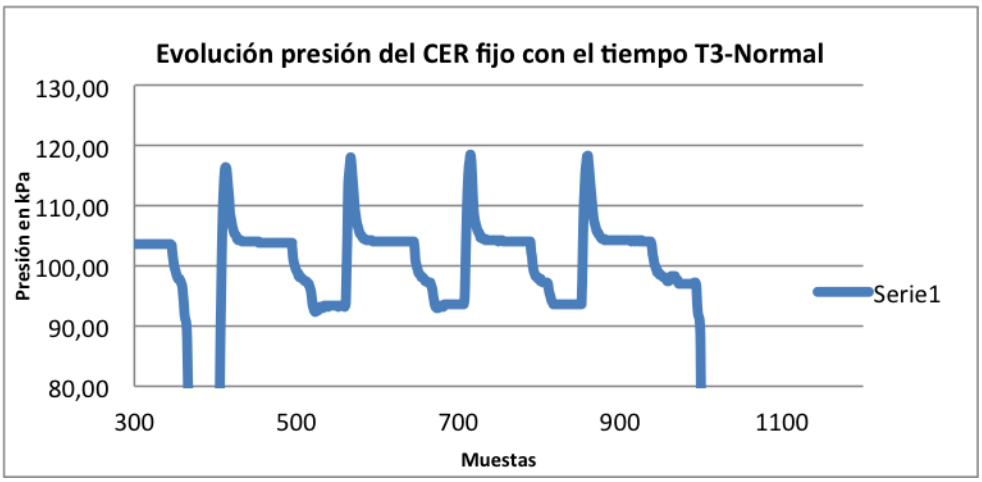

Evolución presión del fuelle con el tiempo T2-Normal
Presión del fuelle en kPa
160
140
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
Muestras
Serie1

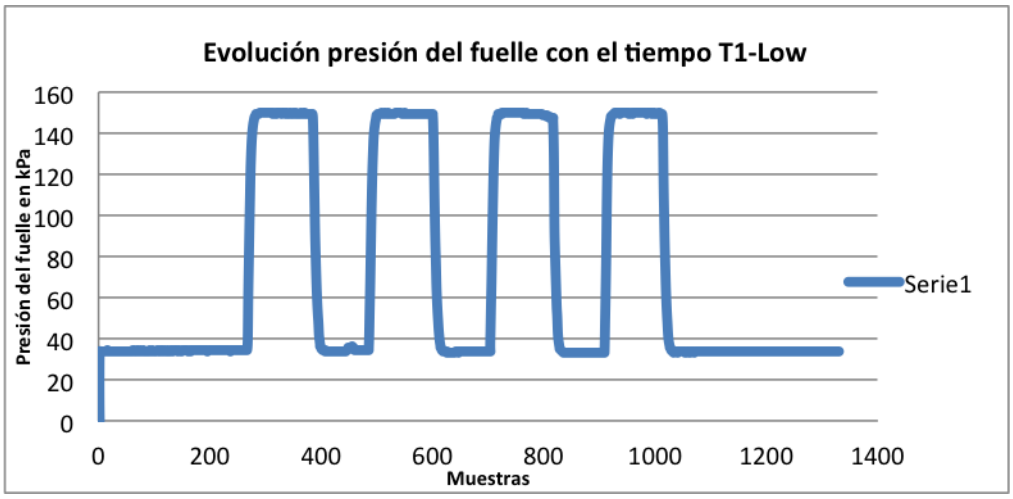

Evolución presión del CER movil con el tiempo T2-Normal
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
400
600
800
1000
1200
1400
Muestras
Serie1

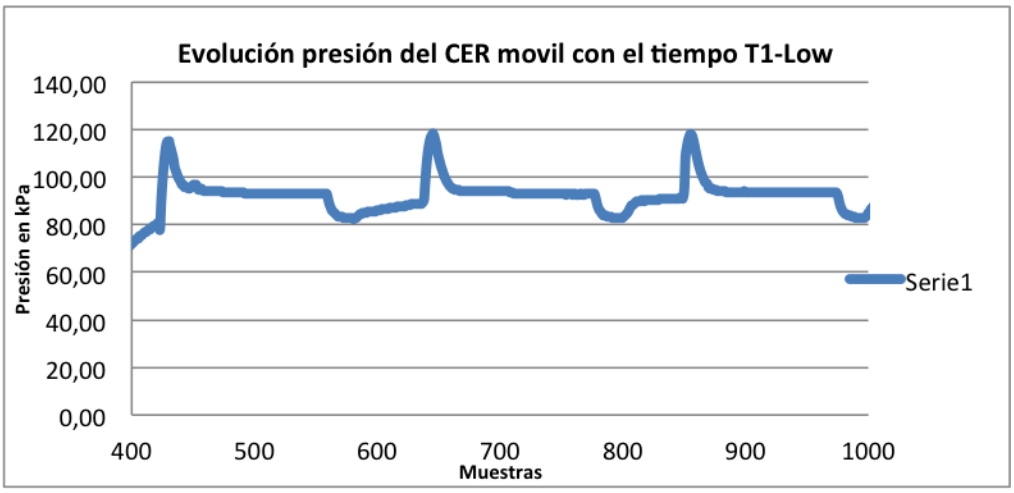

Evolución presión del CER fijo con el tiempo T2-Normal
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
300
500
700
900
1100
Muestras
Serie1

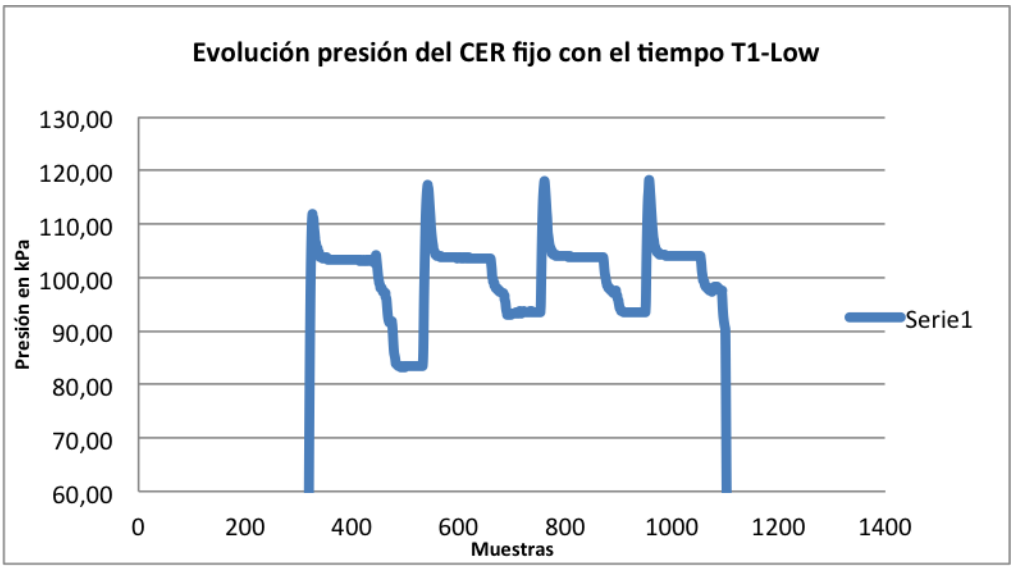

Evolución presión del fuelle con el tiempo T3-Normal
Presión del fuelle en kPa
160
140
120
100
80
60
40
20
0
-100
100
300
500
700
900
1100
Muestras
Serie1

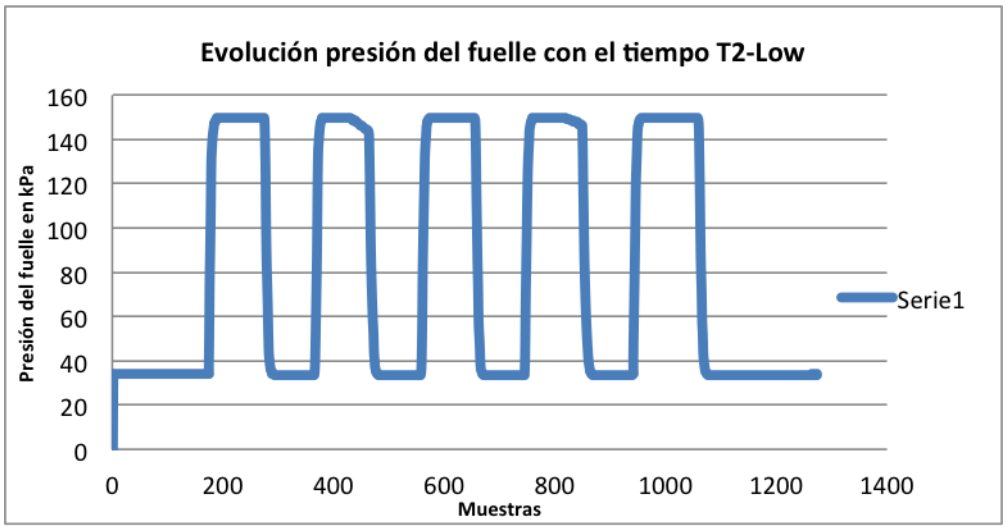

Evolución presión del CER movil con el tiempo T3-Normal
Presión en kPa
140,00
120,00
100,00
80,00
60,00
40,00
20,00
0,00
-20,00
0
200
400
600
800
1000
1200
1400
Muestras
Serie1

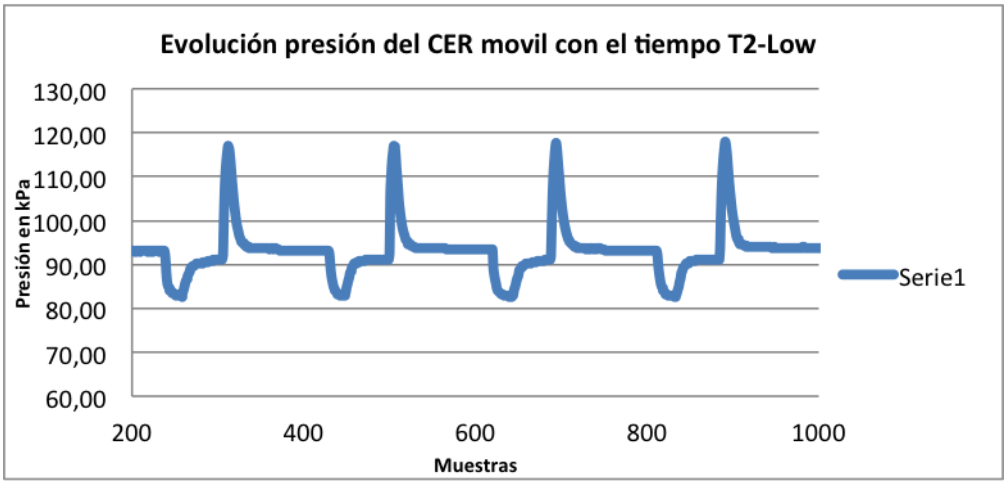

Evolución presión del CER fijo con el tiempo T3-Normal
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
300
500
700
900
1100
Muestas
Serie1

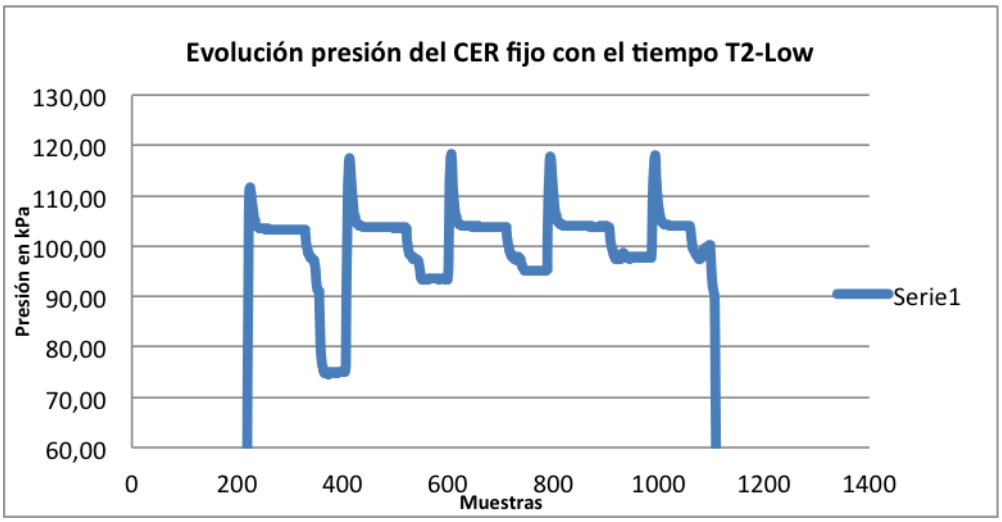

Evolución presión del fuelle con el tiempo T1-Low
Presión del fuelle en kPa
160
140
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
Muestras
Serie1

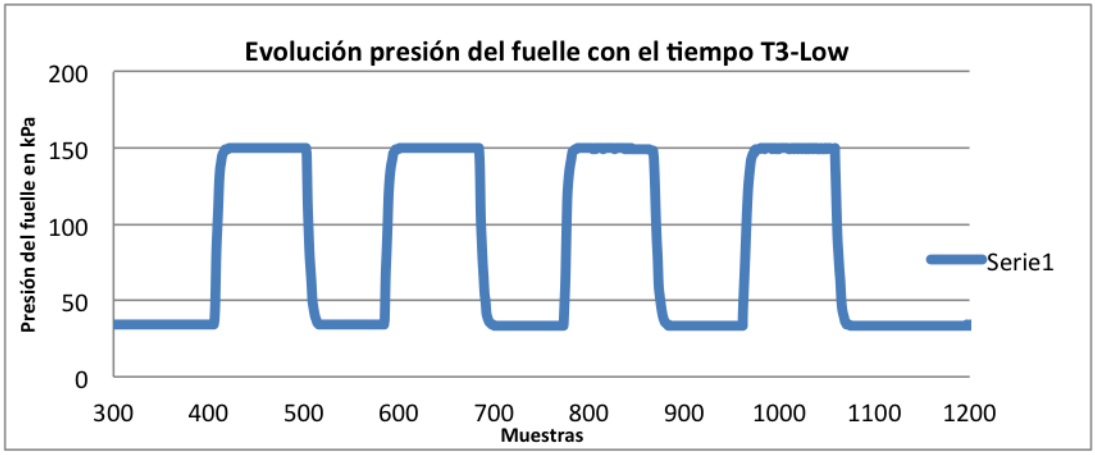

Evolución presión del CER movil con el tiempo T1-Low
Presión en kPa
140,00
120,00
100,00
80,00
60,00
40,00
20,00
0,00
400
500
600
700
800
900
1000
Muestras
Serie1

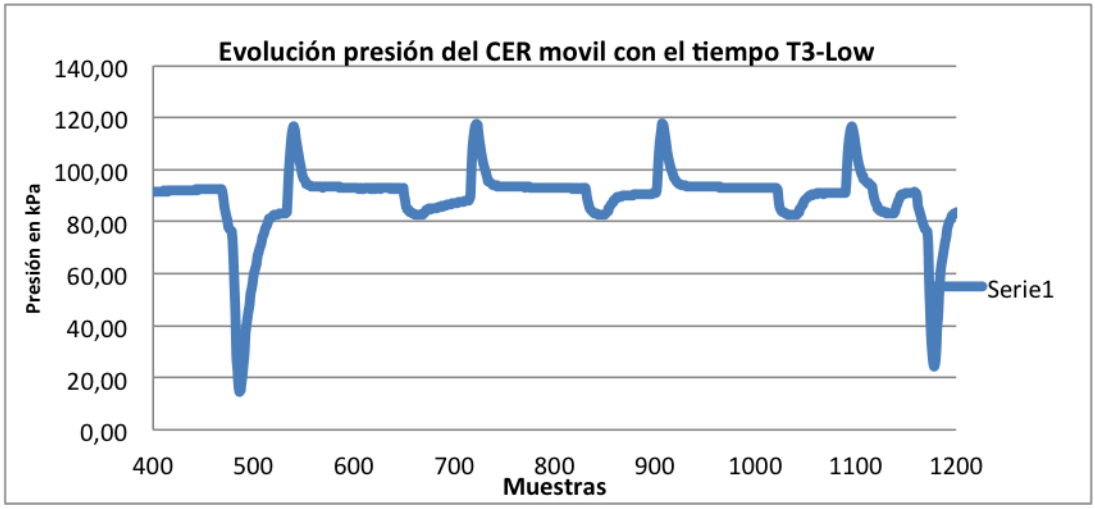

Evolución presión del CER fijo con el tiempo T1-Low
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
0
200
400
600
800
1000
1200
1400
Muestras
Serie1

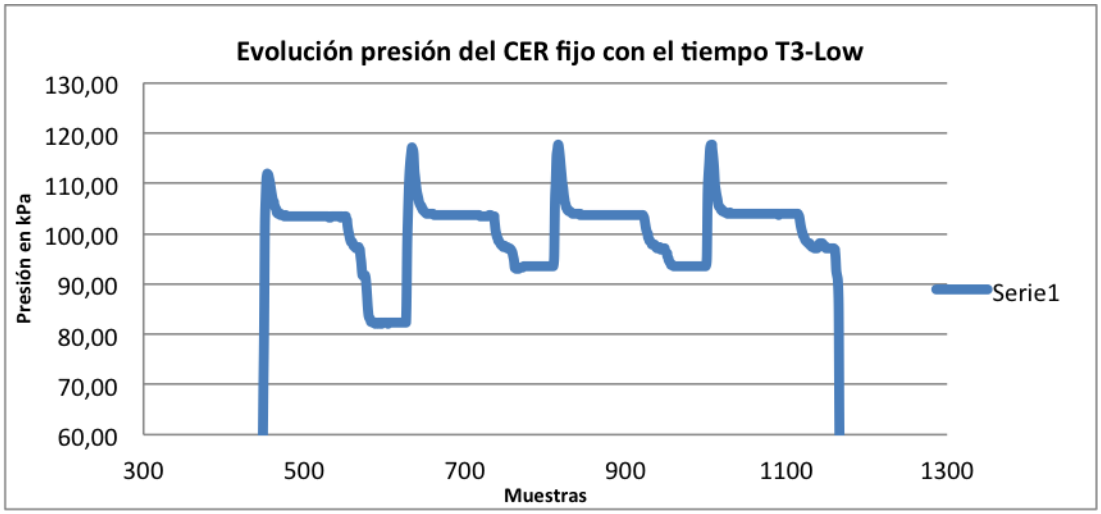

Evolución presión del fuelle con el tiempo T2-Low
Presión del fuelle en kPa
160
140
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
Muestras
Serie1

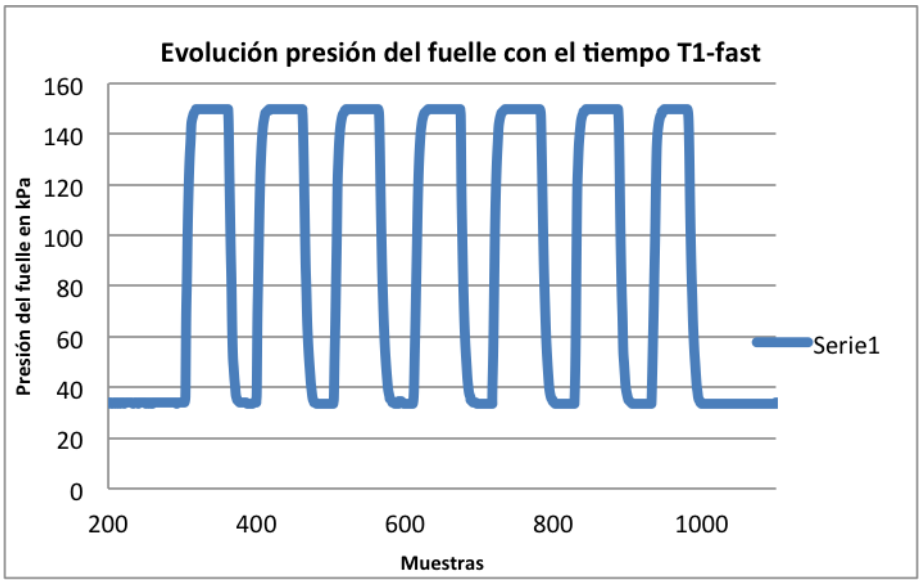

Evolución presión del CER movil con el tiempo T2-Low
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
200
400
600
800
1000
Muestras
Serie1

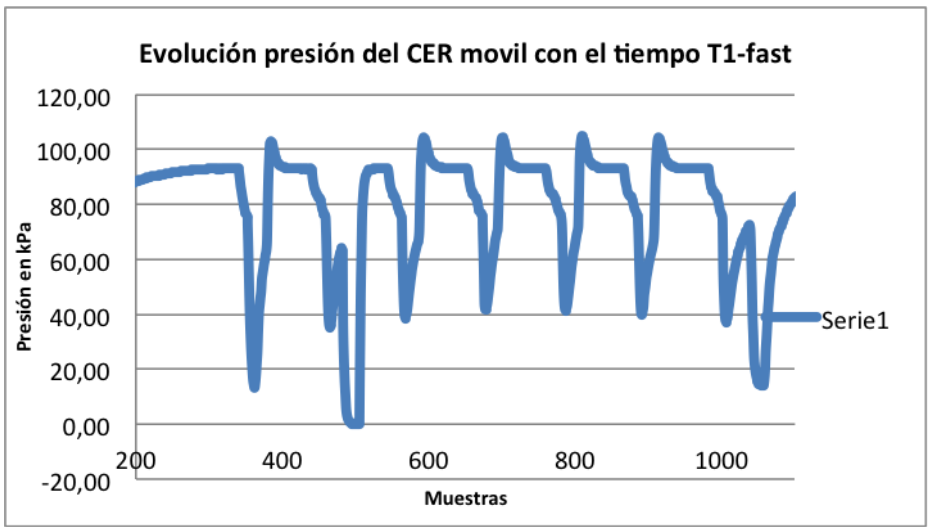

Evolución presión del CER fijo con el tiempo T2-Low
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
0
200
400
600
800
1000
1200
1400
Muestras
Serie1

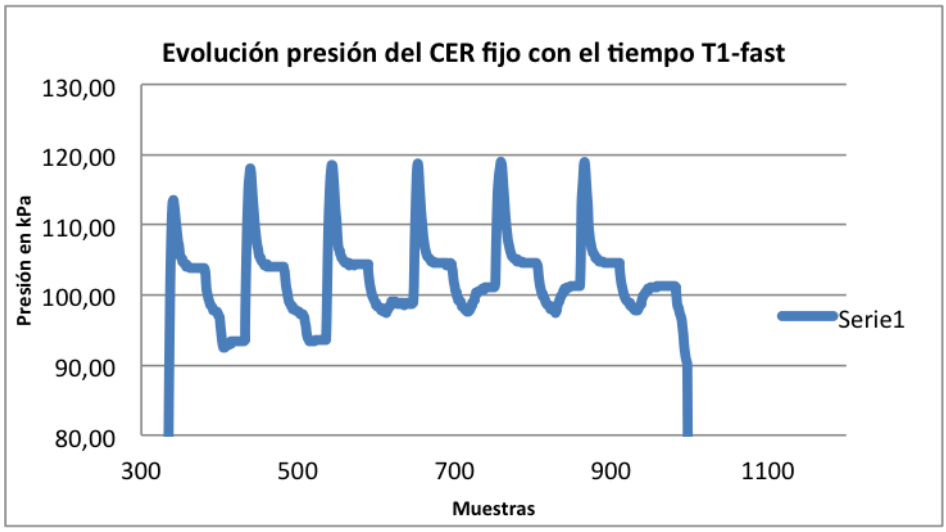

Evolución presión del fuelle con el tiempo T3-Low
Presión del fuelle en kPa
200
150
100
50
0
300
400
500
600
700
800
900
1000
1100
1200
Muestras
Serie1

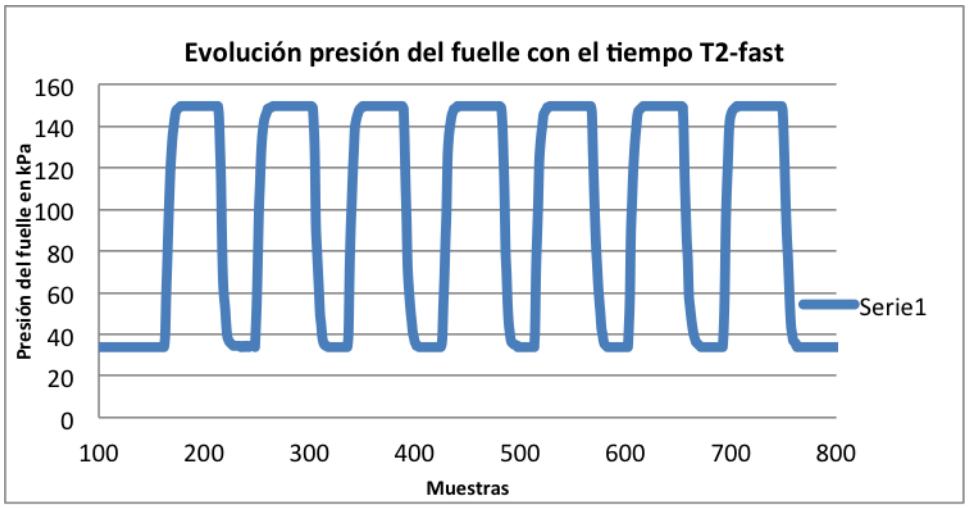

Evolución presión del CER movil con el tiempo T3-Low
Presión en kPa
140,00
120,00
100,00
80,00
60,00
40,00
20,00
0,00
400
500
600
700
800
900
1000
1100
1200
Muestras
Serie1

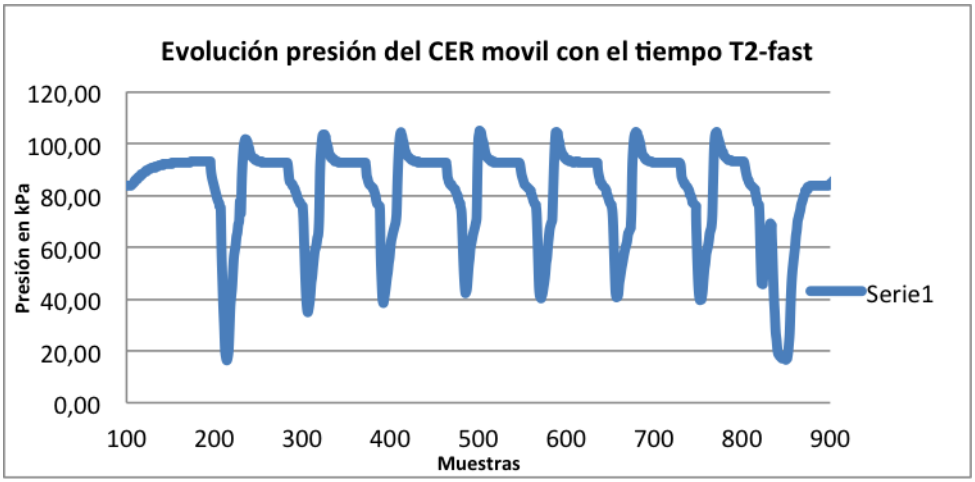

Evolución presión del CER fijo con el tiempo T3-Low
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
300
500
700
900
1100
1300
Muestras
Serie1

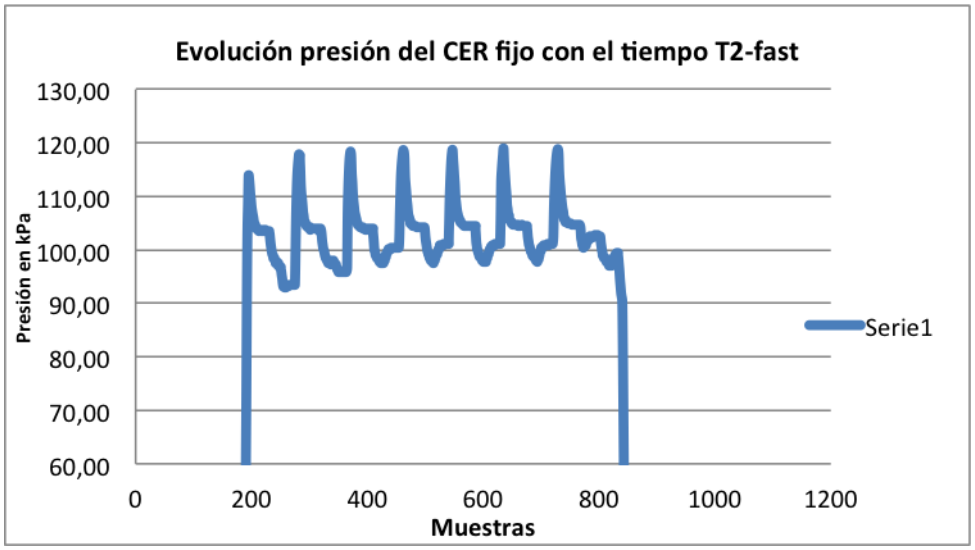

Evolución presión del fuelle con el tiempo T1-fast
Presión del fuelle en kPa
160
140
120
100
80
60
40
20
0
200
400
600
800
1000
Muestras
Serie1

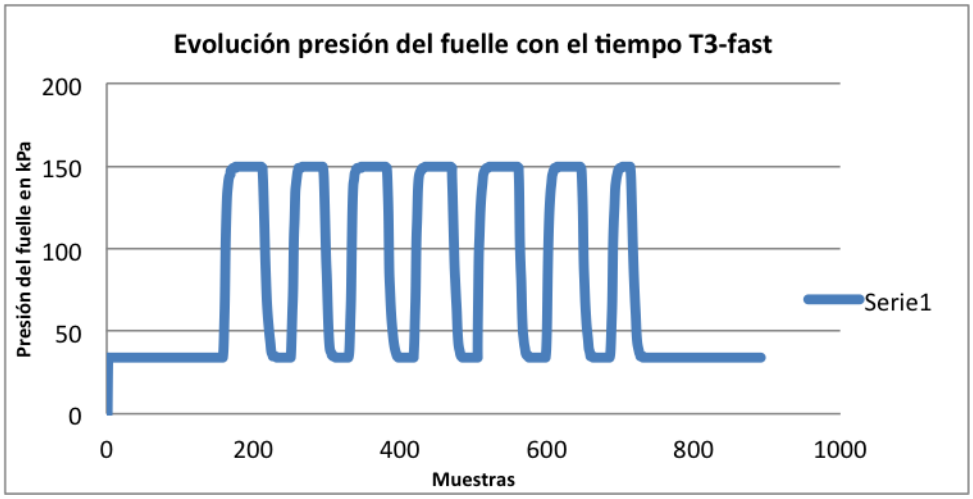

Evolución presión del CER movil con el tiempo T1-fast
Presión en kPa
120,00
100,00
80,00
60,00
40,00
20,00
0,00
-20,00
200
400
600
800
1000
Muestras
Serie1

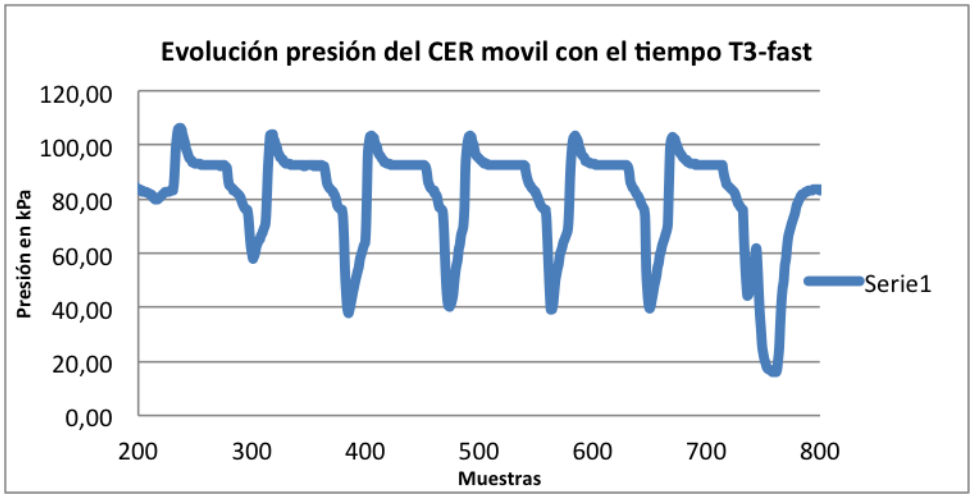

Evolución presión del CER fijo con el tiempo T1-fast
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
300
500
700
900
1100
Muestras
Serie1

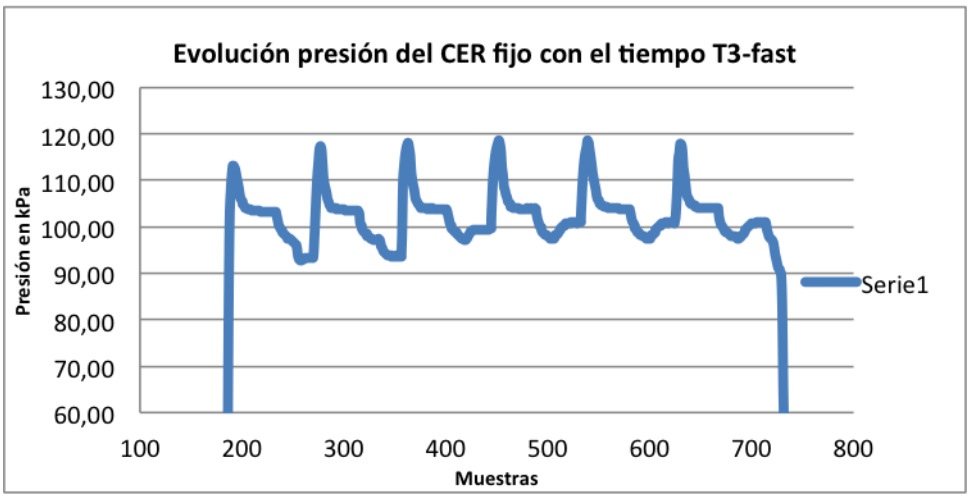

Evolución presión del fuelle con el tiempo T2-fast
Presión del fuelle en kPa
160
140
120
100
80
60
40
20
0
100
200
300
400
500
600
700
800
Muestras
Serie1


Evolución presión del CER movil con el tiempo T2-fast
Presión en kPa
120,00
100,00
80,00
60,00
40,00
20,00
0,00
100
200
300
400
500
600
700
800
900
Muestras
Serie1


Evolución presión del CER fijo con el tiempo T2-fast
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
0
200
400
600
800
1000
1200
Muestras
Serie1


Evolución presión del fuelle con el tiempo T3-fast
Presión del fuelle en kPa
200
150
100
50
0
0
200
400
600
800
1000
Muestras
Serie1


Evolución presión del CER movil con el tiempo T3-fast
Presión en kPa
120,00
100,00
80,00
60,00
40,00
20,00
0,00
200
300
400
500
600
700
800
Muestras
Serie1


Evolución presión del CER fijo con el tiempo T3-fast
Presión en kPa
130,00
120,00
110,00
100,00
90,00
80,00
70,00
60,00
100
200
300
400
500
600
700
800
Muestras
Serie1

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Trabajo final de grado

Grado en ingeniería electrónica industrial y automática

# Sistema de monitorización del prototipo Endoworm 3.0

# 2. Planos

Departamento de ingeniería electrónica

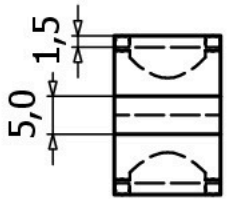
Curso 2015/2016

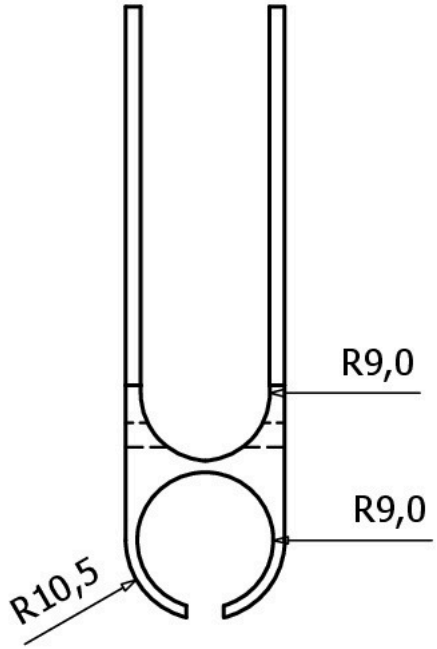
Autora:
Beatriz Almendros Luis

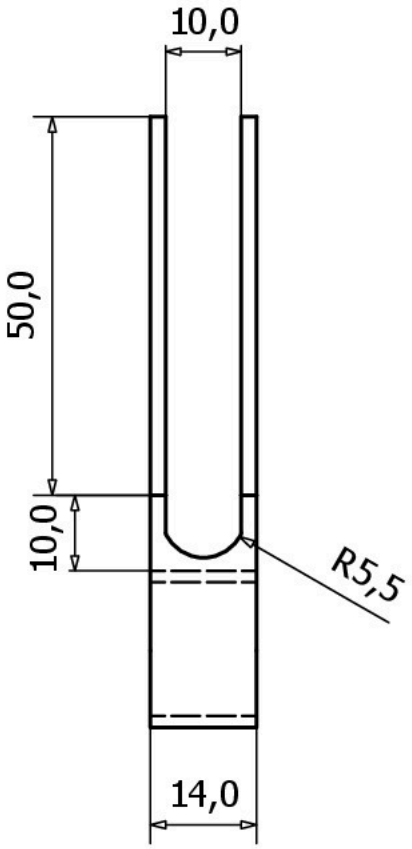
Tutor:
Carlos Sánchez Díaz

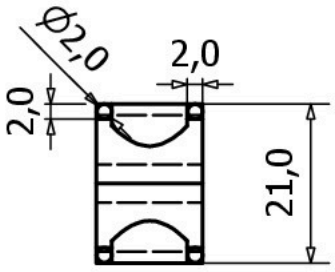
Valencia, septiembre 2016

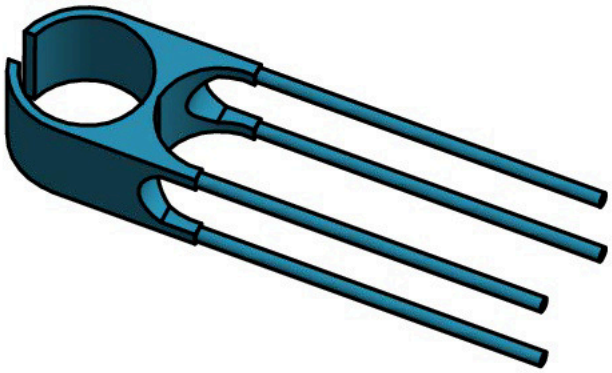
# 2. Planos



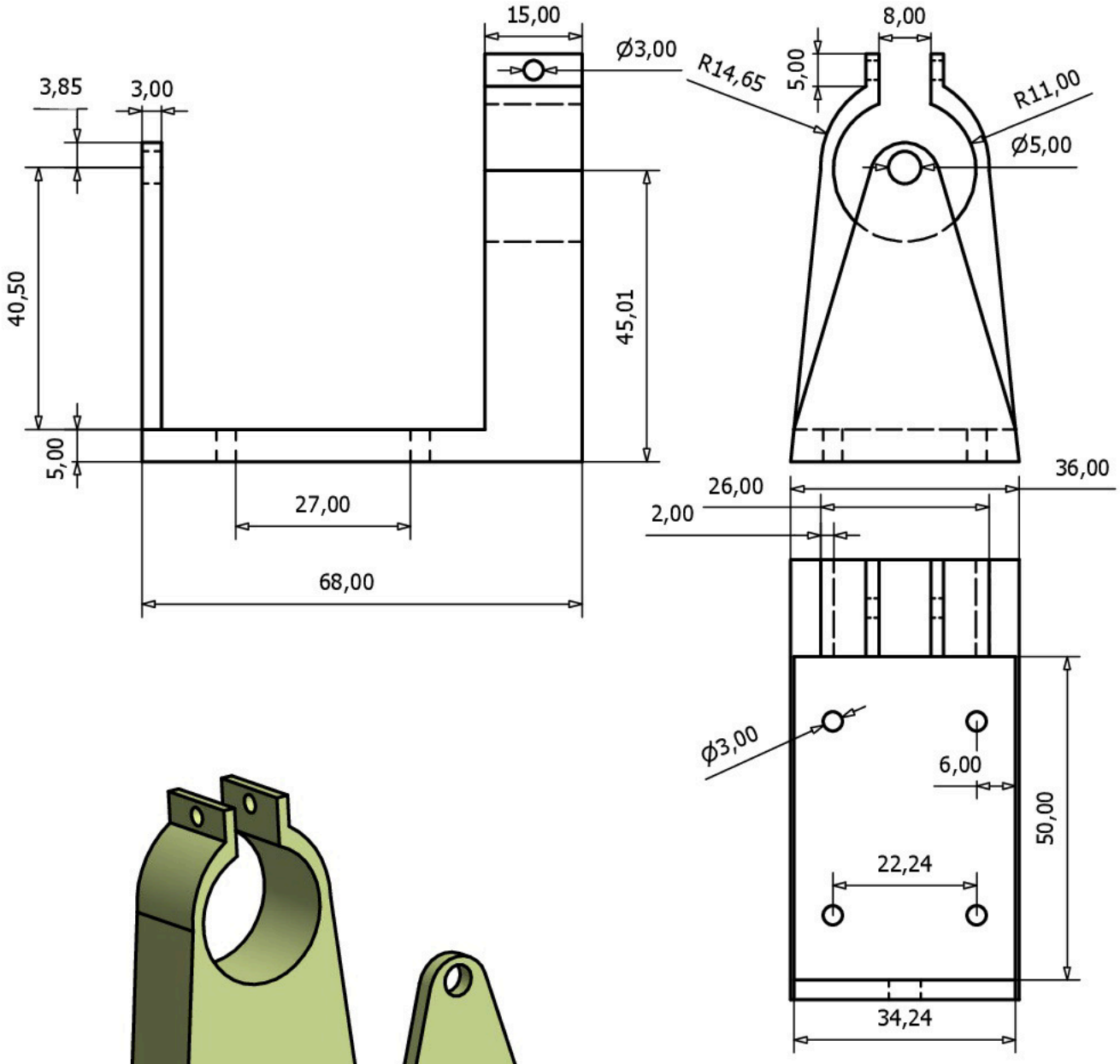



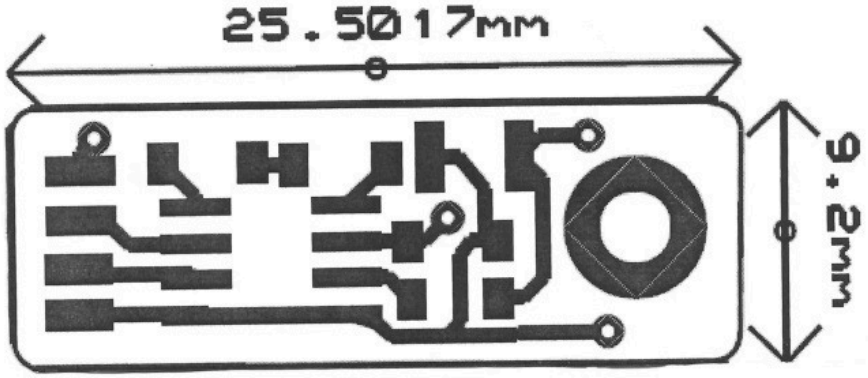
| Medida en mm | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 29/05/2016 | Beatriz Almendros Luis | |
| Revisado por: | 01/06/2016 | Carlos Sánchez Díaz | |
| Escala: 1:1 | Horquilla sujección rectángulo dentado | Proyecto: Endoworm 3.0. | Número del plano: 2016.1 |



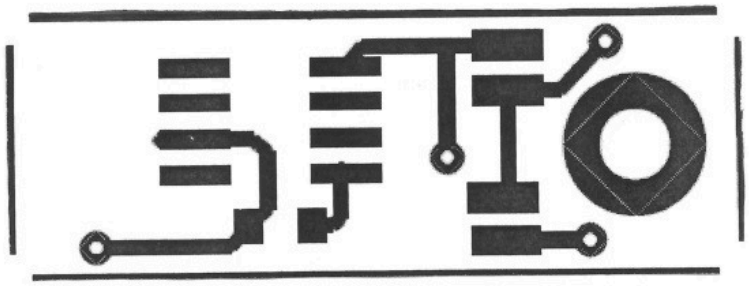



| Medidas en mm | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 29/05/2016 | Beatriz Almendros Luis | |
| Revisado por: | 01/06/2016 | Carlos Sánchez Díaz | |
| Escala: 1:1 | Soporte sensor de posición | Proyecto: Endoworm 3.0. | Número del plano: 2016.2 |

# TOP

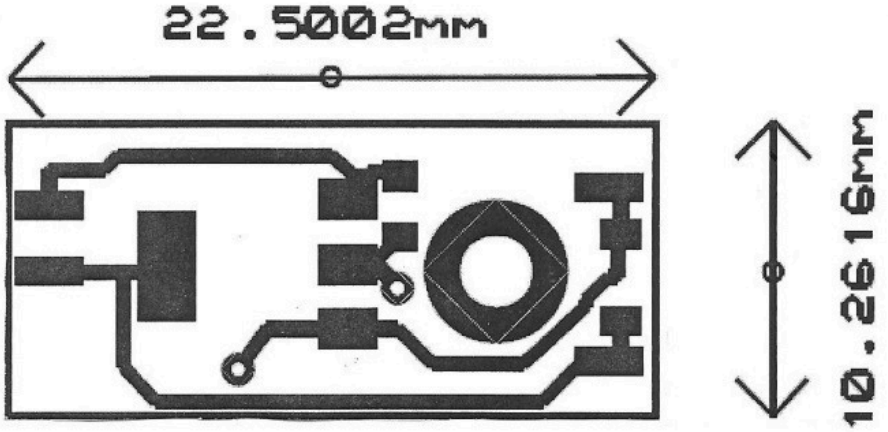



# BOTTOM

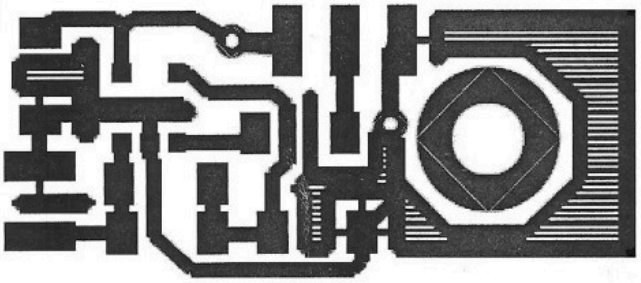


| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA | |
|---|---|---|---|---|
| Dibujado por: | 22/07/2016 | Beatriz Almendros Luis | | |
| Revisado por: | 01/09/2016 | Carlos Sánchez Díaz | | |
| Escala: 4:1 | Pistas PCB sensor de presión Top y Bottom | | Proyecto: Endoworm 3.0. | Número del plano: 2016.3 |

# TOP

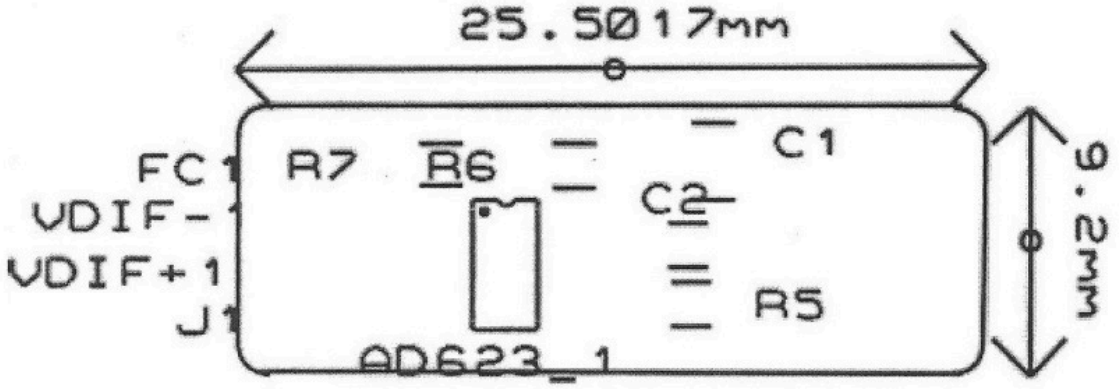



# BOTTOM

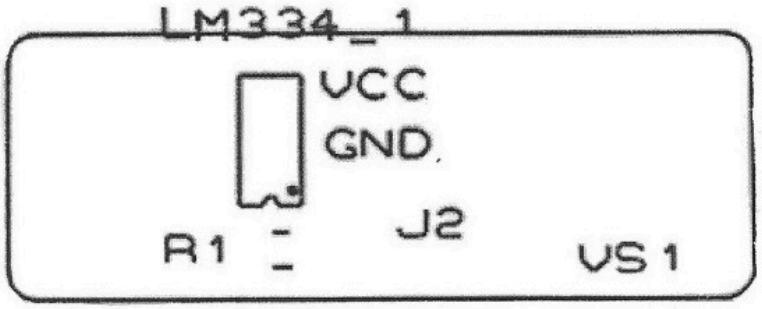


| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 22/07/2016 | Beatriz Almendros Luis | |
| Revisado por: | 01/09/2016 | Carlos Sánchez Díaz | |
| Escala: 4:1 | Pistas PCB regulador de tensión Top y Bottom | Proyecto: Endoworm 3.0. | Número del plano: 2016.4 |

# TOP

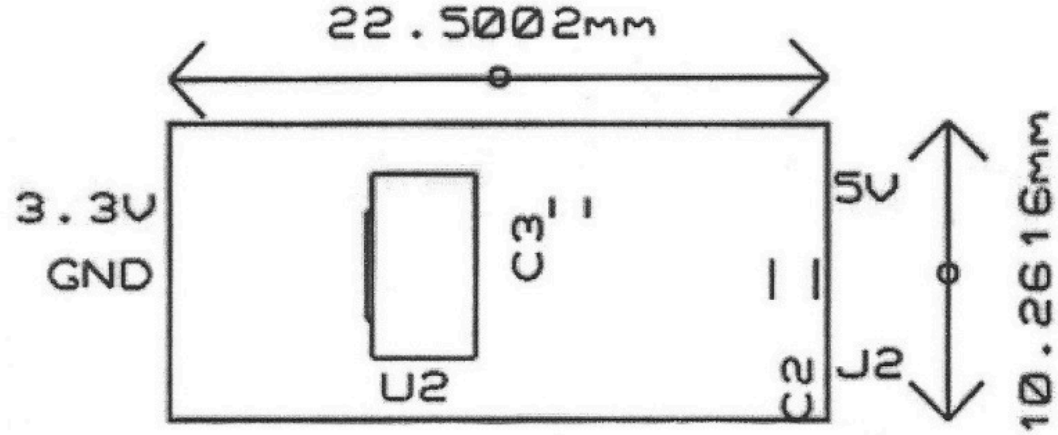



# BOTTOM

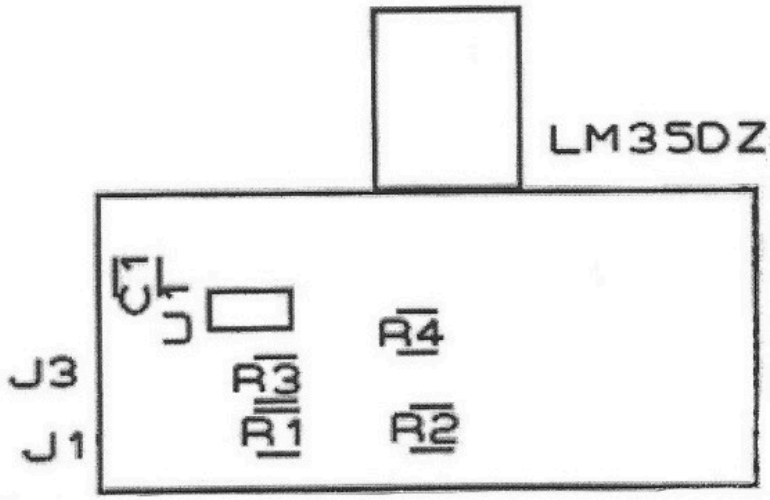



| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 22/07/2016 | Beatriz Almendros Luis | |
| Revisado por: | 01/09/2016 | Carlos Sánchez Díaz | |
| Escala: 4:1 | Componentes PCB sensor de presión Top y Bottom | Proyecto: Endoworm 3.0. | Número del plano: 2016.5 |

# TOP

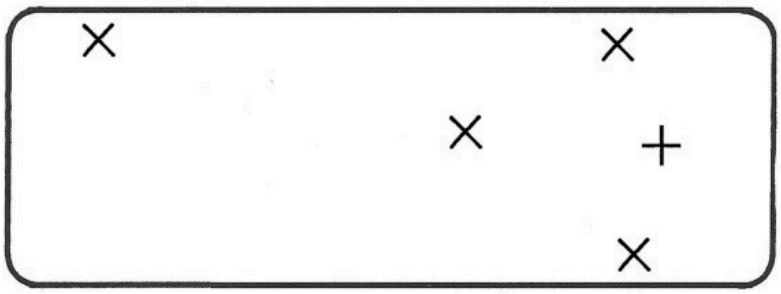



# BOTTOM




| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 22/07/2016 | Beatriz Almendros Luis | |
| Revisado por: | 01/09/2016 | Carlos Sánchez Díaz | |
| Escala: 4:1 | Componentes PCB regulador de tensión Top y Bottom | Proyecto: Endoworm 3.0. | Número del plano: 2016.6 |

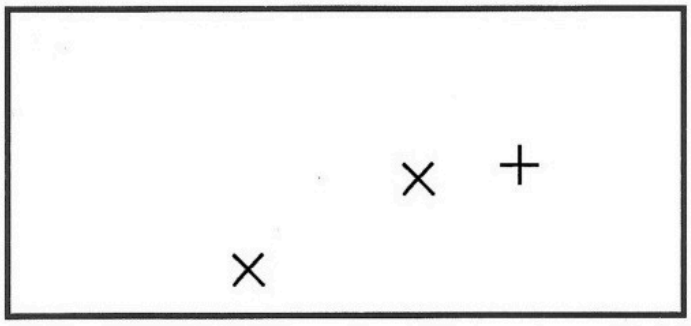
# TOP



+ 2.5mm (1)  × Ø.5mm (4)

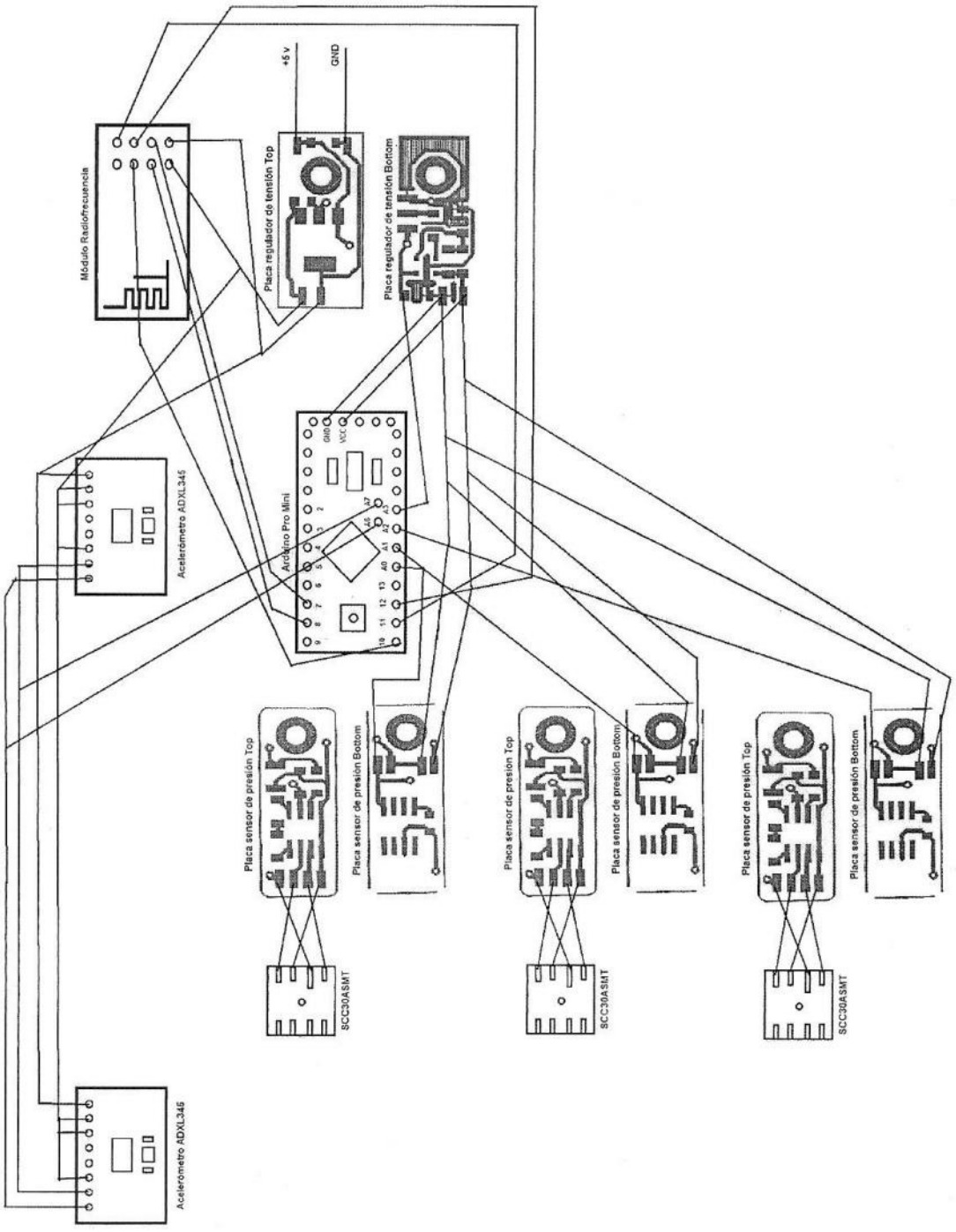
| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 22/07/2016 | Beatriz Almendros Luis | |
| Revisado por: | 01/09/2016 | Carlos Sánchez Díaz | |
| Escala: 4:1 | Taladros PCB sensor de presión | Proyecto: Endoworm 3.0. | Número del plano: 2016.7 |

# TOP



+ 2.5mm (1)  × Ø.5mm (2)

| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA | |
|---|---|---|---|---|
| Dibujado por: | 22/07/2016 | Beatriz Almendros Luis | | |
| Revisado por: | 01/09/2016 | Carlos Sánchez Díaz | | |
| Escala: 4:1 | Taladros PCB regulador de tensión | | Proyecto: Endoworm 3.0. | Número del plano: 2016.8 |




| | Fecha | Nombre | UNIVERSITAT POLITÈCNICA DE VALÈNCIA |
|---|---|---|---|
| Dibujado por: | 05/09/2016 | Beatriz Almendros Luis | |
| Revisado por: | 06/09/2016 | Carlos Sánchez Díaz | |
| Escala: No | Esquema de conexiones sistema completo | Proyecto: Endoworm 3.0. | Número del plano: 2016.9 |

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Trabajo final de grado

Grado en ingeniería electrónica industrial y automática

# Sistema de monitorización del prototipo Endoworm 3.0

## 3. Pliego de condiciones

Departamento de ingeniería electrónica

Curso 2015/2016

Autora:
Beatriz Almendros Luis

Tutor:
Carlos Sánchez Díaz

Valencia, septiembre 2016

# 3. Pliego de condiciones

## 3.1 Definición y alcance

El objeto del presente documento es fijar las condiciones técnicas mínimas que debe cumplir el diseño y desarrollo del sistema de monitorización del prototipo Endorworm 3.0, especificando los requisitos de fiabilidad y funcionalidad del sistema. El ámbito de aplicación de este documento se extiende a todos los sistemas mecánicos, eléctricos y electrónicos que forman parte del diseño. En determinados supuestos se podrán adoptar, por la propia naturaleza del mismo o del desarrollo tecnológico, soluciones diferentes a las exigidas en este documento, siempre que quede suficientemente justificada su necesidad y que no impliquen una disminución de las exigencias mínimas de calidad especificadas en el mismo

## 3.2 Características de los materiales

### 3.2.1 Alimentación

La alimentación a emplear en el sistema es tensión continua de +5 V. Se pretende estipular una tensión de referencia en la cual fijar la alimentación de todos los componentes a emplear. Se suministrará mediante dos cables, alimentación y masa.

### 3.2.2 Sensor de presión

En la selección del sensor de presión el factor principal que decidirá el modelo a escoger es el tamaño, se necesita un sensor lo suficientemente pequeño para no interrumpir el paso del endoscopio. Para ello se ha escogido un sensor de presión absoluta alimentado mediante corriente, el sensor seleccionado es de la marca Honeywell y pertenece a la serie SCC. Se trata de un sensor de tamaño muy reducido que ofrece una gama de rangos de medida muy amplia. En este caso el rango de medida debe ser mayor que la máxima presión que se introducirá en el sistema, esta es de 22 psi (150 kPa) por lo que el rango superior del sensor debe ser superior a los 23 psi. El único modelo que cumple con estas especificaciones dentro de la serie SCC es el modelo SCC30ASMT que ofrece un rango superior de 30 psi. Estos sensores ofrecen un voltaje diferencial a la salida proporcional a la presión ejercida sobre él.

### 3.2.3 Fuente de corriente

Debido a la necesidad de una alimentación para el sensor de presión mediante corriente, distinta a la estipulada en el inicio de este apartado de tensión continua de +5 V, es necesario introducir una fuente de corriente que alimente al sensor.
Los criterios de selección en este caso son; en primer lugar el tamaño de la fuente, seguido de la corriente que es capaz de ofrecer, como la corriente máxima de entrada permitida por el sensor es de 1,5 mA bastará con tomar ese valor como máximo para la fuente. El modelo escogido es el LM334 de la marca Texas instruments, se trata de un componente de tamaño reducido que es capaz de ofrecer hasta 10 mA de corriente a la salida. Su configuración es sencilla, solamente con una resistencia se puede regular la salida.

### 3.2.4 Sensor de temperatura

Uno de los principales factores que modifican el comportamiento de la fuente de corriente es la temperatura, este varia la corriente ofrecida en 1,68 μA por cada grado. Por este motivo es necesario incluir en el sistema un sensor de temperatura que permita calcular la variación de corriente.
El modelo escogido es el LM35DZ de la compañía Texas Instruments, se trata de un sensor alimentado a +5 V que ofrece una tensión a la salida directamente proporcional a la temperatura medida.

### 3.2.5 Sensor de desplazamiento

Un dato importante dentro del funcionamiento del prototipo Endoworm 3.0 es el avance que este es capaz de realizar en cada iteración. Para ello se han empleado dos acelerómetros, uno fijo situado en el inicio del sistema y otro en la zona móvil, que permitirán, mediante la resta del fijo con el móvil, obtener la distancia recorrida. Estos sensores son el modelo ADXL345, se tratan de módulos ya implementados que permiten un montaje y una programación sencilla, uno de los problemas de estos módulos es la tensión a la cual trabajan ya que no es a la estándar marcada al inicio de +5 V, sino que operan con una alimentación de +3,3 V.
En este proyecto no se va a desarrollar la implantación de estos sensores, solamente se va a incluir en el sistema final debido a que su programación ya esta realizada.

### 3.2.6 Microcontrolador

Para el desarrollo de este proyecto es necesario un microcontrolador que permitiera obtener los valores ofrecidos por los sensores y al mismo tiempo permitiera el procesado de los mismos.
La mejor opción es un Arduino, dentro de esta compañía existen muchos modelos dependiendo del número de puertos necesarios y de la capacidad de operación del microcontrolador insertado. En este caso se seleccionó el Arduino Leonardo debido a la capacidad que poseía el microcontrolador de realizar cálculos en coma flotante.
El problema de este modelo era el tamaño para introducirlo dentro del sistema que va acoplado al endoscopio, por lo que se va a utilizar como soporte para el módulo de radiofrecuencia que reciba en el exterior los datos emitidos por el módulo interno. Además será el encargado de realizar los cálculos que permitan depurar los valores obtenidos.
En el caso del Arduino que va interno en el sistema se necesita un modelo, que manteniendo los puertos necesarios para la conexión de todos los sensores, tenga un tamaño lo más reducido posible. Con estas características se ha escogido el Arduino pro mini, este tiene 12 pines digitales, en los cuales vienen integrados los pines de ICSP y 6 pines analógicos, en los cuales están incorporados los dos pines necesarios para el bus serie de datos ($I^2C$), necesarios para los acelerómetros.

### 3.2.7 Sistema de comunicación inalámbrica

Uno de los principales problemas en este sistema es el espacio que debe ocupar, por este motivo se pretende enviar los datos obtenidos mediante un sistema de comunicación inalámbrica que permita eliminar el máximo número de cables posibles. Para ello se va a implementar un transceptor de radiofrecuencia que realiza envíos y recepción de datos, desde valores enteros hasta valores negativos y con decimales. El modelo de este módulo es el NRF2401, se trata de un componente económico y con gran versatilidad en el tratamiento de los datos. Al igual que con los sensores de desplazamiento anteriormente mencionado, estos dispositivos trabajan con una alimentación de +3,3 V, una tensión diferente a la estipulada como estándar del sistema.

### 3.2.8 Regulador de tensión

Dado que dos de los componentes incluidos en el sistema, acelerómetros y módulo de radiofrecuencia, no operan con la tensión fijada de +5 V es necesario introducir un elemento que regule esta tensión de entrada por la necesaria para estos componentes de +3,3 V.

Para ello se va a emplear un regulador de tensión DC/DC de entrada +5 V y salida de +3,3 V. Uno de los factores a tener en cuenta en la elección del modelo es la corriente que este es capaz de ofrecer, sabiendo que un acelerómetro consume 140 μA y el módulo de radiofrecuencia cuando esta en modo emisor es de 115 mA, el regulador de tensión debe ofrecer al menos 140 mA de corriente.

Siguiendo estas condiciones el modelo escogido es el LP2985-33DBVT de la marca Texas instruments, se trata de un modelo con un tamaño mayor al deseado pero capaz de alimentar a todos los componentes presentes en el sistema ya que la corriente que es capaz de ofrecer es de 150 mA.

### 3.2.9 Amplificador operacional

Los datos que son necesarios obtener del sistema son; las 3 tensiones ofrecidas por los sensores de presión, la tensión de salida del sensor de temperatura y los datos ofrecidos por los acelerómetros. En el caso de los dos primeros las tensiones que ofrecen son de mV, las entradas analógicas de los Arduino escogidos solamente son capaces de detectar variaciones de 4,88 mV  este hecho produciría una perdida de información elevada. Para evitar este problema se van a emplear dos amplificadores operacionales que permitan elevar la tensión de mV a V.

Para el caso de los sensores de presión se ha escogido el modelo AD623ARZ de la marca Analog Devices, se trata de un amplificador single supply, se puede alimentar a una tensión y masa, no necesita una alimentación simétrica, este hecho permite mantener la tensión estándar estipulada y alimentar el amplificador entre +5 V y GND. La ganancia necesaria en este caso es de 100 V/V.

El amplificador seleccionado para el sensor de temperatura es el modelo MAX4250EUK-T de la empresa Maxim, se trata de un amplificador operacional con realimentación negativa con una alimentación tipo single supply, al igual que en el caso anterior, con un tamaño muy reducido y rail to rail que significa que es capaz de ofrecer a la salida una tensión muy próxima a la de alimentación. En este caso la ganancia necesaria es de 10 V/V.

### 3.2.10 Encapsulado de los sensores de presión

El encapsulado de los sensores de presión es esencial para el desarrollo del sistema, esto es debido a que los sensores a emplear necesitan de una cavidad estanca para poder realizar la medida correctamente. Para ello se va a encapsular con un cilindro de silicona  silastic 4820.

### 3.2.11 Condensadores

Los condensadores usados para la elaboración del sistema serán de tipo cerámico multicapa del valor determinado en la memoria del proyecto, serán de tipo montaje superficial con montaje máximo de 1210 y mínimo de 0805. Estos tendrán que estar conectados según los planos de circuito en el apartado *2. Planos*.

### 3.2.12 Resistencias

Las resistencias destinadas al desarrollo de las placas del sistema de monitorización deberán tener una tolerancia máxima de ±1%, este se debe a que no se pueden variar los valores de las resistencias ya que se modificaría el comportamiento de todo el sistema. El tamaño de las mismas deben ser al igual que en apartado anterior de montaje superficial y con un montaje máximo de 1210 y mínimo de 0805.

## 3.3 Ejecución

### 3.4.1 Especificaciones

#### *Encapsulado de los sensores de presión*

Este proceso se trata del paso más importante en la ejecución de todo el sistema debido a que una fuga haría que el prototipo no trabaje correctamente y por tanto se obtuvieran datos incorrectos. Una vez soldados los cables al sensor de presión, tal como se comenta en el apartado *1.3.1 Sensor de presión* de la memoria, se introduce en el cilindro de silicona de 1 cm de diámetro. La longitud del encapsulado para cada sensor es de 3,5 cm. El sensor debe estar situado en la zona central, además se incluyen dos tubos de plástico flexible, uno por cada entrada del encapsulado. Estos tubos serán los encargados de permitir el acceso y la salida del aire procedente de la bomba a la cavidad en cuestión. Una vez colocado se añaden dos bridas de tamaño pequeño a los extremos ajustándolas al diámetro del encapsulado sin apretar demasiado. Tras esto se cierran los extremos con silicona transparente que seque al aire, hay que llevar especial cuidado con que todo el área este bien cubierta, también es recomendable prolongar un poco más de silicona sobre el cable y los tubos para mejorar el sellado. Para finalizar se aprietan un poco las bridas (sin llegar a obstruir el paso del aire por el tubo) y se elimina el exceso de silicona. En este momento se debe dejar curar la silicona durante 24 horas, transcurrido el tiempo se comprueba que no haya ninguna fuga conectado uno de los tubos con una de las cavidades y por el otro introduciendo aire con una jeringuilla. En caso afirmativo se continúa terminando el proceso de encapsulado rodeando el cilindro con cinta de alta fijación, la necesidad de añadir este elemento se debe a que la silicona cuando se introduce el aire con mucha presión se hincha e impide que el aire salga de la cavidad, para evitar la deformación elástica que sufre a altas presiones se crea una barrera a su alrededor que impide ese hecho. Las tiras deben ser finas para una mejor adaptación a la forma del cilindro. El resulta se puede observar en la *Figura 11* de la memoria.

### *Diseño y fabricación de placas*

Para una optimización del espacio la mejor opción se trata de lafabricación de unas placas PCB (*Printed Circuit Board*o placa de circuito impreso)en las que colocar los componentes anteriormente comentados,se necesita que estén fijos y bien colocados. Para el diseño de cada una de las placas se ha de saber que componentes irán en cada una de ellas, para ello se van a diferenciar dos placas, una será la encargada de situar cerca de los sensores de presión la fuente de corriente y el amplificador AD623ARZ. Por el otro lado la segunda placa que se va a diseñar albergará el regulador de tensión, el sensor de temperatura y el amplificador operacional MAX4250EUK-T. A la primera se le ha puesto el nombre de placa del sensor de presión y a la segunda placa del regulador de tensión.
Ambos diseños se pueden observar con detalle en el apartado *1.3.2 Diseño e implementación de placas* de la memoria, en este apartado también se puede encontrar el proceso de fabricación explicado detalladamente.

### *Programación del microcontrolador emisor y receptor*

La programación de la placa Arduino se ha realizado con el propio software facilitado por la compañía. Se trata de un programa sencillo y con un lenguaje similar al C, en el Anexo V de la memoria se puede ver un manual de usuario introductorio para el uso del programa.
El programa emisor será el encargado de obtener los datos de los diferentes sensores, convertirlos de bits a valores analógicos y de enviarlos al módulo receptor conectado en el exterior.
En el caso del programa receptor este será el encargado de una vez recibidos los datos realizar los cálculos necesario para la conversión a las unidades deseadas y posteriormente serán visualizados por pantalla.
En el apartado *1.3.1 Materiales empleados/Control y Software del sistema* de la memoria se puede observar todos los pasos y comandos empleados para la programación de los microcontroladores.

### *Montaje sistema completo*

Una vez se tienen preparadas todas las partes del sistema y se ha comprobado su correcto funcionamiento se continúa con la unión de todas las partes. Para tener una idea más clara de cual será la situación de cada elemento se puede observar el *Esquema 7* de la memoria, en este esquema visualiza los 3 carros que son necesarios para su colocación y la posición de cada elemento sobre estos carros.
Todo el sistema de montaje queda explicado en el apartado *1.3.3. Montaje del sistema* de la memoria.

*Montaje sistema receptor*

El sistema receptor comprende al ordenador que servirá de apoyo a la visualización de los datos, al guardado de los mismos y a la alimentación del Arduino, del Arduino Leonardo, el módulo de radiofrecuencia con antena y del cable USB/microUSB que permitirá la conexión de los componentes.

El módulo de radiofrecuencia se conecta a la placa de Arduino, para ello es importante saber cual es la conexión entre ellos, el módulo posee 8 pines de los cuales se van a conectar 7. Los dos primeros se tratan de la alimentación y masa los cuales permiten el funcionamiento del módulo, se conectará la masa a uno de los pines GND del Arduino y la alimentación a la salida de +3,3 V, seguidos se encuentran los pines Cs y Ce que son los pines SPI (*Serial Peripheral Interface o* en castellano interfaz de periféricos serie) que se tratan de los pines necesarios para el intercambio de información entre diferentes dispositivos, en este caso se conectarán a los pines digitales 8 y 7 respectivamente. Por último están los pines ISCP (siglas en inglés de *In-Circuit Serial Programming*o en castellano Programación serie en el circuito*)* estos son los encargados de permitir la programación desde el Arduino al chip del módulo, este puerto realiza la programación directa del dispositivo sin tener que emplear un programador inicial que lo configure, ahora estos pines se colocan en los pines específicos situados en bloque ISCP del Arduino.

Una vez se ha conectado el módulo se conecta la placa al ordenador, para ello se empleará el cable anteriormente mencionado.

Tras haber abierto el software de Arduino y el fichero con la programación del mismo, se configura la placa mediante la opción herramientas de la barra superior, este paso queda mejor descrito en el manual de usuario de Arduino en el Anexo V de la memoria.

Una vez completada la configuración de la placa se carga el programa en la placa mediante el botón en la parte superior.

Tras comprobar que se ha subido el programa correctamente se inicia el programa Clear Terminal, se configura el puerto de entrada y la frecuencia de los datos, y así se comienza con la visualización de los valores que son recibidos, para un mejor entendimiento del uso del programa anterior se puede ver una explicación mas detallada en el apartado *1.3.1 Materiales empleados/Control y Software del sistema*de la memoria.

### 3.4.2 Control de calidad

*Circuito impreso*

Se observará la placa de circuito impreso tras su realización, comprobando que las pistas de cobre de la misma no hayan sufrido ningún daño por un exceso de ácido en el proceso de realización de la misma, se revisará que dicha placa presente un aspecto uniforme y que su diseño a priori se asemeje al realizado a través del programa de diseño.

Al realizar la perforación de la placa, se introducirá una taladradora adecuada a lo especificado en los planos del presente proyecto de manera que se eviten cortocircuitos entre componentes.

#### *Componentes eléctricos*

Se observará que los componentes electrónicos adquiridos no presenten ningún problema tanto en aspecto exterior como en conectividad.

#### *Soldaduras*

Se observarán las soldaduras tras ser realizadas de modo que presenten un aspecto brillante y limpio, evitando así soldaduras frías.

#### *Sensor de presión*

Se observará el sensor visualmente y se medirá a través de un multímetro y de un manómetro la variación de tensión y de presión que se recogen a la salida de ellos, tras realizar la conversión de unidades de V a kPa si son iguales se dará por adecuado el sensor.

#### *Sensor de temperatura*

Se observará el sensor visualmente y se medirá mediante un multímetro y un termómetro el voltaje y la temperatura actual, siendo el resultado igual en ambos casos.

#### *Regulador de tensión*

Se observará el regulador visualmente y se medirá mediante un multímetro su entrada y su salida, siendo la primera es de +5 V y la segunda de +3,3 V.

### 3.4.3 Prueba de servicio

En primer lugar, se realizará una comprobación de las soldaduras utilizando un multímetro electrónico, para ello se colocará dicho aparato en medida de continuidad. Tras esto, colocaremos las puntas del multímetro entre dos elementos serie (entre dos componentes o entre dos soldaduras) de manera que se escuche el pitido emitido al producirse continuidad, si no se recibiera dicha emisión indicaría que existe un problema de conectividad debido a soldadura fría. Antes de todo este proceso, comprobar que el multímetro funciona adecuadamente pinchando las puntas del polímetro entre sí.

Además de corroborar finalmente que tras la conexión total de todo el sistema las medidas proporcionadas estén dentro de unos márgenes de actuación normal. Si las medidas proporcionadas por el sistema fluctúan o son inadecuadas indicará que algún elemento en cuestión del sistema deberá estar funcionando de manera inadecuada por lo que deberá revisarse. La fluctuación de valores podría indicar una fuga en el sistema.

## 3.4 Condiciones económicas

### 3.5.1 Composición de los precios unitarios

El cálculo de los precios de las distintas unidades de la obra es el resultado de sumar los costes directos, los indirectos, los gastos generales y el beneficio industrial.

Se considerarán costes directos:

a) La mano de obra, con sus pluses, cargas y seguros sociales, que intervienen directamente en la ejecución de la unidad de obra.

b) Los materiales, a los precios resultantes a pie de la obra, que queden integrados en la unidad de que se trate o que sean necesarios para su ejecución.

c) Los equipos y sistemas técnicos de la seguridad e higiene para la prevención y protección de accidentes y enfermedades profesionales.

d) Los gastos de personal, combustible, energía, etc., que tenga lugar por accionamiento o funcionamiento de la maquinaría e instalaciones utilizadas en la ejecución de la unidad de obras.

e) Los gastos de amortización y conservación de la maquinaria, instalaciones, sistemas y equipos anteriormente citados.

Se considerarán costes indirectos:

- Los gastos de instalación de oficinas a pie de obra, comunicaciones, edificación de almacenes, talleres, pabellones temporales para obreros, laboratorios, seguros, etc., los del personal técnico y administrativo adscrito exclusivamente a la obra y los imprevistos. Todos esto gastos, se cifrarán en un porcentaje de los costes directos.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Trabajo final de grado

Grado en ingeniería electrónica industrial y automática

# Sistema de monitorización del prototipo Endoworm 3.0

# 4. Presupuesto

Departamento de ingeniería electrónica

Curso 2015/2016

Autora:
Beatriz Almendros Luis

Tutor:
Carlos Sánchez Díaz

Valencia, septiembre 2016

# 4. Presupuesto

## *Resumen del presupuesto por capitulo*

| Capítulo | Importe (€) |
|---|---|
| 1 Componentes electrónicos | 11.760,30 |
| 2 Material Adicional | 735,38 |
| **Presupuesto de ejecución material (PEM)** | **12.495,68** |

**Asciende el presupuesto de ejecución material a la expresada cantidad de DOCE MIL CUATROCIENTOS NOVENTA Y CINCO EUROS CON SESENTA Y OCHO CÉNTIMOS.**

## Presupuesto parcial nº 1 Componentes electrónicos

| Nº | Ud | Descripción | | Medición | Precio | Importe |
|---|---|---|---|---|---|---|
| 1.1 | | Placa Sensor de Presión | | | | |
| | | | Total : | 3,00 | 1.304,63 | **3.913,89** |
| 1.2 | | Placa Regulador de Tensión | | | | |
| | | | Total : | 1,00 | 3.014,79 | **3.014,79** |
| 1.3 | | Conjunto Sensor de Presión | | | | |
| | | | Total : | 3,00 | 94,35 | **283,05** |
| 1.4 | | Hardware de Control | | | | |
| | | | Total : | 1,00 | 2.271,69 | **2.271,69** |
| 1.5 | | Hardware de Comunicación | | | | |
| | | | Total : | 1,00 | 2.255,90 | **2.255,90** |
| 1.6 | | Sensor desplazamiento | | | | |
| | | | Total : | 1,00 | 20,98 | **20,98** |
| | | **Total Presupuesto parcial nº 1 Componentes electrónicos :** | | | | **11.760,30** |

## Presupuesto parcial nº 2 Material Adicional

| Nº | Ud | Descripción | | Medición | Precio | Importe |
|---|---|---|---|---|---|---|
| 2.1 | | Herramientas | | | | |
| | | | Total : | 1,00 | 735,38 | **735,38** |
| | | **Total Presupuesto parcial nº 2 Material Adicional :** | | | | **735,38** |

## 1 Componentes electrónicos

| Código | Ud | Descripción | | Total |
|---|---|---|---|---|
| **1.1** | | Placa que engloba los materiales para el montaje de la placa destinada a la alimentación y toma de datos del sensor de presión. | | |
| | 1,000 | Placa de Cobre para PCB doble cara | 5,540 € | 5,54 € |
| | 0,050 | Químico para procesado PCB | 40,330 € | 2,02 € |
| | 12,000 | Ampliicador de instrumentación ADR23ARZ | 3,215 € | 38,58 € |
| | 12,000 | Fuente de corriente LM334 | 0,996 € | 11,95 € |
| | 1,000 | Resistencia 137 Ohm | 0,042 € | 0,04 € |
| | 1,000 | Resistencia 10 Ohm | 0,592 € | 0,59 € |
| | 1,000 | Resistencia 1 kOhm | 0,668 € | 0,67 € |
| | 1,000 | Resistencia 330 kOhm | 0,040 € | 0,04 € |
| | 1,000 | Condensador Céramico multicapa 100 nF | 0,024 € | 0,02 € |
| | 1,000 | Condensador Céramico multicapa 470 nF | 0,178 € | 0,18 € |
| | 83,000 | Mano de obra de la Placa del Sensor de Presión | 15,000 € | 1.245,00 € |
| | | | **Precio total por** | **1.304,63 €** |
| **1.2** | | Placa que engloba los materiales para el montaje de la placa destinada al alojamiento de regulador de tensión y el sensor de temperatura. | | |
| | 1,000 | Placa de Cobre para PCB doble cara | 5,540 € | 5,54 € |
| | 0,050 | Químico para procesado PCB | 40,330 € | 2,02 € |
| | 1,000 | Regulador de tensión LP2985-33DBVT | 1,295 € | 1,30 € |
| | 1,000 | Sensor de temperatura LM35DZ | 1,148 € | 1,15 € |
| | 1,000 | Amplificador operacional MAX4250EUK+T | 2,890 € | 2,89 € |
| | 1,000 | Condensador Céramico multicapa 470 nF | 0,178 € | 0,18 € |
| | 1,000 | Condensador Céramico multicapa 100 nF | 0,024 € | 0,02 € |
| | 1,000 | Condensador cerámico multicapa 47 uF | 0,772 € | 0,77 € |
| | 1,000 | Resistencia 220 Ohm | 0,129 € | 0,13 € |
| | 2,000 | Resistencia 1.8 kOhm | 0,332 € | 0,66 € |
| | 1,000 | Resistencia 100 Ohm | 0,129 € | 0,13 € |
| | 200,000 | Mano de obra de la Placa del Regular de Tensión | 15,000 € | 3.000,00 € |
| | | | **Precio total por** | **3.014,79 €** |
| **1.3** | | Abarca tanto el sensor como su encapsulado. | | |
| | 1,000 | Sensor de Presión SCC30ASMT | 31,500 € | 31,50 € |
| | 0,100 | Silicona Silastic | 25,800 € | 2,58 € |
| | 2,000 | Brida pequeña | 0,020 € | 0,04 € |
| | 0,100 | Cinta adhesiva | 2,310 € | 0,23 € |
| | 4,000 | Mano de obra Sensor de presión | 15,000 € | 60,00 € |
| | | | **Precio total por** | **94,35 €** |
| **1.4** | | Componentes necesarios para el control del sistema. | | |
| | 1,000 | Arduino Leonardo | 17,440 € | 17,44 € |
| | 1,000 | Arduino pro mini | 4,250 € | 4,25 € |
| | 150,000 | Mano de obra Hardware de Control | 15,000 € | 2.250,00 € |
| | | | **Precio total por** | **2.271,69 €** |
| **1.5** | | Módulos necesarios para la comunicación inalambrica entre los diferentes puntos de control. | | |
| | 2,000 | Móulo de radiorecuencia | 2,950 € | 5,90 € |
| | 150,000 | Mano de obra Hardware de comunicación | 15,000 € | 2.250,00 € |
| | | | **Precio total por** | **2.255,90 €** |
| **1.6** | | Se tratan de los sesores que permiten la medida de la distancia recorrida por el sistema | | |
| | 2,000 | Acelerómetro ADXL345 | 2,990 € | 5,98 € |
| | 1,000 | Montaje de Acelerómetros | 15,000 € | 15,00 € |
| | | | **Precio total por** | **20,98 €** |

## 2 Material Adicional

| Código | Ud | Descripción | | Total |
|---|---|---|---|---|
| **2.1** | | Herramientas empleadas en el proceso de fabricación | | |
| | 1,000 | Multímetro digital | 189,000 € | 189,00 € |
| | 1,000 | Fuente de alimentación regulable | 298,600 € | 298,60 € |
| | 1,000 | Manometro digital | 177,000 € | 177,00 € |
| | 1,000 | Soldador | 26,870 € | 26,87 € |
| | 1,000 | Estaño | 21,260 € | 21,26 € |
| | 1,000 | Cable plano | 22,650 € | 22,65 € |
| | | | **Precio total por** | **735,38 €** |

## *Cuadro de materiales*

| Nº | Código | Designación | Importe | | |
|---|---|---|---|---|---|
| | | | Precio (€ ) | Cantidad | Total (€) |
| 1 | 0033 | Transceptor 2.4GHz NRF24L01 | 2,950 | 2,000 | **5,90** |
| 2 | 0036 | Acelerómetro 3 ejes ADXL345 – digital | 2,990 | 2,000 | **5,98** |
| 3 | 0072 | Pro MINI ATmega328-5v 16MHz | 4,250 | 1,000 | **4,25** |
| 4 | 14458451 | Brida de poliamida para interior de 2,5 x 100 mm. Color blanco | 0,020 | 6,000 | **0,12** |
| 5 | 185-0109 | Soldador Multicore, 0.71mm, Hilo, Punto de Fusión +183 ? +188°C, 40% Plomo, 60% Estaño, 250g | 21,260 | 1,000 | **21,26** |
| 6 | 201348101 | Silastic 3481 Base + Ag. de Curado | 25,800 | 0,300 | **7,74** |
| 7 | 213-2604 | Resistencia de montaje en superficie de película gruesa TE Connectivity, 330k?, ±1%, 0,1W, Película Gruesa, 0805 | 0,040 | 3,000 | **0,12** |
| 8 | 215-2428 | Resistencia de montaje superficial de alta potencia TE Connectivity, 1k?, ±0.1%, 0,1W, Película Fina, 0805, Serie RN73 | 0,668 | 3,000 | **2,00** |
| 9 | 264-4416 | Condensador cerámico multicapa, Kemet, 0,1?F, ±10%, 50V dc SMD, X7R dieléctrico, Carcasa 0805 | 0,024 | 4,000 | **0,10** |
| 10 | 360-093 | Cable Cinta Plano RS Pro, 10 vías, No Apantallado, Anchura 12,7 mm | 22,650 | 1,000 | **22,65** |
| 11 | 435-529 | Placa de Cobre para PCB, RAW 100 X 160MM D/S, Doble Cara, Base PET Reforzado con Fibra de Vidrio, 160 x 100 x 1.6mm | 5,540 | 4,000 | **22,16** |
| 12 | 437-8931 | Medidor de presión digital Digitron PM-20, ±0,2%, 2, 130mbar, 0bar, Diferencial, IP67, AA, LCD, 180g, 155 x 67 x 40mm | 177,000 | 1,000 | **177,00** |
| 13 | 535-9082 | Fuente de corriente programable, LM334M/NOPB, SOIC, 8 pines, 4.9 x 3.9 x 1.45mm | 0,996 | 36,000 | **35,86** |
| 14 | 551-277 | Químico para Procesado RS Pro 551277, Hexahidrato de Cloruro de Hierro en Cristales | 40,330 | 0,200 | **8,07** |
| 15 | 614-4516 | Resistencia de montaje superficial de alta potencia TE Connectivity, 10?, ±0.1%, 0,1W, Película Fina, 0805, Serie RN73 | 0,592 | 3,000 | **1,78** |
| 16 | 616-1454 | Multímetro digital Fluke FLUKE-115 EUR, 10A ac, 600V ac, De Mano, CAT III 600 V | 189,000 | 1,000 | **189,00** |
| 17 | 648-0834 | Condensador cerámico multicapa, Kemet, 470nF, ±10%, 25 V dc, X7R dieléctrico, Carcasa 0805 | 0,178 | 4,000 | **0,71** |
| 18 | 660-6733 | Regulador de tensión LDO, LP2985-33DBVT, 150mA 3,3 V, SOT-23 5 pines ±1.5%, entrada 2,2? 16 V | 1,295 | 1,000 | **1,30** |
| 19 | 667-0961 | Sensor de Presión Absoluta para Gas, 0 ? 30psi | 31,500 | 3,000 | **94,50** |
| 20 | 668-8599 | Resistencia de montaje superficial de alta potencia Arcol, 1,8k?, ±0.1%, 0,1W, Película de Metal, 0805, Serie ACPP | 0,332 | 2,000 | **0,66** |
| 21 | 708-9319 | Resistencia de montaje en superficie de película fina Panasonic, 137?, ±0.1%, 0.125W, Película de Metal, 0805 | 0,042 | 3,000 | **0,13** |
| 22 | 732-7901P | Amplificador operacional MAX4250EUK+T Ruido bajo, 2,4 ? 5,5 V 3MHz Rail to Rail SOT-23, 5 pines | 2,890 | 1,000 | **2,89** |
| 23 | 761-7324 | Arduino Leonardo con cabezales | 17,440 | 1,000 | **17,44** |
| 24 | 778-2620 | Soldador, Weller, 230V, Eléctrico, 40W, Conector Británico | 26,870 | 1,000 | **26,87** |
| 25 | 788-3029 | Condensador cerámico multicapa, TDK, 47?F, ±20%, 10 V dc SMD, X5R dieléctrico, Carcasa 0805 | 0,772 | 1,000 | **0,77** |
| 26 | 806-5522 | Amplificador de Instrumentación, AD623ARZ, 3 ? 12 V 200?V Offset, 800kHz 90dB CMRR, SOIC, 8-Pines | 3,215 | 36,000 | **115,74** |
| 27 | 812-1770 | Resistencia de montaje superficial de alta potencia Vishay, 100?, ±1%, 0,5W, Película Gruesa, 1206, Serie CRCW | 0,129 | 1,000 | **0,13** |

| Nº | Código | Designación | Importe | | |
|---|---|---|---|---|---|
| | | | Precio (€ ) | Cantidad | Total (€) |
| 28 | 812-1849 | Resistencia de montaje superficial de alta potencia Vishay, 220?, ±1%, 0,5W, Película Gruesa, 1206, Serie CRCW | 0,129 | 1,000 | **0,13** |
| 29 | 922-4836 | Sensor de temperatura LM35DZ/NOPB, encapsulado TO-92 3 pines, alimentación 4? 30 V | 1,148 | 1,000 | **1,15** |
| 30 | FTE30V5A | Fuente de alimentación regulable y cortocircuitable de 0 a 30 V y hasta 5 A Salidas fijas a 5 y 12V CC 500mA Indicadores de tensión e intensidad digitales 3 digitos | 298,600 | 1,000 | **298,60** |
| 31 | MB33 | Cinta adhesiva de alta fijación azul croma 50mmx50m | 2,310 | 0,300 | **0,69** |
| | | | | **Total Materiales** | **1.065,70** |

## *Cuadro de mano de obra*

| Nº | Código | Designación | Importe | | |
|---|---|---|---|---|---|
| | | | Precio (€ ) | Cantidad (Horas) | Total (€) |
| 1 | MOA | Soldaje y colocación en el conjunto del sistema | 15,000 | 1,000 | **15,00** |
| 2 | MOHC | Programación de los microcontroladores y montaje de los mismos | 15,000 | 150,000 | **2.250,00** |
| 3 | MOHCC | Configuración, programación y montaje en el sistema | 15,000 | 150,000 | **2.250,00** |
| 4 | MOPRT | DIseño, fabricación y soldadura de la placa del regulador de tensión | 15,000 | 200,000 | **3.000,00** |
| 5 | MOPSP | Diseño, fabricación y soldadura de la placa del sensor de presión | 15,000 | 249,000 | **3.735,00** |
| 6 | MOSP | Soldaje, encapsulado, prueba de funcionamiento y calibración. | 15,000 | 12,000 | **180,00** |
| | | | | **Total mano de obra** | **11.430,00** |