VALIDACIÓN DE LA CAPA FÍSICA Y LA CAPA MAC DEL MODELO DE RED INALÁMBRICA IEEE 802.15.4 APLICADO A UN PROCESO INDUSTRIAL
ANEXOS

ADRIÁN ENRIQUE RODRIGUEZ SANDOVAL
JAIME ANDRÉS LÓPEZ MUÑOZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELÉCTRONICA Y TELECOMUNICACIONES
INGENIERIA EN AUTOMATICA INDUSTRIAL
POPAYÁN
2007

VALIDACIÓN DE LA CAPA FÍSICA Y LA CAPA MAC DEL MODELO DE RED INALÁMBRICA IEEE 802.15.4 APLICADO A UN PROCESO INDUSTRIAL
ANEXOS

ADRIÁN ENRIQUE RODRIGUEZ SANDOVAL
JAIME ANDRÉS LÓPEZ MUÑOZ

Trabajo de grado presentado para optar al título de
Ingeniero en Automática Industrial

Director
VLADIMIR TRUJILLO ARIAS
Ingeniero Electrónico

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELÉCTRONICA Y TELECOMUNICACIONES
INGENIERIA EN AUTOMATICA INDUSTRIAL
POPAYÁN
2007

TABLA DE CONTENIDO

LISTA DE FIGURAS

1. ANEXO A. PROGRAMACIÓN DE LOS DISPOSITIVOS

Cada empresa que suministra dispositivos con tecnología Zigbee, entrega un conjunto de librerías para la implementación de la gestión de datos en cada nodo. En este caso la empresa Microchip establece un conjunto de librerías llamado el Stack Zigbee [1], con las cuales es posible realizar las diferentes configuraciones que requiere cada dispositivo, lo que facilita su utilización en cualquier tipo de aplicación.

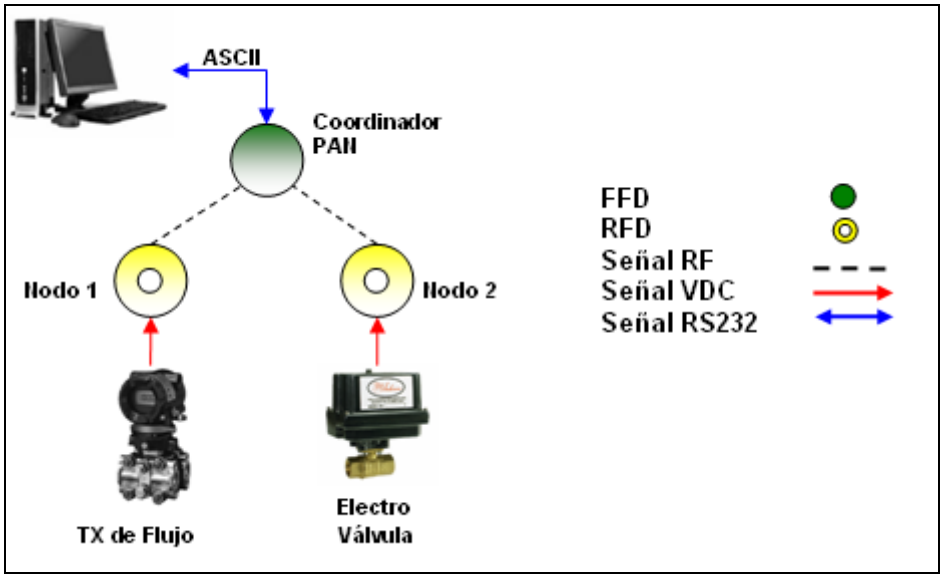

Figura 1 Nodos

Se deben tener en cuenta algunos pasos antes de comenzar a diseñar la aplicación, ellos son:

- Cada dispositivo que desee comunicarse haciendo uso del protocolo Zigbee debe tener una dirección MAC única de 64 bits. Esta dirección esta constituida por un Identificador único de la Organización (OUI, *Organizationally Unique Identifier*) de 24 bits, y los 40 bits restantes corresponden a los asignados por los fabricantes [2].
- Seleccionar el dispositivo con las características físicas de memoria, capacidad de memoria, funcionalidades etc.; que mejor se ajuste a la aplicación. Por ejemplo se presentan los requerimientos de memoria de los dispositivos utilizados en control de iluminación, que cumplen con los requisitos especificados en el Perfil del Stack:
  - Coordinador 36810 ROM bytes
  - Router: 30064 ROM bytes
  - End Device (RFD) 18385 ROM bytes

  En memoria RAM, dependiendo de la aplicación, se requieren 5 bancos disponibles, para un RFD, obviamente se requieren mayores características para los dispositivos FFD, coordinador y router.

- Iniciar el diseño de la aplicación cumpliendo con las normas establecidas de interoperabilidad, funcionamiento y las certificaciones respectivas.

A continuación se presenta la forma en que se debe programar el módulo de control, PIC 18LF4620:

1.1. Programa del Dispositivo FFD

Microchip ofrece un software de configuración llamado ZENA [3], con el cual se configuran los parámetros que caracterizan a cada dispositivo. El programa genera los archivos; myZigbee.c, zigbee.def y zLink.lkr

- ***zigbee.def*** . Provee las definiciones básicas de la configuración del Stack.
- ***myZigBee.c***. Provee toda la inicialización ROM para el Stack, incluyendo los descriptores del dispositivo Zigbee.
- ***zLink.lkr***. Es el scrip utilizado para la configuración del PIC, el linker.

zigbee.def

```
// Created by ZENA(TM)
1. #ifndef _ZIGBEE_DEF
2. #define _ZIGBEE_DEF

Configuracion de la dirección MAC
3. #define MAC_LONG_ADDR_BYTE7 0x00
4. #define MAC_LONG_ADDR_BYTE6 0x04
5. #define MAC_LONG_ADDR_BYTE5 0xA3
6. #define MAC_LONG_ADDR_BYTE4 0x00
7. #define MAC_LONG_ADDR_BYTE3 0x00
8. #define MAC_LONG_ADDR_BYTE2 0x00
9. #define MAC_LONG_ADDR_BYTE1 0x00
10.   #define MAC_LONG_ADDR_BYTE0 0x60

Tipo de dispositivo soportado
11.   #define I_AM_COORDINATOR
12.   #define I_AM_FFD
13.   #define I_SUPPORT_ROUTING

Configuración de desempeño de la comunicación
14.   #define MY_CAPABILITY_INFO 0x8f
15.   #define SUPPORT_END_DEVICE_BINDING
16.   #define CONFIG_ENDDEV_BIND_TIMEOUT (ONE_SECOND * 5)
17.   #define MAX_APL_FRAMES 4
18.   #define MAX_APS_ACK_FRAMES 4
19.   #define MAX_APS_ADDRESSES 0
20.   #define NUM_BUFFERED_INDIRECT_MESSAGES 5
21.   #define I_SUPPORT_BINDINGS
22.   #define MAX_BINDINGS 100
23.   #define MAX_NEIGHBORS 24
24.   #define NUM_BUFFERED_BROADCAST_MESSAGES 3
25.   #define ROUTE_DISCOVERY_TABLE_SIZE 4
26.   #define ROUTING_TABLE_SIZE 16
27.   #define RESERVED_ROUTING_TABLE_ENTRIES 8
```

```
28.   #define NUM_BUFFERED_ROUTING_MESSAGES 4
29.   #define MAX_ENERGY_THRESHOLD 112
30.   #define                    MAC_PIB_macTransactionPersistenceTime
   SYMBOLS_TO_TICKS(480000)
31.   #define RX_BUFFER_SIZE 256
32.   #define RX_BUFFER_LOCATION 0xa00
33.   #define MAC_PIB_macBeaconOrder 15
34.   #define MAC_PIB_macSuperframeOrder 15
35.   #define IEEE_COMPLY
36.   #define MAC_PIB_macBattLifeExt FALSE
37.   #define RF_CHIP CC2420
38.   #define MAC_PIB_macAckWaitDuration (ONE_SECOND/2)
39.   #define FREQUENCY_BAND FB_2400GHz
```

Selección de los pines para la comunicación SPI con el transceiver CC2420

```
40.   #define PHY_CSn LATC0
41.   #define PHY_CSn_TRIS TRISC0
42.   #define PHY_FIFO RB0
43.   #define PHY_FIFO_TRIS TRISB0
44.   #define PHY_FIFOP RB3
45.   #define PHY_FIFOP_TRIS TRISB3
46.   #define PHY_RESETn LATC2
47.   #define PHY_RESETn_TRIS TRISC2
48.   #define PHY_SFD RB2
49.   #define PHY_SFD_TRIS TRISB2
50.   #define PHY_VREG_EN LATC1
51.   #define PHY_VREG_EN_TRIS TRISC1
```

Selección del nivel de consumo de corriente, Obligatorio

```
52.   #define PA_LEVEL 0xEF  // -7 dBm (12.5 mA)
```

Canales permitidos (11 para 2.4GHz)

```
53.   #define ALLOWED_CHANNELS 0x00001000
54.   #define CLOCK_FREQ 4000000
```

Configuracion USART

```
55.   #define BAUD_RATE 19200

56.   #define HEAP_LOCATION 0x0100
57.   #define MAX_HEAP_SIZE 2048
```

Numero de EndPoints

```
58.   #define NUM_USER_ENDPOINTS 2
```

EndPoint

```
59.   #define EP_LEVEL 9
60.   #define MY_MAX_INPUT_CLUSTERS 0
61.   #define MY_MAX_OUTPUT_CLUSTERS 2
```

Perfil utilizado

```
62.   #include "C:\MpZbee\ZigbeeStack\zICLevel.h"
```

El perfil del Stack esta incluido en el archivo ***zICLevel.h*** donde se especifican los diferentes dispositivos, sensores, controladores, etc.; con sus respectivas identificaciones dentro de la red.

myZigbee.c

// Created by ZENA(TM)

Archivos que deben incluirse para llevar a cabo la configuración de la información de los clusters

```
1. #include "zigbee.def"
2. #include "zNWK.h"          ← Funciones de la capa NWK de Zigbee
3. #include "zZDO.h"          ← Funciones del Zigbee Device Object
```

Descriptor del nodo, información de las cualidades del nodo.

```
4. ROM NODE_DESCRIPTOR Config_Node_Descriptor =
5. {
6. 0x02,               // ZigBee End Device
7. 0x00,               // (reserved)
8. 0x00,               // (APS Flags, not currently used)
9. 0x08,               // Frequency Band 2400
10.   MY_CAPABILITY_INFO, // Capability Information
11.   {0x55, 0x05},          // Manufacturer Code
12.   0x7F,                  // Max Buffer Size
13.   {0x7F, 0x00}            // Max Transfer Size
14.   };

15.   ROM NODE_POWER_DESCRIPTOR Config_Power_Descriptor =
16.   {
17.   0x01, //RxPeriodic
18.   0x10, //RechBatt
19.   0x01, //RechBatt
20.   0x0c  //Fill in current power level
21.   };
```

Este código permite identificar al dispositivo dentro del la Red, se describe su información, tipo de configuración, perfil, etc.

```
22.   ROM NODE_SIMPLE_DESCRIPTOR Config_Simple_Descriptors[3] =
23.   {
24.   //-------------------------------------
25.   // ZigBee Device Object Endpoint
26.   // DO NOT MODIFY THIS DESCRIPTOR!!!
27.   //-------------------------------------
28.       {
29.           EP_ZDO,
30.           {MY_PROFILE_ID_LSB,MY_PROFILE_ID_MSB},
31.           {VALVE_SENSOR_DEV_ID_LSB,VALVE_SENSOR_DEV_ID_MSB},
32.           VALVE_SENSOR_DEV_VER,
33.           NO_OTHER_DESCRIPTOR_AVAILABLE,
34.           ZDO_INPUT_CLUSTERS,
35.           {     NWK_ADDR_req,     IEEE_ADDR_req,     NODE_DESC_req,
   POWER_DESC_req,
36.             SIMPLE_DESC_req, ACTIVE_EP_req, MATCH_DESC_req
37.                   , BIND_req, UNBIND_req
38.           },
39.           ZDO_OUTPUT_CLUSTERS,
40.           {     NWK_ADDR_rsp,     IEEE_ADDR_rsp,     NODE_DESC_rsp,
   POWER_DESC_rsp,
41.             SIMPLE_DESC_rsp, ACTIVE_EP_rsp, MATCH_DESC_rsp
42.                   , BIND_rsp, UNBIND_rsp
43.           }
```

```
44.       }
```

El coordinador maneja dos ENDPOINTs uno para el sensor de nivel y el otro para la válvula de control.

```
45.   //--------------------------------------
46.       {
47.           EP_LEVEL,
48.           {MY_PROFILE_ID_LSB,MY_PROFILE_ID_MSB},
49.           {VALVE_SENSOR_DEV_ID_LSB,VALVE_SENSOR_DEV_ID_MSB},
50.           VALVE_SENSOR_DEV_VER,
51.           NO_OTHER_DESCRIPTOR_AVAILABLE,
52.           0,
53.           { NO_CLUSTER },
54.           1,
55.           {
56.               Level_CLUSTER
57.           }
58.       }
59.   ,
60.   //--------------------------------------
61.       {
62.           EP_VALVE,
63.           {MY_PROFILE_ID_LSB,MY_PROFILE_ID_MSB},
64.           {VALVE_SENSOR_DEV_ID_LSB,VALVE_SENSOR_DEV_ID_MSB},
65.           VALVE_SENSOR_DEV_VER,
66.           NO_OTHER_DESCRIPTOR_AVAILABLE,
67.           0,
68.           { NO_CLUSTER },
69.           1,
70.           {
71.               Valve_CLUSTER
72.           }
73.       }
74.   };
```

Este código contiene la información general del Nodo

```
75.   ROM _Config_NWK_Mode_and_Params Config_NWK_Mode_and_Params =
76.   {
77.       nwkcProtocolVersion,                  //Protocol Version
78.       MY_STACK_PROFILE_ID,                  //Stack Profile ID
79.       MAC_PIB_macBeaconOrder,               //Beacon Order
80.       MAC_PIB_macSuperframeOrder,     //Superframe Order
81.       MAC_PIB_macBattLifeExt,               //Battery          Life
   Extension
82.       PROFILE_nwkSecurityLevel,             //Security Level
83.       ALLOWED_CHANNELS                      //Channels to scan
84.   };
```

Una vez generados los anteriores programas, se debe realizar la integración de estos con el Stack Zigbee de Microchip, para dar solución a las necesidades de la aplicación. Los archivos del Stack contienen la lógica para soportar los tipos de aplicaciones del protocolo Zigbee, incluyendo la configuración propuesta por el estándar 802.15.4, pero sólo se habilitará la lógica necesaria dependiendo de la aplicación, ver tablas 1 y 2.

Tabla 1 Estructura del Directorio de Archivos

Nombre del Directorio	**Contenido**
Common	Archivos comunes para el Stack de Microchip
Documentation	Documentación del Stack
ZigbeeStack	Archivos para el manejo del protocolo

Tabla 2 Archivos del Stack Zigbee [1]

Nombre	**Contenido**
SymbolTime.c, .h	Desempeña funciones de sincronización para el Stack de Microchip para el protocolo Zigbee™.
zAPL.h	Archivo de encabezado para la interfase del nivel de aplicación para el Stack. Este es el único archive que el código de aplicación necesita incluir.
zAPS.c, .h	Capa APS del protocolo.
zigbee.h	Constantes genericas del protocolo.
ZigBeeTasks.c, .h	Directivas del flujo del programa a través del las capas del Stack
zMAC.h	Encabezado genérico de la capa MAC de la norma IEEE 802.15.4
zMAC_CC2420.c, .h	Capa MAC IEEE 802.15.4 para el transceiver de Chipcon CC2420.
zNVM.c, .h	Proporciona funciones de almacenamiento en memoria no volátil.
zNWK.c, .h	Capa NWK.
zPHY.h	Encabezado de la capa PHY IEEE 802.15.4.
zPHY_CC2420.c, .h	Capa PHY IEEE 802.15.4 para el transceiver CC2420.

Haciendo uso del software de programación de microcontroladores MPLAB IDE v7.50, junto con el compilador C18, para programar con código en ANSI C, y el Stack Zigbee, que se puede descargar desde su pagina Web [4], se desarrolló el código para cada nodo especifico. Los archivos listados en la tabla 2 fueron incluidos en el proyecto para cada nodo. Ver las figura 2.

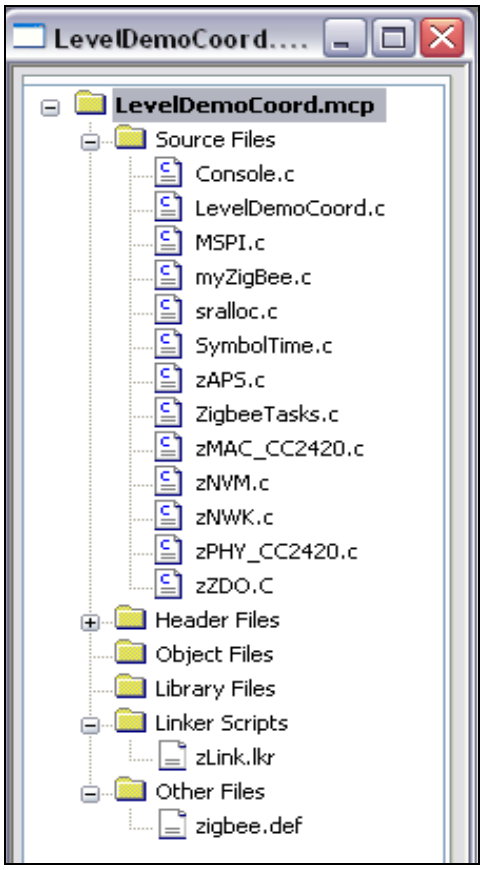

Figura 2 Archivos .C incluidos para el nodo Coordinador

El archivo principal desde donde se gestionan todas las librerías es el LevelDemoCoor.C, el cual sigue los siguientes lineamientos:

1.1.1. Estructura Básica de la Aplicación:

```
while (1)
{
        CLRWDT();
        ZigBeeTasks( &currentPrimitive );

        switch (currentPrimitive)
        {
                case NLME_NETWORK_FORMATION_confirm: …
                case NLME_PERMIT_JOINING_confirm: …
                case NLME_JOIN_indication: …
                case NLME_LEAVE_indication: …
                case NLME_RESET_confirm: …
                case APSDE_DATA_confirm: …

                case APSDE_DATA_indication:

                        switch (params.APSDE_DATA_indication.DstEndpoint)
                                {
                                        case EP_ZDO:

                                        case EP_LEVEL:

                                        case EP_VALVE
                                }

                case NO_PRIMITIVE: …
                default:
                currentPrimitive = NO_PRIMITIVE;
        break;
        }
}
```

El código del programa principal del coordinador PAN que solicita datos del dispositivo identificado con el EndPoint EP_LEVEL es el siguiente:

1.1.2. Solicitud de Datos del Coordinador al RFD de Nivel

```
case APSDE_DATA_indication:
{
…….
case EP_LEVEL:


                    ****************** VERIFICACION ******************
 if ((frameHeader & APL_FRAME_TYPE_MASK) == APL_FRAME_TYPE_KVP)
     {

                ****************** EXTRACCION DE INFO ******************

       frameHeader &= APL_FRAME_COUNT_MASK;
       for(transaction=0; transaction<frameHeader; transaction++)
          {
            sequenceNumber           = APLGet();
            command                  = APLGet();
            attributeId.byte.LSB     = APLGet();
            attributeId.byte.MSB     = APLGet();
            errorCode                = APLGet();

               ****************** VERIFICACION DE ERROR ******************

            if (errorCode != KVP_SUCCESS)
               {
                ConsolePutROMString( (ROM char *) "Mensaje de error recibido\r\n" );
                }
                else
                   {

              ****************** VERIFICACION ID CLUSTER ******************
```

```
            if ((params.APSDE_DATA_indication.ClusterId ==  Level_CLUSTER) &&
(attributeId.Val == Level_CmStr))
                {
                 if (command == APL_FRAME_COMMAND_GET_RES)
                    {
                     ConsolePutROMString( (ROM char *)"Recibido " );
                     data = APLGet();    // Longitud del dato String
                     while (data--)
                      {
                       ConsolePut( APLGet() );      //Capturar Datos
                      }
                     ConsolePutROMString( (ROM char *)"\r\n" );
                    }
                  }
                }// Lectura completa de
              } //los tipos de trama de la transacción
       break;
      default:
     break;
          }
     APLDiscardRx();
    }
  Break;
```

El Stack notifica a la capa de aplicación la recepción de un mensaje a través de la primitiva APSDE_DATA_indication. Esta primitiva retorna con información acerca del mensaje, mientras que el mensaje reside en el buffer. Se usa la función APLGet(), para extraer cada byte del mensaje almacenado en el buffer.
El parámetro DstEndPoint indica el endpoint destino. Si este es un endpoint válido, el mensaje puede ser procesado.
El código verifica que el encabezado de trama APL_FRAME corresponda el valor asignado a APL_FRAME_TYPE_KVP, si esto es cierto, se lleva a cabo la extracción de la información contenida en la trama (*sequence Number, command, attributeId.byte.LSB, attributeId.byte.MSB* y el *errorCode*). Si se verifica que existe error en la comunicación se envía un mensaje vía RS232 a la terminal del coordinador, con la función ConsolePutROMString(). Si no ha ocurrido ningún error se pasa a verificar si la información de Cluster que corresponde al Cluster de nivel y posteriormente se lleva a cabo la extracción de la información del sensor de nivel, bit por bit, del mismo modo se realiza la extracción del la información para el otro dispositivo.

Para enviar un mensaje se implementa lo siguiente.

1. Se verifica que la capa de aplicación está lista para enviar un nuevo mensaje, esto se lleva a cabo “observando” si la bandera ZigBeeReady() esta en TRUE.
2. Bloqueo del sistema con la función ZigBeeBlockTx() para que luego se pase la función anterior a FALSE, Zigbee Listo.
3. Se carga la información del mensaje en el array TxBuffer, usando TxData para indexar las posiciones dentro del mismo. Cuando está completada la operación, TxData debe colocarse en la primera posición del mensaje (e.g. TxData igual a la longitud de los datos).
4. Cargar los parámetros de la primitiva APSDE_DATA_request.
5. Colocar a currentPrimitive en APSDE_DATA_request y llamar a la función ZigbeeTask().

Los mensajes son enviados por la aplicación en dos lugares:

- En APSDE_DATA_indication en proceso, en respuesta a un mensaje recibido.

- En NO_PRIMITIVE en proceso, en respuesta a un evento de aplicación.

El proceso de enviar un mensaje es idéntico para ambos casos.

A continuación se presenta el código implementado para tal efecto.

1.1.3. Envío de Mensaje

```
if (ZigBeeReady())
       {

       if ( myStatusFlags.bits.bLevelSwitchToggled )
          {

            myStatusFlags.bits.bLevelSwitchToggled = FALSE;
            ZigBeeBlockTx();

            TxBuffer[TxData++] = APL_FRAME_TYPE_KVP | 1;
            TxBuffer[TxData++] = APLGetTransId();
            TxBuffer[TxData++] = APL_FRAME_COMMAND_GETACK | (Level_CmStr_DATATYPE << 4);
            TxBuffer[TxData++] = Level_CmStr & 0xFF;

            TxBuffer[TxData++] = (Level_CmStr >> 8) & 0xFF;
            //Mensaje Directo
            params.APSDE_DATA_request.DstAddrMode = APS_ADDRESS_16_BIT;
            params.APSDE_DATA_request.DstEndpoint = EP_LEVEL_RFD;
            params.APSDE_DATA_request.DstAddress.ShortAddr = destinationAddress;
            params.APSDE_DATA_request.ProfileId.Val = MY_PROFILE_ID;
            params.APSDE_DATA_request.RadiusCounter = DEFAULT_RADIUS;
            params.APSDE_DATA_request.DiscoverRoute = ROUTE_DISCOVERY_ENABLE;
            params.APSDE_DATA_request.TxOptions.Val = 0;
            params.APSDE_DATA_request.SrcEndpoint = EP_LEVEL;
            params.APSDE_DATA_request.ClusterId = Level_CLUSTER;

            ConsolePutROMString( (ROM char *)" Solicitando el estado del Nivel..." );
            currentPrimitive = APSDE_DATA_request;
        }
```

1.2. Programa Del Dispositivo RFD

Para configurar un dispositivo como RFD se deben seguir los siguientes pasos:

1. Generar los archivos myZigbee.c, zigbee.def, zlink.lkr con el sofware Zena de Microchip.
2. Crear el proyecto en MPLAB incluyendo los archivos necesarios para la aplicación.
3. Crear el archivo principal teniendo en cuenta los parámetros presentados anteriormente

Se tiene el mismo modelo de programación a excepción de algunas variantes:

Archivo zigbee.def generado por Zena

zigbee.def

```
// Created by ZENA(TM)
```

```
#ifndef _ZIGBEE_DEF
#define _ZIGBEE_DEF
#define MAC_LONG_ADDR_BYTE7 0x00
#define MAC_LONG_ADDR_BYTE6 0x04
#define MAC_LONG_ADDR_BYTE5 0xA3
#define MAC_LONG_ADDR_BYTE4 0x00
#define MAC_LONG_ADDR_BYTE3 0x00
#define MAC_LONG_ADDR_BYTE2 0x00
#define MAC_LONG_ADDR_BYTE1 0x00
#define MAC_LONG_ADDR_BYTE0 0x70     // <--
#define I_AM_END_DEVICE              // <--
#define I_AM_RFD                     // <--
#define INCLUDE_ED_SCAN
#define INCLUDE_ACTIVE_SCAN
#define MY_CAPABILITY_INFO 0x80
#define MAX_APL_FRAMES 4
#define MAX_APS_ACK_FRAMES 4
#define MAX_APS_ADDRESSES 0
#define NUM_BUFFERED_INDIRECT_MESSAGES 0
#define MAX_NEIGHBORS 5
#define NUM_BUFFERED_BROADCAST_MESSAGES 0
#define NUM_BUFFERED_ROUTING_MESSAGES 0
#define MINIMUM_JOIN_LQI 48
#define RX_BUFFER_SIZE 256
#define RX_BUFFER_LOCATION 0xa00
#define MAC_PIB_macBeaconOrder 15
#define MAC_PIB_macSuperframeOrder 15
#define IEEE_COMPLY
#define MAC_PIB_macBattLifeExt FALSE
#define RF_CHIP CC2420
#define MAC_PIB_macAckWaitDuration (ONE_SECOND/2)
#define FREQUENCY_BAND FB_2400GHz
#define PHY_CSn LATC0
#define PHY_CSn_TRIS TRISC0
#define PHY_FIFO RB0
#define PHY_FIFO_TRIS TRISB0
#define PHY_FIFOP RB3
#define PHY_FIFOP_TRIS TRISB3
#define PHY_RESETn LATC2
#define PHY_RESETn_TRIS TRISC2
#define PHY_SFD RB2
#define PHY_SFD_TRIS TRISB2
#define PHY_VREG_EN LATC1
#define PHY_VREG_EN_TRIS TRISC1
#define PA_LEVEL 0xEF  // -7 dBm (12.5 mA)
#define ALLOWED_CHANNELS 0x00001000
#define CLOCK_FREQ 4000000
#define BAUD_RATE 19200
#define HEAP_LOCATION 0x0100
#define MAX_HEAP_SIZE 2048
#define NUM_USER_ENDPOINTS 1
#define EP_LEVEL 3                          // <--
#define MY_MAX_INPUT_CLUSTERS 0
#define MY_MAX_OUTPUT_CLUSTERS 1
#include "C:\MpZbee\ZigbeeStack\zICLevel.h"
#endif
```

El archivo myZigBee.c presenta la siguiente configuración de acuerdo al tipo de dispositivo a programar, en este caso un dispositivo medidor de nivel:

myZigBee.c

```
// Created by ZENA(TM)

#include "zigbee.def"
#include "zNWK.h"
#include "zZDO.h"

ROM NODE_DESCRIPTOR Config_Node_Descriptor =
{
    0x02,               // ZigBee End Device
    0x00,               // (reserved)
    0x00,               // (APS Flags, not currently used)
    0x08,               // Frequency Band 2400
```

```
    MY_CAPABILITY_INFO, // Capability Information
    {0x00, 0x00},       // Manufacturer Code
    0x7F,               // Max Buffer Size
    {0x7F, 0x00}        // Max Transfer Size
};

ROM NODE_POWER_DESCRIPTOR Config_Power_Descriptor =
{
    0x01, //RxPeriodic
    0x02, //DispBatt
    0x02, //DispBatt
    0x0c  //Fill in current power level
};

ROM NODE_SIMPLE_DESCRIPTOR Config_Simple_Descriptors[2] =
{
//--------------------------------------
// ZigBee Device Object Endpoint
// DO NOT MODIFY THIS DESCRIPTOR!!!
//--------------------------------------
    {
        EP_ZDO,
        {MY_PROFILE_ID_LSB,MY_PROFILE_ID_MSB},
        {LEVEL_SENSOR_DEV_ID_LSB,LEVEL_SENSOR_DEV_ID_MSB},
        LEVEL_SENSOR_DEV_VER,
        NO_OTHER_DESCRIPTOR_AVAILABLE,
        ZDO_INPUT_CLUSTERS,
        { NWK_ADDR_req, IEEE_ADDR_req, NODE_DESC_req, POWER_DESC_req,
          SIMPLE_DESC_req, ACTIVE_EP_req, MATCH_DESC_req
        },
        ZDO_OUTPUT_CLUSTERS,
        { NWK_ADDR_rsp, IEEE_ADDR_rsp, NODE_DESC_rsp, POWER_DESC_rsp,
          SIMPLE_DESC_rsp, ACTIVE_EP_rsp, MATCH_DESC_rsp
        }
    }
,
//--------------------------------------
    {
        EP_LEVEL,
        {MY_PROFILE_ID_LSB,MY_PROFILE_ID_MSB},
        {LEVEL_SENSOR_DEV_ID_LSB,LEVEL_SENSOR_DEV_ID_MSB},
        LEVEL_SENSOR_DEV_VER,
        NO_OTHER_DESCRIPTOR_AVAILABLE,
        0,
        { NO_CLUSTER },
        1,
        {
            Level_CLUSTER
        }
    }
};

ROM _Config_NWK_Mode_and_Params Config_NWK_Mode_and_Params =
{
    nwkcProtocolVersion,        //Protocol Version
    MY_STACK_PROFILE_ID,        //Stack Profile ID
    MAC_PIB_macBeaconOrder,     //Beacon Order
    MAC_PIB_macSuperframeOrder, //Superframe Order
    MAC_PIB_macBattLifeExt,     //Battery Life Extension
    PROFILE_nwkSecurityLevel,   //Security Level
    ALLOWED_CHANNELS            //Channels to scan
};
```

Por ultimo se tiene el zLink.lkr, que es el que configura los bancos de memoria del PIC.

El programa principal del RFD implemente el siguiente código para gestión de los datos a enviar, por petición del Coordinador:

1.2.1. Captura de Dato de Nivel RFD

```
if (ZigBeeReady())
       {
         char *ptr;

         ZigBeeBlockTx();
         TxBuffer[TxData++] = APL_FRAME_TYPE_KVP | 1;    // KVP, 1 transaction
         TxBuffer[TxData++] = sequenceNumber;
         TxBuffer[TxData++] = APL_FRAME_COMMAND_GET_RES | (Level_CmStr_DATATYPE << 4);
         TxBuffer[TxData++] = Level_CmStr & 0xFF;         // Attribute ID LSB
         TxBuffer[TxData++] = (Level_CmStr >> 8) & 0xFF;  // Attribute ID MSB

         TxBuffer[TxData++] = KVP_SUCCESS;
         ptr = (char *) &(TxBuffer[TxData]);
         ptr++;
         TxBuffer[TxData] = GetADCString( ptr );// Leer Nivel
         ConsolePutString( (BYTE *)ptr );
         TxData += TxBuffer[TxData] + 1;

         if (params.APSDE_DATA_indication.SrcAddrMode == APS_ADDRESS_NOT_PRESENT)
            {
              params.APSDE_DATA_request.DstAddrMode = APS_ADDRESS_NOT_PRESENT;
            }
          else
            {
              params.APSDE_DATA_request.DstAddrMode = APS_ADDRESS_16_BIT;
              params.APSDE_DATA_request.DstEndpoint =
params.APSDE_DATA_indication.SrcEndpoint;
              params.APSDE_DATA_request.DstAddress.ShortAddr =
params.APSDE_DATA_indication.SrcAddress.ShortAddr;
             }

           params.APSDE_DATA_request.RadiusCounter = DEFAULT_RADIUS;
           params.APSDE_DATA_request.DiscoverRoute = ROUTE_DISCOVERY_ENABLE;
           params.APSDE_DATA_request.TxOptions.Val = 0;
           params.APSDE_DATA_request.SrcEndpoint = EP_LEVEL;

           ConsolePutROMString( (ROM char *)" Enviando mensaje de Nivel.\r\n" );
           currentPrimitive = APSDE_DATA_request;

         }
```

El código para hacer la conversión ADC de la señal eléctrica del sensor de nivel, se muestra a continuación:

1.2.2. Adquisición ADC y Escalización

```
#include <p18f4620.h>
#include <adc.h>
#include <stdlib.h>
#include <string.h>
#include "Console.h"
#include "generic.h"
#include "adcFx.h"

#define NivelLSB      ADRESL
#define NivelMSB      ADRESH

BYTE GetADCString(char *buffer)
{
       typedef union _SIGNED_WORD_VAL
       {
              unsigned int uVal;
              int Val;
              struct
              {
                     BYTE LSB;
                     BYTE MSB;
              } byte;
       }SIGNED_WORD_VAL;

    BYTE            GIEHsave;
    char            *ptr;
    char            *pLevelString;
```

```
SIGNED_WORD_VAL RawLevel;
signed long     ScaledLevel;
BYTE            strLen;
char            LevelString[10];
// Leer ADC (NIvel)
// El nivel del tanque esta dado por la señal de voltaje de 0 a 5Vdc,
// el ADC trabaja con este nivel de voltaje y lo convierte a un
// entero de 10 bits en donde 0 = 0cm
// 1023 = 30 cm.La función es Y = 0.02932X Donde Y = nivel en cm
// y X = bits (10bits 0-1023) Para propósitos de presentación,
// este valor es convertido a un numero de punto flotante
// y multiplicado por 0.02932 cm.
// Alternativamente, este puede dejarse como entero
// y multiplicarse por 2932.  El resultado puede ser escalizado
// con un factor de 10,0000.


// Deshabilitando las interrupciones mientras se adquiere la señal del sensor
GIEHsave = GIEH;
GIEH = 0;

    // Iniciando la conversión
    ConsolePutROMString( (ROM char *)"--> Conversion iniciada...\r\n\r\n" );
    ADON    = 1;
    ADGO    = 1;

    while(ADGO);                        // Conversión terminada??
    RawLevel.byte.MSB = NivelMSB;       // Adquiriendo datos
    RawLevel.byte.LSB = NivelLSB;

// Restaurando las interrupciones.
GIEH = GIEHsave;

if ((RawLevel.uVal == 0x0000) || (RawLevel.uVal == 0xFFFF))
{
    strcpy( buffer, (const char *)"Lectura erronea." );
}
else
{
    // Convertir el nivel del tanque a un nivel escalizado a unidades
    // de 0.0001 cm

    ScaledLevel = ((long)RawLevel.Val * (long)2932)+ (long)564;

     // Convirtiendo el entero a un string y sacando su longitud
     pLevelString = ultoa( ScaledLevel, LevelString );  // Puntero de memoria
     strLen = strlen( LevelString );                    // Longitud del entero


     ptr = buffer;

     // Escribir "0." Si el número está muy cerca a cero; ej: 0, 0.0625, 0.125
     if(strLen < 5)
     {
         *ptr++ = '0';
         *ptr++ = '.';
         memcpy( (void *)ptr, (void *)LevelString, strLen );
         *ptr += strLen;
     }
     // Por otra parte si el número esta completo
     // la parte decimal se coloca en la posición 5
     // ej: 24.1250
     else
     {
         for(;strLen >= 6; strLen--)
             *ptr++ = *pLevelString++;
             *ptr++ = '.';
         for(;strLen > 0; strLen--)
             *ptr++ = *pLevelString++;
     }

     // Escribir cm
     *ptr++ = ' ';
     *ptr++ = 'c';
      *ptr++ = 'm';
     *ptr++ = '\n';
      *ptr++ = '\r';
```

```
    }
    return strlen( buffer );
}
```

2. ANEXO B: CÓDIGO DEL PROGRAMA DEL SISTEMA SUPERVISORIO

A continuación se muestra el código utilizado para la construcción del sistema de supervisión, el cual fue realizado en la plataforma de programación hardware JAVA.

Este programa consta de un programa principal llamado serial.java y de 5 clases adicionales que le dan soporte a las diferentes funciones cumplidas, las cuales son llamadas [5]:

- ConnectionException.java
- conexionSerial.java
- dialogoAlerta.java
- mensajePeticionPuerto.java
- parametrosSeriales.java

2.1. Código serial.java

```
/*
 * serial.java
 *
 * Created on 27 de mayo de 2007, 09:31 AM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package seriales;

/**
 *
 * @author Administrador
 */
import java.io.OutputStream;
import javax.comm.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.Properties;
import java.util.Enumeration;
import objetoComunicacion.parametrosSeriales;
import objetoComunicacion.conexionSerial;
import objetoComunicacion.dialogoAlerta;
import objetoComunicacion.SerialConnectionException;

public class serial extends Frame implements ActionListener {
    final int HEIGHT = 650;
    final int WIDTH  = 610;
    int P;
    String  defaultPort ;
```

```
//se crean Los Menus y botones a Usar
    private MenuBar mb;
    private Menu fileMenu;
    private Menu portMenu;
    private MenuItem openItem;
    private MenuItem closeItem;
    private MenuItem clearItem;
    private MenuItem exitItem;
    private MenuItem p1Item;
    private MenuItem p2Item;
    boolean rfd1;
    private Button      nivelButton;
    private Button      controlButton;
    private Panel buttonPanel;
 // Se crean los paneles de entrada y salida de datos
    private Panel    messagePanel;
    private TextArea messageAreaOut;
    private TextArea messageAreaIn;
    private ConfigurationPanel configurationPanel;
    private parametrosSeriales parametros;
    private conexionSerial conexion;
    private Properties props = null;
    /** Creates a new instance of serial */
    public serial(String args[]) {
        super("Sistema de Supervisión de la Red Inalámbrica IEEE
802.15.4");
      parametros = new parametrosSeriales();
// Configuración del GUI
      addWindowListener(new CloseHandler(this));
// se adicionan los menus y los botones a su respectivo menu
      mb = new MenuBar();
      fileMenu = new Menu("Archivo");
      portMenu = new Menu("Puerto");
      p1Item      =     new MenuItem("COM1");
      p1Item.addActionListener(this);
      portMenu.add(p1Item);
      p2Item      =     new MenuItem("COM2");
      p2Item.addActionListener(this);
      portMenu.add(p2Item);
      fileMenu.add(portMenu);
      openItem = new MenuItem("Abrir Puerto");
      openItem.addActionListener(this);
      fileMenu.add(openItem);
      closeItem = new MenuItem("Cerrar Puerto");
      closeItem.addActionListener(this);
      closeItem.setEnabled(false);
      fileMenu.add(closeItem);
      exitItem = new MenuItem("Salir");
      exitItem.addActionListener(this);
      fileMenu.add(exitItem);
      mb.add(fileMenu);
      setMenuBar(mb);
//se adiciona  los diferentes paneles usados
      messagePanel = new Panel();
      messagePanel.setLayout(new GridLayout(1, 1));
      messageAreaOut = new TextArea();
      messageAreaIn = new TextArea();
      messagePanel.add(messageAreaIn);
      add(messagePanel, "Center");
      configurationPanel = new ConfigurationPanel(this);
      buttonPanel = new Panel();
```

```
// A los Botones Agragados se le Agraga las Etiquetas y se invoca sus
funciones
      nivelButton = new Button("Nodo RFD1(Nivel de Los Tanques)");
      nivelButton.addActionListener(this);
      nivelButton.setEnabled(true);
      buttonPanel.add(nivelButton);
      controlButton = new Button("Nodo RFD2(Señal de Control)");
      controlButton.addActionListener(this);
      controlButton.setEnabled(true);
      buttonPanel.add(controlButton);
      Panel southPanel = new Panel();
      GridBagLayout gridBag = new GridBagLayout();
      GridBagConstraints cons = new GridBagConstraints();
      southPanel.setLayout(gridBag);
      cons.gridwidth = GridBagConstraints.REMAINDER;
      gridBag.setConstraints(configurationPanel, cons);
      cons.weightx = 1.0;
      southPanel.add(configurationPanel);
      gridBag.setConstraints(buttonPanel, cons);
      southPanel.add(buttonPanel);
      add(southPanel, "South");
      parseArgs(args);
      conexion = new conexionSerial(this, parametros,
                                messageAreaOut, messageAreaIn,rfd1);
      setConfigurationPanel();
      Dimension screenSize =
      Toolkit.getDefaultToolkit().getScreenSize();
      setLocation(screenSize.width/2 - WIDTH/2,
                screenSize.height/2 - HEIGHT/2);
      setSize(WIDTH, HEIGHT);
    }
    public void setConfigurationPanel() {
    configurationPanel.setConfigurationPanel();
    }
    public void actionPerformed(ActionEvent e) {
    String cmd = e.getActionCommand();
      // Loads a configuration file.
      if (cmd.equals("Abrir Puerto")) {
            openItem.setEnabled(false);
                portMenu.setEnabled(false);
          Cursor previousCursor = getCursor();
setNewCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
          configurationPanel.setParameters();//////////var
          try {
            conexion.abrirConexion();
          } catch (SerialConnectionException e2) {
            dialogoAlerta ad = new dialogoAlerta(this,
                              "Error Abriendo el Puerto!",
                              "Error Abriendo el Puerto,",
                              e2.getMessage() + ".",
                              "Seleccionar otro Puerto o Configurar de
Nuevo.");
              openItem.setEnabled(true);
                portMenu.setEnabled(true);
            setNewCursor(previousCursor);
            return;
          }
          portOpened();
          setNewCursor(previousCursor);
      }
```

```
      if (cmd.equals("Cerrar Puerto")) {
            portClosed();
                portMenu.setEnabled(true);

      }
       if (cmd.equals("Limpiar")) {
            messageAreaIn.repaint();
      }
      if (cmd.equals("COM1")) {
            defaultPort="COM1";
      }
      if (cmd.equals("COM2")) {
            defaultPort="COM2";
      }
     if (cmd.equals("Nodo RFD1(Nivel de Los Tanques)")) {
    conexion.enviard();
          rfd1 = true;
     }
        if (cmd.equals("Nodo RFD2(Señal de Control)")) {
      conexion.enviard2();
        }
      if (cmd.equals("Salir")) {
          shutdown();
      }
    }
    public void portOpened() {
      openItem.setEnabled(false);
      closeItem.setEnabled(true);
    }
    public void portClosed() {
      conexion.closeConnection();
      openItem.setEnabled(true);
      closeItem.setEnabled(false);
    }
    private void setNewCursor(Cursor c) {
      setCursor(c);
      messageAreaIn.setCursor(c);
      messageAreaOut.setCursor(c);
    }
    private void writeFile(String path) {
        Properties newProps;
        FileOutputStream fileOut = null;
        newProps = new Properties();
        newProps.put("portName", parametros.getPortName());
        newProps.put("baudRate", parametros.getBaudRateString());
        newProps.put("flowControlIn",
parametros.getFlowControlInString());
        newProps.put("flowControlOut",
parametros.getFlowControlOutString());
        newProps.put("parity", parametros.getParityString());
        newProps.put("databits", parametros.getDatabitsString());
        newProps.put("stopbits", parametros.getStopbitsString());
        try {
            fileOut = new FileOutputStream(path);
        } catch (IOException e) {
            System.out.println("No Se Puede Abrir Archivo Para
Escribir");
        }
        newProps.save(fileOut, "Propidades de Demo Serial");
        try {
```

```
            fileOut.close();
        } catch (IOException e) {
            System.out.println("No Se Puede Abrir Archivo Para
Escribir");
        }
    }
     //Función Para Salir
    private void shutdown() {
      conexion.closeConnection();
      System.exit(1);
    }
    private void parseArgs(String[] args) {
      if (args.length < 1) {
          return;
      }
      File f = new File(args[0]);
      if (!f.exists()) {
          f = new File(System.getProperty("user.dir")
                   + System.getProperty("path.separator")
                   + args[0]);
      }
      if (f.exists()) {
          try {
            FileInputStream fis = new FileInputStream(f);
            props = new Properties();
            props.load(fis);
            fis.close();
            loadParams();
          } catch (IOException e) {
          }
      }
    }
    private void loadParams() {
      parametros.setPortName(props.getProperty("portName"));
      parametros.setBaudRate(props.getProperty("baudRate"));
      parametros.setFlowControlIn(props.getProperty("flowControlIn"));
      parametros.setFlowControlOut(props.getProperty("flowControlOut")
);
      parametros.setParity(props.getProperty("parity"));
      parametros.setDatabits(props.getProperty("databits"));
      parametros.setStopbits(props.getProperty("stopbits"));
      setConfigurationPanel();
    }
    class ConfigurationPanel extends Panel implements ItemListener {
      private Frame parent;
      private Label portNameLabel;
      private Choice portChoice;
      private Label baudLabel;
      private Choice baudChoice;
      private Label flowControlInLabel;
      private Choice flowChoiceIn;
      private Label flowControlOutLabel;
      private Choice flowChoiceOut;
      private Label databitsLabel;
      private Choice databitsChoice;
      private Label stopbitsLabel;
      private Choice stopbitsChoice;
      private Label parityLabel;
      private Choice parityChoice;
      public ConfigurationPanel(Frame parent) {
          this.parent = parent;
```

```
          setLayout(new GridLayout(4, 4));
          portNameLabel = new Label("Port Name:", Label.LEFT);
          portChoice = new Choice();
          portChoice.addItemListener(this);
          listPortChoices();
          portChoice.select(parametros.getPortName());
          baudLabel = new Label("Baud Rate:", Label.LEFT);
          baudChoice = new Choice();
          baudChoice.addItem("19200");
baudChoice.select(Integer.toString(parametros.getBaudRate()));
          baudChoice.addItemListener(this);
          flowControlInLabel = new Label("Flow Control In:",
Label.LEFT);
          flowChoiceIn = new Choice();
          flowChoiceIn.addItem("None");
          flowChoiceIn.select(parametros.getFlowControlInString());
          flowChoiceIn.addItemListener(this);
          flowControlOutLabel = new Label("Flow Control Out:",
Label.LEFT);
          flowChoiceOut = new Choice();
          flowChoiceOut.addItem("None");
          flowChoiceOut.select(parametros.getFlowControlOutString());
          flowChoiceOut.addItemListener(this);
          databitsLabel = new Label("Data Bits:", Label.LEFT);
          databitsChoice = new Choice();
          databitsChoice.addItem("8");
          databitsChoice.select(parametros.getDatabitsString());
          databitsChoice.addItemListener(this);
          stopbitsLabel = new Label("Stop Bits:", Label.LEFT);
          stopbitsChoice = new Choice();
          stopbitsChoice.addItem("1");
          stopbitsChoice.select(parametros.getStopbitsString());
          stopbitsChoice.addItemListener(this);
          parityLabel = new Label("Parity:", Label.LEFT);
          parityChoice = new Choice();
          parityChoice.addItem("None");
          parityChoice.select(parametros.getParityString());
          parityChoice.addItemListener(this);
      }
      /**
Configuración de los parámetros de conexión
      */
      public void setConfigurationPanel() {
            portChoice.equals(defaultPort);
            // portChoice.select(parameters.getPortName());
          baudChoice.select(parametros.getBaudRateString());
          flowChoiceIn.select(parametros.getFlowControlInString());
          flowChoiceOut.select(parametros.getFlowControlOutString());
          databitsChoice.select(parametros.getDatabitsString());
          stopbitsChoice.select(parametros.getStopbitsString());
          parityChoice.select(parametros.getParityString());
      }
        public void setParameters() {
            parametros.setPortName(defaultPort);
         // parameters.setPortName(portChoice.getSelectedItem());
            parametros.setBaudRate(baudChoice.getSelectedItem());
parametros.setFlowControlIn(flowChoiceIn.getSelectedItem());
parametros.setFlowControlOut(flowChoiceOut.getSelectedItem());
            parametros.setDatabits(databitsChoice.getSelectedItem());
            parametros.setStopbits(stopbitsChoice.getSelectedItem());
```

```
            parametros.setParity(parityChoice.getSelectedItem());
        }
        void listPortChoices() {
            CommPortIdentifier portId;
            String   defaultPort = "COM1";
            Enumeration en = CommPortIdentifier.getPortIdentifiers();
            // Buscan los Puertos Disponibles
            while (en.hasMoreElements()) {
                portId = (CommPortIdentifier) en.nextElement();
                if (portId.getPortType() ==
CommPortIdentifier.PORT_SERIAL) {
                    portChoice.addItem(portId.getName());
                }
            }

        }
      public void itemStateChanged(ItemEvent e) {
          if (conexion.isOpen()) {
            if (e.getItemSelectable() == portChoice) {
                // Alerta a Usuario.
                dialogoAlerta ad = new dialogoAlerta(parent, "Puerto ya
abierto!",
                                                "El Puerto no Puede",
                                                "Ser Cambiado",
                                                "Mietra este en Uso.");
                setConfigurationPanel();
                return;
            }
            setParameters();
            try {
                // Intenta Cambiar la Configuración del Puerto.
                conexion.setConnectionParameters();
            } catch (SerialConnectionException ex) {
                dialogoAlerta ad = new dialogoAlerta(parent,
                  "Configuración no Soportada!",
                  "Parametros de configuración no soportados,",
                  "Seleccione un nuevo Valor.",
                  "Retornar a la Configuración Previa.");
                setConfigurationPanel();
            }
          } else {
            setParameters();
          }
      }
    }
    class CloseHandler extends WindowAdapter {
      serial sd;
      public CloseHandler(serial sd) {
          this.sd = sd;
      }
      public void windowClosing(WindowEvent e) {
          sd.shutdown();
      }
    }
    public static void main(String[] args) {
        if ((args.length > 0)
          && (args[0].equals("-h")
          || args[0].equals("-help"))) {
          System.out.println("usage: java SerialDemo [configuration
File]");
          System.exit(1);
```

```
      }
      serial Serial = new serial(args);
      Serial.setVisible(true);
      Serial.repaint();
    }
    }
```

2.2. Código: SerialConnectionException.java

```
package objetoComunicacion;

public class SerialConnectionException extends Exception{

    /** Crea a nueva instancia de SerialConnectionException */
    public SerialConnectionException(String str) {
        super(str);
    }
    public SerialConnectionException(){
    super();
    }
}
```

2.3. Código conexionSerial.java

```
package objetoComunicacion;
import javax.comm.*;
import java.io.*;
import java.awt.TextArea;
import seriales.serial;
import objetoComunicacion.parametrosSeriales;
import objetoComunicacion.SerialConnectionException;
import objetoComunicacion.mensajePeticionPuerto;
import java.awt.event.*;
import java.util.TooManyListenersException;

public class conexionSerial implements
SerialPortEventListener,CommPortOwnershipListener{
    private serial parent;
    private TextArea messageAreaOut;
    private TextArea messageAreaIn;
    private OutputStream os;
    private InputStream is;
    private parametrosSeriales parametros;
    private KeyHandler keyHandler;
    private CommPortIdentifier portId;
    private SerialPort sPort;
    boolean rfd1;
    boolean rfd2;
    private boolean open;
    /** Crea una nueva instancia de conexionSerial */
    public conexionSerial(serial parent,
                    parametrosSeriales parametros,
                    TextArea messageAreaOut,
                    TextArea messageAreaIn,boolean rfd1) {
      this.parent = parent;
      this.parametros = parametros;
      this.messageAreaOut = messageAreaOut;
      this.messageAreaIn = messageAreaIn;
        this.rfd1=rfd1;
```

```
      open = false;
    }
    // función que se encarga de enviar un dato al RFD que maneja la
señal de nivel
    public void enviard(){
        char newCharacter = '1';
          try {
         os.write((int)newCharacter);
          } catch (IOException e) {
            System.err.println("errror de escritura: " + e);
          }
        }
    // función que se encarga de enviar un dato al RFD que maneja la
señal de la válvula
    public void enviard2(){
        char newCharacter = '2';
          try {
         os.write((int)newCharacter);
          } catch (IOException e) {
            System.err.println("Error de escritura: " + e);
          }
        }
    //Se  Abre el puerto
    public void abrirConexion()throws SerialConnectionException{

      try {
          portId =
CommPortIdentifier.getPortIdentifier(parametros.getPortName());
      } catch (NoSuchPortException e) {
          throw new SerialConnectionException(e.getMessage());
      }
      try {
          sPort = (SerialPort)portId.open("Serial", 30000);
      } catch (PortInUseException e) {
          throw new SerialConnectionException(e.getMessage());
      }
      try {
          setConnectionParameters();
      } catch (SerialConnectionException e) {
          sPort.close();
          throw e;
      }
      try {
          os = sPort.getOutputStream();
          is = sPort.getInputStream();
      } catch (IOException e) {
          sPort.close();
          throw new SerialConnectionException("Error Abriendo streams
de i/o ");
      }

/*se crea una función que se encarga de enviar
cualquier dato tecleado en el área de salida desde el teclado
sin embargo. no se utiliza en esta aplicación
 */
      keyHandler = new KeyHandler(os);
      messageAreaOut.addKeyListener(keyHandler);

      // se crea la llamada a una función de escucha en el puerto
serial
      try {
```

```
            sPort.addEventListener(this);
        } catch (TooManyListenersException e) {
            sPort.close();
            throw new SerialConnectionException("demaciados listener's
adicionados");
        }
        sPort.notifyOnDataAvailable(true);
        sPort.notifyOnBreakInterrupt(true);
        try {
            sPort.enableReceiveTimeout(30);
        } catch (UnsupportedCommOperationException e) {
        }
        portId.addPortOwnershipListener(this);
        open = true;
    }
    public void setConnectionParameters() throws
SerialConnectionException {
        int oldBaudRate = sPort.getBaudRate();
        int oldDatabits = sPort.getDataBits();
        int oldStopbits = sPort.getStopBits();
        int oldParity   = sPort.getParity();
        int oldFlowControl = sPort.getFlowControlMode();
        try {
            sPort.setSerialPortParams(19200,
                                SerialPort.DATABITS_8,
                                SerialPort.STOPBITS_1,
                                   SerialPort.PARITY_NONE);
        } catch (UnsupportedCommOperationException e) {
            parametros.setBaudRate(oldBaudRate);
            parametros.setDatabits(oldDatabits);
            parametros.setStopbits(oldStopbits);
            parametros.setParity(oldParity);
            throw new SerialConnectionException("Parametro No
Soportado");
        }
        try {
            sPort.setFlowControlMode(parametros.getFlowControlIn()
                                | parametros.getFlowControlOut());
        } catch (UnsupportedCommOperationException e) {
            throw new SerialConnectionException("Control de Flujo No
Soportado");
        }
    }
    public void closeConnection() {
        if (!open) {
            return;
        }
        messageAreaOut.removeKeyListener(keyHandler);
        if (sPort != null) {
            try {
              os.close();
              is.close();
            } catch (IOException e) {
              System.err.println(e);
            }
            sPort.close();
            portId.removePortOwnershipListener(this);
        }

        open = false;
    }
```

```
     public void sendBreak() {
      sPort.sendBreak(1000);
    }
    public boolean isOpen() {
      return open;
    }
    public void serialEvent(SerialPortEvent e) {

      StringBuffer inputBuffer = new StringBuffer();
      int newData = 0;
      switch (e.getEventType()) {
          case SerialPortEvent.DATA_AVAILABLE:
                while (newData != -1) {
                  try {
                      newData = is.read();
                      if (newData == -1) {
                        break;
                      }
                      if ('\r' == (char)newData) {
                        inputBuffer.append('\n');
                      } else {
                        inputBuffer.append((char)newData);
                      }
                  } catch (IOException ex) {
                      System.err.println(ex);
                      return;
                        }
                }
            messageAreaIn.append(new String(inputBuffer));
            break;
          case SerialPortEvent.BI:
            messageAreaIn.append("\n--- Break Recivido ---\n");
      }
    }
     public void ownershipChange(int type) {
      if (type == CommPortOwnershipListener.PORT_OWNERSHIP_REQUESTED)
{
          mensajePeticionPuerto prd = new
mensajePeticionPuerto(parent);
      }
    }
     class KeyHandler extends KeyAdapter {
      OutputStream os;
      public KeyHandler(OutputStream os) {
          super();
          this.os = os;
      }
        public void keyTyped(KeyEvent evt) {
           char newCharacter = evt.getKeyChar();
          try {
         os.write((int)newCharacter);
          } catch (IOException e) {
            System.err.println("Error de escritura: " + e);
          }
        }
    }
}
```

2.4. Código dialogoAlerta.java

```
/*
 * dialogoAlerta.java
 *
  * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package objetoComunicacion;
import java.awt.*;
import java.awt.event.*;
/**
 *
 * @author Administrador
 */
public class dialogoAlerta extends Dialog implements ActionListener{

    /** Creates a new instance of dialogoAlerta */
    public dialogoAlerta(Frame parent,
                    String title,
                    String lineOne,
                    String lineTwo,
                    String lineThree) {
        super(parent, title, true);

      Panel labelPanel = new Panel();
      labelPanel.setLayout(new GridLayout(3, 1));
      labelPanel.add(new Label(lineOne, Label.CENTER));
      labelPanel.add(new Label(lineTwo, Label.CENTER));
      labelPanel.add(new Label(lineThree, Label.CENTER));
      add(labelPanel, "Center");

      Panel buttonPanel = new Panel();
      Button okButton = new Button("OK");
      okButton.addActionListener(this);
      buttonPanel.add(okButton);
      add(buttonPanel, "South");

      FontMetrics fm = getFontMetrics(getFont());
      int width = Math.max(fm.stringWidth(lineOne),
            Math.max(fm.stringWidth(lineTwo),
fm.stringWidth(lineThree)));

      setSize(width + 40, 150);
      setLocation(parent.getLocationOnScreen().x + 30,
                parent.getLocationOnScreen().y + 30);
      setVisible(true);
    }
     public void actionPerformed(ActionEvent e) {
      setVisible(false);
      dispose();
    }
}
```

2.5. Código mensajePeticionPuerto.java

```
/*
 * mensajePeticionPuerto.java
 *
```

```
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package objetoComunicacion;
/*

 */
import java.awt.*;
import java.awt.event.*;
import javax.comm.*;
import seriales.serial;

/**
 *
 * @author Administrador
 */
public class mensajePeticionPuerto extends Dialog implements
ActionListener{
    private serial parent;


    public mensajePeticionPuerto(serial parent) {
        super(parent, "Puerto Solicitado!", true);
      this.parent = parent;

      String lineOne = "Su puerto esta siendo Requerido";
      String lineTwo = "Por Otra Aplicación.";
      String lineThree = "Usted desea ceder su Puerto?";
      Panel labelPanel = new Panel();
      labelPanel.setLayout(new GridLayout(3, 1));
      labelPanel.add(new Label(lineOne, Label.CENTER));
      labelPanel.add(new Label(lineTwo, Label.CENTER));
      labelPanel.add(new Label(lineThree, Label.CENTER));
      add(labelPanel, "Center");

      Panel buttonPanel = new Panel();
      Button yesButton = new Button("Si");
      yesButton.addActionListener(this);
      buttonPanel.add(yesButton);
      Button noButton = new Button("No");
      noButton.addActionListener(this);
      buttonPanel.add(noButton);
      add(buttonPanel, "South");

      FontMetrics fm = getFontMetrics(getFont());
      int width = Math.max(fm.stringWidth(lineOne),
            Math.max(fm.stringWidth(lineTwo),
fm.stringWidth(lineThree)));

      setSize(width + 40, 150);
      setLocation(parent.getLocationOnScreen().x + 30,
                parent.getLocationOnScreen().y + 30);
      setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
      String cmd = e.getActionCommand();

          if (cmd.equals("Si")) {
            parent.portClosed();
```

```
            }

            setVisible(false);
            dispose();
        }

}
```

2.6. Código parametrosSeriales.java

```
/*
 * parametrosSeriales.java
 *
 * Created on 27 de mayo de 2007, 09:44 AM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package objetoComunicacion;
/*
 */
import javax.comm.*;
/**
 *
 * @author Administrador
 */
public class parametrosSeriales {

    private String portName;
    private int baudRate;
    private int flowControlIn;
    private int flowControlOut;
    private int databits;
    private int stopbits;
    private int parity;
    /** Crea una nueva instancia de parametrosSeriales */
    public parametrosSeriales() {
        this("",
           19200,
           SerialPort.FLOWCONTROL_NONE,
           SerialPort.FLOWCONTROL_NONE,
           SerialPort.DATABITS_8,
           SerialPort.STOPBITS_1,
           SerialPort.PARITY_NONE );

    }
 public parametrosSeriales(String portName,
                      int baudRate,
                      int flowControlIn,
                      int flowControlOut,
                      int databits,
                      int stopbits,
                      int parity) {

      this.portName = portName;
      this.baudRate = baudRate;
      this.flowControlIn = flowControlIn;
      this.flowControlOut = flowControlOut;
      this.databits = databits;
```

```
      this.stopbits = stopbits;
      this.parity = parity;
    }
public void setPortName(String portName) {
      this.portName = portName;
    }

    public String getPortName() {
      return portName;
    }

    public void setBaudRate(int baudRate) {
      this.baudRate = baudRate;
    }


    public void setBaudRate(String baudRate) {
      this.baudRate = Integer.parseInt(baudRate);
    }


    public int getBaudRate() {
      return baudRate;
    }

    public String getBaudRateString() {
      return Integer.toString(baudRate);
    }


    public void setFlowControlIn(int flowControlIn) {
      this.flowControlIn = flowControlIn;
    }


    public void setFlowControlIn(String flowControlIn) {
      this.flowControlIn = stringToFlow(flowControlIn);
    }

    public int getFlowControlIn() {
      return flowControlIn;
    }


    public String getFlowControlInString() {
      return flowToString(flowControlIn);
    }


    public void setFlowControlOut(int flowControlOut) {
      this.flowControlOut = flowControlOut;
    }


    public void setFlowControlOut(String flowControlOut) {
      this.flowControlOut = stringToFlow(flowControlOut);
    }

    public int getFlowControlOut() {
      return flowControlOut;
    }
```

```
public String getFlowControlOutString() {
  return flowToString(flowControlOut);
}

public void setDatabits(int databits) {
  this.databits = databits;
}


public void setDatabits(String databits) {

  if (databits.equals("8")) {
      this.databits = SerialPort.DATABITS_8;
  }
}

public int getDatabits() {
  return databits;
}


public String getDatabitsString() {
  switch(databits) {

      case SerialPort.DATABITS_8:
        return "8";
      default:
        return "8";
  }
}


public void setStopbits(int stopbits) {
  this.stopbits = stopbits;
}


public void setStopbits(String stopbits) {
  if (stopbits.equals("1")) {
      this.stopbits = SerialPort.STOPBITS_1;
  }

}


public int getStopbits() {
  return stopbits;
}


public String getStopbitsString() {
  switch(stopbits) {
      case SerialPort.STOPBITS_1:
        return "1";
      default:
        return "1";
  }
}
```

```
    public void setParity(int parity) {
      this.parity = parity;
    }

    public void setParity(String parity) {
      if (parity.equals("None")) {
          this.parity = SerialPort.PARITY_NONE;
      }

    }

    public int getParity() {
      return parity;
    }


    public String getParityString() {
      switch(parity) {
          case SerialPort.PARITY_NONE:
            return "None";

          default:
            return "None";
      }
    }


    private int stringToFlow(String flowControl) {
      if (flowControl.equals("None")) {
          return SerialPort.FLOWCONTROL_NONE;
      }

      return SerialPort.FLOWCONTROL_NONE;
    }


    String flowToString(int flowControl) {
      switch(flowControl) {
          case SerialPort.FLOWCONTROL_NONE:
            return "None";

          default:
            return "None";
      }
    }

}
```

BIBLIOGRAFÍA

[1] Stack ZigBee de Microchip descargado de la página:
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2112&pageId=64
Visitada en el año 2006.

[2] Estándar IEEE Extended Unique Identifiers – EUI-64, visto en la página
https://standards.ieee.org/regauth/oui/forms/
Visitada en el mes de febrero del año 2006.

[3] Software de configuración de Zigbee, ZENA, descargado de la página:
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1999&ty=&dty=&section=&NextRow=&ssUserText=ZENA
Visitada en el mes de octubre del año 2005.

[4] software y manuales del Compilador de C, C18 de Microchip, descargados de la página:
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002
Visitada en el año 2006 y 2007.

[5] Software Java de SUN Microsystems, descargado de la página
http://java.sun.com
Visitada en el mes de marzo del año 2006.