UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN SISTEMA DE CAPTACIÓN Y ALMACENAMIENTO DE LA EVOLUCIÓN DE LA TEMPERATURA EN PUNTOS REMOTOS.

AUTOR: GARCÍA BOLINCHES, JOAN

TUTOR: MARTÍNEZ DE JUAN, JOSÉ LUÍS

Curso Académico: 2016-17

AGRADECIMIENTOS

"A mis padres y a mi tío,

por su apoyo incondicional durante todo el grado"

RESUMEN

Hoy en día, tanto en el ámbito industrial como en el doméstico, se demanda un sensado y monitorización de distintas variables de nuestro entorno (temperatura, humedad, luminosidad, turbidez, consumo de agua, potencia eléctrica consumida, etc). En este contexto, el presente Trabajo Fin de Grado (TFG) tiene como propósito general la adquisición de las temperaturas de un entorno dado, para poder efectuar una posterior monitorización sencilla de ésta información.

En concreto, el objetivo del presente TFG será diseñar un dispositivo que adquiera las diferentes temperaturas de un determinado recinto, en un determinado periodo de tiempo y sea capaz de almacenar estos datos en un microcontrolador, para poder llevar a cabo la posterior descarga y visualización de los mismos. Deberá disponer de un almacenamiento de la evolución de la temperatura captada durante al menos 24 horas. La monitorización de los datos de temperatura adquiridos y almacenados no es objetivo del presente trabajo, pues dicha monitorización se realizará en otro TFG complementario.

Sería interesante que nuestro sistema sea portátil, compacto y autónomo para que pueda cambiar fácilmente de lugar, y además, que pueda funcionar sin la necesidad de estar conectado a la red eléctrica, es decir, siendo alimentado a través de baterías. Será necesario, por tanto, que los consumos eléctricos sean reducidos para aumentar las horas de autonomía del sistema.

También se incluirá en el presente TFG la posibilidad de cambiar algunos parámetros de configuración de nuestro dispositivo, como la frecuencia de muestreo de captación de datos, tiempo total de recogida de muestras y número de muestras a almacenar en memoria. Esto último se ha realizado con la finalidad de diseñar un dispositivo que sea comercializable y adaptable para aplicaciones venideras.

Palabras Clave: Sensor de temperatura. Adquisición y almacenamiento de datos. Instrumentación virtual.

RESUM

Avui en dia, tant a l'àmbit industrial com en la llar, es demanda un sensat y monitorització de distintes variables del nostre entorn (temperatura, humitat, lluminositat, terbolesa, consum d'aigua, potència elèctrica consumida...) . En aquest context, el present Treball Fin de Grau (TFG) té com a propòsit general l'adquisició de les temperatures d'un determinat entorn, per poder dur a terme una posterior monitorització senzilla d'aquesta informació.

En concret, l'objectiu del present TFG serà dissenyar un dispositiu que adquirisca les diferents temperatures d'un determinat recinte, en un determinat període de temps i sigui capaç d'emmagatzemar aquestes dades en un microcontrolador, per poder realitzar la posterior descàrrega i visualització deels mateixos. Deurà disposar d'un emmagatzematge de l'evolució de la temperatura captada durant almenys 24 hores. La monitorització de les dades de temperatura adquirits i emmagatzemats no és objectiu del present treball, ja que aquesta monitorització es farà en un altre TFG complementari a aquest.

Seria interessant que el nostre sistema sigui portàtil, compacte i autònom per a que es puga canviar fàcilment el lloc, i a més a més, funcioni sense necessitat d'estar connectat a la xarxa elèctrica, és a dir, sent alimentat a través de bateries. Serà necessari, per tant, que els consums elèctrics siguin reduïts per augmentar les hores d'autonomia del sistema.

També s'inclourà en el present TFG la possibilitat de canviar alguns paràmetres de la configuració del nostre dispositiu, com la freqüència de mostreig de dades, temps total de recollida de dades y nombre de mostres a emmagatzemar en memòria. Tot açò s'ha realitzat amb la finalitat de dissenyar un dispositiu que sigui comercialitzable i adaptable per a aplicacions futures.

Paraules clau: Sensor de temperatura. Adquisició i emmagatzematge de dades. Instrumentació virtual.

ABSTRACT

Nowadays, in the industrial environment and at home, monitoring of different variables of our environment (temperature, humidity, turbidity, water consumption, luminosity, electrical power consumed ...) is required. For greater control and knowledge of these, the main purpose of this TFG is to capture the different temperatures in a particular environment, to be able to carry out a subsequent simple monitoring of this information.

Specifically, the purpose of this TFG will be to design a device that acquires the different temperatures of a certain environment, in a certain period of time and store this data in a microcontroller, to let the following download and visualization of these temperatures. It must have storage of the evolution of the temperature captured for at least 24 hours. The monitoring of the temperature data acquired and stored is not objective of the present work, this monitoring will be done in another complementary TFG.

It would be interesting that our system can be portable, compact and autonomous for change the place of it; in addition, it is necessary that our system can operate without the need to be connected to the electric network. Our device will be powered by batteries. It will be necessary that the consumptions are reduced to increase the hours of autonomy of the system.

Also included in this TFG is the possibility to change some configuration parameters of our device, such as sampling frequency of data storage, total time of storage and number of samples to be stored in memory. This has been done in order to design a device that is marketable and adaptable to future applications.

Keywords: Temperature sensor. Storage of data. Virtual instrumentation.

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Presupuesto
- Anexos

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO.

LISTADO DE FIGURAS

MEMORIA

CAPÍTULO 1. INTRODUCCIÓN

En el presente capítulo se especificarán los objetivos del proyecto y el alcance del mismo. Para concluir, se detallará el árbol funcional del producto a desarrollar.

1.1. OBJETIVOS DEL DOCUMENTO

1.1.1. OBJETIVO GENERAL

El objetivo principal del presente TFG será diseñar un dispositivo de captación y almacenamiento de la variable temperatura en un entorno dado y en un periodo de tiempo establecido. Este dispositivo debe medir y almacenar las temperaturas sin la necesidad de estar conectado a la red eléctrica, y así hacer mediciones de entornos aislados. Por todo ello, debemos tener en cuenta tres objetivos específicos a la hora de desarrollar la memoria del TFG.

1.1.2. OBJETIVOS ESPECÍFICOS

1. **Selección del sensor de temperatura y desarrollo de la electrónica necesaria para su correcto funcionamiento:** El sensor es indispensable en el circuito de instrumentación para la captación de la temperatura existente en el entorno. Deberá leer un rango de temperaturas de -50ºC a 50ºC, ya que se pretende adaptar el dispositivo final a múltiples entornos. El error máximo permitido para la lectura de temperaturas se establece en ±0,5ºC.

2. **Circuito de acondicionamiento de la señal del sensor:** Incluyendo todos los componentes electrónicos del circuito indispensables para acondicionar la señal entre unos valores de tensión/corriente de salida para la posterior toma de datos en un microcontrolador. En este circuito incluye resistencias, condensadores, transistores, amplificadores operacionales, reguladores, etc. El rango de tensiones de salida lo establecerá el microcontrolador empleado, en nuestro caso será de 0 V a 5 V.

3. **Selección y programación de un microcontrolador:** Para el almacenamiento de los valores de la temperatura captada resulta indispensable la utilización de un

microcontrolador. Hay múltiples opciones en el mercado. Por ello, se debatirá y se justificará qué modelo de microcontrolador se ajusta mejor a los requisitos de nuestro proyecto. Se seleccionará un tipo de microcontrolador con almenos una capacidad de memoria auxiliar (EEPROM) de 1440 datos, para poder así almacenar un elevado número de valores de temperaturas.

1.2. ALCANCE DEL PROYECTO.

El alcance del proyecto se puede sintetizar en los siguientes puntos:

- Captación de la temperatura en diferentes entornos (al menos de-50ºC a +50ºC).
- Almacenamiento en la memoria de un microcontrolador (al menos 1 dato por minuto durante 24 horas, es decir 1440 datos).
- Autonomía de al menos 24 horas.
- Cálculo de distintos parámetros en el prototipo tales como rango de temperaturas, rango de tensiones de salida del circuito, datos captados por día, etc.
- Comunicación mediante puerto USB/bluetooth a PC o sistema visual.
- Monitorización de procesos (TFG complementario).
- Cálculo de errores de medida en nuestro prototipo (±0,5ºC como error máximo)
- Cálculo del tiempo de desarrollo y cálculo de un presupuesto del presente proyecto.

1.3. ÁRBOL FUNCIONAL DEL PRODUCTO A DESARROLLAR.

El árbol funcional del producto a desarrollar se muestra de forma esquemática en la siguiente Figura 1.1.

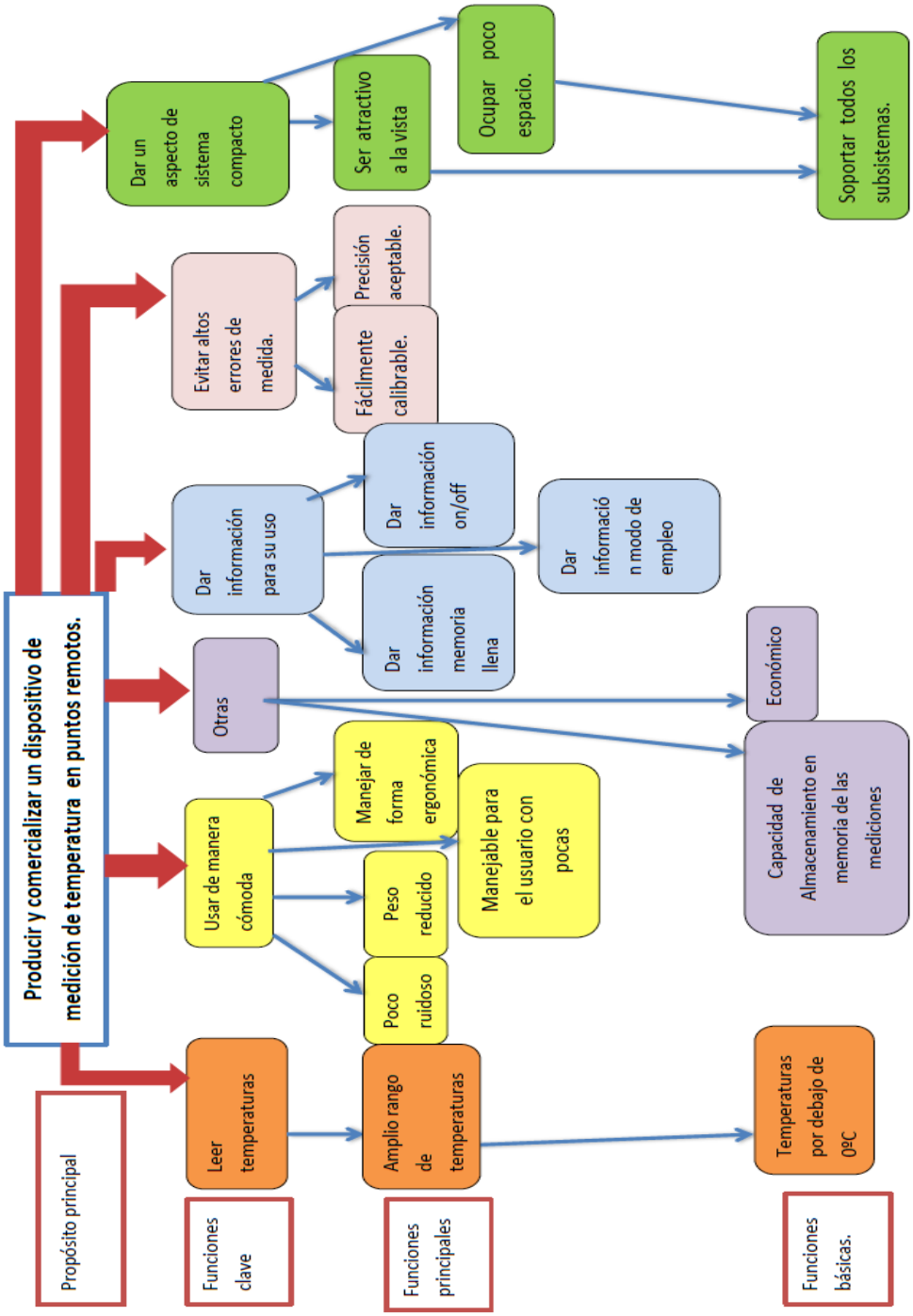


Figura 1.1: Árbol funcional del producto a desarrollar.

CAPÍTULO 2. INTRODUCCIÓN AL PROBLEMA: MOTIVACIÓN, ANTECEDENTES Y JUSTIFICACIÓN.

En el presente capítulo, una vez fijado el objetivo del proyecto en el capítulo anterior, se detalla la motivación para la realización de este trabajo. A continuación se describen algunos equipos con características similares al que se desarrolla en este proyecto, indicando sus principales funcionalidades. Seguidamente, se resalta el interés comercial de este equipo y finalmente, se describen una serie de especificaciones a tener en cuenta a la hora de llevar a cabo el diseño.

2.1. MOTIVACIÓN.

El conocimiento, almacenamiento y regulación de la temperatura de un determinado recinto resulta de gran interés tanto en el ámbito industrial como doméstico. En determinados procesos industriales (plantas de procesamiento de alimentos, cámaras frigoríficas, industrias farmacéuticas, etc.) se requiere conocer cómo evoluciona la temperatura en un periodo de tiempo determinado. En procesos de certificación de determinados productos también resulta esencial conocer la temperatura a la que se han realizado las pruebas durante todo el proceso. De igual forma, puede resultar de gran interés conocer cómo evoluciona la temperatura en un ambiente doméstico, para poder fijar una temperatura dentro de la zona de confort o para mejorar la eficiencia en los sistemas de refrigeración o calefacción.

En todos estos casos puede ser necesaria la adquisición, almacenamiento y posterior monitorización de la temperatura. En este sentido, el presente TFG propone la realización de un equipo que permita medir y almacenar, de forma periódica, los datos de temperatura en una determinada localización en un periodo de tiempo determinado, para posteriormente procesar dicha información.

2.2. DISPOSITIVOS EXISTENTES EN EL MERCADO DE MEDIDA Y ALMACENAMIENTO DE TEMPERATURA.

Buscando información acerca de dispositivos similares a nuestro prototipo, encontramos en el mercado múltiples opciones con diferentes formatos de medición/monitorización de la temperatura. A continuación se detallan dos de estos.

2.2.1 Medisana FTN.

Se trata de un dispositivo de medición de la temperatura por infrarrojos de la compañía Medisana FTN, disponible en Amazon (Figura 2.1), cuyas características más significativas son [1]:

- Precisión: 0.18 °C
- Rápida medición: 1 segundo aproximadamente.
- Almacenamiento de diferentes mediciones en memoria.
- Posibilidad de almacenamiento de 30 mediciones.
- 30 espacios de memoria para guardar los datos obtenidos.
- Señal acústica en caso de fiebre y semáforo de alerta en pantalla.
- Indicadores en triple display de 7 segmentos para visualización de la temperatura del sistema.
- Precio: 35,91 €.

Figura 2.1: Termómetro Medisana FTN de captación por infrarrojos [1].

2.2.2. 40TB Gefran.

Otro ejemplo de dispositivo de medición y almacenamiento algo más complejo sería el modelo 40 TB (*indicador con doble display de temperatura y presión con unidad de alarma)*, de la compañía GEFRAN [2] (Figura 2.2).

Figura 2.2: Visualización del modelo 40TB; indicador con doble display de temperatura y presión con unidad de alarma [2].

PRINCIPALES USOS Y APLICACIONES:

- Indicación y alarma para la temperatura de fusión y presión en las líneas de extrusión.
- Bancos de prueba.
- Plantas de procesamiento de alimentos: Indicación de la temperatura y presión
- Pesaje
- Interruptores de presión (presostatos), termostatos
- Plantas textiles

ALGUNAS DE SUS CARACTERÍSTICAS MÁS IMPORTANTES SON:

- Entradas configurables desde la placa frontal.
- Fácil calibración con el calibrador de deformación de sensibilidad auto-rango.
- Control de la alimentación del sensor (input 1).
- Protección de código seleccionable.
- Posibilidad de configurar la unidad.
- Fuente de alimentación para transmisores.
- Fácil de configurar. Disponibilidad de linealización personalizada.
- Unidades de ingeniería más comunes disponibles en la pantalla o en las etiquetas.
- Adquisición y alarma programable desde 15 ms hasta 120 ms. Con una resolución de 16000 a 4000 divisiones
- Visualización de los valores de las variables.
- 3 alarmas completamente configurables desde la placa frontal
- Precisión 0.2% sobre el fondo de escala ± 1 dígito.
- Protocolo: GEFRAN CENCAL o MODBUS
- Precio aproximado: $799.20

2.3. JUSTIFICACIÓN E INTERÉS COMERCIAL.

Este equipo sería de interés, tal y como se ha explicado en el apartado de motivación, para aquellas empresas que requieran medir y almacenar la temperatura del recinto donde se está llevando a cabo el proceso industrial en cuestión. Serían ejemplos de empresas que tendrían interés en entre producto las siguientes [3, 4, 5]:

- Instalaciones donde se almacenan productos refrigerados, congelados y ultracongelados, pues según la normativa vigente EN12830 [6], en este tipo de instalaciones se deben registrar y documentar las temperaturas de dichos congelados.
- Empresas que realizan la certificación y validación de productos. Cualquier proceso de certificación requerirá un histórico de la temperatura que compruebe el cumplimiento de los puntos de temperatura requeridos.
- Industria farmacéutica. Por ejemplo, las vacunas son sensibles a la temperatura y pueden perder efectividad si estas no se almacenan en unas condiciones de temperatura óptimas.

- Instalaciones de sistemas de calefacción y ventilación. El registro de la temperatura es necesario para la búsqueda de errores en la instalación de sistemas de calefacción y ventilación.
- Industria alimentaria, donde resulta fundamental conocer la temperatura a la que se almacenado un determinado producto para así asegurar que no se ha roto la cadena de frío.
- En el sector de la ganadería o de avicultura para asegurar que la temperatura de estos recintos es la adecuada para una mayor producción.

2.4. ASPECTOS A TENER EN CUENTA EN EL DISEÑO DEL DISPOSITIVO.

El diseño del dispositivo viene condicionado por una serie de pautas que dependen directamente de la monitorización de la temperatura almacenada. Esta monitorización se realizará a partir de instrumentos virtuales, por ejemplo programas diseñados en LabView. Todo esta parte de monitorización con instrumentación virtual es objeto de otro TFG complementario a éste, como ya se ha comentado.

Esta parte complementaria condiciona al diseño del dispositivo en los siguientes aspectos:

- Exige que el dispositivo disponga de un puerto para comunicación con el PC (vía USB, Bluetooth etc.)
- Exige una determinada frecuencia de muestreo para la captación de datos (1 dato de temperatura por minuto).
- Exige un tiempo total de almacenamiento de temperaturas de 24 horas.
- Exige al menos 1640 (200 + 1440) espacios de memoria a guardar en el dispositivo
- Exige una programación del dispositivo determinada. Se debe dejar vacíos los primeros 200 espacios de la memoria EEPROM. Ahí se almacenarán algunos parámetros tales como fecha de la medición, hora de inicio de la medición, temperatura mínima, temperatura máxima, temperatura media etc. Estos parámetros se ilustrarán en los gráficos obtenidos a partir del programa de LabView.
- Exige una autonomía mínima de 24 horas.

CAPÍTULO 3. PARTES DEL DISPOSITIVO A DISEÑAR

En el presente capítulo se expondrán, de forma genérica, las distintas partes en que se divide el proyecto junto con su correspondiente justificación.

3.1. ASPECTOS GENERALES.

Desde un punto de vista meramente general, el proyecto requiere inicialmente de un elemento de sensado de la temperatura del entorno, cuya selección se detallará en capítulos posteriores, el circuito de acondicionamiento electrónico y el circuito de almacenamiento de la información que se realizará mediante un microcontrolador. El proyecto consta, por tanto, de una parte de diseño analógico y otra parte de diseño digital, tal y como se muestra en la Figura 3.1.

Circuito electrónico de captación de la Temperatura (Analógico)

Ejemplo de microcontrolador: placa Arduino (Digital)

Figura 3.1: Esquema ilustrativo de las distintas partes del proyecto [7].

Entrando en mayor profundidad en el circuito electrónico de captación de temperatura se han distinguido dos partes claramente significativas. Una primera parte donde el propósito será la captación de la temperatura, resulta imprescindible el uso de un componente electrónico donde, por ejemplo, su voltaje de salida dependa exclusivamente de la temperatura al que

está sometido, por tanto, se ha utilizado un sensor de temperatura para la captación de la temperatura del recinto.

Una segunda parte dentro del circuito electrónico consistirá en el acondicionamiento de la señal, fijándola entre unos determinados valores que puedan ser procesados por el microcontrolador. Aquí existirán diferentes componentes electrónicos tales como amplificadores operacionales, resistencias, diodos, condensadores etc.

Centrándonos ahora en la parte digital, se tendrá que programar el microcontrolador adecuadamente para almacenar en memoria los diferentes datos de la temperatura adquirida durante al menos 24 horas. Aquí debemos programar distintos parámetros tales como el tipo de memoria donde se almacenarán los datos, frecuencia de muestreo, duración total del muestreo, etc.

3.2. CIRCUITO DE ACONDICIONAMIENTO.

3.2.1. Alimentación

Entrando en profundidad en el análisis del circuito electrónico podemos diferenciar una primera etapa de alimentación. Para alimentar a nuestro circuito se ha procedido a la utilización de baterías ya que es condición necesaria que nuestro dispositivo adquiera temperaturas sin el requisito de estar conectado a la red eléctrica. Será fundamental usar baterías con un valor de voltaje lo más reducido posible, ya que se priorizará hacia un diseño de dispositivo con un consumo mínimo.

3.2.2. Reguladores de tensión

La tensión que proporciona la batería no resulta suficientemente estable (fija) para alimentar los distintos elementos de nuestro circuito (operacionales, sensor, microcontrolador), pues su valor depende del estado de carga de la misma. Además, el nivel de tensión proporcionado por la batería no se adapta necesariamente a las necesidades de estos circuitos electrónicos. Por estos dos motivos utilizaremos reguladores de tensión, que proporcional el nivel de tensión deseado y fijo (independientemente de lo que demanda la carga).

3.2.3. Etapa de amplificación

Se utilizará también un Amplificador Operacional para acondicionar la señal de sensor, y adaptarla a una tensión admisible tanto para la alimentación microcontrolador como para la entrada del convertidor analógico-digital (AD) de dicho microcontrolador. Sería, además, interesante tener la posibilidad de ajustar la ganancia del amplificador por si en algún momento existiese alguna situación donde requeriría un cambio de Arduino® o microcontrolador empleado (por el consumo por ejemplo), y así adecuar unos valores de tensión admisibles para la alimentación/entrada de cada microcontrolador en cuestión. En otras palabras, se pretende diseñar un circuito de acondicionamiento independiente y

acoplable a diversos microcontroladores/convertidores AD existentes en el mercado. Para ello, la ganancia del amplificador se ajustará mediante unos potenciómetros.

3.3. DIAGRAMA DE BLOQUES DEL DISEÑO.

Se procederá en este apartado a la presentación de un diagrama de bloques, Figura 3.2, que servirá como un resumen ilustrativo de las diferentes partes en que se divide el proyecto.

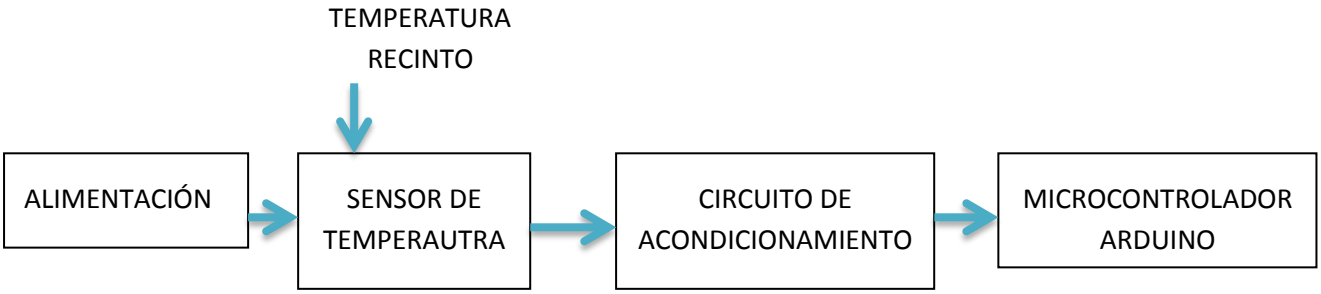


Figura 3.2: Diagrama de bloques del diseño.

CAPÍTULO 4. ALMACENAMIENTO EN MEMORIA

El almacenamiento de la temperatura adquirida se realizará mediante un microcontrolador. En este capítulo se discutirá y se justificará el microcontrolador utilizado. En concreto, se compara las características de un microcontrolador PIC de Microchip con la utilización de una placa de desarrollo de Arduino®, y se compararán aspectos tales como tiempo de desarrollo, flexibilidad, información disponible, requisitos de software específico, coste, etc.

Es indispensable la condición de que el microcontrolador empleado disponga de una salida para comunicación con el PC (vía USB, Bluetooth etc.), porque se requiere una posterior monitorización de los datos captados por el sensor con un programa específico. Esta monitorización será llevada a cabo mediante un TFG complementario a este.

Hay distintos PIC, microcontroladores de la compañía Microchip Technology, que disponen de entradas con una posibilidad de comunicación con PC, es decir, a partir de una conexión de cable USB o conexión externa de Bluetooth. Las placas de la compañía Arduino® disponen la ventaja de tener integrada una entrada USB (en todas sus versiones) para una comunicación con otros dispositivos.

4.1. ARDUINO® VS MICROCONTROLADOR

En el presente apartado se realizará una comparación de ambos elementos con una toma de decisión final sobre cuál se empleará en el proyecto descrito. Todo ello con sus debidas justificaciones.

4.1.1. Introducción a la discusión. Algunos aspectos generales.

ARDUINO®:

En primer lugar, se debe comentar que las placas de Arduino® no se tratan de microcontroladores en sí [8]. Por ejemplo, el modelo de placa Arduino® Mega2560 se trata de todo un circuito integrado en placa basado en el microcontrolador ATMega2560 de la entidad Atmel Corporation. En otras palabras, se podría decir que la compañía Arduino® desarrolla

placas de circuito integrado con un microcontrolador. Todas las placas de la compañía están basadas en microcontroladores de Atmel y otros elementos activos y pasivos (botón de reinicio, pines de entrada salida analógicas/digitales, puertos serie, conector de alimentación, conector USB etc.), tal y como se muestra en la Figura 4.1. También comentar que la presente compañía desarrolla y comercializa sus productos como software libre y de código abierto.

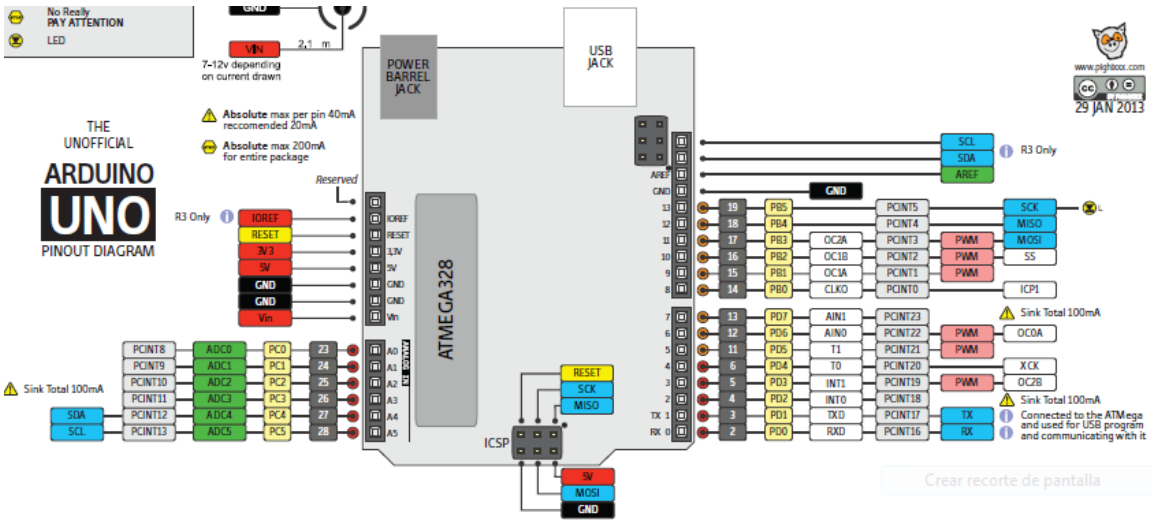


Figura 4.1: Simbología del hardware Arduino® UNO integrado con ATMEGA328 [9].

MICROCONTROLADORES.

Sin embargo los microcontroladores (PIC, AVR, ST, Freescale, TI etc.) son chips que se podrían integrar en cualquier circuito que se diseñe siempre que se programe adecuadamente, se acondicione la señal correspondiente etc. De manera habitual poseen otros periféricos integrados (Timers, PWM, convertidores A/D, puertos, osciladores etc.), tal y como se muestra en la Figura 4.2.

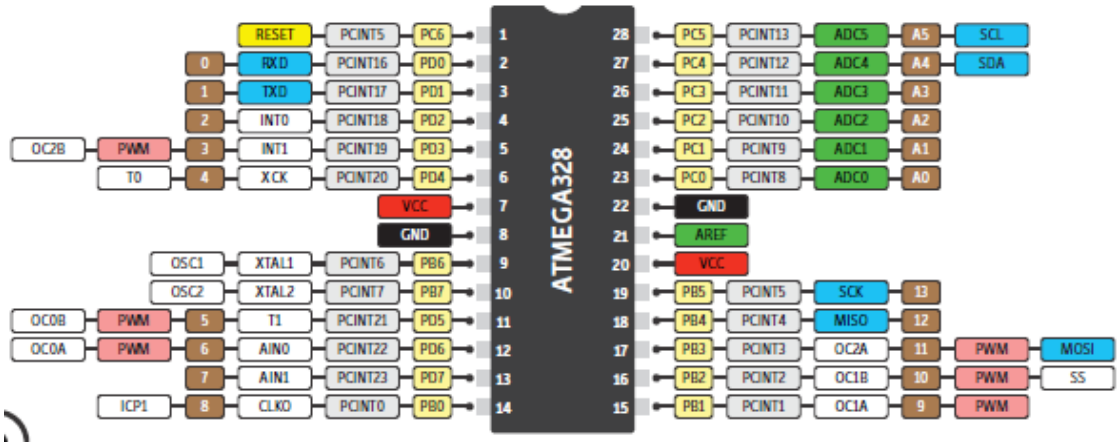


Figura 4.2: Diagrama de pines entrada/salida del microcontrolador ATMEGA328 [10]

Finalmente, comentar que dependiendo de cada aplicación, el tipo de proyecto a llevar a cabo o el tipo de problema a resolver, cada uno tendrá sus ventajas e inconvenientes. Por tanto, se concluye que dependiendo de la aplicación, se nos ajustará mejor las características de uno que las del otro.

4.1.2. Ventajas de cada componente.

Pasemos a continuación a evaluar las ventajas de cada uno por separado.

PLACA ARDUINO®.

En primer lugar procedemos a analizar la placa Arduino®, Figura 4.3. La placa Arduino®, presenta visualmente algunas ventajas, rápidamente se distinguen las entradas Analógicas/Digitales, puertos de comunicación, botón de Reset etc. Por tanto, se puede concluir que todo es más visual y la interfaz de puertos/periféricos viene considerablemente ordenada de fábrica siendo sencillamente localizable cada uno de sus elementos. Estos puertos/periféricos pueden estar internamente conectados al microcontrolador ATMega, eso ya no sería preocupación para el usuario.

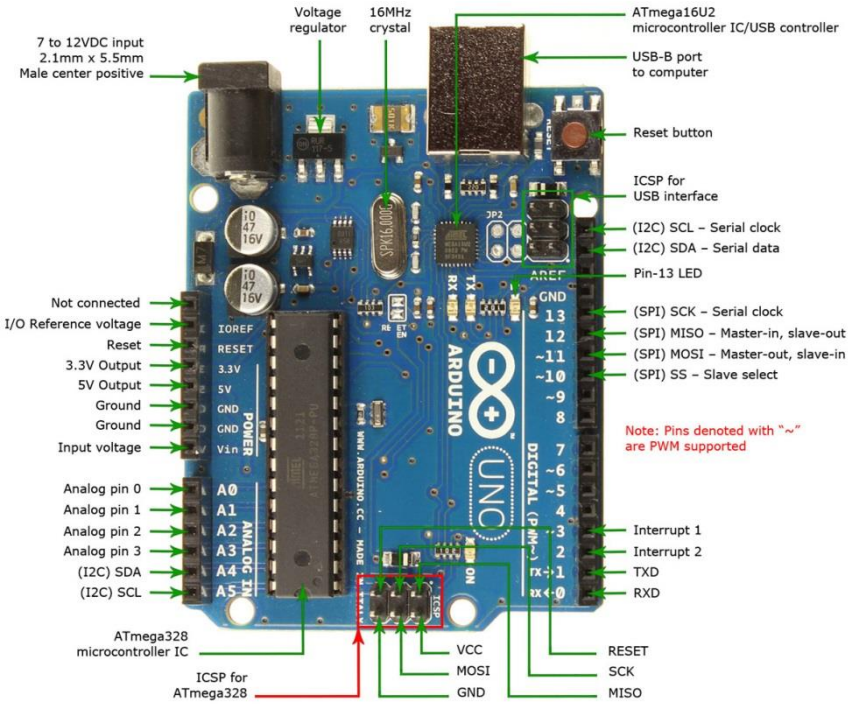


Figura 4.3: Ilustración del modelo de la placa Arduino® UNO. Pines y componentes [11].

Seguidamente, destacaremos Arduino® se puede afirmar que es una comunidad *"Open Source"*. Los usuarios comparten sus resultados, ejemplos de programas realizados, aparte de que las librerías en Arduino® se comercializan de manera libre y están estandarizadas. Esto no sería una ventaja a la hora de llevar a cabo la realización de un proyecto profesional, pero permite, a diferencia de los microcontroladores, una creación y desarrollo de múltiples aplicaciones de manera sencilla y con un tiempo de aprendizaje rápido. Las diferentes librerías

ya están pensadas para la estructura y simbología de cada uno de los distintos modelos de placas de la compañía Arduino®. Existe documentación muy útil para aprender a utilizarlo, muy clara y fácil de comprender, al contrario que en los microcontroladores, donde la documentación es muy densa.

Por tanto podemos extraer la conclusión que con Arduino® obtenemos un grado de éxito en nuestros proyectos de una manera más veloz. Si el alcance de un proyecto debe realizarse con tiempo limitado, probablemente con Arduino® se llegará más rápido a la solución de este por todo lo anteriormente expuesto. Aunque no podemos afirmar si sería lo más óptimo, pero sí se puede concluir que es más funcional.

También existen múltiples *Shields* para distintas aplicaciones realizables con Arduino® (Figura 4.4). Los *Shields* son placas modulares adicionales, comunicables con Arduino®. Se utilizan para ampliar la capacidad y las funciones que este permite realizar. Esto significa una clara simplificación sobre el desarrollo de estas aplicaciones.

Figura 4.4: Arduino® Ethernet Shield, para disponer de un Arduino® con conexion a redes Ethernet [7].

A parte, no necesitamos la descarga de ningún compilador externo a la compañía, ya que Arduino® nos ofrece un software libre con un entorno de desarrollo (IDE). En el fondo, es como si programásemos en Java, ya que este software está basado en el entorno de *Processing* (lenguaje de programación y entorno de desarrollo basado en Java) y en la plataforma de programación *Wiring* (Java). El presente IDE descrito, nos simplificará y ayudará en la programación y desarrollo del software. Este software Arduino® compilará el código programado y lo transferirá a nuestra placa (el software es compatible con todos los modelos de Arduino®).

A diferencia de la mayoría de microcontroladores donde nos debemos descargar el IDE y además el compilador (C18 o C30 serían algunos ejemplos para PICs) para ejecutar así el lenguaje de programación que se requiriese.

Otra ventaja que dispone los modelos Arduino® sería que ya disponen de *bootloader* (gestor de arranque, Figura 4.5). El gestor de arranque es un programa que se inicia en el arranque del sistema, se encarga de preparar el sistema operativo para poder así llevar a cabo el

cumplimiento de todas sus funciones. Este programa ya ha sido instalado previamente en el microcontrolador en cuestión, nos permitirá cargar el código de programación a la placa [12].

Figura 4.5: Ilustración del modo de gestor de arranque de un móvil Android. Por defecto el *bootloader* Android está bloqueado, no sería posible modificación alguna realizada por el usuario [13].

Es decir, a diferencia de la mayoría de microcontroladores, cada modelo Arduino® ya viene precargado con *bootloader*, esto permitirá cargar nuevo código sin la necesidad de un programador adicional.

MICROCONTROLADORES

En primer lugar, comentar que no todos los microcontroladores llevan incorporada la posibilidad de comunicación con el PC, se deberá elegir modelos que lógicamente lleven algún tipo de comunicación con el PC (USB, Bluetooth, etc.) En la siguiente Figura 4.6 se ilustra el esquema del proceso de comunicación con el PC. En el fondo este es el objeto final del uso de este tipo de dispositivos, para así proceder al almacenamiento de los datos captados en memoria.

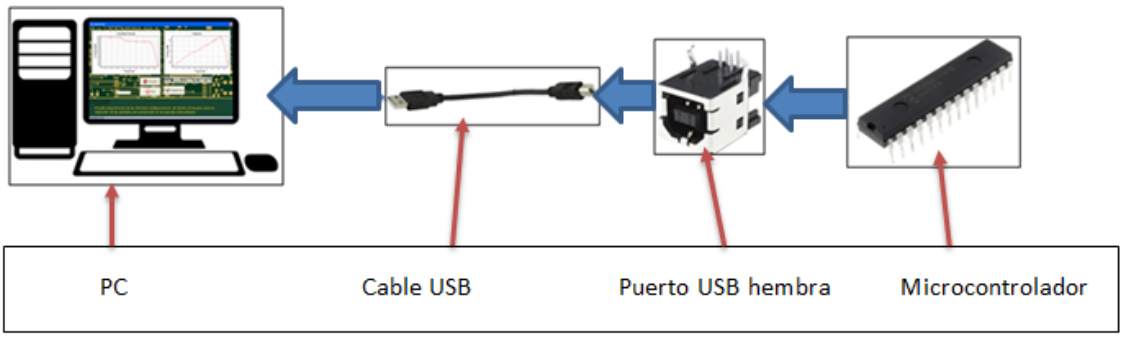


Figura 4.6: Esquema del proceso de comunicación Microcontrolador - PC.

Seguidamente pasamos a evaluar algunas de sus características más distinguibles respecto a Arduino®.

Primeramente, el usuario debe de saber instalar, programar cada microcontrolador en el lenguaje de programación correspondiente y modificar sus registros internos. Por tanto, se deberá proceder a una búsqueda de información más profunda al respecto sobre su funcionamiento interno, electrónica Digital y Analógica etc. Un ejemplo de fuente de información serían sus respectivas hojas de características. Entrando en el tema de los *Shields*, en los microcontroladores deberían ser diseñados e instalados por el usuario teniendo siempre en cuenta los requerimientos del proyecto.

En consecuencia, se requiere un mayor conocimiento para el usuario del hardware del dispositivo a utilizar, lo que si bien nos permitiría una mayor optimización y flexibilidad al proyecto a desarrollar, esto por el contrario resulta en un mayor tiempo de desarrollo. Por tanto esta es la mayor ventaja que dispone un microcontrolador respecto a Arduino® [12].

Figura 4.7: Ilustración del microcontrolador PIC18F2550 [14].

Otra ventaja de los microcontroladores [12, 15] sería que disponemos de una variedad en el mercado mucho más amplia respecto a Arduino® (que únicamente disponemos de una decena de placas aproximadamente y todas muy similares). Podemos buscar el microcontrolador idóneo que se adapte al proyecto a desarrollar (La Figura 4.7 muestra un ejemplo de microcontrolador de la compañía Microchip Technology Inc.).

Desde el punto de vista económico, los microcontroladores ganan la batalla. Son mucho más económicos que los modelos Arduino®, pues tal y como hemos indicado, los modelos Arduino® son placas de desarrollo que no incluyen únicamente el microcontrolador. El precio de un microcontrolador puede fluctuar de 1 € a 5 €, mientras que Arduino® de 20 € a 50 €. Esta diferencia de precio podría considerarse poco importante, pero si se desarrolla un proyecto que conllevará un número elevado de productos a manufacturar, la diferencia de precio sería más que considerable. Utilizando Arduino®, probablemente en el producto final no se estén utilizando parte de su gran número de pines. Por tanto es más eficiente diseñar nuestro propio circuito con microcontrolador. A parte de que se adaptará mejor al proyecto desde el punto de vista presupuestario, también se puede considerar tamaño ocupado como una ventaja y unas especificaciones técnicas más adecuadas y eficientes.

Si queremos trabajar con microcontrolador, tendremos que descargarnos previamente un compilador IDE o software de desarrollo, esto puede ser una ventaja porque elegiríamos el que más se adapte a las características del dispositivo o en el que más familiarizado esté el usuario.

A parte, antes de implementar nuestro diseño en placa, también deberíamos disponer un hardware de entrenamiento para hacer pruebas y experimentaciones sobre nuestro prototipo. Cuando el proyecto funcionase, ya se podría implementar en placa nuestro diseño. Esto ya sería nuestro hardware desarrollado donde existirá el microcontrolador en cuestión a parte de varios componentes electrónicos adicionales, *shields,* etc

Como ejemplo ilustrativo, en la Figura 4.8 mostramos el kit de desarrollo para un microcontrolador PIC específico y en la Figura 4.9 el software de compilación de MPLAB.

Figura 4.8: Kit Desarrollo de Microcontroladores PIC empresa MCE Electronics [16]

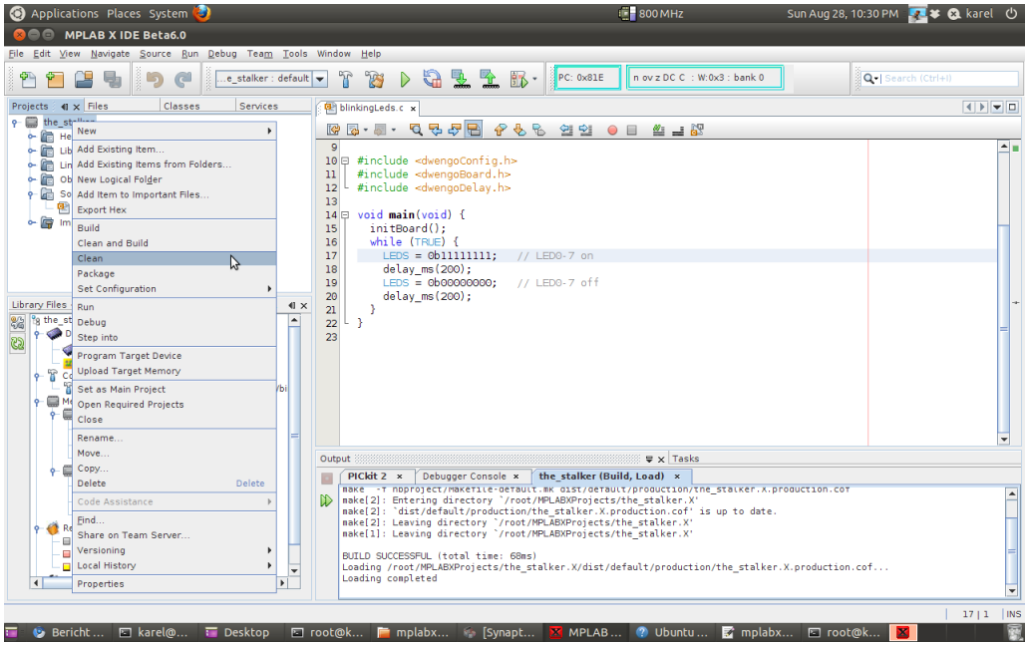


Figura 4.9: Software MPLABX- Microchip para Microcontroladores PIC. [17]

4.1.3. Decisión final. Matriz de decisión del microcontrolador.

Procedemos en el presente apartado a la construcción de una matriz de decisión con una posterior decisión final [18, 19]. Primeramente, se asignará un valor a cada criterio a comparar de acuerdo a las pautas siguientes:

1 = satisface el criterio bastante mal / muy mal

2,3,4 = satisface el criterio moderadamente

5 = satisface el criterio razonablemente bien

6,7,8 = satisface el criterio bien

9 = satisface el criterio muy bien

Seguidamente, a cada criterio a comparar se le asignará una importancia (en tanto por uno). A continuación se procederá a una maximización de cada criterio. Luego se aplicará la suma ponderada a cada alternativa y finalmente se elegirá una opción u otra. La matriz se ilustra en las Figuras 4.10 y 4.11.

	PLACAS ARDUINO®	MICROCONTROLADORES
COMPLEJIDAD PROGRAMACIÓN. (0,08)	**9**	**5**
CONOCIMIENTO DEL HARDWARE DEL MICROCONTROLADOR. (0,08)	**8**	**5**
COSTE. (0,15)	**2**	**9**
TIEMPO DE APRENDIZAJE. (0.15)	**9**	**3**
COMPILADORES. (0,05)	**9**	**8**
TIEMPOS DE DESARROLLO. (0,2)	**9**	**1**
FLEXIBILIDAD (0,025)	**1**	**9**
DISEÑO MÁS COMPACTO. (0,15)	**3**	**9**

Figura 4.10: Matriz de decisión placa Arduino®-microcontrolador. Asignación de valores a cada criterio e importancias.

Nota: Hay que tener en cuenta que hay algunos criterios por ejemplo el criterio coste, que cuando se le asigna un 9 significa que "satisface el criterio muy bien" por lo tanto el coste con un 9 será el menor (más favorable para la toma de decisión final).

A continuación se pasará a la normalización de las valoraciones de la matriz. Seguidamente, se procederá al cálculo del valor suma ponderada para cada alternativa.

	PLACAS ARDUINO®	MICROCONTROLADORES
COMPLEJIDAD PROGRAMACIÓN. (0,08)	**1,00**	**0,56**
CONOCIMIENTO DEL HARDWARE DEL MICROCONTROLADOR. (0,08)	**1,00**	**0,63**
COSTE. (0,15)	**0,22**	**1,00**
TIEMPO DE APRENDIZAJE. (0.15)	**1,00**	**0,33**
COMPILADORES. (0,05)	**1,00**	**0.89**
TIEMPOS DE DESARROLLO. (0,2)	**1,00**	**0,11**
FLEXIBILIDAD (0,025)	**0,11**	**1,00**
DISEÑO MÁS COMPACTO. (0,15)	**0,33**	**1,00**

	PLACA ARDUINO®	MICROCONTROLADORES
	0,74	**0,62**

Figura 4.11: Matriz de decisión placa Arduino®-microcontrolador. Valores normalizados y suma ponderada.

4.1.4. Descripción de los criterios y justificación de los valores de importancias asignadas en la matriz de decisión.

Complejidad de programación: Si se requiere un lenguaje de programación específico. Ambos se programan en C (no requiere, por ejemplo, una programación en ensamblador).

Conocimiento del hardware del microcontrolador. En el caso del PIC se requiere un conocimiento más profundo del mismo, y por tanto, su tiempo de desarrollo es mayor. Es decir, sería un aspecto negativo para el caso del PIC.

Coste: Se debe considerar también que en el caso del PIC no sólo computa el precio del microcontrolador, que es mucho más barato, si no también todos los componentes adicionales que este requiere (reloj, condensadores, conector USB, etc.), así como la realización de la placa de circuito impreso.

Tiempo de aprendizaje: Es el tiempo que se requiere para tener un manejo suficientemente amplio y de cada alternativa. Es una parte importante dentro del tiempo de desarrollo y está también asociado al hecho de que en el caso del PIC se requiere un conocimiento del hardware más profundo, lo que también conlleva un mayor tiempo de puesta a punto de la placa con el microcontrolador.

Compiladores: Si el compilador es o no de libre distribución.

Tiempo de desarrollo: En el caso de Arduino® ya compramos directamente la placa de desarrollo con todas las conexiones necesarias, mientras que en el caso del PIC debemos hacer la placa de circuito impreso con todos los componentes necesarios.

Flexibilidad: Durante el desarrollo del proyecto la jerarquía Arduino® conlleva una menor flexibilidad y menor optimización si se pretende personalizar el trabajo a aplicaciones específicas.

Diseño más compacto: En el caso del PIC como diseñamos la placa junto con todos sus componentes, el resultado final debe ser más compacto. Arduino® es una placa ya diseñada donde adaptaremos a ella nuestro circuito diseñado.

La importancia de cada criterio ha sido asignada de manera que si un criterio favorece la celeridad del desarrollo del proyecto, se le asigna un valor numérico superior respecto a otro criterio considerado desfavorable para la característica descrita, ya que se dispone de un tiempo limitado para la presentación del mismo.

Por tanto, la alternativa seleccionada y que se utilizará en el presente proyecto como microcontrolador será una placa Arduino®. Concretamente se utilizará la placa Arduino® Mega 2560 [20] por las razones que se detallan a continuación:

- Es la que mayor almacenamiento en memoria EEPROM posee de toda la compañía (4 Kb). Esto supone la posibilidad de incrementar la periodicidad de captación o el tiempo que pueden estar tomándose estas captaciones. Hasta 24 horas con una adquisición por cada minuto del día sería suficiente y solo dispondríamos de ¾ de memoria llena aproximadamente.
- El microcontrolador viene instalado en una placa con mayor número de pines para entradas y salidas.
- Compatible con distintos voltajes de alimentación.
- Compatible con diversos convertidores analógicos digitales del mercado, pueden ser una buena opción para la transferencia de las temperaturas captadas al microcontrolador.

Cabe recordar que siempre se debe tener presente cual es el único objetivo del microcontrolador para este proyecto: el almacenamiento en memoria de los datos captados por el sensor de temperatura.

En la Figura 4.12 se muestra una tabla comparativa sobre las características de algunos modelos de placa Arduino®:

Característica de Arduino®	UNO	Mega 2560	Leonardo	DUE
Tipo de microcontrolador	Atmega 328	Atmega 2560	Atmega 32U4	AT91SAM3X8E
Velocidad de reloj	16 MHz	16 MHz	16 MHz	84 MHz
Pines digitales de E/S	14	54	20	54
Entradas analógicas	6	16	12	12
Salidas analógicas	0	0	0	2 (DAC)
Memoria de programa (Flash)	32 Kb	256 Kb	32 Kb	512 Kb
Memoria de datos (SRAM)	2 Kb	8 Kb	2.5 Kb	96 Kb
Memoria auxiliar (EEPROM)	1 Kb	4 Kb	1 Kb	0 Kb

Figura 4.12: Tabla comparativa de las características de distintos modelos de placa Arduino[21]

Por todo lo anteriormente expuesto, el modelo de Arduino® a utilizar en el presente proyecto será Arduino® Mega2560 dada su superioridad en la capacidad de memoria EEPROM.

CAPÍTULO 5.SENSOR DE TEMPERATURA

Para el proyecto, como ya se ha comentado en capítulos anteriores, se necesita un componente que capte la temperatura existente en un determinado recinto. Por tanto, se empleará un sensor de temperatura.

En el presente capítulo, se procederá a la definición y posterior descripción de diversos tipos de sensores de temperatura. Finalmente, se elaborará otra matriz de decisión para así decidir qué sensor se adapta mejor a los requisitos de nuestro proyecto.

5.1. DEFINICIÓN DE SENSOR DE TEMPERATURA.

Los sensores de temperatura y su acondicionamiento están ligados a una parte de la electrónica denominada Instrumentación electrónica [22,23]. Se trata de una rama de la electrónica que se encarga del diseño de distintos dispositivos electrónicos que se utilizan para realizar distintas mediciones. Por tanto, se puede concluir que la instrumentación electrónica abarca múltiples técnicas relacionadas en el diseño de dispositivos electrónicos cuya finalidad es el sensado y procesamiento de la información procedente de las diferentes variables físicas y químicas (variables de instrumentación) que nos rodean, y transformarlas para su posterior procesamiento.

Un sensor de temperatura es un componente electrónico que detecta y nos da información sobre la variable de instrumentación temperatura. Y se encarga del tratamiento y conversión de los distintos valores de la magnitud de entrada temperatura hacia una conversión en distintos valores de una magnitud eléctrica de salida (Figura 5.1).

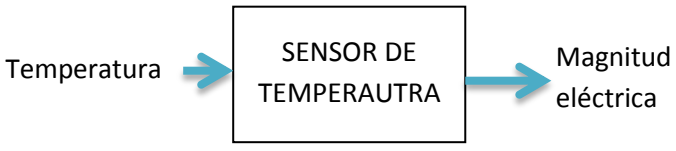


Figura 5.1: Diagrama resumen del concepto sensor de temperatura.

5.2. DESCRIPCIÓN DE VARIOS TIPOS DE SENSORES DE TEMPERATURA.

Hay múltiples sensores en el mercado con diferentes funciones. En este apartado se procederá a comentar las características de varios tipos de sensores de temperatura que se pueden utilizar en circuitos electrónicos y que se adecúan a los requisitos del proyecto.

1. Termistor.

Este sensor se adecua perfectamente a los estándares del aparato que se pretende diseñar. Básicamente un termistor es un sensor de temperatura de tipo resistivo. Su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la temperatura. Operan en un rango de -200º C a + 1000° Tendríamos que informarnos de un modelo de termistor concreto a implementar en el circuito y proceder al análisis de su funcionamiento a partir de su hoja de características. Su principal inconveniente es que la resistencia no varía de forma lineal con la temperatura, con lo cual su acondicionamiento requiere realizar un linealización previa (bien de forma analógica o más tarde de forma digital con un microprocesador). También resulta algo laborioso la calibración del cero y el fondo de escala [18].

2. Termopar.

Otro sensor que se utiliza para medir la temperatura y que hemos estudiado a lo largo del grado es el termopar. El termopar es un sensor de temperatura que se compone de dos metales diferentes, unidos en un extremo. Cuando la unión de los dos metales se calienta o enfría, se produce una tensión que es proporcional a la temperatura.

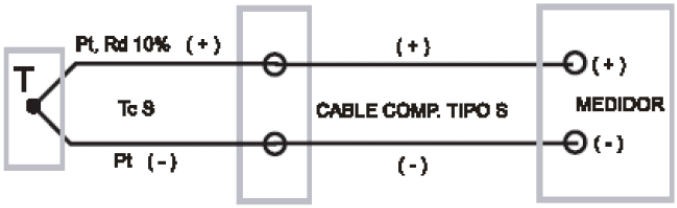


Figura 5.2: Esquema de conexión de un termopar [24].

Los termopares se clasifican siguiendo diferentes criterios como por ejemplo los materiales que constituyen el termopar, tolerancias, desviaciones existentes, etc. Comentar que los termopares más comunes son los tipo J, K y T, de los cuales el termopar tipo K es el más popular debido a su amplio rango de temperaturas y bajo coste. En la Figura 5.3 se detallan las características de estos termopares.

TIPO DE TERMOPAR	DENOMINACIÓN	COMPOSICIÓN Y SÍMBLO	RANGO DE TEMPERATURAS (ºC)	DIÁMETRO DEL ALAMBRE APROPIADO	F.E.M. (mV)
J	Hierro vs Constantán	Fe – CuNi	-200 a 600	3mm – 1mm	-7.89 a 39.13
K	Níquel-cromo vs Níquel	NiCr – Ni	0 a 1000	3mm – 2mm	0 a 41.263
T	Cobre vs constantán	Cu - CuNi	-200 a +700	0.5 mm	-5.6 a 14.86

Figura 5.3: Tabla de características de varios tipos de Termopares estandarizados. Los diferentes rangos de FEM son función de la temperatura existente [25].

3. Sensores RTD

También existen otros sensores de resistencia variable denominados RTD (Resistance Temperature Detector), están basados en la variación de la resistencia de un conductor con la temperatura. A diferencia del Termistor, la variación de Resistencia del sensor es lineal con la temperatura, según la ecuación (5.1). El sensor RTD más popular es el Pt100 (Ro=100Ω a 0°C).

$$R = R_0 \cdot (1 + \alpha \cdot \Delta T) \quad (5.1)$$

4. Sensor LM35.

Otro sensor disponible en el mercado es el denominado LM35. El LM35 es un sensor de temperatura basado en unión semiconductora [32]. A diferencia de otros dispositivos como los termistores en los que la medición de temperatura se obtiene de la medición de su resistencia eléctrica, el LM35 es un integrado que ya dispone su propio circuito de control. Proporciona una salida de voltaje proporcional a la temperatura. La salida es lineal y cada grado Celsius equivale a 10 mV. El rango de medición es de -55ºC (-550mV) a 150ºC (1500 mV).

Entre sus ventajas podemos destacar:

- Poca masa, por lo que tienen una respuesta rápida (cambios de 50ºC detectados en 1,5 – 10 s)
- Si la sonda está aislada eléctricamente, es posible medir componentes activos en funcionamiento
- El acondicionamiento es bastante sencillo

5.3. TOMA DE DECISIÓN SOBRE EL SENSOR A UTILIZAR.

En el presente apartado se presentará un análisis de los diferentes requisitos del sensor idóneo para el proyecto y se procederá a una elección, de la misma manera que en [4.1.3. Decisión

final. Matriz de decisión del microcontrolador], sobre uno de los cuatro sensores descritos anteriormente.

5.3.1. Características deseables en nuestro dispositivo

A continuación se enumerarán algunas de las características a tener en cuenta en la selección del sensor más adecuado para nuestra aplicación. En el fondo serán las características del producto final pero que dependen directamente del sensor que se vaya a utilizar. A partir de todo este conjunto de características, se elaborará posteriormente una matriz de decisión para la elección del sensor, que englobe las características más significativas.

Algunas características deseables en nuestro sensor serían:

- Leer un amplio rango de temperaturas (incluyendo temperaturas por debajo de 0ºC)
- Sea económico.
- Sensibilidad del sensor lineal (o lo más aproximado) y la mayor posible (mayor resolución): nos ahorraríamos tiempo y a la hora de calcular errores de medida.
- Corriente de alimentación (del sensor) no demasiado elevada para evitar calentamientos excesivos del sistema y tener, así, un consumo reducido.
- Precisión de medida adecuada.
- Ocupar poco espacio.
- Soportar todos los subsistemas.
- Manejable con pocas operaciones.
- Que sea de fácil calibración.
- Salida del sensor fácilmente adaptable para una entrada analógica Arduino®.

A continuación, se propondrá una matriz de decisión que optimizará la elección del sensor. Se tendrán en cuenta una serie de criterios importantes, útiles, de obligado cumplimiento etc. a la hora de elegir el sensor. A cada uno de los criterios se le asignará una importancia para el proyecto. El sensor que reúna un mayor número de las cualidades requeridas para el proyecto será el que elijamos.

5.3.2. Matriz de decisión.

Extrayendo del apartado anterior algunas de las características fundamentales tanto como para el sensor utilizado como para el dispositivo final a desarrollar, se ha propuesto la siguiente matriz de decisión [3, 14]. A continuación de cada criterio se ha establecido su importancia correspondiente para el proyecto.Finalmente, se decidirá cuál será el sensor idóneo que encaje con las expectativas de diseño del producto final. La matriz de decisión se ilustra en la Figura 5.4 y en la Figura 5.5 se muestran los valores normalizados junto con la suma ponderada y decisión final.

	Termistor	Pt100	Lm35	Termopar tipo K
Amplio rango de Temperaturas (0,07)	**9**	**5**	**9**	**9**
Económico (0,05)	**8**	**3**	**9**	**4**
Sensibilidad del sensor (0,08)	**6**	**5**	**7**	**9**
Corriente de alimentación (auto calentamiento) (0,15)	**6**	**6**	**9**	**8**
Precisión de medida (0,17)	**9**	**5**	**7**	**8**
Poco espacioso (0,04)	**9**	**5**	**9**	**6**
Soportar todos los subsistemas (0,12)	**9**	**9**	**9**	**9**
Manejable con pocas operaciones (0,1)	**3**	**7**	**9**	**7**
Fácil calibración (0,1)	**2**	**5**	**9**	**7**
Simplicidad del circuito de acondicionamiento (0,12)	**2**	**6**	**7**	**4**

Figura 5.4.: Matriz de decisión del sensor. Asignación de valores a cada criterio e importancias.

	Termistor	Pt100	Lm35	Termopar tipo K
Amplio rango de Temperaturas (0,07)	**1,00**	**0,56**	**0,78**	**1,00**
Económico (0,05)	**0,89**	**0,33**	**1,00**	**0,44**
Sensibilidad del sensor (0,08)	**0,78**	**0,56**	**0,78**	**1,00**
Corriente de alimentación (auto calentamiento) (0,15)	**0,67**	**0,67**	**1,00**	**0,89**
Precisión de medida (0,17)	**1,00**	**0,56**	**0,67**	**0,89**
Poco espacioso (0,04)	**1,00**	**0,56**	**1,00**	**0,67**
Soportar todos los subsistemas (0,12)	**1,00**	**1,00**	**1,00**	**1,00**
Manejable con pocas operaciones (0,1)	**0,33**	**0,78**	**1,00**	**0,78**
Fácil calibración (0,1)	**0,22**	**0,56**	**1,00**	**0,78**
Simplicidad del circuito de acondicionamiento (0,12)	**0,29**	**0,86**	**1,00**	**0,57**

	Termistor	Pt100	Lm35	Termopar tipo K
	0,70	0,67	0,91	0,83

Figura 5.5: Matriz de decisión del sensor. Valores normalizados y suma ponderada.

Por todo lo anteriormente expuesto, la alternativa escogida será el sensor LM35, el cual se muestra en la Figura 5.6.

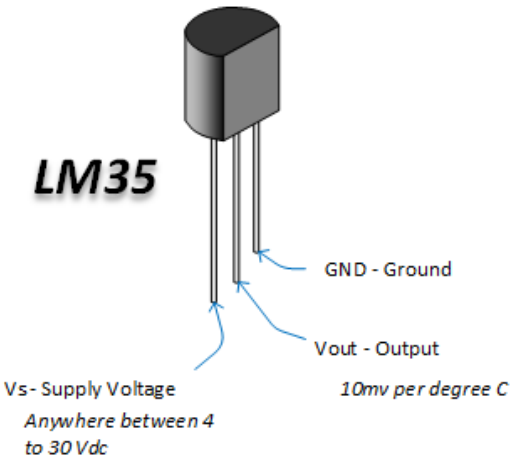


Figura 5.6. : Ilustración del sensor LM35. Conexión de sus pines [26].

5.4. SENSOR LM35. CARACTERÍSTICAS DESCRIPTIVAS Y APLICACIONES.

A continuación, se comentará desde el punto de vista técnico, algunas características de funcionamiento del sensor elegido. Todas ellas extraídas de la hoja de características del fabricante [26].

5.4.1. CARACTERÍSTICAS.

- Calibrado directamente en la escala Celsius (Centígrados)
- Factor de escala lineal 10 mV / °C
- 0,5°C de Precisión (a 25 °C)
- Calibrado para un rango temperaturas de -55 ° C a 150 ° C
- Adecuado para aplicaciones remotas
- Bajo coste debido al ajuste de la oblea del circuito
- Amplio rango de tensión de alimentación: de 4 V a 30 V
- Consumo de corriente escaso: inferior a 60 µA
- Bajo autocalentamiento: 0,08 ° C en aire estancado
- Baja impedancia de salida → 0,1 Ω para carga de 1 mA

5.4.2. APLICACIONES.

Algunos ejemplos de sus diversas aplicaciones serían:

- Fuentes de alimentación.
- Sistemas de gestión de la batería.
- Sondas para multímetros digitales.
- HVAC (Heating, ventilation and air conditioning).
- Electrodomésticos.

5.4.3. BREVE DESCRIPCIÓN DE SUS PROPIEDADES.

Los sensores de temperatura de las series LM35 son circuitos integrados de alta precisión en la variable temperatura. Su tensión de salida es linealmente proporcional a la temperatura en la escala Celsius, y está diseñado para operar entre -55 °C y 150 °C (Vout: -0.55 V a 1.5 V).

El sensor LM35 tiene una ventaja razonable sobre la línea sensores de temperatura calibrados en Kelvin, ya que no se exige al usuario el restar continuamente un valor constante de voltaje de salida para obtener la conversión a la escala Centígrada.

Además, el dispositivo LM35 no requiere calibración externa. Nos proporciona una precisión de medida de unos ±0.25°C a temperatura ambiente y ±0.75 °C de exactitud en un rango de temperatura de -55 ° C a 150 ° C.

Su bajo coste está asegurado mediante el recorte y la calibración del nivel de oblea del circuito, pues se trata de sensores de unión semiconductora. Su baja impedancia de salida, su salida lineal, y la calibración precisa del sensor, hace la interconexión a la lectura y a los circuitos de control realmente sencillo.

El dispositivo puede ser utilizado únicamente con alimentación positiva, o con alimentación positiva y negativa (mayor rango de medida de temperatura). Como consume sólo 60 µA de la alimentación, el dispositivo posee muy bajo autocalentamiento, concretamente menos de 0,1°C en aire estancado. El dispositivo dispone de diversos encapsulados para su implementación física en placa PCB: TO-92, TO-220, TO46-3 y SO8.

En la Figura 5.7 se presenta *footprint* del sensor LM35 empleado en este proyecto, cuyo encapsulado es un TO46-3.

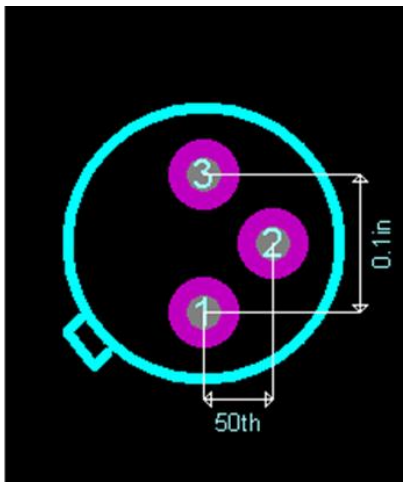


Figura 5.7. Vista previa de la implementación en placa PCB del encapsulado TO46-3 [26].

Para el dispositivo LM35, los pines de la figura con sus correspondientes conexiones serían:

Pin 1: +Vcc
Pin2: Analog output. 10mV/ºC
Pin 3: GND

Según la hoja de características del fabricante [32], la Figura 5.9 muestra la vista inferior del dispositivo seleccionado.

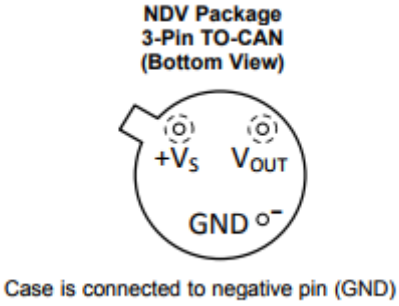


Figura 5.8. Vista inferior del dispositivo LM35 en su formato de encapsulado TO46 [26].

Por otra parte comentar que hay varias configuraciones para este sensor. Mirando la hoja del fabricante consideraremos aquella configuración que permita sensar todo el rango de temperatura para el cual se ha fabricado (-55ºC, 150ºC), lo que requiere una alimentación bipolar, como se verá en los siguientes capítulos.

5.5. ESQUEMA DEL SENSOR LM35 PARA EL RANGO DE TEMPERATURAS COMPLETO.

El rango de alimentación del sensor LM35 viene dado por la hoja de características del fabricante, y es de 4 V a 20 V. Se ha pretendido alimentar el sensor con un valor de tensión continua lo más perfecta posible (sin oscilaciones), por tanto ha sido elegida una tensión de alimentación de **5 V.** La alimentación se realizará a través de baterías de 9 V, con lo cual se requiere de un regulador para obtener una tensión de 5 V regulada.

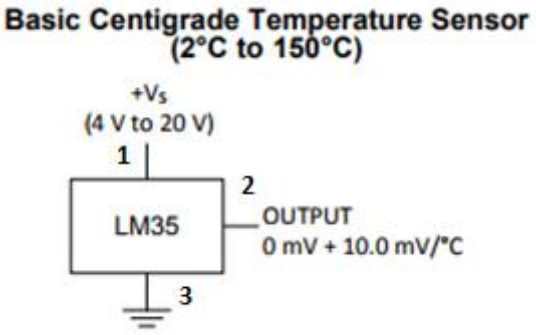


Figura 5.9: Sensor de temperatura centígrado básico. Pines y conexiones. Rangos de tensión de alimentación. [Referencia bibliografica datasheet lm35. [26]

No obstante, en nuestro proyecto se ha pretendido medir todo el rango de temperatura para el cual se ha diseñado el sensor, incluyendo también las temperaturas por debajo de los 0ºC. Por lo tanto, el esquema que se ilustra en la Figura 5.9 no es el que se pretende diseñar ya que con únicamente el sensor podría medir rangos de temperatura de 2ºC a 150ºC.

El esquema a diseñar sería la que se muestra en la Figura 5.10.

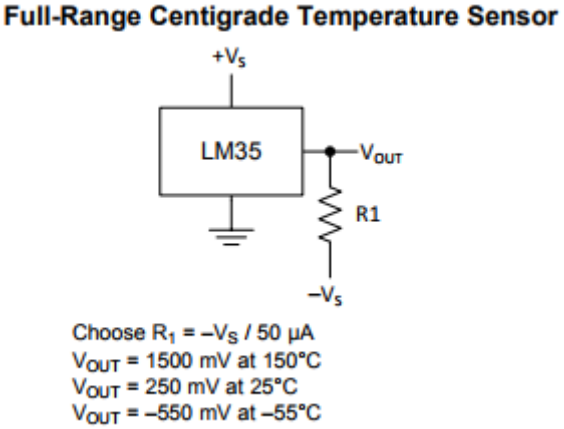


Figura 5.10: Esquema del sensor LM35. Rango de temperaturas completo. [26]

El esquema es idéntico al anterior, pero se incluirá en la salida del sensor una resistencia R1. Esta resistencia es necesaria para que el sensor mida el rango de temperaturas completo .Y su valor viene determinada por la siguiente ecuación:

$$R_1 = \frac{-Vcc}{50\mu A} \tag{5.2}$$

Y además, se conectará a un valor de tensión negativo -V_{cc} (-5V).

Como ya se ha explicado anteriormente, la tensión de alimentación será de 5 V. Sustituyendo en la ecuación (5.2), se ha obtenido un primer valor de diseño.

$$R_1 = 100\ k\Omega \tag{5.3}$$

Para concluir, comentar que se ha conseguido así que el sensor pueda medir a partir de ahora todo el rango de temperaturas completo para el cual ha sido fabricado (incluyendo las temperaturas por debajo de 0ºC).

CAPÍTULO 6. ETAPA DE AMPLIFICACIÓN DE LA SEÑAL DEL SENSOR.

En el presente capítulo se procederá a la explicación y justificación de la necesidad de una etapa de amplificación entre el sensor y la placa de Arduino®. Adicionalmente se discutirá si el convertidor AD de 10 bits interno del Arduino® seleccionado es adecuado para la presente aplicación, o si por lo contrario se requiere un convertidor AD externo que nos dé una mayor precisión en las mediciones de la temperatura.

6.1. CONEXIÓN SENSOR-ARDUINO®.

Vamos a analizar la conexión del sensor con Arduino®. A la salida del sensor, se obtiene una tensión variable con la temperatura en un rango de -0,55 V a 1,5 V. Esos valores de tensión serán leídos por el convertidor AD interno de Arduino® o un convertidor AD externo (luego se analizará), ya que como ya se ha comentado Arduino® trabaja con señales digitales.

Por tanto, parece bastante lógico pensar que se tendrá que adecuar la tensión de salida del sensor a unos valores que puedan ser interpretados por el convertidor AD, no sólo en lo que se refiere a los valores máximos y mínimos de tensión, sino también a la existencia de tensiones negativas. La conexión desde el sensor no será directamente a Arduino®, sino que se procederá a una etapa de amplificación y con nivel de *offset* de la señal que nos da el sensor de temperatura utilizado.

Por ejemplo, el convertidor interno de Arduino® es de 10 bits. Lee una entrada de voltaje de 0 V a 5 V, y por tanto si no acondicionamos la señal aproximándola lo mayor posible a los 5 V no estaremos aprovechando al 100% la resolución que nos da Arduino® y, por tanto, obtendremos errores de medida mayores. Dicho en otras palabras, estaríamos trabajando con un sistema equivalente al deseado pero se comportaría como si tuviese un número de bits inferior (menor resolución). Por tanto, se pretende que para nuestra medición se fije un límite superior de tensión 5 V y un límite inferior de 0 V. Si no se fijasen estos valores, perderíamos precisión como ya se ha explicado.

Como conclusión al razonamiento anterior, cabe comentar que ya se supone bastante lógica la utilización de una etapa de amplificación de la señal antes de utilizar Arduino®.

Por otra parte, tenemos otra complicación, Arduino® solo lee valores de tensión positivos (los valores de tensión negativos los interpretará siempre como 0 V). Pero la salida del sensor sí nos puede dar valores de tensión negativos (para temperaturas por debajo de 0ºC). Por tanto hemos de adaptar también el circuito de acondicionamiento a este requisito y así darnos valores de V_{out} siempre mayores de 0V en la salida del circuito de acondicionamiento.

Por todo lo anteriormente expuesto, se procederá en la etapa de amplificación, a la aplicación de una referencia de *offset* para salir del rango de tensiones negativas, en nuestro caso la mínima es de -550 mV.

6.2. CONVERTIDOR AD.

Por otra parte, debemos convertir la señal analógica que nos está dando el sensor a señal digital, que es como trabaja el microcontrolador. Será, por tanto, necesario utilizar un ADC. Un ADC (Analog-to-Digital Converter**)** es un sistema que convierte mediciones analógicas en correspondientes mediciones digitales, siempre codificadas sobre un número determinado de bits "N".

Estudiando las características la placa Arduino® Mega2560, obsevamos que lleva integradas 16 entradas analógicas con ADC interno de 10 bits [20].

Vamos a razonar si esta resolución nos da un error de medida adecuado, o si es demasiado grande y debemos proceder a adquirir un convertidor AD externo de mayor resolución. Esto lógicamente nos supondría un coste adicional al proyecto (su precio ronda los 15 $) y no sería interesante de realizar si no se obtuviesen unas ventajas realmente significativas en su uso.

Un ADC interesante sería por ejemplo el *ADS1115* de 16 bits (Figura 6.1). El ADS1115 se conecta con relativa sencillez a Arduino®. Además lleva integrado un amplificador de la señal tipo PGA *(programmable-gain amplifier)* que nos permitiría ajustar la ganancia desde 6,144 a 0,256. Aunque independientemente de la ganancia elegida, la máxima tensión que se podrá medir será la de alimentación, 5 V.

Esto supondrá, varias cosas. En primer lugar una gran simplificación en el circuito de acondicionamiento porque ya se dispondría directamente de un operacional con ganancia programable. Por otra parte, como ya se ha comentado anteriormente, se conseguirán unas medidas con precisiones superiores (menor error de medida). Esta ganancia también debería ser programada mediante el código de Arduino® [27,28].

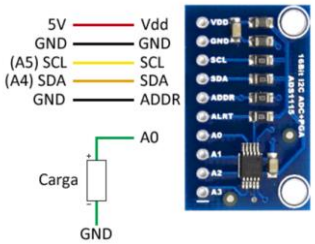


Figura 6.1: Ilustración del modelo ADS1115. Pines y conexiones para funcionamiento con Arduino [27].

En el siguiente apartado se discute si el convertidor AD que incorpora nuestra tarjeta Arduino® es adecuada para la presente aplicación (no se pierde resolución) o, por el contrario, se requiere utilizar un convertidor AD externo con un mayor número de bits (de 16 bits como en el caso de modelo ADS1115).

6.3. ERRORES DE MEDIDA.

Hemos visto en el anterior apartado la elevada simplificación que resultaría en la utilización del convertidor AD externo en nuestro circuito, ya que el circuito de acondicionamiento se reduciría considerablemente en componentes y cálculo.

En el presente apartado se procederá a la realización de varios cálculos sobre los errores de medida del circuito a implementar sin el convertidor AD externo descrito anteriormente. Si surgiesen unos errores en las mediciones elevados, se procederá al uso del convertidor previamente descrito.

Primeramente, se describirán cuáles son los errores de medida de cada componente del dispositivo final por separado y posteriormente se realizará el cálculo correspondiente al error global de medida del dispositivo a desarrollar.

6.3.1. Funcionamiento de la conversión ad.

En primer lugar, y antes de proceder al cálculo de los errores de medida del sistema se hará un pequeño análisis del funcionamiento de un convertidor AD [28]. Para poder fácilmente entender los cálculos que se realizaran a posteriori. Todo ello también con la explicación de algunas de las fórmulas que se utilizarán en los siguientes apartados para calcular los errores de medida.

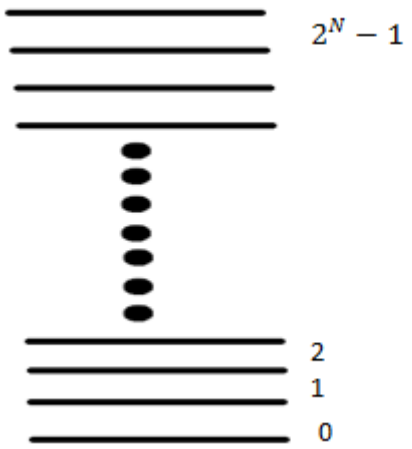


Figura 6.2: Esquema resumen de los intervalos internos presentes un dispositivo ADC de N bits.

Hay que destacar que cuando medimos en un convertidor AD una señal analógica, en el fondo no estamos midiendo todo el valor analógico en sí, sino que se procede a una clasificación

dentro de 2^N niveles o posiciones internas del convertidor tal y como se ilustra en la Figura 6.2. Lógicamente, existirán $2^N - 1$ intervalos, siendo N el número de bits del convertidor [25].

La resolución del convertidor se puede definir como el ancho de cada intervalo, efectuando dicha medida en mV. O en otras palabras, la resolución sería el voltaje a introducir en el convertidor AD, para que se produzca una variación de un bit, en la salida del convertidor (señal digital). En consecuencia, con un mayor número de bits, se poseerían un mayor número de posiciones o intervalos internos en el convertidor, menor será el ancho del intervalo, y en consecuencia mayor resolución o precisión en la medida.

En nuestro modelo de Arduino® se dispone de un convertidor interno de 10 bits. Por tanto se tiene un número de posiciones igual a $2^N - 1 = 1023$. Por tanto, a 5 V que es la tensión máxima que lee el convertidor, es decir, el voltaje que si suministrásemos en la entrada del convertidor se obtendría la conversión máxima (todas las salidas a "1"). A partir de estos datos se obtiene una resolución o ancho entre intervalos en mV de:

$$\varepsilon_{resol} = \frac{5000\ mV}{2^N - 1} = 2.44\ mV \qquad (6.1)$$

6.3.2. Errores de medida de cada componente.

En la Figura 6.3 se muestra un diagrama con cada bloque de nuestra sistema y sus correspondientes rangos de entrada.

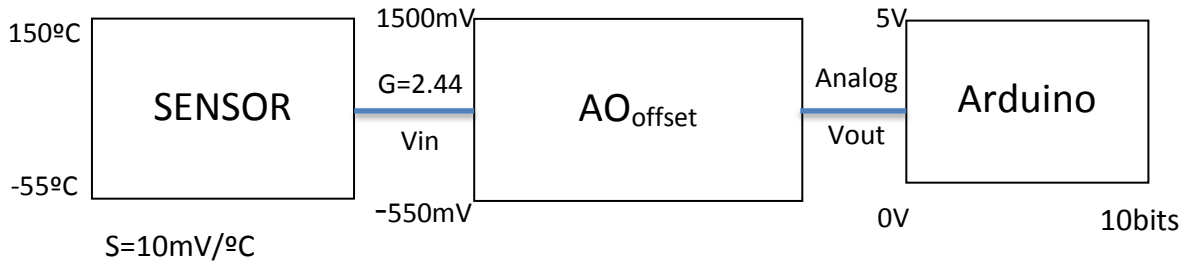


Figura 6.3. : Diagrama de bloques en componentes electrónicos significativos del dispositivo.

Los errores de medida de cada componente serían los siguientes:

- Sensor : ±0.4ºC (Según hoja de características del fabricante)

- Amplificador Operacional: Despreciamos el error de offset de este componente así como el error en la ganancia debida a la tolerancia de las resistencias utilizadas. Estas dos aproximaciones son válidas, pues el operacional utilizado es de muy bajo offset (inferior a 75 µV) y, por otro lado, para un ajuste de la ganancia y *offset* del operacional se han añadido dos potenciómetros. De esta forma, el error introducido por este componente se puede considerar despreciable.
- Convertidor interno ADC Arduino®:

$$\varepsilon_{resol} = \frac{5V}{2^N - 1} \tag{6.2}$$

$$con\ \ N = número\ de\ bits\ del\ dispositivo$$

A continuación se calculará el error de medida cometido en todo el circuito.

6.3.3. Error de medida de todo el circuito.

Seguidamente se procederá a la realización del cálculo de medida global del circuito teniendo siempre presente el esquema de la Figura 6.3. Se realizará el cálculo en ambas direcciones, donde se compararán los errores efectuados con los errores respectivos del sensor LM35 y de Arduino®. Si estos errores no se aproximan en su comparación se procederá a la utilización del convertidor AD de 16 bits previamente descrito.

Por otra parte nos faltaría averiguar la ganancia del Amplificador Operacional. Hasta que no se realizase el diseño final no podemos averiguar dicha ganancia, pero cabe comentar que de manera intuitiva ya se puede realizar un cálculo cualitativo de ella. Vamos a realizar un pequeño análisis de los componentes del circuito y procederemos al razonamiento del cálculo siguiendo una serie de pautas.

Según el esquema de la Figura 6.3 a la entrada del operacional disponemos de un rango de tensiones de entrada de -550 mV a 1500 mV. En primer lugar, lo que se desea es salir del rango de tensiones negativas. Por tanto, ya se ha explicado anteriormente que esto se realizará dando una referencia de *offset* al amplificador operacional. Por lo tanto, en la función de transferencia existirán dos términos bien diferenciados:

Un primer término referente a la elevación de tensión con referencia de *offset*. Este término servirá para modificar el rango de tensiones que se tiene en la entrada, se elevará en 550 mV siendo el nuevo rango de tensiones de 0 mV a 2050 mV. Se consigue así salir del rango de tensiones negativas.

Seguidamente, se pretende conseguir una amplificación de la señal de entrada. En la función de transferencia existirá un segundo término que será el que dictaminará la ganancia de dicho operacional. Por tanto, pasamos a continuación a una etapa de amplificación hasta llegar a una tensión máxima de 5 V para el límite superior de la tensión de entrada, 2,050 V. Así en este momento, ya se puede extraer la ganancia aproximada del Amplificador Operacional aplicando la condición anterior.

Ganancia:

$$G = \frac{V_+}{2{,}05} \tag{6.3}$$

$$con\ \ V_+ = 5V$$

Por tanto el valor de la Ganancia del Amplificador Operacional será aproximadamente de:

$$G = 2{,}44 \tag{6.4}$$

Seguidamente, se procederá ya al cálculo de los errores de medida del circuito completo siguiendo el diagrama de bloques de la Figura 6.3. Finalmente se comparará el cálculo final con el error de medida del respectivo componente:

De de derecha a izquierda:

- Convertidor interno ADC Arduino®:

$$\varepsilon_{resol} = \frac{5000mV}{2^N - 1} = 4{,}88 \text{ mV} \tag{6.5}$$

$$con\ \ N\text{: }n\acute{u}mero\ de\ bits\ del\ dispositivo\text{=}10\ bits$$

- Amplificador Operacional:

$$\varepsilon_{resol.salida.sensor} = \frac{4.88\ mV}{G} \tag{6.6}$$

$$\varepsilon_{resol.salida.sensor} = 2\ mV \tag{6.7}$$

- Sensor:

$$Sensibilidad = \frac{10\ mV}{\text{ºC}} \tag{6.8}$$

$$\varepsilon_{medida.resol.ADC} = 2\ mV \cdot \frac{1\text{ºC}}{10\ mV} = 0{,}2\text{ºC} \tag{6.9}$$

Este último valor se trata del error de medida del sistema, lo comparamos con el error del sensor LM35 extraído de su hoja de características:

$$\varepsilon_{LM35}(\text{ºC}) = 0{,}4\text{ºC} \tag{6.10}$$

Este error de medida, es el que posee como valor típico el sensor a temperatura ambiente (25ºC), tal y como se muestra en la Figura 6.4. Por tanto, se puede extraer la siguiente conclusión a los cálculos anteriores: En el fondo no aprovechamos los 10 bits de resolución que nos da el ADC interno de Arduino® porque el sensor ya tiene un error de medida mayor que el de nuestro sistema. Ya podemos adelantar que sería completamente ineficaz el uso del convertidor externo *ADS1115* de 16 bits previamente descrito.

Electrical Characteristics: LM35, LM35C, LM35D

PARAMETER	TEST CONDITIONS		LM35			LM35C, LM35D			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
Accuracy, LM35, LM35C(1)	T_A = 25°C			±0.4			±0.4		°C
		Tested Limit(2)			±1			±1	
		Design Limit(3)							
	T_A = –10°C			±0.5			±0.5		
		Tested Limit(2)							
		Design Limit(3)						±1.5	
	T_A = T_{MAX}			±0.8			±0.8		
		Tested Limit(2)			±1.5				
		Design Limit(3)						±1.5	
	T_A = T_{MIN}			±0.8			±0.8		
		Tested Limit(2)							
		Design Limit(3)			±1.5			±2	

Figura 6.4: Precisión del sensor LM35 a distintas temperaturas [26].

Para la justificación de la anterior conclusión se procede a realizar el cálculo de izquierda a derecha según la Figura 6.3 para comprobar realmente cuantos bits estamos aprovechando del convertidor AD interno de Arduino®.

- Sensor:

$$\varepsilon_{LM35}(℃) = 0.4℃ \tag{6.11}$$

$$Sensibilidad = \frac{10\ mV}{℃} \tag{6.12}$$

$$\varepsilon_{salida.sensor.LM35}(mV) = 4\ mV \tag{6.13}$$

- Amplificador operacional:

$$\varepsilon_{salida.sensor.LM35}(mV) \cdot G = 9.76\ mV \tag{6.14}$$

- Convertidor interno ADC Arduino®:

$$\varepsilon_{resol} = \frac{5000mV}{2^N - 1} = 9.76\ \text{mV} \tag{6.15}$$

$$N\text{: número de bits del dispositivo} = 9{,}001\ \text{bits} \tag{6.16}$$

Por toda la consiguiente explicación y cálculos realizados, se concluye que la resolución que tiene realmente nuestro sistema es únicamente de 9 bits. Por esta razón, resulta ciertamente inoportuna la utilización del convertidor AD externo ADS1115 ya que si nuestro sistema únicamente aprovecha 9 bits de los disponibles, es ineficaz implementar un sistema que dé resolución mayor únicamente aprovechando 9 bits, aunque se nos simplificase al máximo la estructura, cálculos y diseño del circuito de acondicionamiento a implementar.

CAPÍTULO 7. DISEÑO DEL CIRCUITO PARA EL ACONDICIONAMIENTO DE LA SEÑAL.

En el presente capítulo se establecerá, en primer lugar, un diseño de un circuito para el acondicionamiento de la señal del sensor junto con la justificación de cada uno de los componentes que se han utilizado y qué función desempeñan en dicho circuito. Seguidamente se procederá a un método de análisis del circuito diseñado para establecer así los correspondientes valores de cada uno de los componentes pasivos de dicho circuito que nos conducirán al funcionamiento deseado del mismo.

7.1. AMPLIFICADOR OPERACIONAL UTILIZADO.

Se utilizará el Amplificador Operacional OP07 (Figura 7.1), muy utilizado en los circuitos de instrumentación electrónica por su baja tensión de *offset* o desplazamiento en la entrada V_{IO}.

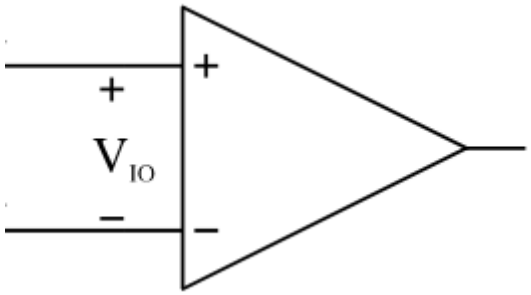


Figura 7.1. : Ilustración de la tensión de desplazamiento ("offset") V_{IO} en la entrada del Amplificador [30].

A continuación se describen algunas características adicionales de este operacional, que también son los parámetros más significativos para la toma de decisión en este tipo de diseños [30]:

- Impedancia de entrada : 45 MΩ
- Factor de rechazo al modo común CMRR (common-mode rejection ratio) : 120 dB
- Velocidad de respuesta slew rate : 0,3 V/µs

Se empleará una configuración para el Amplificador Operacional del tipo no inversor y sumador. Consiguiendo así las características deseadas de amplificación y la elevación de los valores de la tensión de entrada con referencia de offset. El circuito resultante se muestra en la Figura 7.2.

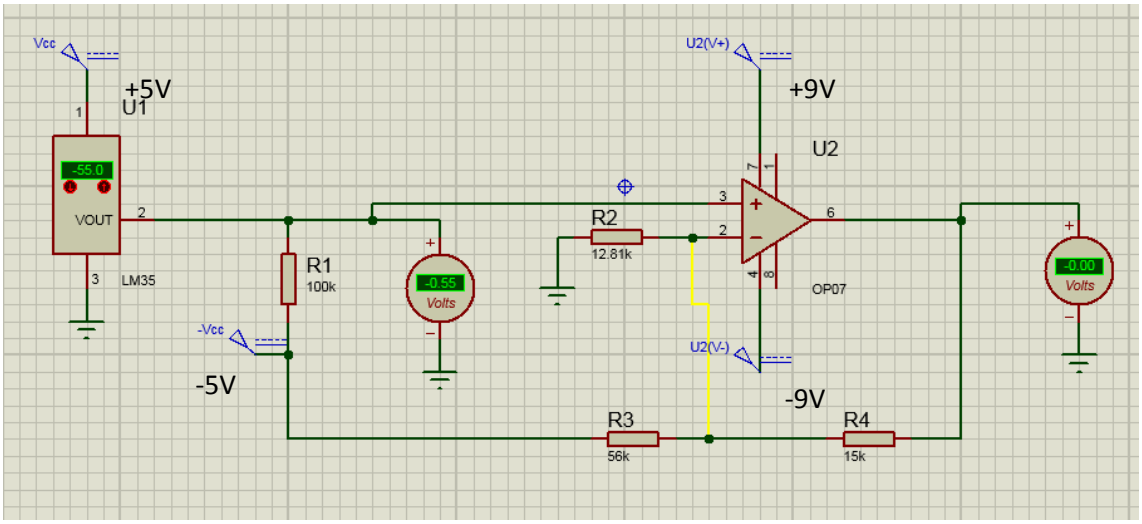


Figura 7.2: Circuito de acondicionamiento del sensor LM35.

7.2. ASPECTOS A TENER EN CUENTA EN EL DISEÑO DEL CIRCUITO DE INSTRUMENTACIÓN.

7.2.1. Alimentación del AO. Condensadores

En un primer lugar se pretendía alimentar el Amplificador Operacional con la misma Vcc que el sensor (menos complejidad electrónica). Sin embargo, nuestro amplificador operacional no puede ser alimentado por los mismos 5 V de la misma fuente que el sensor ya que obtendríamos a la salida una tensión que se saturaría en un valor determinado no pudiendo seguir amplificando la señal. Si el operacional se alimenta a 5V la máxima tensión de salida que puede darnos es de 3 V como máximo, que es la tensión de saturación del operacional con esta alimentación. Como se ha propuesto una señal de salida que varíe entre 0 V y 5 V, para aprovechar el mayor rango dinámico del convertidor AD que incorpora el Arduino®, se alimentará al operacional con los ±9 V que nos proporcionan las pilas incorporadas a la entrada.

Posteriormente se explicará la estructura del circuito de alimentación que se implementará en placa, con sus reguladores para convertir tensión de **9 V** de las baterías (Amplificador Operacional) a **5 V** (sensor y placa de Arduino®).

Por otro lado, entre las patas de alimentación del operacional y tierra se ha procedido a utilizar una serie de **condensadores** con la finalidad de evitar oscilaciones en la señal y eliminar posibles ruidos o interferencias en la entrada de la alimentación del operacional (condensadores de desacoplo).

7.2.2. Utilización de diodo Zener.

De la misma manera, se ha utilizado también un **diodo Zener** de 5,1 V como protección de nuestro Arduino®. Este Zener no deberá funcionar nunca, se ha instalado debido a que la

tensión de salida del operacional podría subir bajo algunas circunstancias (mal funcionamiento) hasta los 9 V de la alimentación, y esa tensión podría dañar el microcontrolador Arduino®. Se puede por tanto concluir que ha sido utilizado por seguridad.

La Figura 7.3 muestra la estructura completa del amplificador operacional utilizado junto con el sensor LM35 a la entrada, la alimentación simétrica (±9 V), los condensadores de desacoplo y el zener de protección de salida.

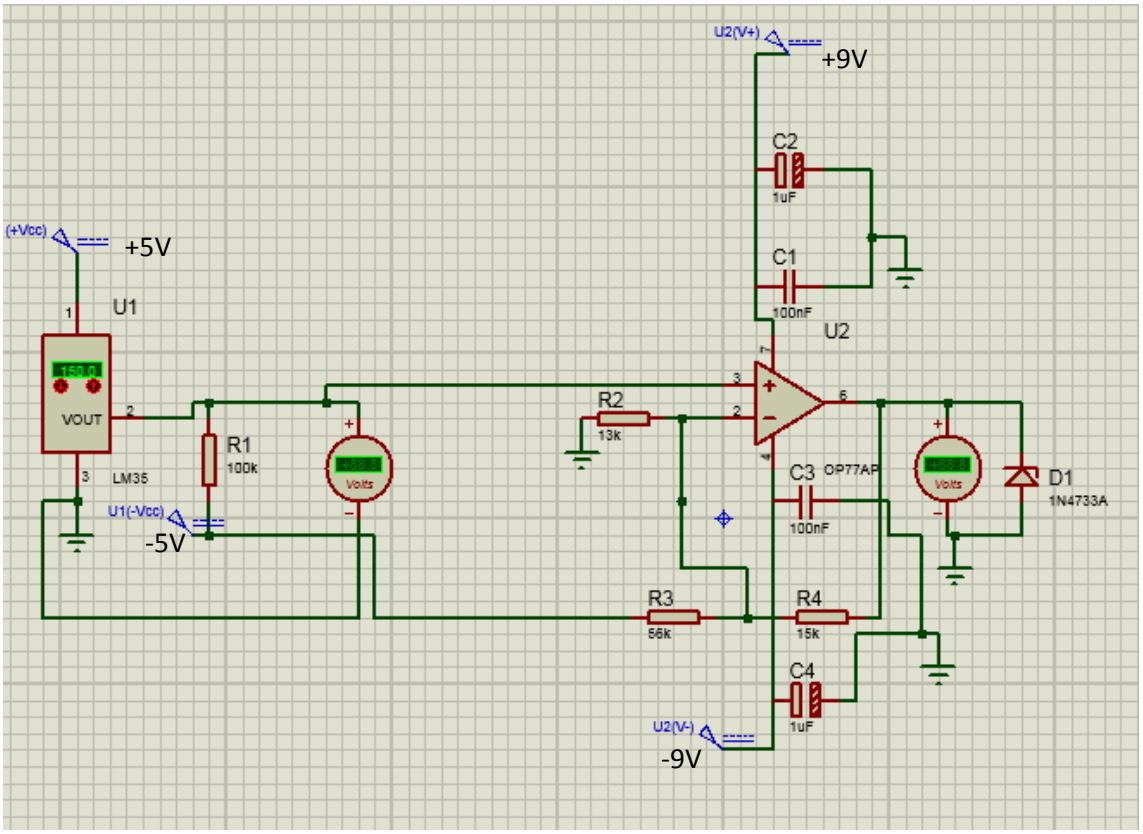


Figura 7.3: Circuito de acondicionamiento del sensor LM35. Esquema final.

7.3. ANÁLISIS DEL CIRCUITO. CÁLCULOS NUMÉRICOS EN LA ETAPA DE AMPLIFICACIÓN.

Seguidamente, se procederá al análisis el circuito de la Figura 7.5. Aplicando el método de análisis por nudos a dicho circuito se obtiene la siguiente expresión para la tensión de salida:

$$V_0 = -V_{cc} \cdot \frac{R_4}{R_3} + V_i \left(1 + \frac{R_4}{R_2} + \frac{R_4}{R_3}\right) \qquad con\ V_{cc} = -5V \tag{7.1}$$

Donde tenemos un primer término de offset: $-V_{cc} \cdot \frac{R_4}{R_3}$, y un segundo término que determina la ganancia $\left(1 + \frac{R_4}{R_2} + \frac{R_4}{R_3}\right)$.

Utilizamos la V_i mínima y máxima como valores límites dados por el sensor de temperatura (condiciones de diseño), junto a los valores de tensiones de salida a los que deseamos acondicionar la señal, así obtenemos los valores de resistencias adecuadas para estas condiciones de diseño.

$$V_i = -0.55V \longrightarrow V_0 = 0V \tag{7.2}$$

$$V_i = 1.5V \longrightarrow V_0 = 5V \tag{7.3}$$

Sustituyendo en (7.2) y (7.3) en (7.1) y agrupando términos obtenemos la siguiente relación:

$$\frac{R_3}{R_4} = 3.\overline{72} \tag{7.4}$$

Vamos a la tabla de valores normalizados para resistencias con una tolerancia del 5% y se obtienen los siguientes valores de acuerdo a la ecuación (7.4):

$$R_4 = 15k\Omega \tag{7.5}$$

$$R_3 = 56k\Omega \tag{7.6}$$

Seguidamente procedemos a calcular el valor de R_2 combinando las expresiones (7.2) y (7.3) con (7.1) y obtenemos:

$$5 = 2.05\left(1 + \frac{R_4}{R_2} + \frac{R_4}{R_3}\right) \tag{7.7}$$

Sustituyendo valores para R_3y R_4 se obtendría el valor de R_2

$$R_2 = 1.28\ k\Omega \tag{7.8}$$

Utilizamos el valor normalizado para R_2 = $13k\Omega$.

Al utilizar valores normalizados para las resistencias los valores de ganancia y offset se ajustan bastante bien a nuestras necesidades. Sin embargo, hay que tener siempre muy presente que en nuestro circuito físico ya montado en placa, con sus resistencias, condensadores, etc existen unas tolerancias en su fabricación, por ejemplo una resistencia de valor 100 kΩ, midiéndola en multímetro nos pueden aparecer 98 kΩ aproximadamente, esto es debido a que todos los componentes electrónicos poseen una determinada tolerancias. Por lo tanto, se ha prestado una especial atención a este tipo de tolerancia ya que nos podrían modificar considerablemente nuestro valor de tensión de salida de nuestro circuito, que es lo que en el fondo nosotros queremos conseguir (valores entre 0 V y 5 V para los límites del sensor).

Por todo lo anteriormente expuesto, se procederá a utilizar dos potenciómetros como sustitución de resistencias R_2 y R_3, cuyos valores serán de 50 kΩ y 100 kΩ respectivamente. Además, lo más importante, los potenciómetros servirán para ajustar el cero y el fondo de escala de nuestro circuito.

Para realizar el ajuste experimental del *offset* y ganancia de nuestro circuito de amplificación procederemos tal y como se explica en el apartado siguiente.

7.4. AJUSTE DE LOS POTENCIÓMETROS.

Para proceder al ajuste de los potenciómetros R_2 y R_3, se ha desconectado, en primer lugar, el pin número 3 de nuestro Amplificador Operacional OP07, es decir, la pata positiva de su tensión de entrada. En ese pin, se ha conectado una fuente de alimentación que se variará entre -0,55 V y a 1,5 V (límites del sensor) y se ajustarán el valor de los potenciómetros para así obtener a la salida unos valores de tensión de 0 V y 5 V respectivamente.

$$V_i = -0.55V \longrightarrow V_0 = 0V \quad (7.2)$$

$$V_i = 1.5V \longrightarrow V_0 = 5V \quad (7.3)$$

En las Figuras 7.4 y 7.5 se muestra las fotografías de los ajustes experimentales del *offset* y el fondo de escala respectivamente [31,32].

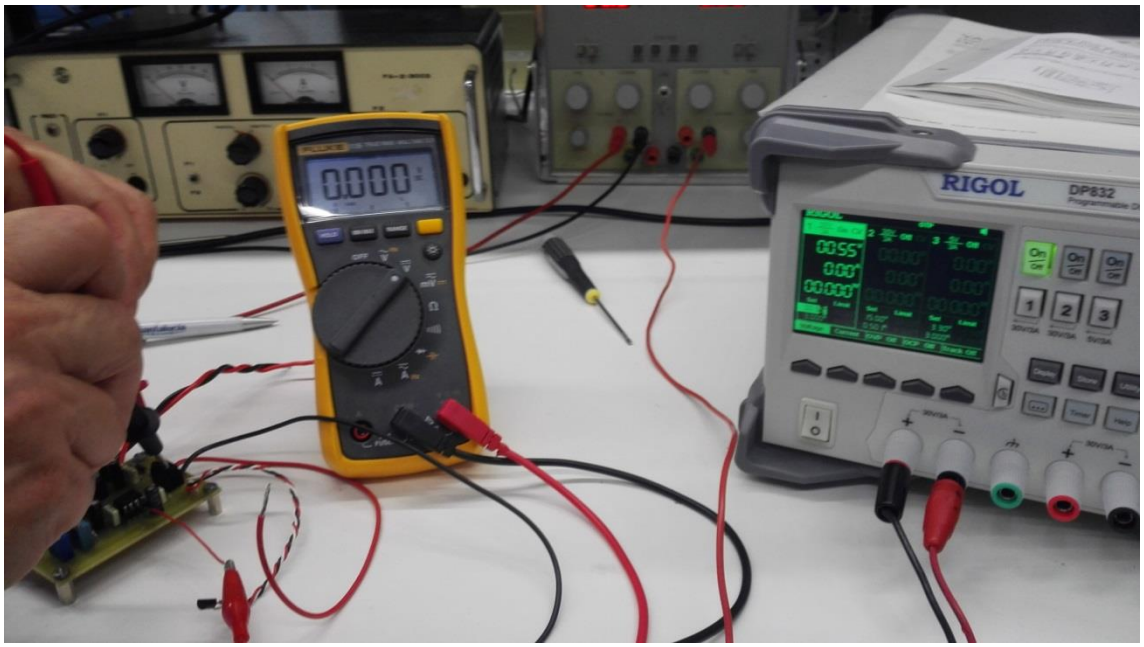


Figura 7.4: Valor de la tensión de salida con una entrada al AO de -0.55V.

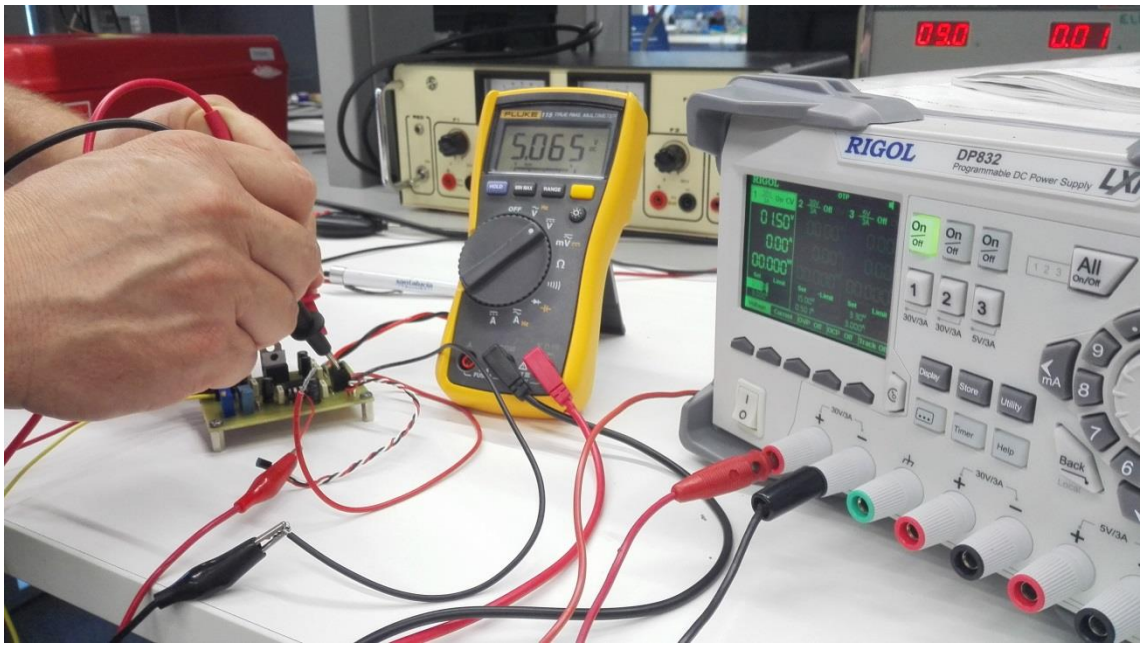


Figura 7.5: Valor de la tensión de salida con una entrada al AO de -0.55V.

CAPÍTULO 8. IMPLEMENTACIÓN FÍSICA DEL DISEÑO. PLACA PCB

En este capítulo se detalla la implementación de la placa de circuito impreso que incluye las baterías, los reguladores de tensión, el sensor, el amplificador y los conectores de entrada y salida.

8.1. ETAPA DE ENTRADA. ALIMENTACIÓN DEL CIRCUITO.

En el presente apartado se comentará, como será nuestra alimentación para el circuito a diseñar, teniendo en cuenta que se requiere una alimentación simétrica (positiva y negativa). Comentar que este apartado, pese a su importancia, no se ha descrito anteriormente en el diseño del circuito, porque resulta innecesario conocer la estructura de la alimentación para realizar los cálculos.

En primer lugar, se debe destacar que las tensiones de alimentación del circuito diseñado son de ±9 V para el operacional y ±5 V para el sensor. Los ±9 V se obtendrán mediante dos pilas de 9 V de tensión conectadas en serie, con el punto central conectado a tierra. La salida de ±5 V se obtendrá a partir de esta mediante dos reguladores de tensión, en concreto se han utilizado los reguladores L7805 y L7905 para los +5 V y -5V, respectivamente. En la Figura 8.1 se muestra el esquema de conexionado de dicha etapa de entrada [33].

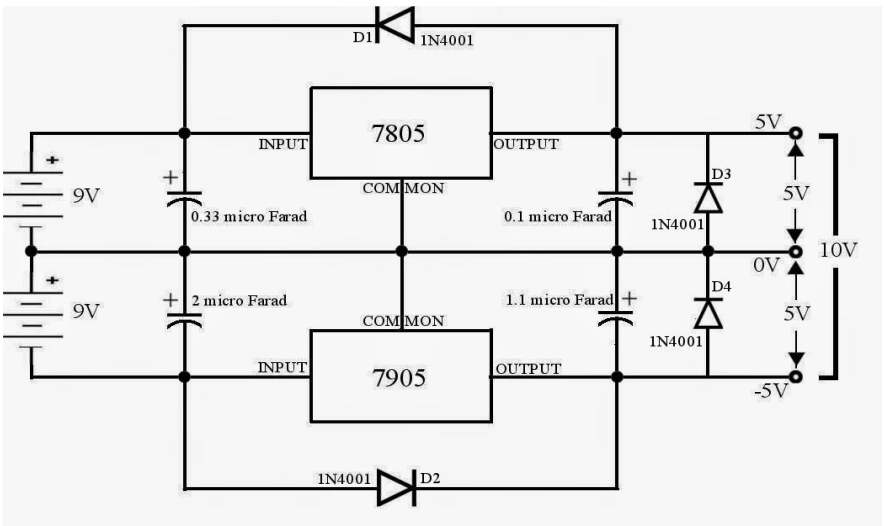


Figura 8.1: Ilustración de la etapa de entrada del circuito [33].

8.2. IMPLEMENTACIÓN FÍSICA DEL CIRCUITO EN PLACA MEDIANTE ORCAD.

Para implementar el circuito de manera física se ha optado por realizar una placa en formato PCB, ya que es la forma de circuito integrado donde todos los componentes electrónicos y cables quedan más compactos y organizados, ocupando el mínimo espacio posible.

Se ha utilizado en primer lugar para el diseño de nuestra placa el software ofimático Orcad [34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. Donde los pasos a seguir para obtener nuestra placa son los siguientes:

1. Esquema del circuito.
2. Organización en placa de los componentes.
3. Circuito impreso PCB.
4. Placa.

8.2.1. Esquema del circuito.

Dentro de la plataforma Orcad, para implementar el esquema de nuestro circuito, hemos utilizado el programa Orcad Capture, donde se irán añadiendo los diferentes componentes electrónicos. No importando demasiado el valor de los componentes en sí, sino la forma, es decir, si nuestro regulador tiene 3 pines (consultando el *datasheet* del fabricante) colocaremos un componente con 3 pines (por ejemplo un conector), ya que no todos los modelos del mercado están guardados en Orcad. Más adelante con el *footprint* (huella) ya se determinará la forma física de cada componente, distribución de pines, distancia entre pines, tamaño etc. Por tanto, el resultado del esquema del nuestro circuito de instrumentación en Orcad Capture quedaría tal y como se muestra en la Figura 8.2.

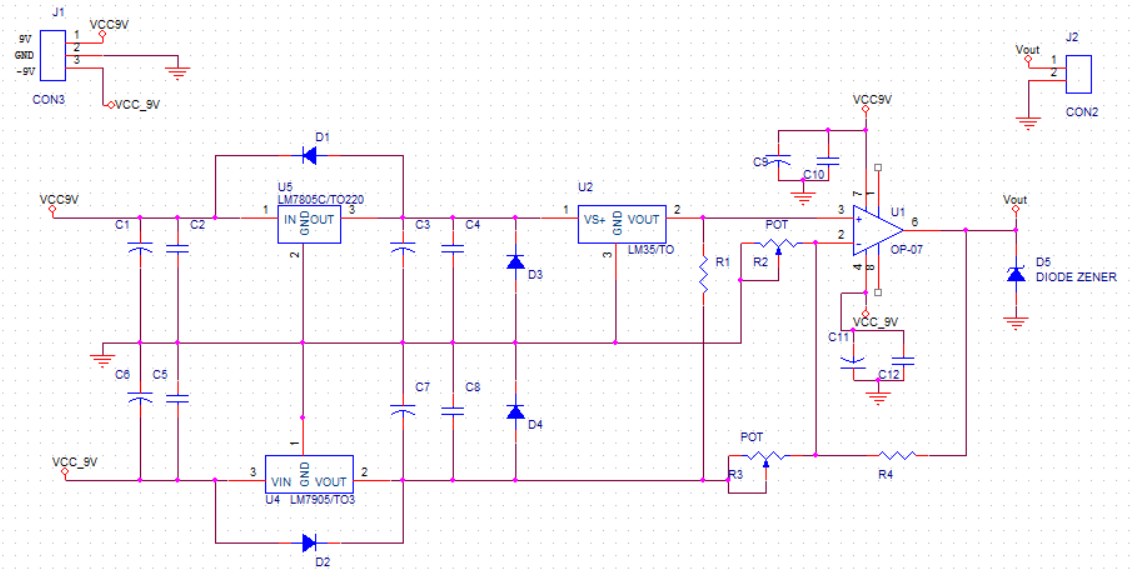


Figura 8.2. : Ilustración en Orcad capture del circuito diseñado.

Seguidamente tendremos que determinar la forma física de cada componente, es decir su *footprint*. Orcad Capture por defecto no añade *footprint* alguno a los componentes, por tanto hemos de añadirlo nosotros (Figura 8.3).

		Location X-Coordinate	Location Y-Coordinate	Name	Part Reference	PCB Footprint
1	SCHEMATIC1 : PAGE1	260	190	INS14025	U5	
2	SCHEMATIC1 : PAGE1	250	Location X-Coordinate	INS415	U4	
3	SCHEMATIC1 : PAGE1	500	190	INS91	U2	
4	SCHEMATIC1 : PAGE1	720	190	INS50	U1	
5	SCHEMATIC1 : PAGE1	740	380	INS6365	R4	
6	SCHEMATIC1 : PAGE1	630	380	INS6739	R3	
7	SCHEMATIC1 : PAGE1	640	210	INS6593	R2	
8	SCHEMATIC1 : PAGE1	590	230	INS4315	R1	
9	SCHEMATIC1 : PAGE1	890	70	INS16792	J2	
10	SCHEMATIC1 : PAGE1	70	50	INS15233	J1	
11	SCHEMATIC1 : PAGE1	840	230	INS7912	D5	
12	SCHEMATIC1 : PAGE1	440	340	INS3791	D4	
13	SCHEMATIC1 : PAGE1	440	220	INS3577	D3	
14	SCHEMATIC1 : PAGE1	270	450	INS2632	D2	
15	SCHEMATIC1 : PAGE1	270	140	INS2532	D1	
16	SCHEMATIC1 : PAGE1	750	310	INS10531	C12	
17	SCHEMATIC1 : PAGE1	710	310	INS10517	C11	
18	SCHEMATIC1 : PAGE1	690	150	INS9269	C10	
19	SCHEMATIC1 : PAGE1	650	150	INS9255	C9	
20	SCHEMATIC1 : PAGE1	390	350	INS663	C8	
21	SCHEMATIC1 : PAGE1	350	350	INS554	C7	
22	SCHEMATIC1 : PAGE1	140	340	INS535	C6	
23	SCHEMATIC1 : PAGE1	170	340	INS644	C5	
24	SCHEMATIC1 : PAGE1	390	220	INS607	C4	
25	SCHEMATIC1 : PAGE1	350	220	INS516	C3	
26	SCHEMATIC1 : PAGE1	170	220	INS588	C2	
27	SCHEMATIC1 : PAGE1	140	220	INS497	C1	

Figura 8.3.: Ilustración de las propiedades de todos los componentes del circuito. Footprints inicialmente vacíos.

El *footprint*, es decir, la huella de cada componente, se puede consultar también en datasheet del fabricante, una vez allí se nos indica la referencia que el componente tiene para PCB, un ejemplo sería la referencia TO-92 como encapsulado; donde esta nomenclatura ya tiene estandarizada la forma del componente (tamaño, distribución de pines, polaridades etc.).

A continuación procederíamos a abrir el programa *Layout* del software Orcad para ver qué librerías contiene y comprobar si los *footprints* estandarizados que da Orcad podríamos adecuarlos a nuestros componentes electrónicos, o si por el contrario debemos modificar las librerias de layout para adaptarlas a nuestros componentes en su forma, distribución de pines, distancia entre pines etc. para poder así obtener una organización en placa más ordenada y eficiente.

Se nos facilita el acceso a las librerías de Orcad Layout de la universidad, donde se posee una buena configuración para cada uno de los componentes que se van a implementar.

A continuación seleccionamos los componentes de nuestro circuito dentro del "Library Manager" de Layout y seguidamente copiamos y pegamos los *footprints* obtenidos a Orcad Capture. A partir de este momento Orcad ya interpreta cual es la forma (encapsulado) de cada componente electrónico del circuito diseñado, para así poder realizar su posterior impresión sobre la placa PCB.

		Location X-Coordinate	Location Y-Coordinate	Name	Part Reference	PCB Footprint
1	SCHEMATIC1 : PAGE1	140	220	INS497	C1	CE10UF/25V
2	SCHEMATIC1 : PAGE1	170	220	INS588	C2	CP4
3	SCHEMATIC1 : PAGE1	350	220	INS516	C3	CE10UF/25V
4	SCHEMATIC1 : PAGE1	390	220	INS607	C4	CP4
5	SCHEMATIC1 : PAGE1	170	340	INS644	C5	CP4
6	SCHEMATIC1 : PAGE1	140	340	INS535	C6	CE10UF/25V
7	SCHEMATIC1 : PAGE1	350	350	INS554	C7	CE10UF/25V
8	SCHEMATIC1 : PAGE1	390	350	INS663	C8	CP4
9	SCHEMATIC1 : PAGE1	650	150	INS9255	C9	CE10UF/25V
10	SCHEMATIC1 : PAGE1	690	150	INS9269	C10	CP4
11	SCHEMATIC1 : PAGE1	710	310	INS10517	C11	CE10UF/25V
12	SCHEMATIC1 : PAGE1	750	310	INS10531	C12	CP4
13	SCHEMATIC1 : PAGE1	270	140	INS2532	D1	DIODO03
14	SCHEMATIC1 : PAGE1	270	450	INS2632	D2	DIODO03
15	SCHEMATIC1 : PAGE1	440	220	INS3577	D3	DIODO03
16	SCHEMATIC1 : PAGE1	440	340	INS3791	D4	DIODO03
17	SCHEMATIC1 : PAGE1	840	230	INS7912	D5	DO41_ZENER
18	SCHEMATIC1 : PAGE1	70	50	INS15233	J1	WEID3
19	SCHEMATIC1 : PAGE1	890	70	INS16792	J2	WEID2
20	SCHEMATIC1 : PAGE1	590	230	INS4315	R1	R1W4
21	SCHEMATIC1 : PAGE1	640	210	INS6593	R2	POTMULTI
22	SCHEMATIC1 : PAGE1	630	380	INS6739	R3	POTMULTI
23	SCHEMATIC1 : PAGE1	740	380	INS6365	R4	R1W4
24	SCHEMATIC1 : PAGE1	720	190	INS50	U1	DIP8S
25	SCHEMATIC1 : PAGE1	500	190	INS91	U2	TO92_LM35
26	SCHEMATIC1 : PAGE1	250	390	INS415	U4	79XX
27	SCHEMATIC1 : PAGE1	260	190	INS14025	U5	78XX

Figura 8.4.: Ilustración de las propiedades de todos los componentes electrónicos. Footprints ya rellenados manualmente con sus correspondientes referencias.

8.2.2. Organización en placa de los componentes.

Generación del archivo .MAX.

Los archivos .MAX son los que utiliza Orcad Layout para generar las placas de circuito impreso. Para generar el archivo .MAX necesitamos el esquema de nuestro circuito creado a partir del Orcad Capture, previamente actualizadas las propiedades de cada componente y añadidos los correspondientes *footprints*. Seguidamente se exportará al software Orcad Layout, procediendo en la opción Create Netlist en la barra de herramientas del Capture. Si no hay errores en los *footprints* ya se tendría nuestro archivo .MAX disponible para imprimir el circuito en la placa.

Una vez dentro de Orcad Layout con el .MAX generado, el aspecto es de todos los componentes muy desordenados y mal distribuidos. El objetivo actual será organizar ya en placa (a tamaño real) la distribución de cada uno de nuestros componentes. Debemos de seguir una serie de directrices muy importantes:

- Ocupar el mínimo espacio posible (nos resultará más económico)
- Los componentes que estén cerca en nuestro esquema electrónico, es recomendable que deban también estarlo en la placa.
- Organizar los componentes pensando ya en la posterior generación de pistas, de tal manera que intentemos sigamos el camino más cercano, a ser posible en linea recta
- Trazar el mínimo número de pistas siguiendo el camino más cercano.

La organización en placa del circuito diseñado se ha hecho siguiendo las directrices anteriores y se ha quedado tal y como se muestra en la Figura 8.5, donde se ha optado por trazar pistas únicamente en la capa trasera. Además, se ha tomado la precaución de situar los condensadores de desacoplo, descritos en el capítulo anterior, lo más cerca posible del correspondiente integrado para así reducir el ruido lo máximo posible.

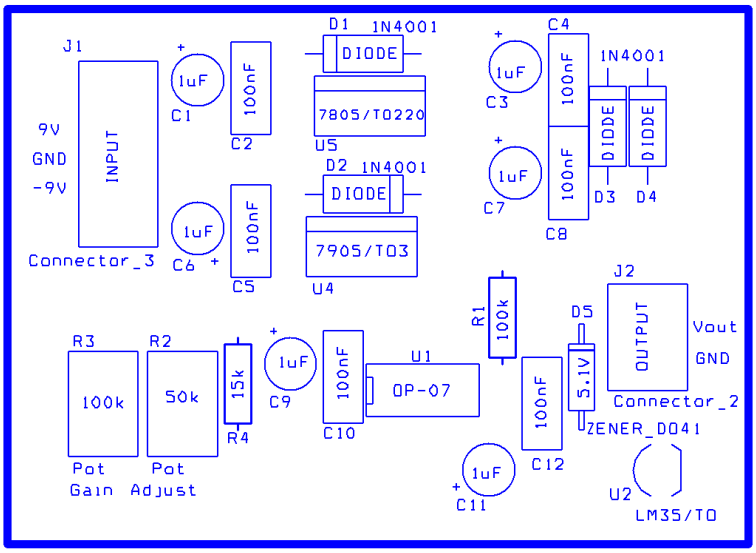


Figura 8.5.: Organización de los componentes en del circuito en placa.

Seguidamente se procederá a la trazada de las pistas. Sería interesante como ya se ha comentado anteriormente, que los componentes con conexiones próximas en el circuito de diseño también lo estén en placa para que las pistas se tracen con el camino más recto posible. El resultado del trazado de dichas pistas se ilustra en la Figura 8.6.

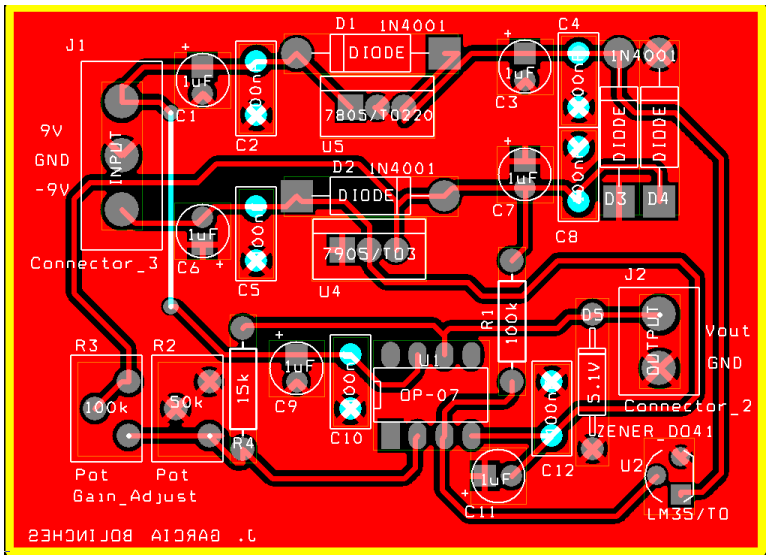


Figura 8.6.: Componentes del circuito en placa junto con el correspondiente trazado de pistas.

8.2.3. Circuito impreso PCB.

A continuación se imprimirá el diseño realizado con Orcad layout para la implementación sobre la placa con diferentes disolventes químicos presentes en el laboratorio. El aspecto final del diseño es el mostrado en la Figura 8.7.

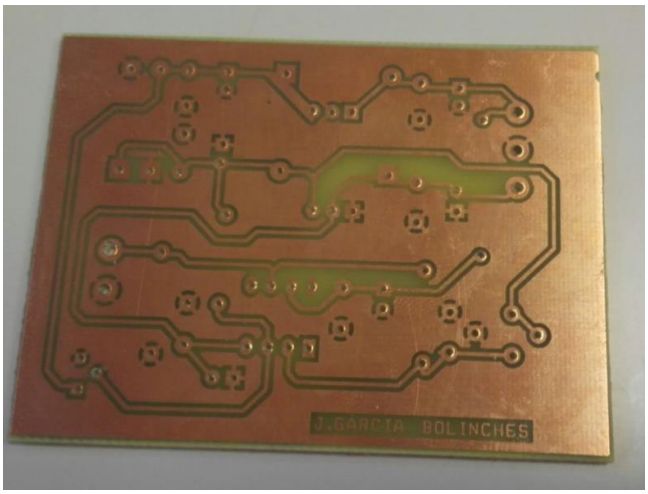


Figura 8.7. : Circuito impreso en placa PCB.

A partir de este momento, solo queda taladrar los huecos donde se introducirán los pines de cada uno de los componentes. Posteriormente se procederá a la soldadura de estos con estaño (Figura 8.8).

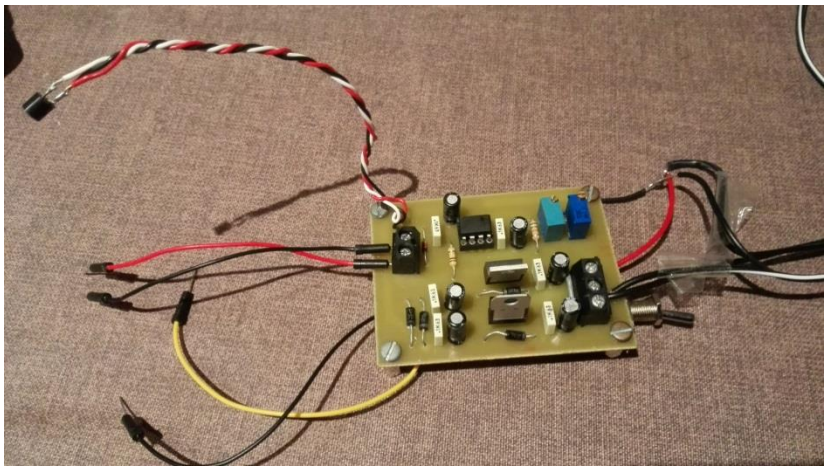


Figura 8.8. : Aspecto final del circuito en placa PCB con todos sus componentes instalados.

CAPÍTULO 9. ETAPA DE CONEXIÓN A ARDUINO®.

En el presente capítulo se comentará como se ha establecido la conexión del circuito a la placa Arduino®. Se mostrará el código utilizado para almacenar los datos de temperatura en memoria y finalmente se mostrarán algunos resultados obtenidos. Este último apartado de resultados corresponde a la visualización mediante el programa LabView de la temperatura almacenada durante 24 h y que han sido cedidos por un TFG complementario a este.

9.1. PRIMERAS CONEXIONES.

En primer lugar, comentar que se ha instalado adicionalmente un interruptor entre la alimentación y la entrada del circuito. Este interruptor cumple la función de cortar o poner en marcha la alimentación de dicho circuito cuando se desee. Es decir, cuando se pretenda iniciar o detener la toma y almacenamiento de los valores de temperatura.

El microcontrolador se alimentará a 5 V como ya se ha comentado a lo largo de la memoria. Por otro lado, la salida de nuestro circuito se conectará con una de las entradas analógicas (entrada A0) de nuestro Arduino®®. La Figura 9.1 muestra la fotografía de dicha conexión.

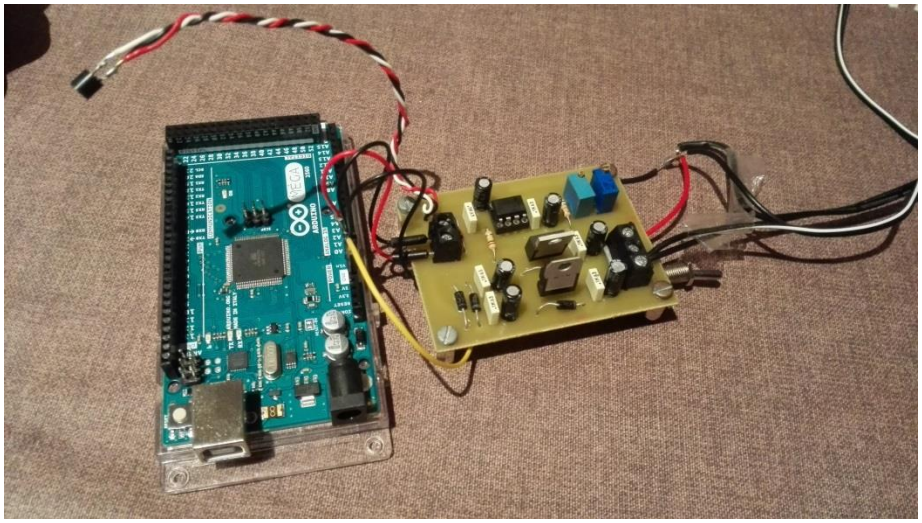


Figura 9.1: Ilustración del dispositivo acabado.

9.2. CÓDIGO ARDUINO® PARA EL ALMACENAMIENTO DE DATOS EN MEMORIA.

Comentar que se ha inicializado el almacenamiento a partir del puesto de memoria número 200, ya que, esos 200 primeros lugares de la memoria son destinados para guardar algunos parámetros complementarios tales como la fecha, hora, día, frecuencia de muestreo etc. Todo esto para que aparezca plasmado en su posterior vista de las mediciones con LabView, como se verá en las Figura 9.3.

Por otro lado también resulta interesante hacer referencia que al iniciar el programa se utiliza una espera de 5 segundos antes de entrar en el bucle por la posible existencia de transitorios en nuestro circuito electrónico. Así que podría existir algún tipo de imprecisión en la primera medida de temperatura captada si no se tuviese esto en cuenta. Se puede asegurar que a partir de dicha espera las medidas ya serán las deseadas.

Comentar también que el programa se ha implementado en la parte del void Setup por una sencilla razón: Arduino® solo lee una vez esta parte del programa. Se ha procedido a dejar el void loop (bucle infinito) vacío. Ya que si se hubiese programado el código en void loop como se hace de manera habitual seguiríamos escribiendo en memoria continuamente incluso cuando estuviese llena empezando de nuevo a escribir datos desde la posición inicial, situación claramente indeseable.

El programa diseñado se ha realizado con la frecuencia de muestreo de 1 dato por minuto con una duración total de recogida de muestras de 24 h (un total de 1440 muestras al día). Esta frecuencia de muestreo se puede modificar a petición del usuario.

El programa desarrollado se muestra en la Figura 9.2.

```
#include <EEPROM.h>
#include <LowPower.h>

const int sensorPin=A0;  //seleccionar entrada del sensor
unsigned int Temp,d,m,b,n,i; //16 bits
//unsigned char 8 bits
int eeAddress=200;//16 bits
byte H,L;
void setup()
{
Serial.begin (9600); //Inicializa Puerto serie
delay (5000); //Esperar antes de entrar en el bucle por si existiesen transitorios, pueden fallar las primeras medidas.
for (n=0; n<1440; n++)
{
  Temp=analogRead(sensorPin); //16bits lecytura sensor
H=highByte(Temp);
L=lowByte(Temp);
EEPROM.write(eeAddress,H);
EEPROM.write(eeAddress+1,L);
eeAddress= eeAddress+2 ;


for (i=0; i<16; i++){
LowPower.powerDown(SLEEP_4S, ADC_OFF,BOD_OFF);} //Apagamos ADC y el circuito BOD durante un minuto de espera entre cada medición.
}

}
void loop ()
{
```

Figura 9.2. : Código Arduino® para almacenar en memoria EEPROM los datos adquiridos.

9.3. COMPROBACIÓN DE UN CORRECTO ALMACENAMIENTO.

En el presente apartado se demostrará que los datos se han guardado en la memoria correctamente tal y como se ha propuesto en el anterior código. Esto se realizará leyendo los datos existentes en memoria.

Se ha programado y se ha dejado actuar el dispositivo durante 24 horas en una vivienda, para un almacenamiento en memoria de los valores de las temperaturas existentes.

Se ha utilizado el programa Labview para una visualización más clara de los datos de temperatura captados. Las Figuras 9.3 y 9.4 muestran los resultados de varios ensayos.

En la Figura 9.3 se muestra la evolución de la temperatura en la habitación de una determinada vivienda. Comentar que existe un pequeño salto sobre las 12 horas aproximadamente. Esto es debido a que ese punto representa el inicio de la medición y el final de la medición el día siguiente.

La Figura 9.4 muestra la evolución de la temperatura en el interior de un frigorífico de uso doméstico. Los saltos de temperatura corresponden a los instantes en que se abre la puerta de dicho frigorífico.

Por todo lo anteriormente expuesto, se puede concluir con que la toma y almacenamiento de datos ha sido coherente. Y en efecto, ha funcionado según lo previsto.

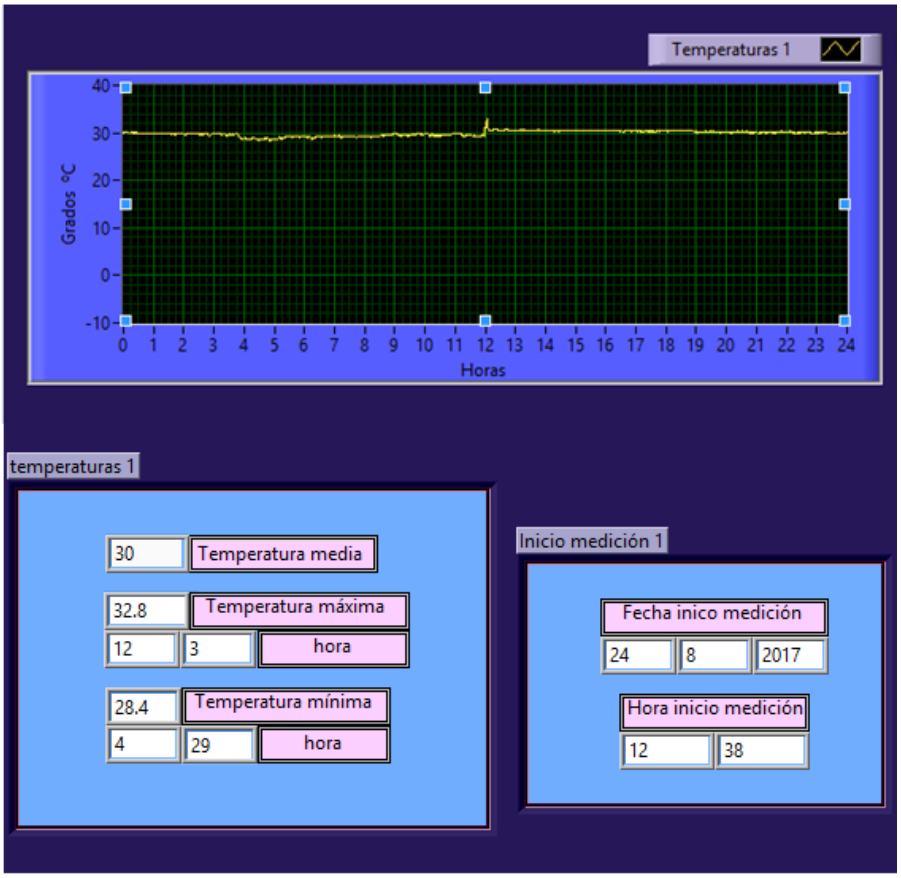


Figura 9.3. : Gráfica de la variación de la temperatura durante 24 horas en una vivienda [46].

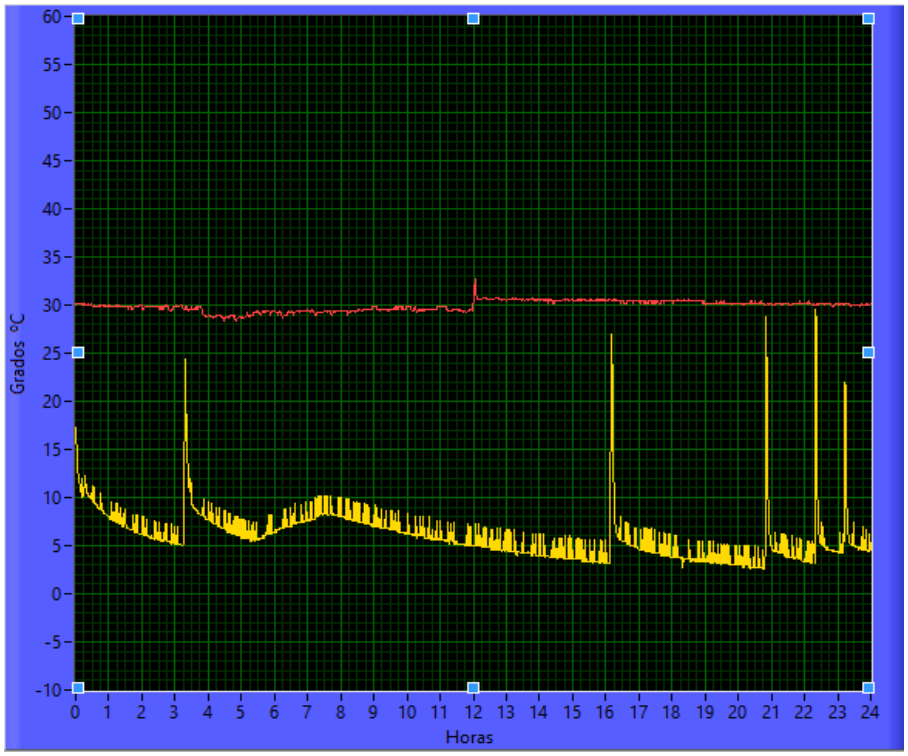


Figura 9.4. : Grafica de la variación de la temperatura durante 24 horas cuando el equipo diseñado se ha introducido en un frigorífico [46].

9.4. CONSUMO Y AUTONOMIA.

En el presente apartado se realizarán una serie de mediciones para determinar el consumo final de nuestro dispositivo para así finalmente emplear la batería con la capacidad que resulte más adecuada. Comentar que se desea una autonomía mínima de 24 horas tal y como se ha descrito en el alcance del proyecto [1.2. ALCANCE DEL PROYECTO].

Comentar que a lo largo del desarrollo del proyecto, se han tomado varias medidas para obtener un consumo total del dispositivo lo más reducido posible. Por ejemplo, en el diseño del circuito se ha optado por seleccionar resistencias del orden de $10\ k\Omega$ a $100\ k\Omega$.

Por otro lado, en el código de programación de Arduino® se ha utilizado la librería <<Low-Power.h>> [45] tal y como se ha podido observar en la Figura 9.2. Esta librería contiene funciones que reducen considerablemente el consumo del mismo. Se ha optado por detener durante el tiempo de espera entre toma de datos, los componentes de la placa con mayores consumos (el ADC y el circuito BOD). Está demostrado que estas funciones reducen considerablemente el consumo de Arduino® respecto con la función típica de espera “delay (msec)” El ensayo para determinar el consumo de nuestro dispositivo queda ilustrado en la Figura 9.5, siendo de 48 mA

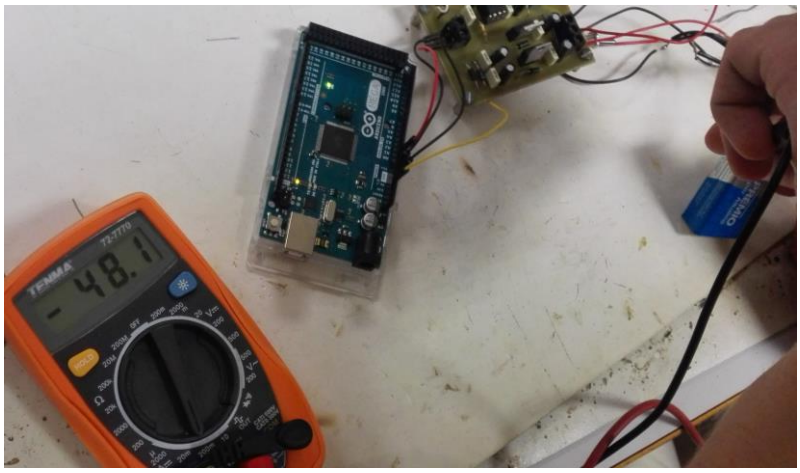


Figura 9.5: Consumo total de nuestro dispositivo.

9.4.1. Cálculo de la capacidad de la batería para obtener la autonomía estipulada.

Al buscarse una autonomía de 24 horas y suponiendo linealidad en el proceso de descarga de la batería, la capacidad de la pila a emplear será de:

$$\frac{Capacidad\_batería\ (mA \cdot h)}{48\ (mA)} \geq 24\ horas \tag{9.1}$$

$$Capacidad\_batería \geq 1152\ mA \cdot h \tag{9.2}$$

El modelo de pila a emplear se ilustra en la Figura 9.6.

Figura 9.6: Batería de la compañía Ultra Life, 9 V, PP3, Dióxido de Manganeso-Litio, 1200 mA·h. [44]

9.4.2. Cálculo de la autonomía del dispositivo.

$$Autonomía = \frac{Capacidad\_batería\ (mA \cdot h)}{Consumo(mA)} = \frac{1200}{48} = 25\ horas \tag{9.3}$$

CAPÍTULO 10. CONCLUSIONES.

En el presente TFG se ha diseñado un dispositivo que permite adquirir los datos de temperatura de un determinado recinto cada minuto durante un espacio de tiempo de 24 h u otra frecuencia de muestreo y tiempo de almacenamiento que pueden ser configurados fácilmente por el usuario. Los datos de temperatura adquiridos son almacenados en memoria para su posterior visualización en un PC mediante el entorno de programación de LabView, aunque este último apartado de visualización corresponde a otro TFG que complementa este.

El sistema desarrollado se alimenta a través de 2 baterías de 9 V, sin necesidad de estar conectado a la red eléctrica. Incluye un interruptor que inicia el proceso de adquisición de los datos, evitando un consumo innecesario del circuito cuando este no está en funcionamiento. En primera instancia se ha seleccionado el sensor más adecuado a esta aplicación, el LM35 por su reducido tamaño, coste, simplicidad en su circuito de acondicionamiento y resolución adecuada para esta aplicación. Para el procesamiento y almacenamiento de los datos de temperatura se ha seleccionado la placa de desarrollo de Arduino® MEGA2560, por su facilidad en el manejo del hardware, su simplicidad en la programación y su reducido tiempo de aprendizaje, lo que redunda en un reducido tiempo de desarrollo. El microcontrolador que incluye esta placa de Arduino® dispone de entradas digitales de 10 bits, lo que supone una precisión suficiente para el sensor de temperatura seleccionado, no siendo necesaria la inclusión de un convertidor analógico-digital externo. Para el acondicionamiento de la tensión de salida del sensor de temperatura se ha optado por un amplificador sumador-no inversor mediante operacional OP07 con bajo nivel de offset, válido para aplicaciones de acondicionamiento electrónico. En dicho circuito de acondicionamiento se han incluido dos potenciómetros que permiten ajustar la ganancia y offset del circuito. Estos ajustes son necesarios puesto que los componentes electrónicos utilizados tienen una determinada tolerancia.

Las resistencias del circuito de acondicionamiento se han seleccionado de valor suficientemente grande para que los consumos sean lo más reducidos posible. Todo esto resulta fundamental para que la batería seleccionada presente el mayor tiempo de autonomía posible. Adicionalmente se ha seleccionado un modelo de Arduino® de bajo consumo y con una programación que permite reducir estos tiempos de consumo mediante la función "*sleep"* de la librería "Low-Power Master". Todo esto ha supuesto un consumo promedio de la placa Arduino® de 30 mA. Seleccionado baterías de 1,2 A·h, esto nos permite una autonomía de 25 horas.

En el presente TFG he puesto en práctica los conocimientos adquiridos a lo largo del grado, por ejemplo: Conocimiento sobre sensores, sobre circuitos de acondicionamiento analógico basado en operacional (diseño analógico), así como la programación de un microcontrolador (diseño digital). Se han manejado las hojas de características de los distintos componentes utilizados, viendo que éstos presentan unas determinadas prestaciones, rangos de operación, determinadas tolerancias y sensibilidades etc. Se ha aprendido a realizar el diseño PCB de estos circuitos mediante Orcad, así como la implementación práctica de dicha placa de circuito impreso y se ha realizado la soldadura de todos los componentes en la placa.

Una vez montados todos los componentes en la placa de circuito impreso se ha ido comprobando el correcto funcionamiento de cada bloque del circuito y se han ido depurando los errores de diseño y montaje.

Finalmente se ha probado el equipo diseñado en dos casos reales, en un caso se ha medido la temperatura de la habitación de una determinada vivienda; y en otro caso se ha medido la temperatura dentro de un frigorífico que se va abriendo y cerrado. Los resultados obtenidos muestran el correcto funcionamiento de la solución propuesta.

El prototipo diseñado resulta de gran interés para los siguientes sectores:

- Industrias alimentarias donde se requiera la supervisión íntegra de la cadena de frío.
- Productos farmacéuticos, ya que algunas vacunas suelen ser sensibles sobre determinadas temperaturas y pueden perder efectividad si estas no se almacenan en unas condiciones de temperatura óptimas.
- Instalaciones donde se almacenan productos refrigerados, congelados y ultracongelados, este tipo de instalaciones deben registrar y documentar las temperaturas de dichos congelados, obligadas por la legislación vigente.
- Instalaciones de sistemas de calefacción y ventilación (HVAC). Resulta necesario un registro de temperaturas para la búsqueda de errores en la instalación de este tipo de sistemas.
- En el sector de la ganadería o de avicultura para asegurar que la temperatura de estos recintos es la adecuada para una mayor producción.

PRESUPUESTO

PRESUPUESTO:

1. NECESIDAD DEL PRESUPUESTO

A continuación se procederá a la valoración económica el producto desarrollado. La elaboración del presupuesto nos permitirá una administración más efectiva de los recursos económicos a emplear en el proyecto. Por tanto, se establecerá de una forma detallada toda la información sobre la inversión financiera que significaría el poder efectuar el desarrollo del producto diseñado.

Cabe destacar que el proyecto ha sido realizado en su totalidad por un Graduado de Ingeniería en Tecnologías Industriales. La unidad monetaria empleada ha sido el Euro (€) y la unidad de tiempo de realización del proyecto ha sido la hora (h). Al tratarse de un proyecto de carácter ingenieril, se ha empleado un criterio realista tanto para la asignación de los precios de los componentes y equipos, como para las horas de trabajo empleadas por el ingeniero.

Al ser el proyecto un trabajo de fin de grado no se incluirá el cálculo del IVA en el mismo.

2. CONTENIDO DEL PRESUPUESTO

Se ilustrarán los diferentes cuadros de precios necesarios para la realización completa del presupuesto del proyecto, así como la definición de las distintas unidades de obra.

2.1. UNIDADES DE OBRA.

A continuación, se presentarán las distintas unidades de obra consideradas para la realización del presupuesto.

- UO1.-Materiales: Componentes electrónicos empleados para el diseño de la placa
- UO2.-Montaje de equipo: Horas empleadas para la realización del diseño del circuito e implementación en placa PCB
- UO3.-Desarrollo de la programación. Tiempo empleado en la implementación del programa de Arduino®
- UO4:-Redacción de la memoria: Tiempo estimado para la redacción completa del documento Trabajo Fin de Grado.

2.2. CUADRO DE PRECIOS

El presupuesto redactado consta de varias partes. En primer lugar se presentarán tres cuadros de precios claramente diferenciados e ilustrados en las figuras

- ***Materiales:*** *Conjunto de materiales empleados para la realización del proyecto*
- ***Material ofimático:*** *Sueldo* que supone al ingeniero el poder utilizar distintos materiales ofimáticos tales como ordenadores o licencias de pago.
- ***Jornales:*** *Conjunto de pagos a realizar al ingeniero por la ejecución de su trabajo.*

2.2.1. Cuadro de precios materiales:

Componente	***Precio/ud (€)***
Condensador 100 nF	0,73
Condensador electrolítico 1 uF	0,11
Pila 9 voltios 800 mAh	5,93
Regulador L7805	0,58
Regulador L7905	0,19
Sensor LM35	0,62
Diodo	0,03
Diodo Zener 5,1 Voltios	0,09
Resistencia 100 k	0,01
Resistencia 15 k	0,01
Potenciómetro 100k	0,33
Potenciómetro 50k	0,33
Amplificador Operacional OP-07	0,09
Arduino® Mega 2560	44,98
Placa PCB	3,00

Figura P.1: Cuadro de precios materiales.

2.2.2. Cuadro de precios del material ofimático.

	Total (€/año)	Total (€/h)
Ordenador	3000,00	1,70
Licencia Office	70,00	0,04
Licencia Orcad	550,00	0,31
Licencia Proteus	5327,00	3,03
Windows 10	100,00	0,06

Figura P.2: Cuadro de precios del material ofimático.

2.2.3. Cuadro de precios jornales:

El presente apartado se expondrá el coste que supone la contratación de un Graduado en Ingeniería en Tecnologías Industriales

Graduado en Ingeniería de Tecnologías Industriales

Concepto	***TOTAL (€/año)***
Sueldo Base	30000,00
Retribución transporte	700,00
Seguridad Social	6000,00
Honorarios	5000,00
Otros (Horas extras, dietas, haberes…)	1500,00

Total anual	43200,00 €
Total jornada	196,36 €
Total por hora	24,55 €

Figura P.3: Cuadro de precios jornales.

Nota: Para la construcción de las tablas anteriores, se ha considerado que una jornada laboral del ingeniero tiene una duración de 8 horas y el número total de jornadas laborables en un año asciende a 220 días.

2.2.4. Cuadro de precios: presupuesto de ejecución del material.

Concepto	Precio unitario	Cantidad	Subtotal
UO1. Materiales			
Condensador 100 nF	0,73	6,00	4,38
Condensador electrolítico 1 uF	0,11	6,00	0,66
Batería ULTRA LIFE 9 V, 1200 mA·h	5,93	2,00	11,86
Regulador L7805	0,58	1,00	0,58
Regulador L7905	0,19	1,00	0,19
Sensor LM35	0,62	1,00	0,62
Diodo	0,03	4,00	0,12
Diodo Zener 5,1 Voltios	0,09	1,00	0,09
Resistencia 100 k	0,01	1,00	0,01
Resistencia 15 k	0,01	1,00	0,01
Potenciómetro 100k	0,33	1,00	0,33
Potenciómetro 50k	0,33	1,00	0,33
Amplificador Operacional OP-07	0,09	1,00	0,09
Arduino® Mega 2560	44,98	1,00	44,98
Placa PCB	3,00	1,00	3
UO2. Montaje de equipo: Diseño del circuito, implementación en placa PCB			
Graduado en Ingeniería en Tecnologías Industriales	24,55	175,00	4295,45
Licencia Orcad	0,31	30,00	9,38
Licencia Proteus	3,03	90,00	272,40

Ordenador	1,70	170,00	289,77
Windows 10	0,06	170,00	9,66
Graduado en Ingeniería en Tecnologías Industriales	24,55	175,00	4295,45
UO3. Desarrollo de la programación			
Graduado en Ingeniería en Tecnologías Industriales	24,55	10,00	245,45
Ordenador	1,70	10,00	17,05
Windows 10	0,056818182	10,00	0,57
UO4: Redacción de la memoria:			
Graduado en Ingeniería en Tecnologías Industriales	24,55	115,00	2822,73
Ordenador	1,70	115,00	196,02
Licencia Office	0,04	115,00	4,57
Windows 10	0,056818182	115,00	6,53

Presupuesto de Ejecución Material	**8236,84**

Figura P.4: Presupuesto de ejecución del material.

El presupuesto de ejecución material asciende a una cantidad total de:

OCHO MIL DOSCIENTOS TREINTA Y SEIS EUROS CON OCHENTA Y CUATRO CÉNTIMOS.

2.2.5. Cuadro de precios: presupuesto de ejecución por contrata.

A todo lo anteriormente descrito, cabe añadir una cantidad considerada para el beneficio industrial de 6% y unos gastos generales del proyecto de un 16%

Concepto	Precio(€)
Presupuesto de Ejecución Material	8236,84
Beneficio Industrial (6%)	494,21
Gastos Generales (16%)	1317,89
Presupuesto Ejecución por Contrata	10048,95

Figura P.5: Presupuesto de ejecución por contrata.

El presupuesto de ejecución por contrata asciende a una cantidad total de:

DIEZ MIL CUARENTA Y OCHO EUROS CON NOVENTA Y CINCO CÉNTIMOS.

REFERENCIAS.

[1] AMAZON.ES: Medisana FTN - *Termómetro por infrarrojos de medición precisa de la temperatura, color gris plateado y verde.* 2015.

[2] GEFRAN AUTOMATION. Gefran 40 TB "*40TB Temperature and pressure double indicator / alarm unit*". "Datasheet". 2015.

[3] LOGTAG RECORDERS – Aplicaciones – *Data Loggers* - URL: https://www.logtag-recorders.com/es/aplicaciones/

[4] INSTRUMENTOS TESTO S.A – *Registro de temperatura* – 2017.

[5] SRC – *Reguladores de temperatura y procesos para industria* - 2016

[6] UNE-EN 12830:2000 - *Registradores de temperatura para el transporte, almacenamiento y distribución de alimentos refrigerados, congelados y ultracongelados y helados. Ensayos, funcionamiento, aptitud de uso.*

[7] Imágenes de Arduino - Store.

[8] DOUTEL F. –*Todo lo que necesitas saber sobre Arduino* – 2015.

[9] Imagen de EZZELINO R. - *Arduino Uno Pinout Diagram* – 2013. URL: http://forum.arduino.cc

[10] Imagen de ATMEL CORPORATION - ATmega328 – *Data Sheet* - 2015

[11] Imagen de TAWIL Y. - *Understanding Arduino UNO Hardware Design* – All about circuits - 2016

[12] TORRES H. - *"Arduino vs Microcontrolador reseña y opinión"* - Het Pro - 2015

[13] Imagen de Espinosa A. – *Desbloquear el Bootloader de un Android* – Pro Android - 2017

[14] Imagen de Microchip Technology Inc. - PIC18F2550 - *Data Sheet* – 2006.

[15] *Arduino vs PIC: La gran batalla* - Nextia Fenix Blog – 2014

[16] Imagen de MC Electronics - *Placa de entrenamiento para PIC de 8 bits - 2015*

[17] Imagen de Microchip Technology Inc. - *MPLAB® X Integrated Development Environment (IDE)* - 2016

[18] Universitat Politécnica de València. PoliformaT : 2016-Proyectos GITI : Recursos: Tema Tema 8.1_Diseño de productos. Metodología y técnicas. Parte 1.pdf.

[19] Universitat Politécnica de València. PoliformaT : 2016-Proyectos GITI : Recursos: Tema 8.2_Diseño de productos. Metodología y técnicas. Parte 2.

[20] Arduino | Products- ArduinoBoardMega2560.

[21] Imagen de Gabriel Rivamar A. - *"Tabla comparativa de Arduinos"* – Sabe Tecnología -2013

[22] ÁLVAREZ ANTÓN, J. C. - Instrumentación electrónica - 2004

[23] PALLÀS ARENY, R. - *Sensores y acondicionadores de señal*- 4ª ed. -Barcelona: Marcombo. 2003..

[24] Imagen de Omega Engineering – Termopar .

[25] Imagen de http://server-die.alc.upv.es

[26] Texas Instruments - *LM35. "Precision Centigrade temperature sensor (-55C to 150C)" – Data Sheet* - 2016

[27] LLAMAS L. - *Entrada analógica de 16 bits con Arduino y ADC ADS1115* - Noviembre 2016.

[28] Texas Instruments. ADS11115. "*ADS111x Ultra-Small, Low-Power, I 2C-Compatible, 860-SPS, 16-Bit ADCs with Internal Reference, Oscillator, and Programmable Comparator*". Diciembre 2016.

[29] Universitat Politécnica de València. PoliformaT: (11431) Tecnología electrónica. *[Tema 8: Convertidores analógico-digital y digital-analógico*.] – 2015.

[30] Analog Devices. *"Next Generation OP07 Ultralow Offset Voltage Operational Amplifier, OP77"* – Data Sheet. - 2015

[31] PLEITE GUERRA, J; VERGAZ BENITO, R.; RUIZ DE MARCOS, J.M. - *Electrónica analógica para ingenieros* – McGraw-Hill – 2009

[32] Universitat Politécnica de València. PoliformaT : - Sistemas Electrónicos. Recursos: *"Aplicaciones del amplificador operacional" – 2015.*

[33] Funny Electronics - *-5V-0-5V Voltage Regulator Using 7805 and 7905* - 2014. URL: http://www.learnerswings.com/

[34] MIRÓ OROZCO I. - *OrCAD Capture. Introducción de esquemas* - Riunet UPV - Abril 2008.

[35] MIRÓ OROZCO I. - *OrCAD Capture. Introducción de esquemas II* - Riunet UPV - Abril 2008.

[36] MIRÓ OROZCO I. - *OrCAD Capture. Entorno de Trabajo.* - Riunet UPV - Abril 2008.

[37] MIRÓ OROZCO I. - *Introducción al OrCAD 15.7.* - Riunet UPV - Abril 2008.

[38] MIRÓ OROZCO I. - *OrCAD Capture. Eligiendo "footprints".* - Riunet UPV - Abril 2008.

[39] MIRÓ OROZCO I. - *OrCAD Layout. Colocando componentes*. - Riunet UPV - Abril 2008.

[40] MIRÓ OROZCO I. - *OrCAD Capture/Layout. Generando el .MAX*. - Riunet UPV - Abril 2008.

[41] MIRÓ OROZCO I. - *OrCAD Layout. Acabando el diseño.* - Riunet UPV - Abril 2008.

[42] MIRÓ OROZCO I. - *OrCAD Layout. Trazando pistas*. - Riunet UPV - Abril 2008.

[43] MIRÓ OROZCO I. - *OrCAD Capture. Actualizando propiedades*. - Riunet UPV - Abril 2008.

[44] Imagen de Farnell Element14 España – Batería Ultra Life, 9 V, PP3, Dióxido de Manganeso-Litio - 1200 mAh.

[45] Rocket Scream Blog– Unleash the rocket scientist in you

http://www.rocketscream.com/blog/

[46] MASCARELL JUSTE, J.B. – *Desarrollo de un sistema para la recuperación y monitorización de temperaturas almacenadas en puntos remotos* – Trabajo Fin de Grado – Septiembre 2017

ANEXOS

MANUALES PARA EL USO DEL DISPOSITIVO

En el presente anexo se presentará, en primer lugar, un manual de uso técnico del dispositivo, dirigido a personal con conocimientos tecnológicos. Seguidamente se presentará otro manual esta vez dirigido para los usuarios que puedan hacer uso del dispositivo desarrollado a lo largo del proyecto.

1. MANUAL TÉCNICO.

En primer lugar, se comentarán algunas características del interfaz de Arduino:

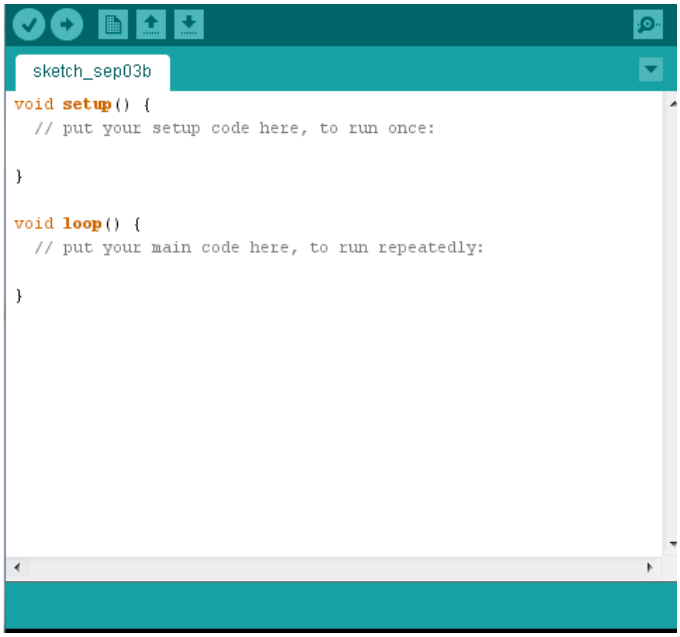


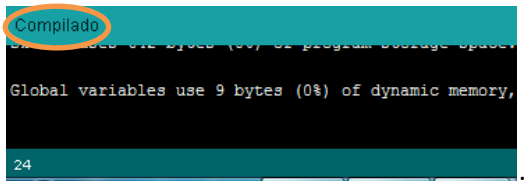
Figura A.1: Interfaz del software de compilación de Arduino®.

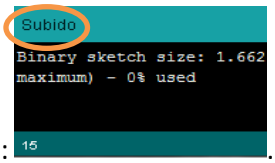
En la parte del *"void Setup"*, Arduino® solo lee una vez esta parte del programa. El *"void loop"* es un bucle que el microcontrolador lee infinitas veces hasta que se procede al corte de la alimentación del mismo.

Para asegurarnos que nuestro Arduino® está conectado correctamente a nuestro PC debemos observar que aparece el nombre del modelo de Arduino® en la parte inferior de la pantalla

junto con el puerto del PC a que está asociada su conexión: 

.

Para proceder a la compilación del código se realizará mediante el botón compilar:  . Si la sintaxis del código escrito es correcta se visualizará el siguiente mensaje en la parte inferior de la pantalla: 

. Para la transferencia del código diseñado a nuestra placa Arduino® se dará al botón transferir:  . Si todo ha ido bien se observará la siguiente ilustración: 

.

A continuación se describirán una serie de pautas sobre la modificación algunos parámetros de medida del dispositivo diseñado. Básicamente se describirán algunas características del código Arduino® diseñado que pueden son modificables para cada aplicación en cuestión.

Primeramente, se describirán los pasos a seguir para la modificación del tiempo de toma de datos y frecuencia de muestreo en el código de programación de Arduino®. Se puede cambiar la frecuencia de muestreo de manera muy simple:

El primer bucle “for” está programado para 1440 iteraciones y con un tiempo de descanso entre iteraciones de 1 minuto. Por tanto para salir del bucle, y dejar de almacenar datos en memoria, se tardan 1440 minutos, o lo que es lo mismo 24 h.

Modificando el tiempo de descanso con la función “LowPower.powerDown” se pueden conseguir tomas de datos con una frecuencia de muestreo diferente. También se deberá proceder a cambiar el número de iteraciones de toma de datos para modificar el tiempo total del almacenamiento de datos.

$$Tiempo\ de\ almacenamiento\ = Número\ de\ iteraciones \cdot\ frecuencia\ de\ muestreo. \quad (A.1)$$

```
#include <EEPROM.h>
#include <LowPower.h>

const int sensorPin=A0;  //seleccionar entrada del sensor
unsigned int Temp,d,m,b,n,i; //16 bits
//unsigned char 8 bits
int eeAddress=200;//16 bits
byte H,L;
void setup()
{
Serial.begin (9600); //Inicializa Puerto serie
delay (5000); //Esperar antes de entrar en el bucle por si existiesen transitorios, pueden fallar las primeras medidas.
for  n=0; n<1440; n++)
{
  Temp=analogRead(sensorPin); //16bits lecytura sensor
H=highByte(Temp);
L=lowByte(Temp);
EEPROM.write(eeAddress,H);
EEPROM.write(eeAddress+1,L);
eeAddress= eeAddress+2 ;

for (i=0; i<16; i++){
LowPower.powerDown(SLEEP_4S, ADC_OFF,BOD_OFF);} //Apagamos ADC y el circuito BOD durante un minuto de espera entre cada medición.
}

}
void loop ()
{
}
```

NÚMERO DE ITERACIONES

FRECUENCIA DE MUESTREO

Figura A.2: Cambio de la frecuencia de muestreo. Parámetros a modificar recuadrados en rojo.

El tiempo de espera entre iteración o la frecuencia de muestreo, en este caso será:

$$4\ seg\ del\ sleep \cdot\ 15\ pasos\ por\ el\ bucle = 60\ segundos\ de\ espera \qquad (A.2)$$

Por tanto, el tiempo total de captación de temperaturas será, en este caso de 24 horas.

Seguidamente, se debe comentar que se ha reservado los primeros huecos de la memoria para cambiar algunas variables tales como el día de la medición, la hora, etc. Estos datos son interesantes de introducir para que en la posterior vista de las gráficas con LabView aparezcan los parámetros descritos, tal y como se visualizan en las Figura 9.3. A continuación en la Figura 10.3 se presentaran un código de programación para el microcontrolador donde se observa cómo cambiar estos parámetros.

```
#include <EEPROM.h>
unsigned int hora, minuto, dia, mes;

void setup() {
  // put your setup code here, to run once:
  Serial.begin (9600); //Inicializa Puerto serie
  hora=13;//Insertar hora de inicio medicóon)
minuto=32;//Insertar minuto de inicio medición)
dia=25;//Insertar día de la medicion)
mes=6;//Insertar número de mes de la medición)
EEPROM.write (13,hora);
EEPROM.write (15,minuto);
EEPROM.write (16,7);// muestra 2017 como año de inicio medicion)
EEPROM.write (17,225);//muestra 2017 como año de inicio medicion)
EEPROM.write (19,mes);
EEPROM.write (21,dia);

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura A.3: Ilustración sobre cómo cambiar el día, hora, etc. de la medición [46].

Nota: El dato del año de la medición se ha introducido directamente como el número "2017" en numeración binaria con su correspondiente separación de 1 byte en 1 byte, que es como almacena los datos internamente Arduino®.

A continuación se comentará brevemente como ajustar correctamente los valores de los potenciómetros.

Siguiendo las ecuaciones (7.2) y (7.3), el potenciómetro R3 (azul oscuro) se empleará para ajustar el cero de nuestro circuito (0 V para -0.55 V de entrada) y el potenciómetro R2 (azul claro) se empleará para ajustar el fondo de escala.

2. MANUAL PARA EL USUARIO FINAL/CLIENTE.

A continuación se describirán una serie de pautas y recomendaciones a seguir para un uso eficiente del dispositivo diseñado, dirigidas al usuario final o cliente.

Las conexiones a realizar serán las siguientes:

La alimentación se realizará desde dos baterías como ya se ha comentado anteriormente. Éstas irán conectadas al conector 1 (de 3 entradas):

- **El terminal + de la pila 1** (cable rojo) se conectará a la entrada 1 del conector 1.
- **El terminal - de la pila 1** (cable negro) se conectará a la entrada 2 del conector 1.
- **El terminal + de la pila 2** (cable rojo) se conectará a la entrada 2 del conector 1.
- **El terminal - de la pila 2** (cable negro) se conectará a la entrada 3 del conector 1.

Por tanto, las conexiones de la alimentación deben quedar de la siguiente manera:

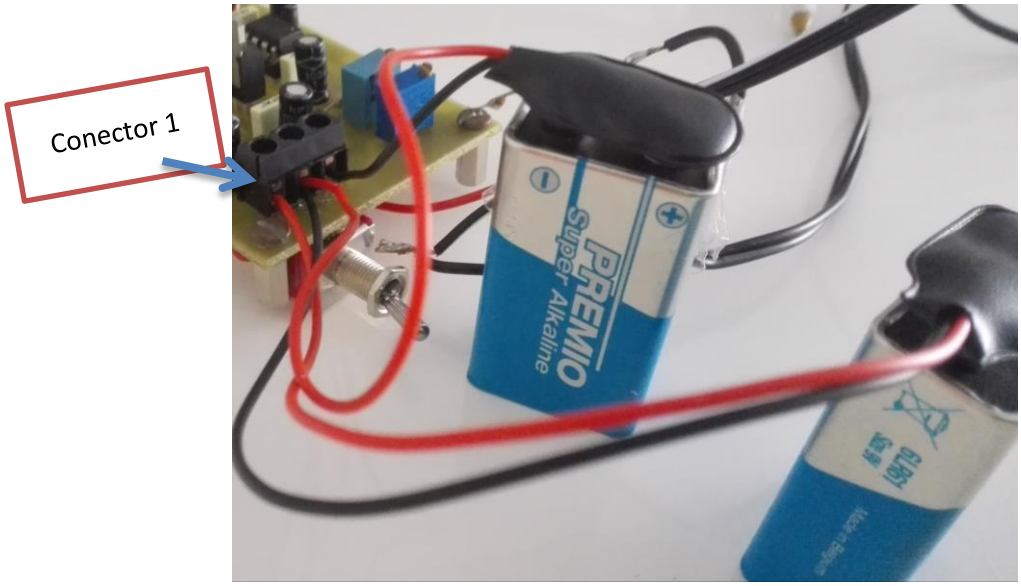
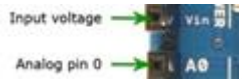
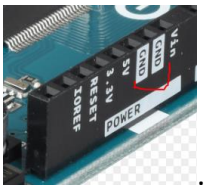


Figura A.4: Ilustración del Conector 1 junto con sus correspondientes conexiones con las baterías.

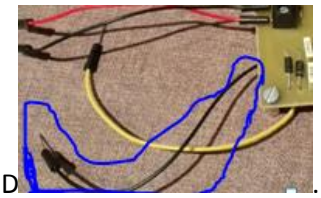
El cable rojo  indica la salida de la señal de la variable medida, es decir, es la que determina la temperatura. Por tanto, el cable rojo se conectará a Arduino mediante la entrada Analógica A0 

. El cable negro del conector de salida  está conectado en la placa a tierra, será conectado al PIN que indica GND en Arduino® 

.

A continuación se procederá a indicar como alimentar Arduino® desde nuestra placa. El cable amarillo  está conectado en nuestra placa a la salida del regulador, por tanto, lleva asociado una tensión de 5 V. Este cable amarillo se conectará al PIN de

alimentación de Arduino® 

de 5V. Tenemos otro cable negro también conectado a tierra en la placa, este cable establecerá la tierra en para la alimentación de Arduino® por tanto también ira conectado a otro PIN de GND  .

En este momento indicar que antes de proceder a la inicialización de la medición de la temperatura se debe comprobar, en primer lugar, una correcta conexión de las baterías al conector 3 de entrada  junto con las conexiones de nuestro circuito de diseño a Arduino.  .

Seguidamente se inicializará la medición. Solo se procederá a poner el interruptor en la posición ON, es decir, la punta del interruptor apuntando hacia arriba  . Así se conseguirá cerrar la alimentación.

A partir de este momento Arduino® ya está almacenando las medidas de las temperaturas. Se dejará actuar en el recinto durante el tiempo deseado. Cuando Arduino® esté activo se mostrará el siguiente pin iluminado. 

Finalmente, cuando deseemos detener las mediciones se procederá a situar el interruptor en la posición OFF  es decir la punta del interruptor apuntando hacia abajo. Así se conseguirá abrir el circuito y por tanto, cortar la alimentación.

Seguidamente se procederá a la descarga y monitorización de las temperaturas captadas mediante el software correspondiente, por ejemplo LabView.