Sensores Inalámbricos
Monitorización Y Control Remoto

Manuel Solera Ferrándiz
Ingeniería Técnica de Informática de Sistemas

Consultor: Jordi Bécares Ferrés

Fecha Entrega: Junio 2011

Resumen

El presente trabajo se circunscribe al área de sistemas empotrados (embedded system).

Basados en el sistema operativo TinyOS y lenguaje NesC para su programación.

Con estos podemos desarrollar una red de sensores inalámbricos que ofrecen una gran versatilidad a la hora de implementar distintas aplicaciones y usos.

El sistema aquí desarrollado pretende servir de base para la consecución de un sistema domótico más elaborado, aunque la utilidad final podría ser diversa.

Para ello se ha utilizado el mínimo de dos circuitos, motas, (para un sistema más “elaborado” necesitaríamos un mayor número de motas y otra gestión de las comunicaciones entre ellas) una servirá de estación base, otra sobre la que aplicaremos el mayor peso de la programación y que llamaremos “RemoteTemp”, procesa, gestiona y recolecta datos de los sensores, comunicándose de forma inalámbrica con la estación base.

RemoteTemp dispone de diversos modos de control de un actuador, alarmas y setpoints modificables; para que el trabajo sea más práctico y útil, se ha desarrollado un prototipo de actuador, conectado a la mota, energiza convenientemente unas bases de corriente de 220V.

Para la monitorización y control del sistema se ha desarrollado una aplicación que denominaremos “RemoteTemp - PC”, con esta, visualizaremos entre otras cosas, datos de los sensores, alarmas y configuración, pudiendo interactuar directamente con la mota.

Nuevamente, para que el sistema resulte más práctico, se ha desarrollado una segunda aplicación llamada “iRemoteTemp” que nos dará acceso al sistema mediante un navegador de internet en cualquier PC y particularmente desde telefonía móvil.

Índice de contenidos

Índice de ilustraciones

1. Introducción

El avance de la tecnología hace que surjan nuevos métodos, sistemas y formas de hacer las cosas, estos son capaces de cambiar nuestros hábitos y costumbres y en definitiva, mejorar nuestra calidad de vida.

La tecnología entre otras cosas, puede hacer más confortable nuestra vida, añadir un plus de seguridad, proporcionarnos información; la tecnología bien usada salva vidas, hace avanzar la sociedad.

1.1. Justificación

Un campo que ofrece un sin fin de posibilidades son los sistemas empotrados y una parcela de estos es la domótica y la inmótica, en el hogar, en los edificios, la industria, proporcionan un control y una gestión de recursos más “inteligente”.

Los sistemas empotrados, redes de sensores inalámbricos, posibilitan una instalación sin cableados, eliminando uno de los inconvenientes de este tipo de instalaciones, a la vez que dotan el sistema de cierta autonomía e “inteligencia”.

Es en este campo de la domótica, donde se centra el presente trabajo, el proyecto puede servir de base para un sistema más complejo y escalable y es extrapolable a otros muchos campos.

1.2. Descripción

El presente proyecto crea un sistema donde los sistemas empotrados, motas, recolectan una serie de lecturas de los sensores disponibles, procesan esta información, toman decisiones y se comunican de forma inalámbrica con una estación base que controla, supervisa y modifica distintos parámetros del sistema.

Las motas pueden actuar físicamente con un actuador para el control de distintos equipos; esta actuación se realiza en función de los distintos modos de control y setpoints establecidos que habilitarán también una serie de alarmas.

También se hace posible el control distante del sistema mediante internet y acceso web, telefonía móvil.

1.3. Objetivos

Desarrollar software para distintos componentes en torno a los sistemas empotrados; sistema de control remoto, monitorización y actuación de una red de sensores inalámbricos.

*** Mota.

-- Adquisición de lecturas de los sensores disponibles (temperatura, luminosidad, tensión de alimentación, sensor de efecto hall)

-- Disparo de alarmas según setpoints (alarmas de alta y baja temperatura, luminosidad, batería baja); otras alarmas que no requieren setpoitn al ser digitales, Hall (disparo sensor magnético) y Boot (aviso de un reset, arranque)

-- Control de un actuador según modo de control automático (temperatura, luminosidad, hall) con puntos de disparo configurables.

-- Cambio modo Aut/Man y control del actuador en modo manual mediante el pulsador USR.
-- Envío lecturas sensores, alarmas y estado actuador a intervalos regulares al software de monitorización y control.
-- Recepción y envío de mensajes desde y hacia el software de control, como Hello message, MAC message, mensajes de reconocimiento de alarmas, cambio de setpoints e intervalo de actualización de lecturas.
-- Reflejar distintas indicaciones mediante leds.
-- Identificación de la mota vía serial.
-- Control de un actuador físico.

*** Aplicación de monitorización y control (PC)
-- Visualización de lecturas sensores, alarmas, estado actuador y setpoints.
-- Requerir distintas actuaciones a la mota, cambio modo de control, operación directa del actuador en modo manual, mensaje Hello, mensaje MAC, modificación de setpoints y reconocimiento de alarmas “digitales” (Hall y Boot).

*** Aplicación de monitorización y control (WEB)
-- Visualización de lecturas sensores, alarmas, estado actuador.
-- Requerir distintas actuaciones a la mota, cambio modo de control, operación directa del actuador en modo manual y reconocimiento de alarmas “digitales” (Hall y Boot).

*** Actuador (hardware).
-- Diseño y construcción de un prototipo de actuador para conectar a la mota capaz de energizar unas tomas de corriente de 220V.

1.4. Enfoque

Se han estudiado las posibilidades de “sistemas empotrados” y de las motas disponibles.

Al disponer solo de dos motas, una tiene que servir de estación base (puente entre el PC y el resto de las motas o mota en este caso), el trabajo se ha centrado principalmente en la segunda mota.

Se requería un software de monitorización y control en PC, se ha optado por realizarlo en VB.NET, que posibilita cierta facilidad en el desarrollo del entorno gráfico; ha sido necesario estudiar la transmisión, estructura, de los mensajes para la comunicación con otros componentes, esto posibilita un mayor conocimiento y control sobre el desarrollo de la aplicación.

Para hacer más práctico el sistema, se pretendía un acceso remoto con navegador de internet y desde telefonía móvil.

Siguiendo con la filosofía de realizar un trabajo eminentemente práctico y útil, se decidió el desarrollo de un prototipo de actuador físico que puede ser activado por la mota.

Se han estudiado las posibilidades de llevar todo esto a cabo y se han marcado prioridades.

Se ha procurado no arrastrar errores y que todas las opciones y funciones fueran operativas según se iban implementando.

Durante el desarrollo se han tomado distintas notas para la posterior documentación.

El desarrollo de las aplicaciones, documentación y aprendizaje es continuo y se entrelazan para la depuración y mejora continua mientras dura el proyecto (desarrollo, test, corrección de errores, optimización).

1.5. Planificación

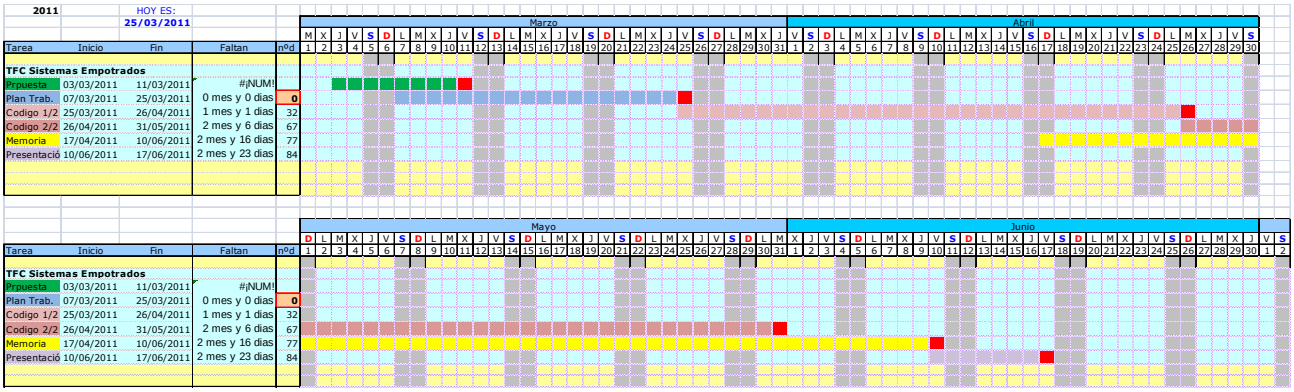


Fig. 1 - Planificación global trabajos

Para la planificación se marcaron unos objetivos básicos principales que se han modificado, ampliando, según se avanzaba en el conocimiento, posibilidades del sistema y disponibilidad de tiempo.

Se ha pretendido dejar cierto grado de libertad en el cumplimiento de los objetivos, desarrollados según prioridades y como último límite los plazos de entrega.

El desarrollo de la aplicación PC, ha ido evolucionando a la par que el desarrollo de la mota para ir verificando el correcto funcionamiento (desarrollo paralelo de diversos hitos).

Depuración soft mota con “testigos leds” y aplicación PC.

Tarea		Inicio	Final
Propuesta		03/03/2011	11/03/2011
Plan Trab.		07/03/2011	25/03/2011
Código 1/2		25/03/2011	26/04/2011
	Documentación	25/03/2011	10/04/2011
	Codificación Modo Manual/Automático	26/03/2011	30/03/2011
	Codificación sensores para funcionamiento Aut.	29/03/2011	06/04/2011
	Codificación alarmas	07/04/2011	11/04/2011
	Codificación transmisiones	09/04/2011	15/04/2011
	Test, depuración, correciones y optimización	10/04/2011	30/04/2011
	Desarrollo 1, aplicación PC	01/04/2011	25/04/2011

Código 2/2		26/04/2011	31/05/2011
	Documentación	26/04/2011	03/05/2011
	Codificación 2, Mota. Ver (1)	27/04/2011	27/05/2011
	Desarrollo 2, aplicación PC	27/04/2011	25/05/2011
	Test, depuración, correciones y optimización	30/04/2011	25/05/2011
	Desarrollo actuador	03/05/2011	18/05/2011
	Implementación base de datos	20/05/2011	24/05/2011
	Desarrollo aplicación Web	11/05/2011	25/05/2011
Memoria		17/04/2011	10/06/2011
Presentación		10/06/2011	17/06/2011

Fig. 2 - Planificación tareas

(1) Las novedades, modificaciones más destacables realizadas en la mota, RemoteTemp, en la segunda versión, son las siguientes:

- Implementado sensor Hall (HallEffect) Se establece una señal (hallReady) pasado un tiempo, de otro modo se dispara el evento hall fired al habilitar el sensor.
- Implementada alarma Hall
- Implementado recepción msg reconocimiento (ACK) alarma Hall
- Implementado control por efecto Hall
- Implementada alarma Boot
- Implementado recepción msg reconocimiento (ACK) alarma Boot
- Implementada recepción msg cambio modo de control
- Se reestructura números de mensaje (50)
- Se añade mensajes para cambio Aut/Man y conexión/desconexión actuador.
- Implementación y procesamiento recepción de mensaje para cambio de configuración (en configuration).
- Implementación de la interfaz "RadioCou" para implementar setUpdateInterval.
- Implementación ActuatorMS para conectar actuador exterior, se utiliza GPIO5_CON (Pin 21), accesible desde el conector EXP_CON (JB9) Pin 15, (masa Pin 6)
- Modificaciones para controlar actuador exterior.
- Se modifica gestión Radio start en RadioCou (powerRadio.startDone)
- Se implementa Watchdog en RadioCouP, se establece un timer exclusivo para este, de esta forma se puede aumentar el tiempo del ciclo "process" (más información en el apartado Watchdog).
- Se ponen "semáforos" en MacSender y Configuratión en combinación con sendDone para no volver a enviar mientras se está en proceso de envío.
- MacSenderP, se implementa una nueva funcionalidad, mediante “Printf”, envío por serial de identificación de la mota en el inicio.

1.6. Recursos empleados

1.6.1. Software

-- El S.O. utilizado en el PC es Windows Vista Home Premium.

-- TinyOS, S.O. utilizado por las motas.

-- Utilizamos Cygwin para compatibilizar el sistema con Windows.

-- Eclipse Versión: 3.6.2, IDE utilizado para el desarrollo del software de las motas.

-- Desde Eclipse podemos también realizar la compilación o bien con el comando "make" desde la shell de Cygwin, ejemplo:

$ cd C:/cygwin/opt/tinyos-2.x/apps/BaseStation

$ make cou24

-- Para cargar la mota utilizaremos "meshprog", teniendo en cuenta el puerto, en este caso COM6 (ttyS5), ejemplo:

$ meshprog -t/dev/ttyS5 -f /cygdrive/c/EclipseTinyOS/RemoteTemp/build/cou24/main.srec

-- Microsoft Visual Basic 2008 Express de Visual Studio 2008 para el desarrollo de la aplicación de control y supervisión de las motas.

-- La conexión PC con la mota base la realizaremos con "SerialForwarder", podemos invocar este desde la Shell con:

$ java net.tinyos.sf.SerialForwarder -port 9002 -comm serial@com6:19200

-- Podemos ver los paquetes en Hex con Listen, ejemplo:

$ java net.tinyos.tools.Listen

(Para salir pulsar: <Ctrl + c>)

serial@COM6:19200: resynchronising

00 FF FF 00 01 0B 22 28 01 01 59 00 0B 02 04 0A 00 26 51

00 FF FF 00 01 0B 22 28 01 01 5A 00 0B 02 04 0A 00 26 52

00 FF FF 00 01 0B 22 28 01 01 59 00 0B 02 04 0A 00 26 53

-- Para el estudio de las tramas de transmisión se ha utilizado también el analizador de protocolo de red, Wireshark, Versión 1.4.4.

-- Microsoft Excel, para generar tablas de conversión, ADC-Vin-unidades de ingeniería.

-- Para averiguar las credenciales de una mota en concreto, conectándola por serial, podemos visualizarlo con PrintfClient desde Cygwin en el arranque de la mota.

-- Microsoft SQL Server 2008, servidor de bases de datos, se ha creado la base de datos DBRemoteTemp.

-- Microsoft Visual Web Developer 2008 Express Edition, para el desarrollo de una aplicación web (iRemoteTemp)

-- Internet Information Server (IIS) habilitado para acceso web a iRemoteTemp.

-- Internet Explorer 7, Safari 5.0.3, son los navegadores en los que se ha probado iRemoteTemp desde el PC, además de Safari en un móvil iPhone 4.

1.6.2. Hardware

-- PC

Procesador:	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz 2.39 GHz
Memoria (RAM):	4,00 GB
Tipo de sistema:	Sistema operativo de 32 bits

-- Dos motas COU 1_2 24 A2

Fig. 3 - Mota COU

-- Móvil iPhone 4.

Fig. 4 - iPhone 4

-- Actuador, se ha diseñado y producido un prototipo de actuador para conectar a la mota.

Fig. 5 - Actuador

1.7. Productos obtenidos

1.7.1. RemoteTemp (mota)

RemoteTemp (en mota), captura, procesa, realiza distintas acciones y se comunica de forma inalámbrica con el software base de monitorización y control.

1.7.2. BaseStation (mota)

Una mota cargada con “BaseStation” hace de puente entre el PC, por un puerto USB, y la red de sensores.

1.7.3. RemoteTemp (PC)

Software de monitorización y control del sistema.

1.7.4. iRemoteTemp (web)

iRemoteTemp (web) posibilita el acceso distante al sistema mediante navegador web y telefonía móvil.

1.7.5. Actuador (hardware)

Proporciona a la mota RemoteTemp, el control, accionamiento, de distintos elementos externos (calefacción, iluminación, motores, etc.)

1.8. Descripción de otros capítulos

A continuación se comentará la tecnología usada y los productos obtenidos con más detalle.

2. Antecedentes

2.1. Estado del arte

Los sistemas empotrados (embedded system) conforman un dispositivo informático, hardware y software, que operan frecuentemente en tiempo real.
Tratan de tener una gestión eficiente de unos recursos limitados, reducen tamaño, costes y consumo energético que ha de ser gestionado de forma eficiente, los avances tecnológicos y la producción en masa debido a la creciente demanda, hacen esto posible.
Los sistemas empotrados (motas) pueden conformar redes de sensores inalámbricos (WSN), funcionando de forma autónoma, la red, las comunicaciones, pueden ser autogestionadas por ellos mismos.
Estos sistemas, como tantos otros, comenzaron siendo investigaciones militares.

Esta es una tecnología en auge, su reducido tamaño y versatilidad, lo hacen aptos para la utilización en un sin fin de dispositivos y aplicaciones, dotando a estos de una potencia de cálculo y procesamiento en un reducido tamaño, además de un reducido consumo energético; si además disponen de comunicación inalámbrica, las posibilidades y campos de aplicación se multiplican posibilitando un fácil despliegue de redes con distintas topologías y protocolos de comunicación.

Los campos de aplicación y las posibilidades son innumerables como ya hemos dicho, por citar algunas:
- Domótica e inmótica.
- Medicina (monitorización de pacientes)
- Alerta temprana de incendios (despliegue de red de sensores en un bosque)
- Robótica.
- Aplicaciones militares.
- Automoción.
- Industria (telemetría, control de procesos).
- Seguridad.
- Agricultura.
- Telecomunicaciones.
- Electrónica de consumo.

Los sistemas empotrados admiten distintos sistemas operativos (TinyOS, LiteOS, MicroC/OS-II, eCos) y lenguajes de programación (ensamblador, C/C++, NesC, Basic).

TinyOS, basado en componentes para redes de sensores inalámbricas, es un sistema operativo de código abierto, está desarrollado por un consorcio liderado por la Universidad de California en Berkley en cooperación con Intel Research.

Este es el S.O. que utilizan las motas en el presente trabajo junto con NesC y protocolo de comunicación ZigBee, basado en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (WPAN).

NesC, es una variación del lenguaje C, adaptado a las características de los sistemas empotrados y utilizado con TinyOS.
NesC es un lenguaje orientado a componentes y está especialmente diseñado para programar aplicaciones sobre redes de sensores.
Se estructura en componentes, los componentes están divididos en tres partes: Configuration, Implementation, Module; estos componentes se conectan (wiring) con otros componentes, se comunican mediante una interfaz que define métodos (*commands* y *events*), los componentes proveen o usan las interfaces.
En los módulos, se programan, se implementan, las acciones a llevar a cabo.

2.2. Estudio de mercado

Los sistemas de control en el mercado son numerosos, no son tantos los que utilizan redes de sensores inalámbricos (WSN), sin embargo, existe una creciente demanda de estos dispositivos y sistemas.
Los avances tecnológicos, reducen cada vez más el tamaño, consumo, costes de producción, aumentan potencia, rendimiento y funcionalidades, esto confiere a los sistemas empotrados grandes posibilidades y se augura un crecimiento exponencial.
Algunas firmas dedicadas al desarrollo de sistemas embebidos: Crossbow, Ember, Sentlla, Intel, Sun Microsystems, Nano-RK, BTNode.

3. Descripción funcional

3.1. Sistema total

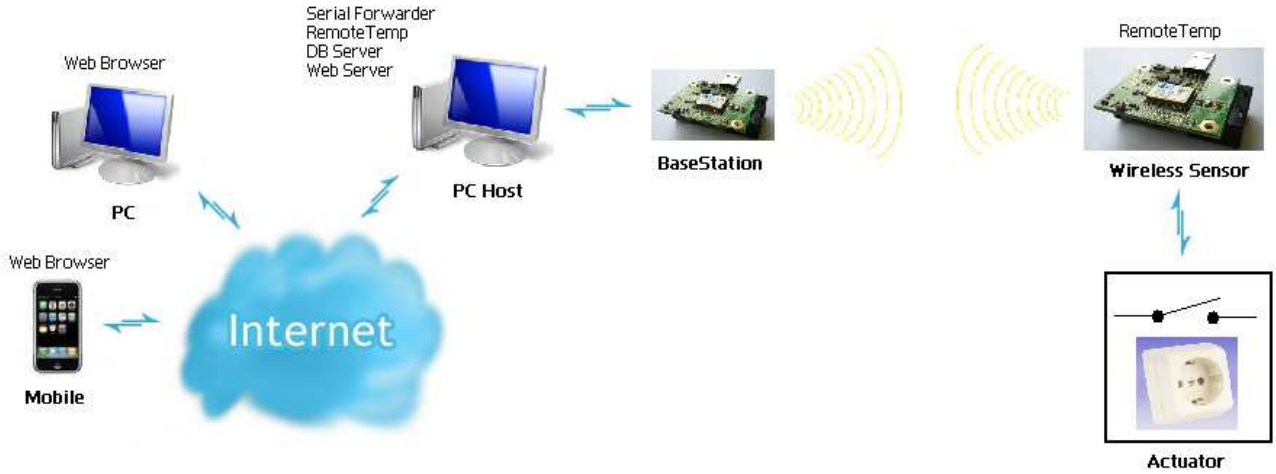


Fig. 6 - Descripción gráfica del sistema en conjunto

Se han desarrollado tres aplicaciones, llamadas “RemoteTemp” para PC en VB.net que realiza la supervisión y control de la mota, la aplicación implementada en la mota “RemoteTemp” y otra aplicación, “iRemoteTemp” para acceso remoto mediante navegador web, además, se ha implementado una base de datos, servidor de base de datos y servidor web junto con la elaboración de un actuador real.
A continuación vemos los distintos componentes y como se interrelacionan.

3.2. PC

3.2.1. RemoteTemp (PC)

La aplicación RemoteTemp (en PC) se conecta a la Mota Base a través de la aplicación “SerialForwarder, gráficamente:

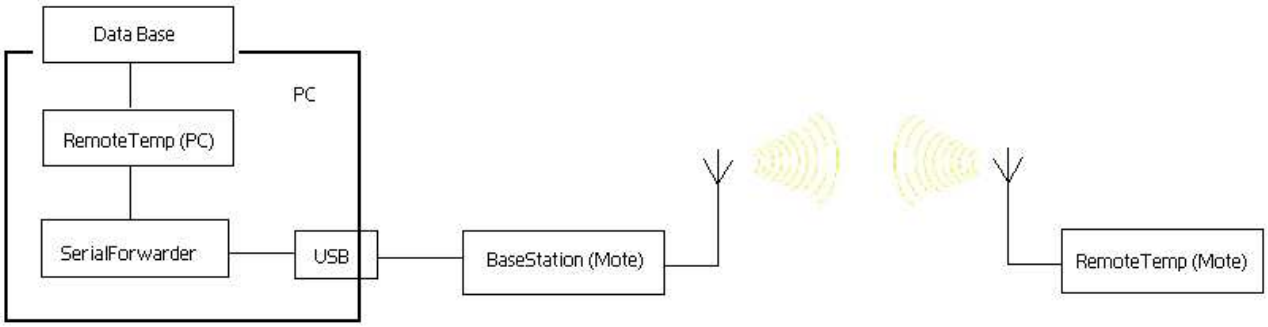


Fig. 7 - Conexiones RemoteTemp PC

También nos podríamos conectar a través de internet utilizando una IP pública.
La base de datos, el servidor de bases de datos, podrían estar localizados en distintos sitios:

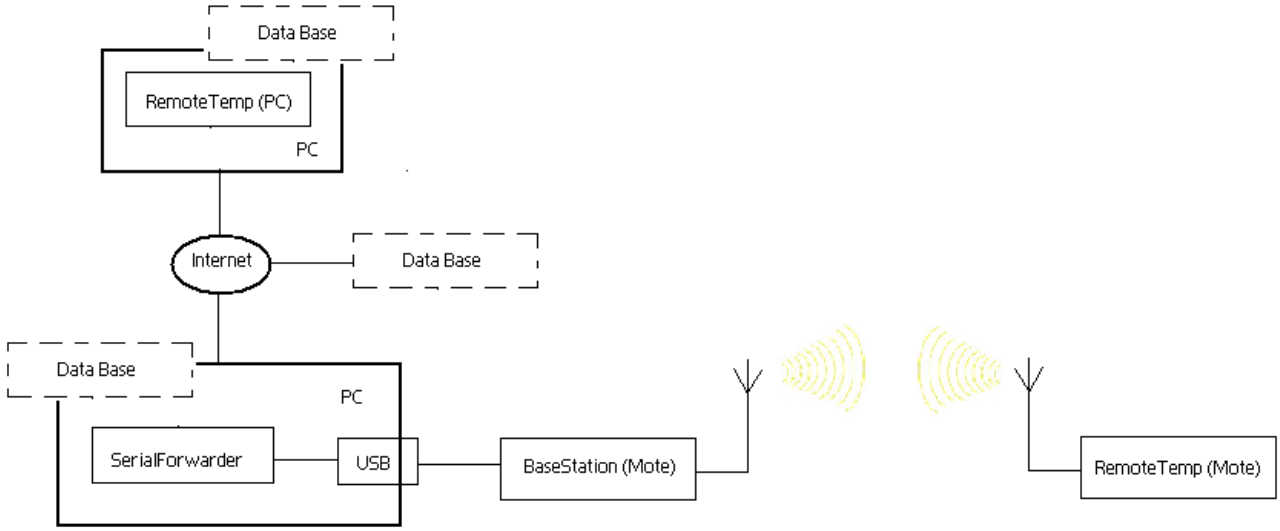


Fig. 8 - Localización componentes sistema

La aplicación RemoteTemp en PC, recibe y visualiza periódicamente las lecturas de los sensores así como las alarmas y estado del actuador, también solicita y visualiza mensajes como “SayHello, MAC y parámetros de configuración.

Para ayudar al estudio y desarrollo del sistema, esta aplicación visualiza distintos mensajes RAW (“en crudo”), intentos de envío/recepción, longitud de mensajes recibidos.

Esta aplicación se conecta con una base de datos para guardar un log de los sensores y atender las peticiones externas que llegan desde “web”.

3.2.2. iRemoteTemp (web)

iRemoteTemp (web) visualiza sensores, alarmas, estado actuador, puede reconocer alarma “Hall”, alarma “Boot”, modificar modo de control y conectar/desconectar el actuador en modo manual.

El sistema dispone de un servidor web conectado a una base de datos que “intermedia” entre el servidor web y RemoteTemp:

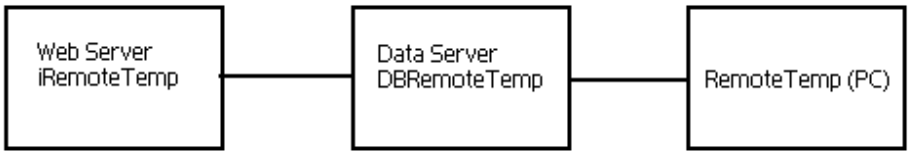


Fig. 9 - Comunicaciones: iRemoteTemp - DBRemoteTemp - RemoteTemp (PC)

3.3. Mota

3.3.1. BaseStation (mota)

Una mota cargada con “BaseStation” hace de puente entre el PC por un puerto USB y la red de sensores.

3.3.2. RemoteTemp (mota)

La aplicación RemoteTemp (en mota) hace uso de los distintos sensores para controlar un actuador que es conectado o desconectado en función de los puntos de actuación prefijados y el sensor seleccionado (temperatura, luminosidad, efecto hall), este control se realiza cuando la mota está en modo automático, en modo manual podemos conectar/desconectar el actuador a voluntad.

El cambio de modo Aut/Man y la conexión/desconexión del actuador en modo manual, se puede realizar mediante el pulsador USR (además de mediante el software de las aplicaciones) quedando reflejadas estas actuaciones por distintos leds.

Se dispone también de unas alarmas que disparan los distintos sensores, con unos puntos de disparos prefijados por defecto (modificables mediante soft de control); el estado de alarma queda reflejado también por un led.

Periódicamente se leen los sensores y se envía la información de estos sensores, junto con el estado de las alarmas y el actuador, a la aplicación de control en el PC, (este periodo es modificable mediante soft de control).

La mota responde con un mensaje “SayHello” cuando recibe un mensaje “SayHello”.
Así mismo la mota envía un mensaje informando de su MAC o un mensaje informando de los parámetros de configuración (alarmas, setpoints, intervalo de actualización) cuando le es requerido.
Otra opción es el envío por serial de la identidad de la mota en el arranque.

3.4. Actuador (hardware)

Para completar el sistema y dotarlo de una utilidad más practica y real se ha diseñado y elaborado un prototipo de actuador que se conecta a la mota y proporciona 220V. en unas tomas de corriente para la actuación de distintos elementos.

4. Descripción detallada

La mota base irá conectada a un puerto USB, esta realizará las comunicaciones entre el resto de motas y el PC, se iniciará Serialforwarder para hacer de puente entre la aplicación RemoteTemp PC y la mota base, ejecutaremos RemoteTemp PC y conectaremos con Serialforwarder.

De otro lado tendremos funcionando una mota con RemoteTemp, conectada a un actuador que proporciona 220V. para operar distintos elementos (iluminación, calefacción, alarmas, etc.).

Por otro lado, RemoteTemp accede a la base de datos DBRemoteTemp alojada en un servidor, para registrar un log de los mensajes recibidos con los distintos valores de los sensores, id, contador, alarmas, estado actuador y fecha/hora.
También accede a la BD para leer y ejecutar los comandos recibidos desde iRemoteTemp (acceso al sistema mediante aplicación web)

iRemoteTemp accede a la BD para proporcionar las lecturas correspondientes y registrar las órdenes que se desean ejecutar.
Internet Information Server (IIS) brindará acceso a iRemoteTemp desde internet.

Esta es la estructura utilizada, no obstante, un servidor web, el DB server y RemoteTemp podrían estar localizados en sitios distintos y distantes.

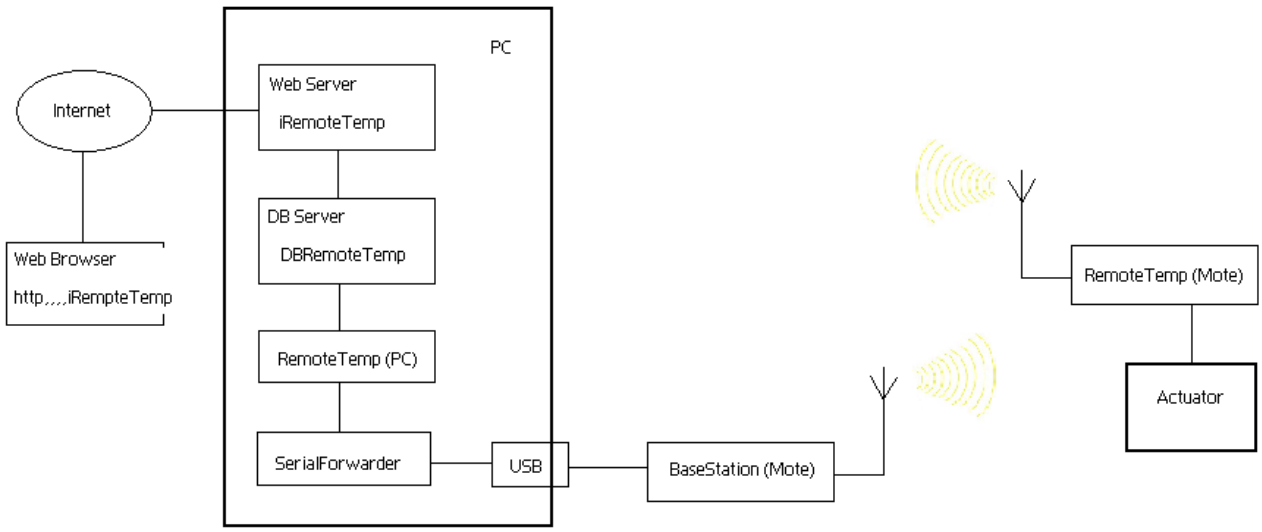


Fig. 10 - Sistema, componentes, conexión

4.1. BaseStation (mota)

Conectada a un puerto USB y cargada con la aplicación “BaseStation”, realiza las comunicaciones entre el resto de motas y el PC, esta mota es alimentada directamente por USB.

Según reza en el README.txt de BaseStation, la indicación de los leds es la siguiente:
The LEDS are programmed to toggle as follows:

RED Toggle	- Message bridged from serial to radio
GREEN Toggle	- Message bridged from radio to serial
YELLOW/BLUE Toggle	- Dropped message due to queue overflow in either direction

Debido a la configuración hardware de nuestra mota, los leds se comportan de la siguiente forma:
Led Rojo, conmuta con el envío de mensajes desde el puerto serie a la radio.
Led Ámbar, conmuta con el envío de mensajes desde la radio al puerto serie.
Led Verde, perdida de mensajes por desbordamiento en cualquier sentido.

4.2. RemoteTemp (mota)

Esta mota, cargada con nuestra aplicación “RemoteTemp” (versión 2), se comunicará de forma inalámbrica (radio 2.4 GHz) con el resto del sistema y puede servir de base para la programación de un sistema domótico más complejo, con más motas, sensores y actuadores.
Puede controlar de forma automática o manual un actuador, informa del estado de los sensores, actuador alarmas y configuración.

4.2.1. Funcionamiento

El funcionamiento, la operatividad que tiene implementada es la siguiente:
Al arrancar se carga con unos valores de configuración por defecto para alarmas y setpoints de control además de un intervalo de actualización de las lecturas de sensores.
Al arrancar se encontrará en modo Manual con el actuador desconectado y modo control Photo si pasamos a Automático.

Como se ha señalado, dispone de dos modos de funcionamiento, automático y manual (indicado por el led ámbar).
En modo manual (led ámbar fijo) podemos conectar/desconectar el actuador (indicado por el led verde) mediante una breve pulsación del botón USR.
Una pulsación mayor de unos dos segundos del pulsador USR pasará el sistema a modo automático (led ámbar intermitente), en este modo, el actuador será controlado según los puntos prefijados (setpoints) del sensor seleccionado en ese momento (Ctl. Temp., Ctl. Photo, Ctl Hall).
El modo control Hall no tiene setpoit, es digital y conectará el actuador al activarse mediante un elemento magnético, permaneciendo el actuador y la alarma activados hasta ser reconocidos.
Al pasar de modo automático a manual, el actuador permanecerá en el último estado que se encontraba en modo automático, ya sea conectado o desconectado.

El led rojo nos indica que existe alguna alarma, disparada a partir de los puntos de alarma prefijados.

- Leds

Indicaciones leds:

LED	ON	OFF	INTERMITENTE
ROJO	Alarma	Sin Alarmas	---
ÁMBAR	Modo Manual	---	Modo Automático (1Hz)
VERDE	Actuador Conectado	Actuador Desconectado	---

Fig. 11 - Indicaciones leds, mota RemoteTemp

- Pulsador USR

Acciones pulsador USR:

Pulsador USR	Modo Automático	Modo Manual
> 2 seg.	Cambia a modo Man.	Cambia a modo Aut.
< 2 seg.	---	Conmuta Actuador

Fig. 12 - Acciones pulsador USR, mota RemoteTemp

- Alarmas - SetPoints

Las alarmas prefijadas son:

- Alarma de alta temperatura.
- Alarma de baja temperatura.
- Alarma de alta luminosidad.
- Alarma de baja luminosidad.
- Alarma de baja batería.
- Alarma Hall (nueva en esta versión)
- Alarma Boot (nueva en esta versión, indica que ha habido un reset, arranque)

Los setpoints prefijadas para el control automático son:

- Setpoint alta temperatura.
- Setpoint baja temperatura.
- Setpoint alta luminosidad.
- Setpoint baja luminosidad.

Más el setpoint:

- Setpoint intervalo de actualización (sensores)

En modo automático, la mota controlará el actuador para tratar de mantener las variables dentro de los setpoints, por ejemplo, al llegar al valor del setpoint baja temperatura, se conectaría el actuador para por ejemplo producir calor y se desconectaría al llegar al setpoint alta temperatura.

La mota dispone también de un intervalo de actualización prefijado por defecto (modificable desde RemoteTemp PC), de forma regular realiza un escáner de los sensores para obtener un valor actualizado.
Estos valores actualizados son utilizados por distintos componentes de la mota para realizar el control automático si procede y chequear las posibles alarmas.

Los valores de los sensores junto con las alarmas, el modo Aut/Man, estado del actuador (aut/man, conectado/desconectado) que incluye también el modo de control (temperatura, luminosidad, hall), y un contador de mensajes, son enviados periódicamente con este mismo intervalo, mediante un mensaje a la aplicación de control del PC RemoteTemp (Mensaje tipo 40).

- Mensaje tipo 40 (sensores)

La estructura de este mensaje es la siguiente (mensaje tipo 40):

Sincronismo	1 Byte
Dirección destino	2 Bytes (broadcast: ff ff)
Dirección origen	2 Bytes
Longitud mensaje (payload)	1 Byte
ID grupo	1 Byte
Tipo mensaje	1 Byte

Payload:

ID Mota	*2 Bytes*
Temperatura	*2 Bytes*
Luminosidad	*2 Bytes*
Batería	*2 Bytes*
Actuador	*1 Byte*
Alarmas	*1 Byte*
Contador	*2 Bytes*

El byte de alarmas esta codificado de la siguiente forma:

/ *** alarmData ****

* *bit0= alarmHighTemp*

* *bit1= alarmLowTemp*

* *bit2= alarmHighPhoto*

** bit3= alarmLowPhoto*
** bit4= alarmLowBat*
** bit5= alarmHall*
***
** bit7= alarmBoot*
**/*

El byte de actuador esta codificado de la siguiente forma:
/ *** actuator ****
** bit0= Aut/Man (1= Automatic, 0= Manual))*
** bit1= (1= Connected, 0= Disconnected)*
** bit2= 1 Temperature control*
** bit3= 1 Photo control*
** bit4= 1 Hall control*
**/*

- Otros tipos de mensaje

Además del envío periódico de este mensaje la mota envía otros tipos de mensaje en respuesta a los requerimientos recibidos desde la aplicación de supervisión y control del PC.

Al recibir un mensaje tipo 41 (Hello Message) responde con otro mensaje “Hello Message”.
El Payload del mensaje es el siguiente:

ID Mota	*1 Byte*
Contador	*1 Byte*

(Para ahorrar recursos, aunque se contempla un ID de mota de dos bytes en la actual versión de Tinyos, nos será suficiente con 1 byte, hasta 256 motas.)

Al recibir un mensaje tipo 50 se le requerirá a la mota uno de los siguientes mensajes indicados en el payload, ejemplo:
“Requerimiento” 51 (MAC mensaje). Acción: la mota responde obteniendo su MAC y enviándola en un mensaje.
“Requerimiento” 52 (configuración). Acción: la mota enviará un mensaje con los parámetros de configuración actuales, puntos de alarma, setpoints e intervalo de actualización.

Listado de códigos de mensaje que se pueden enviar a la mota requiriendo una acción:

```
AM_REQUESTMSG                = 50,        // solicita una acción, los siguientes, son opciones de solicitud
    AM_MACMSG                = 51,        // solicita envío MAC
    AM_CONFIGMSG             = 52,        // solicita envío configuración
    AM_HALLACKMSG            = 53,        // reconocimiento alarma Hall
    AM_BOOTACKMSG            = 54,        // reconocimiento alarma Boot
    AM_CTLTEMPMSG            = 55,        // selecciona control por temperatura
    AM_CTLPHOTOMSG           = 56,        // selecciona control por sensor lumínico
    AM_CTLHALLMSG            = 57,        // selecciona control por hall
    AM_AUTMANMSG             = 58,        // cambio modo Aut/Man
    AM_ACTUATORTOGGLEMSG     = 59,        // cambio estado actuador ON/OFF

AM_SETCONFIGMSG              = 60,        // mensaje cambio de configuración, los siguientes, son opciones de cambio
    AM_SETCONFIGALARMMSG     = 61,        // cambio configuración alarmas
    AM_SETCONFIGSETPOINTSMSG = 62,        // cambio configuración Set Points
```

Además disponemos de los siguientes mensajes ya mencionados:

```
AM_SENSORSMSG  = 40,  // código identificativo (tipo msg) enviado por la mota para indicar el envío del mensaje (valor sensores, alarmas, estado actuador, contador)
AM_HELLOMSG    = 41   // código en envío y recepción de Hello mensaje
```

Todos ellos con una estructura definida en “RemoteTemp.h”

- Función identificación Mota

Una nueva funcionalidad añadida en esta versión, es la posibilidad de averiguar las “credenciales” de la mota, para ello se ha dispuesto que en el arranque envíe por serial unas frases mostrándonos su identificación, esta funcionalidad se sirve de “printf”.

El proceso para llevarlo a cabo es el siguiente:

1.- Conectar la mota a identificar en el puerto USB correspondiente.

2.- Esperar unos segundos (para que después “PrintfClient” la detecte).

3.- En la shell de Cygwin ejecutar “PrintfClient” (para salir pulsar: <Ctrl + c>)

4.- Pulsar el botón reset de la mota.

El output obtenido es como sigue:

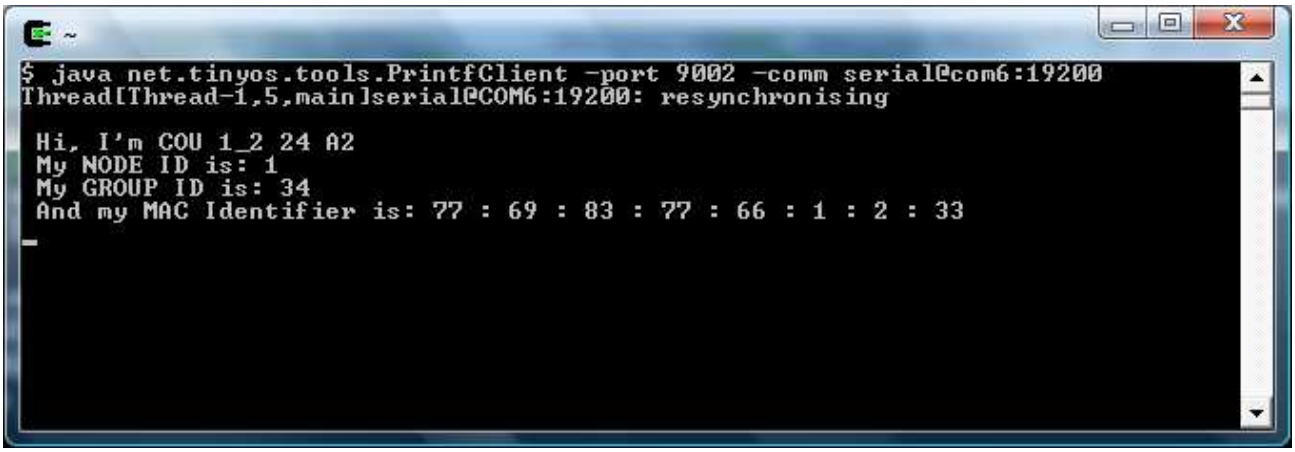


Fig. 13 - PrintfClient, output identificación mota

- Watchdog

También se ha dotado de watchdog a la aplicación.
Este realizará un reset de la mota si se produce un bloqueo de esta.

Watchdog dispone de un timer independiente, en el caso del micro que utilizamos (ATmega 1281), este puede ser configurado para ser "disparado" únicamente con los siguientes tiempos:
16, 32, 64ms y 0.125, 0.25, 0.5, 1, 2, 4, 8 seg.
Deberemos restear este timer antes de que se dispare y realice un reset a la mota.

La disposición del reset del watchdog timer sería conveniente instalarla en un punto que se repitiera cíclicamente, de esta forma detectaríamos también por ejemplo la entrada en un bucle sin fin (aunque si esto sucediera habría que revisar el código), en el caso de RemoteTemp, cíclicamente enviamos un mensaje con el estado de sensores y otros, pero puesto que este ciclo se puede repetir, si así lo deseamos, a intervalos mayores de 8 seg (8 seg.es el tiempo máximo programable en timer del watchdog), se ha optado por habilitar un timer que reseteará el watchdog timer cada 4 seg. si no se produce un bloqueo de la mota.

4.2.2. Componentes

La aplicación está compuesta de módulos (terminados en P), configuraciones (terminados en C), interfaces y la inclusión de ficheros de cabecera (.h)

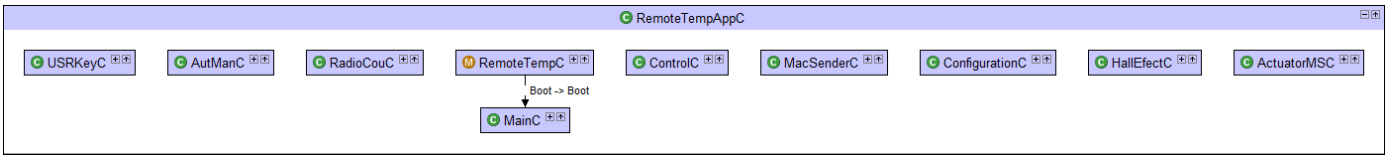


Fig. 14 - Componentes RemoteTemp

Los distintos componentes utilizados en RemoteTemp son:

-- ActuatorMS, ActuatorMSC, ActuatorMSP.
Controla un actuador real externo.
Provee la interfaz con:

actuatorMS0On();

actuatorMS0Off();

actuatorMS0Toggle();

Disponemos así del control de una salida digital.
Se utiliza GPIO5_CON (Pin 21), accesible desde el conector EXP_CON (JB9) Pin 15, (masa Pin 6) para conectar el actuador a la mota.

-- AutMan, AutManC, AutManP.
Mide el tiempo que ha sido pulsado el botón USR para realizar el cambio Aut/Man o conexión/desconexión del actuador, cuando se ha producido uno de estos cambios lanza un “signal” que es recogido por un event de Control.

-- Configuration, ConfigurationC, ConfigurationP.
Gestiona los parámetros de configuración (alarmas, setpoints, intervalo de actualización) que pasa a los módulos que lo requieren como Control.
Envía un mensaje con los parámetros de configuración cuando le es requerido.

-- Control, ControlC, ControlP.
Se encarga del control del actuador y gestión de alarmas, recibe los valores actualizados de los sensores de RadioCOU y los cambios realizados mediante el pulsador USR de AutMan.
Envía el estado de alarma y actuador a RadioCOU para su inclusión en el mensaje periódico.
La configuración de los parámetros de alarma y setpoints es recibida de Configuration.
Implementa y gestiona las alarmas Hall y Boot (nos indica que se ha producido un reset, arranque)

-- HallEfectC, HallEfectP.
Implementa sensor Hall (HallEffect). Se establece una señal (hallReady) pasado un tiempo, de otro modo se dispara el evento hall fired al habilitar el sensor.
Cuando se dispara, se lo comunica a Control “call Control.setCurrentHall(TRUE);”

-- MacSender, MacSenderC, MacSenderP.
Obtiene la MAC de la mota y envía un mensaje con esta cuando se le requiere.
Mediante “Printf” envía un mensaje por serial en el inicio informándonos de sus “credenciales”.

-- RadioCou, RadioCouC, RadioCouP.
Basado en SensorReporter de “blinkCou”, modificado y ampliado para cubrir las necesidades requeridas, “conecta” la radio, chequea los sensores cada cierto intervalo de tiempo y pasa estos valores para ser procesados a ”Control”, recibe el estado de alarma y actuador de “Control” y envía periódicamente un mensaje (tipo 40) con el estado de los sensores, alarmas, actuador (estructura del mensaje tipo 40, pág. 21).
Se encarga de recibir y contestar los mensajes “HelloMsg”.
Se encarga de recibir también el resto de mensajes (anteriormente descritos) que llegan y realizar las llamadas pertinentes según tipo de mensaje.
Por describir alguno:
Al recibir la solicitud para obtener la MAC de la mota, realizará una llamada a MacSender para que se encargue de procesar la solicitud.
Al recibir la solicitud de lectura de configuración de la mota, realizará una llamada a Configuration para que se encargue de este requerimiento.
Implementa, provee “setUpdateInterval” en esta segunda versión, se ha añadido RadioCou (interfaz).
Implementa Watchdog.

-- RemoteTempAppC, RemoteTempC.
El Main de la aplicación.

-- USRKeyC
Provee las interfaces Get y Notify del pulsador USR que serán utilizadas por AutMan para la gestión del pulsador USR.

-- RemoteTemp.h
Este fichero está incluido en la mayoría de los componentes, define constantes y la estructuras de los mensajes.

4.3. RemoteTemp PC

La aplicación de control y supervisión desde el PC ha sido realizada en VB.NET con Microsoft Visual Basic 2008 Express de Visual Studio 2008, las dos vistas siguientes pertenecen a este entorno de desarrollo.

(RemoteTemp Versión 1)

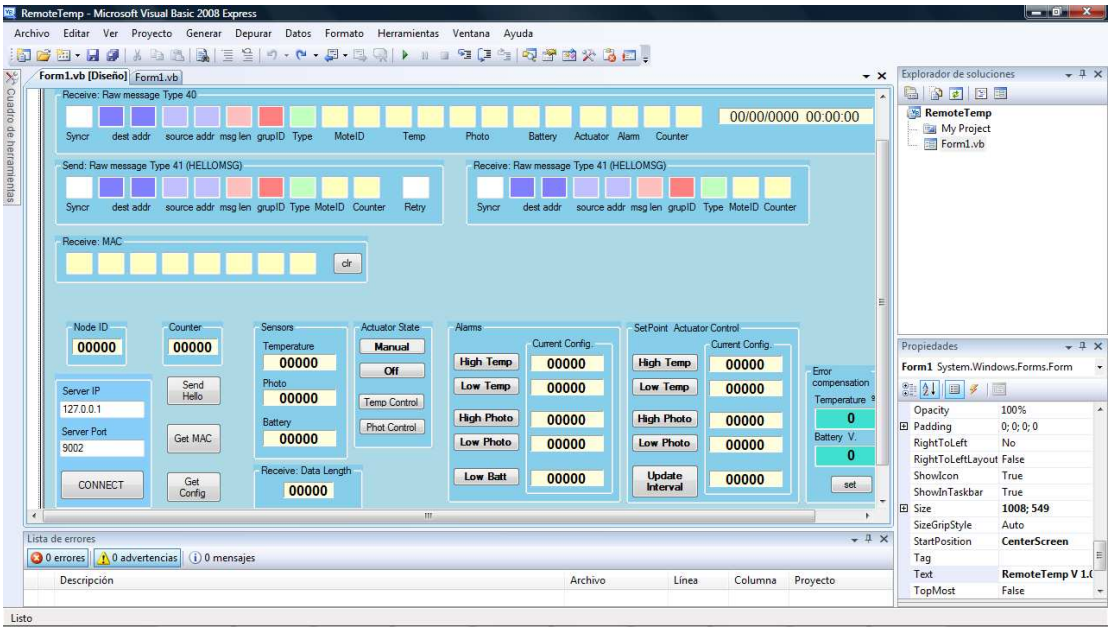


Fig. 15 - RemoteTemp PC, entorno de desarrollo

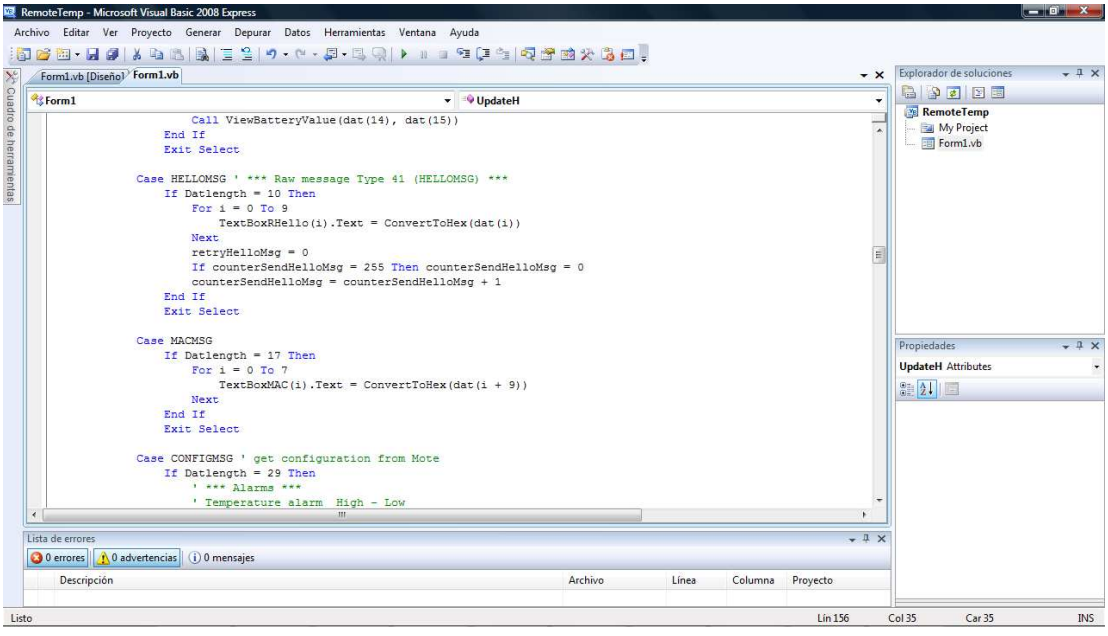


Fig. 16 - Entorno de desarrollo Microsoft VB.NET

En bin --> Release, disponemos de una versión ejecutable de RemoteTemp, (no se ha publicado ninguna versión para distribución e instalación).

De modo didáctico, se visualizan distintos mensajes en crudo (RAW), de esta forma sirve para analizar y estudiar estas estructuras a la par que facilita el estudio y desarrollo de la aplicación y el sistema.

Para el estudio y análisis de las tramas nos hemos servido también del analizador de protocolo de red, Wireshark, en la siguiente figura, podemos apreciar entre otros, el envío de un mensaje desde Serialforwarder a la aplicación PC.

Las tramas se ven duplicadas, esto es debido a que Wireshark solo capta el tráfico en el puerto ethernet, ve la salida hacia el exterior y la vuelta, para esto a la hora de conectar RemoteTemp PC con SerialForwarder deberemos utilizar una IP pública en lugar de localhost (127.0.0.1) y si disponemos de router, habilitar la IP local del PC y el puerto (9002 en nuestro caso) en Virtual Server.

Además de este modo se posibilita la conexión remota y distante a través de Internet.

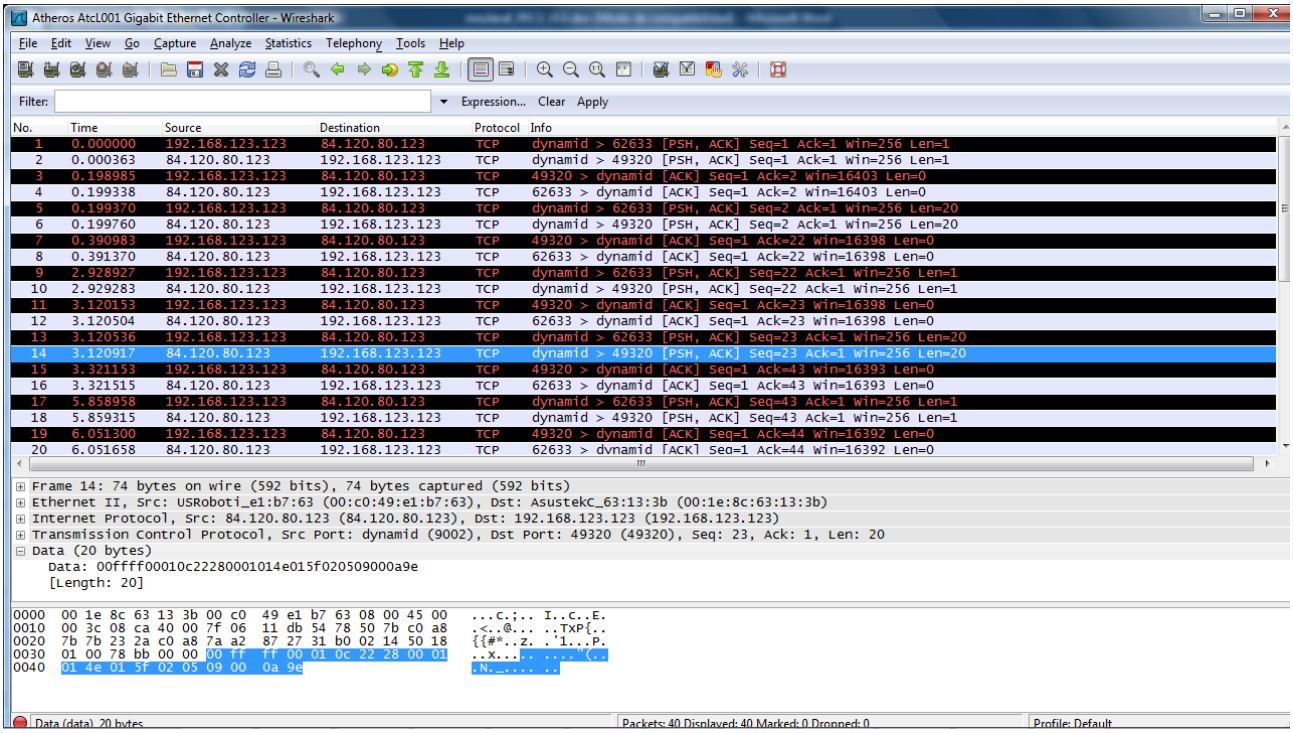


Fig. 17 - Analizador de protocolo de red, Wireshark

La interfaz gráfica de nuestra aplicación (versión 3) es la siguiente:

Fig. 18 - Interfaz gráfica RemoteTemp PC

En ella podemos apreciar el mensaje “RAW” tipo 40, que se actualizará regularmente (la mota lo envía a intervalos regulares) con una indicación de la fecha y hora de recepción.

Otros mensajes que podemos apreciar son el envío y recepción del HelloMsg que enviaremos con el botón Send Hello, en el cuadro Send de HelloMsg podemos apreciar los “Retry” envíos que han sido necesarios para obtener una respuesta de la mota pues en esta se produce en ocasiones perdida de eventos receive, posiblemente por interferencias (esto se tendrá que investigar si se dispone de tiempo, entre otros utilizo teclado y ratón bluetooth que utiliza también 2.4 GHz, además de otros dispositivos “vecinos” que pueden estar utilizando esta banda, sin embargo las emisiones desde la mota con RemoteTemp, parecen no fallar).

La MAC la obtendremos al pulsar el botón Get MAC, el Receive: MAC dispone de un botón clr para borrar la MAC para que al volver a solicitarla veamos que se recibe.

El siguiente botón disponible es Get Config, con él enviaremos un mensaje a la mota solicitando el envío de su configuración actual, puntos de alarma, setpoints para el control automático e intervalo de actualización en milisegundos, 1 Seg. por defecto.
Todo esto es modificable, con los botones “>>” podemos copiar la configuración o setpoints a enviar para facilitar su edición, modificar el parámetro que nos interese y enviar la nueva configuración a la mota.
Si la configuración se ha cambiado correctamente, veremos aparecer los nuevos valores modificados.

Estos mensajes se envían una sola vez al pulsar el correspondiente botón, para reintentarlo, si no se recibe el mensaje, deberemos pulsar nuevamente.
En otros puntos, podemos apreciar el ID del nodo, el contador de mensajes (reflejo del mensaje recibido regularmente) el último valor de los sensores, el estado del actuador y las alarmas, además de la longitud de los mensajes según se van recibiendo.

En el grupo de controles “Actuator State” visualizamos el estado del actuador pudiendo operar sobre ellos, cambio de modo Automático/Manual, conectar/desconectar el actuador manualmente (debe estar en modo manual) y elegir el sensor para el control en modo automático.
Estos mensajes, sí que se envían repetidas veces de forma automática en caso de no recibir la oportuna respuesta de la mota.

Las alarmas son indicativas, excepto las alarmas Hall y Boot que permanecerán activas hasta ser reconocidas (hay que pulsar sobre ellas para enviar un ACK), el cursor cambia de forma sobre estas alarmas para indicarnos que podemos actuar sobre ellas (cursor Hand, mano con el dedo índice extendido).

La alarma Hall nos indica que ha sido activado el sensor de efecto hall y Boot nos indica que viene de un reset o arranque (se habrá cargado la configuración por defecto, cambiarla si es necesario), ambas alarmas permanecerán hasta ser reconocidas.

Para corregir en parte el error de los sensores se dispone de un “cuadro” a tal efecto.
La conversión a unidades de ingeniería se realiza en esta aplicación, de esta forma se descarga la mota de algunos cálculos y se evita tener que reprogramar las motas si realizáramos cambios en los cálculos (imaginar una red de motas numerosas).
El sensor de luminosidad, no es corregido ni convertido, la hoja de características es algo “escasa”, con una definición poco precisa resistencia-Lux, junto con una “reacción” logarítmica, hacen difícil la conversión por ejemplo a Lúmenes.

La tensión de batería está dividida por dos mediante dos resistencias según se aprecia en el esquema de la mota:

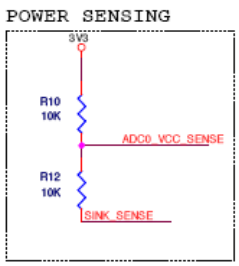


Fig. 19 - Esquema Power Sensing

Las funciones de conversión utilizadas son las siguientes (VREF= 2560 mV.):

Temperatura:

```
'Vin= ADC * Vref/1024
' Temperatura en ºC
(vIn - 100) / 10 - 40 + tempErrorCompensa
```

Tensión bateria:

```
'Vin= ADC * 2 * Vref/1024
'Battery V.
vIn / 1000 + battErrorCompensa
```

En esta última versión, RemoteTemp se conecta con una base de datos para registrar las lecturas, alarmas y estado actuador, esto puede posibilitar en una versión posterior, visualizar un log de los registros o un histórico gráfico, gestión de usuarios y privilegios entre otras posibilidades (esta base de datos, la utilizará iRemoteTemp, acceso web, para visualizar los distintos parámetros).

RemoteTemp, también se sirve de esta BD para ejecutar los comandos recibidos de iRemoteTemp, leerá el ultimo comando registrado en una tabla, se lo enviará a la mota y lo marcará como ejecutado en el registro de la tabla.

En “Web User” podemos ver el ultimo usuario web que envió el comando para ser ejecutado.
Será conveniente en próxima versión, la gestión automática del tamaño de la base de datos para limitar el tamaño de esta, de momento, se deberá realizar de forma manual.

En el cuadro de conexión pondremos la IP y el puerto para la conexión con SerialForwarder y pulsaremos conectar, una vez conectados se habilitarán los botones para el envío de mensajes y se visualizarán los mensajes que periódicamente se reciben, previamente debe estar conectada la mota base, ejecutándose SerialForwarder y servidor BD operativo.

A continuación podemos ver algunas salidas de la aplicación:

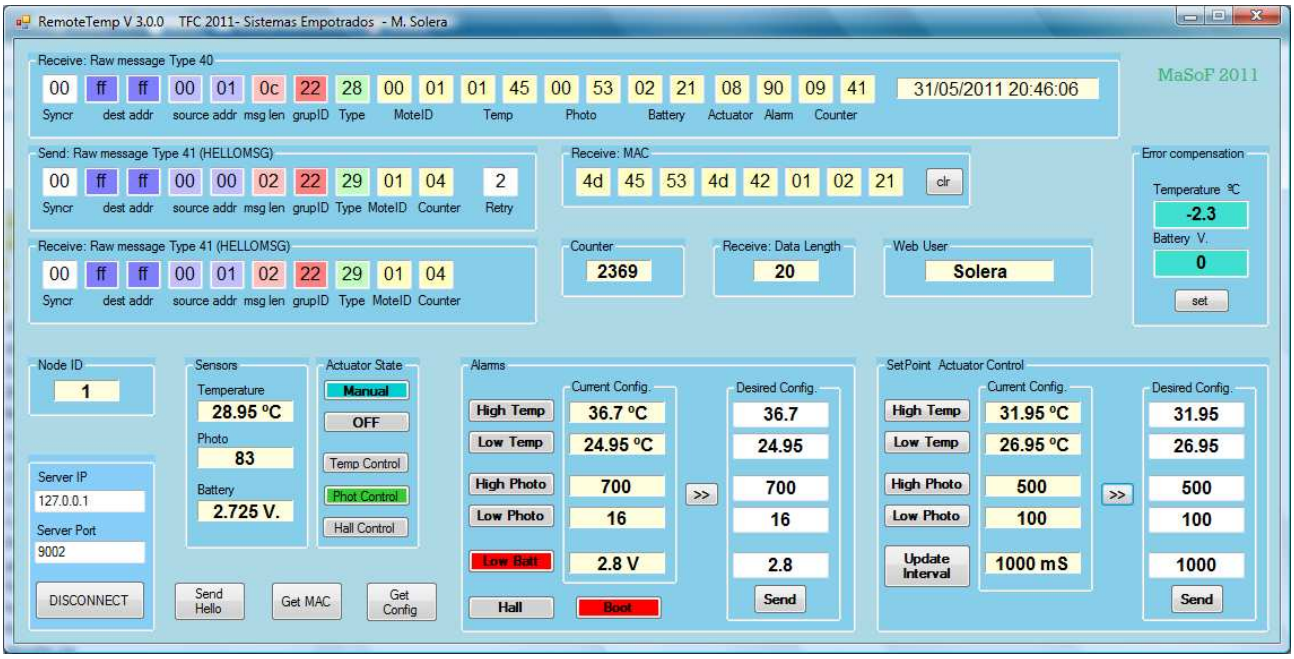


Fig. 20 - RemoteTemp PC (1)

Actuador en automático, conectado al descender la luminosidad (79) del setpoint (100):

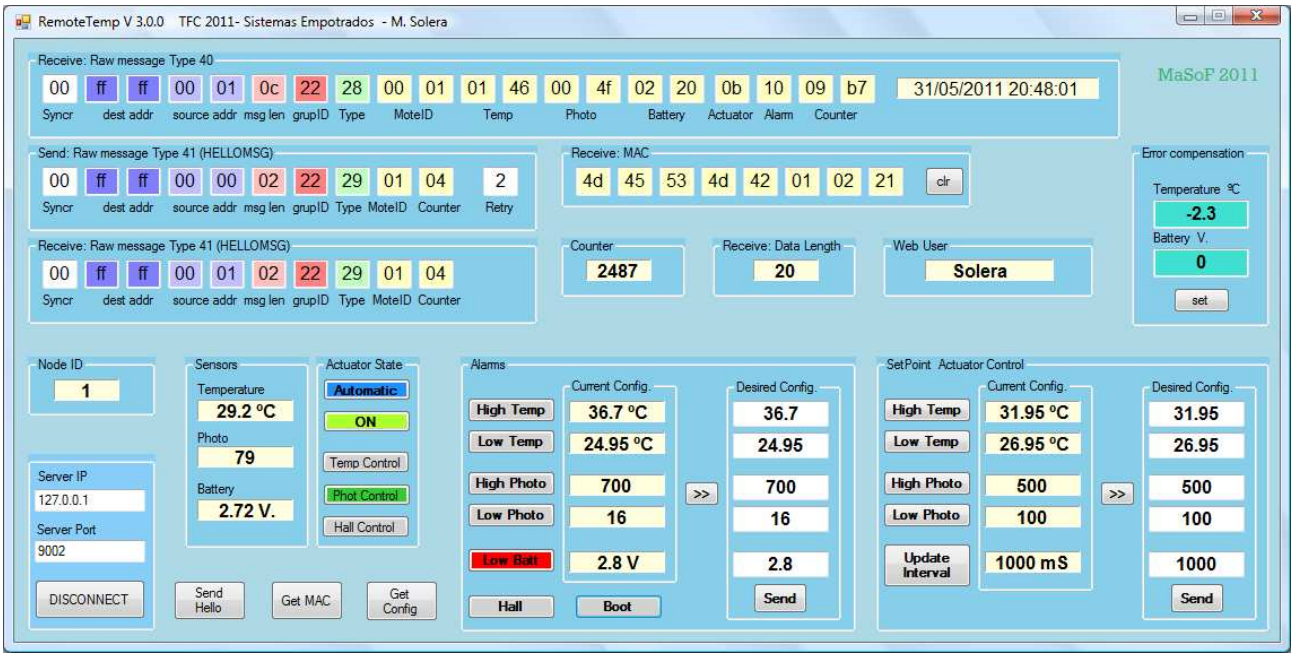


Fig. 21 - RemoteTemp PC (2)

Actuador en automático, MAC, recepción de HelloMsg al segundo intento, control Hall activado y alarma hall esperando ser reconocida:

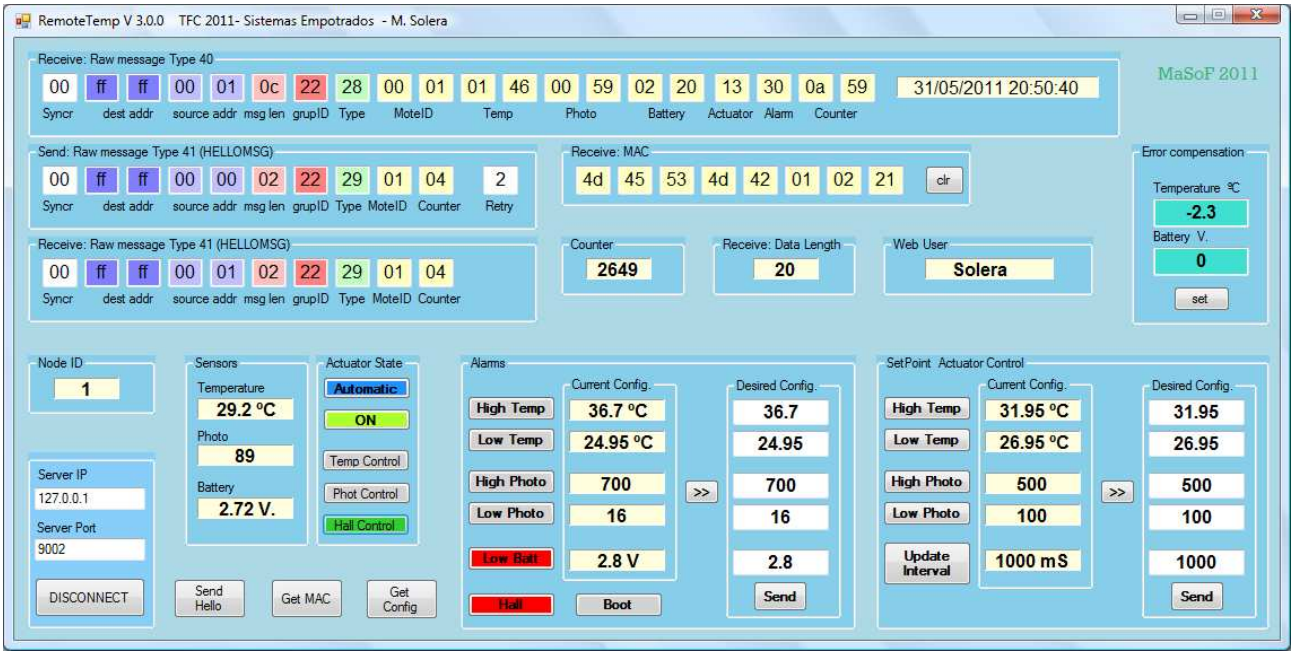


Fig. 22 - RemoteTemp PC (3)

Se ha cambiado el punto de alarma “High Temp” a 31 y se ha disparado:

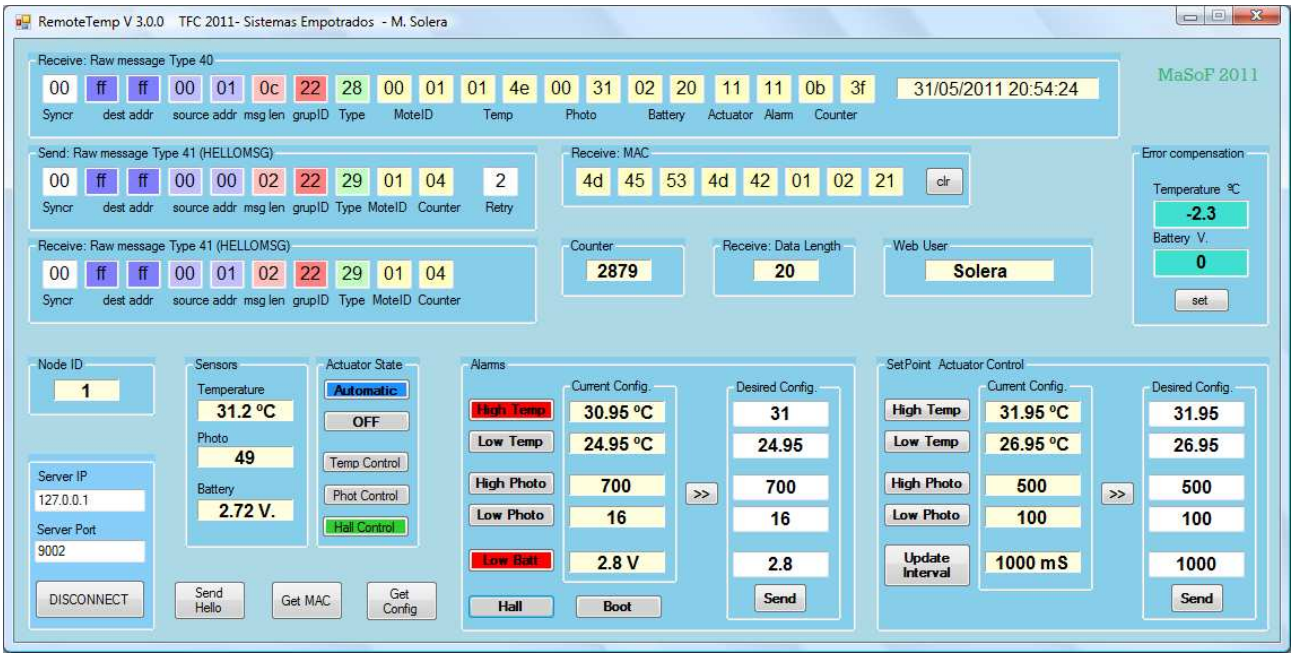


Fig. 23 - RemoteTemp PC (4)

4.4. iRemoteTemp Web

Para facilitar el acceso remoto desde internet mediante navegador web y acceso desde telefonía móvil, se ha creado iRemoteTemp.

Desarrollado con Microsoft Visual Web Developer 2008 Express Edition, utiliza ASP y Visual Basic para su programación.

El objetivo era comprobar la posibilidad y viabilidad de realizar el acceso al sistema sobre todo mediante un teléfono móvil, esta ha sido una primera toma de contacto con este entorno de programación y si bien todavía falta mucho por hacer, es totalmente operativo.

Primeramente para acceder nos requerirá un usuario y contraseña:

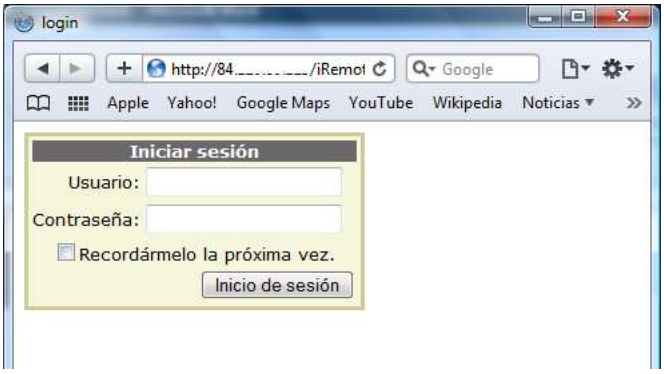


Fig. 24 - iRemoteTemp, pagina se inicio, usuario-contraseña

Comprobado y funcionando en Internet Explorer 7 y Safari 5.0.3:

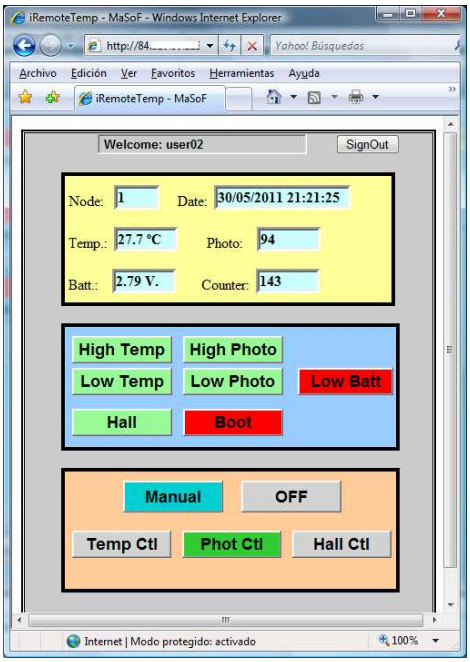


Fig. 25 - iRemoteTemp en Internet Explorer 7

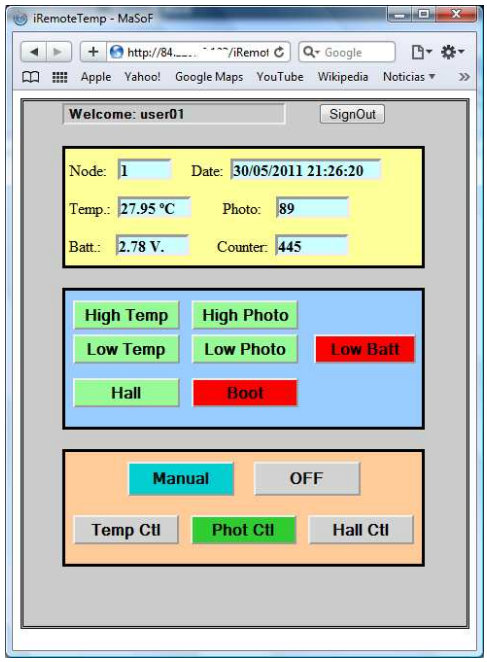


Fig. 26 - iRemoteTemp en Safari 5.0.3

La interfaz gráfica se ha realizado pensando en un móvil, el móvil en cuestión es un iPhone 4, algunas líneas de código están dedicadas a él y es en este móvil en el que está probada la aplicación, no se ha tenido la oportunidad de momento de probarlo en otros Smartphone.

Fig. 27 - iRemoteTemp en smartphone

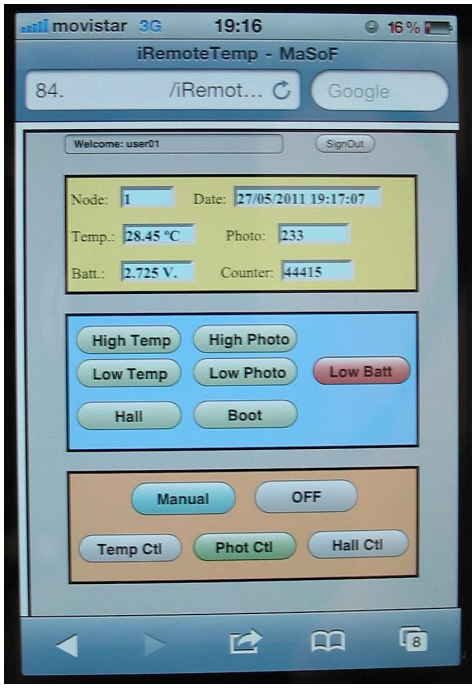


Fig. 28- iRemoteTemp detalle smartphone

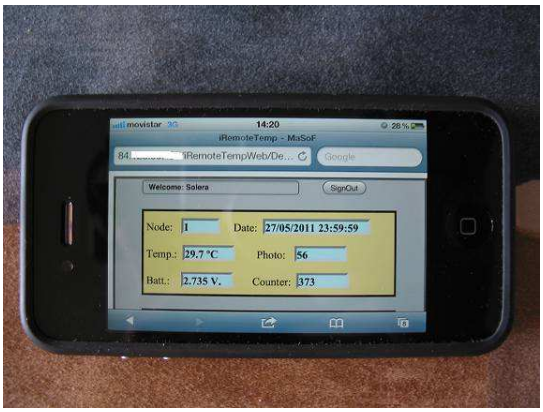


Fig. 29 - iRemoteTemp detalle cambio posición smartphone

iRemoteTemp, utiliza una base de datos (DBRemoteTemp) como “intermediario” entre esta y la aplicación RemoteTemp PC.
De esta base de datos obtendrá las lecturas y en esta BD registrará los comandos que desea ejecutar, podemos reconocer las alarmas “Hall”, “Boot” y operar el actuador (cambio modo Aut/Man, conexión/Desconexión y modo de control).

La página se actualiza cada dos segundos de forma automática, esto puede producir un parpadeo en la recarga si la conexión no es lo suficientemente rápida, como ya se ha comentado, falta mucho por hacer, en próxima versión sería conveniente entre otras cosas, actualizar solamente los cambios (valores, botones) en vez de toda la página (utilización de AJAX).

4.7. Actuador (hardware)

Para completar el sistema y dotarlo de una utilidad más practica y real se ha diseñado y elaborado un prototipo de actuador que se conecta a la mota y proporciona 220V. en unas tomas de corriente.

Se ha utilizado GPIO5_CON (Pin 21), accesible desde el conector EXP_CON (JB9) Pin 15, (masa Pin 6) para conectar el actuador a la mota.

La conexión a la mota se realiza mediante un optoacoplador para aislar un componente de otro.
Con el relé utilizado se pude proporcionar 2200 W. a 220V.
Se ha incluido una fuente de alimentación en el mismo circuito con un pequeño transformador y un regulador 7812.
La activación del actuador y disponibilidad de 220V. en las tomas de corriente es indicada por un led rojo apreciable en el exterior de la caja.
El transistor NPN es un viejo MC140 y el optoacoplador un PC817.
La excitación del optoacoplador se realiza con un consumo menor de 4mA.
Se han realizado unos primeros cálculos teóricos para las resistencias y después se han reajustado si ha sido necesario en “pruebas de laboratorio”.

Esquema:

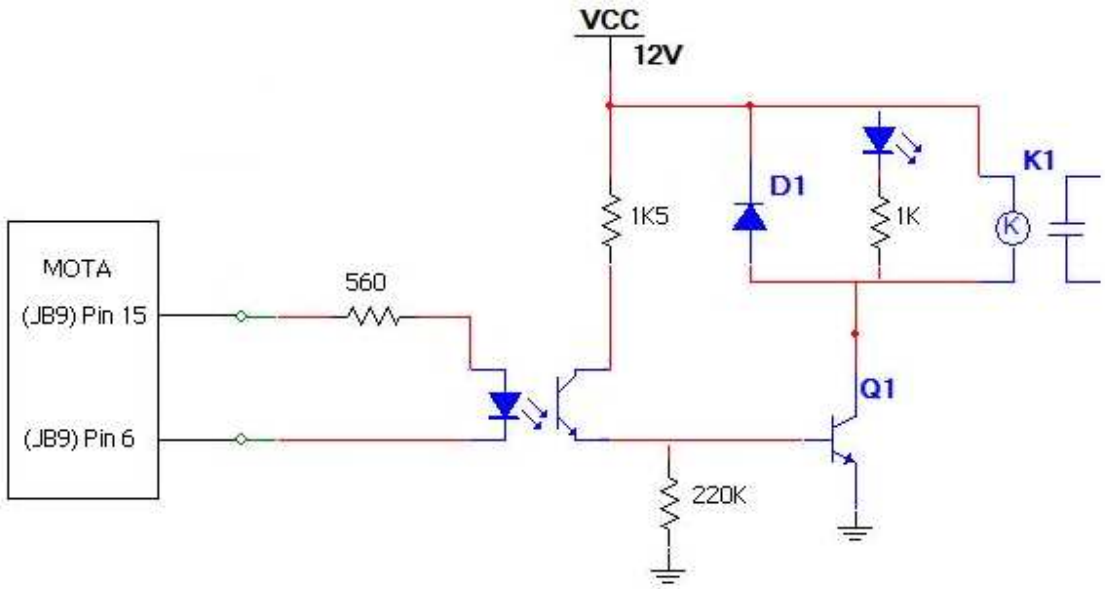
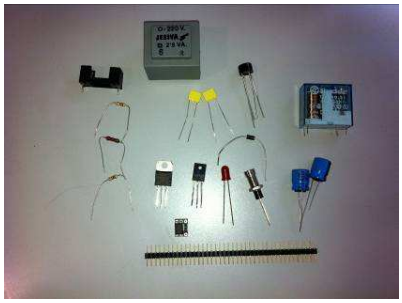
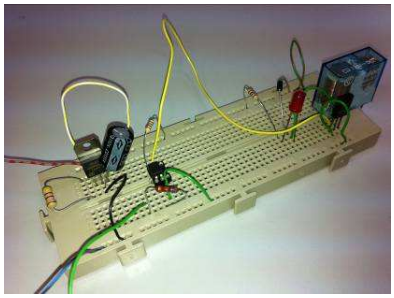
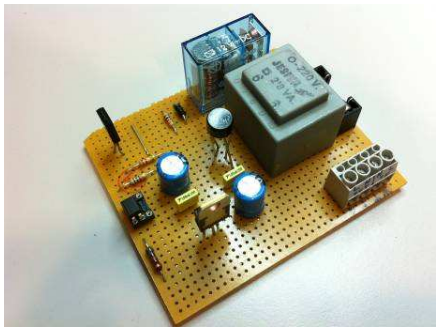
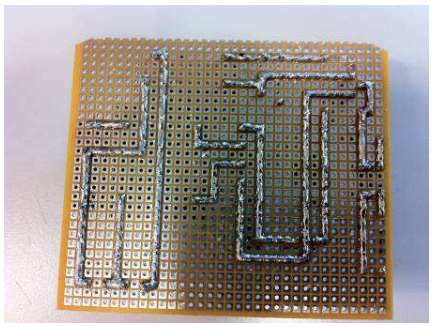


Fig. 30 - Esquema actuador

Detalles del actuador:



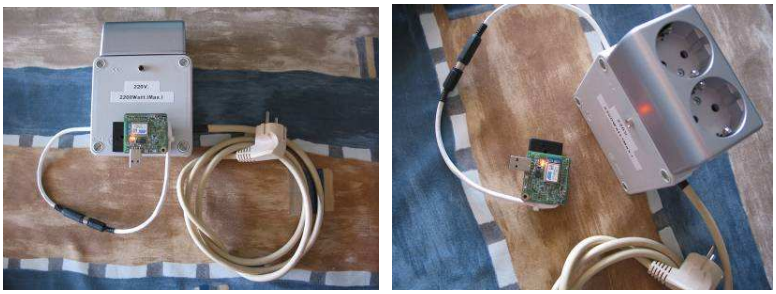



Fig. 31 - Detalles Actuador

5. Otros componentes

5.1. IIS

Internet Information Server (IIS) nos proporcionará el servicio web para el sistema operativo Microsoft Windows, iRemoteTemp se sirve de este para su acceso.

IIS viene preinstalado en Windows Vista pero deberemos habilitarlo para su funcionamiento y configurar las oportunas autorizaciones.

Puede ser necesario abrir el puerto 80 en router y firewall.

Para crear un sitio Web de IIS local con una carpeta virtual

http://msdn.microsoft.com/es-es/library/a1zz9df4(VS.90).aspx

Configuring Authentication in IIS 7

http://technet.microsoft.com/en-us/library/cc733010(WS.10).aspx

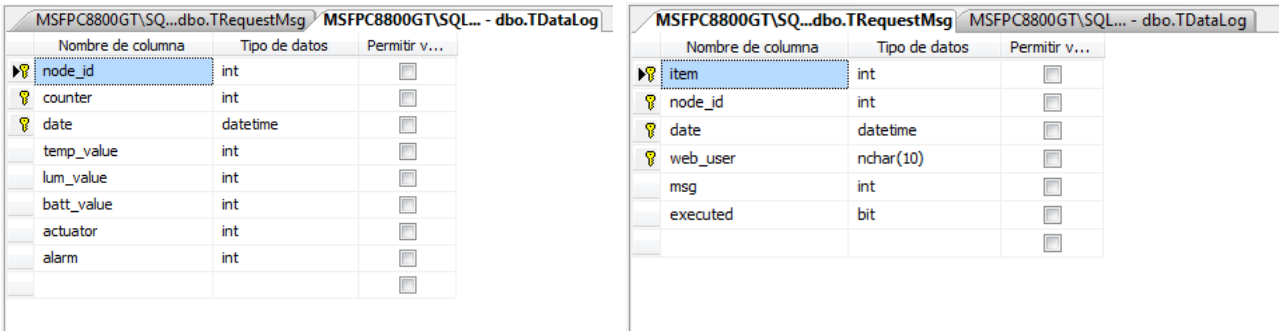
5.2. Base de Datos

Microsoft SQL Server 2008 es el servidor de bases de datos utilizado.

Se han creado una sencilla base de datos “DBRemoteTemp” con dos tablas (TDataLog y TRequestMsg) donde se registran por RemoteTemp PC los datos recibidos de la mota y los comandos recibidos de iRemoteTemp para ser ejecutados en la mota mediante RemoteTemp PC.

El campo “executed” de TRequestMsg estará a cero y se pondrá a uno si el comando (msg) ha sido ejecutado.

Tablas de DBRemoteTemp:



MSFPC8800GT\SQ...dbo.TRequestMsg | MSFPC8800GT\SQL... - dbo.TDataLog

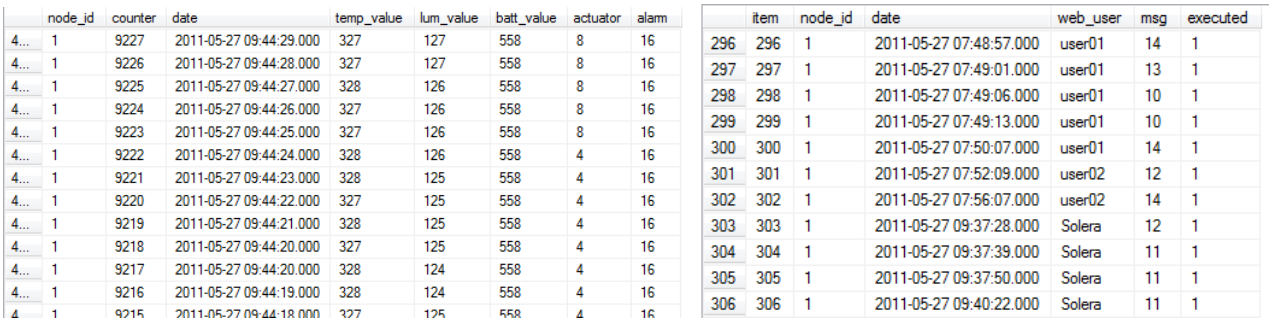
Nombre de columna	Tipo de datos	Permitir v...
node_id	int	
counter	int	
date	datetime	
temp_value	int	
lum_value	int	
batt_value	int	
actuator	int	
alarm	int	

MSFPC8800GT\SQ...dbo.TRequestMsg | MSFPC8800GT\SQL... - dbo.TDataLog

Nombre de columna	Tipo de datos	Permitir v...
item	int	
node_id	int	
date	datetime	
web_user	nchar(10)	
msg	int	
executed	bit	

Fig. 32 - Tablas de DBRemoteTemp

Detalle registros tablas:



	node_id	counter	date	temp_value	lum_value	batt_value	actuator	alarm
4...	1	9227	2011-05-27 09:44:29.000	327	127	558	8	16
4...	1	9226	2011-05-27 09:44:28.000	327	127	558	8	16
4...	1	9225	2011-05-27 09:44:27.000	328	126	558	8	16
4...	1	9224	2011-05-27 09:44:26.000	327	126	558	8	16
4...	1	9223	2011-05-27 09:44:25.000	327	126	558	8	16
4...	1	9222	2011-05-27 09:44:24.000	328	126	558	4	16
4...	1	9221	2011-05-27 09:44:23.000	328	125	558	4	16
4...	1	9220	2011-05-27 09:44:22.000	327	125	558	4	16
4...	1	9219	2011-05-27 09:44:21.000	328	125	558	4	16
4...	1	9218	2011-05-27 09:44:20.000	327	125	558	4	16
4...	1	9217	2011-05-27 09:44:20.000	328	124	558	4	16
4...	1	9216	2011-05-27 09:44:19.000	328	124	558	4	16
4	1	9215	2011-05-27 09:44:18.000	327	125	558	4	16

	item	node_id	date	web_user	msg	executed
296	296	1	2011-05-27 07:48:57.000	user01	14	1
297	297	1	2011-05-27 07:49:01.000	user01	13	1
298	298	1	2011-05-27 07:49:06.000	user01	10	1
299	299	1	2011-05-27 07:49:13.000	user01	10	1
300	300	1	2011-05-27 07:50:07.000	user01	14	1
301	301	1	2011-05-27 07:52:09.000	user02	12	1
302	302	1	2011-05-27 07:56:07.000	user02	14	1
303	303	1	2011-05-27 09:37:28.000	Solera	12	1
304	304	1	2011-05-27 09:37:39.000	Solera	11	1
305	305	1	2011-05-27 09:37:50.000	Solera	11	1
306	306	1	2011-05-27 09:40:22.000	Solera	11	1

Fig. 33 - Detalle registros tablas DBRemoteTemp

Se deben proporcionar los permisos adecuados “NT AUTHORITY\Servicio de red” (Usuario ASP).

Cómo: Configurar la seguridad de SQL Server para aplicaciones .NET
http://support.microsoft.com/kb/815154

6. Viabilidad técnica

Con el hardware y software desarrollado, el proyecto es totalmente funcional y operativo, ofrece monitorización y control tanto a nivel local como remoto, además de un dispositivo actuador que hace más versátil y practico el sistema.

7. Valoración económica

Se puede realizar una estimación de costos orientativa o aproximada, los precios pueden tener grandes variaciones dependiendo de cantidades, distribuidores, materiales, equipos elegidos.

La mayor partida económica se la llevarían los costes de desarrollo, el tiempo dedicado al estudio, desarrollo y documentación del sistema (la experiencia podría reducir esta partida en futuros desarrollos).

La instalación podría ser como mucho de unas horas y el mantenimiento requerido es mínimo.

Una implementación comercial tendría unos costes adicionales pero diluiría los costes de desarrollo.

El software de desarrollo utilizado es gratuito, sin embargo, para una utilización comercial cierto software podría tener un coste.

Para un acceso remoto se necesita una conexión a internet que puede tener un coste de unos 40 € mensuales más una conexión de datos para el móvil de 20 €.

Se necesitan un mínimo de dos motas y un actuador.

Si consideramos un periodo de desarrollo de 4 meses con una media de 22 días laborables o efectivos y una media de 6 horas diarias a un precio hora de 30 € tendremos un coste de desarrollo de 4 * 22 * 6 * 30= 15.840 €

Concepto	Cantidad	Precio	Total
Desarrollo	1	15840	15840
Motas	2	35	70
Actuador	1	30	30
Ordenador	1	900	900
S.O.	1	150	150
Monitor	1	200	200
Soft. Desarrollo	Varios	Free	0
Internet	1	60	60
Smartphone	1	600	600
			17850

Fig. 34 - Tabla valoración económica

8. Conclusiones

8.1. Conclusiones

Se marcaron unos objetivos básicos y unos objetivos adicionales, enumerados a lo largo del presente documento, condicionados por el tiempo disponible y la superación de algunos inconvenientes técnicos que suponían un reto.
Ha sido necesario documentarse y aprender a utilizar varios lenguajes y entornos de desarrollo.
Los objetivos básicos quedan cumplidos en un 100% y los “adicionales” en un 90%, echando de menos alguna función como el salvar en memoria interna de las motas los valores modificados de configuración para ser recuperados después de un reset (de momento se suple con la alarma “Boot” que nos indica que ha habido un reset y funciona con los valores por defecto, brindando la posibilidad de modificarlos).

8.2. Propuesta de mejoras

Son muchas las propuestas de mejora posibles, siempre se pueden añadir nuevas funcionalidades, optimizar código, gráficos, comunicación con otras motas para ampliar la red de sensores.
Una función a implementar en próxima versión es la indicada anteriormente, salvar la configuración modificada en las motas para recuperarla después de un reset.
Algún modo de control más como calefacción/refrigeración, noche/día, modo ahorro de energía.
En cuanto al software de monitorización y control, la base de datos nos aporta muchas posibilidades, históricos, gráficos, control de acceso y permisos; es necesario una mayor gestión de la base de datos.
Se trata de un sistema “privado”, instalado en principio de forma local y acceso remoto mediante IP pública, una opción a contemplar es el envío de esta IP por email cuando sea modificada.
Incluir las funciones de serialforwarder en la aplicación para poder prescindir de su uso.

En cuanto a la aplicación web, ha sido un primer contacto con el entorno de desarrollo y aunque cumple correctamente con los objetivos, se debe optimizar para ser más eficaz, además de complementar con una serie de opciones más.

8.3. Autoevaluación

El trabajo, la experiencia, ha resultado muy interesante; se han visto nuevos sistemas, lenguajes y entornos de desarrollo, una iniciación en ellos en mayor o menor medida, aportando nuevos conocimientos y experiencia.
No cabe duda que queda mucho por estudiar y aprender; los objetivos han sido cumplidos, siendo estos modificados, ampliados o redirigidos a medida que se adquirían nuevos cocimientos del tema y se descubrían nuevas posibilidades.

9. Glosario

-- Embedded systems: Sistemas embebidos, Sistemas empotrados,

-- ADC: Analog-to-Digital Converter, dispositivo que convierte un valor continúo en un valor discreto digital.

-- WSN: wireless sensor network, redes de sensores inalámbricas.

-- Domótica: sistemas para automatización de viviendas.

-- Inmótica: automatización de edificios, “edificios inteligentes”.

-- WPAN: wireless personal area network.

-- ZigBee: protocolos de alto nivel de comunicación inalámbrica.

-- S.O.: sistema operativo.

-- TinyOS: sistema operativo basado en componentes para redes de sensores inalámbricas.

-- NesC: (Network Embedded Systems C) lenguaje de programación basado en C, especialmente diseñado para programar aplicaciones sobre redes de sensores.

-- Motas o motes: pequeño dispositivo compuesto básicamente de un microprocesador con memoria, sensores, una radio de baja potencia y una batería (el nombre proviene del previsible cada vez más reducido tamaño).

-- ACK: ACKNOWLEDGEMENT (acuse de recibo)

10. Bibliografía

-- Wiki UOC
http://cv.uoc.edu/app/mediawiki14/

-- Web oficial de TinyOS
http://www.tinyos.net

-- Tinyos programming
http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf

-- NesC
http://nescc.sourceforge.net/

-- NesC 1.1 reference manual
http://nescc.sourceforge.net/papers/nesc-ref.pdf

-- Datasheet microcontrolador atmega1281
http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf

-- Datasheet sensor de temperatura MCP9700A
http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf

-- Datasheet sensor de luminosidad PDV-P90003-1
http://www.advancedphotonix.com/ap_products/pdfs/PDV-P9003-1.pdf

-- Datasheet sensor de efecto Hall BU520001gul-e
http://www.rohm.com/products/databook/sensor/pdf/bu52001gul-e.pdf

-- Datasheet ZigBit™ 2.4 GHz Wireless Modules ATZB-24-A2/B0
http://www.atmel.com/dyn/resources/prod_documents/doc8226.pdf

-- Esquema mota COU
http://eimtcollab.uoc.edu/softcou24_12/mota_cou_1_2_24A2_2011.pdf

-- Porting the ZigBit 900 Platform to TinyOS
http://disco.ethz.ch/theses/hs08/mb900_tinyos.pdf

-- **M**icro**s**oft **D**eveloper **N**etwork
http://msdn.microsoft.com/

-- Datasheet PC817 Series Photocoupler
http://www.classiccmp.org/rtellason/chipdata/pc817.pdf

11. Anexos

11.1. Programación, Compilación, Carga

Para la programación nesC nos servimos del IDE Eclipse, con este también podemos compilar la aplicación; ver salida siguiente.

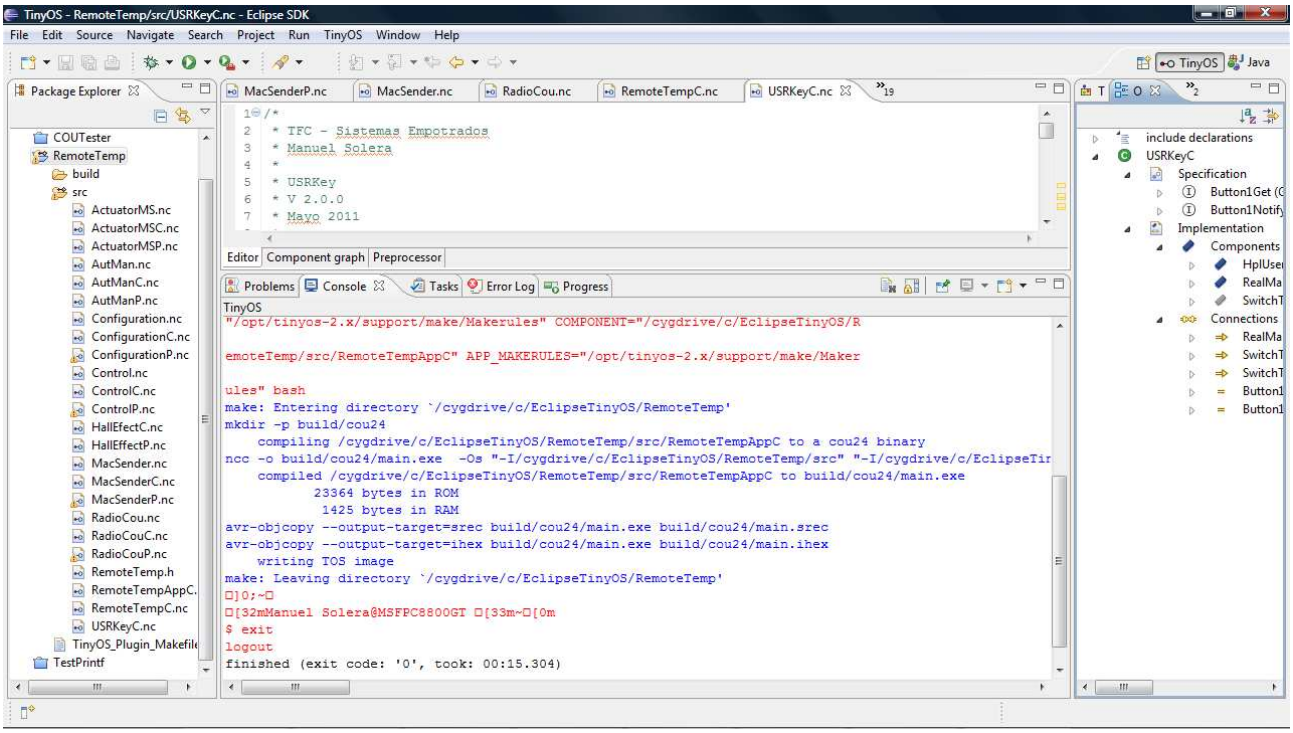


Fig. 35 - IDE Eclipse

Una vez compilado, la carga en la mota se ha realizado desde el shell Cygwin (pulsar botón RST para iniciar la carga):

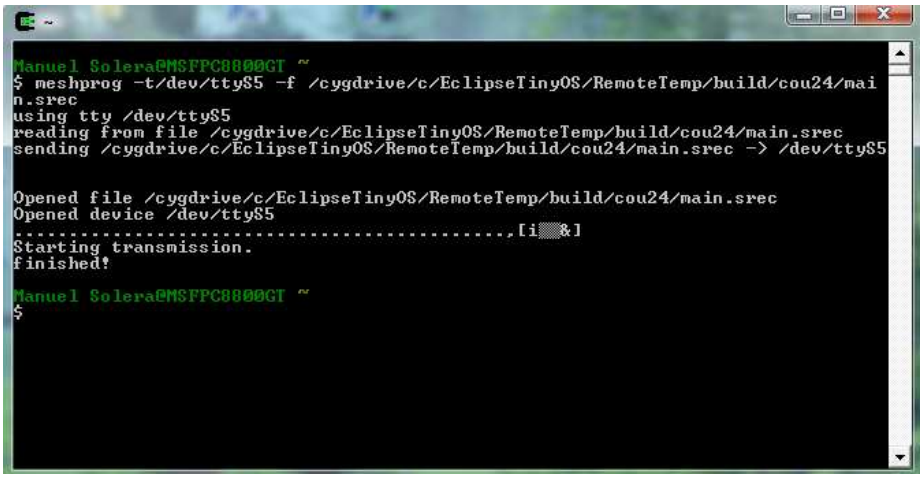


Fig. 36 - Carga en la mota

- Serialforwarder

Invocamos Serialforwarder desde la Shell de Cygwin:

$ java net.tinyos.sf.SerialForwarder -port 9002 -comm serial@com6:19200

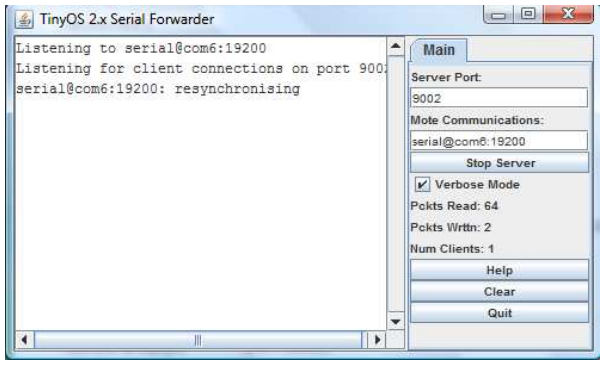


Fig. 37 - Serialforwarder

Podemos apreciar el número de puerto por el que servirá a la aplicación RemoteTemp PC (9002), la comunicación con la mota (COM 6 a una velocidad de 19200), paquetes leídos desde la mota, paquetes escritos/enviados a la mota y el número de clientes conectados (uno, RemoteTemp PC).