Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Código fuente

correspondiente al

Proyecto Fin de Carrera

Red de sensores inalámbricos para monitorización de instalaciones eléctricas de baja tensión

autor José Antonio Tarifa Galisteo

tutor D. Juan Manuel Escaño González

Sevilla, 2011

Índice general

Listado con los códigos de los archivos empleados en todo el trabajo. Antes se describe el contenido y funcionalidad de cada uno de los ficheros, así como los pasos generales necesarios para compilar la aplicación de TinyOS e instalarla en los motes inalámbricos.

Los códigos no contienen acentos gráficos (tildes) de forma intencionada (evita incompatibilidades de caracteres).

1.1 Organización de ficheros

Se dividirán en dos bloques: aplicación de TinyOS y programa de MATLAB.

1.1.1 Archivos TinyOS

- `MaxwellAppC.nc`. Archivo nesC de componente de configuración. Agrega todos los módulos necesarios (realizando el cableado) para componer la aplicación principal.
- `MaxwellC.nc`. Archivo nesC de componente de tipo módulo. Implementa la funcionalidad específica de la aplicación del trabajo.
- `Maxwell.h`. Fichero de cabecera con constantes para una configuración general de la aplicación (identificadores, valores por defecto...).
- `Sensores.h`. Fichero de cabecera que almacena constantes para realizar las conversiones de sensores auxiliares.
- `Dft.h`. Fichero de cabecera con constantes relacionadas con la recogida de datos de ondas de tensión e intensidad y parámetros del algoritmo FFT.
- `Mensajes.h`. Fichero de cabecera con declaraciones de las estructuras de mensajes utilizadas (radio, diseminación, puerto serie).
- `Makefile`. El fichero `makefile` contiene las reglas de compilación, así como banderas específicas para determinadas librerías.
- `volumes-stm25p.xml`. Definición de los bloques de memoria flash a utilizar.

1.1.2 Archivos MATLAB

- `monitor.m`. Código principal para ejecutar el sistema de recogida y envío de datos en MATLAB. También se incluye el servidor web HTTP.
- `procdata.m`. Función auxiliar usada en `monitor.m` para separar los datos que se reciben por el puerto serie.

1.2 Compilación del programa

Para la compilación de la aplicación se deberá abrir Cygwin y acceder al directorio en el cual se encuentren los archivos fuente (normalmente en `cygwin\opt\tinyos-2.x\apps\`) y usar las siguientes líneas según lo que se requiera:

- Compilación para **nodo de datos**, sin utilizar depuración:

  ```
  make telosb priority
  ```

- Compilación para **nodo de datos**, con opción de depuración (esta opción hay que usarla con cuidado: si se usan todas las macros para la depuración la imagen binaria sobrepasa los límites de la memoria ROM, no se pueden activar todas las macros simultáneamente):

  ```
  CFLAGS+=-DDEBUGMODE make telosb priority
  ```

- Compilación para **nodo base (o raíz)**, sin depuración:

  ```
  CFLAGS+=-DROOT_MOTE make telosb priority
  ```

- Compilación para **nodo base (o raíz)**, utilizando depuración:

  ```
  CFLAGS+=-DROOT_MOTE\ -DDEBUGMODE make telosb priority
  ```

La imagen ya compilada aparecerá en el mismo directorio que los archivos fuente, en `build\telosb\`.

1.3 Instalación del programa en los motes

La opción más sencilla para instalar la imagen de la aplicación en un sensor es conectando éste directamente al PC en el cual se tenga dicha imagen compilada (a través del puerto USB). Hecho esto, se deberá escribir en Cygwin

```
make telosb reinstall,ID bsl,COM-1
```

`ID` es el identificador que se usará en la red inalámbrica (recuérdese el criterio que se ha propuesto para distinguir a nodos raíz de nodos de datos). `COM` indica el puerto al cual está conectado el sensor (se puede usar el comando `motelist`).

Si solo hay un mote conectado al ordenador, puede escribirse

```
make telosb reinstall,ID bsl,auto
```

O simplemente

```
make telosb reinstall,ID
```

Hecho esto comenzará el volcado del archivo imagen a la memoria del sensor.

La otra opción es instalar la imagen (para nodos base y ya compilada) a través de la pasarela NSLU2 con el *firmware* Tmote Connect. En este caso, el comando es

```
make telosb reinstall,ID netbsl,Dir.IP:Puerto
```

`ID` es el identificador que se usará en la red inalámbrica. `Dir.IP` indica la dirección IP que tiene la pasarela (por ejemplo `192.168.1.100`) y `Puerto` indica el control del mote (`10001` o `10002` dependiendo el puerto USB de la pasarela).

1.4 MaxwellAppC.nc

```
1   /*
2   Copyright (c) 2011, Jose Antonio Tarifa Galisteo
3   All rights reserved.
4
5   Redistribution and use in source and binary forms, with or without
6   modification, are permitted provided that the following conditions
7   are met:
8
9       * Redistributions of source code must retain the above copyright
10      notice, this list of conditions and the following disclaimer.
11      * Redistributions in binary form must reproduce the above copyright
12      notice, this list of conditions and the following disclaimer in the
13      documentation and/or other materials provided with the
14      distribution.
15      * Neither the name of the University of Seville nor the names of
16      its contributors may be used to endorse or promote products derived
17      from this software without specific prior written permission.
18
19  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
20  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
21  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
22  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
23  THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
24  INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
25  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
27  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
28  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
29  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
30  OF THE POSSIBILITY OF SUCH DAMAGE.
31  */
32
33  /*-----------------------------------------------------------------------------
34     Archivo:    MaxwellAppC.nc
35     Autor:      Jose Antonio Tarifa Galisteo
36     Correo e:   jostargal@alum.us.es
37     Sitio web:  http://alumno.us.es/j/jostargal/
38     Fecha:      17/05/2011
39     Revision:   0517
40     Descrip:    Componente de tipo configuracion de la aplicacion principal.
41                 Se estructura en dos grandes bloques: signatura (en este caso
42                 vacia) e implementacion (declaracion de componentes a usar y
43                 posterior cableado de interfaces).
44
45     Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
46     Universidad de Sevilla                    http://www.us.es/
47  -----------------------------------------------------------------------------*/
48
49
50  // Inclusion ficheros de cabecera
```

```
51      #include "Maxwell.h"
52      #include <StorageVolumes.h>
53
54
55      #warning .........(c) 2011 by Jose Antonio Tarifa Galisteo [0517]
56      #warning .........Higher Technical School of Engineering. University of Seville
57
58
59      // Avisos (warnings) sobre la aplicacion a compilar (nodo base/datos)
60      #ifdef ROOT_MOTE
61         #ifdef DEBUGMODE
62            #warning Compilando para NODO RAIZ [debug]
63         #else
64            #warning Compilando para NODO RAIZ
65         #endif
66      #else
67         #ifdef DEBUGMODE
68            #warning Compilando para NODO DE DATOS [debug]
69         #else
70            #warning Compilando para NODO DE DATOS
71         #endif
72      #endif
73
74
75      configuration MaxwellAppC {
76
77      }  // Fin de SIGNATURE
78
79
80      implementation {
81
82      // Implementacion de la configuracion MaxwellAppC
83
84      /*-------------------------------------------------------------------------------
85
86                COMPONENTES A UTILIZAR
87
88      -------------------------------------------------------------------------------*/
89
90      //   COMPONENTES COMUNES (nodos base y de datos) ------------------------------
91
92         // Componente (modulo) principal
93         components MaxwellC;
94
95         // Secuencia de arranque del sistema TinyOS
96         components MainC;
97
98         // Manejo de los LED del mote
99         components LedsC;
100
101        // Temporizador general para el arranque del programa
102        components new TimerMilliC() as TemporizadorA;
103
104        // Programador de tareas con prioridad
105        components PrioritySchedulerC;
106        // Tarea de muy alta prioridad (no tiene funcionalidad)
107        components new VeryHighTaskC() as VHPTask;
```

```
108
109         // Control de interfaces para mensajes radio
110         components ActiveMessageC;
111
112         // Activacion del protocolo de recoleccion de datos
113         components CollectionC as Collector;
114         // Envio de mensajes radio usando recoleccion de datos
115         components new CollectionSenderC(COLLECTION_ID);
116
117         // Activacion del protocolo de diseminacion de datos
118         components DisseminationC;
119         // Difusion de datos via radio
120         components new DisseminatorC(DissMsg, DISSEMINATOR_ID) as DissC;
121
122         // Control de interfaces para mensajes puerto serie
123         components SerialActiveMessageC as Serial;
124
125
126     //   COMPONENTES NODOS BASE (raiz) ------------------------------------------
127
128     #ifdef ROOT_MOTE
129
130         // Temporizador para 'watchdog' (solo nodo raiz)
131         components new TimerMilliC() as TemporizadorB;
132
133     #endif
134
135
136     //   COMPONENTES NODOS DE DATOS ---------------------------------------------
137
138     #ifndef ROOT_MOTE
139
140         // Tarea para actualizacion de mensajes diseminados
141         components new VeryLowTaskC() as VLPTaskActual;
142         // Tarea de calculo de valores medios, eficaces, potencias...
143         components new LowTaskC() as LPTaskCalcPot;
144         // Tarea de calculo de la DFT de las señales leidas
145         components new LowTaskC() as LPTaskCalcDFT;
146         // Tarea para el procesamiento de sensores auxiliares
147         components new LowTaskC() as LPTaskProcSens;
148         // Tarea para enviar mensajes via radio
149         components new LowTaskC() as LPTaskEnvio;
150         // Tarea para controlar las conversiones a realizar
151         components new HighTaskC() as HPTaskConv;
152
153         // Convertidor A-D del MSP430 para tension e intensidad
154         components new Msp430Adc12ClientAutoRVGC() as ConvertVI;
155         // Convertidor A-D del MSP430 para sensores auxiliares
156         components new Msp430Adc12ClientAutoRVGC() as ConvertSENS;
157
158         // Abstraccion de entradas-salidas digitales
159         components HplMsp430GeneralIOC as GeneralIOC;
160
161         // Almacenamiento en memoria flash externa
162         components new ConfigStorageC(VOLUME_CONFIG);
163
164     #endif
```

```
165
166
167     /*-----------------------------------------------------------------------------
168
169               WIRING O CABLEADO
170
171     -----------------------------------------------------------------------------*/
172
173     //   CABLEADO COMUN (nodos base y de datos) -----------------------------------
174
175        MaxwellC.Boot -> MainC;
176
177        MaxwellC.Leds -> LedsC;
178
179        MaxwellC.TimerInit -> TemporizadorA;
180
181        PrioritySchedulerC.Leds -> LedsC;
182        MaxwellC.VHT -> VHPTask;
183
184        MaxwellC.RadioControl -> ActiveMessageC;
185
186        MaxwellC.RoutingControl -> Collector;
187        MaxwellC.RootControl -> Collector;
188        MaxwellC.Receive -> Collector.Receive[COLLECTION_ID];
189        MaxwellC.Send -> CollectionSenderC;
190
191        MaxwellC.DisseminationControl -> DisseminationC;
192        MaxwellC.Actualizacion -> DissC;
193        MaxwellC.ValorDis -> DissC;
194
195        MaxwellC.SerialControl -> Serial;
196
197
198     //   CABLEADO NODOS BASE (raiz) ----------------------------------------------
199
200     #ifdef ROOT_MOTE
201
202        MaxwellC.TimerWDT -> TemporizadorB;
203
204        MaxwellC.SerialSend -> Serial.AMSend[SERIALTX_ID];
205        MaxwellC.SerialRcv -> Serial.Receive[SERIALRX_ID];
206        MaxwellC.SerialPkt -> Serial;
207
208     #endif
209
210
211     //   CABLEADO NODOS DE DATOS -------------------------------------------------
212
213     #ifndef ROOT_MOTE
214
215        MaxwellC.ResourceVI -> ConvertVI;
216        MaxwellC.OverflowVI -> ConvertVI;
217        MaxwellC.MultiChannelVI -> ConvertVI;
218        MaxwellC.AdcConfigureVI <- ConvertVI;
219
220        MaxwellC.ResourceSENS -> ConvertSENS;
221        MaxwellC.OverflowSENS -> ConvertSENS;
```

```
222         MaxwellC.MultiChannelSENS -> ConvertSENS;
223         MaxwellC.AdcConfigureSENS <- ConvertSENS;
224 
225         MaxwellC.CargaA -> GeneralIOC.Port23;     // 6-pin header (U28) pin#3
226         MaxwellC.CargaB  -> GeneralIOC.Port26;    // 6-pin header (U28) pin#4
227 
228         MaxwellC.ConfigSto -> ConfigStorageC.ConfigStorage;
229         MaxwellC.MountSto  -> ConfigStorageC.Mount;
230 
231         MaxwellC.ActValores -> VLPTaskActual;
232         MaxwellC.CalcPotencia -> LPTaskCalcPot;
233         MaxwellC.CalcDFT -> LPTaskCalcDFT;
234         MaxwellC.ProcSensor -> LPTaskProcSens;
235         MaxwellC.EnvioDatos -> LPTaskEnvio;
236         MaxwellC.Conversion -> HPTaskConv;
237 
238     #endif
239 
240 
241     }  // Fin de IMPLEMENTATION
242 
243 
244     //                    Fin de archivo 'MaxwellAppC.nc'
```

1.5 MaxwellC.nc

```
1   /*
2   Copyright (c) 2011, Jose Antonio Tarifa Galisteo
3   All rights reserved.
4
5   Redistribution and use in source and binary forms, with or without
6   modification, are permitted provided that the following conditions
7   are met:
8
9       * Redistributions of source code must retain the above copyright
10      notice, this list of conditions and the following disclaimer.
11      * Redistributions in binary form must reproduce the above copyright
12      notice, this list of conditions and the following disclaimer in the
13      documentation and/or other materials provided with the
14      distribution.
15      * Neither the name of the University of Seville nor the names of
16      its contributors may be used to endorse or promote products derived
17      from this software without specific prior written permission.
18
19  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
20  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
21  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
22  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
23  THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
24  INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
25  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
27  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
28  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
29  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
30  OF THE POSSIBILITY OF SUCH DAMAGE.
31  */
32
33  /*-----------------------------------------------------------------------------
34     Archivo:    MaxwellC.nc
35     Autor:      Jose Antonio Tarifa Galisteo
36     Correo e:   jostargal@alum.us.es
37     Sitio web:  http://alumno.us.es/j/jostargal/
38     Fecha:      17/05/2011
39     Revision:   0517
40     Descrip:    Componente de tipo modulo con funcionalidad para la aplicacion
41                 principal. Contiene dos grandes bloques: signatura (interfaces
42                 usadas y proporcionadas por el modulo) e implementacion
43                 (definicion de comandos y manejo de eventos).
44
45     Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
46     Universidad de Sevilla                    http://www.us.es/
47  -----------------------------------------------------------------------------*/
48
49  // Inclusion ficheros de cabecera
```

```
50      #include "Maxwell.h"
51      #include "Dft.h"
52      #include "Sensores.h"
53      #include "Mensajes.h"
54
55
56      module MaxwellC {
57
58      // Signatura del modulo MaxwellC. Indica interfaces usadas y/o proporcionadas
59
60         uses {
61      //   INTERFACES USADAS COMUNES (nodos base y de datos) -------------------------
62
63            interface Boot;
64
65            interface Leds;
66
67            interface Timer<TMilli> as TimerInit;
68
69            interface TaskBasic as VHT;
70
71            interface SplitControl as RadioControl;
72
73            interface StdControl as RoutingControl;
74            interface RootControl;
75            interface Receive;
76            interface Send;
77
78            interface StdControl as DisseminationControl;
79            interface DisseminationUpdate<DissMsg> as Actualizacion;
80            interface DisseminationValue<DissMsg> as ValorDis;
81
82
83      //   INTERFACES USADAS NODOS BASE ----------------------------------------------
84      #ifdef ROOT_MOTE
85
86            interface Timer<TMilli> as TimerWDT;
87
88            interface SplitControl as SerialControl;
89            interface AMSend as SerialSend;
90            interface Receive as SerialRcv;
91            interface Packet as SerialPkt;
92
93      #endif
94
95
96      //   INTERFACES USADAS NODOS DE DATOS ------------------------------------------
97      #ifndef ROOT_MOTE
98
99            interface Resource as ResourceVI;
100           interface Msp430Adc12Overflow as OverflowVI;
101           interface Msp430Adc12MultiChannel as MultiChannelVI;
102
103           interface Resource as ResourceSENS;
104           interface Msp430Adc12Overflow as OverflowSENS;
105           interface Msp430Adc12MultiChannel as MultiChannelSENS;
106
```

```
107             interface SplitControl as SerialControl;
108
109             interface HplMsp430GeneralIO as CargaA;
110             interface HplMsp430GeneralIO as CargaB;
111
112             interface ConfigStorage as ConfigSto;
113             interface Mount as MountSto;
114
115             interface TaskBasic as ActValores;
116             interface TaskBasic as CalcPotencia;
117             interface TaskBasic as CalcDFT;
118             interface TaskBasic as ProcSensor;
119             interface TaskBasic as Conversion;
120             interface TaskBasic as EnvioDatos;
121
122         #endif
123         }  // Fin de interfaces USADAS
124
125
126         //   INTERFACES PROPORCIONADAS NODOS DE DATOS ---------------------------------
127         #ifndef ROOT_MOTE
128         provides {
129             interface AdcConfigure<const msp430adc12_channel_config_t*>
130               as AdcConfigureVI;
131             interface AdcConfigure<const msp430adc12_channel_config_t*>
132               as AdcConfigureSENS;
133         }  // Fin de interfaces PROPORCIONADAS
134         #endif
135
136         }  // Fin de SIGNATURE
137
138
139         implementation {
140
141         // Implementacion del modulo MaxwellC
142
143         /*-------------------------------------------------------------------------------
144
145                   VARIABLES Y CONSTANTES GLOBALES
146
147         -------------------------------------------------------------------------------*/
148
149             // Estructura de mensajes de TinyOS. Variable para envio radio
150             message_t radiopkt;
151
152
153         #ifdef ROOT_MOTE
154
155             // Estructura de mensajes de TinyOS. Variable para envio por puerto serie
156             message_t serialpkt;
157
158             // Bandera para indicar si el puerto serie esta enviando datos (1) o no (0).
159             norace uint8_t serialsend_status = 0;
160
161         #else
162
163
```

```
164         /*
165         Array bidimensional de almacenamiento de las lecturas del ADC12.
166         El primer indice indica en cual de los buffers se esta escribiendo/leyendo.
167         El segundo, el numero de muestra.
168            Componentes = NUMSAMP*MAXBUF
169            Tamanio = (2*NUMSAMP*MAXBUF)B
170         */
171         uint16_t raw[MAXBUF][NUMSAMP];
172
173         /*
174         Array para almacenar las lecturas del ADC correspondientes a los sensores
175         auxiliares. Se toman 16 medidas de cada uno de los sensores. Por ejemplo,
176         si hay 4 sensores, el orden sera 1-2-3-4-1-2-3-4-1-2-...-3-4
177         (en total 4*16=64 componentes).
178            Componentes = 16*NUMSENS
179            Tamanio = (32*NUMSENS)B
180         */
181         uint16_t sensor[16*NUMSENS];
182
183         /*
184         Variables para indicar sobre que buffer se esta trabajando. El array 'raw'
185         puede tener varios buffer para poder leer en uno a la vez que se escribe
186         en otro. Estas variables acceden a cada buffer en el momento correspondiente.
187         */
188         norace uint8_t bufesc = 0;   // Buffer de escritura
189         norace uint8_t buflec = 0;   // Buffer de lectura
190
191         /*
192         Banderas para el control de las tareas y recursos. El valor '0' indica que
193         no se esta ejecutando (libre). Uno distinto de '0' indica proceso activo
194         (tambien puede indicar el estado en el que se encuentra).
195         */
196         norace uint8_t adc_status = 0;      // Uso del convertidor A-D
197         norace uint8_t calcp_status = 0;    // Tarea de calculo de potencias
198         norace uint8_t sens_status = 0;     // Procesamiento sensores auxiliares
199         norace uint8_t dft_status = 0;      // Calculo de la DFT de las medidas
200         norace uint8_t send_status = 0;     // Estado envio de paquetes via radio
201
202         /*
203         Bandera para indicar si hay que escribir en la memoria flash externa del
204         nodo. Se activa cuando hay cambios en los parametros de distorsion maxima,
205         temperatura o tension umbrales.
206         */
207         norace bool wr_memory = FALSE;
208
209         /*
210         Banderas para indicar desbordamiento en el convertirdor A-D. El
211         desbordamiento puede deberse a fallo de memoria o de tiempo de muestreo.
212         En caso de que exista, los datos almacenados no se procesaran (pueden ser
213         erroneos).
214         */
215         norace bool vi_overflow = FALSE;    // Medidas de tension e intensidad
216         norace bool sens_overflow = FALSE;  // Medidas de sensores auxiliares
217
218         /*
219         Valores de umbral o limite. 'thd_umbral' especifica cual es el valor maximo
220         de distorsion permitido para la tension (componente 0) y para intensidad (1).
```

```
221         'temp_umbral' establece un valor maximo permitido para la temperatura interna
222         del microcontrolador del nodo. 'volt_umbral' para la tension de alimentacion.
223         */
224         norace uint32_t thd_umbral[2];
225         norace uint16_t temp_umbral;
226         norace uint16_t volt_umbral;
227
228         /*
229         CONFIGURACION DEL ADC12
230         Se usan dos tipos de configuraciones distintas.
231         La primera realiza las conversiones de los canales analogicos A0 y A1,
232         correspondientes a los sensores de tension (V) e intensidad (I).
233         La segunda configura la conversion de los otros sensores adicionales, como
234         temperatura interna, tension de alimentacion...
235         Dentro de cada configuracion, es necesaria la estructura 'config_t' para el
236         primer canal, y despues el array 'adc12memctl_t' para el resto de canales que
237         se convierten.
238         */
239         // Estructura que contiene la configuracion del ADC para el primer canal (A0)
240         const msp430adc12_channel_config_t configVI = {
241            inch: INPUT_CHANNEL_A0,             // Primer canal para convertir
242            sref: REFERENCE_VREFplus_AVss,      // Referencias de tension
243            ref2_5v: REFVOLT_LEVEL_2_5,         // Usar generador de tension interno
244            adc12ssel: SHT_SOURCE_SMCLK,        // Reloj de S&H (1bMHz)
245            adc12div: SHT_CLOCK_DIV_1,          // Divisor del reloj S&H
246            sht: SAMPLE_HOLD_4_CYCLES,          // Ciclos para el mantenedor (H)
247            sampcon_ssel: SAMPCON_SOURCE_SMCLK, // Reloj tiempos de muestreo (1bMHz)
248            sampcon_id: SAMPCON_CLOCK_DIV_1     // Divisor para el reloj de muestreo
249         };
250
251         // Estructura (en array) para configurar el otro canal (A1)
252         adc12memctl_t memctlVI[1];
253
254         // Estructura que contiene la configuracion del ADC para el 1er canal (Temp.)
255         const msp430adc12_channel_config_t configSENS = {
256            inch: TEMPERATURE_DIODE_CHANNEL,    // Primer canal para convertir
257            sref: REFERENCE_VREFplus_AVss,      // Referencias de tension
258            ref2_5v: REFVOLT_LEVEL_2_5,         // Usar generador de tension interno
259            adc12ssel: SHT_SOURCE_SMCLK,        // Reloj de S&H (1bMHz)
260            adc12div: SHT_CLOCK_DIV_1,          // Divisor del reloj S&H
261            sht: SAMPLE_HOLD_64_CYCLES,         // Ciclos para el mantenedor (H)
262            sampcon_ssel: SAMPCON_SOURCE_SMCLK, // Reloj tiempos de muestreo (1bMHz)
263            sampcon_id: SAMPCON_CLOCK_DIV_1     // Divisor para el reloj de muestreo
264         };
265
266         // Estructura (en array) para configurar el resto de canales auxiliares
267         adc12memctl_t memctlSENS[NUMSENS-1];
268
269         /*
270         Variables para envio de mensajes radio. Las siguientes variables almacenan
271         las medidas que se van realizando o resultados de procesamiento. Cuando se
272         envie el paquete de datos, la informacion se toma de estas variables.
273         */
274         norace uint16_t vmeddata[4];     // Tensiones medias
275         norace uint16_t imeddata[4];     // Intensidades medias
276         norace uint32_t vrmsdata[4];     // Tensiones eficaces (rms)
277         norace uint32_t irmsdata[4];     // Intesidades eficaces (rms)
```

```
278         norace uint32_t pactdata[4];     // Ppotencias activas
279         norace uint16_t fdpdata[4];      // Factores de potencia
280         norace uint32_t thddata[2];      // Distorsion (tension e intensidad)
281         norace uint16_t tempvoltdata[2]; // Temperatura y tension de alimentacion
282
283         storage_struct permdata;   // Estructura de almacenamiento en flash
284
285      #endif
286
287
288      /*-----------------------------------------------------------------------------
289
290                FUNCIONES AUXILIARES
291
292      -----------------------------------------------------------------------------*/
293
294         /*
295         Funcion para conocer si un nodo determinado es de tipo root (base) o de tipo
296         de datos.
297         Se divide el identificador del nodo TOS_NODE_ID entre la constante simbolica
298         ROOT_DIVIDER. En caso de que el resto sea cero, se considerara nodo raiz.
299            Argumentos entrada:  N/A
300            Argumentos salida:   valor booleano. Si es nodo raiz=TRUE; en caso
301                                 contrario, devuelve FALSE.
302         */
303         bool IsRoot() {
304            return ((TOS_NODE_ID%ROOT_DIVIDER == 0) ? TRUE : FALSE);
305         }  // Fin FUNCTION IsRoot
306
307
308         /*
309         Funcion que arranca o iniciar los distintos componentes que se usan en el
310         sistema. Si se diera algun tipo de fallo, la funcion vuelve a intentar el
311         arranque. Esto se realiza hasta un numero de veces definido por la constante
312         MAXATTEMPT.
313            Argumentos entrada:  uint8 'device'. Indica el tipo de componente a
314                                 iniciar, segun la lista existente en 'Maxwell.h'
315            Argumentos salida:   error_t. Indica si ha habido algun fallo al intentar
316                                 iniciar el componente, o si todo ha ido
317                                 correctamente.
318         */
319         error_t Arrancar(uint8_t device) {
320            uint8_t attempt = 0;
321            switch (device) {
322               case RADIO:
323                  do {
324                     attempt++;
325                  } while (attempt<=MAXATTEMPT &&
326                           (call RadioControl.start() != SUCCESS));
327                  break;
328               case DISSEM:
329                  do {
330                     attempt++;
331                  } while (attempt<=MAXATTEMPT &&
332                           (call DisseminationControl.start() != SUCCESS));
333                  break;
334               case ROUTING:
```

```
335                  do {
336                     attempt++;
337                  } while (attempt<=MAXATTEMPT &&
338                           (call RoutingControl.start() != SUCCESS));
339                  break;
340               case ROOT:
341                  do {
342                     attempt++;
343                  } while (attempt<=MAXATTEMPT &&
344                           (call RootControl.setRoot() != SUCCESS));
345                  break;
346               case SERIAL:
347                  do {
348                     attempt++;
349                  } while (attempt<=MAXATTEMPT &&
350                           (call SerialControl.start() != SUCCESS));
351                  break;
352               // Este caso solo se da en sensores de recogida de datos
353               #ifndef ROOT_MOTE
354               case ADCVI:
355                  do {
356                     attempt++;
357                  } while (attempt<=MAXATTEMPT &&
358                           (call ResourceVI.request() != SUCCESS));
359                  break;
360               #endif
361
362            }  // Fin del SWITCH device
363
364            if (attempt>MAXATTEMPT)
365               return FAIL;
366            else
367               return SUCCESS;
368
369         }  // Fin FUNCTION Arrancar
370
371
372         /*
373         Funcion para inicializar variables de la aplicacion. Se inicializan los
374         vectores globales, se fijan umbrales predeterminados y se configura el
375         convertidor A-D. Tambien se establecen las salidas digitales.
376            Argumentos entrada:  N/A
377            Argumentos salida:   N/A
378         */
379         void InicioMote (void) {
380      #ifdef ROOT_MOTE
381            // No hace nada
382      #else
383
384            int8_t i = 0;
385            int16_t j = 0;
386
387            pinfo("Iniciando variables...");
388
389            for (i=0; i<MAXBUF; i++) {
390               for (j=0; j<NUMSAMP; j++) {
391                  raw[i][j] = (uint16_t)0;
```

```
392                }
393             }
394
395             for (j=0; j<(16*NUMSENS); j++) {
396                sensor[j] = (uint16_t)0;
397             }
398
399             /*
400             En caso de que se trate de un nodo de datos, hay que inicializar las
401             estructuras auxiliares (resto de canales)para el control del
402             convertidor A-D.
403             */
404             memctlVI[0].inch=INPUT_CHANNEL_A1;
405             memctlVI[0].sref=REFERENCE_VREFplus_AVss;
406
407             memctlSENS[0].inch=SUPPLY_VOLTAGE_HALF_CHANNEL;
408             memctlSENS[0].sref=REFERENCE_VREFplus_AVss;
409             memctlSENS[1].inch=INPUT_CHANNEL_A2;
410             memctlSENS[1].sref=REFERENCE_VREFplus_AVss;
411             memctlSENS[2].inch=INPUT_CHANNEL_A3;
412             memctlSENS[2].sref=REFERENCE_VREFplus_AVss;
413
414             /*
415             Activacion de los pines correspondientes al control de la carga para que
416             funcionen como pines de entrada/salida general.
417             */
418             call CargaA.selectIOFunc();
419             call CargaB.selectIOFunc();
420
421             call CargaA.makeOutput();  // El pin se activa como salida digital
422             call CargaB.makeOutput();
423          #endif
424
425          }  // Fin FUNCTION InicioMote
426
427
428          /*
429          Funcion para incrementar en una unidad una variable de 8b sin signo. Cuando
430          alcanza el valor maximo '255', la funcion no tiene efecto sobre la variable.
431             Argumentos entrada:  puntero a variable de tipo uint8_t.
432             Argumentos salida:   N/A
433          */
434          void increase(uint8_t* x) {
435             if ((*x) != 255) {
436                (*x)++;
437             }
438          }  // Fin FUNCTION increase
439
440
441          /*
442          Funcion para decrementar en una unidad una variable de 8b sin signo. Cuando
443          alcanza el valor minimo '0', la funcion no tiene efecto sobre la variable.
444             Argumentos entrada:  puntero a variable de tipo uint8_t.
445             Argumentos salida:   N/A
446          */
447          void decrease(uint8_t* x) {
448             if (*x) {
```

```
449             (*x)--;
450          }
451       }  // Fin FUNCTION decrease
452
453
454    /*
455    ------------------------------------------------------------------------------
456
457             MANEJO DE COMANDOS
458
459    ------------------------------------------------------------------------------
460    */
461
462    //   COMANDOS NODOS DE DATOS ------------------------------------------------
463
464    #ifndef ROOT_MOTE
465
466       /*
467       El comando es utilizado por el ADC12 cuando necesita conocer la
468       configuracion requerida (configuracion de canales de tension e
469       intensidad).
470          Tamanio variables = N/A
471       */
472       async command const msp430adc12_channel_config_t*
473        AdcConfigureVI.getConfiguration() {
474
475       //.......... CODIGO EJECUTABLE
476
477          return &configVI;
478       }  // Fin COMMAND AdcConfigureVI.getConfiguration
479
480
481       /*
482       El comando es utilizado por el ADC12 cuando necesita conocer la
483       configuracion requerida (configuracion de canales de sensores auxiliares:
484       temperatura, tension alimentacion...).
485          Tamanio variables = N/A
486       */
487       async command const msp430adc12_channel_config_t*
488        AdcConfigureSENS.getConfiguration() {
489
490       //.......... CODIGO EJECUTABLE
491
492          return &configSENS;
493       }  // Fin COMMAND AdcConfigureSENS.getConfiguration
494
495    #endif
496
497
498    /*------------------------------------------------------------------------------
499
500             MANEJO DE EVENTOS
501
502    ------------------------------------------------------------------------------*/
503
504    //   EVENTOS COMUNES (nodos base y de datos) ---------------------------------
505
```

```
506         /*
507         Este evento se activa cada vez que el mote arranca, tras haber un fallo
508         en la alimentacion, pulsar el boton Reset o enviar un comando remoto de
509         reinicio.
510            Tamanio variables = N/A
511         */
512         event void Boot.booted() {
513
514         //.......... CODIGO EJECUTABLE
515
516            welcome();
517
518            /*
519            Comprobacion inicial del ID de nodo. Si se ha compilado para un nodo
520            raiz, el resto de dividir el ID entre la constante ROOT_DIVIDER debera
521            ser cero; para nodos de datos, debera ser distinto de cero.
522            */
523
524      #ifdef ROOT_MOTE
525            // Comprobacion para un nodo base
526            if(IsRoot()) {
527               pinfo("Nodo raiz arrancando");
528            }
529            else {
530               perror("ID mote incorrecto");
531               phalt();
532               return;  // Fin de ejecucion
533            }
534      #else
535            // Comprobacion para un nodo de datos
536            if(!IsRoot()) {
537               pinfo("Nodo datos arrancando");
538            }
539            else {
540               perror("ID mote incorrecto");
541               phalt();
542               return;  // Fin de ejecucion
543            }
544      #endif
545
546            call TimerInit.startPeriodic (1024U);  // Temporizador para prueba LED
547
548         }  // Fin EVENT Boot.booted
549
550
551         /*
552         Evento que se ejecuta cada vez que el temporizador (TimerInit) vence.
553            Tamanio variables = 1B
554         */
555         event void TimerInit.fired() {
556
557         //.......... VARIABLES Y CONSTANTES LOCALES
558
559            static uint8_t contador = 1;
560
561         //.......... CODIGO EJECUTABLE
562
```

```
563             pinfo("Comprobacion LED");
564
565             if(contador >= 8) {
566                call TimerInit.stop();  // Detener el temporizador
567                call Leds.set(0);       // Apagar todos los LED
568
569          #ifdef ROOT_MOTE
570                /*
571                En caso de que se trate de un nodo raiz, se puede llamar directamente
572                a la activacion del sistema radio.
573                */
574                if (Arrancar(RADIO) != SUCCESS) {
575                   perror("Radio no arranca");
576                   return;  // Fin de ejecucion
577                }
578          #else
579                /*
580                Caso de nodo de datos, se inician variables y montaje del bloque
581                de memoria externa.
582                */
583                InicioMote();  // Inicializacion del nodo
584
585                if (call MountSto.mount() == SUCCESS) {
586                   call Leds.led2On();
587                   pinfo("ExtMem montando");
588                   // Esperar a evento MountSto.mountDone
589                }
590                else {
591                   perror("ExtMem montado");
592                }
593          #endif
594
595             }
596
597             call Leds.set(contador);
598             contador = contador<<1;    // Multiplicar contador*2
599
600          }  // Fin EVENT TimerInit.fired
601
602
603          /*
604          El evento avisa de que el sistema de radio del sensor ha sido iniciado.
605          Para arrancarlo, se debe llamar a RadioControl.start.
606             Tamanio variables = 1B
607          */
608          event void RadioControl.startDone(error_t err) {
609
610          //.......... VARIABLES Y CONSTANTES LOCALES
611
612             /*
613             Variable para contabilizar los intentos realizados al iniciar la radio
614             de forma correcta, sin errores.
615             */
616             static uint8_t attempt = 0;
617
618          //.......... CODIGO EJECUTABLE
619
```

```
620             if (err == SUCCESS) {
621                // Radio activada e iniciada correctamente
622                psuccess("Radio iniciada");
623                if (Arrancar(DISSEM) == SUCCESS) {
624                   // Protocolo de diseminacion iniciado correctamente
625                   psuccess("Disem. iniciada");
626                   if(Arrancar(ROUTING) == SUCCESS) {
627                      // Protocolo de enrutamiento iniciado correctamente
628                      psuccess("Routing iniciado");
629
630       #ifdef ROOT_MOTE
631                         // Se trata de un nodo base o raiz
632                         if (Arrancar(ROOT) == SUCCESS) {
633                            // El nodo se ha establecido como raiz correctamente
634                            psuccess("Nodo como root");
635                            if (Arrancar(SERIAL) == SUCCESS) {
636                               // Comunicacion puerto serie iniciada correctamente
637                               pinfo("SerialCOM arrancando");
638                            }
639                         }
640                         wdt_start();   //Arranque WDT para nodo raiz
641                         call TimerWDT.startPeriodic (512U); //Temporiz. para WDT
642
643       #else
644                         // Se trata de un nodo de datos
645                         if (Arrancar(ADCVI) == SUCCESS) {
646                            /* Llamada al convertidor ADC12 para iniciar las medidas
647                            realizada correctamente. Esperar confirmacion en evento
648                            ResourceVI.granted.  */
649                            pinfo("ADC12 (VI) arrancando");
650                         }
651       #endif
652                   }
653                }
654             }
655             else {
656                // La radio se ha activado, pero no se ha iniciado correctamente
657                attempt++;  // Reintento
658                if (attempt<=MAXATTEMPT) {
659                   perror("Radio reiniciando");
660                   Arrancar(RADIO);
661                }
662                else {
663                   perror("Radio no puede iniciar");
664                   phalt();
665                   return;
666                }
667             }
668
669          }  // Fin EVENT RadioControl.startDone
670
671
672          /*
673          El evento se recibe cuando la radio se ha detenido (por cualquier motivo:
674          comando RadioControl.stop, existe algun problema y se para...).
675             Tamanio variables = N/A
676          */
```

```
677         event void RadioControl.stopDone(error_t err) {
678
679         //.......... CODIGO EJECUTABLE
680
681            if (err == SUCCESS) {
682               psuccess("Radio detenido");
683            }
684            else {
685               perror("Detener Radio");
686            }
687
688         }  // Fin EVENT RadioControl.stopDone
689
690
691         /*
692         El evento se recibe cuando se ha realizado (o intentado) el envio de
693         un paquete de datos via radio. Antes se ha debido llamar al comando
694         Send.send.
695         En caso de un nodo raiz, hay que declarar el evento, pero no es
696         necesaria ninguna accion.
697         Para nodos de datos, si hay que realizar llamadas a otros comandos.
698            Tamanio variables = N/A
699         */
700      #ifdef ROOT_MOTE
701         event void Send.sendDone(message_t* m, error_t err) {
702
703         }  // Fin EVENT Send.sendDone [nodo base]
704      #else
705         event void Send.sendDone(message_t* m, error_t err) {
706
707         //.......... CODIGO EJECUTABLE
708
709            if (err == SUCCESS) {
710               psuccess("Radio. Datos enviados");
711            }
712            else {
713               perror("Radio. Envio de datos fallido");
714            }
715            atomic send_status = 0; // Radio libre para enviar datos
716            call Leds.led2Off();
717
718
719            if (call ResourceVI.request() == SUCCESS) {
720               // Peticion ADC12 para v-i correcta
721            }
722            else {
723               // Devuelto EBUSY: ya se ha realizado una solicitud
724               pinfo("ADC12 (VI) ya solicitado");
725            }
726
727         }  // Fin EVENT Send.sendDone [nodo datos]
728      #endif
729
730
731         /*
732         El evento se recibe cuando se ha recibido un paquete de datos via radio,
733         proveniente de otro/s nodo/s.
```

```
734         En caso de un nodo raiz, debe recoger el contenido del mensaje y
735         transferirlo a un sistema externo mediante el puerto serie (USB).
736         Para nodos de datos, hay que declarar el evento, pero no hay que realizar
737         ninguna accion (un nodo de este tipo no tiene que procesar los mensajes).
738            Tamanio variables = 3B (nodo raiz) / N/A (nodo datos)
739         */
740      #ifdef ROOT_MOTE
741         event message_t* Receive.receive
742          (message_t* msg, void* payload, uint8_t len) {
743
744         //.......... VARIABLES Y CONSTANTES LOCALES
745
746            uint8_t i = 0;
747            PotenciaMsg* potpkt = (PotenciaMsg*)payload; // Puntero a carga util
748
749         //.......... CODIGO EJECUTABLE
750
751            call Leds.led1On();
752
753            if (serialsend_status == 0) {
754               // Puerto serie libre para transmitir
755               SerialMsg* basepkt = (SerialMsg*)call SerialPkt.getPayload
756               (&serialpkt, sizeof(SerialMsg)); // Puntero a carga util paquete serie
757
758               basepkt -> nodeid = potpkt -> nodeid;  // ID de nodo
759
760               /*
761               Bucle para traspasar datos de un tipo de mensaje a otro.
762                  Iteraciones: 4
763               */
764               for (i=0; i<4; i++) {
765                  basepkt -> vmedia[i] = potpkt -> vmedia[i];
766                  basepkt -> imedia[i] = potpkt -> imedia[i];
767                  basepkt -> vrms[i] = potpkt -> vrms[i];
768                  basepkt -> irms[i] = potpkt -> irms[i];
769                  basepkt -> potact[i] = potpkt -> potact[i];
770                  basepkt -> fdp[i] = potpkt -> fdp[i];
771               }  // Fin bucle FOR (indice i)
772               /*
773               Bucle para traspasar datos de un tipo de mensaje a otro.
774                  Iteraciones: 2
775               */
776               for (i=0; i<2; i++) {
777                  basepkt -> thd[i] = potpkt -> thd[i];
778                  basepkt -> tempvolt[i] = potpkt -> tempvolt[i];
779               }  // Fin bucle FOR (indice i)
780
781               if (call SerialSend.send
782                 (AM_BROADCAST_ADDR, &serialpkt, sizeof(SerialMsg)) == SUCCESS) {
783                  // Peticion de envio correcta
784                  atomic serialsend_status = 1; // Puerto serie ocupado transmitiendo
785               }
786               else {
787                  // Error al intentar enviar el mensaje
788               }
789            }
790            else {
```

```
791                // Puerto serie ocupado
792             }
793
794             call Leds.led1Off();
795
796             return msg;
797
798          }  // Fin EVENT Receive.receive [nodo base]
799       #else
800          event message_t* Receive.receive
801           (message_t* msg, void* payload, uint8_t len) {
802
803          //.......... CODIGO EJECUTABLE
804
805             return msg;
806
807          }  // Fin EVENT Receive.receive [nodo datos]
808       #endif
809
810
811          /*
812          El evento se activa cuando se han diseminado nuevos valores a la red (caso
813          del nodo raiz) o bien cuando se han recibido dichos valores (nodos datos).
814          Para el caso de un nodo de datos, se llama a la tarea correspondiente para
815          procesar estos valores.
816             Tamanio variables = N/A
817          */
818       #ifdef ROOT_MOTE
819          event void ValorDis.changed() {
820
821          //.......... CODIGO EJECUTABLE
822
823             call Leds.led2Off();
824             psuccess("Radio. Mensaje disem. enviado");
825
826          }  // Fin EVENT ValorDis.changed [nodo base]
827
828       #else
829          event void ValorDis.changed() {
830
831          //.......... CODIGO EJECUTABLE
832
833             call ActValores.postTask();   // LLamada tarea actualizacion de valores
834             psuccess("Radio. Mensaje disem. recibido");
835          }  // Fin EVENT ValorDis.changed [nodo datos]
836
837       #endif
838
839
840          /*
841          Evento para avisar del arranque de la comunicacion serie (puerto USB) del
842          mote. Se inicia mediante la llamada SerialControl.start.
843             Tamanio variables = N/A
844          */
845          event void SerialControl.startDone(error_t err) {
846
847          //.......... CODIGO EJECUTABLE
```

```
848
849             if (err == SUCCESS) {
850                psuccess("SerialCOM iniciado");
851             }
852             else {
853                perror("Inicio SerialCOM");
854             }
855
856          }  // Fin EVENT SerialControl.startDone
857
858
859          /*
860          Evento para avisar de la parada de la comunicacion serie del mote.
861          Puede deberse a diversas causas: parada por comando SerialControl.stop,
862          algun problema del sistema...
863             Tamanio variables = N/A
864          */
865          event void SerialControl.stopDone(error_t err) {
866
867          //.......... CODIGO EJECUTABLE
868
869             if (err == SUCCESS) {
870                psuccess("SerialCOM detenido");
871             }
872             else {
873                perror("Detener SerialCOM");
874             }
875
876          }  // Fin EVENT SerialControl.stopDone
877
878
879
880       //   EVENTOS NODOS BASE (RAIZ) ------------------------------------------------
881
882       #ifdef ROOT_MOTE
883          /*
884          Evento que se ejecuta cada vez que el temporizador (TimerWDT) vence. Solo
885          se usa para resetear el 'watchdog' del nodo raiz o base.
886             Tamanio variables = N/A
887          */
888          event void TimerWDT.fired() {
889
890          //.......... CODIGO EJECUTABLE
891
892             wdt_touch();   // Reseteo del WDT
893
894          }  // Fin EVENT TimerWDT.fired
895
896
897          /*
898          El evento se recibe cuando se ha realizado (o intentado) el envio de
899          un paquete de datos via USB (puerto serie). Antes se ha debido llamar al
900          comando SerialSend.send.
901          Este evento solo se especifica para nodos raiz o base.
902             Tamanio variables = N/A
903          */
904          event void SerialSend.sendDone(message_t* msg, error_t error) {
```

```
905
906         //.......... CODIGO EJECUTABLE
907
908            if((&serialpkt == msg) && (error == SUCCESS)) {
909               call Leds.led0Off();
910               psuccess("Serial. Datos enviados");
911            }
912            else {
913               call Leds.led0On();
914               perror("Serial. Envio datos fallido");
915            }
916            atomic serialsend_status = 0; // Puerto serie libre
917
918         }  // Fin EVENT SerialSend.sendDone
919
920
921         /*
922         El evento se recibe cuando se recibe un mensaje a traves del puerto serie.
923         Despues, se extraen los datos y se reenvian usando diseminacion.
924         Este evento solo se especifica para nodos raiz o base.
925            Tamanio variables = 13B
926         */
927         event message_t* SerialRcv.receive
928          (message_t* msg, void* payload, uint8_t len) {
929
930         //.......... VARIABLES Y CONSTANTES LOCALES
931
932            DissMsg disspkt;                        // Estructura datos diseminacion
933            DissMsg* dissmsg = (DissMsg*)payload;  // Puntero a mensaje diseminacion
934
935         //.......... CODIGO EJECUTABLE
936
937            call Leds.led2On();
938            pinfo("Serial. Mensaje recibido");
939
940            /*
941            Extraccion de datos del mensaje recibido a traves del puerto serie para
942            enviarlos al paquete radio de diseminacion.
943            */
944            disspkt.nodeid = dissmsg -> nodeid;
945            disspkt.function = dissmsg -> function;
946            disspkt.valueA = dissmsg -> valueA;
947            disspkt.valueB = dissmsg -> valueB;
948
949            // Envio de los nuevos valores a traves de la red de difusion
950            call Actualizacion.change(&disspkt);
951
952            pinfo("Radio enviando mensaje disem.");
953
954            return msg;
955
956         }  // Fin EVENT SerialRcv.receive
957      #endif
958
959
960      //   EVENTOS NODOS DE DATOS ------------------------------------------------
961
```

```
962     #ifndef ROOT_MOTE
963
964        /*
965        El evento se senializa cuando se ha obtenido acceso al recurso compartido
966        ADC12 (para medidas de ondas de tension e intensidad). Para solicitar
967        el acceso, se usa el comando ResourceVI.request.
968           Tamanio variables = N/A
969        */
970        event void ResourceVI.granted() {
971
972        //.......... CODIGO EJECUTABLE
973
974           psuccess("Acceso a ADC12 (VI)");
975     //    wdt_start();   // Activacion del WDT para nodos de datos
976           call Conversion.postTask();   // Llamada a la tarea de conversiones
977
978        }  // Fin EVENT ResourceVI.granted
979
980
981        /*
982        El evento se senializa cuando se ha obtenido acceso al recurso compartido
983        ADC12 (para medidas de otros sensores adicionales). Para solicitar el
984        acceso, se usa el comando ResourceSENS.request.
985           Tamanio variables = N/A
986        */
987        event void ResourceSENS.granted() {
988
989        //.......... CODIGO EJECUTABLE
990
991           psuccess("Acceso a ADC12 (SENS)");
992           if (call MultiChannelSENS.configure
993           (&configSENS, memctlSENS, NUMSENS-1, &sensor[0], NUMSAMPSENS, SAMPPERSENS)
994           == SUCCESS) {
995              // Configuracion realizada correctamente
996              sens_overflow = FALSE;  // Bajada bandera desbordamiento
997              call Leds.led0Off();
998              if (call MultiChannelSENS.getData() == SUCCESS) {
999                 // Recogida de datos arrancada correctamente
1000                atomic adc_status = 1;  // ADC12 ocupado conviertiendo
1001                call Leds.led1On();
1002             }
1003             else {
1004                // Error al iniciar las conversiones
1005                perror("Conversion (SENS) no iniciada");
1006             }
1007          }
1008          else {
1009             // Error al realizar la configuracion del ADC12
1010             perror("ADC12 (SENS) no configurado");
1011          }
1012       }  // Fin EVENT ResourceSENS.granted
1013
1014
1015       /*
1016       Evento para indicar que las conversiones (tension e intensidad) ya estan
1017       disponibles en el buffer indicado cuando se inicio la conversion
1018       (comando MultiChannelVI.getData).
```

```
1019            Tamanio variables = N/A
1020          */
1021          async event void MultiChannelVI.dataReady
1022           (uint16_t *buffer, uint16_t numSamples) {
1023
1024          //.......... CODIGO EJECUTABLE
1025
1026             atomic adc_status = 0;  // ADC12 libre
1027             call Conversion.postTask();   // Preparacion tarea nueva conversion
1028             call Leds.led1Off();
1029
1030          }  // Fin EVENT MultiChannelVI.dataReady
1031
1032
1033          /*
1034          Evento para indicar que las conversiones (sensores auxiliares) ya estan
1035          disponibles en el buffer indicado cuando se inicio la conversion
1036          (comando MultiChannelSENS.getData).
1037             Tamanio variables = N/A
1038          */
1039          async event void MultiChannelSENS.dataReady
1040           (uint16_t *buffer, uint16_t numSamples) {
1041
1042          //.......... CODIGO EJECUTABLE
1043
1044             atomic adc_status = 0;  // ADC12 libre
1045             call ProcSensor.postTask();   // Preparacion tarea procesar sensores
1046             call Leds.led1Off();
1047
1048          }  // Fin EVENT MultiChannelSENS.dataReady
1049
1050
1051          /*
1052          Este evento indica cuando ha ocurrido un desbordamiento en la conversion
1053          (tension e intensidad) debido a problemas de temporizacion.
1054             Tamanio variables = N/A
1055          */
1056          async event void OverflowVI.conversionTimeOverflow() {
1057
1058          //.......... CODIGO EJECUTABLE
1059
1060             atomic vi_overflow = TRUE; // Activacion bandera desbordamiento
1061             call Leds.led0On();
1062             perror("ADC12 (VI) Time OF");
1063
1064          }  // Fin EVENT OverflowVI.conversionTimeOverflow
1065
1066
1067          /*
1068          Este evento indica cuando ha ocurrido un desbordamiento en la conversion
1069          (tension e intensidad) debido a problemas en la memoria de escritura.
1070             Tamanio variables = N/A
1071          */
1072          async event void OverflowVI.memOverflow() {
1073
1074          //.......... CODIGO EJECUTABLE
1075
```

```
1076            atomic vi_overflow = TRUE; // Activacion bandera desbordamiento
1077            call Leds.led0On();
1078            perror("ADC12 (VI) Mem OF");
1079
1080         }  // Fin EVENT OverflowVI.memOverflow
1081
1082
1083         /*
1084         Este evento indica cuando ha ocurrido un desbordamiento en la conversion
1085         (sensores auxiliares) debido a problemas de temporizacion.
1086            Tamanio variables = N/A
1087         */
1088         async event void OverflowSENS.conversionTimeOverflow() {
1089
1090         //.......... CODIGO EJECUTABLE
1091
1092            atomic sens_overflow = TRUE;  // Activacion bandera desbordamiento
1093            call Leds.led0On();
1094            perror("ADC12 (SENS) Time OF");
1095
1096         }  // Fin EVENT OverflowSENS.conversionTimeOverflow
1097
1098
1099         /*
1100         Este evento indica cuando ha ocurrido un desbordamiento en la conversion
1101         (sensores auxiliares) debido a problemas en la memoria de escritura.
1102            Tamanio variables = N/A
1103         */
1104         async event void OverflowSENS.memOverflow() {
1105
1106         //.......... CODIGO EJECUTABLE
1107
1108            atomic sens_overflow = TRUE;  // Activacion bandera desbordamiento
1109            call Leds.led0On();
1110            perror("ADC12 (SENS) Mem OF");
1111
1112         }  // Fin EVENT OverflowSENS.memOverflow
1113
1114
1115         /*
1116         El evento tiene lugar cuando se ha montado el volumen determinado de la
1117         memoria externa flash del nodo (previamente se ha debido hacer una
1118         llamada a MountSto.mount).
1119            Tamanio variables = N/A
1120         */
1121         event void MountSto.mountDone(error_t error) {
1122
1123         //.......... CODIGO EJECUTABLE
1124
1125            if (error == SUCCESS) {
1126               // Montado del volumen correcto
1127               if (call ConfigSto.valid() == TRUE) {
1128                  // Estado del volumen correcto
1129                  if (call ConfigSto.read(MEM_ADDR, &permdata,
1130                     sizeof(permdata)) == SUCCESS) {
1131                     // Peticion de lectura de memoria realizada correctamente
1132                     pinfo("ExtMem leyendo volumen");
```

```
1133                  }
1134                  else {
1135                     // Error al intentar leer la memoria
1136                     perror("ExtMem no puede leerse");
1137                  }
1138               }
1139               else {
1140                  // Volumen no valido. Efectuar operacion 'commit'
1141                  perror("ExtMem volumen no valido");
1142
1143                  if (call ConfigSto.commit() == SUCCESS) {
1144                     // Peticion de 'commit' realizada con exito
1145                  }
1146                  else {
1147                     // Error al llamar a 'commit'
1148                  }
1149               }
1150            }
1151            else {
1152               // Error en el montado del volumen
1153               perror("ExtMem no puede montarse volumen");
1154            }
1155
1156         }  // Fin EVENT MountSto.mountDone
1157
1158
1159         /*
1160         El evento se ejecuta cuando se ha efectuado una lectura de la memoria
1161         flash externa del mote (previamente se ha debido realizar la llamada
1162         ConfigSto.read).
1163            Tamanio variables = N/A
1164         */
1165         event void ConfigSto.readDone(storage_addr_t addr, void* buf,
1166          storage_len_t len, error_t err) {
1167
1168         //.......... CODIGO EJECUTABLE
1169
1170            if (err == SUCCESS) {
1171               // Memoria leida correctamente
1172               psuccess("ExtMem volumen montado");
1173               memcpy(&permdata, buf, len);
1174               if (permdata.version == FW_VER) {
1175                  /*
1176                  La version de los datos en memoria es la correcta. Se leen los
1177                  umbrales de distorsion, temperatura y tension y se guardan en las
1178                  variables adecuadas (RAM).
1179                  */
1180                  pinfo("ExtMem version correcta");
1181                  atomic thd_umbral[0] = permdata.thd_threshold[0];
1182                  atomic thd_umbral[1] = permdata.thd_threshold[1];
1183                  atomic temp_umbral = permdata.temp_threshold;
1184                  atomic volt_umbral = permdata.volt_threshold;
1185
1186                  // Arranque de radio (y posteriormente resto de dispositivos)
1187                  if (Arrancar(RADIO) != SUCCESS) {
1188                     perror("Radio no arranca");
1189                  }
```

```
1190                }
1191                else {
1192                   /*
1193                   La version de los datos en memoria no coincide con la actual.
1194                   Se escriben los valores por defecto
1195                   */
1196                   pinfo("ExtMem version incorrecta");
1197
1198                   // Autor
1199                   permdata.author[0] = 0x4A; // J
1200                   permdata.author[1] = 0x41; // A
1201                   permdata.author[2] = 0x54; // T
1202                   permdata.author[3] = 0x47; // G
1203                   permdata.author[4] = 0x00; // NULL char
1204                   permdata.author[8] = 0x00; // NULL char (no quitar linea)
1205
1206                   // Version del firmware instalado
1207                   permdata.version = FW_VER;
1208
1209                   // Escritura de los umbrales por defecto
1210                   permdata.thd_threshold[0] = THD_THRES_VOLT;
1211                   permdata.thd_threshold[1] = THD_THRES_INT;
1212                   permdata.temp_threshold = TEMP_THRES;
1213                   permdata.volt_threshold = VOLT_THRES;
1214
1215                   // Umbrales por defecto a usar por el nodo (RAM)
1216                   atomic thd_umbral[0] = THD_THRES_VOLT;
1217                   atomic thd_umbral[1] = THD_THRES_INT;
1218                   atomic temp_umbral = TEMP_THRES;
1219                   atomic volt_umbral = VOLT_THRES;
1220
1221                   if (call ConfigSto.write
1222                   (MEM_ADDR, &permdata, sizeof(permdata)) == SUCCESS) {
1223                      // Peticion de escritura de nuevos datos correcta
1224                   }
1225                   else {
1226                      // Error al intentar escribir en memoria
1227                   }
1228                }
1229             }
1230             else {
1231                // Ha habido error al leer de la memoria
1232             }
1233
1234          }  // Fin EVENT ConfigSto.readDone
1235
1236
1237          /*
1238          Evento para indicar que se ha escrito en memoria externa flash.
1239          Anteriormente, se ha tenido que llamar al comando ConfigSto.write.
1240             Tamanio variables = N/A
1241          */
1242          event void ConfigSto.writeDone(storage_addr_t addr, void *buf,
1243           storage_len_t len, error_t err) {
1244
1245          //.......... CODIGO EJECUTABLE
1246
```

```
1247            if (err == SUCCESS) {
1248               // Escritura realizada correctamente
1249               if ((addr == MEM_ADDR) && (len == sizeof(permdata))) {
1250                  // Direccion de memoria y tamanio escrito correctos
1251                  psuccess("ExtMem escrita");
1252                  wdt_stop();    // Parada del WDT
1253                  if (call ConfigSto.commit() == SUCCESS) {
1254                     // Peticion 'commit' realizada correctamente
1255                  }
1256                  else {
1257                     // Error al enviar la peticion 'commit'
1258                  }
1259               }
1260               else {
1261                  // Error en la direccion de memoria y/o bytes escritos
1262               }
1263            }
1264            else {
1265               // Error al escribir en memoria
1266            }
1267
1268         }  // Fin EVENT ConfigSto.writeDone
1269
1270
1271         /*
1272         Este evento senializa cuando los datos estan realmente fijados en la
1273         memoria externa flash, es decir, hasta que no se llama al comando
1274         ConfigSto.commit y se recibe este evento, los datos pueden no estar
1275         fijados en la memoria.
1276            Tamanio variables = N/A
1277         */
1278         event void ConfigSto.commitDone(error_t err) {
1279
1280         //.......... CODIGO EJECUTABLE
1281
1282            call Leds.led2Off();
1283            if (err == SUCCESS) {
1284               psuccess("Commit done");
1285               wdt_stop();    // Parada del WDT
1286               if (Arrancar(RADIO) != SUCCESS) {
1287                  pinfo("Radio ya arrancada");
1288                  atomic wr_memory = FALSE;
1289               }
1290               else {
1291                  pinfo("Arrancando radio");
1292               }
1293            }
1294            else {
1295               perror("Commit");
1296            }
1297
1298         }  // Fin EVENT ConfigSto.commitDone
1299
1300      #endif
1301
1302
1303      /*-------------------------------------------------------------------------
```

```
1304
1305                MANEJO DE TAREAS (events)
1306
1307        ------------------------------------------------------------------------------*/
1308
1309        //   TAREAS COMUNES (nodos base y de datos) ----------------------------------
1310
1311           /*
1312           Esta tarea no realiza ninguna accion. El programador de tareas de prioridad
1313           (componente PrioritySchedulerC) falla si no se incluye esta tarea.
1314              Tamanio variables = N/A
1315           */
1316           event void VHT.runTask() {
1317
1318           }  // Fin de EVENT VHT.runTask
1319
1320
1321        //   TAREAS NODOS DE DATOS ---------------------------------------------------
1322
1323        #ifndef ROOT_MOTE
1324
1325           /*
1326           La tarea se encarga de organizar las conversiones que se llevaran a cabo en
1327           el sensor (tanto de V-I como de los sensores auxiliares). Tambien actualiza
1328           los indicadores de buffer de lectura/escritura y de lanzar la tarea de
1329           procesamiento (CalcPotencia).
1330              Tamanio variables = 1B
1331           */
1332           event void Conversion.runTask() {
1333
1334           //.......... VARIABLES Y CONSTANTES LOCALES
1335
1336              static uint8_t num_conv = 0;  // Contador numero conversiones 'correctas'
1337
1338           //.......... CODIGO EJECUTABLE
1339
1340              if (num_conv >= MAXCONV) {
1341                 // Se ha superado el numero de conversiones maximas fijado
1342                 num_conv = 0;
1343                 if (call ResourceVI.release() == SUCCESS) {
1344                    // Liberacion del ADC12 para tension-intensidad correcta
1345                    if (call ResourceSENS.request() == SUCCESS) {
1346                       // Peticion ADC12 para sensores auxiliares correcta
1347                    }
1348                    else {
1349                       // Devuelto EBUSY: ya se ha realizado una solicitud
1350                       pinfo("ADC12 (SENS) ya solicitado");
1351                    }
1352                 }
1353                 else {
1354                    // Error al liberar ADC12 (v-i)
1355                    call Conversion.postTask();
1356                    perror("ADC12 (VI) no liberado");
1357                 }
1358              }
1359              else {
1360                 /*
```

```
1361              (No se ha superado el numero de conversiones maximo)
1362              Se comprueba: que en una conversion anterior no haya habido
1363              desbordamiento del buffer, la tarea que calcula potencia este
1364              finalizada y que el bloque de calculo de la DFT tenga los valores
1365              copiados correctamente.
1366              */
1367              if (vi_overflow==FALSE && calcp_status==0 && dft_status!=1) {
1368                 atomic buflec = bufesc;       // Actualizacion del buffer de lectura
1369                 call CalcPotencia.postTask(); // Llamada a tarea de calculo de pot.
1370                 num_conv++;
1371                 // Actualizacion del buffer de escritura para la proxima conversion
1372                 atomic {
1373                    bufesc++;
1374                    bufesc=bufesc%MAXBUF;
1375                 }
1376              }
1377              else {
1378                 // Debido a desbordamiento, o tareas no finalizadas a tiempo
1379              }
1380
1381
1382              if (call MultiChannelVI.configure
1383              (&configVI, memctlVI, 1, &(raw[bufesc][0]), NUMSAMP, SAMPPER)
1384               == SUCCESS) {
1385                 // Configuracion realizada correctamente
1386                 atomic vi_overflow = FALSE;   // Bajada bandera desbordamiento
1387                 call Leds.led0Off();
1388                 if(call MultiChannelVI.getData() == SUCCESS) {
1389                    // Recogida de datos arrancada correctamente
1390                    atomic adc_status = 1;
1391                    call Leds.led1On();
1392                 }
1393                 else {
1394                    // Error al iniciar las conversiones
1395                    perror("Conversion (VI) no iniciada");
1396                    call Conversion.postTask();
1397                 }
1398              }
1399              else {
1400                 // Error al realizar la configuracion del ADC12
1401                 perror("ADC12 (VI) no configurado");
1402                 call Conversion.postTask();
1403              }
1404
1405           }
1406
1407           wdt_touch();   // Reseteo del WDT
1408
1409        }  // Fin EVENT Conversion.runTask
1410
1411
1412        /*
1413        Tarea para calcular valores significativos a partir de las ondas de
1414        tension e intensidad. Se calculan tension/intensidad medias, valor eficaz
1415        de tension/intensidad, potencia activa y reactiva, energia electrica y
1416        factor de potencia.
1417           Tamanio variables = 52B
```

```
1418          */
1419          event void CalcPotencia.runTask() {
1420
1421          //.......... VARIABLES Y CONSTANTES LOCALES
1422
1423             uint32_t vmed=0;  // Tension media (Vm)
1424             uint32_t imed=0;  // Intensidad media (Im)
1425             uint64_t vrms=0;  // Valor eficaz de tension al cuadrado (Vrms^2)
1426             uint64_t irms=0;  // Valor eficaz de intensidad al cuadrado (Irms^2)
1427             uint64_t pact=0;  // Potencia activa (P)
1428             uint64_t prea=0;  // Potencia reactiva al cuadrado (Q^2)
1429             uint64_t ener=0;  // Energia electrica (E)
1430             uint16_t fdp=0;   // Factor de potencia en % al cuadrado
1431
1432             uint8_t i=0;
1433
1434             static uint8_t contador = 0;
1435
1436          //.......... CODIGO EJECUTABLE
1437
1438             atomic calcp_status = 1;   // Inicio de calculo de potencias
1439
1440             /*
1441             Bucle de calculo de los diferentes indicadores de las ondas de tension e
1442             intensidad. Para la integracion de las senales se usa la regla de Simpson.
1443                Iteraciones: N-2
1444             */
1445             for (i=2; i<=250; i+=4) {
1446                /*
1447                Calculo de valores medios. El punto inicial (indice 0 para la tension)
1448                y el final se tratan fuera del bucle. El segundo punto del intervalo
1449                de integracion tendra indice (2) y se multiplica por el factor 4.
1450                El tercer punto del intervalo coincide con el primero del siguiente,
1451                por ello se multiplica por 2.
1452                Para la intensidad es analogo, con un desfase de indices de +1 respecto
1453                a la tension.
1454                */
1455                vmed+=((raw[buflec][i])<<2)+((raw[buflec][i+2])<<1);
1456                imed+=((raw[buflec][i+1])<<2)+((raw[buflec][i+3])<<1);
1457
1458                /*
1459                Calculo de valores rms al cuadrado. El procedimiento es similar al de
1460                los valores medios, solo que ahora se multiplica cada muestra por si
1461                misma (usando precision de 64b).
1462                */
1463                vrms+=(((uint64_t)(raw[buflec][i])*(uint64_t)(raw[buflec][i]))<<2)+
1464                (((uint64_t)(raw[buflec][i+2])*(uint64_t)(raw[buflec][i+2]))<<1);
1465                irms+=(((uint64_t)(raw[buflec][i+1])*(uint64_t)(raw[buflec][i+1]))<<2)+
1466                (((uint64_t)(raw[buflec][i+3])*(uint64_t)(raw[buflec][i+3]))<<1);
1467
1468                /*
1469                Calculo de la potencia activa. Como las muestras de tension e
1470                intensidad no estan tomadas en el mismo instante, se usa interpolacion
1471                lineal con dos muestras de tension consecutivas. El resultado se
1472                multiplica por la correspondiente muestra de intensidad.
1473                Con los resultados obtenidos, se aplica integracion de Simpson.
1474                */
```

```
1475                pact+=((uint64_t)(raw[buflec][i]+raw[buflec][i+2])*
1476                (uint64_t)(raw[buflec][i+1]))<<2;   // Factor 4
1477                pact+=((uint64_t)(raw[buflec][i+2]+raw[buflec][i+4])*
1478                (uint64_t)(raw[buflec][i+3]))<<1;   // Factor 2 (extremo doble)
1479
1480            }  // Fin bucle FOR (indice i)
1481
1482            /*
1483            Fin calculo valores medios. Se añade la ultima muestra (afectada por el
1484            factor 4). Para completar el periodo de integracion de la senal, se asume
1485            que la ultima muestra (tendria indice 256) coincide con la primera
1486            (ind. 0) de ahi el factor 2 que se usa.
1487            La intensidad tiene un desfase de muestras de +1.
1488            */
1489            vmed+=((raw[buflec][0])<<1)+((raw[buflec][254])<<2);
1490            imed+=((raw[buflec][1])<<1)+((raw[buflec][255])<<2);
1491
1492            /*
1493            Fin calculo valores eficaces (cuadrado). Se usa la misma aproximacion que
1494            en los valores medios. De nuevo se usa precision de 64b.
1495            */
1496            vrms+=(((uint64_t)(raw[buflec][0])*(uint64_t)(raw[buflec][0]))<<1)+
1497            (((uint64_t)(raw[buflec][254])*(uint64_t)(raw[buflec][254]))<<2);
1498            irms+=(((uint64_t)(raw[buflec][1])*(uint64_t)(raw[buflec][1]))<<1)+
1499            (((uint64_t)(raw[buflec][255])*(uint64_t)(raw[buflec][255]))<<2);
1500
1501            /*
1502            Fin calculo de la potencia activa. Tambien se usa la aproximacion de la
1503            ultima muestra del periodo (ind. 256) por la primera (0).
1504            */
1505            pact+=(uint64_t)(raw[buflec][0]+raw[buflec][2])*
1506            (uint64_t)(raw[buflec][1]);
1507            pact+=(uint64_t)(raw[buflec][254]+raw[buflec][0])*
1508            (uint64_t)(raw[buflec][255]);
1509
1510            /*
1511            Ajuste de los valores de integracion. La regla de Simpson tiene un factor
1512            de h/3, donde h=625us (328/2^19). Hay que dividir entre el tiempo de
1513            integracion, que resulta 4*T=128*h (80ms). El factor a aplicar quedaria
1514            1/(128*3). Hay que multiplicar por 10 para usar la macro de redondeo,
1515            resultando 5/192.
1516            */
1517            vmed=(5*vmed)/192;
1518            imed=(5*imed)/192;
1519            vrms=(5*vrms)/192;
1520            irms=(5*irms)/192;
1521            /*
1522            En potencia activa se debe de dividir ademas entre 2 debido a la
1523            interpolacion lineal realizada (factor 5/384).
1524            */
1525            pact=(5*pact)/384;
1526
1527            // Redondeos de variables
1528            round(vmed);
1529            round(imed);
1530            round(vrms);
1531            round(irms);
```

```
1532            round(pact);
1533
1534            /*
1535            En caso de la energia electrica, basta multiplicar la potencia activa por
1536            el tiempo de integracion 4*T=80ms (328/2**12). Para usar redondeo, hay
1537            que multiplicar la cantidad por 10, resultando un factor 205/256.
1538            Este calculo debe realizarse DESPUES del de potencia activa.
1539            */
1540            ener=(205*pact)>>8;
1541            round(ener);
1542
1543            /*
1544            Calculo de la potencia reactiva (cuadrado). Se resta a la potencia
1545            aparente S (Vrms*Irms) el cuadrado de la potencia activa, todo en
1546            precision 64b.
1547            */
1548            prea=(uint64_t)((uint64_t)vrms*(uint64_t)irms)-
1549            (uint64_t)((uint64_t)pact*(uint64_t)pact);
1550
1551            /*
1552            Calculo del factor de potencia (cuadrado). Se divide la potencia activa al
1553            cuadrado entre la aparente tambien al cuadrado. El resultado se da en
1554            tanto por ciento al cuadrado.
1555            */
1556            fdp=(uint16_t)((uint64_t)(10000*(uint64_t)pact*(uint64_t)pact)/
1557            (uint64_t)((uint64_t)vrms*(uint64_t)irms));
1558
1559            /*
1560            El bloque switch determina los cuatro bloques de recogida de datos que
1561            se realizan, segun el valor de la constante 'MAXCONV'.
1562            Los datos grabados seran enviados posteriormente via radio.
1563            */
1564            switch (contador) {
1565               case 0:
1566                  atomic vmeddata[0] = (uint16_t)vmed;
1567                  atomic imeddata[0] = (uint16_t)imed;
1568                  atomic vrmsdata[0] = (uint32_t)vrms;
1569                  atomic irmsdata[0] = (uint32_t)irms;
1570                  atomic pactdata[0] = (uint32_t)pact;
1571                  atomic fdpdata[0] = (uint16_t)fdp;
1572                  break;
1573
1574               case (MAXCONV/4):
1575                  atomic vmeddata[1] = (uint16_t)vmed;
1576                  atomic imeddata[1] = (uint16_t)imed;
1577                  atomic vrmsdata[1] = (uint32_t)vrms;
1578                  atomic irmsdata[1] = (uint32_t)irms;
1579                  atomic pactdata[1] = (uint32_t)pact;
1580                  atomic fdpdata[1] = (uint16_t)fdp;
1581                  break;
1582
1583               case (MAXCONV/2):
1584                  atomic vmeddata[2] = (uint16_t)vmed;
1585                  atomic imeddata[2] = (uint16_t)imed;
1586                  atomic vrmsdata[2] = (uint32_t)vrms;
1587                  atomic irmsdata[2] = (uint32_t)irms;
1588                  atomic pactdata[2] = (uint32_t)pact;
```

```
1589                atomic fdpdata[2] = (uint16_t)fdp;
1590                break;
1591
1592             case (3*MAXCONV/4):
1593                atomic vmeddata[3] = (uint16_t)vmed;
1594                atomic imeddata[3] = (uint16_t)imed;
1595                atomic vrmsdata[3] = (uint32_t)vrms;
1596                atomic irmsdata[3] = (uint32_t)irms;
1597                atomic pactdata[3] = (uint32_t)pact;
1598                atomic fdpdata[3] = (uint16_t)fdp;
1599                break;
1600          }
1601
1602          // Actualizacion del contador de conversiones
1603          contador++;
1604          contador=contador%MAXCONV;
1605
1606          if (dft_status == 0) {
1607             // Se llama a la tarea de calculo DFT solo si esta ha terminado
1608             call CalcDFT.postTask();
1609          }
1610
1611          atomic calcp_status = 0;   // Finalizacion del calculo de potencia
1612
1613       }  // Fin de EVENT CalcPotencia.runTask
1614
1615
1616       /*
1617       Tarea de ejecucion del algoritmo FFT. Realiza un calculo de la DFT a
1618       traves de un algoritmo de Transformada Rapida de Fourier (FFT).
1619       Los valores iniciales de la secuencia se encuentran en el vector 'y' end
1620       forma de parte real e imaginaria. Al final del proceso, los resultados de
1621       la FFT se encuentran tambien en dicho vector.
1622       Despues se realiza el computo del modulo de cada bin de frecuencia, asi
1623       como otros calculos relacionados.
1624          Tamanio variables: (19*N+R+8*ARM+65)B
1625       */
1626       event void CalcDFT.runTask() {
1627
1628       //.......... VARIABLES Y CONSTANTES LOCALES
1629
1630          /*
1631          Coeficientes de la ventana Blackman a aplicar a la secuencia de entrada.
1632          Si el tamano de la secuencia es 2N, solo se almacena la mitad, pues la
1633          ventana tiene simetria respecto a la muestra central N.
1634          Los valores originales se han multiplicado por 2^18 y posteriormente
1635          aplicado un redondeo.
1636             Componentes = N
1637             Tamanio = (4*N)B
1638          */
1639    #ifdef DEBUGMODE
1640          static uint32_t wndw[] = {
1641    #else
1642          const uint32_t wndw[] = {
1643    #endif
1644          0U,58U,232U,524U,937U,1475U,2144U,2949U,
1645          3899U,5000U,6261U,7692U,9302U,11102U,13100U,15308U,
```

```
1646            17734U,20388U,23279U,26414U,29801U,33446U,37353U,41525U,
1647            45965U,50673U,55647U,60884U,66378U,72123U,78109U,84324U,
1648            90756U,97390U,104207U,111188U,118313U,125558U,132899U,140310U,
1649            147762U,155228U,162677U,170078U,177401U,184613U,191682U,198577U,
1650            205264U,211712U,217892U,223771U,229322U,234517U,239329U,243734U,
1651            247709U,251235U,254292U,256864U,258939U,260504U,261553U,262078U };
1652
1653            /*
1654            Vector de aplicacion de inversion binaria. Reordenamiento de las
1655            componentes de las muestras de la senal para conseguir que el resultado
1656            de aplicar la DFT este en el orden correcto.
1657            Generado a partir de N muestras, y multiplicado su valor por 2, con el fin
1658            de poder separar las muestras en parte real e imaginaria mas facilmente.
1659               Componentes = N
1660               Tamanio = (N)B
1661            */
1662        #ifdef DEBUGMODE
1663            static uint8_t order[] = {
1664        #else
1665            const uint8_t order[] = {
1666        #endif
1667            0,64,32,96,16,80,48,112,8,72,40,104,24,88,56,120,
1668            4,68,36,100,20,84,52,116,12,76,44,108,28,92,60,124,
1669            2,66,34,98,18,82,50,114,10,74,42,106,26,90,58,122,
1670            6,70,38,102,22,86,54,118,14,78,46,110,30,94,62,126 };
1671
1672            /*
1673            Valores de las raices complejas de la unidad.
1674            Solo se representa la parte real de la raiz (coseno). La parte imaginaria
1675            se obtiene rotando las muestras del vector.
1676            Los coeficientes originales han sido multiplicados por 32767 (maximo
1677            permitido para enteros con signo de 16b) y despues se aplica un redondeo.
1678               Componentes = N
1679               Tamanio = (2*N)B
1680            */
1681        #ifdef DEBUGMODE
1682            static int16_t expcomp[] = {
1683        #else
1684            const int16_t expcomp[] = {
1685        #endif
1686            32767,32728,32609,32412,32137,31785,31356,30852,
1687            30273,29621,28898,28105,27245,26319,25329,24279,
1688            23170,22005,20787,19519,18204,16846,15446,14010,
1689            12539,11039,9512,7962,6393,4808,3212,1608,
1690            0,-1608,-3212,-4808,-6393,-7962,-9512,-11039,
1691            -12539,-14010,-15446,-16846,-18204,-19519,-20787,-22005,
1692            -23170,-24279,-25329,-26319,-27245,-28105,-28898,-29621,
1693            -30273,-30852,-31356,-31785,-32137,-32412,-32609,-32728 };
1694
1695            /*
1696            Vector de potencias para el indice 'q' del algoritmo FFT. Se memorizan
1697            los valores necesarios para 2^(m-1), con m=1,2,3...r.
1698               Componentes = R
1699               Tamanio = (R)B
1700            */
1701            const uint8_t pot[] = {1,2,4,8,16,32};
1702
```

```
1703            /*
1704            Matriz para alojar la transformada compleja. El segundo indice indica si
1705            se trata de parte real (0) o imaginaria (1). En teoria, la DFT resultante
1706            tendria 2N componentes. sin embargo, al ser la secuencia de entrada real,
1707            solo bastan N+1 componentes para definirla totalmente (simetrias).
1708               Componentes = (N+1) dobles (real e imaginaria)
1709               Tamanio = (8*(N+1))B
1710            */
1711            static int32_t y[N+1][2];
1712
1713            /*
1714            Vector para almacenar los modulos (al cuadrado) del resto de las
1715            componentes de la transformada DFT.
1716               Componentes = N
1717               Tamanio = (4*N)B
1718            */
1719            static uint32_t modulus[N];
1720
1721            // Modulo (cuadrado) de la componente continua (DC)
1722            static uint32_t modulusdc;
1723
1724            /*
1725            Variable para indicar el canal analogico a recuperar en DFT. El algoritmo
1726            FFT solo procesa un canal, de forma que cada vez que se llama trabaja con
1727            datos de tension e intensidad alternativamente (A0 y A1 respec.)
1728            */
1729            static uint8_t chan = 0;
1730
1731            /*
1732            Vector para guardar la estimacion de frecuencia fundamental de las ondas
1733            de tension (0) e intensidad (1).
1734               Componentes = 2
1735               Tamanio = 4B
1736            */
1737            static uint16_t freq[2];
1738
1739            /*
1740            Array bidimensional para almacenar los indicadores de distorsion armonica.
1741            La primera componente indica si el valor es para el armonico de tension
1742            (canal analogico 0) o de intensidad (canal A1).
1743               Componentes = 2*ARM
1744               Tamano = (8*ARM)B
1745            */
1746            static uint32_t inddist[2][ARM];
1747
1748            /*
1749            Array de dos componentes para almacenar el valor de la tasa de distorsion
1750            armonica total (THD).
1751            La primera componente (0) guarda el valor de tension y la segunda (1) el
1752            de intensidad.
1753               Componentes  2
1754               Tamano = 16B
1755            */
1756            static uint64_t thd[2];
1757
1758            /*
1759            Variables para el control de la distorsion de acuerdo a los limites
```

```
1760            establecidos.
1761            'lbcounter' es la variable contador del algoritmo leaky
1762            bucket. Si se supera el limite LB_UPT, se indica que existe exceso de
1763            distorsion en el sistema. Para volver al estado normal, es necesario que
1764            el valor de 'lbcounter' disminuya hasta LB_LWT (histeresis).
1765            'dft_calc' contabiliza el numero de procesamientos DFT (tanto de
1766            tension como intensidad) realizados. Cada vez que se llega a LB_D
1767            calculos, decrementa 'lbcounter' en una unidad, y se reinicia a 0.
1768            */
1769            static uint8_t lbcounter = 0;
1770            static uint8_t dft_calc = 0;
1771
1772            // Indices auxiliares del algoritmo
1773            uint8_t m=0;
1774            uint8_t q=0;
1775            uint8_t k=0;
1776            uint8_t n=0;
1777            uint8_t inda=0;
1778            uint8_t indb=0;
1779
1780            // Almacenamiento de la parte real e imag. de la exponencial compleja
1781            int32_t ur=0;
1782            int32_t ui=0;
1783
1784            // Variables auxiliares (valores intermedios)
1785            int32_t auxa=0;
1786            int32_t auxb=0;
1787            int32_t auxc=0;
1788            int32_t auxd=0;
1789
1790         //.......... CODIGO EJECUTABLE
1791
1792            switch (dft_status) {
1793               case 0:  // Estado inicial
1794
1795                  atomic dft_status = 1;  // Copiando datos a vector 'y'
1796
1797                  for (k=0; k<N; k+=2) {
1798                     // Tratamiento muestras PARES
1799                     // Parte real de las muestras 0,2,4,...,N-2
1800                     y[k][0]=(int32_t)
1801                     (((uint64_t)(raw[buflec][(order[k]<<1)+chan])*10*
1802                     (uint64_t)wndw[order[k]])>>18);
1803                     // Parte imaginaria de las muestras 0,2,4,...,N-2
1804                     y[k][1]=(int32_t)
1805                     (((uint64_t)(raw[buflec][(order[k]<<1)+2+chan])*10*
1806                     (uint64_t)wndw[order[k]+1])>>18);
1807
1808                     // Tratamiento muestras IMPARES
1809                     // Parte real de las muestras 1,3,5,...,N-1
1810                     y[k+1][0]=(int32_t)
1811                     (((uint64_t)(raw[buflec][(order[k+1]<<1)+chan])*10*
1812                     (uint64_t)wndw[NN-1-order[k+1]])>>18);
1813                     // Parte imaginaria de las muestras 1,3,5,...,N-1
1814                     y[k+1][1]=(int32_t)
1815                     (((uint64_t)(raw[buflec][(order[k+1]<<1)+2+chan])*10*
1816                     (uint64_t)wndw[NN-2-order[k+1]])>>18);
```

```
1817
1818                    // Redondeos de las componentes
1819                    round(y[k+1][1]);
1820                    round(y[k][1]);
1821                    round(y[k+1][0]);
1822                    round(y[k][0]);
1823
1824                 }  // Fin bucle FOR (indice k)
1825
1826                 call CalcDFT.postTask();
1827                 atomic dft_status = 2;  // Datos copiados correctamente
1828                 break;
1829
1830              case 2:  //
1831
1832                 atomic dft_status = 3;  // Calculando DFT de la secuencia compleja
1833
1834                 /*
1835                 Bucle principal que realiza el calculo de la DFT a la secuencia
1836                 (compleja y de tamanio N)
1837                 */
1838                 for (m=1; m<=R; m++) {     // m=1,2,3...R (N=2^R). Iteraciones: R
1839                    /* Expresion teorica: q=2**(m-1). Los valores estan almacenados
1840                    en el vector 'pot'.  */
1841                    q=pot[m-1];
1842                    for (k=0; k<q; k++) {   // k=0,1,2...q-1. Iteraciones: q
1843                       // Expresion de u: exp(-j*pi*k/q) (exponencial compleja)
1844                       ur=expcomp[k*(uint8_t)(N/q)]; // Parte real
1845                       // Calculo de la parte imaginaria
1846                       if(2*k<q) {
1847                          // Casos k=0,1,2...q/2-1
1848                          ui=expcomp[NMED+k*(uint8_t)(N/q)];
1849                       }
1850                       else {
1851                          // Casos k=q/2,q/2+1...q-1
1852                          ui=-expcomp[-NMED+k*(uint8_t)(N/q)];
1853                       }
1854                       for (n=0; n<N; n+=(2*q)) {
1855                       // n=0,2q,4q...N-1 (incremento de 2q). Iteraciones: N/(2q)
1856                          inda=n+k;   // Indice auxiliar n+k
1857                          indb=n+k+q; // Indice auxiliar n+k+q
1858                          auxa=(int32_t)((((int64_t)ur*(int64_t)y[indb][0]-
1859                          (int64_t)ui*(int64_t)y[indb][1])*10)/32767);
1860                          auxb=(int32_t)((((int64_t)ur*(int64_t)y[indb][1]+
1861                          (int64_t)ui*(int64_t)y[indb][0])*10)/32767);
1862                          round(auxa); round(auxb);
1863                          y[indb][0]=y[inda][0]-auxa;            // Re{ y(n+k+q) }
1864                          y[indb][1]=y[inda][1]-auxb;            // Im{ y(n+k+q) }
1865                          y[inda][0]=(y[inda][0]<<1)-y[indb][0]; // Re{ y(n+k) }
1866                          y[inda][1]=(y[inda][1]<<1)-y[indb][1]; // Im{ y(n+k) }
1867                       } // Fin del tercer bucle FOR (indice n)
1868                    } // Fin del segundo bucle FOR (indice k)
1869                 } // Fin bucle FOR principal (indice m)
1870
1871                 call CalcDFT.postTask();
1872                 atomic dft_status = 4;  // DFT sec. compleja calculada correctamente
1873                 break;
```

```
1874
1875                case 4:  //
1876
1877                   atomic dft_status = 5;  // Calculando DFT secuencia original
1878
1879                   /*
1880                   En este punto, se tiene calculada la DFT de la secuencia compleja de
1881                   tamano N, formada a partir de la muestra original real de tamano 2N.
1882                   El siguiente bloque calcula la DFT de la secuencia original a
1883                   partir de esta ultima.
1884                   */
1885
1886                   /* Ultima componente G(N)
1887                   Solo hay que modificar la parte real (imaginaria=0). Hay que dividir
1888                   entre 64 (N) y multiplicar por 2**16, es decir, multiplicar por
1889                   2**10. Despues, aplicar la ganancia coherente y redondear.
1890                   */
1891                   y[N][0]=(y[0][0]-y[0][1])<<10;
1892                   y[N][0]/=CGAIN_ADJ;  // Ajuste de la ganancia coherente
1893                   round(y[N][0]);
1894                   y[N][1]=0;
1895
1896                   /* Primera componente G(0)
1897                   Solo hay que modificar la parte real (imaginaria=0). En este caso,
1898                   hay que dividir entre 128 (2N) y multiplicar por 2**16, esto es,
1899                   multiplicar por 2**9. Despues, aplicar la ganancia coherente y
1900                   redondear.
1901                   */
1902                   y[0][0]=(y[0][0]+y[0][1])<<9;
1903                   y[0][0]/=CGAIN_ADJ;  // Ajuste de la ganancia coherente
1904                   round(y[0][0]);
1905                   y[0][1]=0;
1906
1907                   /* Componente G(N/2)
1908                   Dividir entre 128 (2N) y multiplicar por 2**16 (i.e. multiplicar
1909                   por 2**9). Despues, aplicar la ganancia coherente y redondear.
1910                   */
1911                   y[NMED][1]=((-y[NMED][1])<<9)/CGAIN_ADJ; // Cambio de signo IMAG
1912                   y[NMED][0]=(y[NMED][0]<<9)/CGAIN_ADJ;
1913                   round(y[NMED][0]);
1914                   round(y[NMED][1]);
1915
1916                   /*
1917                   Bucle para calcular el resto de componentes. Se realizan el calculo
1918                   de dos componentes cada vez: G(k) y G(N-k), aprovechando las
1919                   propiedades de simetria.
1920                   Es decir: G(1) y G(N-1), G(2) y G(N-2)... hasta G(N/2-1) y G(N/2+1).
1921                   Los resultados sobreescriben a los anteriores (vector 'y').
1922                      Iteraciones: N/2-1
1923                   */
1924                   for (k=1; k<NMED; k++) {   // k=1,2,3...N/2-1
1925                      ur=expcomp[k];          // Parte real de la exp. compleja
1926                      ui=expcomp[NMED+k];     // Parte imaginaria
1927                      // Valores auxiliares para los calculos
1928                      auxa=y[k][0]+y[N-k][0];
1929                      auxb=y[k][1]+y[N-k][1];
1930                      auxc=y[N-k][0]-y[k][0];
```

```
1931                    auxd=y[k][1]-y[N-k][1];
1932
1933                    /* Operaciones a realizar: dividir entre 128 (2N) y multiplicar
1934                    por 2**16, es decir, multiplicar por 2**9. Posteriormente,
1935                    ajustar la ganancia coherente y redondear.
1936                    */
1937                    y[k][0]=((auxa+(int32_t)(((int64_t)ur*(int64_t)auxb-
1938                    (int64_t)ui*(int64_t)auxc)/32767))<<9)/CGAIN_ADJ;  // Re{G(k)}
1939                    y[k][1]=((auxd+(int32_t)(((int64_t)ur*(int64_t)auxc+
1940                    (int64_t)ui*(int64_t)auxb)/32767))<<9)/CGAIN_ADJ;  // Im{G(k)}
1941                    y[N-k][0]=((auxa+(int32_t)(((int64_t)ui*(int64_t)auxc-
1942                    (int64_t)ur*(int64_t)auxb)/32767))<<9)/CGAIN_ADJ;  // Re{G(N-k)}
1943                    y[N-k][1]=((-auxd+(int32_t)(((int64_t)ur*(int64_t)auxc+
1944                    (int64_t)ui*(int64_t)auxb)/32767))<<9)/CGAIN_ADJ;  // Im{G(N-k)}
1945
1946                    round(y[k][0]);
1947                    round(y[k][1]);
1948                    round(y[N-k][0]);
1949                    round(y[N-k][1]);
1950
1951                 }  // Fin bucle FOR (indice k)
1952
1953                 call CalcDFT.postTask();
1954                 atomic dft_status = 6;  // DFT secuencia original calculada OK
1955                 break;
1956
1957              case 6:
1958
1959                 atomic dft_status = 7;  // Calculando parametros armonicos
1960
1961                 dft_calc ++;   // Otro calculo DFT realizado
1962
1963                 // Modulo (al cuadrado) de la componente de continua (frec=0)
1964                 modulusdc = y[0][0]*y[0][0];
1965
1966                 /*
1967                 Bucle de calculo de los modulos (elevados al cuadrado) del resto de
1968                 componentes (armonicos). En total se tienen N valores.
1969                 Si la frecuencia de muestreo de los valores es 'fs', la componente
1970                 k-esima (comenzando en 0) corresponde a la frecuencia
1971                 ((k+1)*fs)/(2N).
1972                    Iteraciones: N
1973                 */
1974                 for (k=1; k<=N; k++) {
1975                    modulus[k-1]=(uint32_t)((y[k][0]*y[k][0])+(y[k][1]*y[k][1]));
1976                 }  // Fin bucle FOR (indice k)
1977
1978
1979                 /*
1980                 Calculo de la frecuencia fundamental estimada.
1981                 Se usa una interpolacion de segundo grado, usando el armonico a la
1982                 frecuencia fundamental (50Hz) y los inmediatamente anterior y
1983                 posterior.
1984                 Se calcula tanto para onda de tension como intensidad ('chan').
1985                 */
1986                 freq[chan]=(uint16_t)((((9*y[3][1]+(y[4][1]<<4)+7*y[5][1])<<8)*100)/
1987                 (41*(y[3][1]+(y[4][1]<<1)+y[5][1])));
```

```
1988
1989                /*
1990                Bucle de calculo de los indicadores de distorsion armonica
1991                individual. Se trabaja con los armonicos a multiplos de la
1992                frecuencia fundamental (50 Hz). Debido a que el vector 'modulus'
1993                contiene mas componentes ademas de los armonicos fundamentales, se
1994                deben de saltar dichas componentes (operacion 4*k+3 del bucle).
1995                El resultado se da en tanto por mil y despues elevado al cuadrado.
1996                El almacenamiento en el vector 'inddist' se produce segun la
1997                variable 'chan' (canal A0-v o A1-i).
1998                   Iteraciones: ARM
1999                */
2000                for (k=0; k<ARM; k++) {
2001                   inddist[chan][k]=(uint32_t)(((uint64_t)1000000*
2002                   (uint64_t)(modulus[4*k+3]))/modulus[3]);
2003                }  // Fin bucle FOR (indice k)
2004
2005                call CalcDFT.postTask();
2006                atomic dft_status = 8;
2007                break;
2008
2009             case 8:
2010
2011                atomic dft_status = 9;
2012
2013                /*
2014                Bucle para hallar la tasa de distorsion total armonica o THD.
2015                Al igual que el bucle anterior, es necesario pasar por alto las
2016                componentes que no se correspondan con frecuencias fundamentales
2017                (50, 100, 150...).
2018                El resultado se da en tanto por mil y elevado al cuadrado,
2019                guardado segun el valor de la variable 'chan'
2020                (canal A0-v o A1-i).
2021                   Iteraciones: ARM-1
2022                */
2023                thd[chan]=0;
2024                for (k=1; k<ARM; k++) {
2025                   thd[chan]+=(uint64_t)(modulus[4*k+3]);
2026                }  // Fin bucle FOR (indice k)
2027                thd[chan]=((uint64_t)1000000*(uint64_t)thd[chan])/
2028                ((uint64_t)modulus[3]);
2029
2030                atomic thddata[chan] = (uint32_t)thd[chan];
2031
2032                // Actualizacion del proximo canal a tratar (alternativo 0-1-0-1...)
2033                atomic {
2034                   chan++;
2035                   chan=chan%2;
2036                }
2037
2038                /*
2039                Bloque para el control de distorsion. thd[0] controla la tension y
2040                thd[1] la intensidad. En caso de que se supere/n el/los umbral/es
2041                establecido/s, se incrementa la variable 'lbcounter' en una unidad.
2042                */
2043                if (thd[0]>thd_umbral[0] || thd[1]>thd_umbral[1]) {
2044                   increase(&lbcounter);
```

```
2045                }
2046
2047                /*
2048                Se comprueban los umbrales maximos y minimos. Caso de que se llegue
2049                al maximo, se activa la salida digital asociada. Para desactivarla,
2050                es necesario que 'lbcounter' disminuya hasta el valor de umbral
2051                minimo.
2052                */
2053                if (lbcounter >= LB_UPT) {
2054                   call CargaB.set();
2055                   pinfo("HIGH dist.");
2056                }
2057                else if (lbcounter <= LB_LWT) {
2058                   call CargaB.clr();
2059                   pinfo("LOW dist.");
2060                }
2061
2062                if (dft_calc >= LB_D) {
2063                   dft_calc = 0;  // Reinicio del contador de procesamientos DFT
2064                   decrease(&lbcounter);   // Factor de olvido disminuye 'lbcounter'
2065                }
2066
2067
2068                atomic dft_status = 0;  // Indicadores armonicos calculados OK.
2069
2070                // Fin calculo DFT
2071
2072                break;
2073
2074             }  // Fin de SWITCH (dft_status)
2075
2076          }  // Fin de EVENT CalcDFT.runTask
2077
2078
2079          /*
2080          Tarea para procesar los sensores auxiliares. Cada vez que se ejecuta, se
2081          realizan las lecturas de los sensores auxiliares (array 'sensor').
2082          Se toman 16 valores consecutivos para cada sensor (en total 16*NUMSENS
2083          conversiones). Despues, se aplica un procedimiento de sobremuestreo para
2084          conseguir dos bits adicionales de resolucion (14b en total).
2085          Por ultimo, dependiendo de la magnitud a medir, se le aplican factores de
2086          conversion para conseguir unos valores mas comodos de interpretar.
2087             Tamanio variables = 17B
2088          */
2089          event void ProcSensor.runTask() {
2090
2091          //.......... VARIABLES Y CONSTANTES LOCALES
2092
2093             uint32_t temp = 0;   // Temperatura interna del MSP430 (decimas de ºC)
2094             uint32_t volt = 0;   // Tension de alimentacion AVcc (mV)
2095             uint32_t sen1 = 0;   // Sensor auxiliar n.1 [NO DEFINIDO]
2096             uint32_t sen2 = 0;   // Sensor auxiliar n.2 [NO DEFINIDO]
2097             uint8_t i=0;         // Contador auxiliar bucle for
2098
2099          //.......... CODIGO EJECUTABLE
2100
2101             atomic sens_status = 1; // Flag indicando que la tarea se esta ejecutando
```

```
2102
2103            if (sens_overflow == FALSE) { // No ha habido desbordamiento del ADC12
2104               pinfo("Leyendo sensores auxiliares");
2105
2106               /*
2107               Bucle para recorrer el vector de medidas del ADC12, sumando todos los
2108               valores para cada sensor respectivamente.
2109                  Iteraciones: NUMSAMPSENS/NUMSENS
2110               */
2111               for (i=0; i<NUMSAMPSENS; i+=NUMSENS) {
2112                  temp+=(uint32_t)sensor[i];
2113                  volt+=(uint32_t)sensor[i+1];
2114                  sen1+=(uint32_t)sensor[i+2];
2115                  sen2+=(uint32_t)sensor[i+3];
2116               }  // Fin bucle FOR (indice i)
2117
2118               // Ajustes mediante factores para los sensores
2119               temp=((F_TEMP_A*(temp>>2))-F_TEMP_B)/F_TEMP_C; // Temperatura
2120               volt=(F_VOLT_A*(volt>>2))/F_VOLT_B; // Tension de alimentacion
2121
2122               // Almacenamiento para enviar mensaje
2123               atomic tempvoltdata[0]=(uint16_t)temp;
2124               atomic tempvoltdata[1]=(uint16_t)volt;
2125
2126               // Gestion de los umbrales de temperatura y tension alimentacion
2127               if (temp >= (uint32_t)temp_umbral) {
2128                  pinfo("Exceso TEMP");
2129               }
2130               if (volt >= (uint32_t)volt_umbral) {
2131                  pinfo("Exceso TENSION");
2132               }
2133
2134            }
2135
2136            // Pedir liberacion del recurso ADC12 para sensores auxiliares
2137            if (call ResourceSENS.release() == SUCCESS) {
2138               // Peticion de liberacion correcta
2139               atomic sens_status = 0;
2140               call EnvioDatos.postTask();   // Tarea de envio de datos radio
2141
2142               /*
2143               En caso de que haya nuevos valores (p.ej. distorsion maxima) se
2144               almacenan en la memoria flash externa no volatil.
2145               */
2146               if (wr_memory  && (adc_status == 0)) {
2147                  if (call ConfigSto.write(0, &permdata, sizeof(permdata))
2148                  == SUCCESS){
2149                     wdt_stop(); // Desactivar momentaneamente el WDT
2150                     call Leds.led2On();
2151                     pinfo("ExtMem escribiendo");
2152                  }
2153                  else
2154                  {
2155                     perror("ExtMem no se puede escribir");
2156                  }
2157               }
2158
```

```
2159            }
2160            else {
2161               perror("ADC12 (SENS) no liberado");
2162            }
2163
2164         }  // Fin de EVENT ProcSensor.runTask
2165
2166
2167         /*
2168         Esta tarea se encarga del envio de datos a traves de la radio (hacia el
2169         nodo base). Se comprueba si el resto de recursos estan libres (para no
2170         interferir con ellos). Si no es asi, se pospone el envio del mensaje.
2171         Recoge los datos procesados por las tareas de calculo (potencia y DFT),
2172         los empaqueta y se llama al envio del mensaje.
2173            Tamanio variables = 3B
2174         */
2175         event void EnvioDatos.runTask() {
2176
2177         //.......... VARIABLES Y CONSTANTES LOCALES
2178
2179            uint8_t i = 0;                // Variable bucle for
2180            PotenciaMsg* potpkt = NULL;   // Puntero a mensaje de potencia
2181
2182         //.......... CODIGO EJECUTABLE
2183
2184            if (wr_memory == FALSE) {
2185               wdt_touch();   // Reseteo del WDT
2186            }
2187
2188            if (calcp_status == 0 && dft_status == 0 && send_status == 0) {
2189
2190               potpkt = (PotenciaMsg*)call
2191                Send.getPayload(&radiopkt,sizeof(PotenciaMsg));
2192
2193               /*
2194               Traspaso de datos desde variables del programa al paquete de datos
2195               */
2196
2197               potpkt -> nodeid = TOS_NODE_ID;
2198               /*
2199               Bucle para recoger las medidas de cada variable de la tarea de calculo
2200               de potencias.
2201                  Iteraciones: 4
2202               */
2203               for (i=0; i<4; i++) {
2204                  potpkt -> vmedia[i] = vmeddata[i];
2205                  potpkt -> imedia[i] = imeddata[i];
2206                  potpkt -> vrms[i] = vrmsdata[i];
2207                  potpkt -> irms[i] = irmsdata[i];
2208                  potpkt -> potact[i] = pactdata[i];
2209                  potpkt -> fdp[i] = fdpdata[i];
2210               }  // Fin bucle FOR (indice i)
2211
2212               /*
2213               Bucle para recoger las medidas de THD (distorsion armonica),
2214               temperatura y tension de alimentacion.
2215                  Iteraciones: 2
```

```
2216                */
2217                for (i=0; i<2; i++) {
2218                   potpkt -> thd[i] = thddata[i];
2219                   potpkt -> tempvolt[i] = tempvoltdata[i];
2220                }  // Fin bucle FOR (indice i)
2221
2222                if (call Send.send(&radiopkt, sizeof(PotenciaMsg)) == SUCCESS) {
2223                   atomic send_status = 1; // Radio ocupada transmitiendo
2224                   call Leds.led2On();
2225                   pinfo("Radio enviando datos");
2226                }
2227                else {
2228                   perror("Radio. Envio no realizado");
2229                }
2230
2231             }
2232             else {
2233                pinfo("Radio. Envio de datos pospuesto");
2234                call EnvioDatos.postTask();   // Se vuelve a programar la tarea
2235             }
2236
2237          }  // Fin de EVENT EnvioDatos.runTask
2238
2239
2240          /*
2241          Tarea encargada de recibir mensajes de diseminacion (enviados por el nodo
2242          base o raiz). Comprueba si efectivamente el mensaje es para el nodo
2243          correcto ('nodeid'). Si es asi, el comportamiento viene dado por el valor
2244          del campo 'function'.
2245          Puede ocurrir que haya que actualizar los valores almacenados en la
2246          memoria flash externa (bandera 'wr_memory').
2247             Tamanio variables = 2B
2248          */
2249          event void ActValores.runTask() {
2250
2251          //.......... VARIABLES Y CONSTANTES LOCALES
2252
2253             const DissMsg* newdissmsg = call ValorDis.get();
2254
2255          //.......... CODIGO EJECUTABLE
2256
2257             /*
2258             Comprobacion del destino del mensaje. Para difundir a toda la red de
2259             diseminacion, se usa la constante BROAD_ID. En caso contrario, el valor
2260             de 'nodeid' indica el nodo destino
2261             */
2262             if (((newdissmsg -> nodeid) == TOS_NODE_ID) ||
2263              ((newdissmsg -> nodeid) == BROAD_ID)) {
2264                // Es el nodo destino (o es difusion)
2265                switch (newdissmsg -> function) {
2266                   case 1:     // Control de carga A digitalmente
2267                      if ((newdissmsg -> valueA) == 0) {
2268                         call CargaA.clr();
2269                      }
2270                      else {
2271                         call CargaA.set();
2272                      }
```

```
2273                    break;
2274
2275                case 2:     // Especificacion de umbrales de THD
2276                    if (((newdissmsg -> valueA)!=thd_umbral[0]) ||
2277                    ((newdissmsg -> valueB)!=thd_umbral[1])) {
2278                        // Hay valores nuevos
2279                        atomic thd_umbral[0] = (newdissmsg -> valueA);
2280                        atomic thd_umbral[1] = (newdissmsg -> valueB);
2281
2282                        permdata.thd_threshold[0] = (newdissmsg -> valueA);
2283                        permdata.thd_threshold[1] = (newdissmsg -> valueB);
2284
2285                        atomic wr_memory = TRUE;   // Nuevos datos a guardar
2286                    }
2287                    else {
2288                        // No hay valores nuevos
2289                    }
2290                    break;
2291
2292                case 3:     // Especificacion de umbrales de tension y temperatura
2293                    if (((newdissmsg -> valueA)!=temp_umbral) ||
2294                    ((newdissmsg -> valueB)!=volt_umbral)) {
2295                        // Hay valores nuevos
2296                        atomic temp_umbral = (uint16_t)(newdissmsg -> valueA);
2297                        atomic volt_umbral = (uint16_t)(newdissmsg -> valueB);
2298
2299                        permdata.temp_threshold = (newdissmsg -> valueA);
2300                        permdata.volt_threshold = (newdissmsg -> valueB);
2301
2302                        atomic wr_memory = TRUE;   // Nuevos datos a guardar
2303                    }
2304                    else {
2305                        // No hay valores nuevos
2306                    }
2307
2308                    break;
2309
2310                default:
2311                    perror("ID de funcion no valido");
2312                    break;
2313
2314            }  // Fin SWITCH (newdissmsg -> function)
2315        }
2316
2317    }  // Fin de EVENT ActValores.runTask
2318
2319    #endif
2320
2321    }   // Fin de IMPLEMENTATION
2322
2323
2324    //                  Fin de archivo 'MaxwellC.nc'
```

1.6 Maxwell.h

```
1   /*
2   Copyright (c) 2011, Jose Antonio Tarifa Galisteo
3   All rights reserved.
4
5   Redistribution and use in source and binary forms, with or without
6   modification, are permitted provided that the following conditions
7   are met:
8
9       * Redistributions of source code must retain the above copyright
10      notice, this list of conditions and the following disclaimer.
11      * Redistributions in binary form must reproduce the above copyright
12      notice, this list of conditions and the following disclaimer in the
13      documentation and/or other materials provided with the
14      distribution.
15      * Neither the name of the University of Seville nor the names of
16      its contributors may be used to endorse or promote products derived
17      from this software without specific prior written permission.
18
19  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
20  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
21  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
22  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
23  THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
24  INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
25  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
27  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
28  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
29  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
30  OF THE POSSIBILITY OF SUCH DAMAGE.
31  */
32
33  /*-----------------------------------------------------------------------------
34     Archivo:    Maxwell.h
35     Autor:      Jose Antonio Tarifa Galisteo
36     Correo e:   jostargal@alum.us.es
37     Sitio web:  http://alumno.us.es/j/jostargal/
38     Fecha:      17/05/2011
39     Revision:   0517
40     Descrip:    Fichero de cabecera con definicion de constantes simbolicas
41                 generales. Permite una configuracion basica de la aplicacion.
42
43     Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
44     Universidad de Sevilla                    http://www.us.es/
45  -----------------------------------------------------------------------------*/
46
47
48  #ifndef MAXWELL_H_
49     #define MAXWELL_H_
```

```
50
51
52          /*
53          Valor del divisor para los nodos raiz. Los ID de nodo multiplos de esta
54          cantidad se consideran nodos base o raiz (el resto seran nodos de datos).
55          Por defecto: 0,20,40,60,80...
56          */
57          #ifndef ROOT_DIVIDER
58             #define ROOT_DIVIDER 20
59          #endif
60
61
62          // Identificador para enviar un mensaje radio por difusion a todos los nodos.
63          #ifndef BROAD_ID
64             #define BROAD_ID 0xFFFF
65          #endif
66
67
68          /*
69          Identificador de subred para el protocolo de coleccion de datos. Todos los
70          nodos que tengan este identificador enviaran datos hacia el mismo nodo raiz.
71          */
72          #ifndef COLLECTION_ID
73             #define COLLECTION_ID 0x3E
74          #endif
75
76
77          /*
78          Identificador de subred para el protocolo de diseminacion de datos. Todos
79          los nodos que tengan este identificador recibiran mensajes diseminados por
80          el nodo raiz correspondiente.
81          */
82          #ifndef DISSEMINATOR_ID
83             #define DISSEMINATOR_ID 0x2F
84          #endif
85
86          /*
87          Identificador para los mensajes enviados desde el nodo base hasta el PC a
88          traves del puerto serie.
89          */
90          #ifndef SERIALTX_ID
91             #define SERIALTX_ID 0x12
92          #endif
93
94
95          /*
96          Identificador para los mensajes enviados desde el PC hasta el nodo base a
97          traves del puerto serie.
98          */
99          #ifndef SERIALRX_ID
100            #define SERIALRX_ID 0x20
101         #endif
102
103
104         /*
105         Version del firmware instalado actualmente. Es usado para comprobar la
106         version de los datos almacenados en la memoria externa.
```

```
107         Se especifica la version en hexadecimal.
108         */
109         #define FW_VER 0x0205
110
111
112         /*
113         Direccion de memoria a partir de la cual se escribe/lee en los bloques
114         de memoria externa flash. Por defecto, 0.
115         */
116         #define MEM_ADDR 0x00000000
117
118         /*
119         Macros para el control del temporizador 'watchdog' (WDT). Acceso a los
120         registros internos del microcontrolador MSP430F1611.
121         */
122         #define wdt_stop()   WDTCTL = WDTPW + WDTHOLD;
123         #define wdt_start()  WDTCTL = WDTPW + WDTHOLD; \
124                              WDTCTL = WDTPW + WDTCNTCL + WDTSSEL + 0x00;
125         #define wdt_touch()  WDTCTL = WDTPW + WDTCNTCL; \
126                              WDTCTL = WDTPW + WDTHOLD; \
127                              WDTCTL = WDTPW + WDTCNTCL + WDTSSEL + 0x00;
128
129
130         /*
131         Macros para imprimir mensajes de depuracion por pantalla. Para activarlos es
132         necesario especificar en compilacion la bandera 'DEBUGMODE'.
133         Precaucion al utilizar depuracion en nodos de datos (exceso de memoria)
134         */
135         #ifdef DEBUGMODE
136            #include "printf.h"
137            #define welcome() printf("\n\n\tTmote Sky. FW ver.0517 2011 by JATG\n"); \
138                              printfflush();
139            #define pinfo(cad) printf("\n%s",cad);printfflush();
140            #define perror(cad) printf("\nERROR. %s",cad);printfflush();
141            #define psuccess(cad) printf("\nOK. %s",cad);printfflush();
142            #define phalt() printf("\nPROG. DETENIDO");printfflush();
143         #else // No se hace nada
144            #define welcome();
145            #define pinfo(cad);
146            #define perror(cad);
147            #define psuccess(cad);
148            #define phalt();
149         #endif
150
151
152         // Constantes que establecen los valores por defecto para los umbrales
153         #define THD_THRES_VOLT 6400   // Distorsion en tension 8 por ciento
154         #define THD_THRES_INT 250000  // Distorsion en intensidad 50 por ciento
155         #define TEMP_THRES 650        // Temperatura maxima 65.0 grados centigrados
156         #define VOLT_THRES 3500       // Tension maxima alimentacion 3.5 V
157
158
159         /*
160         Macro para efectuar el redondeo de una cantidad. La variable que se reciba
161         como argumento de entrada debera estar multiplicada (previamente) por 10.
162         */
163         #define round(var)   if((var-((var/10)*10)) >= 5) {var/=10; var++;} \
```

```
164                             else {var/=10;}
165
166
167         enum {
168            /*
169            Numero maximo de intentos (fallidos) para arrancar los distintos
170            dispositivos del mote.
171            */
172            MAXATTEMPT = 5,
173            /*
174            Numero maximo de conversiones (de tension e intensidad) que se realizan
175            antes de enviar un paquete de informacion radio.
176            Debido a que se envian cuatro bloques de datos, se recomienda que el
177            valor 'MAXCONV' sea 4,8,16,32...
178            */
179            MAXCONV = 16,
180         };
181
182         enum {
183            RADIO = 1,      // Interfaz radio para envio de mensajes
184            DISSEM = 2,     // Protocolo de diseminacion de datos
185            ROUTING = 3,    // Encaminamiento de protocolo de coleccion de datos
186            ROOT = 4,       // Establecimiento de nodo raiz (prot. coleccion)
187            ADCVI = 5,      // Convertidor A-D para tension e intensidad
188            SERIAL = 6,     // Interfaz serie para envio/recepcion de mensajes
189         };
190
191         /*
192         Estructura con los datos a almacenar en memoria flash externa (no volatil)
193            Tamanio = 23B
194         */
195         typedef struct storage_struct {
196            uint8_t author[9];            // Cadena para autor (max.8)
197            uint16_t version;             // Version guardada en memoria
198            uint32_t thd_threshold[2];    // Valores max. distorsion tens. e intens.
199            uint16_t temp_threshold;      // Valor umbral para temperatura
200            uint16_t volt_threshold;      // Valor maximo de tension de alimentacion
201         } storage_struct;
202
203
204      #endif   // MAXWELL_H_
205
206      //                   Fin de archivo 'Maxwell.h'
```

1.7 Sensores.h

```
1   /*
2   Copyright (c) 2011, Jose Antonio Tarifa Galisteo
3   All rights reserved.
4
5   Redistribution and use in source and binary forms, with or without
6   modification, are permitted provided that the following conditions
7   are met:
8
9       * Redistributions of source code must retain the above copyright
10      notice, this list of conditions and the following disclaimer.
11      * Redistributions in binary form must reproduce the above copyright
12      notice, this list of conditions and the following disclaimer in the
13      documentation and/or other materials provided with the
14      distribution.
15      * Neither the name of the University of Seville nor the names of
16      its contributors may be used to endorse or promote products derived
17      from this software without specific prior written permission.
18
19  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
20  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
21  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
22  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
23  THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
24  INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
25  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
27  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
28  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
29  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
30  OF THE POSSIBILITY OF SUCH DAMAGE.
31  */
32
33  /*-----------------------------------------------------------------------------
34     Archivo:    Sensores.h
35     Autor:      Jose Antonio Tarifa Galisteo
36     Correo e:   jostargal@alum.us.es
37     Sitio web:  http://alumno.us.es/j/jostargal/
38     Fecha:      17/05/2011
39     Revision:   0517
40     Descrip:    Fichero de cabecera con eclaracion de constantes simbolicas para
41                 procesamiento de datos recogidos de sensores.
42
43     Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
44     Universidad de Sevilla                    http://www.us.es/
45  -----------------------------------------------------------------------------*/
46
47
48  #ifndef SENSORES_H_
49     #define SENSORES_H_
```

```
50
51          enum {
52             NUMSAMPSENS = 64,     // Numero de muestras totales
53             NUMSENS = 4,          // Numero de sensores auxiliares
54             SAMPPERSENS = 512,    // Periodo de muestreo para los sensores
55
56             // Constantes para conversion de temperatura interna
57             F_TEMP_A = 100000,
58             F_TEMP_B = 646027200,
59             F_TEMP_C = 232596,
60
61             // Constantes para conversion de tension de alimentacion
62             F_VOLT_A = 250,
63             F_VOLT_B = 819,
64          };
65
66
67       #endif   // SENSORES_H_
68
69
70       //                    Fin de archivo 'Sensores.h'
```

1.8 Dft.h

```
/*
Copyright (c) 2011, Jose Antonio Tarifa Galisteo
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

    * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the
    distribution.
    * Neither the name of the University of Seville nor the names of
    its contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
*/

/*-------------------------------------------------------------------------
   Archivo:    Dft.h
   Autor:      Jose Antonio Tarifa Galisteo
   Correo e:   jostargal@alum.us.es
   Sitio web:  http://alumno.us.es/j/jostargal/
   Fecha:      17/05/2011
   Revision:   0517
   Descrip:    Fichero de cabecera con definiciones con parametros de
               configuracion del convertidor A-D y para el algoritmo de calculo
               de la DFT.

   Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
   Universidad de Sevilla                    http://www.us.es/
-------------------------------------------------------------------------*/


#ifndef DFT_H_
```

```
50          #define DFT_H_
51
52
53          enum {
54             /*
55             Numero de buffers disponibles para recoger las conversiones de tension
56             e intensidad. Como minimo debe haber 2, para poder escribir en uno
57             mientras se escribe en el otro.
58             */
59             MAXBUF = 2,
60             /*
61             Numero total de muestras que se recogen de una vez (tanto de tension
62             como de intensidad) en el buffer.
63             */
64             NUMSAMP = 256, // 128 tension + 128 intensidad (entrelazadas)
65             /*
66             Periodo de muestreo para la tension e intensidad en microsegundos
67             binarios. Debido al uso del principio de tiempo de muestreo reducido, este
68             valor es la mitad del tiempo de muestreo.
69             */
70             SAMPPER = 328, // Equivale a 312.5us (aprox.)
71          };
72
73
74          enum {
75             /*
76             Numero de puntos para hallar su DFT (muestras reales). Debe ser potencia
77             entera de 2. Junto a la frecuencia de muestreo, define las frecuencias
78             en las cuales se encontraran los bins. En este caso, Fs=1600Hz (aprox.) y
79             128 muestras, resulta una resolucion de 12.5Hz.
80             */
81             NN = 128,   // Numero de muestras reales para hallar su DFT (igual a 2N)
82             /*
83             Numero de muestras (complejas) con las que realmente trabaja el algoritmo
84             FFT. Ha de ser igual a NN/2.
85             */
86             N = 64,
87             NMED = 32,  // Debe ser igual a N/2
88             R = 6,      // Logaritmo en base 2 de N
89             /*
90             Numero de armonicos a estudiar para la distorsion (tension e intensidad).
91             Se estudian las frecuencias 50,100,150,...,700,750,800Hz.
92             */
93             ARM = 16,
94             /*
95             Ajuste de la ganancia coherente del enventanado. La constante se calcula
96             multiplicando la ganancia por 2**16. DIVIDIR POR 10
97             */
98             CGAIN_ADJ = 2731, // Ganancia Blackman de 128 puntos = 0.41671875
99          };
100
101
102         // Algoritmo Leacky bucket
103         enum {
104            LB_LWT = 12,   // Limite inferior
105            LB_UPT = 20,   // Limite superior
106            LB_D = 8,      // Factor de olvido
```

```
107         };
108
109
110     #endif   // DFT_H_
111
112
113     //                    Fin de archivo 'Dft.h'
```

1.9 Mensajes.h

```
1   /*
2   Copyright (c) 2011, Jose Antonio Tarifa Galisteo
3   All rights reserved.
4
5   Redistribution and use in source and binary forms, with or without
6   modification, are permitted provided that the following conditions
7   are met:
8
9       * Redistributions of source code must retain the above copyright
10      notice, this list of conditions and the following disclaimer.
11      * Redistributions in binary form must reproduce the above copyright
12      notice, this list of conditions and the following disclaimer in the
13      documentation and/or other materials provided with the
14      distribution.
15      * Neither the name of the University of Seville nor the names of
16      its contributors may be used to endorse or promote products derived
17      from this software without specific prior written permission.
18
19  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
20  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
21  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
22  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
23  THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
24  INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
25  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
27  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
28  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
29  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
30  OF THE POSSIBILITY OF SUCH DAMAGE.
31  */
32
33  /*-----------------------------------------------------------------------------
34     Archivo:    Mensajes.h
35     Autor:      Jose Antonio Tarifa Galisteo
36     Correo e:   jostargal@alum.us.es
37     Sitio web:  http://alumno.us.es/j/jostargal/
38     Fecha:      17/05/2011
39     Revision:   0517
40     Descrip:    Fichero de cabecera con declaracion de las estructuras para los
41                 mensajes que se usan en la aplicacion.
42
43     Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
44     Universidad de Sevilla                    http://www.us.es/
45  -----------------------------------------------------------------------------*/
46
47  #ifndef MENSAJES_H_
48     #define MENSAJES_H_
49
```

```
50          /*
51          Estructura de mensaje utilizada por los nodos de datos para enviar
52          informacion hacia el nodo base o raiz de la red inalambrica.
53             Tamanio = 86B
54          */
55          typedef nx_struct PotenciaMsg {
56             nx_uint16_t nodeid;     // Identificacion del nodo origen (2B)
57             nx_uint16_t vmedia[4];  // Vector tensiones medias (2B*4=8B)
58             nx_uint16_t imedia[4];  // Vector intensidades medias (2B*4=8B)
59             nx_uint32_t vrms[4];    // Vector tensiones eficaces cuad. (4B*4=16B)
60             nx_uint32_t irms[4];    // Vector intensidades eficaces cuad. (4B*4=16B)
61             nx_uint32_t potact[4];  // Vector potencias activas (4B*4=16B)
62             nx_uint16_t fdp[4];     // Vector factores de potencia (2B*4=8B)
63             nx_uint32_t thd[2];     // Distorsion (4B*2=8B)
64             nx_uint16_t tempvolt[2];// Temp. interna y tension alimentacion (2B*2=4B)
65          } PotenciaMsg;
66
67
68          /*
69          Estructura de mensaje utilizada por el nodo raiz para enviar informacion
70          hacia un PC de recogida de datos, o hacia la pasarela Tmote Connect.
71             Tamanio = 86B
72          */
73          typedef nx_struct SerialMsg {
74             nx_uint16_t nodeid;     // Identificacion del nodo origen (2B)
75             nx_uint16_t vmedia[4];  // Vector tensiones medias (2B*4=8B)
76             nx_uint16_t imedia[4];  // Vector intensidades medias (2B*4=8B)
77             nx_uint32_t vrms[4];    // Vector tensiones eficaces cuad. (4B*4=16B)
78             nx_uint32_t irms[4];    // Vector intensidades eficaces cuad. (4B*4=16B)
79             nx_uint32_t potact[4];  // Vector potencias activas (4B*4=16B)
80             nx_uint16_t fdp[4];     // Vector factores de potencia (2B*4=8B)
81             nx_uint32_t thd[2];     // Distorsion (4B*2=8B)
82             nx_uint16_t tempvolt[2];// Temp. interna y tension alimentacion (2B*2=4B)
83          } SerialMsg;
84
85
86          /*
87          Estructura de mensaje utilizada por el nodo raiz para enviar informacion o
88          comandos hacia un nodo concreto de su red, o bien en difusion.
89             Tamanio = 11B
90          */
91          typedef nx_struct DissMsg {
92             nx_uint16_t nodeid;     // Identificacion del nodo destino (2B)
93             nx_uint8_t function;    // Identificador de funcion a realizar (1B)
94             nx_uint32_t valueA;     // Variable A de proposito general (4B)
95             nx_uint32_t valueB;     // Variable B de proposito general (4B)
96          } DissMsg;
97
98
99       #endif   // MENSAJES_H_
100
101
102      //                    Fin de archivo 'Mensajes.h'
```

1.10 Makefile

```
/*
Copyright (c) 2011, Jose Antonio Tarifa Galisteo
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

    * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the
    distribution.
    * Neither the name of the University of Seville nor the names of
    its contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
*/

#-------------------------------------------------------------------------------
#  Archivo:    Makefile
#  Autor:      Jose Antonio Tarifa Galisteo
#  Correo e:   jostargal@alum.us.es
#  Sitio web:  http://alumno.us.es/j/jostargal/
#  Fecha:      17/05/2011
#  Revision:   0517
#  Descrip:    Fichero de tipo makefile para generar la imagen de codigo del
#              programa. Las banderas (CFLAGS) incluyen las librerias
#              necesarias.
#
#  Escuela Tecnica Superior de Ingenieria    http://www.esi.us.es/
#  Universidad de Sevilla                    http://www.us.es/
#-------------------------------------------------------------------------------


# Libreria mensajes en pantalla
```

```
50      CFLAGS += -I$(TOSDIR)/lib/printf
51
52      # Tamanio de la carga util de los mensajes (en LQI: DATA+9B)
53      CFLAGS += -DTOSH_DATA_LENGTH=95
54
55      # Libreria protocolos de red
56      CFLAGS += -I$(TOSDIR)/lib/net
57
58      # Protocolo de red coleccion (version LQI)
59      CFLAGS += -I$(TOSDIR)/lib/net/lqi
60
61      # Protocolo de red diseminacion (version DHV)
62      CFLAGS += -I$(TOSDIR)/lib/net/dhv \
63                -I$(TOSDIR)/lib/net/dhv/interfaces
64
65      # Componente de configuracion de la aplicacion
66      COMPONENT=MaxwellAppC
67
68      # Inclusion de las reglas de compilacion necesarias
69      include $(MAKERULES)
70
71
72      #                     Fin de archivo 'Makefile'
```

1.11 Volumes-stm25p.xml

```
/*
Copyright (c) 2011, Jose Antonio Tarifa Galisteo
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

    * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the
    distribution.
    * Neither the name of the University of Seville nor the names of
    its contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
*/

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!--Archivo:   volumes-stm25p.xml                                            -->
<!--Autor:     Jose Antonio Tarifa Galisteo                                  -->
<!--Correo e:  jostargal@alum.us.es                                          -->
<!--Sitio web: http://alumno.us.es/j/jostargal/                              -->
<!--Fecha:     17/05/2011                                                    -->
<!--Revision:  0517                                                          -->
<!--Descrip:   Definicion de los espacios o bloques de memoria flash         -->
<!--           usados para guardar (y recuperar) informacion de              -->
<!--           configuracion.                                                -->
<!--                                                                         -->
<!--Escuela Tecnica Superior de Ingenieria      http://www.esi.us.es/        -->
<!--Universidad de Sevilla                      http://www.us.es/            -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<volume_table>
   <!-- Bloque de memoria para almacenar valores de configuracion -->
```

```
50          <volume name="CONFIG" size="131072"/>
51      </volume_table>
52
53
54      <!--                  Fin de archivo 'volumes-stm25p.xml'   -->
```

1.12 monitor.m

Las imágenes que aparecen en la página HTML generada por MATLAB están codificadas en formato cadena base64. Debido a su gran extensión en caracteres, se han omitido dichos datos, y en su lugar aparece <image_data>.

```
% ____________________________________________________________________
%
%       Programa MONITOR
% ____________________________________________________________________
%
%   Permite recoger datos procedentes de una red de sensores inalambricos
%   a traves de una pasarela Tmote Connect (protocolos TCP/IP).
%   Estos datos pueden ser mostrados en la ventana de comandos de MATLAB,
%   asi como ser representados de forma grafica. Tambien son enviados
%   en formato HTML para poder visualizarlos desde un explorador web.
%   Se pueden enviar comandos o funciones a un nodo determinado o bien a
%   la red completa (difusion) a traves de la misma pasarela.
%
%   Uso:
%        Para arrancarlo, escribir 'monitor' en la ventana de comandos.
%
%   Utiliza funciones de TCP/UDP/IP Toolbox 2.0.6
%
%   Version 0517. 2011 JATG.



% About TCP/UDP/IP Toolbox 2.0.6
%
% Copyright (c) 1997-2009, Peter Rydesäter
% All rights reserved.
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are
% met:
%
%     * Redistributions of source code must retain the above copyright
%       notice, this list of conditions and the following disclaimer.
%     * Redistributions in binary form must reproduce the above copyright
%       notice, this list of conditions and the following disclaimer in
%       the documentation and/or other materials provided with the distribution
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
```

```
44      % SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
45      % INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
46      % CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
47      % ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
48      % POSSIBILITY OF SUCH DAMAGE.
49
50
51
52      clc;
53      fprintf('..... Programa MONITOR v0517 .....\n\n');
54      fprintf('2011 JATG\njostargal@alum.us.es\nhttp://alumno.us.es/jostargal');
55      pause(1.5);
56      close all;
57      clc;
58
59      % Parametros generales de configuracion
60
61      broadid = 65535;    % Direccion de difusion de la red
62      maxbuf = 10; % Numero maximo de medidas enviadas a la web de monitorizacion
63      dataoff = 1;    % Offset para los datos en forma de vector (vmed,imed...)
64      refreshtime = 4;    % Tiempo de refresco en segundos para la pagina web
65
66
67      handshake='U '; % Cookie para establecimiento de sesion
68
69      % Comprobacion de valores de parametros
70      if ((broadid<0) || (broadid>65535))
71         uiwait(errordlg('El parametro ''broadid'' debe estar entre 0 y 65535
72    incluidos.','Valor ''broadid'''));
73         return;
74      end
75
76      if (refreshtime < 0.5)
77         if (refreshtime <= 0)
78            uiwait(errordlg('El tiempo de actualizacion de pagina ''refreshtime'' debe ser
79    mayor que 0.','Valor ''refreshtime'''));
80            return;
81         else
82            uiwait(warndlg('El tiempo de actualizacion de pagina ''refreshtime'' es
83    demasiado bajo (menor que 0.5s).','Valor ''refreshtime'''),2);
84         end
85      end
86
87      % Se comprueba si existe la variable 'con', que almacena el ID de conexion
88      if (~exist('con','var'))
89         % NO existe: hay que crear una nueva conexion
90         fprintf('No existe ninguna conexion activa.\n');
91         conexdlg=char(inputdlg({'Direccion IP remota (tambien ''localhost'')','Puerto de
92    conexion USB (1,2)'},...
93            'Crear nueva conexion',[1 1;50 50]',{'192.168.1.100','1'}));
94
95         if (isempty(conexdlg))
96            clear all;
97            uiwait(errordlg('No ha aceptado parametros de conexion.','Establecer
98    conexion'));
99            return;
100         end
```

```
101
102         host=conexdlg(1,:);
103         usb=str2double(conexdlg(2,:));
104
105         % Comprobacion de errores en la variable 'host'
106         if((length(findstr('.',host)) ~= 3) && (~strcmp(host,'localhost')))
107            clear all;
108            uiwait(errordlg('Debe escribir una direccion IP en formato decimal, o la
109     palabra ''localhost''.','Nombre de host'));
110            return;
111         end
112
113         if(isempty(usb))
114            usb=1;  % Conexion USB por defecto
115         end
116
117         % Comprobacion de errores en la variable 'usb'
118         if(usb<1 || usb>2)
119            clear all;
120            uiwait(errordlg('Los valores posibles para el puerto USB son 1 y 2.','Puerto
121     USB'));
122            return;
123         else
124            puerto=9000+usb;    % Los puertos disponibles son 9001 y 9002
125         end
126
127         fprintf('Conectando con el servidor remoto en %s:%d ...\n',host,puerto);
128         % Establecimiento de la conexion TCP con la pasarela
129         con = pnet('tcpconnect',host,puerto);
130
131         % Comprobacion de la conexion
132         if(con>=0)
133            % Correcta. 'con' almacena el ID para manejar la conexion
134            fprintf('Conectado. Identificador %d\n',con);
135         else
136            % Ha habido un error. Cerrar la conexion
137            pnet(con,'close');
138            uiwait(errordlg('Error en la conexion. Comprobar direccion IP de la pasarela y
139     puerto USB especificados.','Conexion'));
140            return;
141         end
142
143
144         fprintf('Enviando cookie...\n');
145
146         % Envio de la cookie (handshake) para comenzar la sesion
147         pnet(con,'write',handshake);
148
149         pause(0.1); % Esperar un poco...
150
151         if(strcmp(pnet(con,'read',2,'char'),handshake))
152            % El equipo remoto (GW) ha aceptado la cookie
153            fprintf('Cookie aceptada\n');
154         else
155            % Error al establecer la sesion
156            pnet(con,'close');
157            clear all;
```

```
158             fprintf('Conexion cerrada. Todos los datos borrados\n');
159             uiwait(errordlg('El servidor no ha aceptado la cookie. Comprobar puerto USB,
    ''protocol version'' y ''baudrate''.','Servidor remoto'));
160             return;
161         end
162
163       else
164          % SI existe la variable 'con'. Se comprueba el estado de la conexion
165          switch(pnet(con,'status'))
166             case -1 % Conexion fallida
167                clear all;
168                uiwait(errordlg('Conexion previa fallida. Todos los datos borrados. Vuelva
    a ejecuar el programa.','Conexion'));
169                return;
170             case 0  % Servidor desconectado
171                clear all;
172                uiwait(errordlg('El servidor se ha desconectado. Todos los datos borrados.
    Vuelva a ejecutar el programa.','Servidor remoto'));
173                return;
174             otherwise % Sin error
175                fprintf('Parece que hay una conexion activa con ID=%d\n',con);
176          end
177       end
178
179       % Seleccion aleatoria de un puerto para el arrancar el servidor web
180       % Si existiera un puerto de una conexion anterior, se busca uno nuevo
181       newsrvsock=randi([12001 18999],1);
182       while (exist('srvsock','var') && srvsock==newsrvsock)
183          newsrvsock=randi([12001 18999],1);
184       end
185
186       srvsock=newsrvsock;
187
188       % Crear un socket TCP con el nuevo puerto elegido
189       sockcon=pnet('tcpsocket',srvsock);
190
191       while(sockcon==-1)
192          % Ha habido un fallo al crear el servidor en ese puerto.
193          fprintf('Cambiando puerto servidor web...\n');
194          srvsock=randi([12001 18999],1);
195          sockcon=pnet('tcpsocket',srvsock);
196       end
197
198
199       fprintf('Servidor web de datos en puerto %d\n',srvsock);
200       fprintf('http://localhost:%d\n',srvsock);
201       % Copia al portapapeles de la URL del servidor web
202       clipboard('copy',sprintf('http://localhost:%d',srvsock));
203
204       % Definicion del tiempo maximo de bloqueo en las operaciones de L/E
205       pnet(sockcon,'setreadtimeout',1);
206
207       % Menu de opciones principales
208
209       opt=questdlg('Elija una opcion',...
210          'MONITOR',...
211          'Recibir datos','Enviar datos','Cerrar conexion activa','Recibir datos');
212
213
214
```

```
215       if (isempty(opt))
216          return;
217       end
218
219
220       switch(opt)
221          % ENVIO DE DATOS DESDE EL NODO BASE
222          case 'Enviar datos'
223
224             datos=[23 1 38 40];
225
226             while (1)   % Bucle continuo
227
228                inputdata=char(inputdlg({'ID nodo destino (0, 65535)','Funcion (1,
229    20)','ValorA (0, 4294967295)','ValorB (0, 4294967295)'},...
230                   'Introducir parametros',[1 1 1 1;50 50 50 50]',...
231                   {num2str(datos(1)) num2str(datos(2)) num2str(datos(3))
232    num2str(datos(4))}));
233                datos=str2num(inputdata)';
234
235                if(isempty(datos))
236                   return;
237                end
238
239                fprintf('Espere...   ');
240
241                % Comprobaciones de los datos a enviar
242                if(length(datos)~=4)
243                   fprintf('Error: Se deben de rellenar todos los campos de datos.\n');
244                   datos=[23 1 38 40];
245                elseif((datos(1)<0 || datos(1)>65535))
246                   fprintf('Error: El ID de nodo debe estar entre 0 y 65535.\n');
247                   datos(1)=23;
248                elseif((datos(2)<1 || datos(2)>20))
249                   fprintf('Error: El ID de funcion debe estar entre 1 y 20.\n');
250                   datos(2)=1;
251                elseif((datos(3)<0 || datos(3)>(pow2(32)-1)))
252                   fprintf('Error: valor_a debe estar entre 0 y (2^32)-1.\n');
253                   datos(3)=38;
254                elseif((datos(4)<0 || datos(4)>(pow2(32)-1)))
255                   fprintf('Error: valor-b debe estar entre 0 y (2^32)-1.\n');
256                   datos(4)=40;
257                else
258
259                   % Si todas las comprobaciones son correctas, se procede a
260                   % formatear los datos para enviarlos en bloques de 1B
261                   nodeid=[fix(datos(1)/256) mod(datos(1),256)];
262                   val_a=[mod(fix(datos(3)/pow2(24)),256) ...
263                      mod(fix(datos(3)/pow2(16)),256) ...
264                      mod(fix(datos(3)/256),256) ...
265                      mod(datos(3),256)];
266                   val_b=[mod(fix(datos(4)/pow2(24)),256) ...
267                      mod(fix(datos(4)/pow2(16)),256) ...
268                      mod(fix(datos(4)/256),256) ...
269                      mod(datos(4),256)];
270
271                   % Montaje del paquete a enviar hacia la pasarela
```

```
272                 datosenv=pnet(con,'write',...
273                    cast([19 0 255 255 0 0 11 0 32 ...
274                    fix(datos(1)/256) mod(datos(1),256) ...
275                    datos(2) ...
276                    val_a(1) val_a(2) val_a(3) val_a(4) ...
277                    val_b(1) val_b(2) val_b(3) val_b(4)],'char'));
278
279                 pause(1); % Tiempo de espera entre envios
280
281                 fprintf('Enviados correctamente %d Byte de datos ',datosenv);
282                 % Un ID de nodo 'broadid' representa difusion en la red
283                 if (datos(1) == broadid)
284                    fprintf('en difusion\n');
285                 else
286                    fprintf('a nodo %d\n',datos(1));
287                 end
288
289              end
290
291           end
292
293           % CERRAR CONEXION
294        case 'Cerrar conexion activa'
295           pnet(con,'close');  % Cerrar la conexion activa
296           clear all;
297           fprintf('Conexion cerrada. Todos los datos borrados\n');
298
299           % RECEPCION DE DATOS EN EL NODO BASE (por defecto)
300        case 'Recibir datos'
301
302           % Opcion para representar graficas de los nodos remotos
303           graficas=questdlg('Representar graficas temporales',...
304              'Representar graficas',...
305              'Si','No','Si');
306           if (isempty(graficas))
307              graficas='Si';
308           end
309
310           secnum=0;   % Numero de secuencia recibida
311           indbuf=1;   % Indice para moverse por el buffer de datos
312
313           pkterr=0;
314
315           % Este vector almacena para cada posible nodo remoto el ultimo
316           % numero de paquete recibido, para representar correctamente su
317           % grafica
318           idnodos=uint16(zeros(1,65535));
319
320           % Datos que seran enviados al servidor web de datos
321           datosweb=double(zeros(1,18,maxbuf));
322
323
324           fprintf('\nPara detener el programa, CTRL+C\n\n');
325           fprintf(' Tiempo   CTR  GRP  HID  MotID  Vmed  Imed  Vrms  Irms    P.apar
326    P.act      P.reac    fdp     THD v     THD i   Temp  Volt\n');
327           fprintf('hh:mm:ss   #    #    #     #     mV    mA    mV    mA      uVA
328    uW        uVAR      %%        %%         %%      ºC    mV\n');
```

```
329          fprintf('--:--:--  ---  ---  ---  -----  ----  ----  ----  ----  ---------  --
330     -------  ---------  -----  --------  --------  ----  ----\n');
331
332
333          while (1)   % Bucle continuo
334
335             % Extraccion de la longitud del paquete recibido
336             longitud=pnet(con,'read',1,'uint8','noblock');
337
338             if(~isempty(longitud))
339                % Parece que hay datos
340                pause(0.2);
341                % Recogida de los datos del paquete y posterior
342                % procesamiento usando 'procdata'
343                datos=procdata(uint32(pnet(con,'read',longitud,'uint8','noblock')));
344
345                if(datos ~= -1)
346                   % Datos correctos
347
348                   pkterr=0;
349
350                   tiempo=double(fix(clock));  % Extraccion del tiempo actual
351
352                   if (datos(1) ~= 0)
353                      % Comprobacion del campo de control del mensaje
354                      fprintf('    Valor CTR erroneo:\n');
355                   end
356
357                   % Procesamiento de datos
358                   vmed=2500*double(datos(8+dataoff))/4095;
359                   imed=2500*double(datos(12+dataoff))/4095;
360                   vrms=2500*sqrt(double(datos(16+dataoff)))/4095;
361                   irms=2500*sqrt(double(datos(20+dataoff)))/4095;
362
363     potS=((2500/4095)^2)*sqrt(double(datos(16+dataoff))*double(datos(20+dataoff)));
364                   potP=((2500/4095)^2)*double(datos(24+dataoff));
365
366     potQ=((2500/4095)^2)*sqrt((double(datos(16+dataoff))*double(datos(20+dataoff)))-
367     (double(datos(24+dataoff))^2));
368                   fdp=sqrt(double(datos(28+dataoff)));
369                   thdv=sqrt(double(datos(32)))/10;
370                   thdi=sqrt(double(datos(33)))/10;
371                   temp=double(datos(34))/10;
372                   volt=double(datos(35));
373
374                   vectordatos=[tiempo(4) tiempo(5) tiempo(6) double(datos(1))
375     double(datos(7)) vmed imed vrms irms potS potP potQ fdp thdv thdi temp volt];
376
377                   % Impresion de datos por pantalla
378                   fprintf('%02d:%02d:%02d  % 3d  % 3d  % 3d  % 5d  %4.0f  %4.0f  %4.0f
379     %4.0f  %9.1f  %9.1f  %9.1f  %5.1f  %8.2f  %8.2f  %4.1f  %4.0f\n',...
380                      tiempo(4),tiempo(5),tiempo(6),...%% Horas:minutos:segundos
381                      datos(1),datos(5),datos(6),...
382                      datos(7),...
383                      vmed,imed,vrms,irms,potS,potP,potQ,fdp,thdv,thdi,temp,volt);
384
385                   % Gestion del buffer de datos para la pagina web
```

```
386                     if (indbuf<=maxbuf)
387                        datosweb(:,:,indbuf)=[vectordatos secnum];
388                        indbuf=indbuf+1;
389                        secnum=secnum+1;
390                     else
391                        for ind=1:maxbuf-1
392                           datosweb(:,:,ind)=datosweb(:,:,ind+1);
393                        end
394                        datosweb(:,:,maxbuf)=[vectordatos secnum];
395                        secnum=secnum+1;
396                     end
397
398
399                     switch (graficas)
400                        case 'Si'
401
402                           % Activar una ventana de figura  para el nodo
403                           figure(uint16(datos(7)))
404                           set(uint16(datos(7)),'NumberTitle','off',...
405                              'Name',sprintf('Nodo %d',datos(7)),...
406                              'Toolbar','none');
407
408                           idnodos(datos(7))=idnodos(datos(7))+1;
409
410                           % Grafica para tension e intensidad eficaces
411                           subplot(2,2,1)
412                           hold on;
413                           stem(idnodos(datos(7)),vrms,...
414                              'Marker','x','MarkerSize',8,'LineStyle',':','Color',[0.8471
415    0.1608 0]);
416                           stem(idnodos(datos(7)),irms,...
417                              'Marker','o','MarkerSize',6,'LineStyle','-
418    .','Color',[0.04314 0.5176 0.7804]);
419                           xlabel('Paquetes');
420                           ylabel('Tens. eficaz mV / Int. eficaz mA');
421                           legend('V','I');
422
423                           % Grafica para potencias activa y reactiva
424                           subplot(2,2,2)
425                           hold on;
426                           stem(idnodos(datos(7)),potP,...
427                              'Marker','x','MarkerSize',8,'LineStyle',':','Color',[0.8471
428    0.1608 0]);
429                           stem(idnodos(datos(7)),potQ,...
430                              'Marker','o','MarkerSize',6,'LineStyle','-
431    .','Color',[0.04314 0.5176 0.7804]);
432                           xlabel('Paquetes');
433                           ylabel('P.activa uW / P.reactiva uVAR');
434                           legend('P','Q');
435
436                           % Grafica para factor de potencia y angulo phi
437                           subplot(2,2,3)
438                           hold on;
439                           stem(idnodos(datos(7)),fdp,...
440                              'Marker','x','MarkerSize',8,'LineStyle',':','Color',[0.8471
441    0.1608 0]);
442                           stem(idnodos(datos(7)),(180/pi)*acos(fdp/100),...
```

```
443                             'Marker','o','MarkerSize',6,'LineStyle','-
444     .','Color',[0.04314 0.5176 0.7804]);
445                             xlabel('Paquetes');
446                             ylabel('f.d.p. % / \phiº');
447                             legend('fdp','\phi');
448 
449                             % Grafica para THD de tension e intensidad
450                             subplot(2,2,4)
451                             hold on;
452                             stem(idnodos(datos(7)),thdv,...
453                                'Marker','x','MarkerSize',8,'LineStyle',':','Color',[0.8471
454     0.1608 0]);
455                             stem(idnodos(datos(7)),thdi,...
456                                'Marker','o','MarkerSize',6,'LineStyle','-
457     .','Color',[0.04314 0.5176 0.7804]);
458                             xlabel('Paquetes');
459                             ylabel('THD tension % / THD intensidad %');
460                             legend('V','I');
461 
462                          otherwise
463                       end
464 
465                    else
466                       % Error en la longitud del paquete
467                       pkterr = pkterr+1;
468                       if (pkterr < 5)
469                          fprintf('    Error en la longitud del paquete\n');
470                       else
471                          if (pkterr < 15)
472                             fprintf('    Error en la longitud del paquete. Posible problema
473     de sincronismo\n');
474                          else
475                             fprintf('    Error en la longitud del paquete. Posible problema
476     de sincronismo. Reiniciar?\n');
477                          end
478                       end
479                    end
480 
481                else
482                   pause(0.2); % Espera para recoger otro paquete
483                end
484 
485 
486                conb=pnet(sockcon,'tcplisten');
487                if( conb~=-1 ),
488                   try
489                      [ip,port]=pnet(conb,'gethost');
490                      fprintf('    Conexion desde %d.%d.%d.%d en puerto %d',ip,port);
491                      % Evitar el bloqueo del servidor debido a navegadores o
492                      % conexiones lentas
493                      pnet(conb,'setreadtimeout',2);
494                      pnet(conb,'setwritetimeout',1);
495                      pnet(conb,'readline');
496                      pnet(conb,'printf','HTTP/1.1 200 OK\n');  % Version HTTP
497                      pnet(conb,'printf','Content-Type: text/html\n\n');
498                      pnet(conb,'printf','<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
499     Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">\n');
```

```
500                     pnet(conb,'printf','<html lang="es-es">\n');
501                     if ((refreshtime-fix(refreshtime)) == 0)
502                        pnet(conb,'printf','<head><meta content="text/html; charset=ISO-
503     8859-1" http-equiv="content-type">\n<meta http-equiv="refresh" content="%d;
504     url=./">\n',refreshtime);
505                     else
506                        pnet(conb,'printf','<head><meta content="text/html; charset=ISO-
507     8859-1" http-equiv="content-type">\n<meta http-equiv="refresh" content="%.1f;
508     url=./">\n',refreshtime);
509                     end
510                     % Titulo de la pagina web
511                     if(~exist('tiempo','var'))
512                        pnet(conb,'printf','<title>Recogida de datos -motes- SIN
513     DATOS</title></head>\n');
514                     else
515                        pnet(conb,'printf','<title>Recogida de datos -motes-
516     %2d:%02d:%02d</title></head>\n',tiempo(4),tiempo(5),tiempo(6));
517                     end
518                     pnet(conb,'printf','<body>\n<div style="text-align: center;">\n');
519                     pnet(conb,'printf','<a target="_blank" href="http://www.esi.us.es">
520     <img\n style="border: 0px solid;width: 65px; height: 80px;"\n title="Escuela Técnica
521     Superior de Ingeniería"\n alt="Marca Escuela Técnica Superior de Ingeniería"\n
522     src="<image_data>"></a>\n');
523                     pnet(conb,'printf','       \n');
524                     pnet(conb,'printf','<a target="_blank" href="http://www.us.es">
525     <img\n style="border: 0px solid;width: 80px; height: 80px;"\n title="Universidad de
526     Sevilla"\n alt="Marca Universidad de Sevilla"\n
527     src="<image_data>"></a>\n');
528                     pnet(conb,'printf','</div>\n');
529                     pnet(conb,'printf','<h1 style="text-align: center;">Recogida de datos
530     Tmote Sky + Tmote Connect</h1><hr>\n');
531                     pnet(conb,'printf','<h3 style="text-align:
532     center;">2011   <a target="_blank"
533     href="http://alumno.us.es/j/jostargal/"><i>José Antonio Tarifa
534     Galisteo</i></a></h3>\n');
535                     pnet(conb,'printf','<pre>Direccion IP equipo: %d.%d.%d.%d   Puerto:
536     %d</pre>\n',ip,port);
537                     pnet(conb,'printf','<h2>Últimos %d datos recogidos:</h2>\n',maxbuf);
538                     % Envio de datos procesados
539                     pnet(conb,'printf','<pre> Tiempo   CTR  MotID  Vmed  Imed  Vrms  Irms
540     P.apar     P.act      P.reac    fdp     THD v     THD i   Temp  Volt   Seq</pre>\n');
541                     pnet(conb,'printf','<pre>hh:mm:ss   #     #     mV    mA    mV    mA
542     uVA         uW        uVAR      %%        %%        %%      ºC    mV     #</pre>\n');
543                     pnet(conb,'printf','<pre>--:--:--  ---  -----  ----  ----  ----  ----
544     ---------  ---------  ---------  -----  --------  --------  ----  ----  -----
545     </pre>\n');
546                     for ind=1:maxbuf
547                        pnet(conb,'printf','<pre>%02d:%02d:%02d  % 3d  % 5d  %4.0f  %4.0f
548     %4.0f  %4.0f  %9.1f  %9.1f  %9.1f  %5.1f  %8.2f  %8.2f  %4.1f  %4.0f
549     %05d</pre>\n',...
550                           (datosweb(1,:,ind)));
551                     end
552                     if ((refreshtime-fix(refreshtime)) == 0)
553                        pnet(conb,'printf','<br>La página se actualiza cada %ds. Si no
554     ocurre esto, pulse <a href="./">aquí</a> (También F5).<br><br>\n',refreshtime);
555                     else
```

```
556                     pnet(conb,'printf','<br>La página se actualiza cada %.1fs. Si no
557     ocurre esto, pulse <a href="./">aquí</a> (También F5).<br><br>\n',refreshtime);
558                     end
559                     pnet(conb,'printf','Utiliza funciones de <i>TCP/UDP/IP Toolbox 2.0.6
560     (Peter Rydesäter, Mitthögskolan, Östersund, SWEDEN)</i>\n');
561                     pnet(conb,'printf','<br><br><a target="_blank"
562     href="http://validator.w3.org/check?uri=referer"> <img\n style="border: 0px
563     solid;width: 88px; height: 31px;"\n title="Valid HTML 4.01 Transitional"\n alt="Valid
564     HTML 4.01 Transitional"\n src="<image_data>"></a>\n');
565                     pnet(conb,'printf','\n</body>\n</html>');
566
567                 catch ME
568
569                 end
570                 fprintf('    Cerrando conexion...\n');
571                 pnet(conb,'close');
572                 drawnow;
573             end
574         end
575        otherwise
576     end
577
578
579     %                     Fin de archivo 'monitor.m'
```

1.13 procdata.m

```
function [pdata]=procdata(data)
%
%       Funcion PROCDATA
%
%   Funcion auxiliar para el procesamiento de los datos que se reciben
%   desde la pasarela TCP/IP.
%   Se deben indicar el tamanio de cada campo del mensaje recibido, para
%   que la conversion de valores se haga de forma correcta.
%
%   Uso:
%        PDATA = PROCDATA(DATA)
%
%   Argumentos de entrada:    DATA vector con los datos en crudo (raw)
%                              recibidos desde el puerto.
%   Argumentos de salida:     PDATA. Si el proceso se ha realizado sin
%                              problemas, devuelve los datos procesados.
%                              En caso de error, devuelve -1.
%
%   Version 0517. 2011 JATG.

% CABECERA MENSAJES. Indicar tamano en B de cada elemento
% Chk-Dst-Src-Len-Grp-AMt
headersize=[1 2 2 1 1 1];

% CARGA UTIL. Indicar tamano en B de cada elemento
% NodeID-Vmed-Vmed-Imed-Imed-Vrms-Vrms-Irms-Irms-Pact-Pact-THDv-THDi-
% -Temp-Volt
loadsize=[2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 4 4 2 2];

psize=[headersize loadsize];    % Tamanos de todos componentes msg.

bytecount=0;

if(sum(psize) ~= length(data))  % Tamano de los datos no es el esperado
   pdata=-1;
   return;
end

pdata=uint32(zeros(1,length(psize)));   % Vector de datos procesados

for ind=1:length(psize)
   pdata(ind)=0;
   for indb=psize(ind):-1:1
      bytecount=bytecount+1;
      pdata(ind)=pdata(ind)+(pow2(8)^(indb-1))*data(bytecount);
   end
end
end
%                     Fin de archivo 'procdata.m'
```