Universidad Zaragoza

Trabajo Fin de Grado

Sistemas de adquisición de datos basados en hardware y software libre

Autora

Eva Pardos Campanales

Director

Bonifacio Martín del Brío

Departamento de Ingeniería Electrónica y Comunicaciones
Escuela de Ingeniería y Arquitectura

Zaragoza, Septiembre de 2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

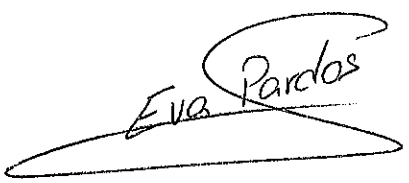
D./Dª. Eva Pardos Campanales,

con nº de DNI 73023416A en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Ingeniería Electrónica y Automática, (Título del Trabajo) SISTEMAS DE ADQUISICIÓN DE DATOS BASADOS EN HARDWARE Y SOFTWARE LIBRE

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de septiembre de 2017



Fdo: _____

Sistemas de adquisición de datos basados en hardware y software libre

RESUMEN

En la actualidad la mayor parte de los sistemas de adquisición de datos se basan en sistemas propietarios. Muchas de las herramientas de mayor difusión pertenecen a la empresa National Instruments, como el muy extendido software LabVIEW y sus tarjetas de adquisición. Además de la calidad de sus productos, estos se distinguen por la facilidad de programación dentro del entorno visual LabVIEW, que permite con tiempos de desarrollo cortos generar entornos muy potentes y profesionales para la visualización de datos y monitorización de un sistema.

En este trabajo se realiza un estudio sobre las herramientas de hardware y software libre actualmente disponibles para el desarrollo de sistemas de adquisición de datos, que puedan resultar una alternativa viable. En primer lugar, se realiza una búsqueda de las diversas alternativas disponibles para, a continuación, centrarse en las más adecuadas y estudiar sus posibilidades. En particular, como tarjeta de adquisición de datos se ha seleccionado la tarjeta Arduino UNO, debido a su difusión y bajo coste, analizándose sus características como tarjeta de adquisición de datos, comparándola con las tarjetas de bajo coste NI-6008 y NI-6009 de National Instruments (prestaciones bajas-medias). Como software de adquisición se han seleccionado dos posibilidades, el entorno libre MyOpenLab y el uso de Processing para la adquisición de datos.

Se han desarrollado librerías de funciones para el manejo de algunos de los sensores más habituales, se ha desarrollado un sistema de adquisición de datos completo a modo de ejemplo del uso de estas herramientas. Uno de los principales problemas de estas herramientas es que la información actualmente disponible es escasa y dispersa, por lo que se ha redactado una guía de instalación y utilización para facilitar su uso.

Pensamos que estas herramientas pueden ser una alternativa viable a los sistemas propietarios, muy especialmente en la docencia, pues facilita que un estudiante pueda trabajar en su propia casa sin coste alguno (la mayor parte de los estudiantes disponen ya de una tarjeta Arduino UNO).

Índice de contenidos

1 INTRODUCCIÓN

1.1 MARCO DE TRABAJO

Este Trabajo Fin de Grado se ha realizado bajo la supervisión de Bonifacio Martín del Brío como director, profesor titular del departamento de Ingeniería Electrónica y Comunicaciones de la Universidad de Zaragoza.

Toda la instrumentación necesaria para la realización de este proyecto ha sido provista por el departamento de Ingeniería Electrónica y Comunicaciones.

1.2 MOTIVACIÓN DEL TFG

Cada vez son más comunes las tareas que requieren adquisición y procesamiento de datos, pero las tarjetas e interfaces comerciales tienes precios muy elevados.

Actualmente, en el departamento de Ingeniería Electrónica y Comunicaciones de la Universidad de Zaragoza, los sistemas de adquisición de datos montados en los laboratorios para uso docente utilizan herramientas de hardware y software propietario pertenecientes a la empresa National Instruments.

El Sistema de Adquisición de Datos utilizado actualmente tiene un precio excesivo, solo el precio de una licencia de software y una tarjeta supera ampliamente los mil euros, por lo que los alumnos solo tienen acceso a realizar prácticas con él en las horas de laboratorio asignadas y el centro no tiene presupuesto para actualizar las licencias anualmente.

La calidad de la enseñanza mejora cuando el alumno tiene acceso a prácticas de laboratorio ya que la actividad experimental es uno de los aspectos clave en el proceso de enseñanza y aprendizaje, tanto por afianzar la fundamentación teórica como por el desarrollo de ciertas habilidades y destrezas para las cuales el trabajo experimental es fundamental.

Este proyecto atiende a la motivación de buscar una alternativa viable a los sistemas propietarios de adquisición de datos actuales ya que el centro no puede permitirse actualizar las licencias del software actual y para facilitar que los estudiantes puedan trabajar en su propia casa sin coste alguno o bajo el mínimo coste posible.

1.3 SISTEMAS DE ADQUISICIÓN DE DATOS

1.3.1 ¿Qué es un Sistema de Adquisición de Datos?

La Adquisición de Datos es el proceso de tomar un conjunto de señales físicas del mundo real, convertirlas en tensiones eléctricas y digitalizarlas para generar datos que puedan ser procesados con un computador.

Como podemos ver en la figura 1, un Sistema de Adquisición de Datos (SAD) se compone de sensores, hardware de medidas y un ordenador con software programable.

Figura 1: Sistema de Adquisición de Datos

Utilizando estos sistemas de medida obtenemos una solución más potente y mejoramos la visualización de los resultados obtenidos.

1.3.2 Hardware de medida.

Existen dos casos típicos de sistemas de adquisición de datos:

- Los basados en **tarjetas de adquisición de datos,** que permiten almacenar y procesar en un computador datos procedentes de sensores, para la medida de cualquier tipo de magnitud física.
- Los basados en **instrumentos de medida**, que conectados a un computador permiten gobernar y automatizar la medida de magnitudes eléctricas desde el computador.

En este proyecto vamos a estudiar SAD basados en tarjetas de adquisición de datos.

Tarjetas de Adquisición de Datos

Una tarjeta de adquisición de datos es un microcontrolador con un sistema de adquisición de datos integrado. Un sistema de adquisición de datos integrado incluye:

- AMUX
- Amplificador de ganancia programable
- S&H
- Conversor A/D
- Interfaz E/S

En la figura 2 podemos ver un diagrama de las partes que constituyen un sistema de adquisición de datos integrado.

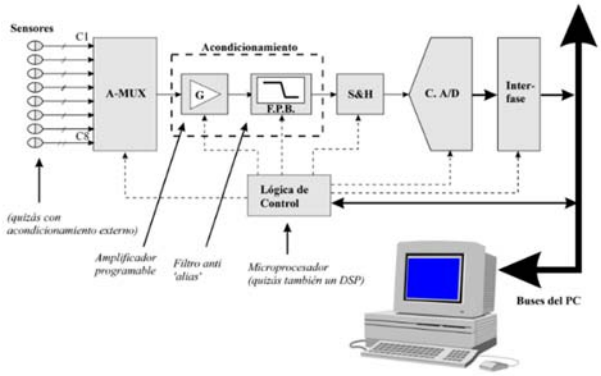


Figura 2: Bloques de una Tarjeta de Adquisición de Datos

Las características principales de una tarjeta de adquisición de datos son:

- Número de canales analógicos
- Tipo de entradas analógicas (referenciadas a tierra o diferenciales)
- Rango de tensión de las entradas analógicas
- Ganancia
- Protección contra sobre-voltajes
- Resolución en bits
- Ritmo de conversión
- Tipo de disparo (por programa, por temporizador, disparo externo)
- Número de entradas y salidas digitales
- Salidas analógicas (número y rango de tensiones)
- Tipo de bus
- Otras: exactitud, linealidad, CMRR, impedancia de entrada, etc.

1.3.3 Software programable

En ambos casos se precisa de un **software de adquisición de datos**, que capture los datos, los almacene y los procese.

1.4 OBJETIVO Y PLANIFICACIÓN DE TFG

En este proyecto estudiamos las tarjetas de adquisición de datos y los softwares de adquisición de datos que utilizamos actualmente y las posibilidades de hardware y software libre que hay disponibles hoy en el mercado, con el fin de proponer una alternativa de SAD libre completo para su posible utilización para uso docente en el departamento de Ingeniería Electrónica y Comunicaciones de la Universidad de Zaragoza.

Para alcanzar este objetivo final se plantearon los siguientes ítems a cumplir:

1. Comenzamos por el estudio de las tarjetas de adquisición de datos utilizadas actualmente y la búsqueda de otras tarjetas de hardware libre de características similares. En este estudio encontramos la tarjeta Arduino UNO, cuya utilización proponemos.

2. Una vez hemos decidido utilizar la tarjeta de Arduino, buscamos alternativas de software libre de adquisición de datos que nos sirvan de alternativa al software utilizado actualmente y que sean compatibles con la tarjeta Arduino UNO. En esta búsqueda hemos encontrado los softwares de MyOpenLab y Processing.

3. Estudio del Software MyOpenLab junto a la tarjeta Arduino Uno. Guía de instalación, exposición de sus características principales y ejemplos de utilización con sensores estándar.

4. Estudio del Software Processing junto a la tarjeta Arduino Uno. Guía de instalación, exposición de sus características principales y un ejemplo de utilización con un sensor estándar.

5. Estudio de una solución mixta: Arduino Uno junto al software de adquisición de datos de Texas Instrument.

6. Desarrollo de un sistema de adquisición de datos completo utilizando la tarjeta Arduino UNO y el software MyOpenLab.

7. Finalmente, se analizan las placas y softwares utilizados y se documenta todo en la memoria del TFG.

Figura 3: Diagrama de Gantt del desarrollo del TFG

2 SISTEMAS DE ADQUISICIÓN DE DATOS PROPIETARIOS Y LIBRES

2.1 TARJETAS DE ADQUISICIÓN DE DATOS

Actualmente, en los laboratorios del departamento, se utiliza la tarjeta NI USB-6009 de la empresa National Instruments.

Nos vamos a centrar en la búsqueda de tarjetas similares, que se adecúen a nuestras necesidades, hardware libre.

Entendemos por Hardware libre, cualquier dispositivo hardware cuyas especificaciones y diagramas esquemáticos son de acceso público.

2.1.1 Búsqueda de Tarjetas de Adquisición de Datos

A continuación, analizamos las características de la tarjeta NI USB-6009 y de su versión más barata, la tarjeta NI USB-6008, y buscamos alternativas a estas de hardware libre.

NI USB-6009

Se trata de un dispositivo DAQ multifunción de bajo coste desarrollado, como ya hemos dicho, por la empresa National Instruments. Nos brinda una funcionalidad básica para aplicaciones como registro de datos simple, medidas portátiles y experimentos académicos de laboratorio.

La empresa National Instruments ha desarrollado un Software para el manejo de estas tarjetas llamado LabView.

El precio de esta tarjeta es de unos 200€ aproximadamente.

Características técnicas de la tarjeta:

- Microcontrolador NI-DAQmx
- 14 bits.
- 12 pines Digitales
- 2 salidas analógicas
- 4 entradas analógica
- Reloj de 24MHz de velocidad
- Conexión USB

En la figura 4 podemos ver el aspecto de la tarjeta de adquisición de datos NI USB-6009.

Figura 4: Tarjeta de Adquisición de Datos NI USB-6009

NI USB-6008

Se trata de una versión algo inferior en prestaciones a la utilizada en las prácticas de laboratorio.

El precio de esta tarjeta es algo inferior al de la tarjeta anterior.

Características técnicas de la tarjeta:

- Microcontrolador NI-DAQmx
- 12 bits.
- 12 pines Digitales
- 4 entradas analógica
- 2 salidas analógicas
- Reloj de 24MHz de velocidad
- Conexión USB

En la figura 5 podemos ver el aspecto de la tarjeta de adquisición de datos NI USB-6008.

Figura 5: Tarjeta de Adquisición de Datos NI USB-6008

Arduino UNO R3

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo (IDE) para su programación. La principal característica del software y del lenguaje de programación de Arduino es su sencillez y su facilidad de uso.

Un factor importante en el éxito de Arduino es la comunidad que apoya todo este desarrollo, comparte conocimiento, elabora librerías para facilitar su uso y publica sus proyectos para que puedan ser replicados, mejorados o sirvan de base para otros proyectos.

Dentro de la familia Arduino, nos hemos decantado por la tarjeta Arduino UNO R3 por ser la más utilizada y más documentada de todas y, por tanto, más utilizada entre los alumnos de la universidad.

El precio de esta tarjeta es de unos 20€ aproximadamente.

Características técnicas de la tarjeta:

- Microcontrolador ATmega328
- 10 bits.
- Voltaje de entrada 7-12V. (recomendado)
- 14 pines Digitales (6 salidas PWM)
- 6 entradas analógica
- 32kB de memoria Flash
- Reloj de 16MHz de velocidad
- Conexión USB

En la figura 6 podemos ver el aspecto de la tarjeta de adquisición de datos Arduino UNO R3.

Figura 6: Tarjeta de Adquisición de Datos Arduino UNO R3

BeagleBone Black

Se trata de otra tarjeta de bajo coste desarrollada por Beagleboard.org, Texas Instruments. Es una plataforma que corre bajo un sistema operativo Linux.

Cuenta con un entorno de programación web basado en Java con instrucciones muy parecidas a las utilizadas por el IDE de Arduino.

Uno de los principales usos de esta plataforma es el procesamiento de visión por computador ya que cuenta con compatibilidad de librerías con OpenCV, cámaras USB y numerosos accesorios para comunicación y pantallas.

El precio de esta tarjeta es de unos 55€ aproximadamente.

Características técnicas de la tarjeta:

- Procesador Sitara AM3359AZCZ100
- 12 bits
- 6 entradas analógica
- 512MB de RAM
- 2GB de memoria Flash
- Reloj de 24.576MHz de velocidad
- Conexión USB y puerto Ethernet

En la figura 7 podemos ver el aspecto de la tarjeta de adquisición de datos BeagleBone Black.

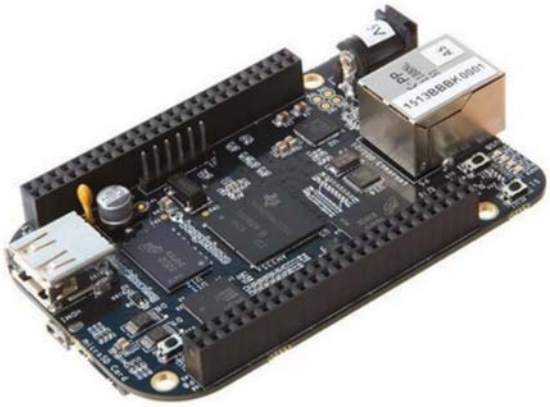


Figura 7: Tarjeta de Adquisición de Datos BeagleBone Black

Raspberry Pi 3 Modelo B

Se trata de otra tarjeta de bajo coste desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza en las escuelas.

Puede utilizar lenguajes de programación de alto nivel como Python, C++ y Java, aunque promueve principalmente Python. El sistema operativo que utiliza es Raspbian, se trata de una versión de Debian adaptada a Raspberry Pi.

El precio de esta tarjeta es de unos 36€ aproximadamente.

Características técnicas de la tarjeta:

- Procesador quad-core ARMv8
- 32 bits
- Todos los pines son digitales, ninguno analógico.
- 1GB de d¡memoria SDRAM (compartida con la CPU)
- Conexión USB, puerto Ethernet, Wifi y BlueTooth

En la figura 8 podemos ver el aspecto de la tarjeta de adquisición de datos Raspberry Pi 3 Modelo B.

Figura 8: Tarjeta de Adquisición de Datos Raspberry Pi 3 Modelo B

WiNode 4

WiNode 4 pertecece a Nanode, una evolución de Arduino que permite conectarse a internet. Sus tarjetas de bajo coste permiten experimentar con el internet de las cosas.

WiNode 4 ha sido diseñado actuar como plataforma inicial para una serie sensores, actuadores y controladores inalámbricos. Está dirigido a aplicaciones que requieren motor, relé o conmutación de alta potencia, así como aplicaciones de registro de datos, guardándolos en una microSD.

El precio de esta tarjeta es de unos 33€ aproximadamente.

Características técnicas de la tarjeta:

- Microcontrolador ATmega328P
- 10 bits
- Voltaje de entrada 10-36V.
- 6 pines Digitales (4 salidas PWM)
- 6 entradas analógica
- Tarjeta microSD
- Reloj en tiempo real
- Conexión USB y módulo inalámbrico RFM12B

En la figura 9 podemos ver el aspecto de la tarjeta de adquisición de datos WiNode 4.

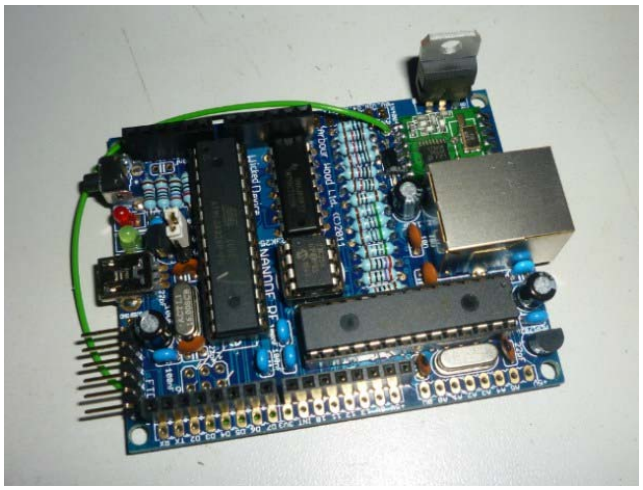


Figura 9: Tarjeta de Adquisición de Datos WiNode 4

Waspmote

Waspmote es un dispositivo desarrollado por la empresa Libelium. Se trata de un equipo autónomo e inalábrico cuyo propósito es el establecimiento de redes sensoriales.

Libelium ha desarrollado una interfaz de programación de aplicaciones (API), el Waspmote IDE.

El precio de esta tarjeta es de unos 200€ aproximadamente.

Características técnicas de la tarjeta:

- Microcontrolador Atmega1281
- 10 bits
- 8 pines Digitales (1 salidas PWM)
- 7 entradas analógica
- 128kB de memoria Flash
- Reloj RTC de 32 KHz de velocidad
- Conexión USB

En la figura 10 podemos ver el aspecto de la tarjeta de adquisición de datos WaspMote.

Figura 10: Tarjeta de Adquisición de Datos Waspmote

2.1.2 Comparativa de las Tarjetas de Adquisición de Datos

	NI USB-6009	NI USB-6008	Arduino UNO R3	BeagleBone Black	Raspberry Pi 3 Modelo B	WiNode 4	Waspmote
Procesador	Micro. NI-DAQmx	Micro. NI-DAQmx	Micro. ATmega328	Proc. Sitara AM3359AZCZ100	Proc. quad-core ARMv8	Micro. ATmega328P	Micro. Atmega1281
Resolución en bits	14 bits	12 bits	10 bits	12 bits	32 bits	10 bits	10 bits
Nº de entradas analógicas	4 entradas 2 salidas	4 entradas 2 salidas	6 entradas	6 entradas	0 entradas	6 entradas	8 entradas
Nº de pines digitales	12 pines	12 pines	14 pines (6 PWM)		Todos los pines son digitales	6 pines (4 PWM)	7 pines (1 PWM)
Memoria			32 KB de memoria Flash	2GB de memoria Flash	1GB de memoria SDRAM (Copartida con la CPU)	En tarjeta microSD	128kB de memoria Flash
Reloj	24MHz	24MHz	16MHz	24.576MHz		RTC	RTC de 32KHz
Conexiones	USB	USB	USB	USB y Ethernet	USB, Ethernet, Wifi y Bluetooth	USB y Módulo inalámbrico RFM12B	USB
Precio	200€	200€	20€	55€	36€	33€	200€

Tabla 1: Comparativa de las Tarjetas de Adquisición de Datos propuestas

2.1.3 ¿Por qué Arduino UNO?

Tras analizar la tabla anterior (tabla 1) podemos ver que en relación prestaciones/precio nos quedamos con las tarjetas WiNode 4, Raspberry Pi 3Modelo B y Arduino UNO R3. Las tres son asequibles por los estudiantes, permiten desarrollar la clase de aplicaciones que hacemos en los laboratorios y, en todos los casos, son hardware libre.

Uno de los factores determinantes para la elección de la tarjeta Arduino es la gran comunidad que apoya este proyecto y que, día a día, publica nuevo contenido, divulga y responde las dudas que cualquier usuario puede tener. Por este motivo es la más utilizada y documentada de todas.

Otro factor importante es su bajo coste, gracias al cual, muchos de los alumnos de la universidad ya disponen de una tarjeta Arduino en su casa.

Se ha incluido el Anexo A para una mejor descripción de las características generales de la tarjeta Arduino UNO.

2.2 SOFTWARE DE ADQUISICIÓN DE DATOS

Actualmente, en los laboratorios del departamento, se utiliza el software LabView de la empresa National Instruments.

Nos vamos a centrar en la búsqueda de softwares similares, que se adecúen a nuestras necesidades, software libre.

Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.

2.2.1 Búsqueda de Software de Adquisición de Datos

Hemos encontrado las siguientes herramientas de programación visual que se pueden utilizar junto a la tarjeta Arduino.

LabView

LabView es una plataforma y entorno de desarrollo para diseñar sistemas con un lenguaje de programación visual gráfico, desarrollado por la empresa National Instruments. Se trata de un software propietario cuya licencia tiene un alto coste, excesivo para que el centro actualice las licencias anualmente y para cualquier estudiante.

Su principal característica es la facilidad de uso para hacer programas relativamente complejos. El lema tradicional de LabView es "*La potencia está en el software*". Se trata de una herramienta de gran capacidad, muy extendida tanto en el mundo laboral como en el académico.

LabView consigue combinarse con todo tipo de hardware y software, tanto del propio fabricante como de otros.

Los programas en LabView se llaman Instrumentos Virtuales (VI), ya que su apariencia y operación imitan instrumentos físicos. Sigue la filosofía de programación descendente (top-down), pudiendo realizarse subprogramas (SubVI) que incorporaríamos dentro de un VI. LabView contiene un conjunto completo de herramientas para la adquisición, análisis, visualización y almacenamiento de datos.

Como podemos ver en la figura 11, se construye una interfaz de usuario, o panel frontal, con controles y visualizadores, y se agrega código en el diagrama de bloques utilizando VIs y estructuras para controlar los objetos del panel frontal.

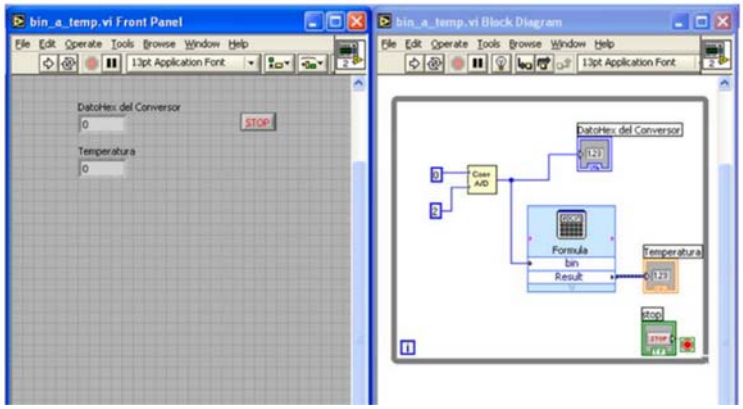


Figura 11: Vista del panel frontal y el diagrama de bloques de LabView

MyOpenLab

Se trata de un software de libre distribución bajo licencia GNU de código abierto, escrito en Java, y por ello multiplataforma, orientado a la realización de aplicaciones de modelado y simulación.

En este programa existe la posibilidad de conexión a través de los puertos del ordenador a diversos tipos de hardware, entre ellos Arduino.

Igual que LabView, sigue la filosofía de programación descendente. Los programas realizados en MyOpenLab se denominan Máaquinas Virtuales (VM), y como en LabView, están compuestos de subprogramas llamados subVM.

Se trata de una herramienta que cuenta con una amplia biblioteca de bloques funcionales para permitir realizar modelos a partir de ellos.

En la figura 12 podemos ver el aspecto de los paneles de circuito y frontal del programa MyOpenLab.

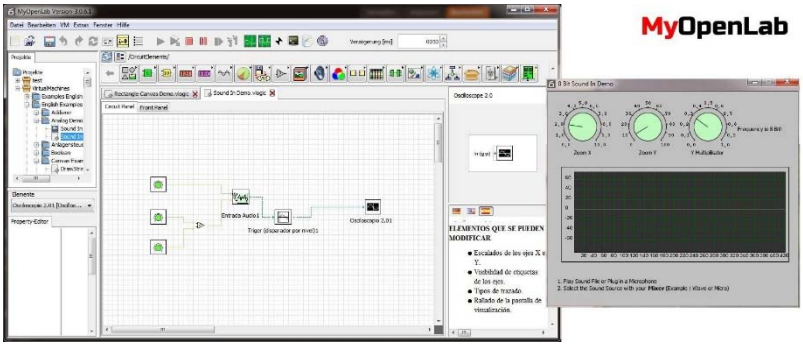


Figura 12: Vista del panel circuito y el panel frontal de MyOpenLab

Processing

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java que se distribuye bajo la licencia GNU.

Ha sido desarrollado por miembros del MIT Media Lab con el objetivo de actuar como herramienta para que artistas, diseñadores visuales y gente ajena al lenguaje de programación, aprendieran a través de una muestra gráfica y visual.

Su interfaz es muy parecida a la de Arduino, incluyendo una ventana visual como complemento al IDE.

En la figura 13 podemos ver las ventanas de programación y visual de este software.

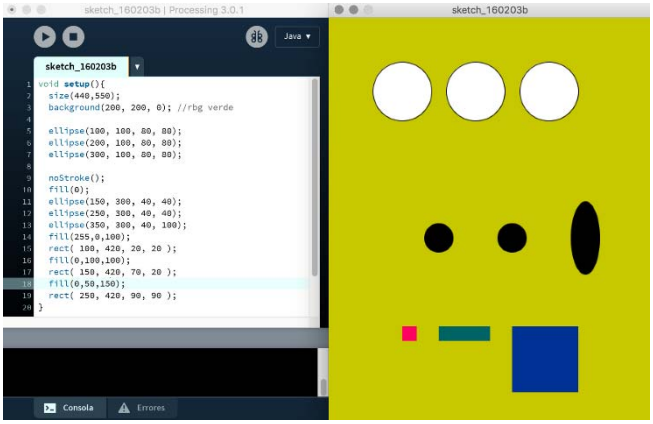


Figura 13: Vista de mimimi de Processing

Scratch for Arduino (S4A)

Scratch es un lenguaje de programación visual desarrollado por el MIT Media Lab. Como en los casos anteriores, se trata de software libre.

Aunque también se puede programar con Scrath, S4A está pensado específicamente para trabajar con Arduino. Se trata de una modificación que añade nuevos bloques para trabajar con sensores y actuadores conectados a Arduino.

Su objetivo es proporcionar una interfaz para programadores de Arduino con funcionalidades como la interacción de varias placas a través de entornos de usuario.

Los objetos de la librería Arduino ofrecen bloques para funcionalidades básicas, como lecturas y escrituras tanto analógicas como digitales, y para funcionalidades de más alto nivel, como bloques para controlar servomotores.

Mediante el protocolo de la PicoBoard, S4A interactúa con Arduino cada 75ms.

Una de sus características más importantes es que se puede crear un proyecto utilizando tantas placas como puertos USB haya disponibles. Funciona con las versiones Duemilanove, Diecimila y UNO.

En la figura 14 podemos ver el aspecto del programa S4A conectado a una tarjeta Arduino.

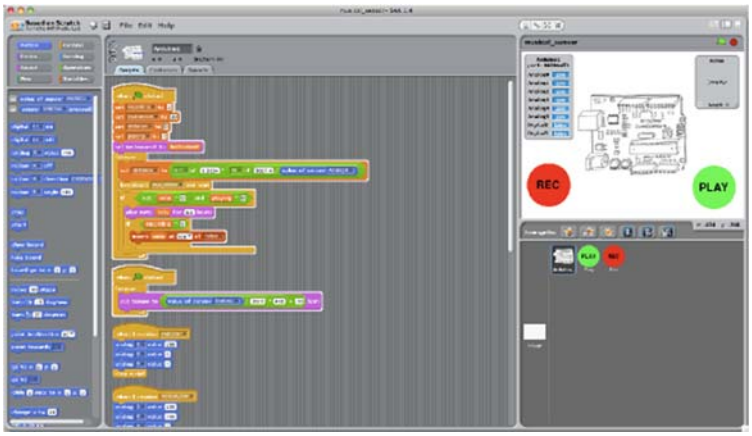


Figura 14: Vista del programa S4A conectado a una tarjeta Arduino

Existen otros softwares como ArduBlock, BlocklyDuino, MBLOCK o MiniBlock que no hemos contemplado por ser de orientación infantil.

2.2.2 Comparativa de los Softwares de Adquisición de Datos

Tras analizar los softwares anteriores, podemos ver que en todos los casos se trata de Softwares gratuitos, salvo el utilizado actualmente, LabView, y que todos nos permiten desarrollar la clase de aplicaciones que hacemos en los laboratorios del departamento.

Las posibilidades gráficas de MyOpenLab junto a su gran potencia de cálculo y procesado de datos le hacen el mejor candidato para la elaboración de proyectos tanto en el aula o el laboratorio como en casa.

Uniendo Arduino y MyOpenLab obtenemos un Sistema de Adquisición de Datos completo, barato y potente para realizar proyectos.

3 SOFTWARE MYOPENLAB

MyOpenLab es una herramienta libre cuyo autor, Carmelo Salafia, la deja disponible bajo licencia GNU. Ha sido traducida y documentada al español por José Manuel Ruiz Gutierrez, profesor del IES Fco. García Pavón en Tomelloso, Ciudad Real.

Como hemos dicho con anterioridad, MyOpenLab es un entorno orientado a la simulación y modelado de sistemas físicos con un amplio campo de aplicaciones.

Está desarrollada en lenguaje Java y por ello resulta portable a distintas plataformas.

Se trata de una herramienta que cuenta con una amplia biblioteca de bloques funcionales que permiten realizar modelos a partir de la conexión de varios de ellos.

Dentro de las herramientas de Instrumentación Virtual, esta es un ejemplo de sencillez y potencia.

3.1 INSTALACIÓN DE MYOPENLAB

Al ser una aplicación desarrollada en Java, lo primero que necesitamos es descargar la última versión. Para este TFG he descargado la versión 8 Update 144 del siguiente enlace:

https://www.java.com/es/download/

Una vez descargada e instalada la última versión de Java, procedemos a descargar el programa MyOpenLab del siguiente enlace:

https://myopenlab.de/download_myopenlab_now.html

En este caso, he descargado la versión 3.9.1 (Windows 64Bit, Java 32 Bit) ya que la versión 3.10.0 tiene un error que impide abrir proyectos guardados con anterioridad.

Una vez completada la descarga, descomprimimos el fichero. MyOpenLab no requiere instalación previa, por lo que podemos iniciarlo en los archivos del tipo “archivo por lotes de Windows”.

3.2 CARACTERÍSTICAS DEL SOFTWARE

3.2.1 Descripción del entorno

Para la creación de proyectos o VM, MyOpenLab sigue la filosofía de programación descendente (top-down), dividiendo el VM en subVM, que pueden encapsular a su vez otros subVM, y otros bloques funcionales.

Los proyectos constan de dos partes:

El circuito, donde se diseña el algoritmo de simulación mediante el enlace de bloques funcionales y subVM. Este conjunto de funciones se editará en la pestaña de "**Panel Circuito**". Podemos observar una imagen de este panel en la figura 15.

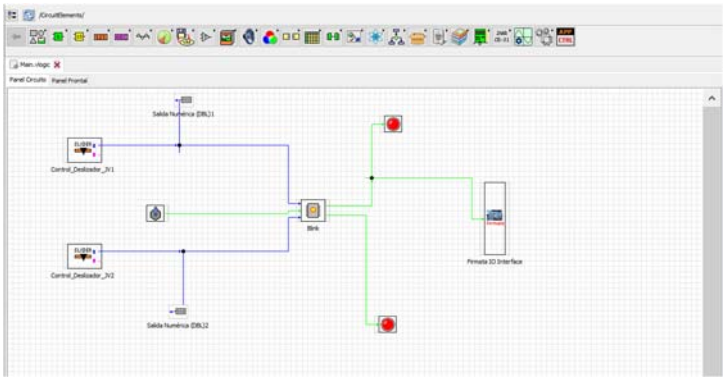


Figura 15: Panel circuito de MyOpenLab

La parte de panel frontal de Visualización, situada en la pestaña de "**Panel Frontal**", donde se colocan los objetos de visualización gráfica asociados al circuito que permiten su visualización durante el modo de visualización. Podemos observar una imagen de este panel en la figura 16.

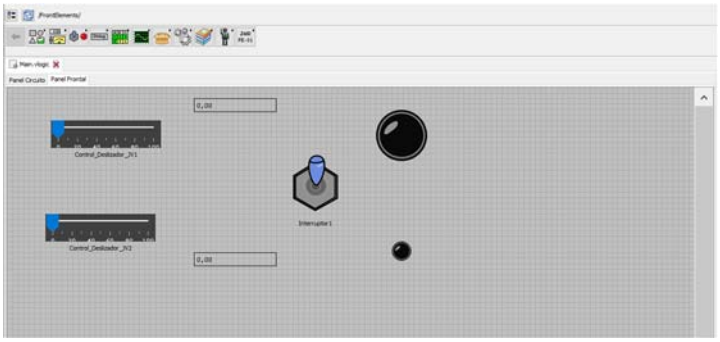


Figura 16: Panel frontal de MyOpenLab

3.2.2 Librerías de elementos

Como hemos dicho anteriormente, MyOpenLab cuenta con una amplia librería de bloques funcionales en ambos paneles. Cada bloque dispone de una ventana editor de propiedades, donde se introducen los parámetros necesarios para su utilización.

Panel de circuito

En la figura 17 podemos ver los distintos bloques de librerías del panel de circuito.

Figura 17: Bloques de librerías del panel de circuito

Vamos a analizar, de izquierda a derecha, los distintos bloques de librerías:

- Decoración: Como podemos ver en la figura 18, en esta librería encontramos elementos decorativos.

Figura 18: Librería de decoración del panel de circuito

- Operaciones digitales: Como podemos ver en la figura 19, en esta librería tenemos operadores lógicos, monoestables, biestables…

Figura 19: Librería de operadores digitales del panel de circuito

- Operaciones con bits: Como podemos ver en la figura 20, en esta librería tenemos operadores lógicos básicos y bloques para desplazar bits.

Figura 20: Librería de operaciones con bits del panel de circuito

- Numéricos: Como podemos ver en la figura 21, en esta librería tenemos operaciones numéricas básicas, conversores de tipo de dato y operaciones matemáticas más avanzadas.

Figura 21: Librería de numéricos del panel de circuito

- Tratamiento de caracteres: Como podemos ver en la figura 22, en esta librería tenemos bloques para unir strings, conversores de formato y bloques para mostrar un string por pantalla.

Figura 22: Librería de tratamiento de caracteres del panel de circuito

- Elementos analógicos: Como podemos ver en la figura 23, en esta librería tenemos bloques de tratamiento de señales analógicas, como un generador de frecuencia.

Figura 23: Librería de elementos analógicos del panel de circuito

- Utilidades: Como podemos ver en la figura 24, en esta librería tenemos bloques contadores, retardos, temporizadores…

Figura 24: Librería de utilidades del panel de circuito

- Ficheros E/S: Como podemos ver en la figura 25, en esta librería tenemos bloques para abrir, leer y guardar un fichero de texto.

Figura 25: Librería de ficheros E/S del panel de circuito

- Comparaciones: Como podemos ver en la figura 26, en esta librería tenemos bloques que comparan elementos numéricos.

Figura 26: Librería de comparaciones del panel de circuito

- Imágenes: Como podemos ver en la figura 27, en esta librería tenemos diferentes bloques de tratamiento de imágenes.

Figura 27: Librería de imágenes del panel de circuito

- Sonido: Como podemos ver en la figura 28, en esta librería tenemos bloques para ejecutar sonidos.

Figura 28: Librería de sonido del panel de circuito

- Color: Como podemos ver en la figura 29, en esta librería tenemos bloques de tratamiento de color en RGB.

Figura 29: Librería de color del panel de circuito

- Pines: Como podemos ver en la figura 30, en esta librería tenemos bloques con pines de entrada y salida para crear subVM.

Figura 30: Librería de pines del panel de circuito

- Vectores y matrices: Como podemos ver en la figura 31, en esta librería tenemos los bloques necesarios para operar con vectores y matrices.

Figura 31: Librería de vectores y matrices del panel de circuito

- Grupo de elementos: Como podemos ver en la figura 32, en esta librería tenemos un bloque que permite agrupar un conjunto de variables u objetos para entregarlos en su salida de forma empaquetada.

Figura 32: Librería de grupos de elementos del panel de circuito

- Canvas: Como podemos ver en la figura 33, en esta librería tenemos bloques para crear y manipular objetos canvas para animaciones.

Figura 33: Librería de canvas del panel de circuito

- Física: Como podemos ver en la figura 34, en esta librería tenemos bloques que recogen modelos físicos de movimientos con el movimiento uniformemente acelerado.

Figura 34: Librería de física del panel de circuito

- Diagramas de flujo: Como podemos ver en la figura 35, en esta librería tenemos todos los bloques necesarios para programar mediante un diagrama de flujo.

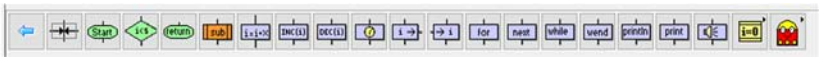


Figura 35: Librería de diagramas de flujo del panel de circuito

- Extras: Como podemos ver en la figura 36, en esta librería tenemos diferentes tipos de controles, como un semáforo o el control con un ratón.

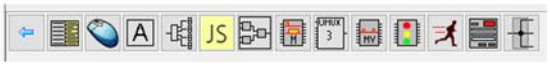


Figura 36: Librería de extras del panel de circuito

- Sockets: Como podemos ver en la figura 37, en esta librería tenemos bloques para conexiones a servidor y cliente.

Figura 37: Librería de sockets del panel de circuito

- Interfaces: Como podemos ver en la figura 38, en esta librería tenemos bloques de conexión con diferentes tarjetas, como la tarjeta Arduino UNO o la Raspberry Pi.

Figura 38: Librería de interfaces del panel de circuito

- JMR-CE-01: Como podemos ver en la figura 39, en esta librería tenemos registradores de varios canales, contador BCD, semáforo, monoestables y biestables…

Figura 39: Librería de JMR-CE-01 del panel de circuito

- Bloque con generador de frecuencia.

- Automatización: Como podemos ver en la figura 40, en esta librería tenemos bloques tales como un regulador PID.

Figura 40: Librería de automatización del panel de circuito

- System_Control: Como podemos ver en la figura 41, en esta librería tenemos bloques para enviar comandos de Linux a Windows y para cerrar máquinas virtuales de java.

Figura 41: Librería de system_control del panel de circuito

Panel frontal

En la figura 18 podemos ver los distintos bloques de librerías del panel frontal.

Figura 42: Bloques de librerías del panel frontal

Vamos a analizar, de izquierda a derecha, los distintos bloques:

- Decoración: Como podemos ver en la figura 43, en esta librería encontramos elementos decorativos.

Figura 43: Librería de decoración del panel frontal

- Numéricos: Como podemos ver en la figura 44, en esta librería tenemos todo tipo de bloques para introducir o representar un dato numérico.

Figura 44: Librería de numéricos del panel frontal

- Booleanos: Como podemos ver en la figura 45, en esta librería tenemos diversos bloques para controlar o mostrar el valor de un dato de tipo booleano.

Figura 45: Librería de booleanos del panel frontal

- Texto: Como podemos ver en la figura 46, en esta librería tenemos bloques para introducir o ver el valor de salida de un string o un fichero de texto.

Figura 46: Librería de texto del panel frontal

- Vectores y matrices: Como podemos ver en la figura 47, en esta librería tenemos bloques para mostrar valores en una lista o tabla.

Figura 47: Librería de vectores y matrices del panel frontal

- Gráficos: Como podemos ver en la figura 48, en esta librería tenemos bloques para mostrar distintas gráficas.

Figura 48: Librería de gráficos del panel frontal

- Extras: Como podemos ver en la figura 49, en esta librería tenemos bloques para mostrar imágenes, datos en displays de 7 segmentos, teclados…

Figura 49: Librería de extras del panel frontal

- Automatización: Como podemos ver en la figura 50, en esta librería tenemos bloques que simulan cintas transportadoras, cilindros neumáticos, barreras…

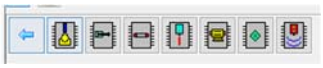


Figura 50: Librería de automatización del panel frontal

- Definido por el usuario: Como podemos ver en la figura 51, en esta librería tenemos bloques para simular cosas diferentes como sensores, teclados numéricos…

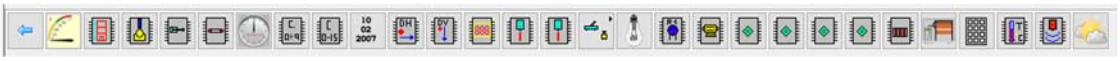


Figura 51: Librería definida por el usuario del panel frontal

- Robótica: Como podemos ver en la figura 52, en esta librería tenemos bloques que simulan robots en 2D y 3D.

Figura 52: Librería de robótica del panel frontal

- JMR-FE-01: Como podemos ver en la figura 53, en esta librería tenemos desde bloques que simulan movimientos de barreras hasta imágenes de tarjetas Arduino.

Figura 53: Librería de JMR-FE-01 del panel frontal

Ventana de ayuda de un bloque

Cada bloque dispone de una ventana de ayuda que nos muestra una explicación del funcionamiento del bloque y de sus entradas y salidas.

Como ejemplo vamos a ver la ventada de ayuda del bloque *Contador Generador de Impulsos* en la figura 54.

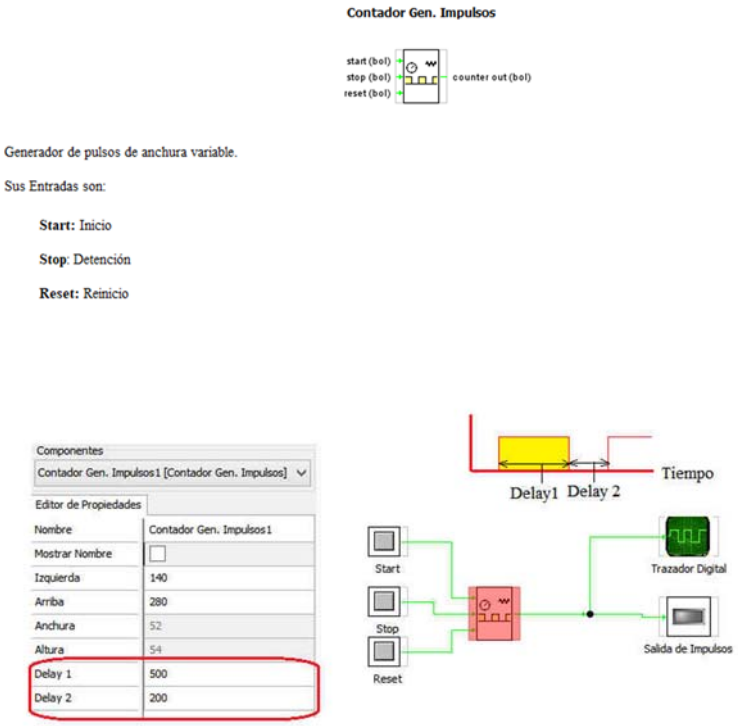


Figura 54: Ventana de ayuda del bloque Contador Generador de Impulsos

Al tratarse de un software de licencia GNU, esta ventana es editable, aunque para editarla necesitamos un editor de html.

Código de un bloque

Pulsando sobre un bloque el botón derecho del ratón, encontramos varias opciones, entre ellas se encuentra *editar código*. Esta opción nos abre una ventana que nos permite visualizar y modificar el código que se ejecuta en ese bloque.

También es editable la imagen que representa ese bloque en la opción *propiedades elemento.*

3.2.3 Arduino en MyOpenLab

La comunicación entre MyOpenLab y Arduino UNO se realiza gracias al protocolo Firmata.

Antes de comenzar, debemos cargar el firmware en la tarjeta Arduino UNO con la ayuda del IDE de Arduino. Esto se encuentra detallado en el Anexo B.

MyOpenLab se conecta a través de la librería Firmata, la podemos encontrar en el panel de circuito, dentro del bloque de librerías Interfaces.

Hemos elegido *Arduino Firmata IO Interface v1.0* porque es la que nos permite configurar los pines conforme a nuestras necesidades, y activar solo los que vayamos a utilizar.

En la figura 55 podemos ver el bloque correspondiente a la firmata elegida.

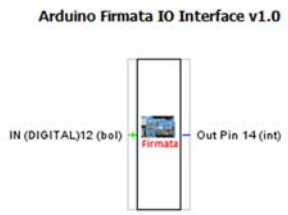


Figura 55: Firmata Arduino en MyOpenLab

Para empezar, abrimos la ventana del editor de propiedades para configurar el puerto de conexiones, COM Port, donde hemos conectado la tarjeta de Arduino, y la velocidad de comunicación con el puerto, Baud, por defecto 57600.

A continuación, configuramos los pines de entrada y salida que utiliza nuestra aplicación.

La numeración de los pines se realiza con valores que van desde el 2 hasta el 19. En la tabla 2 podemos ver una descripción detallada de la configuración de los pines de Arduino UNO.

Pin en MyOpenLab	Tipo de entrada/salida	Pin en Arduino
2	Pin digital	2
3	Pin digital	3
4	Pin digital	4
5	Pin digital	5
6	Pin digital	6
7	Pin digital	7
8	Pin digital	8
9	Pin digital	9
10	Pin digital	10
11	Pin digital	11
12	Pin digital	12
13	Pin digital	13
14	Entrada analógica	A0
15	Entrada analógica	A1
16	Entrada analógica	A2
17	Entrada analógica	A3
18	Entrada analógica	A4
19	Entrada analógica	A5

Tabla 2: Relación de pines Arduino – Firmata MyOpenLab

Como podemos ver en la figura 56, la posibilidad de configuración de cada pin se muestra al hacer clic sobre el pin, las distintas configuraciones son:

- Entrada digital
- Salida digital
- Salida PWM
- Salida Servo
- Entrada analógica

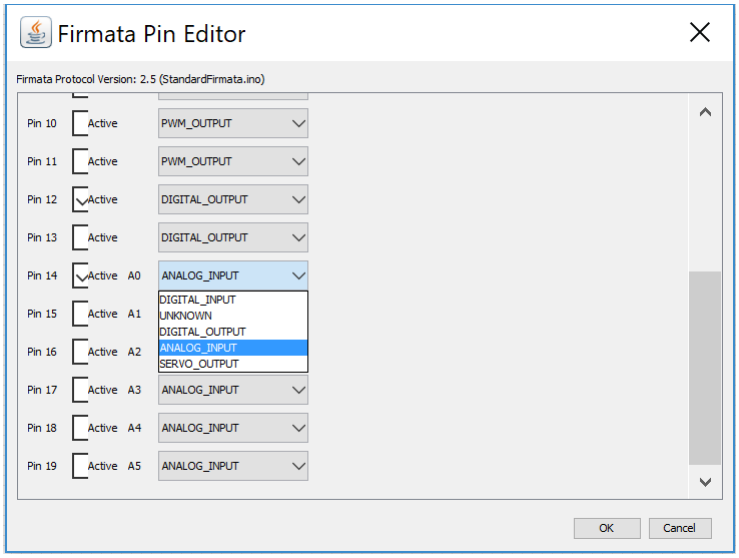


Figura 56: Posibilidades de configuración de los pines de Arduino en MyOpenLab

3.3 EJEMPLOS DE USO CON SENSORES HABITUALES

En este apartado vamos a exponer un conjunto de ejemplos para comprender las posibilidades de la conjunción Arduino + MyOpenLab.

Todos los sensores han sido programados previamente utilizando el IDE de Arduino, para su posterior programación en MyOpenLab. El Anexo C recoge todos estos programas.

Los siguientes anexos recogen los datasheets de los sensores:

- Anexo D: sensor de temperatura.
- Anexo E: sensor de humedad.
- Anexo F: sensor de presión atmosférica.
- Anexo G: sensor de ultrasionidos.
- Anexo H: sensor de luz ambiente.

3.3.1 Sensor de temperatura

Montaje del hardware

En esta primera aplicación, vamos a utilizar un sensor de temperatura LM35 de National Semiconductors.

Se trata de un sensor integrado cuya salida ofrece una tensión proporcional a la temperatura a razón de 10mV por cada Cº. Este sensor opera entre 0 y 100ºC, con una precisión de 0.5ºC a 25ºC.

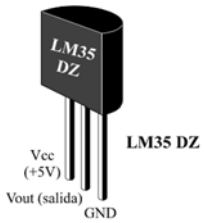


Figura 57: Sensor LM35 de National Semiconductors

Como podemos ver en la figura 57, el circuito integrado cuenta únicamente con tres patillas. Conectaremos la izquierda al pin de salida de 5V, la derecha al pin GND y la central a la entrada analógica A0 de la tarjeta Arduino.

Utilizando dos resistencias de potencia de 33Ω, una a cada lado del sensor, lograremos aumentar la temperatura gradualmente. Incluimos también un diodo LED, conectado al pin digital 13 de la tarjeta Arduino, para programarlo como advertencia al superar una temperatura elevada.

Podemos ver como ha quedado el montaje final en la figura 58.

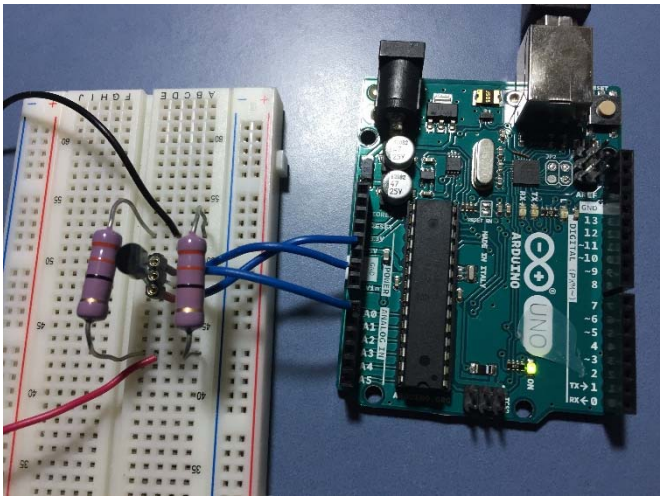


Figura 58: Montaje completo con el sensor de temperatura

Programación en MyOpenLab

Comenzamos en el **panel de circuito**. En este panel debemos colocar la Firmata y hacer la conversión a temperatura.

Colocamos el bloque de la Firmata de Arduino, situado en el bloque de librerías *Interfaces*, configurando el puerto COM y 57600 baudios de velocidad.

En la configuración de los pines, debemos configurar el pin 13 como salida digital y la entrada A0 como entrada analógica. En la tabla 2 del apartado 3.2.3 podemos ver que se corresponden con los pines 13 y 14, respectivamente, de la Firmata.

Con todo configurado, podemos ver que al bloque de la Firmata le han aparecido las patillas necesarias para conectar con los pines 13 y A0 de Arduino.

A continuación, tenemos que transformar la entrada A0 en temperatura, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Teniendo en cuenta que conectamos el sensor a una fuente de 5V, que la resolución de la placa es de 10 bits y que el sensor proporciona 10mV por cada ºC, vamos a calcular la conversión de la tensión que tenemos en A0 a ºC.

$$V_{LSB} = \frac{5v}{2^n - 1} = \frac{5000}{2^{10} - 1} = \frac{5000}{1023}\, mV \qquad \rightarrow \qquad T^{\underline{a}} = \frac{V * V_{LSB}}{10} = V * \frac{500}{1023}$$

Introduciendo esta fórmula bloques de la librería *Numéricos* ya tenemos la temperatura en ºC, por lo que pasamos al **panel frontal** para incluir visualizadores y actuadores.

En el bloque de librerías *Automatismos,* dentro de *JME-FE-01*, encontramos el icono de un termómetro de 0 a 100ºC, lo colocamos en este panel junto a un indicador *LED* situado en el bloque de librerías *Booleanos* y un potenciómetro que encontramos en la librería *Numéricos,* configurándolo con valores entre 0 y 100.

Con estos indicadores colocados, este panel queda como se muestra en la figura 59.

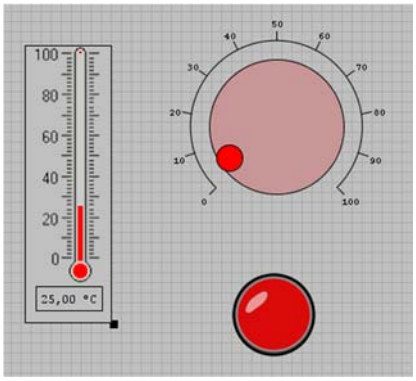


Figura 59: Sensor de temperatura. Panel frontal

Volvemos al **panel de circuito** y vemos que los bloques que hemos añadido al panel frontal han aparecido en este panel.

Queremos mostrar la temperatura en el termómetro, por lo que convertimos el tipo de dato a dbl con un bloque del bloque de librerías de *Numéricos* y lo cableamos.

Con el potenciómetro controlaremos la temperatura de alarma, y cuando esta se supere, se encenderá el led. Para ello, debemos cablear la salida del potenciómetro con la temperatura a un bloque *Mayor o igual que,* que se encuentra en el bloque *Comparadores*, y su salida al LED.

Para terminar, conectamos el LED con la entrada a la Firmata correspondiente al pin 13 de Arduino.

Con los bloques conectados, el panel de circuito queda como se muestra en la figura 60.

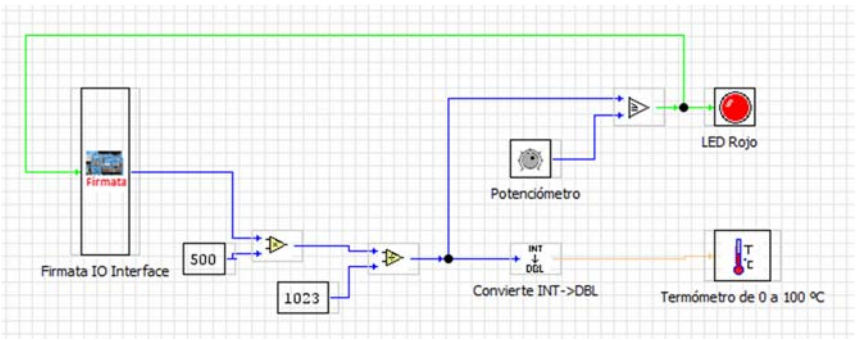


Figura 60 Sensor de temperatura. Panel de circuito

Ahora que hemos terminado de montar ambos paneles, ponemos el panel frontal en **modo ejecución** y aparecerá una ventana como la que se muestra en las figuras 61 y 62.

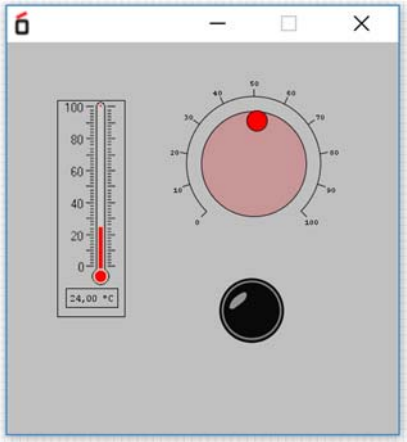


Figura 61: Sensor de temperatura. Modo ejecución 1

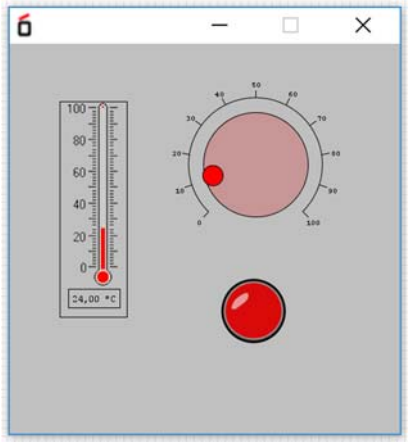


Figura 62: Sensor de temperatura. Modo ejecución 2

3.3.2 Sensor de luz ambiente

Montaje del hardware

Vamos a utilizar una fotorresistencia LDR como sensor de luz ambiente, modelo ST-76.

Se trata de una resistencia cuya impedancia varía en función de la luminosidad, obteniendo su máximo valor en la oscuridad.

Figura 63: Sensor de luz ambiente LDR ST-76

Como podemos ver en la figura 63, el sensor cuenta únicamente con dos patillas. Conectaremos una al pin de salida de 5V y la otra en serie con una resistencia de 10kΩ, que irá al pin GND de la tarjeta Arduino, formando así un divisor resistivo.

Con este sensor, hemos conseguido leer valores máximos de 2.4MΩ y mínimos de 150Ω. Vamos a comprobar si se trata de un día soleado, si está ligeramente nublado, si está nublado del todo o si es de noche en función del valor que la impedancia adquiera.

Los rangos de valores que toma la impedancia son:

- LDR < 500Ω -> Día soleado
- 500Ω < LDR < 1kΩ -> Día ligeramente nublado
- 1kΩ < LDR < 5kΩ -> Día nublado
- LDR > 5kΩ -> Noche

Podemos ver como ha quedado el montaje final en la figura 64.

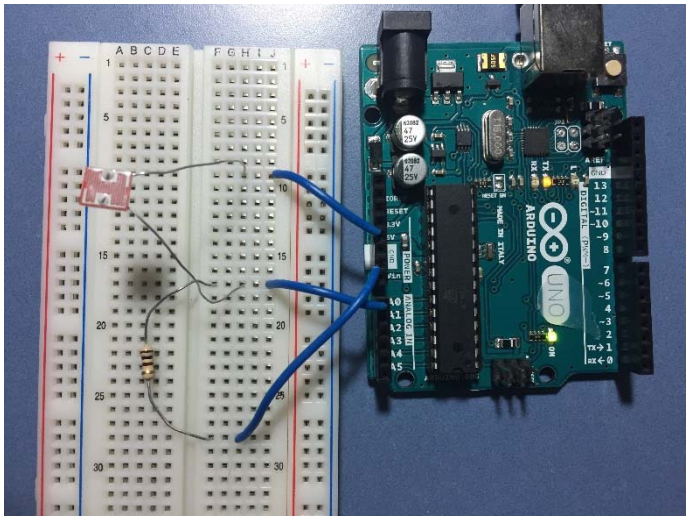


Figura 64: Montaje completo con el sensor de temperatura

Programación en MyOpenLab

Comenzamos en el **panel de circuito**. En este panel debemos colocar la Firmata y hacer la conversión de la señal de entrada a la impedancia del fotorresistor.

Colocamos el bloque de la Firmata de Arduino, situado en el bloque de librerías *Interfaces*, configurando el puerto COM y 57600 baudios de velocidad.

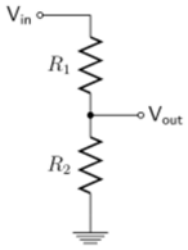
En la configuración de los pines, debemos configurar la entrada A0 como entrada analógica. En la tabla 2 del apartado 3.2.3 podemos ver que se corresponde con el 14 de la Firmata.

Con todo configurado, podemos ver que al bloque de la Firmata le ha aparecido la patilla correspondiente para conectar el pin A0 de Arduino.

A continuación, tenemos que transformar la señal de la entrada A0 en impedancia, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Teniendo en cuenta que conectamos el sensor a una fuente de 5V, que la resolución de la placa es de 10 bits y que el fotorresistor se encuentra en un divisor resistivo, vamos a calcular la conversión de la tensión que tenemos en A0 en impedancia.

$$V_{LSB} = \frac{5v}{2^n - 1} = \frac{5}{2^{10} - 1} = \frac{5}{1023}\ V \qquad ; \qquad V_{OUT} = V_{LSB} * V_{A_0}$$



Como vemos en la figura 65:

$$V_{OUT} = V_{IN} * \frac{R_2}{R_1 + R_2} = 5v * \frac{10k\Omega}{R_{LDR} + 10k\Omega}$$

Figura 65: Divisor resistivo

Por lo tanto: $$R_{LDR} = \frac{1023}{V_{A_0}} * 10k - 10k$$

Introduciendo esta fórmula bloques de la librería *Numéricos* obtenemos la impedancia del fotorresistor, por lo que pasamos al **panel frontal** para incluir visualizadores.

Para visualizar el valor que adquiere la impedancia colocamos un bloque de *Salida Numérica* en el bloque de librerías *Numéricos.* Añadimos un bloque *Mostrar Imagen*, que se encuentra en la librería *Extras* para visualizar una imagen diferente en función del valor de la impedancia.

Con estos indicadores colocados, este panel queda como se muestra en la figura 66.

Figura 66: Sensor de luz ambiente. Panel frontal

Volvemos al **panel de circuito** y vemos que los bloques que hemos añadido al panel frontal han aparecido en este panel.

Queremos mostrar la impedancia en el bloque de *salida numérica*, por lo que cableamos desde la conversión a impedancia hasta el bloque de salida.

Lo primero que debemos hacer es comparar la impedancia mostrada en el bloque de *salida numérica* con los rangos de luz descritos anteriormente. Para esto utilizamos bloques *mayor o*

igual que de la librería *Comparaciones* y bloques *int* de la librería *Numéricos,* con valores de 500Ω, 1kΩ y 5kΩ, de los que saldrán las salidas booleanas de cada rango de impedancias.

Una vez obtenidas las salidas booleanas, incluimos bloques *Pregunta* en cascada, de la librería comparaciones, para determinar qué imagen queremos mostrar. Obtendremos las cuatro imágenes (una para cada rango) utilizando bloques *Imagen Estática* de la librería *Imágenes*, indicando en las propiedades de este bloque la ruta en la que tenemos guardada cada imagen.

Por último, conectamos el final de la cascada de preguntas al bloque del panel frontal en el que queremos mostrar la imagen.

Con todo esto, el panel de circuito queda como se muestra en la figura 67.

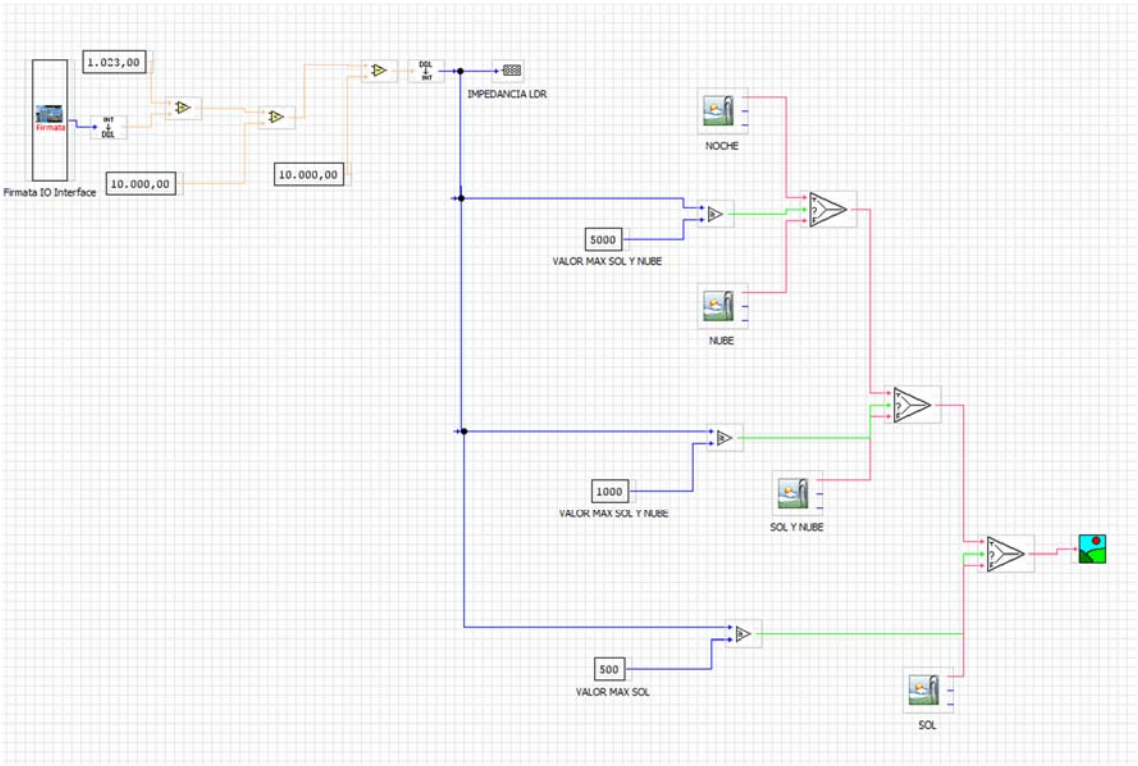


Figura 67: Sensor de luz ambiente. Panel de circuito

Ahora que hemos terminado de montar ambos paneles, ponemos el panel frontal en **modo ejecución** y aparecerá una ventana como la que se muestra en las figuras 68, 69, 70 y 71.

Figura 68: Sensor de luz ambiente. Modo ejecución 1

Figura 69: Sensor de luz ambiente. Modo ejecución 2

Figura 70: Sensor de luz ambiente. Modo ejecución 3

Figura 71: Sensor de luz ambiente. Modo ejecución 4

3.3.3 Sensor de presión atmosférica

Montaje del hardware

Vamos a utilizar un sensor de presión MPX4115AP de Motorola.

Se trata de un sensor integrado cuya salida ofrece una tensión que varía en función de la presión atmosférica.

Figura 72: Sensor de presión MPX4115AP

Como podemos ver en la figura 72, el sensor cuenta con seis patillas, pero solo utilizaremos tres. Conectamos el primer pin, marcado con una muesca, a la entrada A0, el segundo pin a GND y el tercero a 5V.

El datasheet del sensor muestra que se puede calcular la presión, en kPA, a partir de la siguiente fórmula.

$$V_{OUT} = V_{IN} * (0.009 * P - 0.095)$$

Podemos ver como ha quedado el montaje final en la figura 73.

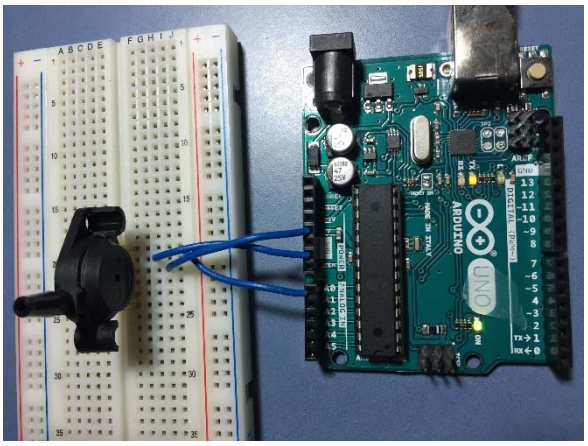


Figura 73: Montaje completo con el sensor de presión

Programación en MyOpenLab

Comenzamos en el **panel de circuito**. En este panel debemos colocar la Firmata y hacer la conversión de la señal de entrada a la presión en hPa.

Colocamos el bloque de la Firmata de Arduino, situado en el bloque de librerías *Interfaces*, configurando el puerto COM y 57600 baudios de velocidad.

En la configuración de los pines, debemos configurar la entrada A0 como entrada analógica. En la tabla 2 del apartado 3.2.3 podemos ver que se corresponde con el 14 de la Firmata.

Con todo configurado, podemos ver que al bloque de la Firmata le ha aparecido la patilla correspondiente para conectar el pin A0 de Arduino.

A continuación, tenemos que transformar la señal de la entrada A0 en presión, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Teniendo en cuenta que conectamos el sensor a una fuente de 5V, que la resolución de la placa es de 10 bits y la fórmula obtenida del datasheet del sensor que relaciona la tensión con la presión, vamos a calcular la conversión de la tensión que tenemos en A0 en presión atmosférica.

$$V_{LSB} = \frac{5v}{2^n-1} = \frac{5}{2^{10}-1} = \frac{5}{1023}\ V \qquad ; \qquad V_{OUT} = V_{LSB} * V_{A_0}$$

$$V_{OUT} = V_{IN} * (0{,}009 * P - 0{,}095) = 5v * (0{,}009 * P - 0{,}095) \rightarrow P = \frac{V_{OUT}+0{,}475}{0{,}045}\ kPa = \frac{V_{OUT}+0{,}475}{0{,}45}\ hPa$$

Por lo tanto: $$P = \frac{V_{A_0} * \frac{5}{1023} + 0{,}475}{0{,}45}\ hPa$$

Introduciendo esta fórmula bloques de la librería *Numéricos* obtenemos la presión atmosférica, por lo que pasamos al **panel frontal** para incluir su visualizador.

Para visualizar el valor que adquiere la presión atmosférica colocamos un bloque de *Salida Numérica* en el bloque de librerías *Numéricos.* El panel quedará como se muestra en la figura 74.

Figura 74: Sensor de presión atmosférica. Panel frontal

Volvemos al **panel de circuito** y vemos el bloque que hemos añadido al panel frontal han aparecido en este panel.

Queremos mostrar el valor de la presión atmosférica en el bloque de *salida numérica*, por lo que cableamos desde la conversión a hPa hasta el bloque de salida.

Con todo esto, el panel de circuito queda como se muestra en la figura 75.

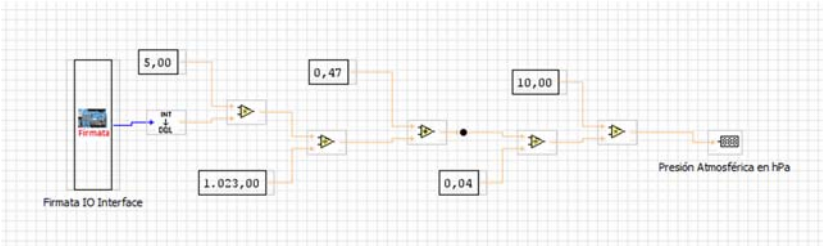


Figura 75: Sensor de presión atmosférica. Panel de circuito

Ahora que hemos terminado de montar ambos paneles, ponemos el panel frontal en **modo ejecución** y aparecerá una ventana como la que se muestra en la figura 76.

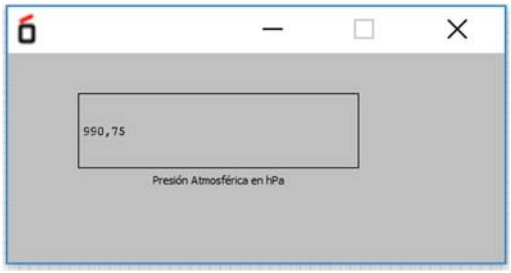


Figura 76: Sensor de luz ambiente. Modo ejecución

3.3.4 Sensor de humedad

Montaje del hardware

Vamos a utilizar un sensor de humedad resistivo HS15P, de Amphenol Advanced Sensors.

Se trata de una resistencia cuya impedancia varía en función de la humedad, obteniendo su valor mínimo teórico con un 100% de humedad en el ambiente de 800Ω y su valor máximo teórico con un 20% de humedad en el ambiente de 8MΩ, a 25ºC.

Figura 77: Sensor de humedad HS15P

Como podemos ver en la figura 77, el sensor cuenta únicamente con dos patillas. Conectaremos una al pin de salida de 5V y la otra en serie con una resistencia de 100Ω, que irá al pin GND de la tarjeta Arduino, formando así un divisor resistivo.

Con este sensor, hemos conseguido leer valores máximos de 1.8MΩ y mínimos de 1k6Ω.

Utilizando la gráfica que relaciona humedad con impedancia del datasheet del sensor, dividiremos la impedancia en 8 rangos.

Los rangos de valores que toma la impedancia son:

- R > 10MΩ -> < 20%
- 10MΩ < R < 1MΩ -> 20% - 30%
- 1MΩ < R < 200kΩ -> 30% - 40%
- 200kΩ < R < 70kΩ -> 40% - 50%
- 70kΩ < R < 25kΩ -> 50% - 60%
- 25kΩ < R < 9kΩ -> 60% - 70%
- 9kΩ < R < 2k5Ω -> 70% - 80%
- 2k5Ω < R < 2kΩ -> 80% - 90%
- 2kΩ < R -> 90% - 1000%

Podemos ver como ha quedado el montaje final en la figura 78.

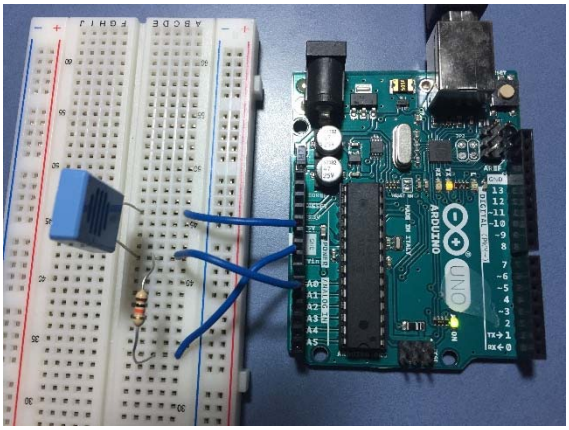


Figura 78: Montaje completo con el sensor de humedad atmosférica

Programación en MyOpenLab

Comenzamos en el **panel de circuito**. En este panel debemos colocar la Firmata y hacer la conversión de la señal de entrada a la impedancia del sensor.

Colocamos el bloque de la Firmata de Arduino, situado en el bloque de librerías *Interfaces*, configurando el puerto COM y 57600 baudios de velocidad.

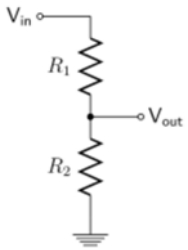
En la configuración de los pines, debemos configurar la entrada A0 como entrada analógica. En la tabla 2 del apartado 3.2.3 podemos ver que se corresponde con el 14 de la Firmata.

Con todo configurado, podemos ver que al bloque de la Firmata le ha aparecido la patilla correspondiente para conectar el pin A0 de Arduino.

A continuación, tenemos que transformar la señal de la entrada A0 en impedancia, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Teniendo en cuenta que conectamos el sensor a una fuente de 5V, que la resolución de la placa es de 10 bits y que el sensor se encuentra en un divisor resistivo, vamos a calcular la conversión de la tensión que tenemos en A0 en impedancia.

$$V_{LSB} = \frac{5v}{2^n - 1} = \frac{5}{2^{10} - 1} = \frac{5}{1023}\ V \qquad ; \qquad V_{OUT} = V_{LSB} * V_{A_0}$$



Como vemos en la figura 79:

$$V_{OUT} = V_{IN} * \frac{R_2}{R_1 + R_2} = 5v * \frac{10k\Omega}{R_{sensor} + 10k\Omega}$$

Figura 79: Divisor resistivo

Por lo tanto: $$R_{sensor} = \frac{1023}{V_{A_0}} * 10k - 10k$$

Introduciendo esta fórmula bloques de la librería *Numéricos* obtenemos la impedancia del sensor, por lo que pasamos al **panel frontal** para incluir visualizadores.

Para visualizar el valor que adquiere la impedancia colocamos un bloque de *Salida Numérica* en el bloque de librerías *Numéricos.* Añadimos un bloque *Salida*, que se encuentra en la librería *Texto* para visualizar el rango de humedad al que pertenece la impedancia calculada.

Con estos indicadores colocados, este panel queda como se muestra en la figura 80.

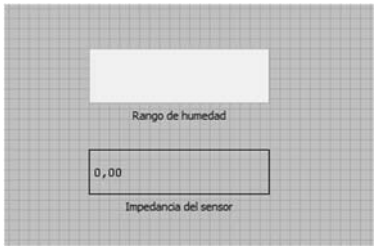


Figura 80: Sensor de humedad. Panel frontal

Volvemos al **panel de circuito** y vemos que los bloques que hemos añadido al panel frontal han aparecido en este panel.

Queremos mostrar la impedancia en el bloque de *salida numérica*, por lo que cableamos desde la conversión a impedancia hasta el bloque de salida.

Lo primero que debemos hacer es comparar la impedancia mostrada en el bloque de *salida numérica* con los rangos de humedad descritos anteriormente. Para esto utilizamos bloques *mayor o igual que* y *menor que* de la librería *Comparadores,* bloques *int* de la librería *Numéricos,* con valores de 10MΩ, 1MΩ, 200kΩ, 70kΩ , 25kΩ, 9kΩ, 2k5Ω y 2kΩ, bloques *Pregunta, también* de la librería *Comparaciones,* y bloques de strings, uno indicando cada rango.

Colocando los bloques *Pregunta* en cascada, a las salidas de los bloques *Comparadores*, obtenemos en la salida el string que nos indica el rango de humedad.

Con todo esto, el panel de circuito queda como se muestra en la figura 81.

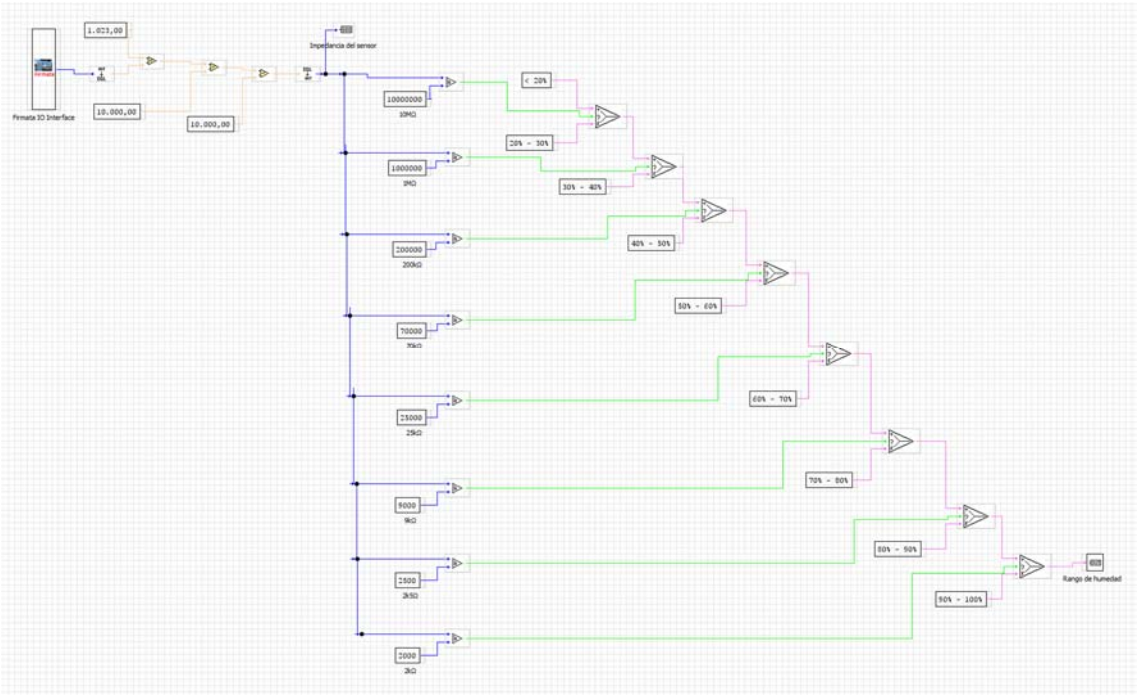


Figura 81: Sensor de humedad. Panel de circuito

Ahora que hemos terminado de montar ambos paneles, ponemos el panel frontal en **modo ejecución** y aparecerá una ventana como la que se muestra en la figura 82.

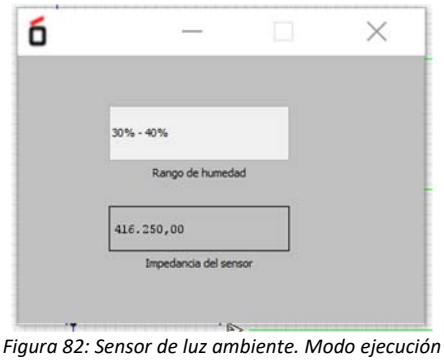


Figura 82: Sensor de luz ambiente. Modo ejecución

3.3.5 Sensor ultrasonido

Montaje del hardware

Vamos a utilizar un sensor de ultrasonidos HC-SR04.

Se trata de un sensor integrado que envía un pulso de alta frecuencia, rebota contra un objeto y se ve reflejado hacia el sensor, que dispone de un micrófono adecuado para esa frecuencia. Midiendo el tiempo entre pulsos estimamos la qué distancia se encuentra el objeto.

El rango de medición teórico es de entre 2cm y 400cm, con una precisión de 3mm.

Como podemos ver en la figura 83, el circuito integrado cuenta con cuatro patillas. Conectaremos la izquierda al pin de salida de 5V, la derecha al pin GND y las centrales, Trig (disparo) y Echo (eco) de izquierda a derecha, a los pines 13 y 12 respectivamente.

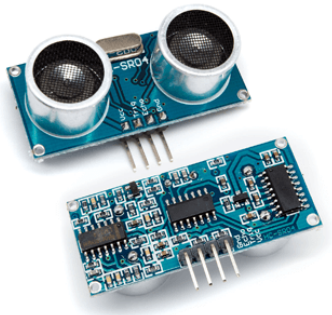


Figura 83: Sensor ultrasonidos HC-SR04

Para calcular la distancia al objeto enviamos pulsos, desde Trig, de 10µs de duración cada 60ms. Echo recibe el pulso tras un tiempo, a partir del cual calculamos la distancia al objeto.

Sabiendo que *distancia = velocidad * tiempo* y que la velocidad del sonido es 343m/s y que tiene que recorrer la distancia al objeto dos veces.

$$Velocidad\ del\ sonido = 343\frac{m}{s} * 100\frac{cm}{m} * \frac{1}{10^6}\frac{s}{\mu s} = 343 * 10^{-4}\frac{cm}{\mu s}$$

$$Distancia(cm) = \frac{Tiempo(\mu s) * 343 * 10^{-4}\frac{cm}{\mu s}}{2}$$

Podemos ver como ha quedado el montaje final en la figura 84.

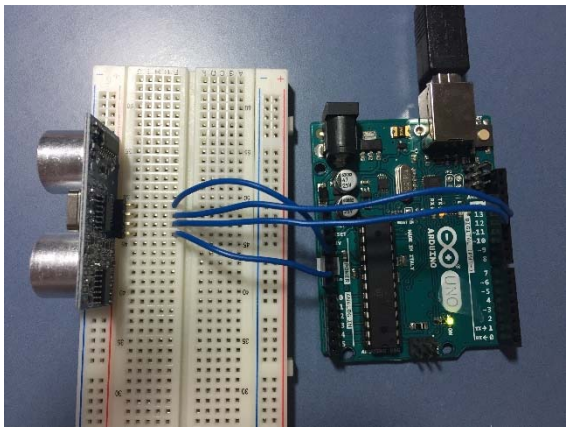


Figura 84: Montaje completo con el sensor de ultrasonidos

Programación en MyOpenLab

Al tratar de programar este sensor, nos hemos encontrado con un impedimento. Necesitamos enviar pulsos de 10µs de duración cada 60ms y la resolución temporal del software es de 1ms. Por este motivo tenemos que descartar el desarrollo de aplicaciones que requieran tiempos inferiores a 1ms.

4 USO DE PROCESSING PARA ADQUISICIÓN DE DATOS

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java que se distribuye bajo la licencia GNU.

Como hemos dicho con anterioridad, ha sido desarrollado por miembros del MIT Media Lab con el objetivo de actuar como herramienta para que artistas, diseñadores visuales y gente ajena al lenguaje de programación, aprendieran a través de una muestra gráfica y visual.

Al estar desarrollada en lenguaje Java, resulta portable a distintas plataformas.

Su interfaz es muy parecida a la de Arduino, ya que este está basado en Processing, incluyendo una ventana visual como complemento al IDE.

4.1 INSTALACIÓN DE PROCESSING

Al ser una aplicación desarrollada en Java, lo primero que necesitamos es descargar la última versión. Para este TFG he descargado la versión 8 Update 144 del siguiente enlace:

https://www.java.com/es/download/

Una vez descargada e instalada la última versión de Java, procedemos a descargar el programa Processing del siguiente enlace:

https://processing.org/download/

En este caso, he descargado la última versión, la 3.3.6.

Una vez completada la descarga, descomprimimos carpeta y, como no requiere instalación previa, podemos iniciarlo mediante el archivo de aplicación.

4.2 CARACTERÍSTICAS DEL SOFTWARE

4.2.1 Estructura y funcionamiento

La estructura del programa es muy similar a la de Arduino, se compone de tres partes: inicialización de variables, setup y bucle infinito. En este caso, al tratarse de un software orientado a diseñadores visuales, el bucle es para dibujar en la pantalla de visualización.

Dispone de una larga serie de funciones muy útiles orientadas al diseño de la pantalla de visualización. Algunas de las más importantes son las siguientes:

- size(x,y) Crea la ventana de visualización
- background(0,0,0) Determina el color de fondo de la ventana en modo RGB

- point(x,y) Crea un punto en la posición x,y del gráfico.
- line(x1,y1,x2,y2) Crea una línea que va del punto 1 al 2.
- triangle(x1,y1,x2,y2,x3,y3) Dibuja un triángulo con vértices 1, 2 y 3.
- fill(0,0,0) Llena del color indicado en modo RGB la figura.
- text("*texto*",x,y) Escribe un texto en la posición x,y.

4.2.2 Conexión con Arduino

Arduino y Processing se comunican a través de un puerto serie. El funcionamiento es el siguiente:

Cargamos en la placa Arduino un código desarrollado en su propio IDE que envíe y reciba las señales que queramos e imprima por pantalla los datos que queramos que reciba Processing.

En el software de Processing comenzamos importando la librería Serial, que se encarga de abrir la comunicación con Arduino con la siguiente línea de código:

```
import processing.serial.*;
```

En el bucle *setup()* del código de programa de Processing debemos incluir las siguientes líneas de código para buscar el puerto serie al que está conectada la placa, abrirla y marcar como separador entre datos un salto de línea.

```
println((Object[])Serial.list());   // Visualizo los puertos serie disponibles en la consola
dataPort = new Serial(this, Serial.list()[0], 9600);  // Abro el primer puerto
dataPort.bufferUntil('\n');  //Separa con un salto de línea
```

Leo el dato que quiero mostrar en cada repetición del bucle con la siguiente línea de código:

```
String inString = dataPort.readStringUntil('\n');  //lee hasta el salto de línea
```

4.3 EJEMPLO DE USO CON UN SENSOR DE ULTRASONIDOS

Vamos a utilizar el sensor de ultrasonidos del apartado 3.3.5. El montaje del hardware será idéntico.

4.3.1 Arduino

Hemos implementado un código en el IDE de Arduino que envía un pulso de ultrasonidos de 10µs de duración desde el disparador y recibe un pulso del eco del sonido. Utilizando la velocidad del sonido y el tiempo transcurrido entre disparo y eco calcula la distancia hasta el objeto en el que ha rebotado el sonido. Para este cálculo hemos utilizado la siguiente ecuación:

$$Distancia(cm) = \frac{Tiempo(\mu s) * 343 * 10^{-4} \frac{cm}{\mu s}}{2}$$

Tras calcular la distancia al objeto imprime en pantalla su valor. Este bucle se repite cada 60ms.

El código implementado puede verse en el Anexo C.

4.3.2 Processing

En el código implementado en Processing distinguimos 3 partes:

- Parte inicial donde importamos la librería que nos permite conectar con Arduino e inicializamos variables.
- Bucle setup donde creamos la ventana en la que vamos a visualizar los resultados y abrimos el puerto serial.
- Bucle Draw donde leemos el dato que nos envía arduino y lo dibujamos en la ventana que hemos creado para graficar los resultados.

Código implementado en Processing

```
import processing.serial.*;  //Importamos la librería Serial
Serial dataPort;     //Nombre del puerto serie

int Xpos;          // Posición horizontal de la gráfica
int Ypos;          // Posición vertical de la gráfica
int lastXpos;       // Coordenada X anterior
int lastYpos;       // Coordenada Y anterior

//---------------- setup ----------------------------------------------------
void setup () {
 // Ventana de 1300 x 600 pixels
 size(1300, 600);
 background(0);        // Background
 Xpos = width+1;       // Posición horizontal inicial

 println((Object[])Serial.list());   // Visualizo los puertos serie disponibles en la consola

 dataPort = new Serial(this, Serial.list()[0], 9600);  // Abro el primer puerto

 dataPort.bufferUntil('\n');  //Separa con un salto de línea

}
//---------------- draw -----------------------------------------------------
void draw () {
 String inString = dataPort.readStringUntil('\n');  //lee hasta el salto de línea

 if (inString != null) {
  inString = trim(inString);    //elimina espacios en blanco

  float inData = float(inString); // convierte el dato
  inData = map(inData, 0, 200, 0, height); // escala el dato en la pantalla
  Ypos= int(height-inData);
```

```
  if (Xpos > width) {   // si se sale de la pantalla
    background(0);     // borra la pantalla
    Xpos = 0;          // inicializa el trazo
    lastXpos= Xpos;    // vuelve x a su posicion inicial
    lastYpos= Ypos;    // vuelve y a su posicion inicial
  }

  stroke(127,34,255);    //color de la traza
  strokeWeight(4);       // tamaño de la traza
  line(lastXpos, lastYpos, Xpos, Ypos);

  lastXpos= Xpos;
  lastYpos= Ypos;

  Xpos++;              // incrementa x
}
}
```

4.3.3 Resultados visuales

En la figura 85 podemos ver una muestra de los resultados obtenidos con el sensor ultrasonidos graficados en la ventana creada en Processing.

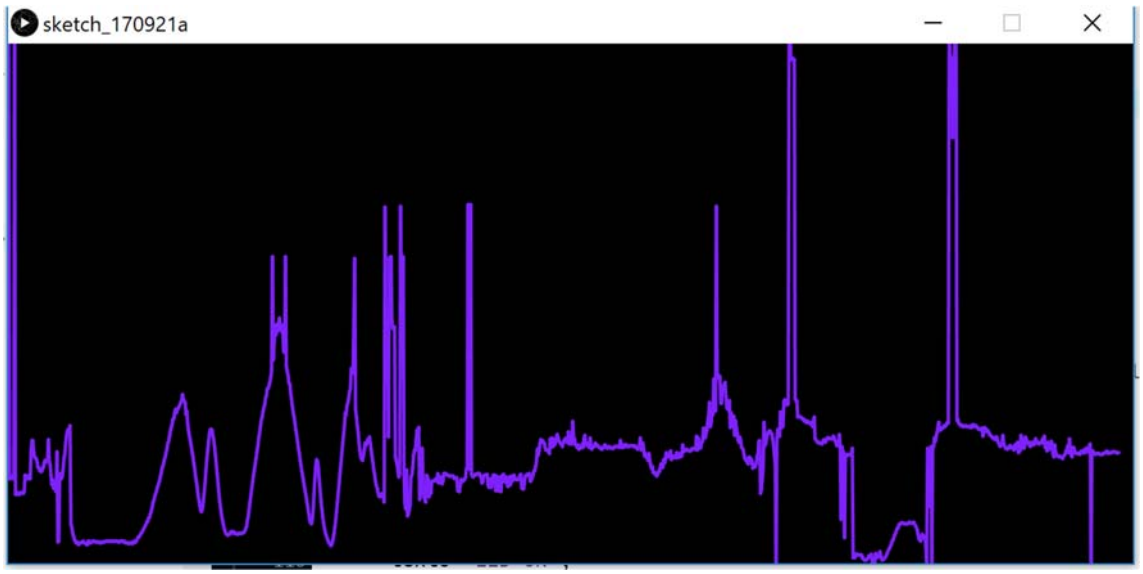


Figura 85: Gráfica construida en Processing para visualizar la respuesta en cm del sensor de ultrasonidos

5 UNA SOLUCIÓN MIXTA: ARDUINO CON LABVIEW

Planteamos la posibilidad de una solución mixta, una tarjeta de adquisición de datos libre con un software propietario.

El software de LabView dispone de una interfaz para Arduino, LIFA Toolkit, disponible de manera gratuita, que permite adquirir datos de Arduino y procesarlos en el entorno de programación gráfica de LabView.

Esta interfaz se encuentra disponible tanto para la versión comercial de LabView como para eloftware LabView Estudents Edition.

Esta opción puede resultar asequible si obtenemos licencias para estudiantes ya que el coste de una licencia comercial supera ampliamente los mil euros.

5.1 ARDUINO EN LABVIEW

Para inicializar y confgurar la comunicación de Arduino con el PC a través de un puerto USB se utiliza un bloque o VI llamado *Arduino Init,* este bloque se encuentra dentro del bloque de librerías que añadimos con el toolkit de Arduino. Este bloque nos permite seleccionar el tipo de placa Arduino, establecer la velocidad de comunicación serial y configurar los pines.

5.2 INSTALACIÓN DEL SOFTWARE NECESARIO

Una vez obtenida la licencia comercial, la licencia de estudiante o la licencia de 45 días gratuita seguimos los siuientes pasos:

- Instalamos los drivers NI-VISA que permiten la comunicación de LabView con diferentes plataformas hardware, se puede descargar a través del siguiente enlace:

  http://www.ni.com/nisearch/app/main/p/bot/no/ap/tech/lang/es/pg/1/sn/catnav:du,n8:3.25.123.1640,ssnav:ndr/

- Instalamos el gestor de descargas JKI VI Package Manager para instalar las herramientas de LabView, se puede descargar a través del siguiente enlace:

  https://vipm.jki.net/

- Instalación del LIFA a través del gestor de descargas.
- Instalación del IDE de Arduino a través del siguiente enlace:

  https://www.arduino.cc/en/main/software

- Cargamos la interfaz de LabView en Arduino, este fichero se ha descargado durante la instalación de LIFA y se llama *LIFA_Base*, lo abrimos con el IDE de Arduino y lo subimos a la placa.

Después de seguir estos pasos podemos utilizar la librería LIFA Toolkit en LabView.

6 DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS COMPLETO BASADO EN HARDWARE LIBRE Y SOFTWARE LIBRE

6.1 INTRODUCCIÓN

Vamos a realizar una estación meteorológica, basada en el hardware Arduino y el software MyOpenLab que hemos analizado anteriormente.

La estación meteorológica debe medir parámetros tales como temperatura, humedad, luminosidad y presión atmosférica.

El objetivo del diseño y programación de esta aplicación no es el de ser montada y encapsulada con el fin de ser usada sino el de estudiar las herramientas utilizadas para analizar si resultan una alternativa viable para su utilización en las aulas.

6.2 DISEÑO DEL HARDWARE

6.2.1 Descripción de los sensores

Sensor de temperatura

Vamos a utilizar el sensor de temperatura LM35 de National Semiconductors, utilizado en el apartado 3.3.1 de este documento.

Se trata de un sensor integrado cuya salida ofrece una tensión proporcional a la temperatura a razón de 10mV por cada Cº. Este sensor opera entre 0 y 100ºC, con una precisión de 0.5ºC a 25ºC.

Sensor de luz ambiente

Vamos a utilizar la fotorresistencia LDR como sensor de luz ambiente, modelo ST-76, utilizada en el apartado 3.3.2 de este documento.

Se trata de una resistencia cuya impedancia varía en función de la luminosidad, obteniendo su máximo valor en la oscuridad.

Los rangos de valores que toma la impedancia y su relación con la luminosidad del día son los siguientes:

- LDR < 500Ω -> Día soleado
- 500Ω < LDR < 1kΩ -> Día ligeramente nublado
- 1kΩ < LDR < 5kΩ -> Día nublado
- LDR > 5kΩ -> Noche

Sensor de presión

Vamos a utilizar el sensor de presión MPX4115AP de Motorola, utilizado en el apartado 3.3.3 de este documento.

Se trata de un sensor integrado cuya salida ofrece una tensión que varía en función de la presión atmosférica.

Sensor de humedad

Vamos a utilizar el sensor de humedad resistivo HS15P, de Amphenol Advanced Sensors, utilizado en el apartado 3.3.4 de este documento.

Se trata de una resistencia cuya impedancia varía en función de la humedad, obteniendo su valor mínimo teórico con un 100% de humedad en el ambiente de 800Ω y su valor máximo teórico con un 20% de humedad en el ambiente de 8MΩ, a 25ºC.

Utilizando la gráfica que relaciona humedad con impedancia del datasheet del sensor, dividiremos la impedancia en 8 rangos.

Los rangos de valores que toma la impedancia son:

- R > 10MΩ -> < 20%
- 10MΩ < R < 1MΩ -> 20% - 30%
- 1MΩ < R < 200kΩ -> 30% - 40%
- 200kΩ < R < 70kΩ -> 40% - 50%
- 70kΩ < R < 25kΩ -> 50% - 60%
- 25kΩ < R < 9kΩ -> 60% - 70%
- 9kΩ < R < 2k5Ω -> 70% - 80%
- 2k5Ω < R < 2kΩ -> 80% - 90%
- 2kΩ < R -> 90% - 1000%

Una humedad relativa superior al 80% indicará que está lloviendo.

6.2.2 Conexión de los sensores a la placa Arduino

El **sensor de temperatura** es un sensor integrado que cuenta con tres patillas, conectaremos la izquierda al pin de salida de 5V, la derecha al pin GND y la central a la entrada analógica A0 de la tarjeta Arduino. De este modo obtendremos en A0 una tensión dependiente de la temperatura.

El **sensor de luz ambiente** es una fotorresistencia LDR que conectaremos en un divisor resistivo con 100Ω. Los extremos del divisor resistivo irán a los pines de la tarjeta de 5V y GND, mientras que su centro estará conectado a la entrada A1 de la tarjeta. De este modo obtendremos en A1 una tensión dependiente del valor de la LDR y, por tanto, de la luminosidad del ambiente.

El **sensor de presión atmosférica** es un sensor integrado que cuenta con seis patillas, pero solo utilizaremos tres. Conectamos el primer pin, marcado con una muesca, a la entrada A2, el

segundo pin a GND y el tercero a 5V. De este modo obtendremos en A2 una tensión dependiente de la presión ambiental.

El **sensor de humedad** es una resistencia que conectaremos en un divisor resistivo con 100Ω. Los extremos del divisor resistivo irán a los pines de la tarjeta de 5V y GND, mientras que su centro estará conectado a la entrada A3 de la tarjeta. De este modo obtendremos en A3 una tensión dependiente del valor de la impedancia y, por tanto, de la humedad relativa.

Podemos ver como ha quedado el montaje final en la figura 86.

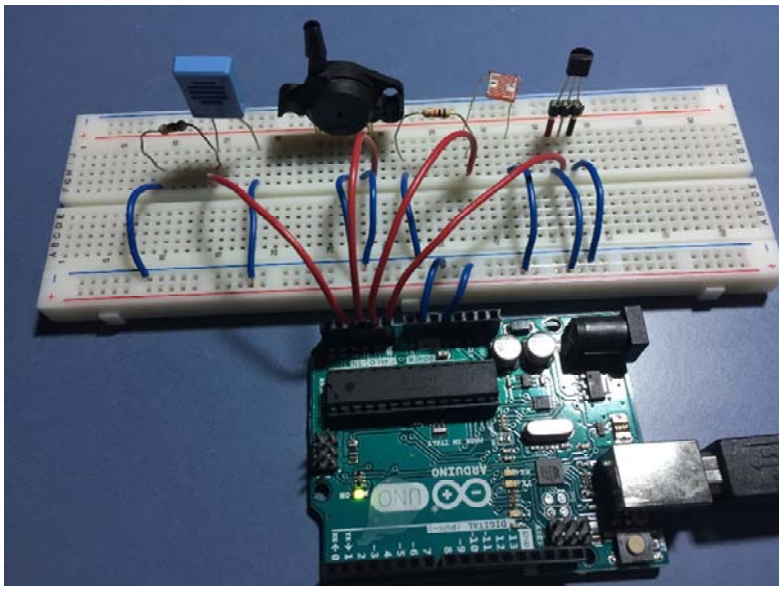


Figura 86: Montaje completo de la estación meteprológica

6.2.3 Adquisición de datos a partir de las tensiones de entrada de los sensores a la placa

Teniendo en cuenta que conectamos el sensor a una fuente de 5V y que la resolución de la placa es de 10 bits.

$$V_{LSB} = \frac{5v}{2^n - 1} = \frac{5}{2^{10} - 1} = \frac{5}{1023}\ V \qquad ; \qquad V_{OUT} = V_{LSB} * V_{A_n}$$

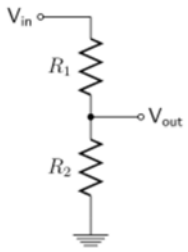
Temperatura

Sabiendo que la tensión es proporcional a la temperatura a razón de 10mV por cada Cº, obtendremos la temperatura a partir de la siguiente fórmula:

$$T^{a} = \frac{V_{A_0} * V_{LSB}}{10} = V_{A_0} * \frac{500}{1023}$$

Luz ambiente

Al conectar la resistencia LDR en serie con una resistencia de 10kΩ, formando un divisor resistivo, obtendremos la impedancia que adopta el sensor a partir de la siguiente fórmula:



$$V_{OUT} = V_{IN} * \frac{R_2}{R_1 + R_2} = 5v * \frac{10k\Omega}{R_{LDR} + 10k\Omega}$$

Por lo tanto: $$R_{LDR} = \frac{1023}{V_{A_1}} * 10k - 10k$$

Figura 87: Divisor resistivo

Presión atmosférica

El datasheet del sensor dice que la relación entre la tensión aportada por el sensor y la presión atmosférica, en kPA, es la siguiente:

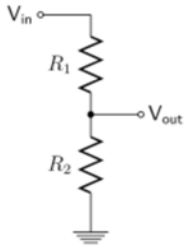
$$V_{OUT} = V_{IN} * (0{,}009 * P - 0{,}095) = 5v * (0{,}009 * P - 0{,}095)$$

Por lo tanto, podremos calcular la presión atmosférica a partir de la siguiente fórmula:

$$P = \frac{V_{A_2} * \frac{5}{1023} + 0{,}475}{0{,}45}\ hPa$$

Humedad relativa

Al conectar el sensor resistivo en serie con una resistencia de 10kΩ, formando un divisor resistivo, obtendremos la impedancia que adopta el sensor a partir de la siguiente fórmula:



$$V_{OUT} = V_{IN} * \frac{R_2}{R_1 + R_2} = 5v * \frac{10k\Omega}{R_{sensor} + 10k\Omega}$$

Por lo tanto: $$R_{sensor} = \frac{1023}{V_{A_3}} * 10k - 10k$$

Figura 88: Divisor resistivo

6.3 PROGRAMACIÓN EN MYOPENLAB

6.3.1 Panel frontal

En este panel vamos a colocar los siguientes visualizadores:

- Un termómetro de 0 a 100ºC que se encuentra en el bloque de librerías *Automatismos,* dentro de *JME-FE-01,* que indicará la temperatura.
- Un bloque *Canvas*, que se encuentra en la librería *Extras* para visualizar una imagen que indicará la luminosidad del día y si está lloviendo.
- Un bloque de *Indicador de Aguja* en el bloque de librerías *Numéricos* que indicará la presión atmosférica en hPa.
- Un bloque *Salida*, que se encuentra en la librería *Texto* para visualizar el rango de humedad relativa.

Con estos indicadores colocados, este panel queda como se muestra en la figura 89.

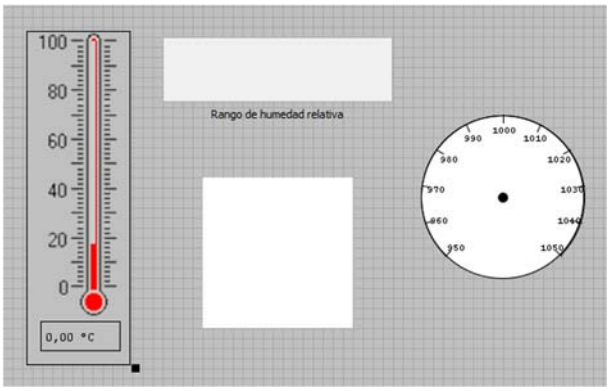


Figura 89: Estación meteorológica. Panel frontal

6.3.2 Panel de circuito

Comenzamos colocando el bloque de la Firmata de Arduino, situado en el bloque de librerías *Interfaces*, configurando el puerto COM y 57600 baudios de velocidad.

En la configuración de los pines, debemos configurar las entradas A0, A1, A2 y A3 como entradas analógicas. En la tabla 2 del apartado 3.2.3 podemos ver que se corresponden con los pines 14, 15, 16 y 17 de la Firmata.

Con todo configurado, podemos ver que al bloque de la Firmata le ha aparecido las patillas correspondientes para conectar los pines A0, A1, A2 y A3 de Arduino.

Sensor de temperatura

Tenemos que transformar la señal de la entrada A0 en temperatura, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Una vez colocados, conectamos la entrada al pin 14 de la Firmata y la salida al visualizador del termómetro.

Con los bloques colocados, los bloques del panel de circuitos para representar la temperatura quedan como se muestra en la figura 90.

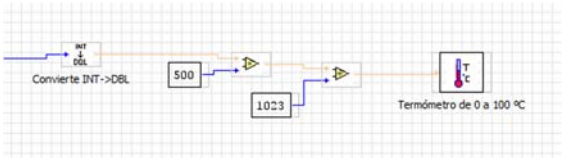


Figura 90: Panel de circuito de conversión a temperatura de la estación meterológica

Sensor de luz ambiente

A continuación, tenemos que transformar la señal de la entrada A1 en impedancia, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Ahora debemos comparar la impedancia obtenida con los rangos de luz ambiente descritos anteriormente. Para esto utilizamos bloques *mayor o igual que* de la librería *Comparadores* y bloques *int* de la librería *Numéricos,* con valores de 0Ω, 500Ω, 1kΩ y 5kΩ.

Una vez colocados, conectamos la entrada al pin 15 de la Firmata. Dejamos las salidas para más adelante.

Con los bloques colocados, los bloques del panel de circuitos para representar la luminosidad quedan como se muestra en la figura 91.

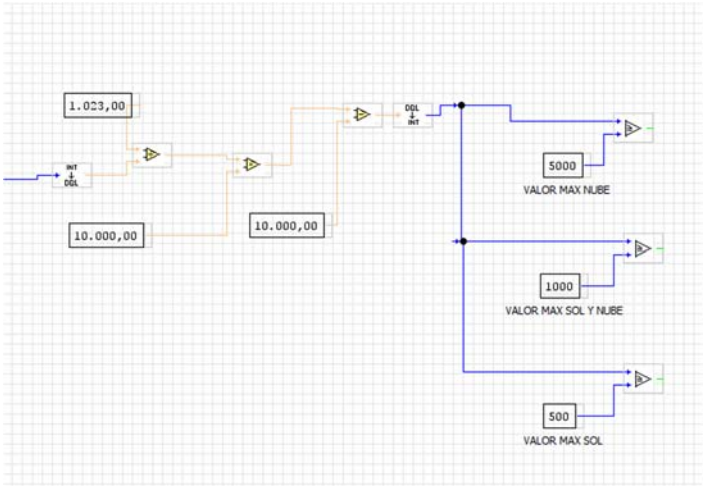


Figura 91: Panel de circuito de conversión a luz ambiente de la estación meterológica

Sensor de presión atmosférica

A continuación, tenemos que transformar la señal de la entrada A2 en presión atmosférica, para ello utilizamos bloques del bloque de librerías *Numéricos.*

Una vez colocados, conectamos la entrada al pin 16 de la Firmata y la salida al barómetro del panel frontal.

Con los bloques colocados, los bloques del panel de circuitos para representar la presión atmosférica quedan como se muestra en la figura 92.

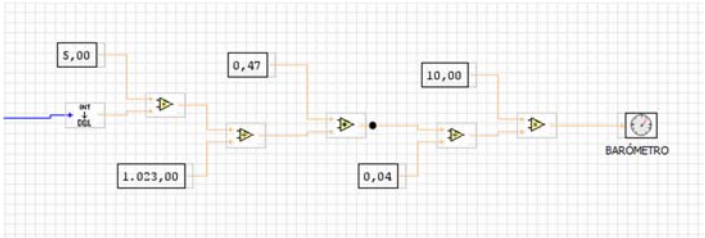


Figura 92: Panel de circuito de conversión a presión atmosférica de la estación meterológica

Sensor de humedad

Ahora debemos comparar la impedancia obtenida con los rangos de humedad descritos anteriormente. Para esto utilizamos bloques *mayor o igual que* de la librería *Comparadores,* bloques *int* de la librería *Numéricos,* con valores de 10MΩ, 1MΩ, 200kΩ, 70kΩ, 25kΩ, 9kΩ, 2k5Ω y 2kΩ, bloques *Pregunta, también* de la librería *Comparaciones,* y bloques de strings, uno indicando cada rango.

Conectamos la entrada a los bloques numéricos con el pin 17 de la Firmata.

Colocando los bloques *Pregunta* en cascada, a las salidas de los bloques *Comparadores*, obtenemos en la salida el string que nos indica el rango de humedad y lo unimos al bloque de salida string del panel frontal.

Utilizando un bloque *menor que* de la librería *Operadores digitales* obtenemos una salida tipo booleana que nos indicará si el rango de humedad es suficientemente elevado como para señalizar que está lloviendo, lo dejamos sin conectar hasta más adelante.

Con los bloques colocados, los bloques del panel de circuitos para representar la humedad quedan como se muestra en la figura 93.

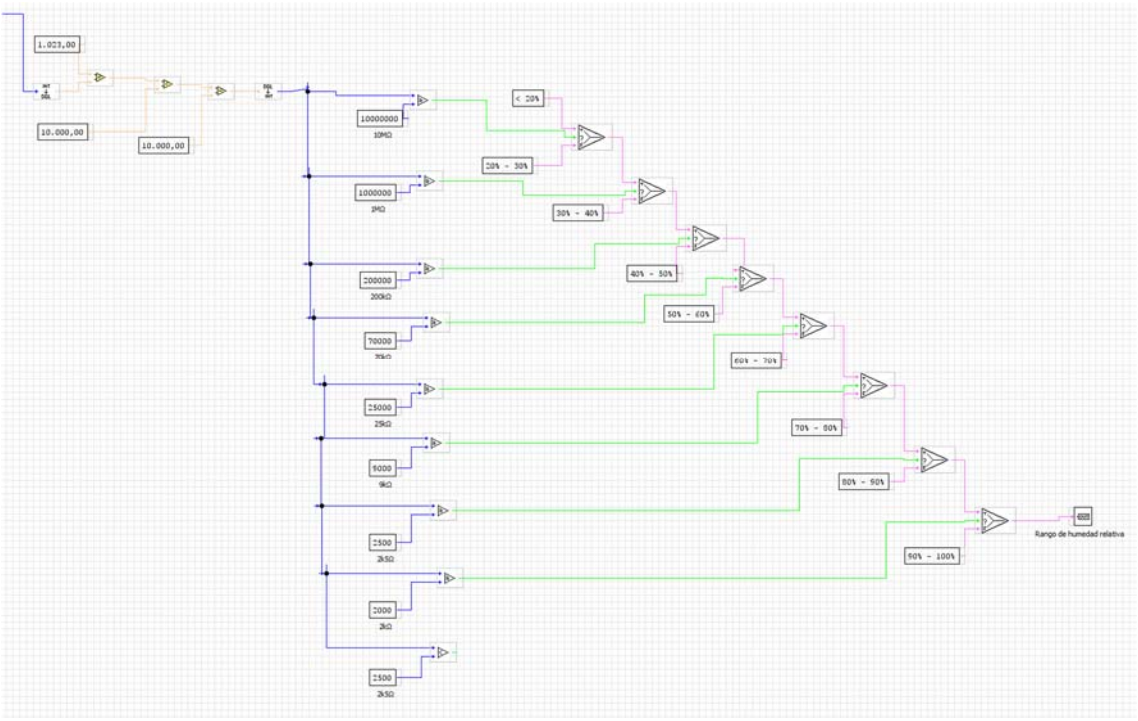


Figura 93: Panel de circuito de conversión a humedad de la estación meterológica

Estación meteorológica completa

Actualmente la situación es la siguiente:

- Los 4 pines de la Firmata están conectados en los flujos correspondientes.
- El flujo de la temperatura está unido al visualizador del termómetro.
- El flujo de la presión atmosférica está unido al barómetro.
- El flujo de la humedad está unido a la salida que indica a su rango, pero tenemos una salida booleana que indica si está lloviendo que hay que unir a una imagen de lluvia.
- El flujo de luz ambiente tiene 3 salidas booleanas que hay que unir a sus correspondientes imágenes.

Para completar el panel, tenemos que unir las 4 salidas booleanas al bloque de imagen del panel frontal. Para esto vamos a utilizar bloques *Pregunta* en cascada, de la librería comparaciones, para determinar qué imagen queremos mostrar. Obtendremos las cinco imágenes (una para cada rango de luminosidad y otra para lluvia) utilizando bloques *Imagen Estática* de la librería *Imágenes*, indicando en las propiedades de este bloque la ruta en la que tenemos guardada cada imagen.

Por último, conectamos el final de la cascada de preguntas al bloque del panel frontal en el que queremos mostrar la imagen.

Estos bloques quedan como se muestra en la figura 94.

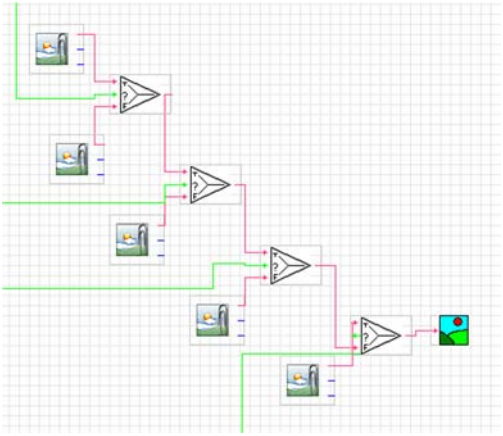


Figura 94: bloques de decisión de imagen de salida

Uniendo las 4 señales booleanas que activan la pregunta a las 3 del flujo de luminosidad y la de lluvia hemos completado el panel de circuito de la estación meteorológica.

El panel de circuito completo queda como en la figura 95.

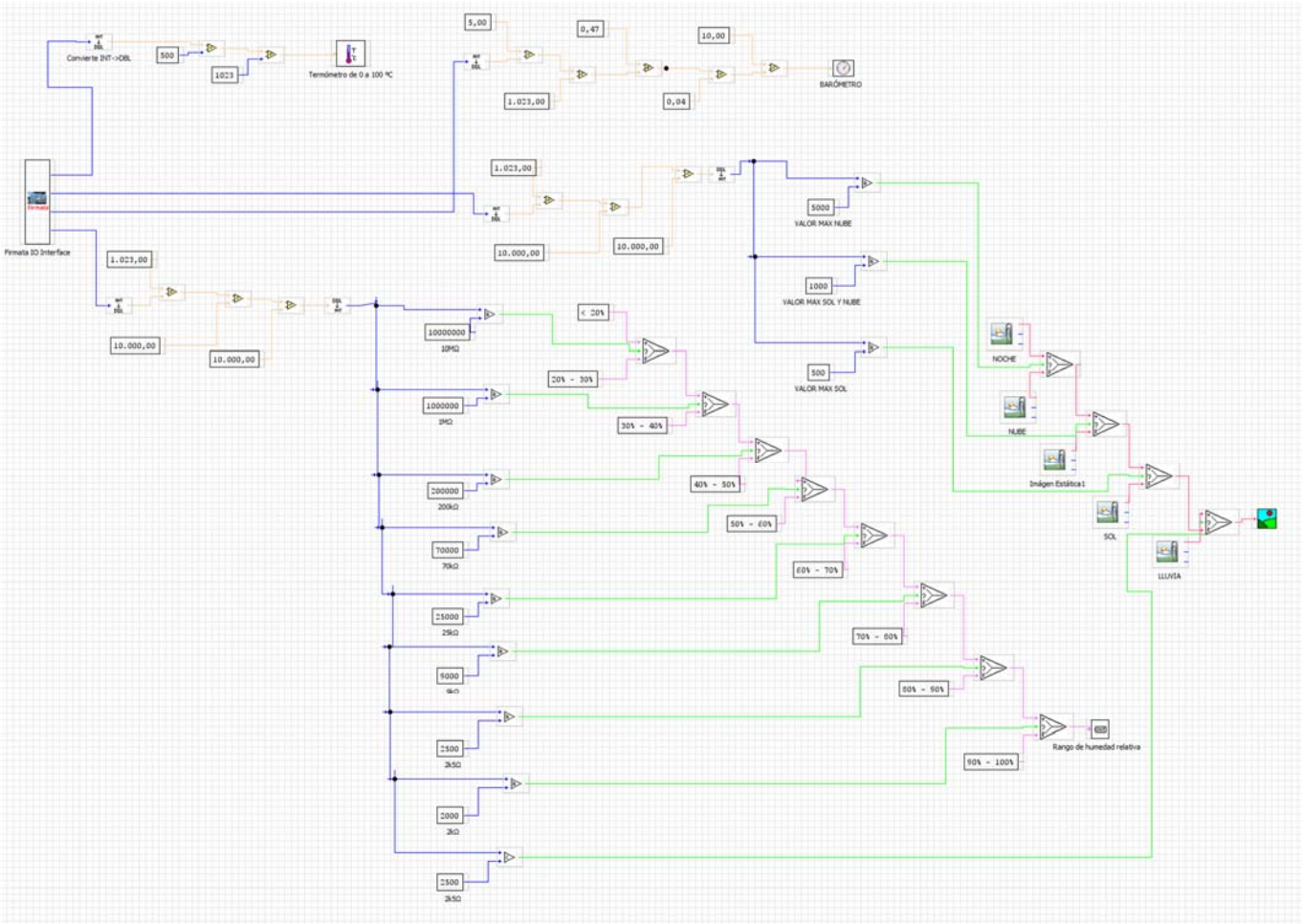


Figura 95: Panel de circuito de la estación meteorológica

6.3.3 Modo visualización

Ahora que hemos terminado de montar ambos paneles, ponemos el panel frontal en **modo ejecución**.

Modificando la temperatura y la iluminación obtenemos paneles como los de las figuras 96, 97, 98 y 99.

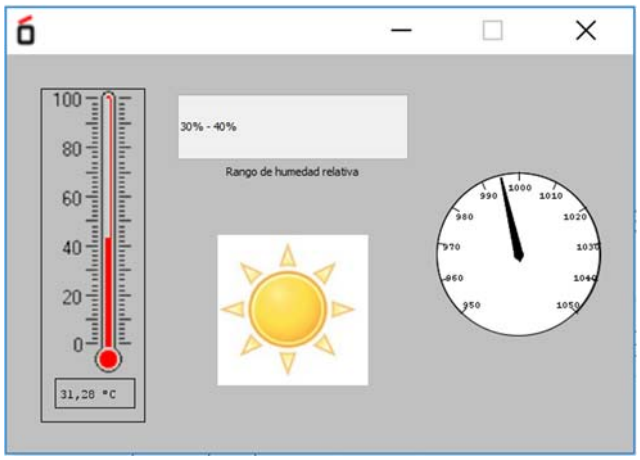


Figura 96: Estación meteorológica. Modo visualización 1

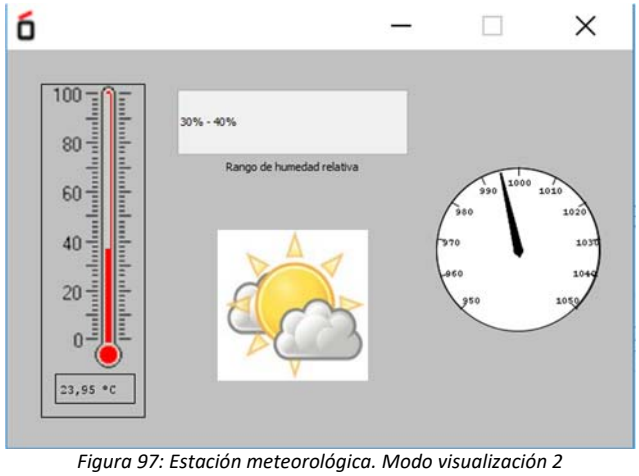


Figura 97: Estación meteorológica. Modo visualización 2

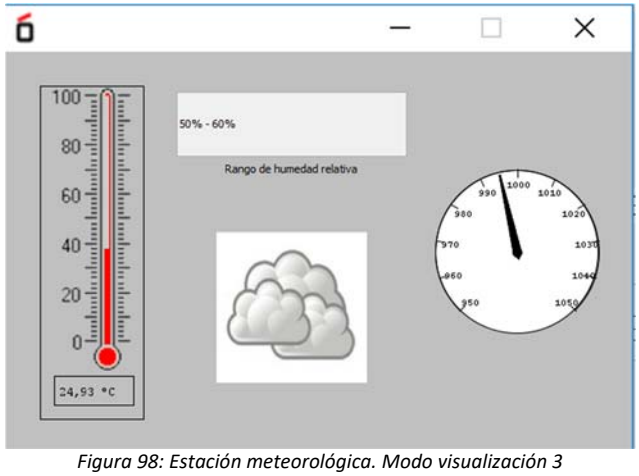


Figura 98: Estación meteorológica. Modo visualización 3

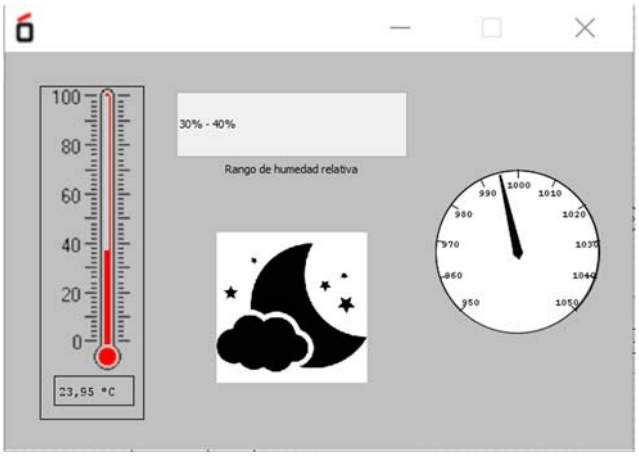


Figura 99: Estación meteorológica. Modo visualización 4

7 CONCLUSIONES Y TRABAJO FUTURO

Comparando las tarjetas de adquisición de datos podemos ver que las prestaciones de la tarjeta de National Instruments son mejores que las de la placa Arduino UNO, por su mayor resolución en bits, su mayor frecuencia de reloj y porque además de entradas también dispone de salidas digitales. Aun con todo esto, no hemos encontrado ningún impedimento a la hora de utilizar la placa de Arduino como una tarjeta de adquisición de datos.

En cuanto a la utilización de MyOpenLab cabe destacar que resulta complicado encontrar cualquier tipo de información, tanto de su instalación como de su uso, y que la mayor parte de la información encontrada proviene de José Manuel Ruiz Gutierrez, profesor del IES Fco. García Pavón en Tomelloso, Ciudad Real, quien se ha encargado de traducir y documentar al español todo el software.

A la hora de instalar el software nos hemos encontrado con que no podíamos utilizar su última versión, la 3.10.0, ya que aún tiene errores básicos como no poder abrir un proyecto guardado. Por lo que hemos utilizado la versión anterior, la 3.9.9.

Al esudiar el funcionamiento de MyOpenLab nos hemos dado cuenta de que hay pequeños errores, como que si borras un proyecto cuando aún está abierto la aplicación se bloquea, que la tarjeta debe estar conectada antes de abrir el software o no la leerá o que las ventanas de ayuda de los bloques de librerías no siempre están desarrolladas.

Hemos encontrado errores algo mayores, como que esta versión del software falla al crear subVM, lo que resulta muy útil en cuanto las aplicaciones crecen, permitiendo una programación más limpia y más fácil de mantener, o que algunos bloques no funcionan como deberían. Los bloques cuyo funcionamiento es erróneo son los siguientes:

- El bloque *Lista de Objetos Canvas,* de la librería *Canvas* del panel de circuito, cuyo bloque mostramos en la figura 100, cuyo funcionamiento debería permitirnos la opción de añadir o borrar el objeto canvas de entrada al grupo de objetos de salida, siempre incluye el objeto canvas en el grupo de objetos.

Figura 100: Entradas y salidas del bloque Lista de Objetos Canvas

- El bloque *Simple Pregunta,* de la librería *Comparaciones,* cuyo bloque mostramos en la figura 101, cuyo funcionamiento correcto sería transmitir la variable de entrada a la salida en el caso de que la entrada booleana fuese true, y sacar un valor nulo en caso de que fuese false, siempre transmite la variable, independientemente de la segunda entrada.

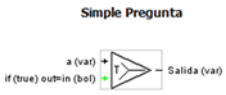


Figura 101: Entradas y salidas del bloque Simple Pregunta

También hemos observado que la visualización en los bloques del panel frontal no es muy precisa, como ejemplo podemos ver en la figura 102 como se muestra la temperatura en un termómetro al variarla.

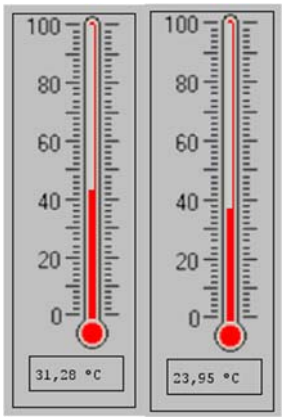


Figura 102: Indicadores de temperatur

Aun con todo esto, el verdadero problema que nos hemos encontrado a la hora de utilizar el software es que la **resolución temporal** de MyOpenLab es un impedimento a la hora de desarrollar aplicaciones que requieran una medida de tiempo inferior a 1ms.

Como no hemos encontrado este impedimento programando con el IDE de Arduino, pero este no tiene un entorno visual, podríamos encontrar una solución intermedia que supondría programar la adquisición de datos en el código StandarFirmata del IDE de Arduino y transmitir los resultados a la Firmata de MyOpenLab.

Utilizando el ejemplo del sensor de ultrasonidos, tendríamos que programar en el IDE de Arduino que el pin conectado al disparador emita pulsos de 1µs cada 60 ms, calcular el tiempo que transcurre hasta que el eco recibe un pulso, convertir ese tiempo en distancia y transmitírselo a MyOpenLab como entrada analógica. De este modo, desde MyOpenLab veríamos una señal de entrada analógica después de pasar por el conversor AD de la placa.

Con la utilización del IDE de Arduino como software de adquisición de datos y Processing como entorno visual, el problema de la resolución temporal también queda solventado. Procesing es muy potente como entorno visual carece de las librerías de visualización que hacen que la parte visual de LabView o MyOpenLab sea tán práctica.

La solución mixta de Arduino con LabView puede ser una opción factible ya que existe un programa académico de que trata de facilitar el uso de LabView en la docencia, pero este plantea fuertes condiciones y además está sujeto a cambios. Hasta hace poco, el programa académico implicaba acuerdos entre el centro docente y National Instruments y la gestión, por parte del profesor, de todas y cada una de las licencias de todos y cada uno de los estudiantes, que tenía que validar caso a caso con National Instruments. Recientemente el programa académico ha cambiado y ha sido sustituido por el suministro de licencias a los estudiantes siempre y cuando el centro renueve anualmente sus licencias, lo que para doce puestos de laboratorio supone miles de euros al año, un coste totalmente inasumible para los actuales presupuestos docentes.

REFERENCIAS Y BIBLIOGRAFÍA

[1] B Martín, A. Bono
Apuntes Instrumentación Electrónica, Universidad de Zaragoza
2015

[2] http://www.ni.com/data-acquisition/what-is/esa/

[3] https://es.wikipedia.org/wiki/Adquisición_de_datos

[4] http://revistas.proeditio.com/iush/quid/article/view/49/49

[5] https://www.fsf.org/about/what-is-free-software

[6] https://es.wikipedia.org/wiki/Hardware_libre

[7] https://aprendiendoarduino.wordpress.com/2016/03/28/que-es-arduino-hw-libre/

[8] https://store.arduino.cc/arduino-uno-rev3

[9] https://mecatronicauaslp.wordpress.com/2014/02/28/introduccion-a-beaglebone/

[10] https://blogthinkbig.com/4-alternativas-arduino-beaglebone-raspberrypi-nanode-waspmote

[11] http://es.engadget.com/2013/04/22/beaglebone-black-1ghz-45-dolares/

[12] https://es.wikipedia.org/wiki/Raspberry_Pi#Raspberry_Pi_3_Modelo_B

[13] https://www.raspberrypi.org/downloads/

[14] http://www.nanode.eu/what-is-nanode/

[15] http://www.nanode.eu/products/#tab-4

[16] http://www.libelium.com/products/waspmote/hardware/

[17] http://www.ni.com/pdf/manuals/371303n.pdf

[18] http://www.ni.com/pdf/manuals/375295b.pdf

[19] http://www.ni.com/pdf/manuals/375296b.pdf

[20] https://aprendiendoarduino.wordpress.com/2016/03/31/comunidad-arduino/

[21] https://es.wikipedia.org/wiki/LabVIEW

[22] https://es.slideshare.net/cperezm78/programaciongraficadearduinojosemanuelruizgutierrez2011

[23] https://processing.org

[24] https://es.wikipedia.org/wiki/Processing

[25] http://s4a.cat/index_es.html

[26] https://descubrearduino.com/alternativas-graficas-programar-arduino/#Scratch_for_Arduino_S4A

[27] https://myopenlab.de/download_myopenlab_now.html

[28] https://www.java.com/es/download/

[29] http://internetaula.ning.com/profile/JoseManuelRuizGutierrez

[30] https://campustecnologicovirtual.es/blogs/diseno-y-simulacion-con-myopenlab-la-alternativa-gratuita-al-labview-1

[31] https://myopenlab.de/faq.html

[32] https://myopenlab.es

[33] http://myopenlab.wixsite.com/myopenlab/arduino-c15n2

[34] https://myopenlab.de/interfaces/firmata.html

[35] https://www.arduino.cc/en/main/software

[36] https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

[37] https://myopenlab.de/cms-media/g/myopenlab/dokumente/canvas.pdf

[38] https://ccollins.wordpress.com/2013/12/11/how-to-measure-air-pressure-with-arduino/

[39] https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

[40] L. Sanromán Hernández

Control de nivel de agua y rellenado automático para pruebas de fiabilidad en placas de inducción.

Trabajo fin de grado, 2015

[41] https://pastebin.com/Pw23tFJK

[42] http://diymakers.es/arduino-processing-primeros-pasos/

GLOSARIO

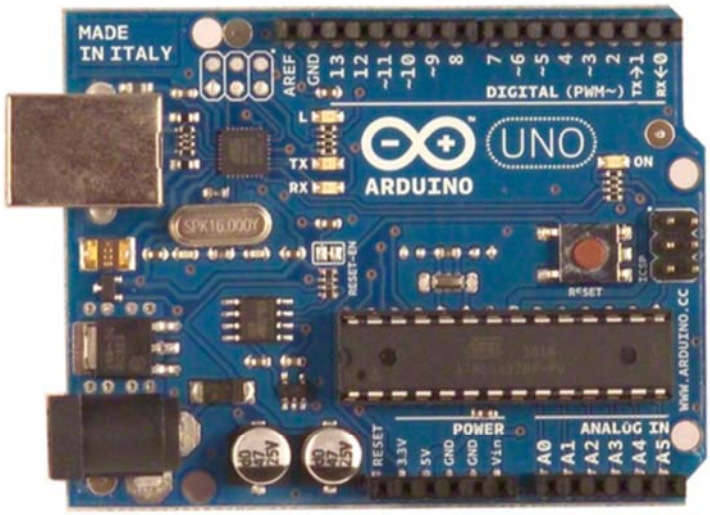
- **SAD:** Sistema de Adquisición de Datos
- **TFG:** Trabajo Fin de Grado
- **AMUX:** Multiplexor analógico
- **S&H:** Circuito de muestreo-retención (Sample & Hold)
- **PWM:** Pulse Width Modulation (Modulación por ancho de pulso)
- **IDE:** Integrated Development Environment (Entorno de desarrollo integrado)
- **RAM:** Random Access Memory (Memoria de acceso aleatorio)
- **USB:** Universal Serial Bus (Bus universal en serie)
- **CPU:** Central Processing Unit (Unidad de procesamiento central)
- **MicroSD:**
- **RTC:** Real Time Clock (Reloj en tiempo real)
- **API:** Application Programming Interface (interfaz de programación de aplicaciones)
- **DAQ:** Data Adquisition (adquisición de datos)
- **VI:** Virtual Instruments (Instrumentos virtuales)
- **GNU:** General Public License (Licencia pública general)
- **VM:** Virtual Machine
- **MIT:** Instituto Tecnológico de Massachusetts
- **GND:** Ground (Tierra o masa)
- **RGB:** Red Green Blue (Rojo verde azul)

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Anexo A – Características técnicas de la placa Arduino UNO

Arduino UNO



Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

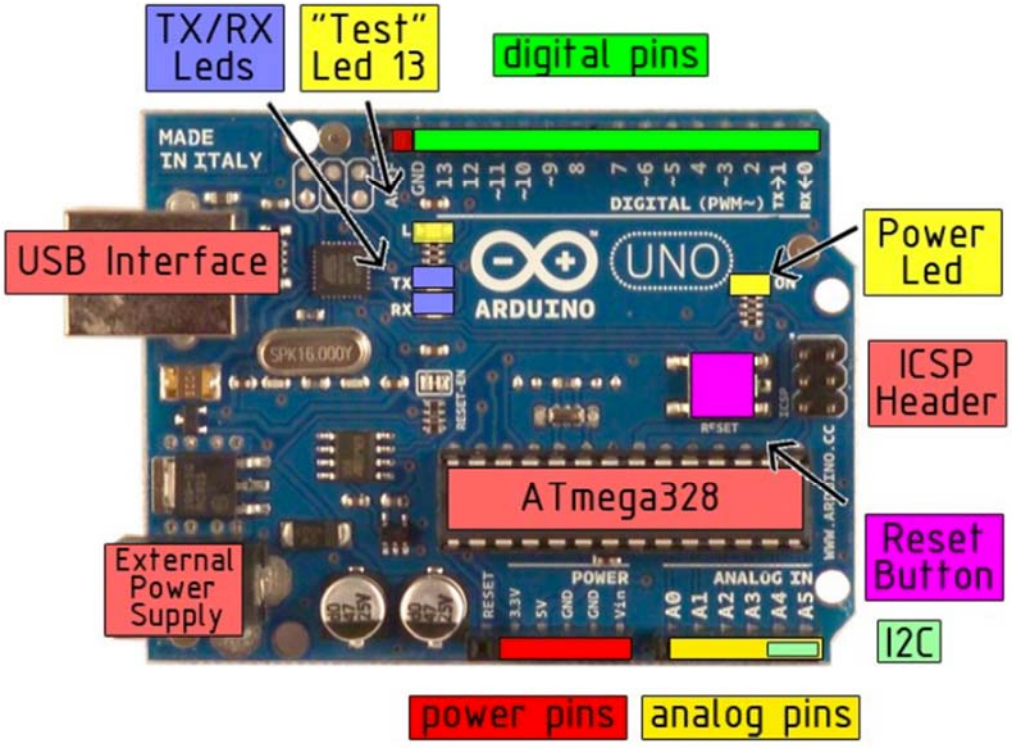
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Anexo B – Preparación de la tarjeta Arduino UNO para su utilización en MyOpenLab

La comunicación entre MyOpenLab y Arduino UNO se realiza gracias al protocolo Firmata.

Para poder utilizar Arduino junto a MyOpenLab, debemos comenzar cargando el Firmware de la Firmata en la tarjeta Arduino, para ello, descargamos el IDE de Arduino del siguiente enlace:

https://www.arduino.cc/en/main/software

Una vez descargado e instalado el IDE, procedemos a seleccionar placa y puerto en el desplegable de herramientas, tal y como se muestra en la figura 103.

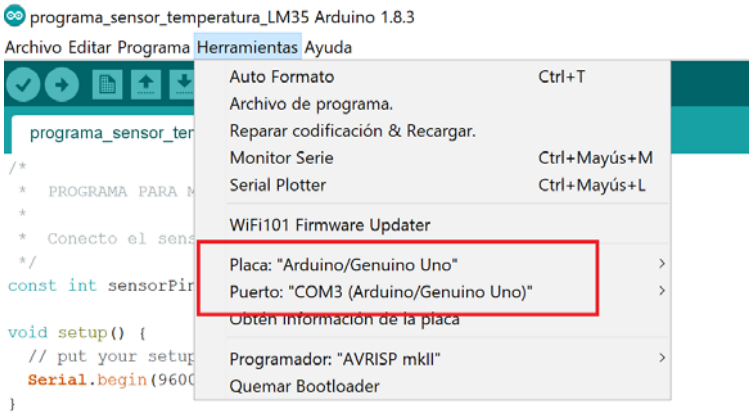


Figura 103: Configuracion de la tarjeta en el IDE de Arduino

A continuación, buscamos el código del Firmware en los desplegables Archivo -> Ejemplos -> Firmata -> StandardFirmata. Tal y como se muestra en la figura 104.

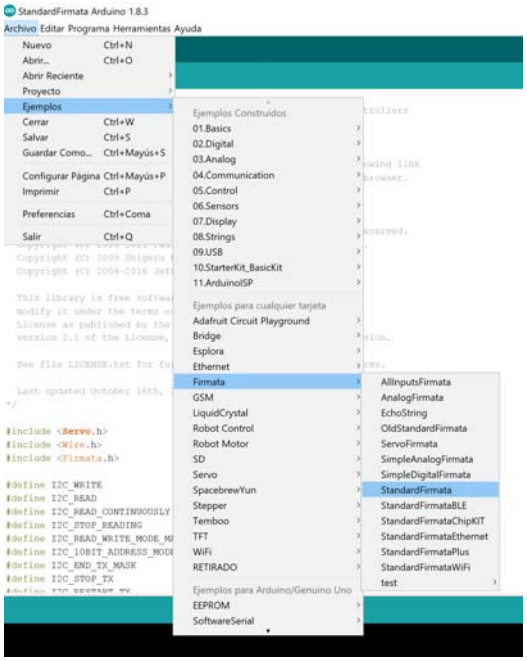


Figura 104: Búsqueda del Firmware en el IDE de Arduino

Antes de subir el cófigo de StandardFirmata a la tarjeta, debemos asegurarnos de que no tenemos abierto ningún otro IDE de Arduino ni MyOpenLab, en caso contrario no se cargará bien y no nos permitirá acceder a la placa desde el software de MyOpenLab.

Una vez obtenido el código lo subimos a la placa y podemos empezar a controlarla desde MyOpenLab.

Anexo C – Programación de los sensores en el IDE de Arduino

Sensor de temperatura LM35 de National Semiconductors.

```
const int sensorPin = A0;
const int led = 13;
const int temperaturaAlarma = 40;

void setup() {
        //  Abro el puerto serie y fijo la velocidad en 9600 baudios
        Serial.begin(9600);
        //Configuro el pin 13 como salida
        pinMode(led, OUTPUT);
}

void loop() {
         //  Obtengo la tensión de salida del sensor y la transformo en ºC
        int value = analogRead(sensorPin);
        float milivolts = (value / 1023.0) * 5000;
        float celsius = milivolts / 10;
        //  Muestro el resultado por pantalla
        Serial.print(celsius);
         Serial.println(" C");
        //  Si la temperatura es superior a mi temperatura de alarma, enciendo el led
        if(celsius > temperaturaAlarma){
                digitalWrite(led, HIGH);
        }
        else{
                digitalWrite(led, LOW);
        }
        //  Repito el bucle cada 100ms
        delay(100);
}
```

Sensor de luz ambiente LDR ST-76

```
const int sensorPin = A0;

void setup() {
        //  Abro el puerto serie y fijo la velocidad en 9600 baudios
        Serial.begin(9600);
}
```

```
void loop() {
        //  Obtengo la tensión del divisor resistivo
        int value = analogRead(sensorPin);
        int tension = (value / 1023.0) * 5000;    //en mV
        Serial.println(tension);
        //  Repito el bucle cada 100ms
        delay(100);
}
```

Sensor de presión atmosférica MPX4115AP

```
const int sensorPin = A0;

void setup() {
        //  Abro el puerto serie y fijo la velocidad en 9600 baudios
        Serial.begin(9600);
}

void loop() {
        //  Obtengo la tensión del sensor
        double value = analogRead(sensorPin);
        double tension = (value / 1023.0) * 5;  //en voltios
        double presion = ((tension + 0.475)/0.045)*10; // en hPa
        Serial.println(presion);
        //  Repito el bucle cada 100ms
        delay(100);
}
```

Sensor de humedad HS15P

```
const int sensorPin = A0;

void setup() {
        //  Abro el puerto serie y fijo la velocidad en 9600 baudios
         Serial.begin(9600);
}

void loop() {
         //  Obtengo la tensión del divisor resistivo
         int value = analogRead(sensorPin);
         float tension = float((value / 1023.0) * 5);  //en V
         float impedancia = (50 - 10 * tension) / tension; // en k

         Serial.println(tension);
         Serial.println(impedancia);
         Serial.println("");
```

```
        //  Repito el bucle cada 100ms
        delay(3000);
}
```

Sensor de ultrasonidos HC-SR04

```
long tiempo;
float distancia;
const int disparo = 13; //pin con el que realizo el disparo
const int eco = 12; //pin en el que recibo el eco

void setup() {
        //  Abro el puerto serie y fijo la velocidad en 9600 baudios
        Serial.begin(9600);
        pinMode(disparo, OUTPUT);
        pinMode(eco, INPUT);
}

void loop() {
        //Mando un pulso de 10 microsegundos en la patilla trig
        digitalWrite(disparo, HIGH);
        delayMicroseconds(10);
        digitalWrite(disparo, LOW);

        //Leo el tiempo en microsegundos en el que la patilla eco está a 1
        tiempo = pulseIn(eco, HIGH); //microsegundos ida y vuelta
        //Calculo la distancia
        distancia = float((tiempo * 0.0343) / 2);//en cm

        Serial.println(distancia);

        delay(60);        //Un pulso de 10 microsegundos cada 60 milisegundos
}
```

Anexo D – Características técnicas del sensor de temperatura

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 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 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 μA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 μA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

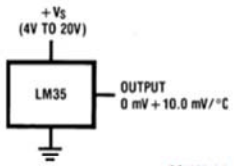
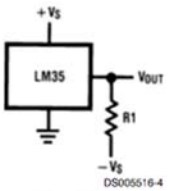


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

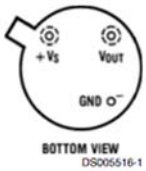


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

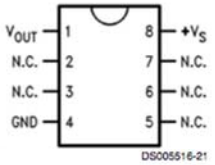
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

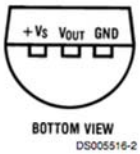
SO-8
Small Outline Molded Package



N.C. = No Connection

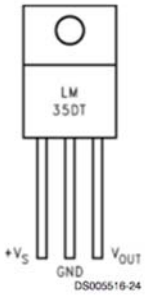
Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package



Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60˚C to +180˚C
TO-92 Package,	−60˚C to +150˚C
SO-8 Package,	−65˚C to +150˚C
TO-220 Package,	−65˚C to +150˚C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300˚C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260˚C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55˚C to +150˚C
LM35C, LM35CA	−40˚C to +110˚C
LM35D	0˚C to +100˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
	T_A=−10˚C	±0.3			±0.3		±1.0	˚C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25˚C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

LM35

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy, LM35, LM35C (Note 7)	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=−10˚C	±0.5			±0.5		±1.5	˚C
	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	˚C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	˚C
Accuracy, LM35D (Note 7)	T_A=+25˚C				±0.6	±1.5		˚C
	T_A=T_{MAX}				±0.9		±2.0	˚C
	T_A=T_{MIN}				±0.9		±2.0	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±2.0		±0.4	±2.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.1		±0.01	±0.1		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	80		56	80		µA
	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25˚C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	2.0		0.2	2.0		µA
	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Note 1: Unless otherwise noted, these specifications apply: −55˚C≤T_J≤+150˚C for the LM35 and LM35A; −40˚≤T_J≤+110˚C for the LM35C and LM35CA; and 0˚≤T_J≤+100˚C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2˚C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400˚C/W, junction to ambient, and 24˚C/W junction to case. Thermal resistance of the TO-92 package is 180˚C/W junction to ambient. Thermal resistance of the small outline molded package is 220˚C/W junction to ambient. Thermal resistance of the TO-220 package is 90˚C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: 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.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/˚C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in ˚C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

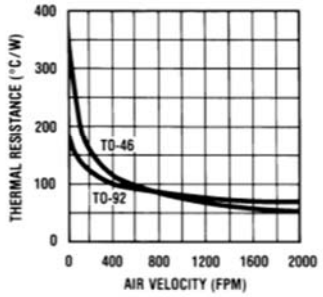
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

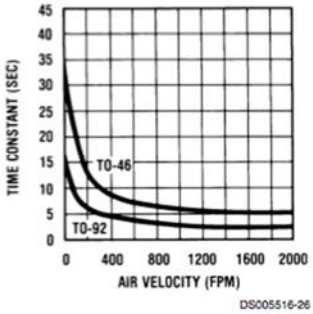
Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

Typical Performance Characteristics

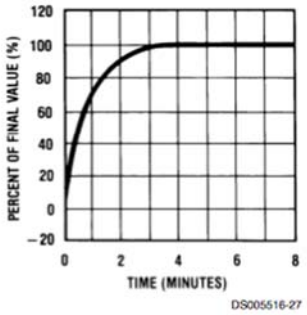
Thermal Resistance Junction to Air



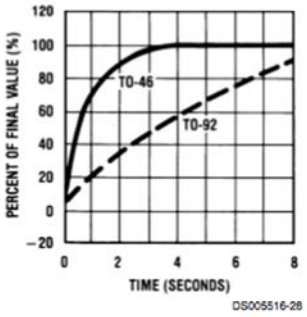
Thermal Time Constant



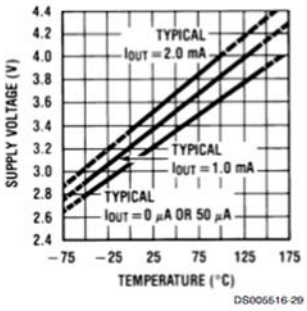
Thermal Response in Still Air



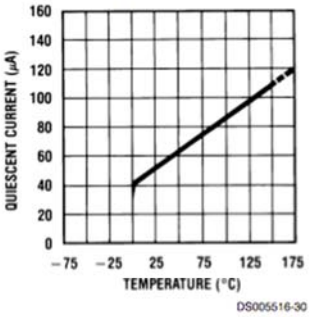
Thermal Response in Stirred Oil Bath



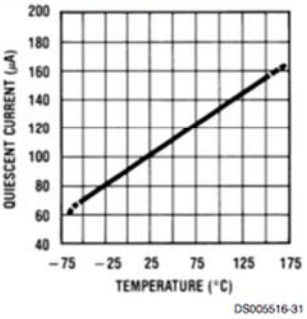
Minimum Supply Voltage vs. Temperature



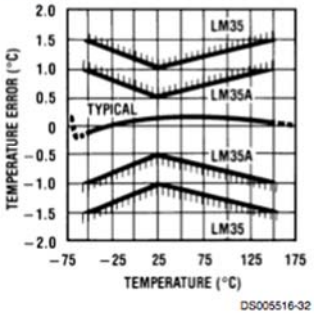
Quiescent Current vs. Temperature (In Circuit of *Figure 1*.)



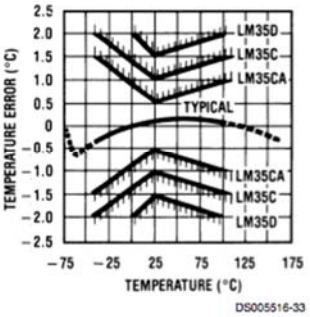
Quiescent Current vs. Temperature (In Circuit of *Figure 2*.)



Accuracy vs. Temperature (Guaranteed)

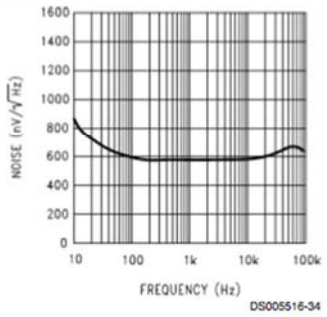


Accuracy vs. Temperature (Guaranteed)

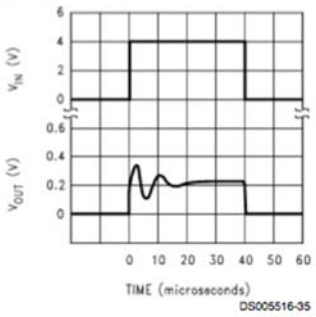


Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01˚C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V– terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400˚C/W	100˚C/W	180˚C/W	140˚C/W	220˚C/W	110˚C/W	90˚C/W
Moving air	100˚C/W	40˚C/W	90˚C/W	70˚C/W	105˚C/W	90˚C/W	26˚C/W
Still oil	100˚C/W	40˚C/W	90˚C/W	70˚C/W			
Stirred oil	50˚C/W	30˚C/W	45˚C/W	40˚C/W			
(Clamped to metal, Infinite heat sink)	(24˚C/W)				(55˚C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications

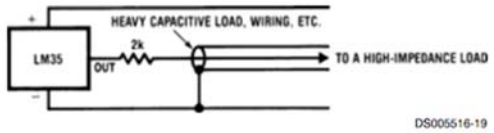


FIGURE 3. LM35 with Decoupling from Capacitive Load

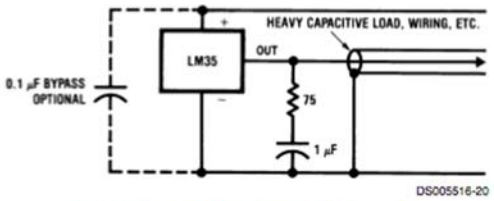


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in *Figure 5*, *Figure 6* or *Figure 8* it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 μF from output to ground are often useful. These are shown in *Figure 13*, *Figure 14*, and *Figure 16*.

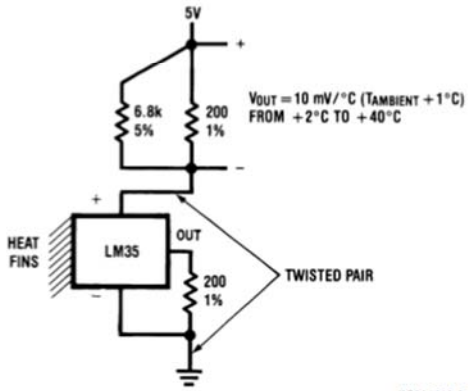


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

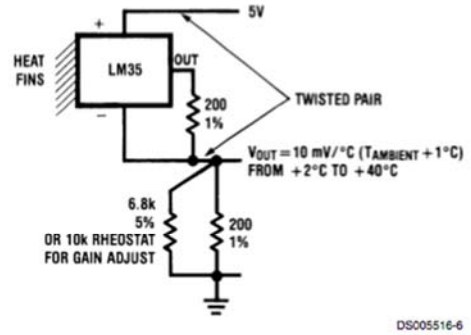


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

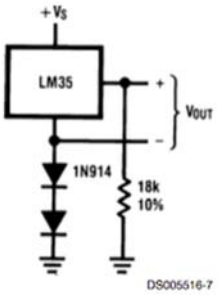


FIGURE 7. Temperature Sensor, Single Supply, −55° to +150°C

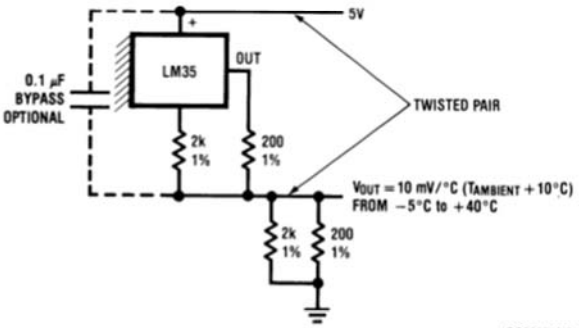


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

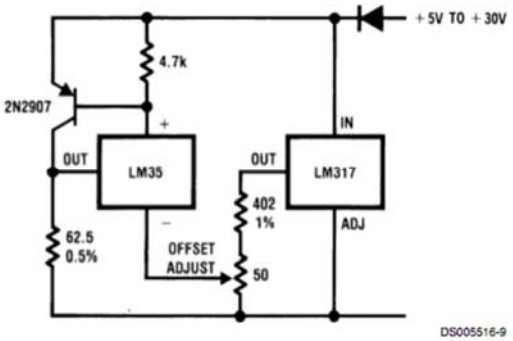


FIGURE 9. 4-To-20 mA Current Source (0°C to +100°C)

Typical Applications (Continued)

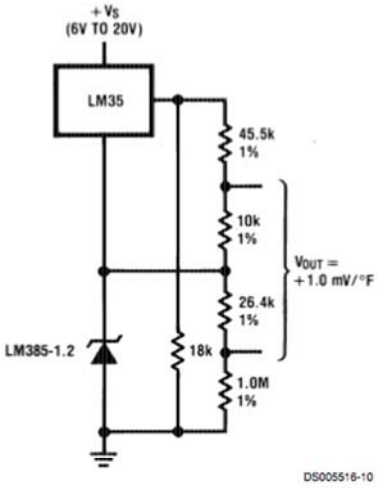


FIGURE 10. Fahrenheit Thermometer

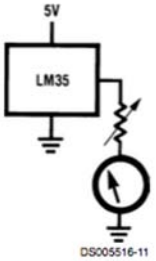


FIGURE 11. Centigrade Thermometer (Analog Meter)

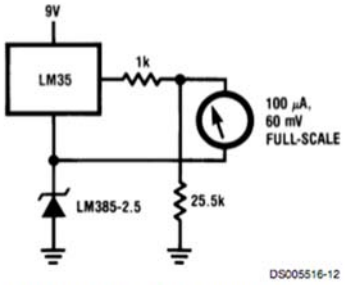


FIGURE 12. Fahrenheit ThermometerExpanded Scale Thermometer (50˚ to 80˚ Fahrenheit, for Example Shown)

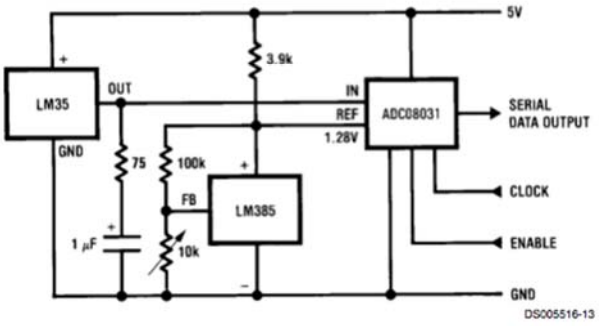


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128˚C Full Scale)

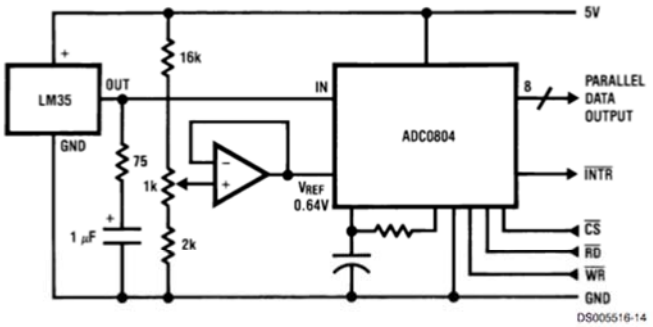
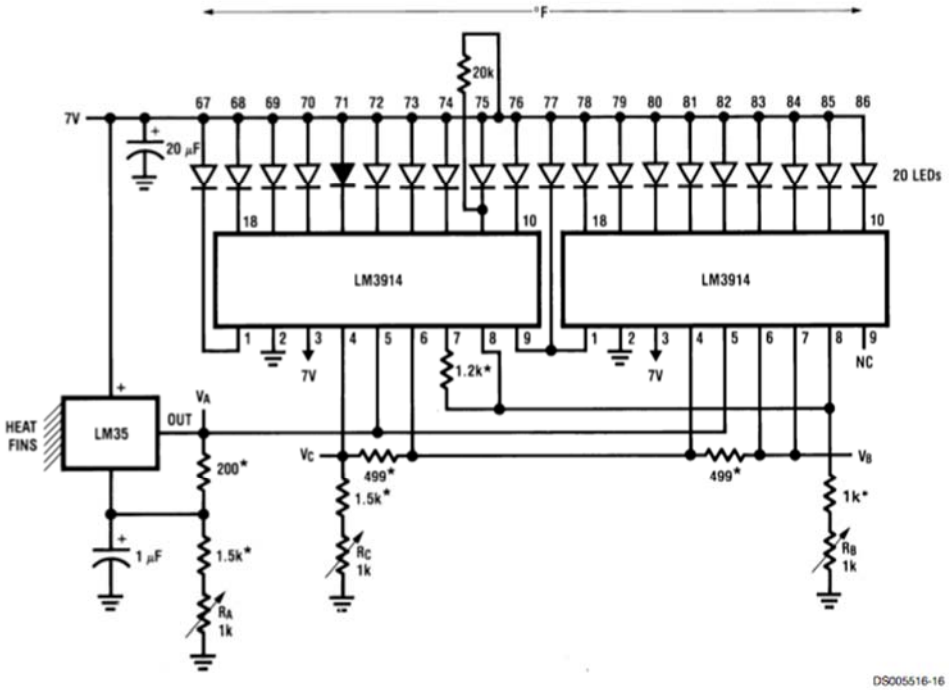


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128˚C Full Scale)

Typical Applications (Continued)



*=1% or 2% film resistor
Trim R_B for V_B=3.075V
Trim R_C for V_C=1.955V
Trim R_A for V_A=0.075V + 100mV/°C x $T_{ambient}$
Example, V_A=2.275V at 22°C

FIGURE 15. Bar-Graph Temperature Display (Dot Mode)

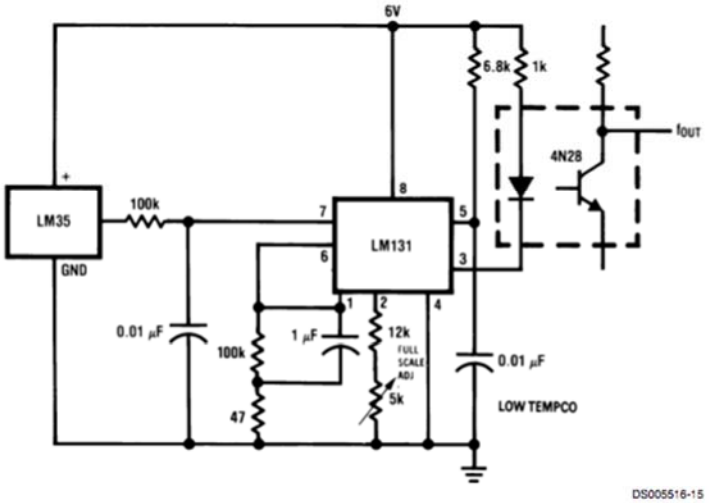
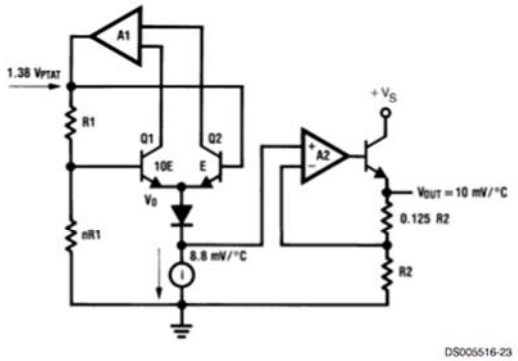
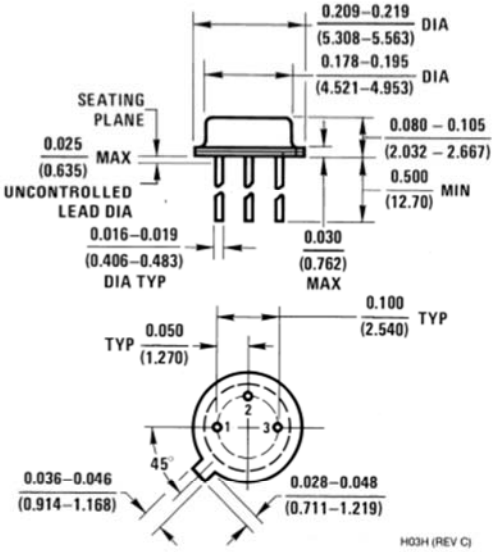


FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

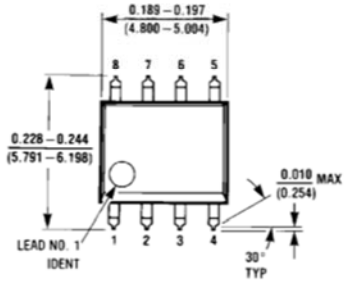
Block Diagram

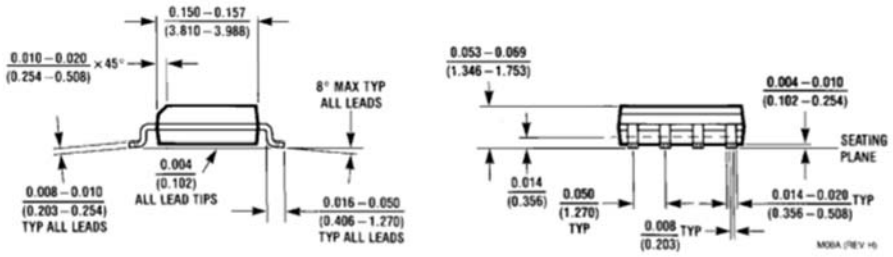


Physical Dimensions inches (millimeters) unless otherwise noted



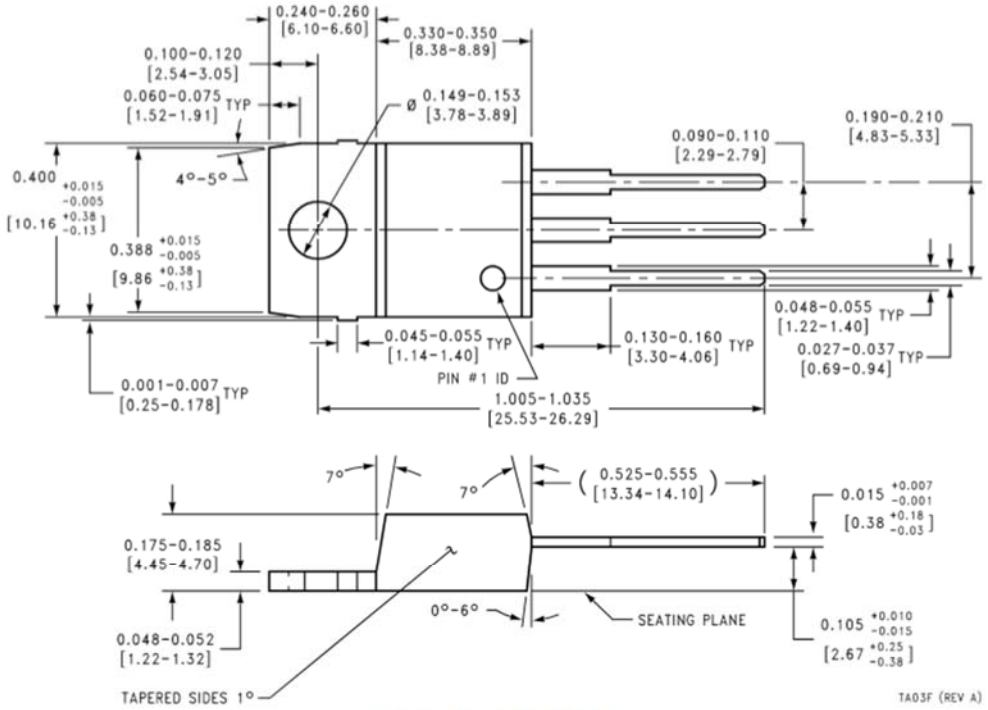
TO-46 Metal Can Package (H)
Order Number LM35H, LM35AH, LM35CH, LM35CAH, or LM35DH
NS Package Number H03H





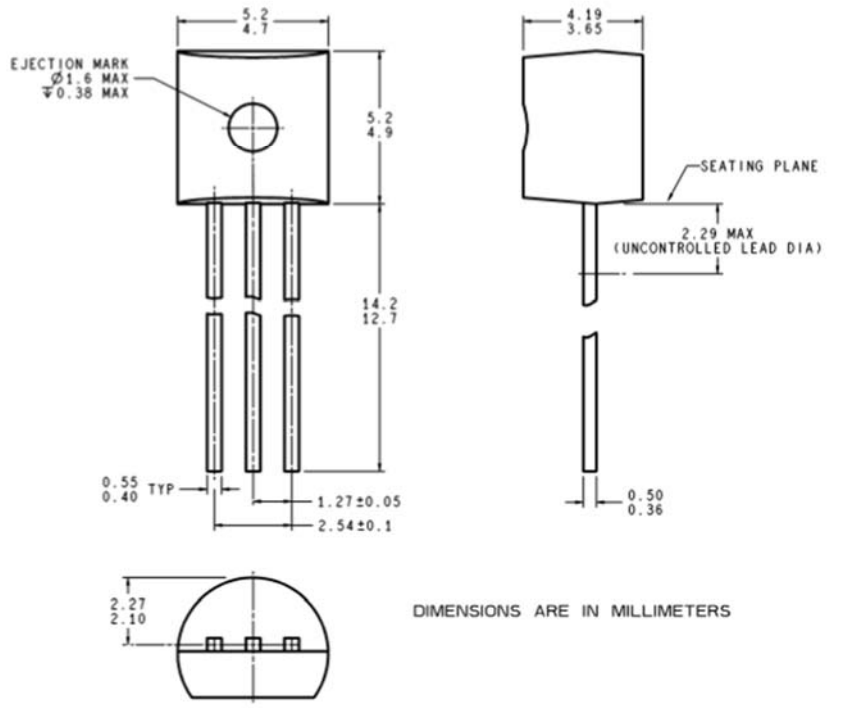
SO-8 Molded Small Outline Package (M)
Order Number LM35DM
NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Power Package TO-220 (T)
Order Number LM35DT
NS Package Number TA03F

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



TO-92 Plastic Package (Z)
Order Number LM35CZ, LM35CAZ or LM35DZ
NS Package Number Z03A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

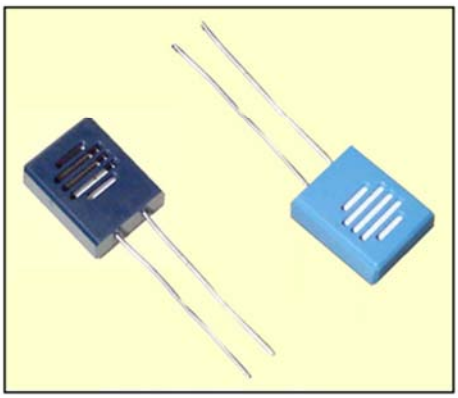
National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

Anexo E – Características técnicas del sensor de humedad

HUMIDITY SENSORS: TYPE HS12P, HS15P
RELATIVE HUMIDITY SENSOR

DESCRIPTION:
Non-refresh type humidity sensor made of polymer.

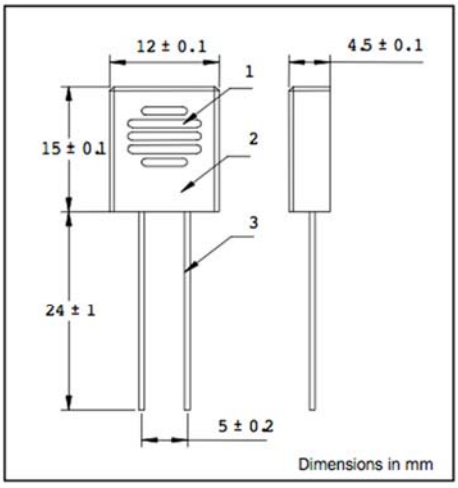


FEATURES:
- Good long term reliability
- Cost effective performance
- HS15P water resistive
- Typical applications include humidity monitors, humidity controllers, air conditioners, humidifiers, dehumidifiers, automatic ventilation

DATA:

CODE	HS12P	HS15P
Operating Temperature	0 to 50°C	
Operating Humidity	20 to 90% RH (without condensing)	20 to 100% RH
Impedance at 25°C 50% RH	60 kohm +/-30 kohm (+/-5% RH)	
Rated Voltage	AC.1V rms	
Rated Frequency	50 Hz to 1 kHz	
Consumption Power	0.3 mW	

DIMENSIONS:



CODING:
HS12P: quick response type
HS15P: water resistant type

WARNING:
Use only within the specified limits.
Do not disassemble and change any parts.
Do not apply DC voltage or DC bias.
Do not immerse into water or any solution.

1. Filter
2. Case ..... ABS: Dark blue (HS12P)
..... Light blue (HS15P)
3. Lead wire ..... Sn-Pb plated Cu
..... Diameter: 0.6mm

Data sheet D-HS12/15P-1

Crown Industrial Estate, Priorswood Rd
Taunton, Somerset TA2 8QY UK
Tel +44 (0)1823 335200
Fax +44 (0)1823 332637

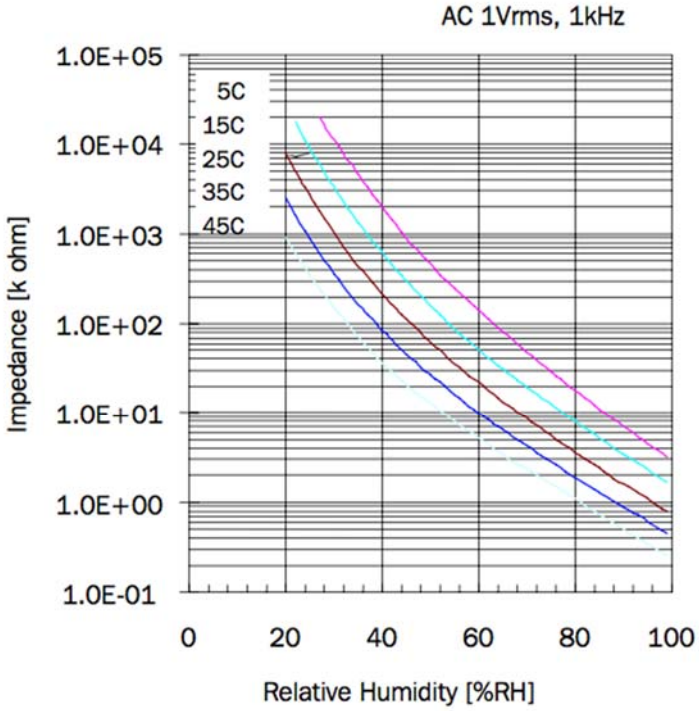
808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

967 Windfall Road
St Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969

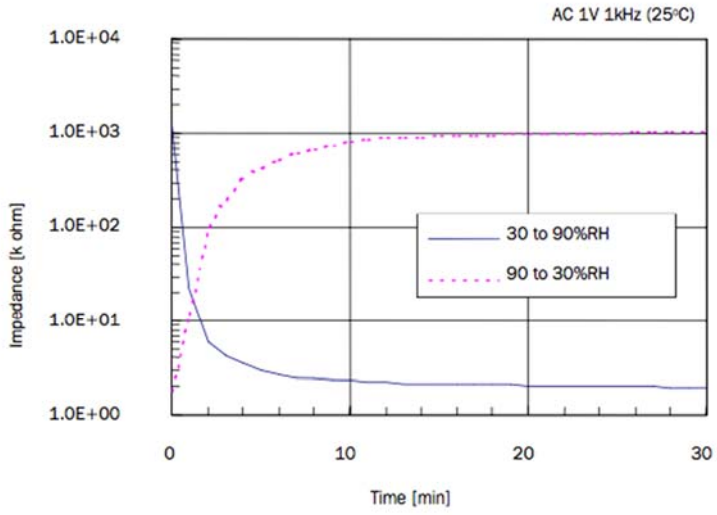
HUMIDITY SENSORS: TYPE HS12P, HS15P

RELATIVE HUMIDITY SENSOR

TYPICAL HUMIDITY CURVE:



TYPICAL RESPONSE CURVE:



Crown Industrial Estate, Priorswood Rd
Taunton, Somerset TA2 8QY UK
Tel +44 (0)1823 335200
Fax +44 (0)1823 332637

808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

967 Windfall Road
St Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969

Anexo F – Características técnicas del sensor de presión atmosférica

MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document by MPX4115/D

senseon

Integrated Silicon Pressure Sensor Altimeter/Barometer Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

MPX4115 SERIES

OPERATING OVERVIEW INTEGRATED PRESSURE SENSOR 15 to 115kPa (2.18 to 16.7 psi) 0.2 to 4.8 Volts Output

The MPX4115 series is designed to sense absolute air pressure in an altimeter or barometer (BAP) applications. Motorola's BAP sensor integrates on–chip, bipolar op amp circuitry and thin film resistor networks to provide a high level analog output signal and temperature compensation. The small form factor and high reliability of on–chip integration makes the Motorola BAP sensor a logical and economical choice for application designers.

Features

- 1.5% Maximum Error over 0° to 85°C
- Ideally suited for Microprocessor or Microcontroller–Based Systems
- Patented Silicon Shear Stress Strain Gauge
- Available in Absolute, Differential and Gauge Configurations
- Durable Epoxy Unibody Element
- Easy–to–Use Chip Carrier Option

Application Examples

- Altimeter
- Barometer

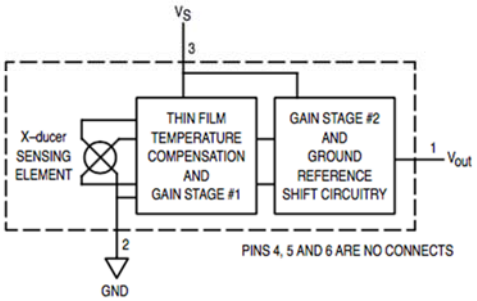


Figure 1. Fully Integrated Pressure Sensor Schematic

BASIC CHIP CARRIER ELEMENT CASE 867–08, STYLE 1

PORT OPTION CASE 867B–04, STYLE 1

NOTE: Pin 1 is the notched pin.

PIN NUMBER			
1	V_{out}	4	N/C
2	Grd	5	N/C
3	V_S	6	N/C

NOTE: Pins 4, 5, and 6 are internal device connections. Pin 1 is noted by the notch in the Lead. Do not connect to external circuitry or ground.

Senseon and X–ducer are trademarks of Motorola, Inc.

REV 3

MOTOROLA

MPX4115 SERIES

MAXIMUM RATINGS(1)

Parametrics	Symbol	Value	Unit
Overpressure(2) (P1 > P2)	P_{max}	400	kPa
Burst Pressure(2) (P1 > P2)	P_{burst}	1000	kPa
Storage Temperature	T_{stg}	−40° to +125°	°C
Operating Temperature	T_A	−40° to +125°	°C

1. T_C = 25°C unless otherwise noted.
2. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

OPERATING CHARACTERISTICS (V_S = 5.1 Vdc, T_A = 25°C unless otherwise noted, P1 > P2)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range(1)		P_{OP}	15	—	115	kPa
Supply Voltage(2)		V_S	4.85	5.1	5.35	Vdc
Supply Current		I_o	—	7.0	10	mAdc
Minimum Pressure Offset(3) @ V_S = 5.1 Volts	(0 to 85°C)	V_{off}	0.135	0.204	0.273	Vdc
Full Scale Output(4) @ V_S = 5.1 Volts	(0 to 85°C)	V_{FSO}	4.725	4.794	4.863	Vdc
Full Scale Span(5) @ V_S = 5.1 Volts	(0 to 85°C)	V_{FSS}	—	4.59	—	Vdc
Accuracy(6)	(0 to 85°C)	—	—	—	±1.5	$\%V_{FSS}$
Sensitivity		V/P	—	46	—	mV/kPa
Response Time(7)		t_R	—	1.0	—	mS
Output Source Current at Full Scale Output		I_{o+}	—	0.1	—	mAdc
Warm–Up Time(8)		—	—	20	—	mSec
Offset Stability(9)		—	—	±0.5	—	$\%V_{FSS}$

Decoupling circuit shown in Figure 3 required to meet electrical specifications.

MECHANICAL CHARACTERISTICS

Characteristic	Symbol	Min	Typ	Max	Unit
Weight, Basic Element (Case 867)	—	—	4.0	—	Grams
Common Mode Line Pressure(10)	—	—	—	690	kPa

NOTES:

1. 1.0kPa (kiloPascal) equals 0.145 psi.
2. Device is ratiometric within this specified excitation range.
3. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
4. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
5. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
6. Accuracy (error budget) consists of the following:
   - Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.
   - Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   - Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure at 25°C.
   - TcSpan: Output deviation over the temperature range of 0° to 85°C, relative to 25°C.
   - TcOffset: Output deviation with minimum pressure applied, over the temperature range of 0° to 85°C, relative to 25°C.
   - Variation from Nominal: The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS} at 25°C.
7. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
8. Warm–up is defined as the time required for the product to meet the specified output voltage after the Pressure has been stabilized.
9. Offset stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.
10. Common mode pressures beyond what is specified may result in leakage at the case–to–lead interface.

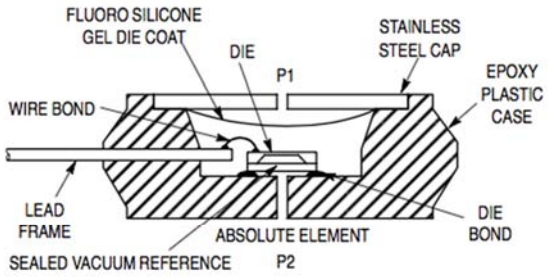


Figure 2. Cross–Sectional Diagram (Not to Scale)

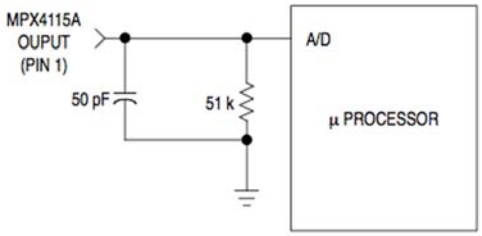


Figure 3. Decoupling Filter for Sensor to Microprocessor Interface

Figure 2 illustrates the absolute sensing chip in the basic chip carrier (Case 867). A fluorosilicone gel isolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the sensor diaphragm. The MPX4115A series pressure sensor operating characteristics, and internal reliability and qualification tests are based on use of dry air as the pressure media. Media, other than dry air, may have adverse effects on sensor performance and long–term reliability. Contact the factory for information regarding media compatibility in your application.

Figure 3 shows a typical decoupling circuit for interfacing the integrated MAP sensor to the A/D input of a microprocessor. Proper decoupling of the power supply is recommended.

Figure 4 shows the sensor output signal relative to pressure input. Typical, minimum, and maximum output curves are shown for operation over a temperature range of 0° to 85°C . (The output will saturate outside of the specified pressure range.)

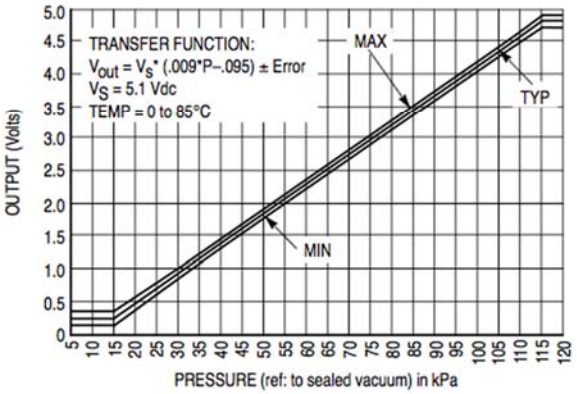


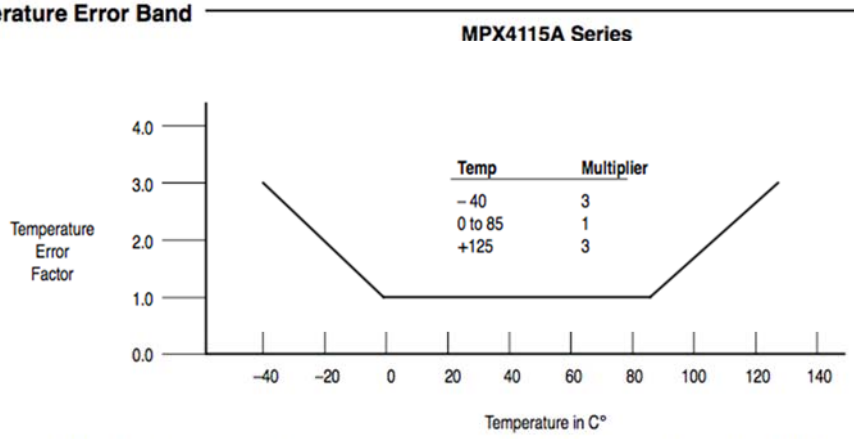
Figure 4. Output versus Absolute Pressure

MPX4115 SERIES

Transfer Function

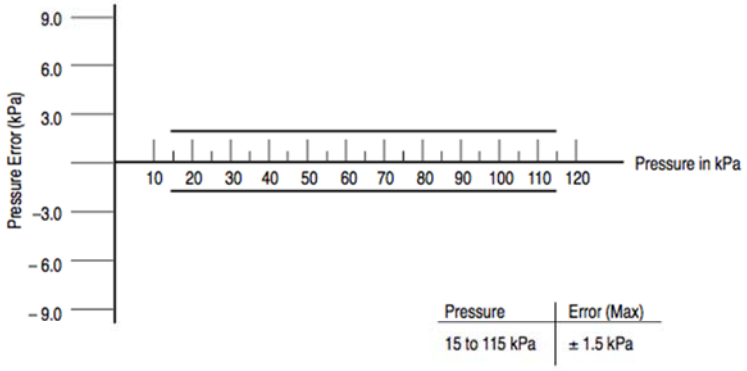
Nominal Transfer Value: $V_{out} = V_S (P \times 0.009 - 0.095)$
+/– (Pressure Error x Temp. Factor x 0.009 x V_S)
$V_S = 5.1\ V \pm 0.25\ Vdc$

Temperature Error Band



NOTE: The Temperature Multiplier is a linear response from 0° to –40°C and from 85° to 125°C.

Pressure Error Band

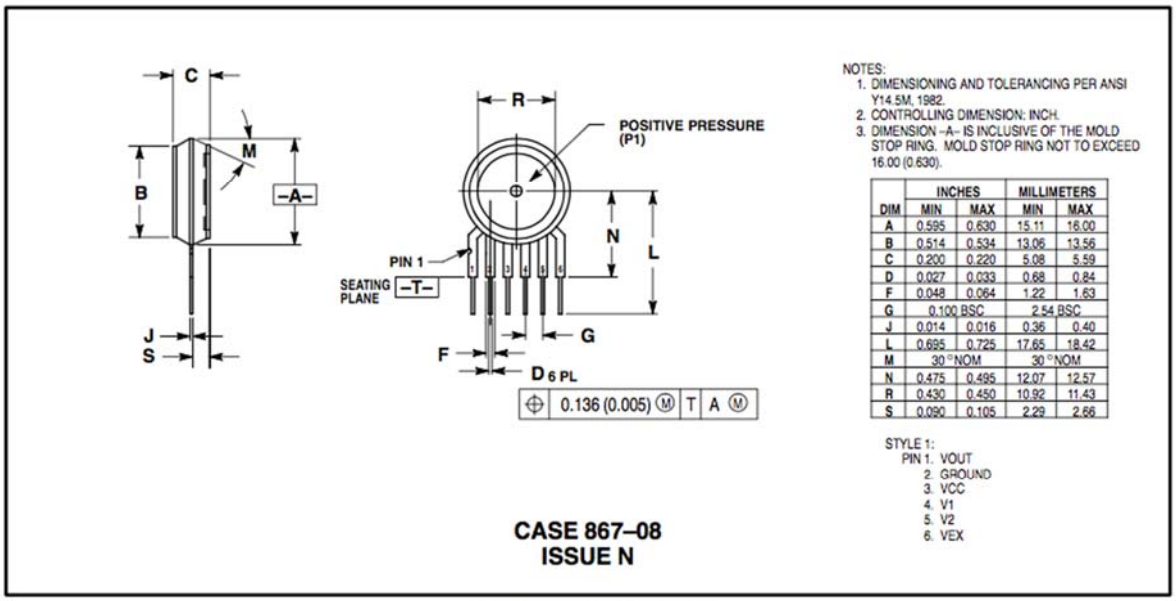


ORDERING INFORMATION

The MPX4115A BAP Sensor is available in the Basic Element package or with pressure port fittings that provide mounting ease and barbed hose connections.

Device Type	Options	Case No.	MPX Series Order No.	Marking
Basic Element	Absolute, Element Only	Case 867–08	MPX4115A	MPX4115A
Ported Elements	Absolute, Ported	Case 867B–04	MPX4115AP	MPX4115AP
	Absolute, Stove Pipe Port	Case 867E–03	MPX4115AS	MPX4115A
	Absolute, Axial Port	Case 867F–03	MPX4115ASX	MPX4115A

PACKAGE DIMENSIONS



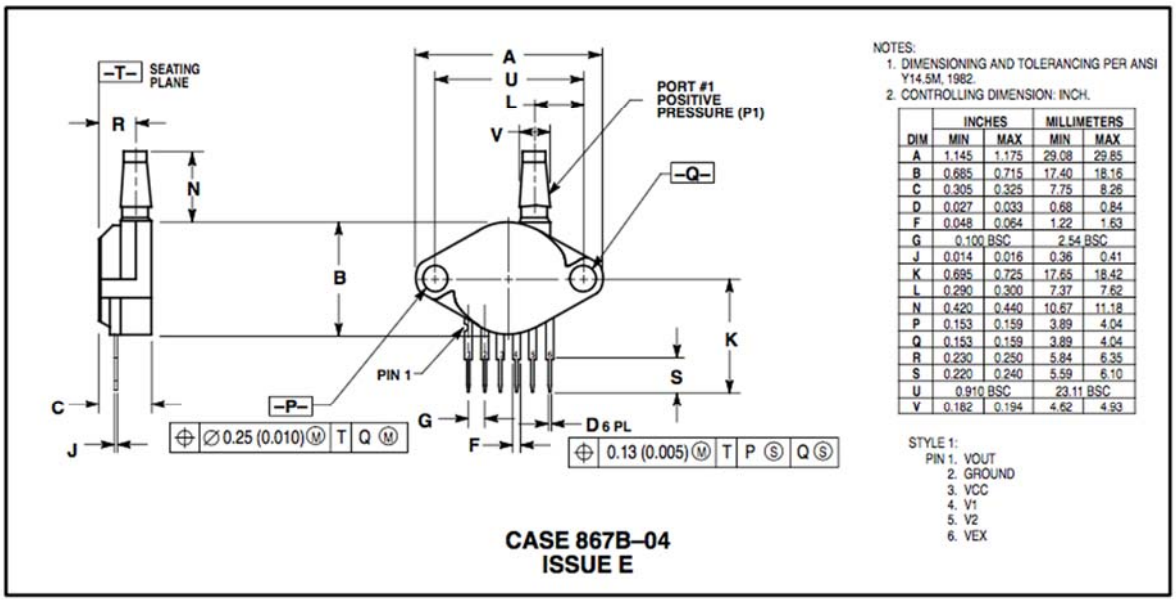
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION –A– IS INCLUSIVE OF THE MOLD STOP RING. MOLD STOP RING NOT TO EXCEED 16.00 (0.630).

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.595	0.630	15.11	16.00
B	0.514	0.534	13.06	13.56
C	0.200	0.220	5.08	5.59
D	0.027	0.033	0.68	0.84
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.40
L	0.695	0.725	17.65	18.42
M	30° NOM		30° NOM	
N	0.475	0.495	12.07	12.57
R	0.430	0.450	10.92	11.43
S	0.090	0.105	2.29	2.66

STYLE 1:
PIN 1. VOUT
2. GROUND
3. VCC
4. V1
5. V2
6. VEX

CASE 867–08
ISSUE N

BASIC ELEMENT (A, D)



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	1.145	1.175	29.08	29.85
B	0.685	0.715	17.40	18.16
C	0.305	0.325	7.75	8.26
D	0.027	0.033	0.68	0.84
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.41
K	0.695	0.725	17.65	18.42
L	0.290	0.300	7.37	7.62
N	0.420	0.440	10.67	11.18
P	0.153	0.159	3.89	4.04
Q	0.153	0.159	3.89	4.04
R	0.230	0.250	5.84	6.35
S	0.220	0.240	5.59	6.10
U	0.910 BSC		23.11 BSC	
V	0.182	0.194	4.62	4.93

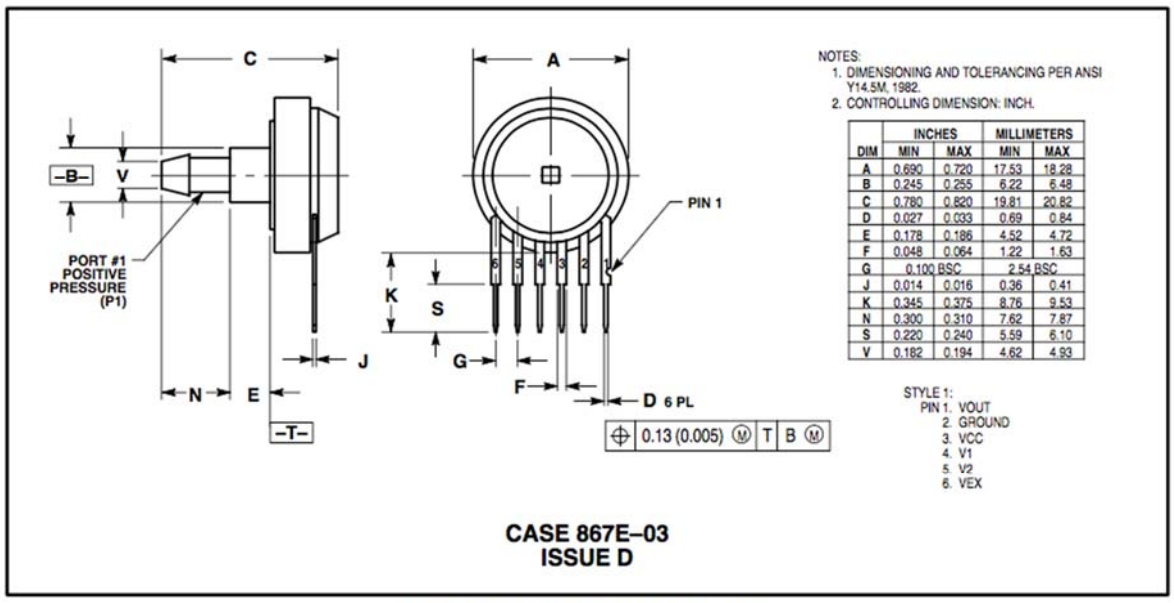
STYLE 1:
PIN 1. VOUT
2. GROUND
3. VCC
4. V1
5. V2
6. VEX

CASE 867B–04
ISSUE E

PRESSURE SIDE PORTED (AP, GP)

MPX4115 SERIES

PACKAGE DIMENSIONS—CONTINUED



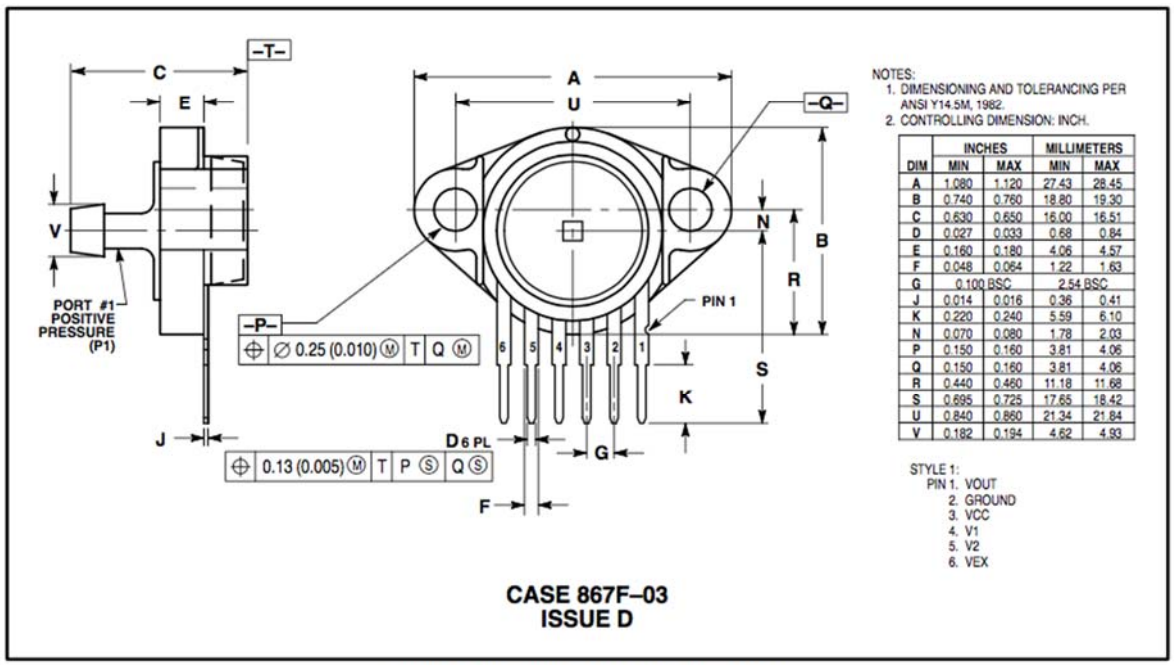
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	0.690	0.720	17.53	18.28
B	0.245	0.255	6.22	6.48
C	0.780	0.820	19.81	20.82
D	0.027	0.033	0.69	0.84
E	0.178	0.186	4.52	4.72
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.41
K	0.345	0.375	8.76	9.53
N	0.300	0.310	7.62	7.87
S	0.220	0.240	5.59	6.10
V	0.182	0.194	4.62	4.93

STYLE 1:
PIN 1. VOUT
2. GROUND
3. VCC
4. V1
5. V2
6. VEX

CASE 867E–03
ISSUE D

PRESSURE SIDE PORTED (AS, GS)



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	1.080	1.120	27.43	28.45
B	0.740	0.760	18.80	19.30
C	0.630	0.650	16.00	16.51
D	0.027	0.033	0.68	0.84
E	0.160	0.180	4.06	4.57
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.41
K	0.220	0.240	5.59	6.10
N	0.070	0.080	1.78	2.03
P	0.150	0.160	3.81	4.06
Q	0.150	0.160	3.81	4.06
R	0.440	0.460	11.18	11.68
S	0.695	0.725	17.65	18.42
U	0.840	0.860	21.34	21.84
V	0.182	0.194	4.62	4.93

STYLE 1:
PIN 1. VOUT
2. GROUND
3. VCC
4. V1
5. V2
6. VEX

CASE 867F–03
ISSUE D

PRESSURE SIDE PORTED (ASX, GSX)

Anexo G – Características técnicas del sensor de ultrasonidos

HC-SR04 Ultrasonic Range Finder

Manual

Features

- Distance measurement range: 2cm - 400cm
- Accuracy: 0.3cm
- Detect angle: 15 degree
- Single +5V DC operation
- Current comsuption: 15mA

Fig. 1

How It Works

HC-SR04 consists of ultrasonic transmitter, receiver, and control circuits. When trigged it sends out a series of 40KHz ultrasonic pulses and receives echo from an object. The distance between the unit and the object is calculated by measuring the traveling time of sound and output it as the width of a TTL pulse.

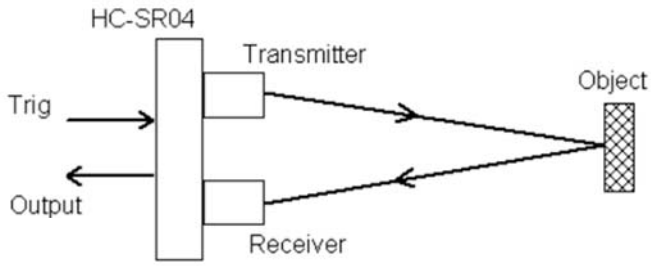


Fig. 2

How To Use It

To measure distance you need to generate a trig signal and drive it to the Trig Input pin. The trig signal leve must meet TTL level requirements (i.e. High level > 2.4V, low level < 0.8V) and its width must be greater than 10us. At the same time you need to monitor the Output pin by measuring the pulse width of output signal. The detected distance can be calculated by the formula below.

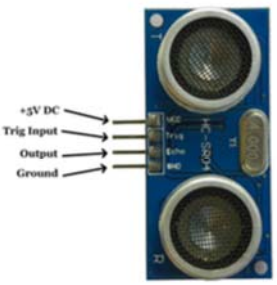


Fig.3

$$\text{Distance} = \frac{\text{Pulse Width} * \text{Sound Speed}}{2}$$

where the pulse width is in unit of second and sound speed is in unit of meter/second. Normally sound speed is 340m/s under room temperature.

www.AccuDIY.com

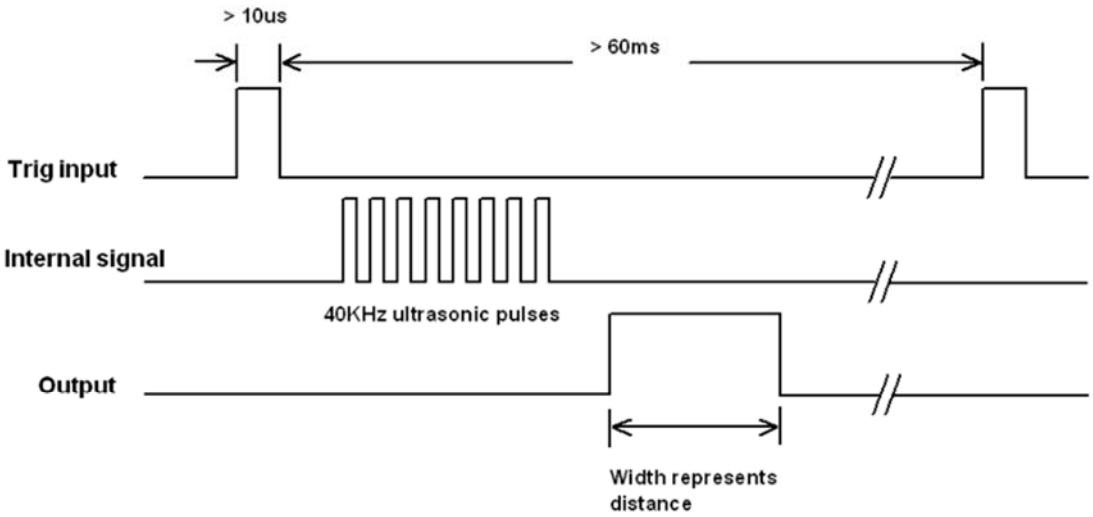


Fig. 4

Notes: 1. The width of trig signal must be greater than 10us
2. The repeat interval of trig signal should be greater than 60ms to avoid interference between connective measurements.

Specifications

Parameters	Specification
Operating Voltage	+5V DC
Operating Current	15mA
Operating Frequency	40KHz
Maximum Distance	400cm
Minimum Distance	2cm
Detect Angle	15 degree
Resolution	0.3cm
Input Trig Signal	>10us TTL pulse
Output Signal	TTL pulse with width representing distance
Weight	
Dimension	45 x 20 x 15 mm

Anexo H – Características técnicas del sensor de luz ambiente

Al no encontrar el datasheet del sensor utilizado, el ST-76, incluimos uno de funcionamiento similar, el N5AC-501085.

LDR CdS PHOTO CELL

SPECIFICATION

MODEL NO. N5AC-501085

Absolute Maximum Ratings

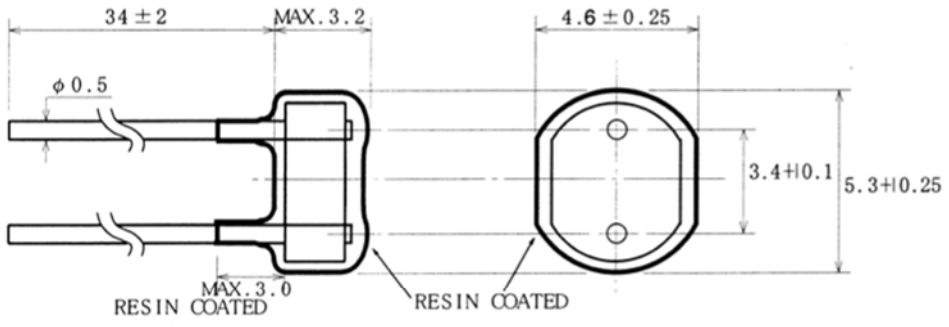
Allowable Power Dissipation (P_D) ····· 50 mW (Derate above 25℃ : 1.0 mW/℃)
Maximum Applied Voltage (V_{MAX}) ····· 100 V_{DC}
Ambient Temperature Range (T_A) ····· -30 ~ +60 ℃

Photo-electric Characteristics (at 25℃)

PARAMETER	SYMBOL	MIN.	MAX.	UNITS
Light Resistance at 10 Lux	R_L	50	100	kΩ
Gamma Value at 10~100 Lux	γ	0.85(Typ.)		——
Dark Resistance (10 sec. after shut off 10 Lux)	R_D	5	——	MΩ
Peak Spectral Response	λ_P	550	650	nm

※ Pre-measurement condition : Exposed in 500 lux for more than 3 hours.

Dimensions (unit : mm)



LDR CdS PHOTO CELL	PAGE 1/1	SPEC. NO. NCS-950203-01	REV. A
	• NIPPON CERAMIC CO., LTD.		
MODEL NO. N5AC-501085	APPROVED BY	CHECKED BY	