Sistemas basados en microprocesador.

Sistema inalámbrico para el control de la calefacción.

Alberto García Izquierdo
Ismael Pérez Rojo

Índice

1. Descripción del proyecto.

El proyecto se realiza para la presentación en la asignatura Sistemas basados en microprocesadores del grado en ingeniería electrónica de la universidad de Burgos.

Este proyecto está elaborado por Ismael Pérez Rojo y Alberto García Izquierdo.

La finalidad del mismo es el control y monitorización remota de la temperatura de diferentes estancias mediante el encendido y apagado del sistema de calefacción.

Se tratarán tres salas diferenciadas, dónde se evaluará el nivel de temperatura y se actuará según las necesidades del usuario, marcadas de forma manual o a través de la web.

El desarrollo se basa en la plataforma de hardware libre Arduino. También constará de un módulo de conexión a internet y de un servidor remoto. Tendrá también otros componentes discretos para la obtener temperaturas y operar con el dispositivo.

Como todo sistema de control tendrá un modo de funcionamiento manual desde una interface manual, y un modo automático a través de internet.

2. Objetivos del proyecto.

Los objetivos principales propuestos por el profesor de la asignatura son:

- Desarrollo de un sistema de control a elegir por los alumnos. 100%

- Utilizar una plataforma de hardware libre Arduino. Microcontrolador Atmega328P. 100%

- Utilizar al menos una entrada/salida digital y una entrada analógica. 100%

- Realizar una recogida de información y tratamiento de datos a distancia. 100%

Se han desarrollado otros apartados extras por los autores del proyecto.

3. Componentes utilizados. Descripción y funciones.

3.1. Arduino

3.1.1. Introducción Arduino.

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

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. En este proyecto se ha usado el Atmega328p.

Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa.

Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, y controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador.

Como hemos dicho es una plataforma de hardware libre lo que quiere decir que se dispone de los diseños a nivel de usuario pudiendo modificar parte de ellos y adaptándonos a las necesidades de cada proyecto.

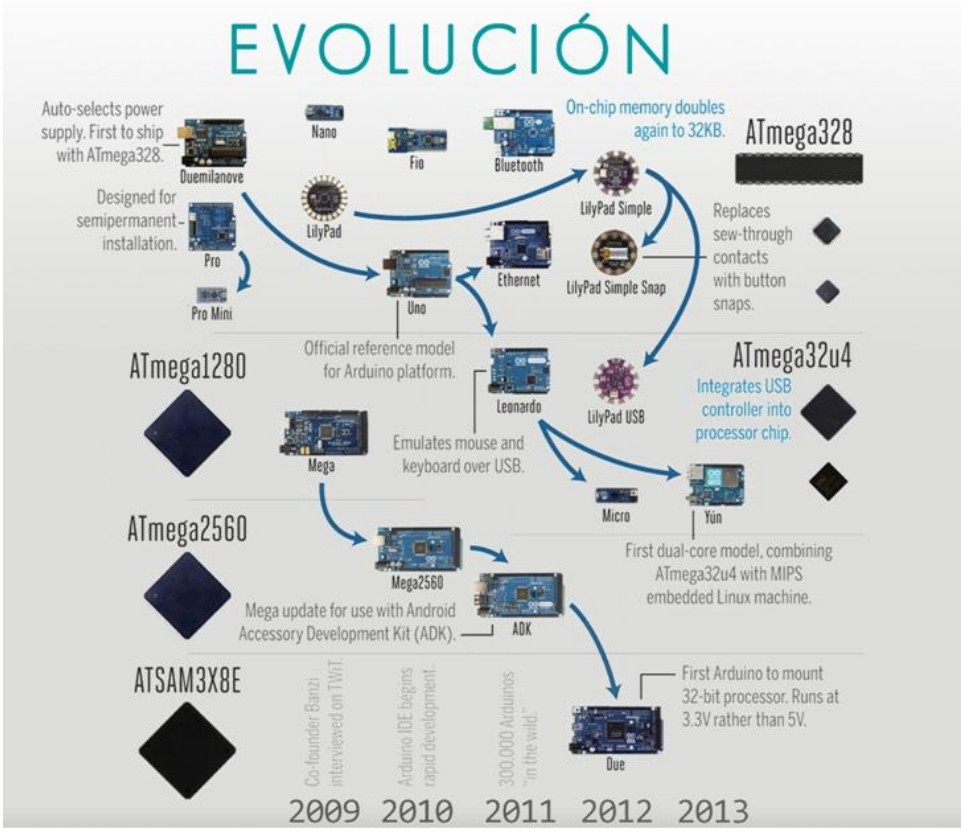
3.1.2. Hardware elegido para el proyecto.

Existen diferentes placas de arduino. Por lo que la primera tarea del trabajo es elegir la adecuada que llegue adaptarse al proyecto.

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

A parte de estos casos existen la placa, Arduino Nano, Arduino Mini, Arduino Micro, Arduino Fio… Estas placas son de tamaño menor y preparadas para cuando nuestro proyecto requiera de un controlador pequeño. No es el caso donde la necesidad del tamaño sea un problema.

Vamos a ver la evolución de las placas en los últimos años:



En la figura anterior vemos las placas que existen actualmente. Insistimos en la prioridad de dar con la placa adecuada en cada trabajo.

Se debe elegir siguiendo una serie de principios:

- Espacio crítico: NANO, FIO (Xbee), Pro Mini, LilyPad (flexible, cosido a la ropa)
- Número de entradas/salidas: Arduino Mega(16/54)/Arduino Due(12/54)
- Tamaño del código: Arduino Uno (32Kib), Arduino Mega (265KiB), Arduino Due (512KieB).
- Potencia de procesamiento: Arduino Uno (16MHz), Arduino Due (84MHz).

Relativizando estos aspectos a nuestro proyecto, el arduino que usaríamos es el Arduino Mega, el tamaño de nuestro código se acerca a 32KiB por lo que es posible que un Arduino UNO se quede corto, y las entradas y salidas que tendríamos no sería suficiente con un Arduino UNO.

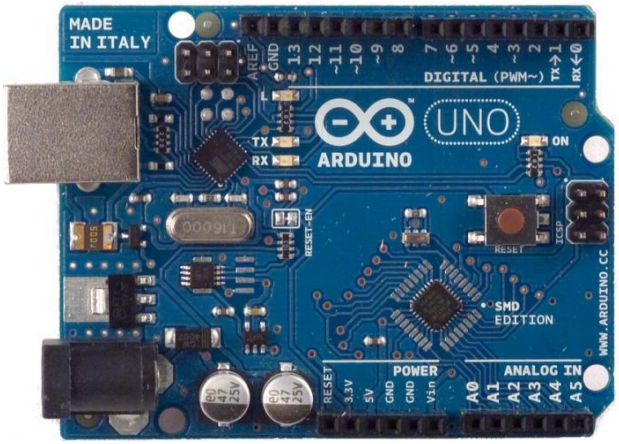
Puesto que el departamento de Electrónica nos aportó para la realización del trabajo el Arduino UNO nos adaptamos a realizar el sistema con esta placa, sabiendo que para su implementación puesto que necesitaríamos más entradas/salidas y más capacidad de memoria usaremos un Arduino MEGA. Para implementar este código a un Arduino Mega habría que tener en cuenta el patillaje y algunas configuraciones del puerto serial.

Como no necesitamos velocidad el Arduino Due queda descartado.

Todo lo que sigue a continuación de este proyecto está desarrollado para un Arduino UNO.

3.1.3. Arduino UNO

Se trata de un Arduino UNO SMD. A nivel de programación lo usaremos de la misma manera que un Arduino UNO.



Características Principales.

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

El esquema se puede obtener de:

http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

Alimentación: Arduino UNO puede estar alimentado por dos vías:

Conexión USB (que proporciona 5 V). En nuestro caso no lo alimentaremos porque cortocircuitaremos los pines serial usados TX, RX del puerto COM del ordenador con los del Xbee.

Jack de alimentación (que en nuestro caso es una fuente de 8V).

Por los terminales Vin y Gnd, de esta manera alimentaremos el arduino Nano con una tensión de una fuente de 12V. En la pcb diseñada será de 8V.

Si se conectan dos fuentes de alimentación por dos canales distintos el arduino por defecto escoge automáticamente una y desprecia las demás.

Valores de entrada y de salida: en función de cómo este siendo utilizado en pin, tendremos:

Salida y entrada digital: los valores de salida pueden ser o 0 V (LOW) o 5 V

(HIGH), y se interpretara una entrada de entre 0 y 2 V como LOW y de entre

3 y 5 V como HIGH.

Salida analógica: los valores de salida van desde 0 V a 5 V en un rango de

0 a 255 (precisión de 8 bits) valores intermedios.

Entrada analógica: los valores de entrada van desde 0 V a 5 V en un rango

de 0 a 1023 (precisión de 10 bits) valores intermedios.

La intensidad máxima de todos estos pines es de 40 mA.

Normalmente, todo el circuito electrónico que Arduino controlara se monta sobre

una placa de prototipos o breadboard, y el conexionado se realiza con cables tipo

jumper (es importante utilizar este tipo de cables porque no suelen romperse en los

zócalos):

3.1.4. Instalación IDE Arduino

Para poder escribir un programa en Arduino se siguen los siguientes pasos.

Utilizamos Windows.

- Descargar e instalar el Java Runtime Enviroment (J2RE).
- Descargar última versión del IDE Arduino.

http://arduino.cc/es/Main/Software

Instalamos el IDE de Arduino. Una vez instalado necesitamos encontrar el driver para la conexión. Instalar FTDI USB Drivers

Conectamos el arduino y no buscamos el driver automáticamente, si no que en configuración avanzada seleccionamos el directorio de la carpeta de arduino donde tenemos ubicado el driver.

Ahora si podemos empezar a programar el arduino.

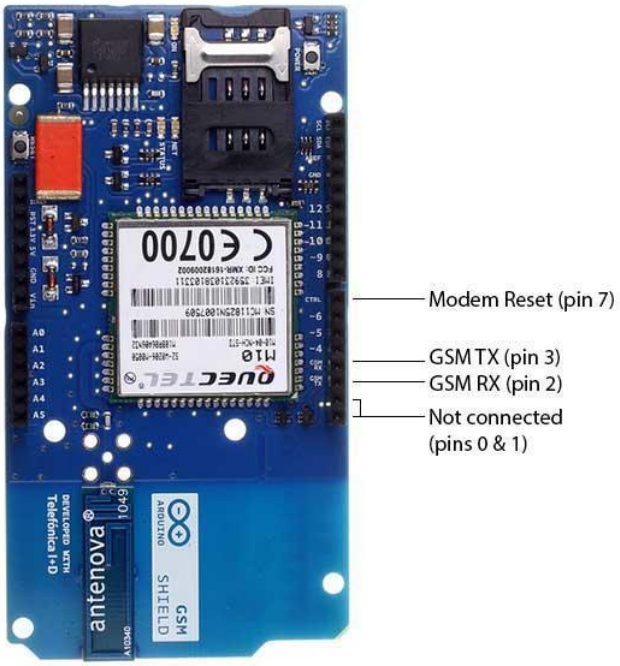
Debemos cargar en la carpeta de librerías, la librería del GSM y la de LCD que en las últimas versiones del entorno de desarrollo ya vienen incorporadas.

3.2. Shield GSM

Este shield es una parte importante del proyecto, es el encargado de realizar la comunicación con un servidor propio alojado en internet. Se encargará de enviar y recibir datos para establecer una comunicación. De esta manera podremos monitorizar y actuar de forma remota.

Este shield generará una conexión a internet a través de una tarjeta SIM que añadiremos. De esta manera no necesitaremos conectarnos a una red local que exista por las proximidades. Esto es una ventaja si se quiere instalar en un área sin acceso a ninguna red.

Podremos conectarnos a la red GSM y enviar o recibir mensajes SMS, realizar o recibir llamadas o enviar datos a servidores a través de la red de teléfono. Aunque actualmente ya existen placas con similar funcionamiento, esta es la considerada oficial desarrollada entre el team de Arduino y la compañía Telefónica. Según indican en la página oficial, es compatible con los modelos UNO, MEGA y Leonardo.



Vemos como utiliza los pines 2,3 para realizar la conexión al Arduino, de manera que estos dos pines ya quedan inutilizados para realizar más tareas.

La librería GSM.h la utilizaremos a la hora de realizar el código.

Esta placa la usaremos para conectarnos como cliente a un servidor donde interactuamos con los datos del programa.

Se podrá enviar un mensaje de texto de manera que sea un sistema de alarma, por lo que recibiremos un sms a nuestro teléfono si se cumplen unas características fijadas por los autores del proyecto.

3.3. LCD.

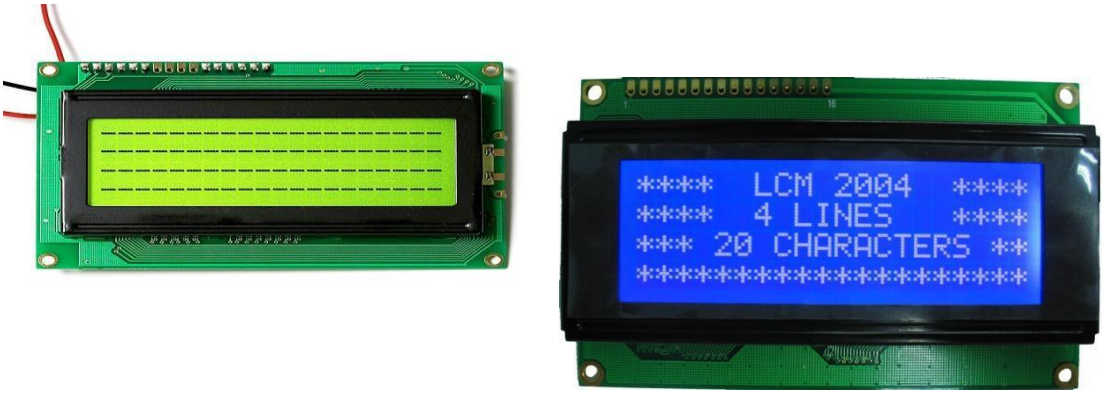
Se trata de nuestra interface de modo manual, hemos creado una especie de menú a través de una serie de pulsadores/potenciómetros donde nos iremos desplazando por el recorrido del programa.

El LCD se encargará de monitorizar cada menú así como todos los datos (temperaturas, estados…) a tiempo real.

Se trata de una LCD 20x4 (20 caracteres, 4 columnas).

Usaremos la librería liquidcristal.h.

http://arduino.cc/es/Tutorial/LiquidCrystal



El LCD tiene un interfaz paralelo, significando esto que el microcontrolador tiene que manipular varios pines del interfaz a la vez para controlarlo. El interfaz consta de los siguientes pines:

Un pin de selección de registro (RS) que controla en qué parte de la memoria del LCD estás escribiendo datos. Puedes seleccionar bien el registro de datos, que mantiene lo que sale en la pantalla, o un registro de instrucción, que es donde el controlador del LCD busca las instrucciones para saber cual es lo siguiente que hay que hacer.

El pin de lectura/escritura (R/W)que selecciona el modo de lectura o el de escritura.

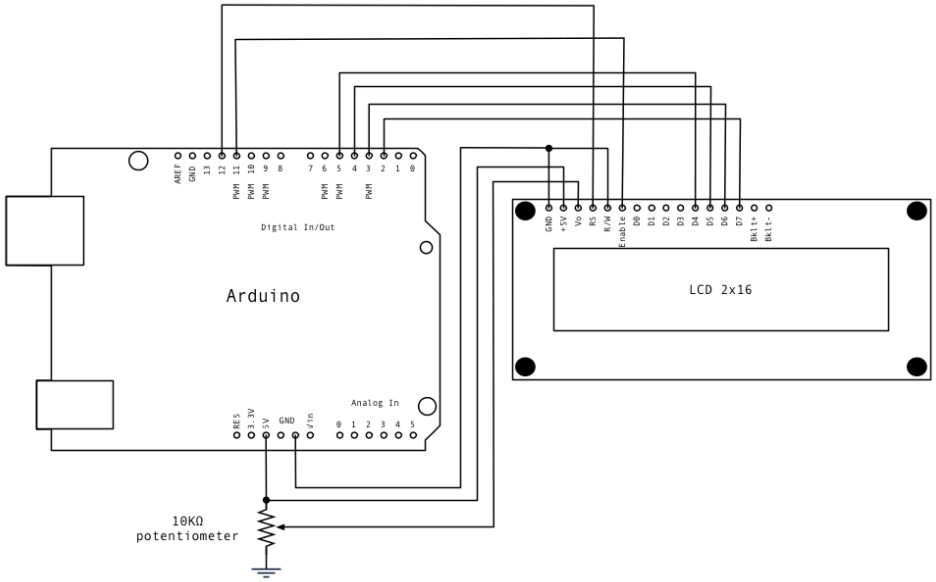
Un pin para habilitar (enable) que habilita los registros.

8 pines de datos (D00-D07). Los estados de estos pines (nivel alto o bajo) son los bits que estás escribiendo a un registro cuando escribes, o los valores de lectura cuando estás leyendo.

Hay también un pin de contraste del display (Vo), pines de alimentación (+5V y GND) y pines de retro-iluminación (Bklt+ y Bklt-), que te permiten alimentar el LCD, controlar el contraste del display, o encender y apagar la retro-iluminación, respectivamente.

El proceso de controlar el display involucra la colocación de los datos que componen la imagen de lo que quieres mostrar, en los registros de datos, y luego, colocar las instrucciones, en el registro de instrucciones. La librería LiquidCrystal te simplifica todo este proceso de forma que no necesitas saber las instrucciones de bajo nivel.

Los LCD-s compatibles con Hitachi pueden ser controlados de dos modos: 4 bits u 8 bits. El modo de 4 bits requiere siete pines de E/S de Arduino, mientras el modo de 8 bits requiere 11 pines. Para mostrar texto en la pantalla, puedes hacer la mayoría de las cosas en modo 4 bits, por lo que el ejemplo muestra cómo controlar un LCD de 2x16 en modo de 4 bits.



El conexionado para la LCD de 20x4 es exactamente el mismo. Las funciones de scroll de texto en pantalla para una de 20x4 funcionan en las líneas pares a la vez.

3.4. Servidor web.

Las paginas que hemos creado que contienen las funciones y los datos están alojadas y se ejecutan en un servidor remoto al que accedemos mediante una dirección IP o mediante una dirección URL. En nuestro caso: www.ubumicros.hol.es.

Este servicio está disponible de forma gratuita, pero con ciertas limitaciones. Es necesario que el servidor permita la ejecución de php para que todo funcione, ya que la mayor parte del código está escrito en este lenguaje.

Las funciones que tendrá será recoger los datos enviados por arduino, recoger los datos introducidos por el usuario y mostrar y tener disponibles todos los datos anteriores. También servirá para validar el origen de los datos y verificar usuarios, para que los datos no puedan ser manipulados por cualquier persona sin conocer las claves de acceso.

3.5. Otros componentes

Usaremos en principio una protoboard para el conexionado de la electrónica y probar el correcto funcionamiento del sistema.

Para el proceso cíclico de la barra de menús usaremos un potenciómetro (22KΩ) y un pulsador, entrada analógica y digital respectivamente.

Condensadores para eliminar el ruido del circuito 1uF.

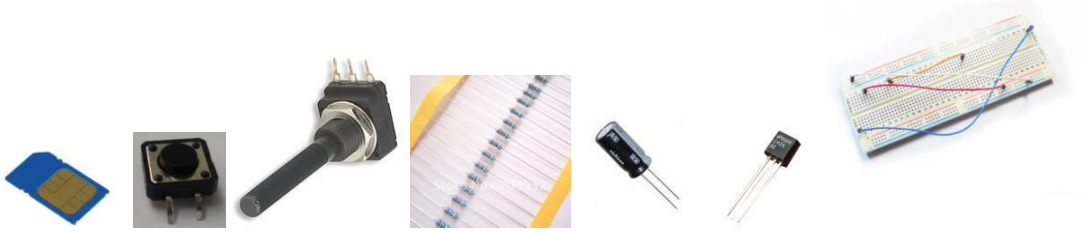
Resistencias de protección 270Ω(1/4W) para los led que simularán las salidas digitales de los relés que irán conectados al sistema de potencia de climatización.

Un potenciómetro de 10KΩ para el ajuste del contraste de la LCD.

Tarjeta SIM para el módulo GSM.

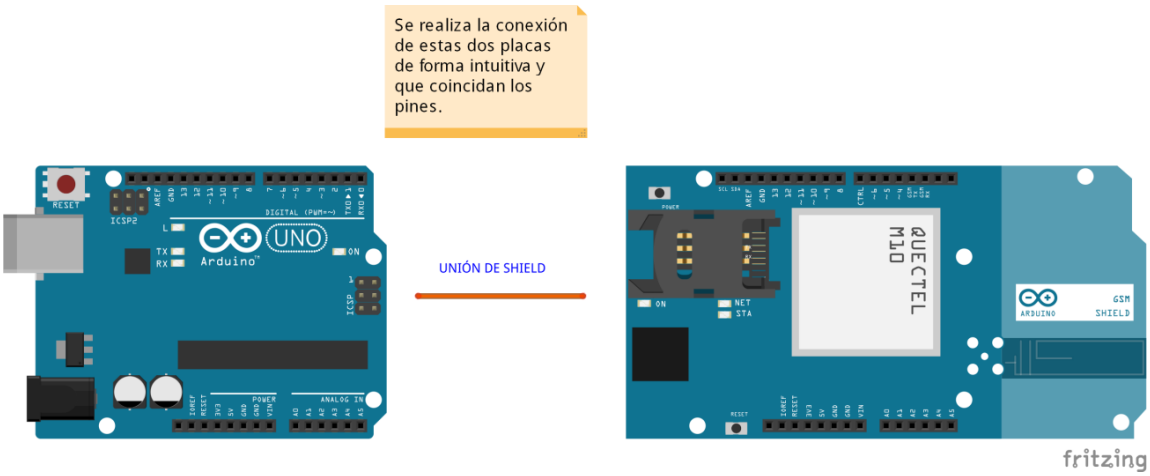
Sensor LM35 para monitorizar las temperaturas de cada sala.

Cableado.

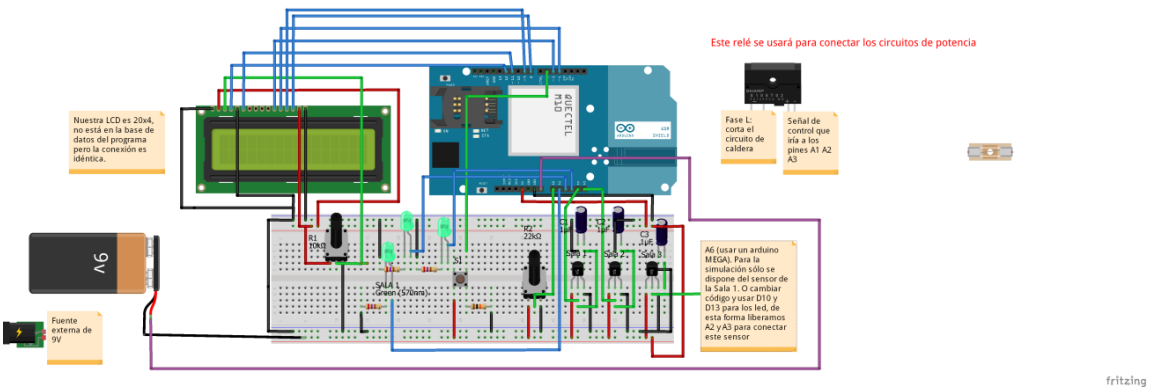


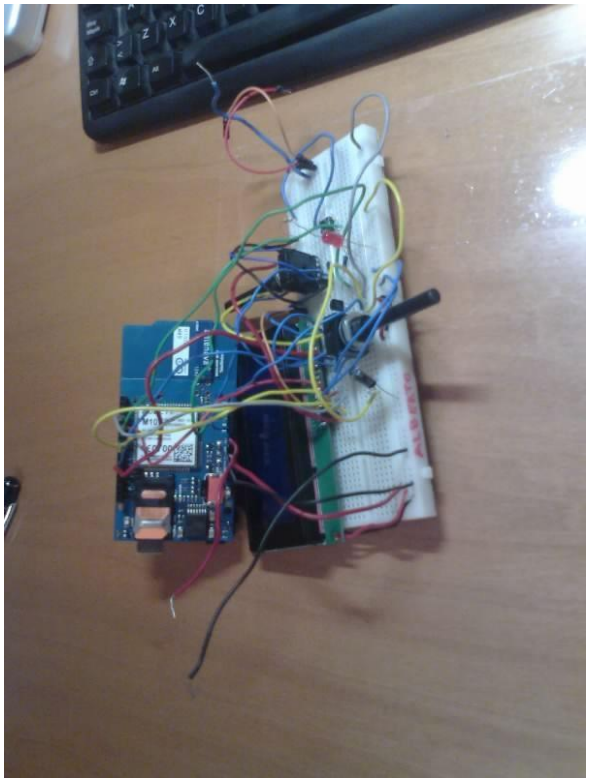
4. Esquemas de conexionado.

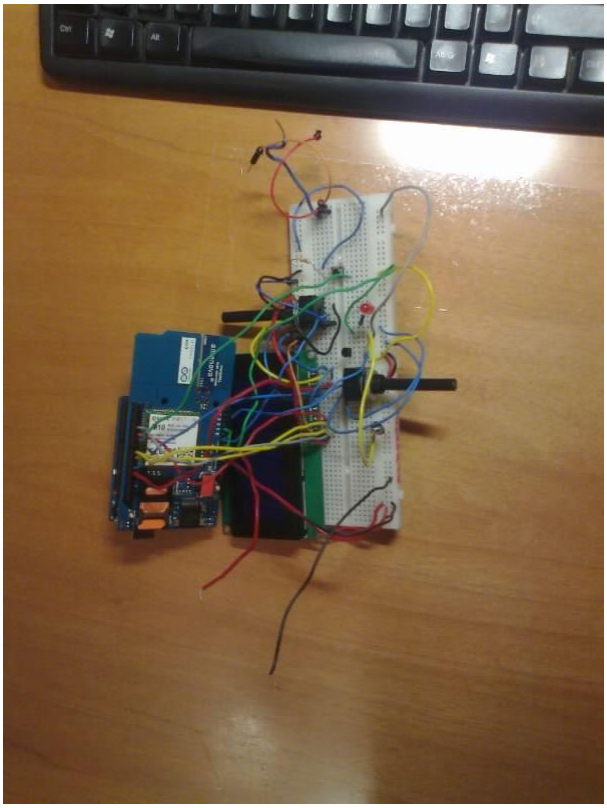
Primero conectamos las placas de forma automática sabiendo que quedan inutilizados los pines 2 y 3 que se usan para el GSM Shield.

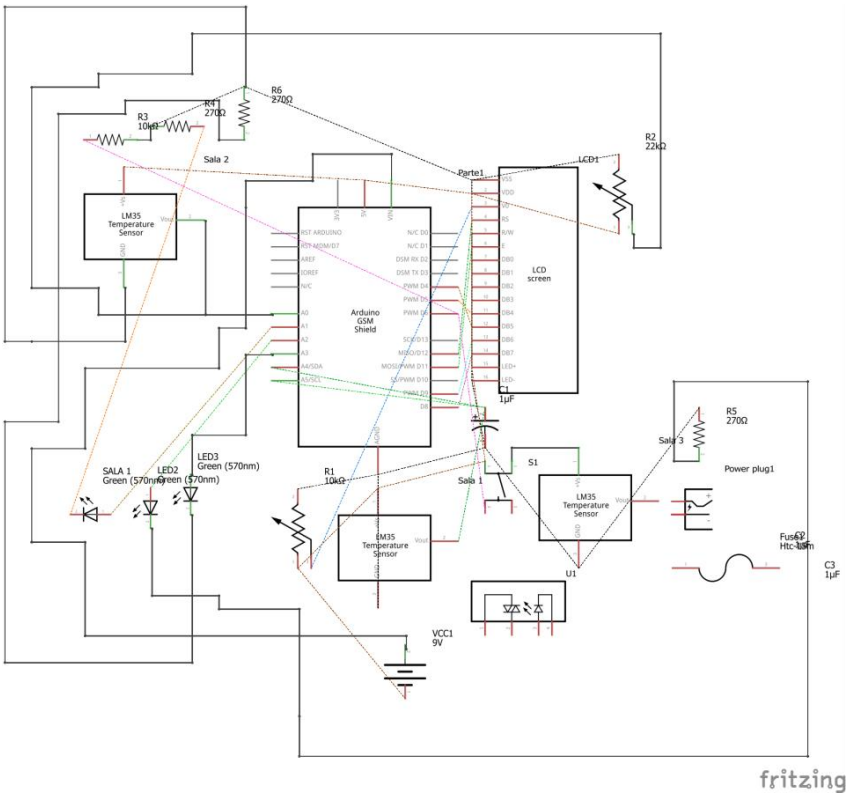


Procedemos al conexionado de todo lo demás:









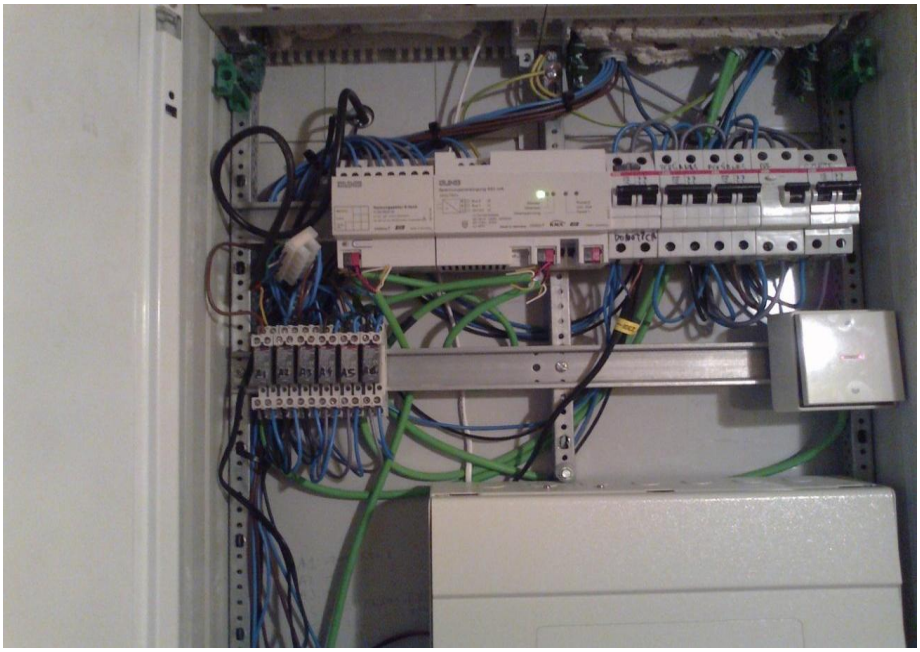
El programa Fritzing con el que está hecho el esquema con los dibujos te permite realizar el circuito esquemático pero al autorutear se quedan conexiones sin ejecutarse. También permite hacer la PCB pero esto no es competencia de la asignatura, aunque un acabado en PCB para posterior ensamblaje dentro del cuadro sería lo apropiado en la instalación.

5. Circuito e instalación.

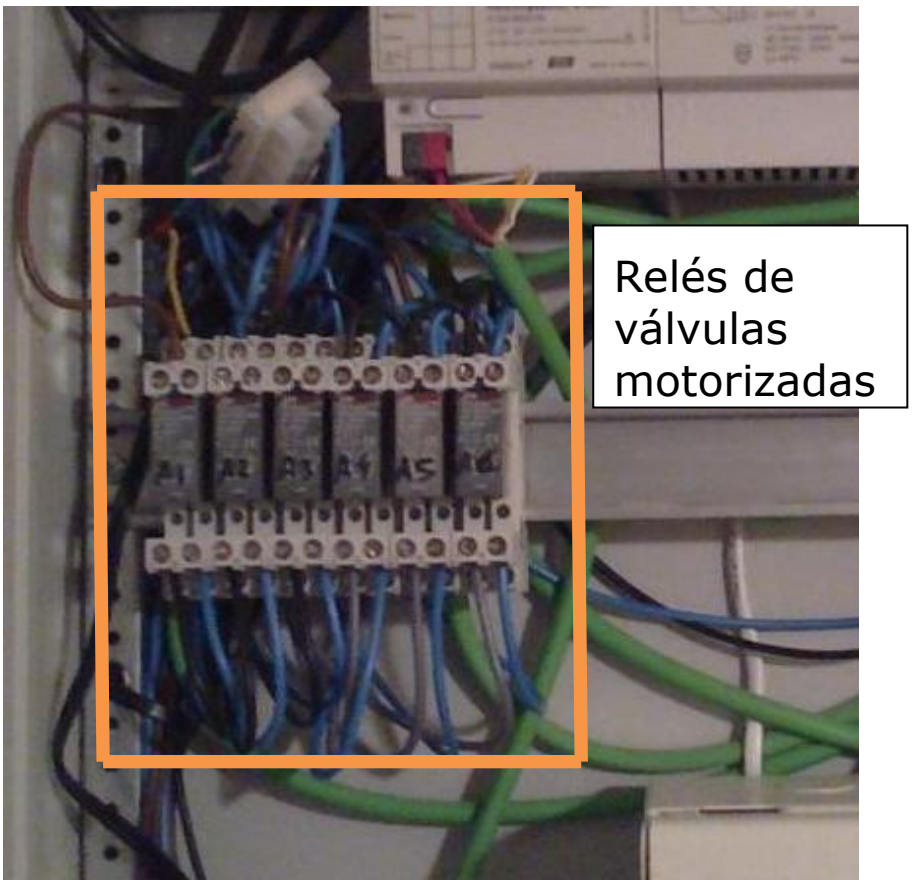
Actuaremos sobre unas válvulas motorizadas de dos vías que son las que impulsan el agua de cada circuito al los radiadores. La caldera mientras esté encendida mantendrá el agua en circulación a la temperatura fijada en la caldera. Cuando las válvulas motorizadas no estén activadas se cierran.

Válvulas motorizadas:

Para actuar sobre estas válvulas, lo haremos mediante unos relés que pueden observarse en el cuadro eléctrico:



En detalle:

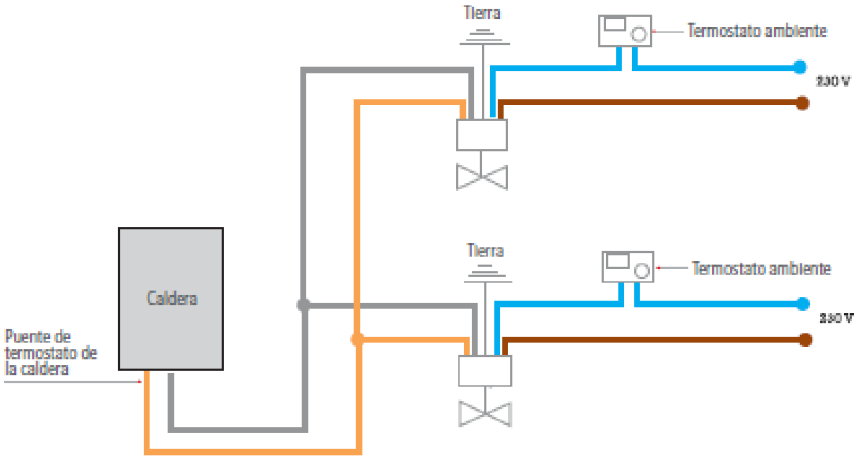


En el apartado de simulación como no disponemos de este cuadro un led indicará el estado de activación o desactivación del relé.

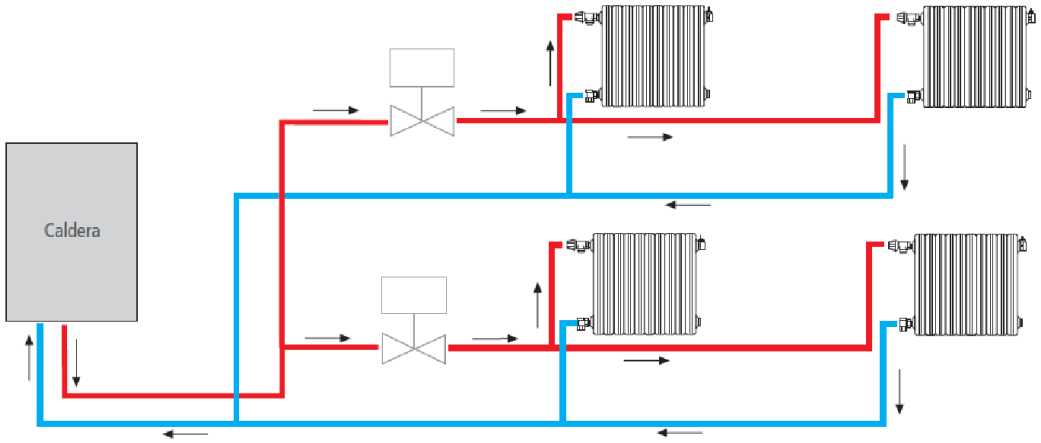
Como el sistema de control desarrollado por la plataforma Arduino permite salidas de 5V y 40mA como máximo no podemos actuar sobre el circuito de potencia, por lo que usaremos unos relevadores para poder cortar las líneas que alimentan cada circuito de climatización.

Las válvulas motorizadas están colocadas a la salida del circuito, próximo a la caldera.

Un ejemplo de cómo será la instalación eléctrica de las válvulas se puede ver en la imagen. El cable de control de la caldera estará inoperativo ya que la señal se control de las válvulas la daremos desde nuestro propio sistema de control. El termostato de la figura representa nuestro sistema, pero además existe un relé entre nuestro sistema y la válvula.



La instalación hidráulica, como se puede apreciar en la imagen es sencilla. En nuestro caso existe un tercer circuito, pero es idéntico o a los otros 2 representados. El agua caliente circulará por los circuitos cuya válvula esté activada, calentando la estancia.



6. Procedimiento y Estructura del Sistema de Control.

Se trata de un proceso de automatización y control de tres temperaturas. En concreto son tres salas, separadas eléctricamente por tres circuitos que son los que realmente procederemos a su activación/desactivación.

Las tres salas:

SALA 1: Dispondrá de una temperatura actual definida por el sensor de temperatura LM35; el usuario fijará una temperatura de referencia (web/LCD) y el circuito de activación se encenderá cuando la temperatura actual sea inferior a la fijada, de este modo se activará el relé encargado de activar el calentador de la sala. La sala comenzará a calentarse y cuando alcance la temperatura de referencia desactivará el circuito y se apagará el calentador.

SALA 2: Dispondrá de una temperatura actual definida por el sensor de temperatura LM35; el usuario fijará una temperatura de referencia (web/LCD) y el circuito de activación se encenderá cuando la temperatura actual sea inferior a la fijada, de este modo se activará el relé encargado de activar el calentador de la sala. La sala comenzará a calentarse y cuando alcance la temperatura de referencia desactivará el circuito y se apagará el calentador.

SALA 3: Dispondrá de una temperatura actual definida por el sensor de temperatura LM35; el usuario fijará una temperatura de referencia (web/LCD) y el circuito de activación se encenderá cuando la temperatura actual sea inferior a la fijada, de este modo se activará el relé encargado de activar el calentador de la sala. La sala comenzará a calentarse y cuando alcance la temperatura de referencia desactivará el circuito y se apagará el calentador.

El control de las salas en todo momento es individual, sin interferencias entre ellas, de este modo se controla de una manera real e independiente las salas que queramos.

Existen dos modos de funcionamiento: Los modos los fijaremos a través de la LCD, la primera pantalla que aparecerá será la de fijar AUTOMATICO/MANUAL.

Una vez que alimentamos el arduino y comienza la secuencia del programa, aparecerá una pantalla de inicio con el título y el nombre de los creadores del proyecto. Después desaparecerá la pantalla en cosa de unos segundos definidos por código y aparecerá la pantalla de selección de modo.

El flujo del programa lo vamos llevando a través de un pulsador y un potenciómetro. El pulsador es una entrada digital que funciona a modo de “enter” para guardar configuración y pasar al siguiente menú. El potenciómetro funciona a modo de selector, es una entrada analógica definida de varias formas según la función que realicemos: SI/NO, 10ºC/26ºC, Manual/Automático…

Modos de funcionamiento:

- Manual: Si elegimos manual, se fijará todo a través de la LCD, con el potenciómetro seleccionaremos la temperatura de referencia que queremos

alcanzar, los circuitos que queremos encender y el proceso se quedará funcionando de modo manual hasta que reiniciemos el ciclo. Se enviarán datos a la web de manera orientativa, pero cualquier cambio de la web no se verá afectado en este modo.

- Automático: Si en la primera pantalla seleccionamos automático entrará en la función automático y leerá y escribirá en los ficheros del servidor web. Por lo tanto se efectuarán las acciones que se desarrollen en la web.

Al tener el GSM shield todo quedará impregnado en un servidor que podremos acceder de forma remota y cambiar las configuraciones cuando se desee.

Por ejemplo: Queremos encender la calefacción de un sitio lejano, al cual se llegará en un tiempo de 2 horas, y queremos que esté a la temperatura ideal cuando lleguemos. Esto es posible en nuestro sistema.

Podremos monitorizar todas las temperaturas actuales de las salas desde cualquier lugar, accediendo a la página web. También se observarán las temperaturas en la pantalla LCD en el lugar de control.

Cuando hemos seleccionado todo y se queda la pantalla final con las temperaturas y la indicación de ON/OFF de cada sala dependiendo si está o no activada según las condiciones anteriormente descritas; existe una cuarta fila, donde se escribirá una frase de 20 caracteres máximo determinada por el usuario.

Esta frase se debe cargar por el monitor serial del arduino así que si queremos dejar algún mensaje en esa fila debemos arrancar el arduino con el USB para arrancar el monitor serial. Escribiremos la frase y todo seguirá igual, sólo que en la pantalla final aparecerá la frase deseada.

Esto está a modo indicativo, si se quiere dejar algún tipo de recado y que se lea por la LCD. Por ejemplo: “Abrir las ventanas”, “He salido”, “Vuelvo por la noche”. Es un factor añadido al proyecto.

Hay otra función extra para darle al proyecto más valor añadido que es un sistema de alarma, aprovechando la existencia del GSM y aprendiendo todas sus funcionalidades (enviar/recibir web), existe la posibilidad de enviar mensajes de texto SMS a cualquier móvil.

De esta manera si la temperatura supera un valor impuesto por código, enviará un sms de alerta al usuario. “Problema de incendio o fallo en el sistema”. La temperatura máxima fijada por código hemos puesto cerca de 30ºC para poder simular esta función, falseando el sensor de temperatura con el calor de la mano. En la realidad ese valor indicará o incendio o avería en algún componente electrónico.

Se observa que manejamos varias entradas/salidas del arduino, aprovechamos en gran parte el gsm shield y utilizamos el puerto serial. Manejamos una interaface LCD, y resolvemos las comunicaciones vía web.

7. Fundamentos teóricos.

Los pines del Arduino pueden configurarse como entradas o salidas. Este documento explica el funcionamiento de los pines en esos modos. Si bien el título de este documento se refiere a los pines digitales, es importante señalar que la gran mayoría de los pines analógicos de Arduino (Atmega), pueden configurarse y utilizarse, exactamente de la misma manera que los pines digitales.

Propiedades de los Pines Configurados como Entrada (INPUT)

Los pines de Arduino (Atmega) por defecto son de entrada, por lo que no es necesario configuraros explícitamente como entradas con pinMode(). Se dice que los pines configurados como entradas están en estado de alta impedancia. Una forma de explicar esto es que los terminales de entrada hacen demandas extremadamente pequeñas en el circuito que están muestreando, se dice que equivale a una resistencia en serie de 100 megaohmio frente al pin. Esto significa que se necesita muy poca corriente para pasar el pin de entrada de un estado a otro, y puede hacer posible el uso de los pines para tareas como la utilización de un sensor capacitivo al tacto, la lectura de un LED como un fotodiodo, o la lectura de un sensor analógico con un esquema como el RCTime.

Esto también significa sin embargo, que los terminales de entrada sin conectar nada a ellos, o con los cables conectados a ellos sin estar conectados a otros circuitos, reflejarán cambios aparentemente aleatorios en el estado de pin, recogiendo el ruido eléctrico del entorno, o el acoplamiento capacitivo del estado de un pin próximo.

Nosotros tenemos el pulsador S1 como este tipo de entrada digital.

Propiedades de los Pines Configurados como salida (OUTPUT)

Los pines configurados como salida (OUTPUT) con pinMode() se dice que están en un estado de baja impedancia. Esto significa que puede proporcionar una cantidad sustancial de corriente a otros circuitos. Los pines del Atmega pueden proporcionar corriente positiva o proporcionar corriente negativa de hasta 40 mA (miliamperios) a otros dispositivos o circuitos. Esta es suficiente corriente para la brillante luz de un LED (no te olvides de la resistencia en serie), o para utilizar muchos sensores por ejemplo, pero no es corriente suficiente para utilizar la mayoría de relés, solenoides o motores.

Los cortocircuitos en los pines de Arduino, o intentos de extraer mucha corriente de ellos, pueden dañar o destruir los transistores de salida en el pin, o dañar completamente el chip Atmega. A menudo, esto se traducirá en un pin del microcontrolador "muerto", pero el resto del chip seguirá funcionando adecuadamente. Por esta razón es buena idea conectar los pines de salida a otros dispositivos con resistencias de 470Ω o 1k, limitando la corriente máxima que desde los pines es requerida para una aplicación particular.

En nuestro caso cada led que simula la salida de un relé va acompañado de una resistencia:

Como no usaremos el pin digital 13 de Arduino para conectar el LED necesitamos una resistencia de unos 200 ohmios (el valor estándar más cercano es de 220 ohmios, pero como no disponíamos de ellas pusimos de 270 ohmios), el cálculo:

- Un LED normal necesita de entre 5 y 20 mA para encenderse, por lo que necesitaremos unos 15mA para que el LED luzca bien.
- Cada salida del arduino proporciona 5 voltios.
- En el LED caerán 2 voltios aproximadamente, dependerá del color del led.
- En la resistencia deben caer unos 3 voltios.

Por lo tanto:

5 - 2 = 3 voltios de tensión

Aplicando la ley de Ohm (Tensión = Intensidad * Resistencia):

V = I * R

Puesto que conocemos V e I podremos calcular R:

R = V / I

Con los valores:

R = 3 / 0.015 = 200 ohmios

(donde "0.015" serán los 15mA que necesita el LED pasados a amperios)

Por lo tanto necesitaremos la resistencia del mercado que más se aproxime a 200 ohmios, la que tenemos a mano es de 270 ohmios.

Pins de Entrada Analógica

Conversor A/D

El controlador Atmega que usa Arduino lleva incluido un conversor analógico-digital (A/D) de 6 canales. Tiene una resolución de 10 bits, retornando enteros desde 0 a 1023. Mientras que el uso principal de estos pines por los usuarios de Arduino es para la lectura de sensores analógicos, estos pines tienen también toda la funcionalidad de los pines de entrada-salida de propósito general (GPIO) (al igual que los pines 0 - 13).

Consecuentemente, si un usuario necesita más pines de propósito general de entrada-salida, y no se está usando ningún pin analógico, estos pines pueden usarse como GPIO.

Mapeo de Pins

Los pines de Arduino correspondientes a los pines analógicos son desde el 14 al 19. Observa que esto son pines de Arduino y no corresponden con los números de los pines físicos del chip Atmega. Los pines analógicos, pueden usarse de manera idéntica que los digitales, así que por ejemplo, podrías ver un código como este para configurar un pin analógico y establecerlo a HIGH:

```
pinMode(14,                                    OUTPUT);
digitalWrite(14,                                HIGH);
```

Resistencias Pullup

Los pines analógicos también tienen resistencias pullup, las cuales funcionan igual que en los pines digitales. Se activan cuando usamos instrucciones como la siguiente

digitalWrite(14, HIGH); // activa la resistencia pullup en el pin analógico 0

Mientras el pin es una entrada (input).

Ten en cuenta que si, por descuido, activamos una resistencia pullup, mientras usamos algunos sensores, los valores obtenidos por analogRead() se verán afectados. La mayoría de los usuarios usarán las resistencias pullup cuando usen un pin analógico como digital.

La función utilizada es analogRead():

Lee el valor de tensión en el pin analógico especificado. La placa Arduino posee 6 canales (8 canales en el Mini y Nano y 16 en el Mega)conectados a un conversor analógico digital de 10 bits. Esto significa que convertirá tensiones entre 0 y 5 voltios a un número entero entre 0 y 1023. Esto proporciona una resolución en la lectura de: 5 voltios / 1024 unidades, es decir, 0.0049 voltios (4.9 mV)por unidad. El rango de entrada puede ser cambiado usando la función analogReference().

El conversor tarda aproximadamente 100 microsegundos (0.0001 segundos) en leer una entrada analógica por lo que se puede llevar una tasa de lectura máxima aproximada de 10.000 lecturas por segundo.

Ese valor de 0 a 1023 es con el que trabajaremos en las funciones potenciometro_Si_No, potenciómetro_T…

Valor analógico	<513	>513
potenciómetro_Si_No	ON	OFF
potenciómetro_Si_No	Manual	Automático

Vamos a explicar la función potenciometro_T

Al igual que la anterior, lee un valor analógico, pero en este caso nos interesa que comprenda un rango de temperaturas.

Esta será la temperatura de referencia que el usuario marca a través del potenciómetro.

Se podrá cambiar también esa temperatura de referencia desde la web pero entonces no estaríamos en modo manual donde leemos esta entrada.

Fijamos el rango que queremos, en este caso de 10ºC a 26ºC temperaturas ambiente razonables para acondicionar una sala.

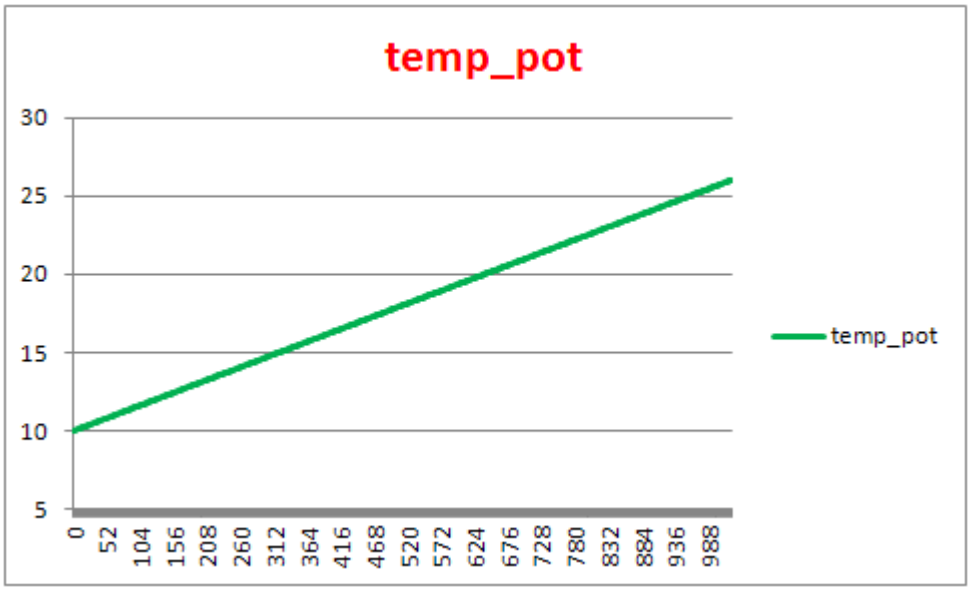
Sabemos que el rango va de 0 a 1023 por lo que desarrollamos la ecuación: $temp_{pot} = (valor_{pot1} + 639.375) \div 63.9375$

Lectura analógica	temp_pot
0	10
1	10,0156403
2	10,0312805
3	10,0469208
4	10,0625611
5	10,0782014
6	10,0938416
7	10,1094819
8	10,1251222
9	10,1407625
10	10,1564027
11	10,172043
12	10,1876833
13	10,2033236
14	10,2189638
15	10,2346041
16	10,2502444
17	10,2658847
18	10,2815249
19	10,2971652
20	10,3128055

No están representados todos los valores

1003	25,6871945
1004	25,7028348
1005	25,7184751
1006	25,7341153
1007	25,7497556
1008	25,7653959
1009	25,7810362
1010	25,7966764
1011	25,8123167
1012	25,827957
1013	25,8435973
1014	25,8592375
1015	25,8748778
1016	25,8905181
1017	25,9061584
1018	25,9217986
1019	25,9374389
1020	25,9530792
1021	25,9687195
1022	25,9843597
1023	26

Desarrollamos los valores en una tabla Excel y sacamos la gráfica que se rige por la ecuación mencionada anteriormente.



Se puede acotar el rango y mejorar la sensibilidad de una entrada analógica cambiando la tensión de referencia usada por el conversor A/D del arduino para ello se puede usar la función analogreference.

Configura el voltaje de referencia usado por la entrada analógica.La función analogRead() devolverá un valor de 1023 para aquella tensión de entrada que sea igual a la tensión de referencia. Las opciones son:

- DEFAULT: Es el valor de referencia analógico que viene por defecto que de 5 voltios en placas Arduino de y de 3.3 voltios en placas Arduino que funcionen con 3.3 voltios.
- INTERNAL: Es una referencia de tensión interna de 1.1 voltios en el ATmega168 o ATmega328 y de 2.56 voltios en el ATmega8.
- EXTERNAL: Se usará una tensión de referencia externa que tendrá que ser conectada al pin AREF.

Filtrado de la señal analógica:

Para evitar que el ruido del circuito nos proporcione mediciones falsas, se acopla el condensador que aparece en los esquemas y así filtramos el ruido que nos afecta a la señal. Se debe colocar entre la entrada analógica y masa.

Comparativa de temperaturas y Histéresis

Si comparamos continuamente la tensión de referencia con la obtenida por el sensor LM35 obtenemos lo siguiente:

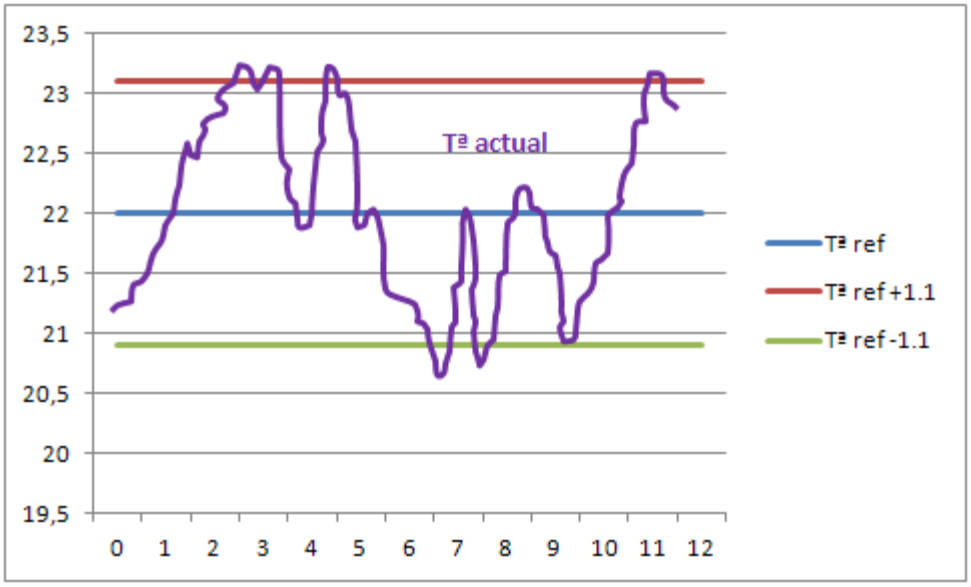
	S_1 ON	S_1 OFF	S_2 ON	S_2 OFF	S_3 ON	S_3 OFF
Tªref<T_sensor	OFF	OFF	OFF	OFF	OFF	OFF
Tªref>T_sensor	SALA 1 ON	OFF	SALA 2 ON	OFF	SALA 3 ON	OFF

Como vemos en la tabla se debe cumplir la condición que la temperatura que queramos sea mayor que la que existe y que hayamos previamente activado el circuito. Entonces se activarán los relés de salida que en nuestro caso se simulará con un LED.

Cuando alcancemos el punto crítico, si esta temperatura sube y baja lo suficientemente rápido tendremos un problema, ya que el led empezará a parpadear, encenderse y apagarse continuamente hasta que se estabilice. Con un led no pasa nada pero si se trata de activar calderas, el arranque de esas máquinas supone mucha energía de activación que conlleva un gasto importante.

Por lo que realizaremos la media de 10 temperaturas y un ciclo de histéresis para conseguir un rango mayor y no ocurra esto.

Observamos la siguiente gráfica:



La temperatura en morado es la temperatura que mide el sensor. Cuando alcance la de referencia (azul) no pasará nada, cuando llegue a la máxima cambiará la salida a 0 para que deje de calentar, entonces, si se producen cambios bruscos que pasen a cruzar la Tªref (azul) no cambiará su estado.

La salida pasará a 1 (aplicamos calor) cuando pise la Tªref (verde). Si existen en la activación oscilaciones que lleguen a sobrepasar la azul no pasaría de nuevo nada. Hasta que alcance la roja y se termina el ciclo.

LM35

Este es el sensor seleccionado para medir la temperatura. Se adjuntará su datasheet al final del documento.

Deberemos alimentarlo a 5 V y conocer su patillaje.

Se conocerá su sensibilidad. 10mV/ºC. De esta manera queda demostrada la función de transferencia usada en la función sensor_temp.

Este sensor para que admita temperaturas negativas, se debe alimentar con una fuente simétrica o realizar un circuito con dos diodos para simular temperaturas negativas. En este caso como la temperatura de las salas las suponemos siempre positivas, no haría falta.

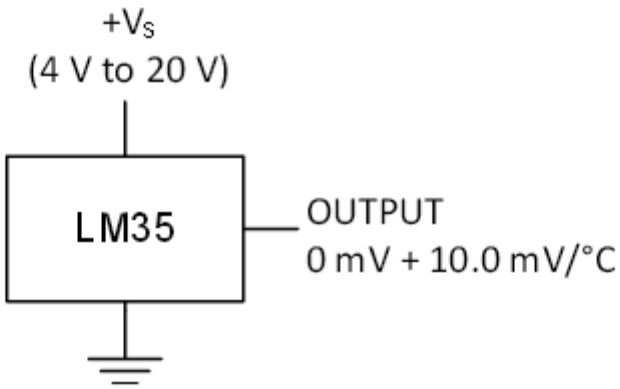
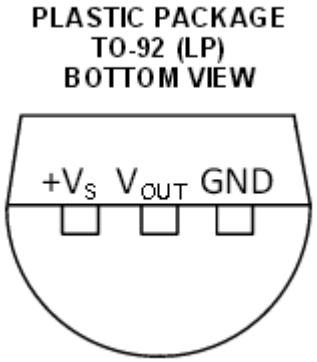
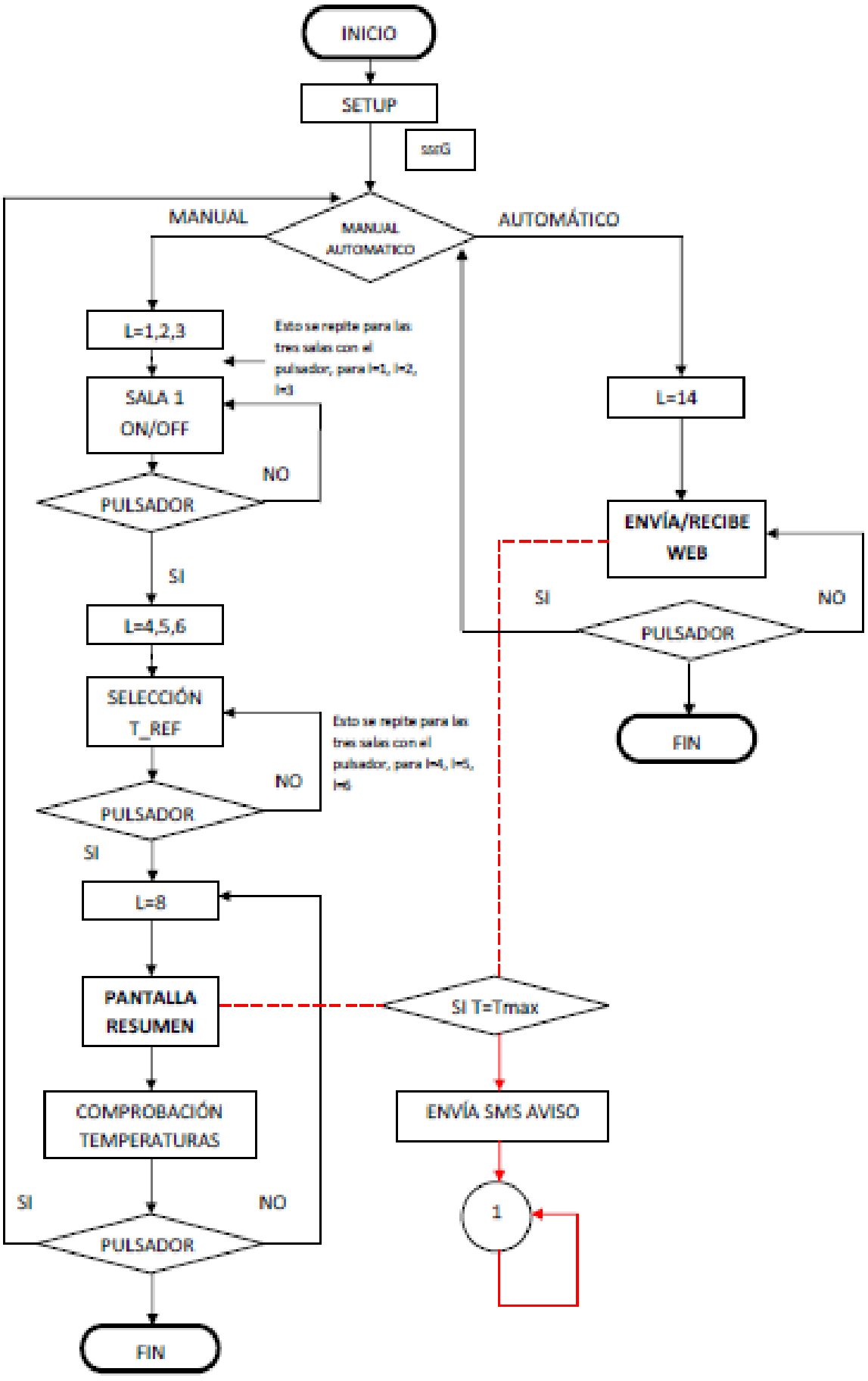


Figure 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



8. Flujo del programa de arduino.

Se ha desarrollado un diagrama de flujo para observar el funcionamiento y seguimiento del transcurso del sistema.


INICIO
SETUP
MANUAL
MANUAL AUTOMATICO
AUTOMÁTICO
L=1,2,3
Esto se repite para las tres salas con el pulsador, para l=1, l=2, l=3
SALA 1 ON/OFF
NO
PULSADOR
SI
L=4,5,6
SELECCIÓN T_REF
Esto se repite para las tres salas con el pulsador, para l=4, l=5, l=6
NO
PULSADOR
SI
L=8
PANTALLA RESUMEN
COMPROBACIÓN TEMPERATURAS
SI
NO
PULSADOR
FIN
L=14
ENVÍA/RECIBE WEB
SI
NO
PULSADOR
FIN
SI T=Tmax
ENVÍA SMS AVISO
1

8.1. Inicio.

Indicamos las librerías que usaremos para controlar los diferentes módulos.

LCD: **#include <LiquidCrystal.h>**

GSM: **#include <GSM.h>**

Conversión de datos char: **#include <stdio.h>** **#include <stdlib.h>**

Declaramos las variables globales que usamos:

Definimos valores constantes:

Nombre Variable	Tipo Variable	Valor	Uso
PINNUMBER	Global	Vacío	Pin de la tarjeta SIM en este caso la tarjeta SIM la hemos desactivado el PIN para entrar fácil y evitar problemas
GPRS_APN	Global	gprs-service.com	APN del operador de la tarjeta
GPRS_LOGIN	Global	Vacío	Login del operador APN. Se debe pedir a la operadora.
GPRS_PASSWORD	Global	Vacío	Password del operador APN
server	Char	www.ubumicros.hol.es	Servidor WEB
port	int	80	Puerto por defecto
lecturas	int	{1,2,3,4,5,6}	array que guarda los datos leídos de la web
Ismael	word	6655	Contraseña para escribir datos
dir	char	{"GET /textos/webRefSala1.txt","GET	Direcciones de los ficheros

		/textos/webRefSala2.txt","GET /textos/webRefSala3.txt","GET /textos/webSala1.txt","GET /textos/webSala2.txt","GET /textos/webSala3.txt"}	
tiempo_ant	long	0	iniciamos tiempo anterior a 0
delta	long	20	tiempo en segundos para comunicación GSM
Str2	char	Introduce texto	Char donde guardamos el texto que mostramos
pulsador	int	6	Pin del pulsador
l	int	0	Variable que crea el flujo del programa
n	int		Variable que recorre Str2
n1	int		Valor que devuelve la función Serial Char
M_A	bool		0-Manual 1-Automático
s_1	bool		Variable que muestra la activación o desactivación de la sala 1
s_2	bool		Variable que muestra la activación o desactivación de la sala 2
s_3	bool		Variable que muestra la activación o desactivación de la sala 3
p_1	bool		Potenciómetro ON/OFF

salida1	bool		Estado de la salida//encendido/apagado
salida2	bool		Estado de la salida//encendido/apagado
salida3	bool		Estado de la salida//encendido/apagado
modo	bool	0	Auto/Manual
estado_pulsador	int	0	Estado que recoge de la lectura del pulsador
grados_s1	float		Variable que recoge los grados del sensor LM35 de la sala 1.
grados_s2	float		Variable que recoge los grados del sensor LM35 de la sala 2.
grados_s3	float		Variable que recoge los grados del sensor LM35 de la sala 3.
grados_s1_ant	float	-1	Variable que recoge el último valor de temperatura para realizar la actualización en caso de cambio sustancial
grados_s2_ant	float	-1	Variable que recoge el último valor de temperatura para realizar la actualización en caso de cambio sustancial
grados_s3_ant	float	-1	Variable que recoge el último valor de temperatura para realizar la actualización en caso de

			cambio sustancial
pinLM35_s1	int	A4	Entrada analógica del sensor de temperatura LM35. A4. Sala 1
pinLM35_s2	int	A5	Entrada analógica del sensor de temperatura LM35. A5. Sala 2
pinLM35_s3	int	A6	Entrada analógica del sensor de temperatura LM35. A6. Sala 3
temp_pot	float		Temperatura que leemos de seleccionar con el potenciómetro
temp_pot_ant	float	-1	Temperatura de referencia anterior
temp_pot_s1	float		Temperatura de referencia de la sala 1
temp_pot_s2	float		Temperatura de referencia de la sala 2
temp_pot_s3	float		Temperatura de referencia de la sala 3
valor_pot1	int	A0	Pin de lectura del potenciómetro.

8.2. Setup

Esta función void setup(){}; sólo se inicia una vez. Cada vez que se resetea el arduino esta función se ejecuta una vez y jamás se vuelve a ejecutar a no ser que reiniciemos el sistema. Es una función de configuración, dónde le vamos a establecer las configuraciones oportunas que establecemos.

- Inicializamos la LCD.
- Conectamos el GSM.

- Realizamos una verificación de conexión de GSM.
- Establecemos el texto que aparecerá durante 5 segundos en la LCD.
- Establecemos los pines analógicos y digitales como salidas/entradas.

Esta configuración se guardará siempre que no se reinicie el sistema.

8.3. Modo manual.

Una vez que entramos en la función void loop(){} que se realizará de manera cíclica siempre que no la paremos en algún sitio determinado. Esta función se repetirá infinitamente.

La variable global l es la que nos indica el flujo del programa.

En la segunda pantalla de la LCD después de la de inicio seleccionaremos mediante el potenciómetro el modo manual/automático. A medida que giramos el potenciómetro se nos mostrará en la LCD el valor en el que estamos. Una vez que queramos ese valor (manual/automático) pulsamos el pulsador.

El pulsador es una entrada digital (5V/0V) mediante la lectura de esa entrada nosotros sabremos si está a nivel alto (5V) o a nivel bajo (0V). En cuanto se encuentre a nivel alto pasamos de menú, de tal manera que incrementamos una unidad la variable l y pasaríamos a la siguiente pantalla.

En el modo manual deberemos ir pasando de pantallas y eligiendo el valor de temperatura que deseemos y si queremos o no activar esa sala. Luego al final se quedará en una pantalla final de monitorización donde automáticamente se irá observando el valor del las temperaturas y si está o no activado ese circuito.

8.4. Modo automático.

Cuando en la segunda pantalla seleccionamos modo automático, el sistema se irá automáticamente a la pantalla final y se realizarán las funciones de enviar y recibir de la web.

De esta manera tenemos una comunicación establecida vía web y se podrá controlar de forma remota.

Lo primero que se comprueba es que no se haya pulsado el pulsador para cambiar de modo. Si no se ha pulsado se miden las temperaturas para actualizar los datos de las variables y se continúa con el resto del código.

Si se ha superado el tiempo establecido en el parámetro delta, entonces se envían los datos a la web para actualizarlos y luego se leen del servidor los datos que se hayan introducido en el mismo por si hay que actualizar los parámetros de funcionamiento.

El resto del código actualizará los valores de temperatura en la pantalla de los tres circuitos, que son iguales. Dentro de este código se comprueba si la sala está activada o no, y si lo está, si temperatura medida es superior o inferior a la referencia fijada

desde la página web para activar o desactivar las salidas de cada circuito. Si la sala está desactivada, la temperatura se mostrará, pero la salida estará siempre desactivada.

La activación y desactivación de las salidas tiene un ciclo de histéresis fijado en 1.1ºC con la misión de que la salida no oscile. Esto se ha conseguido mediante la expresión que está en las condiciones dentro de las sentencias IF y ELSE IF.

En el modo manual la activación y desactivación de las salidas tiene el mismo funcionamiento.

Este ciclo se repetirá indefinidamente salvo que salgamos del modo mediante el pulsador.

8.5. Funciones.

Hemos dividido el trabajo en varias funciones que llamaremos varias veces a lo largo del flujo del programa para ahorrar código y poder extraerlas a otros proyectos.

- **F_Salas_ONOFF:**
  - Esta función aparece en la tercera pantalla de la LCD y se repite para las tres salas. Muestra la activación o desactivación de cada relé correspondiente a cada circuito.
  - Llamamos a la función potenciometro_Si_No que es la encargada de seleccionar un SI/No mediante la entrada analógica de un potenciómetro.
  - Introducimos como parámetro de entrada a_l que adquirirá el número de la sala cada vez que se la llame en la función principal
  - Es una función vacía (tipo void) por lo que no retornamos ningún valor, pero si guardamos valores en variables globales s_1…

- **F_Salas_ONOFF:**
  - Después de pasar la función anterior pasamos a seleccionar la temperatura deseada en la siguiente pantalla del menú.
  - Esta función a su vez llamará a la función potenciometro_T donde seleccionaremos la temperatura.
  - Iremos muestreando esa temperatura en la pantalla con una serie de filtros para que no esté variando continuamente. Controlamos la sensibilidad del dato en todo momento guardando estados anteriores y comparando (ver fundamentos teóricos).

- **potenciometro_Si_No:**
  - Es una función tipo boolean porque en un principio se pensó pasar ese valor, pero por una serie de circunstancias se quitó el retorno y actuaría como una función vacía.
  - Esta función sabe en qué lugar de la pantalla LCD está de esta manera lee una entrada analógica y determina si el valor es menor o mayor que la mitad de la resolución que ofrece el arduino.
    - Si valor<513=ON/Manual
    - Si valor>513=OFF/Automático
  - Muestra por pantalla cada estado.

- **potenciometro_T:**
  - Lee una entrada analógica A0.
  - Resuelve una ecuación que rige valores de 10º a 26ºC

- **sensor_temp:**
  - Es la función donde se lee del sensor de temperatura LM35 el valor que se obtiene.
  - Se hace una media de 10 valores para obtener una medida más exacta.

- **serial_char:**
  - En la pantalla final lee cuando haya algo en el puerto serie y escribe en la última línea de la LCD esa frase en forma de aviso.

- **enviar:**
  - Se llamará en los dos modos automático/manual, para enviar constantemente los datos a tiempo real al servidor.

- **leeweb:**
  - Función que lee de la web cada archivo de texto.
  - Como parámetro de entrada tiene la dirección del archivo
  - Como dato de salida retorna el archivo leído.
  - Se llamará cada vez para cada dato.

- **recibir2:**
  - Esta función recorre todos las direcciones del servidor de las que se leen datos y lee los datos de cada una de ellas.
  - Llamaremos aquí a la función anterior.
  - Se asignan los datos leídos a las variables globales correspondientes.

9. Presupuesto.

MATERIALES	CANTIDAD	PRECIO UNIDAD	PRECIO	ENLACE DE COMPRA
ARDUINO UNO	1	22	22	http://www.bricogeek.com/shop/arduino/305-arduino-uno.html
SHIELD GSM	1	78,9	78,9	http://www.bricogeek.com/shop/shields-arduino/540-arduino-gsm-shield.html
SENSOR DE TEMPERATURA LM35	3	1,7	5,1	TIENDA FÍSICA
CONDENSADORES 1uF	3	0,5	1,5	TIENDA FÍSICA
RESISTENCIAS (10KOHMIOS, 270KOHMIOS)	4	0,15	0,6	TIENDA FÍSICA
LED	3	0,2	0,6	TIENDA FÍSICA
RELÉ DE ACTIVACIÓN	3	7,6	22,8	http://www.bricogeek.com/shop/kits-electronica-para-montar/427-kit-rele-20a.html
PULSADOR	1	0,5	0,5	http://www.bricogeek.com/shop/componentes/298-pulsador-switch-12mm.html
POTENCIÓMETRO	2	2	4	TIENDA FÍSICA
LCD 20X4	1	16,1	16,1	http://www.bricogeek.com/shop/pantallas-lcd-oled/178-pantalla-lcd-20x4-caracteres-negro-verde.html
PROTOBOARD	1	10	10	TIENDA FÍSICA
CABLEADO Y OTROS	1	10	10	TIENDA FÍSICA
HERRAMIENTA (FUENTE DE ALIMENTACIÓN, OSCILOSCOPIO…)	1	10	10	TIENDA FÍSICA
			182,1	
		INCREMENTO +IVA(21%)	**220,34**	

TAREA	HORAS	PRECIO/HORA	PRECIO
INGENIERO INDUSTRIAL ELECTRÓNICO. ALBERTO	40	22	880
INGENIERO INDUSTRIAL ELECTRÓNICO. ISMAEL	40	22	880
INGENIERO DE SOFTWARE. ALBERTO	30	19	570
INGENIERO DE SOFTWARE. ISMAEL	30	19	570
INSTALADOR. ALBERTO	2	12	24
INSTALADOR. ISMAEL	2	12	24
DESARROLLO DE MEMORIA	20	20	400
	164		**3348**

TOTAL	
DESARROLLO PROYECTO	3348
MATERIALES	220,34
	3568,3

Todos los materiales están comprados en distribuidor oficial, se podría abaratar el coste del material en fabricantes chinos, ya que como se trata de hardware libre estas placas puedes encontrarlas en muchos sitios, con precios más asequibles. Incluso al disponer de los esquemas abiertos, podíamos aprovechar el diseño y fabricarlas nosotros mismos.

Bricogeek y Mouser son sitios aceptable para la adquisición de estos módulos.

Las horas de trabajo son orientativas ya que ha sido un trabajo arduo.

Se ha dividido en varias facetas, ya que se ha trabajado como distintos puestos de trabajo.

10. Programa en el servidor.

10.1. Inicio.

Es la página que muestra el servidor por defecto cuando entramos en la dirección del servidor desde un navegador web. Existen varios nombres de página que el servidor tomará como página de inicio. En nuestro caso ponemos el nombre default.php.

Nuestra página de inicio tendrá un texto de bienvenida, un enlace a la página de logout por si tenemos una sesión abierta que queramos cerrar y un formulario para introducir los datos de usuario y contraseña para crear una sesión. Estos datos serán enviados a la página de login para su verificación y creación de sesión.

Si la sesión existe en el momento de abrir esta página, se mostrará un enlace a la página de formulario en vez del formulario de validación.

10.2. Login.

Esta página no se muestra en ningún momento, solo tiene código php. Es llamada cuando se inserta un usuario y contraseña en la página default y su misión es contrastar los valores introducidos en el formulario con la clave de acceso. Si la clave es correcta se creará una sesión y nos llevará a la página de formulario. Si es incorrecta nos devuelve a la página default. En esta página se introducirán también el usuario y contraseña que validará al usuario, el valor con el que se comparan los valores introducidos.

10.3. Formulario.

La página de nombre formulario tiene como misión la introducción de datos para el control de la caldera en el modo automático.

A esta página se puede acceder cuando se ha iniciado una sesión, de lo contrario se mostrara un mensaje y un enlace a la página de inicio, al formulario para iniciar sesión. Los formularios para introducir los datos están escritos en leguaje HTML y toda la página muestra texto plano sin apenas formato.

Cuando se introducen los datos en los campos del formulario, estos se escriben en archivos de texto de forma análoga a como lo hacen en la página de datos, pero esta vez los datos son introducidos desde la propia página mediante el método POST en los formularios.

Lo primero que vemos es un desplegable para la selección de la temperatura de una lista. La temperatura seleccionada será el valor que el formulario envía cuando se pulse el botón que tiene la función de enviar. Las acciones de estos formularios contienen un código PHP que escribe el dato seleccionado en un archivo de texto que corresponde al que almacena la temperatura que hemos seleccionado.

El siguiente formulario tiene por acción lo mismo que el anterior, pero esta vez solo escribe un uno o un cero según la opción seleccionada mediante el ratón y después de dar al botón de enviar.

Posteriormente tenemos unas líneas donde se muestra la referencia introducida en la web. Esto sirve para monitorizar la temperatura de referencia actual almacenada en el servidor y compararla con la temperatura actual real. Estas líneas escritas en PHP leen los archivos del servidor que contiene los datos que se quieren leer y los muestran como texto.

Después se muestran de forma idéntica las temperaturas que hay en las salas, las temperaturas que miden los sensores. Estas se monitorizan tanto en el modo automático como en el manual, tanto si la salas están activadas como si no. este dato siempre es interesante.

Después se leen los datos de las salas introducidos en la web, igual que las referencias. En este caso leemos el archivo y escribimos un texto u otro según el dato sea un uno o un cero, y por lo tanto hayamos introducido que la caldera tiene que funcionar o estar apagada. El texto reflejara este estado.

El siguiente texto muestra la fecha guardada en el servidor y que mostraremos con el propósito de saber cuál ha sido la última conexión y verificar que el sistema está bien. Si este dato está actualizado de acuerdo con el tiempo delta de envío que hemos introducido en el programa, entonces sabremos que la conexión está funcionando y que por lo tanto las temperaturas son actuales. Este dato de fecha se actualiza en el servidor a la vez que el resto de datos cuando son enviados por arduino. La fecha se lee del servidor.

Posteriormente tenemos unas líneas donde se muestra la referencia introducida en el modo manual de arduino. Esto sirve para monitorizar la temperatura de referencia y compararla con la actual. Estas líneas escritas en PHP leen los archivos del servidor que contiene los datos que se quieren leer y los muestran como texto. Son datos enviados por arduino y que la página de datos guarda en el servidor cuando los recoge.

Del mismo modo vemos el estado de las salas en arduino. Estos datos se representan igual que los de las salas que se introducen por la web. Pero en este caso los datos llegan de arduino, tanto si esta en modo automático como si esta en modo manual. Esto sirve de monitorización remota y también sirve para comprobar el estado de la conexión y el funcionamiento del sistema en modo automático, ya que estos valores después del tiempo delta deben coincidir con los introducidos en la página web.

Posteriormente mostraremos el dato modo, que no indica si el sistema está funcionando en modo automático o en modo manual. La pagina lee si este dato tiene como valor un uno o un cero y escribe “modo automático” si es un uno, y “modo manual” si no lo es.

Por último se muestra el estado de las salidas de la caldera. Esto sirve para comprobar si las bombas de la calefacción están funcionando en ese instante. Sirve tanto para monitorizar como para comprobar el sistema cuando está en modo automático. Si es un uno el dato, la bomba estará funcionando, sino estará cerrada la válvula. El texto se mostrara en rojo cuando la bomba funcione y en azul cuando no funcione.

10.4. Datos.

Aquí haremos las conexiones con arduino para enviar datos. Todos los datos que queramos almacenar y sean enviados por arduino serán recogidos en esta página.

Esta es la única página que no necesita una sesiona activa para acceder. Si entramos en ella desde el navegador solo veremos el nombre de la página, el resto estará en blanco.

El código php que contiene sirve para recoger los datos que se envían con el método GET, es decir que los datos vienen incluidos como parámetros en la propia dirección que llama a esta página.

Los datos recogidos se guardan en variables y si una de estas, que hace las funciones de contraseña ($comprobación), coincide con el valor configurado en arduino, entonces las variables se escribirán de forma permanente en archivos de texto en el directorio especificado para cada archivo.

También quedará registrada la fecha y la hora de la última conexión con arduino. Con el propósito de verificar el correcto funcionamiento de la conexión.

10.5. Logout.

Esta página no muestra nada en el navegador, solo destruye la sesión que esté iniciada y redirige a la página de inicio. Bastará con pinchar alguno de los enlaces que dirige a esta página para cerrar la sesión.

11. Resultados.

11.1. Variables controladas y monitorizadas.

Los objetivos quedan superados con creces. Tenemos exceptuando la D13 todas las salidas/entradas de la placa usándose.

Se controlan:

- Entradas digitales: Pulsador
- Entradas analógicas: sensores de temperatura y potenciómetro.
- Salidas digitales: relé de activación (led de simulación)
- Se controlan las salidas digitales para la LCD
- Librería LCD.
- Servidor web: html, php.
- Librería GSM.

Es un proyecto bastante completo y con una cierta dificultad que ha sido superado con éxito. Además se trata de un proyecto real que se puede llevar a cabo en cualquier instalación que se parezca a la descrita en el proyecto.

11.2. Vídeo de demostración.

12. Posibles mejoras.

- Memorizar el estado de funcionamiento y las referencias en la memoria Eeprom para reanudar el estado después de un corte en el suministro eléctrico.
- Utilizar un reloj para programar horarios desde la web.
- Añadir control sobre otros dispositivos.
- Creación de un registro de temperaturas para crear gráficas o estadísticas.
- Estimar el gasto de energía.
- Implementar un teléfono con el módulo GSM, teclado, altavoces y micrófono.
- Programar el sistema para que obtenga la hora de la web y programar horarios.
- Programar directamente en el servidor para que en función de la hora que sea y de los horarios seleccionados se active o desactive el sistema.
- Obtener la temperatura exterior de la casa.
- Dividir en más estancias la casa.
- Guardar en el servidor los datos en una base de datos para tener un registro más ordenado de la información y poder operar y manipular los datos de forma más ordenada y eficiente.
- Crear una base de datos de usuarios, para registrar sus entradas, cambiar contraseñas y nombres de usuario.
- Ensamblar el sistema en una PCB con su posterior caja estanca para poder ser colocada en cualquier tipo de armario eléctrico.
- Utilizar modos de comunicación inalámbrica Xbee para llevar las temperaturas de las salas de forma remota, evitando así cablear la instalación.
- Utilizar un circuito de desplazamiento para ahorrar en entradas y salidas del arduino con tres salidas podríamos tener 8 virtuales (multiplexor).

13. Código.

13.1. Arduino.

13.1.1. GSMMicros_final_v6_comentado.ino

```
/*
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++

Programa Sistemas basados en microprocesadores. Alberto García
Izquierdo. Ismael Pérez Rojo

Se elabora el código correspondiente a la plataforma Arduino.

Programa que realiza dos modos:
----Manual: Selección de calderas que queremos activar, junto a la
monitorización de la temperatura actual y
la elección de una temperatura de referencia que ejecutará una salidas
de relé (led) para simular la activación de cada válvula que
representa una zona.

---Automático:Mediante el módulo GSM se conectará arduino a un
servidor web donde realizará consultas y escrituras concretas para
realizar las órdenes que le demos
desde cualquier dispositivo (tablet, móvil, PC)

Extra: Podremos escribir un texto que se ejecutará en la LCD en forma
de aviso, el cual escribiremos de la web, además realizaremos una
función de alerta que nos avisará
mediante un SMS del posible fallo o avería de la instalación.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++

*/
```

```
// include the library code:
#include <LiquidCrystal.h>    //Libreria pra el LCD

// libraries
#include <GSM.h>              //Libreria del módulo GSM
#include <stdio.h>            //Libreria necesaria para hacer las
conversiones de datos de texto
#include <stdlib.h>           //Libreria necesaria para hacer las
conversiones de datos de texto

// PIN Number
#define PINNUMBER ""          //Definimos el PIN de la tarjeta GSM

// APN data
#define GPRS_APN       "gprs-service.com" // APN del operador de la
tarjeta.
#define GPRS_LOGIN      ""                // GPRS login. Vacío si no
existe.
#define GPRS_PASSWORD  ""                 // GPRS password. Vacío si
no existe.

// Objetos de la librería GSM
GSMClient client;
GPRS gprs;
GSM gsmAccess;

char server[] = "www.ubumicros.hol.es";  //Servidor al que nos
conectamos.

int port = 80; // port 80 is the default for HTTP.  Puerto de
conexión.

int lecturas[]={1,2,3,4,5,6};  //array que guarda los datos leidos de
la web
word ismael=6655;             //contraseña para escribir datos en la
web.
char* dir[]={"GET /textos/webRefSala1.txt","GET
/textos/webRefSala2.txt","GET /textos/webRefSala3.txt","GET
/textos/webSala1.txt","GET /textos/webSala2.txt","GET
/textos/webSala3.txt"};
                            //Direcciones de las que leemos los datos
del servidor. De aqui tomamos los datos introducidos en la web.
long tiempo_ant=0;  //iniciamos tiempo anterior a cero.
long delta=20;    //intervalo de tiempo en segundos para establecer
comunicacion GSM


int n; //Variable que recorre caracter a caracter el Str2

int n1=0;
char Str2[20]="Introduce texto";
float Temp;
float Temp1;
```

```
int pulsador = 6;  //Definimos el pin de conexión del pulsador
int l=0;           //Variable que recorre los menús del programa
bool p=1;
bool M_A;
bool s_1;          //Variable que muestra la activación o desactivación
de la sala 1
bool s_2;          //Variable que muestra la activación o desactivación
de la sala 2
bool s_3;          //Variable que muestra la activación o desactivación
de la sala 3
bool p_1;          //Valor que devuelve la función potenciometro en
función de si ponemos ON o OFF
bool salida1;      //Estado de la salida//encendido/apagado
bool salida2;      //Estado de la salida//encendido/apagado
bool salida3;      //Estado de la salida//encendido/apagado
bool modo=0;       //Modo auto o manual. Extensible para informar de
otros estados también.
int estado_pulsador = 0;

float grados_s1;         // Variable que recoge los grados del sensor
LM35 de la sala 1.
float grados_s2;         // Variable que recoge los grados del sensor
LM35 de la sala 2.
float grados_s3;         // Variable que recoge los grados del sensor
LM35 de la sala 3.
float grados_s1_ant=-1;   //Variable que recoge el último valor de
temperatura para realizar la actualización en caso de cambio
sustancial
float grados_s2_ant=-1;   //Variable que recoge el último valor de
temperatura para realizar la actualización en caso de cambio
sustancial
float grados_s3_ant=-1;   //Variable que recoge el último valor de
temperatura para realizar la actualización en caso de cambio
sustancial

int pinLM35_s1 = A4;     //Entrada analógica del sensor de temperatura
LM35. A4. Sala 1
int pinLM35_s2 = A5;     //Entrada analógica del sensor de temperatura
LM35. A5. Sala 2
int pinLM35_s3 = A6;     //Entrada analógica del sensor de temperatura
LM35. A6. Sala 3

float temp_pot;          //Temperatura que leemos de seleccionar con el
potenciómetro
float temp_pot_ant=-1;
float temp_pot_s1;       //Temperatura de referencia de la sala 1
float temp_pot_s2;       //Temperatura de referencia de la sala 2
float temp_pot_s3;       //Temperatura de referencia de la sala 3
int valor_pot1=A0 ;      //Pin de lectura del potenciómetro.
//Analógico.
LiquidCrystal lcd(12, 11, 5, 4, 9, 8);  //Pines usados de la LCD.
//char texto;
```

```
void setup(){

  delta=delta*1000;     //convertimos segundos en milisegundos para
comparar valores con la función millis().

  lcd.begin(20, 4);     //Tipo de display. 20 carcteres. 4 líneas.
  //Serial.begin(9600);

  //Serial.println("inicio");
  lcd.setCursor(0,2);         //ubicamos el curson intruduciendo las
coordenadas en el LCD. (Fila, Columna)
  lcd.print("  Conectando...");  //Texto en el LCD.

   boolean notConnected = true;  //Variable que indica el estado de
conexión del GSM.
      while(notConnected)           //Intenta conectar siempremientras
no lo esté.
      {
      if((gsmAccess.begin(PINNUMBER)==GSM_READY) &
      (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN,
GPRS_PASSWORD)==GPRS_READY)){
      notConnected = false;
            //introducimos el PIN, APN, USER, PASWORD. Si conectamos,
reseeamos laa variable y no lo volvemos a intentar.
//          lcd.clear();
//          lcd.setCursor(0,2);
//          lcd.print("  Conectado");
//          delay(500);
      //Serial.println("conectado");
      }
      else
      {
      //lcd.print("Not connected");
      lcd.clear();
      lcd.setCursor(0,2);
      lcd.print("Error de Conexion");
//          delay(500);
//          delay(1000);
      }
      }
      //lcd.clear();

  //Serial.println("     Bienvenido       ");
  //Mesaje de bienvenida.
  lcd.print("     Bienvenido  ");
  lcd.setCursor(0,1);lcd.print("*Sistema Controlado*");
  lcd.setCursor(0,2);lcd.print("****Calefaccion*****");
  lcd.setCursor(0,3);lcd.print("__Ismael___Alberto__");
  delay(5000);

  pinMode(pulsador, INPUT);  //Configuramos el modo del pin del
pulsador como entrada.
```

```
  pinMode(A1, OUTPUT);         //Configuramos los pines de salida como
salidas.
  pinMode(A2, OUTPUT);         //No tienen un nombre asinado los
activamos directamente.
  pinMode(A3, OUTPUT);
  lcd.clear();


}

void loop()
{
   /*
  **********************************************
  **********************************************
  */
//Elección del modo manual o automáticoç
/*
  **********************************************
  **********************************************
  */

      //activamos o desactivamos el dato que indica el modo atomático.
 if(l==14){
  modo=1;    //modo automático
 }
 else{
  modo=0;    //modo manual
 }

  while (l==0){
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Modo_Funcionamiento");
      potenciometro_Si_No();
      lcd.setCursor(11,3);
      lcd.print("Pulse OK");
    delay(400);
    estado_pulsador = digitalRead(pulsador);
      if (estado_pulsador == HIGH & M_A==0){
      l++;
      delay(300);

      //Serial.println("PULSADORRRR");
      while(digitalRead(pulsador) == HIGH){}
      lcd.clear();
  }
  if (estado_pulsador == HIGH & M_A==1){
      l=14;// PONER EL VALOR DONDE EMPIEZA EL AUTOMÁTICO
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Modo automático");
      lcd.setCursor(0,1);
```

```
      lcd.print("  Iniciando...");
      lcd.setCursor(0,2);
      lcd.print("Esto llevara un rato");
      delay(5000);

//         Serial.println("PULSADORRRR");
//           Serial.println(l);
      while(digitalRead(pulsador) == HIGH){}
      //lcd.clear();
  }
  }
  /*
  ************************************************
  ************************************************
  */
  //Activación de las calderas por zonas.
  /*
  ************************************************
  ************************************************
  */

  while (l==1){  //Activar sala 1
      int a1=1;
      F_Salas_ONOFF(a1);
  }

  while (l==2){   //Activar sala 2
      int a1=2;
      F_Salas_ONOFF(a1);
  }
  while (l==3){//Activar sala 3
    int a1=3;
    F_Salas_ONOFF(a1);
  }
  lcd.clear();
  /*
  ************************************************
  ************************************************
  */
  //Selección de temperatura.
  /*
  ************************************************
  ************************************************
  */

  while (l==4){

      int a1=1;
      sensor_temp();
      F_Salas_Temp(a1);
      estado_pulsador = digitalRead(pulsador);
      if (estado_pulsador == HIGH){
      l++;
```

```
    //delay(300);
    temp_pot_s1=temp_pot;
    grados_s1=grados_s1;
    //Serial.println("PULSADORRRR");
    while(digitalRead(pulsador) == HIGH){}
    lcd.clear();
}
}
while (l==5){

    int a1=2;
    sensor_temp();
    F_Salas_Temp(a1);
    estado_pulsador = digitalRead(pulsador);
    if (estado_pulsador == HIGH){
    l++;
    //delay(300);
    temp_pot_s2=temp_pot;
    grados_s2=grados_s2;
    //Serial.println("PULSADORRRR");
    while(digitalRead(pulsador) == HIGH){}
    lcd.clear();
}
}
while (l==6){

    int a1=3;
    sensor_temp();
    F_Salas_Temp(a1);
    estado_pulsador = digitalRead(pulsador);
    if (estado_pulsador == HIGH){
    l++;
    //delay(300);
    temp_pot_s3=temp_pot;
    grados_s3=grados_s3;
    //Serial.println("PULSADORRRR");
    while(digitalRead(pulsador) == HIGH){}
    lcd.clear();
}
}
/*
**********************************************
**********************************************
*/
//Entrada de texto
/*
**********************************************
**********************************************
*/
while (l==7){
    lcd.print("hola");
    //Serial.print("hola");
    if (estado_pulsador == HIGH){
```

```
      l++;



      while(digitalRead(pulsador) == HIGH){}
      lcd.clear();
  }

  }
  /*
  ***********************************************
  ***********************************************
  */
  //Resumen total
  /*
  ***********************************************
  ***********************************************
  */

  while (l==8){  //Modo manual.

      estado_pulsador = digitalRead(pulsador);  //leemos el estado del
pulsador.
      if (estado_pulsador == HIGH){
      l=0;
      delay(200);
      lcd.clear();
      }

      sensor_temp();

      if (abs(grados_s1 - grados_s1_ant) > 0.2){// Si la diferencia es
mayor que 0.5 muestra el valor nuevo si no no. Así evitamos problemas
de ruido
      grados_s1_ant=grados_s1;
      }

      if (abs(grados_s2 - grados_s2_ant) > 0.2){// Si la diferencia es
mayor que 0.5 muestra el valor nuevo si no no. Así evitamos problemas
de ruido
      grados_s2_ant=grados_s2;
      }

      if (abs(grados_s3 - grados_s3_ant) > 0.2){// Si la diferencia es
mayor que 0.5 muestra el valor nuevo si no no. Así evitamos problemas
de ruido
      grados_s3_ant=grados_s3;
      }


      //Cambiamos la salida de la caldera si procede

      if(s_1==1 & temp_pot_s1-1.1>(grados_s1_ant)){
```

```
digitalWrite(A1,HIGH);  //Orden de encendido de la salida
lcd.setCursor(0,0);
lcd.print("S1: ");
lcd.print(grados_s1_ant);lcd.print("/");lcd.print(temp_pot_s1);
lcd.print("   ON");
salida1=1;  //estado de la salida. // Caldera encendida
}
else if(s_1==1 & temp_pot_s1+1.1<(grados_s1_ant)){

digitalWrite(A1,LOW);   //Orden de apagado de la salida
lcd.setCursor(0,0);
lcd.print("S1: ");
lcd.print(grados_s1_ant);lcd.print("/");lcd.print(temp_pot_s1);
lcd.print("  OFF");
salida1=0;         //estado de la salida. // Caldera apagada.
}
else if(s_1==0){  //si la caldera esta off, que se apague.
digitalWrite(A1,LOW);   //Orden de apagado de la salida
lcd.setCursor(0,0);
lcd.print("S1: ");
lcd.print(grados_s1_ant);lcd.print("/");lcd.print(temp_pot_s1);
lcd.print("  OFF");
salida1=0;         //estado de la salida. // Caldera apagada.
}
else{
//actualizamos las teperaturas actuales.
lcd.setCursor(0,0);
lcd.print("S1: ");
lcd.print(grados_s1_ant);lcd.print("/");lcd.print(temp_pot_s1);

}

if(s_2==1 & temp_pot_s2-1.1>(grados_s2_ant)){
digitalWrite(A2,HIGH);  //Orden de encendido de la salida
lcd.setCursor(0,1);
lcd.print("S2: ");
lcd.print(grados_s2_ant);lcd.print("/");lcd.print(temp_pot_s2);
lcd.print("   ON");
salida2=1;  //estado de la salida. // Caldera encendida
}
else if(s_2==1 & temp_pot_s2+1.1<(grados_s2_ant)){
digitalWrite(A2,LOW);   //Orden de apagado de la salida
lcd.setCursor(0,1);
lcd.print("S2: ");
lcd.print(grados_s2_ant);lcd.print("/");lcd.print(temp_pot_s2);
lcd.print("  OFF");
salida2=0;         //estado de la salida. // Caldera apagada.
}
else if(s_2==0){  //si la caldera esta off, que se apague.
digitalWrite(A2,LOW);   //Orden de apagado de la salida
lcd.setCursor(0,1);
lcd.print("S2: ");
lcd.print(grados_s2_ant);lcd.print("/");lcd.print(temp_pot_s2);
```

```
      lcd.print("  OFF");
      salida2=0;         //estado de la salida. // Caldera apagada.
      }
      else{
      //actualizamos las teperaturas actuales.
      lcd.setCursor(0,1);
      lcd.print("S2: ");
      lcd.print(grados_s2_ant);lcd.print("/");lcd.print(temp_pot_s2);

      }

      if(s_3==1 & temp_pot_s3-1.1>(grados_s3_ant)){
      digitalWrite(A3,HIGH);  //Orden de encendido de la salida
      lcd.setCursor(0,2);
      lcd.print("S3: ");
      lcd.print(grados_s3_ant);lcd.print("/");lcd.print(temp_pot_s3);
      lcd.print("   ON");
      salida3=1;  //estado de la salida. // Caldera encendida
      }

      else if(s_3==1 & temp_pot_s3+1.1<(grados_s3_ant)){
      digitalWrite(A3,LOW);   //Orden de apagado de la salida
      lcd.setCursor(0,2);
      lcd.print("S3: ");
      lcd.print(grados_s3_ant);lcd.print("/");lcd.print(temp_pot_s3);
      lcd.print("  OFF");
      salida3=0;         //estado de la salida. // Caldera apagada.
      }
      else if(s_3==0){  //si la caldera esta off, que se apague.
      digitalWrite(A3,LOW);   //Orden de apagado de la salida
      lcd.setCursor(0,2);
      lcd.print("S3: ");
      lcd.print(grados_s3_ant);lcd.print("/");lcd.print(temp_pot_s3);
      lcd.print("  OFF");
      salida3=0;         //estado de la salida. // Caldera apagada.
      }
      else{
      //actualizamos las teperaturas actuales.
      lcd.setCursor(0,2);
      lcd.print("S3: ");
      lcd.print(grados_s3_ant);lcd.print("/");lcd.print(temp_pot_s3);

      }


   //serial_char();
   //Indicamos en la ultima linea de la pantalla que el modo es manual
   lcd.setCursor(0,3);
   //lcd.print(Str2);
   lcd.print("   Modo manual");
   //Serial.println(Str2);
   //for(n1=n1;n1<(20-n1);n1++){
//Serial.println("forrrrr");
```

```
      //  lcd.print(" ");
   //}

      //Si ha transcurrido el tiempo suficiente se inicia la
comunicacion con la web.

 unsigned long tiempo=millis();

   if(tiempo-tiempo_ant>delta){  //Se compara el incremento de tiempo
con el tiempo programado.
      tiempo_ant=tiempo;            //Se asigna el tiempo actual al
tiempo anterior.
      enviaweb();                   //Se envían los datos a la web
para monitorizacion remota.
    }
  }

      //activamos o desactivamos el dato que indica el modo atomático.
 if(l==14){
  modo=1;
 }
 else{
  modo=0;
 }

  while (l==14){

      //delay(1000);

      //Serial.println("while 14 ");
      estado_pulsador = digitalRead(pulsador);
      if (estado_pulsador == HIGH){
      l=0;
      delay(200);
      lcd.clear();
      }
      sensor_temp();

      //lcd.print("recibiendo");
      //lcd.print(grados_s1);

      unsigned long tiempo=millis();      //asigna el tiempo
transcurrido a la variable tiempo.

   //Si ha transcurrido el tiempo suficiente se inicia la comunicacion
con la web.
   //Se asigna el tiempo actual al tiempo anterior.

      if(tiempo-tiempo_ant>delta){
      tiempo_ant=tiempo;
            enviaweb();             //envía datos a la web
            recibir2();             //lee datos de la web
      }
```

```
      //las funciones de enviar y recibir ocupan bastante tiempo,
durante este proceso la respuesta del botón
      //se volverá mas lenta, pese a haber deteccion del pulsador
dentro de la función recibir.

      //Serial.println("fin de lectura");
      //Serial.println(grados_s1);

      lcd.clear();      //Borra el LCD.

      if(l!=0){   //si se ha pulsado el pulsador no seguimos con el
código. Evita retardos.
      if(s_1==1 & temp_pot_s1-1.1>(grados_s1)){ //ciclo de histeresis
            digitalWrite(A1,HIGH);  //Orden de encendido de la salida
            lcd.setCursor(0,0);
         lcd.print("S1: ");

      lcd.print(grados_s1);lcd.print("/");lcd.print(temp_pot_s1);
            lcd.print("   ON");
            salida1=1;  //estado de la salida. // Caldera encendida

      }
      else if(s_1==1 & temp_pot_s1+1.1<(grados_s1)){  //ciclo de
histeresis

            digitalWrite(A1,LOW);   //Orden de apagado de la salida
            lcd.setCursor(0,0);
            lcd.print("S1: ");

      lcd.print(grados_s1);lcd.print("/");lcd.print(temp_pot_s1);
            lcd.print("  OFF");
            salida1=0;        //estado de la salida. // Caldera
apagada.
      }
      else if(s_1==0){  //si la caldera esta off, que se apague.
            digitalWrite(A1,LOW);   //Orden de apagado de la salida
            lcd.setCursor(0,0);
            lcd.print("S1: ");

      lcd.print(grados_s1);lcd.print("/");lcd.print(temp_pot_s1);
            lcd.print("  OFF");
            salida1=0;        //estado de la salida. // Caldera
apagada.
      }
      else{
      //actualizamos las teperaturas actuales.
            lcd.setCursor(0,0);
            lcd.print("S1: ");

      lcd.print(grados_s1);lcd.print("/");lcd.print(temp_pot_s1);
      }

      if(s_1==2 & temp_pot_s2-1.1>(grados_s2)){ //ciclo de histeresis
```

```
            digitalWrite(A2,HIGH);  //Orden de encendido de la salida
            lcd.setCursor(0,1);
            lcd.print("S2: ");

      lcd.print(grados_s2);lcd.print("/");lcd.print(temp_pot_s2);
            lcd.print("   ON");
            salida2=1;  //estado de la salida. // Caldera encendida

      }
      else if(s_2==1 & temp_pot_s2+1.1<(grados_s2)){  //ciclo de
histeresis
            digitalWrite(A2,LOW);   //Orden de apagado de la salida
            lcd.setCursor(0,1);
            lcd.print("S2: ");

      lcd.print(grados_s2);lcd.print("/");lcd.print(temp_pot_s2);
            lcd.print("  OFF");
            salida2=0;        //estado de la salida. // Caldera
apagada.
      }
      else if(s_2==0){  //si la caldera esta off, que se apague.
            digitalWrite(A2,LOW);   //Orden de apagado de la salida
            lcd.setCursor(0,1);
            lcd.print("S2: ");

      lcd.print(grados_s2);lcd.print("/");lcd.print(temp_pot_s2);
            lcd.print("  OFF");
            salida2=0;        //estado de la salida. // Caldera
apagada.
      }
      else{
      //actualizamos las teperaturas actuales.
            lcd.setCursor(0,1);
            lcd.print("S2: ");

      lcd.print(grados_s2);lcd.print("/");lcd.print(temp_pot_s2);
      }

      if(s_3==1 & temp_pot_s3-1.1>(grados_s3)){ //ciclo de histeresis
            digitalWrite(A3,HIGH);  //Orden de encendido de la salida
            lcd.setCursor(0,2);
            lcd.print("S3: ");

      lcd.print(grados_s3);lcd.print("/");lcd.print(temp_pot_s3);
            lcd.print("   ON");
            salida3=1;  //estado de la salida. // Caldera encendida

      }
      else if(s_3==1 & temp_pot_s3+1.1<(grados_s3)){  //ciclo de
histeresis
            digitalWrite(A3,LOW);   //Orden de apagado de la salida
            lcd.setCursor(0,2);
            lcd.print("S3: ");
```

```
      lcd.print(grados_s3);lcd.print("/");lcd.print(temp_pot_s3);
            lcd.print("  OFF");
            salida3=0;  //estado de la salida. // Caldera apagada.
      }
      else if(s_3==0){  //si la caldera esta off, que se apague.
            digitalWrite(A3,LOW);   //Orden de apagado de la salida
            lcd.setCursor(0,2);
            lcd.print("S3: ");

      lcd.print(grados_s3);lcd.print("/");lcd.print(temp_pot_s3);
            lcd.print("  OFF");
            salida3=0;  //estado de la salida. // Caldera apagada.
      }
      else{
      //actualizamos las teperaturas actuales.
            lcd.setCursor(0,2);
            lcd.print("S3: ");

      lcd.print(grados_s3);lcd.print("/");lcd.print(temp_pot_s3);
      }


      }

      //Indicamos en la ultima linea de la pantalla que el modo es
automático.
   lcd.setCursor(0,3);
   //lcd.print(Str2);
   lcd.print("  Modo automatico");

  }


}
```

13.1.2. Enviar.ino

```
/*
--------------------------------------------------------------------
-
Función que envía a la web los datos que están definidos como varibles
globales en el programa.

Introducimos en esta función tantas lineas como variables queramos
enviar al
servidor.
Conectamos al servidor y escribimos la direccion de la pagina que
recoge los datos
```

```
seguido del simbolo '?' y los nombres que tienen en la web los datos
que se recogen
seguidos de la variable que se envía.
Al final cerramos la conexión.
-------------------------------------------------------------------
-
*/


  void enviaweb(){

  delay(200);
  delay(1000);

//Serial.println(client.status());
  client.connect(server,80);               //Conectamos al servidor.
// Serial.println(client.status());
  if (client.connected()){                 //Si establecemos la
conexión:
//Variables que enviamos al servidor.
      client.print("GET /datos.php?");     //Página del servidor que
recibe los datos que enviamos.
      delay (100);
      //Serial.println("enviando datos");
      client.print("ardrefsala1=");        //Variable de la web que lee
GET y queremos cambiar.
      client.print(temp_pot_s1);           //Dato de arduino que
enviamos a la web.
      //Serial.println(ardrefsala1);
      client.print("&ardrefsala2=");       //Variable de la web que lee
GET y queremos cambiar.
      client.print(temp_pot_s2);           //Dato de arduino que
enviamos a la web.
      client.print("&ardrefsala3=");       //Variable de la web que lee
GET y queremos cambiar.
      client.print(temp_pot_s3);           //Dato de arduino que
enviamos a la web.
      client.print("&ardsala1=");          //Variable de la web que lee
GET y queremos cambiar.
      client.print(s_1);                     //Dato de arduino que
enviamos a la web.
      client.print("&ardsala2=");          //Variable de la web que lee
GET y queremos cambiar.
      client.print(s_2);                   //Dato de arduino que
enviamos a la web.
      client.print("&ardsala3=");          //Variable de la web que lee
GET y queremos cambiar.
      client.print(s_3);                   //Dato de arduino que
enviamos a la web.
      client.print("&tempsala1=");         //Variable de la web que lee
GET y queremos cambiar.
      client.print(grados_s1);             //Dato de arduino que
enviamos a la web.
```

```
      client.print("&tempsala2=");          //Variable de la web que lee
GET y queremos cambiar.
      client.print(grados_s2);              //Dato de arduino que
enviamos a la web.
      client.print("&tempsala3=");          //Variable de la web que lee
GET y queremos cambiar.
      client.print(grados_s3);
      client.print("&salida1=");            //Variable de la web que lee
GET y queremos cambiar.
      client.print(salida1);
      client.print("&salida2=");            //Variable de la web que lee
GET y queremos cambiar.
      client.print(salida2);
      client.print("&salida3=");            //Variable de la web que lee
GET y queremos cambiar.
      client.print(salida3);
      client.print("&modo=");               //Variable de la web que lee
GET y queremos cambiar.
      client.print(modo);
      client.print("&comprobacion=");       //Variable de la web que lee
GET y queremos cambiar.
      client.print(ismael);                 //Dato de arduino que
enviamos a la web.
      //Líneas finales para finalizar el envío http.
      client.println(" HTTP/1.1");
      client.println("Host: www.ubumicros.hol.es");  //Modificar el
servidor según la direccion de conexión
      client.println("Connection: close");
    client.println();

    delay(300);

      client.stop();  //Finalizamos la conexión.
      delay(150);
      }
  }
```

13.1.3. F_salas_ONOFF.ino

```
/*
**********************************************************************
***
**********************************************************************
***
Función que ejecuta la activación por zonas de las calderas
mostrándolo en la LCD
**********************************************************************
***
```

```
**********************************************************************
***
*/

void F_Salas_ONOFF(int a1){
    lcd.setCursor(0,0);//Nos posicionamos en el inicio de la LCD
      lcd.print("Activacion  calderas");
      lcd.setCursor(0,1);
      lcd.print("SALA ");lcd.print(a1);lcd.print(": ");//pasamos el
parámetro de cada sala, 1 , 2, 3
      //lcd.setCursor(8,1);

      potenciometro_Si_No();//Entramos a la función del potenciómetro
de elección SI/NO
      lcd.setCursor(0,3);
      lcd.print("Pulse OK");//Indicamos que pulse el pulsador
    //Serial.println("SALE FUNCION");
      estado_pulsador = digitalRead(pulsador);//Comprobamos el estado
del pulsador. El programa pasará por aquí varias veces y en el momento
que pulsemos saltará de pantalla.
      if (estado_pulsador == HIGH){
      l++;
      delay(200);

      //Serial.println("PULSADORRRR");
      while(digitalRead(pulsador) == HIGH){}//Evitamos rebotes
      lcd.clear();//limpiamos pantalla
      switch (a1) {//Bucle switch donde vemos los casos de las
distintas salas y sacamos el valor s_1 global
            case 1:
            if(p_1==1){
            digitalWrite(A1,HIGH);//Poner los led que hagan
faltaaaaaaaaaaaaaaaaa.
            s_1=1;
            }
            else{
            s_1=0;
            }

            break;
            case 2:
            if(p_1==1){
            digitalWrite(A2,HIGH);//Poner los led que hagan
faltaaaaaaaaaaaaaaaaa
            s_2=1;
            }
             else{
            s_2=0;
            }

            break;
            case 3:
            if(p_1==1){
```

```
            digitalWrite(A3,HIGH);//Poner los led que hagan
faltaaaaaaaaaaaaaaaaa
            s_3=1;
            }
            else{
            s_3=0;
            }

            break;

      }

    }
    else{
      }
}
```

13.1.4. F_salas_Temp.ino

```
//FUNCIÓN DONDE ELEGIMOS LA TEMPERATURA DE REFERENCIA Y VA MOSTRANDO
LA TEMPERATURA EN LA LCD JUNTO CON LA LEIDA DEL LM35.

void F_Salas_Temp(int a1){
  potenciometro_T();//llamamos a la función potenciómetro para guardar
las temperaturas de referencia
      lcd.setCursor(0,0);
      lcd.print("       T SALA "); lcd.print(a1);lcd.print("      ");
      lcd.setCursor(0,1);
      lcd.print("Temp_actual: ");
    if(l==4){//SALA 1
      if (abs(grados_s1 - grados_s1_ant) > 0.2){// Si la diferencia es
mayor que 0.2 muestra el valor nuevo si no no. Así evitamos problemas
de ruido


      //Serial.println(temp_pot);
      lcd.print(grados_s1);lcd.print("C");
      grados_s1_ant=grados_s1;//Guardamos el estado para ver si cambia
o no en la nueva medición y mostrarlo o no mostrarlo
      }
      }
      if(l==5){//SALA 2
      if (abs(grados_s2 - grados_s2_ant) > 0.2){// Si la diferencia es
mayor que 0.2 muestra el valor nuevo si no no. Así evitamos problemas
de ruido


      //Serial.println(temp_pot);
```

```
      lcd.print(grados_s2);lcd.print("C");
      grados_s2_ant=grados_s2;
      }
      }

      if(l==6){//SALA 3
      if (abs(grados_s3 - grados_s3_ant) > 0.2){// Si la diferencia es
mayor que 0.2 muestra el valor nuevo si no no. Así evitamos problemas
de ruido


      //Serial.println(temp_pot);
      lcd.print(grados_s3);lcd.print("C");
      grados_s3_ant=grados_s3;
      }
      }


      lcd.setCursor(0,2);
      //El valor del potenciómetro es lo mismo para las tres salas.
      lcd.print("Temp_deseada: ");
      if (abs(temp_pot - temp_pot_ant) > 0.2)// Si la diferencia es
mayor que 0.5 muestra el valor nuevo si no no. Así evitamos problemas
de ruido
  {
   lcd.print(temp_pot);lcd.print("C");
      //Serial.println(temp_pot);

      temp_pot_ant=temp_pot;// Actualizamos el valor anterior al
actual
  }


}
```

13.1.5. Potenciometro_SI_NO.ino

```
/*
*********************************************************************
***
*********************************************************************
***
Función de elección mediante potenciómetro de ON/OFF y de
Manual/Automático
*********************************************************************
***
*********************************************************************
***
*/
```

```
boolean potenciometro_Si_No(){
  if(l==1 || l==2 || l==3){  // ON/OFF para cuando estamos en el menu
de la activación de la caldera.
      valor_pot1= analogRead(A0);

      if(valor_pot1<=513){    //Como la señal va de 0 a 1023 ya que
trabajamos con una resolución de 10bits 2^10=1023
      //Dividimos el valor del potenciómetro por la mitad de manera
que si es menor que 513 aparecerá un ON.
      //Serial.println("ON");
      lcd.setCursor(8,1);
      lcd.print("ON ");
      p_1=1;

      }

      else{//Si no es menor que 513, es decir, es mayor aparecerá OFF
      //Serial.println("OFF");
      lcd.setCursor(8,1);
      lcd.print("OFF");
      p_1=0;
    }

  }
  if(l==0){  //Manual Automático. Elección en el primer menu. El
procedimiento es igual que lo anterior.
      valor_pot1= analogRead(A0);
      if(valor_pot1<=513){
      lcd.setCursor(0,2);
      lcd.print("   Manual          ");
      M_A=0;
      }
      else{
      lcd.setCursor(0,2);
      lcd.print("   Automatico      ");
      M_A=1;//CAMBIAR ESTA VARIABLE PARA EL MENU FINAL DONDE PASA AL
AUTOMATICO
      }
  }
}
```

13.1.6. Potenciometro_T.ino

```
/*
*************************************************************************
***
*************************************************************************
***
```

```
Función de elección mediante potenciómetro de Temperatura (10°C/26°C)
**************************************************************************
***
**************************************************************************
***
*/


void potenciometro_T(){
 valor_pot1= analogRead(A0);

  //Desarrollamos la ecuación de la recta que abarca los valores.
Sabemos que la lectura va de 0 a 1023 y debemos ajustar de forma
  //proporcional a las temperaturas que queramos.
 temp_pot=(valor_pot1+639.375)/63.9375;// Ecuación de la recta que
rige la lectura analógica para colocar el rango de temperatura de 10 a
26




//MANERA CUTRE DE AJUSTAR A TU GUSTO. De forma manual
  //Serial.println(valor_pot1valor_pot1valor_pot1valor_pot1);
//  if(valor_pot1<=64){
//    temp_pot=10;
//    Serial.println(temp_pot);
//  }
//    else if (valor_pot1>64 && valor_pot1<=128){
//    temp_pot=11;
//    Serial.println(temp_pot);
//    }
//
//    else if (valor_pot1>128 && valor_pot1<=192){
//    temp_pot=12;
//    Serial.println(temp_pot);
//    }
//
//    else if (valor_pot1>192 && valor_pot1<=256){
//    temp_pot=13;
//    Serial.println(temp_pot);
//    }
//
//    else if (valor_pot1>256 && valor_pot1<=320){
//    temp_pot=14;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>320 && valor_pot1<=384){
//    temp_pot=15;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>384 && valor_pot1<=238){
//    temp_pot=16;
//    Serial.println(temp_pot);
```

```
//    }
//    else if (valor_pot1>238 && valor_pot1<=448){
//    temp_pot=17;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>448 && valor_pot1<=512){
//    temp_pot=18;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>512 && valor_pot1<=576){
//    temp_pot=19;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>576 && valor_pot1<=640){
//    temp_pot=20;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>640 && valor_pot1<=704){
//    temp_pot=21;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>704 && valor_pot1<=768){
//    temp_pot=22;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>764 && valor_pot1<=832){
//    temp_pot=23;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>832 && valor_pot1<=896){
//    temp_pot=24;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>896 && valor_pot1<=960){
//    temp_pot=25;
//    Serial.println(temp_pot);
//    }
//    else if (valor_pot1>960 && valor_pot1<=1024){
//    temp_pot=26;
//    Serial.println(temp_pot);
//    }
//  else{
//
//}
}
```

13.1.7. Recibir.ino

```
/*
```

```
-----------------------------------------------------------------------
-
Función que lee del servidor los archivos a los que apuntan las
direcciones.

Esta función recibe como parametro una direccion de un archivo del
servidor.

Se conecta al servidor. Escribe la direccion de lectura y después de
la
peticion espera los datos.
Los 2 ultimos datos que se reciben son almacenados en un registro
continuamente.
Cuando se han enviado todos los datos ese registro tendra almacenado
el
valor que queremos leer de temperatura. Caovertimos el dato a tipo
byte.

La función devuelve el dato leido del servidor.
-----------------------------------------------------------------------
-
*/


int leeweb(char dir[]){ //Funcion que lee de la web los archivos de
texto.
                        //Se le pasa como parametro la direccion de la
que lee.

  char reg[3];     //Registro de 3 bytes tipo caracter que  será
convertido a dato byte.

  int ret=99;      //Variable que devuelve la función. La
inicializamos.
                   //Si devuelve este valor, existe error de lectura.
  char c;          //caracter char que se leerá cada vez.


  //Serial.println("available1");
  client.stop();               //Cerramos la conexión previamente para
evitar errores con otras conexiones anteriores
  delay(200);                  //tiempo de espera para el servidor.
  client.connect(server,80);   //conexión por el puerto 80.

  if (client.connected()) {    //Verificamos que estamos conectados al
servidor.
      reg[0]=0;                //Inicializamos el registro a cero.
      reg[1]=0;
      reg[2]=0;
      //Serial.println("available2");
//    client.print(dir);
      client.print(dir);       //Conexión a la direccion de la página
remota .txt
```

```
//    client.println();
    client.println(" HTTP/1.1");
    client.println("Host: www.ubumicros.hol.es");
    //client.println("Connection: close");
    client.println();
    delay(5000);              //Esperamos un tiempo para que el
servidor responda a la petición.

      //delay(150);
      while(client.available()>0){        //Mientras haya datos en la
página por leer se ejecuta la sentencia.
      // Serial.println("available4");
      c = client.read();      //Leemos caracteres de la web.
      // Serial.print(c);
      reg[0]=reg[1];          //desplazamos la respuesta del servidor
en el dato registro.
      reg[1]=c;               //el ultimo dato leido siempre estará en
la posición 1.
      reg[2]=0;
      }

  delay(100);
  client.stop();              //cerramos la conexión.
  }

      ret=atoi(reg);          //Asigna a la variable de salida el
registro de caracteres convertido a byte.

      return ret;             //Valor que devuelve la función. El dato
leido del servidor.
  }
```

13.1.8. Recibir2.ino

```
/*
--------------------------------------------------------------------
-
Función que llama a la función que lee del servidor un numero
determinado de veces y finalmete asigna los valores leidos del
servidor a las variables globales del programa.

Se guardan en un array temporal los datos que devuelve la función
leeweb. A la cual se le pasa una direccion diferente según el indice
del bucle.
Si entre un bucle y otro se pulsa el pulsador de modo, se sale
del bucle para que la función termine lo más pronto posible.
--------------------------------------------------------------------
-
*/
```

```
void recibir2(){

for (int z = 0; z < 6; z++)
  {
//esta función recorre todos las direcciones del servidor de las que
se leen datos y lee los datos de cada una de ellas.
//Posibilidad de escribir una indicacion de este estado en el LCD, ya
que ocupa mucho tiempo.
   //lcd.print("recibiendo");
      lecturas[z]=leeweb(dir[z]);
      //Serial.println(lecturas[i]);
      //Si entre 2 lecturas se pulsa el pulsador se sale de la función
antes de que termine.
      estado_pulsador = digitalRead(pulsador);
      if (estado_pulsador == HIGH){
      z=6;        //salimos del for
      l=0;        //Volvemos al menu de selección de modo.

      delay(200);

      //Serial.println("PULSADORRRR");
      //Indicacion de que se ha pulsado el pulsador.
      while(digitalRead(pulsador) == HIGH){}
      lcd.clear();
      lcd.print("REINICIANDO");
      delay(600);
      }
  }

  //Al finalizar la función asignamos a las variables del programa el
array de datos leidos del servidor según corresponda.
temp_pot_s1=lecturas[0];
temp_pot_s2=lecturas[1];
temp_pot_s3=lecturas[2];
s_1=lecturas[3];
s_2=lecturas[4];
s_3=lecturas[5];

}
```

13.1.9. Sensor_temp.ino

```
/*
*************************************************************************
***
*************************************************************************
***
```

```
Función que recoge el dato del sensor con su correspondiente
adquisición.
*************************************************************************
***
*************************************************************************
***
*/

float sensor_temp(){
    grados_s1=0;
    grados_s2=0;
      grados_s3=0;
  for(int q=0 ; q<10; q++){//Realizamos 10 medidas y realizaremos la
media de esas 10 medidas para alcanzar mayor exactitud de medida.

      //grados_s1 = analogRead(pinLM35_s1);  //SALA 1
    grados_s1 = analogRead(pinLM35_s1)+grados_s1;  //SALA 1


    grados_s2 = analogRead(pinLM35_s2)+grados_s2;     //SALA 2


    grados_s3 = analogRead(pinLM35_s3)+grados_s3;     //SALA 3

    delay(20);
      //Serial.println(grados_s1);
      }
grados_s1 = ((5.0 * grados_s1 * 100.0)/10240.0)-2;  //Diez cuentas
//hace la media. Por eso el 10240.
grados_s2 = ((5.0 * grados_s2 * 100.0)/10240.0)-2;  //Diez cuentas
//hace la media. Por eso el 10240.
grados_s3 = ((5.0 * grados_s3 * 100.0)/10240.0)-2;  //Diez cuentas
//hace la media. Por eso el 10240.
}
```

13.1.10. Serial_char.ino

```
int serial_char(){

if (Serial.available()) {
    // Esperamos a que el puerto serie tenga algo

      delay(100);
```

```
      // limpia pantalla
      lcd.clear();

      while (Serial.available() > 0) {//Cuando hay algo comenzamos a
leer

      char u=Serial.read();//leemos ese caracter
      //Serial.print(c);
      Str2[n]=u;//Lo guardamos en el string
      n=n++;//Sumamos uno para recoger todos los caracteres de la
frase
      //n1=n;

      // display each character to the LCD

    }
  Str2[n]='\0';//cuando haya un valor vacío retornamos
      n=0;

  }
  return(n1);//devolvemos n1
  lcd.clear();
}
```

13.2. Servidor.

13.2.1. Default.php

```
<html>
<head>
<title>Login</title>
</head>
<h3>Bienvenido</h3>
<p> Login</p>
<a href="logout.php">Logout</a>
<p> </p>

<?php
//pagina de bienvenida. iniciamos la sesión.
session_start();
```

```
//Si la sesión no coincide o no se ha iniciado mostramos el formulario
de validacion de usuario.
//Este formulario envía los datos a la pagina de login.
if (!isset($_SESSION["inloggning"]) || $_SESSION["inloggning"] !==
true) {
?>
<form action="login2.php" method = "post">
      <label for="username">Username</label>
      <input type="username" id="usename" name="username"><br /><br />
      <label for="password">Password:</label>
      <input type="password" id="password"    name="password"><br
/><br />
      <button type = "submit">Login</button>
</form>
<?php
//Si existe la session previamente, nos muestra el enlace al
formulario junto con el de cerrar sesión.
} if (isset($_SESSION["inloggning"])){

      echo '<a href="formulario.php">Formulario</a>';
      }
?>


</html>
```

13.2.2. Loguin2.php

```
<?php
// Inicio de sesión
session_start();

// Configuramos el usuario y la contraseña
$anvandarID = "alberto";
$losenord = "ismael";

//comprobamos que los valores introducidos en el forumario coinciden
con los valores configurados arriba.
//Si son iguales se crea una sesión y nos redirige al formulario.

if (isset($_POST["username"]) && isset($_POST["password"])) {

      if ($_POST["username"] == $anvandarID && $_POST["password"] ==
$losenord) {
      $_SESSION["inloggning"] = true;

      header("Location: formulario.php");

      }
```

```
        // Si los datos no coinciden, no redirige a la página de inicio.
        else {header("Location: default.php");}
}
?>
```

13.2.3. Datos.php

```
<!--Recivirá las variables que lleguen del arduino.
        Eviará estas variables a archivos de texto.
        -->

        <html>
        <head>
        <title> Arduino Shield </title>
        </head>
        <body>

        <?php

        // Establecer la zona horaria predeterminada a usar. Disponible
desde PHP 5.1
        date_default_timezone_set('Europe/Madrid');
        //Imprimimos después la fecha actual dandole un formato


        $ardrefsala1 = $_GET['ardrefsala1'];        //leemos el dato de
referencia de la sala 1
        $ardrefsala2= $_GET['ardrefsala2'];         //leemos el dato de
referencia de la sala 2
        $ardrefsala3= $_GET['ardrefsala3'];         //leemos el dato de
referencia de la sala 3
        $ardsala1= $_GET['ardsala1'];               //leemos el estado de
la sala 1
        $ardsala2= $_GET['ardsala2'];               //leemos el estado de
la sala 2
        $ardsala3= $_GET['ardsala3'];               //leemos el estado de
la sala 3
        $tempsala1= $_GET['tempsala1'];             //leemos la
temperatura de la sala 1
        $tempsala2= $_GET['tempsala2'];             //leemos la
temperatura de la sala 2
        $tempsala3= $_GET['tempsala3'];             //leemos la
temperatura de la sala 3
        $comprobacion= $_GET['comprobacion'];    //leemos el dato de
comprobacion para que no se envien
```

```
                                                    //datos desde
dispositivos no deseados.
        $tiempo=date(DATE_COOKIE);                  //formato para
COOKIES. Lee la fecha y la hora local.
        $salida1=$_GET['salida1'];                  //leemos si la caldera
esta funcionando para la sala 1
        $salida2=$_GET['salida2'];                  //leemos si la caldera
esta funcionando para la sala 2
        $salida3=$_GET['salida3'];                  //leemos si la caldera
esta funcionando para la sala 3
        $modo=$_GET['modo'];                        //leemos el modo de
funcionamiento del módulo.
                                                    //manual, automático u
otro.


        //Escribe los datos en archivos txt permanentes.
        //Los datos recibidos anteriormente se guardan en archivos de
texto.

        if($comprobacion==6655){
        //Si se cumple la condicion, el dispositivo es el correcto,
procedemos a escribir los datos
              //que leerá y mostrara la página de formulario.

        echo file_put_contents("./textos/ardRefSala1.txt",$ardrefsala1);
        echo file_put_contents("./textos/ardRefSala2.txt",$ardrefsala2);
        echo file_put_contents("./textos/ardRefSala3.txt",$ardrefsala3);
        echo file_put_contents("./textos/ardSala1.txt",$ardsala1);
        echo file_put_contents("./textos/ardSala2.txt",$ardsala2);
        echo file_put_contents("./textos/ardSala3.txt",$ardsala3);
        echo file_put_contents("./textos/tempSala1.txt",$tempsala1);
        echo file_put_contents("./textos/tempSala2.txt",$tempsala2);
        echo file_put_contents("./textos/tempSala3.txt",$tempsala3);
        echo file_put_contents("./textos/ultenv.txt",$tiempo);
        //se guarda la fecha de la ultima

        //escritura de datos.
        echo file_put_contents("./textos/salida1.txt",$salida1);
        echo file_put_contents("./textos/salida2.txt",$salida2);
        echo file_put_contents("./textos/salida3.txt",$salida3);
        echo file_put_contents("./textos/modo.txt",$modo);
        //echo
file_put_contents("./textos/comprobacion.txt",$comprobacion);
        //el código no queda registrado por seguridad.
        }

        header('Location: default.php');
        ?>

        </body>
</html>
```

13.2.4. Formulario.php

```
<!--Formulario para introducir variables y enviarlos a los archivos de
texto para almacenarlos.
      Lee los datos de los archivos de texto y los muestra por
pantalla.-->

<html lang="es">

<head>
      <meta charset="utf-8">
      <title>Variables</title>
</head>

<body>

<?php
session_start();
//Si existe una sesión válida se continúa.
//Si no existe sesión se muestra el enlace a la página de inicio.
if (!isset($_SESSION["inloggning"]) || $_SESSION["inloggning"] !==
true) {
echo 'Acceso restringido <br><a href="default.php">Iniciar
sesión</a>';
header("Location: default.php");
exit;
}
?>

<!--Se especifica el formato del texto y se crean los formularios.-->

<div id="wrapper">
<div id="texto">
      <h1>Variables</h1>
      <p><p>
<a href="logout.php">Logout</a>
<p><p> <!--Se introducen párrafos.-->

<!--Formulario para introducir texto.-->
      <form action=" <?php if(isset($_POST['textAreaField'])){

      file_put_contents("./textos/texto.txt",$_POST['textAreaField']);
                  }?>" method="post">
      <div style="30em;">
            <label for="textAreaField">Texto del display</label>
```

```
            <textarea name="textAreaField" id="textAreaField" value=""
rows="4" cols="20" ></textarea>
            <label for="submitButton"> Envía el texto</label>
            <input type="Submit" name="textareaButton"
id="submitButton" value="envio"/>
      </div>
      </form>

      <!--Formulario para seleccionar la temperatura de la sala 1.
      los datos se escriben en el archivo de texto correspondiente si
se introduce algo-->
      <form action="<?php if(isset($_POST['webrefsala1'])){

      file_put_contents("./textos/webRefSala1.txt",$_POST['webrefsala1
']);
                  }?>" method="post">
      <div style="30em;">
            <label for="webrefsala1">Temperatura Sala1</label>
            <SELECT NAME="webrefsala1" id="webrefsala1" size="1">
                  <OPTION VALUE=10> 10°C
                  <OPTION VALUE=11> 11°C
                  <OPTION VALUE=12> 12°C
                  <OPTION VALUE=13> 13°C
                  <OPTION VALUE=14> 14°C
                  <OPTION VALUE=15> 15°C
                  <OPTION VALUE=16> 16°C
                  <OPTION VALUE=17> 17°C
                  <OPTION VALUE=18> 18°C
                  <OPTION VALUE=19> 19°C
                  <OPTION VALUE=20> 20°C
                  <OPTION VALUE=21> 21°C
                  <OPTION VALUE=22> 22°C
                  <OPTION VALUE=23> 23°C
                  <OPTION VALUE=24> 24°C
                  <OPTION VALUE=25> 25°C
                  <OPTION VALUE=26> 26°C
            </SELECT>
            <label for="submitButton"> Entrar</label>
            <input type="Submit" name="webrefsala1Button"
id="submitButton" value=" Envía"/>
      </div>
      </form>

      <!--Formulario para seleccionar la temperatura de la sala 2.
      los datos se escriben en el archivo de texto correspondiente si
se introduce algo-->
      <form action="<?php if(isset($_POST['webrefsala2'])){

      file_put_contents("./textos/webRefSala2.txt",$_POST['webrefsala2
']);
                  }?>" method="post">
      <div style="30em;">
            <label for="webrefsala2">Temperatura Sala2</label>
```

```
            <SELECT NAME="webrefsala2" id="webrefsala2" size="1">
                  <OPTION VALUE=10> 10°C
                  <OPTION VALUE=11> 11°C
                  <OPTION VALUE=12> 12°C
                  <OPTION VALUE=13> 13°C
                  <OPTION VALUE=14> 14°C
                  <OPTION VALUE=15> 15°C
                  <OPTION VALUE=16> 16°C
                  <OPTION VALUE=17> 17°C
                  <OPTION VALUE=18> 18°C
                  <OPTION VALUE=19> 19°C
                  <OPTION VALUE=20> 20°C
                  <OPTION VALUE=21> 21°C
                  <OPTION VALUE=22> 22°C
                  <OPTION VALUE=23> 23°C
                  <OPTION VALUE=24> 24°C
                  <OPTION VALUE=25> 25°C
                  <OPTION VALUE=26> 26°C
            </SELECT>
            <label for="submitButton"> Entrar</label>
            <input type="Submit" name="webrefsala2Button"
id="submitButton" value=" Envía"/>
      </div>
      </form>


      <!--Formulario para seleccionar la temperatura de la sala 3.
      los datos se escriben en el archivo de texto correspondiente si
se introduce algo-->
      <form action="<?php if(isset($_POST['webrefsala3'])){

      file_put_contents("./textos/webRefSala3.txt",$_POST['webrefsala3
']);
                  }?>" method="post">
      <div style="30em;">
            <label for="webrefsala3">Temperatura Sala3</label>
            <SELECT NAME="webrefsala3" id="webrefsala3" size="1">
                  <OPTION VALUE=10> 10°C
                  <OPTION VALUE=11> 11°C
                  <OPTION VALUE=12> 12°C
                  <OPTION VALUE=13> 13°C
                  <OPTION VALUE=14> 14°C
                  <OPTION VALUE=15> 15°C
                  <OPTION VALUE=16> 16°C
                  <OPTION VALUE=17> 17°C
                  <OPTION VALUE=18> 18°C
                  <OPTION VALUE=19> 19°C
                  <OPTION VALUE=20> 20°C
                  <OPTION VALUE=21> 21°C
                  <OPTION VALUE=22> 22°C
                  <OPTION VALUE=23> 23°C
                  <OPTION VALUE=24> 24°C
                  <OPTION VALUE=25> 25°C
```

```
                <OPTION VALUE=26> 26°C
            </SELECT>
            <label for="submitButton"> Entrar</label>
            <input type="Submit" name="webrefsala3Button"
id="submitButton" value=" Envía"/>
      </div>
      </form>


      <!--Formulario para seleccionar el estado de la sala 1.
      los datos se escriben en el archivo de texto correspondiente si
se introduce algo-->
      <form action="<?php if(isset($_POST['radioButtonField1'])){

      file_put_contents("./textos/webSala1.txt",$_POST['radioButtonFie
ld1']);
                  }?>" method="post">
      <div style="30em;">
            <p>Sala1</p>
            <label for="radioButtonField1">on</label>
            <input type="radio" name="radioButtonField1"
id="radioButtonField1" value="1" />
            <label for="radioButtonField2">off</label>
            <input type="radio" name="radioButtonField1"
id="radioButtonField2" value="0" />
            <label for="submitButton"> caldera</label>
            <input type="Submit" name="Sala1Button" id="submitButton"
value=" Envía"/>
      </div>
      </form>


      <!--Formulario para seleccionar el estado de la sala 2.
      los datos se escriben en el archivo de texto correspondiente si
se introduce algo-->
      <form action="<?php if(isset($_POST['radioButtonField2'])){

      file_put_contents("./textos/webSala2.txt",$_POST['radioButtonFie
ld2']);
                  }?>" method="post">
      <div style="30em;">
            <p>Sala2</p>
            <label for="radioButtonField1">on</label>
            <input type="radio" name="radioButtonField2"
id="radioButtonField1" value="1" />
            <label for="radioButtonField2">off</label>
            <input type="radio" name="radioButtonField2"
id="radioButtonField2" value="0" />
            <label for="submitButton"> caldera</label>
            <input type="Submit" name="Sala2Button" id="submitButton"
value=" Envía"/>
      </div>
      </form>
```

```
	<!--Formulario para seleccionar el estado de la sala 3.
	los datos se escriben en el archivo de texto correspondiente si
se introduce algo-->
	<form action="<?php if(isset($_POST['radioButtonField3'])){

	file_put_contents("./textos/webSala3.txt",$_POST['radioButtonFie
ld3']);
		}?>" method="post">
	<div style="30em;">
		<p>Sala3</p>
		<label for="radioButtonField1">on</label>
		<input type="radio" name="radioButtonField3"
id="radioButtonField1" value="1" />
		<label for="radioButtonField2">off</label>
		<input type="radio" name="radioButtonField3"
id="radioButtonField2" value="0" />
		<label for="submitButton"> caldera</label>
		<input type="Submit" name="Sala3Button" id="submitButton"
value=" Envía"/>
	</div>
	</form>

</div>
</div>


<?php

// $webrefsala1= 'webRefSala1.txt';

///Estado de las temperaturas de referencia introducidas en la web.
//Se muestran las temperaturas en una línea de texto.
echo "Temperatura de referencia Sala1: ";
echo file_get_contents("./textos/webRefSala1.txt");
echo " °C";
echo '<br>';

echo "Temperatura de referencia Sala2: ";
echo file_get_contents("./textos/webRefSala2.txt");
echo " °C";
echo '<br>';

echo "Temperatura de referencia Sala3: ";
echo file_get_contents("./textos/webRefSala3.txt");
echo " °C";
echo '<br>';

echo'<p>';

///Estado de la temperatura real.
//Se muestran las temperaturas en una línea de texto.
```

```
echo "TEMPERATURAS: ";
echo '<br>';

echo "Temperatura de Sala1: ";
echo file_get_contents("./textos/tempSala1.txt");
echo " °C";
echo '<br>';

echo "Temperatura de Sala2: ";
echo file_get_contents("./textos/tempSala2.txt");
echo " °C";
echo '<br>';

echo "Temperatura de Sala3: ";
echo file_get_contents("./textos/tempSala3.txt");
echo " °C";
echo '<br>';

echo'<p>';

///Estado de la calefaccion.
//Se muestra el estado en una línea de texto.
//"caldera funcionando" si el txt contiene un 1
//"Caldera apagada" si tiene otro dato
echo "Caldera Sala1: ";
if(file_get_contents("./textos/webSala1.txt")==1){
      echo "Caldera funcionando";
      }else{
      echo "Caldera apagada";
      }
echo '<br>';

echo "Caldera Sala2: ";
if(file_get_contents("./textos/webSala2.txt")==1){
      echo "Caldera funcionando";
      }else{
      echo "Caldera apagada";
      }
echo '<br>';

echo "Caldera Sala3: ";
if(file_get_contents("./textos/webSala3.txt")==1){
      echo "Caldera funcionando";
      }else{
      echo "Caldera apagada";
      }
echo '<br>';
echo '<br>';


//Muestra la fecha de la última vez que se enviaron datos desde
arduino.
//Última conexión.
```

```
echo "Última conexión: ";
echo file_get_contents("./textos/ultenv.txt");
echo " ";
echo '<br>';
echo '<br>';


///Estado de las temperaturas de referencia que tiene arduino en su
ultima conexión.
//Se muestran las temperaturas en una línea de texto.
echo "REFERENCIAS EN ARDUINO: ";
echo '<br>';

echo "Temperatura de referencia Sala1: ";
echo file_get_contents("./textos/ardRefSala1.txt");
echo " °C";
echo '<br>';

echo "Temperatura de referencia Sala2: ";
echo file_get_contents("./textos/ardRefSala2.txt");
echo " °C";
echo '<br>';

echo "Temperatura de referencia Sala3: ";
echo file_get_contents("./textos/ardRefSala3.txt");
echo " °C";
echo '<br>';
echo '<br>';


///Estado de las salas que tiene arduino en su ultima conexión.
//"caldera funcionando" si el txt contiene un 1
//"Caldera apagada" si tiene otro dato
echo "CALDERAS EN ARDUINO: ";
echo '<br>';

echo "Caldera Sala1: ";
if(file_get_contents("./textos/ardSala1.txt")==1){
      echo "Caldera funcionando";
      }else{
      echo "Caldera apagada";
      }
echo '<br>';

echo "Caldera Sala2: ";
if(file_get_contents("./textos/ardSala2.txt")==1){
      echo "Caldera funcionando";
      }else{
      echo "Caldera apagada";
      }
echo '<br>';

echo "Caldera Sala3: ";
```

```
if(file_get_contents("./textos/ardSala3.txt")==1){
      echo "Caldera funcionando";
      }else{
      echo "Caldera apagada";
      }
echo '<br>';

echo'<p>';

//muestra el modo de funcionamiento de arduino en su ultima conexión.
//automático si el estado es 1
//manual si es distinto de 1
echo "ESTADO ACTUAL: ";
echo '<br>';

if(file_get_contents("./textos/modo.txt")==1){
      echo "Funcionando en modo automático";
      }else{
      echo "Funcionando en modo manual";
      }
echo '<br>';

//muestra funcionamiento de las salas de arduino en su ultima
conexión.
//"Calentando la estancia" si el estado es 1    color rojo del texto
//"En reposo" si es distinto de 1   color azul del texto
if(file_get_contents("./textos/salida1.txt")==1){
      echo "<font color='red'> Calentando la estancia</font>";
      }else{
      echo "<font color= 'blue'> En reposo</font>";
      }
echo '<br>';

if(file_get_contents("./textos/salida2.txt")==1){
      echo "<font color='red'> Calentando la estancia</font>";
      }else{
      echo "<font color= 'blue'> En reposo</font>";
      }
echo '<br>';

if(file_get_contents("./textos/salida3.txt")==1){
      echo "<font color='red'> Calentando la estancia</font>";
      }else{
      echo "<font color= 'blue'> En reposo</font>";
      }
echo '<br>';


echo'<p>';

?>
```

```
</body>

</html>
```

13.2.5. Logout.php

```
<?
//inicio de sesion
session_start();
//cerramos la sesion
$_SESSION['inloggning']=false;
unset($_SESSION['inloggning']);
session_destroy();
session_unset();

//redirigimos a la pagina de inicio
header("location:default.php");

?>
```

14. Bibliografía.

http://www.arduino.cc/

http://arduino.cc/en/Main/ArduinoGSMShield

http://arduino.cc/es/Tutorial/LiquidCrystal

Arduino Curso Práctico de Formación. Oscar Torrente Artero.

http://www.ti.com/lit/ds/symlink/lm35.pdf

Ejercicios de Arduino resueltos.

Introducción a Arduino.

blog.bricogeek.com.

Arduino: La tecnología al alcance de todos.

Getting started with Arduino.

www.arduteka.com/arduino/

http://forum.arduino.cc/index.php/board,49.0.html (para alguna duda acerca del GSM).

http://tutorialfritzing.blogspot.com.es/

Php-cookbook

15. ANEXO

15.1. LM35 Datasheet

15.2. Esquema Conexionado

LM35 Precision Centigrade Temperature Sensors

FEATURES

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

DESCRIPTION

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

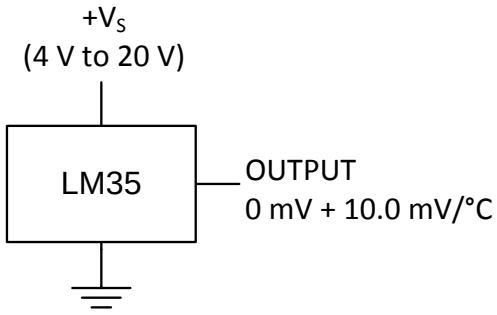
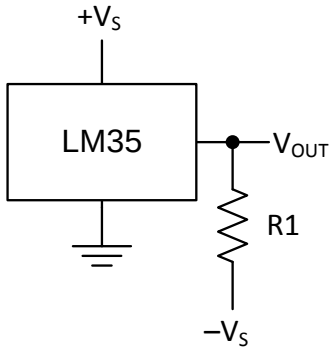


Figure 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



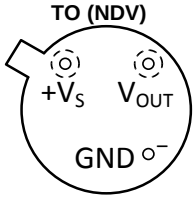
Choose $R_1 = -V_S / 50\ \mu A$
$V_{OUT} = 1500$ mV at 150°C
$V_{OUT} = 250$ mV at 25°C
$V_{OUT} = -550$ mV at −55°C

Figure 2. Full-Range Centigrade Temperature Sensor

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

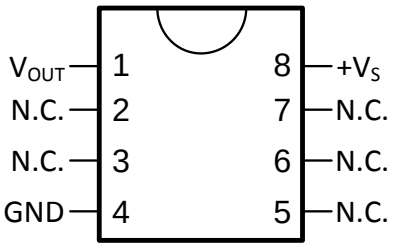
CONNECTION DIAGRAMS

METAL CAN PACKAGE
TO (NDV)



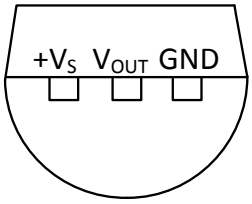
Case is connected to negative pin (GND)

SMALL-OUTLINE MOLDED PACKAGE
SOIC-8 (D)
TOP VIEW

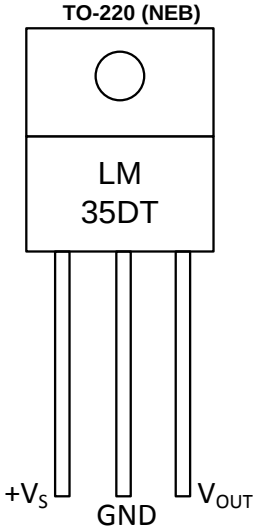


N.C. = No connection

PLASTIC PACKAGE
TO-92 (LP)
BOTTOM VIEW



PLASTIC PACKAGE
TO-220 (NEB)



Tab is connected to the negative pin (GND).

NOTE: The LM35DT pinout is different than the discontinued LM35DP

ABSOLUTE MAXIMUM RATINGS(1)(2)

			MIN	MAX	UNIT
Supply voltage			–0.2	35	V
Output voltage			–1	6	V
Output current				10	mA
Electrostatic discharge (ESD) susceptibility(3)				2500	V
Storage temperature	TO Package		–60	180	°C
	TO-92 Package		–60	150	
	TO-220 Package		–65	150	
	SOIC-8 Package		–65	150	
Lead temperature	TO Package (soldering, 10 seconds)			300	°C
	TO-92 and TO-220 Package (soldering, 10 seconds)			260	
	SOIC Package	Infrared (15 seconds)		220	
		Vapor phase (60 seconds)		215	
Specified operating temperature range: T_{MIN} to T_{MAX}(4)	LM35, LM35A		–55	150	°C
	LM35C, LM35CA		–40	110	
	LM35D		0	100	

(1) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(2) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.
(3) Human body model, 100 pF discharged through a 1.5-kΩ resistor.
(4) Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the APPLICATIONS section.

ELECTRICAL CHARACTERISTICS(1)(2)

PARAMETER	TEST CONDITIONS	LM35A			LM35CA			UNITS (MAX.)
		TYP	TESTED LIMIT(3)	DESIGN LIMIT(4)	TYP	TESTED LIMIT(3)	DESIGN LIMIT(4)	
Accuracy(5)	T_A = 25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A = –10°C	±0.3			±0.3		±1	
	T_A = T_{MAX}	±0.4	±1		±0.4	±1		
	T_A = T_{MIN}	±0.4	±1		±0.4		±1.5	
Nonlinearity(6)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Sensor gain (average slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10**	**+9.9, +10.1**		**+10**		**+9.9, +10.1**	mV/°C
Load regulation(7) $0 \le I_L \le 1$ mA	T_A = 25°C	±0.4	±1		±0.4	±1		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$	**±0.5**		**±3**	**±0.5**		**±3**	
Line regulation(7)	T_A = 25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4 V ≤ V_S ≤ 30 V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	

(1) Unless otherwise noted, these specifications apply: −55°C ≤ T_J ≤ 150°C for the LM35 and LM35A; −40°C ≤ T_J ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ T_J ≤ 100°C for the LM35D. V_S = 5 Vdc and I_{LOAD} = 50 μA, in the circuit of Figure 2. These specifications also apply from +2°C to T_{MAX} in the circuit of Figure 1. Specifications in boldface apply over the full rated temperature range.
(2) Specifications in boldface apply over the full rated temperature range.
(3) Tested Limits are ensured and 100% tested in production.
(4) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(5) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(6) Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(7) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

ELECTRICAL CHARACTERISTICS[1][2] (continued)

PARAMETER	TEST CONDITIONS	LM35A			LM35CA			UNITS (MAX.)
		TYP	TESTED LIMIT[3]	DESIGN LIMIT[4]	TYP	TESTED LIMIT[3]	DESIGN LIMIT[4]	
Quiescent current[8]	V_S = 5 V, 25°C	56	67		56	67		µA
	V_S = 5 V	**105**		**131**	**91**		**114**	
	V_S = 30 V, 25°C	56.2	68		56.2	68		
	V_S = 30 V	**105.5**		**133**	**91.5**		**116**	
Change of quiescent current[7]	4 V ≤ V_S ≤ 30 V, 25°C	0.2	1		0.2	1		µA
	4 V ≤ V_S ≤ 30 V	**0.5**		**2**	**0.5**		**2**	
Temperature coefficient of quiescent current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum temperature for rate accuracy	In circuit of Figure 1, I_L = 0	+1.5		+2	+1.5		+2	°C
Long term stability	T_J = T_{MAX}, for 1000 hours	±0.08			±0.08			°C

(8) Quiescent current is defined in the circuit of Figure 1.

ELECTRICAL CHARACTERISTICS[1][2]

PARAMETER	TEST CONDITIONS	LM35			LM35C, LM35D			UNITS (MAX.)
		TYP	TESTED LIMIT[3]	DESIGN LIMIT[4]	TYP	TESTED LIMIT[3]	DESIGN LIMIT[4]	
Accuracy, LM35, LM35C[5]	T_A = 25°C	±0.4	±1		±0.4	±1		°C
	T_A = –10°C	±0.5			±0.5		±1.5	
	T_A = T_{MAX}	±0.8	±1.5		±0.8		±1.5	
	T_A = T_{MIN}	±0.8		±1.5	±0.8		±2	
Accuracy, LM35D[5]	T_A = 25°C				±0.6	±1.5		°C
	T_A = T_{MAX}				±0.9		±2	
	T_A = T_{MIN}				±0.9		±2	
Nonlinearity[6]	T_{MIN} ≤ T_A ≤ T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Sensor gain (average slope)	T_{MIN} ≤ T_A ≤ T_{MAX}	**+10**	**+9.8, +10.2**		**+10**		**+9.8, +10.2**	mV/°C
Load regulation[7] 0 ≤ I_L ≤ 1 mA	T_A = 25°C	±0.4	±2		±0.4	±2		mV/mA
	T_{MIN} ≤ T_A ≤ T_{MAX}	**±0.5**		**±5**	**±0.5**		**±5**	
Line regulation[7]	T_A = 25°C	±0.01	±0.1		±0.01	±0.1		mV/V
	4 V ≤ V_S ≤ 30 V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	

(1) Unless otherwise noted, these specifications apply: −55°C ≤ T_J ≤ 150°C for the LM35 and LM35A; −40°C ≤ T_J ≤ 110°C for the LM35C and LM35CA; and 0°C ≤ T_J ≤ 100°C for the LM35D. V_S = 5 Vdc and I_{LOAD} = 50 µA, in the circuit of Figure 2. These specifications also apply from +2°C to T_{MAX} in the circuit of Figure 1. Specifications in boldface apply over the full rated temperature range.
(2) Specifications in boldface apply over the full rated temperature range.
(3) Tested Limits are ensured and 100% tested in production.
(4) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(5) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(6) Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(7) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

ELECTRICAL CHARACTERISTICS[1][2] (continued)

PARAMETER	TEST CONDITIONS	LM35			LM35C, LM35D			UNITS (MAX.)
		TYP	TESTED LIMIT[3]	DESIGN LIMIT[4]	TYP	TESTED LIMIT[3]	DESIGN LIMIT[4]	
Quiescent current[8]	V_S = 5 V, 25°C	56	80		56	80		µA
	V_S = 5 V	**105**		**158**	**91**		**138**	
	V_S = 30 V, 25°C	56.2	82		56.2	82		
	V_S = 30 V	**105.5**		**161**	**91.5**		**141**	
Change of quiescent current[9]	4 V ≤ V_S ≤ 30 V, 25°C	0.2	2		0.2	2		µA
	4 V ≤ V_S ≤ 30 V	**0.5**		**3**	**0.5**		**3**	
Temperature coefficient of quiescent current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Minimum temperature for rate accuracy	In circuit of Figure 1, I_L = 0	+1.5		+2	+1.5		+2	°C
Long term stability	T_J = T_{MAX}, for 1000 hours	±0.08			±0.08			°C

(8) Quiescent current is defined in the circuit of Figure 1.
(9) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

TYPICAL PERFORMANCE CHARACTERISTICS

THERMAL RESISTANCE JUNCTION TO AIR

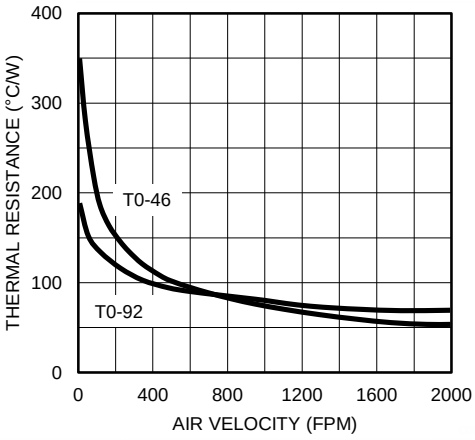


Figure 3.

THERMAL TIME CONSTANT

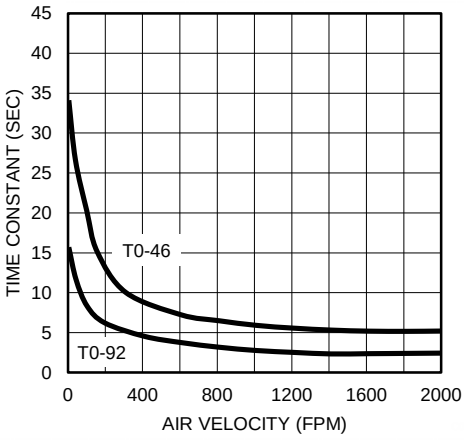


Figure 4.

THERMAL RESPONSE IN STILL AIR

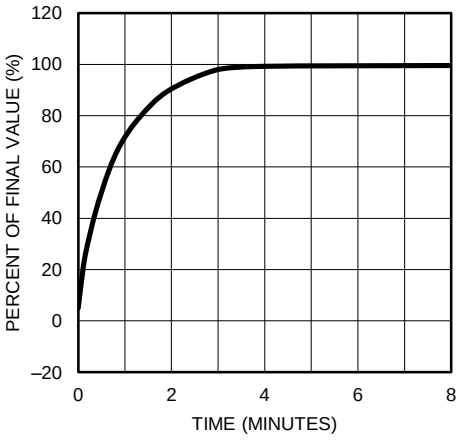


Figure 5.

THERMAL RESPONSE IN STIRRED OIL BATH

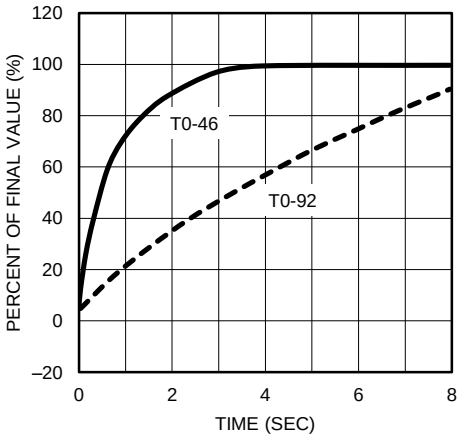


Figure 6.

MINIMUM SUPPLY VOLTAGE VS TEMPERATURE

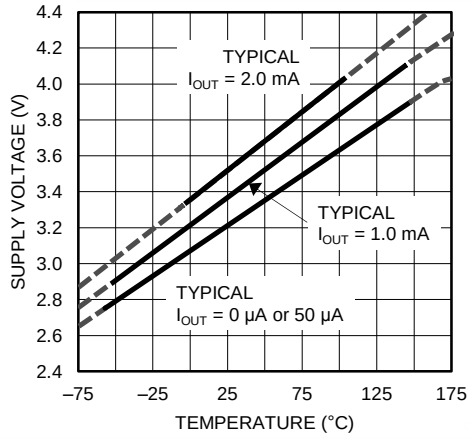


Figure 7.

QUIESCENT CURRENT VS TEMPERATURE (IN CIRCUIT OF Figure 1)

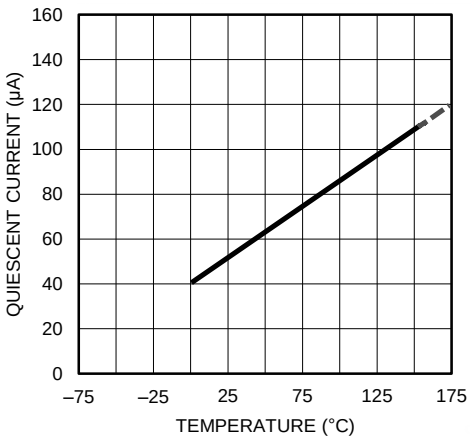


Figure 8.

TYPICAL PERFORMANCE CHARACTERISTICS (continued)

QUIESCENT CURRENT vs TEMPERATURE (IN CIRCUIT OF Figure 2)

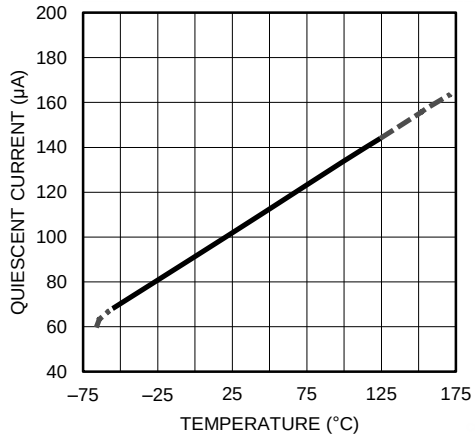


Figure 9.

ACCURACY vs TEMPERATURE (ENSURED)

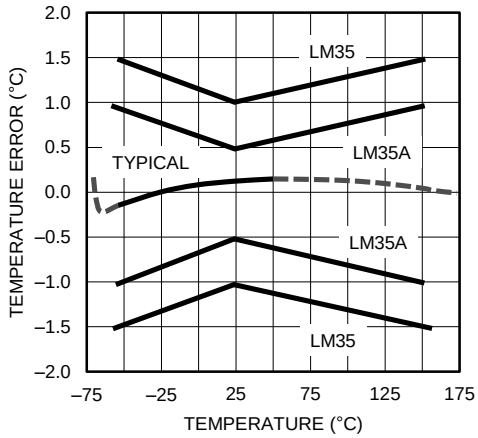


Figure 10.

ACCURACY vs TEMPERATURE (ENSURED)

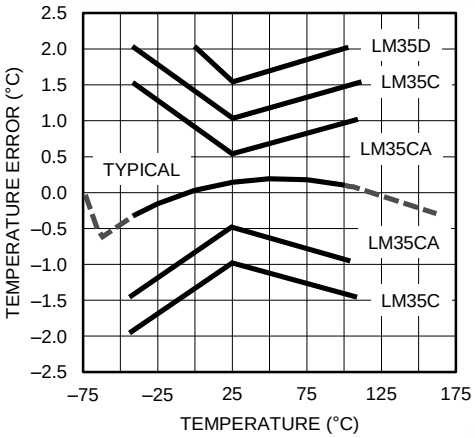


Figure 11.

NOISE VOLTAGE

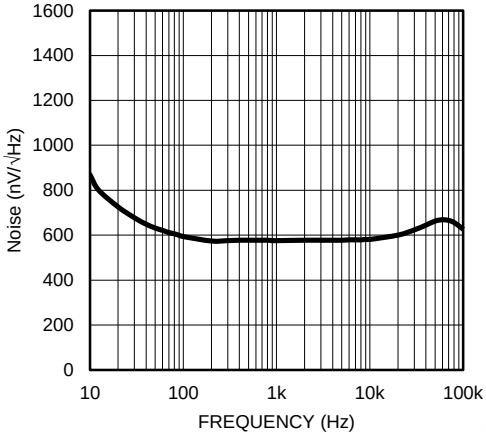


Figure 12.

START-UP RESPONSE

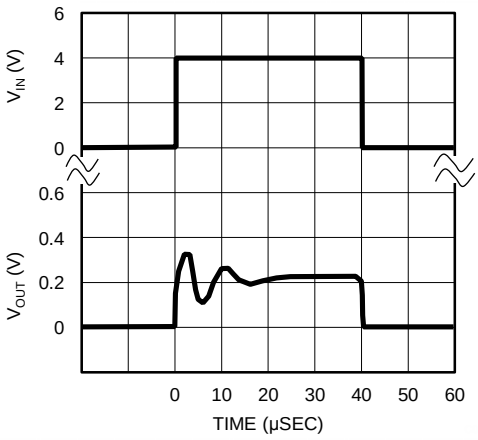


Figure 13.

APPLICATIONS

The LM35 is applied easily in the same way as other integrated-circuit temperature sensors. Glue or cement the device to a surface and the temperature should be within about 0.01°C of the surface temperature.

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

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

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

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

Table 1. Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO, no heat sink	TO(1), small heat fin	TO-92, no heat sink	TO-92(2), small heat fin	SOIC-8, no heat sink	SOIC-8(2), small heat fin	TO-220, no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, Infinite heat sink)	(24°C/W)				(55°C/W)		

(1) Wakefield type 201, or 1-in disc of 0.02-in sheet brass, soldered to case, or similar.
(2) TO-92 and SOIC-8 packages glued and leads soldered to 1-in square of 1/16-in printed circuit board with 2-oz foil or similar.

TYPICAL APPLICATIONS

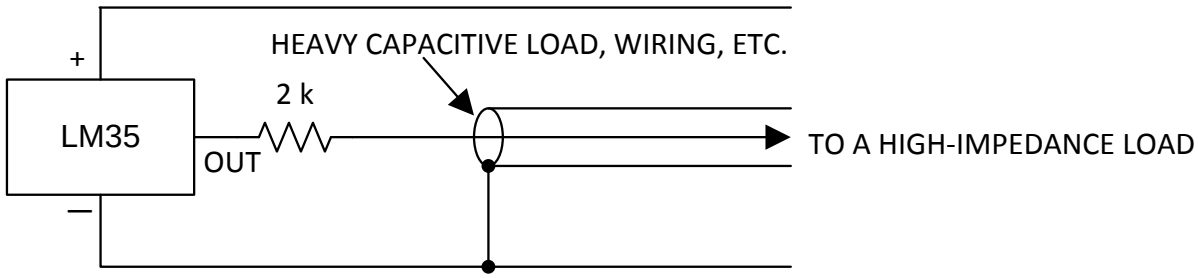


Figure 14. LM35 with Decoupling from Capacitive Load

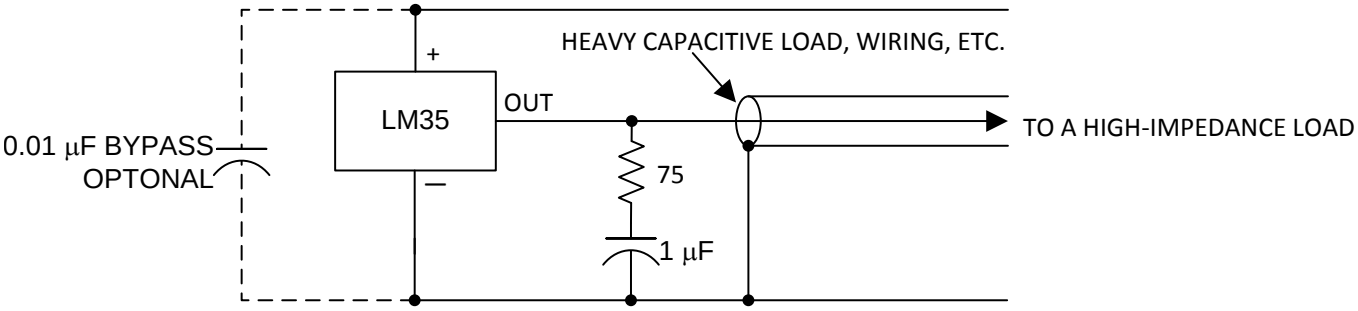


Figure 15. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 alone is able to drive 50 pf without special precautions. If heavier loads are anticipated, isolating or decoupling the load with a resistor is easy (see Figure 14). Or you can improve the tolerance of capacitance with a series R-C damper from output to ground (see Figure 15).

When the LM35 is applied with a 200-Ω load resistor as shown in Figure 16, Figure 17, or Figure 19, the device is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input and not on the output. However, as with any linear circuit connected to wires in a hostile environment, performance is affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, and SCR transients, as the wiring acts as a receiving antenna and the internal junctions act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper, such as 75 Ω, in series with 0.2 or 1 µF from output to ground are often useful. These are shown in Figure 24, Figure 24, and Figure 27.

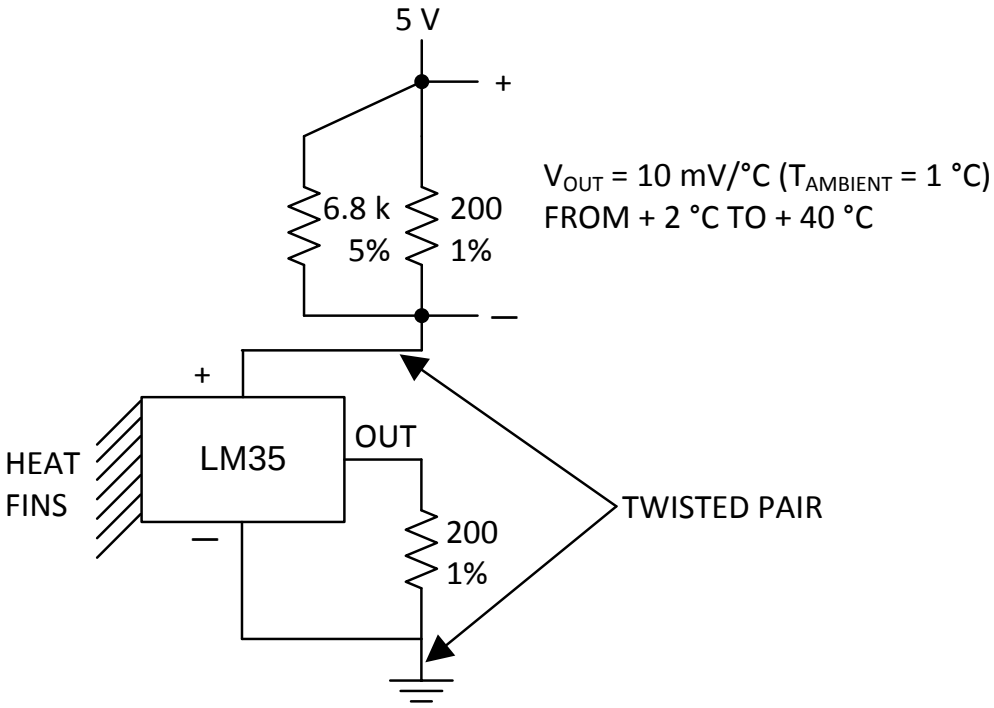


Figure 16. Two-Wire Remote Temperature Sensor (Grounded Sensor)

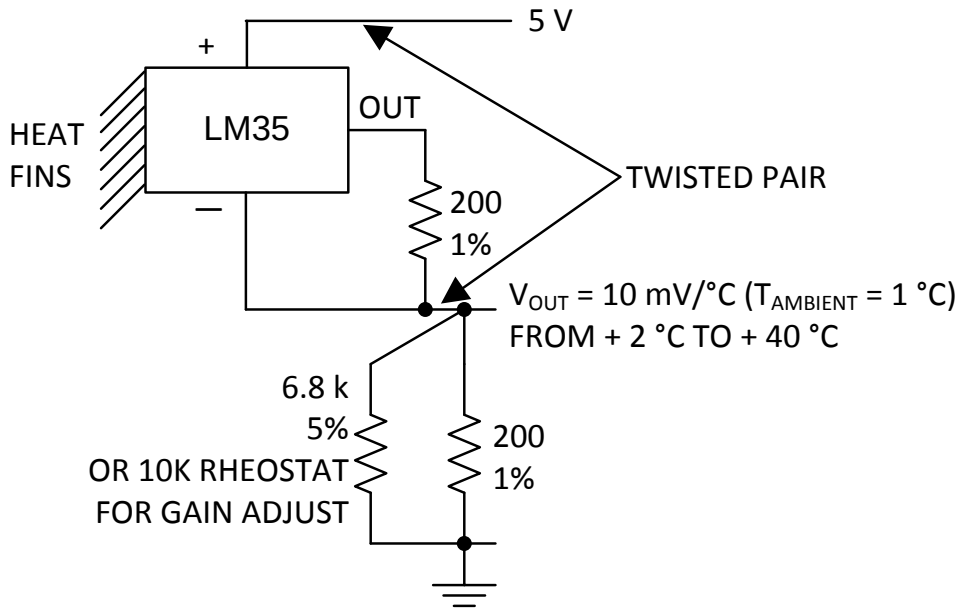


Figure 17. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

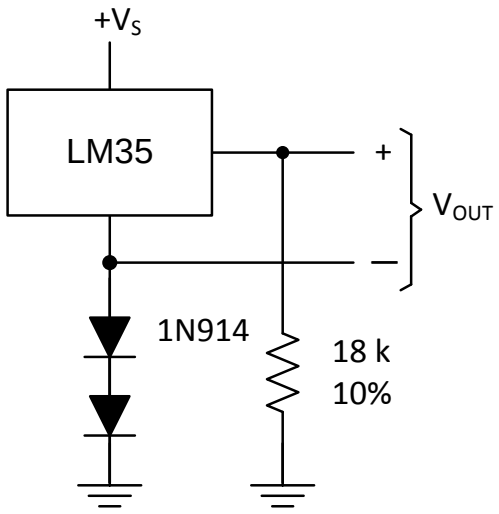


Figure 18. Temperature Sensor, Single Supply (−55° to +150°C)

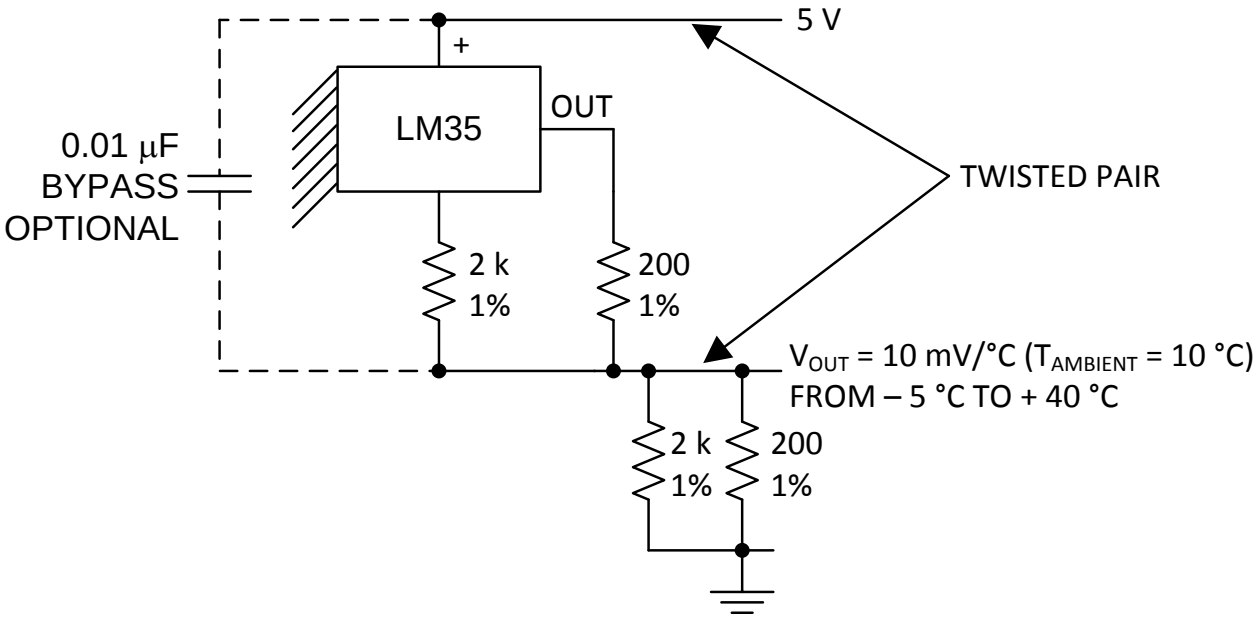


Figure 19. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

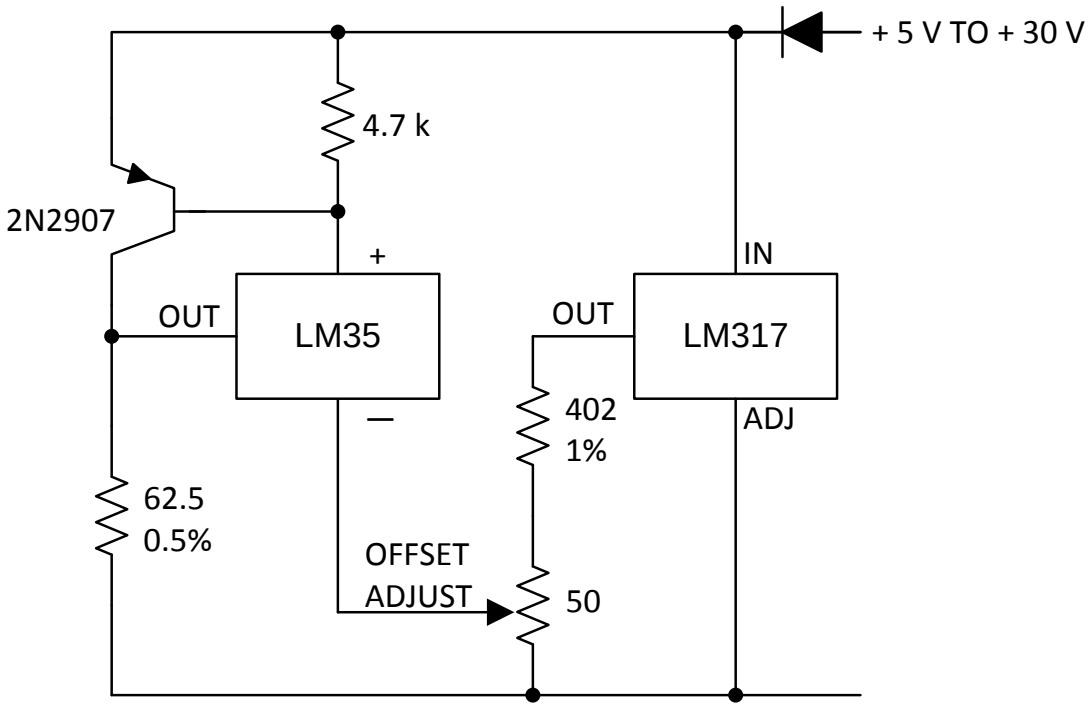


Figure 20. 4-To-20 mA Current Source (0°C to 100°C)

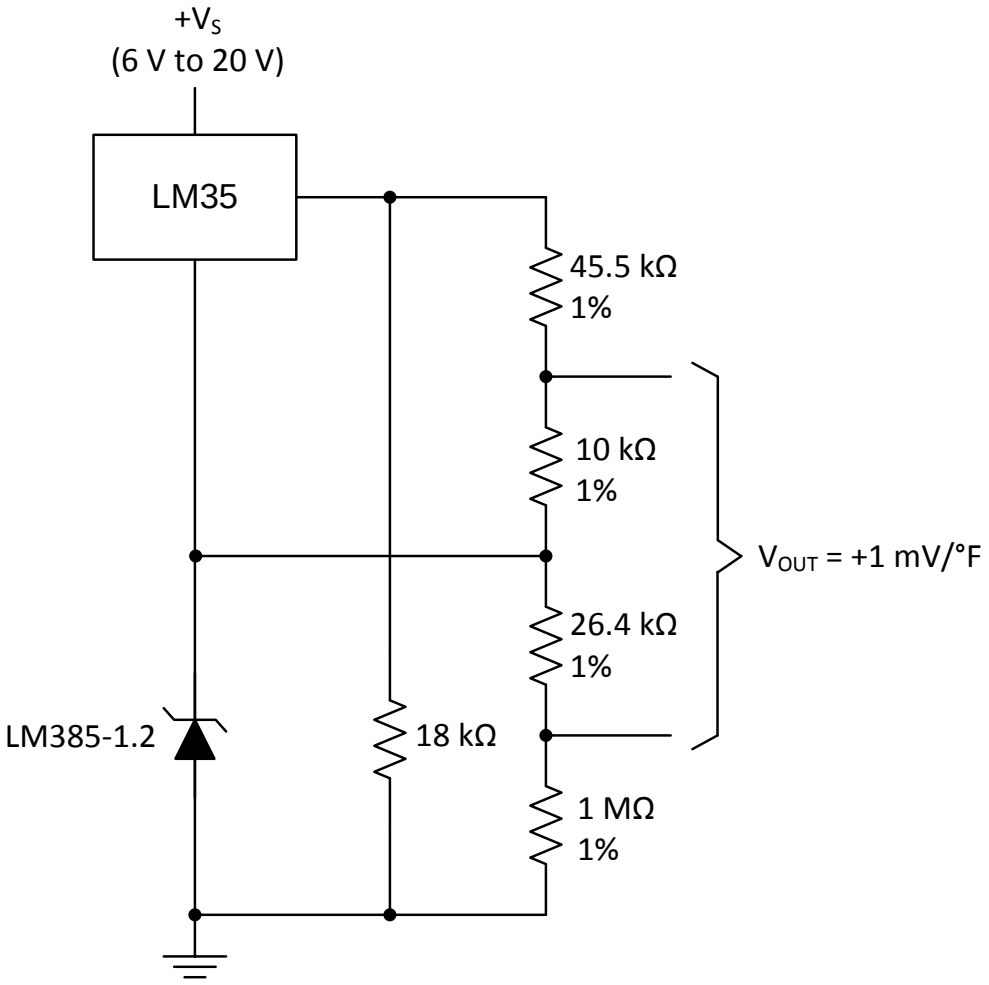


Figure 21. Fahrenheit Thermometer

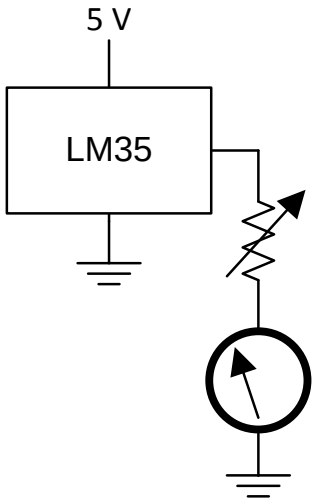


Figure 22. Centigrade Thermometer (Analog Meter)

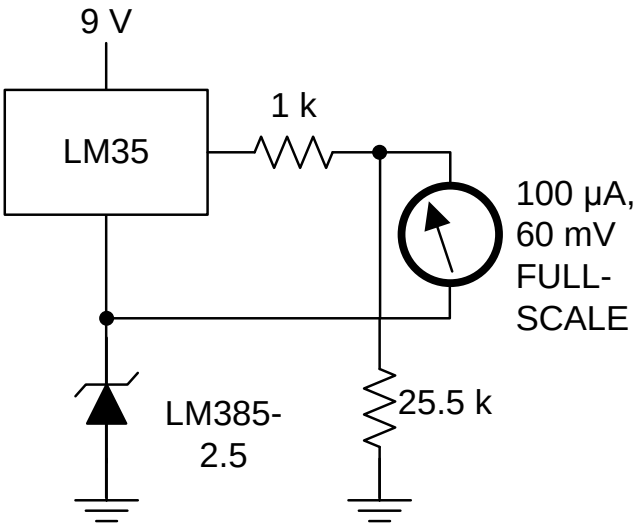


Figure 23. Fahrenheit Thermometer, Expanded Scale Thermometer (50°F to 80°F, for Example Shown)

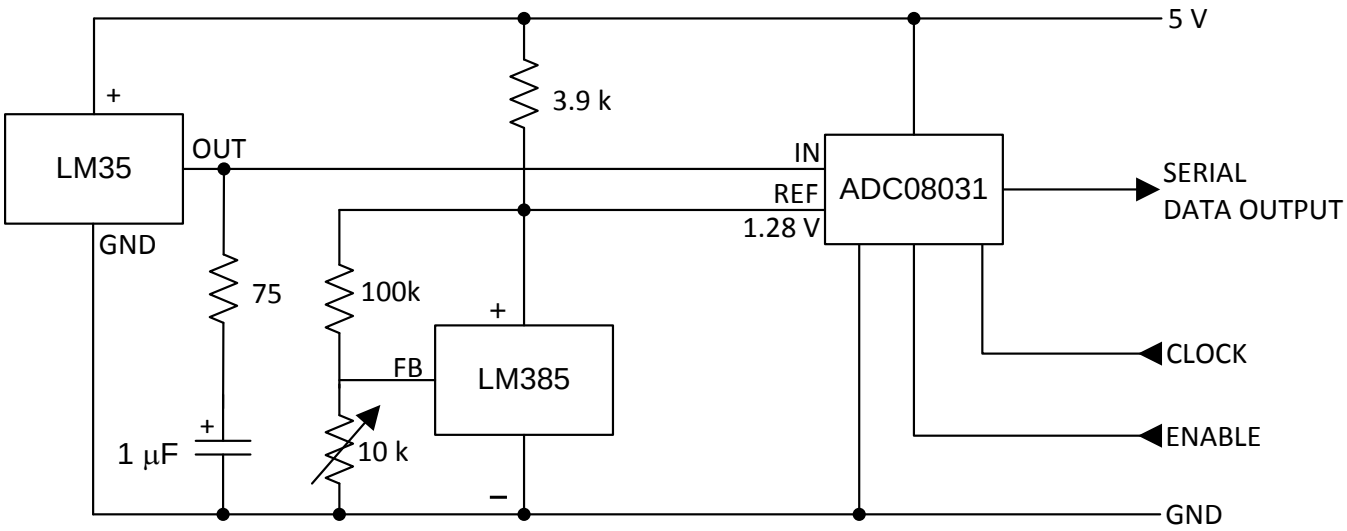


Figure 24. Temperature To Digital Converter (Serial Output) (128°C Full Scale)

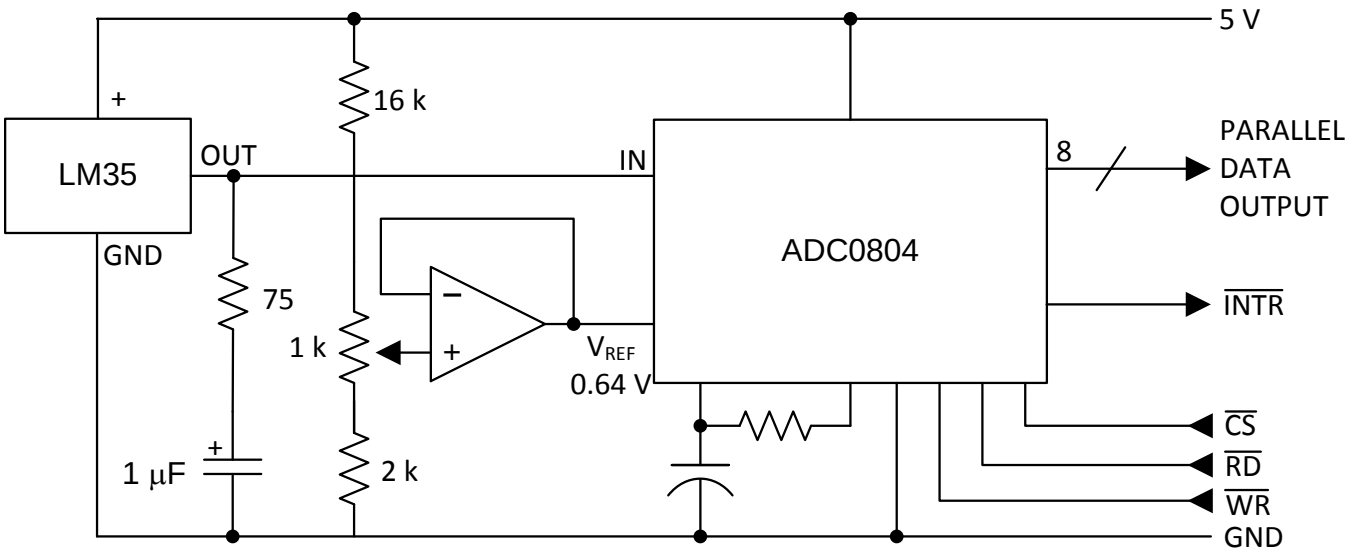
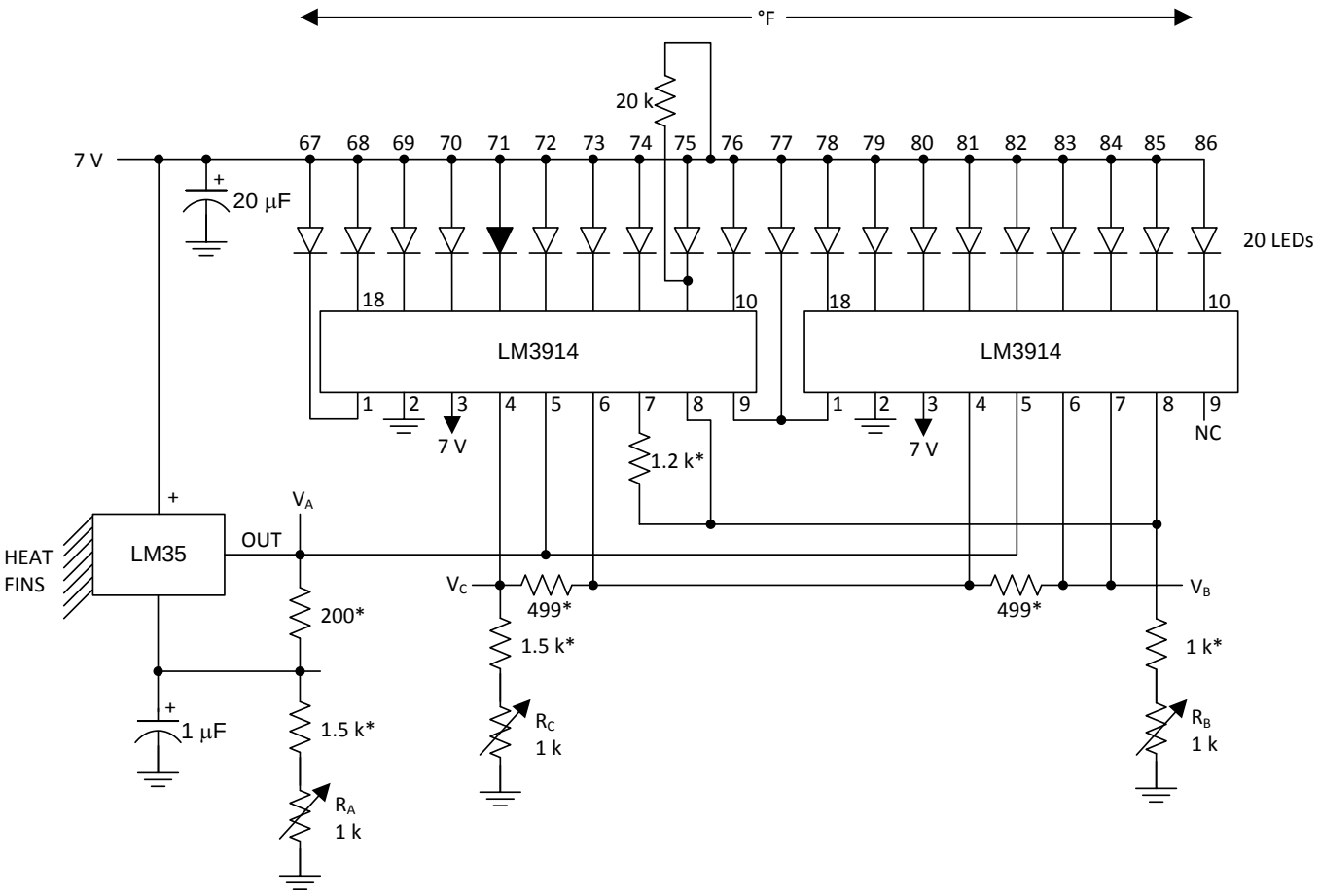


Figure 25. Temperature To Digital Converter
(Parallel TRI-STATE Outputs for Standard Data Bus to µP Interface.)
(128°C Full Scale)



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

Figure 26. Bar-Graph Temperature Display (Dot Mode)

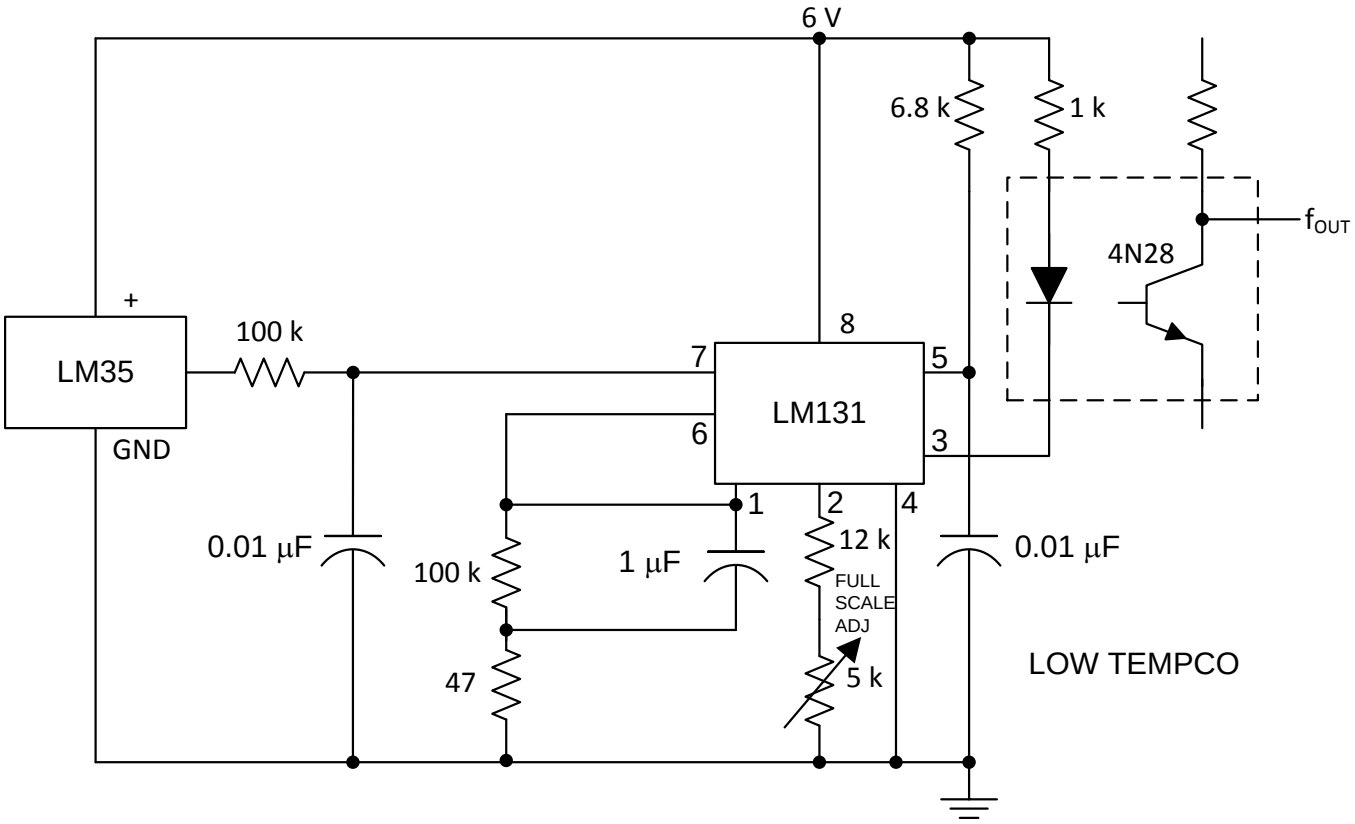
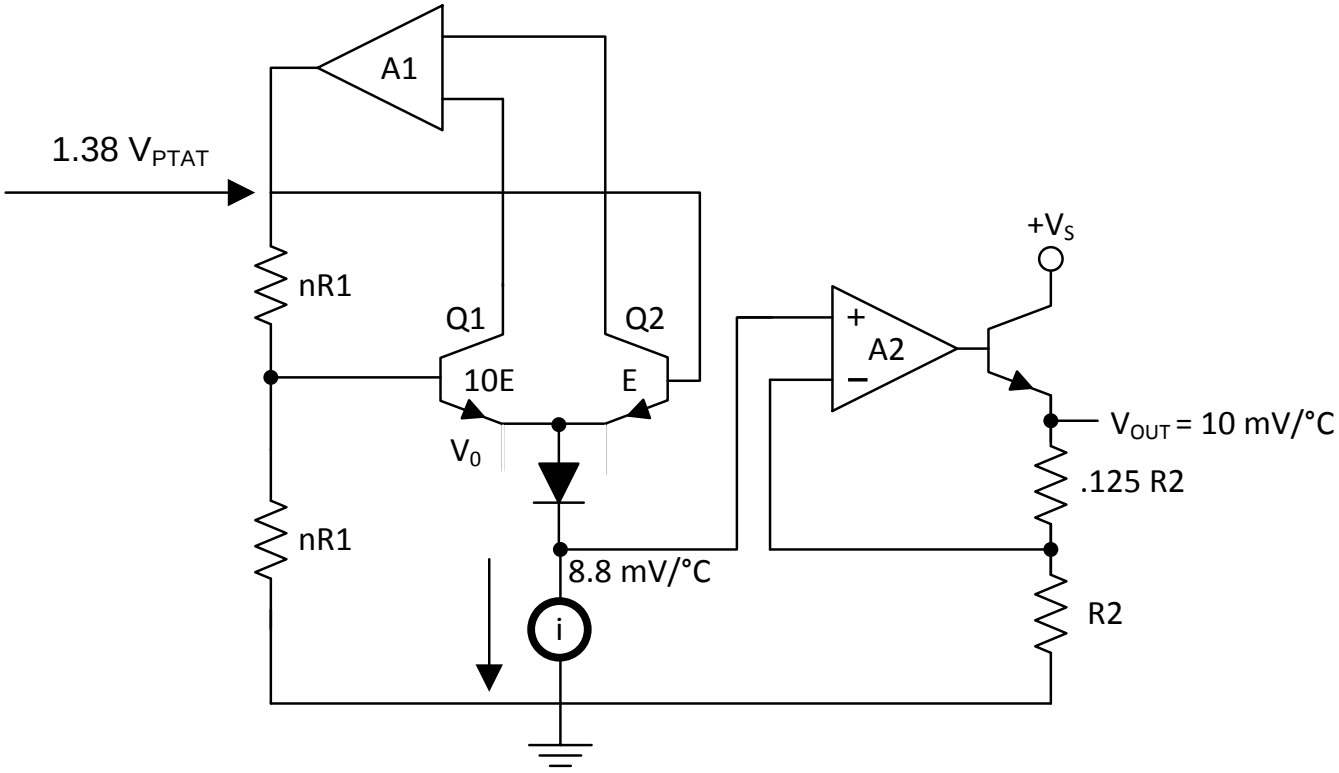


Figure 27. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to 150°C; 20 to 1500 Hz)

BLOCK DIAGRAM



REVISION HISTORY

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

- Changed *W* to Ω 1
- Changed *W* to Ω 3
- Changed *W* to Ω 9

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
LM35AH	ACTIVE	TO	NDV	3	1000	TBD	Call TI	Call TI	-55 to 150	LM35AH	Samples
LM35AH/NOPB	ACTIVE	TO	NDV	3	1000	Green (RoHS & no Sb/Br)	Call TI | POST-PLATE	Level-1-NA-UNLIM	-55 to 150	LM35AH	Samples
LM35CAH	ACTIVE	TO	NDV	3	1000	TBD	Call TI	Call TI	-40 to 110	LM35CAH	Samples
LM35CAH/NOPB	ACTIVE	TO	NDV	3	1000	Green (RoHS & no Sb/Br)	Call TI | POST-PLATE	Level-1-NA-UNLIM	-40 to 110	LM35CAH	Samples
LM35CAZ/LFT4	ACTIVE	TO-92	LP	3	2000	Green (RoHS & no Sb/Br)	CU SN	N / A for Pkg Type		LM35 CAZ	Samples
LM35CAZ/NOPB	ACTIVE	TO-92	LP	3	1800	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type	-40 to 110	LM35 CAZ	Samples
LM35CH	ACTIVE	TO	NDV	3	1000	TBD	Call TI	Call TI	-40 to 110	LM35CH	Samples
LM35CH/NOPB	ACTIVE	TO	NDV	3	1000	Green (RoHS & no Sb/Br)	Call TI | POST-PLATE	Level-1-NA-UNLIM	-40 to 110	LM35CH	Samples
LM35CZ/LFT1	ACTIVE	TO-92	LP	3	2000	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type		LM35 CZ	Samples
LM35CZ/LFT4	ACTIVE	TO-92	LP	3	2000	TBD	Call TI	Call TI			Samples
LM35CZ/NOPB	ACTIVE	TO-92	LP	3	1800	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type	-40 to 110	LM35 CZ	Samples
LM35DH	ACTIVE	TO	NDV	3	1000	TBD	Call TI	Call TI	0 to 70	LM35DH	Samples
LM35DH/NOPB	ACTIVE	TO	NDV	3	1000	Green (RoHS & no Sb/Br)	POST-PLATE	Level-1-NA-UNLIM	0 to 70	LM35DH	Samples
LM35DM	NRND	SOIC	D	8	95	TBD	Call TI	Call TI	0 to 100	LM35D M	
LM35DM/NOPB	ACTIVE	SOIC	D	8	95	Green (RoHS & no Sb/Br)	SN | CU SN	Level-1-260C-UNLIM	0 to 100	LM35D M	Samples
LM35DMX	NRND	SOIC	D	8	2500	TBD	Call TI	Call TI	0 to 100	LM35D M	
LM35DMX/NOPB	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	SN | CU SN	Level-1-260C-UNLIM	0 to 100	LM35D M	Samples
LM35DT	NRND	TO-220	NEB	3	45	TBD	Call TI	Call TI	0 to 100	LM35DT	

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
LM35DT/NOPB	ACTIVE	TO-220	NEB	3	45	Green (RoHS & no Sb/Br)	CU SN	Level-1-NA-UNLIM	0 to 100	LM35DT	Samples
LM35DZ	OBSOLETE	TO-92	LP	3		TBD	Call TI	Call TI			
LM35DZ/LFT1	ACTIVE	TO-92	LP	3	2000	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type		LM35 DZ	Samples
LM35DZ/LFT2	ACTIVE	TO-92	LP	3	2000	Green (RoHS & no Sb/Br)	CU SN	N / A for Pkg Type		LM35 DZ	Samples
LM35DZ/LFT4	ACTIVE	TO-92	LP	3	2000	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type		LM35 DZ	Samples
LM35DZ/LFT7	ACTIVE	TO-92	LP	3	2000	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type		LM35 DZ	Samples
LM35DZ/NOPB	ACTIVE	TO-92	LP	3	1800	Green (RoHS & no Sb/Br)	SN | CU SN	N / A for Pkg Type	0 to 100	LM35 DZ	Samples
LM35H	ACTIVE	TO	NDV	3	1000	TBD	Call TI	Call TI	-55 to 150	LM35H	Samples
LM35H/NOPB	ACTIVE	TO	NDV	3	1000	Green (RoHS & no Sb/Br)	Call TI | POST-PLATE	Level-1-NA-UNLIM	-55 to 150	LM35H	Samples

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

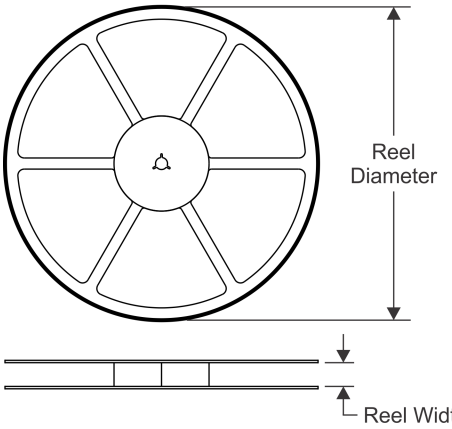
(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

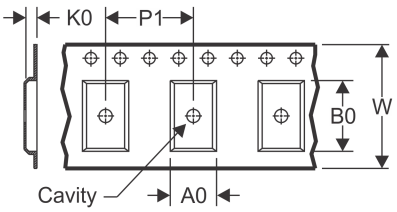
(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

TAPE AND REEL INFORMATION

REEL DIMENSIONS

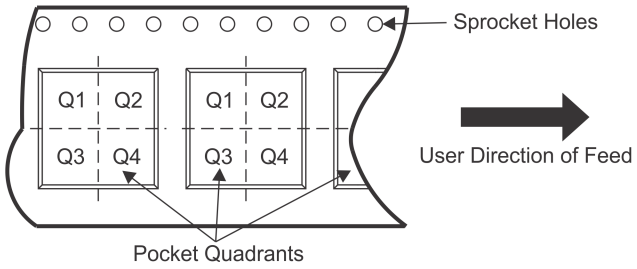


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

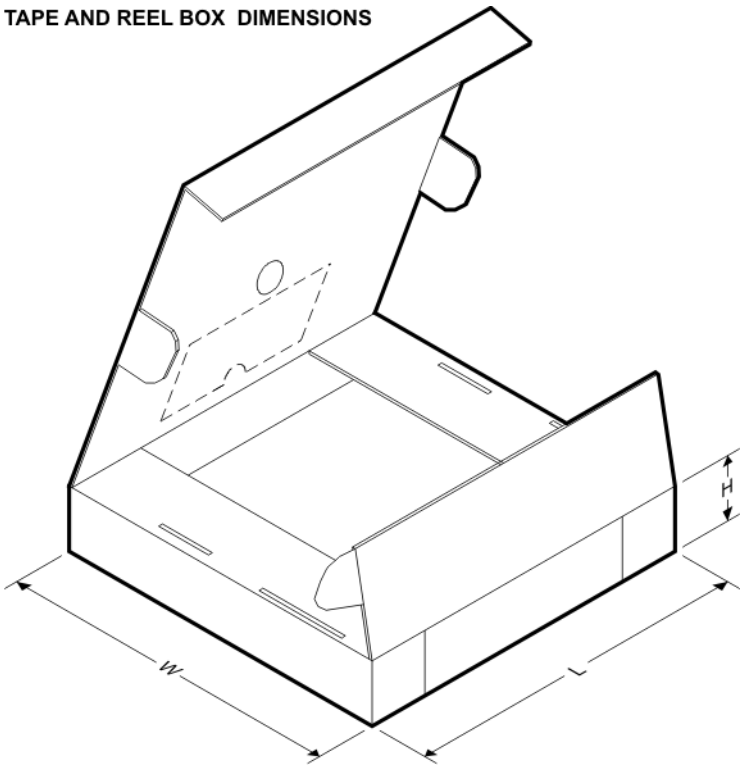
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
LM35DMX	SOIC	D	8	2500	330.0	12.4	6.5	5.4	2.0	8.0	12.0	Q1
LM35DMX/NOPB	SOIC	D	8	2500	330.0	12.4	6.5	5.4	2.0	8.0	12.0	Q1

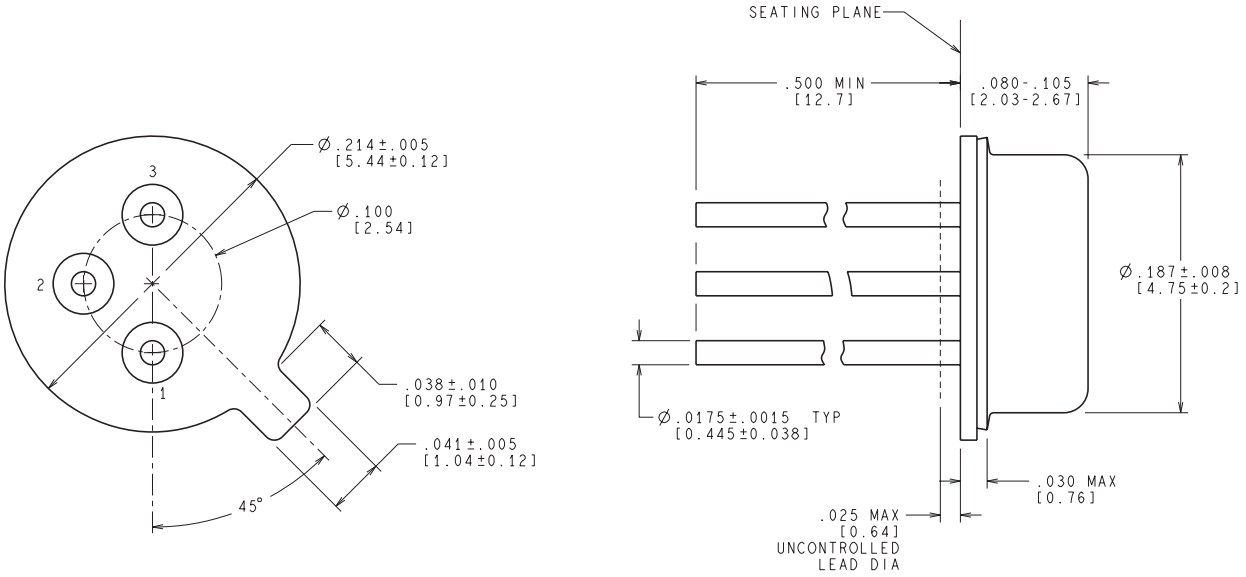
TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
LM35DMX	SOIC	D	8	2500	367.0	367.0	35.0
LM35DMX/NOPB	SOIC	D	8	2500	367.0	367.0	35.0

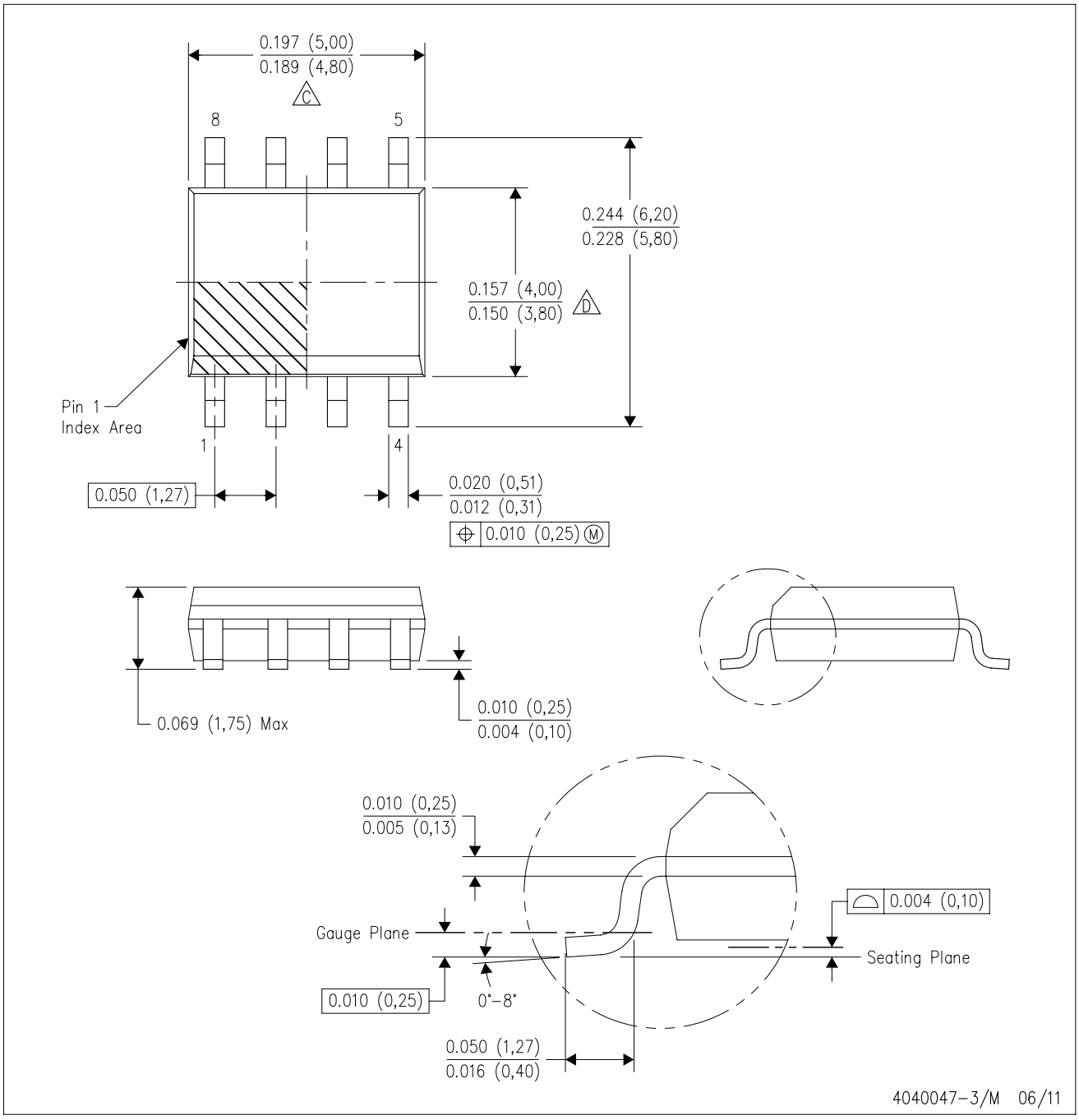
NDV0003H



CONTROLLING DIMENSION IS INCH
VALUES IN [ ] ARE IN MILLIMETERS

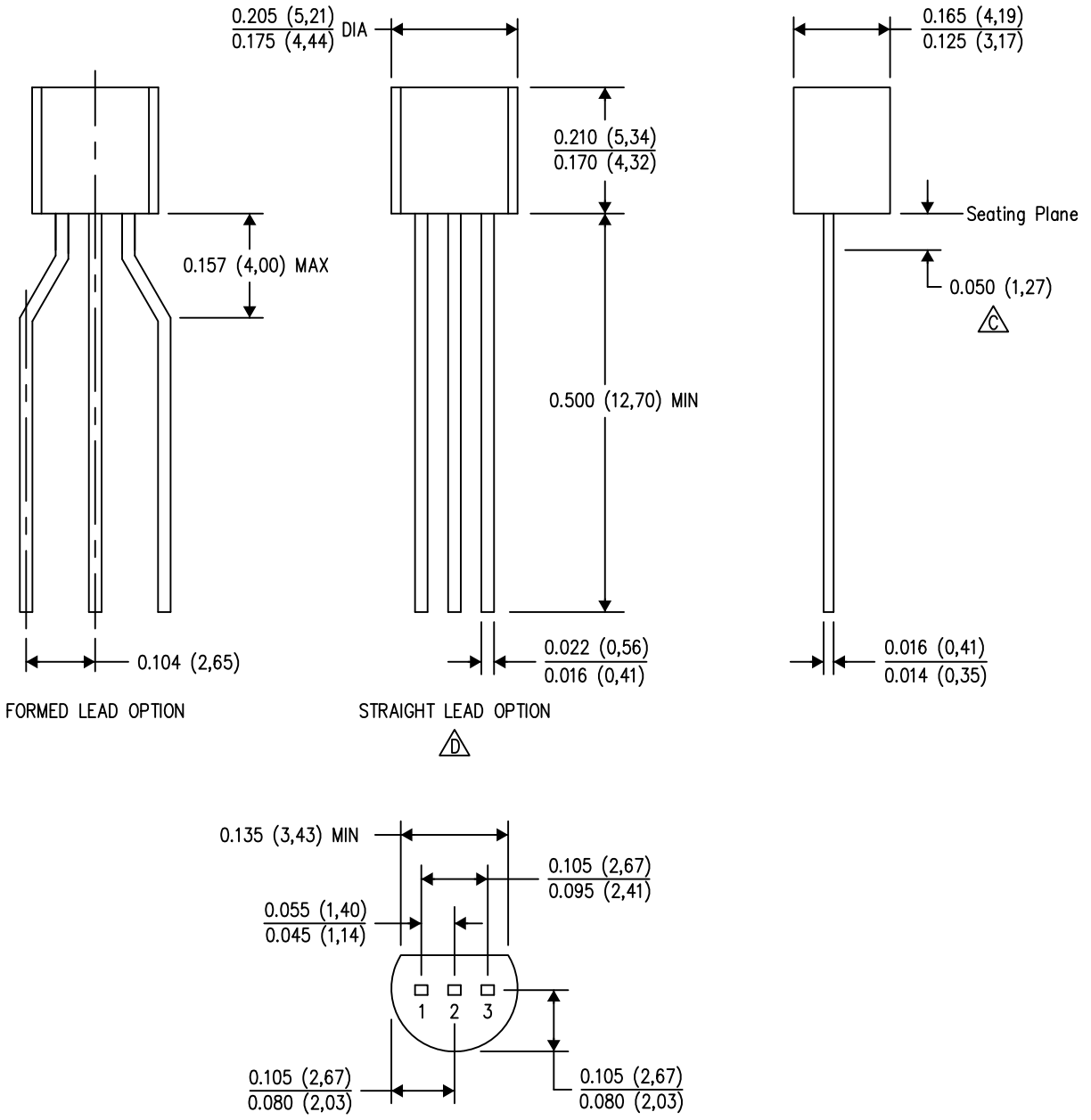
H03H (Rev F)

D (R–PDSO–G8) PLASTIC SMALL OUTLINE



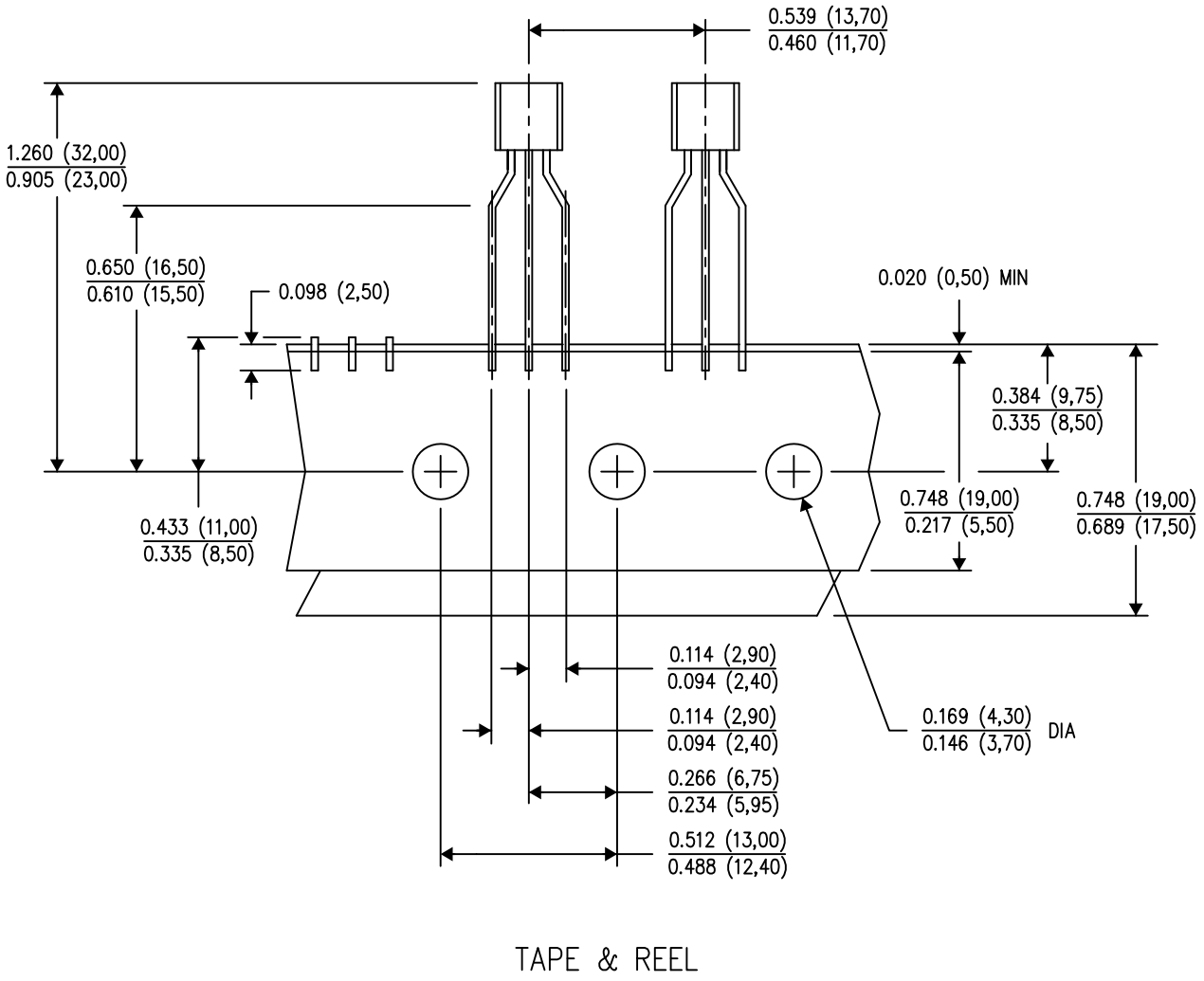
NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.006 (0,15) each side.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0.017 (0,43) each side.
E. Reference JEDEC MS–012 variation AA.

LP (O–PBCY–W3) PLASTIC CYLINDRICAL PACKAGE



NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Lead dimensions are not controlled within this area.
D. Falls within JEDEC TO–226 Variation AA (TO–226 replaces TO–92).
E. Shipping Method:
Straight lead option available in bulk pack only.
Formed lead option available in tape & reel or ammo pack.
Specific products can be offered in limited combinations of shipping mediums and lead options.
Consult product folder for more information on available options.

LP (O–PBCY–W3) PLASTIC CYLINDRICAL PACKAGE



NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Tape and Reel information for the Formed Lead Option package.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

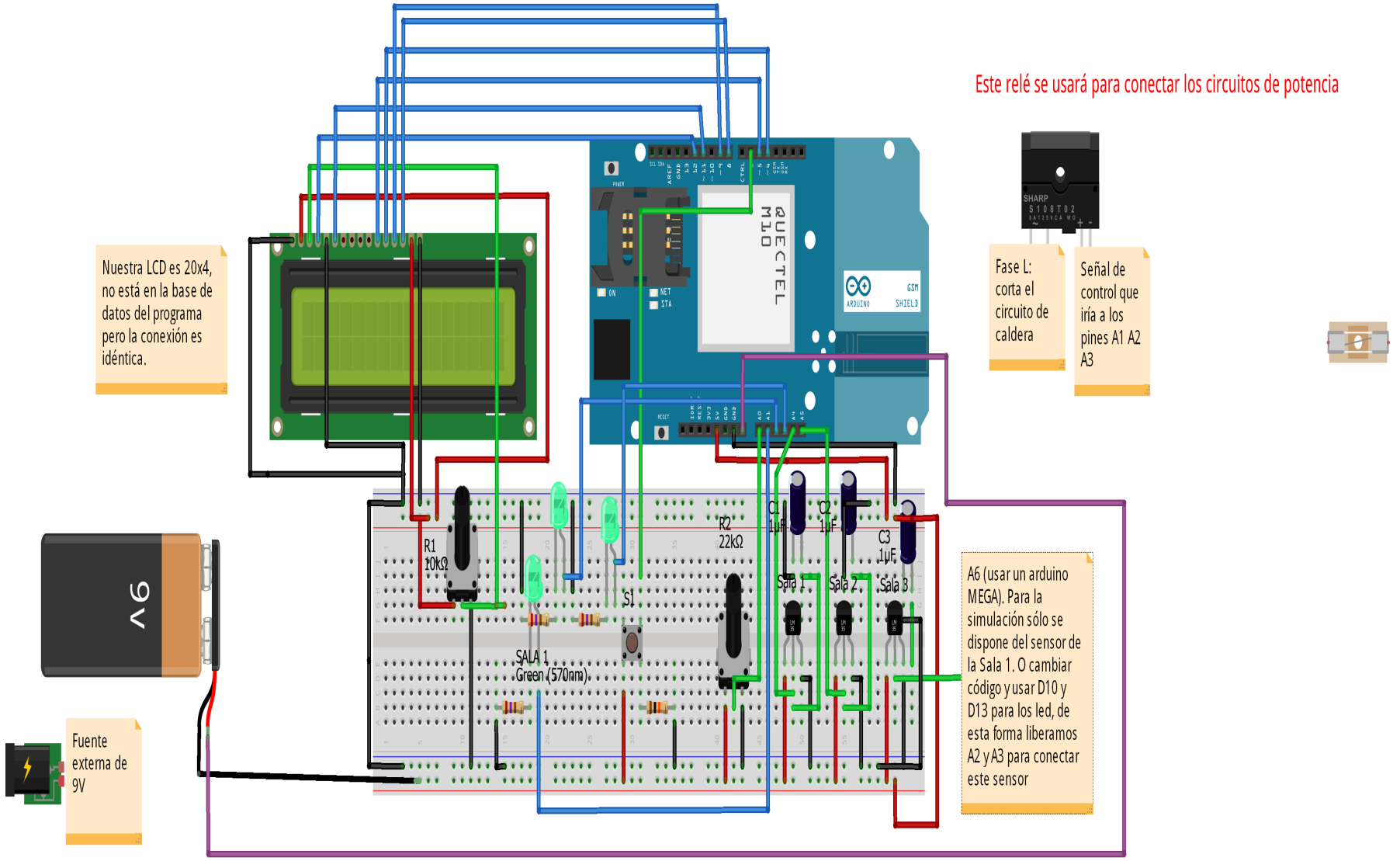
No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265


Este relé se usará para conectar los circuitos de potencia
Nuestra LCD es 20x4, no está en la base de datos del programa pero la conexión es idéntica.
Fase L: corta el circuito de caldera
Señal de control que iría a los pines A1 A2 A3
9V
Fuente externa de 9V
R1 10kΩ
R2 22kΩ
C1 1µF
C2 1µF
C3 1µF
S1
Sala 1
Sala 2
Sala 3
SALA 1 Green (570nm)
A6 (usar un arduino MEGA). Para la simulación sólo se dispone del sensor de la Sala 1. O cambiar código y usar D10 y D13 para los led, de esta forma liberamos A2 y A3 para conectar este sensor
fritzing