WeatherLAN:
Sensor de temperatura y humedad con Arduino

Héctor Fiel
contacto@hfiel.es
http://weatherlan.hfiel.es

28 de junio de 2011

Resumen

Sistema de control de temperatura y humedad bajo la plataforma libre Arduino, que permite el registro de la información recogida en una base de datos consultable de forma gráfica mediante página web.

El proyecto busca minimizar el coste y maximizar la sencillez de uso, realizando un desarrollo a pequeña escala que facilite su posterior implementación en aquellos entornos donde las soluciones comerciales existentes no sean adecuadas.

En el desarrollo software del proyecto se usan exclusivamente soluciones libres y de uso gratuito.

Índice

Parte I

Anexos al documento

Como anexos a este documento tenemos los siguientes puntos:

- Documento de instrucciones, con los datos completos necesarios para la cosntrucción del sensor
- Código Arduino, con el código fuente empleado en la parte HW del sensor
- Código web, con una parte del código empleado en la generación de las gráficas.
- Código SQL de creación de tablas, con los datos para la creación de las tablas en MySQL, así como los campos necesarios.

Parte II

Presentación del proyecto

1. Introducción

Este proyecto de investigación se engloba dentro de la *Beca de Investigación en las Nuevas Tecnologías Aplicadas al Ámbito Cultural* que el autor está desarrollando en la *Consejería de Cultura* de la *Junta de Andalucía*.

El presente texto abarca todo el desarrollo del prototipo inicial, estando dividido en 4 secciones:

1. *Presentación del proyecto:* la presente parte introduce los objetivos del proyecto y ofrece una idea general del mismo.
2. *Anteproyecto:* detalla las ideas iniciales de las primeras fases de investigación, muestra las tecnologías y herramientas escogidas para el desarrollo, así como la estimación temporal y económica del mismo.

3. *Primeras pruebas:* los primeros pasos con los sensores y el sistema Arduino permitieron definir correctamente algunos aspectos aparecidos en el anteproyecto, así como descubrir algunos problemas existentes, que han sido subsanados de cara al prototipo final.

4. *Desarrollo final:* mediante explicaciones detalladas del desarrollo y fabricación del prototipo final, tanto a nivel de hardware como de software, ofrece todos los datos que permiten reproducir el sistema para uso propio, así como la puesta en producción del sistema completo.

5. *Conclusiones:* las pruebas y resultados del prototipo, así como la evaluación de los objetivos alcanzados por el sistema, permiten cerrar el proyecto generando una adecuada imagen del conjunto final.

1.1. Fundamentación

Este proyecto nace al detectar dos necesidades importantes que no se encuentran cubiertas correctamente en la actualidad:

1. Control de salas de archivo y arte: Las organizaciones dedicadas al archivo y almacenamiento de documentos u obras de arte, necesitan garantizar la correcta conservación de los materiales gestionados. Para lograrlo es preciso disponer de un sistema de control de temperatura y humedad que permita llevar un registro adecuado y preciso de las condiciones ambientales existentes en cada momento, y permita definir correctamente las medidas de almacenamiento adecuadas.

2. Ahorro de gastos en salas de CPD: En todos los CPD[1] existen equipos de aire acondicionado que permiten una adecuada refrigeración de la sala y equipos. Sin embargo, la mayoría de centros pequeños no disponen de un sistema que permita controlar la temperatura real, y el sistema de refrigeración se suele configurar para obtener la mínima temperatura posible. El correcto ajuste de la temperatura puede suponer un importante ahorro energético, sin llegar a poner en riesgo la salud del equipamiento informático.

En ambas situaciones, los sistemas comerciales destinados a este fin pueden resultar extremadamente caros y los centros más pequeños no pueden asumir los costes asociados. Se opta entonces por soluciones que no resuelven todas las necesidades, o directamente se omiten los sistemas de control. Este proyecto pretende ofrecer una

[1] *Centro de Proceso de Datos*: sala donde se concentran los sistemas informáticos centrales de una organización.

alternativa con un coste bajo que facilite su implementación en aquellos entornos que no pueden permitirse el acceso a las soluciones existentes.

2. Objetivos del proyecto

Se buscan los siguientes objetivos:

- Desarrollar un sistema de medición, almacenamiento y consulta de temperatura y humedad, con dos sondas de temperatura y una sonda de humedad. El sistema debe permitir, como mínimo:
  - Medir la temperatura y humedad instantánea.
  - Almacenar los datos en un servidor de bases de datos.
  - Realizar consultas remotas mediante Web a la base de datos.
  - Ampliación sencilla del número de sondas.
- Emplear herramientas y plataformas libres en el proyecto.
- Buscar el mínimo coste posible para el proyecto, manteniendo niveles de calidad aceptables.
- Alcanzar un nivel de desarrollo que permita el uso real en un entorno de producción, incluso a nivel comercial.
- Documentar adecuadamente todas las fases para permitir tanto el seguimiento del proyecto como su reproducción por terceras partes.
- Obtener resultados finales sobre la funcionalidad y resultados del proyecto.
- Liberar online todo el material generado, tanto en código como documentación, en la página Web del autor, `http://weatherlan.hfiel.es`. Esta liberación se hará conforme se vayan generado documentos, para permitir un seguimiento de la evolución del proyecto. Para la misma se hará uso de diferentes licencias libres, detalladas en la siguiente sección.

3. Licencias del proyecto

En todo el desarrollo del proyecto *weatherlan* se usan licencias libres, que permiten el uso y la modificación por terceros. Se diferencia el licenciamiento de la documentación y de los programas generados, usando licencias diferentes para cada caso. A continuación se resume el contenido y alcance de las licencias, aunque se indica la necesidad de consultar los textos completos originales para resolver cualquier duda relacionada con el uso de las mismas.

Sin interferencia con el uso de las licencias ni su alcance, el proyecto completo se halla amparado en el registro de la propiedad intelectual como obra del autor.

- *Documentación*: el presente texto así como los manuales del sistema se liberan bajo licencia Creative Commons CC - BY - SA 3.0, que permite el uso, reproducción y modificación libre de los materiales bajo las siguientes condiciones:

  1. Cualquier uso o modificación de la obra debe identificar al autor original.
  2. Cualquier obra derivada debe compartirse bajo la misma licencia.

  La licencia completa puede consultarse en:
  `http://creativecommons.org/licenses/by/3.0/es/`

- *Software*: el software del proyecto se libera bajo la licencia GPL. Esta licencia permite el uso, modificación y transferencia libre del software asociado, siempre que cualquier se cumplan las siguientes condiciones:

  1. El usuario es libre de usar el software.
  2. El usuario es libre de distribuir el software.
  3. El usuario es libre de modificar el software.
  4. El usuario es libre de distribuir sus modificaciones, siempre que lo haga bajo la misma licencia GPL, y permita el acceso al código fuente modificado sin restricciones.

  La licencia completa puede consultarse en:
  `http://gnu.org/copyleft/gpl/`
  Adicionalmente, se incluye una copia de la licencia junto con los programas.

3.1. Uso comercial del proyecto

Las licencias escogidas no impiden en ningún caso el uso comercial del proyecto. No obstante, y debido a las mismas, cualquier modificación que se haga en el mismo

deberá hacerse disponible bajo las mismas licencias. Esto incluye los documentos y manuales, así como el código fuente usado en la plataforma *Arduino*, o en el servidor Web asociado.

3.2. Descarga

Una vez completado y presentado el proyecto, todos los materiales de documentación, esquemas y software, serán publicados en la web `http://weatherlan.hfiel.es`.

Parte III

Anteproyecto

En la presente parte se procede a describir las elecciones y estimaciones llevadas a cabo en la primera parte del proyecto, antes de proceder a la construcción y programación del primer prototipo. Como se podrá observar más adelante, algunas de las ideas iniciales se tuvieron que modificar conforme se fue avanzando en el desarrollo.

4. Elección de tecnologías

Para el desarrollo del proyecto se han seleccionado inicialmente las siguientes tecnologías y software:

- Plataforma *Arduino* como sistema hardware, junto con su entorno de desarrollo software.
- Conectividad Ethernet para la comunicación del equipo hardware y el servidor de datos.
- Servidor Web *Apache* para la publicación de las páginas Web.
- Servidor de bases de datos *MySQL* para el almacenamiento y consulta de los datos.
- Lenguaje de programación *PHP* para la generación dinámica de las páginas de consulta y registro de datos.
- Librerías *Open Flash Charts 2* y *jQuery* para la generación de los gráficos de temperatura y humedad en la página Web.

4.1. Entorno de desarrollo y producción

Todo el software a utilizar durante el desarrollo es libre y multiplataforma (para sistemas *Windows*, *GNU/Linux* y *Mac OS X*), y no hay restricciones a la hora de escoger un entorno en concreto, tanto para la programación del sistema Arduino, como para la parte Web del proyecto. De cara a la utilización en un entorno de

producción, la parte Arduino una vez terminada es completamente autónoma, y el desarrollo Web es un estándar portable a cualquier plataforma, lo que permite su uso en prácticamente cualquier entorno existente, siempre que se cumplan los requisitos mínimos de los programas utilizados.

5. Plataforma Arduino

5.1. Presentación de la plataforma

Arduino (`http://arduino.cc`) es una plataforma basada en microcontrolador que integra en una pequeña placa PCB[2] todos los componentes necesarios para crear un sistema informático completo. Sus principales características, en su última versión, son:

- Basado en microcontrolador ATmega 328, funcionando a 16MHz.
- Memoria SRAM de 2KB
- Memoria Flash de 32KB
- Memoria EEPROM de 1KB
- 14 entradas y salidas digitales
- 6 entradas y salidas analógicas
- Conexión USB para comunicación con un ordenador y alimentación.
- Diversos módulos de expansión, incluyendo módulos de comunicación para Ethernet, Wifi, Bluetooth, ZigBee o GPRS, así como módulos GPS y de ampliación de memoria.
- Programación mediante herramientas libres y gratuitas, disponibles para los principales sistemas operativos.
- Coste reducido, al basarse en componentes simples disponibles comercialmente en grandes cantidades.

5.2. Posibilidades de la plataforma

La unión de la potencia de procesamiento del microcontrolador, la conectividad exterior tanto analógica como digital y los módulos de comunicaciones disponibles en la plataforma, combinados con otros circuitos electrónicos disponibles comercialmente, permiten la elaboración de sistemas complejos que hace unos años sólo estaban al alcance de empresas con los recursos necesarios. Todo esto, favorecido por el reducido coste, la suave curva de aprendizaje de la plataforma y la facilidad de

[2] *Printed Circuit Board*, placa de circuito impreso.

su programación, acerca a aficionados y pequeñas empresas el desarrollo económico de proyectos y prototipos con plena funcionalidad. Arduino funciona como corazón de los sistemas, aportando capacidad de proceso y conexionado, y la programación específica unida a los circuitos adecuados permite lograr los objetivos concretos de cada proyecto. La combinación de múltiples placas Arduino permite la elaboración de sistemas distribuidos, ampliando el alcance y capacidad de los sistemas. Ejemplos de uso actuales de la plataforma incluyen sistemas de medición, automatismo y control de procesos, domótica,...

5.3. Arduino como plataforma libre

Se considera que Arduino es una plataforma libre dado que las instrucciones de fabricación completas están disponibles para cualquier usuario, incluyendo los esquemas eléctricos y el software que lo hace funcionar. Esto permite que cualquier persona pueda construir su propio Arduino, incluso de forma casera, con las modificaciones que considere necesarias. Dicha característica ha propiciado la aparición de diversos clones, algunos idénticos al original, y otros con diversas adaptaciones, siendo la práctica totalidad compatibles entre si. De igual forma, existen múltiples extensiones para la plataforma que amplían sus capacidades, y que han sido desarrolladas tanto por los creadores originales como por terceras partes.

Esta apertura del sistema ha permitido una amplia difusión, y la aparición durante los últimos años de muchos proyectos, algunos muy complejos y avanzados, basados en la plataforma. Cualquier persona o empresa puede desarrollar y comercializar tanto clones de la plataforma como extensiones, con la única limitación de no ponerle el nombre Arduino a ninguna de las creaciones.

5.4. Familias Arduino disponibles

La plataforma Arduino presenta múltiples versiones, la mayoría fruto de la evolución del sistema. Dentro de todas las opciones destacamos tres familias principales:

1. Familia básica, orientada a pequeños proyectos y prototipado, y que incluye tanto versiones normales como algunas de prestaciones superiores para proyectos con mayor requerimiento de potencia o conexionado. Es la familia que suele emplearse para el desarrollo inicial de los proyectos, y para la que existen el mayor número de extensiones. Esta familia es la escogida para desarrollar este proyecto.

2. Familia Nano y Mini, orientada a proyectos empotrados y despliegues a media-

na escala, basada en placas más pequeñas. Suele emplearse en proyectos ya depurados que pasan a su fase de producción, o donde se requiere un tamaño reducido.

3. Familia Lillypad, orientada a proyectos de ropa inteligente, pensada para integrar una plataforma Arduino en los tejidos (como por ejemplo, para integrar en una chaqueta botones que permitan controlar un reproductor MP3).

5.5. Versiones

La plataforma Arduino surgió en el año 2000, y desde entonces ha pasado por diversas revisiones. La versión actual, y la escogida para este proyecto, es la Duemilanove. Por lo general, el cambio de versión no supone la incompatibilidad con los programas desarrollados para las versiones anteriores, y sirve para introducir mejoras en el sistema, como mayor potencia, memoria o nº de E/S.

Igualmente surgen nuevas versiones del software asociado, que introducen mejoras en las funciones disponibles en la programación de la plataforma. Se debe diferenciar el software interno de la plataforma (el que la hace funcionar y ejecuta los programas) y el software de desarrollo que se utiliza en un ordenador para crear los programas.

6. Servidor Web y de bases de datos

Para el lado software, como se ha mencionado anteriormente, se han seleccionado el servidor Web Apache (`http://httpd.apache.org`), combinado con la base de datos MySQL (`http://mysql.com`) y el lenguaje PHP (`http://www.php.net`). Estas tres tecnologías son ampliamente utilizadas en conjunto en proyectos de páginas Web, y su utilización y configuración están ampliamente documentadas. Los tres elementos son software libre, y su explicación queda fuera del ámbito del proyecto.

7. Costes estimados del proyecto

Dado que las herramientas software a emplear son todas de libre uso y sin coste, los gastos estimados del proyecto radican exclusivamente en la adquisición del hardware, principalmente de la placa Arduino junto con su expansión de comunicaciones Ethernet[3]. Además se requerirán componentes adicionales para completar el proyecto, como las sondas de temperatura y humedad. Tras consultar diversas tiendas en Internet, la evaluación inicial de costes es la siguiente:[4]

Material	**Unidades**	**Precio Unitario**
Arduino Duemilanove	1	22 €
Ethernet Shield	1	30 €
Sonda de temperatura	2	2,5 €
Sonda de humedad	1	18 €
	Total	**75 €**

A estos componentes principales habrá que añadir el coste de algunos adicionales, como cables, resistencias o condensadores, que permitan el conexionado de todos los elementos, y cuyo precio es muy bajo (en torno a céntimos de euro por unidad). Igualmente son necesarias algunas herramientas, como placa de prototipos o soldador, de precio mayor pero reutilizables, y de las cuales ya se dispone de proyectos anteriores.

[3]A la hora de publicar esta documentación, ya se encuentran disponibles tanto la nueva versión Arduino UNO, que es completamente compatible con las anteriores, como de la placa ethernet, sin variaciones apreciables en el precio.

[4]En estos costes no se incluyen impuestos ni gastos de transporte

8. Fases y planificación temporal

Se indican a continuación las fases que se plantean, junto con su planificación temporal previa. No obstante, tanto las fases como la temporización podrán sufrir variaciones conforme avance el proyecto.

8.1. Anteproyecto, evaluación y adquisición de la plataforma Arduino

Introducción y presentación del proyecto, elección de tecnologías, valoración y decisión de compra sobre las diferentes versiones de la plataforma y componentes a utilizar. Adquisición de componentes. Esta fase está representada en el presente documento. Se han invertido aproximadamente 5 horas en el proceso.

8.2. Aprendizaje básico de la plataforma

Configuración del entorno de desarrollo, conexión y pruebas iniciales de la plataforma. Aprendizaje básico de la metodología de programación. Conexionado de la expansión Ethernet. Se estiman entre 8 y 12 horas.

8.3. Combinación inicial de los componentes

Esquemas eléctricos, combinación de los componentes en la placa de prototipado, aprendizaje de la programación que permita la interacción con los mismos. Se estiman entre 2 y 4 horas.

8.4. Versión preliminar

Versión preliminar de la parte Arduino que permita la consecución de los objetivos referentes a la lectura de las medidas. Se estiman entre 12 y 18 horas.

8.5. Versión estable

Versión estable de la parte Arduino, que incluya el traslado de los componentes a una placa definitiva, la correcta medición y el procesado de los datos. Calibración exacta de las medidas de temperatura y humedad. Limpieza y corrección de comentarios y código de programación. Depuración del proceso que optimice tanto velocidad como memoria. Se estiman entre 8 y 10 horas.

8.6. Conexión y comunicaciones

Conexión Ethernet. Almacenamiento de la información en la base de datos remota y desarrollo de la web de consulta. Se estiman entre 12 y 18 horas.

8.7. Evaluación e implementación de mejoras

Consideración de mejoras y sus posibilidades de implementación en tiempo razonable. Se estiman hasta 20 horas, en su caso.

8.8. Documentación

Finalización de la documentación y preparación de presentaciones del proyecto. Se estiman entre 8 y 12 horas.

8.9. Cierre del proyecto

Preparación de la información asociada al proyecto (código, esquemas, documentos) para permitir su acceso online a terceras personas. Se estima 1 hora.

Parte IV

Primeras pruebas

Una vez completada la fase inicial de investigación, comienza el verdadero desarrollo del sistema con las pruebas iniciales. Durante las mismas, dado que se han ido descubriendo características más concretas de las plataformas escogidas, se han efectuado modificaciones en los planteamientos y elecciones previos, que han permitido avanzar correctamente hacia el prototipo final. Igualmente se han encontrado problemas que han obligado a replantear algunos aspectos del circuito. Durante estas pruebas se han utilizado placas de prototipado que han permitido modificar la interconexión de los circuitos de manera sencilla.

Todos los pasos que se detallan a continuación, tanto a nivel de hardware como de software, se han ido documentando en la web del autor, para permitir un seguimiento de la evolución del proyecto. No obstante, dicha información se traslada a este documento para facilitar el acceso a la misma. Se ha respetado por lo tanto la evolución temporal del proyecto, y por consiguiente sus avances, y sus fallos. No obstante, dado que mucha de la información original resulta superflua, se ha reestructurado para reducir su tamaño: por ejemplo, originalmente se ofrecían hasta 4 versiones del código, practicamente idénticas, mientras que aquí sólo se incluyen las dos versiones con las modificaciones más importantes. A pesar de que esta fase dista mucho de ser el producto final, se considera que puede resultar de utilidad a otros usuarios que quieran modificar el proyecto, para evitar que incurran en los mismos problemas que se han encontrado a lo largo del desarrollo.

Se considera por lo tanto necesario recomendar al lector que, en caso de que su interés sea simplemente reproducir el prototipo, avance hasta la parte V del documento, omitiendo por completo los siguientes apartados.

9. Componentes

Tras la elección de todas la tecnologías se procede a la adquisición de los componentes para comenzar las primeras versiones del sensor. El corazón del sistema es la placa Arduino Duemilanove junto con el módulo Ethernet Shield. Respecto a las sondas de temperatura y humedad, tenemos el circuto 808H5V5 para la sonda de humedad, y los circuitos MCP9700A y LM35 para las sondas de temperatura. Se van a realizar pruebas con ambos circuitos de temperatura de cara a evaluar cual es el más adecuado para nuestro proyecto. Además, se buscará la forma de mejorar

la precisión de las lecturas, y para ello deberemos describir algunos aspectos de la arquitectura interna de Arduino.

9.1. Sensores MCP9700A, LM35 y 808H5V5

En primer lugar, daremos algunos datos concretos sobre los circuitos a emplear. No obstante, se recomienda consultar las hojas de características del fabricante, que ofrecen mucha mayor información de cara al correcto uso de los mismos.

9.1.1. MCP9700A

Comenzaremos describiendo la sonda de temperatura MCP9700A[5] de la casa Microchip. Tiene un rango entre -40ºCy 125ºC y un error típico de +-1ºC. Puede alimentarse con cualquier tensión entre 2,3V y 5,5V, y entrega 10mV por ºC, con valores de salida entre 100mV (a -40ºC) y 1,75V (a 125ºC). Debido a la relación que presenta entre temperatura y voltaje, habrá que hacer un sencillo cálculo para traducir las lecturas de mV a los grados que representa (lo veremos más adelante, en la parte del código). El cálculo concreto se puede obtener en la hoja de características del dispositivo.

9.1.2. LM35

El segundo sensor de temperatura, el LM35[6] de la casa National, tiene un rango entre -55ºC a 150ºC, con un error típico de 0,75ºC. La alimentación oscila entre los 4 y los 30 voltios, y entrega 10mV por ºC, con valores de 0mV (0ºC) hasta 1500mV (150ºC). Notar que decimos que mide desde -50ºC, y sin embargo entrega 0mV a 0ºC. Esto se debe a que se trata de un sensor de grados centígrados que, sin añadirle ningún componente, entrega valores de mV directamente proporcionales a la temperatura positiva en grados centígrados, por ejemplo 170mV para una temperatura de 17º, y 1V para una temperatura de 100ºC. Esto facilita enormemente la utilización en aquellos entornos que no necesitan de lecturas de temperatura negativas (como es nuestro caso). En caso de necesitar valores negativos, es necesario añadir a la salida una resitencia conectada a la tensión de alimentación en negativo, pero esta aplicación queda fuera de las necesidades de nuestro proyecto actual.

[5] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en027103

[6] http://www.national.com/ds/LM/LM35.pdf

9.1.3. 808H5V5

Respecto a la sonda de humedad, modelo 808H5V5[7] de Sencera, mide entre el 0 % y el 100 % de humedad relativa, teniendo la mayor precisión entre el 30 % y el 80 % (medida a 25ºC de temperatura). El error típico en la medida es del +-4 % durante el primer año de uso, a partir del cual pierde en torno a 1 % adicional por cada año. La sonda puede operar entre -40ºC y 85ºC, y tiene un tiempo de respuesta de 15s como máximo. La alimentación es exclusivamente a 5V, y los valores de salida son 0,8V para un 0 % de humedad relativa, y 3,9V para un 100 %.

9.2. Entradas analógicas en Arduino, nivel de referencia y conversión A/D

Para poder obtener el mejor resultado de los sensores, necesitamos entender correctamente el funcionamiento de la conversión analógico/digital que lleva a cabo el microprocesador de Arduino. Como podemos comprobar por la descripción de la placa, disponemos de 6 de estas entradas[8], con una resolución de 10 bits (esto es, que miden $2^{10} = 1024$ niveles diferentes en la señal recibida) y que admiten valores entre 0V y 5V (por defecto). Podemos ajustar este rango de entrada mediante la selección de la señal de referencia analógica. En la versión de Arduino que estamos empleando (con un ATMega 328) disponemos de las siguientes referencias:

- 5V (la que viene por defecto)
- 1,1V (generada de manera interna por la placa Arduino)
- Externa, que la recibe por medio del pin AREF, y podemos usar para establecer una referencia propia.

Estos valores serán de gran importancia en la precisión de las medidas, dado que nuestros sensores (tanto de temperatura como de humedad) entregan un voltaje de salida concreto para cada medición, que dependerá del modelo de circuito empleado. En el mejor de los casos, nuestros sensores nos entregarían una señal entre 0V y 5V, que nos permitiera aprovechar todo el margen disponible, pero hemos visto que esto no sucede así, y deberemos escoger adecuadamente la referencia para poder obtener la precisión adecuada. Dado que tenemos dos referencias predeterminadas en Arduino (5V y 1,1V), estudiemos lo que supone cada una en relación a la precisión.

[7] `http://www.sensorelement.com/humidity/808H5V5.pdf`

[8] Como se podrá comprobar más adelante, estas 6 entradas se obtienen mediante multiplexación, lo que ha tenido consecuencias imprevistas.

9.2.1. Refencia de 5V

Si escogemos el rango de 0V a 5V, con 10 bits de precisión (los 1024 niveles mencionados) tenemos que cada 4,88mV de incremento de la señal (aproximadamente, 5000/1024) suponen una diferencia de 1 bit. Esto quiere decir que una señal que tuviera 0V nos daría el nivel 0, y una con 5V el nivel 1023. Cualquier señal intermedia generaría un nivel proporcional, y por ejemplo 2.3V nos darían como nivel 471 ($2,5 \times 1024/5 = 471,04$, redondeando al entero más cercano). Cualquier señal negativa o igual a cero nos dará nivel 0. Cualquier señal de 5V o superior nos dará como nivel 1023.

9.2.2. Referencia de 1,1V

Si empleáramos la referencia de 1,1V, cada bit representaría 1,07mV. Esto nos permite medir con mayor precisión señales más pequeñas, pero ya no podemos superar los 1,1V (cualquier señal superior daría como nivel 1023).

9.3. Niveles de salida de los componentes

Como vemos, los componentes que tenemos no entregan valores entre 0 y 5 voltios, sino que el rango de salida es mucho menor. Esto nos hace perder mucha precisión en las medidas, y para solucionarlo vamos a emplear la referencia interna de 1,1V. Sin embargo, como se ha podido observar, el rango de salida de los sensores sobrepasa con creces los 1,1V, por lo que usar esta referencia nos impone algunas restricciones, e incluso hace necesario adaptar uno de los sensores. No obstante esta elección, se incluye el diseño de circuitos y el código para las dos señales de referencia, dado que puede que en algunas circunstancias sea más útil la referencia por defecto.

9.3.1. Adaptando las salidas a 1,1V

Con respecto a los sensores de temperatura, y dado que la máxima salida está en torno a los 1,5V y 1,75V según el modelo, podemos emplear directamente la referencia de 1,1V, asumiendo que la contrapartida es una limitación en la máxima temperatura a medir: cuando se superen los 1,1V en la salida del sensor, el conversor de Arduino siempre leerá 1,1V (nivel digital 1023). En la sonda MCP9700A, el valor máximo que corresponde a los 1100mV son 60º, temperatura más que suficiente para nuestro proyecto, manteniendo la lectura de la temperatura mínima en -40ºC,

correspondiente a 100mV. Con respecto al sensor LM35, los 1100mV corresponden a 110ºC, y la mínima temperatura se sigue manteniendo en 0ºC a 0mV.

Sin embargo, esta referencia de 1,1V nos impide usar directamente el sensor de temperatura, dado que la salida correspondiente al 100 % de humedad relativa se sitúa en 3,9V, y en el caso de este sensor sí es probable que alcancemos valores muy elevados de humedad, y por lo tanto de voltaje de salida. Para poder usar los dos sensores de temperatura y el de humedad de forma simultánea con esta referencia, debemos reducir la salida del sensor de humedad, escalándola al nivel adecuado.

Para escalar la salida de 3,9V y convertirla en 1,1V, el mecanismo más sencillo que tenemos es un divisor de tensión, cuyo diagrama es[9]:

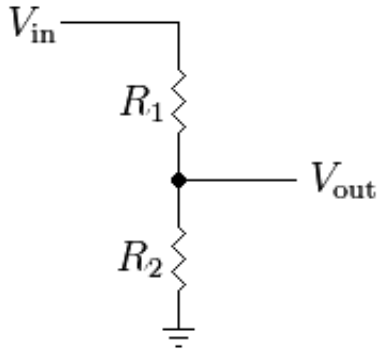

Figura 1: **Divisor de tensión** *[Wikimedia Commons]*

Y la relación que debemos cumplir es:

$$V_0 = \frac{R_2}{R_1 + R_2} \cdot V_i$$

Queremos que nuestra señal sea 0,28 veces la señal original ($1, 1/3, 9 = 0, 28$), de forma que la salida máxima del sensor es 1,1V, y la mínima de 0,8V se convierte en 0,224V.Para implementarlo, usaremos una resistencia fija para R2, de un valor comercial común (1kΩ), y para R1 usaremos un potenciómetro que nos permita un ajuste preciso hasta obtener la relación buscada. Con los valores que tenemos para los voltajes de entrada y salida, y el valor indicador para R2, el valor de R1 que debemos ajustar con el potenciómetro es de 2571,43Ω, por lo tanto usaremos un potenciómetro de 4,7kΩ, valor comercial facilmente disponible. Dado que las resistencias comerciales tiene una cierta tolerancia, el valor final real en el potenciómetro variará un poco

[9]Imagen del divisor de tensión obtenida de Wikimedia Commons, licencia CC - BY - SA 3.0. `http://commons.wikimedia.org/wiki/File:Resistive_divider.png`

respecto a este cálculo. Para el ajuste preciso, procederemos a alimentar el divisor con 5V, y ajustaremos el potenciómetro hasta obtener 1,4V exactos a la salida.

Con esta modificación, ya podemos emplear los tres sensores de forma simultánea con la referencia de 1,1V.

10. Conexionado de los componentes

La conexión de los componentes se hace en una placa de prototipado, que permite probar diferentes configuraciones de manera cómoda. La alimentación de los componentes se realiza desde el puerto de 5V de la placa Arduino, y el circuito se cierra en el puerto de tierra (GND). En los siguientes diagramas, se ha representado el terminal positivo de alimentación de los circuitos conectado al cable de color rojo, mientras que el negativo es de color negro. Las salidas analógicas de cada sensor emplean cables de diferentes colores.

10.1. Conexión para referencia de 5V

A la hora de conectar los componentes, vamos a utilizar condensadores de desacoplo entre las patillas de alimentación y tierra de cada elemento, para filtrar el ruido en la señal. Estos condensadores son de un valor muy reducido (1μf) y no afectan al funcionamiento del circuito. Los componentes se alimentarán a partir del pin de 5V de Arduino, y la tierra irá conectada al pin GND. La salida de cada componente se conectará a una entrada analógica (0, 1 y 2). Es importante respetar correctamente los pines positivos y negativos de cada componente. Por ejemplo, en nuestro sensor de temperatura, tal y como se ha representado aquí, el pin positivo es el de la izquierda, mientras que en el de humedad es el de la derecha. Si empleamos la referencia de 5V, la conexión es muy sencilla y directa, como vemos a continuación. Recordar que los 3 condensadores (C1, C2 y C3) son de 1μf:

10.2. Conexión para referencia de 1,1V

Si por el contrario vamos a utilizar la referencia de 1,1V, con el divisor de tensión, la conexión será la que mostramos a continuación. De nuevo los condensadores son de 1μf, y con respecto a los componentes del divisor, la resistencia es de 1kΩ y el potenciómetro de 4,7kΩ.

Recordar que el potenciómetro debe estar correctamente ajustado y conectado

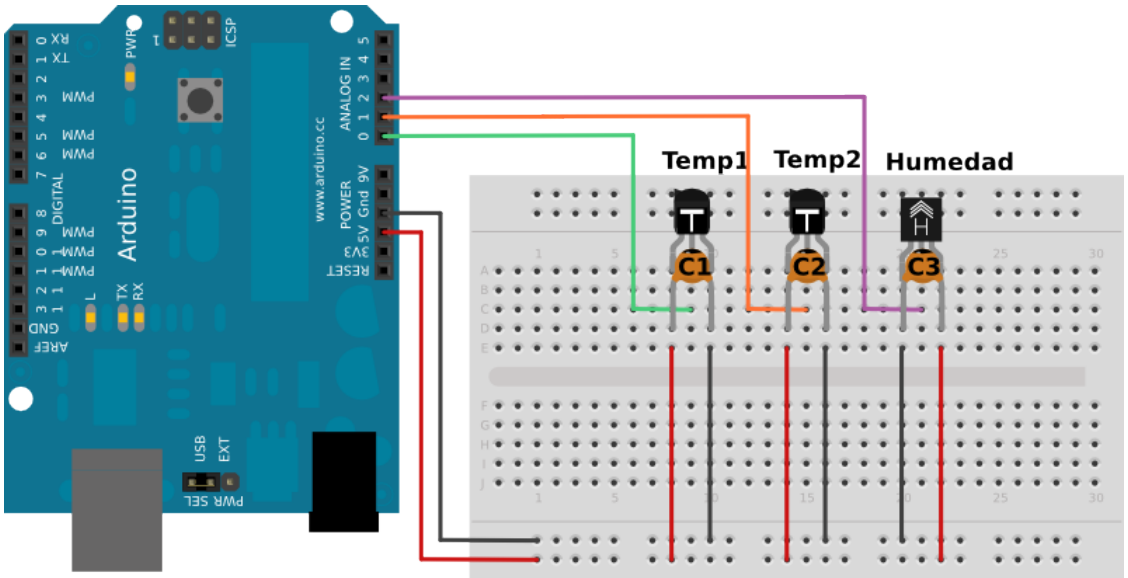

Figura 2: **Conexionado pruebas, versión 5V**

para permitir una lectura precisa, tal y como se menciona en el apartado 9.3.1. La conexión de la patilla central del sensor de humedad va a una de las patillas del potenciómetro (aquí se ha representado con el pequeño cable amarillo), y la otra patilla se conecta tanto a la entrada analógica de Arduino como a tierra por medio de la resistencia R1. La tercera patilla del potenciómetro (la que comparte lado con la salida del sensor de humedad) se deja sin conectar. Es importante remarcar que, a pesar de emplear otra referencia, los circuitos se siguen alimentando a 5V.

10.3. Imágenes del montaje

11. Código inicial

En esta primera versión no empleamos el módulo Ethernet, y nos centraremos en realizar las lecturas de las entradas analógicas. Ante las dos posibles referencias para los voltajes de entrada, el código de ambas versiones es prácticamente idéntico, a excepción del ajuste de la señal de referencia (5V ó 1,1V) y el consiguiente cálculo para convertir la señal leída en la entrada al valor en mV.

Comenzamos por la versión más sencilla, la de 5V, y después indicaremos las modificaciones necesarias para la versión a 1,1V. Veremos las ecuaciones que necesitamos para convertir los datos recibidos de las entradas analógicas en valores de temperatura y humedad relativa, ecuaciones que obtenemos de las hojas de carac-

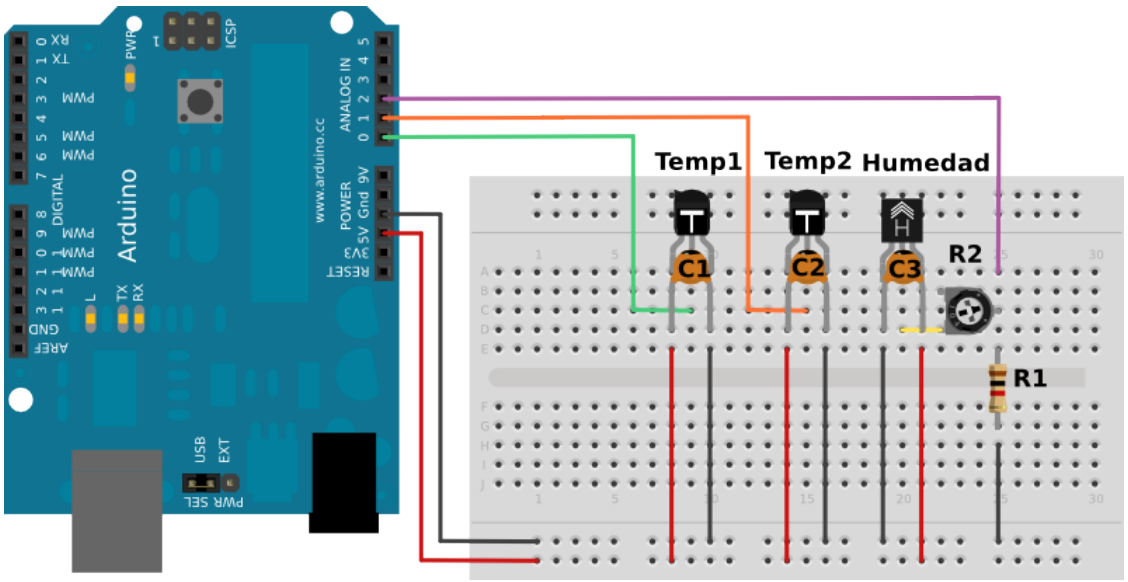

Figura 3: **Conexionado pruebas, versión 1,1V**

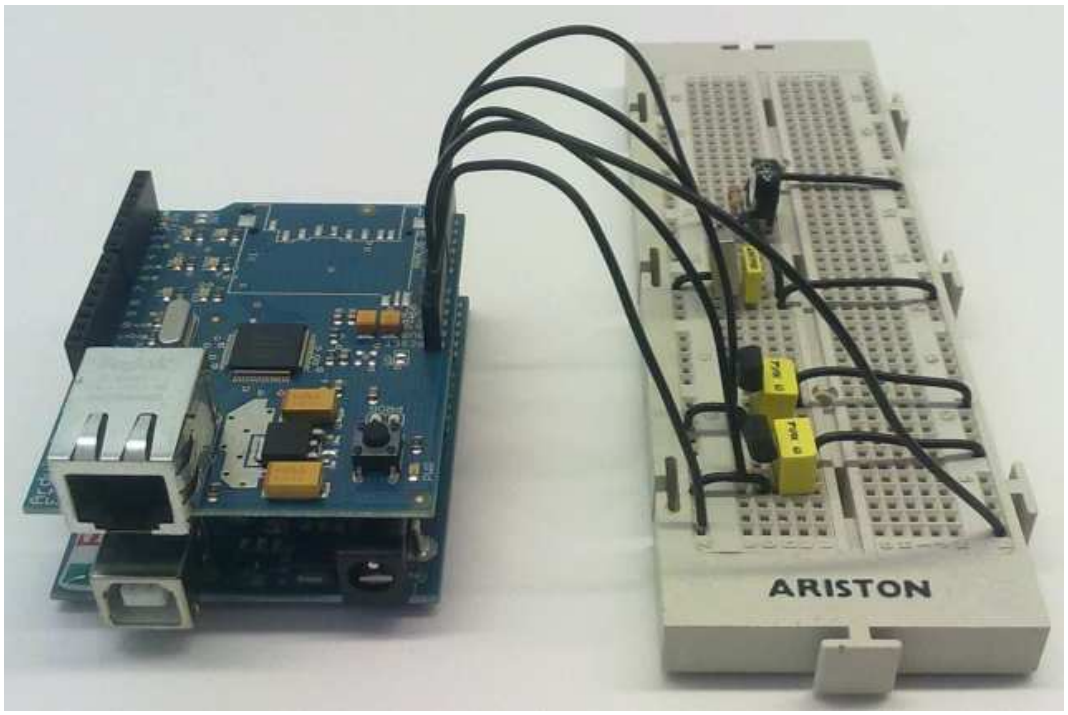

Figura 4: **Pruebas protoboard, versión 1,1V** (Vista frontal)

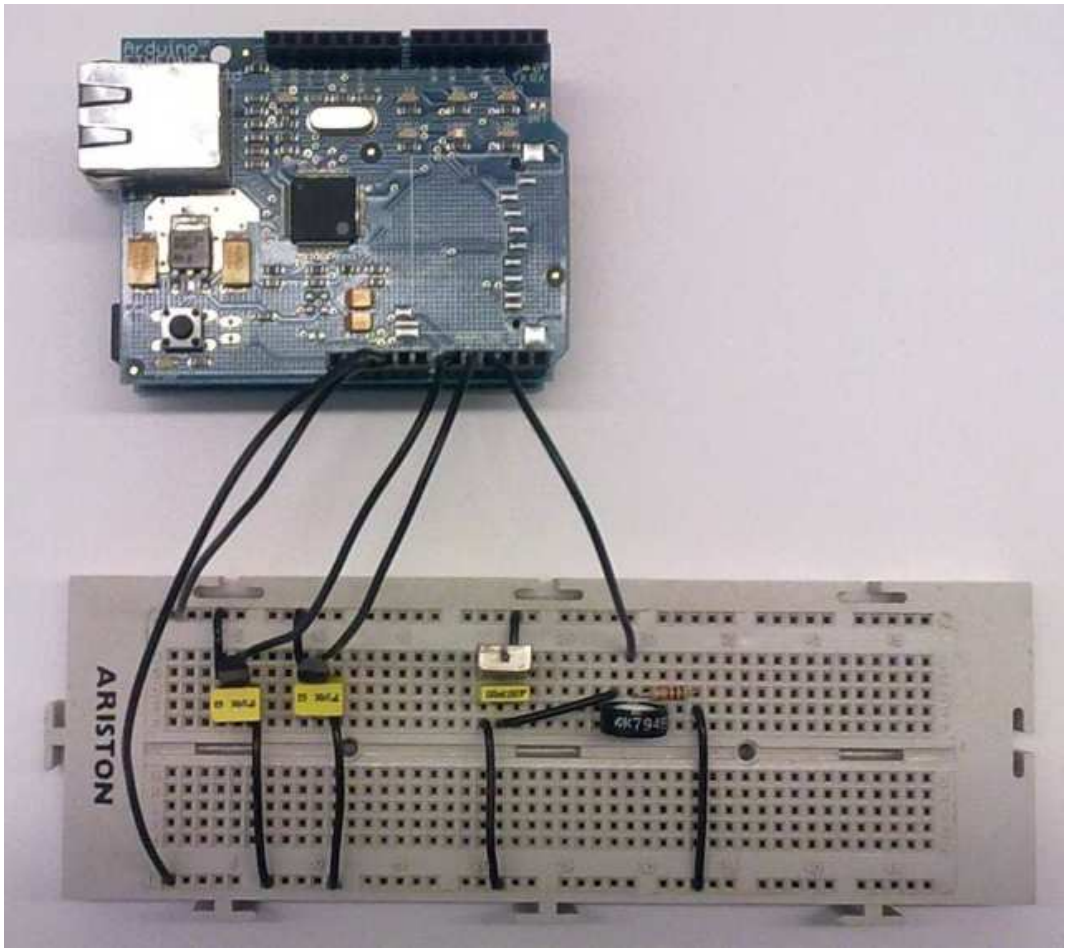

Figura 5: **Pruebas protoboard, versión 1,1V** (Vista superior)

terísticas de los dispositivos. Dado que disponemos de dos tipos distintos de sensores de temperatura, necesitaremos dos ecuaciones diferentes en función del dispositivo utilizado.

11.1. Conversión de la señal de entrada en mV

Como se vio en el apartado 9.2, la señal analógica de entrada se convierte en un nivel digital entre 0 y 1023. Para poder usar este nivel, deberemos volver a convertir este nivel digital en los mV correspondientes. Esta conversión dependerá de la referencia establecida.

En el caso de la referencia a 5V, la conversión se realiza mediante la ecuación:

$$V_{mV} = \frac{Entrada \times 5000}{1024}$$

Para la versión a 1,1V, usamos una ecuación parecida, teniendo en cuenta la nueva referencia:

$$V_{mV} = \frac{Entrada \times 1100}{1024}$$

11.2. Conversión de la señal en mV a los valores reales

Debemos convertir estos valores en mV a sus correspondientes valores físicos (ºC y % de humedad relativa). En todos los casos, debemos obtener las ecuaciones para la conversión de las hojas de características de cada dispositivo.

11.2.1. Sensores de temperatura

Sensores de temperatura En el caso de las sondas de temperatura MCP9700A, vemos que la señal que entregan a 0ºC es de 500mV, así pues al valor recibido en mV desde la entrada analógica le restaremos esta cantidad. Además, el valor que se obtiene habrá que dividrlo por 10, dado que cada grado de temperatura supone un incremento de 10mV en la señal de salida. Por lo tanto, la ecuación es:

$$T_C = \frac{V_{mV} - 500}{10}$$

Con respecto a los sensores LM35, ya están calibrados para grados centígrados. Por lo tanto, la salida en mV es directamente proporcional a la temperatura, y sólo deberemos dividir el valor por 10 (cada grado supone un incremento de 10mV en la salida):

$$T_C = \frac{V_{mV}}{10}$$

Estas ecuaciones son las mismas tanto para la señal de referencia de 5V como para la señal de 1,1V, dado que no estamos realizando ninguna adaptación sobre la salida de los sensores.

11.2.2. Sensor de humedad

En el caso del sesor de humedad 808H5V5, la ecuación sufrirá modificaciones en función de la señal de referencia, dado que en uno de los casos utilizamos un divisor de tensión. En la versión a 5V, comprobando las hojas de características vemos que el sensor entrega 0,8V para un 0 % de HR y 3,9V para el 100 %. Por lo tanto, a la señal medida en mV le restamos 800mV, y para obtener el dato en porcentaje, dividimos el resultado entre 31 ($3900 - 800 = 3100$):

$$\%HR = \frac{V_{mV} - 800}{31,0}$$

Si empleamos la referencia de 1,1V, deberemos hacer los cálculos para adaptar estos valores. Con el divisor de tensión empleado, las señales de salida normales, 0,8V a 3,9V, se convierten en 0,224V a 1,092V. Esto supone que al valor medido en mV deberemos restarle 224mV, y para convertir el resultado a porcentaje, dividimos por 8,68 ($1092 - 224 = 868$):

$$\%HR = \frac{V_{mV} - 224}{8,68}$$

11.3. Código Arduino, versión 1

A pesar de emplear dos referencias, el código utilizado en ambos casos es muy similar, y por lo tanto no vamos a repetir todo el código. A continuación mostramos el código para 5V, y en el mismo se incluye (comentado para evitar su ejecución) el código para la versión a 1,1V.

Con respecto a esta versión a 1,1V, la primera de las variaciones es el cambio de la tensión de referencia con la instrucción *analogReference(INTERNAL)*, y además tendremos que adaptar las ecuaciones empleadas para hacer la conversión de las lecturas, tal y como se explica en las ecuaciones de los apartados anteriores.

Por otra parte, aunque tenemos dos tipos de sensores de temperatura diferentes, en el proyecto sólo se usan dos de manera simultánea. En el código se usa por defecto el sensor MCP9700A, aunque se incluye también el código para el LM35, de nuevo comentado.

En caso de querer cambiar la referencia a 1,1V, o usar el sensor LM35, lo único que hay que hacer es descomentar las líneas adecuadas, y comentar las originales.

En caso de querer utilizar este código, se recomienda descargar los ficheros fuente correspondientes de la web del autor, dado que copiar y pegar el código puede introducir fallos en el mismo.

11.4. Ejecución y resultados

Mediante el entorno de desarrollo de Arduino, se ha compilado el código y se ha subido a la placa, donde comienza a ejecutarse inmediatamente. Si nos conectamos al monitor serie incluido en el entorno, obtenemos las lecturas de los sensores. Tal y como se comenta en el código, esta primera versión tiene algunos problemas que se observan en el resultado:

- Baja estabilidad de las medidas: las lecturas de las tensiones en las entradas analógicas varían mucho entre cada ejecución (esto es especialmente notable en las lecturas del sensor de humedad).
- Diferencia de temperatura entre los dos sensores: cada sensor entrega una temperatura diferente, problema que se puede atribuir al error interno que presenta cada componente[10].

[10]Como se detallará más adelante, este error, cuyo origen real es completamente distinto, provocará una profunda modificación en el proyecto. No obstante la verdadera causa no se detecta hasta

- Resultados sin calibrar: los valores mostrados difieren del valor real de temperatura y humedad en la habitación, dado que no se han calibrado los componentes. item
- En la versión a 5V, debido a la pérdida de precisión, los saltos entre diferentes medidas de los sensores de temperatura son bastante grandes. Dado que el rango del sensor de humedad se adapta mejor a los 5V, no presenta problemas al respecto.

El primero de estos problemas se puede solucionar de forma sencilla en las siguientes versiones del código, mediante el uso de medias.

Los dos problemas referidos a la calibración de los sensores se dejan para una etapa posterior, dado que es posible que el montaje actual (en una placa de prototipado) introduzca ruido en las señales que afecte a las medidas.

Con respecto al problema de precisión en las temperaturas, se soluciona empleando la referencia de 1,1V, por lo tanto se considera resuelto.

Un ejemplo de la salida del programa (en concreto, de la versión a 1,1V usando sensores LM35) se puede observar en la figura 6. Se puede comprobar la elevada variabilidad del sensor de humedad, y la precisión en las medidas: la variación mínima que pueden detectar los sensores de temperatura está en torno a los 0,11ºC, mientras que el salto mínimo observado en la versión de 5V está en torno a los 0,51ºC.

11.5. Código Arduino, versión 2

Con respecto a la siguiente versión del código, se han introducido algunas modificaciones importantes, con el fin de llevar a cabo pruebas de mayor alcance y duración, que permitan evaluar mejor el funcionamiento del sistema:

- Se mejora la estabilidad de las medidas. Para ello, se toman 5 muestras de cada sensor de manera continua, y se obtiene la media de las mismas. Esto suaviza los resultados, y se evita mucho del ruido presente en la anterior versión.
- Dado que se ha comprobado que el sistema funciona, aunque sea de forma rudimentaria, se va a evaluar la adecuación de los sensores. Para ello, el código se modifica de forma que incluya los 4 sensores de temperatura disponibles (2 LM35 y 2 MCP9700A), junto con el sensor de humedad 808H5V5. Esta modificación también implica cambios en el circuito.

más adelante, y se prefiere mostrar la evolución real del proyecto tal y como se ha producido.

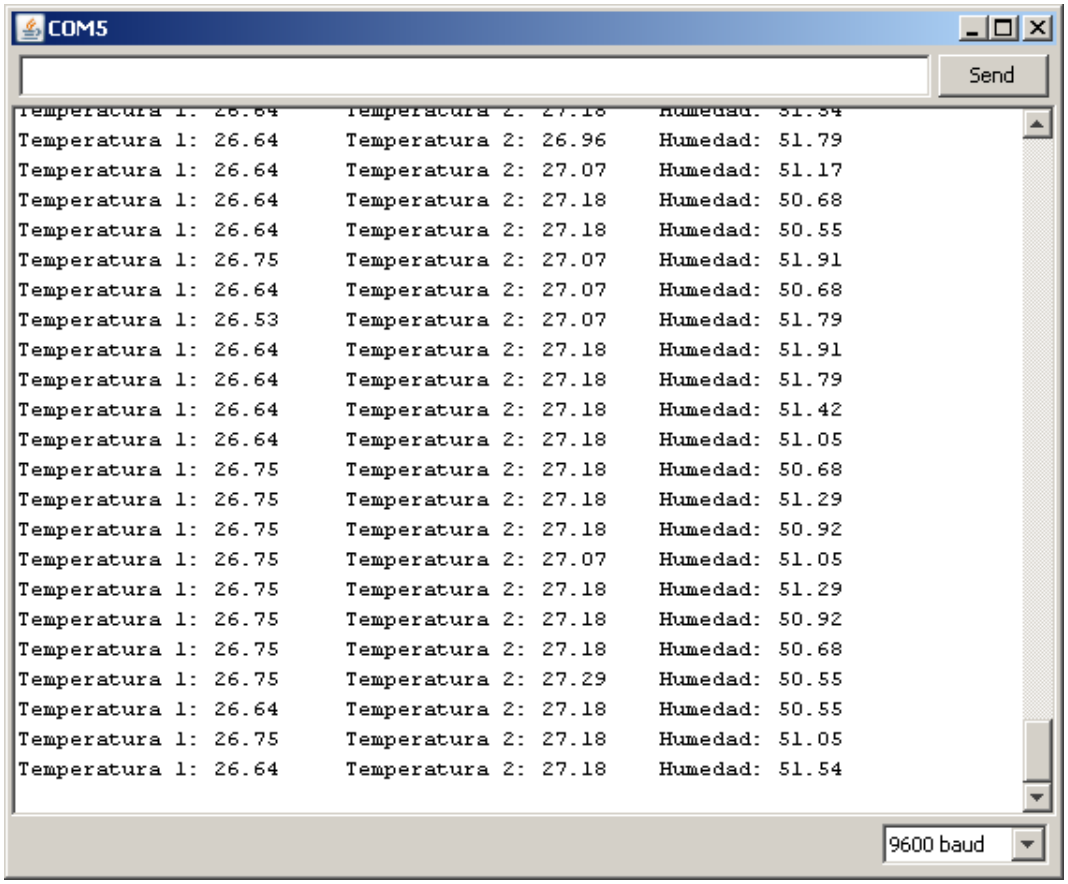

Figura 6: **Ejecución código, versión 1** (Monitor Serie)

De nuevo, y para determinar si el uso de diferentes señales de referencia introduce mejoras en las mediciones, se incluyen en el código las ecuaciones para ambas referencias.

Se debe tener en cuenta que esta versión tiene como finalidad las pruebas del sistema, y que a pesar de emplear los 4 sensores de temperatura, el prototipo final sigue teniendo como objetivo sólo 2 sensores.

11.6. Ejecución y resultados

Como se verá a continuación, estas pruebas han puesto de manifiesto un problema importante en el sistema, y cuya causa se ha tardado bastante en descubrir.

Hasta ahora todas las pruebas que se han realizado han sido con el sistema conectado por USB a la máquina de desarrollo, pero para poder llevar a cabo las pruebas en el entorno final se necesita que el sistema sea completamente independiente, por lo tanto se ha conectado un transformador a la entrada de alimentación de la placa Arduino. Siguiendo las recomendaciones de la página de Arduino, se ha empleado una tensión de 9V para asegurar un correcto funcionamiento.

En estas pruebas se ha comparado la medición de las lecturas con un sensor de temperatura y humedad comercial.

Los datos iniciales al encender el sistema coinciden en gran medida con los datos del sensor comercial, salvo por el ya conocido desajuste entre las mediciones de los sensores de temperatura. Respecto al sensor de humedad, la coincidencia es perfecta entre ambos sistemas, más ahora con la mejora de la estabilidad.

Durante los primeros minutos de funcionamiento las mediciones de temperatura de los sensores LM35 han ido en aumento progresivo, hasta que finalmente se han estabilizado en aproximadamente 2ºC por encima de la temperatura marcada por el otro sensor. Dado que dicha variación es muy elevada, se ha procedido a comprobar la temperatura real con otro termómetro (uno analógico de alcohol), y efectivamente se comprueba que el sensor arduino marca 2ºC por encima de la temperatura real. Es posible que se deba al calentamiento del propio sensor LM35 durante el funcionamiento, aunque el fabricante indica un calentamiento de 0.1ºC, lo que está muy lejos de la variación apreciada.

El aumento de la temperatura se detecta por igual en las dos sondas, mientras que el sensor de humedad no muestra variación sobre la medida. Además, se detecta una elevada variabilidad en las mediciones consecutivas, con saltos de hasta 1 grado en un sólo segundo. Se revisará el sistema de cálculo de las medias.

En vista de los resultados, se prepara una nueva prueba empleando de forma simultánea las sondas LM35 y MCP9700, para comprobar si el problema del aumento de temperatura se produce en todos los sensores (lo que señalaría al código, al propio arduino, o a problemas de interferencia en el ciruito como origen del problema) o sólo en un tipo de sondas.

Con los 4 sensores en el mismo entorno se observa, con una temperatura real en torno a los 25ºC, que ambas sondas LM35 muestran una temperatura superior a las sondas MCP9700, estando las LM35 en torno a los 1,5ºC ó 2ºC por encima, y las MCP entre 0,5ºC y 1,5ºC. Las cuatro sondas muestran alguna variación entre medidas consecutivas, aunque las LM35 son apreciablemente más estables, presentando las MCP saltos bastante bruscos de más de 2ºC.

Trasladamos el sistema a otro entorno donde la temperatura real es más baja (sobre los 18ºC), y aquí las sondas LM35 muestran una temperatura más cercana a la real (entre 0,5ºC y 1ºC de diferencia, siempre por arriba), con una muy elevada estabilidad, mientras que las MCP presentan oscilaciones muy fuertes en mediciones consecutivas, llegándose incluso a observar saltos bruscos de más de 6ºC entre dos medidas consecutivas, lo que resulta imposible de justificar a tenor de las características del componente. Esta elevada inestabilidad impide obtener datos reales sobre las lecturas de las sondas MCP. De nuevo, el sensor de humedad es completamente correcto y estable, aunque esto puede deberse al elevado tiempo que necesita este sensor para adaptarse a un cambio brusco en las condiciones ambientales de humedad (en torno a 15s), lo que suaviza mucho las variaciones de su tensión de salida. Como dato, indicar que este último entorno de pruebas contiene una elevada cantidad de equipo informático y electrónico, lo que puede suponer la existencia de importantes interferencias electromagnéticas.

Dado que la única diferencia entre ambas pruebas es el entorno, este comportamiento tan variable hace pensar en posibles problemas de interferencia, que pueden ser debidos al montaje del circuito (las protoboard suelen dar problemas de ruido) o incluso a la alimentación externa utilizada. Se van a llevar a cabo otras pruebas para intentar localizar el origen exacto del problema, probando con cables apantallados, preparando el montaje en placa final de los componentes, e intentando ajustar la calibración de los componentes tal y como indica el fabricante.

11.7. Problemas con los sensores de temperatura

El proceso de pruebas que se ha llevado a cabo ha permitido determinar el problema en las mediciones. Con los 4 sensores de temperatura conectados, se observa que los problemas de incremento de las temperaturas se dan en todas las sondas por igual. Tras consultar a fondo las hojas de características y diversos sitios en

internet, se descubre que el problema radica en la interacción entre el conversor analógico-digital de arduino y los sensores de temperatura.

11.7.1. Multiplexación en el sensor

El procesador Atmega 328 que incorpora el arduino incluye un único conversor ADC, el cual se multiplexa para permitir la lectura de las 7 entradas analógicas. Cambiar entre dos entradas hace que en la práctica, la señal se corte en la entrada original y se establezca en la nueva entrada seleccionada. En la mayoría de componentes (por ejemplo, en potenciómetros) esto no tiene ninguna repercusión en la señal, pero en el caso de los sensores de temperatura, se produce un problema relacionado con la arquitectura de los sensores.

Los circuitos que incluyen internamente los sensores de temperatura presentan, de cara a la interconexión con otros componentes, una muy alta impedancia y se comportan en algunos aspectos como un condensador de elevada capacidad. Esto provoca que al cambiar muy rápido a la entrada analógica de uno de ellos no de tiempo a estabilizar la señal, y por lo tanto las medidas salen falseadas. Dado que el sensor de humedad no se comporta de esta forma, no presenta el problema.

En los foros de soporte de arduino se plantean algunas soluciones para este problema, y la más extendida es la comentada en el tutorial de sensores de Ladyada (`http://www.ladyada.net/learn/sensors/tmp36.html`), donde se mencionan posibles problemas al conectar múltiples sensores y su solucion (`http://www.adafruit.com/blog/2010/01/29/how-to-multiplex-analog-readings-what-can-go-wrong-with-high-impedance-sensors-and-how-to-fix-it/`), la cual se basa en pausar las lecturas entre las entradas analógicas, haciendo el siguiente proceso:

1. Cambiar a la entrada que se desea leer, mediante `analogRead(entrada)`, sin asignarla a ninguna variable.
2. Esperar algunos segundos, para permitir que se estabilice la señal.
3. Hacer la lectura de la señal: `variable=analogRead(entrada)`.

Este proceso se debe repetir para cada puerto de entrada analógico que vayamos a utilizar. Es importante tener en cuenta que esta pausa afecta al cambio de un puerto a otro, pero no es necesaria para realizar múltiples lecturas sucesivas de un mismo puerto, mecanismo utilizado para obtener una lectura más estable usando la media de varias lecturas consecutivas.

Poniendo en práctica este sistema se han probado los 4 sensores de temperatura disponibles, haciendo sucesivas pruebas con diferentes tiempos de descanso entre los

cambios de entrada. De los 20ms de espera en la configuración original, se han ido incrementando poco a poco las pausas, hasta llegar a los 120 segundos en el caso máximo.

Se comprueba que el método logra mejorar las lecturas de los sensores de temperatura, y los valores en exceso medidos con anterioridad (2º de media) se han reducido, aunque dicha disminución ha sido muy pequeña, en torno a medio grado en el mejor de los casos. Como contrapartida, los tiempos de lectura conjuntos de todos los sensores superan los 10 minutos, de tal forma que hacen al sistema muy poco práctico, impidiendo de hecho la consulta de la temperatura en tiempo real.

Es posible que utilizar este método junto a la calibración de los sensores permita obtener mediciones correctas, pero de cara a utilizar el sistema en entornos grandes, donde se necesite preparar y calibrar muchos sensores, el tiempo de preparación puede resultar demasiado elevado, y aumentar en exceso la complejidad de la solución buscada.

A la vista de estos resultados, se determina que encontrar una solución adecuada se opta por replantear el sistema de medición de la temperatura

12. Versión 3: sensores digitales

Debido a los resultados poco satisfactorios con los sensores de temperatura analógicos utilizados hasta ahora, se ha optado por eliminar los sensores analógicos de temperatura y comenzar a usar sensores digitales. Con estos cambios, se ha logrado por fin un sistema estable y apto para su uso, y se ha dado por concluida la parte de desarrollo del proyecto.

12.1. Sensores DS18B20

Se inicia un nuevo proceso de investigación de cara a evaluar las diferentes opciones de componentes en el mercado, centrando la búsqueda en los dispositivos completamente digitales.

Durante el mismo, se encuentran los sensores digitales de la casa Maxim (`http://www.maxim-ic.com/`, antigua Dallas Semiconductor), mencionados en los propios foros de Arduino. En concreto se trata de la familia de componentes digitales 1-Wire (`http://www.maxim-ic.com/products/1-wire/`), que incluye, entre otros, diversos sensores de temperatura.

Tras analizar sus características, se determina que son la mejor opción disponible para el proyecto, dado que presentan diversas ventajas:

- Listos para usar: no requieren ningún tipo de calibración.
- Amplia disponibilidad asegurada en el mercado.
- Permiten expandir el nº de sensores de manera mucho más simple que con los sensores analógicos.

La mencionada ventaja de la expansión viene como consecuencia del sistema de conexión que usan estos sensores. El bus 1-Wire permite la conexión de hasta 64 componentes de esta familia en un mismo canal digital de datos, usando por lo tanto una única entrada digital. Dentro de este canal, la identificación de los dispositivos individuales se realiza mediante la consulta de un número de serie único para cada dispositivo que se establece en fábrica.

Además, los sistemas 1-Wire permiten alimentarse de dos formas: de manera directa con 3 cables (un cable para tensión, otro para tierra y otro para datos) o en el denominado modo parásito, donde el cable de datos lleva la propia alimentación, produciendo que en la práctica sólo se usen 2 cables para conectar el dispositivo, aunque el uso de esta configuración implica un retardo adicional de 750ms en cada lectura.

En concreto, de todos los sensores disponibles en la mencionada familia, se ha optado por utilizar los circuitos DS18B20. Estos sensores presentan las siguientes características:

- Rango de temperatura de -55 a 125 ºC
- Precisión seleccionable de 9 a 12 bits
- Salida seleccionable en grados centígrados o farenheit
- Alimentación de 3 a 5,5V
- Precision media de ±0,5 ºC
- Tiempo de conversión de 750ms (caso peor máximo, a 12 bits)

Como contrapartida estos circuitos presentan, por unidad, un coste algo mayor que los analógicos, en torno a 5€por unidad, aunque la compra en grandes cantidades reduce la diferencia (en internet se pueden adquirir packs de 30 unidades con un coste por circuito de menos de 1€). Para el nuevo montaje, se han adquirido dos unidades.

12.2. Nuevo diseño del circuito y el código

El uso de estos sensores ha implicado cambios en el circuito. Los sensores de temperatura se conectan ahora a la entrada digital (se ha utilizado la número 2) y necesitan de su correspondiente alimentación. Con respecto a esta, empleamos el modo directo (3 cables), dado que acelera la velocidad de las lecturas frente al modo parásito. Se han eliminado además los condensadores asociados a los sensores de temperatura analógicos, dado que no tienen utilidad en estos dispositivos digitales.

Como resultado del cambio de circuitos, se ha podido llevar a cabo otra modificación importante en el diseño. Dado que en las señales analógicas ya sólo tenemos el sensor de temperatura, el uso de la señal de referencia de 1,1V (y el consiguiente divisor de tensión para la humedad) ya no resulta útil, y se ha podido establecer de nuevo la referencia en 5V y conectar directamente el sensor 808H5V5 a la señal analógica. Esto elimina los posibles fallos asociados al ajuste del potenciómetro.

En lo relativo al código, existe una completa librería genérica para utilizar los dispositivos 1-Wire en arduino, y otra más específica para utilizar los DS18B20. Esto ha permitido que los primeros pasos con los nuevos sensores hayan sido muy sencillos, con resultados francamente positivos. Se han obtenido lecturas de muy alta precisión y estabilidad desde el primer momento. Como es lógico, el código ha sufrido algunos cambios, pero la existencia de las mencionadas librerías para arduino ha permitido realizar las modificaciones en muy poco tiempo. Entre otros cambios, cabe destacar que estos sensores entregan una medida absoluta y de total precisión en cada consulta a los mismos, y por lo tanto no es necesario establecer mecanismos para convertir la medición en grados, ni calcular la media de varias medidas para estabilizar la señal.

12.3. Nuevas pruebas y montaje final

Con el sistema modificado, se han llevado a cabo más pruebas para comprobar la validez del montaje. Se ha comprobado la estabilidad y corrección de las medidas, comparando los datos con otros termómetros que han permitido comprobar la veracidad de las mediciones. Una vez terminados los test, se ha procedido a trasladar los componentes desde la placa de prototipado a la placa de circuito final, con los componentes soldados, dando al sistema un aspecto mucho más acabado y profesional, así como mayor robustez.

12.4. Comunicación ethernet y almacenamiento de registros

Una vez que se ha probado el sistema completo, se procede a implementar el código para la conexión mediante cable ethernet en el propio arduino.

La comunicación ethernet por HTTP permite dos cosas:

- Acceder por HTTP al propio arduino, para comprobar en tiempo real la temperatura y humedad existentes[11].
- Registrar en un servidor remoto dichos valores de forma periódica.

Mediante una consulta HTTP por el mecanismo GET, el arduino lanza al servidor una cadena con toda la información de las mediciones actuales de temperatura y humedad. El servidor recibe la cadena, la procesa mediante un script en PHP, y da a los datos el formato adecuado para su almacenamiento.

Los datos recogidos se vuelcan en una base de datos MySQL. Esta información, una vez almacenada, puede ser consultada a través del mismo servidor usando gráficas horarias, que permiten analizar con comodidad la evolución de la temperatura y humedad a lo largo de un día, así como los valores medios. El sistema permite seleccionar el sensor y la fecha de consulta, y genera un gráfico interactivo en flash con los datos almacenados. Este gráfico se genera mediante la librería libre Open Flash Charts V2 (`http://teethgrinder.co.uk/open-flash-chart-2/`).

12.4.1. Sincronización NTP

Entre los datos enviados por el sensor al servidor, se encuentran la fecha y la hora de la lectura. Estos datos los obtiene el sensor mediante la sincronización con un servidor de tiempo NTP.

Al emplear este sistema, se garantiza que las lecturas han sido tomadas a intervalos completamente regulares y precisos, y se facilita la comparación de registros entre fechas.

Además, este sistema evita problemas con los cambios de horario de verano, dado que se emplea la hora UTC (universal), y no la hora local de un país concreto. De esta forma, durante los cambios de hora se evita que aparezcan dos registros con la misma hora. Será el propio servidor Web el que haga los cálculos de conversión

[11] A pesar de la utilidad de esta funcionalidad, presenta un inconveniente que se explica en 12.6.2, y que se debe tener en cuenta en caso de querer utilizarla.

de hora necesarios para mostrar la hora local, mediante el uso de las funciones de conversión de fecha y hora presentes en MySQL.

12.5. Pruebas de larga duración

Una vez listo el montaje final y el código para las comunicaciones, se procede a llevar a cabo una prueba de larga duración, con el sistema funcionando durante varios días de forma continuada. De cara a permitir el uso independiente del sistema durante estas pruebas, se ha conectando un módulo POE[12] para alimentar el conjunto. De esta forma, el sistema se puede desplegar en cualquier punto con sólo tirar un cable de red, por donde viajan tanto los datos como la corriente necesaria para el funcionamiento.

Con el montaje definitivo, el prototipo se ha puesto en pruebas reales en un entorno de producción. Para ello, aprovechando la existencia de un departamento de archivo en el centro de trabajo, donde se efectúa un registro periódico de temperatura y humedad, se ha ubicado el sistema completo junto al sensor en uso actualmente, y se ha dejado funcionando varios días recopilando datos. A la hora de escribir esto, el sistema lleva funcionando 15 días de forma continuada.

12.6. Resultados de las pruebas

Durante el primer día se detectaron 2 problemas, ambos asociados al módulo ethernet.

12.6.1. Problemas de inicialización ethernet

La versión de la placa ethernet usada en el montaje del prototipo tiene un problema con el sistema de inicialización, que obliga a pulsar el botón de reinicio de la placa una vez que se ha alimentado. Hasta que este proceso no se lleva a cabo, el módulo es incapaz de establecer una conexión ethernet correctamente. Esto supone que ante un corte de corriente, el sistema se inicia y comienza a tomar datos de temperatura y humedad, pero no es capaz de comunicar los mismos por red, con lo cual pierde toda la utilidad. No obstante, este problema, que esta localizado y documentado en la web de arduino, se ha solucionado mediante la conexión de un condensador entre las señales de tierra y reset de la placa de prototipado. Además, en la actual versión de la placa ethernet, que ya está disponible en las tiendas, la

[12]Power Over Ethernet

inicialización del módulo se produce correctamente de manera automática. Por lo tanto se recomienda evitar la adquisición de las placas antiguas que aún quedan en stock. Para diferenciarlas, las nuevas placas corregidas llevan una ranura para tarjetas microSD[13] que no estaba presente en la versión anterior (lo cual supone además un valor añadido para la placa).

12.6.2. Bloqueos del sistema

Al hacer numerosos accesos al servidor web integrado para comprobar el correcto funcionamiento del sistema, se produce un bloqueo del sistema, debido al agotamiento de los recursos de arduino. Esto se debe a que la librería ethernet empleada no gestiona las peticiones de los clientes con la suficiente rapidez, y ante varias solicitudes simultáneas es incapaz de liberar los recursos a tiempo. Se ha encontrado una librería ethernet no oficial que soluciona esto acelerando la respuesta a las peticiones, pero aún no se ha probado en el prototipo, dado que no se ha logrado hacer funcionar en conjunto con las otras librerías empleadas.

Dado que una vez montado el sistema de registro en BBDD no ha sido necesario volver a consultar directamente al servidor integrado en arduino, el bloqueo no ha vuelto a producirse. En caso necesario, se puede probar con la librería no oficial, pero quizás la mejor opción pasa por eliminar el servidor integrado y usar sólo el sistema de almacenamiento y consulta en BBDD.

12.6.3. Otros problemas detectados

Aparte de estos dos problemas mencionados, los únicos fallos detectados han sido la pérdida de 10 registros por cortes en la red local que une el sistema con el equipo que almacena la base de datos (problemas completamente ajenos al sistema). Teniendo en cuenta que lleva más de 4500 registros seguidos, el número es bastante positivo.

12.6.4. Calidad de las medidas

Los resultados son altamente satisfactorios. La medición de la temperatura es completamente estable, y de una precisión tan elevada que permite identificar cuando hay personas en la sala (dato observable en las pequeñas oscilaciones que se producen durante las horas de trabajo). Las mediciones coinciden completamente

[13]Cuidado: una versión aún mas antigua tiene una ranura para tarjetas SD, no microSD, y también se debe evitar.

con las del sensor en uso actualmente, y presentan la ventaja del registro continuo, lo que permite analizar la evolución por las noches y los fines de semana, momentos en los cuales hasta ahora no se realizaban anotaciones.

Y en referencia al sensor de humedad, la extremada sensibilidad y velocidad del mismo han permitido rastrear eventos completamente desconocidos de la sala. Se detectaron unas fuertes variaciones de humedad registradas en intervalos muy pequeños de tiempo, las cuales se achacaron inicialmente a un mal funcionamiento o montaje del sensor, pero a lo largo de los días los patrones observados se repitieron con bastante exactitud, como por ejemplo, una caída brusca de la humedad a una hora concreta de la mañana.

Al observar que dichos patrones desaparecían durante los fines de semana, se comenzó a estudiar con detalle los mismos para asociarlos a actividad del personal de la sala. Se ha logrado comprobar que el sistema registra de manera inmediata cuando se abren y cierran las puertas de la sala, e incluso se han podido determinar los momentos de entrada del vigilante durante las rondas nocturnas. Se ha observado que el mero hecho de permanecer cerca del sensor unos instantes altera la medición de la humedad, lo que da una muestra de la elevada sensibilidad del mismo. Y aunque este hecho permite mantener un registro de enorme exactitud, debe tenerse en cuenta de cara a escoger la adecuada ubicación del sistema, dado que la interacción accidental con el mismo podría introducir variaciones no deseadas en las medidas y afectar a su validez.

13. Conclusiones de la etapa de desarrollo

Los resultados y mediciones obtenidos superan con creces las espectativas buscadas, y la funcionalidad del sistema de consulta gráfica aporta una información muy valiosa, que supera ampliamente a la ofrecida por el sistema en uso actualmente.

Debido a estos satisfactorios resultados, se considera que la parte de desarrollo del proyecto ha terminado. Dado que el prototipo aporta una funcionalidad muy superior al sistema que existía previamente en la ubicación de pruebas, se ha decidido mantenerlo en uso de manera indefinida, e incluso se valora la posibilidad de construir una segunda unidad para instalarla de manera definitiva en dicha ubicación.

Además, una vez visto que el sistema tiene posibilidades reales de uso, se ha diseñado una placa de circuitos propia para la conexión de los componentes, aunque debido a los costes y tiempos aún no se ha podido fabricar. No obstante, el diseño de la misma se añade a la documentación del desarrollo final.

Parte V

Desarrollo final

La presente parte del documento detalla la construcción completa de un sensor plenamente funcional, tanto a nivel de hardware como de software.

13.1. Advertencia de seguridad

Antes de comenzar con la descripción del sistema final, se debe hacer una advertencia sobre algunos aspectos del mismo:

- Inyecciones SQL: el sistema no comprueba ni filtra en modo alguno los datos recibidos, ya sean desde el sensor, o los que introduzca el usuario. Esto es un problema de seguridad, dado que abre la puerta a posibles inyecciones SQL en la base de datos, con el riesgo que esto supone, incluyendo pérdida de datos (ya sean del registro de los sensores, o de otros datos almacenados en el mismo servidor), o escalada de privilegios en el sistema anfitrión.

  Dado que la base de datos emplea unos permisos concretos para el trabajo con los registros, el máximo riesgo teórico es la inserción o modificación indebida de los registros almacenados en la propia base de datos. No obstante, es necesario tener esto en cuenta de cara a su utilización en entornos sensibles a ataques.

- Uso abierto en Internet: debido al punto antes comentado, se desaconseja completamente el uso del sistema en servidores accesibles al público desde Internet. No entra dentro del objetivo del proyecto el proporcionar al código de las mínimas medidas de seguridad necesarias para este uso, y sería necesaria una importante modificación del código Web para permitir un uso seguro en entornos abiertos.

- Falsificación de medidas: dado que ni el sensor ni el servidor llevan a cabo validación de los datos recibidos (ni del origen de los mismos), resulta muy sencillo falsificar datos. Por lo tanto, el sistema en su estado actual no está indicado para aquellos entornos donde la veracidad de los datos almacenados resulte vital.

Por todos estos motivos, no puedo dar garantía alguna sobre el sistema, incluyendo su correcto funcionamiento, ni posibles problemas de seguridad que pueda

implicar su uso. Siguiendo la filosofía de la mayoría del software existente, el sistema se entrega "tal cual", y no puedo asegurar su funcionamiento libre de errores, ni las posibles consecuencias de su utilización [14].

14. Prototipo final

Este apartado detalla los diferentes aspectos del prototipo final weatherlan, que incluyen:

- Hardware
  - Plataforma Arduino con las extensiones para conectividad Ethernet.
  - Sensores y materiales adicionales necesarios para completar el sensor.
- Software
  - Entorno de desarrollo Arduino, con el código utilizado en el sensor para capturar y registrar los datos de los sensores.
  - Servidor de registros bajo Debian GNU/Linux[15]
    - Servidor MySQL, para el almacenamiento de los datos de registro.
    - Servidor Web, implementado mediante HTML y PHP, que gestiona la consulta y el registro de los datos.
    - Servidor NTP, para la sincronización horaria de los sensores.

15. Hardware

15.1. Componentes

Una vez completado el prototipo del proyecto, los siguientes componentes han sido utilizados:

- 1 placa *Arduino Duemilanove*

[14] No obstante, funciona, y dudo mucho que llegue a obligarte a formatear tu ordenador.

[15] Es posible emplear cualquier otro sistema operativo y servidor (siempre que admita PHP y MySQL), pero por razones lógicas, dado el objetivo de precio y el uso de licencias libres, se considera razonable emplear GNU/Linux.

- 1 módulo *Arduino Ethernet Shield*
- 1 placa de prototipado *Arduino Protoshield*
- 2 sensores de temperatura *Maxim DS18B20*
- 1 sensor de humedad *808H5V5*
- 1 resistencia 4,7kΩ
- 1 condensador cerámico de 47nF[16]
- 3 cables de Audio-CD MPC-2, para la extensión de los sensores desde la protoshield Cualquier cable de 3 hilos es adecuado para la conexión, pero he usado el MPC-2 (que es de 4 hilos, y he desechado uno) simplemente porque tenía muchos sin usar a mano.
- Hilo de conexión de 0,28mm, para la interconexión de componentes en la *protoshield*
- Barra de contactos de tamaño estandar (1"), para la conexion de la *protoshield* a la placa ethernet, y la conexiones de los conectores MPC-2 a la *protoshield*
- Funda termoretractil para la protección de los contactos de los sensores

Los aspectos relevantes de los principales componentes se detallan en la siguiente sección.

Adicionalmente, para alimentar el sistema durante las pruebas finales de forma remota e independiente, se ha utilizado un inyector de corriente *POE*[17] de la marca *D-Link*, concretamente el modelo *DWL-P200*. No obstante, la placa Arduino puede alimentarse de manera sencilla desde muy diversos orígenes, tal y como se detalla en su página Web.

Como herramientas, se han usado:

- Soldador de estaño para la soldadura de los componentes electrónicos.
- Estaño fino, con el fundente ya incorporado para facilitar el soldado.
- Alicates de corte para rematar las patillas de los componentes y cables una vez finalizados.
- Multímetro digital, para comprobar las conexiones de los circuitos.

[16]Sólo necesario en caso de emplear una antigua Ethernet Shield. Las versiones actuales (Ethernet Shield V2) no lo necesitan, y de hecho no se incluye en el esquema final del circuito, ni en el diseño final de la placa PCB. Todas las placas nuevas llevan incorporada una ranura para tarjetas SD, mientras que las viejas (afectadas por el problema) no la traen.

[17]Power Over Ethernet

15.1.1. Arduino Duemilanove

Como ya se ha comentado en la parte correspondiente del anteproyecto, la placa Arduino Duemilanove presenta 14 entradas/salidas digitales y 6 entradas/salidas analógicas, así como diversas conexiones para alimentación y control.

De estas posibles entradas y salidas, se ha usado el puerto analógico 0 como entrada para el sensor de humedad y el puerto digital 2 como bus de entrada y salida 1-Wire para la conexión con los sensores digitales de temperatura.

La placa dispone de un conector USB usado en la programación y de un conector de alimentación eléctrica que admite un elevado rango de voltajes de entrada, con un valor recomendado entre 7 y 12V. Durante la programación se usa la alimentación directamente desde el puerto USB, una vez completado el proceso se utiliza la entrada de alimentación utilizando el módulo POE descrito en el apartado 15.1, configurado para funcionar a 12v.

En la placa existe igualmente un botón de reset, que permite hacer un reinicio en caliente del sistema sin tener que desenchufarlo.

Como detalle, se debe tener en cuenta que al utilizar la conexión USB, cada vez que la unidad se conecta al equipo de desarrollo, esta se reinicia de manera automática.

15.1.2. Arduino Ethernet Shield

La placa Ethernet se conecta sobre la placa Arduino, y permite seguir accediendo a todos los conectores de la misma sin interferencias.

Esta placa dispone de un conector RJ-45 que permite conectividad ethernet 10/100Mb. Existe una librería para usarla de manera sencilla con Arduino, que implementa, entre otras funciones, una pila TCP/IP, así como un cliente y servidor Web. Igualmente, la placa dispone de varios LED que permiten ver el estado de la conectividad, así como un botón que permite resetear de manera simultánea la placa Arduino y el módulo Ethernet.

Hay que indicar que la placa Ethernet usa de manera exclusiva algunos puertos digitales de Arduino, los cuales ya no pueden usarse para entrada/salida digital de manera directa (aunque siempre es posible desconectar la placa mediante software y usar los puertos, y luego volver al conectar la placa). En concreto, los puertos 10 a 13 quedan ocupados una vez que la conexión ha sido inicializada.

15.1.3. Arduino Protoshield

La placa presenta conexiones directas para todas las señales de Arduino, tanto analógicas como digitales y de alimentación, junto con un pequeño espacio de placa perforada donde se lleva a cabo la conexión de los dispositivos.

15.1.4. Sensor DS18B20

Se han utilizado dos unidades de este sensor de temperatura, conectadas mediante alimentación directa (3 cables) y usando una de las entradas digitales de Arduino.

15.1.5. Sensor 808H5V5

El sensor de humedad (1 unidad) se conecta a una entrada analógica de Arduino. Tal y como se ha descrito anteriormente, la entrada analógica utilizada es la 0.

15.2. Montaje de los sensores

Dado el escaso número de elementos, el montaje de los componentes en la placa protoshield es bastante sencillo.

Los sensores digitales de temperatura se han conectado usando las instrucciones del fabricante, habiéndose escogido la alimentación directa del circuito para acelerar la respuesta de los mismos. Se ha conectado el pin de alimentación VCC a la salida de 5V disponible en la protoshield, y la conexión GND a la tierra. Con respecto al conector de datos 1-Wire, la salida de ambos circuitos se ha unido al puerto digital 2 de Arduino, insertando una resistencia en serie entre dicho puerto y la alimentación VCC del circuito, tal y como se especifica en las instrucciones de conexión del fabricante . Esto permite utilizar una única entrada digital para gestionar ambas sondas de temperatura.

El sensor de humedad se conecta igualmente a la alimentación y tierra, y su salida analógica se ha conectado al puerto analógico 0 de Arduino de manera directa.

El esquema de la conexión en protoboard se puede encontrar en la figura 11.

Igualmente, se incluye el esquema eléctrico del circuito en la figura 12.

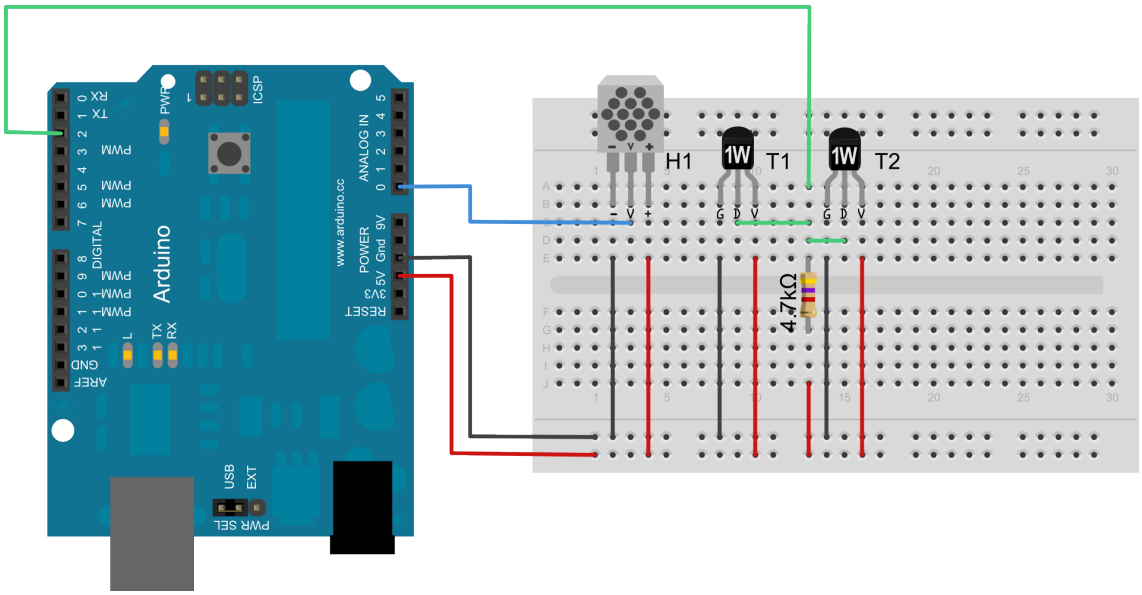

Figura 7: **Protoboard: Conexionado final componentes**

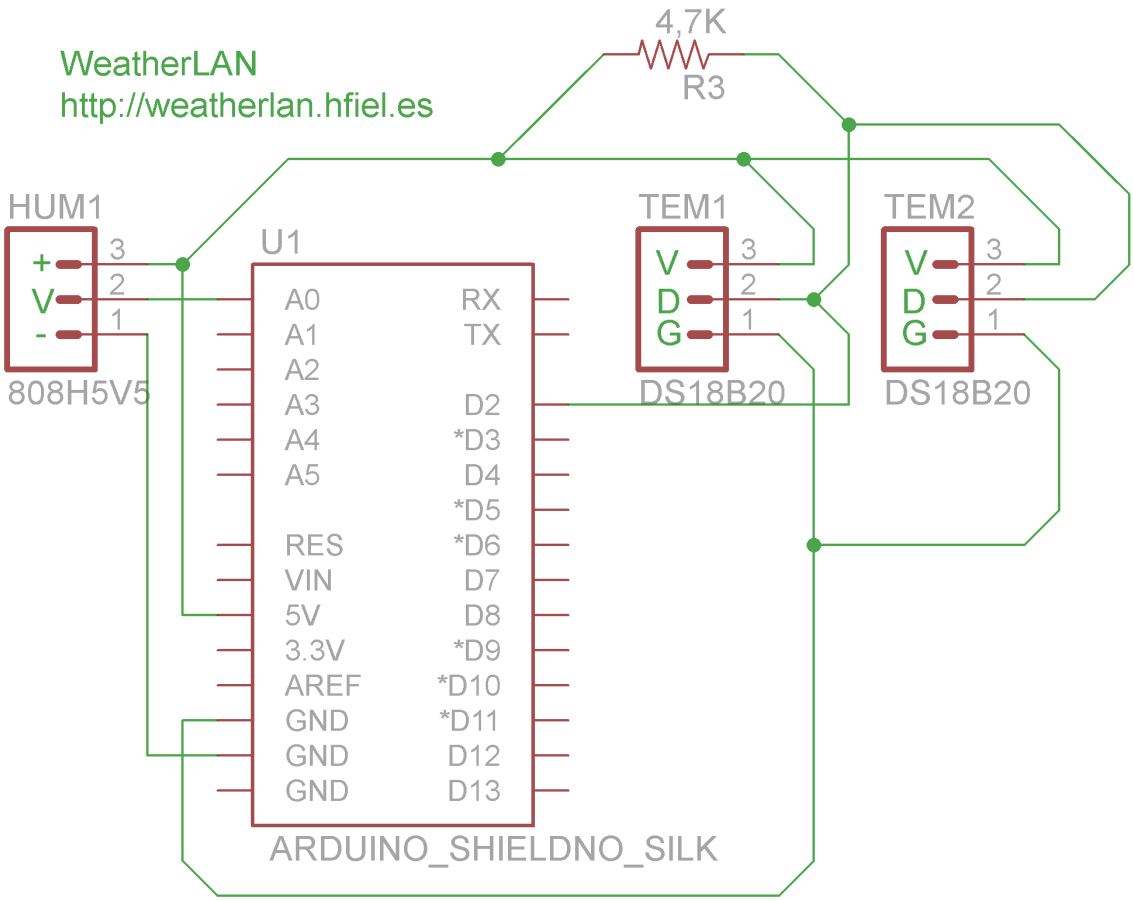

Figura 8: **Esquema eléctrico: Conexionado final componentes**

El fabricante del sensor de humedad indica la posibilidad conectar un condensador cerámico de 100nF entre los pines de alimentación y tierra para reducir el ruido inducido por líneas de alimentación no estables. Esta práctica es común ante el uso de todos los sensores de naturaleza analógica. Aunque en las pruebas realizadas no se ha observado ruido apreciable, y por lo tanto no ha sido necesario emplear ningún condensador, es posible que sea necesario en otros entornos.

Se ha usado la barra de contactos de 1"para conectar la protoshield encima de la placa Ethernet, y además se ha utilizado para permitir la conexión de los sensores sin tener que soldar los mismos a la placa, pensando en un futuros reemplazos de las piezas por cuestiones de mantenimiento, especialmente en el caso del sensor de humedad.

En este aspecto, cada sensor se ha conectado a la placa mediante los cables MPC-2. Para ello, se ha cortado uno de los extremos de cada cable y se ha soldado a las patillas de los sensores. El otro extremo se conecta a los de la barra de contactos que se incorporan a la protoshield.

Este montaje se ha pasado posteriormente a la placa protoshield para arduino, respetando las mismas conexiones entre componentes.

15.3. Diseño placa PCB

Aunque el uso de la placa protoshield resulta adecuado para nuestra unidad de pruebas, de cara a la producción en mayores cantidades su uso es altamente inadecuado, en comparación con una placa PCB, debido a las siguientes razones:

1. Coste: cada placa protoshield tiene un coste bajo (alrededor de 3€por unidad), pero cuando se usan varias el coste conjunto resulta alto. El coste de una sóla placa PCB es mucho más alto, pero por lo general las placas PCB a medida sólo se pueden solicitar en cantidades elevadas, lo que proporciona un coste final por placa muy reducido (hablamos de algunas decenas de céntimos por unidad).
2. Tiempo: soldar los componentes en la placa protoshield es laborioso, dado que requiere unirlos mediante cables a medida. Además, es relativamente sencillo cometer un error en la soldadura. Usar una placa PCB elimina prácticamente la posibilidad de errores, y acelera el tiempo de montaje de cada unidad.
3. Calidad de las señales: el uso de cables, como en el caso de la protoshield, puede llegar a inducir ruido en las señales analógicas. De cara a dar robustez general al sistema, y particularmente ante su utilización con elevado ruido electromagnético, se hace aconsejable la utilización de la placa PCB.

4. Resultado final: el aspecto de la placa protoshield es el de un prototipo. Adecuado para pruebas y primeras versiones, pero no válido para ofrecer un producto final. El resultado en placa PCB es mucho más profesional, y resulta casi obligatorio de cara a presentar el sistema a potenciales clientes.

Dado que este proyecto pretende la construcción de un prototipo, no se ha fabricado la placa PCB, debido exclusivamente a razones económicas (el precio de la única unidad necesaria resultaría demasiado elevado para justificar su fabricación). No obstante, sí se ha diseñado la placa, para que pueda ser usada en aquellas situaciones que lo requieran.

Es importante indicar que, por razones de diseño de la PCB, se han empleado 2 de los terminales de tierra (GND) existentes en los pines de conexión de Arduino. Estas dos conexiones no aparecen en el esquema de protoboard (figura 11), pero sí aparecen en el resto de esquemas. A nivel práctico, es indistinto conectar la línea de tierra a cualquiera de los pines GND de la plataforma, pero la disponibilidad de diversos pines facilita la creación de placas PCB sin cruces entre las pistas.

En caso de desear utilizar el diseño, se deben utilizar los ficheros disponibles en la web de este proyecto en lugar de las imágenes que aparecen a continuación, para asegurar las dimensiones adecuadas de la placa. Se incluyen las imágenes para la cara de pistas (figura 13), y para la cara de textos (figura 14).

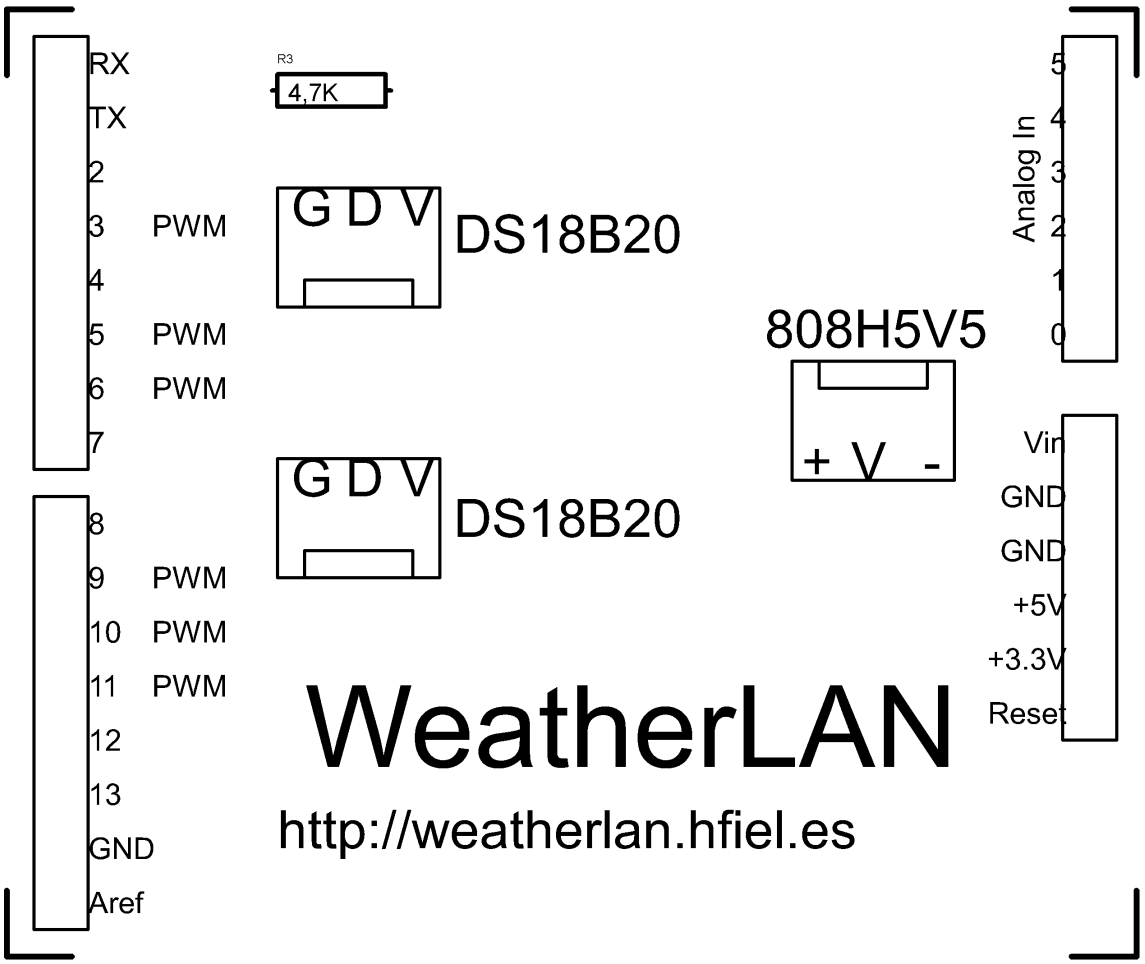

Figura 9: **Placa PCB, cara de serigrafiado**

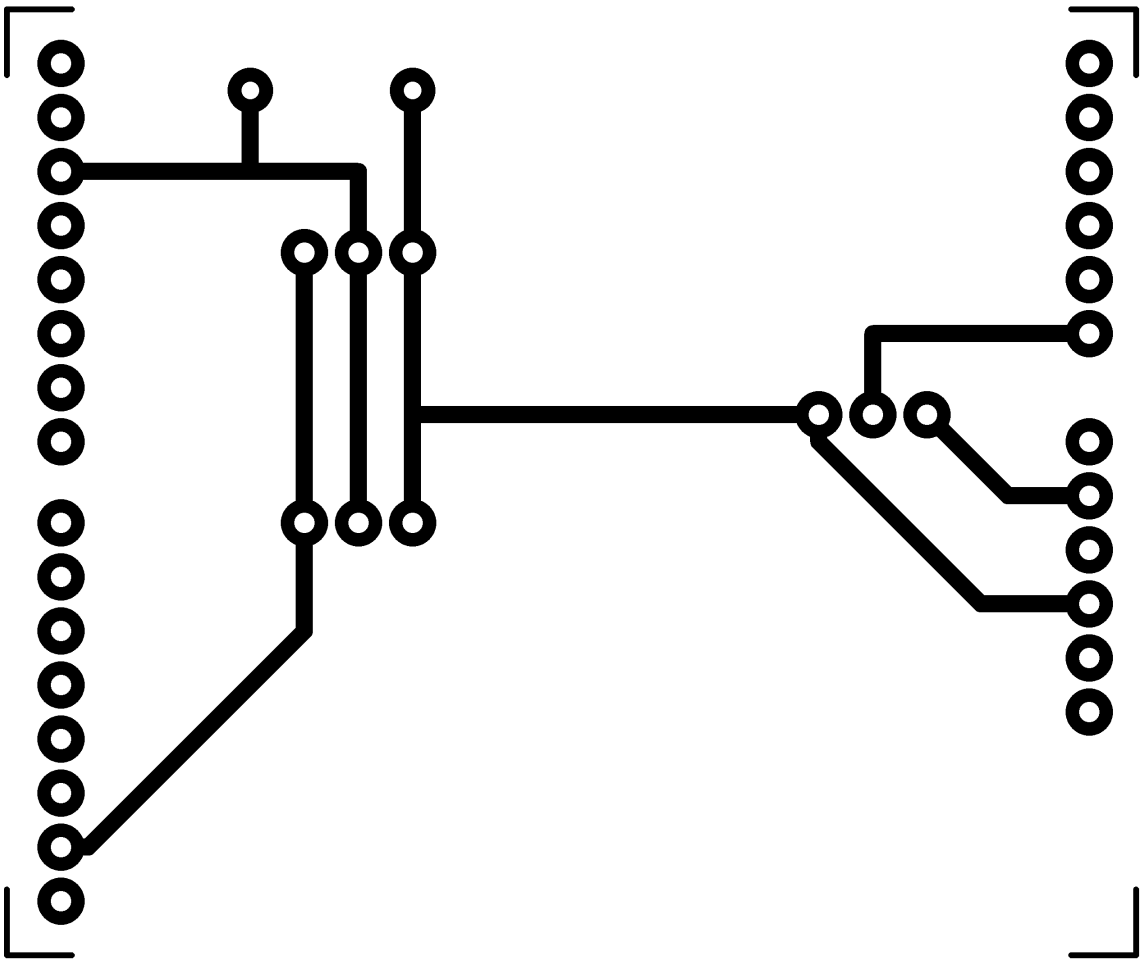

Figura 10: **Placa PCB, cara de pistas**

16. Software: sensor

16.1. IDE Arduino

El sistema se ha desarrollado con el IDE de Arduino disponible en `arduino.cc`, y tomando como punto de partida los ejemplos y código de referencia existente en la mencionada página Web.

Es importante aclarar que el sistema se ha desarrollado y funciona en la versión 18 del IDE, compatible con la versión Duemilanove. No ha sido probado en versiones posteriores (necesarias para el uso del modelo Arduino UNO), pero es de suponer que no deben producirse problemas de portabilidad.

Además de las librerías internas del entorno de desarrollo, se utilizan diversas librerías adicionales para añadir funcionalidad al sistema, entre las que tenemos:

- Librería de gestión de tiempo para la compatibilidad con NTP: `http://www.arduino.cc/playground/Code/Time`
- Librería de datagramas UDP: `https://bitbucket.org/bjoern/arduino_osc/src/14667490521f/libraries/Ethernet/`
- Librería 1-Wire para el uso de los sensores Dallas/Maxim: `http://milesburton.com/index.php/Dallas_Temperature_Control_Library`

Notar que estas librerías deben instalarse en carpetas concretas del sistema, las cuales se indican en el código fuente del programa Arduino.

16.2. Descripción general del programa

El programa consta de las siguientes secciones, tal y como aparecen en el código:

- **Declaraciones y carga de librerías**. Esta sección define las librerías a utilizar, así como las variables necesarias para el funcionamiento del programa.
- Inicialización. En esta parte se inicializan los componentes (como el módulo ethernet), así como las direcciones de los sensores digitales y el sistema de sincronización horaria NTP. Se muestran por puerto serie los datos de inicialización.

- **Programa principal**. El programa principal lleva a cabo las llamadas a las funciones necesarias para el desarrollo del programa, ejecutándose de manera continua, según la siguiente temporización:

  1. El sistema, mediante sus variables internas de tiempo, comprueba si ha pasado un segundo completo.
  2. Una vez cada segundo, se desarrollan las siguientes acciones:
     - Se toman los datos de los sensores y se almacenan en sus variables correspondientes
     - Los datos leídos se muestran por serie (para facilitar la depuración del sistema en los procesos de desarrollo).
     - Se comprueba si han pasado 5 minutos, y en caso afirmativo, se envían los últimos datos recopilados al servidor web. En concreto, se comprueba si estamos en un minuto acabado en 0 ó 5, para hacer los envíos de datos exactamente en esos intervalos (PE, a las 10:00, 10:05, 10:10...). Una vez enviados, se cierra la conexión.
  3. Se vuelve al comienzo del programa

- **Funciones NTP**. Tras el programa principal, encontramos las funciones auxiliares para la sincronización NTP. Dicha sincronización se ajusta de manera automática, no necesitando que el programa haga proceso alguno para mantener la hora correcta, una vez se ha conectado correctamente al servidor NTP.

- **Funciones lectura sensores**. Tenemos a continuación las funciones de lectura de los sensores, tanto de los digitales de temperatura como del analógico de humedad. Estas funciones devuelven los valores leídos en variables del programa principal.

- **Funciones Cliente Web**. Por último, tenemos la función que envía los datos al servidor de registro. Toma los datos de las mediciones y la hora, y genera una petición HTTP en forma de cadena mediante el mecanismo GET, que lanza al servidor remoto para su procesado.

16.3. Lecturas y configuración de los sensores

16.3.1. Sensores de temperatura

Los sensores de temperatura generan un dato en coma flotante. Dado que la lectura de la temperatura del sensor es directa (entrega el dato correcto, en grados centigrados) no es necesario llevar a cabo ningún proceso de adaptación de la medida, y sólo es necesaria una variable para almacenar cada temperatura.

La lectura del dato se lleva a cabo mediante la correspondiente función de la librería 1-Wire.

16.3.2. Configuracion de la dirección 1-Wire

La dirección 1-Wire, que es única para cada sensor y viene impuesta de fábrica, se busca de manera automática al comienzo del programa. Esto evita tener que identificar de manera manual cada sensor con su dirección para poder emplearlos en el código.

El sistema de búsqueda de direcciones identifica todos los sensores 1-Wire presentes en el bus. Esta búsqueda es determinista (si se conectan siempre los mismos sensores, el orden en el que aparecen es siempre el mismo), pero el orden de identificación no tiene que ver con su colocación en el bus. Esto implica que, a priori y sin conectar el sistema, es imposible saber con antelación cual es el sensor asignado como 1 y cual el asignado como 2. Esta identificación deberá hacerse manualmente tras la programación (por ejemplo, observando la salida serie y tomando entre los dedos uno de los sensores para ver cual aumenta sus lecturas), y antes de posicionar el sensor en su ubicación definitiva.

16.3.3. Sensor de humedad

El sensor de humedad es analógico, y como tal tiene el problema, ya mencionado, de la inestabilidad de la medida y del ruido asociado a la misma. Al igual que en versiones anteriores del código, la versión final opta por almacenar 5 lecturas consecutivas del sensor, espaciadas 10ms cada una, y obtener la media de las mismas, para minimizar en la medida de lo posible el ruido electromagnético presente en la medida. Estas lecturas se llevan a cabo en un array de 5 elementos float.

Tal y como se explica en 11.2.2, una vez tomados los valores, se convierten a mV (teniendo en cuenta que trabajamos con una referencia de 5V), y mediante los datos del fabricante se convierten al porcentaje de humedad relativa

16.4. Configuración de variables del sensor

En la primera parte del código (tras la carga de librerias y la definicion de pines) se localiza la configuración de los parámetros de red. Estos parámetros deberán adaptarse a las necesidades y configuración de la red local existente en cada entorno antes de poder usar el sensor.

- **MAC**. La dirección MAC se define mediante software, dado que el módulo ethernet no establece una propia. Es importante que esta dirección sea única para cada sensor, o de lo contrario se producirán problemas en la red.

  La dirección se indica en hexadecimal (anteponiendo *0x* a cada pareja de dígitos hexadecimales).

  ```
  byte mac[] = { 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC };
  ```

- **IP del sensor**. La dirección IP, al igual que la MAC, debe ser única en la red. Esta IP es la que se registra en la BBDD como origen de las medidas, y permite identificar de manera unívoca a cada sensor de la red.

  Adicionalmente, es posible saber si un sensor esta encendido haciendo *ping* a esta dirección IP.

  ```
  byte ip[] = { 192, 168, 1, 2 };
  ```

- **Mascara de red**.

  ```
  byte subnet[] = {255, 255, 255, 0};
  ```

- **Gateway** (puerta de enlace)

  ```
  byte gw[] = {192, 168, 1, 1};
  ```

- **IP del servidor de registro**. Esta IP identifica al servidor de registro de los datos. El sistema no puede emplear DNS, por lo que es obligatorio acceder por IP.

  Es posible emplear un servidor fuera de la subred del sensor (dado que el sistema admite el uso de gateway), pero este punto no ha sido probado en el prototipo final.

  ```
  byte server[] = {192, 168, 1, 100};
  ```

- **IP del servidor NTP**. El servidor NTP, que debe indicarse también mediante IP, puede ser interno o externo. De nuevo, no es posible emplear DNS para resolver un nombre a IP.

  El sistema se ha probado con servidores de la red interna (el propio servidor de registros) y externos (en concreto, se ha probado con el Real Observatorio de la Armada, en *hora.roa.es*, que indica la hora oficial en España).

  ```
  byte SNTP_server_IP[] = {150, 214, 94, 5};
  ```

- **Carpeta de la aplicación web en el servidor de registro**. Esta es la carpeta de Apache que contiene el fichero *index.php*. Por ejemplo, si dicho fichero *index.php* esta en `http://10.66.3.100/weatherlan/index.php`, la carpeta a indicar es `weatherlan`.

  ```
  char directorio[]= "weatherlan";
  ```

17. Software: servidor

17.1. Servidor de registros

El servidor se compone de dos partes: una base de datos, donde se almacena la información recogida de los sensores, y una página web que procesa los datos recibidos y permite las consultas al usuario.

17.1.1. Servidor MySQL

La BBDD sensor consta de dos tablas:

- *Sensores*: en esta tabla se almacenan los identificadores de los sistemas hardware para facilitar su manipulación posterior. Por cada dirección IP (de cada uno de los sensores) se asocia un nombre, así como una zona horaria.
- *Medidas*: esta tabla recoge los datos recibidos desde los sensores. Contiene campos que almacenan la dirección ip, temperaturas y humedad de los sensores, así como fecha y hora de la toma toma de datos.

17.1.2. Servidor Web Apache

El servidor web apache cumple dos propósitos. Por un lado, sirve al usuario las paginas web de consulta de datos. Por otro, recibe y procesa la información recogida por los sensores y la envía al servidor de BBDD.

Ambos procesos se llevan a cabo mediante el lenguaje PHP.

De cara a la consulta, el usuario selecciona un sensor mediante una lista desplegable. Se genera entonces el calendario asociado a dicho sensor, y tras la selección de la fecha se genera el gráfico correspondiente.

17.1.3. Calendario de selección de fechas

El calendario para selección de fechas se genera mediante Unobstrusive Calendar Widget v5 (`http://www.frequency-decoder.com/2009/09/09/unobtrusive-date-picker-widget-`

Mediante el mismo se facilita la selección de fechas válidas, dado que permite indicar que fechas pueden marcarse.

Para lograrlo, se extraen de la BBDD las fechas existentes para el sensor escogido, y se le pasan al widget para que sólo esas fechas puedan ser seleccionadas por el usuario.

17.1.4. Gráficos de medidas

La generación de gráficos para las consultas por parte del usuario se realiza con la ayuda de la librería Open Flash Chart 2 (`http://teethgrinder.co.uk/open-flash-chart-2/`). Esta librería genera un gráfico en flash que permite la interactividad con los datos representados.

El sistema utiliza un marco flash vacío, que contiene todos los elementos necesarios para dibujar los gráficos (tales como tipos de linea, puntos, ejes...). Es en este marco donde se vuelcan los datos mediante la generación de un fichero JSON, que el marco recibe y representa.

El fichero JSON se genera mediante lenguaje PHP, con las consultas adecuadas al servidor de BBDD.

Se generan dos tipos de gráficos:

- Datos de consulta diarios.
- Datos de media diarios.

En función del gráfico solicitado, se genera un fichero JSON u otro. La diferencia entre ambos consiste en la consulta SQL, que en un caso obtiene todos los datos individuales de un día concreto, y en el otro los datos medios, que se calculan de manera directa en el propio servidor MySQL.

17.1.5. Servidor NTP

De cara a mantener el correcto sincronismo en las medidas de los sensores, se activa un servicio NTP en el servidor, con el fin de establecer la hora en cada weatherlan. Es posible emplear tanto servidores NTP ya existentes en Internet, o implementar un servidor propio en el servidor de datos. De esta manera se facilita la integración en aquellos entornos donde no se disponga de salida a internet, o donde está esté limitada en algún aspecto.

17.2. Configuración de la BBDD

La BBDD se crea en el servidor MySQL por cualquiera de los procedimientos existentes para ello, aunque por comodidad se recomienda *PHPMyAdmin*. Una vez creada, las tablas y campos necesarios se generan mediante el script SQL incluido en el proyecto, `weatherlan_mysql_vacia.sql`.

El usuario que emplea la aplicación requiere de los siguientes permisos:

- SELECT
- INSERT
- UPDATE

17.2.1. TimeZones

Para el funcionamiento de la aplicación, es obligatorio que el sistema MySQL tenga configurado correctamente el soporte para *TimeZones*.

El sistema utiliza las funciones internas de MySQL para gestionar las conversiones de fecha y hora de zonas horarias, y no funcionará si no dispone de las tablas de zonas horarias instaladas.

Antes de proceder a la instalación de las mismas, es necesario comprobar si ya están presentes en el sistema, ya que si se sobreescriben con otras versiones, puede causar problemas en su servidor.

Para comprobar si las tablas están disponibles y correctamente instaladas, se puede usar la siguiente consulta SQL:

```
SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
```

Se debe obtener como resultado:

```
'2004-01-01 13:00:00'
```

Si como respuesta se obtiene `NULL`, las tablas no están disponibles en su sistema, y deberá proceder a su instalación

Para más información respecto a las funciones de conversión, y a la instalación de las tablas, se pueden consultar los siguientes enlaces:

- Funcionalidad de fecha y hora en MySQL: `http://dev.mysql.com/doc/refman/5.0/en/time-zone-support.html`
- Instalación de las tablas de TimeZones: `http://dev.mysql.com/doc/refman/5.0/en/mysql-tzinfo-to-sql.html`

17.3. Instalación de la aplicación Web

La instalación de la aplicación web, una vez cumplidos todos los requisitos, conlleva la copia de los ficheros del proyecto al servidor, y su posterior configuración.

17.3.1. Directorio de instalación

La web de consulta y registro de datos necesita estar dentro de un directorio en el servidor, que es la carpeta que se ha indicado en 16.4.

Es decir, que si el servidor está en la dirección: `http://192.168.1.100` la página web tiene que estar ubicada en `http://192.168.1.100/weatherlan`

El nombre concreto de la carpeta, o el uso de subdominios, no afectan a la funcionalidad, pero si la página se cuelga directamente de la raíz del servidor, no funciona correctamente.

17.4. Configuración de la aplicación Web

Tras copiar los ficheros de la web en el directorio seleccionado, se debe configurar la aplicación para que pueda acceder a la BBDD.

Hay que modificar el fichero `weatherlan/content/config.php` adaptando las siguientes variables según la configuración específica de la instalación:

- Nombre del servidor (por lo general, `localhost`, salvo que la página Web y el servidor MySQL se encuentren en máquinas diferentes):

```
$servidor = "localhost";
```

- Nombre de la BBDD creada:

```
$bbdd = "BASE_DE_DATOS";
```

- Usuario de acceso a la BBDD:

```
$usuario = "USUARIO_MYSQL";
```

- Clave de acceso a la BBDD:

```
$clave = "CLAVE_MYSQL";
```

- Zona horaria (por defecto, Europe/Madrid), según el formato TimeZone de MySQL[18]:

```
$zonahoraria = "Europe/Madrid";
```

Con todos estos parámetros configurados, el servidor debe comenzar a registrar los datos recibidos por los sensores, y permitir su consulta sin mayores problemas.

18. Utilización del sistema completo

Una vez que el servidor esté funcionando, y los sensores se hayan configurado y desplegado, el sistema comienza su funcionamiento de manera autónoma.

El sensor envía los datos al servidor cada 5 minutos, y desde ese momento están disponibles para su consulta Web, accediendo a `http://servidor/carpeta/index.php`

19. Pruebas de funcionamiento

19.1. Entorno de pruebas

El entorno de pruebas del proyecto ha sido, de nuevo, la sala de archivo de la Delegación Provincial de Cultura en Córdoba.

[18]Importante: esta configuración será la que defina a que hora comienza y acaba el día en la generación de las gráficas. Si se indica un dato que no es el adecuado, la interpretación que se pueda hacer de los datos registrados será incorrecta.

El sensor ha estado ubicado en una posición central de la sala, cercano a la pared y apoyado sobre uno de los conductos de ventilación (que no estaba en uso, y por lo tanto no ha afectado a las mediciones), y alejado de cualquier persona. El sensor de humedad y uno de los sensores de temperatura se han fijado al conducto de manera que sobresalieran por la parte superior, y el otro sensor de forma que sobresaliera por la parte inferior, siempre sin tocar nada de forma directa, ni entre ellos.

Las pruebas se han llevado a cabo durante 15 días, del 17/8/10 al 1/9/10.

19.2. Alimentación del sistema weatherlan

Las pruebas finales se han llevado a cabo con un módulo POE DWL-P200, configurado para 12V. Se ha comprobado que el sistema funciona correctamente y es capaz de alimentar a todos los componentes conectados de manera correcta y estable.

20. Problemas detectados

A continuación hacemos un seguimiento de los problemas detectados en las fases de desarrollo, y su estado actual.

20.1. Problema inicio ethernet

Como se comentó en el desarrollo inicial del prototipo (véase 12.6.1), las pruebas iniciales obligaban a efectuar un reset manual del sistema al alimentarlo, debido al diseño del módulo ethernet empleado en el sistema.

Este fallo, detectado y documentado en los foros de soporte de arduino, ya se ha resuelto en las versiones actuales del módulo ethernet. No obstante, para aquellas versiones afectadas, existen varias soluciones funcionales. La que se ha empleado en nuestro caso pasa por conectar un pequeño condensador cerámico entre los terminales de tierra y reset, lo que fuerza a la aparición de una señal de nivel bajo en dicho terminal de reinicio durante el primer inicio del sistema. Esto es equivalente a pulsar el botón de reset cada vez que se alimenta el sistema, y permite la correcta inicialización de todo el conjunto.

20.2. Bloqueos del sistema

Los bloqueos que se observaban en el sistema al efectuar diversas consultas web no se han vuelto a observar, dado que ya no se emplea la consulta al servidor interno, y el programa se ha limpiado mucho desde las versiones iniciales, especialmente en lo referente a salida de datos por serie para permitir la depuración del prototipo.

No obstante, para prevenir posibles bloqueos, se puede emplear el sistema de *watchdog* de arduino.

20.2.1. Watchdog

El propio sistema integra una funcionalidad de perro guardián, que fuerza un reset completo del sistema si este no responde durante un tiempo preestablecido. Por desgracia, en la versión Duemilanove, el uso de esta herramienta requiere reprogramar el procesador Atmega integrado en el sistema arduino, y para hacerlo es necesario un dispositivo con el que no contamos actualmente para este proyecto. En las versiones actuales de Arduino Uno, el watchdog viene integrado y activado en el bootloader, lo que facilita su uso.

Para emplear el watchdog, tan sólo hay que añadir unas pocas instrucciones al código del programa, sin alterar para nada su funcionalidad. Para más información, se puede consultar la página `http://blog.bricogeek.com/noticias/arduino/como-utilizar-watchdog-con-arduino/`

20.3. Pérdida de datos

Como se comentó en las pruebas de prototipo, siguen apareciendo pequeños cortes en los registros, asociados a la saturación de la red. En este caso, no hay una solución definitiva que permita resolverlo, dado que cuando el sistema hardware lanza una cadena al servidor de registros no comprueba si este la ha recibido correctamente.

La solución pasaría por reimplementar el código para que arduino comprobara con el servidor la correcta recepción, y en su caso reintentará la comunicación. Sin embargo, esto implica demasiadas modificaciones como para considerarlo en esta versión del sistema, y puede conllevar incluso el uso de sistemas de memoria adicionales en arduino, como por ejemplo tarjetas SD, que permitan almacenar los datos recogidos hasta su envío al servidor.

Teniendo en cuenta que las pérdidas de datos suponen menos del 0,01 % de las mediciones, no se considera necesario en el estado actual del proyecto pasar a estos desarrollos. No obstante, se considera interesante su inclusión en futuras revisiones, especialmente dado que los nuevos módulos ethernet para arduino ya incluyen de serie un slot micro-SD completamente funcional.

20.4. Excesiva sensibilidad sensor humedad

La elevada sensibilidad del sensor de humedad sigue presente, y provoca que los datos registrados de forma consecutiva presenten una elevada variabilidad entre ellos. Aunque hay maneras de reducir esta variabilidad mediante técnicas de suavizado (*smoothing*) de la señal, al emplearlas se pierde precisión de los sensores. Por lo tanto, se ha optado por mantener los datos tal y como se reciben, y de esta forma preservar la veracidad de los datos recopilados.

No obstante, en caso de considerarse necesario establecer un suavizado de los datos, se recomienda mantener el código de arduino sin modificaciones, para poder guardar las medidas reales inalteradas, y suavizar la representación de los datos mediante las consultas PHP a la BBDD. Con esto, siempre se dispondrá de los datos reales exactos.

Parte VI

Cierre del proyecto

21. Mejoras del proyecto

El proyecto, en su estado actual, se considera completo y cerrado. No obstante, como en cualquier otro proyecto, queda mucho espacio para mejorarlo.

Algunos aspectos del mismo, especialmente la parte relativa a la Web de consultas, pueden ser ampliamente mejorados. Entre otras posibles ampliaciones, podemos indicar:

- Automatización del incremento de sensores, dado que el software está pensado exclusivamente para los 3 sensores empleados. No debe resultar especialmente complejo automatizar la gestión del número de sensores de temperatura (recordemos que el protocolo 1-Wire permite hasta 64 dispositivos en un único bus).
- Exportación de los datos a ficheros CSV, que permitiría trasladar los datos almacenados a otras aplicaciones para su procesamiento.
- Mejoras en el análisis estadístico de los datos, generando informes de valores máximos y mínimos entre periodos determinados.
- Implementación de seguridad, para evitar el acceso a los datos por personal no autorizado.
- Sistema de alertas, que avise al usuario por email cuando un parámetro supere unos valores de control.

22. Conclusiones

Tras todo el proceso de desarrollo, y una vez efectuadas todas las pruebas, se puede concluir que el proyecto cumple con los objetivos marcados:

- Es preciso y estable. Los datos recogidos indican que resulta adecuado para la medición y registros contínuos de temperatura y humedad, y resulta más práctico que la toma de datos manual.

- El coste es bajo, y la fabricación del mismo está al alcance de cualquier persona que tenga unos conocimientos básicos de soldadura. La instalación del sistema de registros no requiere de configuraciones especiales, y es asequible para cualquier persona con los conocimientos adecuados de técnico de sistemas informáticos o equivalentes.
- La escalabilidad, aunque no ha podido probarse debido a condicionantes económicos, está garantizada, dado que todo el sistema está pensado para la utilización con múltiples sensores de manera simultánea.

Por todo esto, podemos considerar a WeatherLAN como un proyecto completo y plenamente funcional, aunque por supuesto puede ser ampliamente mejorado.

Aunque el sistema, tal y como se describe aquí, no cubre todas las posibles necesidades que pueden presentarse en las ilimitadas situaciones donde pueda ser de aplicación, es un punto de partida interesante, y más que suficiente para pequeños usos. La disponibilidad del código y la documentación, mediante las licencias empleadas, permiten la adaptación del mismo a otras situaciones.

23. Valoración final

A nivel personal, este proyecto me ha permitido aprender el uso y posibilidades de la plataforma Arduino, y el proceso ha sido muy gratificante. La amplia difusión del sistema, así como la disponibilidad de documentación en Internet, facilitan en gran medida el trabajo con la misma.

La plataforma es muy potente y cómoda de utilizar, y permite llevar a cabo proyectos electrónicos que resultarían muy complejos si hubiera que desarrollar todo el sistema desde cero. Ciertamente, resulta muy interesante para aficionados, o incluso usos profesionales con pequeños requerimientos de potencia, y es una herramienta magnífica para la enseñanza.

Aunque el trabajo ha sido en ocasiones complicado (por ejemplo, la búsqueda de los problemas asociados a los sensores analógicos de temperatura), he aprendido mucho durante el desarrollo, y me abre muchas posibilidades de cara a futuros proyectos. Sin lugar a dudas, este no será el último trabajo que desarrolle con esta plataforma.

Quiero aprovechar para dar mi enhorabuena a todos los integrantes del proyecto Arduino, y a toda la gente que colabora diariamente con el mismo.

Parte VII

Anexo 1: instrucciones

24. Introducción

24.1. Acerca de este documento

Este documento reúne toda la información necesaria para la fabricación e instalación del proyecto WeatherLAN.

Para obtener más información respecto al mismo, se ruega consultar la memoria final del proyecto, disponible en el sitio web `http://weatherlan.hfiel.es`.

24.2. WeatherLAN

WeatherLAN es un sistema de control de temperatura y humedad bajo la plataforma libre Arduino (`http://arduino.cc`), que permite el registro de la información recogida en una base de datos consultable de forma gráfica mediante página web.

El proyecto busca minimizar el coste y maximizar la sencillez de uso, realizando un desarrollo a pequeña escala que facilite su posterior implementación en aquellos entornos donde las soluciones comerciales existentes no sean adecuadas.

En el desarrollo software del proyecto se usan exclusivamente soluciones libres y de uso gratuito.

24.3. Advertencia de seguridad

Antes de comenzar con las instrucciones, se debe hacer una advertencia sobre algunos aspectos del mismo:

- Inyecciones SQL: el sistema no comprueba ni filtra en modo alguno los datos recibidos, ya sean desde el sensor, o los que introduzca el usuario. Esto es un problema de seguridad, dado que abre la puerta a posibles inyecciones SQL en

la base de datos, con el riesgo que esto supone, incluyendo pérdida de datos (ya sean del registro de los sensores, o de otros datos almacenados en el mismo servidor), o escalada de privilegios en el sistema anfitrión.

Dado que la base de datos emplea unos permisos concretos para el trabajo con los registros, el máximo riesgo teórico es la inserción o modificación indebida de los registros almacenados en la propia base de datos. No obstante, es necesario tener esto en cuenta de cara a su utilización en entornos sensibles a ataques.

- Uso abierto en Internet: debido al punto antes comentado, se desaconseja completamente el uso del sistema en servidores accesibles al público desde Internet. No entra dentro del objetivo del proyecto el proporcionar al código de las mínimas medidas de seguridad necesarias para este uso, y sería necesaria una importante modificación del código Web para permitir un uso seguro en entornos abiertos.

- Falsificación de medidas: dado que ni el sensor ni el servidor llevan a cabo validación de los datos recibidos (ni del origen de los mismos), resulta muy sencillo falsificar datos. Por lo tanto, el sistema en su estado actual no está indicado para aquellos entornos donde la veracidad de los datos almacenados resulte vital.

Por todos estos motivos, no puedo dar garantía alguna sobre el sistema, incluyendo su correcto funcionamiento, ni posibles problemas de seguridad que pueda implicar su uso. Siguiendo la filosofía de la mayoría del software existente, el sistema se entrega "tal cual", y no puedo asegurar su funcionamiento libre de errores, ni las posibles consecuencias de su utilización [19].

[19] No obstante, funciona, y dudo mucho que llegue a obligarte a formatear tu ordenador.

25. Montaje del sensor

25.1. Materiales

Los materiales necesarios para la fabricación del sensor son los siguientes[20]:

- 1 placa *Arduino Duemilanove*
- 1 módulo *Arduino Ethernet Shield*
- 1 placa de prototipado *Arduino Protoshield*
- 2 sensores de temperatura *Maxim DS18B20*
- 1 sensor de humedad *808H5V5*
- 1 resistencia 4,7kΩ
- 1 condensador cerámico de 47nF[21]

Adicionalmente, se han usado los siguientes elementos para completar el sistema:

- 3 cables de Audio-CD MPC-2, para la extensión de los sensores desde la protoshield. Cualquier cable de 3 hilos es adecuado para la conexión, pero he usado el MPC-2 (que es de 4 hilos, y he desechado uno) simplemente porque tenía muchos sin usar a mano.
- Hilo de conexión de 0,28mm, para la interconexión de componentes en la *protoshield*
- Barra de contactos de tamaño estandar (1"), para la conexion de la *protoshield* a la placa ethernet, y la conexiones de los conectores MPC-2 a la *protoshield*
- Funda termoretractil para la protección de los contactos de los sensores

[20]Importante: el proyecto, y el prototipo fabricado, se han desarrollado en la versión Arduino Duemilanove. Actualmente hay versiones más modernas disponibles (en concreto, Arduino Uno), pero no debería suponer excesivo problema pasar a la nueva versión.

[21]Opcional, ver siguiente sección para más detalles

25.1.1. Sobre el condensador

El condensador de 47nF sólo es necesario en caso de emplear una antigua Ethernet Shield.

Algunas de las antiguas placas Ethernet tienen un problema de inicialización, que les impide funcionar correctamente sin llevar a cabo un reset en caliente del sistema. Mediante el condensador, que se conecta entre el pin de reset (RESET) y de tierra (GND) de Arduino, se logra solucionar el error, y el conjunto arranca sin problemas.

Las versiones actuales (Ethernet Shield V2) no lo necesitan, y de hecho no se incluye en el esquema final del circuito, ni en el diseño final de la placa PCB, dado que es ya es bastante difícil encontrar estas placas en el mercado. Todas las placas nuevas llevan incorporada una ranura para tarjetas SD, mientras que las viejas (afectadas por el problema) no la traen, y esto puede usarse para diferenciarlas.

25.2. Esquemas eléctricos

Para facilitar la fabricación, se incluyen 2 esquemas, que permiten una sencilla conexión en protoboard o protoshield. Adicionalmente, se incluye el esquema de la placa PCB diseñada para el proyecto.

El primer esquema es de la conexión en protoboard, que se puede encontrar en la figura 11.

El esquema eléctrico, en la figura 12, muestra la conexión de los distintos bloques.

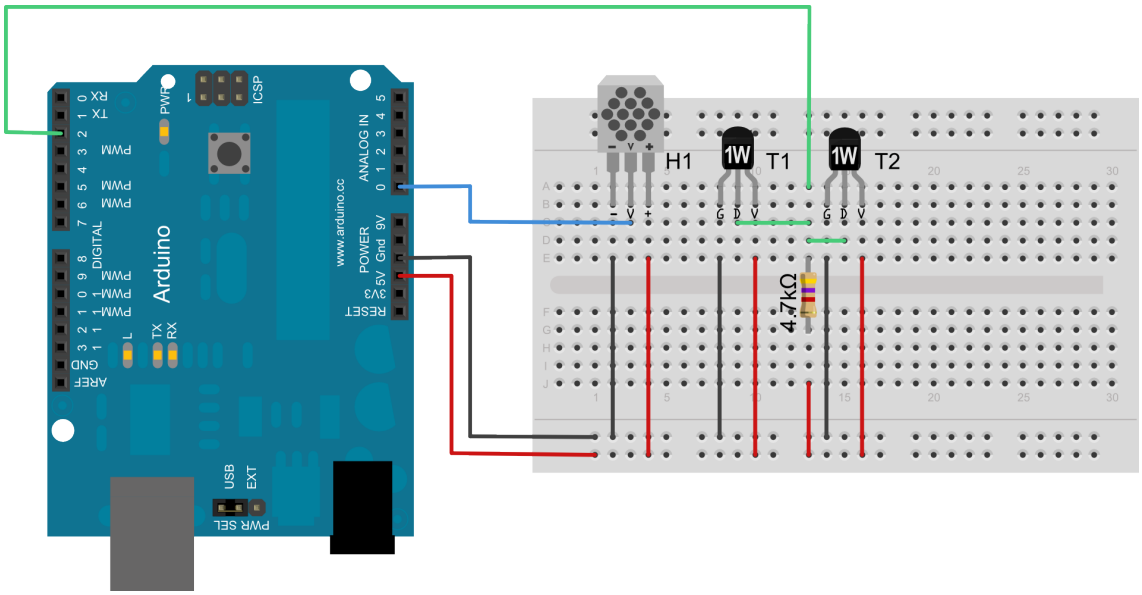

Figura 11: **Protoboard: Conexionado componentes**

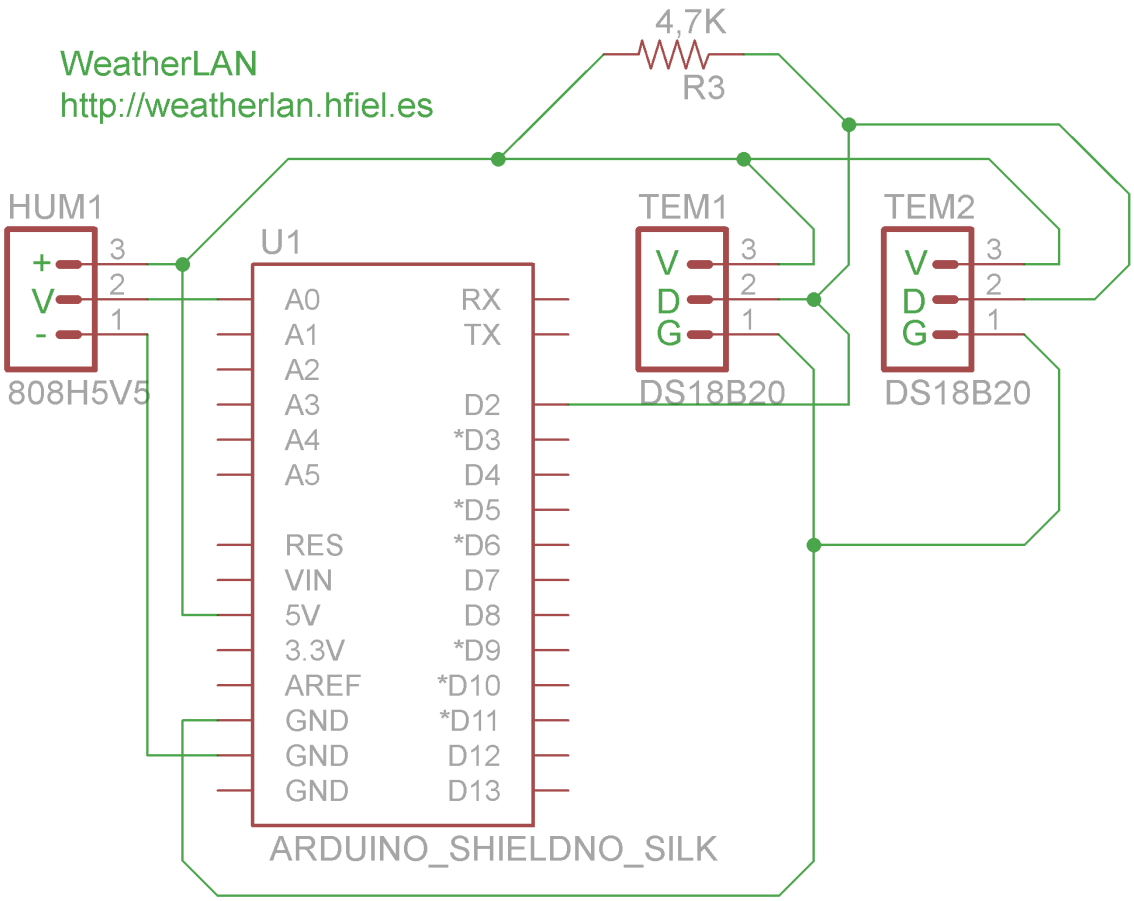

Figura 12: **Esquema eléctrico: Conexionado componentes**

25.3. Placa PCB

Se ha diseñado una placa PCB específica para el proyecto, que facilita la conexión de los componentes con un acabado mucho más profesional.

Es importante indicar que, por razones de diseño de la PCB, se han empleado 2 de los terminales de tierra (GND) existentes en los pines de conexión de Arduino. Estas dos conexiones no aparecen en el esquema de protoboard (figura 11), pero sí aparecen en el resto de esquemas. A nivel práctico, es indistinto conectar la línea de tierra a cualquiera de los pines GND de la plataforma, pero la disponibilidad de diversos pines facilita la creación de placas PCB sin cruces entre las pistas.

En caso de desear utilizar el diseño, se deben utilizar los ficheros disponibles en la web de este proyecto en lugar de las imágenes que aparecen a continuación, para asegurar las dimensiones adecuadas de la placa. Se incluyen las imágenes para la cara de pistas (figura 13), y para la cara de textos (figura 14).

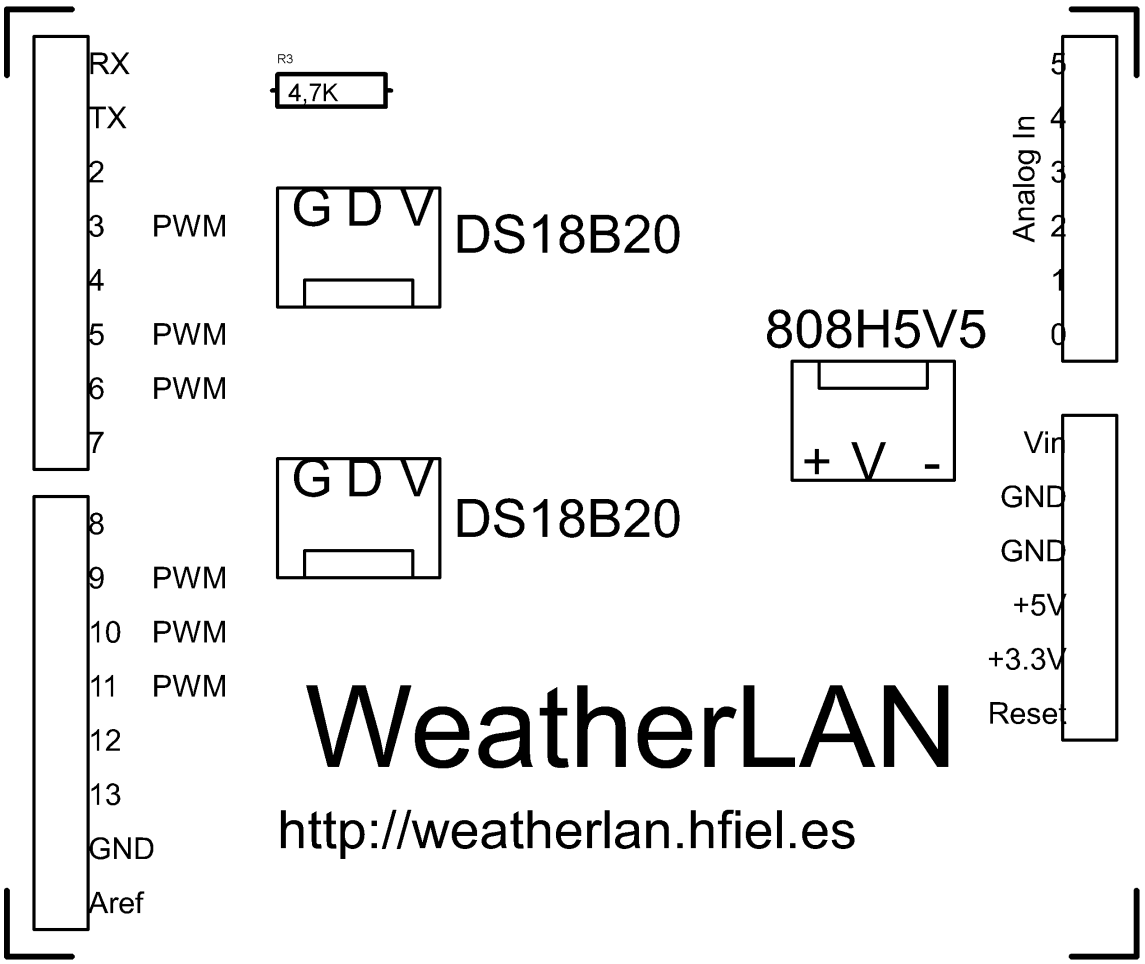

Figura 13: **Placa PCB, cara de serigrafiado**

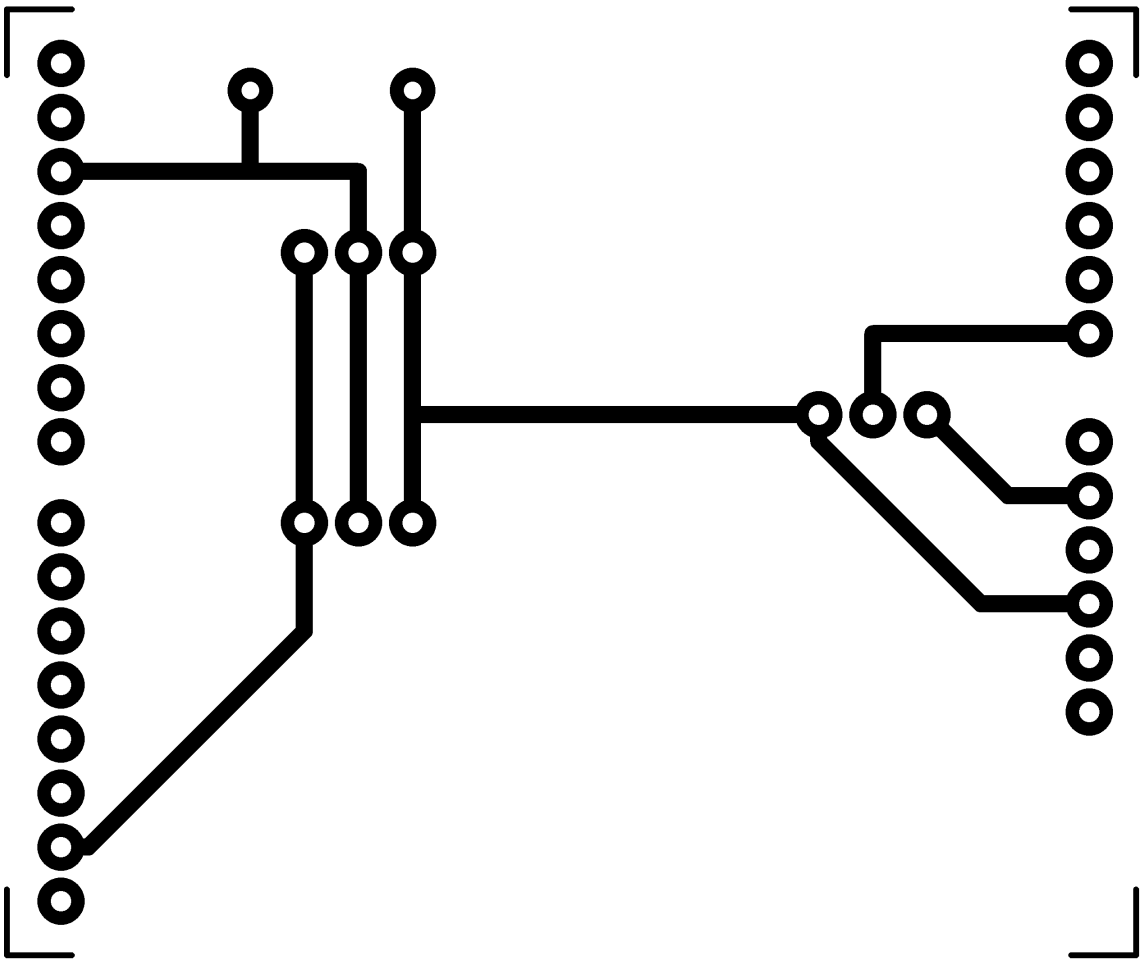

Figura 14: **Placa PCB, cara de pistas**

25.4. Descripción

Internamente, el sistema funciona a 5 voltios (los circuitos empleados funcionan a dicha tensión), y la referencia de las entradas analógicas de Arduino se establece también a 5 voltios (configuración por defecto).

25.4.1. Sensores de temperatura

Los dos sensores de temperatura DS18B20 se conectan mediante alimentación directa (3 cables, un cable para tensión, otro para tierra y otro para datos) a la placa Arduino.

Se ha conectado el pin de alimentación VCC (V) a la salida de 5V de Arduino , y la conexión GND (G) a la tierra. Con respecto al conector de datos 1-Wire (D), la salida de ambos circuitos se ha unido al puerto digital 2 de Arduino, insertando una resistencia de 4,7kΩ en serie entre dicho puerto y la alimentación VCC del circuito.

25.4.2. Sensor de humedad

El sensor de humedad se conecta a la alimentación (+) y tierra (-), y su salida analógica (V) se ha conectado al puerto analógico 0 de Arduino de manera directa.

25.5. Ensamblado

Una vez unidos soldados los componentes a la protoshield, las tres placas se ensamblan una sobre otra, quedando Arduino + Ethernet + Protoshield

25.6. Alimentación

Para alimentar el sistema Arduino, tienes muchas opciones. Si no sabes que hacer, consulta la web `http://arduino.cc`.

La más sencilla es conectarlo mediante USB al ordenador o un adaptador de corriente USB (como los empleados en algunos móviles), y con eso tienes bastante.

Otra opción es utilizar un transformador independiente. Arduino admite voltajes

entre 7 y 12 voltios sin problemas, por lo que un transformador de 9V es perfecto. Sólo hay que conectarlo en la clavija que hay en la placa Arduino.

También puedes usar la opción de alimentar Arduino directamente mediante los pines de alimentación de la placa. Esto se suele hacer cuando usas un panel solar, o una batería, pero ten cuidado en este caso, no vayas a meterle una tensión incorrecta y frías la placa.

La última opción es utilizar un sistema de alimentación PoE (Power Over Ethernet). Esta opción manda por el cable ethernet los datos y la alimentación, y de esta forma puedes poner el sensor muy lejos de un enchufe. El problema es que es la opción más cara.

26. Programación del sensor

Una vez ensamblado, el sensor debe programarse empleando el IDE Arduino (disponible en `http://arduino.cc`). La versión utilizada en el desarrollo es la 0018, aunque hay versiones más modernas que deberían funcionar correctamente. No obstante, no se han probado.

26.1. Preparación del entorno

El sistema necesita una serie de librerías auxiliares, todas ellas disponibles junto con el código en la web del proyecto.

Se indican las rutas para entornos de desarrollo bajo MS Windows, pero la configuración debería ser extrapolable a otros entornos con sólo ajustar adecuadamente las carpetas.

26.2. Librerías en Mis Documentos

Por lo general, las librerías adicionales para usar con Arduino se ubican dentro de la carpeta **Mis Documentos/Arduino** del usuario actual. En el fichero ZIP con el código del proyecto, las librerías que se necesitan en dicha ubicación se pueden encontrar dentro de **codigo/librerias/carpeta mis documentos**. Sólo hay que copiar la carpeta **Arduino** que se encuentra allí hasta la carpeta textbfMis Documentos del sistema.

26.3. Librerías en la carpeta de instalación Arduino

Debido al uso de una librería para el envío de datagramas UDP, hay que copiar 4 ficheros en la carpeta de instalación del IDE Arduino. En el ZIP, los ficheros que hay que copiar se encuentran dentro de **codigo/librerias/carpeta arduino**. Dentro hay una carpeta **libraries/Ethernet** que contiene los 4 ficheros. Sólo hay que soltar la carpeta **libraries** dentro de la carpeta del IDE (ya debería haber una carpeta con el nombre **libraries** allí, si no la ves estás mirando en el sitio equivocado).

26.4. Código del programa

El código principal del sensor (en el ZIP se encuentra dentro de la carpeta **codigo/weatherlan**) es el fichero **weatherlan.pde**. Por requisitos del IDE Arduino, este fichero debe estar en una carpeta con el mismo nombre del programa (en este caso, **weatherlan**).

Adicionalmente, hay una versión especial en la carpeta **codigo/weatherlan_web** que incluye un servidor web local en el propio sensor. Esta versión permite consultar en tiempo real los datos del sensor, accediendo mediante navegador a la IP del mismo. No obstante, esta versión no resulta estable y puede bloquear el sensor si se accede muchas veces seguidas al mismo mediante web. No se recomienda su uso en producción.

26.4.1. Configuración del código

Para poder usar el sistema, hay que configurar cada sensor con una serie de datos.

La parte posterior (Servidor de registro) te explica en más detalle el servidor, y algunos de los parámetros que aquí se configuran.

En la primera parte del fichero **weatherlan.pde** (tras la carga de librerias y la definicion de pines) se localiza la configuración de los parámetros de red. Estos parámetros deberán adaptarse a las necesidades y configuración de la red local existente en cada entorno antes de poder usar el sensor.

- **MAC**. La dirección MAC se define mediante software, dado que el módulo ethernet no establece una propia. Es importante que esta dirección sea única para cada sensor, o de lo contrario se producirán problemas en la red.

La dirección se indica en hexadecimal (anteponiendo *0x* a cada pareja de dígitos hexadecimales).

```
byte mac[] = { 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC };
```

- **IP del sensor**. La dirección IP, al igual que la MAC, debe ser única en la red. Esta IP es la que se registra en la BBDD como origen de las medidas, y permite identificar de manera unívoca a cada sensor de la red.

  Adicionalmente, es posible saber si un sensor esta encendido haciendo *ping* a esta dirección IP.

  ```
  byte ip[] = { 192, 168, 1, 2 };
  ```

- **Mascara de red**.

  ```
  byte subnet[] = {255, 255, 255, 0};
  ```

- **Gateway** (puerta de enlace)

  ```
  byte gw[] = {192, 168, 1, 1};
  ```

- **IP del servidor de registro**. Esta IP identifica al servidor de registro de los datos. El sistema no puede emplear DNS, por lo que es obligatorio acceder por IP.

  Es posible emplear un servidor fuera de la subred del sensor (dado que el sistema admite el uso de gateway), pero este punto no ha sido probado en el prototipo final.

  ```
  byte server[] = {192, 168, 1, 100};
  ```

- **IP del servidor NTP**. El servidor NTP, que debe indicarse también mediante IP, puede ser interno o externo. De nuevo, no es posible emplear DNS para resolver un nombre a IP.

  El sistema se ha probado con servidores de la red interna (el propio servidor de registros) y externos (en concreto, se ha probado con el Real Observatorio de la Armada, en *hora.roa.es*, que indica la hora oficial en España).

  ```
  byte SNTP_server_IP[] = {150, 214, 94, 5};
  ```

- **Carpeta de la aplicación web en el servidor de registro**. Esta es la carpeta de Apache que contiene el fichero *index.php*. Por ejemplo, si dicho fichero *index.php* esta en `http://10.66.3.100/weatherlan/index.php`, la carpeta a indicar es `weatherlan`.

  ```
  char directorio[]= "weatherlan";
  ```

26.5. Programación del sensor

Una vez que el código se ha modificado correctamente, hay que conectar la placa Arduino al PC mediante USB, y proceder a su programación.

Una vez completado el proceso de conexión, se compila el código y se envía a la placa Arduino.

Puedes encontrar detalles concretos al proceso en la web del proyecto Arduino (`http://arduino.cc`), que te explican los pasos concretos para lograr conectar correctamente el equipo y subir un programa al mismo.

27. Servidor de registro

El servidor de registro de datos es un sistema GNU/Linux con un servidor LAMP (Linux, Apache, MySQL, PHP). Puede emplearse cualquier otro sistema operativo (por ejemplo, MS Windows) donde se instalen los componentes indicados.

27.1. Requisitos

- Servidor WEB Apache 2
- PHP 5 ó superior
- MySQL 5 ó superior (ver punto 3 de este documento)
- Acceso a servidor NTP (público en internet, o interno)

El sistema necesita un servidor NTP, pero puede emplearse uno de los muchos existentes en Internet, como por ejemplo el del Real Observatorio de la Armada, en *hora.roa.es*, que indica la hora oficial en España. La IP de dicho servidor es 150.214.94.5

27.2. Instalación de los ficheros

Los ficheros de la página web se encuentran (en el ZIP) en la carpeta **web/weatherlan**. Esta carpeta se debe copiar al directorio de apache (normalmente, **htdocs**, o **www**, según el sistema utilizado).

Realmente el nombre de la carpeta no es importante, pero si es necesario que los ficheros estén dentro de una carpeta. Si cambias el nombre de la carpeta, tendrás que ajustar el código del sensor como se indicó anteriormente.

27.3. Creación de la BBDD

La aplicación necesita una BBDD en MySQL. Lo más sencillo es emplear PHPMyAdmin para crear la base de datos, el usuario de acceso, y las tablas.

La BBDD puede tener cualquier nombre, y deberás crear un usuario y clave asociado a la misma que tenga los siguientes permisos:

- SELECT
- INSERT
- UPDATE

Una vez que tengas la BBDD, hay que crear las tablas. Para ello, dentro del ZIP hay un script SQL llamado **script bbdd/weatherlan_mysql_vacia.sql** que crea automaticamente las tablas y los campos adecuados. Este fichero puedes usarlo desde PHPMyAdmin, en la pestaña importar.

También hay un fichero llamado **datos_prueba.sql** que contiene bastantes registros, para probar que el sistema funciona correctamente.

27.3.1. Configurar las TimeZones

El sistema hace uso de las funcionalidades de conversión de fecha y hora de MySQL. Por lo tanto, requiere que las tablas de TimeZones estén correctamente configuradas en la base de datos. En muchas instalaciones por defecto estas tablas no están incluidas, y deberás añadirlas a mano antes de poder usar la página de consulta.

Importante: antes de proceder a la instalación de las mismas, es necesario comprobar si ya están presentes en el sistema, ya que si se sobreescriben con otras versiones, puede causar problemas en el servidor.

Para comprobar si las tablas están disponibles y correctamente instaladas, se puede usar la siguiente consulta SQL:

```
SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
```

Se debe obtener como resultado:

```
'2004-01-01 13:00:00'
```

Si como respuesta se obtiene `NULL`, las tablas no están disponibles en tu sistema, y deberás proceder a su instalación

Para más información respecto a las funciones de conversión, y a la instalación de las tablas, se pueden consultar los siguientes enlaces:

- Funcionalidad de fecha y hora en MySQL: `http://dev.mysql.com/doc/refman/5.0/en/time-zone-support.html`
- Instalación de las tablas de TimeZones: `http://dev.mysql.com/doc/refman/5.0/en/mysql-tzinfo-to-sql.html`

27.4. Configuración de la aplicación web

Cuando todo esté listo, sólo queda configurar la aplicación web para que pueda acceder a la BBDD.

Dentro de la carpeta de apache (recuerda, **htdocs** o **www**), hay que modificar el fichero `weatherlan/content/config.php` adaptando las siguientes variables según la configuración específica de la instalación:

- Nombre del servidor (por lo general, `localhost`, salvo que la página Web y el servidor MySQL se encuentren en máquinas diferentes):

  ```
  $servidor = "localhost";
  ```

- Nombre de la BBDD creada:

  ```
  $bbdd = "BASE_DE_DATOS";
  ```

- Usuario de acceso a la BBDD:

  ```
  $usuario = "USUARIO_MYSQL";
  ```

- Clave de acceso a la BBDD:

```
$clave = "CLAVE_MYSQL";
```

- Zona horaria (por defecto, Europe/Madrid), según el formato TimeZone de MySQL[22]:

```
$zonahoraria = "Europe/Madrid";
```

Con todos estos parámetros configurados, el servidor debe comenzar a registrar los datos recibidos por los sensores, y permitir su consulta sin mayores problemas.

28. Uso del sistema

Nota: próximamente pondré más capturas de pantalla y mejoraré las explicaciones de uso. De todas formas, la página Web es bastante simple de usar.

Una vez que tengas todo preparado, sólo queda conectar el sensor a la red local, encenderlo, y esperar a que comience a registrar datos. Como el sistema guarda los datos cada 5 minutos en punto (por ejemplo, a las 12:00:00, a las 12:05:00...) es posible que tengas que esperar un rato hasta que aparezca el primer registro.

Una vez que tengas datos, sólo hay que consultarlos desde la página web que has copiado en apache. Por ejemplo, si tu servidor está en la IP 192.168.1.100, y has usado la carpeta weatherlan, tendrás que ir con tu navegador hasta `http://192.168.1.100/weatherlan/index.php`.

En la figura 15 puedes ver una captura de la página. En la parte superior de la página tienes varias pestañas, que te permiten selección entre la vista de datos diarios, las medias y la configuración de los sensores. Las dos primeras opciones te permiten escoger el sensor mediante una lista desplegable.

[22]Importante: esta configuración será la que defina a que hora comienza y acaba el día en la generación de las gráficas. Si se indica un dato que no es el adecuado, la interpretación que se pueda hacer de los datos registrados será incorrecta.

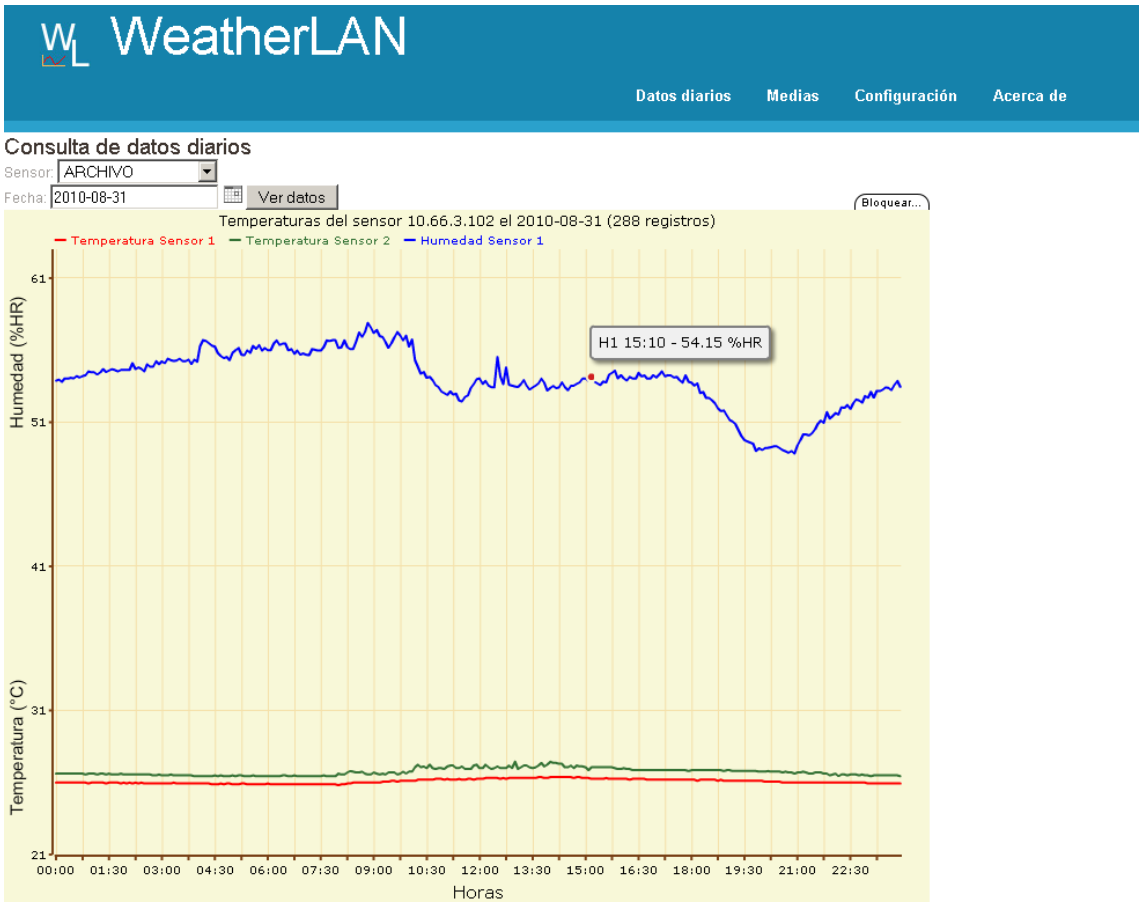

Figura 15: **Interfaz Web**

Cuando tengas seleccionado el sensor, aparece un calendario (figura 16), donde sólo puedes hacer clic en las fechas que contienen datos (el resto de fechas aparecen sombreadas y en cursiva). De forma predeterminada, la fecha seleccionada es la última sobre la que haya datos.

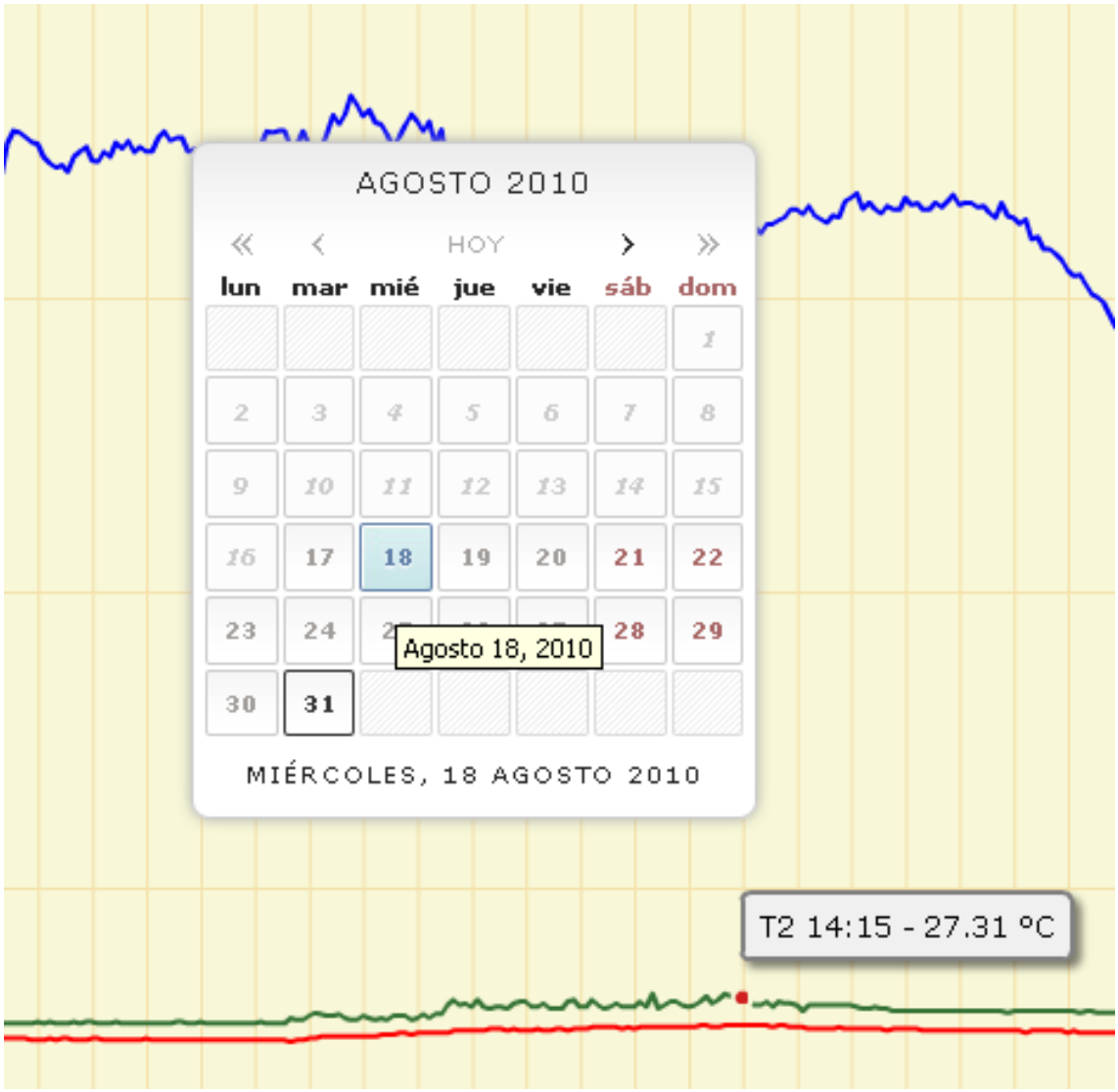

Figura 16: **Selección de fechas**

Cuando hayas escogido la fecha, aparece la ventana de gráficos, que te mostrará los tres valores almacenados, ordenados por horas. Si mueves el ratón sobre uno de los datos, te muestra el valor exacto y la hora a la que se recogió.

En el caso de que haya cortes en los registros (es decir, si alguna de las medidas se ha perdido en el camino desde el sensor hasta el servidor), aparece un aviso en la parte superior, y el corte se indica con dos pequeños triángulos rojos en el punto

exacto donde se haya producido el fallo.

28.0.1. Exportar gráficas

Si usas el navegador Mozilla Firefox, podrás exportar la gráfica en formato PNG. Para ello, haz clic con el botón derecho sobre la gráfica, y selecciona la opción **Save as PNG**.

Esta opción no funciona correctamente en Internet Explorer, y no la he probado con otros navegadores.

28.0.2. Configurar nombres

En la parte superior de la web hay una pestaña que te permite acceder a la configuración de los sensores. Desde aquí podrás escoger un sensor, y configurar su nombre, así como la zona horaria del mismo. Por defecto, la zona que se establece es Europe/Madrid, pero puedes ajustarla para que corresponda con la que estés usando. De esta forma, las gráficas se ajustan adecuadamente con tu reloj local.

Parte VIII

Anexo 2: Código Arduino

Código correspondiente al sensor Arduino para WeatherLAN. Fichero: **weatherlan.pde**

```
/*
WeatherLAN: Sensor de temperatura y humedad con arduino
http://weatherlan.hfiel.es/

 Copyright 2011 Hector Fiel


 This file is part of WeatherLAN.

    WeatherLAN is free software: you can redistribute it and
       /or modify
```

```
    it under the terms of the GNU General Public License as
       published by
    the Free Software Foundation, either version 3 of the
       License, or
    (at your option) any later version.

    WeatherLAN is distributed in the hope that it will be
       useful,
    but WITHOUT ANY WARRANTY; without even the implied
       warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
       See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General
       Public License
    along with WeatherLAN.  If not, see <http:www.gnu.org/
       licenses/>.


Autor: Hector Fiel (http://hfiel.es)
Fichero: weatherlan_web.pde
Version: 1
Revision: Version final.
Enlace: http://weatherlan.hfiel.es
Importante: Codigo probado en Arduino Duemilanove, en la
   version 0018 del IDE.
Seguir las instrucciones para la instalacion de las
   librerias UDP necesarias (indicadas mas abajo, en la
   carga de la libreria).

Resumen version: version para 2 sensores de temperatura
   DS18B20, y un sensor de humedad 808H5V5.
Referencia de 5V, salida por serie y web, almacenamiento en
   BBDD remota
Sincronizacion de tiempo con servidor NTP
Busqueda automatica direccion sensores 1-Wire


Reconocimientos:
El codigo en general se ha desarrollado mediante los
   ejemplos de la web de arduino.cc
 http://arduino.cc/en/Tutorial/HomePage
```

```
El codigo de servidor y cliente web se basa en los ejemplos
   de las librerias ethernet de arduino.cc
 http://arduino.cc/en/Reference/Ethernet
El codigo para NTP se basa en el ejemplo TimeNTP.PDE de la
   libreria time de arduino.cc
 http://www.arduino.cc/playground/Code/Time
El codigo para los sensores de temperatura se basa en los
   ejemplos de arduino.cc y
la libreria de temperatura 1-Wire de Miles Burton y el
   ejemplo multiple.pde
 http://www.arduino.cc/playground/Learning/OneWire
 http://milesburton.com/index.php/
     Dallas_Temperature_Control_Library

Descripcion:

La medida de temperatura se realiza mediante sensores
   digitales de Dallas/Maxim que funcionan con el
protocolo 1-Wire (permiten el envio y recepcion de datos
   usando una sola linea digital, con hasta
64 dispositivos compartiendo dicha linea). Se usa la
   configuracion 1-Wire Normal (fuente de
alimentacion externa, total de 3 hilos) para agilizar la
   lectura de datos de los sensores. Se puede
usar el modo de alimentacion parasito (dos hilos) pero esto
   implica un retardo de 750ms en cada lectura.
Los sensores se conectan a dos conectores en la placa, el
   orden de los mismos viene definido por
la busqueda del bus que efectua la libreria, y es
   determinista (el sensor identificado como 0 siempre
sera el mismo, si no se cambian los sensores).

La medida de humedad se realiza mediante un sensor analogico
     conectado al DAC del Arduino.

Se realiza almacenamiento en BBDD, mediante una peticion web
     por GET al servidor remoto, donde se
ejecuta un servidor Web Apache con PHP y una BBDD MySQL
La peticion es de la forma:

http:\\<IP servidor>\<carpeta weatherlan>\sensor.php?reg=1&
   t1=14.1&t2=14.2&h1=50.5&hora=12&minuto=25&dia=28&mes=5&
   anyo=2010
```

```
donde los datos en cada campo get se insertan mediante las
   lecturas de los sensores.
La IP del servidor se indica en las variables de
   inicializacion de ethernet.
La carpeta de instalacion de weatherlan se indica en la
   funcion de conexion "envioweb".
Los parametros enviados por get son:
t1: temperatura sonda 1
t2: temperatura sonda 2
h1: humedad sonda 1
hora: hora de la medicion
minuto: minuto de la medicion
dia: dia de la medicion
mes: mes de la medicion
anyo: anyo de la medicion
Si un campo de sensor (t1, t2, h1) no se incluye, la bbdd
   almacena NULL en ese campo.

INFORMACION SOBRE PROTOCOLO DE TIEMPO NTP:
El protocolo NTP permite sincronizar la hora con un servidor
    de internet. Esto se usa para poder hacer
registros en el servidor remoto en tiempos exactos (PE, cada
    15 minutos). El envio de la fecha y hora
desde el arduino permite garantizar que los datos se
   almacenan con la fecha exacta, sin depender de la
hora del sistema del servidor con PHP y la BBDD. Es
   importante tener en cuenta que el sistema completo
trabaja con hora UTC (universal), y no con la hora espanola.
    Esto permite evitar problemas con los
cambios de horario de verano (como tener dos registros con
   la misma hora el dia que se retrasa la hora).
Esto hay que tenerlo en cuenta en el servidor web a la hora
   de procesar los datos para mostrarlos.
Para el correcto funcionamiento, el servidor MyQSL de
   registro de datos debe tener configuradas correctamente
   las TimeZones

Los tres servicios de red (cliente web, servidor web y
   cliente ntp) se implementan mediante funciones separadas,
    para
facilitar el codigo principal.

//****
PINES ARDUINO EMPLEADOS:
```

```
//****
Pines usandos en las entradas analogicas de Arduino:
0: 808H5V5

Pines usados en las entradas digitales de Arduino:
3: 1 Wire (2 sensores DS18B20)

*/

//****
//CARGA DE LIBRERIAS
//****
//Se indica la libreria y la direccion de descarga
#include <Ethernet.h>  //Libreria Modulo Ethernet arduino (
   incluida en el IDE)

#include <Time.h>      //Libreria de tiempo arduino
//http://www.arduino.cc/playground/Code/Time

#include <UdpBytewise.h> //Libreria UDP de bjoern@cs.
   stanford.edu, para mensajes UDP
//http://bitbucket.org/bjoern/arduino_osc/src/14667490521f/
   libraries/Ethernet/
//NOTA: las librerias UDP tienen que estar en el directorio
   libraries\Ethernet del software IDE Arduino,
//no funcionan en el directorio de librerias del usuario
//hay que incluuir 4 ficheros:
//UdpRaw.h
//UdpRaw.cpp
//UdpBytewise.h
//UdpBytewise.cpp

//Librerias onewire
#include <OneWire.h> //Libreria para protocolo 1-Wire
//(incluida en la descarga de DallasTemperature.h, mas abajo
   )
//http://www.pjrc.com/teensy/td_libs_OneWire.html

#include <DallasTemperature.h> //libreria para los sensores
   de temperatura Dallas/Maxim
//http://milesburton.com/index.php/
   Dallas_Temperature_Control_Library
```

```
//****
//DEFINICION DE PINES
//****
//Pin analogico a usar en el sensor de humedad
#define puertoH1 0

//Pin digital para los sensores de temperatura
#define ONE_WIRE_BUS 2


//****
//DECLARACION DE VARIABLES PARA CONEXIONES ETHERNET
//****


//****
//DIRECCION MAC
//****
//la direccion mac debe ser unica para cada dispositivo de
   la red
//esta viene puesta en el ejemplo de Arduino, aqui se ha
   cambiado el ultimo
//valor para evitar problemas con otros ejemplos (era 0xED,
   se pone 0xCD)
 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xCD };

//****
//IP SENSOR WEATHERLAN
//****
//la direccion ip tambien tiene que ser unica en la red,
   ponemos una que este libre en
//nuestro rango de direcciones (dependera de la
   configuracion de red de cada uno)
 byte ip[] = { 10, 66, 3, 101 };

//****
//GATEWAY (PUERTA DE ENLACE)
//****
//puerta de enlace (dependera de la configuracion de red de
   cada uno)
 byte gw[] = {10, 66, 0, 1};

//****
//MASCARA DE SUBRED
```

```
//****
//subred (dependera de la configuracion de red de cada uno)
 byte subnet[] = {255, 255, 255, 0};

//****
//IP SERVIDOR BBDD
//****
//definimos la ip del servidor remoto donde se va a conectar
    arduino para
//almacenar los datos, ajustar segun la IP del servidor de
   cada uno
 byte server[] = {10, 66, 3, 100};


//****
//IP SERVIDOR NTP
//****
//IP del servidor NTP, ajustar al servidor que quiera usar
   cada uno
//Por defecto, se usa la misma IP que el servidor de
   registros BBDD (habra que activar el servicio NTP)
byte SNTP_server_IP[] = {10, 66, 3, 100};
//en caso de no usar servidor NTP local, prueba con alguno
   de estos
//byte SNTP_server_IP[] = {150, 214, 94, 5};   //hora.roa.es
   : Real Observatorio de la Armada, hora ofical en espana
//byte SNTP_server_IP[] = { 192, 43, 244, 18}; // time.nist.
   gov
//byte SNTP_server_IP[] = { 130,149,17,21};    // ntps1-0.cs
   .tu-berlin.de
//byte SNTP_server_IP[] = { 192,53,103,108};   // ptbtime1.
   ptb.de

//****
//PUERTO DE ESCUCHA PARA EL SERVIDOR WEB LOCAL DEL SENSOR
//****
//puerto donde escucha peticiones web arduino, por defecto
   el 80 (HTTP)
Server servidor = Server(80);


//****
//DECLARACION DE VARIABLES PARA ALMACENAMIENTO DE DATOS
//****
```

```
//Variables donde almacenar las lecturas, con inicializacion
    a 0
//En el caso de la humedad, se usa un array para poder hacer
//multiples lecturas y obtener una media, dado que son datos
//analogicos y pueden oscilar bastante entre dos medidas.
//En las temperaturas, el sensor entrega un dato final y
   estable
//por lo que no necesitamos hacer medias
//Arrays donde almacenar datos crudos de humedad
 int vhumedad[5] = {0,0,0,0,0};
//variables para almacenar las medias
 float mhumedad = 0.0;
//variables con datos finales
 float temperaturaT1 = 0.0;
 float temperaturaT2 = 0.0;
 float humedad = 0.0;

//****
//DECLARACION DE VARIABLES PARA SINCRONIZACION DE TIEMPO NTP
//****

//VARIABLE AUXILIAR PARA ALMACENAR EL VALOR DE TIEMPO NTP
// auxiliar para saber cuando se mostro el reloj digital
 time_t prevDisplay = 0;

//VARIABLE DE DESFASE DE TIEMPO NTP
//(NO DEBERIA SER NECESARIO MODIFICARLA)
//desfase de hora: el sistema siempre recibe y entrega hora
   UTC, y por lo tanto
// esta variable siempre se mantiene a 0. Para usar tiempo
   local, habra que cambiar su valor
// Establece el valor al desfase en segundos con tu tiempo
   local.
 const  long timeZoneOffset = 0L;

//****
//DELAY ENTRE LECTURAS (ms) PARA EL SENSOR DE HUMEDAD
//****
 int delay_lectura = 20;

//****
//DECLARACIONES PARA EL BUS 1-WIRE (SENSORES DE TEMPERATURA)
//****
```

```
//Creamos una instancia del objeto onewire
 OneWire oneWire(ONE_WIRE_BUS);
//pasamos la referencia del objeto a la libreria de
   DallasTemperature
 DallasTemperature sensors(&oneWire);

//****
//DIRECCIONES SENSORES 1-WIRE
//****
//variables para almacenar la direccion de los sensores
 DeviceAddress sensorTemp1;
 DeviceAddress sensorTemp2;
//Establecemos la precision del sensor (9,10,11,12 bytes).
   Afecta al tiempo de respuesta
#define TEMPERATURE_PRECISION 12


//CONFIGURACIoN DE ARDUINO
void setup() {

  //Inicializacion del puerto serie a 9600 baudios
   Serial.begin(9600);
   Serial.println("Sensor de temperatura y humedad.
      Inicializando...");

   //Establecemos la referencia analogica en DEFAULT (5V)
   analogReference(DEFAULT);

   //1-WIRE
   //incializacion sensores onewire
   sensors.begin();
   //busqueda de Dispositivos  1-wire, por metodo del indice
   // localizamos los dispositivos en el bus
   Serial.print("Buscando dispositivos 1-Wire...");
   Serial.print("Encontrados ");
   Serial.print(sensors.getDeviceCount(), DEC);
   Serial.println(" dispositivos.");
   //mostramos el modo de alimentacion de los dispositivos
   Serial.print("Alimentacion parasita: ");
   if (sensors.isParasitePowerMode()) Serial.println("ON");
   else Serial.println("OFF");

   //****
   //BUSQUEDA DE DIRECCIONES POR INDICE
```

```
//****
//busqueda de direcciones por indice (Necesaria para
   obtener las direcciones la primera vez)
if (!sensors.getAddress(sensorTemp1, 0)) Serial.println("
   No se puede encontrar la direccion del dispositivo 0")
   ;
if (!sensors.getAddress(sensorTemp2, 1)) Serial.println("
   No se puede encontrar la direccion del dispositivo 1")
   ;
//mostramos las direcciones obtenidas
Serial.print("Direccion sensor 1 (id 0): ");
printAddress(sensorTemp1);
Serial.println();
Serial.print("Direccion sensor 2 (id 1): ");
printAddress(sensorTemp2);
Serial.println();

//establecemos la precision de los sensores
sensors.setResolution(sensorTemp1, TEMPERATURE_PRECISION)
   ;
sensors.setResolution(sensorTemp2, TEMPERATURE_PRECISION)
   ;

//mostramos la precision establecida en cada sensor
Serial.print("Resolucion sensor 1: ");
Serial.print(sensors.getResolution(sensorTemp1), DEC);
Serial.println();

Serial.print("Resolucion sensor 2: ");
Serial.print(sensors.getResolution(sensorTemp2), DEC);
Serial.println();

//INICIALIZACION ETHERNET Y SERVIDOR
Ethernet.begin(mac, ip, gw, subnet);

//****
//INICIALIZACION SERVIDOR WEB LOCAL
servidor.begin();

//SINCRONIZACION NTP
//establece la sincronia de tiempos y la mantiene
//NO ES NECESARIO ACTUALIZAR LA SINCRONIZACION A MANO
//*****
```

```
    //IMPORTANTE: SI AL ENCENDER EL ARDUINO NO SE LOGRA
       SINCRONIZAR,
    //NO SE INICIA LA EJECUCION
    //*****
    setSyncProvider(getNtpTime);
    while(timeStatus()== timeNotSet)
     ; // Espera hasta que se recibe sincronizacion

    //finalizamos la inicializacion
    Serial.print("Finalizada inicializacion. Direccion IP: ")
       ;
    for (int i = 0; i < 4; i++) {
       Serial.print(int(ip[i]));
       if (i<3)
       {
        Serial.print(".");
       }
     }
     Serial.println();
}

//PROGRAMA PRINCIPAL
void loop() {

  //PROCESAMOS CADA SEGUNDO
  if( now() != prevDisplay) //si la hora ha cambiado (ha
     pasado 1 segundo)
  {
    //actualizamos la variable de hora con el tiempo actual
    prevDisplay = now();
    //mostramos la hora por serie
    digitalClockDisplay();
    //leemos la temperatura de los sensores,
    medida_temperatura();
    //leemos la humedad
    medida_humedad();
    //imprimo sensores por puerto serie
    muestraSerie();

    //escritura en servidor remoto cada 5 minutos
    //Se incluye aqui dentro (en proceso cada 1s) para que
       se haga 1 sola escritura por segundo
    //si la ponemos fuera, y el ciclo total tarda menos de 1
       s, haria varias escrituras en un solo ciclo
```

```
    //El registro se hace cuando minuto es 0 y segundo es 0
       (hora en punto)
    //y tambien cuando minuto es multiplo de 5 (el modulo
       con 5 es cero) y segundo es 0
    //para hacer la escritura una sola vez, en el segundo
       indicado
    //
    if ( (minute()==0 && second()==0) || (minute()%5==0 &&
       second()==0) )
    {
      //envio al servidor remoto
      envioweb(temperaturaT1, temperaturaT2, humedad);
    }

  }

   //****
   //MOSTRAR DATOS EN SERVIDOR WEB LOCAL
   //****
   //llamo al servidor web local para mostrar datos
   servidorweb();

  //esperamos 1ms antes de volver a empezar
   delay(1);

}

   //****
   //FUNCION DE SERVIDOR WEB LOCAL
   //****
   //funcion servidorweb, para mostrar resultados
      directamente por web en la IP del arduino
void servidorweb()
{

  //SERVIDOR WEB
  //El codigo es una adaptacion directa del servidor Web
     incluido en los ejemplos de Arduino
  //solo se han traducido y ampliado los comentarios, y
     adaptado la salida para que muestre
  //los valores de temperatura y humedad calculados
  //El servidor responde en la IP indicada en el comienzo
     del programa, en la parte de
  //inicializacion ethernet, por la variable ip[]
```

```
//creamos una nueva instancia del servidor, y la ponemos
   como disponible
//para que el cliente se pueda conectar
Client cliente =  servidor.available();
//Comprobamos si se crea el servidor correctamente
if (cliente) {
  //variable auxiliar: una peticion http de un cliente
     termina con una linea en blanco
  boolean current_line_is_blank = true;
  //mientras tengamos un cliente conectado
  while (cliente.connected()) {
    //si el cliente esta disponible
    if (cliente.available()) {
      // leemos la peticion del cliente
      char c = cliente.read();
      //si hemos llegado al final de la linea (hemos
         recibido un
      //caracter de salto de linea) y la linea esta vacia,
          la peticion
      //http ha finalizado, por lo tanto ya podemos mandar
          la respuesta
      if (c == '\n' && current_line_is_blank) {
        // devolvemos una cabecera HTTP estandar
        cliente.println("HTTP/1.1 200 OK");
        cliente.println("Content-Type: text/html");
        cliente.println();

        //comienza el codigo que se va a ver en la pagina

        //mostramos la IP del sensor
          cliente.print("<h1>WeatherLAN: sensor de
             temperatura y humedad</h1><a href='http://");
            for (int i = 0; i < 4; i++) {
              cliente.print(int(server[i]));
              if (i<3)
              {
               cliente.print(".");
              }
            }
          cliente.print("/weatherlan/index.php'");
          cliente.print(">Historico de datos</a><br />
             Sensor en IP: ");
            for (int i = 0; i < 4; i++) {
```

```
            cliente.print(int(ip[i]));
            if (i<3)
            {
             cliente.print(".");
            }
          }
        cliente.println("<br />Fecha: ");
        //fecha y hora
        cliente.print(day());
        cliente.print("/");
        cliente.print(month());
        cliente.print("/");
        cliente.print(year());
        cliente.print(" ");
        cliente.print(hour());
        cliente.print(":");
        cliente.print(minute());
        cliente.print(":");
        cliente.print(second());
        // Salida de los valores de los sensores
        cliente.print("<br /><h2>Temperatura T1: ");
        cliente.print(temperaturaT1);
        cliente.print(" &ordm C</h2><br /> <h2>
           Temperatura T2: ");
        cliente.print(temperaturaT2);
        cliente.print(" &ordm C</h2><br /> <h2>Humedad:
           ");
        cliente.print(humedad);
        //la ultima escritura la hacemos con println
           para que salte de linea al finalizar
        cliente.println(" %HR</h2><br />");
      break;
    }
    //esta parte es para saber si ya ha terminado la
       peticion del cliente
    if (c == '\n') {
      // estamos empezando una nueva linea
      current_line_is_blank = true;
    } else if (c != '\r') {
      // hemos recibido algun caracter en la linea
         actual.
      current_line_is_blank = false;
    }
  }
```

```
    }
    //cerramos el cliente
    cliente.stop();
    // esperamos para permitir al navegador recibir los
       datos
    delay(1);
    }
}


//funcion envioweb, para almacenar datos en servidor remoto
void envioweb(float t1, float t2, float h1)
{

 Client client(server, 80);
 Serial.println("Envio de datos al servidor web:");
  //inicializacion cliente web por ethernet

 //creamos una clase cliente para que conecte al servidor
    indicado en el puerto 80
    if (client.connect()) {
      Serial.print("Conectado OK: ");
      //comenzamos la peticion http
      client.print("GET /weatherlan/index.php?reg=1&t1=");
      Serial.print("GET /weatherlan/index.php?reg=1&t1=");
      //ponemos las variables
      client.print(t1);
      Serial.print(t1);
      client.print("&t2=");
      Serial.print("&t2=");
      client.print(t2);
      Serial.print(t2);
      client.print("&h1=");
      Serial.print("&h1=");
      client.print(h1);
      Serial.print(h1);
      client.print("&hora=");
      Serial.print("&hora=");
      client.print(hour());
      Serial.print(hour());
      client.print("&minuto=");
      Serial.print("&minuto=");
      client.print(minute());
      Serial.print(minute());
```

```
        client.print("&dia=");
        Serial.print("&dia=");
        client.print(day());
        Serial.print(day());
        client.print("&mes=");
        Serial.print("&mes=");
        client.print(month());
        Serial.print(month());
        client.print("&anyo=");
        Serial.print("&anyo=");
        client.print(year());
        Serial.print(year());
        //terminamos indicando el protocolo HTTP y un salto de
            linea
        //client.println(" HTTP/1.1");
        //Serial.print(" HTTP/1.1");
        //linea en blanco para que termine la peticion
        client.println();
        Serial.println();
      }
      else
      {
        Serial.println("Error al conectar. Compruebe la
           conexion.");
      }

      //paramos el cliente
      client.stop();
      delay(1);
}

//FUNCIONES MOSTRAR HORA POR SERIAL
void digitalClockDisplay(){
  // digital clock display of the time
  Serial.print("Fecha: ");
  Serial.print(day());
  Serial.print("/");
  Serial.print(month());
  Serial.print("/");
  Serial.print(year());
  Serial.print(" HORA: ");
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
```

```
  Serial.println();
}

void printDigits(int digits){
  // utility function for digital clock display: prints
     preceding colon and leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

//FUNCIONES NTP

//obtiene la hora del servidor NTP, devuelve como Unsigned
   Long, mediante UDP
unsigned long getNtpTime()
{
  sendNTPpacket(SNTP_server_IP);
  delay(1000);
  if ( UdpBytewise.available() ) {
    for(int i=0; i < 40; i++)
       UdpBytewise.read(); // ignore every field except the
          time
    const unsigned long seventy_years = 2208988800UL +
       timeZoneOffset;
    return getUlong() -  seventy_years;
  }
  return 0; // return 0 if unable to get the time
}

//envia paquete NTP al servidor para hacer la consulta por
   UDP
unsigned long sendNTPpacket(byte *address)
{
  UdpBytewise.begin(123);
  UdpBytewise.beginPacket(address, 123);
  UdpBytewise.write(B11100011);   // LI, Version, Mode
  UdpBytewise.write(0);     // Stratum
  UdpBytewise.write(6);  // Polling Interval
  UdpBytewise.write(0xEC); // Peer Clock Precision
  write_n(0, 8);     // Root Delay & Root Dispersion
  UdpBytewise.write(49);
  UdpBytewise.write(0x4E);
```

```
  UdpBytewise.write(49);
  UdpBytewise.write(52);
  write_n(0, 32); //Reference and time stamps
  UdpBytewise.endPacket();
}

//lee un Unisgned Long para UDP
unsigned long getUlong()
{
    unsigned long ulong = (unsigned long)UdpBytewise.read()
      << 24;
    ulong |= (unsigned long)UdpBytewise.read() << 16;
    ulong |= (unsigned long)UdpBytewise.read() << 8;
    ulong |= (unsigned long)UdpBytewise.read();
    return ulong;
}

//escribe bytes para UDP
void write_n(int what, int how_many)
{
  for( int i = 0; i < how_many; i++ )
    UdpBytewise.write(what);
}

//medida del sensor de humedad
void medida_humedad()
{

  //LECTURA DE VALORES EN LAS ENTRADAS ANALOGICAS
  //la lectura del dato en crudo (entero, de 0 a 1023) se
     obtiene con analogRead(pin)
  //tomamos 5 muestras del sensor para mejorar la
     estabilidad de las lecturas,
  //con una pequena pausa entre cada medida para permitir al
      conversor
  //analogico-digital recuperarse entre cada medida
  for (int i=0; i<5; i++)
  {
    analogRead(puertoH1);
    delay(delay_lectura);
    vhumedad[i] = analogRead(puertoH1);
  }

  //CALCULO DE VALORES MEDIOS Y CONVERSION A mV
```

```
  //inicializo a cero el acumulador
    mhumedad = 0.0;

  //sumamos las 5 lecturas
    for (int i=0; i<5; i++)
    {
      mhumedad = mhumedad + float(vhumedad[i]);
    }

  //Dividimos el valor sumado entre 5 para obtener la media.
  //y el resultado lo multiplicamos por 5000 (5V) y
     dividimos por 1024 para
  //convertirlo a mV
  mhumedad = ((mhumedad / 5.0)*5000.0)/1024.0;

  //CONVERSION A VALORES REALES, EN LAS UNIDADES
     CORRESPONDIENTES (%HR)
  //estas conversiones se obtienen de las ecuaciones
     indicadas en las hojas de caracteristicas
  //del componente.

  //En la humedad, los valores de salida normales del
     circuito son 0,8V para el 0% a 3,9V para el 100%
  //Tomamos la medida en mV y le restamos 800mv (voltaje de
     salida a 0%HR)
  //y el resultado lo dividimos por 31 (dado que el rango de
     salida del sensor, 800mV a 3900mV,
  //supone 3100mV de amplitud, y queremos convertir a %, por
     lo tanto dividimos entre 100, resultando 31)
  humedad = (mhumedad-800.0)/31;
}

void medida_temperatura()
{
  //lectura de temperatura de los sensores 1-wire
  sensors.requestTemperatures(); // Envia por el bus 1-wire
     el comando para leer temperaturas

  //almacena los valores recibidos de los sensores en las
     variables
  temperaturaT1 = sensors.getTempC(sensorTemp1);
  temperaturaT2 = sensors.getTempC(sensorTemp2);
}
```

```
void muestraSerie()
{
   //IMPRESION DE LOS VALORES POR EL PUERTO SERIE
    Serial.print("Temp 1: ");
    Serial.print(temperaturaT1);
    Serial.print("C; \t Temp 2: ");
    Serial.print(temperaturaT2);
    Serial.print("C; \t Humedad: ");
    Serial.print(humedad);
    Serial.println("%HR");
}


//funciones 1-wire
//Muestra por serie la direccion de un dispositivo
void printAddress(DeviceAddress deviceAddress)
{
  for (uint8_t i = 0; i < 8; i++)
  {
    // zero pad the address if necessary
    if (deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
  }
}
```

Parte IX

Anexo 3: Código web

Código PHP (parcial) correspondiente al servidor web. El resto del código se incluye en los descargables del proyecto.

Fichero **data.php**, generacion de gráficas mediante Open Flash Chart y JSON.

```
<?php
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

//WeatherLAN: Sensor de temperatura y humedad con arduino
```

```
//http://hfiel.es
//
// Copyright 2011 Hector Fiel - contacto@hfiel.es
//
//
// This file is part of WeatherLAN.
//
//    This program is free software: you can redistribute it
    and/or modify
//    it under the terms of the GNU General Public License
   as published by
//    the Free Software Foundation, either version 3 of the
   License, or
//    (at your option) any later version.
//
//    This program is distributed in the hope that it will
   be useful,
//    but WITHOUT ANY WARRANTY; without even the implied
   warranty of
//    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the
//    GNU General Public License for more details.
//
//    You should have received a copy of the GNU General
   Public License
//    along with this program.  If not, see <http://www.gnu.
   org/licenses/>.
//
//Fichero: data.php
//Version: 3 (version final)
//Descripcion: genera los datos JSON para crear los graficos
    de valores diarios
//mediante OFC, a partir del sensor seleccionado y la fecha
   indicada
//Las consultas SQL a la BBDD incluyen la conversion de hora
    UTC a hora local, segun la
//zona horaria definida en el fichero de configuracion
//Cuando se extraen los datos, se comprueba que haya
   continuidad en los registros
//y no se hayan producido cortes (se debe recibir un dato
   nuevo cada 5 minutos exactos,
//comenzando a las 00:00 de cada dia). En caso de detectar
   dos registros con una separacion
//mayor a 5 minutos, se indica en el titulo y se muestra un
```

```
   simbolo para reflejar el
//corte en la linea del grafico.
//Las divisiones del eje Y (temperatura y humedad) se
   calculan en funcion de los valores
//maximo y minimo encontrados en los datos
//Las divisiones del eje X (horas) se calculan en funcion
   del numero de registros recibidos
//para que siempre se pueda leer correctamente la etiqueta
   de la hora
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////



//si tenemos fecha y sensor seleccionados, los almacenamos.
//En caso contrario, paramos la ejecucion
if (isset($_GET['fecha']) and isset($_GET['ip']))
{
        $fecha=$_GET['fecha'];
        $ip=$_GET['ip'];
}
else
{
die();
}

//fichero de configuracion con los datos de BBDD y zona
   horaria
require_once "content/config.php";

//conectamos a la BBDD
mysql_connect ($servidor, $usuario, $clave);
mysql_select_db ($bbdd);

//incluimos la libreria de OFC
include 'php-ofc-library/open-flash-chart.php';

//consulta con los valores del sensor y la fecha
   seleccionada
//Para la seleccion de los datos, dado que fecha y hora se
   almacenan en UTC, primero los pasamos al horario de
   Europe/Madrid
```

```
//usando la funcion CONVERT_TZ. Para sacar los datos de hora
    convertimos el valor mediante la funcion TIME.
//Para la seleccion de los datos de la fecha concreta,
   convertimos mediante la funcion DATE
$consultasql="SELECT t1,t2,h1, TIME(CONVERT_TZ(fechahora,'
   UTC','".$zonahoraria."')) AS hora FROM medidas where ip
   ='".$ip."' and DATE(CONVERT_TZ(fechahora,'UTC','".
   $zonahoraria."'))='".$fecha."' ORDER BY hora";

//leemos los datos
$resultado = mysql_query ($consultasql);

//si no tenemos respuesta de la consulta terminamos la
   ejecucion mostrando el error
if (!$resultado) {
        die('Consulta fallida: ' . mysql_error());
}

//variables para poner el maximo y minimo en el eje Y
//el maximo se inicializa a -1000.0 y el minimo a 1000.0
//para que siempre haya un dato real mayor o menor
   respectivamente

$maximo=-1000.0;
$minimo=1000.0;

//variable para ver si se han perdido datos
//entre registros, inicializada a las 00:00
//se usa para comprobar que los datos se han recibido cada 5
    minutos
$horaanterior="00:00";
$horaactual="00:00";
//convertimos la cadena de texto con la hora anterior al
   formato tiempo
$horaanteriortime=strtotime($horaanterior);

//variable para indicar que hemos tenido cortes en el
   registro
$haycorte=0;

//calculamos el numero de resultados
$cuantos=mysql_num_rows($resultado);

//variables para almacenar los datos de cada sensor
```

```
$datos_t1=array();
$datos_t2=array();
$datos_h1=array();

//variable para la etiqueta del eje X
$etiqx=array();

//saco los datos de la BBDD en mi propio array
$arraydatos=array();
//mientras tenga datos, proceso cada fila
while($row = mysql_fetch_array($resultado))
{
 $arraydatos[]=$row;
}

//si tengo datos, proceso
if ($cuantos != 0)
{
        //datos del sensor de temperatura 1
        //(incluye las horas del eje x)

        //por cada elemento fila
        foreach($arraydatos as $numcol => $row)
        {
                //ponemos el valor de cada punto (
                   temperatura) en el array $val
                $val=floatval($row['t1']);
                //buscamos maximo o minimo
                if ($val<$minimo)
                {
                        $minimo=$val;
                }
                if ($val>$maximo)
                {
                        $maximo=$val;
                }
                //comprobamos si hemos tenido diferencia de
                   mas de 5 minutos
                $horaactual=date("H:i", (mktime(substr($row
                   ['hora'],0,2),substr($row['hora'],3,2),0,
                   substr($fecha,5,2),substr($fecha,8,2),
                   substr($fecha,0,4))));
                $horaactualtime=strtotime($horaactual);
                //obtenemos la diferencia en minutos
```

```
$difertime=($horaactualtime-
   $horaanteriortime)/60;
//guardamos la hora actual como anterior,
   para la siguiente vuelta
$horaanteriortime=$horaactualtime;
//si la diferencia es de 5 minutos, creamos
   la etiqueta de forma normal
if ($difertime==5 or ($horaactual=="00:00"))
{
        //ponemos la etiqueta para el eje X
           en $label
        $label[]=date("H:i", (mktime(substr(
           $row['hora'],0,2),substr($row['
           hora'],3,2),0,substr($fecha,5,2),
           substr($fecha,8,2),substr($fecha
           ,0,4))));
        //definimos el punto
        $d = new dot($val);
        ///ponemos en el array $datos_t1 los
            nuevos puntos, como toltip su
           valor, y con los estilos del
           punto
        $datos_t1[]=$d->size(3)->colour('#
           D02020')->tooltip( 'T1 #x_label#
           - #val# &#xB0;C' );
}
//si tenemos una separacion diferente a 5
   minutos, destacamos la etiqueta
if ($difertime!=5  and ($horaactual
   !="00:00"))
{
        $haycorte=1;
        $label[]="*".date("H:i", (mktime(
           substr($row['hora'],0,2),substr(
           $row['hora'],3,2),0,substr($fecha
           ,5,2),substr($fecha,8,2),substr(
           $fecha,0,4))));
        //definimos el punto como estrella
        $d = new bow($val);
        ///ponemos en el array $datos_t1 los
            nuevos puntos, como toltip su
           valor, y con los estilos del
           punto
        $datos_t1[]=$d->size(10)->halo_size
```

```
                    (2)->colour('#D02020')->tooltip(
                    'T1 #x_label# - #val# &#xB0;C' );
        }

}
//t2
foreach($arraydatos as $numcol => $row)
{
        //ponemos el valor de cada punto (
           temperatura) en el array $val
        $val=floatval($row['t2']);
        //buscamos maximo o minimo
        if ($val<$minimo)
        {
                $minimo=$val;
        }
        if ($val>$maximo)
        {
                $maximo=$val;
        }
        //comprobamos si hemos tenido diferencia de
           mas de 5 minutos
        $horaactual=date("H:i", (mktime(substr($row
           ['hora'],0,2),substr($row['hora'],3,2),0,
           substr($fecha,5,2),substr($fecha,8,2),
           substr($fecha,0,4))));
        $horaactualtime=strtotime($horaactual);
        //obtenemos la diferencia en minutos
        $difertime=($horaactualtime-
           $horaanteriortime)/60;
        //guardamos la hora actual como anterior,
           para la siguiente vuelta
        $horaanteriortime=$horaactualtime;
        //si la diferencia es de 5 minutos, creamos
           la etiqueta de forma normal
        if ($difertime==5 or ($horaactual=="00:00"))
        {
                //definimos el punto
                $d = new dot($val);
                ///ponemos en el array $datos_t1 los
                    nuevos puntos, como toltip su
                   valor, y con los estilos del
                   punto
                $datos_t2[]=$d->size(3)->colour('#
```

```
                D02020')->tooltip( 'T2 #x_label#
                - #val# &#xB0;C' );
        }
        //si tenemos una separacion diferente a 5
           minutos, destacamos la etiqueta
        if ($difertime!=5  and ($horaactual
           !="00:00"))
        {
                $haycorte=1;
                //definimos el punto como estrella
                $d = new bow($val);
                ///ponemos en el array $datos_t1 los
                    nuevos puntos, como toltip su
                   valor, y con los estilos del
                   punto
                $datos_t2[]=$d->size(10)->halo_size
                   (2)->colour('#D02020')->tooltip(
                   'T2 #x_label# - #val# &#xB0;C' );
        }
}
//h1
foreach($arraydatos as $numcol => $row)
{
        //ponemos el valor de cada punto (
           temperatura) en el array $val
        $val=floatval($row['h1']);
        //buscamos maximo o minimo
        if ($val<$minimo)
        {
                $minimo=$val;
        }
        if ($val>$maximo)
        {
                $maximo=$val;
        }
        //comprobamos si hemos tenido diferencia de
           mas de 5 minutos
        $horaactual=date("H:i", (mktime(substr($row
           ['hora'],0,2),substr($row['hora'],3,2),0,
           substr($fecha,5,2),substr($fecha,8,2),
           substr($fecha,0,4))));
        $horaactualtime=strtotime($horaactual);
        //obtenemos la diferencia en minutos
        $difertime=($horaactualtime-
```

```
                $horaanteriortime)/60;
            //guardamos la hora actual como anterior,
                para la siguiente vuelta
            $horaanteriortime=$horaactualtime;
            //si la diferencia es de 5 minutos, creamos
                la etiqueta de forma normal
            if ($difertime==5 or ($horaactual=="00:00"))
            {
                    //definimos el punto
                    $d = new dot($val);
                    ///ponemos en el array $datos_t1 los
                        nuevos puntos, como toltip su
                        valor, y con los estilos del
                        punto
                    $datos_h1[]=$d->size(3)->colour('#
                        D02020')->tooltip( 'H1 #x_label#
                        - #val# &#37;HR' );
            }
            //si tenemos una separacion diferente a 5
                minutos, destacamos la etiqueta
            if ($difertime!=5  and ($horaactual
                !="00:00"))
            {
                    $haycorte=1;
                    //definimos el punto como estrella
                    $d = new bow($val);
                    ///ponemos en el array $datos_t1 los
                        nuevos puntos, como toltip su
                        valor, y con los estilos del
                        punto
                    $datos_h1[]=$d->size(10)->halo_size
                        (2)->colour('#D02020')->tooltip(
                        'H1 #x_label# - #val# &#37;HR' );
            }
        }
}


$chart = new open_flash_chart();
if ($cuantos != 0)
{
        if ($haycorte==0)
        {
                $titulo="Temperaturas del sensor ".$ip." el
```

```
                    ".$fecha." (".$cuantos." registros)";
        }
        else
        {
                $titulo="Temperaturas del sensor ".$ip." el
                    ".$fecha." (".$cuantos." registros) AVISO
                    : CORTES EN EL REGISTRO";
        }
}
else
{
$titulo="No hay datos para la fecha seleccionada";
}
if ($haycorte==0)
{
        $chart->set_title( new title ($titulo));
}
else
{
        $charttitulo=new title ($titulo);
        $charttitulo->set_style("{font-size: 15px; color: #
            DF0101}");
        $chart->set_title( $charttitulo);
}

//definimos el nuevo elemento linea y asignamos los valores
if ($cuantos != 0)
{
$linea_t1 = new line();
$linea_t1->set_values( $datos_t1);
$linea_t1->set_colour('#FF0000'); //rojo
$linea_t1->set_key( "Temperatura Sensor 1", 10 );
$linea_t2 = new line();
$linea_t2->set_values($datos_t2);
$linea_t2->set_colour( '#347235' ); //verde
$linea_t2->set_key( "Temperatura Sensor 2", 10 );
$linea_h1 = new line();
$linea_h1->set_values($datos_h1);
$linea_h1->set_colour( '#0000FF' ); //azul
$linea_h1->set_key( "Humedad Sensor 1", 10 );
}
//eje X
$x_labels = new x_axis_labels();
$x_labels->set_labels( $label );
```

```
$x_axis = new x_axis();
$x_axis->set_labels($x_labels);

switch ($cuantos)
{
 case ($cuantos <= 30):
  $x_labels->set_steps(3);
  $x_axis->set_steps(1);
  break;
 case ($cuantos <= 80):
 $x_labels->set_steps(6);
 $x_axis->set_steps(3);
  break;
 case ($cuantos <= 140):
 $x_labels->set_steps(10);
 $x_axis->set_steps(5);
  break;
 case ($cuantos <= 200):
 $x_labels->set_steps(12);
 $x_axis->set_steps(6);
  break;
 case ($cuantos <= 260):
 $x_labels->set_steps(16);
 $x_axis->set_steps(8);
  break;
 case ($cuantos <= 300):
 $x_labels->set_steps(18);
 $x_axis->set_steps(9);
  break;
 case ($cuantos > 300):
 $x_labels->set_steps(30);
 $x_axis->set_steps(15);
  break;
}

// metemos linea en el grafico:
if ($cuantos != 0)
{
$chart->add_element( $linea_t1 );
$chart->add_element( $linea_t2 );
$chart->add_element( $linea_h1 );
}
//definimos eje y
```

```
$y_axis = new y_axis();
$y_axis->set_range( round($minimo) -5, round($maximo) +5, 10
    );
$y_legend = new y_legend( 'Temperatura (&#xB0;C)

   Humedad (&#37;HR)' );
$y_legend->set_style( '{font-size: 15px; color: #000000}' );
$chart->set_y_legend( $y_legend );
$x_legend = new x_legend( 'Horas' );
$x_legend->set_style( '{font-size: 15px; color: #000000}' );
$chart->set_x_legend( $x_legend );

//incluimos ejes en el grafico
$chart->add_y_axis( $y_axis );
$chart->set_x_axis( $x_axis );

echo $chart->toPrettyString();

?>
```

Parte X

Anexo 4: Código SQL de creación de tablas

Script SQL para la creación de tablas y campos. Fichero: **weatherlan_mysql_vacia.sql**

```
-- phpMyAdmin SQL Dump
-- version 3.2.4
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Tiempo de generacion: 17-06-2011 a las 13:10:24
-- Version del servidor: 5.1.41
-- Version de PHP: 5.3.1

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=
   @@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=
   @@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=
   @@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Base de datos: `sensor`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `medidas`
--

CREATE TABLE IF NOT EXISTS `medidas` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ip` varchar(15) COLLATE utf8_spanish_ci NOT NULL,
  `t1` float DEFAULT NULL,
  `t2` float DEFAULT NULL,
  `h1` float DEFAULT NULL,
  `fechahora` datetime NOT NULL DEFAULT '0000-00-00
     00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=
   utf8_spanish_ci AUTO_INCREMENT=4292 ;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `sensores`
--

CREATE TABLE IF NOT EXISTS `sensores` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ip` varchar(15) COLLATE utf8_spanish_ci NOT NULL,
  `nombre` varchar(30) COLLATE utf8_spanish_ci NOT NULL,
  `tz` varchar(50) COLLATE utf8_spanish_ci DEFAULT 'Europe/
     Madrid',
  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=
    utf8_spanish_ci AUTO_INCREMENT=6 ;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
    */;
/*!40101 SET CHARACTER_SET_RESULTS=
    @OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
    */;
```