XVIII SIMPOSIO DE INGENIERÍA ELÉCTRICA (SIE-2019)

Biblioteca de funciones para configuración y control de módulos PmodRF2

Function library for configuration and control of PmodRF2 modules

Adrian Acuña Pérez[1], Carlos A. Bazán Prieto[2], Ernesto Rivero Pérez[3]

1- Adrian Acuña Pérez. Universidad Central "Marta Abreu" de las Villas, Cuba. E-mail: aacuna@uclv.cu

2-Carlos Alberto Bazán Prieto. Universidad Central "Marta Abreu" de las Villas, Cuba. E-mail: cabazan@uclv.edu.cu

3- Ernesto Rivero Pérez. Universidad Central "Marta Abreu" de las Villas, Cuba. E-mail: ernestorp@uclv.cu

Resumen: El propósito del Estándar IEEE 802.15.4 es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos ubicuos con bajo costo y velocidad. Este estándar especifica el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con bajas tasas de transmisión de datos (LR-WPAN). Los módulos PmodRF2 utilizan el Estándar IEEE 802.15.4-2003 y proporcionan suporte de radio frecuencia para aplicaciones con microprocesadores. Estos módulos son configurables y no programables; los registros de control son los encargados de modificar y controlar su funcionamiento. El medio de comunicación principal entre los módulos y los microprocesadores es un bus SPI. En la actualidad no se dispone de una biblioteca que posea las funciones necesarias para implementar un sistema de comunicación empleando estos dispositivos. En este trabajo se desarrolla e implementa una biblioteca que permite el manejo de los registros de control del módulo PmodRF2. Esta biblioteca, lib_PmodRF2, se implementa en lenguaje C++ para placas de desarrollo Arduino. Se presenta una aplicación de comunicación entre dos módulos PmodRF2 conectados a placas Arduino, configurados con el uso de lib_PmodRF2.

Abstract: *The purpose of IEEE 802.15.4 Standard is to define the basic network levels to service a specific type of wireless personal area network (WPAN) focused on enabling communication between ubiquitous devices with low cost and speed. This standard specifies the physical level and access control of low-rate wireless personal area networks (LR-WPAN). PmodRF2 modules use IEEE 802.15.4-2003 Standard and provide radio frequency support for applications with microprocessor. These modules are configurable and not programmable; control registers are responsible for modifying and controlling their operation. At present, there is no library that has the necessary functions to implement a communication system using these devices. The main communication medium between the modules and the microprocessors is an SPI bus. In this work, a library that allows the management of control registers of PmodRF2 module is developed and implemented. This library, lib_PmodRF2, is implemented in C ++ language for Arduino development boards. A communication application is presented between two PmodRF2 modules connected to Arduino boards, configured with the use of lib_PmodRF2.*

Palabras Clave: Comunicación inalámbrica; Estándar IEEE 802.15.4; Módulos PmodRF2; Arduino.

Keywords: *Wireless communication; IEEE 802.15.4 Standard; PmodRF2 modules; Arduino.*

1. Introducción

El desarrollo científico-tecnológico actual, impulsa a la humanidad a implementar nuevas herramientas y modernizar o adaptar las existentes en una interminable búsqueda por mejorar la calidad de vida. Es innegable el auge de las tecnologías y técnicas de comunicaciones que invaden la cotidianidad. Cada día es mayor la tendencia a utilizar medios inalámbricos para el intercambio de información. Además, la miniaturización de los componentes, ha propiciado la creación de equipos portátiles que permiten emplear estructuras de redes muy complejas con un despliegue de elementos muy sencillo.

Para el desarrollo de estos dispositivos fue necesaria la creación de un estándar que regulara la base de la comunicación y a su vez optimizara el consumo de energía, con vista a extender la duración de sus baterías. Así surgieron varios grupos de trabajo en el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE Standards Association) y se definieron algunos estándares, para disímiles tecnologías en Redes de Área Personal (PAN). Uno de estos es el Estándar IEEE 802.15.4, que define la capa física (PHY) y el control de acceso al medio (MAC) para Redes de Área Personal Inalámbricas (WPAN) con baja tasa de transmisión (LR-WPAN) (IEEE Standards Association, 2015).

El Estándar IEEE 802.15.4 es la base de las Redes de Sensores Inalámbricas (WSN), las cuales han aprovechado en gran medida la evolución tecnológica. Las WSN tienen numerosas aplicaciones potenciales entre las que se incluyen el monitoreo de salud, la agricultura de precisión, sistemas de detección de intrusos, monitoreo de las condiciones ambientales, detección precoz de desastres naturales, rastreo de objetos y otros escenarios de aplicación, relevantes tanto para la industria como para la sociedad en general (Dargie and Poellabauer, 2010).

Los módulos PmodRF2 fabricados por Digilent (Digilent Inc.) utilizan el Estándar IEEE 802.15.4-2003 (LAN/MAN Standards Committee, 2003), estos añaden comunicación inalámbrica a proyectos sencillos, con microprocesadores. Además logran que dispositivos portátiles como computadoras personales, teléfonos, sensores y actuadores utilizados en domótica, entre otros, puedan comunicarse e interoperar. Por lo antes expuesto, es necesario disponer de funciones para configurar y manejar los módulos PmodRF2. Existen algunas bibliotecas implementadas para manejar el circuito integrado principal de estos módulos (MRF24J40) pero carecen de otras funcionalidades específicas de PmodRF2 (Circuitar, 2015b, Palsson, 2011).

Este trabajo tiene como objetivo desarrollar e implementar una biblioteca de funciones en lenguaje C++, usando placas de desarrollo Arduino, para la configuración y control de módulos PmodRF2. La biblioteca permite a los usuarios desarrollar fácilmente proyectos y aplicaciones con placas de desarrollo Arduino y módulos PmodRF2, sin necesidad de dominar las operaciones con los registros de control de los módulos.

2. Metodología

El módulo PmodRF2 fue creado como interfaz para agregar comunicación inalámbrica de radio frecuencia (RF) a cualquier placa de sistema Digilent. Digilent Inc. (2016) describe todas las características de los módulos PmodRF2. Algunas de estas características son: voltaje de alimentación de 3.3 volts, conector de 12 pines denominado J1 por el fabricante, la principal vía de comunicación del módulo con el microcontrolador anfitrión es un bus de Interfaces Periféricas Series (SPI) (Leens, 2009) y el circuito integrado principal de estos módulos es el MRF24J40 (Microchip Technology Inc., 2010).

El circuito integrado MRF24J40 es fabricado por Microchip (Microchip Technology Inc.) cuya función es la de un transceptor de radio frecuencia compatible con el Estándar IEEE 802.15.4-2003. Este opera en la banda de 2.4 GHz y puede alcanzar una tasa de transmisión de 250 kbps o 625 kbps con el modo turbo. MRF24J40 es compatible con los protocolos ZigBee®, MiWi™, MiWi P2P (Flowers and Yang, 2007) y otros propios para el trabajo con redes inalámbricas. El chip permite crear elementos de una WPAN con bajo costo y bajo consumo. Además, puede comunicarse con una gran variedad de microcontroladores utilizando un bus SPI de cuatro líneas, un pin de interrupción, uno para despertar y uno para reestablecer el circuito integrado a su estado inicial (Microchip Technology Inc., 2010).

El Estándar IEEE 802.15.4 define el nivel físico (PHY) y el control de acceso al medio (MAC) en LR-WPAN. Estas redes de comunicaciones de bajo costo proveen conectividad inalámbrica en aplicaciones con alimentación limitada y requerimientos de rendimiento bajos. El objetivo principal de este tipo de redes es una fácil instalación, transferencia de datos segura, operación en rangos cortos, costos extremadamente bajos, y una razonable vida de la batería manteniendo un protocolo simple y flexible (IEEE Standards Association, 2015).

El circuito integrado MRF24J40 brinda soporte de hardware para varias de las funciones del estándar IEEE 802.15.4-2003 como lo son: detección de energía, detección de portadora, tres modos de CCA (*Clear Channel Assessment*), algoritmo de CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*), retransmisión automática de paquetes, envío automático de paquetes ACK (*Acknowledgment*), cuatro buffers independientes para la transmisión y un mecanismo de seguridad que puede

encriptar y desencriptar para la subcapa MAC y capas superiores. Estas funciones reducen la carga de procesamiento, permitiendo el uso de microcontroladores de 8 bits de bajo costo. Un conjunto de registros del circuito integrado son utilizados para la configuración de dichas funciones y otras que además posee (Microchip Technology Inc., 2010).

Es posible utilizar el PmodRF2 con una amplia gama de microcontroladores. El circuito integrado MRF24J40 y el módulo PmodRF2 tienen compatibilidad con las familias de microcontroladores PIC de Microchip. Dicha empresa provee dos bibliotecas de software denominadas "MiWi™ P2P/Star" y "MiWiMesh" (Yang, 2009a, Yang, 2009b) para la utilización de estos módulos con sus microcontroladores de 8-bit, 16-bit y 32-bit. Estas bibliotecas utilizan el protocolo MiWi™ propiedad de la misma empresa (Microchip Technology Inc., 2010).

Las librerías necesarias para incorporar MiWi™ a proyectos con microcontroladores son de código abierto, pero están limitadas a su utilización con microcontroladores de Microchip y su circuito integrado MRF24J40. Por otra parte, la incorporación de ZigBee® a un proyecto no está restringida, pero si lo está el acceso a su implementación. También es posible utilizar los PmodRF2 con placas de desarrollo fabricadas por Digilent o Xilinx. Las placas de desarrollo Arduino (Goilav and Geoffrey, 2016, McRoberts, 2013) cumplen con los requisitos antes planteados y son más económicas que las Digilent o Xilinx (Digilent Inc., Xilinx).

Existen una serie de bibliotecas implementadas para la utilización del circuito integrado MRF24J40 con las placas Arduino. La mayoría de estas bibliotecas están enfocadas a la comunicación a nivel MAC IEEE 802.15.4 y utilizan otras versiones de MRF24J40 (Circuitar, 2015a). El resto de las bibliotecas existentes (Circuitar, 2015b, Palsson, 2011), están enfocadas en el MRF24J40 pero poseen características que son suprimidas en los módulos PmodRF2 y carecen de otras funcionalidades como los modos de recepción, los filtros de tramas, modo de sueño, etc.

3. Resultados y discusión

La biblioteca lib_PmodRF2 se implementa con el objetivo de facilitar la utilización de los módulos PmodRF2 en aplicaciones con placas de desarrollo Arduino. Esta biblioteca logra abstraer a los usuarios de Arduino del sistema de registros de control que comandan el módulo. La biblioteca lib_PmodRF2 se utiliza como medio de

configuración y control de los módulos PmodRF2 desde placas Arduino y permite a los usuarios desarrollar proyectos de forma sencilla. La biblioteca lib_PmodRF2 implementa una clase de C++ (nombrada PmodRF2) siguiendo la estrategia de la programación orientada a objetos. La biblioteca lib_PmodRF2 está compuesta por dos archivos: *PmodRF2.h* (*header*, donde se realizan las declaraciones) y *PmodRF2.cpp* (*source*, donde se implementan las funciones).

El archivo *PmodRF2.h* posee definidos todos los registros del circuito integrado MRF24J40. Entender por "definir los registros" a la acción de asociar la dirección de memoria del registro con un nombre bien conocido. En las definiciones de estos registros se toma la nomenclatura utilizada por el fabricante y se utiliza el prefijo "REG_" como distinción de registro (ej. 'REG_RXMCR'). En el caso de los bits muy utilizados dentro de los registros, para el chequeo de estado y control, estos se definen con el prefijo "REG_BIT_" y el nombre utilizado por el fabricante (ej. 'REG_BIT_TXNSTAT') y los valores utilizados son las posiciones de estos dentro del registro (de 0 a 7).

En el archivo *PmodRF2.h* también contiene definiciones de las direcciones de inicio de los FIFO (*First In First Out*). Para esto se utilizan los nombres de los FIFO (sin espacios) y el sufijo "startOf" (ej. 'TXNFIFOstartOf'). Además, en este fichero se definen ocho máscaras de bits para identificar fácilmente, en el registro INTSTAT, la interrupción ocurrida. Para estas máscaras se utiliza el nombre de la interrupción y el prefijo "MSK_INTSTAT_" (ej. 'MSK_INTSTAT_RXIF').

Los formatos utilizados para tramas de salida y entrada son definidos como estructuras en el fichero *PmodRF2.h* y se muestran en la Figura 1a y 1b respectivamente. Las estructuras de tramas de salida incluyen los atributos *"sent"*, *"retries"* y *"channelBusy"* para representar el estado de la trama en el intento de transmisión. Por otra parte, las estructuras de tramas de entrada incluyen los atributos *"isValid"*, *"lqi"* y *"rssi"* para representar la validez de las mismas y los valores de LQI (*Link Quality Indication*) y RSSI (*Receiver Signal Strength Indicator*) obtenidos.

```
uint8_t     sent:1;
uint8_t     retries:2;
uint8_t     channelBusy:1;
```

a)

```
uint8_t     isValid:1;
uint8_t     lqi;
uint8_t     rssi;
```

b)

Figura 1. Atributos de estado de trama: a) trama de salida, b) trama de entrada.

Las tramas del estándar IEEE 802.15.4 poseen los atributos mostrados en la Figura 2a mientras que las tramas de *beacons* y datos poseen los atributos mostrados en las Figuras 2b y 2c respectivamente.

```
uint16_t    frameControl;
uint8_t     sequenceNumber;
uint16_t    PANId;
uint16_t    Address;
```

a)

```
uint16_t    superFrameSpecification;
uint8_t     GTSspecification;
uint8_t     pendingADDspecification;
uint8_t     beaconPayloadSize;
uint8_t     beaconPayload[112];
```

b)

```
uint8_t     dataPayloadSize;
uint8_t     dataPayload[116];
```

c)

Figura 2. Atributos de formato de trama: a) atributos generales de tramas IEEE 802.15.4, b) atributos de trama de *beacon*, c) atributos de trama datos.

Dentro de la clase PmodRF2 se declaran una trama de entrada y una de salida de datos, *"inDataFrame"* y *"outDataFrame"*, y una trama de entrada y una de salida *beacon*, *"inBeaconFrame"* y *"outBeaconFrame"*.

Además de las definiciones ya mencionadas el archivo *PmodRF2.h* contiene la definición de la clase PmodRF2. Esta definición consiste en la declaración de los atributos y funciones que poseerán los objetos de la clase PmodRF2. La Tabla 1 muestra una descripción de los atributos principales de la clase PmodRF2.

Tabla 1. Atributos de la clase PmodRF2.

Atributo	Descripción
PIN_ChipSelect	Guarda el número del pin utilizado en la placa Arduino para la señal ~CS del PmodRF2.
PIN_Reset	Guarda el número del pin utilizado en la placa Arduino para la señal $\overline{\text{RESET}}$del PmodRF2.
PIN_Wake	Guarda el número del pin utilizado en la placa Arduino para la señal WAKE del PmodRF2.
currentPANId	Guarda el identificador de la PAN a la que pertenece el dispositivo.
currentShortAddress	Guarda la dirección corta del dispositivo.
currentChannel	Guarda el canal en el cual está operando el dispositivo.
PAN_Coord	Identifica e dispositivo como un coordinador de red.

Coord	Identifica e dispositivo como un coordinador.
BSN	Guarda el número de secuencia de las tramas *beacon*.
DSN	Guarda el número de secuencia de las tramas de datos.
periodSlpClk	Guarda el período del reloj de sueño calculado con la calibración.
WAKETIME	Guarda el valor del registro WAKETIME (calculado con relación a periodSlpClk).

La clase PmodRF2 posee otra serie de atributos que son utilizados como banderas de interrupción y estado (ver Tabla 2). Además, en esta clase se definen un conjunto de atributos que funcionan como punteros para escribir o leer de los respectivos FIFO (ver Tabla 3).

Tabla 2: Banderas de interrupción y estado.

Atributo	Descripción
flag_slp	Bandera de interrupción SLPIF.
flag_wake	Bandera de interrupción WAKEIF.
flag_rx	Bandera de interrupción RXIF.
flag_txn	Bandera de interrupción TXNIF.
sleepingFlag	Bandera de estado del modo de sueño.

Tabla 3: Punteros de lectura y escritura en FIFO.

Atributo	Descripción
TXNFIFOptr	Puntero para la escritura en el TX Normal FIFO.
RXFIFOptr	Puntero para la lectura del RX FIFO.

La función principal de la clase es el constructor, función que se utiliza para definir un objeto de esta clase. El constructor se nombra igual que la clase y recibe como parámetros los números de los pines que utilizará la placa Arduino para las señales de ~CS, RESET y WAKE, como se observa en la Figura 3.

```
PmodRF2(int p_ChipSelect, int p_Reset, int p_Wake);
```

Figura 3. Definición del constructor de la clase PmodRF2.

Todas las operaciones de la biblioteca están basadas en la lectura y escritura de registros del MRF24J40. Debido a esto las funciones base de la biblioteca lib_PmodRF2 son las

mostradas en la Tabla 4 y sus definiciones se muestran en la Figura 4. Estas funciones de lectura y escritura utilizan la comunicación SPI entre la placa Arduino y el PmodRF2. Por esto es necesario incluir al inicio del archivo *PmodRF2.h* la directiva *#include "SPI.h"* que posibilitará la utilización de la biblioteca SPI existente para la plataforma Arduino.

Tabla 4. Funciones de lectura y escritura de registros.

Función	Descripción
sacrRead	Lectura de registro de dirección corta.
sacrWrite	Escritura de registro de dirección corta.
lacrRead	Lectura de registro de dirección larga.
lacrWrite	Escritura de registro de dirección larga.

```
/// Short Address Control Registers
void sacrWrite(uint8_t address, uint8_t value);
uint8_t sacrRead (uint8_t address);

/// Long Address Control Registers
void lacrWrite(uint16_t address, uint8_t value);
uint8_t lacrRead (uint16_t address);
```

Figura 4. Definiciones de funciones de lectura y escritura de registros.

La clase PmodRF2 tiene implementados el restablecimiento por *hardware*, el restablecimiento de la máquina de estado RF y el restablecimiento por software (completo e individual). Las definiciones de cada uno de estos restablecimientos son mostradas en la Figura 5.

```
void hardwareReset ();          void powerMgrReset ();
void stateMachineReset ();      void basebandReset ();
void fullSoftReset ();          void macReset ();
```

Figura 5. Definiciones de funciones de restablecimiento.

La función *initializingModuleRF* se encarga de inicializar el módulo PmodRF2. Esta es la función que debe encabezar cualquier aplicación que utilice la biblioteca lib_PmodRF2.

Atendiendo a lo relacionado con los modos de recepción y los filtros por tipo de trama se implementan en la clase PmodRF2 funciones que establezcan estos modos y filtros (ver Figuras 6 y 7).

```
void setNormalMode ();
void setErrorMode ();
void setPromisMode ();
```

Figura 6. Definición de funciones de establecimiento de modos de recepción.

```
void setNoFrameFilter ();
void setCommandOnlyFilter ();
void setDataOnlyFilter ();
void setBeaconOnlyFilter ();
```

Figura 7. Definición de funciones de establecimiento de filtros de tramas.

Además, la clase PmodRF2 posee funciones para establecer y consultar el identificador de la PAN a la cual está asociado el dispositivo, así como la dirección corta que lo identifica dentro de la misma, Las definiciones de estas funciones se muestran en la Figura 8.

```
/// PANId Operations
void setPANId (uint16_t panid);
uint16_t getPANId ();

/// Short Address Operations
void setShortAddress (uint16_t shortaddress);
uint16_t getShortAddress ();
```

Figura 8. Definición de funciones de establecimiento y consulta de identificador de PAN y de dispositivo.

Los objetos pertenecientes a la clase PmodRF2 también pueden establecer y consultar el canal de operación utilizando las funciones mostradas en la Figura 9.

```
/// Channel Operations
void setChannel (uint8_t channel);
uint8_t getChannel ();
```

Figura 9. Definición de funciones de establecimiento y consulta de canal de operación.

La función *initializingNBe* se encarga de inicializar los parámetros pertinentes a una red NonBeacon-Enabled en un dispositivo, atendiendo a su función dentro de la misma, según los atributos *"Coord"* y *"PAN_Coord"*. En el caso que un objeto de la clase PmodRF2 posea ambos atributos *"Coord"* y *"PAN_Coord"* en '0' significaría que se desempeñará como dispositivo.

En la clase PmodRF2 se implementa una función llamada *loadBeaconFrameIntoTXNFIFO***,** para cargar la trama *"outBeaconFrame"* dentro del TX Normal FIFO (ver Figura 10). Con esta función es posible transmitir una trama *beacon* utilizando el buffer para transmisión normal de datos. En las ocasiones en que se utilice este método para transmitir la trama *beacon*, se debe asegurar que el dispositivo al cual se envíe esta trama utilice el modo de recepción Error o Promiscuo. Al transmitir

la trama *beacon* desde el TX Normal FIFO el campo FCS añadido al paquete será incorrecto y este es reconocido en el dispositivo receptor como un paquete con errores.

Además, se implementa una función *loadDataFrameIntoTXxxFIFO* para cargar la trama *"outDataFrame"* a partir de la dirección que se pase como parámetro, esto posibilita la selección del FIFO por el cual enviar la trama. También, se implementa la función *loadPacketFromRXFIFO* que se encarga de leer la trama recibida del RX FIFO y almacenarla según corresponda en *"inDataFrame"* o *"inBeaconFrame"* dependiendo del tipo de trama recibida. Esta última función establecerá a '1' el atributo *"isValid"* de la trama recibida y una vez procesada esta se recomienda restablecer el valor a '0' para expresar que la trama ya fue leída. La declaración de estas dos últimas funciones enconadas, es mostrada en la Figura 10.

```
/// Frames Operations
void loadBeaconFrameIntoTXNFIFO ();
void loadDataFrameIntoTXxxFIFO (uint16_t TXxxFIFOptr);
void loadPacketFromRXFIFO ();
```

Figura 10. Definiciones de funciones para operaciones con tramas.

Las funciones mostradas en la Figura 11 son la encargada de habilitar las fuentes de interrupción del PmodRF2 y la encargada de realizar los procedimientos necesarios y habilitar las banderas correspondientes cuando ocurre una interrupción.

```
/// Interrupts Operations
void enableInterrupts ();
void interruptHandler ();
```

Figura 11. Definiciones de funciones para operaciones con interrupciones.

Las funciones mostradas en la Figura 12 son las encargadas de habilitar y deshabilitar la recepción. Estas operaciones son necesarias para el procedimiento de lectura de una trama del RX FIFO.

```
/// RX Circuit Operations
void enableRX();
void disableRX();
```

Figura 12: Definiciones de funciones para operaciones con el sistema de circuitos de recepción.

El incremento y la obtención de los números de secuencia de las tramas de datos y de *beacons* se realizan a través de las funciones mostradas en la Figura 13. Mientras que la operación de enviar la trama almacenada en el TX Normal FIFO se ordena a través de la función *sendN*, su definición se observa en la Figura 14.

```
/// Sequence Number Operations
void increaseDSN ();
void increaseBSN ();
uint8_t getDSN ();
uint8_t getBSN ();
```

Figura 13. Definiciones de funciones para operaciones con los números de secuencias.

```
/// TX Operations
void sendN();
```

Figura 14. Definición de función para transmitir (TX Normal FIFO).

En lo concerniente al modo de sueño se definen las funciones mostradas en la Figura 15, las cuales se encargarán de: habilitar la salida inmediata del modo de sueño, salir del modo de sueño utilizando la señal WAKE, salir del modo de sueño utilizando el registro WAKECON, entrar inmediatamente en modo de sueño, calibrar el reloj de sueño, establecer el período de sueño, entrar en modo de sueño temporizado (con el período previamente establecido).

```
/// Sleep and Wake
void enableImmediateWake ();
void immediateWakePin ();
void immediateWakeReg ();
void immediateSleep ();
void calibrateSleepClock ();
void setSleepPeriod (uint32_t sleepPeriod);
void startSleepPeriod ();
```

Figura 15. Definiciones de funciones para entrada y salida del modo de sueño.

La clase PmodRF2 también posee funciones para el cálculo de RSSI. Estas funciones se encargarán de establecer el número de símbolos a promediar para calcular el RSSI y además a establecer el modo por el cual se calcula. La definición de dichas funciones se muestra en la Figura 16.

```
/// RSSI Operations
void setRSSIavgSymbls (uint8_t RSSINUM);
uint8_t rssiFirmwareRequest ();
void setRSSImode2 ();
void clrRSSImode2 ();
```

Figura 16. Definiciones de funciones para el cálculo de RSSI.

La función *setTurboMode* establece el modo turbo, el cual permite alcanzar razones de transmisión de 625 kbps. Su definición se observa a continuación en la Figura 17.

```
/// TURBO Mode
void setTurboMode ();
```

Figura 17. Definición de función para establecer modo turbo.

Finalmente la clase PmodRF2 provee una función para inicializar un temporizador que posee el módulo PmodRF2 dentro de su sistema de circuitos MAC (ver Figura 18). Este

temporizador de 16 bits cuenta períodos de 8 micro segundos disminuyendo el valor cargado en él inicialmente. Una vez el temporizador alcanza el valor cero se activa la interrupción HSYMTMRIF.

```
/// MAC Timer
void setMACtimer (uint16_t macTIME); // Period = 8 us
```

Figura 18. Definición de función para activar el temporizador MAC.

La implementación de una aplicación con los módulos PmodRF2 utilizando como microcontrolador anfitrión una placa Arduino es un proceso relativamente sencillo. Se debe comenzar por incluir la biblioteca lib_PmodRF2 en el *sketch*. A continuación, se debe declarar el objeto de la clase PmodRF2. En la función *setup* del *sketch* se debe definir el pin de interrupción de la placa Arduino como una entrada y establecer la interrupción por flanco de caída. En la misma función del *sketch* se debe inicializar la comunicación SPI y ejecutar las funciones *hardwareReset* e *initializingModuleRF* de forma consecutiva, para restablecer los registros de control del PmodRF2 y establecer algunos a sus valores recomendados. A partir de este punto se puede proceder a establecer los parámetros de red del dispositivo y los atributos que definen su funcionamiento (*"Coord" y "PAN_Coord"*). Luego se llama a la función *initializingNBe* para establecer el dispositivo en una red NonBeacon-Enabled. A partir de este momento se puede proceder a modificar las tramas y configuraciones del módulo así como utilizar las funciones para ejecutar operaciones atendiendo a las necesidades de la aplicación y el proyecto a implementar. La descripción de este proceso se muestra en el esquema de la Figura 19.

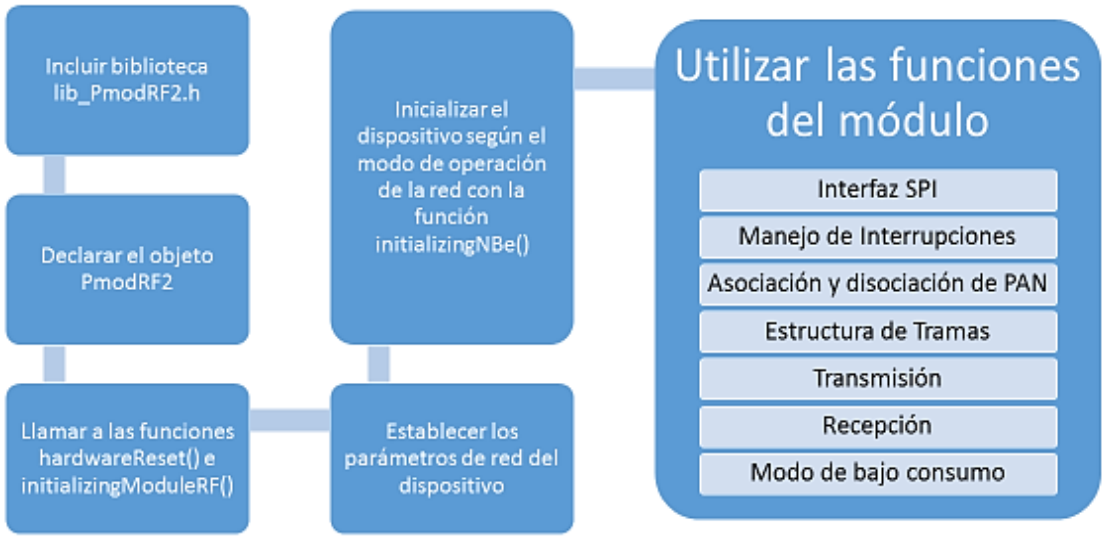
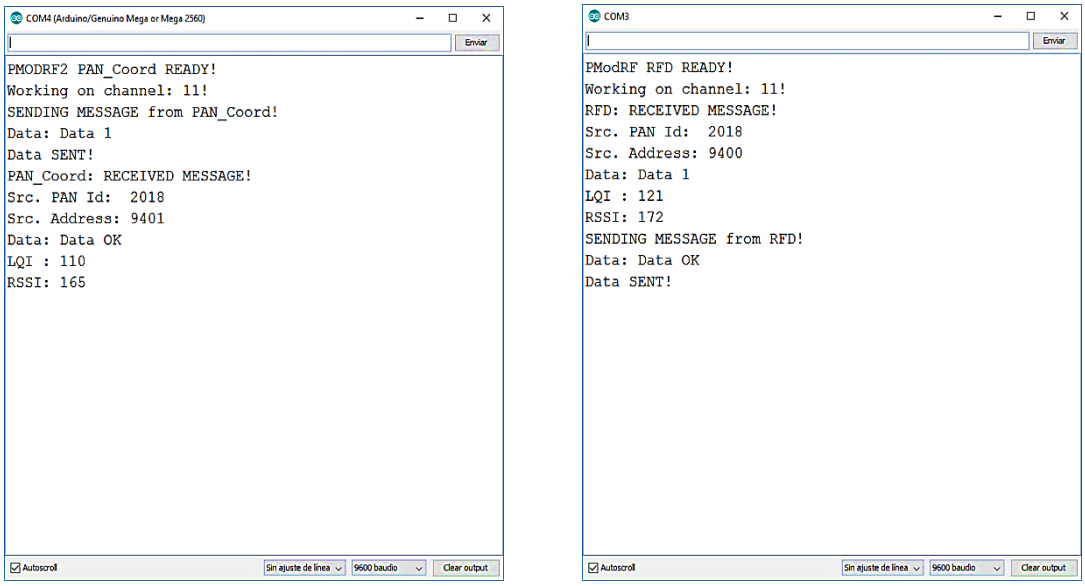


Figura 19. Proceso para la implementación de una aplicación utilizando la biblioteca lib_PmodRF2.

La Figura 20 muestra el resultado de ejecutar una aplicación donde se intercambian datos entre dos placas de desarrollo Arduino usando módulos PmodRF2. El Arduino Mega 2560 envía el dato "Data 1" (ver Figura 20a) el cual es recibido correctamente por el Arduino UNO. Mientras que el Arduino UNO envía el dato "Data OK" (ver Figura 20b) e igualmente es recibido sin errores por el Arduino Mega 2560.



a) b)

Figura 20. Muestra de la corrida de una aplicación de intercambio de datos: a) Arduino Mega 2560 (PAN Coord.), b) Arduino UNO (RFD).

4. Conclusiones

Los registros de control del circuito integrado MRF24J40 son los encargados de modificar y controlar la actuación del módulo PmodRF2, siempre basado en el Estándar IEEE 802.15.4-2003. Es posible utilizar los módulos PmodRF2 conectados a las placas Arduino, mediante el protocolo SPI. Con el desarrollo de la biblioteca en C++, es posible realizar la configuración y establecer una comunicación entre los dispositivos. La implementación de aplicaciones que utilicen módulos PmodRF2 para comunicarse se transforma en una tarea sencilla al utilizar la biblioteca lib_PmodRF2. Esta biblioteca facilita el uso de las funciones del módulo abstrayendo al programador del manejo de los registros de control. Además, en la biblioteca están implementadas funciones que permiten, de una manera simple, utilizar funcionalidades complejas del PmodRF2. Esto comprueba los beneficios que posee la biblioteca lib_PmodRF2 para proyectos futuros.

5. Referencias bibliográficas

CIRCUITAR. 2015a. *Nanoshield_MRF24J40: Arduino library to use with the MRF24J40 Nanoshield (MRF24J40 A/B/C/D from Microchip)* [Online]. Available: https://github.com/circuitar/Nanoshield_MRF24J40.

CIRCUITAR. 2015b. *Nanoshield_MRF: Arduino library to use the MRF24J40 802.15.4 2.4GHz wireless radio from Microchip* [Online]. Available: https://github.com/circuitar/Nanoshield_MRF.

DARGIE, W. & POELLABAUER, C. 2010. *Fundamentals of wireless sensor networks: theory and practice*, John Wiley & Sons.

DIGILENT INC. *Digilent: Electrical Engineering Store, FPGA, Microcontrollers and Instrumentation* [Online]. Available: https://store.digilentinc.com/.

DIGILENT INC. 2016. PmodRF2™ Reference Manual.

FLOWERS, D. & YANG, Y. 2007. MiWi™ Wireless Networking Protocol Stack. *Microchip Technology Inc. application note*.

GOILAV, N. & GEOFFREY, L. 2016. *Arduino: Aprender a desarrollar para crear objetos inteligentes*, Ediciones ENI.

IEEE STANDARDS ASSOCIATION 2015. IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs). *IEEE Computers Society*.

LAN/MAN STANDARDS COMMITTEE 2003. Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). *IEEE Computer Society*.

LEENS, F. 2009. An introduction to I 2 C and SPI protocols. *IEEE Instrumentation & Measurement Magazine*, 12**,** 8-13.

MCROBERTS, M. 2013. *Beginning Arduino*, Apress.

MICROCHIP TECHNOLOGY INC. *Microchip Technology: Home* [Online]. Available: https://www.microchip.com/.

MICROCHIP TECHNOLOGY INC. 2010. MRF24J40 DataSheet.

PALSSON, K. 2011. *Mrf24j40-arduino-library: arduino driver for the mrf24j40 802.15.4 modules* [Online]. Available: https://github.com/karlp/Mrf24j40-arduino-library.

XILINX. *Xilinx - Adaptable. Intelligent.* [Online]. Available: https://www.xilinx.com/.

YANG, Y. 2009a. Microchip Wireless (MiWi™) Application Programming Interface–MiApp. *Microchip Inc.[online] Available: http://www.microchip.com/downloads/en/AppNotes*.

YANG, Y. 2009b. Microchip Wireless (MiWi™) Media Access Controller–MiMAC.