



ANEXO B

Instalación y uso del driver LoRa para sistemas embebidos.

JOSÉ DANIEL RODRÍGUEZ MUNCA

Tutor Académico: Manuel Sierra Castañer

Tutor Profesional: Álvaro Gutiérrez Martín

**UNIVERSIDAD DE POLITÉCNICA DE MADRID
UNIVERSIDAD COMPLUTENSE DE MADRID
MASTER INTERUNIVERSITARIO EN ESTRATEGIAS Y TECNOLOGÍAS PARA
EL DESARROLLO
MADRID - ESPAÑA
2016**

TABLA DE CONTENIDO

INSTALACIÓN Y USO DEL DRIVER LORA PARA SISTEMAS EMBEBIDOS ARDUINO.....	5
Estructura de archivos que componen el driver	5
Instalación de la librería para el uso con Arduino	7
Uso de librería para el control de dispositivos LoRa en aplicaciones o sketch de Arduino ...	7
Ejemplo de aplicación: red punto a punto (aplicación ping-pong)	12
Ejemplo de aplicación: recepción de datos de varios dispositivos remotos (red punto - multipunto).....	14
INSTALACIÓN Y USO DEL DRIVER LORA PARA SISTEMAS EMBEBIDOS SDIN ...	17
Estructura de archivos que componen el driver	17
Uso de librería para el control de dispositivos LoRa en aplicaciones para SDIN	20
Ejemplo de aplicación: red punto a punto (aplicación ping-pong) con módulos SDIN	22
Ejemplo de aplicación: recepción de datos de varios dispositivos remotos (red punto - multipunto) con módulos SDIN	24
BIBLIOGRAFÍA	27

LISTADO DE FIGURAS

<i>Figura 1. Estructura de archivos y carpetas del driver propuesto para el manejo de dispositivos LoRa utilizando Arduino.</i>	<i>5</i>
<i>Figura 2. Gestión entre archivos del driver propuesto para el manejo de dispositivos LoRa. .</i>	<i>6</i>
<i>Figura 3. Base de aplicación o “sketch” para utilizar transceiver LoRa con IDE Arduino. ..</i>	<i>11</i>
<i>Figura 4. Principio de funcionamiento de aplicación “ping-pong”.....</i>	<i>13</i>
<i>Figura 5. Esquema de conexión para LoRa1278 de NiceRF y Arduino UNO.....</i>	<i>13</i>
<i>Figura 6. Tipología estrella para comunicación con diversos dispositivos Arduino.</i>	<i>15</i>
<i>Figura 7. Estructura de archivos y carpetas del driver propuesto para el manejo de dispositivos LoRa utilizando FreeRTOS –SDIN.....</i>	<i>17</i>
<i>Figura 8. Gestión entre archivos del driver propuesto para el manejo de dispositivos LoRa para los módulos SDIN.</i>	<i>19</i>
<i>Figura 9. Principio de funcionamiento de aplicación “ping-pong” con SDIN.</i>	<i>23</i>
<i>Figura 10. Esquema de conexión para LoRa1278 de NiceRF y módulo SDIN.</i>	<i>23</i>
<i>Figura 11. Tipología estrella para comunicación con diversos dispositivos.</i>	<i>25</i>

LISTADO DE TABLAS

<i>Tabla 1. Instrucción para selección de transceiver LoRa en archivos radio.h.....</i>	<i>7</i>
<i>Tabla 2. Banda de frecuencia ISM compatibles con los transceivers de radio LoRa.....</i>	<i>9</i>
<i>Tabla 3. Combinación de los principales parámetros configurables para los LoRa</i>	<i>10</i>
<i>Tabla 4. Comportamiento de los registros tipo flag o banderas de los estados finales de transmisión o recepción de datos para los LoRa.</i>	<i>12</i>
<i>Tabla 5. Nomenclatura utilizada para la conexión entre la tarjeta LoRa1278 de NiceRF y la tarjeta Arduino UNO.....</i>	<i>14</i>
<i>Tabla 6. Nomenclatura utilizada para la conexión entre la tarjeta LoRa1278 de NiceRF y el módulo SDIN</i>	<i>24</i>

INSTALACIÓN Y USO DEL DRIVER LORA PARA SISTEMAS EMBEBIDOS ARDUINO

Estructura de archivos que componen el driver

El driver está compuesto por una serie de archivos con *código fuente* y archivos tipo *header* organizados en dos subcarpetas. La estructura organizada de archivos se presenta en la *Figura 1*.

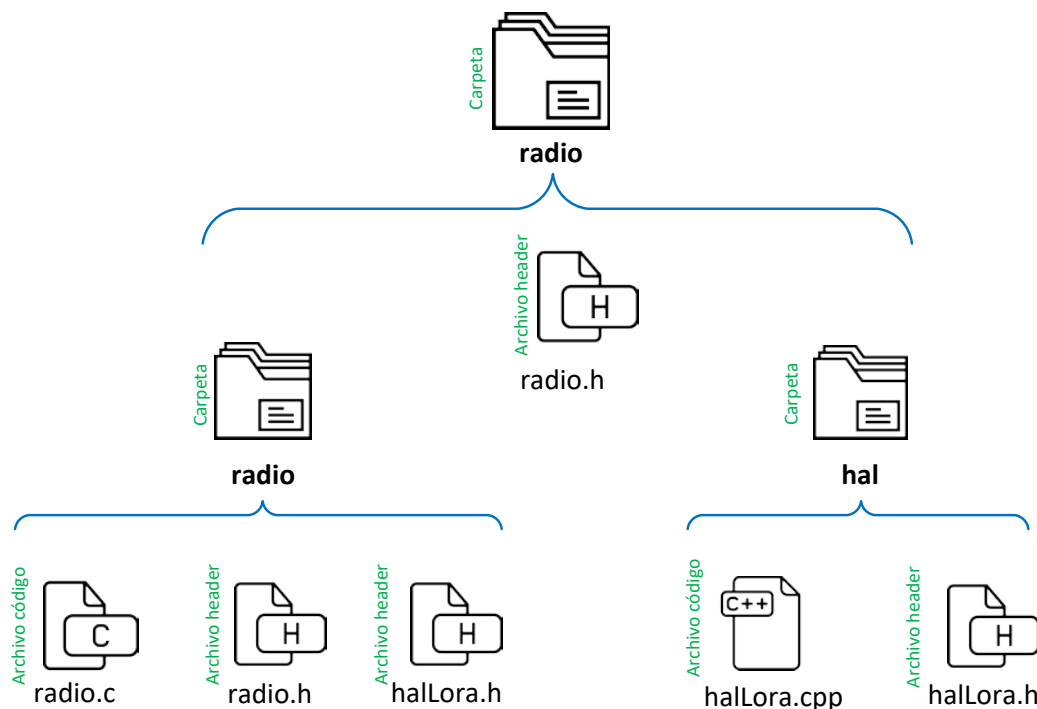


Figura 1. Estructura de archivos y carpetas del driver propuesto para el manejo de dispositivos LoRa utilizando Arduino.

Fuente: Esquema propuesto por el autor

La carpeta principal se denomina **radio** y está compuesta por los elementos:

- **Carpeta radio:** contiene los archivos necesarios para el funcionamiento del driver en aplicaciones o “*sketch*” para Arduino.
 - **radio.h:** el *header file* contiene la definición o selección del transceiver LoRa utilizado por la aplicación de Arduino. Es suficiente con definir:

Instrucción		Comentario
#define CFG_1272_SEMTECH	1	Selecciona el CHIP SX1272 de SEMTECH
#define CFG_1278_DRF1278F	1	Selecciona el CHIP SX1278 de DORJI versión de tarjeta DRF1278F
#define CFG_1278_NICERF1278	1	Selecciona el CHIP SX1278 de NiceRF versión de tarjeta LORA1278

- **Carpeta radio:** la carpeta contiene el código fuente, junto la definición de variables y funciones necesarias para la configuración de los dispositivos LoRa.
 - **radio.h:** contiene la definición de variables y funciones para la configuración de los LoRa a nivel de software relacionado con el código presentado en **radio.c**.
 - **radio.c:** contiene las funciones para la configuración de los LoRa a nivel de software.
 - **halLora.h:** cuenta con la definición de funciones del archivo **halLora.cpp** (*HAL o hardware abstraction layer*).
- **Carpeta hal:** la carpeta contiene el código fuente relacionado con la capa de más bajo nivel que configura los periféricos internos de Arduino.
 - **halLora.h:** cuenta con la definición de la nomenclatura asignada a los pines o terminales de control de los dispositivos LoRa.
 - **halLora.cpp:** contiene el código fuente de la capa de más bajo nivel que configura los periféricos internos de arduino (interrupciones por cambio de flanco, comunicación SPI, comunicación UART, cambios de estados en terminales). A esta capa se le conoce como *HAL* o capa de abstracción de hardware.

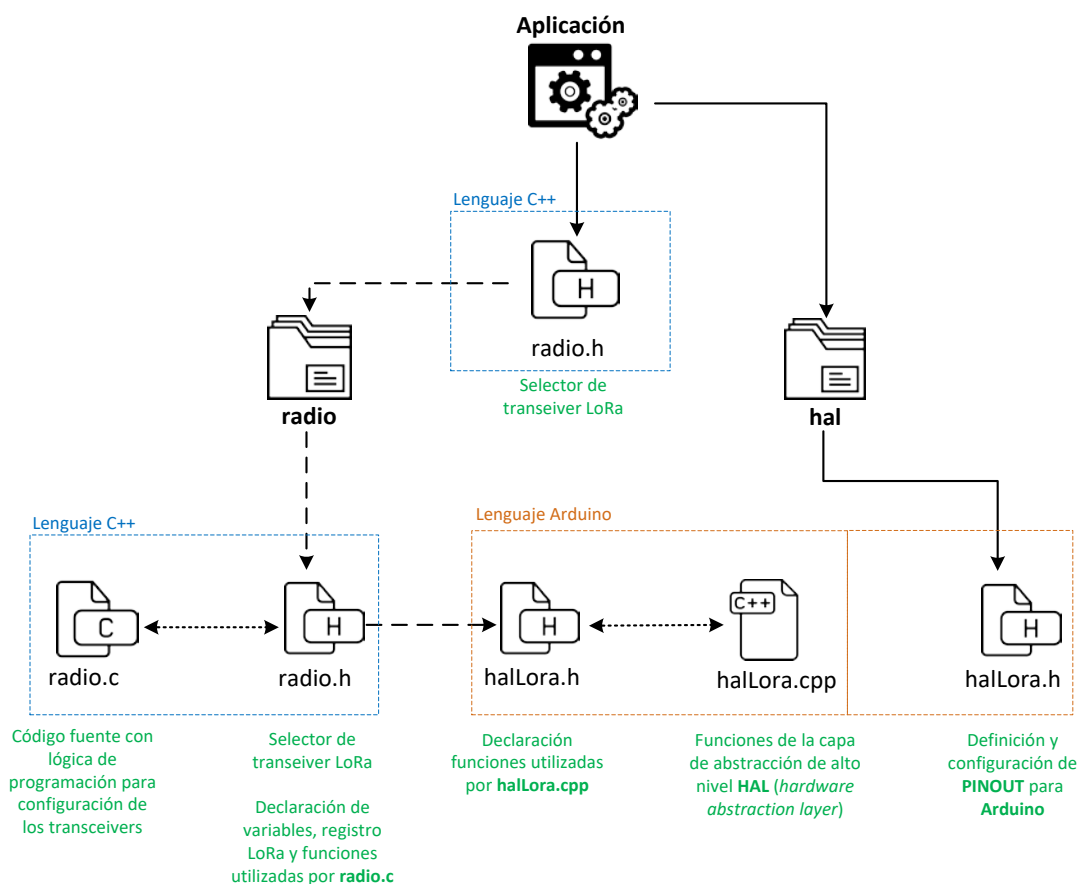


Figura 2. Gestión entre archivos del driver propuesto para el manejo de dispositivos LoRa.

Fuente: Esquema propuesto por el autor

La *Figura 2* presenta la gestión entre los diferentes archivos que componen el driver planteado.

Resulta estratégica la organización de los diferentes archivos debido a que el driver busca que el código presentado en *radio.c* se comporte como una capa de alto nivel, manteniendo el código invariante al uso de diferentes sistemas embebidos como Arduino, módulos SDIN o tarjetas de STMicroelectronics; mientras que el código presentado en *halLora.cpp* corresponde a una capa de alto bajo nivel de abstracción de hardware que de acuerdo al tipo de sistema embebido utilizado pueda ser ajustado para realizar la correcta homologación al microcontrolador utilizado.

Instalación de la librería para el uso con Arduino

La instalación del driver para el control de los dispositivos LoRa utilizando Arduino (se recomienda que el equipo de cómputo tenga instalado el IDE Arduino), se realiza copiando los archivos de la carpeta principal **radio** en la carpeta *libraries* de la instalación del *IDE arduino*:

<i>Ruta por defecto para Linux (ubuntu)</i>	<i>Ruta por defecto para Windows</i>
/usr/share/arduino/libraries	C:\Program Files (x86)\Arduino\libraries
	C:\Program Files\Arduino\libraries

Una vez copiado los archivos en la carpeta *libraries* de la *instalación del IDE arduino*, el usuario debe seleccionar el dispositivo LoRa que será utilizado por Arduino. Para seleccionar el transceiver, se debe habilitar una de las siguientes instrucciones en el archivo **radio/radio.h**:

Tabla 1. Instrucción para selección de transceiver LoRa en archivos radio.h.

Fuente: Planteamiento del autor

Instrucción			Selección de transeiver LoRa
#define	CFG_1272_SEMTECH	1	SX1272 de SEMTECH
#define	CFG_1278_DRF1278F	1	SX1278 de DORJI versión de tarjeta DRF1278F
#define	CFG_1278_NICERF1278	1	SX1278 de NiceRF versión de tarjeta LORA1278

Uso de librería para el control de dispositivos LoRa en aplicaciones o sketch de Arduino

Una vez copiado los archivos del driver en la carpeta donde se ha instalado el IDE Arduino, el usuario debe crear una aplicación o sketch y seguir los siguientes pasos:

- Incluir la librerías para control por software (**radio**) y hardware (**halLora**):

```
#include <radio.h>           //Librería de alto nivel para configuración por software de los radio LoRa
#include <hal/halLora.h>      //Librería de bajo nivel para configuración de hardware radio LoRa (HAL)
#include <SPI.h>              //Librería para manejo de comunicación SPI
```

- Definir los pines de control de la tarjeta Arduino con respecto a los transceiver LoRa:

```
//Definición de los pines de control de la tarjeta Arduino
const radio_pinmap radio_pins = {
  .nss =      [número de pin],
  .rxtx = RADIO_UNUSED_PIN,
  .rst =      [número de pin],
  .dio0 =     [número de pin],

  #if (defined CFG_1278_DRF1278F) || (defined CFG_1272_SEMTECH)  //Selección la versión del SX1278
    .dio3 =     [número de pin],
  #endif
  #if defined(CFG_1278_NICERF1278)                               //Selecciona el CHIP SX1278 de NICERF
    .tx_en =    [número de pin],
    .rx_en =    [número de pin],
  #endif
};
```

- Configurar los dispositivos LoRa de acuerdo a las necesidades del usuario. Esto se debe incorporar en la función propia de Arduino “*void setup()*”:

- Iniciar las variables tipo *flag* o *banderas*:

```
//Inicialización de variables del driver
```

```
RADIO.flagTx = 0;
```

```
RADIO.flagRx = 0;
```

```
RADIO.crc = 0;
```

- Definir la frecuencia de trabajo del transceiver:

```
RADIO.freq = [frecuencia de trabajo]; //Frecuencia de la banda ISM compatible con LoRa
```

Teniendo en cuenta que existen bandas de frecuencias de uso privativo (telefonía, internet, radio, entre otras), existen frecuencias de libre uso conocidas como ISM (Industrial, Scientific and Medical) que son frecuencias intencionalmente libres para uso no comercial en aplicaciones de tipo industrial, científico y médico. Se recomienda al usuario verificar la compatibilidad de la frecuencia de trabajo con la *Tabla 2* (bandas de uso ISM).

Tabla 2. Banda de frecuencia ISM compatibles con los transceivers de radio LoRa

Fuente: Texas Instruments (2005)

Banda ISM (MHz)	Frecuencia mínima (MHz)	Frecuencia máxima (MHz)	Ancho de Banda (MHz)	Usos por continente
433	433.050	434.790	1.84	Europa, África y parte del norte de Asia.
869	868	870	2	Europa, África, Asia y Oceanía.
900	902	928	26	Continente americano

- Definir la potencia de transmisión del transceiver:

//Configuración de potencia

RADIO.txpow = [Máxima potencia (dBm)] //Máxima TX potencia

RADIO.imax = [Corriente máxima (mA)] //Por defecto 100mA 45mA <=Imax <= 240mA

La potencia máxima se expresa en dBm y el valor definido debe estar entre 2 a 17.

La corriente máxima está dada en mA y el valor definido se debe encontrar entre 45mA y 240mA. El valor típico de corriente es de 100mA. Para valores significativos de corrientes (mayores a 125mA), se recomienda verificar la corriente máxima de suministro soportada por la fuente o batería de alimentación. Algunos sistemas embebidos como Arduino cuentan con reguladores con corriente máxima de suministro hasta de 120mA, por lo que al realizar una inadecuada configuración en el LoRa, podría destruirse el regulador y dañar la placa del sistema embebido.

- Definir las variables de configuración de la comunicación entre los transceivers LoRa:

//Variables de configuración de la comunicación LoRa

RADIO.sf = [Spread factor SF] //Configuración del Spread factor

RADIO.bw = [Ancho del canal BW] //Configuración del ancho de canal

RADIO.cr = [Coding rate CR] //Configuración de Coding rate

Los dispositivos LoRa cuentan con los parámetros *SF*, *BW* y *CR* que según las combinaciones entre estos, permite determinar el alcance o distancia máxima de recepción de datos, la velocidad de los datos y sobrecarga por corrección o detección de errores en los datos. Estos valores deben ser los mismos en todos los dispositivos radio LoRa de la red inalámbrica.

Los parámetros independientes de configuración se definen como (Semtech SX1272, 2015 & Semtech SX1278, 2015):

BW: Ancho de banda (entre más bajo sea este valor, el tiempo de la transmisión será mayor)

SF rate: Es el factor de alcance expresado como logaritmo de base 2. Cuanto mayor sea este valor, mejor rendimiento tendrá la transmisión de datos.

CR: Tasa de codificación de errores: Entre mayor sea este valor, mayor será la fiabilidad de los datos, pero con una sobrecarga en el tiempo de transmisión.

La *Tabla 3* presenta la combinación de los principales variables independientes para la configuración de la comunicación entre transceivers LoRa. Cabe aclarar que estos valores deben ser los mismos en todos los dispositivos radio LoRa de la red inalámbrica.

Tabla 3. Combinación de los principales parámetros configurables para los LoRa
Fuente: Semtech SX1278 (2015) & Semtech SX1272 (2015)

Ancho de banda BW (kHz)		Factor de alcance SF		Tasa de codificación de errores. CR	
Hardware	Software RADIO.BW	Hardware	Software RADIO.sf	Hardware	Software RADIO.cr
500	BW500	6 → 64 chips / symbol (modo utilizado para FSK)	SF_6	4/5	CR_4_5
250	BW250	7 → 128 chips / symbol	SF_7	4/6	CR_4_6
125	BW125	8 → 256 chips / symbol	SF_8	4/7	CR_4_7
62.5*	BW62_5*	9 → 512 chips / symbol	SF_9	4/8	CR_4_8
41.7*	BW41_7*	10 → 1024 chips / symbol	SF_10		
31.25*	BW31_25*	11 → 2048 chips / symbol	SF_11		
20.8*	BW20_8*	12 → 4096 chips / symbol	SF_12		
15.6*	BW15_6*				
10.4*	BW10_4*				
7.8*	BW7_8*				

*: Aplica solamente para los transceiver SX1278

- Configurar los pines I/O y la comunicación SPI:

hal_init() //Configuración de puertos de Arduino

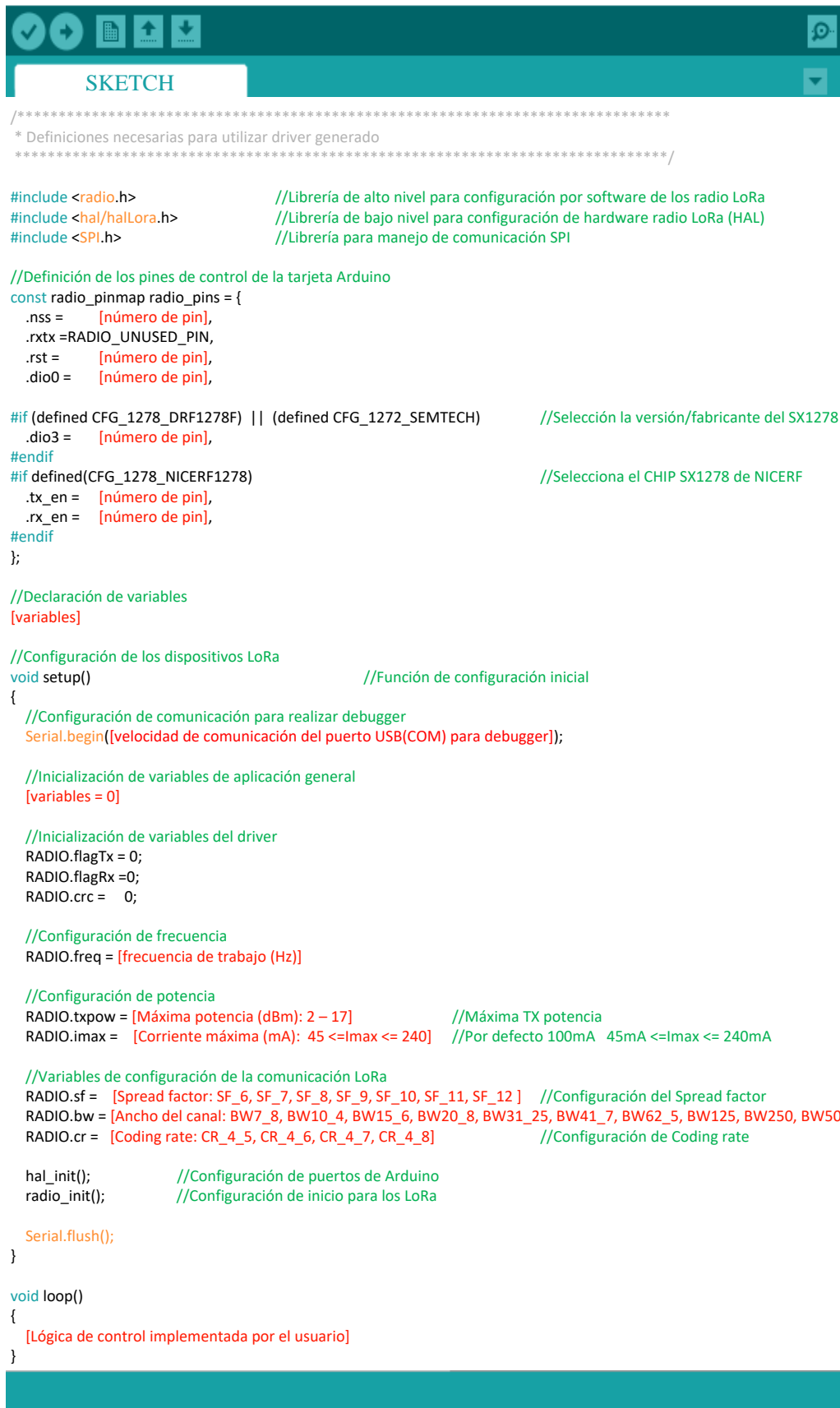
- Configurar los pines I/O y la comunicación SPI:

radio_init(); //Configuración de inicio para los LoRa

Corresponde a la función general que configura todos los parámetros de inicio del transceiver LoRa.

- Crear la aplicación con la lógica propuesta por el desarrollador:

```
void loop()
{
    [Lógica de control implementada por el usuario]
}
```



```

SKETCH

/*****
* Definiciones necesarias para utilizar driver generado
*****/

#include <radio.h>           //Librería de alto nivel para configuración por software de los radio LoRa
#include <hal/halLora.h>     //Librería de bajo nivel para configuración de hardware radio LoRa (HAL)
#include <SPI.h>             //Librería para manejo de comunicación SPI

//Definición de los pines de control de la tarjeta Arduino
const radio_pinmap radio_pins = {
  .nss =    [número de pin],
  .rxtx = RADIO_UNUSED_PIN,
  .rst =    [número de pin],
  .dio0 =   [número de pin],

  #if (defined CFG_1278_DRF1278F) || (defined CFG_1272_SEMTECH)           //Selección la versión/fabricante del SX1278
    .dio3 =   [número de pin],
  #endif
  #if defined(CFG_1278_NICERF1278)                                       //Selecciona el CHIP SX1278 de NICERF
    .tx_en =  [número de pin],
    .rx_en =  [número de pin],
  #endif
};

//Declaración de variables
[variables]

//Configuración de los dispositivos LoRa
void setup()                   //Función de configuración inicial
{
  //Configuración de comunicación para realizar debugger
  Serial.begin([velocidad de comunicación del puerto USB(COM) para debugger]);

  //Inicialización de variables de aplicación general
  [variables = 0]

  //Inicialización de variables del driver
  RADIO.flagTx = 0;
  RADIO.flagRx = 0;
  RADIO.crc = 0;

  //Configuración de frecuencia
  RADIO.freq = [frecuencia de trabajo (Hz)]

  //Configuración de potencia
  RADIO.txpow = [Máxima potencia (dBm): 2 – 17]           //Máxima TX potencia
  RADIO.imax =  [Corriente máxima (mA): 45 <=Imax <= 240] //Por defecto 100mA  45mA <=Imax <= 240mA

  //Variables de configuración de la comunicación LoRa
  RADIO.sf =  [Spread factor: SF_6, SF_7, SF_8, SF_9, SF_10, SF_11, SF_12] //Configuración del Spread factor
  RADIO.bw =  [Ancho del canal: BW7_8, BW10_4, BW15_6, BW20_8, BW31_25, BW41_7, BW62_5, BW125, BW250, BW500]
  RADIO.cr =  [Coding rate: CR_4_5, CR_4_6, CR_4_7, CR_4_8]           //Configuración de Coding rate

  hal_init();           //Configuración de puertos de Arduino
  radio_init();         //Configuración de inicio para los LoRa

  Serial.flush();
}

void loop()
{
  [Lógica de control implementada por el usuario]
}

```

Figura 3. Base de aplicación o “sketch” para utilizar transceiver LoRa con IDE Arduino.

Fuente: Código propuesto por el autor

La *Figura 3* presenta el código propuesto para el uso del driver generado para aplicaciones con dispositivos LoRa y sistemas embebidos Arduino.

Cabe aclarar que de acuerdo al modo de configuración de los dispositivos LoRa, los registros tipo *flags* o *banderas* se comportan de acuerdo a la lógica presentada en la *Tabla 4*.

Tabla 4. Comportamiento de los registros tipo flag o banderas de los estados finales de transmisión o recepción de datos para los LoRa.

Fuente: Propuesta del autor

Registro tipo flag o bandera	Estado	Comentario
<i>RADIO.flagTx</i>	0	En modo transmisión, indica que aún no se ha transmitido la información del buffer <i>RADIO.frameTX</i> de cantidad <i>RADIO.dataLenTX Bytes</i> .
	1	En modo transmisión, indica que se ha transmitido la información del buffer <i>RADIO.frameTX</i> de cantidad <i>RADIO.dataLenTX Bytes</i> .
<i>RADIO.flagRx</i>	0	En modo recepción, indica que aún no se ha recibido información.
	1	En modo recepción, indica que se ha recibido <i>RADIO.dataLenRX Bytes</i> en el buffer <i>RADIO.frameRX</i> y que el usuario debe gestionar la información.
<i>RADIO.crc</i>	0	En modo recepción, indica que si <i>RADIO.flagRx</i> = 1, la información recibida en el buffer <i>RADIO.frameRX</i> es correcta .
	1	En modo recepción, indica que si <i>RADIO.flagRx</i> = 1, la información recibida en el buffer <i>RADIO.frameRX</i> es errada .

El driver también generan las variables:

- ***RADIO.snr***: Calidad de la señal a ruido de los datos recibidos.
- ***RADIO.rssi***: Indicador de fuerza de los datos recibidos.

Ejemplo de aplicación: red punto a punto (aplicación ping-pong)

Como ejemplo se plantea el uso de dos sistemas embebidos, uno utilizado como maestro y el otro utilizado como esclavo.

El dispositivo maestro se encarga de realizar la petición de datos a un dispositivo esclavo ubicado en un sitio remoto, el cual obtiene datos de sensores. El esclavo remite la información solicitada y continuamente se encuentra esperando peticiones del dispositivo esclavo. La

Figura 4 se presenta un diagrama pictórico de la aplicación propuesta: maestro-esclavo (sensor) con la utilización de dispositivo LoRa.



Ambos dispositivos cuentan con la misma configuración: frecuencia, BW, SF y CR

Figura 4. Principio de funcionamiento de aplicación “ping-pong”.
Fuente: Desarrollado por el autor

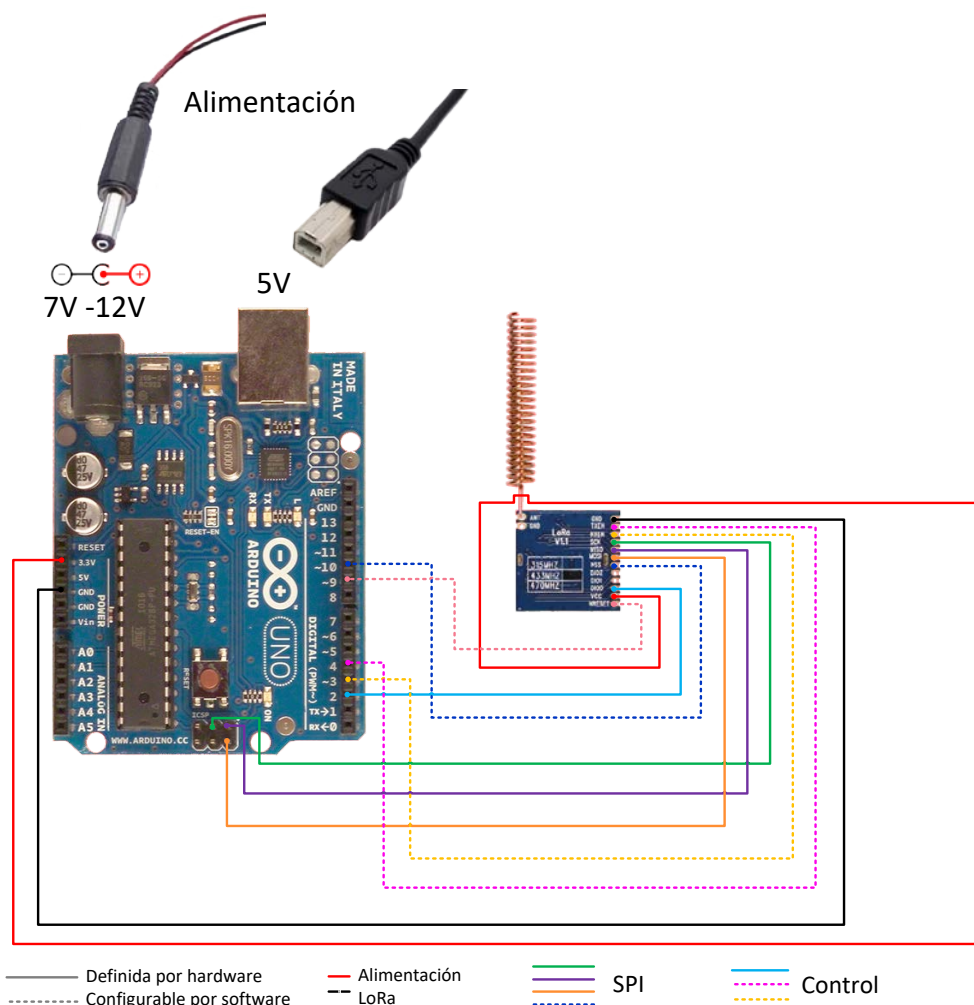


Figura 5. Esquema de conexión para LoRa1278 de NiceRF y Arduino UNO.
Fuente: Esquema propuesto por el autor

La *Figura 5* presenta el esquema de conexión utilizado entre los dispositivos LoRa SX1278 y el sistema embebido Arduino Uno. La *Tabla 5* presenta la nomenclatura utilizada en el transceiver SX1278 y la tarjeta Arduino Uno.

Tabla 5. Nomenclatura utilizada para la conexión entre la tarjeta LoRa1278 de NiceRF y la tarjeta Arduino UNO

Fuente: NiceRF (2015)

Tipo	Nomenclatura estándar	Notación en tarjetas	
		LoRa1278	Arduino UNO
Alimentación	VCC	VCC	3.3V
	GND	GND	GND
SPI	NSS	NSS	10
	MOSI	MOSI	11
	MISO	MISO	12
	SCK	SCK	13
CONTROL	RESET	NRESET	9
	DIO0	DIO0	2
	TXEN	TXEN	3
	RXEN	RXEN	4

El flujograma del código fuente empleado para la prueba se encuentra en el “ANEXO A Desarrollo del driver para el uso de los dispositivos de radio LoRa con sistemas embebidos” y los códigos fuente se presenta en la carpeta *radio/ejemplos/ping_pong* que acompaña al driver.

El sketch *Master_Sx1278_PingPong.ino* corresponde al archivo maestro y el sketch *Esclavo_Sx1278_PingPong.ino* corresponde al código del sistema embebido esclavo.

Ejemplo de aplicación: recepción de datos de varios dispositivos remotos (red punto - multipunto)

El ejemplo plantea un esquema de dispositivos conectados por una red punto-multipunto tipo estrella con un nodo central (circuito maestro) y 3 dispositivos esclavos.

La *Figura 6* presenta el esquema propuesto donde se aprecia la comunicación simultánea de 3 dispositivos hacia un nodo central. Se plantea el uso de esta aplicación con el uso del driver desarrollado para aplicaciones complejas con diversos sensores ubicados en sitios remotos. Cabe resaltar que los dispositivos LoRa deben ser de la misma referencia, estar configurados a la misma frecuencia y con los parámetros SF, BW y CR idénticos.

El dispositivo maestro se debe configurar como dispositivo “pasivo”, es decir continuamente recibir la información de los dispositivos esclavos. Una vez recibe la información de un dispositivo esclavo, remite la confirmación de recepción correcta de datos.

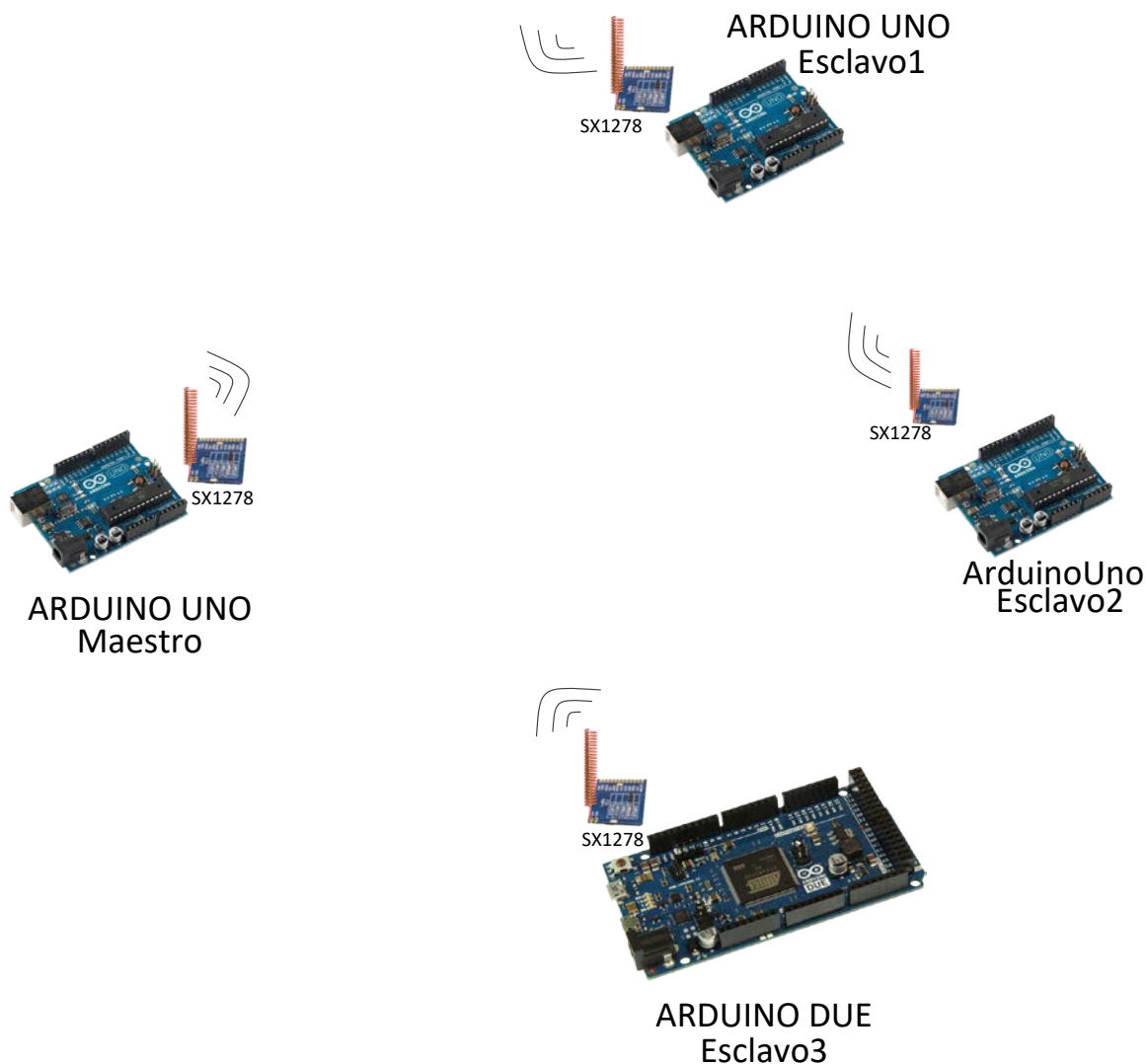


Figura 6. Tipología estrella para comunicación con diversos dispositivos Arduino.

Fuente: Esquema propuesto por el autor

El modem LoRa es *half dúplex*, es decir que los LoRa no puede transmitir o recibir datos en el mismo instante; esto significa que los transceiver LoRa deben ser configurados para transmitir o recibir datos. Cuando se incorpora más de dos dispositivos en un protocolo de comunicación *half dúplex*, se deben establecer estrategias que mitiguen los efectos adversos por colisión indeseada de datos cuando se reciben datos.

Si los dispositivos no cuentan con estrategias para evitar la transmisión de datos sobre el mismo canal de comunicación por parte de los dispositivos esclavos en un mismo instante, los datos llegan errados al dispositivo maestro y la comunicación de datos será errada y

ocasionará un consumo energético innecesario de energía en la transmisión de información por parte de los esclavos.

La solución para evitar la colisión de datos, corresponde a establecer tiempos de muestreo y tiempos aleatorios de “des-sincronización” cuando se transmiten datos.

En el documento Anexo A: “Desarrollo del driver para el uso de los dispositivos de radio LoRa con sistemas embebidos” – apartado “*Ejemplo de aplicación: recepción de datos de varios dispositivos remotos (red punto – multipunto)*”, se describe de manera detallada el proceso de desarrollo de la prueba desarrollada para crear la red punto-multipunto con dispositivos LoRa.

La *Figura 5* presenta el esquema de conexión utilizado entre los dispositivos LoRa SX1278 y el sistema embebido Arduino Uno. La *Tabla 5* presenta la nomenclatura utilizada en el transceiver SX1278 y la tarjeta Arduino Uno.

El código fuente utilizado por los dispositivos maestro y esclavos se presenta en la carpeta *radio/ejemplos/multipunto* que acompaña al driver.

El sketch *Master_Sx1278_Multipunto.ino* corresponde al archivo maestro y el sketch *Esclavo_Sx1278_Multipunto.ino* corresponde al código del sistema embebido esclavo.

INSTALACIÓN Y USO DEL DRIVER LORA PARA SISTEMAS EMBEBIDOS SDIN

Estructura de archivos que componen el driver

El driver está compuesto por una serie de archivos con *código fuente* y archivos tipo *header* organizados en dos subcarpetas. La estructura organizada de archivos se presenta en la *Figura 7*.

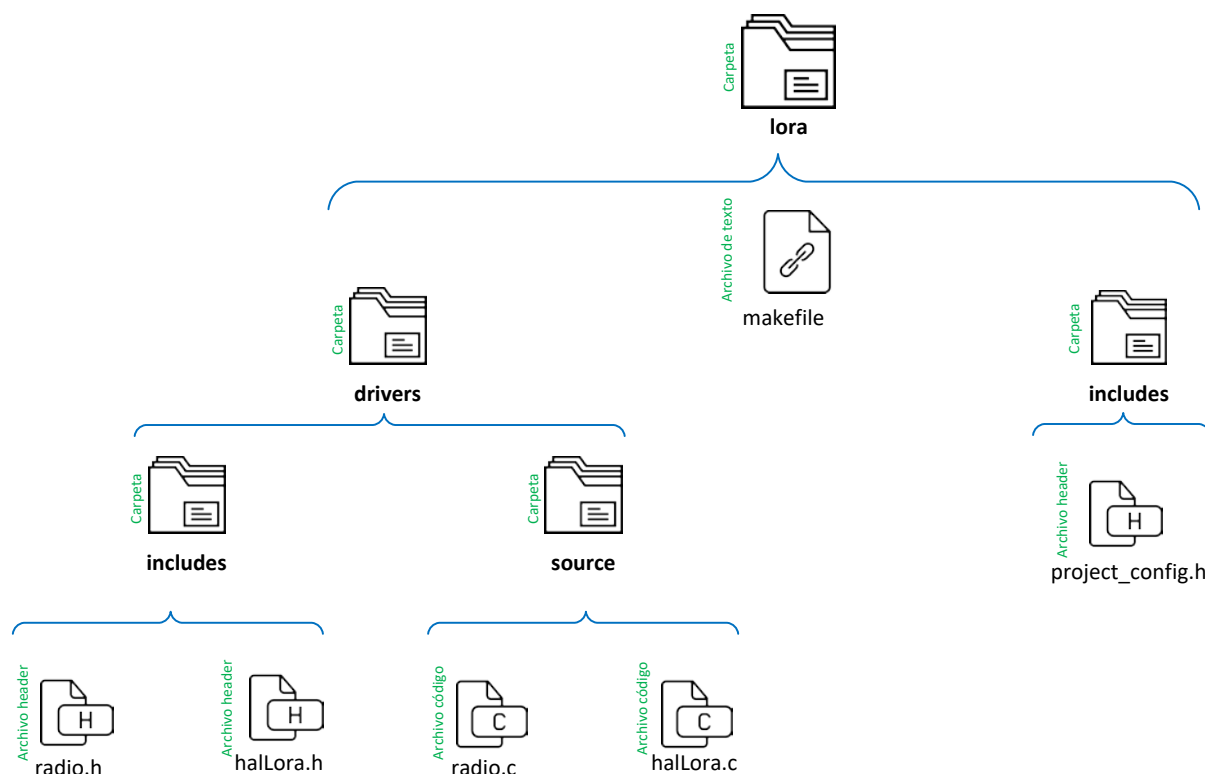


Figura 7. Estructura de archivos y carpetas del driver propuesto para el manejo de dispositivos LoRa utilizando FreeRTOS –SDIN.

Fuente: Esquema propuesto por el autor

La carpeta principal se denomina **lora** y está compuesta por los elementos:

- **Makefile:** es el archivo que cuenta con la información del proyecto para el uso de FreeRTOS. Con respecto al driver, el apartado “DRIVERS_SOURCE= \” debe contener la ruta de las librerías relacionadas con el control de los LoRa:

```

DRIVERS_SOURCE= \
    $(DRIVERS_SOURCE_DIR)/source/halLora.c \
    $(DRIVERS_SOURCE_DIR)/source/radio.c \

```

- **Carpeta drivers:**
 - Carpeta includes:

- **radio.h:** contiene la definición de variables y funciones para la configuración de los LoRa a nivel de software relacionado con el código presentado en **radio.c**.
 - **halLora.h:** cuenta con la definición de funciones del archivo **halLoRa.c** (*HAL o hardware abstraction layer*). Además contiene la definición de la nomenclatura asignada a los pines o terminales de control de los dispositivos LoRa.
- Carpeta source:
 - **radio.c:** contiene las funciones para la configuración de los LoRa a nivel de software.
 - **halLora.c:** contiene el código fuente de la capa de más bajo nivel que configura los periféricos internos de SDIN (interrupciones por cambio de flanco, comunicación SPI, comunicación UART, cambios de estados en terminales). A esta capa se le conoce como *HAL* o capa de abstracción de hardware.
- Carpeta includes:
 - **project_config.h:** el *header file* contiene la definición o selección del transceiver LoRa utilizado por la aplicación de SDIN. Es suficiente con definir:

	Instrucción		Comentario
#define	CFG_1272_SEMTECH	1	Selecciona el CHIP SX1272 de SEMTECH
#define	CFG_1278_DRF1278F	1	Selecciona el CHIP SX1278 de DORJI versión de tarjeta DRF1278F
#define	CFG_1278_NICERF1278	1	Selecciona el CHIP SX1278 de NiceRF versión de tarjeta LORA1278

○

La *Figura 8* presenta la gestión entre los diferentes archivos que componen el driver planteado.

Resulta estratégica la organización de los diferentes archivos debido a que el driver busca que el código presentado en **radio.c** se comporte como una capa de alto nivel, manteniendo el código invariante al uso de diferentes sistemas embebidos como el módulo SDIN, módulos SDIN (microcontrolador de 32 bits STM32f10x) o tarjetas de STMicroelectronics; mientras que el código presentado en **halLora.c** corresponde a una capa de alto bajo nivel de abstracción de hardware que de acuerdo al tipo de sistema embebido utilizado pueda ser ajustado para realizar la correcta homologación al microcontrolador utilizado.

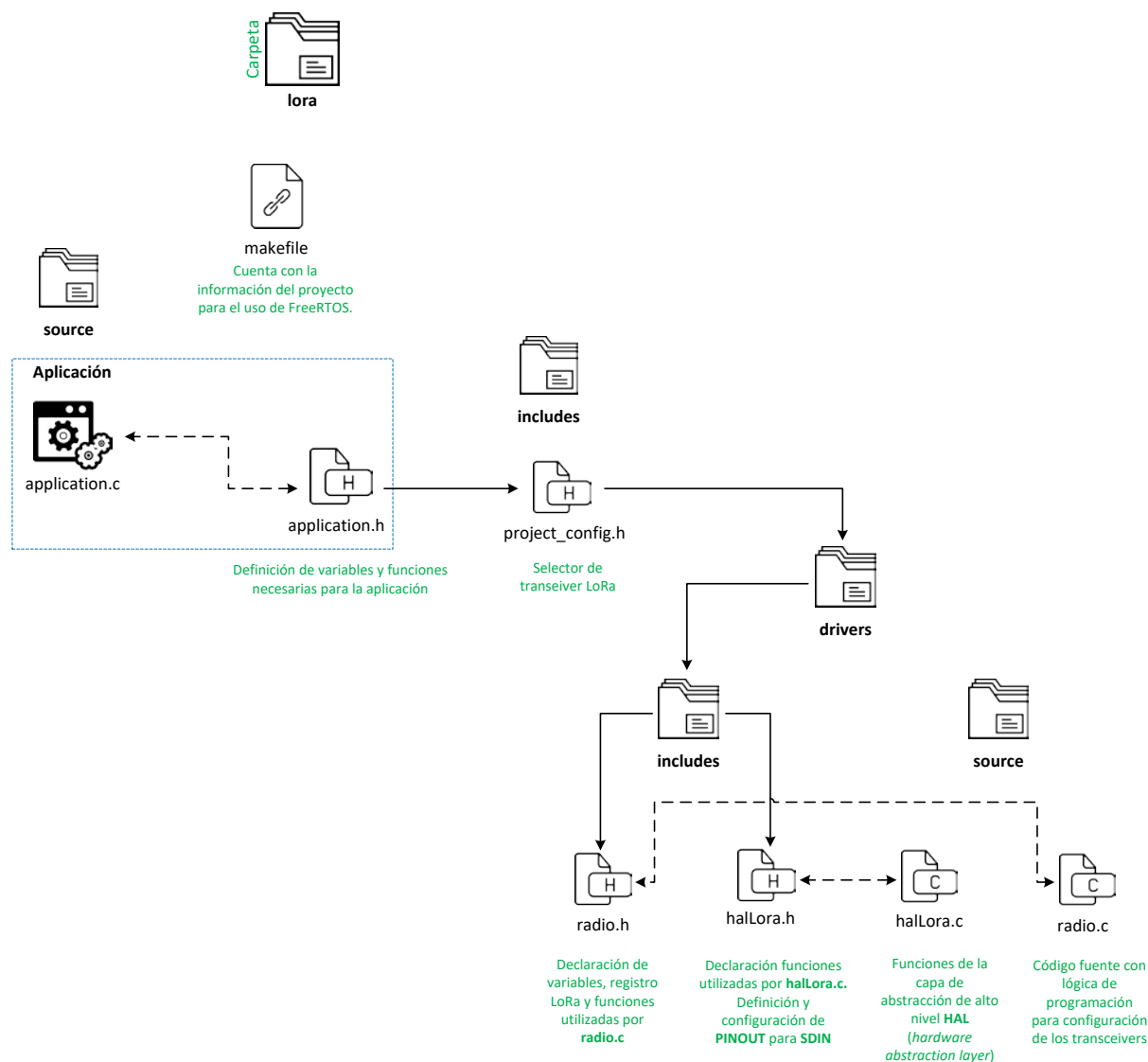


Figura 8. Gestión entre archivos del driver propuesto para el manejo de dispositivos LoRa para los módulos SDIN.

Fuente: Esquema propuesto por el autor

La Figura 8 presenta la gestión entre los diferentes archivos que componen el driver planteado.

Resulta estratégica la organización de los diferentes archivos debido a que el driver busca que el código presentado en **radio.c** se comporte como una capa de alto nivel, manteniendo el código invariante al uso de diferentes sistemas embebidos como Arduino, módulos SDIN o tarjetas de STMicroelectronics; mientras que el código presentado en **halLora.c** corresponde a una capa de alto bajo nivel de abstracción de hardware que de acuerdo al tipo de sistema embebido utilizado pueda ser ajustado para realizar la correcta homologación al microcontrolador utilizado.

Uso de librería para el control de dispositivos LoRa en aplicaciones para SDIN

La instalación del driver para el control de los dispositivos LoRa utilizando SDIN (se recomienda que el equipo de cómputo tenga instalado el IDE Eclipse), se realiza copiando los archivos del fichero principal **lora** junto con la carpeta donde se encuentra instalado los archivos del S.O. FreeRTOS:

- Incluir las librerías para control por software (**radio**) y hardware (**halLora**) en el header file **application.h**:

```
#include "radio.h"           //Librería de alto nivel para configuración por software de los radio LoRa
#include "halLora.h"         //Librería de bajo nivel para configuración de hardware radio LoRa (HAL)
#include "project_config.h"  //Librería para la selección del transceiver LoRa
```

- Definir los pines de control de la tarjeta SDIN con respecto a los transceiver LoRa:

//Definición de los pines de control de la tarjeta SDIN

```
#define LORA_SPI           SPI2
#define LORA_SPI_CLK       RCC_APB1Periph_SPI2
#define LORA_SPI_IRQn      SPI2_IRQn
#define LORA_SPI_PORT      GPIO [letra del puerto]

// SPI CS/NSS
#define LORA_NSS_PIN        GPIO_Pin_[número del Pin]
#define LORA_NSS            LORA_SPI_PORT, LORA_NSS_PIN

// SPI SCK
#define LORA_SCK_PIN        GPIO_Pin_[número del Pin]
#define LORA_SCK            LORA_SPI_PORT, LORA_SCK_PIN

// SPI MISO
#define LORA_MISO_PIN        GPIO_Pin_[número del Pin]
#define LORA_MISO            LORA_SPI_PORT, LORA_MISO_PIN

// SPI MOSI
#define LORA_MOSI_PIN        GPIO_Pin_[número del Pin]
#define LORA_MOSI            LORA_SPI_PORT, LORA_MOSI_PIN

#define LORA_RST_PORT        GPIOA
#define LORA_RST_PIN        GPIO_Pin_[número del Pin]
#define LORA_RST            LORA_RST_PORT, LORA_RST_PIN
#define LORA_RST_MODE        GPIO_Mode_Out_PP
```

- Configurar los dispositivos LoRa de acuerdo a las necesidades del usuario. Esto se debe incorporar en la función propia de **application.c** (ejemplo "TestLora_Task()"):

- Iniciar las variables tipo *flag* o *banderas*:

//Inicialización de variables del driver

```
RADIO.flagTx = 0;
```

```
RADIO.flagRx =0;
```

RADIO.crc = 0;

- Definir la frecuencia de trabajo del transceiver:

RADIO.freq = [frecuencia de trabajo]; //Frecuencia de la banda ISM compatible con LoRa

Teniendo en cuenta que existen bandas de frecuencias de uso privativo (telefonía, internet, radio, entre otras), existen frecuencias de libre uso conocidas como ISM (Industrial, Scientific and Medical) que son frecuencias intencionalmente libres para uso no comercial en aplicaciones de tipo industrial, científico y médico. Se recomienda al usuario verificar la compatibilidad de la frecuencia de trabajo con la *Tabla 2* (bandas de uso ISM).

- Definir la potencia de transmisión del transceiver:

//Configuración de potencia

RADIO.txpow = [Máxima potencia (dBm)] //Máxima TX potencia

RADIO.imax = [Corriente máxima (mA)] //Por defecto 100mA 45mA <=Imax <= 240mA

La potencia máxima se expresa en dBm y el valor definido debe estar entre 2 a 17.

La corriente máxima está dada en mA y el valor definido se debe encontrar entre 45mA y 240mA. El valor típico de corriente es de 100mA. Para valores significativos de corrientes (mayores a 125mA), se recomienda verificar la corriente máxima de suministro soportada por la fuente o batería de alimentación.

- Definir las variables de configuración de la comunicación entre los transceivers LoRa:

//Variables de configuración de la comunicación LoRa

RADIO.sf = [Spread factor SF] //Configuración del Spread factor

RADIO.bw = [Ancho del canal BW] //Configuración del ancho de canal

RADIO.cr = [Coding rate CR] //Configuración de Coding rate

Los dispositivos LoRa cuentan con los parámetros *SF*, *BW* y *CR* que según las combinaciones entre estos, permite determinar el alcance o distancia máxima de recepción de datos, la velocidad de los datos y sobrecarga por corrección o detección de errores en los datos. Estos valores deben ser los mismos en todos los dispositivos radio LoRa de la red inalámbrica.

Los parámetros independientes de configuración se definen como (Semtech SX1272, 2015 & Semtech SX1278, 2015):

BW: Ancho de banda (entre más bajo sea este valor, el tiempo de durante la transmisión será mayor)

SF rate: Es el factor de alcance expresado como logaritmo de base 2. Entre mayor sea este valor, se tendrá un mejor rendimiento en la transmisión de datos.

CR: Tasa de codificación de errores: Entre mayor sea este valor, mayor será la fiabilidad de los datos, pero con una sobrecarga en el tiempo de transmisión.

La *Tabla 3* presenta la combinación de los principales variables independientes para la configuración de la comunicación entre transceivers LoRa. Cabe aclarar que estos valores deben ser los mismos en todos los dispositivos radio LoRa de la red inalámbrica.

- Configurar los pines I/O y la comunicación SPI:

```
hal_init() //Configuración de puertos de SDIN
```

- Configurar los pines I/O y la comunicación SPI:

```
radio_init(); //Configuración de inicio para los LoRa
```

Corresponde a la función general que configura todos los parámetros de inicio del transceiver LoRa.

- Crear la aplicación con la lógica propuesta por el desarrollador:

```
while(1)
{
    [Lógica de control implementada por el usuario]
}
```

- En la función **Application_Start()**, crear la tarea relacionada con la aplicación de uso de los LoRa (ejemplo: TestLora_Task):

```
xTaskCreate(TestLora_Task, "TestLora_Task", TEST_LORA_TASK_STACK, NULL,
TEST_LORA_TASK_PRIORITY, &TestLoraTask_Handle); //TestLora_Task->
PRUEBA UNO A UNO - Esclavo
```

Cabe aclarar que de acuerdo al modo de configuración de los dispositivos LoRa, los registros tipo *flags* o *banderas* se comportan de acuerdo a la lógica presentada en la *Tabla 4*.

El driver también genera las variables:

- **RADIO.snr:** Calidad de la señal a ruido de los datos recibidos.
- **RADIO.rssi:** Indicador de fuerza de los datos recibidos.

Ejemplo de aplicación: red punto a punto (aplicación ping-pong) con módulos SDIN

Como ejemplo se plantea el uso de dos sistemas embebidos, uno utilizado como maestro y el otro utilizado como esclavo.

El dispositivo maestro se encarga de realizar la petición de datos a un dispositivo esclavo ubicado en un sitio remoto, el cual obtiene datos de sensores. El esclavo remite la información solicitada y continuamente se encuentra esperando peticiones del dispositivo esclavo. La

Figura 9 se presenta un diagrama pictórico de la aplicación propuesta: maestro-esclavo (sensor) con la utilización de dispositivo LoRa.



Ambos dispositivos cuentan con la misma configuración: frecuencia, BW, SF y CR

Figura 9. Principio de funcionamiento de aplicación “ping-pong” con SDIN.
Fuente: Desarrollado por el autor

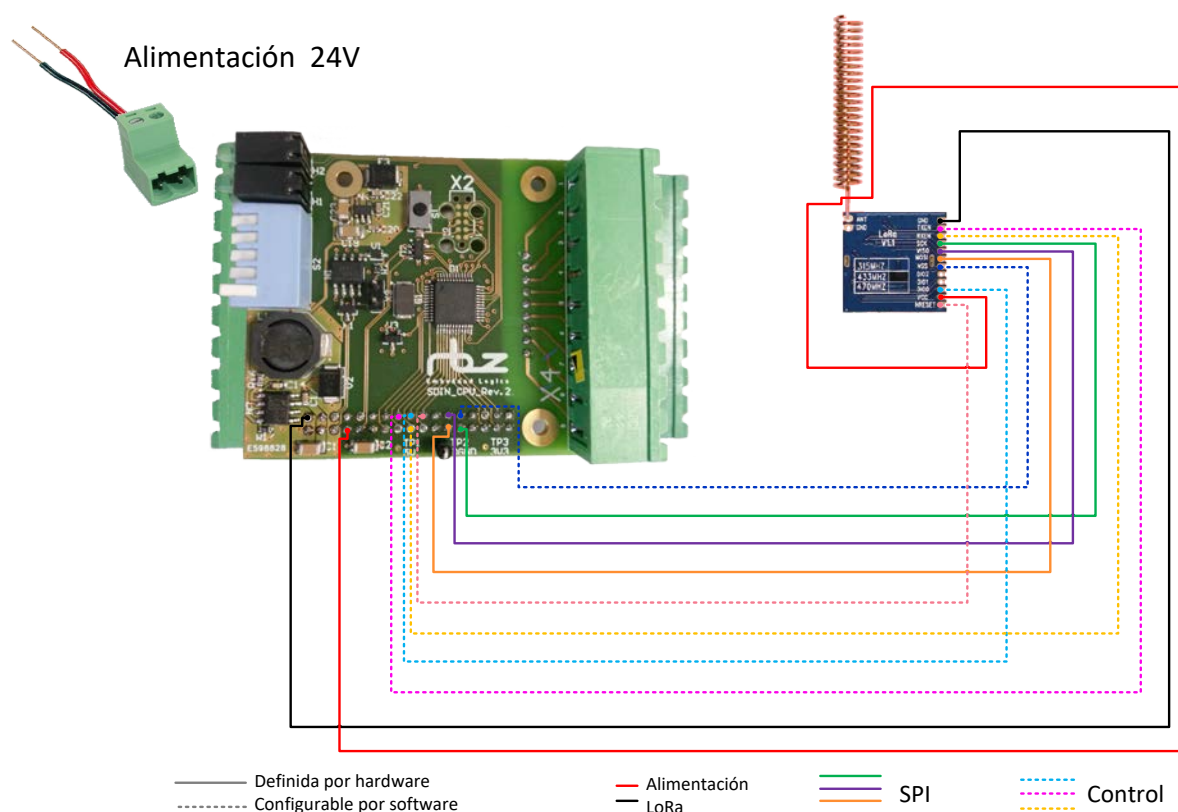


Figura 10. Esquema de conexión para LoRa1278 de NiceRF y módulo SDIN.
Fuente: Esquema propuesto por el autor

La Figura 10 presenta el esquema de conexión utilizado entre los dispositivos LoRa SX1278 y el sistema embebido SDIN. La Tabla 6 presenta la nomenclatura utilizada en el transceiver SX1278 y la tarjeta SDIN.

Tabla 6. Nomenclatura utilizada para la conexión entre la tarjeta LoRa1278 de NiceRF y el módulo SDIN

Fuente: NiceRF (2015)

Tipo	Nomenclatura estándar	Notación en tarjetas	
		LoRa1278	SDIN
Alimentación	VCC	VCC	3.3V
	GND	GND	GND
SPI	NSS	NSS	PB12
	MOSI	MOSI	PB15
	MISO	MISO	PB14
	SCK	SCK	PB13
CONTROL	RESET	NRESET	PA7
	DIO0	DIO0	PA5
	TXEN	TXEN	PA3
	RXEN	RXEN	PA4

El flujograma del código fuente empleado para la prueba se encuentra en el “ANEXO A Desarrollo del driver para el uso de los dispositivos de radio LoRa con sistemas embebidos” y los códigos fuente se presenta en la carpeta *lora/source/application.c* que acompaña al driver.

La tarea se denomina *TestLora_Task()* corresponde al código fuente del dispositivo maestro del sistema embebido SDIN.

Ejemplo de aplicación: recepción de datos de varios dispositivos remotos (red punto - multipunto) con módulos SDIN

El ejemplo plantea un esquema de dispositivos conectados por una red punto-multipunto tipo estrella con un nodo central (circuito maestro) y 3 dispositivos esclavos.

La *Figura 11* presenta el esquema propuesto donde se aprecia la comunicación simultánea de 3 dispositivos hacia un nodo central. Se plantea el uso de esta aplicación con el uso del driver desarrollado para aplicaciones complejas con diversos sensores ubicados en sitios remotos. Cabe resaltar que los dispositivos LoRa deben ser de la misma referencia, estar configurados a la misma frecuencia y con los parámetros SF, BW y CR idénticos.

El dispositivo maestro se debe configurar como dispositivo “pasivo”, es decir continuamente recibir la información de los dispositivos esclavos. Una vez recibe la información de un dispositivo esclavo, remite la confirmación de recepción correcta de datos.

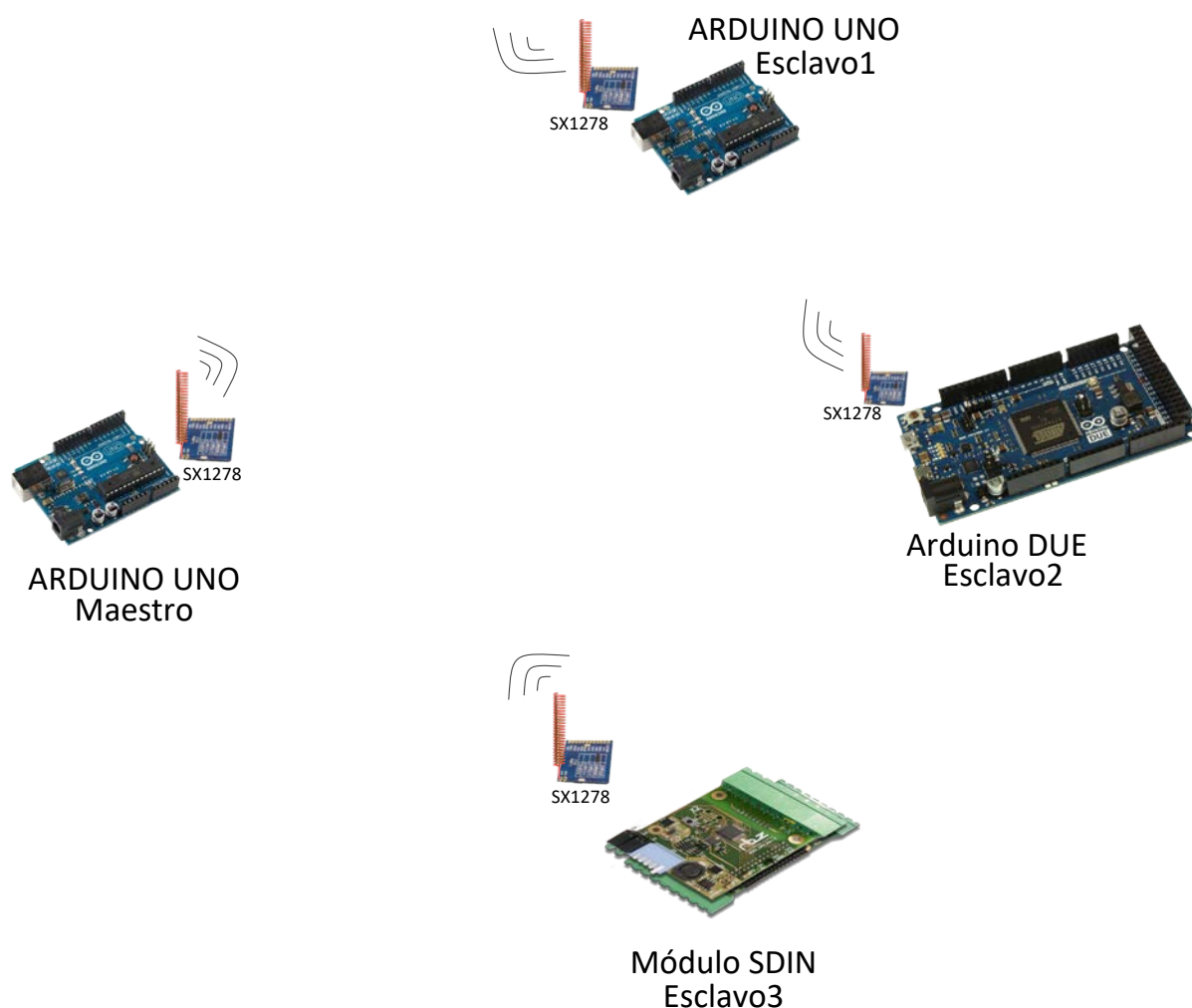


Figura 11. Tipología estrella para comunicación con diversos dispositivos.

Fuente: Esquema propuesto por el autor

El modem LoRa es *half dúplex*, es decir que los LoRa no puede transmitir o recibir datos en el mismo instante; esto significa que los transceiver LoRa deben ser configurados para transmitir o recibir datos. Cuando se incorporan más de dos dispositivos en un protocolo de comunicación *half dúplex*, se deben establecer estrategias que mitiguen los efectos adversos por colisión indeseada de datos cuando se reciben datos.

Si los dispositivos no cuentan con estrategias para evitar la transmisión de datos sobre el mismo canal de comunicación por parte de los dispositivos esclavos en un mismo instante, los datos llegarán errados al dispositivo maestro y la comunicación de datos será errada y ocasiona un consumo energético innecesario de energía en la transmisión de información por parte de los esclavos.

La solución para evitar la colisión de datos, corresponde a establecer tiempos de muestreos y tiempos aleatorios de “des-sincronización” cuando se transmiten datos; es decir evitar con los tiempos adiciones de transmisión de datos la colisión de datos.

En el documento Anexo A: “Desarrollo del driver para el uso de los dispositivos de radio LoRa con sistemas embebidos” – apartado “*Ejemplo de aplicación: recepción de datos de varios dispositivos remotos (red punto – multipunto)*”, se describe de manera detallada el proceso de desarrollo de la prueba desarrollada para crear la red punto-multipunto con dispositivos LoRa.

La *Figura 10* presenta el esquema de conexión utilizado entre los dispositivos LoRa SX1278 y el sistema embebido Arduino Uno. La *Tabla 6* presenta la nomenclatura utilizada en el transceiver SX1278 y la tarjeta SDIN.

El código fuente utilizado por el dispositivo esclavo se presenta en la carpeta *lora/source/application.c* que acompaña al driver.

La tarea se denomina *TestLoraMultiple_Task()* corresponde al código fuente del dispositivo esclavo del sistema embebido SDIN.

BIBLIOGRAFÍA

NiceRF (2015). LoRa1278 100 mW 4 km larga distancia y alta sensibilidad (-139 dBm) 433 MHz módulo de transceptor inalámbrico. Recuperado de <http://es.aliexpress.com/item/2pcs-lot-LoRa1278-100mW-4km-Long-Distance-and-High-Sensitivity-139-dBm-433MHz-Wireless-Transceiver-Module/32461365864.html>, China. Consultado el 22 de junio de 2016

Semtech SX1272 (2015). SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver. Datasheet. Semtech Corporation. Estados Unidos.

Semtech SX1278 (2015). SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver. Datasheet. Semtech Corporation. Estados Unidos.

Texas Instruments (2005). SM-Band and Short Range Device Regulatory Compliance Overview. Estados Unidos.