UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster

CURSO 2017/2018

DESARROLLO E IMPLEMENTACIÓN DE SENSORES INTELIGENTES EN REDES INALÁMBRICAS (STANDARD IEEE1451) BASADO EN UN SISTEMA EMBEBIDO.

Máster en Ingeniería Industrial

ALUMNO:

Luis Núñez Couselo

TUTOR:

José Luis Calvo Rolle

FECHA:

FEBRERO 2018

DESARROLLO E IMPLEMENTACIÓN DE SENSORES INTELIGENTES EN REDES INALÁMBRICAS (STANDARD IEEE1451) BASADO EN UN SISTEMA EMBEBIDO.

Este Trabajo Fin de Máster se centra en análisis y la aplicación del estándar de referencia IEEE1451 y en las redes de sensores inalámbricas (WSN) que se identifican como una de las tecnologías clave en el Internet de las Cosas (IoT), ya que proporcionan la flexibilidad necesaria para disminuir tiempos de instalación, recolección de datos y mantenimiento, y aportan mayor resolución espacial y temporal en el muestreo de las variables de interés a lo que se puede alcanzar con métodos tradicionales.

La revolución tecnológica actual basada en IoT y WSN ofrece enormes oportunidades para las iniciativas de negocio digital, a la vez que nuevos retos ante las demandas de diferente orden que se plantean por los usuarios, empresas y consumidores.

El principal objetivo que se persigue con este proyecto es realizar un ejemplo de aplicación práctica del desarrollo de sensores inteligentes y conocer las posibilidades que aporta la implementación del estándar IEEE1451. Para ello, se describen el proceso, la codificación, la transmisión de la información y el tratamiento de datos de los propios sensores.

DESENVOLVEMENTO E IMPLEMENTACIÓN DE SENSORES INTELIXENTES EN REDES SEN FIOS (STANDARD IEEE1451) BASADO NUN SISTEMA EMBEBIDO.

Este traballo Fin de Máster céntrase na análise e aplicación do estándar de referencia IEEE1451 e nas redes de sensores sen fíos (WSN) que se identifican como unha das tecnoloxías claves no Internet das Cousas (IoT), xa que proporcionan a flexibilidade necesaria para diminuír os tempos de instalación, recolección de datos e mantemento, e aporta maior resolución espacial e temporal na mostraxe das variables de interese ao que se pode acadar con métodos tradicionais.

A actual revolución tecnolóxica baseada en IoT e WSN ofrece enormes oportunidades para iniciativas empresariais dixitais, así como novos retos ás demandas de diferente orde que se plantexan polos usuarios, empresas e consumidores.

O principal obxectivo que se persegue con este proxecto e realizar un exemplo de aplicación práctica do desenvolvemento de sensores intelixentes e coñecer a as posibilidades que aporta a implementación do estándar IEEE1451. Para iso, descríbense o proceso, a codificación, a transmisión de información e o procesamento de datos dos sensores.

DEVELOPMENT AND IMPLEMENTATION OF SMART SENSORS IN WIRELESS NETWORKS (STANDARD IEEE1451) BASED ON EMBEDED SYSTEMS.

This Master's Project focuses on analysis and application of the IEEE1451 reference standard and on wireless sensor networks (WSN) identified as one of the key technologies of the Internet of Things (IoT), as they provide the flexibility to reduce installation times, data collection and maintenance, and greater spatial and temporal resolution in the sampling of the variables of interest to the reachable with traditional methods.

The current technological revolution based on IoT and WSN offers enormous opportunities for digital business initiatives, as well as new challenges to the demands of different order posed by users, companies and consumers.

The main objective of this project is to make an example of practical application of development of intelligent sensors and to know the possibilities of the implementation of the IEEE1451 standard. For this, the process, coding, transmission of information and data processing of the sensors are described.

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster
CURSO 2017/2018

INDICE GENERAL.

Máster en Ingeniería Industrial

ALUMNO:

Luis Núñez Couselo

TUTOR:

José Luis Calvo Rolle

FECHA:

FEBRERO 2018

1 INDICE GENERAL……6

2 MEMORIA....10
2.1 Objeto del proyecto.....10
2.2 Alcance.....10
2.3 Antecedentes y justificación de necesidades....10
2.4 Normas y referencias....11
2.4.1 Disposiciones legales y normas aplicadas....11
2.4.2 Bibliografía....11
2.4.3 Programas de cálculo e Informáticos utilizados en el proyecto....11
2.4.4 Otras referencias....12
2.5 Definiciones y abreviaturas....12
2.6 Marco estratégico del proyecto....12
2.7 Estructuración del trabajo fin de master....12
2.8 Orden de prioridad en los documentos....13
3.1 CONTEXTUALIZACIÓN DEL PROYECTO....15
3.1.1 Introducción a las redes de sensores....15
3.1.2 Clasificación de las topologías de red....16
3.1.3 Estandarización de las redes inalámbricas....19
3.1.4 Redes WSN....19
3.1.5 El IoT (Internet of Things)....20
3.1.6 Redes inteligentes....22
3.1.7 El sensor inteligente....23
3.2 EL ESTÁNDAR IEEE 1451....25
3.2.1 Introducción....25
3.2.2 IEEE 1451. Hacia la convergencia de las redes de sensores inteligentes....25
3.2.3 Inicios del estándar IEEE 1451....25
3.2.4 El estándar IEEE 1451....26
3.2.5 El estándar IEEE 1451.4....27
3.2.6 Sensores plug&play según el estándar IEEE 1451.4....28
3.2.6.1 Interfaz IEEE 1451.4 de Modo Mixto: Sensores Clase I y Clase II....29
3.2.6.2 Transducter Electronic Data Sheet (TEDS)....29
3.2.6.3 La estructura de los TEDS....31
3.2.6.4 Basic TEDS....31
3.2.6.5 Plantillas del estándar para los TEDS....34
3.2.6.6 Hardware para almacenamiento de los TEDS....35
3.2.6.7 La EEPROM DS-2431....35

3.3 DISEÑO Y CÁLCULO DE LOS SENSORES....37
3.3.1 Desarrollo de un sistema de Adquisición de datos inteligente conforme a la norma IEEE 1451.4....37
3.3.2 Implementación del Estándar IEEE1451....37
3.3.2.1 Selección del NCAP....37
3.3.2.2 Configuración del TIM....38
3.3.2.3 Programación....40
3.3.2.4 Pruebas de la EEPROM DS2431....45
3.3.3. Instalación de la memoria EEPROM en el sensor....46
3.3.4 Sensor RTD (PT1000)....46
3.3.4.1 Introducción....46
3.3.4.2 Características técnicas del sensor....50
3.3.4.3 Calibración y acondicionamiento del sensor....51
3.3.4.4 Simulación en OrCAD....53
3.3.4.5 Implementación del estándar IEEE 1451 en una PT1000....56
3.3.4.6 Montaje y resultados....64
3.3.4.7 Programación....68
3.3.5 Sensor LDR....89
3.3.5.1 Introducción....89
3.3.5.2 Características técnicas del sensor....91
3.3.5.3 Calibración del sensor....92
3.3.5.4 Simulación OrCAD....92
3.3.5.5 Implementación del estándar IEEE1451 en una LDR....96
3.3.5.6 Montaje y resultados....101
3.3.5.7 Programación....106
3.3.6 Sensor Termistor (NTC)....126
3.3.6.1 Introducción....126
3.3.6.2 Características técnicas del sensor....128
3.3.6.3 Calibración del sensor....129
3.3.6.4 Simulación OrCAD....129
3.3.6.5 Implementación del estándar IEEE1451 en una NTC....132
3.3.6.6 Montaje y resultados....138
3.3.6.7 Programación....142
3.4 PRUEBAS Y RESULTADOS FINALES....164
3.5 REDES INALAMBRICAS ZIGBEE....178
3.5.1 Introducción a las comunicaciones inalámbricas....178
3.5.2 Clasificación de las redes inalámbricas....179

3.5.3 Arquitectura Básica de una Red XBee 180
3.5.4 El módulo Xbee 181
3.5.5 Redes inalámbricas inteligentes con XBEE (Integración del IEEE1451) 181
3.6 INFORMACIÓN TÉCNICA DE LOS FABRICANTES 189
4 PLANOS 193

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster
CURSO 2017/2018

MEMORIA

Máster en Ingeniería Industrial

ALUMNO:

Luis Núñez Couselo

TUTOR:

José Luis Calvo Rolle

FECHA:

FEBRERO 2018

2 MEMORIA

2.1 Objeto del proyecto.

Este Proyecto presenta un caso de aplicación del estándar IEEE 1451, que consiste en el diseño de una red de sensores de modo que éstos funcionen con una alta capacidad para la autoidentificación, autodescripción y autodiagnóstico.

Lo que se espera es que la aplicación del estándar aporte un mayor grado de versatilidad a las comunicaciones entre sensores, así como que dependa en menor medida de la introducción manual de datos, y que por tanto, sea menos susceptible de error humano, y que todo ello pueda realizarse a un bajo coste.

De los resultados de esta aplicación, podrán sacarse conclusiones acerca del potencial y las limitaciones que este estándar presenta ante el incremento de la demanda del control inalámbrico del IoT.

Este Proyecto se redacta para la Escuela Politécnica Superior de Ferrol con domicilio en Calle Mendizábal s/n, Ferrol, con objeto de que sirva como Trabajo Fin de Máster para el alumno Luis Núñez Couselo.

2.2 Alcance.

El proyecto abarca los siguientes extremos:

- Análisis y aplicación del estándar IEEE1451
- Selección, diseño y desarrollo de sensores inteligentes para las redes inalámbricas de sensores actuales (WSN).
- Dotar a los sensores con la capacidad Plug&Play siguiendo el estándar de referencia IEEE1451
- Recolección y transmisión de los datos.
- Programación de los microcontroladores (Atmel).
- Muestra de resultados obtenidos.

2.3 Antecedentes y justificación de necesidades

El panorama actual se caracteriza por el protagonismo de los sensores específicos, ya que lo que se demanda es la medición, control y parametrización de todo tipo de variables (medioambientales, atmosféricas, fisiológicas,…) para necesidades concretas. Los sensores específicos precisan de protocolos de comunicación específicos, tanto por cable (RS232, USB,…) como inalámbricos (Bluetooth, Zigbee,…).

Esto, si bien tiene como ventaja facilitar una medición óptima, al mismo tiempo supone un obstáculo cuando resulta necesario que los diferentes sensores se comuniquen entre sí, o cuando se requiere comunicación entre las redes en las que se integran, por las incompatibilidad que se generan entre unos y otros.

Este inconveniente resulta perjudicial para el usuario final, en el sentido de que el sistema resulta poco versátil para sus necesidades. Con el estándar IEEE 1451 se alcanzó una solución a esta problemática, ya que implementa una comunicación estándar entre sensores de diferentes tecnologías para que sean capaces de funcionar juntos.

Con este trabajo se ha realizado un ejemplo práctico con Arduino para implementar y demostrar dicho estándar en diferentes sensores y dotarlos de capacidad de auto identificación en las futuras redes inalámbricas que trabajen bajo este estándar.

2.4 Normas y referencias

2.4.1 Disposiciones legales y normas aplicadas

En la redacción de este Proyecto se han tenido en cuenta todas y cada una de las especificaciones contenidas en las Reglamentaciones y Normas que se relacionan a continuación:

- Norma UNE 157001 de Criterios Generales para la elaboración de proyectos.
- IEEE 1451 y IEEE 1451.4-2004 IEEE Stand para una interfaz de transductor inteligente para sensores y actuadores - Protocolos de comunicación en modo mixto y formatos de TEDS
- Normas UNE de dibujo técnico :
  - Norma UNE-EN 1035-95, Cuadro de rotulación
  - Norma UNE-EN 1039-94, Acotación
  - Norma UNE-EN ISO 5455-96, Escalas

2.4.2 Bibliografía

- Estándar IEEE 1451, Standard for Smart Transducers
- Estándar IEEE 1451.4, IEEE Standard for A Smart Transducer Interface for Sensors and Actuators, Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.
- Estándares para interoperabilidad de redes de sensores: IEEE 1451 y Sensor Web Enablement (SWE), Universidad de la Coruña (UDC), año 2014.
- Proyecto, Estudio y elaboración de una guía de implementación del estándar ieee 1451 para redes inalámbricas. Universidad de Navarra, año 2010

También se han realizado consultas en las siguientes páginas Web:

- **www.arduino.cc/** (Página oficial de ARDUINO. Fecha de acceso 25-10-2017)
- **bibing.us.es/proyectos/abreproy/11904/fichero/1-CAPITULO1.pdf** (Introducción al Estándar 1451, fecha de acceso 11-8-2017)
- **http://www.ni.com/white-paper/3469/es/** (Información General sobre el Estándar IEEE 1451.4 de NATIONAL INSTRUMENTS, fecha de acceso 11-8-2017)
- **standards.ieee.org/develop/regauth/tut/** (Guías sobre la implementación del Estándar IEEE 1451.4, fecha de acceso 4-12-2017)
- **standards.ieee.org/develop/regauth/manid/** (Registro del IEEE 1451.4)
- **standards.ieee.org/develop/regauth/manid/public.html** (Lista Pública de Fabricantes con IEEE 1451.4, fecha de acceso 5-12-2017)
- **https://aprendiendoarduino.wordpress.com/tag/xctu/** (Información sobre XCTU y módulos XBEE, fecha de acceso 4-1-2018)

2.4.3 Programas de cálculo e Informáticos utilizados en el proyecto

En la redacción de este proyecto se han utilizado las herramientas informáticas y programas de cálculo que se indican a continuación:

- MICROSOFT WORD 2010 para tratamiento de textos.
- MICROSOFT EXCELL 2010 para hojas de cálculo
- Orcad Pspice Lite 16.6 para diseño y simulación de circuitos.
- ARDUINO 1.8.3 como software de programación de los microcontroladores ATMEL

2.4.4 Otras referencias

Catálogos y hojas de datos de los elementos instalados:

- Datasheet ATmega48PA/88PA/168PA/328P
- Datasheet Platinum Sensing Resistors – Thin Film (Pt100 & Pt1000 Ohm)
- Datasheet GL5528 LDR
- Datasheet NTC THERMISTOR OF MF52-TYPE SERIES SPECIFICATION
- Datasheet Amplificador Operacional uA741
- Datasheet Amplificador de Instrumentación AD623
- Datasheet EEPROM DS2431
- Información técnica sobre Arduino UNO y Xbee

2.5 Definiciones y abreviaturas

A todo lo largo del Proyecto se utilizan una serie de abreviaturas para simplificar la lectura. La primera vez que se utilice una abreviatura, se hará entre paréntesis siguiendo a la palabra que, en lo sucesivo, va a sustituir.

2.6 Marco estratégico del proyecto

Este proyecto no se encuentra en vigor, y se integraría en el Marco Europeo para la Investigación e Innovación, dentro del reto Horizonte2020. Dentro de los desafíos de este, se centra en los de la “Economía y sociedad digital”.

De lo que se trata con este proyecto es de impulsar el desarrollo de la economía digital, reducir costes, fortalecer el sector TIC español como fuente de generación de riqueza y empleo, e impulsar el I+D+i en las industrias del futuro.

Este proyecto tiene su aplicación como mejora de cualquier proyecto actual en el ámbito del IOT basado en redes inalámbricas, como por ejemplo:

- Smart Cities.
- Redes de monitorización de cultivos.
- Parkings inteligentes.
- Industrias del futuro conectadas.
- Gestión de residuos urbanos.
- Etc.

2.7 Estructuración del trabajo fin de master

A lo largo del presente Proyecto se han expuesto los fundamentos que han servido de base para la realización del mismo. Quedan, a juicio del autor del proyecto, suficientemente claros los detalles de características, tipo de materiales y los procedimientos a llevar a cabo.

El resultado final será el desarrollo y estudio de sensores inteligentes según el estándar de referencia IEEE1451 para redes inalámbricas.

El Proyecto se estructura en varias unidades de modo que los distintos documentos básicos, con sus documentos unitarios, son los que se relacionan a continuación:

- Índice General
- Memoria General
- Redes de sensores inteligentes. Análisis y resultados:

  - Contextualización del proyecto
  - Estándar IEEE1451
  - Diseño y cálculo de los sensores
  - Pruebas y resultados finales
  - Redes inalámbricas Zigbee
- Planos relativos a:
  - Circuito y acondicionamiento PT1000
  - Circuito y acondicionamiento LDR
  - Circuito y acondicionamiento NTC

Cada una de estas unidades se estudia separadamente en el anexo correspondiente, al final del cual se incluirán las tablas de soluciones que se consideren necesarias.

2.8 Orden de prioridad en los documentos

En relación con las posibles discrepancias entre los documentos básicos del Proyecto, el orden de prioridad es el que viene indicado de forma general en la UNE 157001, sin más consideraciones, es decir:

1. ESTÁNDAR IEEE1451
2. DISEÑO Y CALCULOS DE LOS SENSORES
3. PRUEBAS Y RESULTADOS FINALES
4. PLANOS

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster
CURSO 2017/2018

REDES DE SENSORES INTELIGENTES. ANÁLISIS Y RESULTADOS FINALES

Máster en Ingeniería Industrial

ALUMNO:

Luis Núñez Couselo

TUTOR

José Luis Calvo Rolle

FECHA

FEBRERO 2018

3.1 CONTEXTUALIZACIÓN DEL PROYECTO

3.1.1 Introducción a las redes de sensores

Una red de sensores consiste en un conjunto de nodos capaces de sensar una variable física, convertirla en datos digitales y comunicar dichos datos a otro nodo adyacente, o a un nodo central donde pueden ser procesados y convertidos en información útil.

Figura 3.1.1.1: Redes de sensores

El desarrollo y puesta en marcha de una red de sensores requiere abordar varias cuestiones:

En primer lugar, solucionar el suministro de energía requerida por los nodos, tanto para el funcionamiento de cada uno de los sensores como para la propia transmisión de los datos entre ellos. Para evitar que la batería de cada nodo se agote, y con ello se frustre la recolección de información, se recurre al uso del protocolo Zigbee y a los paneles solares como sistema de recuperación de carga.

En segundo lugar, optar por una de las dos alternativas tecnológicas actualmente disponibles, la inalámbrica y la cableada, en función de lo que más se adapte a la topología de la red que se está diseñando. El sistema cableado tiene el inconveniente de una menor movilidad, mientras que hay soluciones basadas en sistemas GPS que permiten que una red de sensores inalámbrica transmita datos cualquiera que sea su ubicación, con capacidad de cubrir todo el planeta, monitorizando la información constantemente, con movilidad absoluta y capacidad de crecimiento indefinida.

Figura 3.1.1.2: Red inalámbrica mundial

Actualmente la implementación de las redes de sensores sin hilos está arrancando, ya que se encuentra en fase de investigación. La gran mayoría de investigaciones hasta el día de hoy se han centrado en temas tales como la monitorización ambiental, el impacto de terremotos sobre grandes estructuras, e incluso en la anticipación de desastres naturales, sin mencionar por supuesto las aplicaciones militares que han sido y serán siempre las pioneras. Dentro de las aplicaciones más estrictamente relacionadas con el sector industrial, el mayor desarrollo se ha dado en los ámbitos de la monitorización del estado de los equipos industriales en las plantas, en aplicaciones relacionadas con la domótica o en la agricultura de precisión. En el sector servicios, destacan las implantaciones en la monitorización de pacientes en hospitales y en las aplicaciones para ciudades inteligentes.

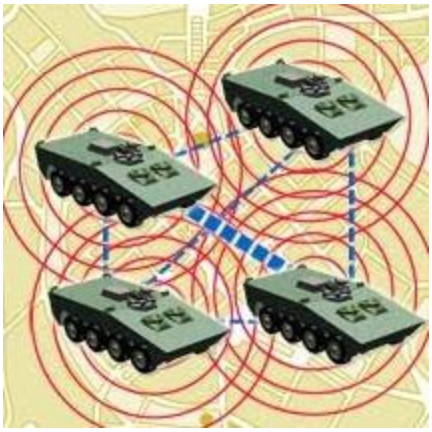


Figura 3.1.1.3: Red inalámbricas militares

En resumen, es de destacar que continúa el constante desarrollo relacionado con las redes inalámbricas de Internet, al objeto de conseguir mayores velocidades de transmisión y tecnologías óptimas que incrementen la conectividad, la seguridad y la fiabilidad de la transmisión de los datos.

3.1.2 Clasificación de las topologías de red

Siendo la topología de una red de sensores la disposición de sus nodos y la de las líneas de conexión entre ellos, existen dos modos de definición de la geometría de una red, la topología física y la topología lógica o de señal:

Topología física: hace referencia a la disposición geométrica de las estaciones de la red, a los cables que la conectan y al trayecto que siguen las señales a través de la conexión física.

Topología lógica: es la trayectoria lógica que sigue una señal a su paso por los nodos de la red.

Las topologías de red básicas son las de tipo árbol, malla, anillo y estrella, y cuando una red combina una o más de estas topologías básicas se dice que estamos ante una topología híbrida:

Malla: en una red de tipo malla, cada uno de los nodos está conectado a todos los demás nodos que componen la red, de modo que las señales pueden viajar entre dos nodos concretos siguiendo diferente camino cada vez.

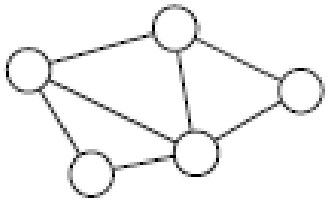


Figura 3.1.2.1: Red tipo Malla

Estrella: en la topología básica en estrella, las estaciones están conectadas directamente a un mismo nodo central, de manera que todas las comunicaciones tienen que pasar obligatoriamente a través de ese punto. La ventaja de este tipo de red es su baja latencia y el bajo consumo de energía que realiza cada nodo, por lo que resulta idónea cuando el alcance de la red es corto. Su desventaja es que dicho alcance no llega a los 100 metros de distancia, lo que puede ser insuficiente en muchos casos.

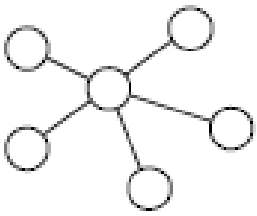


Figura 3.1.2.2: Red tipo Estrella

Árbol: La red en árbol es una topología de red en la que los nodos están colocados en forma de árbol, es decir, las comunicaciones entre nodos se realiza de forma ramificada.

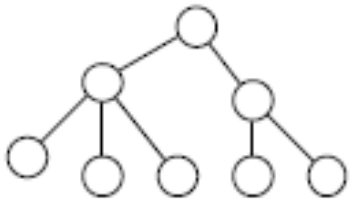


Figura 3.1.2.3: Red tipo Árbol

Bus: una red en bus es una topología que tiene un único canal de comunicaciones (denominado bus, troncal o backbone) al cual se conectan los diferentes dispositivos.

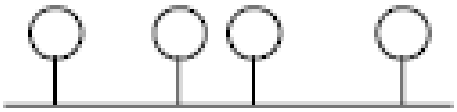


Figura 3.1.2.4: Red tipo Bus

Anillo: una topología de red en anillo se caracteriza porque cada estación tiene una única conexión de entrada y otra de salida, es decir, un transmisor y un receptor que a la vez hace la función de traductor, pasando la señal a la siguiente estación.

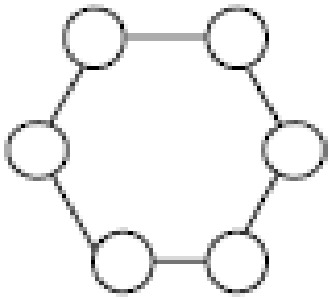


Figura 3.1.2.5: Red tipo Anillo

Respecto a los nodos, existen 3 tipos diferentes:

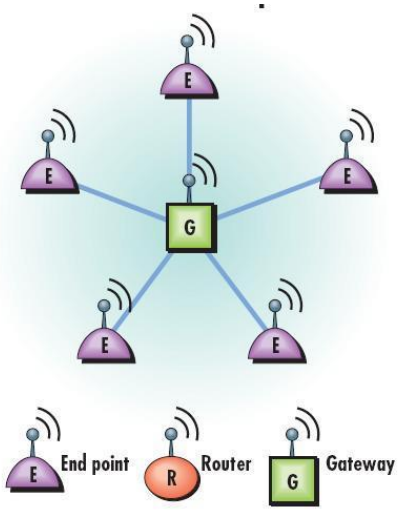


Figura 3.1.2.6 Tipos de Nodos

Nodo Pasarela (denominado *gateway*): es un sistema de hardware o software que permite interconectar redes que utilizan arquitecturas completamente diferentes a fin de que puedan intercambiar información entre ellas.

Nodos Intermedios (denominados *routers*): son nodos capaces de extender el alcance de la red, sortear obstáculos a la transmisión sin hilos y proveer rutas alternativas para el tránsito de mensajes que se envían al *gateway*.

Estos nodos pueden alimentar su funcionamiento a partir de baterías o de otro tipo de fuentes de energía con mayor continuidad, decisión que dependerá de la estructura con la que se dote la red y del propósito de la misma y del de cada uno de los nodos.

Nodos Finales: son dispositivos de funciones reducidas, que se limitan a sensar la información y transmitirla al siguiente nodo de la red, por lo que constituyen la interfaz entre la magnitud física a sensar y la red. No reciben información vía radio, ni por tanto tienen que retransmitirla. Los nodos finales se caracterizan por disponer de varias conexiones de entrada-salida para comunicarse con sensores o actuadores, y habitualmente se opta por alimentarlos con baterías para dotarlos de autonomía.

Sensores y actuadores: son los dispositivos encargados de transformar en una señal eléctrica aquella señal física que se quiere medir, almacenar y tratar. Pueden ser de muy variada tipología, pues actualmente existen opciones para medir todo aquello que resulte necesario.

3.1.3 Estandarización de las redes inalámbricas

Las primeras redes inalámbricas eran únicamente válidas para la situación para la que habían sido concebidas, ya que no disponían de la tecnología que facilitase la interoperabilidad, sino tan solo de un protocolo propietario que aportaba una solución específica a la problemática concreta. Estas redes ad hoc estaban constituidas por nodos capacitados para reenviar datos a los demás nodos de la red en función de la conectividad de esta.

Estos límites consustanciales a la especificidad de las primeras redes inalámbricas fueron superándose con la evolución a las primeras WPANs y WSNs (Wireless Sensor Network), en el caso de los sensores, pasando de soluciones particulares a soluciones estandarizadas. La estandarización de las redes inalámbricas WSN fue responsabilidad del IEEE (Institutye of Electrical and Electronics Engineers), institución redactora de una norma universal para las redes, con multitud de aplicaciones en el ámbito doméstico, empresarial, de servicios, e incluso en el campo de la operación de redes de telecomunicaciones.

3.1.4 Redes WSN

El concepto de WSN puede definirse como una red inalámbrica que está formada por dispositivos autónomos distribuidos espacialmente, que monitorizan condiciones físicas y ambientales mediante sensores.

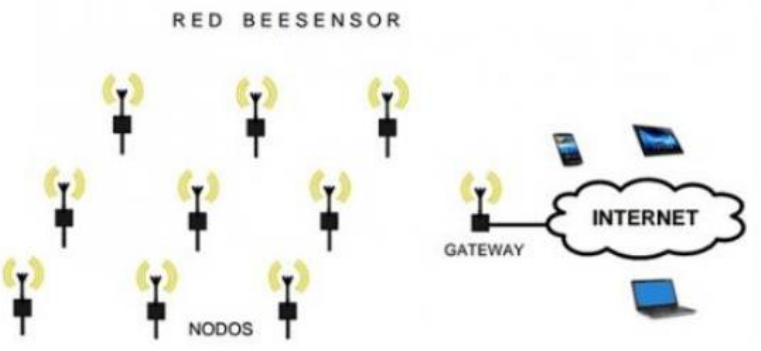


Figura 3.1.4.1: Redes WSN

Cada dispositivo autónomo o nodo está formado por un microcontrolador, sensores y transmisor/receptor, y en conjunto con otros nodos forma una red. Un sensor por sí solo no es capaz de procesar más que una limitada cantidad de datos, pero un grupo de nodos coordinados entre sí tienen suficiente capacidad para medir un medio físico con gran detalle.

Se ha producido una evolución desde las primeras redes de sensores, que consistían en un reducido número de nodos conectados por cable a una estación central de procesamiento de datos.

Hoy en día, los desarrollos en este campo están centrados en redes de sensores distribuidos e inalámbricos que monitorizan diferentes variables físicas, redes WSN capaces de realizar con mayor precisión su tarea específica, cuanto mayor sea el número de sensores o motas desplegadas, y cuanta mejor coordinación se establezca entre ellas.

Actualmente, la posibilidad de utilizar un conjunto amplio de sensores inalámbricos hace posible establecer monitorizaciones en ambientes hostiles a otras formas de transmisión de las comunicaciones, como aquellos en los que existen obstáculos físicos u obstrucciones y cortes de línea. También son óptimos para situaciones en las que el suministrar energía es difícil, ya que los nodos, además de poder funcionar con baterías, consumen menos energía a través de largas distancias en un sistema de comunicación distribuido como es el que caracteriza a las WSN.

El panorama tecnológico actual permite conformar una red distribuida con dispositivos de tamaño diminuto, de bajo coste, y capaces tanto de procesar información como de comunicarse de forma inalámbrica. Además, son de bajo consumo energético tanto cuando operan activamente como cuando permanecen en espera, por lo que son capaces de permanecer disponibles durante largos periodos de tiempo.

Los elementos de una WSN típica son:

- **Sensores**: convierten una magnitud física en señales eléctricas.
- **Nodos (Motas)**: toman datos del sensor y remiten la información a la estación base.
- **Gateway:** establecen la interconexión entre la red de sensores y una red de datos (TCP/IP).
- **Estación base o Recolector de datos**: basado en un ordenador común, un PLC o a un sistema embebido.

3.1.5 El IoT (Internet of Things)

El Internet de las cosas (*Internet of things*, abreviado IoT) tiene dos definiciones complementarias, que resumen a la perfección la magnitud que tiene como oportunidad tecnológica para el sector industrial.

En primer lugar, el IoT es un sistema basado en máquinas y objetos equipados con tecnologías de recopilación de datos, de modo que pueden comunicarse entre sí a través de internet. Es decir, su finalidad es que el mayor número de cosas de la vida cotidiana tengan incorporados dispositivos que permitan que otros equipos las identifiquen y gestionen, de modo similar a como las identifican y gestionan las personas.

En segundo lugar, el IoT es la conexión de internet en la que existen más máquinas u objetos que personas, por lo que también se la denomina el *Internet de todas las cosas* o *Internet en las cosas*. Su finalidad es que estén conectados el máximo de objetos posible, entre sí y también con las personas.

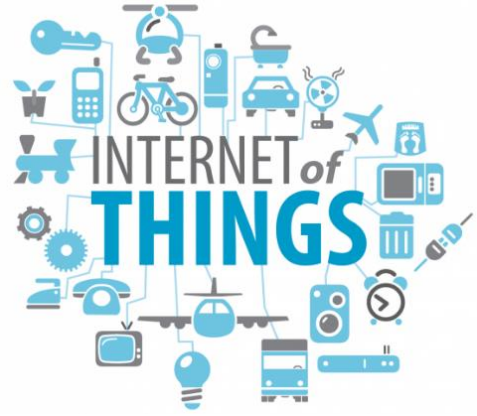


Figura 3.1.5.1: Internet of Things

La propuesta del IoT fue planteada en 1999 por Kevin Ashton, un empresario e ingeniero británico creador del Centro de Investigación Auto-ID de la estadounidense Universidad Tecnológica de Massachusetts (MIT). Según su propuesta, el IoT tenía "el potencial para cambiar el mundo tal y como lo hizo Internet", mediante la conexión de todos los equipos electrónicos que nos rodean, la medición de los parámetros externos a ellos, permitiendo así la automatización de muchas actividades realizadas por el ser humano hasta el momento.

Además de las ventajas tecnológicas que se desprenden de esta propuesta, para la industria resulta de enorme interés debido a las cifras económicas relacionadas con su coste de implementación y el nivel de inversión esperado: para el año 2020, se espera que el coste de los componentes electrónicos que requiere el IoT baje por debajo de un dólar, mientras que las inversiones en el IoT se prevé que asciendan a 1.9 trillones de dólares. Además, se estima que en 2020 podrían existir más de 20.000 millones de dispositivos electrónicos conectados a la Red, unos 3 por cada ser humano del planeta.

Figura 3.1.5.2: *Smart Cities*

La medición de parámetros externos (energía, velocidad temperatura, humedad, actividad física, luz, etc.) de forma automática y sin la interacción del ser humano puede abordarse mediante la utilización de las tecnologías del IoT, que además hace posible que los datos recabados sean transmitidos a un centro de procesamiento en el cual su análisis automático permite la toma de decisiones adecuadas.

El ejemplo más popular a día de hoy es el de los proyectos de Smart Cities, basados en la diseminación de sensores en multitud de puntos fijos y agentes móviles, como vehículos, semáforos, alcantarillas, alumbrado, alarmas,… Con ello, se espera la mejora de la dinámica urbana, por ejemplo, mediante la cuantificación de los peatones que pasan por un determinado lugar para optimizar automáticamente el tráfico en esa zona.

Existen suministradores de dispositivos en internet, pudiendo adquirirse botones, sensores de ultrasonidos, sensores de luz o de distancia, sensores táctiles, acelerómetros sensores de inclinación, potenciómetros, sensores de humedad y temperatura, de altitud, de presión,… todo ello, y cualquier otra cosa que pueda medirse, para la plataforma Arduino.

Las empresas relacionadas actualmente con el IoT también tienen capacidad para diseñar y fabricar sus propios sensores, diferentes a los mencionados para Arduino, de modo que van respondiendo a las necesidades del mercado con sensores ad hoc diseñados para satisfacerlas convenientemente.

En lo que se refiere a este Proyecto, hay que mencionar que entre los diferentes protocolos de comunicación utilizados en IoT, destaca el estándar IEEE 1451, que permite conectar transductores, sensores y actuadores independientes de la red. Sin embargo, la falta de hardware compatible actualmente con este estándar implica que las publicaciones al respecto no hayan gozado de crecimiento suficiente. Por ello, existe espacio disponible para las propuestas relacionadas con la aplicación del estándar a la comunicación entre los diferentes dispositivos disponibles en el mercado.

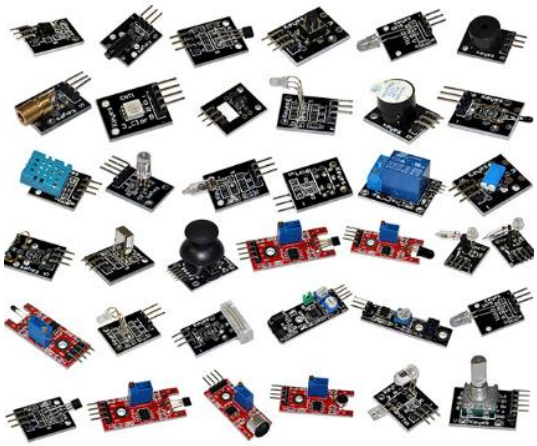


Figura 3.1.5.3: Sensores para Arduino

3.1.6 Redes inteligentes

El origen de las redes de sensores está en proyectos de naturaleza militar, por lo cual no existe mucha información al respecto, ni sobre su evolución inicial. Hacia 1980, se comenzó la investigación en redes de sensores con el proyecto Distributed Sensor Networks (DSN), de la Agencia Militar de Investigación Avanzada de los Estados Unidos.

Posteriormente, se continuó el desarrollo tecnológico a nivel empresarial, con la creación de redes de transductores inteligentes (sensores y actuadores) como solución óptima y económica para una enorme variedad de aplicaciones de medición y control de parámetros de todo tipo.

El futuro desarrollo de estas tecnologías está garantizado, ya que se están convirtiendo en un elemento fundamental en todos los ámbitos de producción de los países industrializados, transformando radicalmente el mercado de los negocios e incrementando la demanda de los mismos en diversos ámbitos y novedosos usos.

Aunque a día de hoy los sensores ya están siendo utilizados en la mayoría de las industrias, en especial, en aquellas más intrínsecamente ligadas a la tecnología como la aeroespacial, la de la automoción, la construcción, el control industrial y la producción de objetos en general, su uso se incrementa sin parar, y con él, la demanda del mercado. Por ello, los fabricantes de sensores indagan métodos de construcción de sensores a un menor coste para bajar su precio, al mismo tiempo que tratan de compatibilizar una mayor facilidad de uso con un diseño capaz de responder a necesidades industriales nuevas y más sofisticadas.

Figura 3.1.6.1: Redes inteligentes

Como se ha adelantado, es aquí donde aparece la necesidad de un estándar que comunique redes de diferente finalidad, creadas por empresas diferentes, que las han dotado de protocolos de comunicación propios y específicos, eficaces pero incompatibles entre sí, por lo que el uso combinado de varias de estar redes se mantiene en un estado de incertidumbre y confusión.

Lo deseable, es que un sensor inteligente sea capaz de actuar adecuadamente bien como un sensor aislado, bien relacionándose con otro sensor o actuador de la misma red, o bien como un nodo inteligente dentro de la red (Randy, 2000). Esta interoperabilidad entre nodos de diferentes redes, puede conseguirse mediante la implementación de varias interfaces que operen con ellas.

Es necesario que existan distinto tipo de redes, específicas y por ello óptimas para abordar situaciones y problemáticas diferentes, pero el sector industrial también ha sido consciente de la necesidad de lograr la interoperabilidad entre ellas, clarificando la definición, funcionalidad y estándares de comunicación de los sensores inteligentes.

Por ello, el Comité Técnico de Tecnología de Sensores del IEEE ha patrocinado una serie de proyectos para abordar esta necesidad, denominados en conjunto IEEE P1451, y que consisten en el desarrollo de una familia de normas de interfaces transductores inteligentes para conectar a las redes.

La estandarización de estas interfaces contiene unas especificaciones comunes para que los fabricantes las sigan en sus diseños específicos de transductores y de conectividad con las redes, lo que redunda en favor de una menor incertidumbre, del más rápido desarrollo de sensores y actuadores inteligentes, así como en el abaratamiento de su fabricación y, por tanto, en su expansión en el mercado.

3.1.7 El sensor inteligente

Un sensor es un transductor que convierte una magnitud física (temperatura, fuerza, luminosidad…) en una magnitud eléctrica (voltaje, corriente, impedancia).

Los Sensores Inteligentes (*Smart sensors*), son dispositivos que dan a la señal analógica recogida el formato que necesita, es decir, facilitan una interfaz con el sistema de medida más adecuado. Por tanto, además de la función de detección realiza funciones de procesamiento y comunicación mediante un microprocesador o microcontrolador integrados, motivo por el cual es denominado sensor inteligente.

En la mayoría de ocasiones, la intefaz que aporta el microprocesador o microcontrolador son específicos, creados ad hoc por una compañía determinada para una necesidad determinada, de modo que son muy efectivos en el cumplimiento de su finalidad, pero también son poco versátiles.

Figura 3.1.7.1: Sensores inteligentes

En todo caso su coste de adquisición será mayor que el de un sensor convencional, a lo que hay que añadir los costes de mantenimiento, fiabilidad, etc. Por tanto, su utilización debe venir justificada por no ser adecuados los sensores convencionales para alcanzar el objetivo establecido, o para solucionar la problemática a abordar.

La información que típicamente ofrecen los sensores inteligentes, es:

- Datos de la magnitud medida.
- Información sobre el contexto de interpretación de los datos proporcionados.
- Información de calibración.

Debido a su finalidad específica, el nivel de complejidad de un sensor inteligente puede ser muy variado, y por ello se habla de distintos grados de "inteligencia" del sensor, en función de las funciones que incluya:

- Adaptación y acondicionamiento.
- Conversión A/D
- Comunicaciones/Transmisión de la información

Las características deseables que debe tener un sensor inteligente son:

- Auto-identificación.
- Auto-descripción.
- Auto-diagnóstico.
- Auto-calibración.
- Capacidad de coordinación con otros nodos.
- Procesamiento de datos.
- Notificación de alertas.
- Fácil instalación.

3.2 EL ESTÁNDAR IEEE 1451

3.2.1 Introducción

El desarrollo de soluciones complejas de sensorización y control se ha encontrado con el límite de la heterogeneidad de sensores de diferentes fabricantes, que operan en diferentes ámbitos y no definen el sistema de interoperabilidad con otras redes, plataformas y aplicaciones.

El estándar IEEE 1451 ha venido a facilitar una solución al problema de interoperabilidad a nivel de sensores y actuadores, mediante la definición de una interfaz común que permite el funcionamiento de transductores inteligentes con capacidades de auto-identificación, auto-descripción y auto-diagnosis.

Es decir, la familia de estándares IEEE 1451 define el sistema de identificación de las características de cada sensor, los protocolos para transferir información que resultan aplicables a todos los sensores, los formatos de los datos y las operaciones posibles sobre sensores y actuadores.

De este modo, es posible desarrollar modelos universales de redes abstrayéndonos del fabricante, ya que no es necesario un transductor específico para cada una de las opciones del mercado. El resultado es que se simplifican al máximo las tareas de instalación y gestión de nodos sensores, haciendo posible la implementación de soluciones 100% *plug&play*, lo que incrementa la aceptación de esta tecnología por parte de usuarios inexpertos, y por tanto resulta en un incremento de uso de la misma.

3.2.2 IEEE 1451. Hacia la convergencia de las redes de sensores inteligentes

Como se ha expuesto, el problema en esta cuestión radica en que cada una de las redes de sensores inteligentes cuyos datos son susceptibles de consulta a través de Internet, utiliza sus propios estándares, protocolos y formatos de representación de datos.

Por tanto, las características deseables de los sensores de la red son:

- Auto-identificación.
- Auto-diagnóstico.
- Confiabilidad.
- Capacidad de coordinación con otros nodos.
- Funciones software y de tratamiento digital de la señal.
- Protocolos de control y de interfaz de red estándares
- Fácil instalación.

3.2.3 Inicios del estándar IEEE 1451

En conjunto, la familia de estándares IEEE 1451 está concebida para trabajar de forma sincronizada facilitando la conexión de sensores y actuadores en un dispositivo o en una red. Su origen está en la reunión celebrada en el mes de septiembre de 1993 entre el Instituto Nacional de Estándares y Tecnología (NIST) y el Comité del IEEE, que giró en torno a la posibilidad de crear una interfaz estándar en las comunicaciones de sensores inteligentes.

Para llevar a cabo este objetivo, se acordó la creación de una interfaz común de comunicación entre transductores inteligentes, resolviéndose la formación de cuatro grupos de trabajo iniciales, a los que se añadieron otros posteriormente. Los grupos de trabajo iniciales fueron denominados P1451.1, P1451.2, P1451.3 y P1451.4.

Norma		Estado
1451.0	TEDS y Comandos Comunes	Aprobado (2007)
1451.1	Interfaz NCAP - Computador	Aprobado (1999) - Requiere revisión
1451.2	Interfaz punto a punto (TII)	Aprobado (1997) - Requiere revisión
1451.3	Buses Multi-Drop	Aprobado (2003) - Requiere revisión
1451.4	Interfaz Analógico	Aprobado (2004)
1451.5	Wireless (WiFi, Zigbee, etc)	Aprobado (2007)
1451.6	CAN Bus	Pendiente
1451.7	RFID	Pendiente

Familia IEEE 1451.0

Figura 3.2.3.1 Familia IEEE 1451 (8 normas)

La finalidad del grupo de trabajo P1451.1 es la definición de un modelo de objetos común para transductores inteligentes, así como el desarrollo de las especificaciones de interfaz requeridas para los componentes de dicho modelo.

La definición de un módulo de interfaz inteligente transductor (STIM), un transductor de hoja de datos electrónicos (TEDS), y una interfaz digital para acceder a los datos fue la misión asignada al grupo de trabajo P1451.2.

El grupo P1451.3 se creó para definir una interfaz de comunicación digital para los sistemas de distribución multipunto (Sistemas *Multidrop*).

Finalmente, el objetivo de P1451.4 se estableció en la definición de una combinación de modo de protocolo de comunicación para transductores inteligentes.

El propósito subyacente, por tanto, es permitir a los fabricantes que los elementos construidos de un sistema sean interoperables. El concepto de sensores inteligentes para controlar la interoperabilidad de redes, la idea de sensor/interoperabilidad de la red, fue concebida por estos grupos de trabajo y ha sido presentada en dos congresos sobre sensores, celebrados en Boston y Chicago.

Para lograr ese objetivo, la familia IEEE 1451 clasifica las partes de un sistema en dos categorías de dispositivos, una de las cuales es la red de procesador de aplicaciones capaces (NCAP), que funciona como una puerta de enlace entre la red de los usuarios y los módulos de interfaz del transductor (TIM). El NCAP es un dispositivo que tiene dos interfaces:

- La interfaz física para los usuarios de la red no se especifica en ninguna de estas normas, ya que lo que proporciona el IEEE 1451.1 es un modelo de objetos lógicos para esta interfaz, pudiendo optar el fabricante por el uso de otras aplicaciones.
- La interfaz de comunicaciones entre el NCAP y TIMs se especifica en los otros resultados de IEEE 1451 (como el IEEE 1451.5). Asimismo, los distintos fabricantes pueden construir sus propias NCAPs y TIMs, que serán interoperables en el caso de que cumplan con lo previsto en el IEEE 1451.

3.2.4 El estándar IEEE 1451

El enfoque del estándar IEEE 1451 goza de gran aceptación, ya que viene acompañado de una reducción de costes mediante el uso de microcontroladores que provoca que los fabricantes de sensores sean proclives a aceptar un estándar que haga posible la interoperabilidad de las redes de sensores inteligentes.

Como se ha mencionado, la familia de estándares proporciona una interfaz común para transductores inteligentes con capacidades (auto-identificación, auto-descripción y auto-diagnosis) que facilita la configuración de sensores *plug&play* con independencia de la tecnología o forma de comunicación específicas de los mismos.

En la ARQUITECTURA se definen 3 partes:

1. Network Capable Application Processor (NCAP).

NCAP: es el hardware y software que hace de pasarela entre los distintos TIMs y la red de usuario y controla al TIM a través de una interfaz digital.

Existen 3 formas de acceder desde la red de usuario al NCAP: mediante Servicios WEB, protocolo IEEE1451.0 y IEEE1451.1

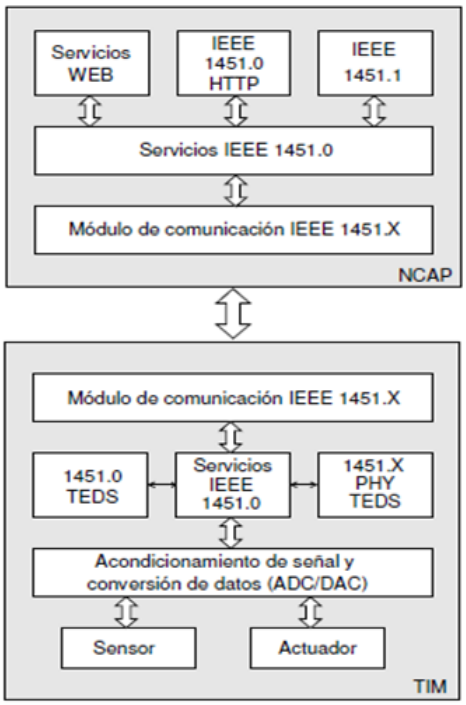
2. Transducer Interface Module (TIM).

TIM: formada por los transductores, electrónica de conversión y procesado de señal. Contiene los TEDS.

TEDS: hojas de datos (en formato digital) con los datos del transductor: fabricante, nº de modelo, nº de serie, rango, sensibilidad, factor de escala y los parámetros importantes del sensor o actuador.

3. Interfaz entre NCAP y TIM.

En este caso, el elegido es el IEEE1451.4



Arquitectura del conjunto de estándares IEEE 1451

3.2.5 El estándar IEEE 1451.4

Actualmente, está enormemente extendida la presencia de sensores en la rutina diaria e industrial de nuestro entorno, informando sobre el funcionamiento de multitud de objetos, e influyendo en la calidad de vida del individuo (por ejemplo, los frigoríficos informan de cualquier alteración de temperatura que pueda poner en riesgo su mantenimiento o la preservación de los alimentos que contiene), pero también en la seguridad de la comunidad (por ejemplo, el control de estado de un reactor nuclear).

Aunque la consulta y procesamiento de la información es casi siempre de tipo digital, los sensores que la recogen son en muchas ocasiones analógicos, por las ventajas que ofrecen: son de coste reducidos, extremadamente fiables y muy resistentes, es decir, son capaces de medir una magnitud que destruiría, o al menos dañaría, a un sensor digital. Son susceptibles de ser utilizados en ambientes de riesgo no aptos para los sensores digitales, como por ejemplo el interior de un motor.

Su inconveniente reside en que fácilmente pueden fallar sin que sea detectable dicho fallo, ni siquiera por ellos mismos, de modo que pueden reportar normalidad de funcionamiento cuando en realidad el sistema está siendo inestable. Este es el problema de de los sensores analógicos denominado calibración, es decir, el proceso de ajustamiento de los datos recogidos por un sensor a las condiciones reales.

El problema de calibración de los sensores analógicos viene a ser abordado por la norma IEEE 1451.4, mediante la combinación de la robustez y la fiabilidad de los sensores analógicos con la inteligencia de los equipos digitales. Por un lado, la norma IEEE 1451.4 aporta una interfaz estándar y el protocolo mediante el cual un sensor puede describirse a través de una red.

En segundo lugar, también corrige otro de los errores más comunes de un sensor analógico, como es el de la transcripción incorrecta de la información de calibración. Los sensores analógicos tienen una tensión de salida proporcional a la magnitud a medir, independientemente de para cual fue diseñado.

En los sensores analógicos, esta información se introduce en la plataforma de adquisición (normalmente, un PC) manualmente por una persona, que la ha consultado en la hoja de datos del sensor (datasheet) suministrada por cada fabricante, por lo que un error humano en este proceso puede volver inútil un sensor, y toda la información procedente del mismo.

La IEEE 1451.4 prevé el uso de los "Transducer Electronic DataSheet" o TEDS, que son datasheets contenidos en una memoria EEPROM situados en algún punto del montaje del sensor, que además de dotar a dicho sensor una capacidad de "auto-identificación", trasladan automáticamente todos sus datos al sistema (el PC), eliminando las posibilidades de error de una transcripción manual.

Además, el provecho de la esta capacidad de auto-identificación marcada por la norma 1451.4 se manifiesta en otros usos, ensayos y entornos, por ejemplo, en redes formadas por multitud de sensores diferentes, aunque aparentemente iguales, o simplemente en ambientes de difícil acceso para personas o en ambientes peligrosos.

3.2.6 Sensores plug&play según el estándar IEEE 1451.4

La norma IEEE1451.4 define el modo en que se incorpora una función de auto-identificación a sensores y actuadores analógicos, con el fin de compatibilizar los transductores ya existentes en el mercado con el modelo IEEE1451. Se añade a los sensores antiguos la capacidad para almacenar TEDS, y que puedan heredar habilidades de auto-descripción y auto-identificación para la operación *plug&play*.

En el estándar IEEE1451.4 define una interfaz de modo mixto con dos señales:

- Una señal analógica de tensión o corriente que representa una magnitud física (temperatura, presión, fuerza, etc.) medida por el sensor.
- Una señal digital para leer los datos TEDs o configurar el transductor.

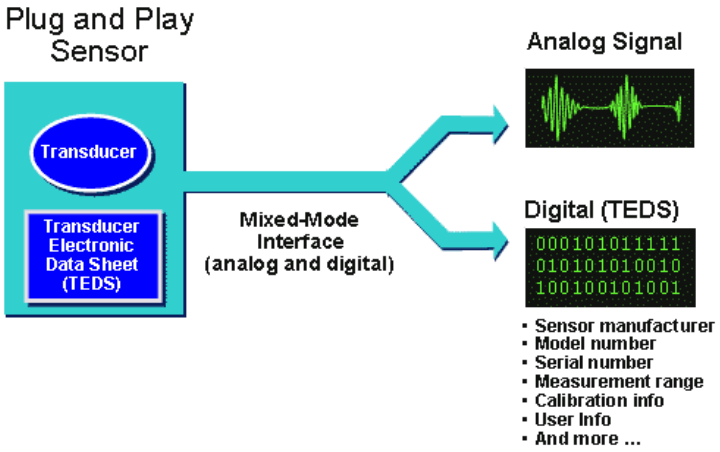


Figura 3.2.6.1: Interfaz de modo mixto

El proceso de instalación y configuración de un sistema de adquisición de datos tradicional, requiere que los parámetros fundamentales del sensor sean introducidos manualmente. Esto es así con lo relativo al rango, sensibilidad y los factores de escala, a fin de que el software sea capaz de usar y escalar adecuadamente los datos del sensor. Un sistema equipado con sensores y actuadores IEEE 1451.4 puede automatizar este paso de configuración, y también incrementar la integridad y la fiabilidad generales del sistema.

Los componentes principales del estándar IEEE 1451.4 son dos: los TEDS y la interfaz de modo mixto. Aunque existen otras tecnologías para sensores inteligentes que ofrecen la capacidad *plug&play*, el IEEE 1451.4 es único en su ámbito porque mantiene la salida analógica del sensor.

3.2.6.1 Interfaz IEEE 1451.4 de Modo Mixto: Sensores Clase I y Clase II

En los sensores de Clase 1, la señal digital se comparte con la señal analógica en las mismas líneas. El sensor incluye la EEPROM y la circuitería para la conmutación basada en una corriente de polarización, que por lo general es sólo una resistencia y un diodo o dos.

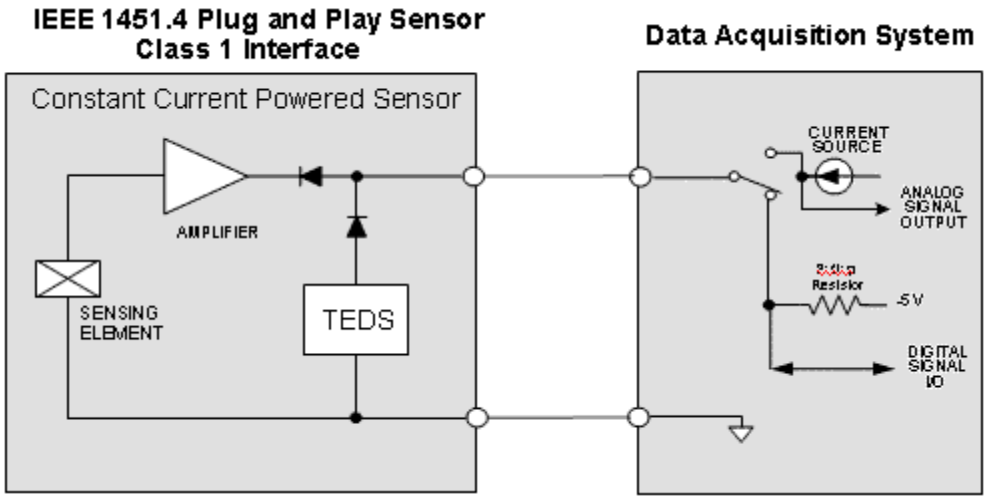


Figura 3.2.6.1.1: Interfaz de Dos Cables Clase 1 para Sensores ICP

En los sensores de Clase 2 de múltiples hilos tendremos una interface de modo mixto con las 2 señales (analógica y digital) separadas en paralelo. Esto permite la implementación de transductores *plug&play* con prácticamente cualquier tipo de sensor o actuador, incluyendo termopares, RTDs, termistores, sensores de puente, celdas electroquímicas y sensores de lazos de corriente de 4-20 ma.

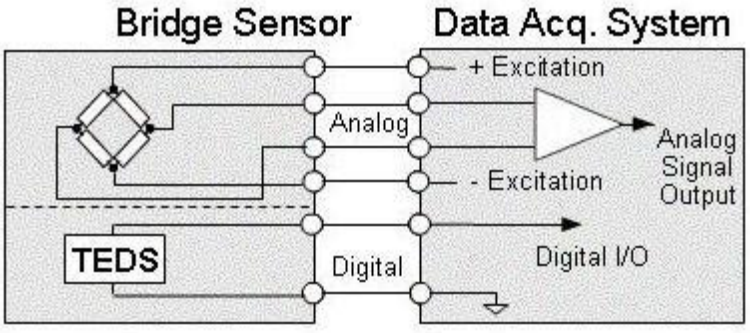


Figura 3.2.6.1.2: Interfaz Clase 2 con las dos vías por separado

3.2.6.2 Transducter Electronic Data Sheet (TEDS)

TEDS (Transducer Electronic Data Sheet) es un método estandarizado de almacenar datos como la identificación, la calibración, los datos de corrección y la información relacionada con el fabricante de un transductor (sensor o actuador).

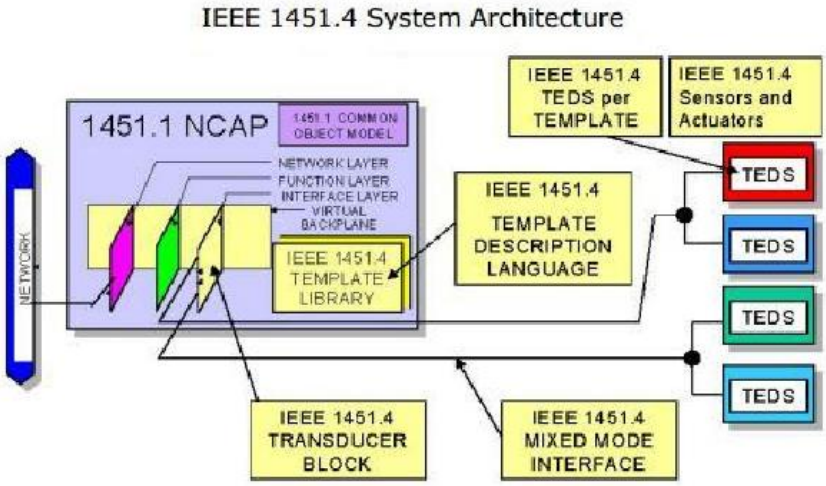


Figura 3.2.6.2.1: Arquitectura IEEE 1451.4

Los TEDS tienen una naturaleza similar a la de los *datasheets* proporcionados por los proveedores, ya que contienen información relativa al sensor, como el nombre del fabricante, tipos del sensor, número de serie, rango de medidas, precisión, datos de calibración y formatos de datos estandarizados.

Son almacenados en una memoria EEPROM, caracterizada por su capacidad para ser borrada y reprogramada entre 100.000 y un millón de veces, y especialmente porque puede ser leída un número ilimitado de veces. Esto implica una gran reducción de la cantidad de papel necesaria, ya que disminuye la necesidad de realizar copias físicas de los *datasheets* de cada uno de los incontables sensores analógicos que hay ahora mismo en circulación en el mundo, número que aumenta cada día que debido al proceso de globalización informática actual, en la que para cada acción hay una automatización respaldada por un sensor.

Pueden resaltarse las siguientes ventajas y beneficios de su uso:

- Permiten la auto-identificación de sensores o actuadores, pudiendo identificarse o auto-describirse al *host* o a la red.
- Proporcionan auto-documentación a largo plazo: los TEDS pueden almacenar y actualizar información como la relativa a la localización del sensor, la fecha de recalibración, los registros de reparación y también datos relativos al mantenimiento.
- Reducen el error humano mediante la transferencia automática de TEDS a la red o sistema, eliminando así los parámetros de sensado manual que inducen a errores.
- Facilitan la instalación de campo, actualización y mantenimiento de los sensores, ayudando a reducir los costes de ciclo de vida (total) de los sistemas de sensores, ya que estas tareas no pueden realizarse vía *plug&play* de los sensores.
- Proporcionan capacidades de *plug&play*. Un TIM y NCAP basados en IEEE 1451 pueden conectarse con un medio de comunicaciones físico estandarizado y operar sin ningún cambio en el software.

En resumen, los TEDS permiten a un sensor inteligente determinado “entenderse” dentro de una red con otros sensores, por lo que constituyen, sin lugar a dudas, la clave más potente del estándar IEEE 1451

Es importante recordar que IEEE 1451 (salvo 1451.4) no tiene TEDS (plantillas para control de un sensor) específicas para cada tipo de sensor (termómetro, barómetro, presencia…), sino que dentro de una TEDS común para todo tipo de sensor se establece la magnitud física a medir, rangos, sensibilidad…. Este es uno de los motivos por los cuales es estándar IEEE 1451.

3.2.6.3 La estructura de los TEDS

Independientemente de la interfaz utilizada, IEEE 1451.4 divide los TEDS en secciones que juntas forman el TED completo.

Basic TEDS	Manufacturer ID	43 (Acme Accelerometer Company)
	Model Number	7115
	Version Letter	B
	Serial Number	X001891
Standard and Extended TEDS (fields will vary according to transducer type)	Calibration Date	Jan 29, 2000
	Sensitivity @ ref. condition (S ref)	1.0094E+03 mV/g
	Physical measurement range	± 50 g
	Electrical output range	± 5 V
	Reference frequency (f ref)	100.0 Hz
	Quality factor @ fref (Q)	300 E-3
	Temperature coefficient	-0.48 %/°C
	Reference temperature (T ref)	23 °C
	Sensitivity direction (x,y,z)	x
User Area	Sensor Location	Strut 3A
	Calibration due date	April 15, 2002

Tabla 3.2.6.3.1. Estructura de una TEDS

La primera sección son los *Basic TEDS*, los cuales comprimen la información relativa a la identificación esencial. Típicamente, un TEDS IEEE estándar que defina las propiedades fundamentales de un sensor particular, seguirá una estructura de "Basic TEDS". Opcionalmente, puede ser ampliado con una plantilla de calibración, denominada Standard and Extended TEDS. Finalmente, existe una sección llamada User Area, que puede ser utilizada a criterio del usuario. Como ya se ha comendato, la información de los TEDS es programada dentro de un chip EEPROM, y la distribución de la estructura de los datos es la que se muestra la siguiente figura:

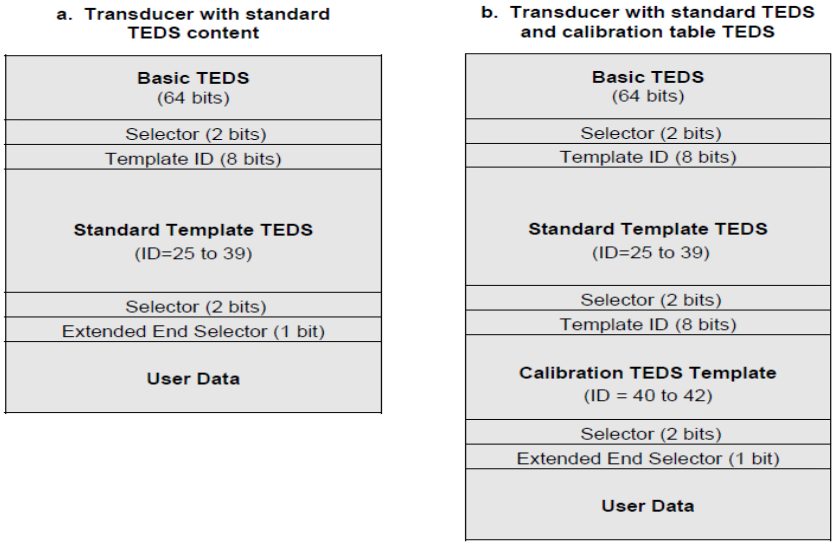


Figura 3.2.6.3.1: Información contenida en la EEPROM

3.2.6.4 Basic TEDS

La norma IEEE 1451.4 define un formato estándar para los datos de TEDS que hacen posible la auto-identificación de sensores, por lo que los TEDS describen todo lo necesario para que se realice la medición por los sensores.

Establece unos *Basic TEDS* en el transductor, con una longitud de 64 bits, sección en la que se almacena la información relativa al fabricante, modelo, versión, código de serie. El *Basic TEDS* únicamente identifica al transductor, e incluye el número de identificación del fabricante (Manufacturer ID) en 14 bits, el número del modelo (15 bits), la letra de la versión (5-bits con código de caracteres), el número de versión (6 bits) y el número de serie del dispositivo (24 bits).

Table 2—Basic TEDS content

Manufacturer ID	Model number	Version letter	Version number	Serial number
14 bits (17–16381) [a]	15 bits (0–32767)	5 bits Char (A–Z, data type Chr5)	6 bits (0–63)	24 bits (0–16777215)

	Bit Length	Allowable Range
Manufacturer ID	14	17 – 16381
Model Number	15	0-32767
Version Letter	5	A-Z (data type Chr5)
Version Number	6	0-63
Serial Number	24	0-16777215

Figura 3.2.6.4.1: Contenido de las Basic TEDS.

El *Manufacturer ID* es número de identificación de fabricante, que es una codificación estandarizada de los fabricantes existentes. Muchos IDs fueron asignados a los primeros fabricantes que usaron ésta norma, y la asignación de los futuros IDs es gestionada por el IEEE. Los IDs asignados están disponibles en un archivo de texto en ASCII, en la página web del IEEE.

En la siguiente imagen vemos un ejemplo de cómo cubrir el campo *MANOFACTURER ID*

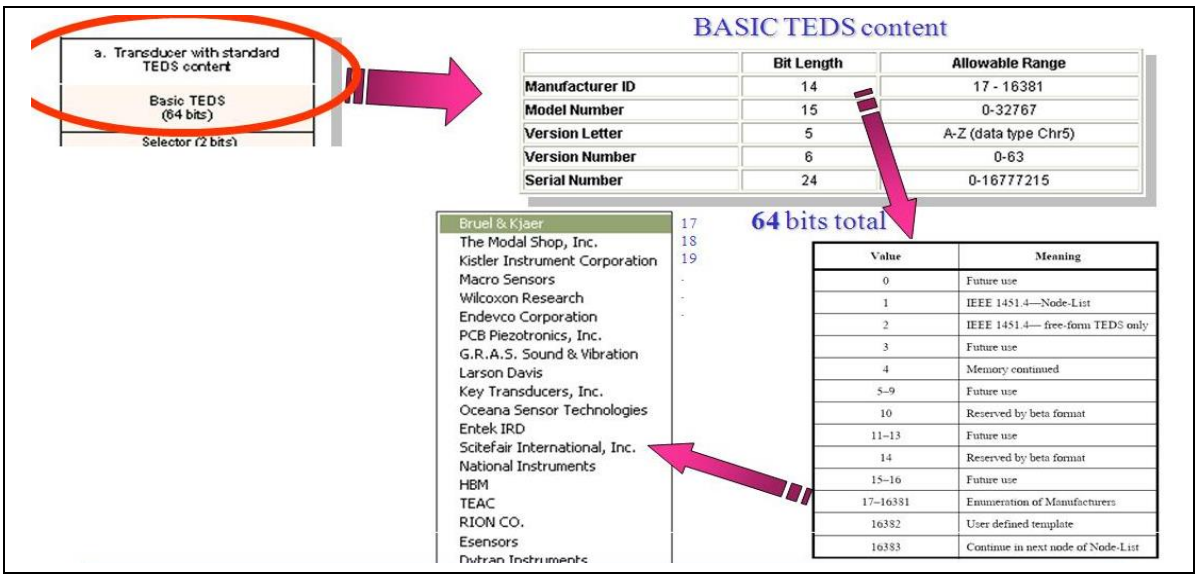


Figura 3.2.6.4.2: Ejemplo de MANOFACTURER ID

Como el *MANOFACTURER ID* es un número que asigna la asociación del estándar IEEE, para obtenerlo hay que registrarse y solicitar que incluyan a la empresa en la lista pública de fabricantes IEEE1451.4:
(https://standards.ieee.org/develop/regauth/manid/public.html)

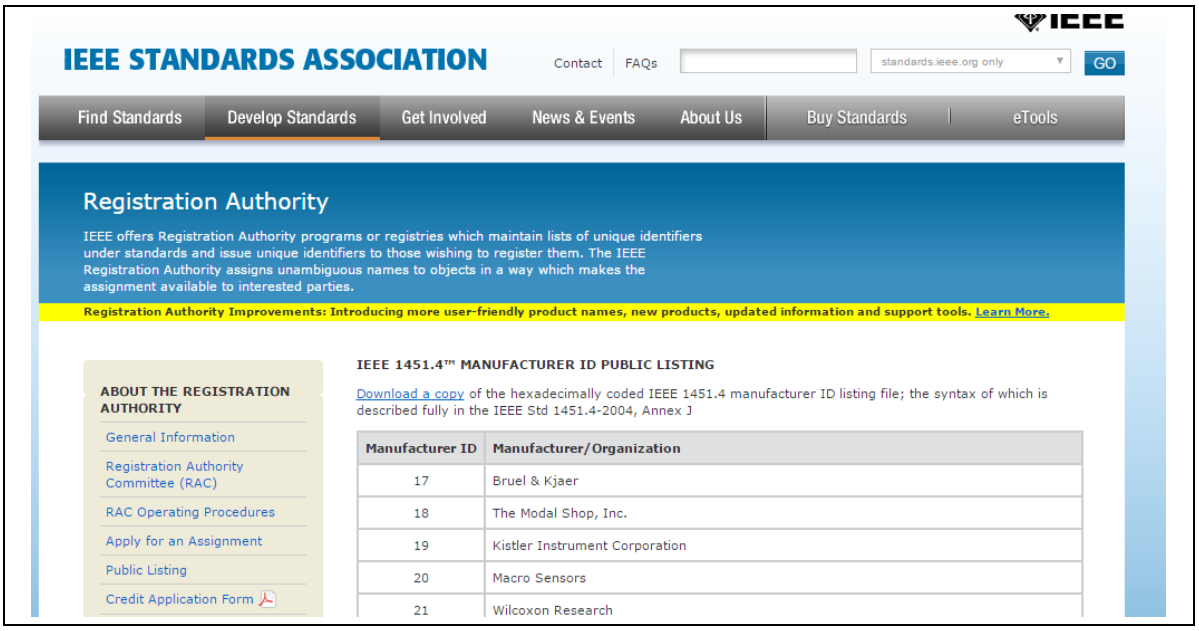


Figura 3.2.6.4.3: Página IEEE ESTÁNDAR ASSOCIATION

El número de modelo se deja a elección del fabricante, por lo que lo habitual es que se corresponda con la codificación numérica que el fabricante utiliza para el control de sus productos.

En la siguiente imagen vemos un ejemplo como cubrir los campos *VERSION LETTER* y *VERSION NUMBER*.

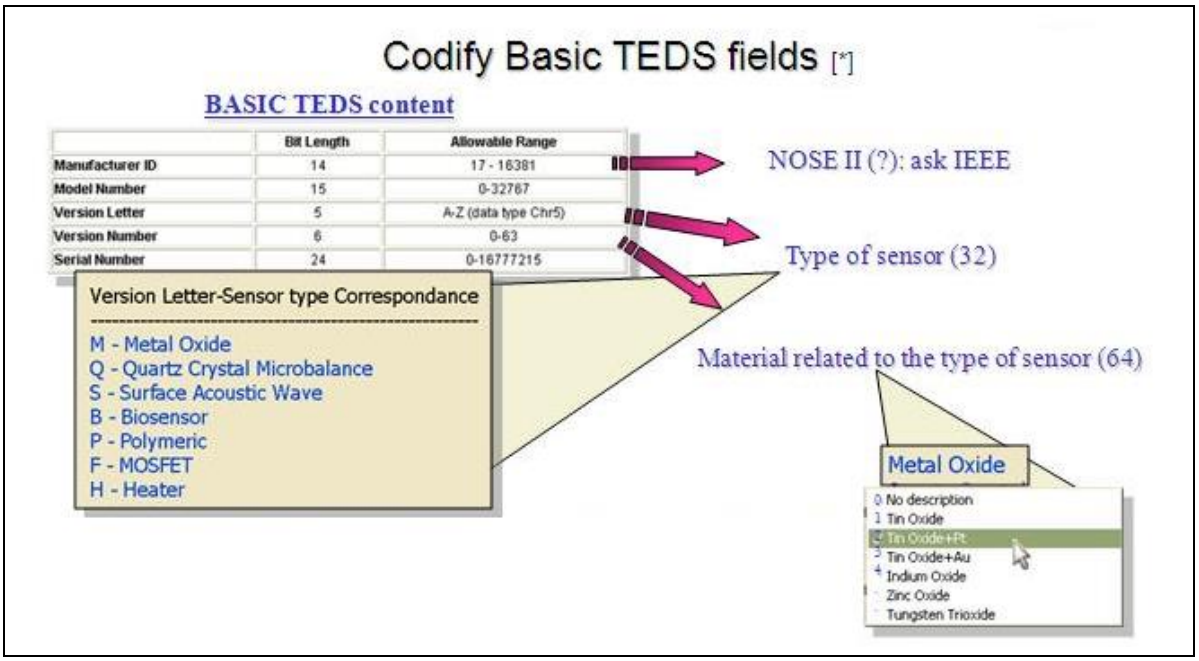


Figura 3.2.6.4.4: Ejemplo de los campos *VERSION LETTER* y *VERSION NUMBER*

El *selector* es un campo que describe si la plantilla es o no una plantilla definida según el estándar IEEE 1451.4. Este campo es de 2 bits (un rango de 0 a 3), con el siguiente significado:

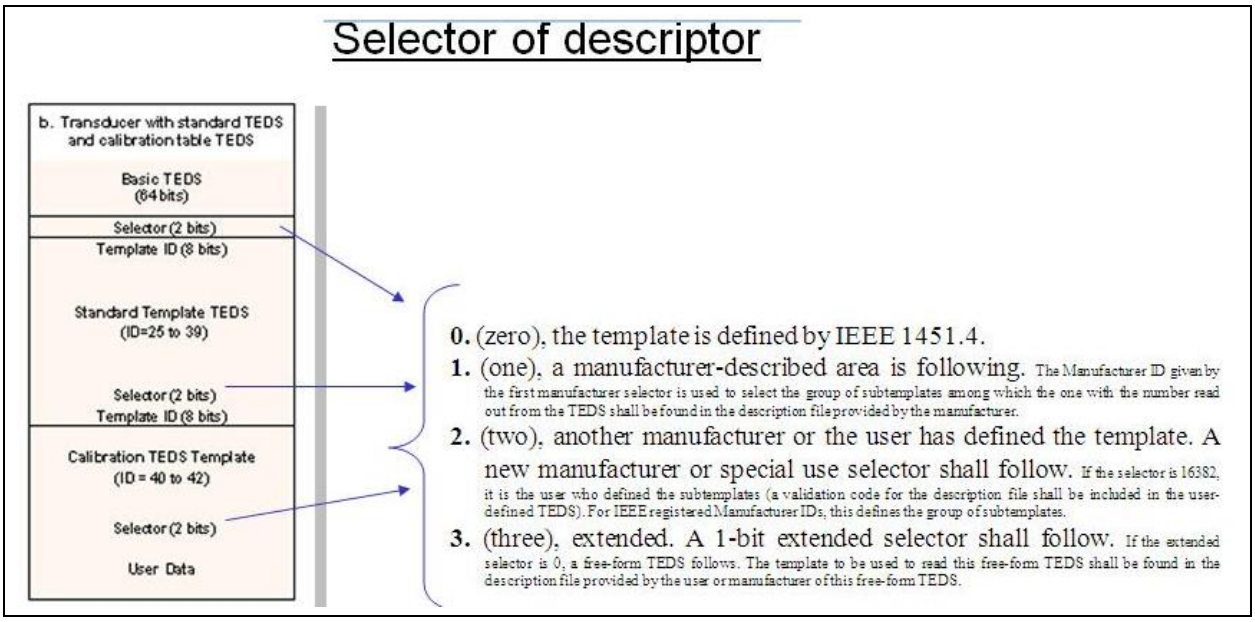


Figura 3.2.6.4.5: Campo “Selector”

3.2.6.5 Plantillas del estándar para los TEDS

Lo que caracteriza al estándar IEEE 1451.4 dentro de la familia IEEE 1451, es que define una colección de plantillas (*templates*) específicas para cada tipo de sensor (termómetro, barómetro…), que se alojan junto con las TEDS, y son referenciadas por éstas. En su Anexo A, el estándar IEEE 1451.4 contiene todas las plantillas para transductores comunes.

IEEE standard templates

Type	Template ID	Name of Template
Transducer Type Templates	25	Accelerometer & Force
	26	Charge Amplifier (w/ attached accelerometer)
	43	Charge Amplifier (w/ attached force transducer)
	27	Microphone with built-in preamplifier
	28	Microphone Preamplfiers (w/ attached microphone)
	29	Microphones (capacitive)
	30	High-Level Voltage Output Sensors
	31	Current Loop Output Sensors
	32	Resistance Sensors
	33	Bridge Sensors
	34	AC Linear/Rotary Variable Differential Transformer (LVDT/RVDT) Sensors
	35	Strain Gage
	36	Thermocouple
	37	Resistance Temperature Detectors (RTDs)
	38	Thermistor
	39	Potentiometric Voltage Divider
Calibration Templates	40	Calibration Table
	41	Calibration Curve (Polynomial)
	42	Frequency Response Table

Figura 3.2.6.4.3: Relación de plantillas para TEDS

Las plantillas numeradas del 25 al 39 contienen propiedades necesarias para tipos de transductores específicos. Las plantillas 40, 41, y 42 son plantillas de calibración, y se pueden utilizar con las plantillas de tipo de transductor.

3.2.6.6 Hardware para almacenamiento de los TEDS

Una TEDS, en esencia, es un dispositivo de memoria conectado al transductor y contiene la información necesaria relativa a un dispositivo de medida o al sistema de control de la interfaz del transductor.

En cualquier caso, una TEDS puede ser puesta en la práctica de dos formas:

- Primero, la TEDS puede residir en la memoria integrada, típicamente una EEPROM, en el propio transductor que es conectado al dispositivo de medida o al sistema de control.
- Segundo, una TEDS puede ser virtual, es decir, puede existir como un fichero de datos al que se puede acceder según el dispositivo de medida o el sistema de control. Es muy práctico en situaciones en las cuales la memoria integrada no puede estar disponible.

En el caso en que la TEDS esté en formato texto, la estructura Tipo/Longitud/Valor (TLV) es utilizada para proporcionar un directorio que dé acceso a las diferentes secciones del texto de la TEDS, usando XML para el contenido de la información.

En la utilización de estructuras TLVs, el campo "Tipo" es una etiqueta de 1 octeto que identifica la propia TLV, de modo similar a la funciones en HTML o a las etiquetas en XML. El campo "Longitud" es el que especifica el número de los octetos del campo “Valor”, y el campo "Valor" son los datos, la información como tal. Cada entrada puede estar compuesta de uno o varios TLVs. La estructura y el tipo de datos del campo de valor son definidos en la especificación para TEDS.

La parte central del estándar 1451.4 es el uso de una memoria digital de solo lectura (EEPROM), consistente en chips incorporados al sensor analógico que almacenan la hoja de datos del sensor, así como la información de identificación del sensor (tipo, fabricante, número de serie). Cuando se conecta a un sensor tipo 1451.4 habilitado para la adquisición de datos, el chip EEPROM transmite el TEDS al sistema, de una manera similar a como se conectan un ratón o una impresora a un PC, a través del USB.

El estándar, sin embargo, no determina el punto en él se debe instalar el chip EEPROM en el sensor analógico. El chip puede ser colocado en el interior de la cubierta del sensor, en el conector del sensor que va al equipo de toma de datos, o incluso en el interior del cable del sensor. Esto permite que el sensor analógico pueda ser instalado en ambientes peligrosos, con el chip EEPROM asentado en la zona de menor riesgo, que por lo general es el otro extremo del cable.

3.2.6.7 La EEPROM DS-2431

La memoria EEPROM requerida por el estándar 1451.4 para almacenar las TED es la DS2431. Es una memoria que permite la comunicación por un solo hilo y contiene una ROM de 64 bits grabada por láser con un número de registro único.

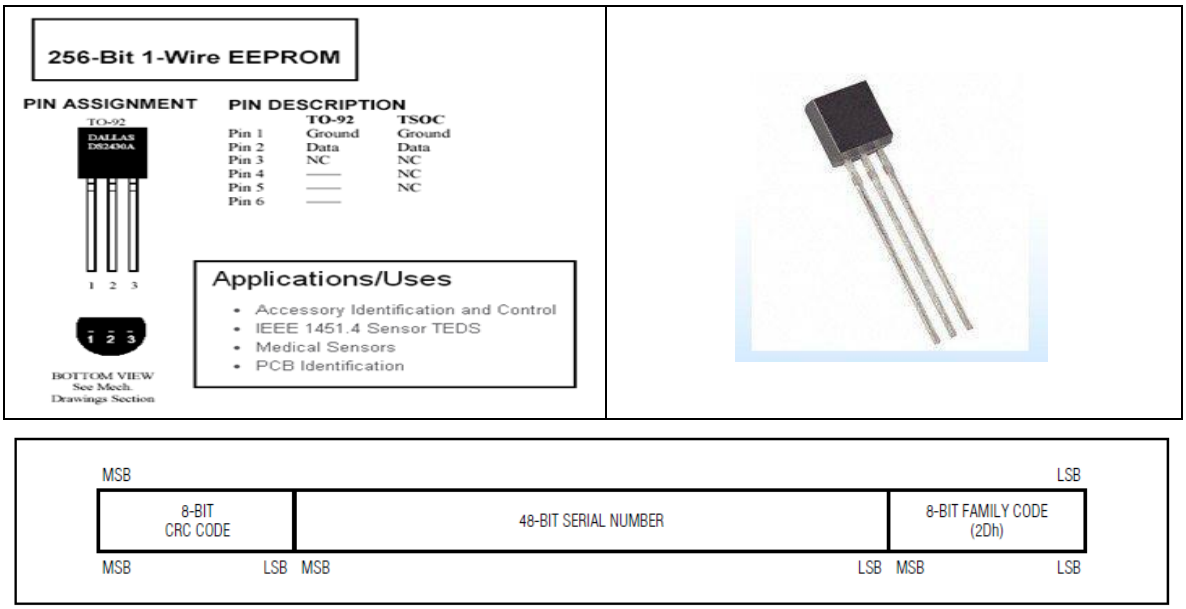


Figure 3. 64-Bit Lasered ROM

Figura 3.2.6.4.3: EEPROM DS 2431

La DS2431 es una memoria EEPROM de 1024-bit con protocolo *1-Wire*®. Es un chip organizado en cuatro páginas de memoria de 256 bits. Los datos son escritos en un espacio de 8-bytes, que son verificados y copiados en la memoria EEPROM. Como características especiales, cabe señalar que las cuatro páginas de memoria pueden ser individualmente escritas en modo protegido o puestas dentro de *EPROM-emulationmode*, donde los bits solo pueden ser cambiados de estado 1 a 0. La comunicación de la DS2431 se realiza a través del protocolo estándar *1-wire*.

Cada dispositivo tiene su propio número de registro que es inalterable y único en una ROM de 64-bits, que es gravado mediante laser dentro del chip. El número de registro es utilizado para tratar el dispositivo en entornos de red multipunto *1-wire*.

Esta especialmente diseñada para la calibración de los sensores analógicos, incluida la norma IEEE 1451.4. Sus características generales son las siguientes:

- 1024 Bits de memoria EEPROM particionada en 4 páginas de 256 bits.
- Páginas de memoria individual que pueden ser permanentemente protegidas de escritura o puestas en el modo de emulación (“escritura a 0”).
- Histéresis de conmutación y filtrado para optimizar el rendimiento en presencia de ruido.
- IEC 1000-4-2 Level 4 ESD Protection (±8kV Contact, ±15kV Air, typical)
- Escritura y lectura con un rango de voltaje de 2.8V a 5,25V, en un margen de temperatura entre -40ºC a +85ºC.
- Comunicación con el Host con una señal digital simple de 15.4kbps o 125kbps usando *1-wire Protocol*.

3.3 DISEÑO Y CÁLCULO DE LOS SENSORES

3.3.1 Desarrollo de un sistema de Adquisición de datos inteligente conforme a la norma IEEE 1451.4

Este Proyecto pretende presentar las bases de desarrollo de un sistema basado en TEDS de clase II para la adquisición inteligente, que cumpla la normativa IEEE 1451.4, que pueda aplicarse a cualquier sensor genérico, y de ésta forma aprovechar las múltiples ventajas que brinda esta norma. Para ello, se define la estructura de las plantillas de datos de los sensores estableciendo un protocolo de comunicación identificativo de los datos relevantes para la diferenciación de los sensores, tales como el dispositivo de origen, su nombre y la medición del sensor.

Anteriormente, se ha explicado resumidamente el contenido de la norma y el funcionamiento básico de ésta, incidiendo en los puntos clave. A continuación se explicará con ejemplos el funcionamiento de los sensores, así como las ventajas que aporta el estándar para los de tipo analógico.

Después, se mostrarán los pasos seguidos para el desarrollo del sensor inteligente Clase II, y posteriormente se mostrarán los resultados del desarrollo llevado a cabo, así como una lista de posibles mejoras a realizar a partir de lo expuesto. Finalmente, se documentan y realizan las pruebas correspondientes, en principio sobre simulaciones de datos obtenidos, y a continuación sobre las medidas reales obtenidas.

3.3.2 Implementación del Estándar IEEE1451

La base de este proyecto es la de desarrollar un sistema que adapte un sensor genérico en un sensor inteligente conforme a la norma IEEE 1451.4.

Para la implementación se ha seguido el siguiente procedimiento cumpliendo con el estándar 1451:

1) Selección del NCAP
2) Configuración del TIM
3) Programación
4) Pruebas

3.3.2.1 Selección del NCAP

Se ha elegido el Arduino UNO como dispositivo NCAP para leer las señales analógicas de los sensores y decodificar la información de las TEDS.

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos.

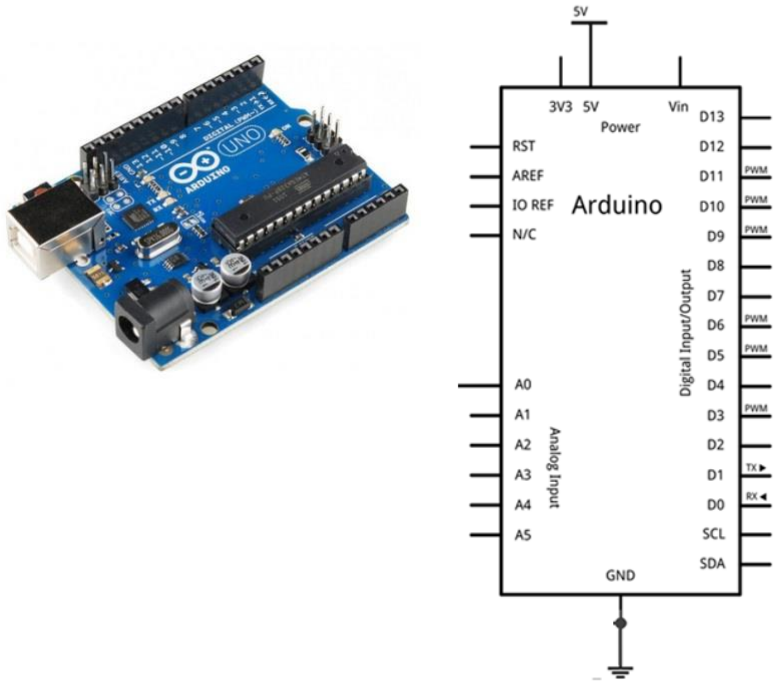


Figura 3.3.2.1.1: Arduino UNO

Es importante tener en cuenta a la hora de leer los sensores analógicos, que el arduino UNO consta de un microcontrolador "ATmega328P" que tiene 6 ADCs de 10 bits.

2^{10} = 1024 divisiones (de 0 a 1023) del voltaje de referencia.

0V = 0

5V = 1023

3.3.2.2 Configuración del TIM

El conjunto del sensor más la memoria EEPROM DS2431 (TEDS) forman el bloquee del TIM.

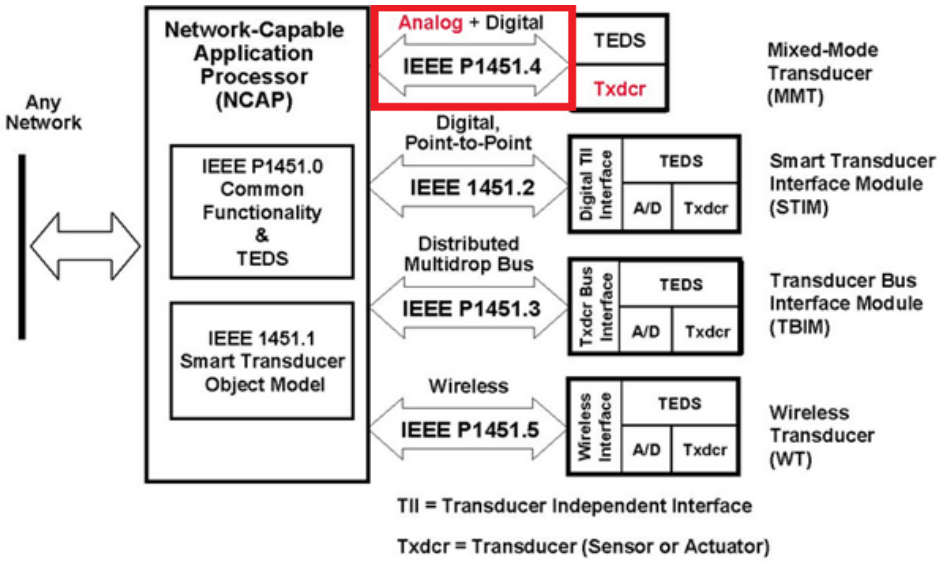


Figura 3.3.2.2.1: Interfaz IEEE 1451.4

Debido al material disponible, se ha decidido adaptar el sensor al interfaz TEDS de clase II de múltiples cables, en el que la adquisición de los datos analógicos y digitales van por vías diferentes, por ser la interfaz más sencilla de implementar.

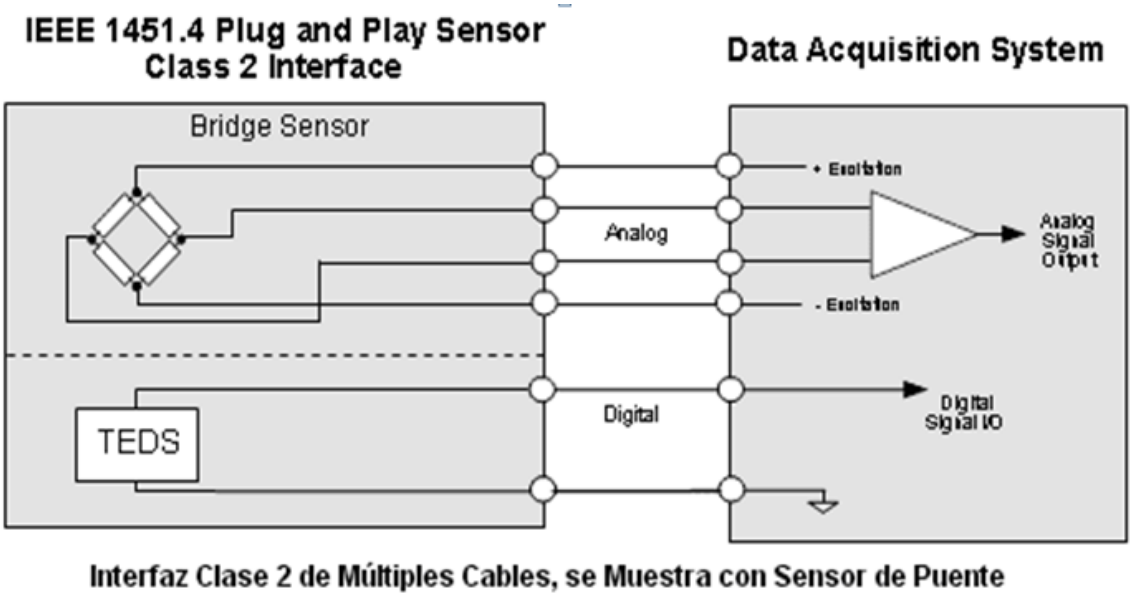
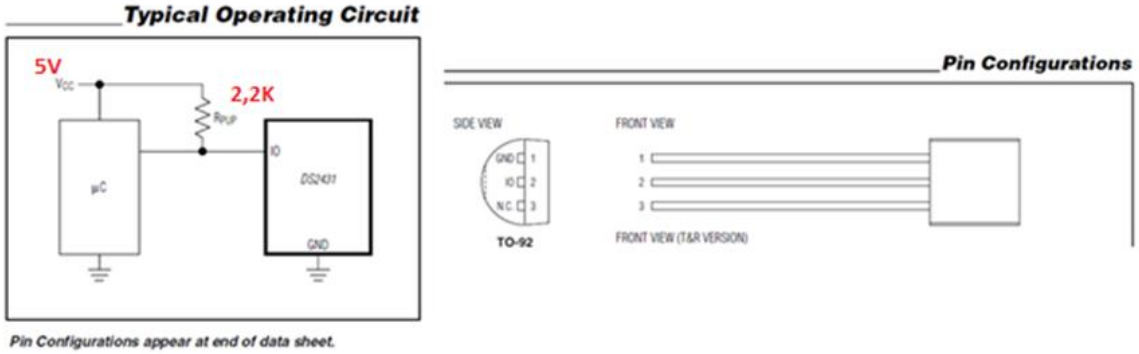


Figura 3.3.2.2: Interfaz Clase 2 Múltiples Cables

La memoria EEPROM utilizada para almacenar las TED es la DS2431, que es la requerida en el estándar 1451.4. Es una memoria que permite la comunicación por un solo hilo y contiene una ROM de 64 bits grabada por láser con un número de registro único. Este número es utilizado para identificar el sensor dentro de una red.

Una vez instalada en la protoboard se realizan las primeras pruebas de la memoria EEPROM siguiendo el esquema del circuito del datasheet de la DS2431



PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
IO PIN: GENERAL DATA					
1-Wire Pullup Voltage	V_{PUP}	2.8		5.25	V
1-Wire Pullup Resistance	R_{PUP}	0.3		2.2	kΩ
Input Capacitance	C_{IO}			1000	pF

Figura 3.3.2.3: Esquema de montaje de la EEPROM DS2431

Se realiza una primera prueba de lectura y escritura de la EEPROM cargando los siguientes datos en hexadecimal:

```
dat[0] = 0x09;
dat[1] = 0x44;
dat[2] = 0xB2;
dat[3] = 0xBC;
dat[4] = 0x64;
dat[5] = 0xEB;
dat[6] = 0x6D;
dat[7] = 0x8D;

WriteRow(0,dat);
```

```
dat[0] = 0xB5;
dat[1] = 0xD5;
dat[2] = 0xC3;
dat[3] = 0x18;
dat[4] = 0x0C;
dat[5] = 0x70;
dat[6] = 0xCB;
dat[7] = 0x68;

WriteRow(1,dat);
```

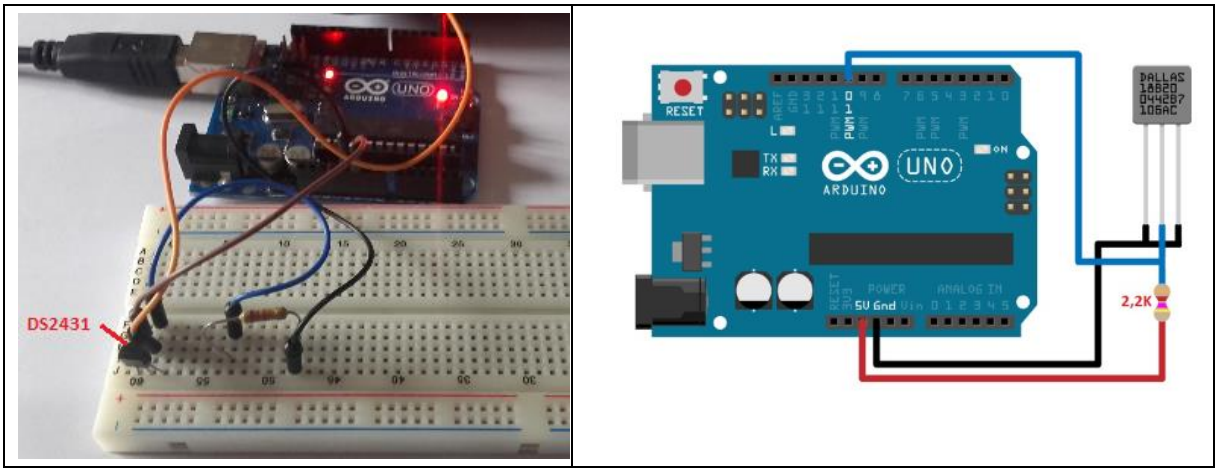


Figura 3.3.2.4: Pruebas EEPROM DS2431

3.3.2.3 Programación

CÓDIGO PRUEBA ESCRITURA/LECTURA DS2431

```
#include <OneWire.h>


OneWire  ds(2);    // 1-wire on pin 2
byte     addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)
void setup(void)
{
  Serial.begin(9600);
}


void loop(void)
{
  //byte i;
  byte dat[13];
```

```
SearchAddress(addr); //lectura y comprobación de la ROM

//Grabamos DATOS DE LA PLANTILLA en la EPROM DS2431

dat[0] = 0x09;
dat[1] = 0x44;
dat[2] = 0xB2;
dat[3] = 0xBC;
dat[4] = 0x64;
dat[5] = 0xEB;
dat[6] = 0x6D;
dat[7] = 0x8D;

WriteRow(0,dat);

dat[0] = 0xB5;
dat[1] = 0xD5;
dat[2] = 0xC3;
dat[3] = 0x18;
dat[4] = 0x0C;
dat[5] = 0x70;
dat[6] = 0xCB;
dat[7] = 0x68;

WriteRow(1,dat);

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;
```

```
 WriteRow(2,dat);

 dat[0] = 0x00;
 dat[1] = 0x00;
 dat[2] = 0x00;
 dat[3] = 0x00;
 dat[4] = 0x00;
 dat[5] = 0x00;
 dat[6] = 0x00;
 dat[7] = 0x00;

WriteRow(3,dat);

 dat[0] = 0x00;
 dat[1] = 0x00;
 dat[2] = 0x00;
 dat[3] = 0x00;
 dat[4] = 0x00;
 dat[5] = 0x00;
 dat[6] = 0x00;
 dat[7] = 0x00;

 WriteRow(15,dat);

 ReadAllMem();    //print all mem content

 while(1);
}

void SearchAddress(byte* address) //Search for address and confirm it
// lee el codigo gravado por LASER de la EEPROM que la hace única
{
 int i;
 if ( !ds.search(address))
 {
```

```
  Serial.print("No device found.\n");
  ds.reset_search();
  delay(250);
  return;
 }

 Serial.print("ADDR= ");
 for( i = 0; i < 8; i++)
 {
  Serial.print(address[i], HEX);
  Serial.print(" ");
 }

 if ( OneWire::crc8( address, 7) != address[7])
 {
    Serial.print("CRC is not valid, address is corrupted\n");
    return;
 }

 if ( address[0] != 0x2D)
 {
    Serial.print("Device is not a 1-wire Eeprom.\n");
    return;
 }
 Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
 int i;
 ds.reset();
 ds.select(addr);
 ds.write(0x0F,1);  // comando Write ScratchPad
 ds.write(TA1,1);
 ds.write(TA2,1);
```

```
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);        // Read Scratchpad

  for ( i = 0; i < 13; i++)
    data[i] = ds.read();
}

void CopyScratchPad(byte* data)
{
  ds.reset();
  ds.select(addr);
  ds.write(0x55,1);  // Copy ScratchPad
  ds.write(data[0],1);
  ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
  ds.write(data[2],1);
  delay(25); // Waiting for copy completion
  //Serial.print("Copy done!\n");
}

void ReadAllMem()
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0xF0,1);  // Read Memory
  ds.write(0x00,1);  //Read Offset 0000h
  ds.write(0x00,1);

  for ( i = 0; i < 144; i++) //whole mem is 144
  {
    Serial.print("byte");
```

```
    Serial.print("(");
    Serial.print(i);
    Serial.print(") ");
    Serial.print(ds.read(), HEX);
    Serial.println(" ");
  }
  Serial.println();
}

void WriteRow(byte row, byte* buffer)
{
  int i;
  if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
    return;              //The remaining are for the 64 bits register page

  WriteReadScratchPad(row*8, 0x00, buffer);

  CopyScratchPad(buffer);
}
```

3.3.2.4 Pruebas de la EEPROM DS2431

Una vez se carga el programa y se pone en funcionamiento se hace la comprobación por el monitor serie de que todo funciona correctamente:

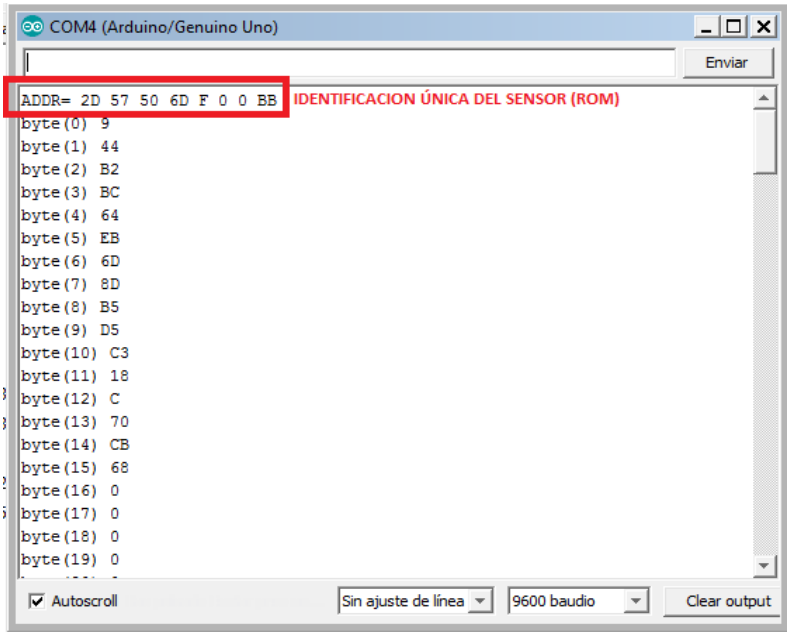


Figura 3.3.2.4.1: Monitor serie Arduino. Lectura del contenido de la EEPROM.

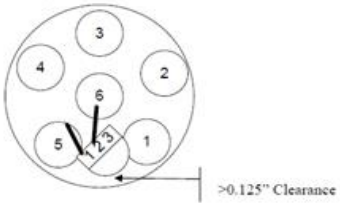
3.3.3. Instalación de la memoria EEPROM en el sensor

Para integrar la memoria EEPROM en los sensores se propone como solución utilizar el siguiente material:

Component	Digikey Part Number
DS2431 EEPROM	DS2431+-ND
Switchcraft EN3C6F	SC1162-ND

Tabla 3.3.3.1: Material para integración de la EEPROM en la sonda

Lo primero es instalar la memoria EEPROM DS2431 en el conectar “Swichcraft EN3C6F”. Para ello se suelda siguiendo el siguiente esquema:



Pin	Standard Wire Color	Two 2 Wire Sensors	Two 3 Wire Sensors	One 3 Wire and one 2-Wire Sensors
1	Red	N/U	Excitation [2]	Excitation [2]
2	Orange	Sensor 1+ [1]	Sensor 1 Out [3]	Sensor 1 Out [3]
3	Yellow	Sensor 1- [1]	Exc Monitor [4]	Exc Monitor [4]
4	Green	Sensor 2+ [1]	Sensor 2 Out [3]	Sensor 2 + [1]
5	Black	Sensor 2- [1]	Gnd [5]	Sensor 2 - [1]
6	Brown	ID (n/c)		

Figura 3.3.3.2: Instalación DS2431 en conector Swichcraft EN3C6F

Una vez integrada la EEPROM en el conector se procede al montaje del mismo en la sonda:

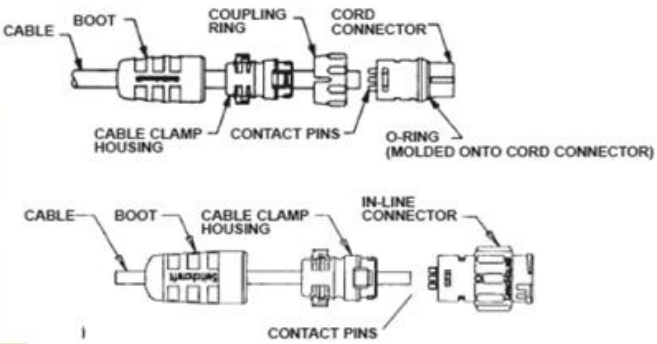


Figura 3.3.2.3: Sonda

3.3.4 Sensor RTD (PT1000)

3.3.4.1 Introducción

El sensor PT1000 es un sensor de tipo RTD (*resistance temperature detector*). Es un detector de temperatura resistivo, es decir, varía su resistencia con la variación de temperatura.

Su símbolo, en el que se indica una variación lineal con coeficiente de temperatura positivo, es el siguiente.

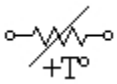


Figura 3.3.4.1.1: Símbolo RTD

El sensor PT1000 consiste en un arrollamiento muy fino, normalmente de platino bobinado entre capas de material aislante y protegido por un material cerámico. El material que forma el conducto “palatino” posee un coeficiente de resistencia alta, que es el que determina la variación de resistencia por cada grado que cambia su temperatura.

Por lo general las sondas PT1000 industriales se fabrican encapsuladas de la misma forma que los termopares, dentro de un tubo de acero inoxidable u otro material (vaina). En un extremo está el elemento sensible (Sensor RTD) y en el otro están los terminales eléctricos de los cables de conexión.

Figura 3.3.4.1.2: Vaina típica de un sensor RTD industrial

La nomenclatura del sensor RTD se compone del material, de la resistencia en Ohmios a 0ºC y de la clase del sensor de temperatura .En el caso de la PT1000 a 0°C tiene una resistencia de 1000 ohms, y al aumentar la temperatura aumenta su resistencia eléctrica.

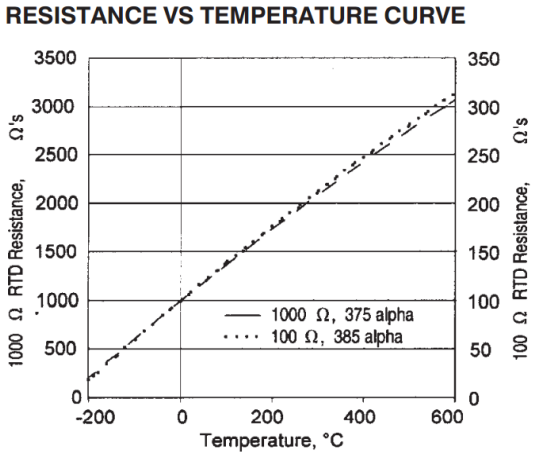


Figura 3.3.4.1.3: Curva Resistencia-Temperatura PT1000

El incremento de la resistencia de la PT1000 de platino no se realiza de forma lineal, sino de forma creciente, de modo que mediante el uso de tablas es posible encontrar la temperatura exacta que le corresponde.

La variación del valor de resistencia puede ser representada de manera polinómica conforme la siguiente ecuación.

$$R_T = R_0(1 + \alpha\Delta T + \beta\Delta T^2 + \gamma\Delta T^3 ...)$$ [Ecuación 3.3.4.1.1]

Por lo general, la variación es bastante lineal en márgenes amplios de temperatura. Como los valores de α, β, etc son valores muy pequeños, la ecuación se puede aproximar a la siguiente expresión:

$$R = R_0 \cdot (1 + \alpha \cdot \Delta T) \qquad \text{[Ecuación 3.3.4.1.2]}$$

Dónde:

- R_0 es la resistencia a la temperatura de referencia T_0
- ΔT es la desviación de temperatura respecto a T_0 $(\Delta T = T - T_0)$
- α es el coeficiente de temperatura del conductor a 0 °C.

Materiales de las RTDs

Los materiales más empleados para la construcción de sensores RTD suelen ser conductores como el platino, el niquel-hierro, el cobre, el molibdeno, etc. Las propiedades de algunos de estos materiales se muestran en la siguiente tabla:

	RANGO DE OPERACIÓN	PRECISIÓN
	(ºC)	(grados)
Platino	-200 a 950	0.01
Níquel	-150 a 300	0.50
Cobre	-200 a 120	0.10

Tabla 3.3.4.1.1: Propiedades de los materiales

Los sensores RTDs que utilizan el platino como sensor son los más usados ya que entre sus ventajas, destacan por tener mejor linealidad, mayor margen de temperatura [-200ºC, 800ºC] y mayor rapidez.

Clases de Tolerancia

Las clases establecidas por la norma IEC 751:1995 especifican la clase de tolerancia para las RTDs según la siguiente tabla:

Clases de Tolerancia	Ecuación de Tolerancia (ºC)
Clase A	± (0.15 + 0.002 · | t |)
Clase B	± (0.30 + 0.005 · | t |)
Clase C	± (0.40 + 0.009 · | t |)
Clase D	± (0.60 + 0.0018 · | t |)

Tabla 3.3.4.1.2: Ecuaciones de tolerancia según clases de tolerancia

Conexión de la PT1000

Existen 3 modos de conexión para las RTD (Pt1000), pero cada uno de ellos requiere un instrumento lectura diferente. Lo que se busca es determinar exactamente la resistencia eléctrica del sensor de platino sin que influya en la lectura la resistencia de los cables.

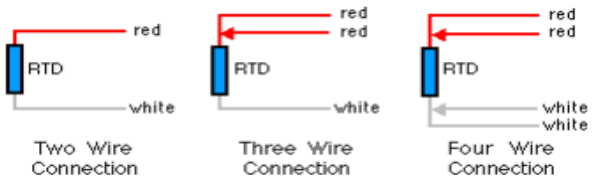


Figura 3.3.4.1.4: Conexionado de las RTD

PT1000 a 2 hilos: es el modo más sencillo de conexión (pero el menos recomendado) es con solo 2 cables. Este modo solo se recomienda para medición máximo a 10 metros ya que a partir de ahí el sensor pt1000 puede tener pérdidas de señal.

PT1000 a 3 hilos: El modo de conexión a 3 hilos es el más común y más utilizado en los procesos industriales ya que resuelve bien el problema del error generado por los cables. El único requisito es que los tres cables tengan la misma resistencia eléctrica pues el sistema de medición está basado en el "puente de Wheatstone". La mayoría de los equipos industriales vienen preparados para realizar la conexión a 3 hilos.

PT1000 a 4 hilos: El método de 4 hilos es el más preciso de todos y se usa para laboratorio. Los 4 cables pueden ser distintos (de distinta resistencia) pero debido a esto el instrumento lector es más caro.

Ventajas y desventajas de los sensores PT1000

Ventajas

- Los sensores Pt1000 siendo más caros y mecánicamente no tan rígidos como las termocúpulas, superan a estas en aplicaciones a bajas temperaturas (-100 a 200°).
- Las Pt1000 pueden entregar precisiones de una décima de grado con la ventaja de que no se descompone gradualmente con el tiempo entregando lecturas erróneas, sino que normalmente el circuito se abre, con lo cual el dispositivo medidor detecta inmediatamente la falta del sensor, pudiendo generarse una alarma de aviso.
- Además la Pt1000 permite ser colocada a cierta distancia del medidor sin mayor problema (hasta unos 30 metros) pudiendo ser utilizado con cable de cobre convencional para hacer la extensión.
- Tienen salida de gran amplitud.
- El rango de medida de temperatura es amplio.
- Sensibilidad a cambios de temperaturas altas y diez veces mayor a la de un termopar.
- Excelente linealidad.
- Alta exactitud, estabilidad, repetitibilidad y resistencia de choques térmicos.

Desventajas

- Velocidad de reacción baja, comparada con un termopar o termistor.
- Están afectadas por el auto-calentamiento.
- Son inestables ante las vibraciones o choques mecánicos

Aplicaciones de las RTD

La correcta selección de un sensor RTD depende de la aplicación para lo que se van a emplear. Las aplicaciones más habituales de estos sensores suelen ser:

- En procesos de inyección de plásticos
- Control de temperatura de motores
- Procesos alimenticios
- Procesos industriales
- Equipos de empaque

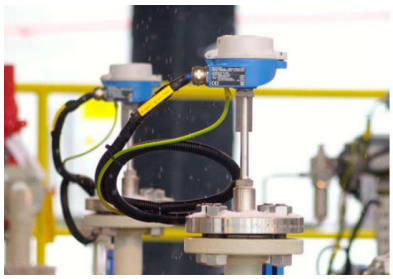


Figura 3.3.4.1.5: RTD en entornos industriales

3.3.4.2 Características técnicas del sensor

Las especificaciones según el fabricante del sensor PT1000 utilizado en este trabajo son las siguientes:

Pt1000 Elements, Thin Film (1000 Ohm)

- Pt1000 elements to IEC 751 Class A and B
- For use from –50°C to +500°C
- Thin film construction
- Suitable for surface & immersion applications where protected
- Vibration resistant

Specifications:

Sensor type:	Pt1000 (1000 Ohms @0°C)
Construction:	Thin film, 10mm tails
Temperature range:	-50°C to +500°C
Ice point resistance:	1000Ω
Fundamental interval (0°C to 100°C):	385Ω (nominal)
Self heating:	<0.5oC/mW
Thermal response:	0.1s
Stability:	±0.05%

Resistance	Dimensions (width x length)	Tolerance Class	RS order code
Pt1000	2.0 x 10mm	Class A	362-9907
Pt1000	2.0 x 10mm	Class B	362-9913

Figura 3.3.4.2.1: Especificaciones de la PT1000

Parámetro	Platino (Pt)
Resistividad ($\mu\Omega cm$)	10.6
$\alpha(\Omega/\Omega/C)$	0.00385
$R_0(\Omega)$	25, 50, 100, 200
margen (°C)	-200 a +850

Figura 3.3.4.2.2: Parametros principales de la RTD de platino

Coeficientes del Callendar – Van Dusen:

Uno de los parámetros requerido por la plantilla del estándar IEEE1451 para la RTD son los parámetros de la **Callendar-Van Dusen** .

La **ecuación de Callendar-Van Dusen** es una ecuación que describe la relación entre la resistencia (R) y la temperatura (T) de las RTD.

$$R_t = R_0\left\{1 + At + Bt^2 - C(t-100)t^3\right\}$$ [Ecuación 3.3.4.2.1]

$$A = \alpha + \frac{\alpha \cdot \delta}{100} \qquad B = -\frac{\alpha - \delta}{100^2} \qquad C = -\frac{\alpha - \beta}{100^4}$$ [Ecuación 3.3.4.2.2]

Different Coefficients for (alpha)			
Coefficient	**Value**	**Value**	**Value**
α	0,003850	0,003926	0,003911
δ	1,4999		
β	0,10863		
A	$3{,}9083e^{-3}$	$3{,}9848e^{-3}$	$3{,}9692e^{-3}$
B	$-5{,}775e^{-7}$	$-5{,}870e^{-7}$	$-5{,}8495e^{-7}$
C	$-4{,}18301e^{-12}$	$-4{,}000e^{-12}$	$-4{,}2325e^{-12}$

These three values represent the three principal specifications for RTD's.

- 0,003850 //°C: Standard DIN 43760, IEC 751, named Europeen Industrial Standard.
- 0,003926 //°C: Require pur platinum (99,999%), named U.S. Industrial Standard.
- 0,3911 //°C: Often named U.S. Industrial Standard.

Figura 3.3.2.3: Coeficientes RTD

Standard Temperature	Coefficient α (/°C)	A (/°C)	B (/°C²)	C (/°C⁴)
DIN 43760	0.00385	3.9080×10^{-3}	-5.8019×10^{-7}	-4.2735×10^{-12}
American	0.003911	3.9692×10^{-3}	-5.8495×10^{-7}	-4.2325×10^{-12}
ITS-90	0.003926	3.9848×10^{-3}	-5.870×10^{-7}	-4.0000×10^{-12}

Tabla 3.3.4.2.1: Coeficientes Callendar-Van Dusen

Los parámetros para la PT-1000 son tomados del estándar DIN 43760. La tabla anterior muestra los parámetros principales de un sensor PT-1000.

3.3.4.3 Calibración y acondicionamiento del sensor

El valor de la PT1000 varía en función de la temperatura siguiendo la siguiente ecuación lineal:

$$R_{PT1000}=R_0*(1+\alpha*T) \qquad \text{[Ecuación 3.3.4.3.1]}$$

$R_0=1000$

$\alpha=0{,}003908$

Dando valores a "T" se obtienen los valores correspondientes de resistencia del sensor a una temperatura determinada:

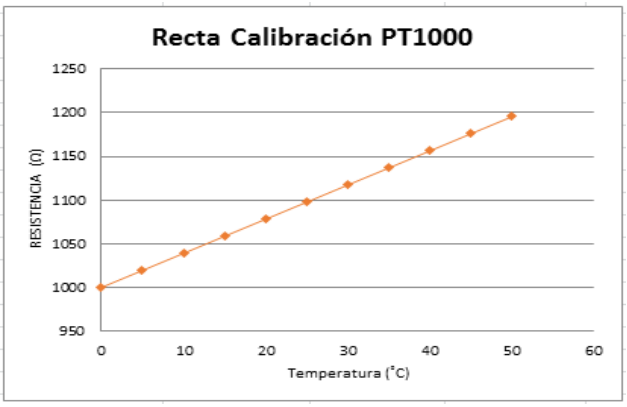


Figura 3.3.4.3.1: Recta de calibración PT1000

Tensión de salida del puente de Wheastone (Valor V_s)

El circuito de acondicionamiento empleado para la PT1000 es el montaje en puente de Wheastone donde Vs es la tensión de salida del puente:

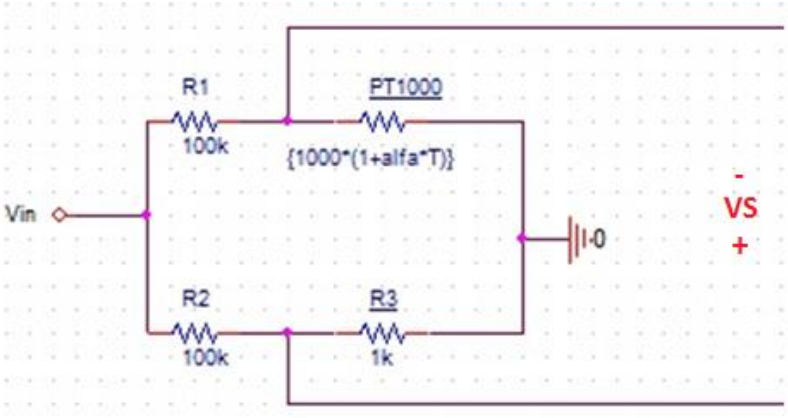


Figura 3.3.4.3.2: Vs Puente de Wheastone

$$V_S = \left(\frac{R_3}{R_2 + R_3} - \frac{R_{PT1000}}{R_{PT1000} + R_1} \right) V_{in}$$

[Ecuación 3.3.4.3.2]

A **T=0 ˚C** se quiere que **V_s** sea igual a **0**, por lo tanto, **R_3=R_0 y R_2=R_1=n*R_0**. Se escogen las resistencias R_3=R_0=1K Ω, R_2=R_1=100kΩ. La función del puente de Wheatstone es, mediante unas simplificaciones, conseguir que la tensión de salida sea lineal. Llegando, finalmente, a la siguiente ecuación:

$$\boldsymbol{V_s = V_{s+} - V_{s-} = \frac{n*\alpha*T}{(n+1)^2} * V\text{in}}$$

[Ecuación 3.3.4.3.3]

Siendo n=100 y el alfa característico de esta PT100 α=0,003908.

Teniendo en cuenta que el puente de Wheatstone se alimenta con 5V se obtiene que **V_s** es igual a **0 V** para una temperatura de **0 ˚C** y **9,56 mV** para **50 ˚C**.

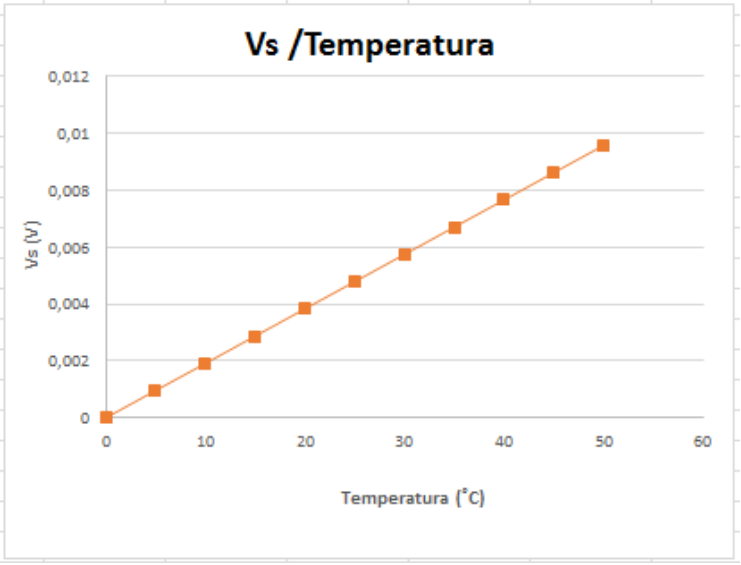


Figura 3.3.4.3.3: Vs/Temperatura PT1000

3.3.4.4 Simulación en OrCAD

Circuito de acondicionamiento (puente de wheastone)

Se realiza el montaje en OrCAD del del puente de wheastone asignándole a la PT1000 una resistencia de valor igual a la ecuación de la recta de calibración que define el comportamiento del sensor.

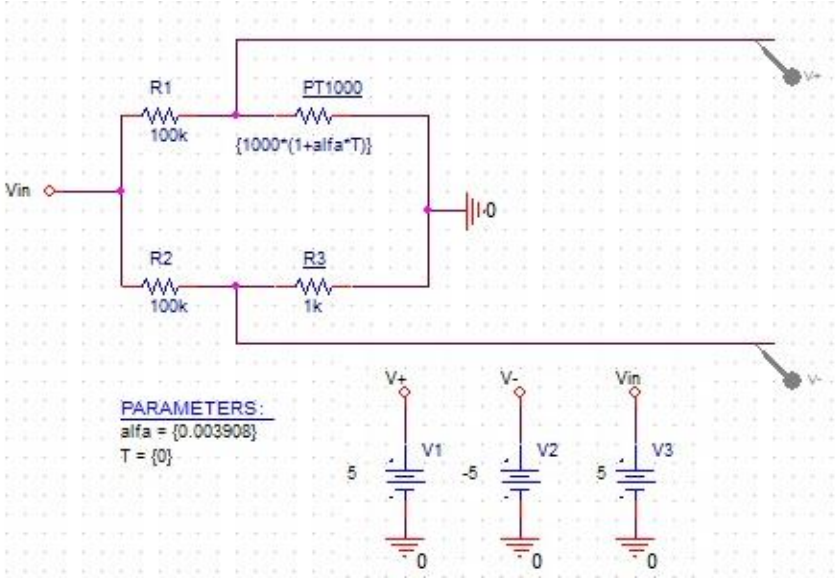


Figura 3.3.4.4.1: Puente de Wheastone

Una vez definidos los valores de alfa y la temperatura, se realiza un barrido continuo, por ejemplo, desde 0 hasta 50 (que es el rango de grados centígrados entre los que opera el medidor) con un incremento unidad.

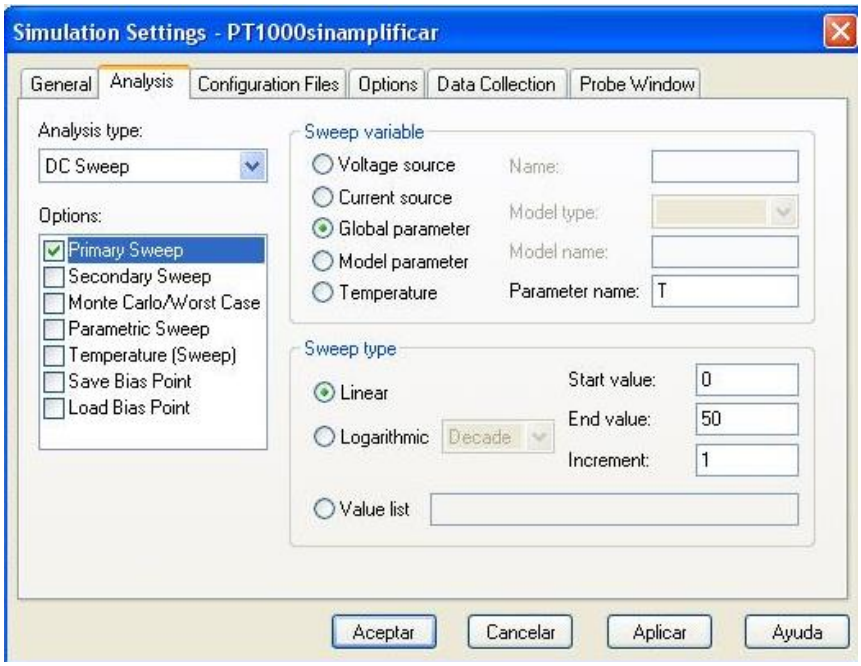


Figura 3.3.4.4.2: Configuración de la simulación

Al simular, midiendo la tensión de salida de todo el circuito (Vs), se obtiene la siguiente gráfica:

Para T=0˚C; Vs=0V
Para T= 50˚C; Vs=9,56 mV

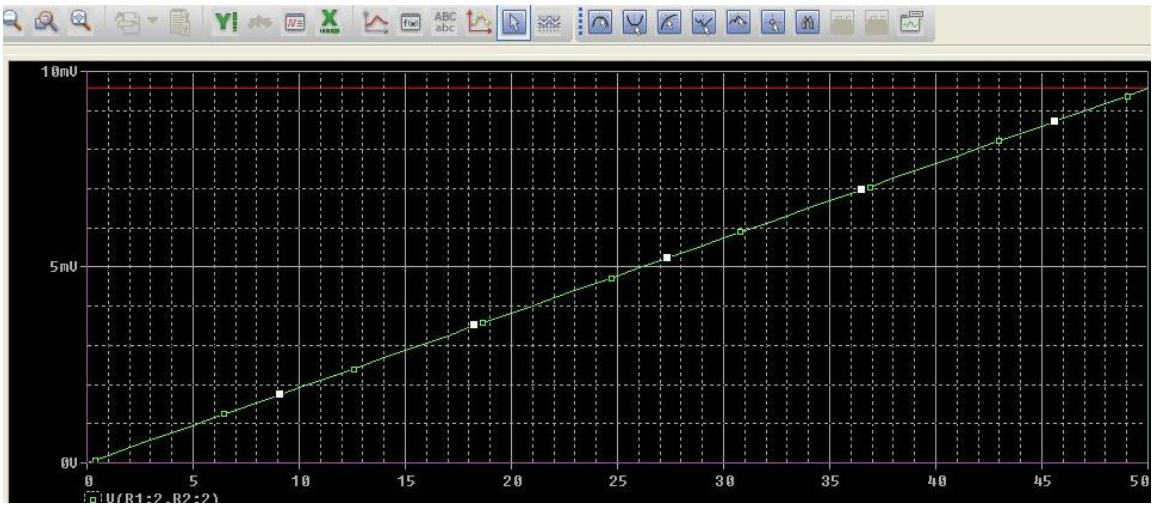
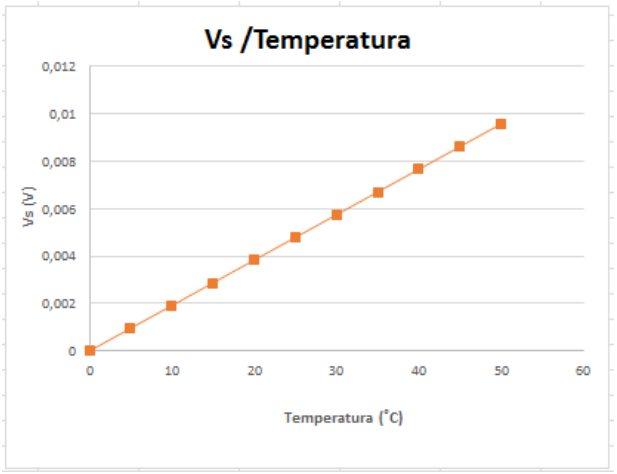


Figura 3.3.4.4.3: Gráfica de la tensión de salida puente de Wheastone

Una vez simulado el circuito, se observa que la gráfica de la simulación coincide con los resultados esperados calculados en EXCEL.



Temperatura (˚C)	R_{PT1000} (Ω)	Vs (V)
0	1000	0
5	1019,54	0,00095775
10	1039,08	0,0019155
15	1058,62	0,00287325
20	1078,16	0,003831
25	1097,7	0,00478875
30	1117,24	0,0057465
35	1136,78	0,00670424
40	1156,32	0,00766199
45	1175,86	0,00861974
50	1195,4	0,00956749

Figura 3.3.4.4.4:Tensión de salida del puente de Wheastone obtenida en Excel

Etapa de amplificación de la señal

Las entradas analógicas de Arduino sirven para medir voltajes entre 0 y 5V. Para una alimentación del puente con el sensor a 5V se obtiene a su salida una tensión de 0 a 9,56 mV como hemos visto en el apartado anterior. Por ello, es necesario utilizar un circuito adaptador de escala que amplifique la salida para obtener mejor resolución de la medida.

Ganancia de amplificación

Como se pretende que a 50 ˚C la tensión de salida sea 5V, se tendrá que ampliar la tensión obtenida en el puente de Wheatstone. Esto se hará con el operacional AD623, cuya ganancia deberá ser igual a 5V/0.00957V = 523,013.

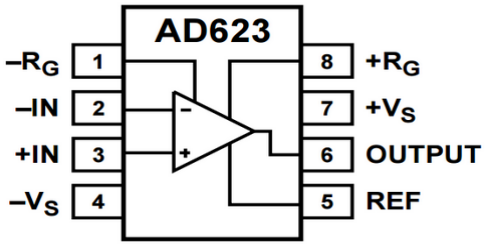


Figura 3.3.4.4.5: Esquema de un AD623

Valor de R_G

Al usar un operacional AD623, R_G se calcula como

$$R_G = \frac{100\ k}{(G-1)} \qquad \text{[Ecuación 3.3.4.4.1]}$$

Dado que la ganancia es 523,013, R_G=191,57 Ω.

Con el dato de R_G se puede montar ya el circuito para obtener a la salida una tensión amplificada que varíe entre 0 y 5 V. En el montaje se empleará una R_G de 192 Ω (3 resitencias de 4 Ω + 1 resistencia de 180 Ω.)

A continuación se realiza el montaje del circuito amplificador en OrCAD:

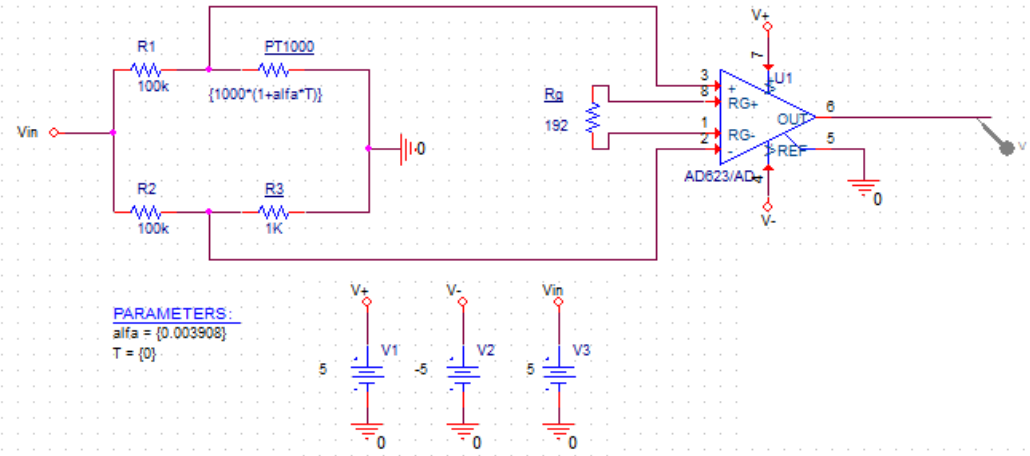


Figura 3.3.4.4.6: Puente de Wheastone con etapa de amplificación

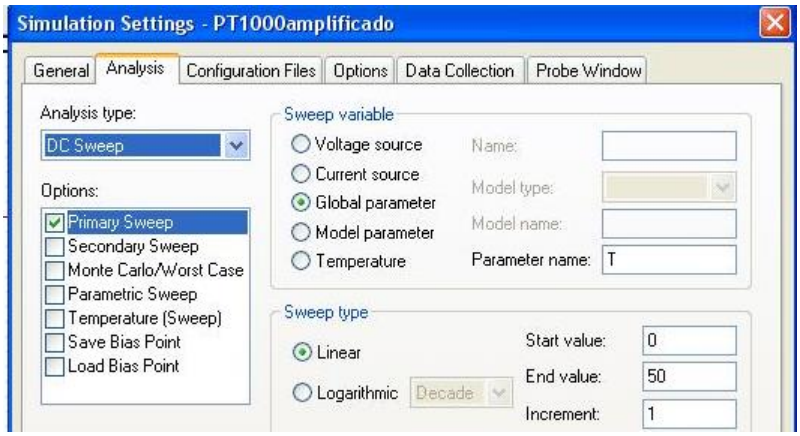


Figura 3.3.4.4.7: Configuración de la simulación

Una vez simulado el circuito, se observa que la gráfica de la simulación coincide con los resultados esperados calculados en EXCEL.

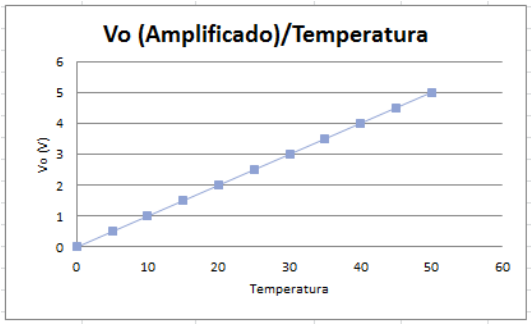


Figura 3.3.4.4.8: Curva de salida de la etapa de amplificación obtenida en Excel

Temperatura (˚C)	R_{PT1000} (Ω)	Vs (V)	Vo Amplificado (V)
0	1000	0	0
5	1019,54	0,00095775	0,500915303
10	1039,08	0,0019155	1,001830607
15	1058,62	0,00287325	1,50274591
20	1078,16	0,003831	2,003661214
25	1097,7	0,00478875	2,504576517
30	1117,24	0,0057465	3,00549182
35	1136,78	0,00670424	3,506407124
40	1156,32	0,00766199	4,007322427
45	1175,86	0,00861974	4,508237731
50	1195,4	0,00957749	5,009153034

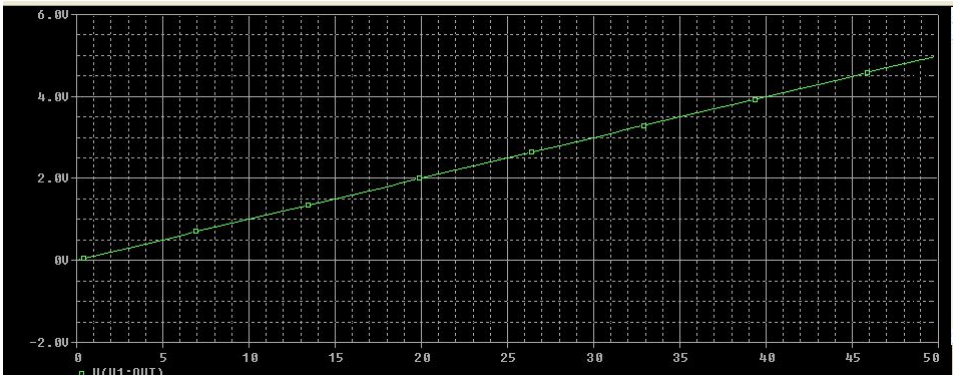


Figura 3.3.4.4.9: Simulación en OrCAD de la Salida de la etapa de amplificación / Entrada de señal al Arduino

Se ha logrado que la tensión de salida se amplie, variará entre 0 y 5 de forma lineal.

3.3.4.5 Implementación del estándar IEEE 1451 en una PT1000

Para la implementación del diseño de un sensor inteligente PT1000 se realiza los pasos indicados en el estándar IEEE1451.4:

1) Se emplea la arquitectura clase 2 de múltiples hilos por su sencilla integración. Tendremos una interface de modo mixto con 2 señales separadas: una analógica (la lectura del sensor) y otra digital (los datos de la TED).
2) El sensor PT1000 y la memoria EEPROM DS2431 formarán el TIM
3) El Arduino Uno será el encargado de leer, decodificar e interpretar la información trabajando como NCAP.

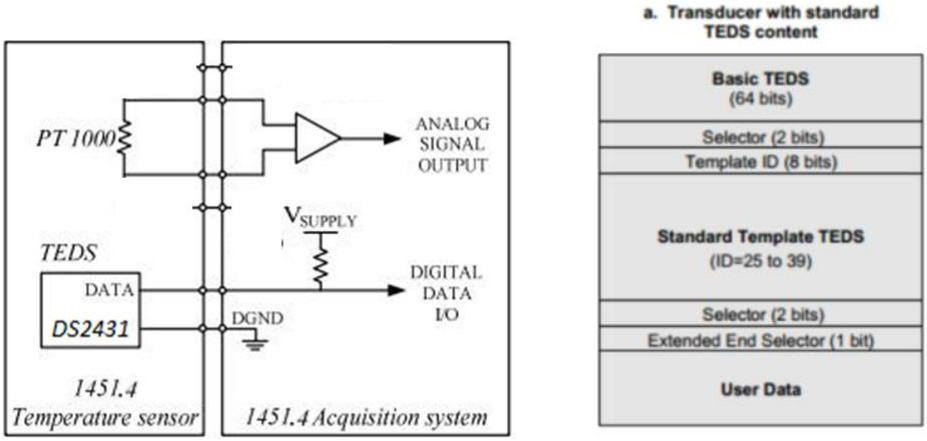


Figura 3.3.4.5.1: Esquema PT1000 Estándar IEEE1451.4

4) La TED (Transducer Electronic Data Sheets) es la información del sensor. Para sensores RTDs está tabulada en el ANEXO A , plantilla (ID=37) del ESTÁNDAR IEEE1451.4
5) La memoria EEPROM DS2431 será la encargada de almacenar la información digital de la TED del sensor PT1000
6) Se procede a cubrir los campos con la información del fabricante de la BASIC TED+SELECTOR+STANDAR TEMPLATE
7) Finalmente se codifica esta información y se almacena en la EEPROM DS2431

Una vez definidos estos parámetros se procede a la implementación del estándar en un sensor PT1000.

IEEE1451.4, ANEXO A, PLANTILLA ID=37 para RTDs

Table A.14—Resistance temperature detectors (RTDs) template (ID = 37) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 37)	—
Measurement	—	%MinPhysVal	Minimum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	°C
	—	%MaxPhysVal	Maximum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	°C
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 2, "Resistance Sensor"	—
	—	%MinElecVal	Minimum electrical output	CAL	11	ConRes (0 to 2.05 k, step 1)	Ω
	—	%MaxElecVal	Maximum electrical output	CAL	13	ConRes (0 to 8.2 k, step 1)	Ω
	—	%MapMeth	Mapping Method	ID	—	Assign = 5, "RTD"	—
	Select Case—R0 Resistance				2	Select Case	—
	Case 0	%RTDCoef_R0	Resistance R0	ID	—	Assign = 100.0	Ω
	Case 1	%RTDCoef_R0	Resistance R0	ID	—	Assign = 120.0	Ω
	Case 2	%RTDCoef_R0	Resistance R0	ID	—	Assign = 1000.0	Ω
	Case 3	%RTDCoef_R0	Resistance R0	ID	20	ConRelRes (1 to 12.5k, ±4.5 ppm)	Ω
	Select Case—RTD Curve (Callendar-Van Dusen Coefficients)				3	Select Case	—
	Case 0	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.8100E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –6.0200E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –6.000E-12	$1/C^3$
	Case 1	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9083E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = –5.7750E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = –4.183E-12	$1/C^3$

Table A.14—Resistance temperature detectors (RTDs) template (ID = 37) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 2	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9692E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = −5.8495E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = −4.229E-12	$1/C^3$
	Case 3	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9739E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = −5.8700E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = −4.39E-12	$1/C^3$

Table A.15—Resistance temperature detectors (RTDs) template (ID = 37) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
	Case 4	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9787E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = −5.8685E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = −4.160E-12	$1/C^3$
	Case 5	%RTDCoef_A	CVD Coefficient A	ID	—	Assign = 3.9888E-3	1/C
		%RTDCoef_B	CVD Coefficient B	ID	—	Assign = −5.915E-7	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	—	Assign = −3.816E-12	$1/C^3$
	Case 6	%RTDCoef_A	CVD Coefficient A	ID	13	ConRes (3.8E-3 to 4E-3, step 2.5E-8)	1/C
		%RTDCoef_B	CVD Coefficient B	ID	10	ConRes (−6.1E-7 to −5.6E-7, step 5E-11)	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	7	ConRes (−6E-12 to −3E-12, step 2.3E-14)	$1/C^3$
	Case 7	%RTDCoef_A	CVD Coefficient A	ID	32	Single	1/C
		%RTDCoef_B	CVD Coefficient B	ID	32	Single	$1/C^2$
		%RTDCoef_C	CVD Coefficient C	ID	32	Single	$1/C^3$
	—	%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation or power supply	—	%ExciteAmplNom	Excitation current, nom	CAL	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
	—	%ExciteAmplMax	Excitation current, max	ID	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
			Total bits required for TEDS (range):		135 to 251 bits		

BASIC TED

Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
Manufacturer ID	0	14	00 0000 0000 0000
Model Number	0	15	000 0000 0000 0000
Version Letter	0	5	0 0000
Version Number	0	6	00 0000
Serial Number	0	24	0000 0000 0000 0000 0000 0000
BITS Totales		64	

SELECTOR

Property / Command	Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
Selector	The template is defined by IEEE1451.4	0	2 bits	00
BITS Totales			2 bits	

PLANTILLA Resistance temperature detectors (RTDs) template (ID = 37)

Property / Command	Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
TEMPLATE	Template ID	37	8 bits	0010 0101
%MinPhysVal	Minimum temperature	-50 (ºC)	11 bits	000 1001 0110
%MaxPhysVal	Maximum temperature	500 (ºC)	11 bits	010 1011 1100
%ElecSigType	Transducer Electrical Signal Type	-	-	String = "2, Resistance sensor"
%MinElecVal	Minimum electrical output	807(Ω)	11 bits	011 0010 0111
%MaxElecVal	Maximum electrical output	2925(Ω)	13 bits	0 1011 0110 1101
%MapMeth	Mapping Method	-	-	String = "5, RTD"
Select Case—R0 Resistance (%RTDCoef_R0)	Resistance R0	Case 2 R0=1000 (Ω)	2 bits	10
Select Case—RTD Curve (Callendar-Van Dusen Coefficients)	CVD Coefficient A CVD Coefficient B CVD Coefficient C	Case 1 A=3.9083E-3 (1/C) B=–5.7750E-7 (1/C2) C=–4.183E-12 (1/C3)	3 bits	001
%RespTime	Sensor Response Time	0,1 (Seg)	6 bits	10 1101
%ExciteAmplNom	Excitation current, nom	3 mA = 0,003 (A)	8 bits	1011 0010
%ExciteAmplMax	Excitation current, max	40mA= 0.040 (A)	8 bits	1110 1100
%CalDate	Calibration Date	Desde 1/2/1998 hasta 1/2/2018 = 7330 dias	16 bits	0001 1100 1010 0010
%CalInitials	Calibration initials	LNC	15 bits	01100 01110 00011
%CalPeriod	Calibration period	2 años = 730 dias	12 bits	0010 1101 1010
%MeasID	Measurement location ID	Plantilla nº 0	11 bits	000 0000 0000
BITS Totales			135 bits	

CODIGO COMPLETO BINARIO

0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000
0000	1001	0100	0100	1011	0010	1011	1100
0110	0100	1110	1011	0110	1101	1000	1101
1011	0110	0101	1101	1000	0011	1001	0100
0100	1100	0111	0000	1100	1011	0110	1000
0000	0000	0000	0000	0000	0000	0000	0000

CODIGO COMPLETO HEXADECIMAL (32 bytes)

00 00 00 00 00 00 00 00 09 44 B2 BC 64 EB 6D 8D B6 5D 83 94 4C 70 CB 68
00 00 00 00 00 00 00 00

CÁLCULO Y OPERACIONES (SEGÚN IEEE1451):

7.4.5.3.3 ConRes

The constant resolution data type assigns a value according to this formula:

property name = <*start_value*> + [<*tolerance*> x <*teds_value*>]

%MinPhysVal	Minimum temperature	CAL	11	ConRes (−200 to 1,846, step 1)	°C

Para -50ºC

%MinPhysVal = -200+1*X= -50

X=150 = 1001 0110

%MaxPhysVal	Maximum temperature	CAL	11	ConRes (−200 to 1,846, step 1)	°C

Para 500ºC

%MaxPhysVal=-200+1*X=500

X=700 = 10 1011 1100

Modelo matemático de una RTD:

$$R_T = R_0 \cdot (1 + \alpha \cdot \Delta T)$$

Para -50ºC

Rt=1000(1+0,00385 *(-50))= 807,5 Ω

Para 500ºC

Rt=1000(1+0,00385 *(500))= 2925 Ω

%MinElecVal	Minimum electrical output	CAL	11	ConRes (0 to 2.05 k, step 1)	Ω

%MinElecVal=0+1*X=807 Ω

X=807 =11 0010 0111

%MaxElecVal	Maximum electrical output	CAL	13	ConRes (0 to 8.2 k, step 1)	Ω

%MaxElecVal=0+1*X=2925 Ω

X=2925=101 1011 0110

7.4.5.3.4 ConRelRes

The constant relative resolution is calculated using the following equation.

property name = <*start_value*> * [1 + 2 * <*tolerance*>] ^ <*teds_value*>

%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s

%RespTime=$1*10^{-6}[1+2*0,46]^X$=0,1 S

$1*10^{-6}[1,292]^X$=0,1 S

X*ln(1,292)=ln(100000)

X=45= 10 1101

%ExciteAmplNom	Excitation current, nom	CAL	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A

%ExciteAmplNom=$1*10^{-6}[1+2*0,023]^X$=0,003 A

$1*10^{-6}[1,046]^X$=0,003

X*ln(1,046)=ln(3000)

X=178= 1011 0010

%ExciteAmplMax	Excitation current, max	ID	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A

%ExciteAmplMax=$1*10^{-6}[1+2*0,023]^X$=0,04 A

$1*10^{-6}[1,046]^X$=0,003

X*ln(1,046)=ln(40000)

X=236= 1110 1100

7.4.5.2 Standard data and string types <dtype>

7.4.5.2.1 Date

The date data type is an integer that counts days since 1 January 1998. The first day, 1 January 1998 is day 0. All "1" in RAW TEDS (binary) shall denote property not used.

Example:

```
%CalDate, "date of calibration", CAL, 14, Date, "", ""
```

The binary input for this might be:

```
00000000011111
```

which would be translated into:

1 February 1998.

Note that 14 bits provides dates into the year 2032.

Desde 1/2/1998 hasta 1/2/2018 = 7330 dias
001 1100 1010 0010

7.4.5.2.5 Chr5

This is a character set intended to use a minimal number of bits for the characters. Table 9 provides the encoding.

Example:

```
%CalInitials, "person who calibrated", CAL, 15, Chr5, "", ""
```

The binary input for this might be:

00001 00010 00011

which would be translated into:

ABC

Table 9—Chr5 Values

		lsb							
		000	001	010	011	100	101	110	111
msb	00	Space	A	B	C	D	E	F	G
	01	H	I	J	K	L	M	N	O
	10	P	Q	R	S	T	U	V	W
	11	X	Y	Z	,	.	/	_	@

LNC= 01100 01110 00011

3.3.4.6 Montaje y resultados

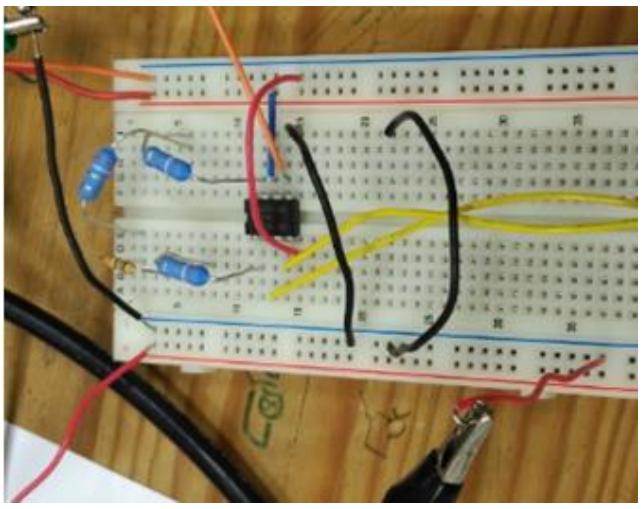
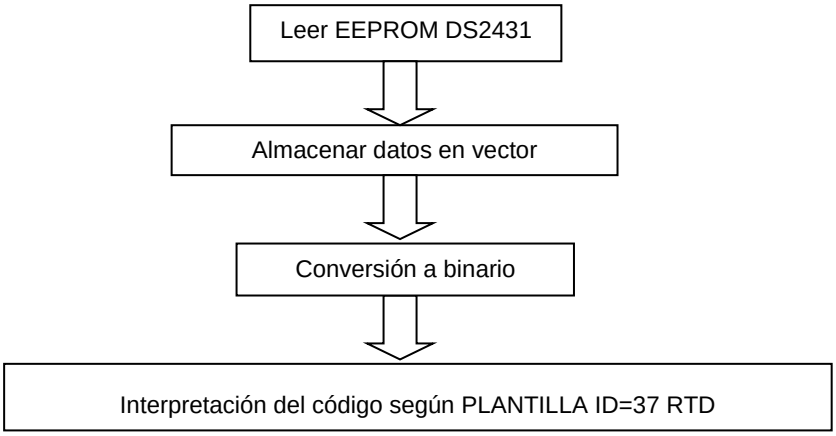


Figura 3.3.4.6.1: Montaje PT1000 en Puente de Wheastone

Una vez realizado el montaje se procede a cargar el programa y a visualizar los resultados en el monitor serie de Arduino.

Lo primero es programar el Arduino para leer el contenido de la EEPROM y almacenarlo en una matriz para luego poder tratar los datos.



Una vez leída la información de la EEPROM se procede a la conversión a binario.

CODIGO COMPLETO HEXADECIMAL (32 bytes)

00 00 00 00 00 00 00 00 09 44 B2 BC 64 EB 6D 8D B6 5D 83 94 4C 70 CB 68 00 00 00
00 00 00 00 00

Durante el proceso de decodificación, se identifica el ID de la plantilla (en este caso ID=37), para posteriormente interpretar el código bit a bit siguiendo el patrón de la plantilla (Anexo A, estándar IEEE1451.4)

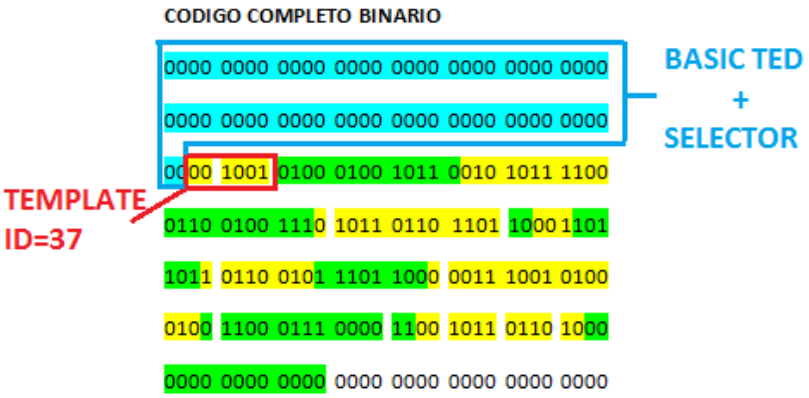


Figura 3.3.4.6.2: Patron de la plantilla ID=37

El primer resultado que se muestra por pantalla es el código de la ROM de la EEPROM. Este código es único e identifica al sensor dentro de la red.

Figura 3.3.4.6.3: Monitor serie Arduino. Código ROM EEPROM sensor PT1000

Lo siguiente que se muestra por pantalla es los datos leídos de la EEPROM y que han sido almacenados

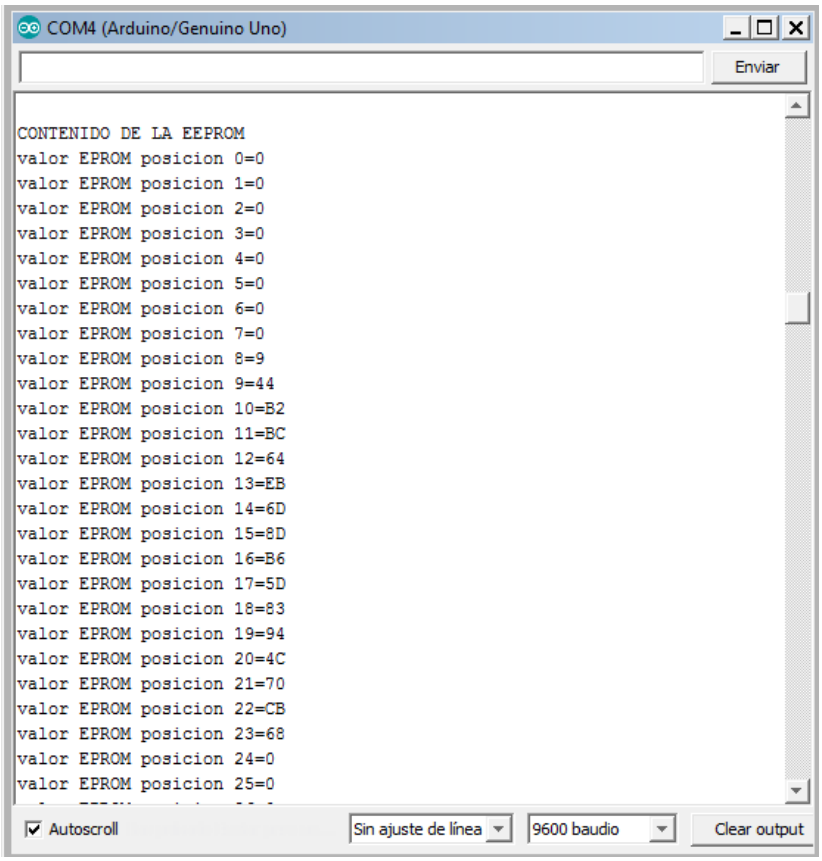


Figura 3.3.4.6.4: Monitor serie Arduino. Contenido EEPROM PT1000

Una vez almacenado los datos se procede a la conversión a binario.

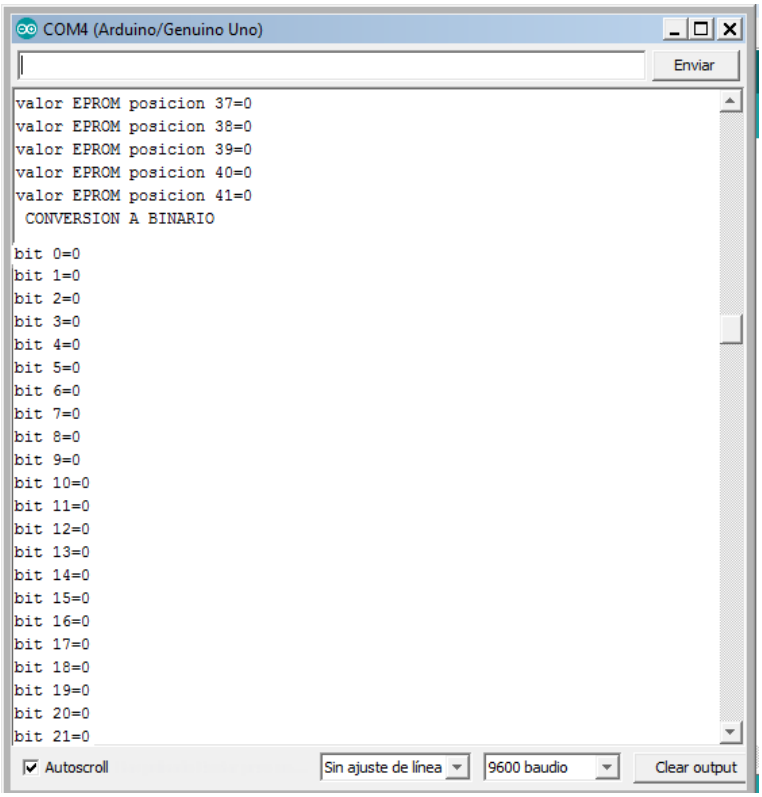


Figura 3.3.4.6.5: Monitor serie Arduino.Decodificación binaria EEPROM PT1000

Y finalmente se interpreta este código para mostrar los datos de la plantilla y la lectura del sensor.

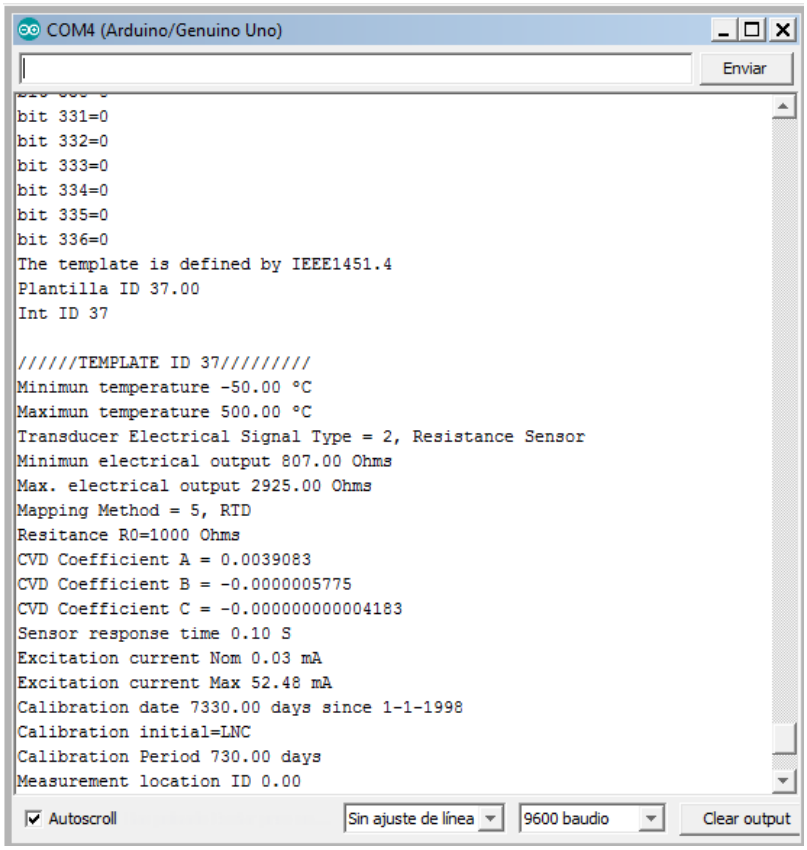


Figura 3.3.4.6.6: Monitor serie Arduino.Interpretación plantilla ID 37 PT1000

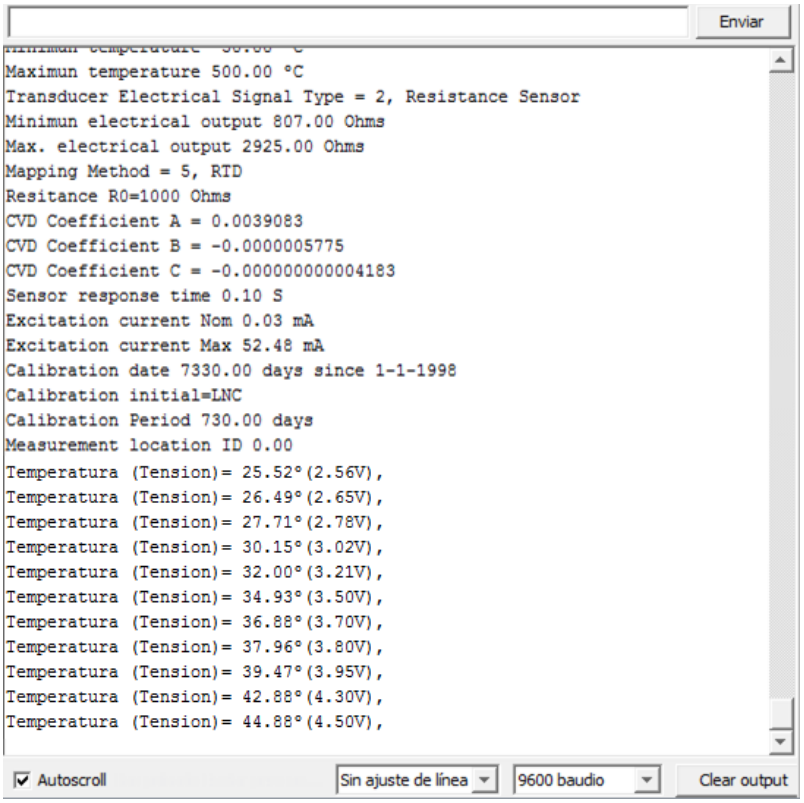


Figura 3.3.4.6.7: Monitor serie Arduino.Resultados de la lectura del sensor PT1000

3.3.4.7 Programación

Para obtener una ecuación que relacione la tensión que le llega a la entrada analógica del Arduino con la temperatura que mide el sensor se utilizan las siguientes ecuaciones:

La salida de la primera etapa de medición (puente de wheastone):

$$V_s = V_{s+} - V_{s-} = \frac{n*\alpha*T}{(n+1)^2} * Vin = \frac{100*0{,}003908*Temperatura}{(100+1)^2} * 5$$

[Ecuación 3.3.4.7.1]

La salida de la etapa de amplificación o lo que lo mismo, la entrada al Arduino:

$$Vi = Vs*G = Vs*523{,}013$$ [Ecuación 3.3.4.7.2]

Con estas 2 ecuaciones y despejando la temperatura en función de la tensión de entrada al Arduino V_i se obtiene la ecuación necesaria para introducir en la programación del microcontrolador:

$$Temperatura = 9{,}9817*Vi$$ [Ecuación 3.3.4.7.3]

CÓDIGO ARDUINO: ESCRITURA/LECTURA EEPROM DS2431 DE LA INFORMACION DEL SENSOR PT1000

```
#include <OneWire.h>


OneWire  ds(2);   // 1-wire on pin 2
byte    addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)
void setup(void)
{
 Serial.begin(9600);
}


void loop(void)
{
 //byte i;
 byte dat[15];


 SearchAddress(addr); //lectura y comprobación de la ROM


 //Grabamos los DATOS DE LA PLANTILLA en la EPROM DS2431
```

```
dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(0,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x09;
dat[1] = 0x44;
dat[2] = 0xB2;
dat[3] = 0xBC;
dat[4] = 0x64;
dat[5] = 0xEB;
dat[6] = 0x6D;
dat[7] = 0x8D;

WriteRow(1,dat); //(posición donde empieza a guardar los datos)

dat[0] = 0xB6;
dat[1] = 0x5D;
dat[2] = 0x83;
dat[3] = 0x94;
dat[4] = 0x4C;
dat[5] = 0x70;
dat[6] = 0xCB;
dat[7] = 0x68;

WriteRow(2,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
```

```
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(3,dat);  //(posición donde empieza a guardar los datos)
dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(4,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(5,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
```

```
  dat[4] = 0x00;
  dat[5] = 0x00;
  dat[6] = 0x00;
  dat[7] = 0x00;

  WriteRow(6,dat);  //(posición donde empieza a guardar los datos)

  dat[0] = 0x00;
  dat[1] = 0x00;
  dat[2] = 0x00;
  dat[3] = 0x00;
  dat[4] = 0x00;
  dat[5] = 0x00;
  dat[6] = 0x00;
  dat[7] = 0x00;

  WriteRow(15,dat);  //(posición donde empieza a guardar los datos)

  ReadAllMem();    //print all mem content

  while(1);
}

void SearchAddress(byte* address) //Search for address and confirm it
// Esta función lee el código grabado por LASER de la EEPROM que la hace única
{
  int i;
  if ( !ds.search(address))
  {
    Serial.print("No device found.\n");
    ds.reset_search();
    delay(250);
    return;
  }
```

```
  Serial.print("ADDR= ");
  for( i = 0; i < 8; i++)
  {
    Serial.print(address[i], HEX);
    Serial.print(" ");
  }
  if ( OneWire::crc8( address, 7) != address[7])
  {
     Serial.print("CRC is not valid, address is corrupted\n");
     return;
  }

  if ( address[0] != 0x2D)
  {
     Serial.print("Device is not a 1-wire Eeprom.\n");
     return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);       // Read Scratchpad
```

```
  for ( i = 0; i < 13; i++)
    data[i] = ds.read();
 }

 void CopyScratchPad(byte* data)
 {
  ds.reset();
  ds.select(addr);
  ds.write(0x55,1);  // Copy ScratchPad
  ds.write(data[0],1);
  ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
  ds.write(data[2],1);
  delay(25); // Waiting for copy completion
  //Serial.print("Copy done!\n");
 }

 void ReadAllMem()
 {
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0xF0,1);  // Read Memory
  ds.write(0x00,1);  //Read Offset 0000h
  ds.write(0x00,1);

  for ( i = 0; i < 144; i++) //whole mem is 144
  {
    Serial.print("byte");
    Serial.print("(");
    Serial.print(i);
    Serial.print(") ");
    Serial.print(ds.read(), HEX);
    Serial.println(" ");
  }
  Serial.println();
```

```
}

void WriteRow(byte row, byte* buffer)
{
 int i;
 if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
  return;              //The remaining are for the 64 bits register page

 WriteReadScratchPad(row*8, 0x00, buffer);

 /*  Print result of the ReadScratchPad
 for ( i = 0; i < 13; i++)
 {
  Serial.print(buffer[i], HEX);
  Serial.print(" ");
 }
 */
 CopyScratchPad(buffer);

}
```

CÓDIGO ARDUINO: EEPROM DS2431 LECTURA Y DECODIFICACIÓN DE LA INFORMACION DEL SENSOR PT1000

```
#include <OneWire.h>

OneWire  ds(2);    // 1-wire on pin 2
byte     addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)

unsigned char EPROM[145]; //vector de la EPROM DS2431
unsigned char BINARIO[336];
int valor;
int posicion;
int i; //i es la posición del byte de la EPROM[] en la que nos encontramos
int j; //j es la posición de bit del vector BINARIO[]
```

```
float base;
float ID;
int IDint; //variable para pasar ID (tipo float) a IDint (entero)
float pot=0; //
float Vi;
float Vo;
float Temperatura;

//variables para Template ID 37
float minphisval; //Minimun Temperature
float maxphisval; //Maximun Temperature
float minelecval; //Minimun electrical output
float maxelecval; //Max. electrical output
int R0; //R0 Resistance
float RTDcurve; //RTD curve
double RespTime; // Sensor response time
double ExciteAmplNom; //Excitation current, nom
double ExciteAmplMax; //Excitation current, max
float CalDate; // Calibration date
char Calinitials; //Calibration initials
float CalPeriod; //Calibration Period
float MeasID; //Measurement location ID



void setup()
{
Serial.begin(9600);

Serial.println("ROM identificacion del sensor");
SearchAddress(addr); //lectura y comprobación de la ROM
Serial.println(" ");

ReadAllMem();    //print all mem content
```

```
      //visualizo todo el vector EPROM[] 42 bytes


      Serial.println("CONTENIDO DE LA EEPROM ");
       for ( i = 0; i < 42; i++){ // "i" va a ser un valor que enviaremos al SLAVE como valor de
posición de la EPROM


       Serial.print("valor EPROM posicion ");
       Serial.print(i);
       Serial.print("=");
       Serial.println(EPROM[i],HEX); //mostramos el valor almacenado
       delay(100);
       }
      //convierto los datos del vector EPROM[] a binario y almaceno los datos en el vector
BINARIO[]


       for ( i = 0; i < 42; i++){

      /* Serial.println(" ");
        Serial.print("///////////////byte ");
        Serial.println(i);
      */
         //i es la posición del byte que nos encontramos de la EPROM


        valor=EPROM[i];


        //usamos una variable "j" para posicionarnos de 7 al 0 (8bits de derecha a izquierda))
        for ( j = 7; j >= 0; j--){


         posicion=i*8+j;


      /*   Serial.print("bit ");
         Serial.println(posicion); //mostramos valores de J
      */


         //detectamos si es par o impar y hacemos las divisiones
          if (valor & 0x01){
```

```
      //Serial.println("es impar");
      valor=valor-1;
      valor=valor/2;
      BINARIO[posicion]=1;// como es impar el resto vale 1
      }
      else {
      //Serial.println("es par");
      valor=valor/2;
      BINARIO[posicion]=0;
      }

/*   Serial.println(BINARIO[posicion]); //mostramos valores de J
     Serial.println(" ");
     delay(100);
*/
   }
 }
Serial.print(" ");
Serial.println("CONVERSION A BINARIO ");

 for ( i = 0; i < 337; i++){
 base=BINARIO[i];
 Serial.print("bit ");
 Serial.print(i);
 Serial.print("=");
 Serial.println(BINARIO[i]);
 }


 //SELECTOR (2 bits 64 y 65)
 if (BINARIO[64]==0 & BINARIO[65]==0) {
      Serial.println("The template is defined by IEEE1451.4 ");
  }
```

```
    //decodificacion e identifiación de la ID de la plantilla, 8 bits del 66 al 73 del vector BINARIO[]

     for ( i = 66; i < 74; i++){
     base=BINARIO[i];
     ID=ID+base*pow(2,73-i);
      }

     Serial.print("Plantilla ID ");
     Serial.println(ID);
      IDint= int(ID);
      IDint= IDint+1;
     Serial.print("Int ID ");
     Serial.println(IDint);


    }//fin de void setup


    void loop()
    {



     //PLANTILLA ID 37
     if (IDint == 37){

     Serial.println(" ");
     Serial.println("//////TEMPLATE ID 37/////////");


    //    delay(1000);
        minimun_temperature();
    //    delay(1000);
        maximun_temperature();
```

```
//    delay(1000);
   Serial.println("Transducer Electrical Signal Type = 2, Resistance Sensor");
//    delay(1000);
   minimun_electrical_output();
//    delay(1000);
   Max_electrical_output();
//    delay(1000);
   Serial.println("Mapping Method = 5, RTD");
//    delay(1000);
   R0_resistance();
//    delay(1000);
   RTD_curve();
//    delay(1000);
   Sensor_response_time();
//    delay(1000);
   Excitation_current_nom();
//    delay(1000);
   Excitation_current_max();
//    delay(1000);
   Calibration_date();
//    delay(1000);
   Calibration_initials();
 //    delay(1000);
   Calibration_period();
//    delay(1000);
   Measurement_location_ID();
//    delay(1000);


  IDint = 0; //reseteamos la variable ID

//LECTURA DEL SENSOR

  for ( i = 0; i < 11; i++){ // mido 10 valores
```

```
   Vi=analogRead(0);
   Vo=(5*Vi)/1023;

   Temperatura=9.9817*Vo;
   Serial.print("Temperatura (Tension)= ");
   Serial.print(Temperatura);
   Serial.print("º");
   Serial.print("(");
   Serial.print(Vo);
   Serial.print("V");
   Serial.print(")");
   Serial.println(", ");
   delay(1000);
  }

  }// fin IF ID37

} //Fin de void loop()


///FUNCIONES  PLANTILLA 37/////////////////////////////////////////////////////


void minimun_temperature(){
      //Minimun Temperature("%Min Phys Val")
      for ( i = 74; i < 85; i++){  //bits del 74 al 84
      base=BINARIO[i];
      minphisval=minphisval+base*pow(2,84-i);
      delay(100);
      }
      minphisval=minphisval-200;
      Serial.print("Minimun temperature ");
      Serial.print(minphisval);
      Serial.println(" ºC");
}
```

```
void maximun_temperature(){
      //Maximun Temperature ("%Max Phys Val")
      for ( i = 85; i < 96; i++){  //bits del 85 al 95
      base=BINARIO[i];
      maxphisval=maxphisval+base*pow(2,95-i);
      delay(100);
      }
      maxphisval=maxphisval-200;
      Serial.print("Maximun temperature ");
      Serial.print(maxphisval);
      Serial.println(" ºC");
}

void minimun_electrical_output(){
      //Minimun electrical output ("%Min Elec Val")
      for ( i = 96; i < 107; i++){  //bits del 96 al 106
      base=BINARIO[i];
      minelecval=minelecval+base*pow(2,106-i);
      delay(100);
      }
      Serial.print("Minimun electrical output ");
      Serial.print(minelecval);
      Serial.println(" Ohms");
}

void Max_electrical_output(){
      //Max. electrical output
      for ( i = 107; i < 120; i++){  //bits del 107 al 119
      base=BINARIO[i];
      maxelecval=maxelecval+base*pow(2,119-i);
      delay(100);
      }
      Serial.print("Max. electrical output ");
      Serial.print(maxelecval);
```

```
        Serial.println(" Ohms");
    }

    void R0_resistance(){
        //R0 resistance (%RTDCoef_R0, selectcase, bits 120 y 121)
        if (BINARIO[120]==0 & BINARIO[121]==0) {
        Serial.println("Resitance R0=100 Ohms ");
        }
        else if (BINARIO[120]==0 & BINARIO[121]==1) {
        Serial.println("Resitance R0=120 Ohms ");
        }
        else if (BINARIO[120]==1 & BINARIO[121]==0) {
        Serial.println("Resitance R0=1000 Ohms ");
        }
        else if (BINARIO[120]==1 & BINARIO[121]==1) {
        Serial.println("Resitance R0= ");
        }
    }

    void RTD_curve(){
        //RTD curve (selectcase, bits 122,123 y 124)
    //    if (BINARIO[122]==0 & BINARIO[123]==0 & BINARIO[124]==0) {
    //     Serial.println("CVD Coefficient A = 0.00381 ");
    //     Serial.println("CVD Coefficient B = -0.000000602");
    //     Serial.println("CVD Coefficient C = -0.000000000006");
    //     }
    //     else if (BINARIO[122]==0 & BINARIO[123]==0 & BINARIO[124]==1) {
    //     Serial.println("CVD Coefficient A = 0.0039083 ");
    //     Serial.println("CVD Coefficient B = -0.0000005775 ");
    //     Serial.println("CVD Coefficient C = -0.000000000004183");
    //     }
    //     else if (BINARIO[122]==0 & BINARIO[123]==1 & BINARIO[124]==0) {
    //     Serial.println("CVD Coefficient A = 0.0039692");
    //     Serial.println("CVD Coefficient B = -0.000000584");
    //     Serial.println("CVD Coefficient C = -0.000000000004");
```

```
//      }
//      else if (BINARIO[122]==0 & BINARIO[123]==1 & BINARIO[124]==1) {
//      Serial.println("CVD Coefficient A = 0.0039739");
//      Serial.println("CVD Coefficient B = -0.000000587");
//      Serial.println("CVD Coefficient C = -0.000000000004");
//      }
//      else if (BINARIO[122]==1 & BINARIO[123]==0 & BINARIO[124]==0) {
//      Serial.println("CVD Coefficient A = 0.0039787");
//      Serial.println("CVD Coefficient B = -0.000000586");
//      Serial.println("CVD Coefficient C = -0.000000000004");
//      }
//      else if (BINARIO[122]==1 & BINARIO[123]==0 & BINARIO[124]==1) {
//      Serial.println("CVD Coefficient A = 0.003988");
//      Serial.println("CVD Coefficient B = -0.000000591");
//      Serial.println("CVD Coefficient C = -0.000000000003");
//      }
//      else if (BINARIO[122]==1 & BINARIO[123]==1 & BINARIO[124]==0) {
//      Serial.println("CVD Coefficient A,B,C = custom");
//      }
//      else if (BINARIO[122]==1 & BINARIO[123]==1 & BINARIO[124]==1) {
//      Serial.println("CVD Coefficient A,B,C = custom");
//      }

        //COMO ESTA FUNCION OCUPA DEMASIADA MEMORIA DEL MICRO ME QUEDO CON
EL CASO QUE ME INTERESA
        if (BINARIO[122]==0 & BINARIO[123]==0 & BINARIO[124]==1) {
        Serial.println("CVD Coefficient A = 0.0039083 ");
        Serial.println("CVD Coefficient B = -0.0000005775 ");
        Serial.println("CVD Coefficient C = -0.000000000004183");
        }
    }

    void Sensor_response_time(){
        //Senor response time
        for ( i = 125; i < 131; i++){  //bits del 125 al 130
        base=BINARIO[i];
```

```
        RespTime=RespTime+base*pow(2,130-i); //CUIDADO las potencias se usan variables FLOAT si no redondea MAL!!!
        delay(100);
        }
        RespTime= 0.000001*pow(1.292,RespTime);
        Serial.print("Sensor response time ");
        Serial.print(RespTime);
        Serial.println(" S");

    }

    void Excitation_current_nom(){
        //Excitation current, nom.
        for ( i = 67; i < 75; i++){  //bits del 131 al 138
        base=BINARIO[i];
        ExciteAmplNom=ExciteAmplNom+base*pow(2,74-i);
        delay(100);
        }
        ExciteAmplNom= 0.001*pow(1.0471285480509,ExciteAmplNom);
        Serial.print("Excitation current Nom ");
        Serial.print(ExciteAmplNom);
        Serial.println(" mA");
    }

    void Excitation_current_max(){
        //Excitation current, max
            for ( i = 139; i < 147; i++){  //bits del 139 al 146
        base=BINARIO[i];
        ExciteAmplMax=ExciteAmplMax+base*pow(2,146-i);
        delay(100);
        }
        ExciteAmplMax= 0.001*pow(1.0471285480509,ExciteAmplMax);
        Serial.print("Excitation current Max ");
        Serial.print(ExciteAmplMax);
        Serial.println(" mA");
    }
```

```
void Calibration_date(){
    //Calibration date
    for ( i = 147; i < 163; i++){  //bits del 147 al 162
    base=BINARIO[i];
    CalDate=CalDate+base*pow(2,162-i); //CUIDADO las potencias se usan variables FLOAT si no redondea MAL!!!
    delay(100);
    }
    Serial.print("Calibration date ");
    Serial.print(CalDate);
    Serial.println(" days since 1-1-1998");
}

 void Calibration_initials(){
    //Calibration initial, bits 163 al 177
    if (BINARIO[163]==0 & BINARIO[164]==1 & BINARIO[165]==1 & BINARIO[166]==0 & BINARIO[167]==0 & BINARIO[168]==0 & BINARIO[169]==1 & BINARIO[170]==1 & BINARIO[171]==1 & BINARIO[172]==0 & BINARIO[173]==0 & BINARIO[174]==0 & BINARIO[175]==0 & BINARIO[176]==1 & BINARIO[177]==1 ) {
    Serial.println("Calibration initial=LNC");
     }

}

void Calibration_period(){
    //Calibration period
    for ( i = 178; i < 190; i++){  //bits del 178 al 189
    base=BINARIO[i];
    CalPeriod=CalPeriod+base*pow(2,189-i);
    delay(100);
    }
    Serial.print("Calibration Period ");
    Serial.print(CalPeriod);
    Serial.println(" days");
}
```

```
void Measurement_location_ID(){
    //Measurement location ID
    for ( i = 190; i < 201; i++){  //bits del 190 al 200
    base=BINARIO[i];
    MeasID=MeasID+base*pow(2,200-i);
    delay(100);
    }
    Serial.print("Measurement location ID ");
    Serial.println(MeasID);
}

//FUNCIONES DE LA EEPROM DS2431 ONE WIRE
void SearchAddress(byte* address) //Search for address and confirm it
// Esta funcion lee el código ROM gravado por LASER de la EEPROM que la hace única
{
 int i;
 if ( !ds.search(address))
 {
  Serial.print("No device found.\n");
  ds.reset_search();
  delay(250);
  return;
 }

 Serial.print("ADDR= ");
 for( i = 0; i < 8; i++)
 {
  Serial.print(address[i], HEX);
  Serial.print(" ");
 }

 if ( OneWire::crc8( address, 7) != address[7])
 {
    Serial.print("CRC is not valid, address is corrupted\n");
```

```
    return;
  }

  if ( address[0] != 0x2D)
  {
    Serial.print("Device is not a 1-wire Eeprom.\n");
    return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);        // Read Scratchpad

  for ( i = 0; i < 13; i++)
    data[i] = ds.read();
}

void CopyScratchPad(byte* data)
{
  ds.reset();
  ds.select(addr);
  ds.write(0x55,1);  // Copy ScratchPad
```

```
 ds.write(data[0],1);
 ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
 ds.write(data[2],1);
 delay(25); // Waiting for copy completion
 //Serial.print("Copy done!\n");
}

void ReadAllMem()
{
 int i;
 ds.reset();
 ds.select(addr);
 ds.write(0xF0,1);  // Read Memory
 ds.write(0x00,1);  //Read Offset 0000h
 ds.write(0x00,1);

 for ( i = 0; i < 144; i++) //whole mem is 144
 {
//   Serial.print("byte");
//   Serial.print("(");
//   Serial.print(i);
//   Serial.print(") ");
//   Serial.print(ds.read(), HEX);
   EPROM[i]=ds.read();
   Serial.println(" ");
 }
 Serial.println();
}
void WriteRow(byte row, byte* buffer)
{
 int i;
 if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
   return;             //The remaining are for the 64 bits register page

 WriteReadScratchPad(row*8, 0x00, buffer);
```

```
/*  Print result of the ReadScratchPad
for ( i = 0; i < 13; i++)
{
  Serial.print(buffer[i], HEX);
  Serial.print(" ");
}
*/
CopyScratchPad(buffer);

}
```

3.3.5 Sensor LDR

3.3.5.1 Introducción

Una fotorresistencia LDR (*Light-Dependent Resistor*) es un sensor de luz de tipo resistivo que consiste en un dispositivo en el cual su resistencia varía en función de la luz que incide sobre el mismo. Se puede emplear esta variación para medir, a través de las entradas analógicas de un microcontrolador, una estimación del nivel de luz.

Su comportamiento es el siguiente:

- Mayor luminosidad = menor resistencia eléctrica
- Menor luminosidad = mayor resistencia eléctrica

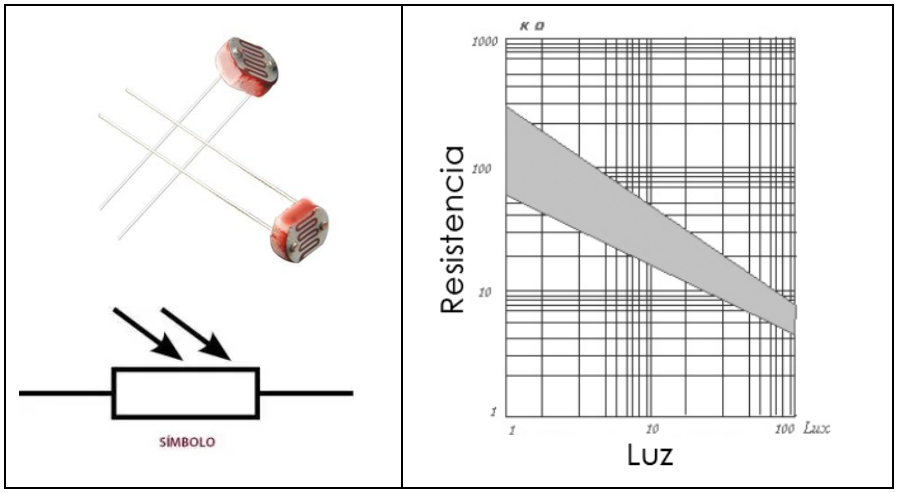


Figura 3.3.5.1.1: Variación de la resistencia de una LDR en función de la luminosidad incidente

En la LDR, la relación entre la intensidad lumínica y el valor de la resistencia no tiene un comportamiento lineal. Estos valores pueden ser obtenidos del Datasheet del componente.

Modelo	Voltaje (V)	Temperatura (°C)	Pico espectral (nm)	Resistencia luz brillante (KΩ)	Resistencia oscuridad (KΩ)	gamma	Tiempo respuesta (ms)
GL5516	150	-30°+70°	540	5-10	500	0.5	30
GL5528	150	-30°+70°	540	10-20	1000	0.6	25
GL5537-1	150	-30°+70°	540	20-30	2000	0.6	25
GL5537-2	150	-30°+70°	540	30-50	3000	0.7	25
GL5539	150	-30°+70°	540	50-100	5000	0.8	25
GL5549	150	-30°+70°	540	100-200	10000	0.9	25

Figura 3.3.5.1.2: Valores provenientes del Datasheet de la LDR empleada

Tal y como se aprecia en la tabla anterior, un fotoresistor disminuye su resistencia a medida que aumenta la luz incidente sobre él. Los valores típicos son de 1 Mohm en total oscuridad y 10-20 Ohm bajo luz brillante.

Tienen el inconveniente que siempre existirán pequeñas variaciones entre LDR´s, incluso dentro de la misma familia, ocasionados por la fabricación del componente.

Este inconveniente hace que estas pequeñas diferencias, supongan grandes variaciones en la medición, por lo que no es posible emplear estos valores de forma directa sin un proceso previo de calibración.

Lo primero que debemos de efectuar después de esta calibración, es el acondicionamiento de la señal obtenida del sensor. Para ello, el circuito más sencillo es el divisor de tensión, también conocido como divisor de voltaje.

Divisor de tensión: Mediante un par de resistencias en serie, es posible repartir la tensión suministrada por la fuente entre los terminales de estas.

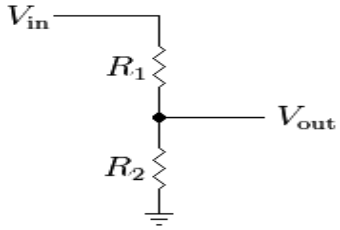


Figura 3.3.5.1.3: Divisor de tensión común

Se conecta Vin a uno de los extremos de las resistencias. Al conducirse corriente a través de estas dos resistencias, se produce un voltaje en el punto donde se unen, Vout, cuyo valor puede determinarse con la fórmula:

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

[Ecuación 3.3.5.1.1]

En este caso con el sensor, la LDR es R1, y R2 es una resistencia que se ha dimensionado de un (1k). Se utiliza el divisor de tensión con la LDR para obtener un voltaje variable Vout de acuerdo a la cantidad de luz percibida.

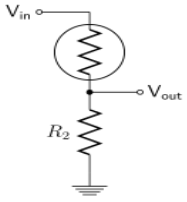


Figura 3.3.5.1.4 Divisor de tensión empleado

Utilizando está configuración en el montaje, en el divisor se obtiene que a **mayor luz, mayor será el voltaje** a la salida (cuanta más luz incida sobre la LDR, menor será la resistencia eléctrica y mayor será la tensión a la salida).

La principal ventaja de las LDR frente a otros sensores ópticos es que es un sensor más económico, y la principal diferencia es la velocidad de respuesta y la linealidad. Mientras que los fotodiodos son casi lineales y muy rápidos, las LDR son más lentas y menos lineales.

Las aplicaciones más sencillas y habituales de las LDR son las relacionadas con la automatización de la iluminación. Por ejemplo:

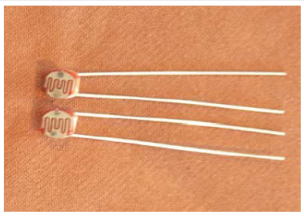
- Sensor crepuscular.
- Detector de luz y oscuridad
- Encendido automático de luces exteriores y farolas
- Iluminación automática en automóviles

Figura 3.3.5.1.5: Ejemplos de distintas aplicaciones de las LDR

3.3.5.2 Características técnicas del sensor

Las especificaciones según el fabricante del sensor LDR utilizado en este trabajo son las siguientes:

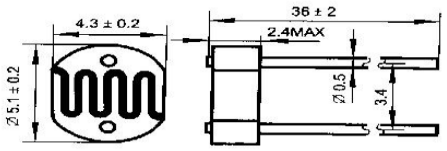
CdS PHOTOCONDUCTIVE CELLS **GL5528**



- Epoxy encapsulated
- Quick response
- Small size
- High sensitivity
- Reliable performance
- Good characteristic of spectrum

Light Resistance at 10Lux (at 25℃)	8～20KΩ
Dark Resistance at 0 Lux	1.0MΩ(min)
Gamma value at 100-10Lux	0.7
Power Dissipation(at 25℃)	100mW
Max Voltage (at 25℃)	150V
Spectral Response peak (at 25℃)	540nm
Ambient Temperature Range:	- 30～+70℃

Outline



Modelo	Voltaje (V)	Temperatura (°C)	Pico espectral (nm)	Resistencia luz brillante (KΩ)	Resistencia oscuridad (KΩ)	gamma	Tiempo respuesta (ms)
GL5516	150	-30°+70°	540	5-10	500	0.5	30
GL5528	150	-30°+70°	540	10-20	1000	0.6	25
GL5537-1	150	-30°+70°	540	20-30	2000	0.6	25
GL5537-2	150	-30°+70°	540	30-50	3000	0.7	25
GL5539	150	-30°+70°	540	50-100	5000	0.8	25
GL5549	150	-30°+70°	540	100-200	10000	0.9	25

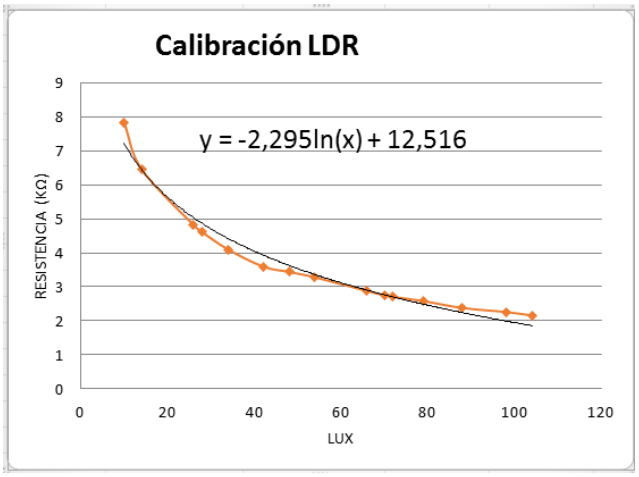
Figura 3.3.5.2.1: Especificaciones de la LDR

3.3.5.3 Calibración del sensor

El procedimiento a seguir para calibrar el sensor es obtener y registrar los valores de resistencia de la LDR cuando es sometida a una iluminación dada. Para realizar esta tarea hemos empleado un polímetro para medir el valor resistivo de la LDR y un luxómetro para medir la intensidad lumínica que incide en el sensor en cada momento.

Figura 3.3.5.3.1: Calibración de la LDR mediante el luxómetro del teléfono móvil

Tras realizar el ensayo, he registrado y representado gráficamente los valores en una tabla EXCEL, y generado la curva de calibración mediante una aproximación logarítmica.



Lux	KΩ
10	7,84
14	6,47
26	4,82
28	4,61
34	4,1
42	3,6
48	3,44
54	3,28
66	2,88
70	2,75
72	2,71
79	2,58
88	2,38
98	2,26
104	2,16

Figura 3.3.5.3.2. Aproximación de la curva logarítmica de la LDR

De esta forma se obtiene la ecuación de la curva de calibración que define el comportamiento del sensor frente a la variación de la intensidad lumínica.

3.3.5.4 Simulación OrCAD

Circuito de acondicionamiento (divisor de tensión):

Se realiza el montaje en OrCAD del divisor de tensión asignándole a la LDR una resistencia de valor igual a la ecuación de la curva logarítmica que define el comportamiento del sensor.

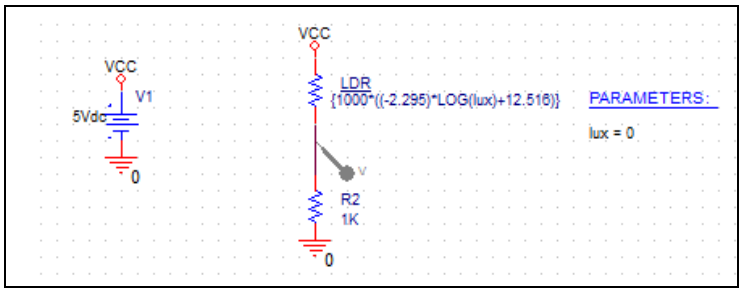


Figura 3.3.5.4.1 Divisor de tensión con LDR

$$R_{LDR} = -2{,}295 \times \ln(LUX) + 12{,}516 \qquad \text{[Ecuación 3.3.5.4.1]}$$

$$V_{out} = V_{cc} \times \frac{R_2}{R_{LDR}+R_2} \qquad \text{[Ecuación 3.3.5.4.2]}$$

$$V_{out} = 5V \times \frac{1K\Omega}{R_{LDR}+1K\Omega} \qquad \text{[Ecuación 3.3.5.43]}$$

Una vez simulado el circuito, se observa que la gráfica de la simulación coincide con la calculada en EXCEL.

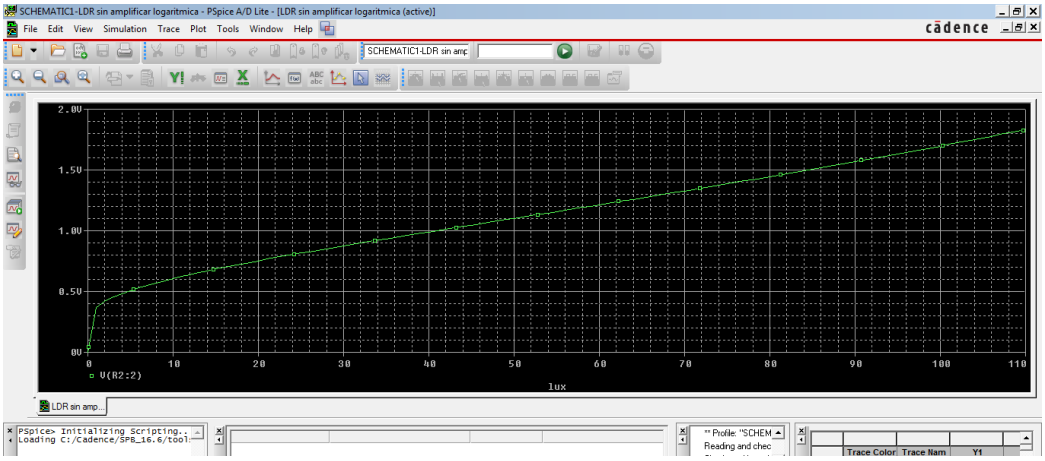
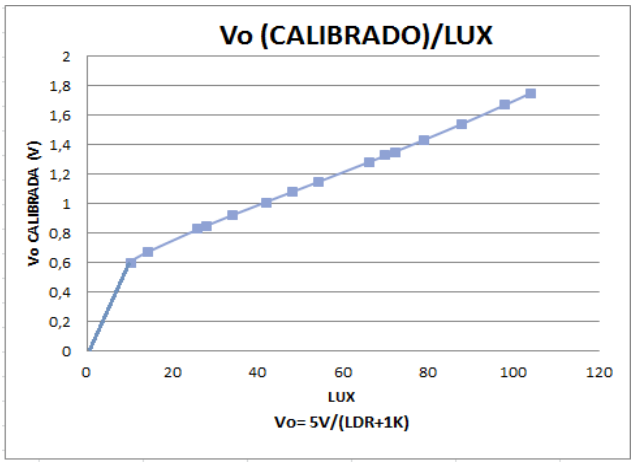


Figura 3.3.5.4.2: Simulación en OrCAD de la salida Vout



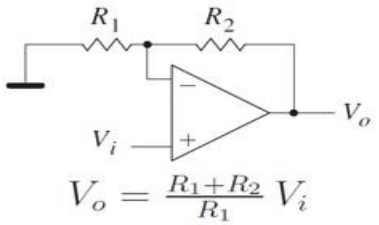
Lux	KΩ C.CALIBRACIÓN	Vo CALIBRADO
10	7,231567212	0,607417746
14	6,459363429	0,670298484
26	5,038668445	0,827997107
28	4,868590649	0,851993315
34	4,423002596	0,921998452
42	3,938048226	1,012545802
48	3,63159368	1,079542021
54	3,361281613	1,146451994
66	2,900742367	1,281807289
70	2,765703419	1,327773179
72	2,701051257	1,35096751
79	2,488117179	1,433438082
88	2,240512011	1,542966045
98	1,993499636	1,670285822
104	1,857122886	1,750012232

Figura 3.3.5.4.3 Curva de salida Vout obtenida en Excel

Etapa de amplificacion de la señal

Las entradas analógicas de Arduino sirven para medir voltajes entre 0 y 5V. Para una alimentación del sensor a 5V se obtiene a su salida una tensión de 0 a 1,8V, como hemos visto en el apartado anterior.

Por ello, es necesario utilizar un circuito adaptador de escala que amplifique la salida para obtener mejor resolución de medida.



Circuito 3.2.1. Amplificador no inversor común

Según los datos obtenidos en el OrCAD, mirando la tensión máxima que nos muestra y la tensión de 5 voltios que queremos obtener para este mismo valor, sacamos la ganancia que necesitamos en el adaptador de escala. Con ello y con la ecuación del circuito de la imagen anterior (asignándole a R1 el valor de 1KΩ), calculamos el valor de R2 como aproximadamente 2KΩ.

$$G = \frac{V_0}{V_i} = \frac{5}{1.8326} = 2.7284 \qquad \text{[Ecuación 3.2.1]}$$

$$G = \left(1 + \frac{R_2}{R_1}\right) = 2.7284 \qquad \text{[Ecuación 3.2.1]}$$

$$SI\ R_1 = 1K\Omega, entonces\ R_2 = 1{,}7284K\Omega \qquad \text{[Ecuación 3.2.1]}$$

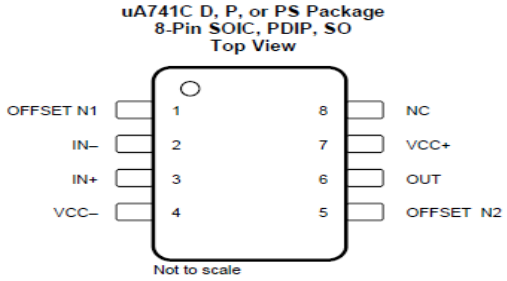


Figura 3.3.5.4.4: Esquema de un uA741

A continuación se realiza el montaje del circuito amplificador en OrCAD:

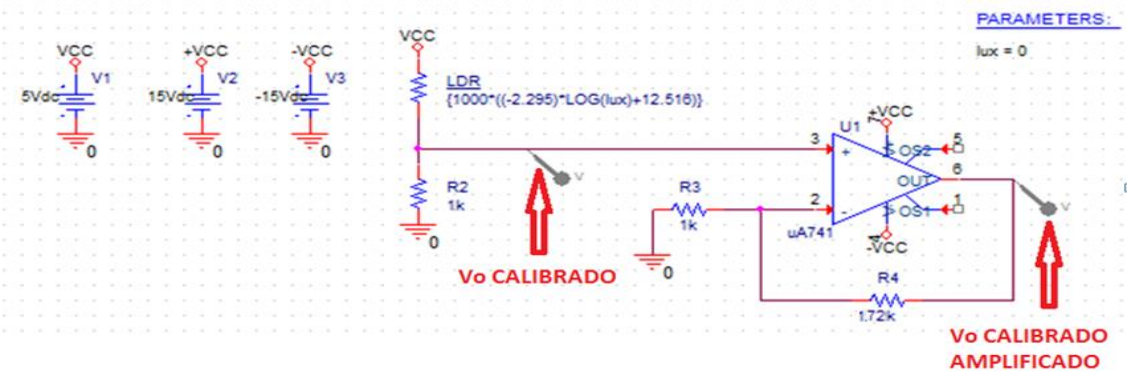


Figura 3.3.5.4.5 Amplificación de la señal LDR

Una vez simulado el circuito, se observa que la gráfica de la simulación coincide con la calculada en EXCEL.

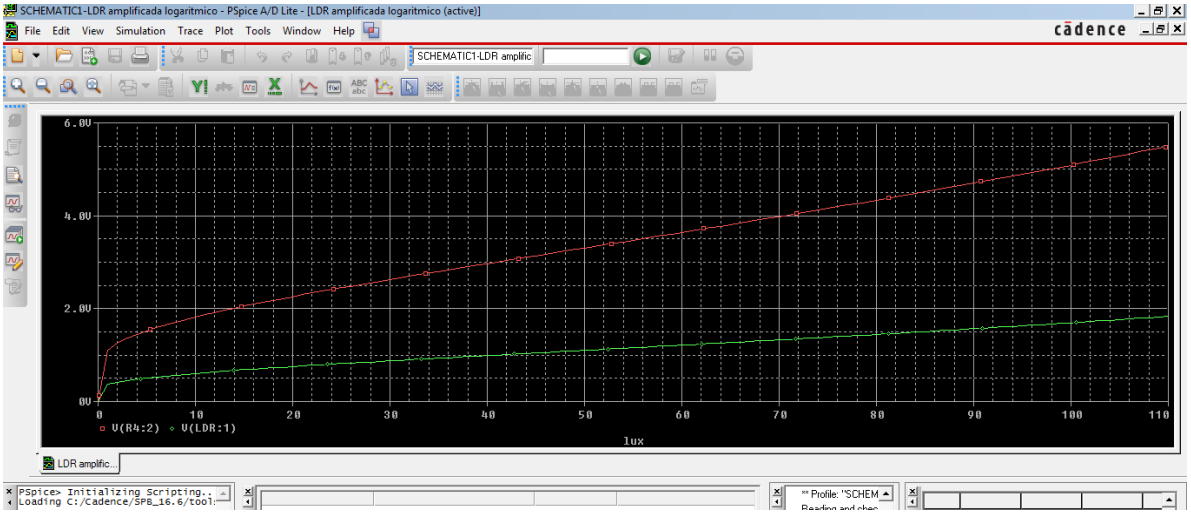
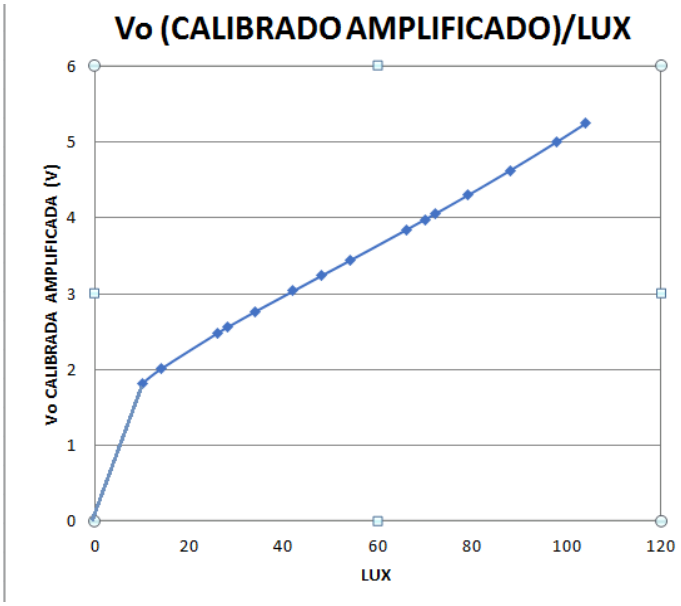


Figura 3.3.5.4.6. Simulación en OrCAD de la Salida de la etapa de amplificación / Entrada de señal al Arduino



Lux	KΩ C.CALIBRACIÓN	Vo CALIBRADO	Vo CALIBRADO AMPLIFICADO
10	7,231567212	0,607417746	1,822253237
14	6,459363429	0,670298484	2,010895453
26	5,038668445	0,827997107	2,48399132
28	4,868590649	0,851993315	2,555979944
34	4,423002596	0,921998452	2,765995357
42	3,938048226	1,012545802	3,037637405
48	3,63159368	1,079542021	3,238626062
54	3,361281613	1,146451994	3,439355981
66	2,900742367	1,281807289	3,845421868
70	2,765703419	1,327773179	3,983319537
72	2,701051257	1,35096751	4,052902529
79	2,488117179	1,433438082	4,300314247
88	2,240512011	1,542966045	4,628898134
98	1,993499636	1,670285822	5,010857465
104	1,857122886	1,750012232	5,250036696

Figura 3.3.5.4.7 Curva de salida de la etapa de amplificación obtenida en Excel

3.3.5.5 Implementación del estándar IEEE1451 en una LDR

Se implementa el diseño de un sensor inteligente LDR de la misma forma que el sensor PT1000, siguiendo lo indicado en el estándar IEEE1451 4:

1) Se emplea la arquitectura clase 2 de múltiples hilos por su sencilla integración. Tendremos una interface de modo mixto con 2 señales separadas: una analógica (la lectura del sensor) y otra digital (los datos de la TED).

2) El sensor LDR y la memoria EEPROM DS2431 formarán el el TIM.

3) El Arduino Uno será el encargado de leer, decodificar e interpretar la información trabajando como NCAP.

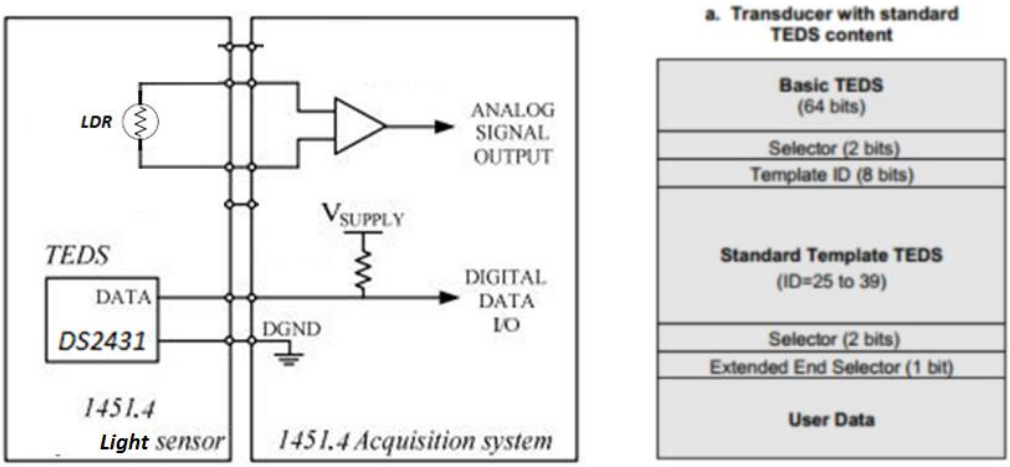


Figura 3.3.5.5.1 Esquema LDR Estándar IEEE1451.4

4) La TED (Transducer Electronic Data Sheets) es la información del sensor. Para sensores RTDs está tabulada en el ANEXO A , plantilla (ID=32) del ESTÁNDAR IEEE1451.4

5) La memoria EEPROM DS2431 será la encargada de almacenar la información digital de la TED del sensor LDR

6) Se procede a cubrir los campos con la información del fabricante de la BASIC TED+SELECTOR+STANDAR TEMPLATE

7) Finalmente se codifica esta información y se almacena en la EEPROM DS2431

Una vez definidos estos parámetros se procede a la implementación del estándar en un sensor LDR.

IEEE1451.4, ANEXO A, PLANTILLA ID=32 para sensores Resistivos

Table A.9—Resistance sensors template (ID = 32) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 32)	—
Measurement	Select Case—Physical Measurand				6	Select Case	—
	Cases 0–45	%MinPhysVal	Minimum physical value	CAL	32	Single	Various[a]
		%MaxPhysVal	Maximum physical value	CAL	32	Single	Various[a]
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 2, "Resistance Sensor"	—
	Select Case—Full-Scale Electrical Value Precision				2	Select Case	—
	Case 0	%MinElecVal	Minimum resistance	CAL	7	ConRes (0 to 1.3k, step 10)	Ω
		%MaxElecVal	Maximum resistance	CAL	7	ConRes (0 to 1.3k, step 10)	Ω
	Case 1	%MinElecVal	Minimum resistance	CAL	10	ConRes (0 to 1M, step 1k)	Ω
		%MaxElecVal	Maximum resistance	CAL	10	ConRes (0 to 1M, step 1k)	Ω
	Case 2	%MinElecVal	Minimum resistance	CAL	16	ConRes (0 to 65.4k, step 1)	Ω
		%MaxElecVal	Maximum resistance	CAL	16	ConRes (0 to 65.4k, step 1)	Ω
	Case 3	%MinElecVal	Minimum resistance	CAL	32	Single	Ω
		%MaxElecVal	Maximum resistance	CAL	32	Single	Ω
	—	%MapMeth	Mapping Method	ID	2	Enumeration: Linear | Inverse m/(x+b) | Inverse b+m/x | Inverse 1/(b+m/x)	—
	—	%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Power supply	—	%ExciteAmplNom	Excitation current, nominal	ID	8	ConRelRes (1E-6 to 0.12, ±2%)	A
	—	%ExciteAmplMax	Excitation current, max	ID	8	ConRelRes (1E-6 to 0.12, ±2%)	A

Table A.9—Resistance sensors template (ID = 32) summary *(continued)*

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
Total bits required for TEDS (range):					172 to 222 bits		

[a]Units for %MinPhysVal and %MaxPhysVal are determined by value of the Select Case "Physical Measurand" as summarized in Table A.22.

BASIC TED

Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
Manufacturer ID	0	14	00 0000 0000 0000
Model Number	0	15	000 0000 0000 0000
Version Letter	0	5	0 0000
Version Number	0	6	00 0000
Serial Number	0	24	0000 0000 0000 0000 0000 0000
BITS Totales		64	

SELECTOR

Property / Command	Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
Selector	The template is defined by IEEE1451.4	0	2 bits	00
BITS Totales			2 bits	

PLANTILLA Resistance sensors template (ID = 32)

Property / Command	Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
TEMPLATE	Template ID	32	8 bits	0010 0000
Select Case: Physical Measurand	Physical Measurand	Case 46	6 bits	10 1110
%MinPhysVal	Minimum physical value	0 (Lux)	32 bits	0000 0000 0000 0000 0000 0000 0000 0000
%MaxPhysVal	Maximum physical value	100 (Lux)	32 bits	0000 0000 0000 0000 0000 0000 0110 0100
%ElecSigType	Transducer Electrical Signal Type	-	-	String = “2, Resistance sensor”
Select Case: Full-Scale Electrical Value Precision	Full-Scale Electrical Value Precision	Case 1 0 to 1M	2 bits	01
%MinElecVal	Minimum resistance	10 (KΩ)	10 bits	00 0000 1010
%MaxElecVal	Maximum resistance	1000 (KΩ)	10 bits	11 1110 1000
%MapMeth	Mapping Method	“0” Linear y=mx+b	2 bits	00
%RespTime	Response Time	0,025 (Seg)	6 bits	10 0111
%ExciteAmplNom	Excitation current, nominal	3 mA = 0,003 (A)	8 bits	1100 1100
%ExciteAmplMax	Excitation current, max	40mA= 0.040 (A)	8 bits	1110 1100
%CalDate	Calibration Date	Desde 1/1/1998 hasta 1/2/2018 = 7330 dias	16 bits	0001 1100 1010 0010
%CalInitials	Calibration initials	LNC	15 bits	01100 01110 00011
%CalPeriod	Calibration period	2 años = 730 dias	12 bits	0010 1101 1010
%MeasID	Measurement location ID	Plantilla nº 1	11 bits	000 0000 0001
BITS Totales			189 bits	

CODIGO COMPLETO BINARIO

0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000
0000	1000	0010	1110	0000	0000	0000	0000
0000	0000	0000	0000	0000	0000	0000	0000
0000	0000	0110	0100	0100	0000	1010	1111
1010	0000	1001	1110	1100	1011	1011	0000
0111	0010	1000	1001	1000	1110	0001	1001
0110	1101	0000	0000	0001	0000	0000	0000

CODIGO COMPLETO HEXADECIMAL (31 bytes)

00 00 00 00 00 00 00 00 08 2E 00 00 00 00 00 00 00 64 40 AF A0 9F 33 B0 72 89 8E 19 6D 00 10 00

CÁLCULO Y OPERACIONES (SEGÚN IEEE1451):

7.4.5.3.3 ConRes

The constant resolution data type assigns a value according to this formula:

property name = <*start_value*> + [<*tolerance*> x <*teds_value*>]

%MinElecVal	Minimum resistance	CAL	10	ConRes (0 to 1M, step 1k)	Ω

%MinElecVal=0+1*X=10 KΩ

X=10KΩ= 00 0000 1010

%MaxElecVal	Maximum resistance	CAL	10	ConRes (0 to 1M, step 1k)	Ω

%MaxElecVal=0+1*X=1000 KΩ

X=1000 KΩ =11 1110 1000

7.4.5.3.4 ConRelRes

The constant relative resolution is calculated using the following equation.

property name = <*start_value*> * [1 + 2 * <*tolerance*>] ^ <*teds_value*>

%RespTime	Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s

%RespTime=$1*10^{-6}[1+2*0,15]^{X}$=0,025 S

$1*10^{-6}[1,3]^{X}$=0,025 S

X*ln(1,3)=ln(25000)

X=39= 10 0111

%ExciteAmplNom	Excitation current, nominal	ID	8	ConRelRes (1E-6 to 0.12, ±2%)	A

%ExciteAmplNom=$1*10^{-6}[1+2*0,02]^X$=0,003 A
$1*10^{-6}[1,04]^X$=0,003
X*ln(1,04)=ln(3000)
X=204= 1011 0010

%ExciteAmplMax	Excitation current, max	ID	8	ConRelRes (1E-6 to 0.12, ±2%)	A

%ExciteAmplMax=$1*10^{-6}[1+2*0,02]^X$=0,04 A
$1*10^{-6}[1,04]^X$=0,003
X*ln(1,04)=ln(40000)
X=236= 1110 1100

3.3.5.6 Montaje y resultados

El primer montaje que se realiza, es el circuito de medida del sensor sin la etapa de amplificación, para comprobar el buen funcionamiento del sensor LDR.

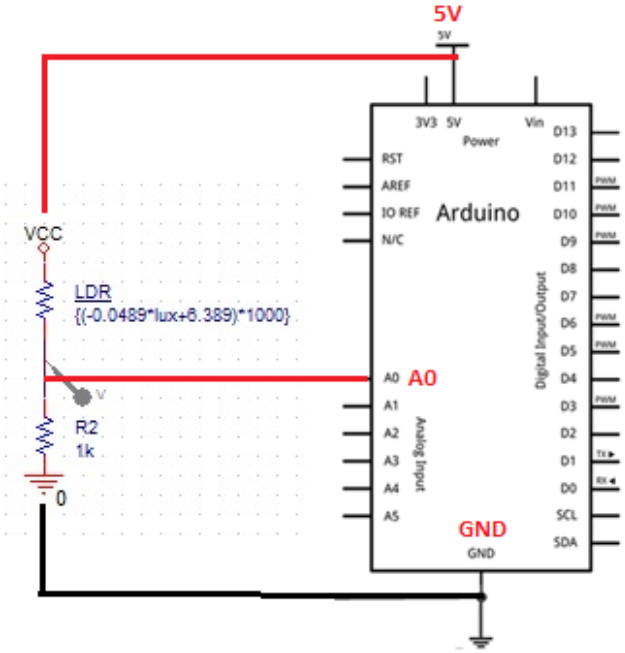


Figura 3.3.5.6.1: Esquema de conexionado del divisor de tensión al Arduino

Este circuito es alimentado por los 5V de la fuente de Arduino en Vin, generando un voltaje en Vout que se puede leer con sus entradas analógicas. Al variar la resistencia de la LDR, observamos que el voltaje de salida Vout que recoge nuestro Arduino varía de acuerdo a los cálculos y simulación realizados. De esta forma, se puede detectar la cantidad de luz que hay en el ambiente.

Comprobado este circuito, se procede a añadirle la etapa de amplificación, para la cual es necesario una fuente de alimentación de ±15V. Una vez montada la última parte del circuito, se realizan las pruebas oportunas y se vuelven a contrastar los resultados con los obtenidos en cálculos y simulación.

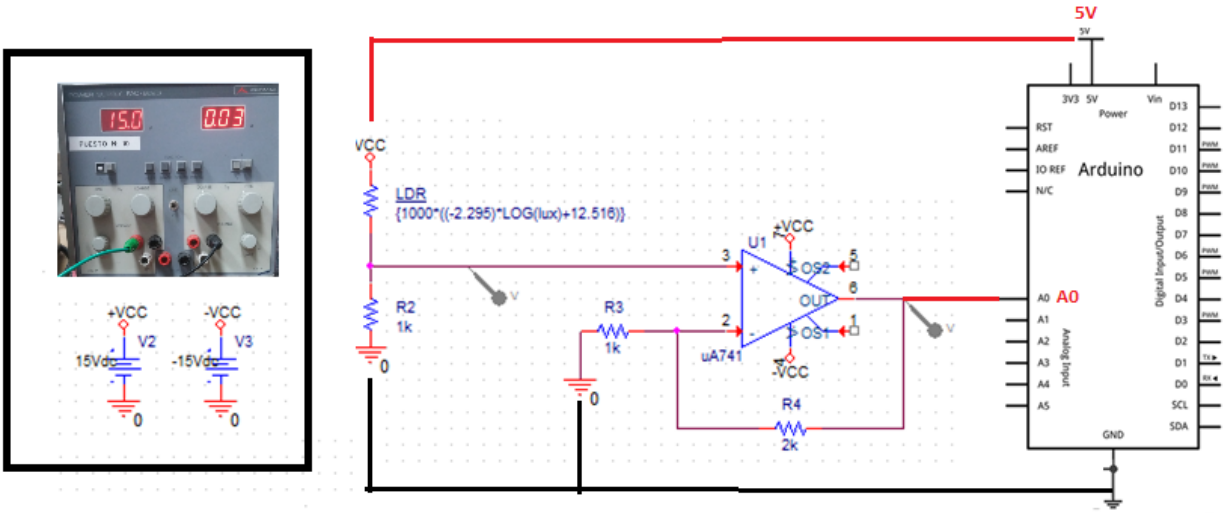


Figura 3.3.5.6.2: Esquema de conexionado al Arduino con etapa de amplificación LDR

Para contrastar estos resultados, se emplea el mismo método que se ha utilizado en la calibración. Por medio de una linterna y un luxómetro se varía el índice de luminosidad sobre la LDR y se comprueba que los resultados obtenidos se corresponden con los mostrados en el monitor serie. Con las tensiones usamos el mismo método, a luminosidad alta se comprueba que tiende a 5V y tapándola pasa a aproximarse a 0V.

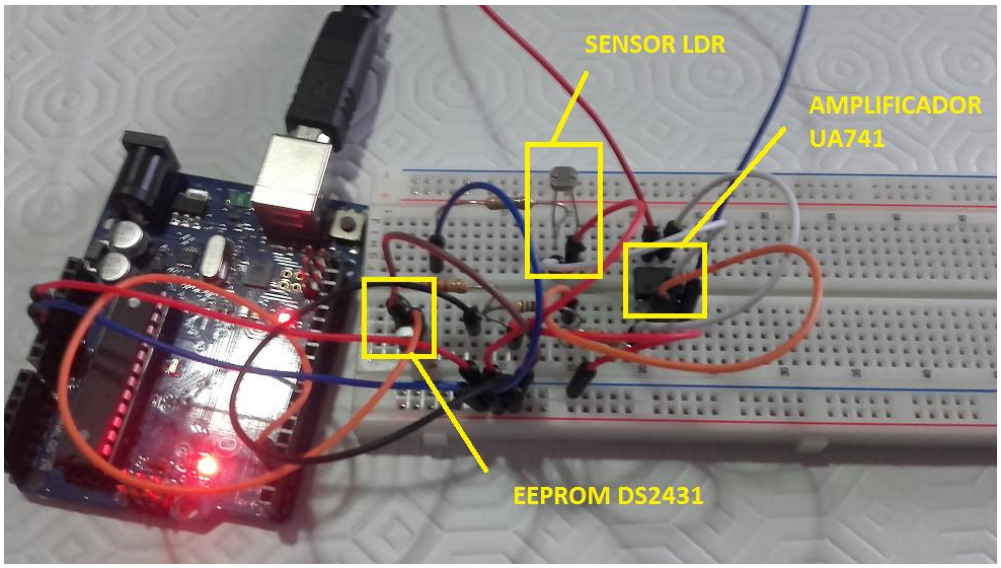
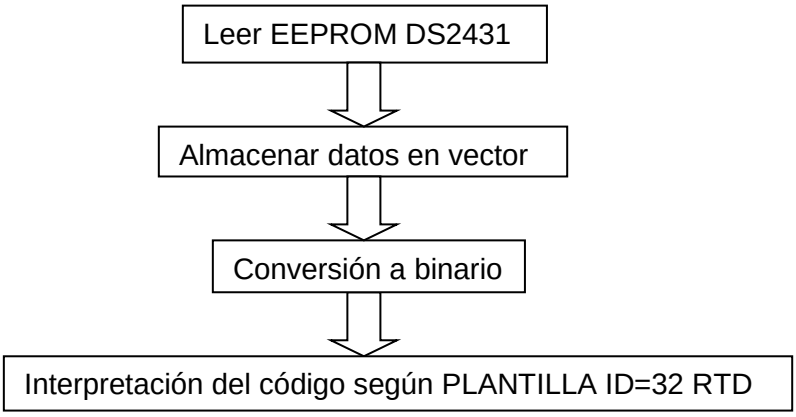


Figura 3.3.5.6.3: Prueba del montaje final LDR

Una vez realizado el montaje se procede a cargar el programa y a visualizar los resultados en el monitor serie de Arduino.

Lo primero es programar el Arduino para leer el contenido de la EEPROM y almacenarlo en una matriz para luego poder tratar los datos.



Una vez leída la información de la EEPROM se procede a la conversión a binario.

CODIGO COMPLETO HEXADECIMAL (31 bytes)

00 00 00 00 00 00 00 00 08 2E 00 00 00 00 00 00 00 64 40 AF A0 9F 33 B0 72 89 8E
19 6D 00 10 00

Durante el proceso de decodificación, se identifica el ID de la plantilla (en este caso ID=32), para posteriormente interpretar el código bit a bit siguiendo el patrón de la plantilla (Anexo A, estándar IEEE1451.4)

Figura 3.3.5.6.4: Patron de la plantilla ID=32

El primer resultado que se muestra por pantalla es el código de la ROM de la EEPROM. Este código es único e identifica al sensor dentro de la red.

Figura 3.3.5.6.5: Monitor serie Arduino. Código ROM EEPROM sensor LDR

Lo siguiente que se muestra por pantalla es los datos leídos de la EEPROM y que han sido almacenados

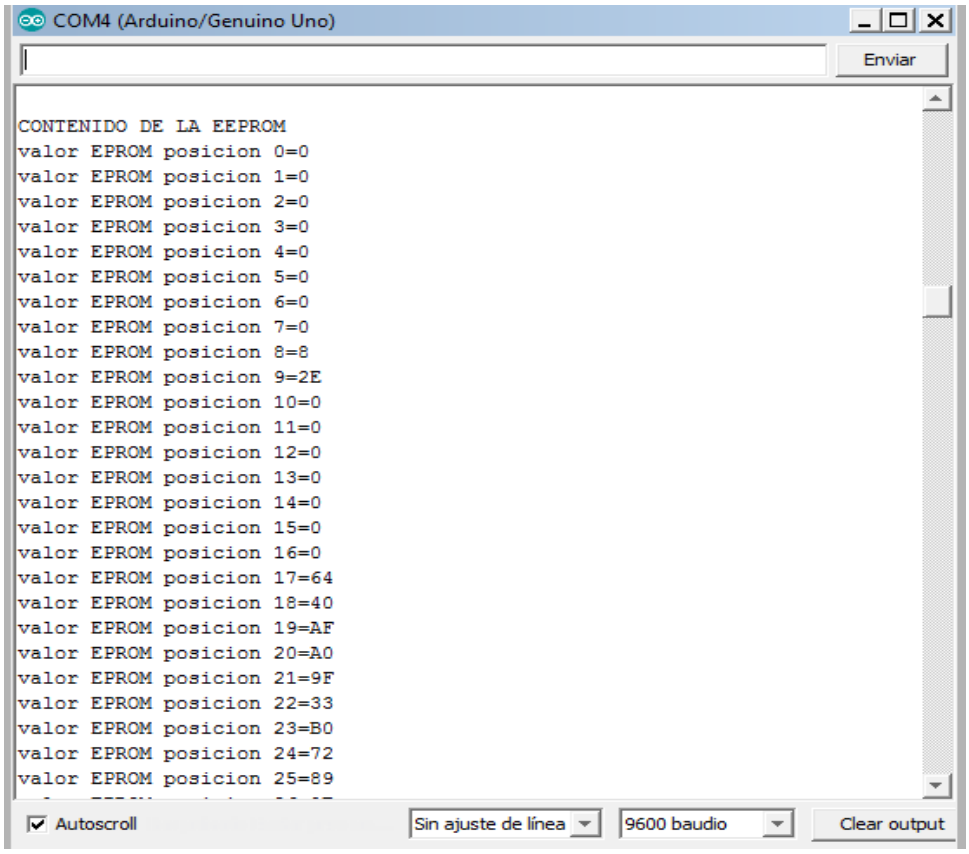


Figura 3.3.5.6.6: Monitor serie Arduino. Contenido EEPROM LDR

Una vez almacenado los datos se procede a la conversión a binario.

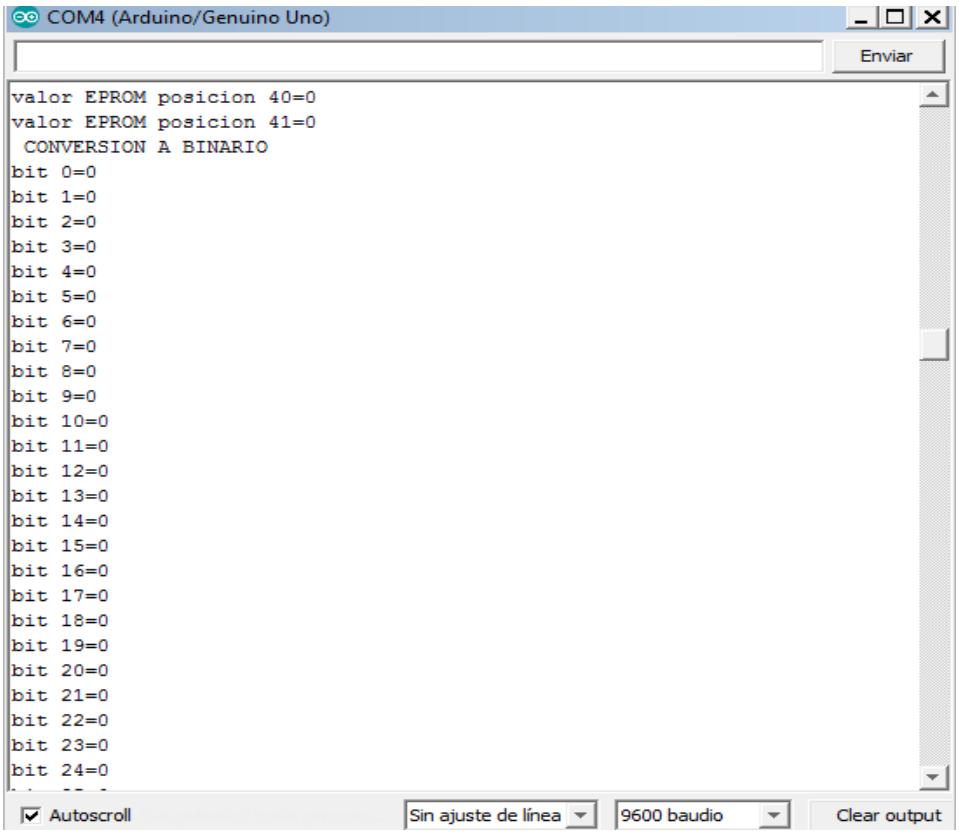


Figura 3.3.5.6.7: Monitor serie Arduino.Decodificación binaria EEPROM LDR

Y finalmente se interpreta este código para mostrar los datos de la plantilla y la lectura del sensor.

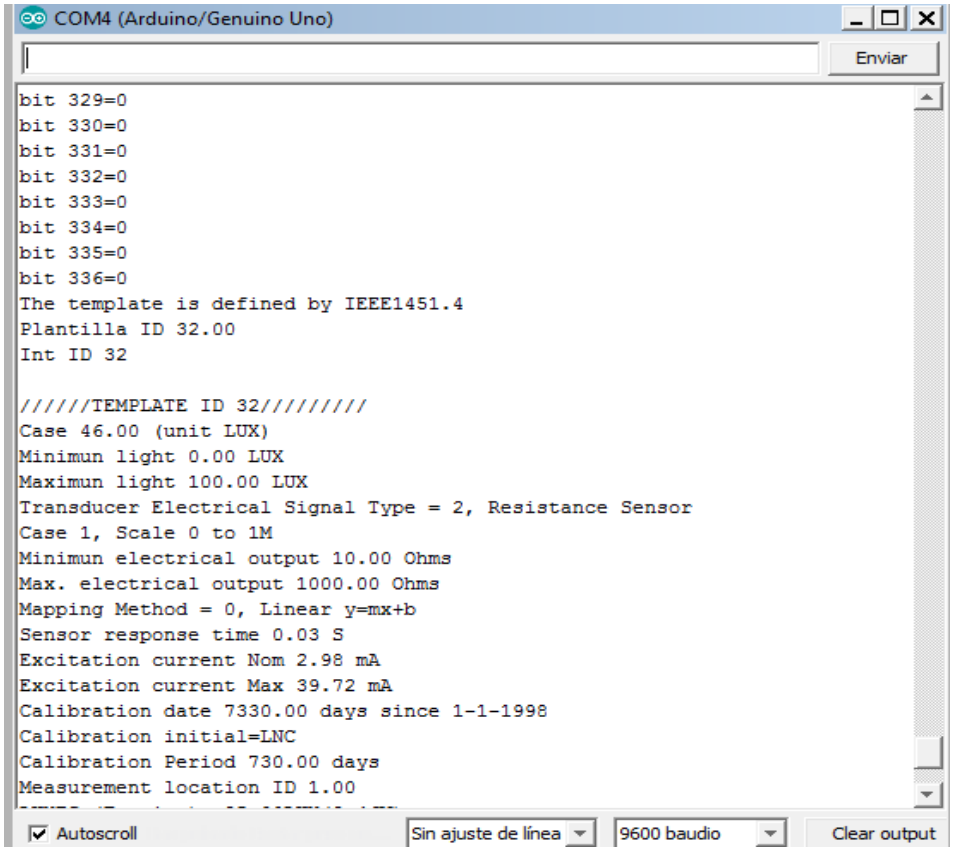


Figura 3.3.5.6.8: Monitor serie Arduino.Interpretación plantilla ID 32 LDR

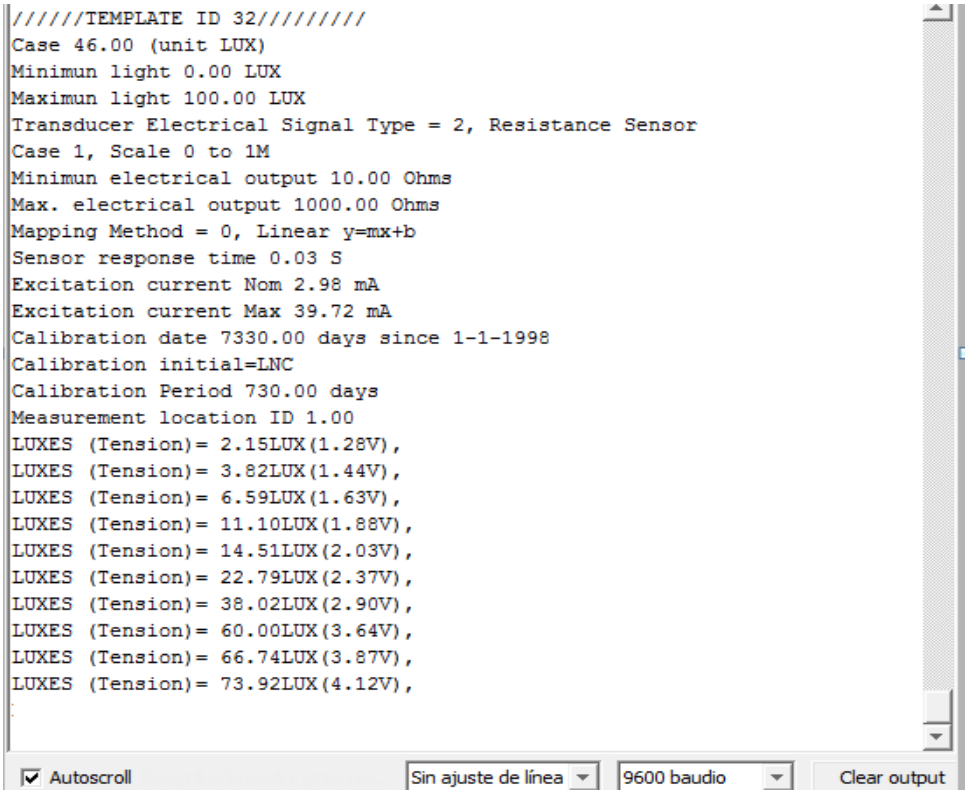


Figura 3.3.5.6.9: Monitor serie Arduino.Resultados de la lectura del sensor LDR

3.3.5.7 Programación

Para obtener una ecuación que relacione la tensión que le llega a la entrada analógica del Arduino con la intensidad lumínica (LUX) que incide en el sensor se utilizan las siguientes ecuaciones:

- La ecuación de la curva logarítmica que describe el sensor LDR:

$$R_{LDR} = -2.295 \times \ln(LUX) + 12.516 \quad \text{[Ecuación 5.1]}$$

- La salida de la primera etapa de medición:

$$V_0 = \frac{5}{R_{LDR}+1} \quad \text{[Ecuación 5.2]}$$

- La salida de la etapa de amplificación o lo que lo mismo, la entrada al Arduino:

$$V_i = V_0\left(1 + \frac{R_2}{R_1}\right) = V_0\left(1 + \frac{1{,}7284K\Omega}{1K\Omega}\right) = V_0 \times 2{,}7284 \quad \text{[Ecuación 5.3]}$$

Con estas 3 y despejando los LUX en función de la tensión de entrada al Arduino V_i se obtiene la ecuación necesaria para introducir en la programación del microcontrolador:

$$LUX = e^{\frac{13.516-\frac{15}{V_1}}{2.295}} \quad \text{[Ecuación 5.4]}$$

CÓDIGO ARDUINO: ESCRITURA/LECTURA EEPROM DS2431 DE LA INFORMACION DEL SENSOR LDR

```
#include <OneWire.h>


OneWire  ds(2);  // 1-wire on pin 2
byte  addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)
void setup(void)
{
 Serial.begin(9600);
}


void loop(void)
{
 //byte i;
 byte dat[15];


 SearchAddress(addr); //lectura y comprobación de la ROM


//Grabamos DATOS DE LA PLANTILLA en la EPROM DS2431


 dat[0] = 0x00;
```

```
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(0,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x08;
dat[1] = 0x2E;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(1,dat); //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x64;
dat[2] = 0x40;
dat[3] = 0xAF;
dat[4] = 0xA0;
dat[5] = 0x9F;
dat[6] = 0x33;
dat[7] = 0xB0;

WriteRow(2,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x72;
dat[1] = 0x89;
dat[2] = 0x8E;
```

```
dat[3] = 0x19;
dat[4] = 0x6D;
dat[5] = 0x00;
dat[6] = 0x10;
dat[7] = 0x00;

WriteRow(3,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(4,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(5,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
```

```
  dat[5] = 0x00;
  dat[6] = 0x00;
  dat[7] = 0x00;

  WriteRow(6,dat);  //(posición donde empieza a guardar los datos)

  dat[0] = 0x00;
  dat[1] = 0x00;
  dat[2] = 0x00;
  dat[3] = 0x00;
  dat[4] = 0x00;
  dat[5] = 0x00;
  dat[6] = 0x00;
  dat[7] = 0x00;

  WriteRow(15,dat);  //(posición donde empieza a guardar los datos )
  ReadAllMem();    //print all mem content

  while(1);
}

void SearchAddress(byte* address) //Search for address and confirm it
// Esta función lee el código grabado por LASER de la EEPROM que la hace única
{
  int i;
  if ( !ds.search(address))
  {
    Serial.print("No device found.\n");
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ADDR= ");
  for( i = 0; i < 8; i++)
```

```
  {
   Serial.print(address[i], HEX);
   Serial.print(" ");
  }

  if ( OneWire::crc8( address, 7) != address[7])
  {
     Serial.print("CRC is not valid, address is corrupted\n");
     return;
  }

  if ( address[0] != 0x2D)
  {
     Serial.print("Device is not a 1-wire Eeprom.\n");
     return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);        // Read Scratchpad

  for ( i = 0; i < 13; i++)
```

```
    data[i] = ds.read();
  }

  void CopyScratchPad(byte* data)
  {
   ds.reset();
   ds.select(addr);
   ds.write(0x55,1);  // Copy ScratchPad
   ds.write(data[0],1);
   ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
   ds.write(data[2],1);
   delay(25); // Waiting for copy completion
   //Serial.print("Copy done!\n");
  }

  void ReadAllMem()
  {
   int i;
   ds.reset();
   ds.select(addr);
   ds.write(0xF0,1);  // Read Memory
   ds.write(0x00,1);  //Read Offset 0000h
   ds.write(0x00,1);

   for ( i = 0; i < 144; i++) //whole mem is 144
   {
    Serial.print("byte");
    Serial.print("(");
    Serial.print(i);
    Serial.print(") ");
    Serial.print(ds.read(), HEX);
    Serial.println(" ");
   }
   Serial.println();
  }
```

```
void WriteRow(byte row, byte* buffer)
{
 int i;
 if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
  return;              //The remaining are for the 64 bits register page

 WriteReadScratchPad(row*8, 0x00, buffer);

 /*  Print result of the ReadScratchPad
 for ( i = 0; i < 13; i++)
 {
  Serial.print(buffer[i], HEX);
  Serial.print(" ");
 }
 */
 CopyScratchPad(buffer);

}
```

CÓDIGO ARDUINO: EEPROM DS2431 LECTURA Y DECODIFICACIÓN DE LA INFORMACION DEL SENSOR LDR

```
#include <OneWire.h>

OneWire  ds(2);    // 1-wire on pin 2
byte     addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)

unsigned char EPROM[145];
unsigned char BINARIO[336];

int valor;
int posicion;
int i; //i es la posición del byte de la EPROM[] en la que nos encontramos
```

```
int j; //j es la posición de bit del vector BINARIO[]
float base;
float ID;
int IDint; //variable para pasar ID (tipo float) a IDint (entero)
float pot=0; //
float Vi;
float Vo;
float LUX;

//variables para Template ID 32
float PhisMeasurand; //Minimun Temperature
float minphisval; //Minimun Temperature
float maxphisval; //Maximun Temperature
float minelecval; //Minimun electrical output
float maxelecval; //Max. electrical output
double RespTime; // Sensor response time
double ExciteAmplNom; //Excitation current, nom
double ExciteAmplMax; //Excitation current, max
float CalDate; // Calibration date
char Calinitials; //Calibration initials
float CalPeriod; //Calibration Period
float MeasID; //Measurement location ID


void setup()
{
Serial.begin(9600);

 Serial.println("ROM identificacion del sensor");
 SearchAddress(addr); //lectura y comprobación de la ROM
 Serial.println(" ");

ReadAllMem();   //print all mem content

//visualizo todo el vector EPROM[] 42 bytes
```

```
      Serial.println("CONTENIDO DE LA EEPROM ");
      for ( i = 0; i < 42; i++){ // "i" va a ser un valor que enviaremos al SLAVE como valor de posición de la EPROM

      Serial.print("valor EPROM posicion ");
      Serial.print(i);
      Serial.print("=");
      Serial.println(EPROM[i],HEX); //mostramos el valor almacenado
      delay(100);
      }


     //convierto los datos del vector EPROM[] a binario y almaceno los datos en el vector BINARIO[]

      for ( i = 0; i < 42; i++){

     /* Serial.println(" ");
       Serial.print("///////////////byte ");
       Serial.println(i);
     */
       //i es la posición del byte que nos encontramos de la EPROM

       valor=EPROM[i];

       //usamos una variable "j" para posicionarnos de 7 al 0 (8bits de derecha a izquierda))
       for ( j = 7; j >= 0; j--){

        posicion=i*8+j;

     /*   Serial.print("bit ");
        Serial.println(posicion); //mostramos valores de J
     */

        //detectamos si es par o impar y hacemos las divisiones
```

```
      if (valor & 0x01){ //comparacion binaria bit a bit AND (si da 1 impar si da 0 es par)

      //Serial.println("es impar");
      valor=valor-1;
      valor=valor/2;
      BINARIO[posicion]=1;// como es impar el resto vale 1
      }
      else {
      //Serial.println("es par");
      valor=valor/2;
      BINARIO[posicion]=0; //como es par el resto vale 0
      }

/*    Serial.println(BINARIO[posicion]); //mostramos valores de J
     Serial.println(" ");
     delay(100);
*/
   }
  }

  Serial.print(" ");
  Serial.println("CONVERSION A BINARIO ");

  for ( i = 0; i < 337; i++){
  base=BINARIO[i];
  Serial.print("bit ");
  Serial.print(i);
  Serial.print("=");
  Serial.println(BINARIO[i]);
  }


  //SELECTOR (2 bits 64 y 65)
  if (BINARIO[64]==0 & BINARIO[65]==0) {
      Serial.println("The template is defined by IEEE1451.4 ");
```

```
      }


    //decodificacion e identifiación de la ID de la plantilla, 8 bits del 66 al 73 del vector BINARIO[]

      for ( i = 66; i < 74; i++){
      base=BINARIO[i];
      ID=ID+base*pow(2,73-i);
       }

      Serial.print("Plantilla ID ");
      Serial.println(ID);
       IDint= int(ID);
       IDint= IDint+1;
      Serial.print("Int ID ");
      Serial.println(IDint);


    }//fin de void setup

    void loop()
    {



      //PLANTILLA ID 32
      if (IDint == 32){

      Serial.println(" ");
      Serial.println("//////TEMPLATE ID 32/////////");

        Physical_Measurand(); //Llamo a la funcion %Physical Measurand
    //    delay(1000);
        minimun_light(); //Llamo a la funcion ("%Min Phys Val")
    //    delay(1000);
```

```
    maximun_light();//Llamo a la funcion ("%Max Phys Val")
//    delay(1000);
    Serial.println("Transducer Electrical Signal Type = 2, Resistance Sensor");
//    delay(1000);

Full_Scale(); //Llamo a la funcion Full-Scale Electrical Value

  minimun_electrical_output(); //Llamo a la funcion  ("%Min Elec Val")
//    delay(1000);
    Max_electrical_output();//Llamo a la funcion  ("%Max Elec Val")
//    delay(1000);
    Serial.println("Mapping Method = 0, Linear y=mx+b");
//    delay(1000);

    Sensor_response_time();
//    delay(1000);
    Excitation_current_nom();
//    delay(1000);
    Excitation_current_max();
//    delay(1000);
    Calibration_date();
//    delay(1000);
    Calibration_initials();
 //    delay(1000);
    Calibration_period();
//    delay(1000);
    Measurement_location_ID();
//    delay(1000);


  IDint = 0;


  //LECTURA DEL SENSOR
```

```
 for ( i = 0; i < 10; i++){ // mido 10 valores

  Vi=analogRead(0);
  Vo=(5*Vi)/1023;

  LUX=exp((13.516-(15/Vo))/2.295);
  Serial.print("LUXES (Tension)= ");
  Serial.print(LUX);
  Serial.print("LUX");
  Serial.print("(");
  Serial.print(Vo);
  Serial.print("V");
  Serial.print(")");
  Serial.println(", ");
  delay(1000);
 }

 }// fin IF ID32

} //Fin de void loop()



///FUNCIONES PLANTILLA 32//////////////////////////

void Physical_Measurand(){
    //%Physical_Measurand
        for ( i = 74; i < 80; i++){  //bits del 74 al 79
    base=BINARIO[i];
    PhisMeasurand=PhisMeasurand+base*pow(2,79-i);
    delay(100);
    }
    Serial.print("Case ");
    Serial.print(PhisMeasurand);
    Serial.println(" (unit LUX)");
```

```
}

void minimun_light(){
      //Minimun light("%Min Phys Val")
      for ( i = 80; i < 112; i++){  //bits del 80 al 111
      base=BINARIO[i];
      minphisval=minphisval+base*pow(2,111-i);
      delay(100);
      }
      minphisval=minphisval+0;
      Serial.print("Minimun light ");
      Serial.print(minphisval);
      Serial.println(" LUX");
}

void maximun_light(){
      //Maximun Light ("%Max Phys Val")
      for ( i = 112; i < 144; i++){  //bits del 112 al 143
      base=BINARIO[i];
      maxphisval=maxphisval+base*pow(2,143-i);
      delay(100);
      }
      maxphisval=maxphisval+0;
      Serial.print("Maximun light ");
      Serial.print(maxphisval);
      Serial.println(" LUX");
}

void Full_Scale(){

if (BINARIO[144]==0 & BINARIO[145]==1) { //bits del 144 al 145
      Serial.println("Case 1, Scale 0 to 1M");
      }
}
```

```
void minimun_electrical_output(){
      //Minimun electrical output ("%Min Elec Val")
      for ( i = 146; i < 156; i++){  //bits del 146 al 155
      base=BINARIO[i];
      minelecval=minelecval+base*pow(2,155-i);
      delay(100);
      }
      Serial.print("Minimun electrical output ");
      Serial.print(minelecval);
      Serial.println(" Ohms");
}


void Max_electrical_output(){
      //Max. electrical output
      for ( i = 156; i < 166; i++){  //bits del 156 al 165
      base=BINARIO[i];
      maxelecval=maxelecval+base*pow(2,165-i);
      delay(100);
      }
      Serial.print("Max. electrical output ");
      Serial.print(maxelecval);
      Serial.println(" Ohms");
}
void Sensor_response_time(){
     //Senor response time
     for ( i = 168; i < 174; i++){  //bits del 168 al 173
     base=BINARIO[i];
     RespTime=RespTime+base*pow(2,173-i);
     delay(100);
     }
     RespTime= 0.000001*pow(1.3,RespTime);
     Serial.print("Sensor response time ");
     Serial.print(RespTime);
```

```
        Serial.println(" S");

    }

    void Excitation_current_nom(){
        //Excitation current, nom.
        for ( i = 174; i < 182; i++){  //bits del 174 al 181
        base=BINARIO[i];
        ExciteAmplNom=ExciteAmplNom+base*pow(2,181-i);
        delay(100);
        }
        ExciteAmplNom= 0.001*pow(1.04,ExciteAmplNom);
        Serial.print("Excitation current Nom ");
        Serial.print(ExciteAmplNom);
        Serial.println(" mA");
    }

    void Excitation_current_max(){
        //Excitation current, max
            for ( i = 182; i < 190; i++){  //bits del 182 al 189
        base=BINARIO[i];
        ExciteAmplMax=ExciteAmplMax+base*pow(2,189-i);
        delay(100);
        }
        ExciteAmplMax=ExciteAmplMax+34;
        ExciteAmplMax= 0.001*pow(1.04,ExciteAmplMax);
        Serial.print("Excitation current Max ");
        Serial.print(ExciteAmplMax);
        Serial.println(" mA");
    }


    void Calibration_date(){
        //Calibration date
        for ( i = 190; i < 205; i++){  //bits del 190 al 205
```

```
        base=BINARIO[i];
        CalDate=CalDate+base*pow(2,205-i);
        delay(100);
        }
        Serial.print("Calibration date ");
        Serial.print(CalDate);
        Serial.println(" days since 1-1-1998");
    }

        void Calibration_initials(){
        //Calibration initial, bits 206 al 220
            if (BINARIO[206]==0 & BINARIO[207]==1 & BINARIO[208]==1 & BINARIO[209]==0 &
BINARIO[210]==0 & BINARIO[211]==0 & BINARIO[212]==1 & BINARIO[213]==1 &
BINARIO[214]==1 & BINARIO[215]==0 & BINARIO[216]==0 & BINARIO[217]==0 &
BINARIO[218]==0 & BINARIO[219]==1 & BINARIO[220]==1 ) {
        Serial.println("Calibration initial=LNC");
         }

    }

    void Calibration_period(){
        //Calibration period
        for ( i = 221; i < 233; i++){  //bits del 221 al 232
        base=BINARIO[i];
        CalPeriod=CalPeriod+base*pow(2,232-i);
        delay(100);
        }
        Serial.print("Calibration Period ");
        Serial.print(CalPeriod);
        Serial.println(" days");
    }

    void Measurement_location_ID(){
        //Measurement location ID
        for ( i = 233; i < 244; i++){  //bits del 233 al 243
        base=BINARIO[i];
```

```
      MeasID=MeasID+base*pow(2,243-i);
      delay(100);
      }
      Serial.print("Measurement location ID ");
      Serial.println(MeasID);
}

//FUNCIONES DE LA EEPROM ONE WIRE

void SearchAddress(byte* address) //Search for address and confirm it
// Esta funcion lee el codigo gravado por LASER de la EEPROM que la hace unica
{
 int i;
 if ( !ds.search(address))
 {
  Serial.print("No device found.\n");
  ds.reset_search();
  delay(250);
  return;
 }

 Serial.print("ADDR= ");
 for( i = 0; i < 8; i++)
 {
  Serial.print(address[i], HEX);
  Serial.print(" ");
 }
 if ( OneWire::crc8( address, 7) != address[7])
 {
    Serial.print("CRC is not valid, address is corrupted\n");
    return;
 }

 if ( address[0] != 0x2D)
 {
```

```
    Serial.print("Device is not a 1-wire Eeprom.\n");
    return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);       // Read Scratchpad

  for ( i = 0; i < 13; i++)
    data[i] = ds.read();
}

void CopyScratchPad(byte* data)
{
  ds.reset();
  ds.select(addr);
  ds.write(0x55,1);  // Copy ScratchPad
  ds.write(data[0],1);
  ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
  ds.write(data[2],1);
  delay(25); // Waiting for copy completion
  //Serial.print("Copy done!\n");
```

```
}

void ReadAllMem()
{
 int i;
 ds.reset();
 ds.select(addr);
 ds.write(0xF0,1);  // Read Memory
 ds.write(0x00,1);  //Read Offset 0000h
 ds.write(0x00,1);

 for ( i = 0; i < 144; i++) //whole mem is 144
 {
//   Serial.print("byte");
//   Serial.print("(");
//   Serial.print(i);
//   Serial.print(") ");
//   Serial.print(ds.read(), HEX);
   EPROM[i]=ds.read();
   Serial.println(" ");
 }
 Serial.println();
}

void WriteRow(byte row, byte* buffer)
{
 int i;
 if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
   return;              //The remaining are for the 64 bits register page

 WriteReadScratchPad(row*8, 0x00, buffer);

 /*  Print result of the ReadScratchPad
 for ( i = 0; i < 13; i++)
 {
```

```
    Serial.print(buffer[i], HEX);
    Serial.print(" ");
  }
  */
  CopyScratchPad(buffer);
  }
```

3.3.6 Sensor Termistor (NTC)

3.3.6.1 Introducción

El término “termistor” proviene de *Thermally Sensitive Resistor*.

El termistor un termómetro de resistencia, o una resistencia cuyo valor depende de la temperatura. Está hecho de óxidos metálicos, presionado en un cordón, disco o forma cilíndrica y luego encapsulado con un material impermeable tal como epoxi o vidrio.

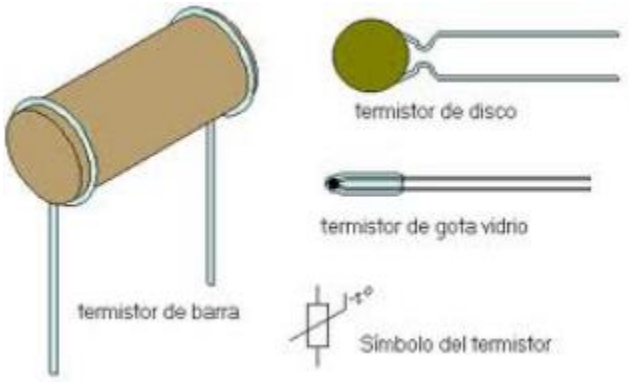


Figura 3.3.6.1.1 Diferentes tipos de termistores

Existen dos tipos de termistor:

- NTC (*Negative Temperature Coefficient*) – coeficiente de temperatura negativo.
- PTC (*Positive Temperature Coefficient*) – coeficiente de temperatura positivo (también llamado posistor).

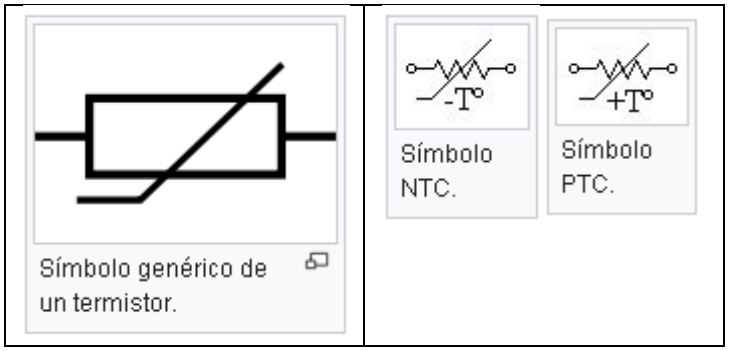
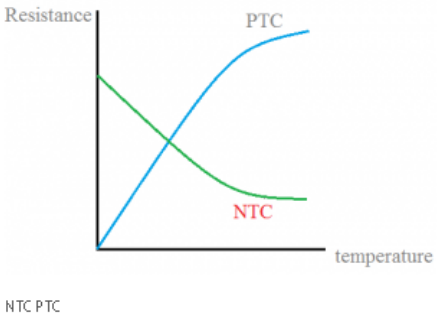


Figura 3.3.6.1.2 Simbolos de los Termistores

Se fabrican con unos materiales semiconductores con coeficiente de temperatura negativo (cuando aumenta la temperatura la resistencia disminuye) y otros con coeficiente de temperatura positivo (cuando aumenta la temperatura la resistencia aumenta).

Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura, creando una variación en la concentración de portadores. Para los termistores NTC, al aumentar la temperatura, aumentará también la concentración de portadores, por lo que la resistencia será menor, de ahí que el coeficiente sea negativo. Para los termistores PTC, en el caso de un semiconductor con un dopado muy intenso, éste adquirirá propiedades metálicas, tomando un coeficiente positivo en un margen de temperatura limitado. Usualmente, los termistores se fabrican a partir de óxidos semiconductores, tales como el óxido férrico, el óxido de níquel, o el óxido decobalto.



NTC PTC

Figura 3.3.6.1.3 Curva NTC y PTC

Los termistores tienen como característica destacable su no linealidad. En los proyectos donde se busca esta propiedad es necesario estudiar su rango de temperatura y escoger aquel en el cual su comportamiento es más lineal.

Para un termistor NTC, la característica es hiperbólica. Para pequeños incrementos de temperatura, se darán grandes incrementos de resistencia. Por ejemplo, el siguiente modelo caracteriza la relación entre la temperatura y la resistencia mediante dos parámetros:

$$R_T = A \cdot e^{\frac{B}{T}} \qquad \text{[Ecuación 3.3.6.1.1]}$$

$$A = R_0 \cdot e^{\frac{-B}{T_0}} \qquad \text{[Ecuación 3.3.6.1.2]}$$

Donde:

- R_T es la resistencia del termistor NTC a la temperatura T (K)
- R_0 es la resistencia del termistor NTC a la temperatura de referencia T_0 (K)
- B es la temperatura característica del material, entre 2000 K y 5000 K, es diferente dependiendo del material que se utilice.

Por analogía a los sensores RTD, podría definirse un coeficiente de temperatura equivalente , que para el modelo de dos parámetros quedaría:

$$\alpha = \frac{1}{R_T} \cdot \frac{dR_T}{dT} = -\frac{B}{T^2} \qquad \text{[Ecuación 3.3.6.1.3]}$$

Los termistores son fáciles de usar, de bajo costo, robusto, y responden de manera predecible a los cambios en la temperatura. Algunos de los usos más comunes de los termistores son en los termómetros digitales, en los coches para medir la temperatura del aceite y refrigerante, y en aparatos electrodomésticos tales como hornos y refrigeradores, pero también se encuentran en casi cualquier aplicación que requiera control de calefacción o refrigeración circuitos impresos de electrónica.

Ventajas:

La ventaja más importante es su pequeña medida, lo que permite velocidades de respuesta muy altas.

Desventajas:

Su principal desventaja es su falta de estabilidad en el tiempo y su gran dispersión en comparación con las termorresistencia, que pueden fabricarse con valores de resistencia superiores, mayor exactitud y valores normalizados universalmente que garantizan su intercambio sin calibración previa.

Aplicaciones:

- Su aplicación más frecuente es como sensor de temperatura para mediaciones rápidas en sondas manuales que acompañan a los termómetros portátiles electrónicos.
- Accionamiento retardo de relés.
- Estabilización de tensiones.

3.3.6.2 Características técnicas del sensor

Las especificaciones según el fabricante del sensor NTC utilizado en este trabajo son las siguientes:

*** Specification**

Model	R25	B value	Dissipation	Time Constant	Temperature Range
MF52	100Ω-10KΩ	3100K			
MF52	200Ω-10KΩ	3270K			
MF52	500Ω-15KΩ	3470K			
MF52	1KΩ-50KΩ	3600K	≥2.5mW/℃ in static air	≤7S in static air	-40℃~+120℃
MF52	5KΩ-50KΩ	3950K			
MF52	10KΩ-100KΩ	4050K			
MF52	10KΩ-100KΩ	4150K			
MF52	20KΩ-500KΩ	4300K			

Normal specification Resistance & Temperature Table of MF52-type (Unit : KΩ)

R25 / B / T(℃) Rt	10 KΩ	50 KΩ	100 KΩ	50 KΩ	50 KΩ	100 KΩ	100 KΩ	150 KΩ
	3950	3950	4000	4050	4150	4150	4300	4500
-30	181.70	908.30	1790.00					
-25	133.30	666.50	1321.00					
-20	98.88	494.50	984.70					
-15	74.10	370.50	740.80					
-10	56.06	280.30	562.30					
-5	42.80	214.00	430.50					
0	98.96	164.80	332.30	168.80	172.00	344.10	352.40	576.70
5	25.58	127.90	257.50	131.30	132.20	264.30	270.00	433.20
10	20.00	99.98	201.10	101.00	102.40	204.80	208.30	328.40
15	15.76	78.79	158.20	79.28	80.03	160.10	161.90	250.90
20	12.51	62.55	125.40	62.78	63.00	125.00	136.70	193.30
25	10.00	50.00	100.00	50.00	50.00	100.00	100.00	150.00
30	8.048	40.24	80.29	39.98	39.76	79.51	78.35	117.30
35	6.518	32.59	64.87	32.16	31.89	63.77	62.37	92.28
40	5.312	26.56	57.72	26.10	25.73	51.45	49.94	73.11
45	4.354	21.77	43.10	21.35	20.88	41.76	40.22	58.28
50	3.588	17.94	35.42	17.72	17.04	34.08	32.56	46.74
55	2.974	14.87	29.26	14.36	13.99	27.97	26.40	37.71
60	2.476	12.38	24.30	11.92	11.53	23.06	21.53	30.58
65	2.072	10.36	20.27	9.938	9.541	19.08	17.69	24.94
70	1.743	8.717	16.99	8.317	7.929	15.86	14.62	20.45
75	1.473	7.364	14.31	6.991	6.621	13.24	12.20	16.85
80	1.250	6.248	12.10	5.906	5.552	11.10	10.05	13.94
85	1.065	5.324	10.27	5.012	4.674	9.348	8.376	11.60
90	0.911	4.555	8.758	4.271	3.950	7.900	7.004	9.680
95	0.7824	3.912	7.495	3.654	3.349	6.698	5.894	8.118
100	0.6744	3.372	6.438	3.316	2.849	5.698	4.978	6.836
105	0.5836	2.918	5.550	2.701	2.438	4.875	4.215	5.780
110	0.5066	2.533	4.801	2.336	2.093	4.186	3.580	4.904

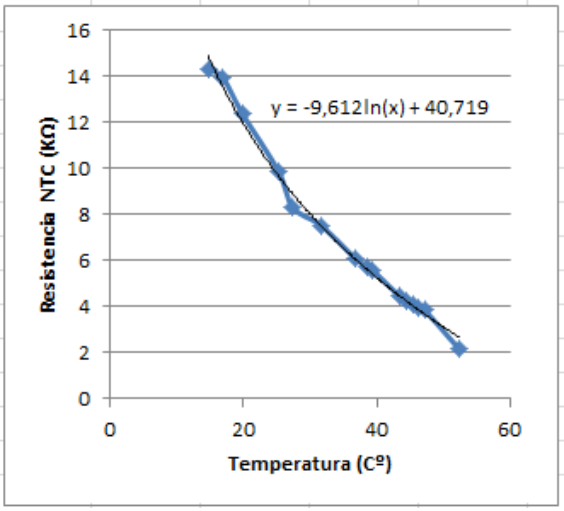
Figura 3.3.2.3: Especificaciones de la NTC

3.3.6.3 Calibración del sensor

De la misma forma que se ha calibrado el sensor LDR, el procedimiento a seguir para calibrar el sensor NTC es obtener y registrar los valores de resistencia de la cuándo es sometido a una temperatura dada. Para realizar esta tarea se utiliza un polímetro para medir el valor resistivo de la NTC sumergida en agua y un termómetro un para tomar las diferentes medidas de temperatura.

Figura 3.3.6.3.1: Calibración de la NTC mediante un termómetro

Después realizar el ensayo, he registrado y representado gráficamente los valores en una tabla EXCEL y generando la curva de calibración mediante una aproximación logarítmica.



Temperatura	KΩ
14,7	14,34
16,9	13,93
19,8	12,39
25,2	9,89
27,4	8,31
31,6	7,53
36,7	6,12
38,6	5,72
39,3	5,6
43,5	4,44
44,5	4,27
45,5	4,09
46,3	3,97
47,2	3,84
52,3	2,19

Figura 3.3.6.3.2: Aproximación de la curva logarítmica de la NTC

De esta forma se obtiene la ecuación de la curva de calibración que define el comportamiento del sensor frente a la variación de la temperatura.

3.3.6.4 Simulación OrCAD

Circuito de acondicionamiento (divisor de tensión):

Se realiza el montaje en OrCAD del divisor de tensión asignándole a la NTC una resistencia de valor igual a la ecuación de la curva logarítmica que define el comportamiento del sensor.

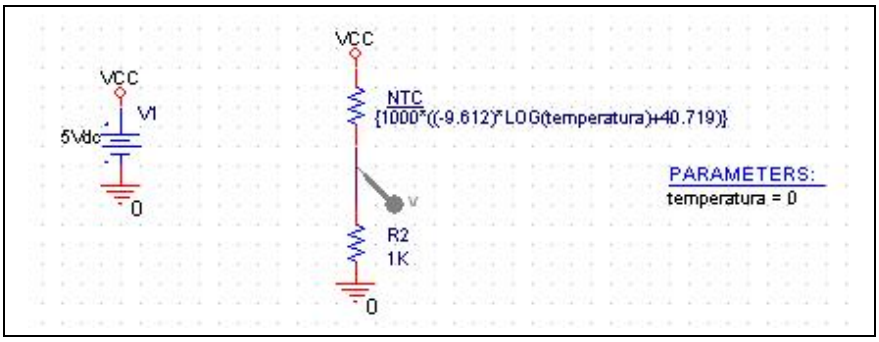


Figura 3.3.6.4.1: Divisor de tensión empleado

$$R_{NTC} = -9{,}612 \times \ln(LUX) + 40{,}719 \qquad \text{[Ecuación 3.3.6.4.1]}$$

$$V_{out} = V_{cc} \times \frac{R_2}{R_{NTC}+R_2} \qquad \text{[Ecuación 3.3.6.4.2]}$$

$$V_{out} = 5V \times \frac{1K\Omega}{R_{NTC}+1K\Omega} \qquad \text{[Ecuación 3.3.6.4.3]}$$

Una vez simulado el circuito, se observa que la gráfica de la simulación coincide con la calculada en EXCEL.

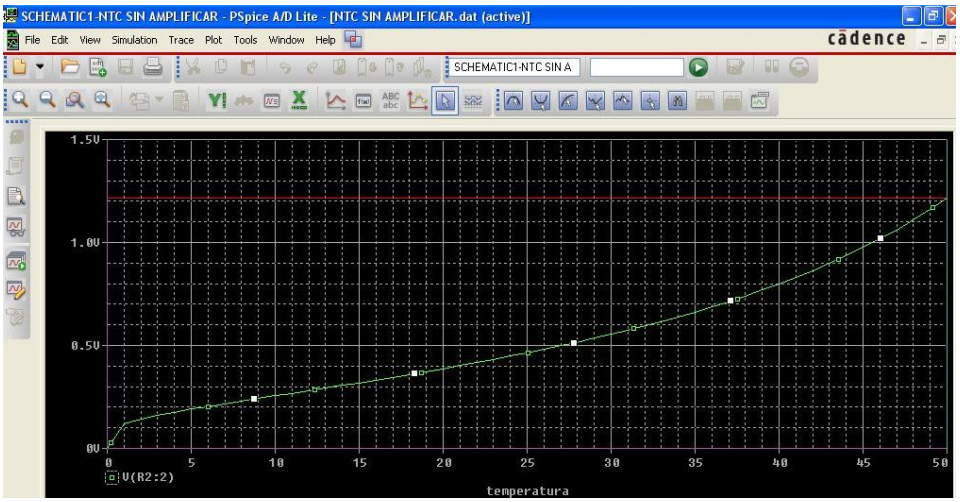
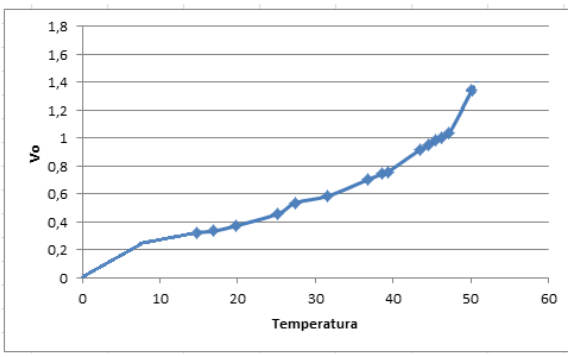


Figura 3.3.6.4.2: Simulación en OrCAD de la salida Vout



Temperatura Cº	KΩ	V0
14,7	14,34	0,325945
16,9	13,93	0,334896
19,8	12,39	0,373413
25,2	9,89	0,459137
27,4	8,31	0,537057
31,6	7,53	0,586166
36,7	6,12	0,702247
38,6	5,72	0,744048
39,3	5,6	0,757576
43,5	4,44	0,919118
44,5	4,27	0,948767
45,5	4,09	0,982318
46,3	3,97	1,006036
47,2	3,84	1,033058
50	2,19	1,214601

Figura 3.3.6.4.3: Curva de salida Vout obtenida en Excel

Etapa de amplificacion de la señal

Las entradas analógicas de Arduino sirven para medir voltajes entre 0 y 5V. Para una alimentación del sensor a 5V se obtiene a su salida una tensión de 0 a 1,21V, como hemos visto en el apartado anterior.

Por ello, es necesario utilizar un circuito adaptador de escala que amplifique la salida para obtener mejor resolución de medida.

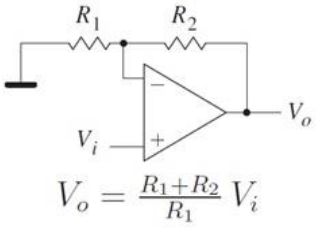


Figura 3.3.6.4.4: Amplificador no inversor común

Según los datos obtenidos en el OrCAD, mirando la tensión máxima que se muestra y la tensión de 5 voltios que se quiere obtener para este mismo valor, sacamos la ganancia necesaria del adaptador de escala. Con ello y la ecuación del circuito de la imagen anterior (asignándole a R1 el valor de 1KΩ) se calcula el valor de R2.

$$G = \frac{V_0}{V_i} = \frac{5}{1.2146} = 4{,}1165 \qquad \text{[Ecuación 3.3.6.4.4]}$$

$$G = \left(1 + \frac{R_2}{R_1}\right) = 4{,}1165 \qquad \text{[Ecuación 3.3.6.4.5]}$$

$$\text{SI } R_1 = 1K\Omega, \text{entonces } R_2 = 3{,}1165K\Omega \qquad \text{[Ecuación 3.3.6.4.6]}$$

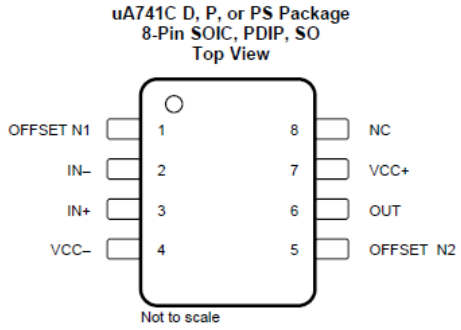


Figura 3.3.6.4.5: Esquema de un uA741

A continuación se realiza el montaje del circuito amplificador en OrCAD:

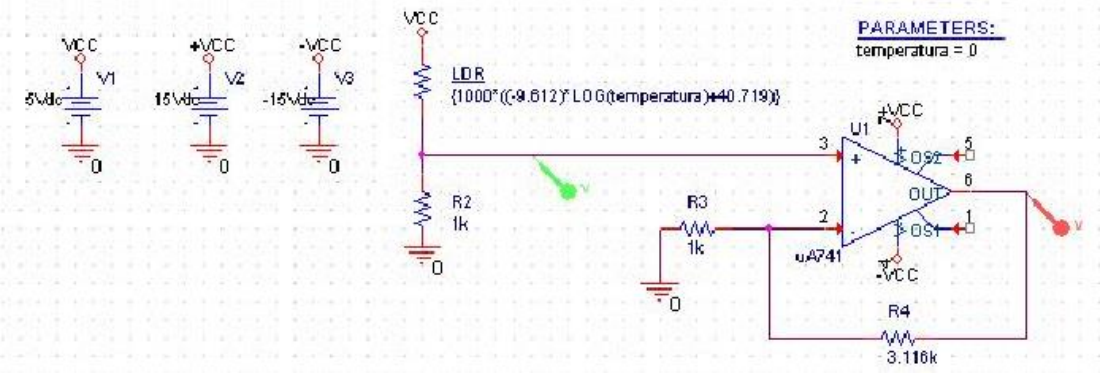


Figura 3.3.6.4.6: Esquema de conexionado etapa de amplificación NTC

Una vez simulado el circuito, se observa que la gráfica de la simulación coincide con la calculada en EXCEL.

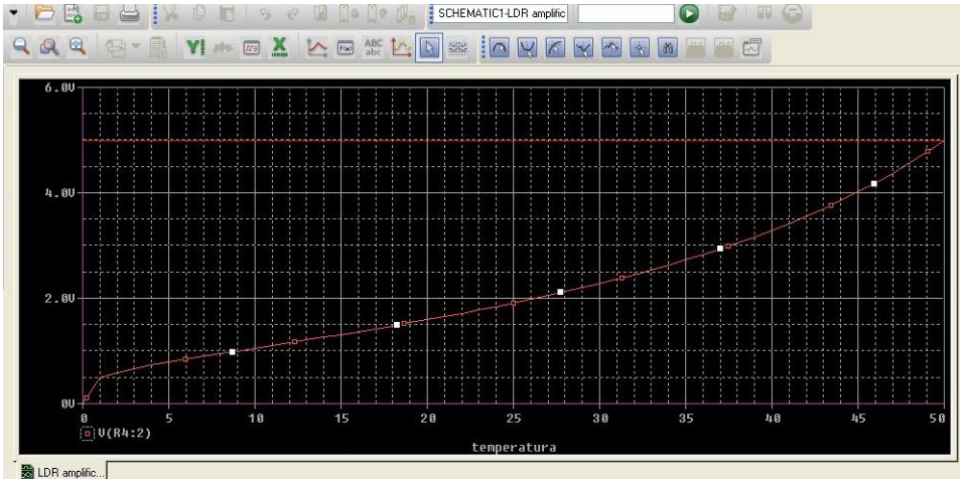
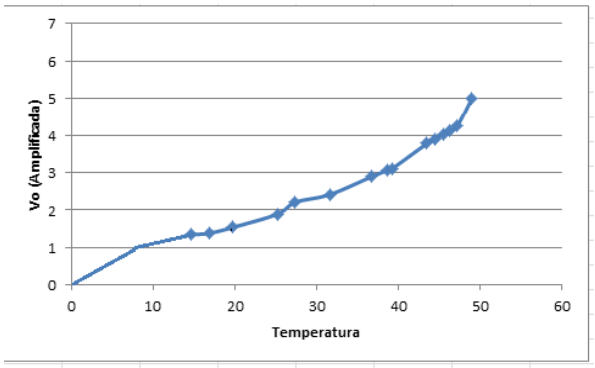


Figura 3.3.6.4.7: Simulación en OrCAD de la Salida de la etapa de amplificación / Entrada de señal al Arduino



Temperatura Cº	KΩ	V0	V0 Amplificada
14,7	14,34	0,325945	1,341753585
16,9	13,93	0,334896	1,378600134
19,8	12,39	0,373413	1,537154593
25,2	9,89	0,459137	1,890036731
27,4	8,31	0,537057	2,210794844
31,6	7,53	0,586166	2,412954279
36,7	6,12	0,702247	2,890800562
38,6	5,72	0,744048	3,062872024
39,3	5,6	0,757576	3,118560606
43,5	4,44	0,919118	3,783547794
44,5	4,27	0,948767	3,905597723
45,5	4,09	0,982318	4,043713163
46,3	3,97	1,006036	4,141348089
47,2	3,84	1,033058	4,252582645
50	2,19	1,214601	5,0324

Figura 3.3.6.4.8: Curva de salida de la etapa de amplificación obtenida en Excel

3.3.6.5 Implementación del estándar IEEE1451 en una NTC

Para la implementación del diseño de un sensor inteligente NTC de la misma forma que el sensor PT1000, siguiendo lo indicado en el estándar IEEE1451 4:

1) Se emplea la arquitectura clase 2 de múltiples hilos por su sencilla integración. Tendremos una interface de modo mixto con 2 señales separadas: una analógica (la lectura del sensor) y otra digital (los datos de la TED)
2) El sensor NTC y la memoria EEPROM DS2431 formarán el TIM
3) El Arduino Uno será el encargado de leer, decodificar e interpretar la información trabajando como NCAP.

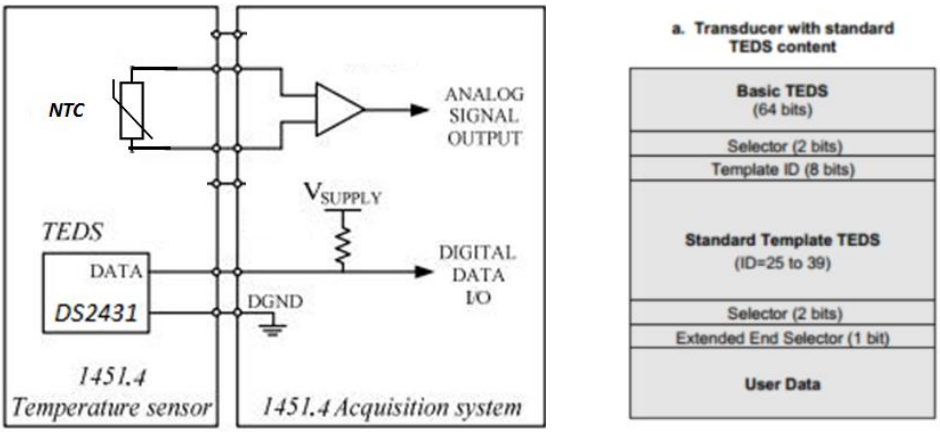


Figura 3.3.6.5.1: Esquema NTC Estándar IEEE1451.4

4) La TED (Transducer Electronic Data Sheets) es la información del sensor. Para sensores RTDs está tabulada en el ANEXO A, plantilla (ID=38) del ESTÁNDAR IEEE1451.4.
5) La memoria EEPROM DS2431 será la encargada de almacenar la información digital de la TED del sensor NTC.
6) Se procede a cubrir los campos con la información del fabricante de la BASIC TED+SELECTOR+STANDAR TEMPLATE.
7) Finalmente se codifica esta información y se almacena en la EEPROM DS2431.
8) Una vez definidos estos parámetros se procede a la implementación del estándar en un sensor NTC.

IEEE1451.4, ANEXO A, PLANTILLA ID=38 para Termistores

Table A.16—Thermistor template (ID = 38) summary

Function	Select	Property/Command	Description	Access	Bits	Data type (and range)	Units
ID	—	TEMPLATE	Template ID	—	8	Integer (value = 38)	—
Measurement	—	%MinPhysVal	Minimum temperature	CAL	11	ConRes (−200 to 1,846, step 1)	°C
	—	%MaxPhysVal	Maximum temperature	CAL	11	ConRes (−200 to 1,846, step 1)	°C
Electrical signal output	—	%ElecSigType	Transducer Electrical Signal Type	ID	—	Assign = 2, "Resistance Sensor"	—
	—	%MinElecVal	Minimum resistance output	CAL	18	ConRes (0 to 262k, step 1)	Ω
	—	%MaxElecVal	Maximum resistance output	CAL	18	ConRes (0 to 262k, step 1)	Ω
	—	%MapMeth	Mapping Method	ID	—	Assign = 6, "Thermistor"	—
	—	%RTDCoef_R0	Resistance of thermistor at 0°C	ID	20	ConRelRes (10 to 5.5E+6, ±6.3ppm)	Ω
	—	%SteinhartA	Steinhart-Hart Coefficient A	ID	32	Single	1/C
	—	%SteinhartB	Steinhart-Hart Coefficient B	ID	32	Single	1/C
	—	%SteinhartC	Steinhart-Hart Coefficient C	ID	32	Single	1/C
	—	%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s
Excitation or power supply	—	%ExciteAmplNom	Nominal current excitation	CAL	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
	—	%ExciteAmplMax	Maximum current excitation	ID	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A
	—	%SelfHeating	Self-heating constant	ID	5	ConRelRes (0.25E-3 to 16E-3, ±7.4%)	W/°C
Calibration information	—	%CalDate	Calibration Date	CAL	16	DATE	—
	—	%CalInitials	Calibration initials	CAL	15	CHR5	—
	—	%CalPeriod	Calibration period	CAL	12	UNINT	days
Misc.	—	%MeasID	Measurement location ID	USR	11	UNINT	—
			Total bits required for TEDS (range):		263 bits		

BASIC TED

Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
Manufacturer ID	0	14	00 0000 0000 0000
Model Number	0	15	000 0000 0000 0000
Version Letter	0	5	0 0000
Version Number	0	6	00 0000
Serial Number	0	24	0000 0000 0000 0000 0000 0000
BITS Totales		64	

SELECTOR

Property / Command	Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
Selector	The template is defined by IEEE1451.4	0	2 bits	00
BITS Totales			2 bits	

PLANTILLA Thermistor template (ID = 38)

Property / Command	Descripción	Valor (Unidades)	Nº de Bits	Código - Binario
TEMPLATE	Template ID	38	8 bits	0010 0110
%MinPhysVal	Minimum temperature	-40º (C)	11 bits	000 1010 0000
%MaxPhysVal	Maximum temperature	120º (C)	11 bits	001 0100 0000
%ElecSigType	Transducer Electrical Signal Type	-	-	String = "2, Resistance sensor"
%MinElecVal	Minimum resistance output	500(Ω)	18 bits	00 0000 0001 1111 0100
%MaxElecVal	Maximum resistance output	182000(Ω)	18 bits	10 1100 0110 1111 0000
%MapMeth	Mapping Method	-	-	String = "6, Thermistor"
%RTDCoef_R0	Resistance of thermistor at 0°C	100000(Ω)	20 bits	0000 0000 0100 1110 1100
%SteinhartA	Steinhart-Hart Coefficient A	996 (1/C)	32 bits	0000 0000 0000 0000 0000 0011 1110 0100
%SteinhartB	Steinhart-Hart Coefficient B	45031 (1/C)	32 bits	0000 0000 0000 0000 1010 1111 1110 0111
%SteinhartC	Steinhart-Hart Coefficient C	741840 (1/C)	32 bits	0000 0000 0000 1011 0101 0001 1101 0000
%RespTime	Sensor Response Time	7 (Seg)	6 bits	11 1100
%ExciteAmplNom	Nominal current excitation	3 mA = 0,003 (A)	8 bits	1011 0010
%ExciteAmplMax	Maximum current excitation	40mA= 0.040 (A)	8 bits	1110 1100
%SelfHeating	Self-heating constant	2,5 (mW/Cº)	5 bits	0 0110
%CalDate	Calibration Date	Desde 1/1/1998 hasta 1/2/2018 = 7330 días	16 bits	0001 1100 1010 0010
%CalInitials	Calibration initials	LNC	15 bits	01100 01110 00011
%CalPeriod	Calibration period	2 años = 730 días	12 bits	0010 1101 1010
%MeasID	Measurement location ID	Plantilla nº 2	11 bits	000 0000 0010
BITS Totales			137 bits	

CODIGO COMPLETO BINARIO

0000 0000 0000 0000 0000 0000 0000 0000

0000 0000 0000 0000 0000 0000 0000 0000

0000 1001 1000 0101 0000 0001 0100 0000

0000 0000 0111 1101 0010 1100 0110 1111

0000 0000 0000 0100 1110 1100 0000 0000

0000 0000 0000 0011 1110 0100 0000 0000

0000 0000 1010 1111 1110 0111 0000 0000

0000 1011 0101 0001 1101 0000 1111 0001

1100 01011 1011 0000 1100 0011 1001 0100

0100 1100 0111 0000 1100 1011 0110 1000

0000 0001 0000 0000

CODIGO COMPLETO HEXADECIMAL (41 bytes)

00 00 00 00 00 00 00 00 09 85 01 40 00 7D 2C 6F 00 04 EC 00 00 03 E4
00 00 AF E7 00 0B 51 D0 F2 CB B0 C3 94 4C 70 CB 68 01 00

CÁLCULO Y OPERACIONES (SEGÚN IEEE1451):

7.4.5.3.3 ConRes

The constant resolution data type assigns a value according to this formula:

property name = <*start_value*> + [<*tolerance*> x <*teds_value*>]

%MinPhysVal	Minimum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	ºC

Para -40ºC

%MinPhysVal = -200+1*X= -40 ºC

X=160 = 000 1010 0000

%MaxPhysVal	Maximum temperature	CAL	11	ConRes (–200 to 1,846, step 1)	ºC

Para 120ºC

%MaxPhysVal=-200+1*X=120 ºC

X=320 = 001 0100 0000

%MinElecVal	Minimum resistance output	CAL	18	ConRes (0 to 262k, step 1)	Ω

%MinElecVal=0+1*X=500 Ω

X=500 =00 0000 0001 1111 0100

%MaxElecVal	Maximum resistance output	CAL	18	ConRes (0 to 262k, step 1)	Ω

%MaxElecVal=0+1*X=182000 Ω

X=182000=10 1100 0110 1111 0000

7.4.5.3.4 ConRelRes

The constant relative resolution is calculated using the following equation.

property name = <*start_value*> * [1 + 2 * <*tolerance*>] ^ <*teds_value*>

%RTDCoef_R0	Resistance of thermistor at 0°C	ID	20	ConRelRes (10 to 5.5E+6, ±6.3ppm)	Ω

10 000ppm=1%

6,3ppm=0,00063%

%RTDCoef_R0=10[1+2*0,0000063]^{X}=100 000 Ω

10[1,0000126]^{X}=100 000

X*ln(1,0000126)=ln(10000)

X=1260= 0000 0000 0100 1110 1100

%RespTime	Sensor Response Time	ID	6	ConRelRes (1E-6 to 7.9, ±15%)	s

%RespTime=1*10^{-6}[1+2*0,15]^{X}=7 Seg

1*10^{-6}[1,3]^{X}=7 Seg

X*ln(1,3)=ln(7000000)

X=60= 11 1100

%ExciteAmplNom	Nominal current excitation	CAL	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A

%ExciteAmplNom=1*10^{-6}[1+2*0,023]^{X}=0,003 A

1*10^{-6}[1,046]^{X}=0,003

X*ln(1,046)=ln(3000)

X=178= 1011 0010

%ExciteAmplMax	Maximum current excitation	ID	8	ConRelRes (1E-6 to 120E-3, ±2.3%)	A

%ExciteAmplMax=1*10^{-6}[1+2*0,023]^{X}=0,04 A

1*10^{-6}[1,046]^{X}=0,003

X*ln(1,046)=ln(40000)

X=236= 1110 1100

%SelfHeating	Self-heating constant	ID	5	ConRelRes (0.25E-3 to 16E-3, ±7.4%)	W/°C

%ExciteAmplMax=0,25*10^{-3}[1+2*0,074]^{X}=0,0025 W/Cº

0,25*10^{-3} [1,148]^{X}=0,0025

X*ln(1,148)=ln(10)

X=6= 0 0110

Steinhart - Hart Equation $1/T = A+B(LnR)+C(LnR)^3$

$$1/T_1 = A + B(LnR_1) + C(LnR_1)^3$$
$$1/T_2 = A + B(LnR_2) + C(LnR_2)^3$$
$$1/T_3 = A + B(LnR_3) + C(LnR_3)^3$$

T_1=0ºC	→	R1=98960 Ω
T_2=50ºC	→	R2=3512 Ω
T_1=100ºC	→	R3=674 Ω

$$1/273K=A+B(ln98960)+C(ln98960)^3$$
$$1/323K=A+B(ln3512)+C(ln3512)^3$$
$$1/373K=A+B(ln674)+C(ln674)^3$$

A=0,001003829 → 1/C (A)= 996
B=0,000222068 → 1/C (B)= 45031
C=0,00001348 → 1/C (C)= 741840

3.3.6.6 Montaje y resultados

Se realiza el montaje del circuito siguiendo el esquema diseñado en ORCAD. Una vez montado circuito, se realizan las pruebas oportunas y se vuelve a contrastar los resultados con los obtenidos en cálculos y simulación.

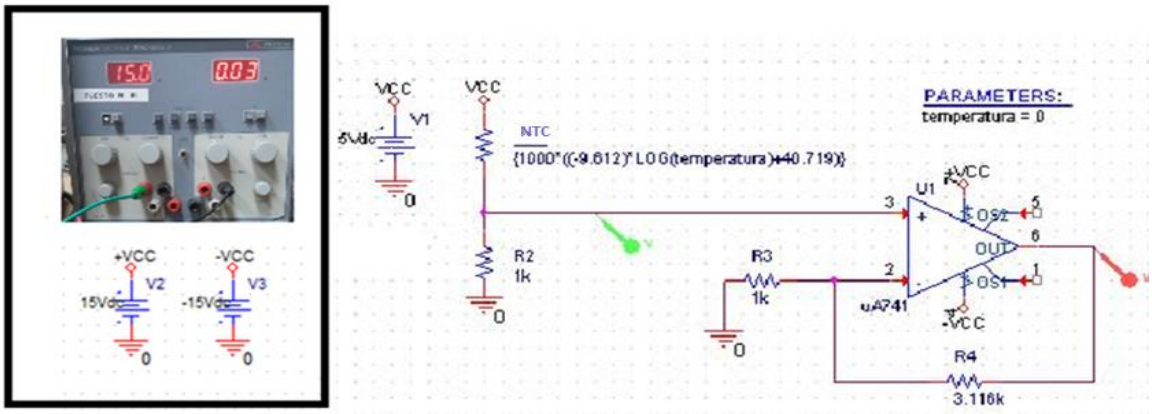


Figura 3.3.6.6.1: Esquema de conexionado etapa de amplificación NTC

Para contrastar estos resultados, se emplea el mismo método que se ha utilizado en la calibración. Por medio de una fuente de calor y un termómetro se varía la temperatura sobre la NTC y se comprueba que los resultados obtenidos en la simulación se corresponden con los mostrados en el monitor serie.

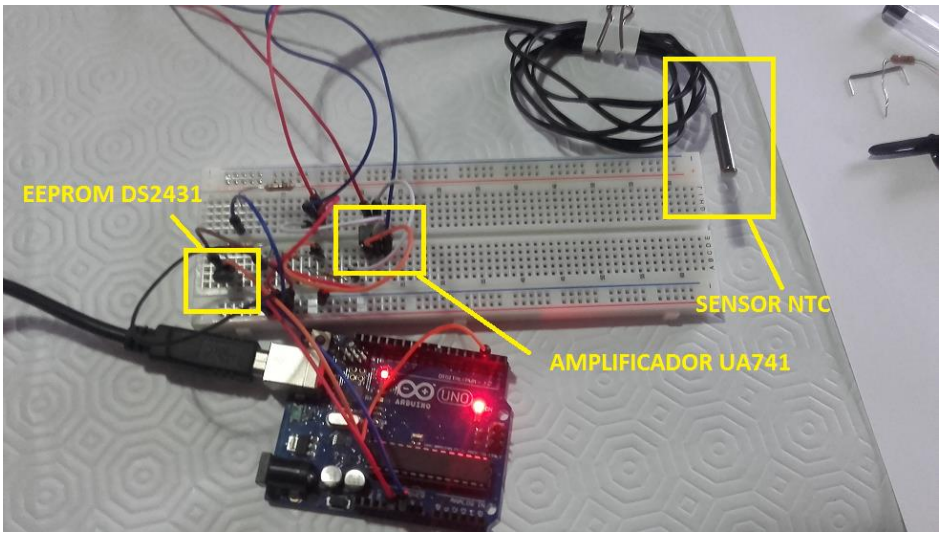
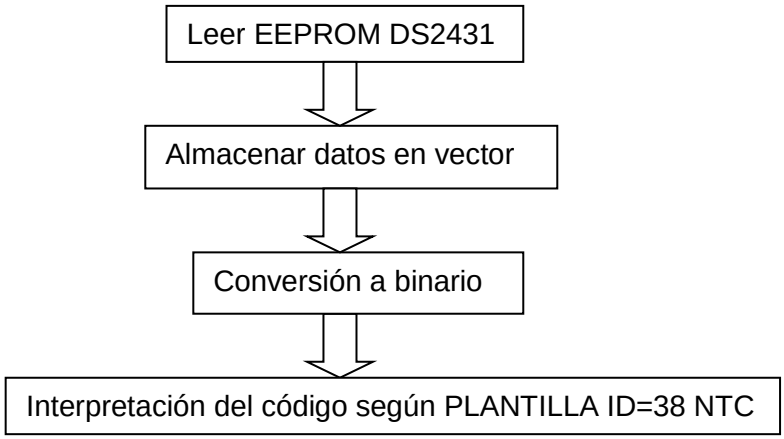


Figura 3.3.6.6.2: Prueba del montaje final NTC

Para ello una vez realizado el montaje se procede a cargar el programa y a visualizar los resultados en el monitor serie de Arduino.

Lo primero es programar el Arduino para leer el contenido de la EEPROM y almacenarlo en una matriz para luego poder tratar los datos.



Una vez leída la información de la EEPROM se procede a la conversión a binario.

CODIGO COMPLETO HEXADECIMAL (41 bytes)

00 00 00 00 00 00 00 00 09 85 01 40 00 7D 2C 6F 00 04 EC 00 00 03 E4 00 00 AF
E7 00 0B 51 D0 F2 CB B0 C3 94 4C 70 CB 68 01 00

Durante el proceso de decodificación, se identifica el ID de la plantilla (en este caso ID=38), para posteriormente interpretar el código bit a bit siguiendo el patrón de la plantilla (Anexo A, estándar IEEE1451.4)

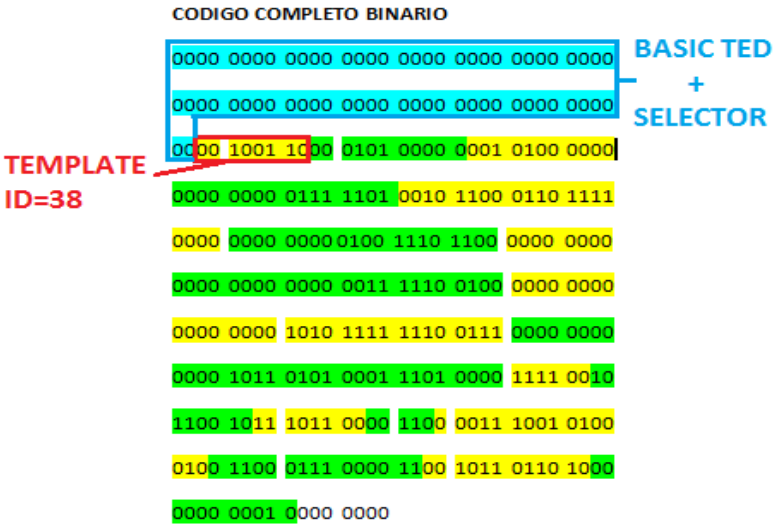


Figura 3.3.6.6.3: Patron de la plantilla ID=38

El primer resultado que se muestra por pantalla es el código de la ROM de la EEPROM. Este código es único e identifica al sensor dentro de la red.

Figura 3.3.6.6.3: Monitor serie Arduino. Código ROM EEPROM sensor NTC

Lo siguiente que se muestra por pantalla es los datos leídos de la EEPROM y que han sido almacenados:

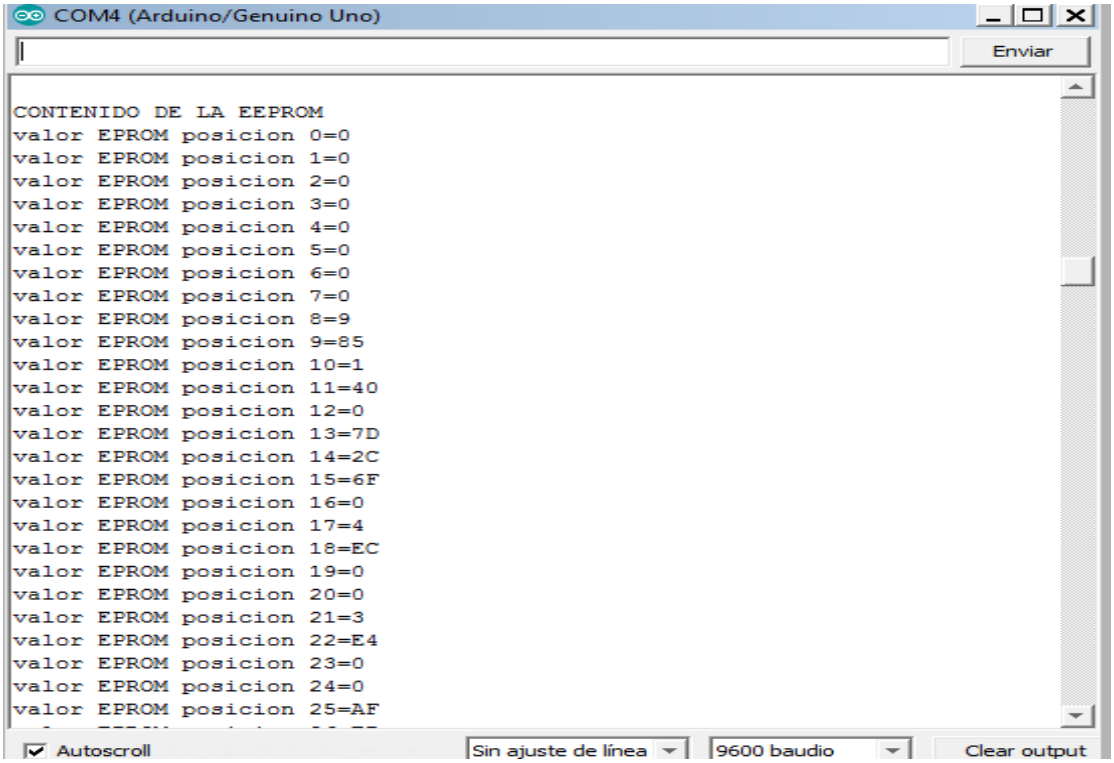


Figura 3.3.6.6.4: Monitor serie Arduino. Contenido EEPROM NTC

Una vez almacenado los datos se procede a la conversión a binario.

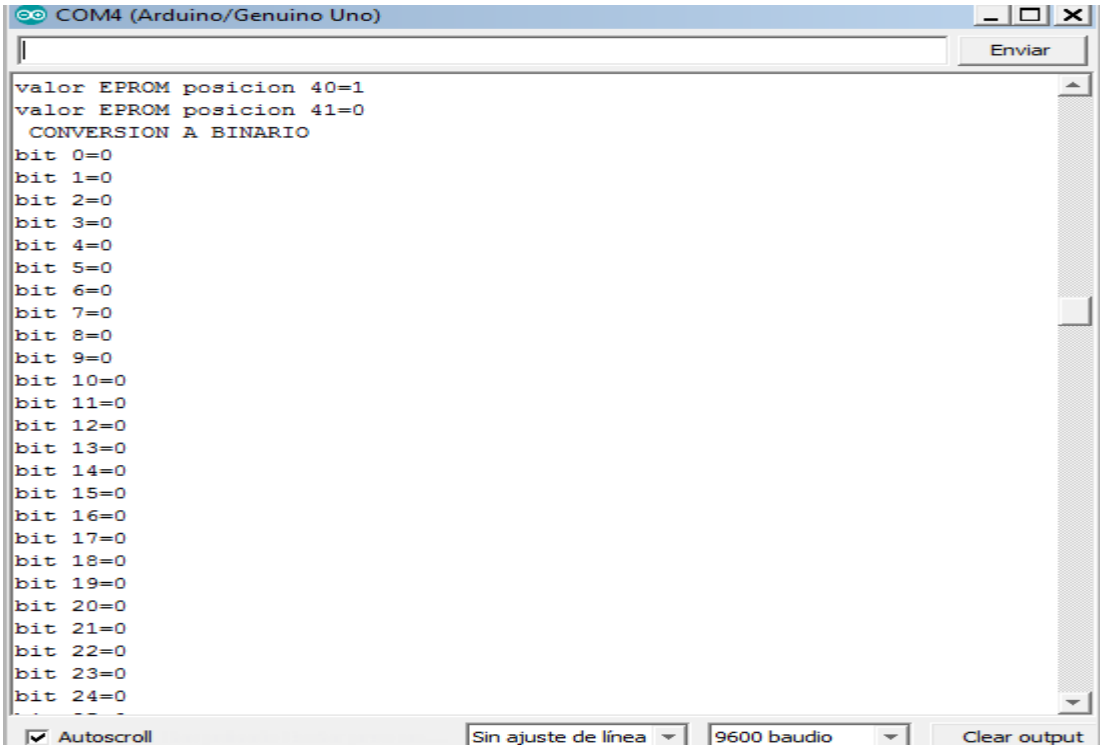


Figura 3.3.6.6.5: Monitor serie Arduino.Decodificación binaria EEPROM NTC

Y finalmente se interpreta este código para mostrar los datos de la plantilla y la lectura del sensor.

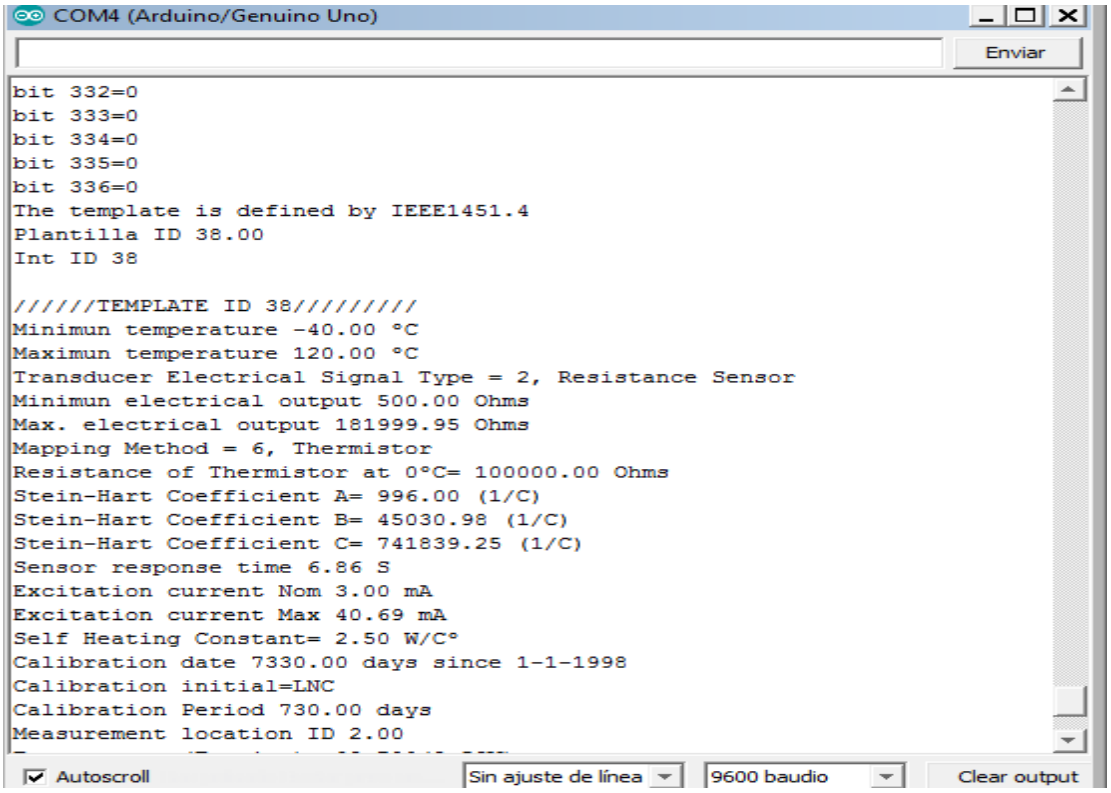


Figura 3.3.6.6.6: Monitor serie Arduino.Interpretación plantilla ID 38 NTC

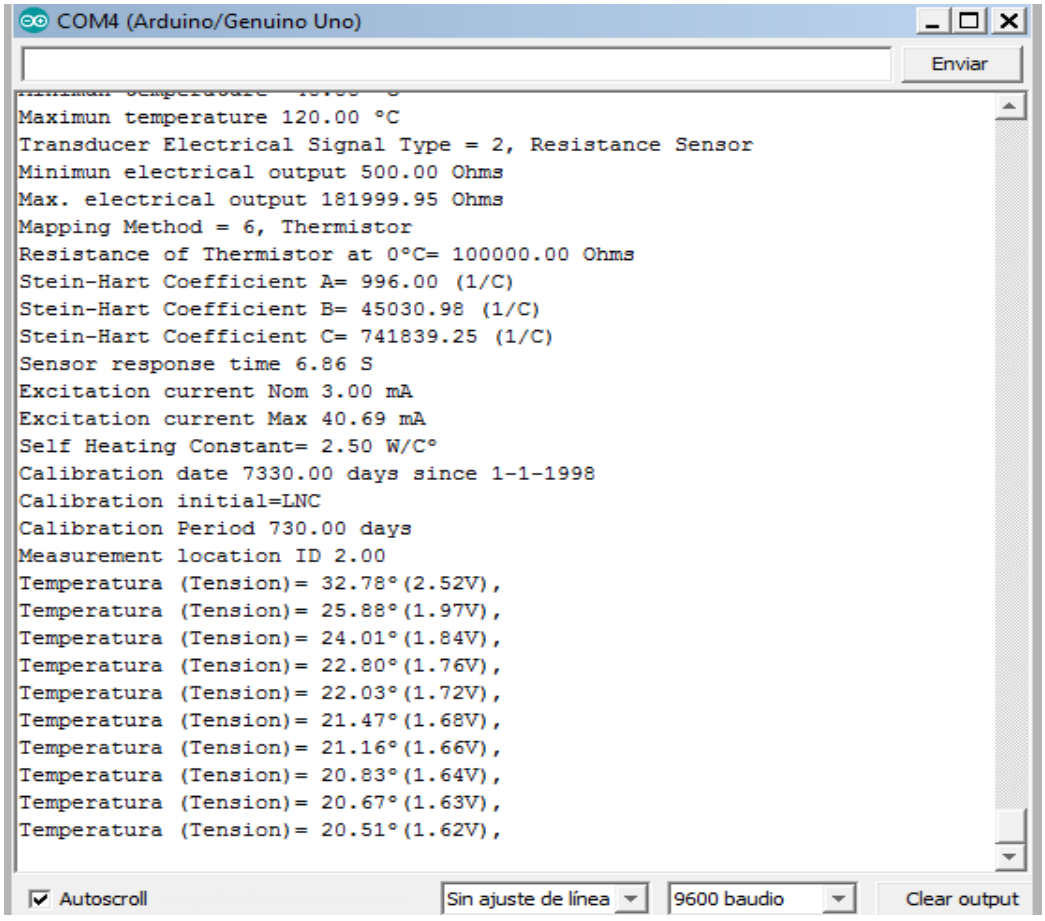


Figura 3.3.6.6.7: Monitor serie Arduino.Resultados de la lectura del sensor NTC

3.3.6.7 Programación

Para obtener una ecuación que relacione la tensión que le llega a la entrada analógica del Arduino con la temperatura que mide el sensor se utilizan las siguientes ecuaciones:

- La ecuación de la curva logarítmica que describe el sensor NTC:

$$R_{NTC} = -9.612 \times \ln(\text{temperatura}) + 40.719$$ [Ecuación 3.3.6.7.1]

- La salida de la primera etapa de medición(divisor de tensión):

$$V_0 = \frac{5}{R_{NTC}+1}$$ [Ecuación 3.3.6.7.2]

- La salida de la etapa de amplificación o lo que lo mismo, la entrada al Arduino:

$$V_i = V_0\left(1 + \frac{R_2}{R_1}\right) = V_0\left(1 + \frac{3{,}1165K\Omega}{1K\Omega}\right) = V_0 \times 4{,}1165$$ [Ecuación 3.3.6.7.3]

Con estas 3 ecuaciones y despejando la temperatura en función de la tensión de entrada al Arduino V_i se obtiene la ecuación necesaria para introducir en la programación del microcontrolador:

$$\text{Temperatura} = e^{\frac{41.719-\frac{20.58}{V_1}}{9.612}}$$ [Ecuación 3.3.6.7.4]

CÓDIGO ARDUINO: ESCRITURA/LECTURA EEPROM DS2431 DE LA INFORMACION DEL SENSOR NTC

```
#include <OneWire.h>


OneWire  ds(2);   // 1-wire on pin 2
byte    addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)
void setup(void)
{
 Serial.begin(9600);
}


void loop(void)
{
 //byte i;
 byte dat[15];


 SearchAddress(addr); //lectura y comprobación de la ROM


//Grabamos los DATOS DE LA PLANTILLA en la EPROM DS2431
```

```
dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(0,dat);  //(posicion donde empieza a guardar los datos)

dat[0] = 0x09;
dat[1] = 0x85;
dat[2] = 0x01;
dat[3] = 0x40;
dat[4] = 0x00;
dat[5] = 0x7D;
dat[6] = 0x2C;
dat[7] = 0x6F;
  ..
WriteRow(1,dat); //(posicion donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x04;
dat[2] = 0xEC;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x03;
dat[6] = 0xE4;
dat[7] = 0x00;

WriteRow(2,dat);  //(posicion donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0xAF;
```

```
dat[2] = 0xE7;
dat[3] = 0x00;
dat[4] = 0x0B;
dat[5] = 0x51;
dat[6] = 0xD0;
dat[7] = 0xF2;

WriteRow(3,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0xCB;
dat[1] = 0xB0;
dat[2] = 0xC3;
dat[3] = 0x94;
dat[4] = 0x4C;
dat[5] = 0x70;
dat[6] = 0xCB;
dat[7] = 0x68;

WriteRow(4,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x01;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
dat[4] = 0x00;
dat[5] = 0x00;
dat[6] = 0x00;
dat[7] = 0x00;

WriteRow(5,dat);  //(posición donde empieza a guardar los datos)

dat[0] = 0x00;
dat[1] = 0x00;
dat[2] = 0x00;
dat[3] = 0x00;
```

```
  dat[4] = 0x00;
  dat[5] = 0x00;
  dat[6] = 0x00;
  dat[7] = 0x00;

  WriteRow(6,dat);  //(posición donde empieza a guardar los datos)

  dat[0] = 0x00;
  dat[1] = 0x00;
  dat[2] = 0x00;
  dat[3] = 0x00;
  dat[4] = 0x00;
  dat[5] = 0x00;
  dat[6] = 0x00;
  dat[7] = 0x00;

  WriteRow(15,dat);

  ReadAllMem();    //print all mem content

  while(1);
}

void SearchAddress(byte* address) //Search for address and confirm it
// Esta funcion lee el codigo grabado por LASER de la EEPROM que la hace unica
{
  int i;
  if ( !ds.search(address))
  {
    Serial.print("No device found.\n");
    ds.reset_search();
    delay(250);
    return;
  }
  Serial.print("ADDR= ");
```

```
  for( i = 0; i < 8; i++)
  {
    Serial.print(address[i], HEX);
    Serial.print(" ");
  }

  if ( OneWire::crc8( address, 7) != address[7])
  {
     Serial.print("CRC is not valid, address is corrupted\n");
     return;
  }

  if ( address[0] != 0x2D)
  {
     Serial.print("Device is not a 1-wire Eeprom.\n");
     return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);       // Read Scratchpad
```

```
  for ( i = 0; i < 13; i++)
    data[i] = ds.read();
 }

 void CopyScratchPad(byte* data)
 {
  ds.reset();
  ds.select(addr);
  ds.write(0x55,1);  // Copy ScratchPad
  ds.write(data[0],1);
  ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
  ds.write(data[2],1);
  delay(25); // Waiting for copy completion
  //Serial.print("Copy done!\n");
 }

 void ReadAllMem()
 {
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0xF0,1);  // Read Memory
  ds.write(0x00,1);  //Read Offset 0000h
  ds.write(0x00,1);

  for ( i = 0; i < 144; i++) //whole mem is 144
  {
    Serial.print("byte");
    Serial.print("(");
    Serial.print(i);
    Serial.print(") ");
    Serial.print(ds.read(), HEX);
    Serial.println(" ");
  }
  Serial.println();
```

```
}

void WriteRow(byte row, byte* buffer)
{
 int i;
 if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
  return;              //The remaining are for the 64 bits register page

 WriteReadScratchPad(row*8, 0x00, buffer);

 /*  Print result of the ReadScratchPad
 for ( i = 0; i < 13; i++)
 {
  Serial.print(buffer[i], HEX);
  Serial.print(" ");
 }
 */
 CopyScratchPad(buffer);
     }
```

CÓDIGO ARDUINO: LECTURA DE LA EEPROM DS2431 Y DECODIFICACIÓN DE LA INFORMACION DEL SENSOR NTC

```
#include <OneWire.h>

OneWire  ds(2);    // 1-wire on pin 2
byte     addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)

unsigned char EPROM[145];
unsigned char BINARIO[336];
int valor;
int posicion;
int i; //i es la posición del byte de la EPROM[] en la que nos encontramos
int j; //j es la posición de bit del vector BINARIO[]
float base;
```

```
float ID;
int IDint; //variable para pasar ID (tipo float) a IDint (entero)
float pot=0; //
float Vi;
float Vo;
float Temperatura;

//variables para Template ID 38
float minphisval; //Minimun Temperature
float maxphisval; //Maximun Temperature
float minelecval; //Minimun electrical output
float maxelecval; //Max. electrical output
float R0NTC; //R0 Resistance NTC
float SteinhardtA; //Stein-Hart Coefficient A
float SteinhardtB; //Stein-Hart Coefficient B
float SteinhardtC; //Stein-Hart Coefficient C
float RTDcurve; //RTD curve
double RespTime; // Sensor response time
double ExciteAmplNom; //Excitation current, nom
double ExciteAmplMax; //Excitation current, max
float SelfHeating; //Self Heating Constant
float CalDate; // Calibration date
char Calinitials; //Calibration initials
float CalPeriod; //Calibration Period
float MeasID; //Measurement location ID

void setup()
{
Serial.begin(9600);

Serial.println("ROM identificacion del sensor");
SearchAddress(addr); //lectura y comprobación de la ROM
Serial.println(" ");

ReadAllMem();   //print all mem content
```

```
//visualizo todo el vector EPROM[] 42 bytes

 Serial.println("CONTENIDO DE LA EEPROM ");
 for ( i = 0; i < 42; i++){ // "i" va a ser un valor que enviaremos al SLAVE como valor de posición de la EPROM

 Serial.print("valor EPROM posicion ");
 Serial.print(i);
 Serial.print("=");
 Serial.println(EPROM[i],HEX); //mostramos el valor almacenado
 delay(100);
 }

 //convierto los datos del vector EPROM[] a binario y almaceno los datos en el vector BINARIO[]

 for ( i = 0; i < 42; i++){

/* Serial.println(" ");
  Serial.print("///////////////byte ");
  Serial.println(i);
*/
  //i es la posición del byte que nos encontramos de la EPROM

  valor=EPROM[i];

  //usamos una variable "j" para posicionarnos de 7 al 0 (8bits de derecha a izquierda))
  for ( j = 7; j >= 0; j--){

   posicion=i*8+j;

/*  Serial.print("bit ");
   Serial.println(posicion); //mostramos valores de J
*/
```

```
      //detectamos si es par o impar y hacemos las divisiones
       if (valor & 0x01){ //comparacion binaria bit a bit AND (si da 1 impar si da 0 es par)

       //Serial.println("es impar");
       valor=valor-1;
       valor=valor/2;
       BINARIO[posicion]=1;// como es impar el resto vale 1
       }
       else {
       //Serial.println("es par");
       valor=valor/2;
       BINARIO[posicion]=0; //como es par el resto vale 0
       }

/*    Serial.println(BINARIO[posicion]); //mostramos valores de J
     Serial.println(" ");
     delay(100);
*/
    }
  }

  Serial.print(" ");
  Serial.println("CONVERSION A BINARIO ");

  for ( i = 0; i < 337; i++){
  base=BINARIO[i];
  Serial.print("bit ");
  Serial.print(i);
  Serial.print("=");
  Serial.println(BINARIO[i]);
  }

  //SELECTOR (2 bits 65 y 66)
  if (BINARIO[65]==0 & BINARIO[66]==0) {
      Serial.println("The template is defined by IEEE1451.4 ");
```

```
      }

    //decodificacion e identifiación de la ID de la plantilla, 8 bits del 66 al 73 del vector
BINARIO[]

      for ( i = 66; i < 74; i++){
      base=BINARIO[i];
      ID=ID+base*pow(2,73-i);
       }

      Serial.print("Plantilla ID ");
      Serial.println(ID);
       IDint= int(ID);
       IDint= IDint+1;
      Serial.print("Int ID ");
      Serial.println(IDint);

    }//fin de void setup

    void loop()
    {

      //PLANTILLA ID 38
      if (IDint == 38){

      Serial.println(" ");
      Serial.println("//////TEMPLATE ID 38/////////");

    //   delay(1000);
        minimun_temperature(); //Llamo a la funcion ("%Min Phys Val")
    //   delay(1000);
        maximun_temperature();//Llamo a la funcion ("%Max Phys Val")
    //   delay(1000);
        Serial.println("Transducer Electrical Signal Type = 2, Resistance Sensor");
    //   delay(1000);
        minimun_electrical_output(); //Llamo a la funcion  ("%Min Elec Val")
```

```
//    delay(1000);
    Max_electrical_output();//Llamo a la funcion  ("%Max Elec Val")
//    delay(1000);
    Serial.println("Mapping Method = 6, Thermistor");
//    delay(1000);
    R0_resistanceNTC();
//    delay(1000);
   SteinHartA();
//    delay(1000);
   SteinHartB();
//    delay(1000);
   SteinHartC();
//    delay(1000);

    Sensor_response_time();
//    delay(1000);
    Excitation_current_nom();
//    delay(1000);
    Excitation_current_max();
//    delay(1000);
    SelfHeatingConstant();
//    delay(1000);
    Calibration_date();
//    delay(1000);
    Calibration_initials();
  //    delay(1000);
    Calibration_period();
//    delay(1000);
    Measurement_location_ID();
//    delay(1000);

  IDint = 0; //reseteamos la variable ID

    //LECTURA DEL SENSOR
```

```
      for ( i = 0; i < 10; i++){ // mido 10 valores

        Vi=analogRead(0);
        Vo=(5*Vi)/1023;

        Temperatura=exp((41.719-(20.58/Vo))/9.612);
        Serial.print("Temperatura (Tension)= ");
        Serial.print(Temperatura);
        Serial.print("º");
        Serial.print("(");
        Serial.print(Vo);
        Serial.print("V");
        Serial.print(")");
        Serial.println(", ");
        delay(1000);
      }

      }// fin IF ID38

    } //Fin de void loop()

    ///FUNCIONES PLANTILLA 38/////////////////////////

    void minimun_temperature(){
        //Minimun Temperature("%Min Phys Val")
        for ( i = 74; i < 85; i++){  //bits del 74 al 84
        base=BINARIO[i];
        minphisval=minphisval+base*pow(2,84-i); //CUIDADO las potencias se usan variables
FLOAT si no redondea MAL!!!
        delay(100);
        }
        minphisval=minphisval-200;
        Serial.print("Minimun temperature ");
        Serial.print(minphisval);
        Serial.println(" ºC");
    }
```

```
void maximun_temperature(){
      //Maximun Temperature ("%Max Phys Val")
      for ( i = 85; i < 96; i++){  //bits del 85 al 95
      base=BINARIO[i];
      maxphisval=maxphisval+base*pow(2,95-i);
      delay(100);
      }
      maxphisval=maxphisval-200;
      Serial.print("Maximun temperature ");
      Serial.print(maxphisval);
      Serial.println(" ºC");
}

void minimun_electrical_output(){
      //Minimun electrical output ("%Min Elec Val")
      for ( i = 96; i < 114; i++){  //bits del 96 al 113
      base=BINARIO[i];
      minelecval=minelecval+base*pow(2,113-i);       delay(100);
      }
      Serial.print("Minimun electrical output ");
      Serial.print(minelecval);
      Serial.println(" Ohms");
}

void Max_electrical_output(){
      //Max. electrical output
      for ( i = 114; i < 132; i++){  //bits del 114 al 131
      base=BINARIO[i];
      maxelecval=maxelecval+base*pow(2,131-i);
      delay(100);
      }
      Serial.print("Max. electrical output ");
      Serial.print(maxelecval);
      Serial.println(" Ohms");
```

```
}

void R0_resistanceNTC(){
    //R0 resistance (%RTDCoef_R0,)
    for ( i = 132; i < 152; i++){  // bits 132 y 151
    base=BINARIO[i];
    R0NTC=R0NTC+base*pow(2,151-i);
    delay(100);
    }
    R0NTC= 100000*pow(1.0000126,R0NTC)-1604.89;
    Serial.print("Resistance of Thermistor at 0ºC= ");
    Serial.print(R0NTC);
    Serial.println(" Ohms");
}

void SteinHartA(){
    //Stein-Hart Coefficient A
    for ( i = 152; i < 184; i++){  //bits del 152 al 183
    base=BINARIO[i];
    SteinhardtA=SteinhardtA+base*pow(2,183-i);
    delay(100);
    }

    Serial.print("Stein-Hart Coefficient A= ");
    Serial.print(SteinhardtA);
    Serial.println(" (1/C)");
}

void SteinHartB(){
    //Stein-Hart Coefficient B
    for ( i = 184; i < 216; i++){  //bits del 184 al 215
    base=BINARIO[i];
    SteinhardtB=SteinhardtB+base*pow(2,215-i);
    delay(100);
    }
```

```
        Serial.print("Stein-Hart Coefficient B= ");
        Serial.print(SteinhardtB);
        Serial.println(" (1/C)");
    }

    void SteinHartC(){
        //Stein-Hart Coefficient C
        for ( i = 216; i < 248; i++){  //bits del 216 al 247
        base=BINARIO[i];
        SteinhardtC=SteinhardtC+base*pow(2,247-i);
        delay(100);
        }

        Serial.print("Stein-Hart Coefficient C= ");
        Serial.print(SteinhardtC);
        Serial.println(" (1/C)");
    }

    void Sensor_response_time(){
        //Senor response time
        for ( i = 248; i < 254; i++){  //bits del 248 al 253
        base=BINARIO[i];
        RespTime=RespTime+base*pow(2,253-i);
        delay(100);
        }
        RespTime= 0.000001*pow(1.3,RespTime);
        Serial.print("Sensor response time ");
        Serial.print(RespTime);
        Serial.println(" S");
    }

    void Excitation_current_nom(){
        //Excitation current, nom.
        for ( i = 254; i < 262; i++){  //bits del 254 al 261
```

```
        base=BINARIO[i];
        ExciteAmplNom=ExciteAmplNom+base*pow(2,261-i);
        delay(100);
        }
        ExciteAmplNom= 0.001*pow(1.046,ExciteAmplNom);
        Serial.print("Excitation current Nom ");
        Serial.print(ExciteAmplNom);
        Serial.println(" mA");
    }

    void Excitation_current_max(){
        //Excitation current, max
            for ( i = 262; i < 270; i++){  //bits del 262 al 269
        base=BINARIO[i];
        ExciteAmplMax=ExciteAmplMax+base*pow(2,269-i);
        delay(100);
        }
        ExciteAmplMax= 0.001*pow(1.046,ExciteAmplMax);
        Serial.print("Excitation current Max ");
        Serial.print(ExciteAmplMax);
        Serial.println(" mA");
    }

    void   SelfHeatingConstant(){

        //Self Heating Constant
        for ( i = 270; i < 275; i++){  //bits del 270 al 274
        base=BINARIO[i];
        SelfHeating=SelfHeating+base*pow(2,274-i);
        delay(100);
        }
        SelfHeating= 0.00025*pow(1.148,SelfHeating);
        SelfHeating=2.5;
        Serial.print("Self Heating Constant= ");
        Serial.print(SelfHeating);
```

```
            Serial.println(" W/Cº");
    }

    void Calibration_date(){
        //Calibration date
        for ( i = 275; i < 291; i++){  //bits del 275 al 290
        base=BINARIO[i];
        CalDate=CalDate+base*pow(2,290-i);
        delay(100);
        }
        Serial.print("Calibration date ");
        Serial.print(CalDate);
        Serial.println(" days since 1-1-1998");
    }

    void Calibration_initials(){
        //Calibration initial, bits 291 al 305
            if (BINARIO[291]==0 & BINARIO[292]==1 & BINARIO[293]==1 & BINARIO[294]==0 &
BINARIO[295]==0 & BINARIO[296]==0 & BINARIO[297]==1 & BINARIO[298]==1 &
BINARIO[299]==1 & BINARIO[300]==0 & BINARIO[301]==0 & BINARIO[302]==0 &
BINARIO[303]==0 & BINARIO[304]==1 & BINARIO[305]==1 ) {
        Serial.println("Calibration initial=LNC");
         }
    }

    void Calibration_period(){
        //Calibration period
        for ( i = 306; i < 318; i++){  //bits del 306 al 317
        base=BINARIO[i];
        CalPeriod=CalPeriod+base*pow(2,317-i);
        delay(100);
        }
        Serial.print("Calibration Period ");
        Serial.print(CalPeriod);
        Serial.println(" days");
    }
```

```
void Measurement_location_ID(){
    //Measurement location ID
    for ( i = 318; i < 329; i++){  //bits del 318 al 328
    base=BINARIO[i];
    MeasID=MeasID+base*pow(2,328-i);
    delay(100);
    }
    Serial.print("Measurement location ID ");
    Serial.println(MeasID);
}

//FUNCIONES DE LA EEPROM ONE WIRE

void SearchAddress(byte* address) //Search for address and confirm it
// Esta función lee el código gravado por LASER de la EEPROM que la hace única
{
 int i;
 if ( !ds.search(address))
 {
  Serial.print("No device found.\n");
  ds.reset_search();
  delay(250);
  return;
 }

 Serial.print("ADDR= ");
 for( i = 0; i < 8; i++)
 {
  Serial.print(address[i], HEX);
  Serial.print(" ");
 }

 if ( OneWire::crc8( address, 7) != address[7])
 {
```

```
    Serial.print("CRC is not valid, address is corrupted\n");
    return;
  }

  if ( address[0] != 0x2D)
  {
    Serial.print("Device is not a 1-wire Eeprom.\n");
    return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);       // Read Scratchpad

  for ( i = 0; i < 13; i++)
    data[i] = ds.read();
}

void CopyScratchPad(byte* data)
{
  ds.reset();
  ds.select(addr);
```

```
  ds.write(0x55,1);  // Copy ScratchPad
  ds.write(data[0],1);
  ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
  ds.write(data[2],1);
  delay(25); // Waiting for copy completion
  //Serial.print("Copy done!\n");
}

void ReadAllMem()
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0xF0,1);  // Read Memory
  ds.write(0x00,1);  //Read Offset 0000h
  ds.write(0x00,1);

  for ( i = 0; i < 144; i++) //whole mem is 144
  {
//    Serial.print("byte");
//    Serial.print("(");
//    Serial.print(i);
//    Serial.print(") ");
//    Serial.print(ds.read(), HEX);
    EPROM[i]=ds.read();
    Serial.println(" ");
  }
  Serial.println();
}

void WriteRow(byte row, byte* buffer)
{
  int i;
  if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
    return;              //The remaining are for the 64 bits register page
```

```
  WriteReadScratchPad(row*8, 0x00, buffer);

  /*  Print result of the ReadScratchPad
  for ( i = 0; i < 13; i++)
  {
    Serial.print(buffer[i], HEX);
    Serial.print(" ");
  }
  */
  CopyScratchPad(buffer);
}
```

3.4 PRUEBAS Y RESULTADOS FINALES

Una vez implementado el estándar IEEE1451 en los 3 sensores, se realiza un último montaje para comprobar y demostrar el funcionamiento del sistema de auto-identificación.

En el montaje se utilizan 3 memorias DS2431 programadas con el *datasheet* de cada uno de los sensores diseñados en el anexo anterior:

- Sensor PT1000
- Sensor LDR
- Sensor NTC

Para simplificar el montaje, como los sensores han sido acondicionados y amplificados para proporcionar una señal analógica de 0-5V a su salida adaptada a la lectura de las entradas analógicas de Arduino, se ha utilizado un potenciómetro que simulará dicha señal y representará la salida de cualquiera de los 3 sensores.

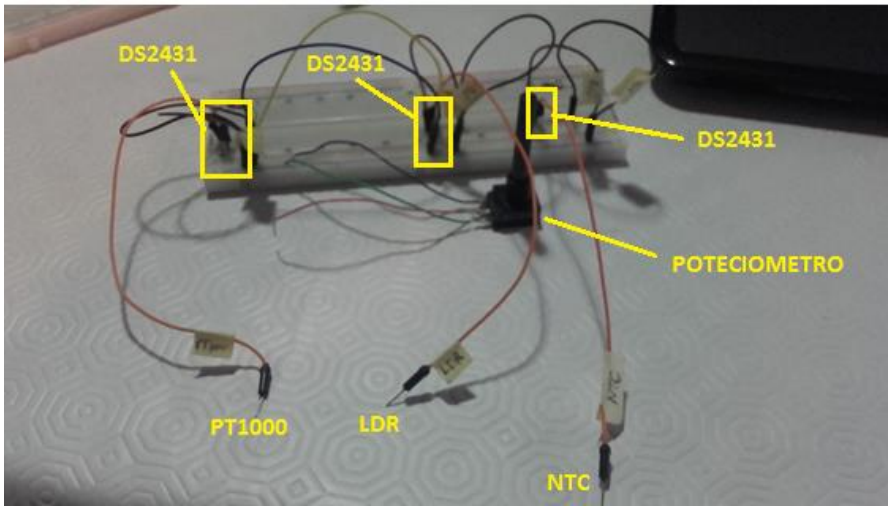


Figura 3.4.1 Montaje de pruebas estándar IEEE1451.4

Para las primeras pruebas, lo primero que se hace es conectar el potenciómetro a la entrada analógica A0 de arduino y la memoria DS2431 de la PT1000 al pin digital 2.

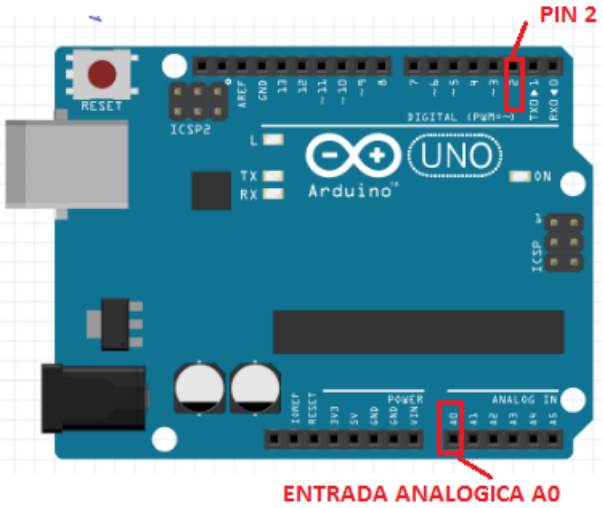


Figura 3.4.2: Montaje de pruebas estándar IEEE1451.4

Una vez arrancado el programa, Arduino pasa a leer el contenido de la memoria EEPROM, lo almacena, decodifica el código, e interpreta el contenido de la información, detectando el tipo de sensor es (en este caso la plantilla ID=37 RTD).

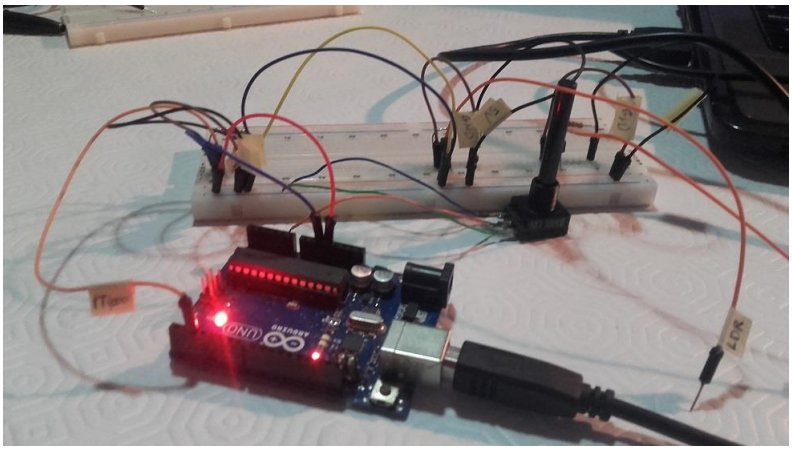


Figura 3.4.3: Montaje y prueba del sensor PT1000

Para finalizar, se muestra por el monitor serie de Arduino el contenido de la plantilla así como la lectura analógica de este sensor en su escala, unidades de medida y nivel de tensión.

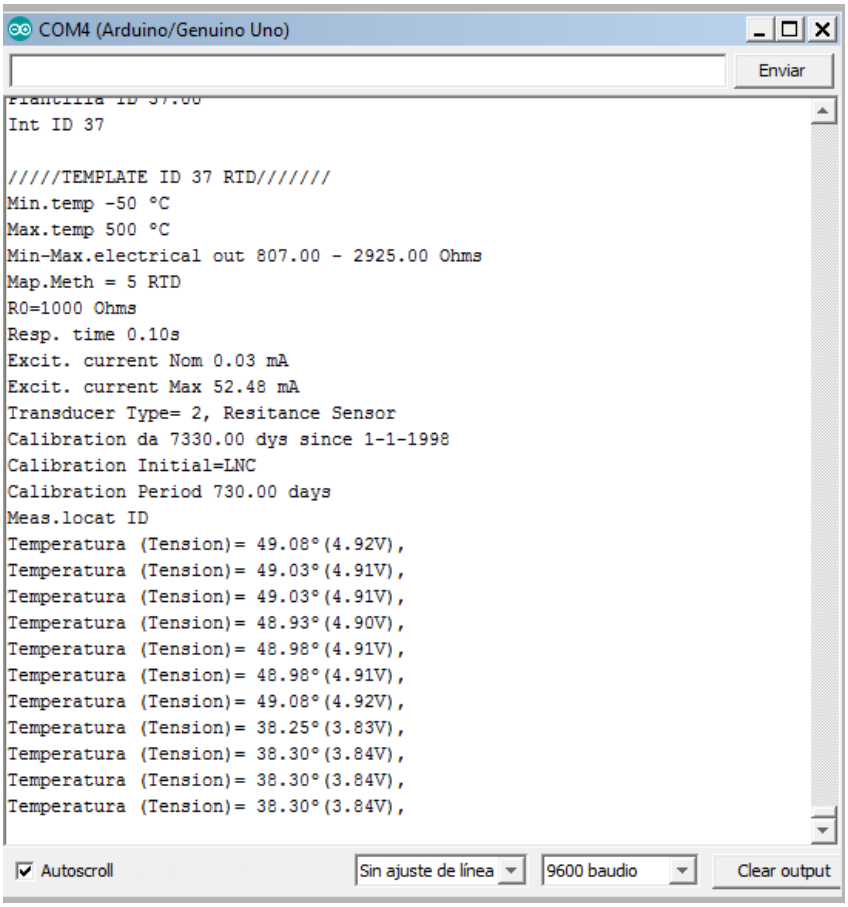


Figura 3.4.4: Monitor serie Arduino. Resultados PT1000

A continuación se desconecta la PT1000 de Arduino y se procede a conectar el sensor LDR. Una vez se ejecuta el siguiente ciclo de programa se observa que la detección y decodificación de la LDR se realiza con éxito de la misma forma que el anterior sensor.

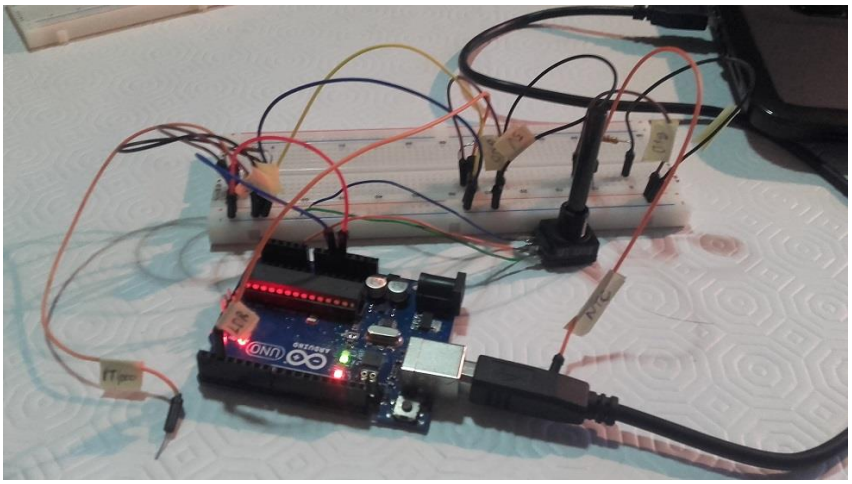


Figura 3.4.5: Montaje y prueba del sensor LDR

Igual que en sensor anterior, se muestra por el monitor serie de Arduino el contenido de la plantilla así como la lectura analógica de este sensor en su escala, unidades de medida y nivel de tensión.

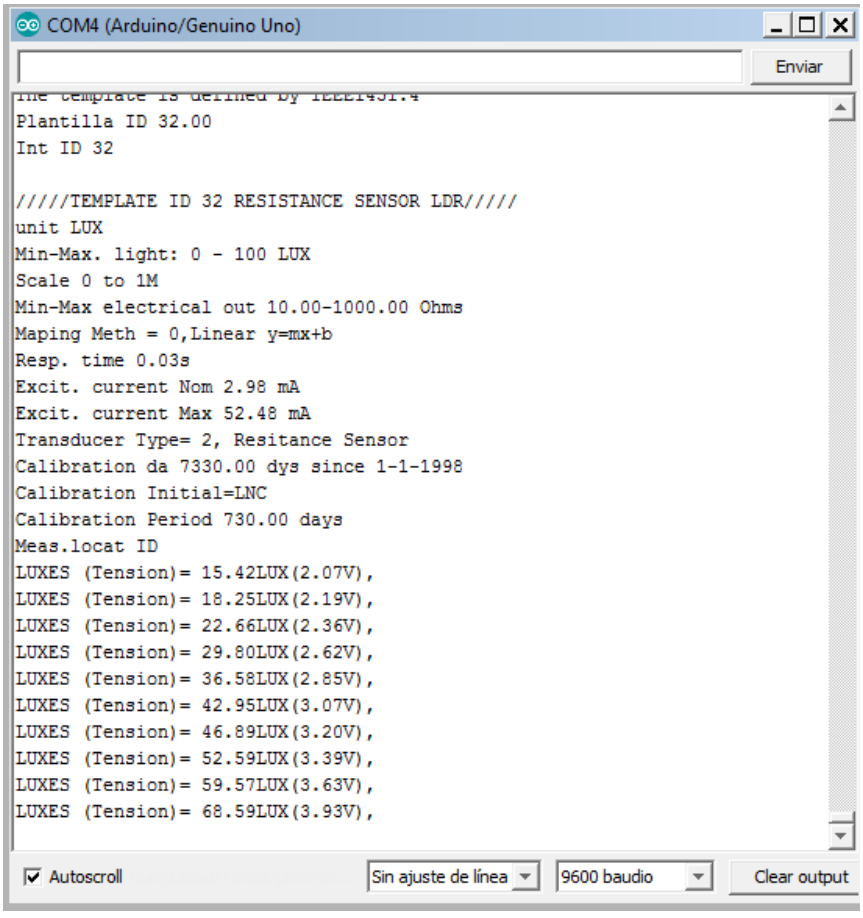


Figura 3.4.6: Monitor serie Arduino. Resultados LDR

Por último, se realiza la misma operación con el sensor NTC obteniendo el resultado esperado.

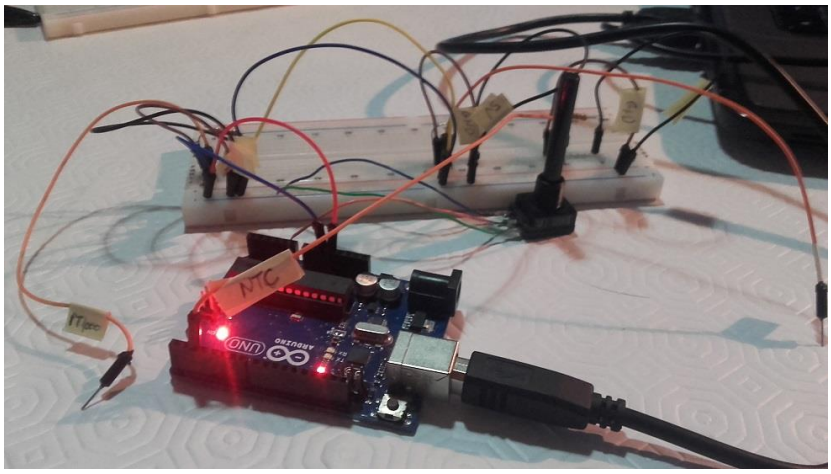


Figura 3.4.7: Montaje y prueba del sensor NTC

Se muestra por el monitor serie de Arduino el contenido de la plantilla así como la lectura analógica de este sensor en su escala, unidades de medida y nivel de tensión

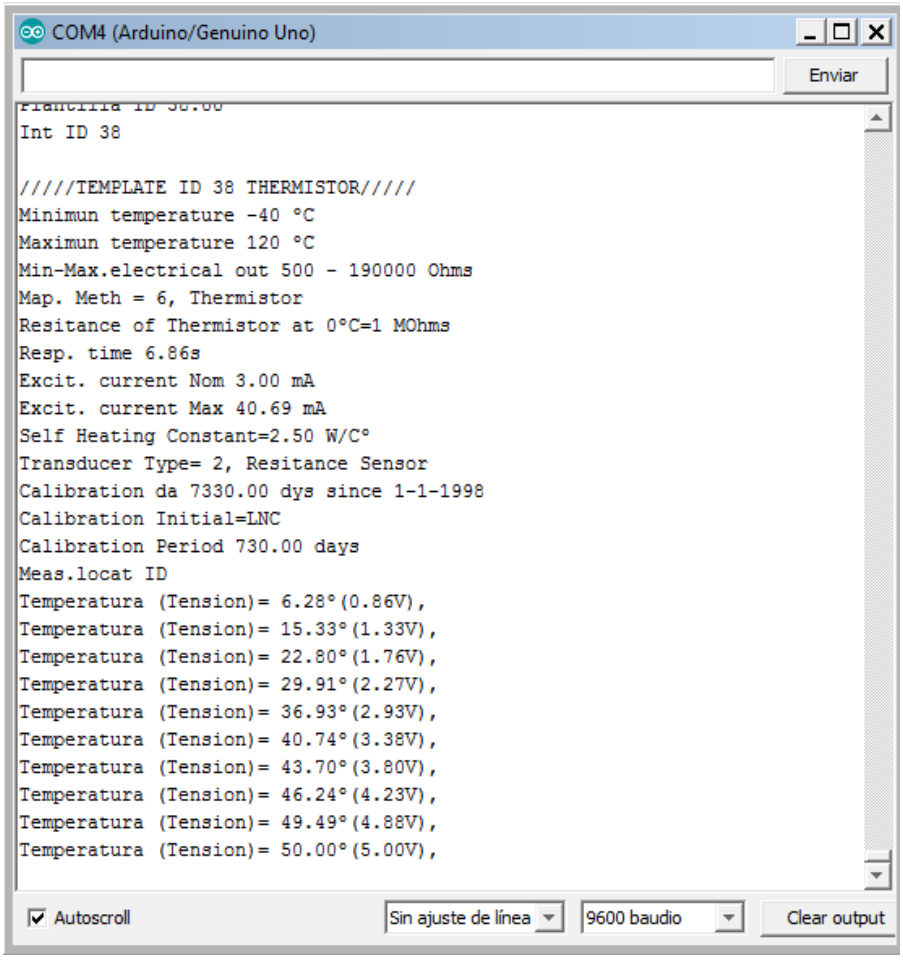


Figura 3.4.8: Monitor serie Arduino. Resultados NTC

CÓDIGO ARDUINO: PRUEBA 3 SENSORES ESTÁNDAR IEEE1451.4

```
#include <OneWire.h>


OneWire  ds(2);    // 1-wire on pin 2
byte     addr[8];  // Contains the eeprom unique ID (ROM 64bits: 8bit family code, 48 bit serial NUMBER y 8bit CRC)
unsigned char EPROM[145];
unsigned char BINARIO[336];
int valor;
int posicion;
int i; //i es la posición del byte de la EPROM[] en la que nos encontramos
int j; //j es la posición de bit del vector BINARIO[]
float base;
float ID;
int IDint;
float pot=0; //
float Vi;
float Vo;
float Temperatura;
float LUX;


void setup()
{
Serial.begin(9600);

}//fin de void setup


void loop()
{
delay(100);
//TRATAMIENTO DATOS DE LA EEPROM DS2431


Serial.println("ROM identificacion del sensor");
SearchAddress(addr); //lectura y comprobación de la ROM
Serial.println(" ");


ReadAllMem();    //print all mem content
```

```
      //visualizo todo el vector EPROM[] 42 bytes


        Serial.println("CONTENIDO DE LA EEPROM");
       for ( i = 0; i < 42; i++){ // "i" va a ser un valor que enviaremos al SLAVE como valor de
posición de la EPROM


       Serial.print("valor EPROM posicion ");
       Serial.print(i);
       Serial.print("=");
       Serial.println(EPROM[i],HEX); //mostramos el valor almacenado
       delay(100);
       }


      //convierto los datos del vector EPROM[] a binario y almaceno los datos en el vector
BINARIO[]


       for ( i = 0; i < 42; i++){


     /* Serial.println(" ");
       Serial.print("////////////////byte ");
       Serial.println(i);
     */
        //i es la posición del byte que nos encontramos de la EPROM


        valor=EPROM[i];


        //usamos una variable "j" para posicionarnos de 7 al 0 (8bits de derecha a izquierda))
        for ( j = 7; j >= 0; j--){


         posicion=i*8+j;


     /*   Serial.print("bit ");
         Serial.println(posicion); //mostramos valores de J
     */


         //detectamos si es par o impar y hacemos las divisiones
           if (valor & 0x01){


           //Serial.println("es impar");
           valor=valor-1;
```

```
          valor=valor/2;
          BINARIO[posicion]=1;// como es impar el resto vale 1
          }
          else {
          //Serial.println("es par");
          valor=valor/2;
          BINARIO[posicion]=0; //como es par el resto vale 0
          }

    /*    Serial.println(BINARIO[posicion]); //mostramos valores de J
          Serial.println(" ");
          delay(100);
    */
       }
      }

      Serial.print(" ");
      Serial.println("CONVERSION A BINARIO ");

      for ( i = 0; i < 337; i++){
      base=BINARIO[i];
      Serial.print("bit ");
      Serial.print(i);
      Serial.print("=");
      Serial.println(BINARIO[i]);
      }

      //SELECTOR (2 bits 65 y 66)
      if (BINARIO[65]==0 & BINARIO[66]==0) {
          Serial.println("The template is defined by IEEE1451.4 ");
       }


     //decodificacion e identifiación de la ID de la plantilla, 8 bits del 66 al 73 del vector
BINARIO[]

      for ( i = 66; i < 74; i++){
      base=BINARIO[i];
     ID=ID+base*pow(2,73-i); //CUIDADO las potencias se usan variables FLOAT si no
redondea MAL!!!
```

```
        }

       Serial.print("Plantilla ID ");
       Serial.println(ID);
        IDint= int(ID);
        IDint= IDint+1; //sumamos uno porque al pasar de FLOAT a INT redondea a la baja (ej 37.00=36)
       Serial.print("Int ID ");
       Serial.println(IDint);



       //PLANTILLA ID 32
       if (IDint == 32){

       Serial.println(" ");
       Serial.println("/////TEMPLATE ID 32 RESISTANCE SENSOR LDR/////");
       Serial.println("unit LUX");
       Serial.println("Min-Max. light: 0 - 100 LUX");
       Serial.println("Scale 0 to 1M");
       Serial.println("Min-Max electrical out 10.00-1000.00 Ohms ");
       Serial.println("Maping Meth = 0,Linear y=mx+b");
       Serial.println("Resp. time 0.03s");
       Serial.println("Excit. current Nom 2.98 mA");
       Serial.println("Excit. current Max 52.48 mA");
       datos();



        //LECTURA DEL SENSOR

       for ( i = 0; i < 10; i++){ // mido 10 valores

        Vi=analogRead(0);
        Vo=(5*Vi)/1023;

        LUX=exp((13.516-(15/Vo))/2.295);
        Serial.print("LUXES (Tension)= ");
        Serial.print(LUX);
        Serial.print("LUX");
        Serial.print("(");
```

```
  Serial.print(Vo);
  Serial.println("V), ");


  delay(1000);
 }
ID=0;
delay(20000);
software_Reset();
 }// fin IF ID32


 //PLANTILLA ID 37
 if (IDint == 37){


 Serial.println(" ");
 Serial.println("/////TEMPLATE ID 37 RTD///////");

 Serial.println("Min.temp -50 ºC ");
 Serial.println("Max.temp 500 ºC ");
 Serial.println("Min-Max.electrical out 807.00 - 2925.00 Ohms");
 Serial.println("Map.Meth = 5 RTD ");
 Serial.println("R0=1000 Ohms ");
//  Serial.println("CVD Coef A = 0.0039083");
//  Serial.println("CVD Coef B = -0.0000005775");
//  Serial.println("CVD Coef C = -0.000000000004183");
 Serial.println("Resp. time 0.10s");
 Serial.println("Excit. current Nom 0.03 mA");
 Serial.println("Excit. current Max 52.48 mA");
 datos();


 //LECTURA DEL SENSOR

 for ( i = 0; i < 11; i++){ // mido 10 valores

  Vi=analogRead(0);
  Vo=(5*Vi)/1023;
```

```
   Temperatura=9.9817*Vo;
   Serial.print("Temperatura (Tension)= ");
   Serial.print(Temperatura);
   Serial.print("º");
   Serial.print("(");
   Serial.print(Vo);
   Serial.println("V), ");

   delay(1000);
  }
 ID=0;
 delay(20000);
 software_Reset();
  }// fin IF ID37


  //PLANTILLA ID 38
  if (IDint == 38){

  Serial.println(" ");
  Serial.println("/////TEMPLATE ID 38 THERMISTOR/////");

  Serial.println("Minimun temperature -40 ºC");
  Serial.println("Maximun temperature 120 ºC");
  Serial.println("Min-Max.electrical out 500 - 190000 Ohms");
  Serial.println("Map. Meth = 6, Thermistor ");
  Serial.println("Resitance of Thermistor at 0ºC=1 MOhms");
//  Serial.println ("Stein-Hart Coefficients A=996 (1/C)");
//  Serial.println ("Stein-Hart Coefficients B=45030.98 (1/C)");
//  Serial.println ("Stein-Hart Coefficients C=741839.25 (1/C)");
  Serial.println("Resp. time 6.86s");
  Serial.println("Excit. current Nom 3.00 mA");
  Serial.println("Excit. current Max 40.69 mA");
  Serial.println("Self Heating Constant=2.50 W/Cº");
  datos();


    //LECTURA DEL SENSOR
```

```
  for ( i = 0; i < 10; i++){ // mido 10 valores

    Vi=analogRead(0);
    Vo=(5*Vi)/1023;

    Temperatura=exp((41.719-(20.58/Vo))/9.612);
    Serial.print("Temperatura (Tension)= ");
    Serial.print(Temperatura);
    Serial.print("º");
    Serial.print("(");
    Serial.print(Vo);
    Serial.println("V), ");

    delay(1000);
  }
 ID=0;
 delay(20000);
 software_Reset();
  }// fin IF ID38


} //Fin de void loop()

//-------------------------------------------------------------------------------------


//FUNCIONES DE LA EEPROM ONE WIRE

void SearchAddress(byte* address) //Search for address and confirm it
// Esta funcion lee el codigo gravado por LASER de la EEPROM que la hace unica
{
  int i;
  if ( !ds.search(address))
  {
    Serial.print("No device found.\n");
    ds.reset_search();
    delay(250);
    return;
```

```
  }

  Serial.print("ADDR= ");
  for( i = 0; i < 8; i++)
  {
    Serial.print(address[i], HEX);
    Serial.print(" ");
  }

  if ( OneWire::crc8( address, 7) != address[7])
  {
     Serial.print("CRC is not valid, address is corrupted\n");
     return;
  }

  if ( address[0] != 0x2D)
  {
     Serial.print("Device is not a 1-wire Eeprom.\n");
     return;
  }
  Serial.println();
}

void WriteReadScratchPad(byte TA1, byte TA2, byte* data)
{
  int i;
  ds.reset();
  ds.select(addr);
  ds.write(0x0F,1);  // comando Write ScratchPad
  ds.write(TA1,1);
  ds.write(TA2,1);
  for ( i = 0; i < 8; i++)
    ds.write(data[i],1);

  ds.reset();
  ds.select(addr);
  ds.write(0xAA);        // Read Scratchpad

  for ( i = 0; i < 13; i++)
```

```
    data[i] = ds.read();
  }

  void CopyScratchPad(byte* data)
  {
   ds.reset();
   ds.select(addr);
   ds.write(0x55,1);  // Copy ScratchPad
   ds.write(data[0],1);
   ds.write(data[1],1);  // Send TA1 TA2 and ES for copy authorization
   ds.write(data[2],1);
   delay(25); // Waiting for copy completion
   //Serial.print("Copy done!\n");
  }

  void ReadAllMem()
  {
   int i;
   ds.reset();
   ds.select(addr);
   ds.write(0xF0,1);  // Read Memory
   ds.write(0x00,1);  //Read Offset 0000h
   ds.write(0x00,1);

   for ( i = 0; i < 144; i++) //whole mem is 144
   {
  //   Serial.print("byte");
  //   Serial.print("(");
  //   Serial.print(i);
  //   Serial.print(") ");
  //   Serial.print(ds.read(), HEX);
     EPROM[i]=ds.read();
     Serial.println(" ");
   }
   Serial.println();
  }

  void WriteRow(byte row, byte* buffer)
  {
```

```
  int i;
  if (row < 0 || row > 15) //There are 16 row of 8 bytes in the main memory
    return;              //The remaining are for the 64 bits register page

  WriteReadScratchPad(row*8, 0x00, buffer);

  /*  Print result of the ReadScratchPad
  for ( i = 0; i < 13; i++)
  {
    Serial.print(buffer[i], HEX);
    Serial.print(" ");
  }
  */
  CopyScratchPad(buffer);

}

void software_Reset()
// Restarts program from beginning but
// does not reset the peripherals and registers
{
  asm volatile ("  jmp 0");
}

void datos(){
  Serial.println("Transducer Type= 2, Resitance Sensor ");
  Serial.println("Calibration da 7330.00 dys since 1-1-1998");
  Serial.println("Calibration Initial=LNC ");
  Serial.println("Calibration Period 730.00 days ");
  Serial.println("Meas.locat ID");
}
```

3.5 REDES INALAMBRICAS ZIGBEE

3.5.1 Introducción a las comunicaciones inalámbricas

Las redes inalámbricas utilizan ondas de radio para conectar los dispositivos, sin la necesidad de utilizar cables de ningún tipo. En algunos casos se utilizan en respaldo o sustitución de redes cableadas preexistentes, mientras que en otros casos se utilizan para nuevas utilidades, ya que pueden proporcionar acceso a datos desde ubicaciones muy remotas, que los sistemas cableados no son capaces de ofrecer. La infraestructura inalámbrica sirve a muchos propósitos y puede ser construida a muy bajo coste en comparación con las alternativas cableadas tradicionales.

Los protocolos más utilizados en redes inalámbricas y comunicaciones son: Wifi, WiMaX, Bluetooh, Zigbee, 3G-4G y GSM.

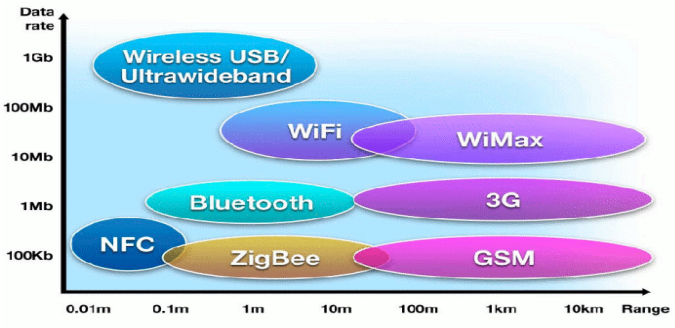


Figura 3.5.1.1: Protocolos de comunicaciones inalámbricas

Este proyecto fin de Master se centra en las comunicaciones tipo ZigBee, debido a las ventajas que ofrece.

ZigBee es la denominación de un conjunto de protocolos caracterizados por su alto nivel de comunicación inalámbrica, concebidos para su utilización con modulos de radio digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas WSN. Su ámbito es el de las aplicaciones que requieren por un lado, seguridad en las comunicaciones, por otro, una baja tasa de envío de datos, y finalmente una maximización de la vida útil de las baterías.

El ZigBee es una de las opciones más adecuadas para las aplicaciones relacionadas con el IoT, ya que al estar basado en redes inalámbricas requiere que la comunicación entre nodos funcionen con baterías, es decir, que el consumo de energía sea mínimo y el coste económico de los dispositivos sea bajo.

COMPARATIVA PRINCIPALES ESTÁNDARES DE COMUNICACIÓN			
	ZigBee	Bluetooth	WiFi
Estándar	802.15.4	802.15.1	802.11
AB	250 kbps	24 Mbps	Hasta 54Mbps
Consumo	20.8 mA en Tx 6 µA en reposo	40 mA en Tx 0.2 mA en reposo	400 mA en Tx 20 mA en reposo
Conexión	Se puede conectar a la red un gran número de dispositivos	Su conexión es limitada	Muchas redes conectadas a un administrador principal
Ámbito de aplicación	Dispositivos portátiles Domótica, iluminación sensores...	Dispositivos portátiles	Internet y redes de ordenadores

Figura 3.5.1.2: Comparativa de los principales estándares de comunicación

3.5.2 Clasificación de las redes inalámbricas

Los tipos de redes inalámbricas dependen de su alcance y del tipo de onda electromagnética utilizada. Según su alcance, de mayor a menor, nos encontramos las siguientes redes:

- Redes de área personal inalámbrica (WPAN: wireless personal area networks).
- Red de área metropolitana (WMAN: wireless metropolitan-area networks).
- Redes de área local inalámbrica (WLAN: wireless local area networks).
- Redes de área extendida inalámbrica (WWAN: wireless wide area networks).

Comparación tecnologías inalámbricas

Tipo de red	WWAN (Wide)	WMAN (Metropolitan)	WLAN (Local)	WPAN (Personal)
Estándar	GSM/GPRS/UMTS	IEEE 802.16	IEEE 802.11	IEEE 802.15
Certificación		WiMAX	WiFi	Bluetooth, ZigBee
Velocidad	9,6/170/2000 Kb/s	15-134 Mb/s	1-54 Mb/s	Hasta 721 Kb/s
Frecuencia	0,9/1,8/2,1 GHz	2-66 GHz	2,4 y 5 GHz Infrarrojos	2,4 GHz
Rango	35 Km	1 – 50 Km	30 - 150 m	10 m
Técnica radio	Varias	Varias	FHSS, DSSS, OFDM	FHSS
Itinerancia (roaming)	Sí	Sí (802.16e)	Sí	No
Equivalente a:	Conex. telef. (módem)	ADSL, CATV	LAN	Cables de conexión

Figura 3.5.2.1: Clasificación de las redes inalámbricas

Puede hacerse una representación gráfica de los tipos de red inalámbrica de acuerdo con su alcance, de modo que se refleje como las redes de menor alcance se insertan en las redes de mayor alcance:

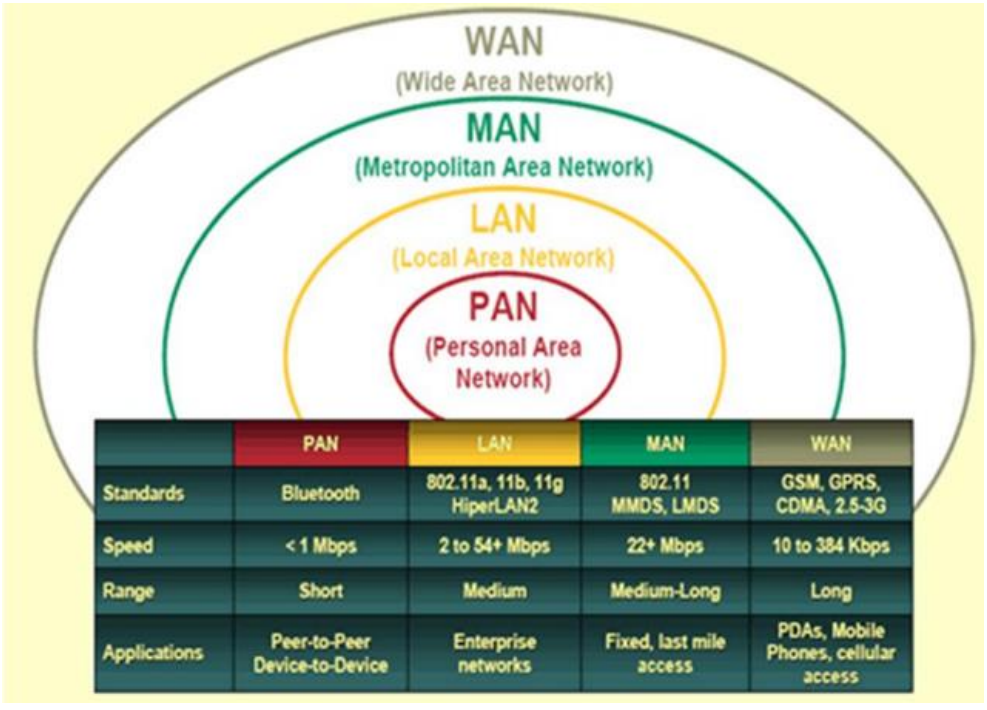


Figura 3.5.2.2: Alcance de las redes inalámbricas

3.5.3 Arquitectura Básica de una Red XBee

Una red XBee está formada por 3 tipos de elementos: un único dispositivo coordinador (C), varios dispositivos *routers* (R) y varios dispositivos finales o *end points* (E).

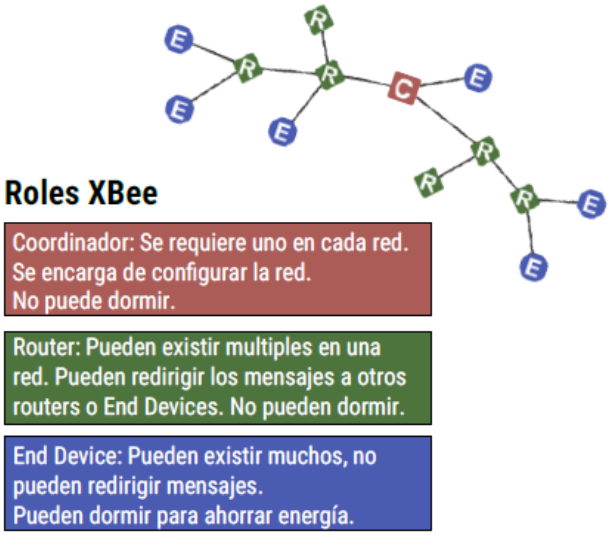


Figura 3.5.3.1: Arquitectura de redes Xbee

El Coordinador: es el nodo que tiene la única función de conformar una red, responsable de establecer el canal de comunicaciones y del PAN ID (identificador de red) para toda ella. Una vez establecidos los parámetros, el Coordinador puede formar una red, permitiendo unirse a él a dispositivos *Routers* y *End Points*, y a partir de ese momento el Coordinador hace las funciones de *Router*, esto es, participa en el enrutado de paquetes y es origen y/o destinatario de información.

Los *Routers*: es un nodo que determina la mejor ruta para enrutar un paquete de información mediante la creación y mantenimiento de información sobre la red. Para ello, un *Router* debe unirse a una red Zigbee antes de poder actuar como tal, retransmitiendo paquetes de otros routers o de *end points*.

End Point o End Device: los dispositivos finales no tienen capacidad de enrutar paquetes, sino que deben interactuar siempre a través de su nodo padre, ya sea este un Coordinador o un *Router*, es decir, no pueden enviar información directamente a otro *end device*. Precisamente por ese motivo, normalmente este tipo de dispositivos van alimentados con baterías, ya que el consumo es menor al no tener que realizar funciones de enrutamiento.

Los módulos XBee son versátiles a la hora de establecer diversas topologías de red. De este modo, Zigbee Soporta tres tipos de topologías de red de las explicadas hasta el momento:

- **Estrella:** ofrece una larga vida útil como consecuencia del bajo consumo que requiere.
- **Malla:** ofrece una alta confiabilidad, al existir múltiples rutas para alcanzar un destino.
- **Árbol:** ofrece los beneficios de las dos anteriores, siendo una topología del tipo Mesh-Star.

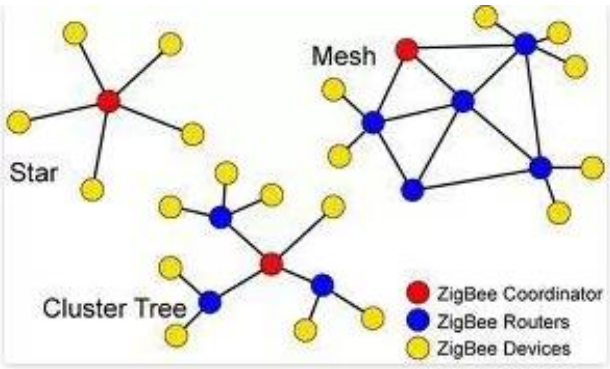


Figura 3.5.3.1: Tipologías de red Zigbee

3.5.4 El módulo Xbee

El protocolo ZigBee fue creado por Zigbee Alliance para facilitar que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas, y se basó en el estándar de comunicaciones para redes inalámbricas IEEE 802.15.4.

Las comunicaciones ZigBee se realizan en la banda libre de 2.4GHz. A diferencia de bluetooth no utiliza FHSS (*Frequency hooping*), sino que realiza las comunicaciones a través de un canal.

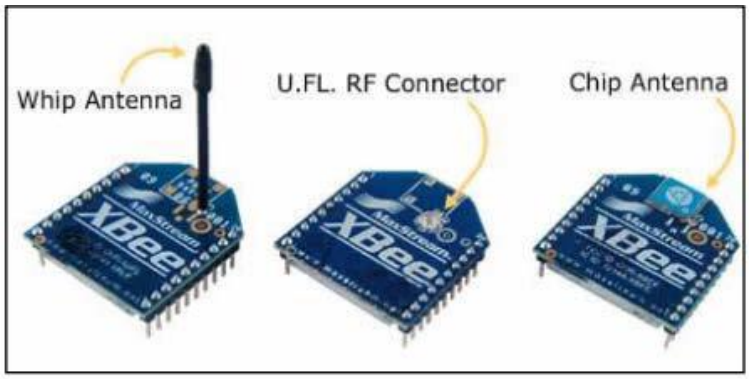


Figura 3.5.4.1: Chips Xbee.

Está siendo especialmente utilizado para redes de sensores en entornos industriales, médicos, agrícolas, domóticos, etc.

Los dispositivos Xbee integran un transmisor-receptor de ZigBee y un procesador en un mismo módulo, lo que permite desarrollar aplicaciones de manera rápida y sencilla. .

3.5.5 Redes inalámbricas inteligentes con XBEE (Integración del IEEE1451)

En este Proyecto, lo que se propone es una red de sensores tipo Xbee, Arduino, y dispositivos de medición inteligentes en topología de estrella.

De lo que se trata es de que un Arduino con XBEE configurado como Router lea toda la información del sensor inteligente, tanto la procedente de las TEDS como la propia señal analógica, y la transmita por ZigBEE a otro Arduino configurado como Coordinador, para que este último decodifique e interprete dicha información.

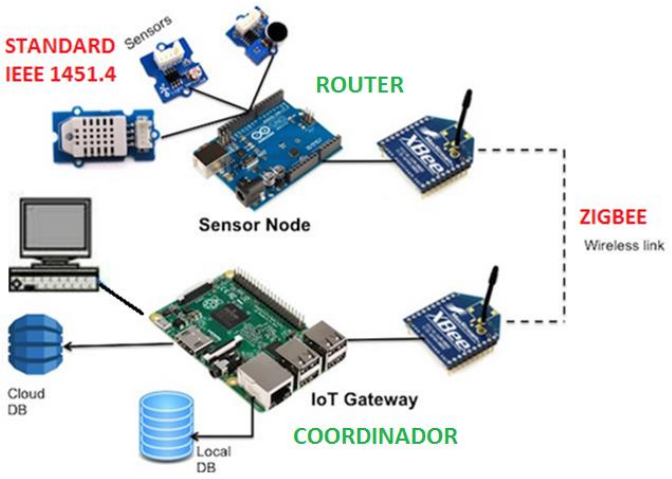


Figura 3.5.5.1: Red Inteligente Xbee

Los módulos Zigbee están diseñados para ser capaces de comunicarse entre ellos sin requerirse que ningún otro equipo intermedio gestione la información en el proceso.

Con la librería de los XBEE, cada Arduino es capaz de analizar en tiempo real los mensajes que se intercambian los módulos Zigbee, leyendo cada uno de los datos que se envían a través de los puertos serie, y es capaz de discriminar si el mensaje afecta a la lógica del programa mediante la utilización de flags de control.

En el siguiente diagrama se detalla el funcionamiento del módulo comentado:

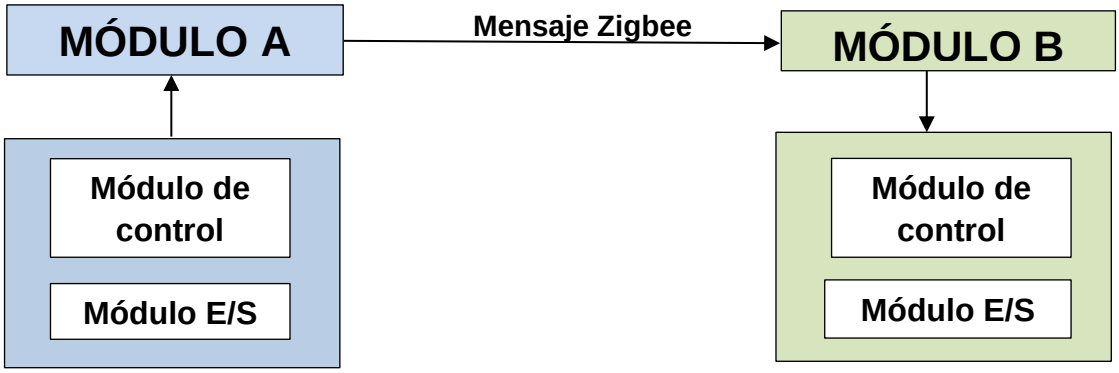


Figura 3.5.5.2: Comunicación Zigbee

Módulo E/S: el Arduino dispone de entradas y salidas para gestionar el control del sistema.

Módulo de control: permite al Arduino programar y configurar la lógica de control del sistema Zigbee.

Intercambio de mensajes Zigbee: cada módulo Zigbee dispone de unos emisores y receptores que permiten el intercambio de mensajes.

El módulo de control del sistema emisor se encarga de codificar los mensajes para enviar al módulo receptor, que los decodifica. Esta codificación se realiza según lo establecido en una serie de mensajes almacenados en una memoria *buffer*, que se van leyendo de acuerdo con un ciclo de reloj del microcontrolador del Arduino.

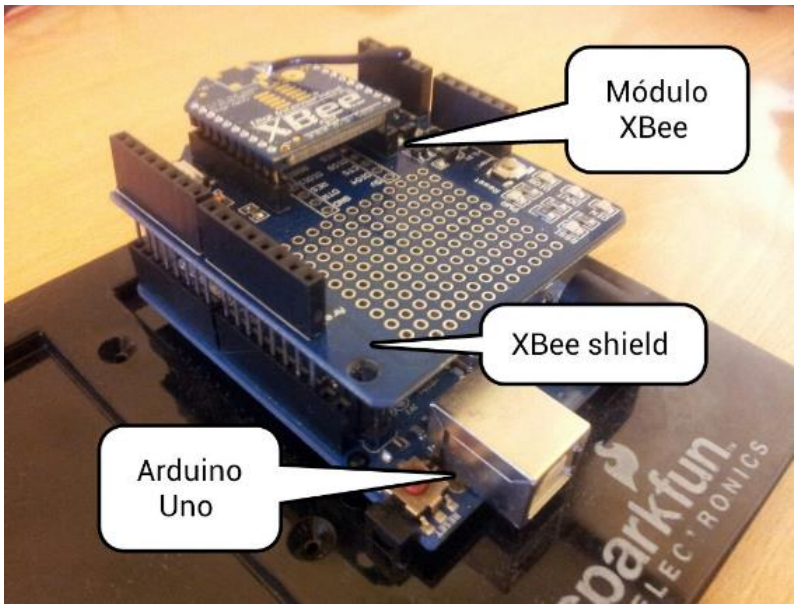


Figura 3.5.5.3: Conjunto Arduino Xbee

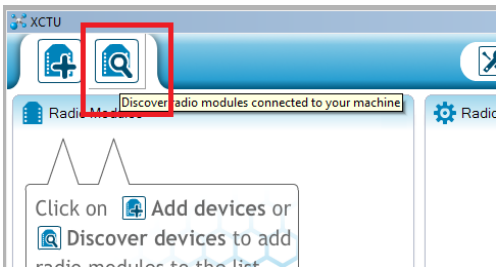
Por tanto, para comunicar dos Xbee de Arduino se configura uno en modo Coordinador y el otro en modo Router:

- Coordinador: es el nodo que recibirá y decodificará los datos
- Router: es el nodo que enviará los datos

Software X-CTU

Para configurar y usar los módulos XBee en Coordinador o Router es necesario descargar e instalar el software multiplatforma *XBee Configuration and Test Utility* (XCTU), que permite interactuar con los módulos mediante un interfaz gráfico e incluye un conjunto de herramientas que hacen muy sencillo configurar y probar los módulos XBee.

Una vez conectado el Arduino e instalado el software, se selecciona la herramienta de Buscar módulo Xbee. A continuación, procede configurar un módulo XBee, por lo que es necesario seleccionar el modo de Configuración y de entre los módulos correspondientes, seleccionar el puerto COM del USB al que está conectado



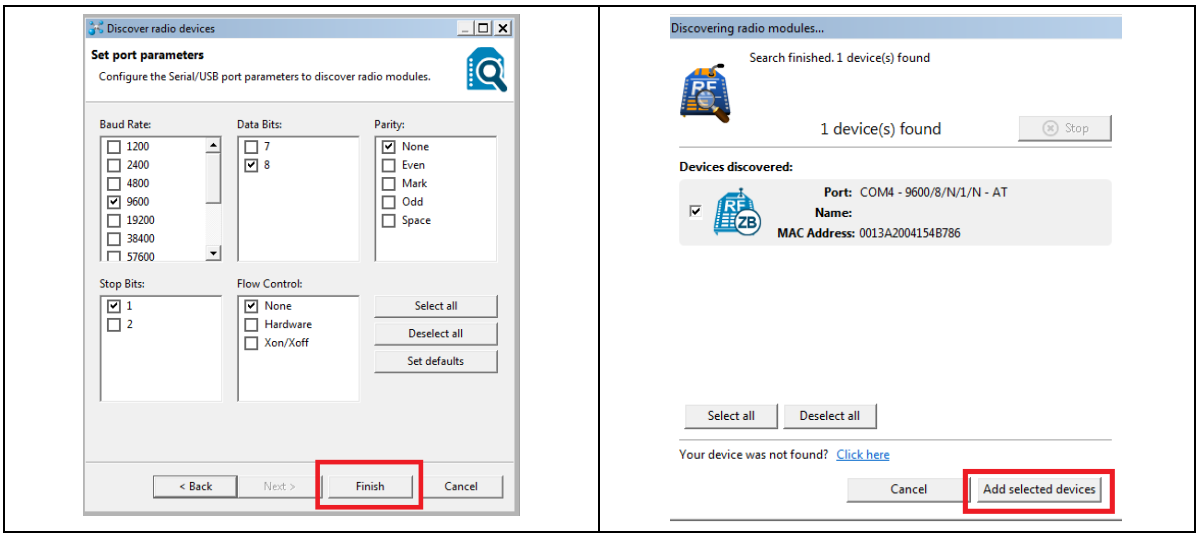


Figura 3.5.5.4: XCTU primeros pasos

Cada módulo de XBee tiene una dirección única de 64 bits, esta dirección se llama MAC y es análogo a la MAC de las tarjetas de red o wifi. El valor de 64 bits está compuesto por los parámetros *Serial Number High* (SH) y *Serial Number Low* (SL), que aparecen impresos en la parte trasera del módulo. El valor SH es generalmente el mismo para todos los módulos XBee (0013A200). La dirección 000000000000FFFF está reservada para mandar un mensaje de *broadcast*.

Figura 3.5.5.5: MAC del XBEE

Una vez conectados en el programa pasamos a modificar los parámetros de los nodos y a configurar el X-CTU

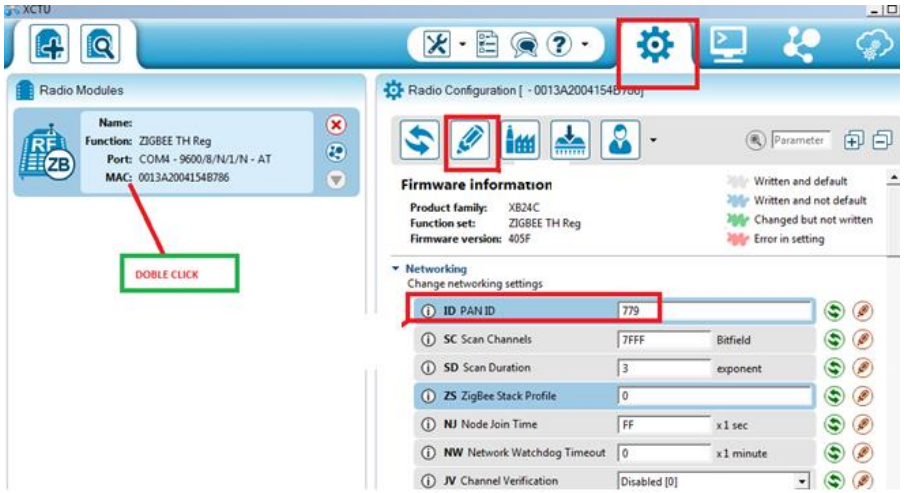


Figura 3.5.5.6: Configuración de los parámetros de los nodos en XCTU

CONFIGURACION DE PARAMETROS DEL XBEE COORDINADOR

Para configurar el nodo COORDINADOR se realiza la siguiente configuración:

- **ID Pan Id:** por ejemplo 779 (se configuran los 2 con la misma ID)

Figura 3.5.5.7: ID - PAN ID

- **CE Coordinator Enable** : Enable [1]

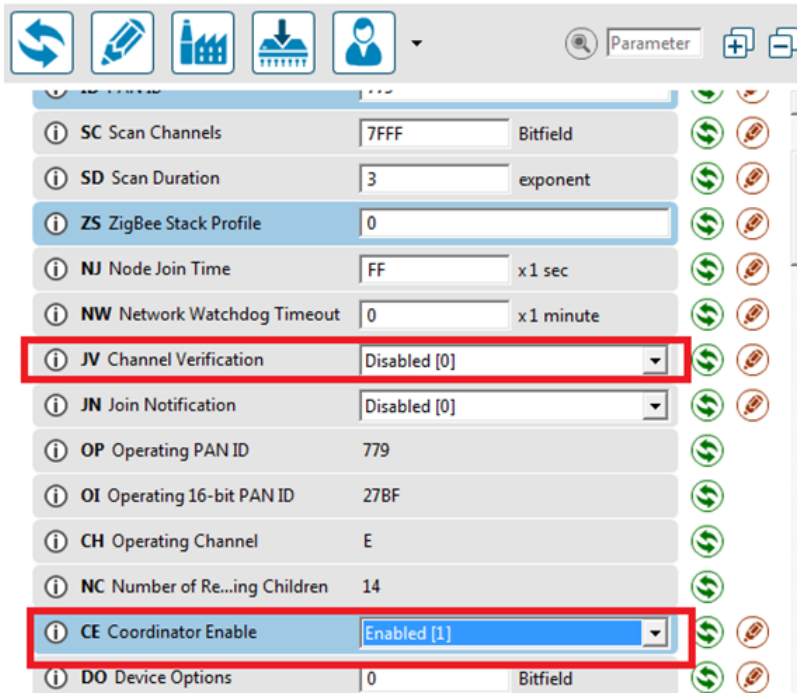


Figura 3.5.5.8: Parámetro CE Coordinator Enable

- **DL Destination Address**: Podemos 0 (cero) porque solo va a ser 1 coordinador y 1 router. Otra opción es poner FFFF (configuración broadcast) o poner los datos del ROUTER 414F9145

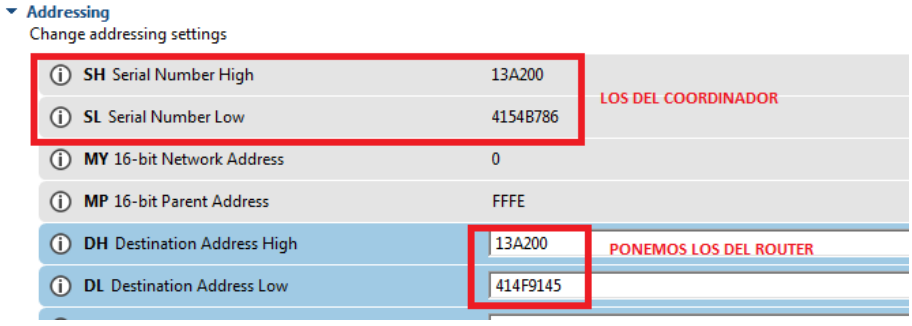


Figura 3.5.5.9: Parámetros Destination Address High y Low

- **DH Destination Address:** Podemos poner 0 (cero) porque solo va a ser 1 coordinador y 1 router

Otra opción es poner FFFF (configuración *broadcast*) o poner los datos del ROUTER 13A200

CONFIGURARCION DE LOS PARAMETROS DEL XBEE ROUTER

Para configurar el nodo ROUTER se realiza la siguiente configuración:

- **ID Pan Id:** al igual que antes, por ejemplo 779 (se configuran los 2 con la misma ID)
- **JV Channel verification:** Enable [1]
- **DL Destination Address:** Podemos 0 (cero) porque solo va a ser 1 coordinador y 1 router. Otra opción es poner FFFF (configuración broadcast o poner los datos del COORDINADOR 4154B786

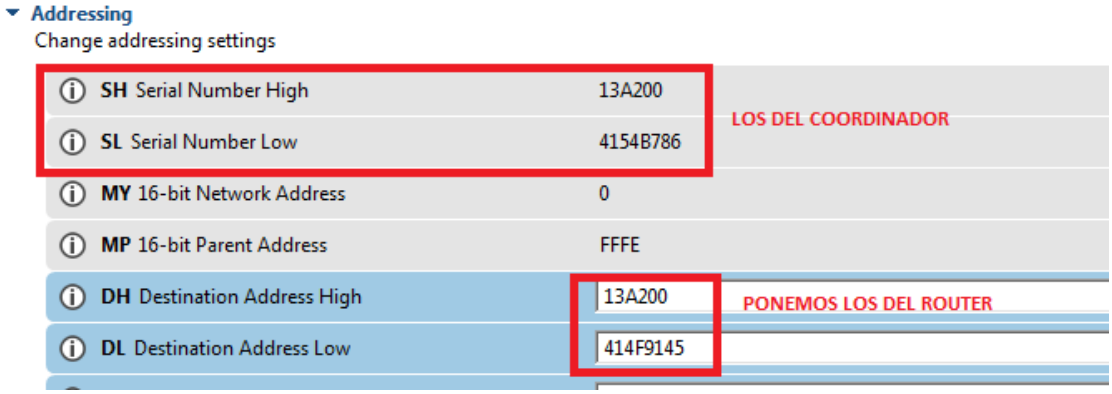


Figura 3.5.5.10: Parámetros Destination Address High y Low

- **DH Destination Address:** Podemos 0 (cero) porque solo va a ser 1 Coordinador y 1 Router
  - Otra opción es poner FFFF (configuración broadcast)
  - Otra opción es Poner los datos del ROUTER 13A200

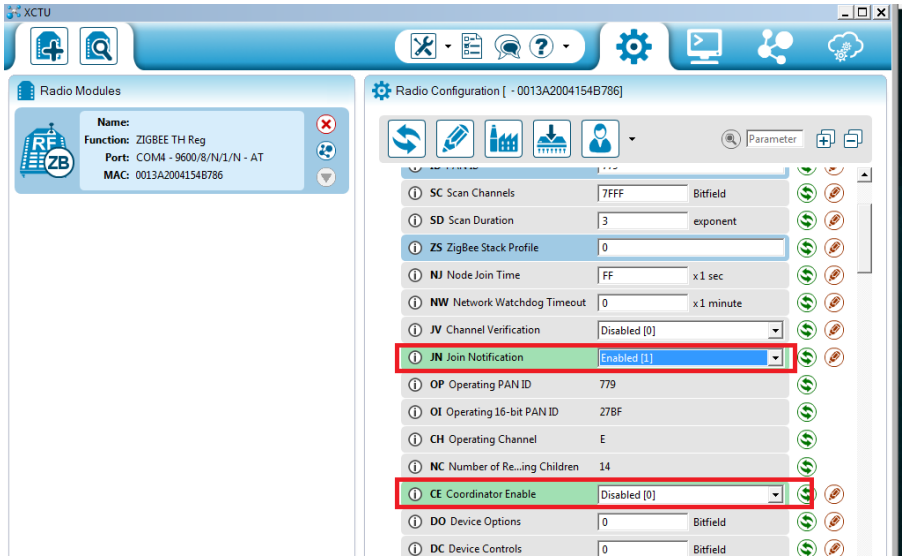


Figura 3.5.5.11: Parámetros Join Notification y Coordinator Enable

Conectamos los 2 XBEE de la siguiente forma y probamos a mandar mensajes usando la consola del software XCTU para ver que funciona todo correctamente

Figura 3.5.5.12: Módulos de Arduino utilizados

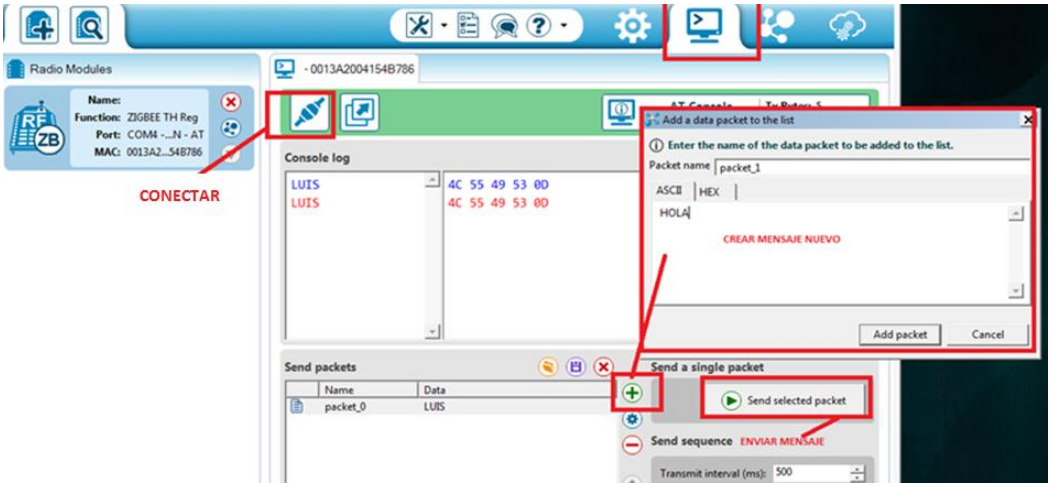


Figura 3.5.5.13: Consola del XCTU

La siguiente prueba consiste en cargar un programa en ambos Arduinos, abrir el monitor serie y probar a mandar mensajes.

1) En la imagen inferior se muestra el interfaz de lectura del puerto COM5 (Arduino Emisor) y COM6 (Arduino receptor). Es este interfaz es posible enviar comandos a través de la línea de comando y utilizar el botón enviar para confirmar el envío. En esta primera imagen se muestra el interfaz de consola COM6 al lado izquierdo y COM5 al lado derecho.

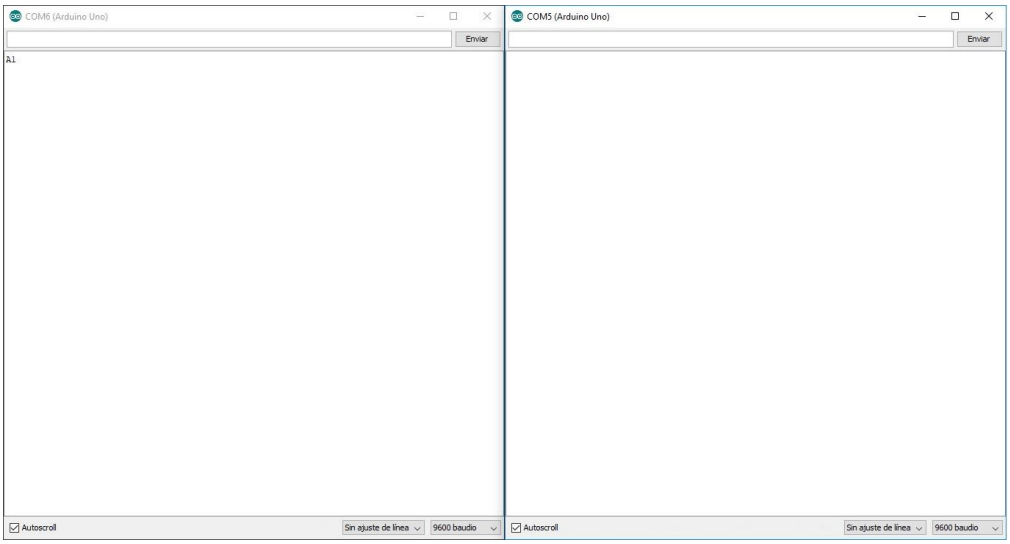


Figura 3.5.5.14: Consola del XCTU

2) A continuación escribimos el texto que queremos enviar al otro arduino en el COM5, en este caso enviamos el comando A2.
3) Finalmente pulsamos el botón “Enviar” y confirmamos el envío del parámetro. En la imagen inferior observamos la recepción del parámetro se ha realizado con éxito.

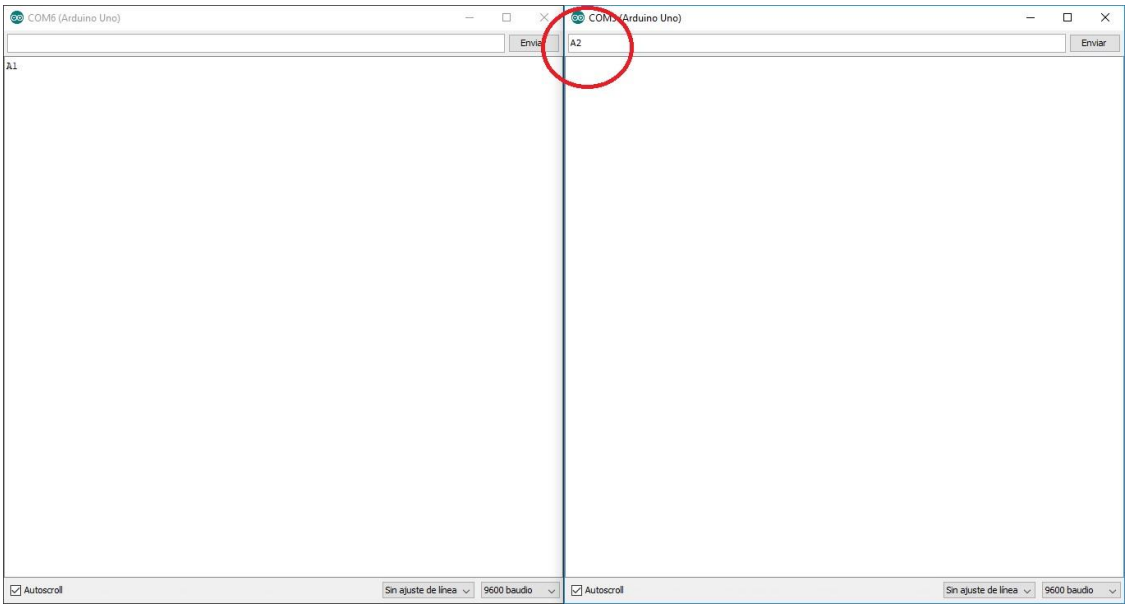


Figura 3.5.5.15: Monitor Serie. Envío de comandos

Para finalizar se realiza el montaje en el que el ROUTER lee la información del sensor inteligente (TEDS + señal analógica), la transmita por ZigBEE a otro Arduino configurado como COORDINADOR y este último decodifica e interpreta la información mostrando el resultado por el monitor serie de Arduino.

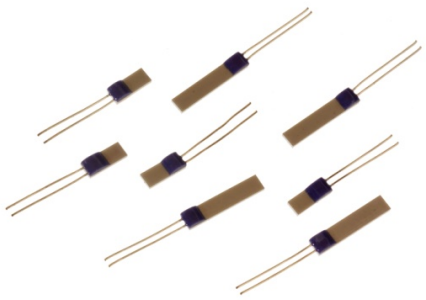
Figura 3.5.5.16: Montaje Final Xbee

3.6 INFORMACIÓN TÉCNICA DE LOS FABRICANTES

Data sheet

Platinum Thin Film Detectors

Platinum Sensing Resistors – Thin Film (Pt100 & Pt1000 Ohm)



Pt100 Elements, Thin Film (100 Ohm)

- Pt100 elements to IEC751 Class A, B and 1/3DIN
- For use from –50°C to +500°C
- Thin film construction
- Suitable for surface & immersion applications where protected
- Vibration resistant

Specifications:

Sensor type:	Pt100 (100 Ohms @ 0°C)
Construction:	Thin film, 10mm tails
Temperature range:	-50°C to +500°C
Ice point resistance:	100Ω
Fundamental interval (0°C to 100°C):	38.5Ω (nominal)
Self heating:	<0.5°C/mW
Thermal response:	0.1s
Stability:	±0.05%

Resistance	**Dimensions** (width x length)	**Tolerance Class**	**RS order code**
Pt100	2 x 5.0mm	Class A	611-7788
Pt100	2 x 5.0mm	Class B	611-7801
Pt100	2 x 5.0mm	Class B	290-5070 (Packet of 5)
Pt100	2 x 10mm	Class A	362-9799
Pt100	2 x 10mm	Class B	237-1607
Pt100	2.0 x 10mm	1/3DIN	362-9812
Pt100	2.0 x 2.3mm	Class A	362-9834
Pt100	2.0 x 2.3mm	Class B	362-9840
Pt100	2.0 x 2.3mm	1/3DIN	362-9856

Data Ref: RS025/0812

Data sheet

Platinum Thin Film Detectors

Platinum Sensing Resistors – Thin Film (Pt100 & Pt1000 Ohm)

Pt100 Elements (continued)

Resistance	**Dimensions** (width x length)	**Tolerance Class**	**RS order code**
Pt100	1.2 x 1.6mm	Class A	666-7362
Pt100	1.2 x 1.6mm	Class B	666-7353
Pt100	1.0 x 3.0mm	Class A	666-7359
Pt100	1.0 x 3.0mm	Class B	666-7356

Pt1000 Elements, Thin Film (1000 Ohm)

- Pt1000 elements to IEC 751 Class A and B
- For use from –50°C to +500°C
- Thin film construction
- Suitable for surface & immersion applications where protected
- Vibration resistant

Specifications:

Sensor type:	Pt1000 (1000 Ohms @0°C)
Construction:	Thin film, 10mm tails
Temperature range:	-50°C to +500°C
Ice point resistance:	1000Ω
Fundamental interval (0°C to 100°C):	385Ω (nominal)
Self heating:	<0.5oC/mW
Thermal response:	0.1s
Stability:	±0.05%

Resistance	**Dimensions** (width x length)	**Tolerance Class**	**RS order code**
Pt1000	2.0 x 10mm	Class A	362-9907
Pt1000	2.0 x 10mm	Class B	362-9913

NTC THERMISTOR OF MF52-TYPE SERIES SPECIFICATION

*** Outline：**

The MF52 thermistor is a small-sized,epoxy-resin coated NTC resistor made from new-type material with new craftsmanship.It is featured with advantages including high precision and quick reaction

*** Application：**

Air conditioners, heating facilitied,electronic thermometers ,fluid level sensors,automobile electronics and electronic table-calendars.

*** Features：**

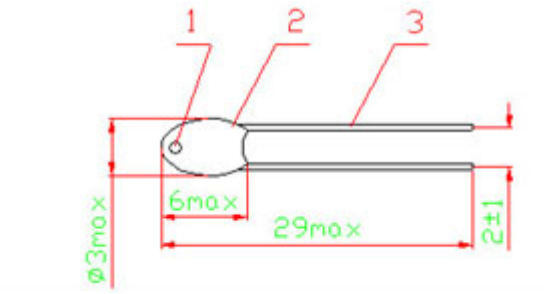
1.High testing precision;
2.Small and quick in reaction;
3.Long and good service;
4.Good interconvertibility and consistency.

*** Part NO.：**

MF52	E	103	H	L	347
①	②	③	④	⑤	⑥

① Drop-like NTC thermistor
② E：Epoxy-resin coated package S：Silicone coated package
③ R25: 10KΩ-103
④ Tolerance: F：±1% G：±2% H：±30% J：±5% K：±10%
⑤ L：B25/50 H：B25/85 T：Special
⑥ B-value：347：3470 338：3380 we adopted the former three digits

*** Dimensions(mm) :**



*** Specification**

Model	R25	B value	Dissipation	Time Constant	Temperature Range
MF52	100Ω-10KΩ	3100K			
MF52	200Ω-10KΩ	3270K			
MF52	500Ω-15KΩ	3470K			
MF52	1KΩ-50KΩ	3600K	≥2.5mW/℃	≤7S	-40℃~+120℃
MF52	5KΩ-50KΩ	3950K	in static air	in static air	
MF52	10KΩ-100KΩ	4050K			
MF52	10KΩ-100KΩ	4150K			
MF52	20KΩ-500KΩ	4300K			

Remarks:

1) Tolerance of the resistance: F : ±1% G : ±2% H : ±3% J : ±5% K : ±10% .
2) The Tolerance of the B-value is ±1% in response with a rated resistance for which the precision is ±1%, The tolerance of B-value is±2% under other circumstances.
3) Products with specifications unmentioned in the table above are available upon customers' request.

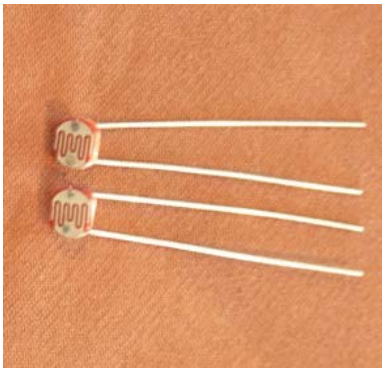
*** Cautions :**

1) The two ends of the lead is not supposed to be loaded with excess pulling stress,owing to the small size and small welding spot of MF52-srs products.
2) Soldering is supposed to be done 5mm away from the root of the lead,and only for a brief moment.
3) Thermistor of MF52-srs are not supposed to be exposed directly in water while working.

Normal specification Resistance & Temperature Table of MF52-type (Unit：KΩ)

R25 / B / T(℃) / Rt	10 KΩ	50 KΩ	100 KΩ	50 KΩ	50 KΩ	100 KΩ	100 KΩ	150 KΩ
	3950	3950	4000	4050	4150	4150	4300	4500
-30	181.70	908.30	1790.00					
-25	133.30	666.50	1321.00					
-20	98.88	494.50	984.70					
-15	74.10	370.50	740.80					
-10	56.06	280.30	562.30					
-5	42.80	214.00	430.50					
0	98.96	164.80	332.30	168.80	172.00	344.10	352.40	576.70
5	25.58	127.90	257.50	131.30	132.20	264.30	270.00	433.20
10	20.00	99.98	201.10	101.00	102.40	204.80	208.30	328.40
15	15.76	78.79	158.20	79.28	80.03	160.10	161.90	250.90
20	12.51	62.55	125.40	62.78	63.00	125.00	136.70	193.30
25	10.00	50.00	100.00	50.00	50.00	100.00	100.00	150.00
30	8.048	40.24	80.29	39.98	39.76	79.51	78.35	117.30
35	6.518	32.59	64.87	32.16	31.89	63.77	62.37	92.28
40	5.312	26.56	57.72	26.10	25.73	51.45	49.94	73.11
45	4.354	21.77	43.10	21.35	20.88	41.76	40.22	58.28
50	3.588	17.94	35.42	17.72	17.04	34.08	32.56	46.74
55	2.974	14.87	29.26	14.36	13.99	27.97	26.40	37.71
60	2.476	12.38	24.30	11.92	11.53	23.06	21.53	30.58
65	2.072	10.36	20.27	9.938	9.541	19.08	17.69	24.94
70	1.743	8.717	16.99	8.317	7.929	15.86	14.62	20.45
75	1.473	7.364	14.31	6.991	6.621	13.24	12.20	16.85
80	1.250	6.248	12.10	5.906	5.552	11.10	10.05	13.94
85	1.065	5.324	10.27	5.012	4.674	9.348	8.376	11.60
90	0.911	4.555	8.758	4.271	3.950	7.900	7.004	9.680
95	0.7824	3.912	7.495	3.654	3.349	6.698	5.894	8.118
100	0.6744	3.372	6.438	3.316	2.849	5.698	4.978	6.836
105	0.5836	2.918	5.550	2.701	2.438	4.875	4.215	5.780
110	0.5066	2.533	4.801	2.336	2.093	4.186	3.580	4.904

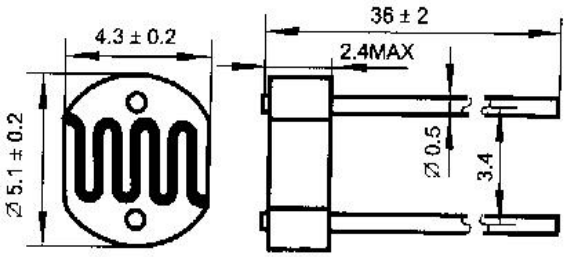
CdS PHOTOCONDUCTIVE CELLS GL5528



- ▲ Epoxy encapsulated
- ▲ Quick response
- ▲ Small size
- ▲ High sensitivity
- ▲ Reliable performance
- ▲ Good characteristic of spectrum

Light Resistance at 10Lux (at 25℃)	8～20KΩ
Dark Resistance at 0 Lux	1.0MΩ(min)
Gamma value at 100-10Lux	0.7
Power Dissipation(at 25℃)	100mW
Max Voltage (at 25℃)	150V
Spectral Response peak (at 25℃)	540nm
Ambient Temperature Range:	- 30～+70℃

Outline



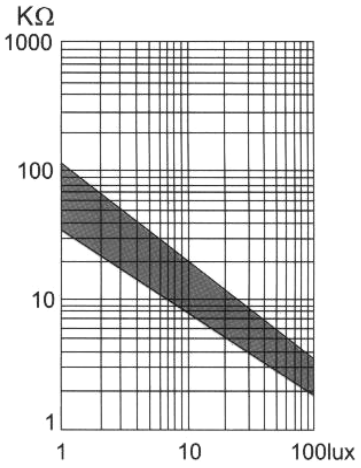
Measuring Conditions

1. Light Resistance:
   measured at 10 lux with standard light A (2854k color temperature) and 2h pre-illumination at 400-600 lux prior to testing.
2. Dark Resistance:
   measured 10 seconds after pulsed 10 lux.
3. Gamma Characteristic:
   between 10 lux and 100 lux and given by

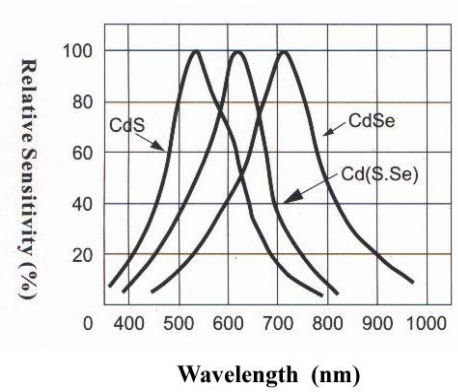
   $$T = \frac{\log (R10/R100)}{\log (100/10)} = \log (R10/R100)$$

   R10, R100 cell resistance at 10 lux and 100 lux.
   The error of T is +0.1.
4. Pmax:
   Max. power dissipation at ambient temperature of 25°C.
5. Vmax:
   Max. voltage in darkness that may be applied to the cell continuously.

Illuminance Vs. Photo Resistance



Spectral Response



LIDA OPTICAL&ELECTRONIC CO., LTD.

254,Zhong zhou road, Nanyang, Henan,P.R.C
TEL: +86-377-6313 0034
FAX: +86-377-6315 2372
E-mail: sale@nylida.com
Http://www.nylida.com

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

Data Sheet **AD623**

FEATURES

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

APPLICATIONS

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

GENERAL DESCRIPTION

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

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

Table 1. Low Power Upgrades for the AD623

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

FUNCTIONAL BLOCK DIAGRAM

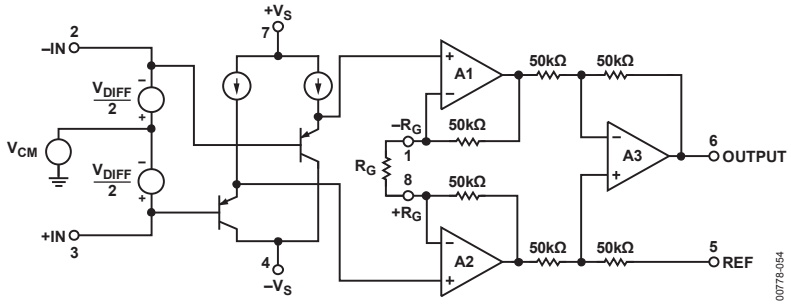


Figure 1.

Rev. E
Document Feedback

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
Technical Support www.analog.com

µA741 General-Purpose Operational Amplifiers

1 Features

- Short-Circuit Protection
- Offset-Voltage Null Capability
- Large Common-Mode and Differential Voltage Ranges
- No Frequency Compensation Required
- No Latch-Up

2 Applications

- DVD Recorders and Players
- Pro Audio Mixers

3 Description

The µA741 device is a general-purpose operational amplifier featuring offset-voltage null capability.

The high common-mode input voltage range and the absence of latch-up make the amplifier ideal for voltage-follower applications. The device is short-circuit protected and the internal frequency compensation ensures stability without external components. A low-value potentiometer may be connected between the offset null inputs to null out the offset voltage as shown in Figure 12.

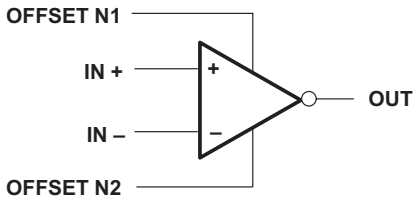
The µA741C device is characterized for operation from 0°C to 70°C.

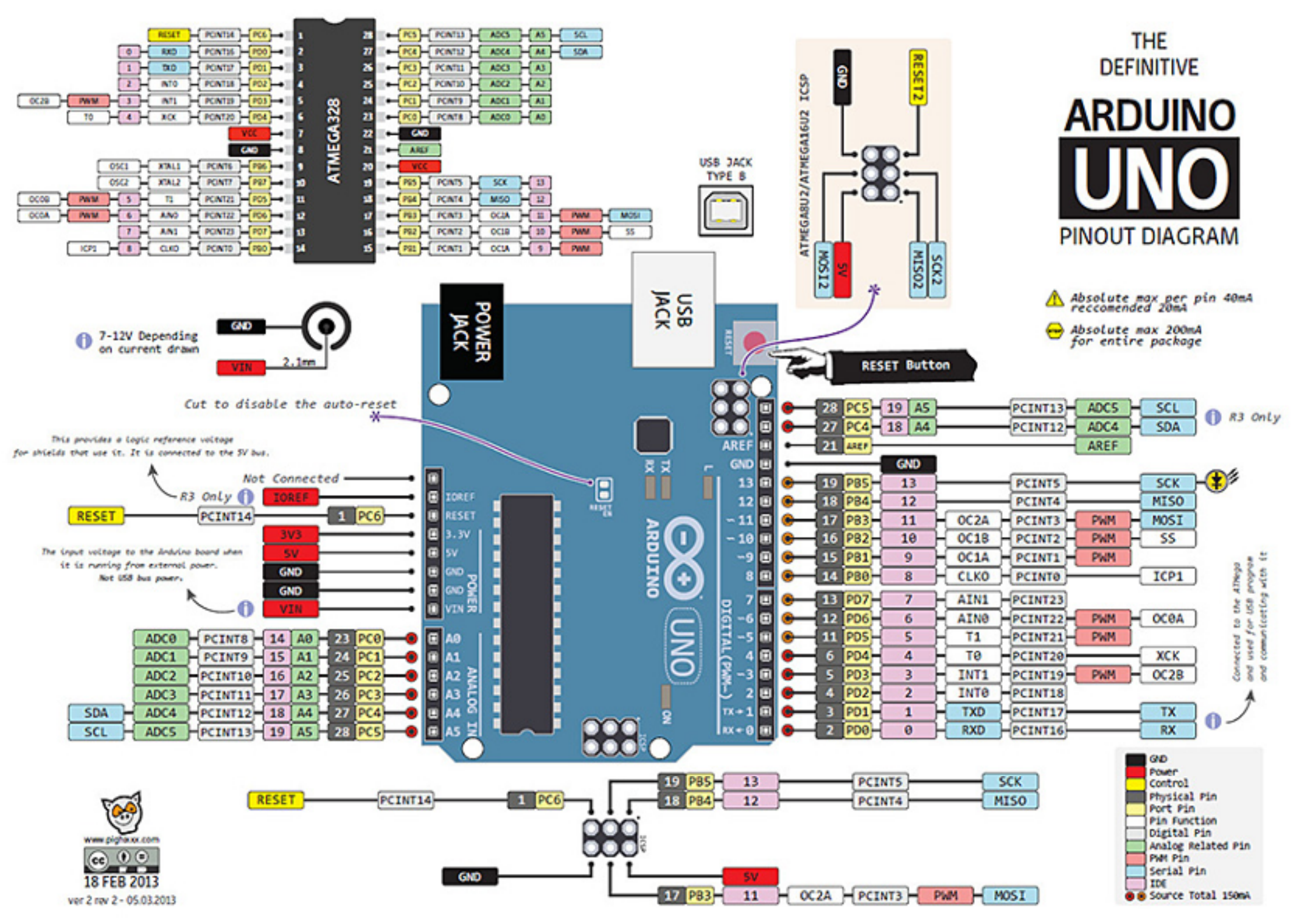
Device Information(1)

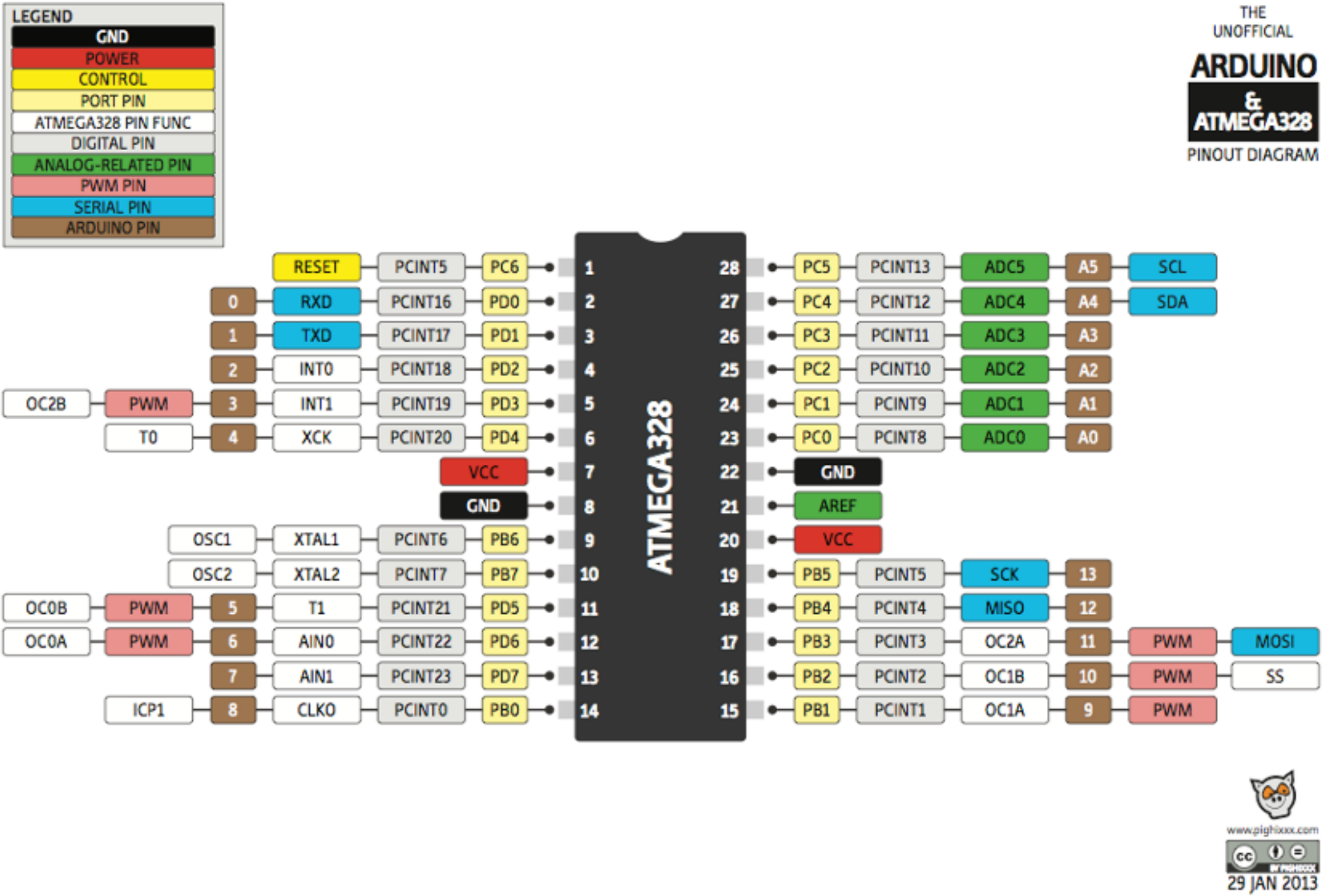
PART NUMBER	PACKAGE	BODY SIZE (NOM)
µA741CD	SOIC (8)	4.90 mm × 3.91 mm
µA741CP	PDIP (8)	9.81 mm × 6.35 mm
µA741CPS	SO (8)	6.20 mm × 5.30 mm

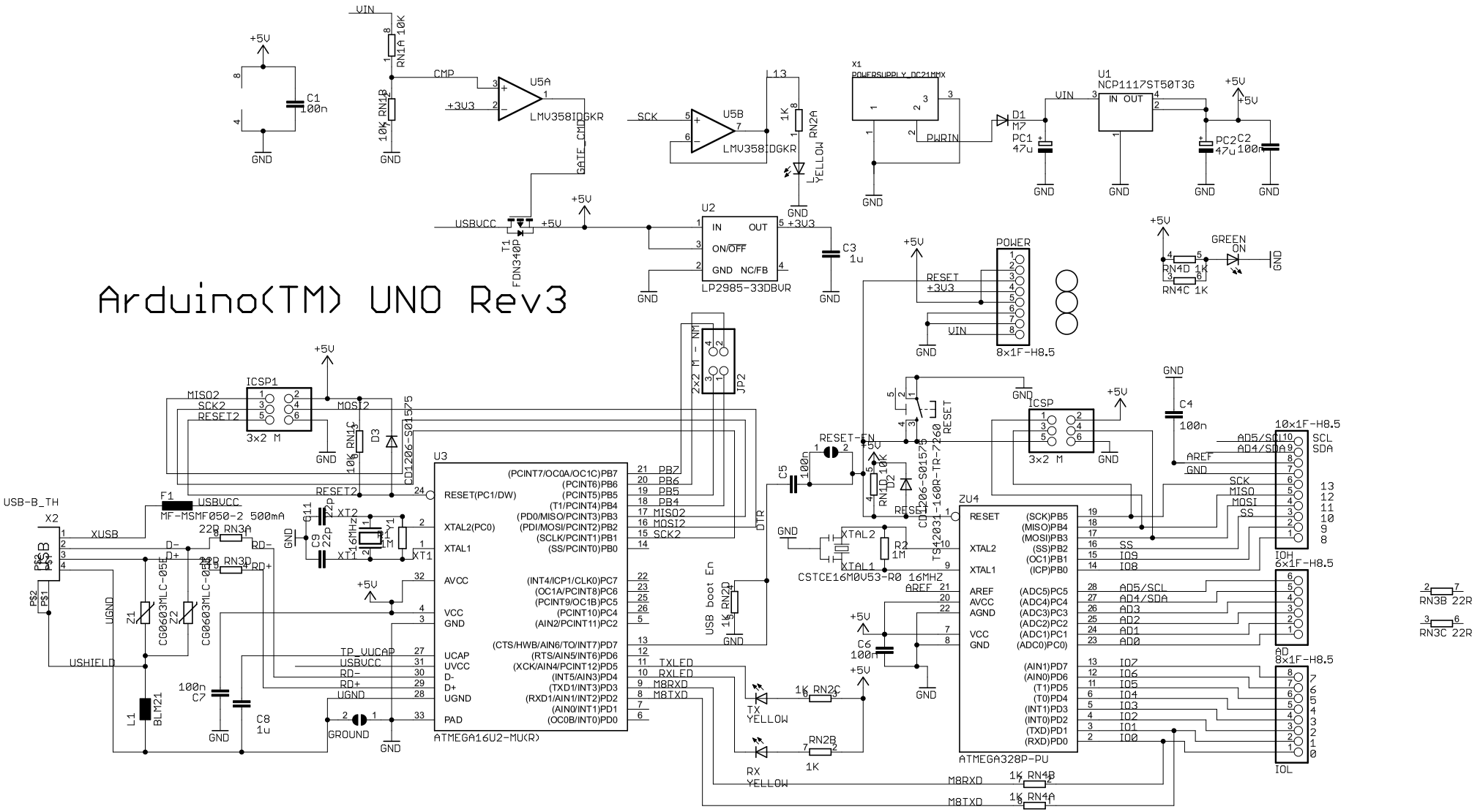
(1) For all available packages, see the orderable addendum at the end of the data sheet.

Simplified Schematic

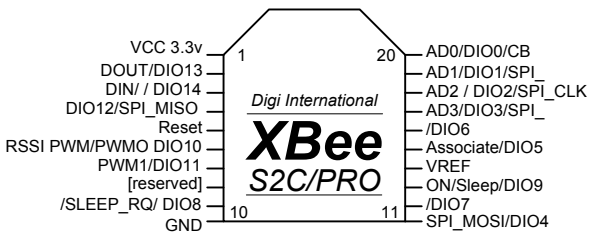



THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
ATMEGA8U2/ATMEGA16U2 ICSP
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
GND
VIN
2.1mm
Cut to disable the auto-reset
This provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Connected to the ATmega and used for USB program and communicating with it
www.pighixx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

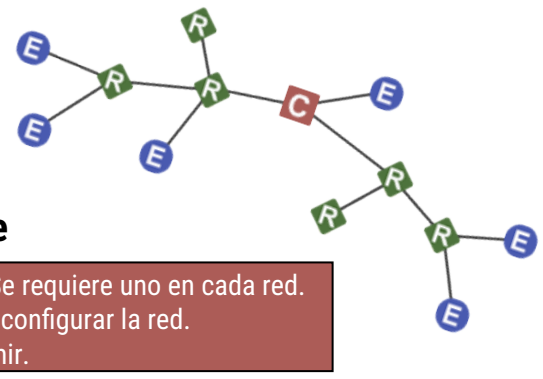

LEGEND
GND
POWER
CONTROL
PORT PIN
ATMEGA328 PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
ARDUINO PIN
THE UNOFFICIAL
ARDUINO & ATMEGA328
PINOUT DIAGRAM
ATMEGA328
RESET PCINT5 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 7
GND 8
OSC1 XTAL1 PCINT6 PB6 9
OSC2 XTAL2 PCINT7 PB7 10
OC0B PWM 5 T1 PCINT21 PD5 11
OC0A PWM 6 AIN0 PCINT22 PD6 12
7 AIN1 PCINT23 PD7 13
ICP1 8 CLKO PCINT0 PB0 14
28 PC5 PCINT13 ADC5 A5 SCL
27 PC4 PCINT12 ADC4 A4 SDA
26 PC3 PCINT11 ADC3 A3
25 PC2 PCINT10 ADC2 A2
24 PC1 PCINT9 ADC1 A1
23 PC0 PCINT8 ADC0 A0
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 SCK 13
18 PB4 PCINT4 MISO 12
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM
www.pighixxx.com
29 JAN 2013



GUÍA DE REFERENCIA PARA **XBee S2C PRO**



XBee es un microcontrolador fabricado por digi, el cual utiliza el protocolo Zigbee. XBee utiliza 3.3V y tienen un espaciado entre pines menor que el utilizado en un protoboard. Dado lo anterior es recomendable utilizar una tarjeta o kit para que sea más fácil su uso.



Roles XBee

Coordinador: Se requiere uno en cada red.
Se encarga de configurar la red.
No puede dormir.

Router: Pueden existir multiples en una red. Pueden redirigir los mensajes a otros routers o End Devices. No pueden dormir.

End Device: Pueden existir muchos, no pueden redirigir mensajes.
Pueden dormir para ahorrar energía.

S2C

Especificaciones			
	Voltaje de operación: 2.1 – 3.6V Corriente de operación: 33mA@3.3V Rango en interiores: 60 metros Rango en línea vista: 1200 metros Max valor lectura analógica: 1.2V	Pines I/O: 13 Entradas analógas: 4 *Configurable como red mesh* *Red auto reparable*	Firmware: Zigbee, Digimesh, 802.15.4 Velocidad de transmisión RF: 250kbps Frecuencia: 2.4GHz - 2.5GHz Temperatura operación: -40 a 85°C

S2C PRO

Especificaciones			
	Voltaje de operación: 2.7 – 3.6V Corriente de operación: 31mA@3.3V Rango en interiores: 90 metros Rango en línea vista: 3200 metros Max valor lectura analógica: 1.2V	Pines I/O: 13 Entradas analógas: 4 *Configurable como red mesh* *Red auto reparable*	Firmware: Zigbee, Digimesh, 802.15.4 Velocidad de transmisión RF: 250kbps Frecuencia: 2.4GHz - 2.5GHz Temperatura operación: -40 a 85°C

Modos XBee

Transparente: Los dispositivos actúan como un reemplazo de cable serial. Cuando los datos RF son recibido, el dispositivo envía los datos a través del puerto serie. Utilice la interfaz de modo de comando AT para configurar los parámetros del dispositivo.
Comando: Basada en tramas, amplía el nivel en que una aplicación host puede interactuar con las capacidades de red del dispositivo. Cuando esta en modo API, el dispositivo contiene todos los datos que entran y salen en marcos que definen operaciones o eventos dentro del dispositivo.

Setup XBee

Conecta el Xbee a un adaptador TTL a Serial como un FTDI
Utiliza el software gratuito X-CTU para configurar el módulo XBee
Baud:9600 – FC: Hardware – Data Bits 8 – Parity: None – Stop Bits: 1

Ajustes Básicos

PAN ID: es la red a la cual se conectará el módulo. Si es 0, el XBee se asociará a cualquiera que esté disponible
DH/DL: Es la dirección del módulo de destino. Se utiliza para enviar información a un XBee en específico. Si se configura en 0 enviará datos solo al coordinador. Si se configura en 0x0000000000FFFF hará un broadcast (envío a todos los módulos de la red)

Ajustes Pin

Para poder trabajar con los pines como entradas/salidas en un XBee, debe estar configurado en modo API.
D0 - Configura el pin en 0 para comenzar a leer datos
IR - realiza una lectura del pin cada XX milisegundos

Trama API para envío de comandos AT remotos

Byte	Ejemplo	Descripción
0	0x7E	Byte de inicio – indica el comienzo del paquete de datos (frame)
1	0x00	Largo – Número de bytes (ChecksumByte# – 1 – 2)
2	0x10	
3	0x17	Tipo de mensaje - 0x17 significa que es solicitud de comando AT
4	0x01	Frame ID – secuencia del paquete
5	0x00	Dirección de destino de 64-bi t(número de serie)
6	0x13	MSB es el byte 5, LSB es el byte 12
7	0xA2	
8	0x00	0x0000000000000000 = Coordinador
9	0x40	0x000000000000FFFF = Broadcast
10	0x8B	
11	0x78	
12	0x4E	
13	0xFF	Dirección de la red de destino
14	0xFE	(configúralo como 0xFFFE para enviar un bodcast)
15	0x02	Opción del comando remoto (configuralo como 0x02 para aplicar los cambios)
16	0x44 (D)	Nombre del comando AT (Dos caracteres ASCII)
17	0x34 (2)	
18	0x05	Parámetro del comando
19	0x25	Checksum

Conexión con Arduino:
Arduino TX se conecta la RX de XBee (Data IN)
Arduino RX se conecta al TX de XBee (Data Out)

Integración con Arduino:
Los datos enviados utilizando Serial.print() saldrán por el puerto TX del Arduino, que estará conectado al RX del módulo XBee. Si XBee está en modo AT, se transmitirá inalámbricamente hacia el destino. Los datos recibidos en el módulo XBee serán enviados al puerto serial.

Ejemplo para Arduino: Lectura de un valor análogo utilizando modo API

```
// XBee remoto: AT, XBee base: API
if (Serial.available() >= 21) { // Nos aseguramos que ha llegado el mensaje completo
if (Serial.read() == 0x7E) { // 7E es el byte de inicio
  for (int i = 1; i<19; i++) { // descartamos los bytes hasta llegar los datos análogos
    byte discardByte = Serial.read();
}
int analogMSB = Serial.read(); // Lee el primer byte del dato análogo
analogLSB = Serial.read(); // Lee el segundo byte del dato análogo
int analogReading = analogLSB + (analogMSB * 256); }
 }
}
```

Ejemplo Arduino: Cambiar la configuración de un pin en un XBee remoto

```
// XBee remoto: AT, XBee base: API
Serial.write(0x7E); // byte de inicio
Serial.write((byte)0x0); // Largo MSB (siempre 0)
Serial.write(0x10); // Largo LSB
Serial.write(0x17); // 0x17 es el tipo de mensaje para enviar comandos AT
Serial.write((byte)0x0); // Frame ID (no solicitamos repuesta)
Serial.write((byte)00); // Envía los 64 bit de la dirección de destino
Serial.write((byte)00); // (Enviando 0x000000000000FFFF (broadcast))
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write(0xFF);
Serial.write(0xFF);
Serial.write(0xFF); // Red de destino
Serial.write(0xFE); // (enviar 0xFFFE si es desconocida)
Serial.write(0x02); // configurar 0x02 para aplicar los cambios
Serial.write('D'); // Comando AT : D1
Serial.write('1');
Serial.write(0x05); // Configura D1 para ser 5  (Digital Out HIGH)
long chexsum = 0x17 + 0xFF + 0xFF +0xFF + 0xFE + 0x02 + 'D' + '1' + 0x05;
Serial.write( 0xFF – (chexsum & 0xFF)); // Checksum
```

Trama API para recepción de datos I/O

Byte	Ejemplo	Descripción
0	0x7E	Byte de inicio - indica el comienzo del paquete de datos (frame)
1	0x00	
2	0x14	
3	0x92	Tipo de frame 0x92 indica que es un muestreo de las entradas del XBee
4	0x00	Dirección de origen de 64-bit (número de serie)
5	0x13	MSB es el byte 4, LSB es el byte 11
6	0xA2	
7	0x00	
8	0x40	
9	0x8B	
10	0x78	
11	0x4E	
12	0xA4	Dirección de 16-bit de la red de origen
13	0x02	
14	0x01	Opciones de recepción: 01 = Packet acknowledged 02 = Broadcast packet
15	0x01	Número de muestras. Siempre debe ser 1 dadas las limitaciones de XBee
16	0x00	Máscara para el canal digital, indica que pines estan configurados como DIO
17	0x30	
18	0x01	Máscara para el canal análogo, indica cuales pines están configurados como ADC
19	0x00	Lectura de los canales digitales. Estos dos bytes contienen los estados de
20	0x20	los pines configurados como DIO
21	0x02	Lectura del canal análogo.
22	0x0C	Cada canal entrega 2 bytes con el resultado de la lectura del ADC
23	0x20	Checksum (0xFF - la suma de todos los bytes desde el byte 3 a hasta el último)

Modo Sleep

End Device puede dormir para ahorrar energía. Un End Device que solo despierta cada 5 minutos para enviar datos puede solo estar despierto por 6 segundos en un día.

SM – 4 = Cyclic Sleep
SP - Sleep time (hasta 28 segundos)
SN - Número de ciclos sleep
ST – Tiempo que permanecerá despierto

Pin I/O Opciones

0 –Disabled
1 –N/A
2 –ADC
3 –Digital IN
4 –Digital OUT, LOW
5 –Digital OUT, HIGH

Máscara para canal digital

Primer Byte
n/a n/a n/a D12 D11 D10 n/a n/a
Segundo Byte
D7 D6 D5 D4 D3 D2 D1 D0
Ejemplo:
0x00 0x13 = 0000 0000 0000 1101
Pins D3, D2 y D0

Máscara para canal análoogo

(volt) n/a n/a n/a A3 A2 A1 A0
Ejemplo:
0x05 = 0000 0101 = Pin A2 and A0

DS2431

1024-Bit, 1-Wire EEPROM

General Description

The DS2431 is a 1024-bit, 1-Wire® EEPROM chip organized as four memory pages of 256 bits each. Data is written to an 8-byte scratchpad, verified, and then copied to the EEPROM memory. As a special feature, the four memory pages can individually be write protected or put in EPROM-emulation mode, where bits can only be changed from a 1 to a 0 state. The DS2431 communicates over the single-conductor 1-Wire bus. The communication follows the standard 1-Wire protocol. Each device has its own unalterable and unique 64-bit ROM registration number that is factory lasered into the chip. The registration number is used to address the device in a multidrop, 1-Wire net environment.

Applications

Accessory/PCB Identification

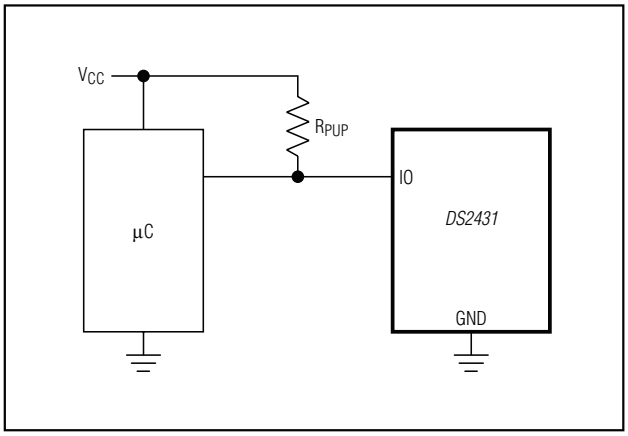
Medical Sensor Calibration Data Storage

Analog Sensor Calibration Including IEEE P1451.4 Smart Sensors

Ink and Toner Print Cartridge Identification

After-Market Management of Consumables

Typical Operating Circuit



Pin Configurations appear at end of data sheet.

Features

- **1024 Bits of EEPROM Memory Partitioned Into Four Pages of 256 Bits**
- **Individual Memory Pages Can Be Permanently Write Protected or Put in EPROM-Emulation Mode ("Write to 0")**
- **Switchpoint Hysteresis and Filtering to Optimize Performance in the Presence of Noise**
- **IEC 1000-4-2 Level 4 ESD Protection (±8kV Contact, ±15kV Air, Typical)**
- **Reads and Writes Over a Wide Voltage Range from 2.8V to 5.25V from -40°C to +85°C**
- **Communicates to Host with a Single Digital Signal at 15.4kbps or 125kbps Using 1-Wire Protocol**
- **Also Available as Automotive Version Meeting AEC-Q100 Grade 1 Qualification Requirements (DS2431-A1; Refer to the IC Data Sheet for Details)**

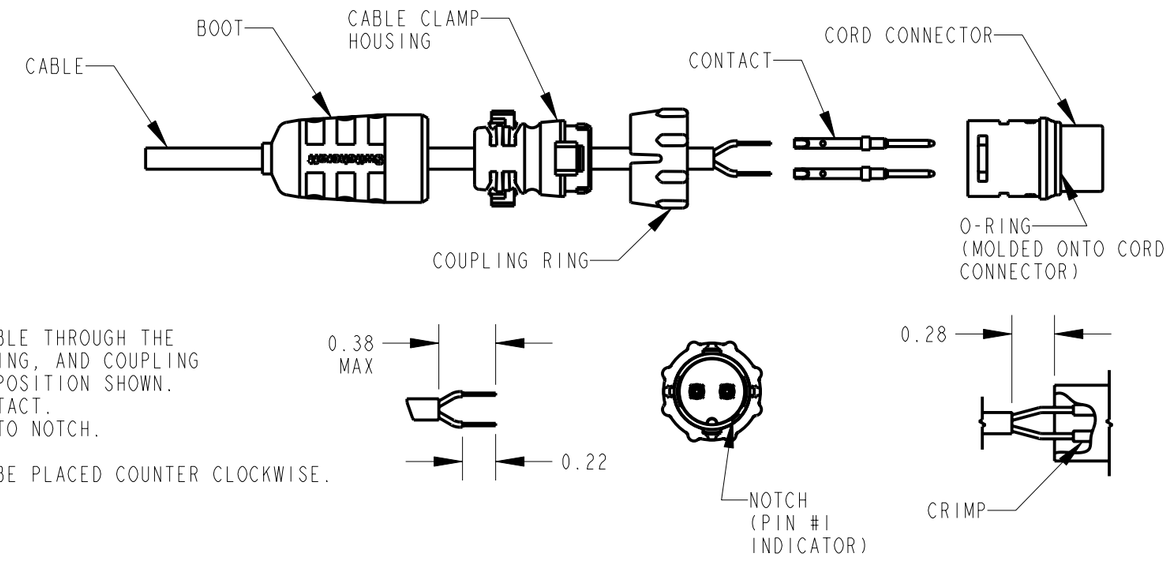
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
DS2431+	-40°C to +85°C	3 TO-92
DS2431+T&R	-40°C to +85°C	3 TO-92
DS2431P+	-40°C to +85°C	6 TSOC
DS2431P+T&R	-40°C to +85°C	6 TSOC
DS2431G+U	-40°C to +85°C	2 SFN (6mm x 6mm)
DS2431G+T&R	-40°C to +85°C	2 SFN (6mm x 6mm) (2.5k pcs)
DS2431GA+U	-40°C to +85°C	2 SFN (3.5mm x 6.5mm)
DS2431GA+T&R	-40°C to +85°C	2 SFN (3.5mm x 6.5mm) (2.5k pcs)
DS2431Q+T&R	-40°C to +85°C	6 TDFN-EP* (2.5k pcs)
DS2431X-S+	-40°C to +85°C	3x3 UCSPR (2.5k pcs)
DS2431X+	-40°C to +85°C	3x3 UCSPR (10k pcs)

Note: *The leads of TO-92 packages on tape and reel are formed to approximately 100-mil (2.54mm) spacing. For details, refer to the package outline drawing.*
+Denotes a lead(Pb)-free/RoHS-compliant package.
T&R = Tape and reel.
**EP = Exposed pad.*

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.

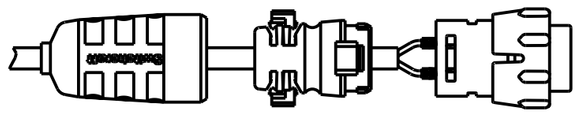
19-4675; Rev 13; 3/12



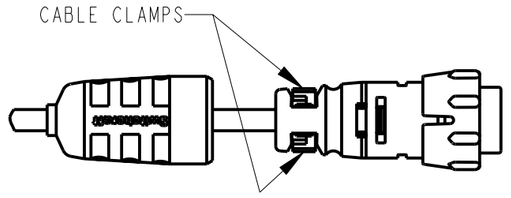
STEP 1
STRIP CABLE AS SHOWN.
FEED THE END OF THE CABLE THROUGH THE BOOT, CABLE CLAMP HOUSING, AND COUPLING RING IN THE ORDER AND POSITION SHOWN.
CRIMP CONDUCTOR TO CONTACT.
CONTACT #1 TO BE NEXT TO NOTCH.

**REMAINING CONTACTS TO BE PLACED COUNTER CLOCKWISE.

STEP 2
ALIGN COUPLING RING'S TABS WITH CORD CONNECTOR'S SIDE NOTCHES AND PUSH THE COUPLING RING ONTO CORD CONNECTOR.



STEP 3
PUSH THE CABLE CLAMP HOUSING FORWARD UNTIL IT LOCKS INTO THE CONNECTOR BODY AND SNAP THE TWO CLAMPS INTO IT'S COMPARTMENTS.



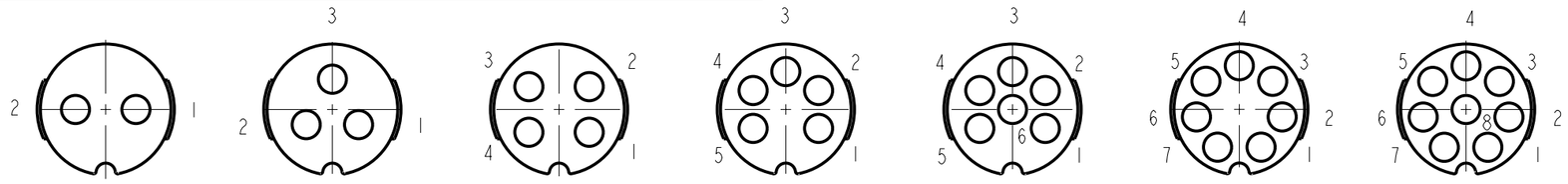
STEP 4
PUSH THE BOOT ALL THE WAY FORWARD TO SEAT TIGHTLY ONTO THE CABLE CLAMP HOUSING.

SPECIFICATIONS

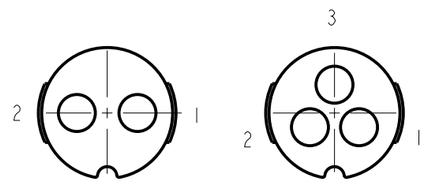
MECHANICAL:
SHOCK: MIL-STD 202 METHOD 213B, COND. K.
VIBRATION: MIL-STD 202 METHOD 201
LIFE: 300 INSERTION/WITHDRAWAL CYCLES (MINIMUM)

ELECTRICAL
DIELECTRIC WITHSTANDING VOLTAGE: 1,000 VAC
INSULATION RESISTANCE: 100 MEGOHMS (MIN) AT 77°F
CONTACT RESISTANCE: 5.0 MILLOHMS MAX.
CURRENT RATING: 7.5 AMPS (#20 CONTACT)
6.5 AMPS (7 & 8 PIN #20 CONTACT)
13.0 AMPS (#16 CONTACT)

ENVIRONMENTAL
TEMPERATURE LIMITS: -40°C TO +65°C (NON-OPERATING)
MOISTURE RESISTANCE: MIL-STD 202 METHOD 106F
INSULATION RESISTANCE: MIL-STD 202 METHOD 302, COND. B
THERMAL SHOCK: MIL-STD 202 METHOD 107G
SALT SPRAY: MIL-STD 202 METHOD 101D, COND. B
WATER TIGHTNESS TEST: U.S. COAST GUARD CFR 46 PART 110.20



2 - #20 | 3 - #20 | 4 - #20 | 5 - #20 | 6 - #20 | 7 - #20 | 8 - #20



2 - #16 | 3 - #16

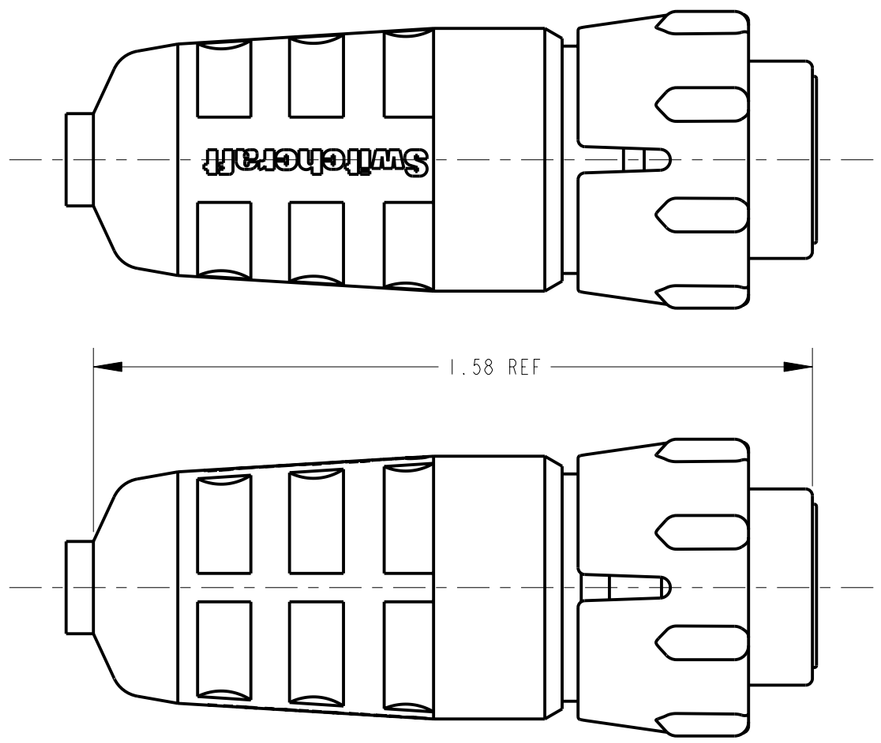
CONTACT ARRANGEMENTS

SHOWN ARE REAR VIEWS OF FEMALE CORD CONNECTORS

MATERIALS:
CORD CONNECTOR SHELL, CONTACT LOCKING DISK, COUPLING RING AND CABLE CLAMP ASSEMBLY:
THERMOPLASTIC POLYMER GLASS FIBER, FLAME RETARDANT

REAR BOOT AND CONNECTOR SHELL INTERIOR:
THERMOPLASTIC RUBBER

CONTACTS: COPPER BASE ALLOY GOLD-PLATED OVER NICKEL UNDERPLATE



CUSTOMER DRAWING

STANDARD[1] ← → OPTIONAL[2]

E N 3 C F

SERIES
CONECTOR TYPE
C - CORD
NUMBER OF CONTACTS
2,3,4,5,6,7,8
GENDER
F - FEMALE
CONTACT SIZE
16 - #16
AVAILABLE IN
2 AND 3 PIN ONLY

[1]STANDARD CONFIGURATION
- CONTACT SIZE: #20
- CONTACT STYLE: SOLDER CUP
- CONTACT PLATING: GOLD
- CONTACTS FACTORY INSTALLED
- PACKAGED IN BULK

[2]OPTIONAL CONFIGURATION
- LEAVE BLANK FOR STANDARD CONFIGURATION

REV	ECO NUMBER	DATE	BY	APVD
G	22232	6-7-98	SG	RB
F	21282	2-5-97	SG	RB
E	20897	2-5-97	SG	RB

REVISIONS

☆ STAR SYMBOL DENOTES CRITICAL DIMENSION

UNLESS OTHERWISE SPECIFIED
1. ALL DIMENSIONS IN INCHES
-TWO PLACE DECIMALS ±0.01
-THREE PLACE DECIMALS ±0.005
-ANGLES ±1°
-ALL DIA. CONCENTRIC WITHIN 0.005 T.I.R.
2. FEATURES ON THE SAME CENTERLINE MUST BE ALIGNED WITHIN ±0.002
3. REMOVE ALL BURRS

DO NOT SCALE DRAWING

SIZE	WIDTH	MULT	LBS/M	TEMPER

FINISH: SPEC No.
MATERIAL: SPEC No.
FIRST USED ON
SCALE: 3:1

DATE DRAWN	BY	CHKD	APVD
3-4-94	SG	SG 2-6-95	FL 2-6-95

Switchcraft®

SHEET 1 OF 1

NAME: FEMALE CORD CONNECTOR
PART No.: EN3C_F
REV: G

Pro/Engineer CAD File

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster
CURSO 2017/2018

PLANOS

Máster en Ingeniería Industrial

ALUMNO:

Luis Núñez Couselo

TUTOR

José Luis Calvo Rolle

FECHA

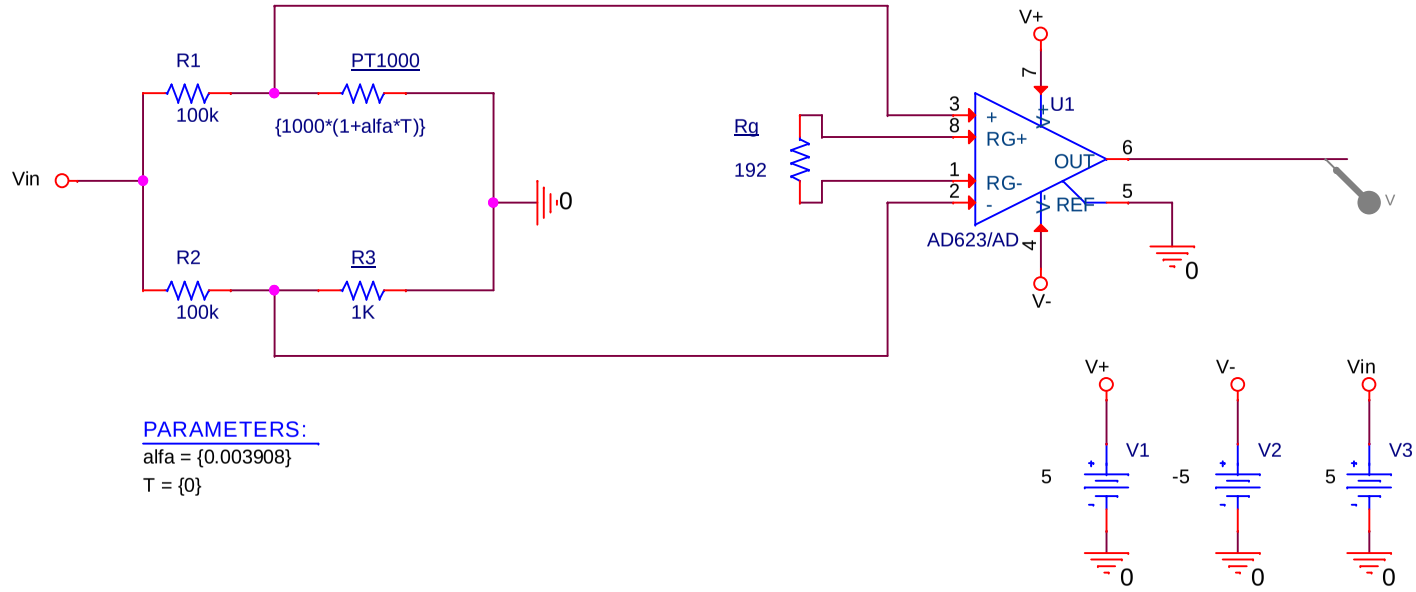
FEBRERO 2018

4 PLANOS

4.1- Circuito sensor PT1000

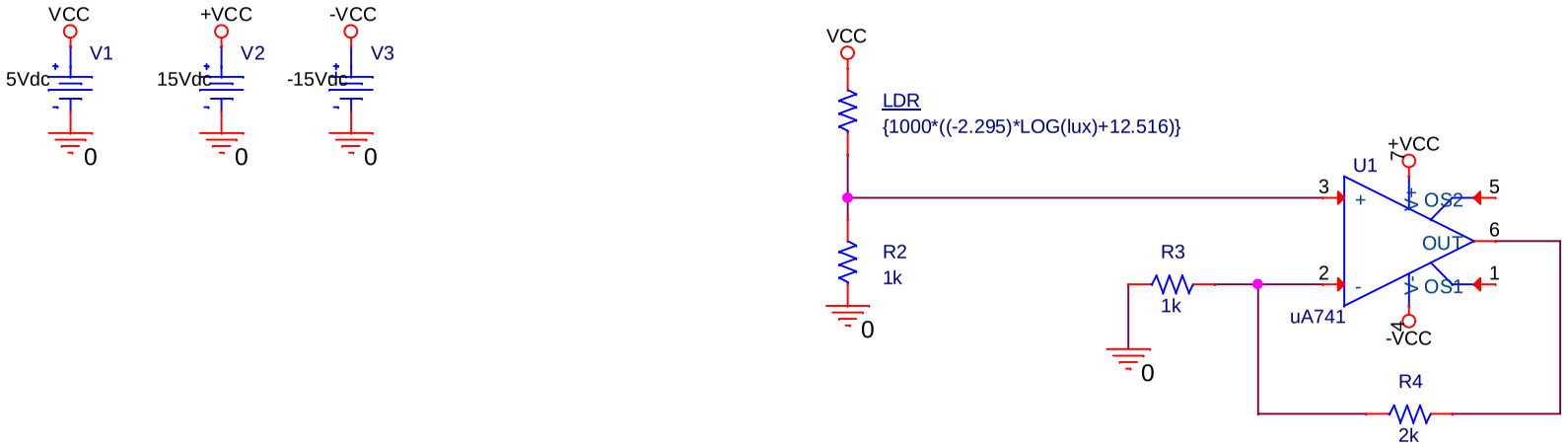
4.2- Circuito sensor LDR

4.3- Circuito sensor NTC



PARAMETERS:
alfa = {0.003908}
T = {0}

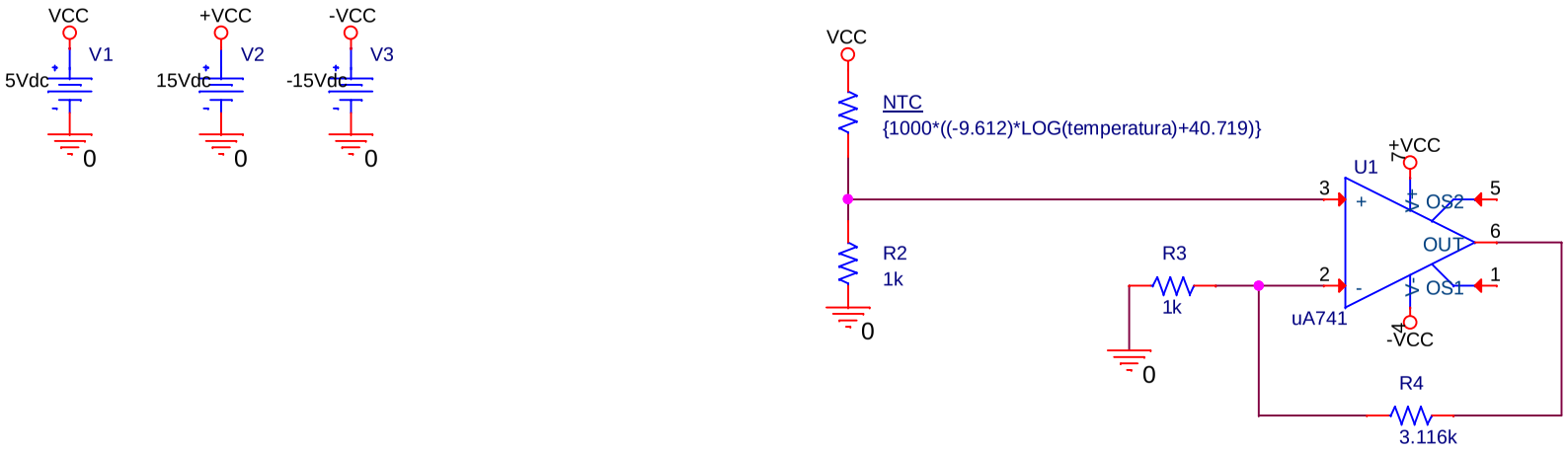
Title	CIRCUITO SENSOR PT1000	
Size A	Document Number 000000000	Rev 0
Date: Saturday, January 27, 2018	Sheet 1 of 1	



PARAMETERS:

lux = 0

Title: CIRCUITO SENSOR LDR			
Size: A	Document Number: 000000000		Rev: 0
Date: Tuesday, November 14, 2017		Sheet 1 of 1	



Title: CIRCUITO SENSOR NTC		
Size: A	Document Number: 000000000	Rev: 0
Date: Saturday, December 30, 2017	Sheet 1 of 1	