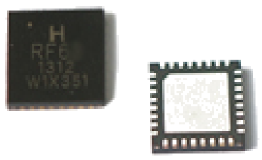
RF69 Communication Example

This chapter will guide the user to carry out a pair of RF69 transmitting and receiving communication experiment through HopeDuino. RF69 is a main push wireless transceiver chip belonging to HopeRF, with +20dBm transmit power, -120dBm sensitivity and link budget up to 140dB. The RFM series module is designed by the chip，the specific models are RFM69, RFM69C, RFM69H and RFM69HC. They all have small size, high performance, easy for users to debug.

1. Tools and software needed to be prepared



- Arduino IDE version 1.0.5
- HopeDuino board (two pieces)
  (If you have not used the HopeDuino board, please refer to the 《AN0002-HopeDuino Platform Construction Guideline》)
- USB cable(Type A to Type B)
- Module based on RF69 chip design, such as RFM69、RFM69C、RFM69H and RFM69HC.

RFM69

RFM69H

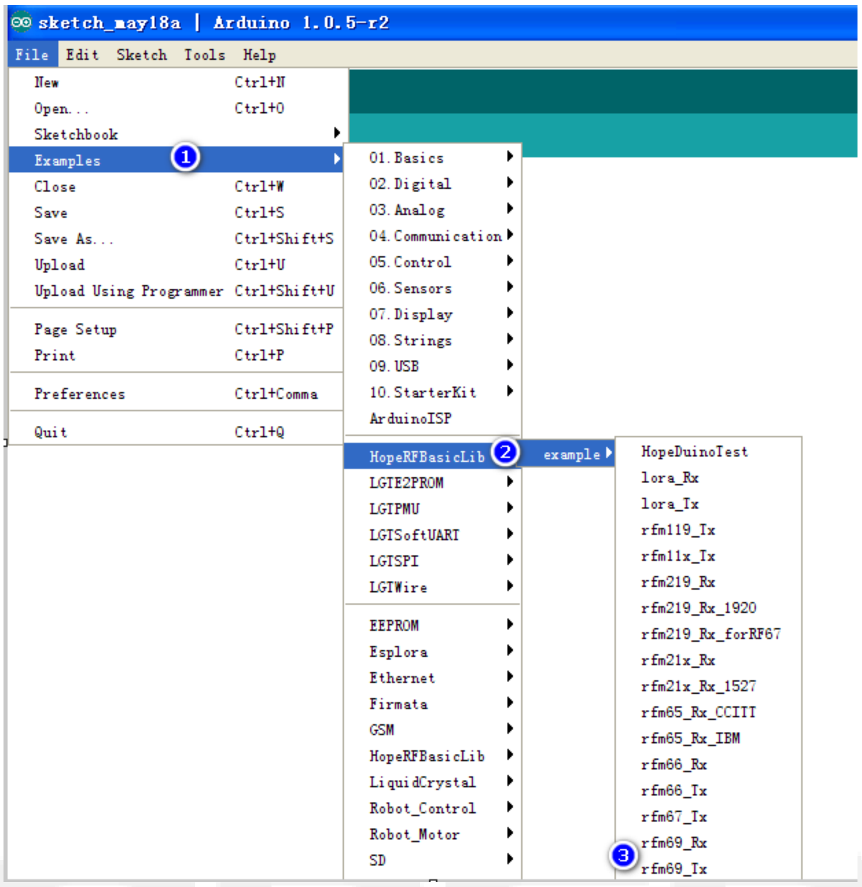
RFM69C

RFM69HC

2. Hands-on experiment

- The two RFM69x modules (with conversion board) are inserted into the corresponding HopeDuino board.
- Connect the two HopeDuino boards to PC with USB cable.
- Open Arduino IDE interface, Click 【Files】→【Examples】→【HopeRFBasicLib】→【example】→【rfm69_Tx】, as shown below:
- Open Arduino IDE interface, Click 【Files】→【Examples】→【HopeRFBasicLib】→【example】→【rfm69_Rx】, as shown below:

⚠Notice: You couldn't find [HopeRFBasicLib] in [Examples] because you didn't install the HSP provided by HopeRF. Please refer to 《AN0002-HopeDuino Platform Construction Guideline》.

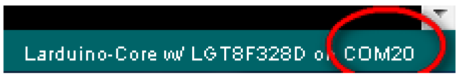


- At this time the Tx program and Rx program have been opened, please compile the download programs according to the corresponding COM port.

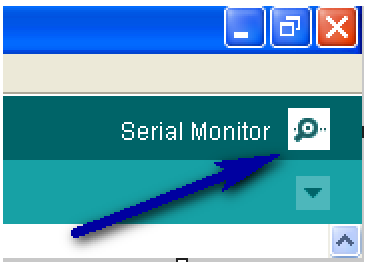
Notice:

1. Do not know how to compile the download code, please refer to 《AN0002-HopeDuino Platform Construction Guideline》

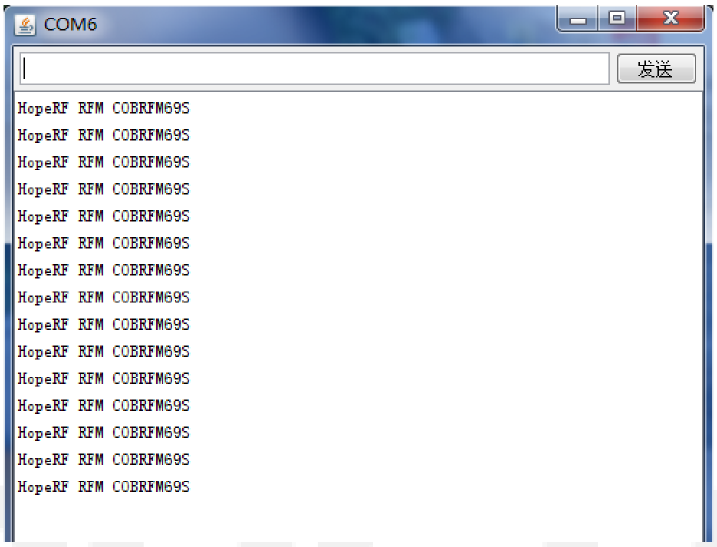
2. HopeDuino platform support multiple boards connected to the same PC. But you need to modify the parameters manually. So you need to pay special attention to which COM port when you download the program every time. COM port is in the lower right corner of Arduino interface, as shown below.



- After the two programs are downloaded, the Tx board will transmit a packet of data through module RFM69x. The Rx board will receive a packet of data through module RFM69x periodically and upload the data to PC through UART (USB). At this point, you can set the COM of Arduino IDE as the port connected with Rx board. Open the “Serial Monitor” , as shown below.



- Click the "Serial Monitor", pop up the serial port assistant interface, as shown below. Window will display the received data message.



```
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
HopeRF RFM COBRFM69S
```

⚠ Notice:

1. The receiving program enables UART library function. On the description of library function UART, please refer to the "HopeDuino_UART" library file. It is also stored in the HopeRFLib.

3. Program Explanation

- rfm69_Tx.ino Case explanation

```
#include <HopeDuino_RFM69.h>                          // Call the corresponding library file.
rf69Class radio;                                          // Define variable radio for RF69.
byte str[21] = {'H','o','p','e','R','F',' ','R','F','M',' ','C','O','B','R','F','M','6','9','S'};    //Message to be transmitted

void setup()
{
radio.Modulation          = FSK;              //Modulation mode is FSK
radio.COB                 = RFM69;         //Module is RFM69
radio.Frequency           = 434000;           // Target frequency is 434MHz
radio.OutputPower         = 10+18;            //Output power is 10dBm
radio.PreambleLength      = 16;            //Preamble length is 16 bytes
radio.FixedPktLength      = true;             //Message length is fixed
radio.PayloadLength       = 21;               //Payload length is 21 bytes
radio.CrcDisable          = true;             //Disable CRC for true
radio.AesOn               = false;         //Disable AES function
radio.SymbolTime          = 416000;           //Rate is 2.4Kbps
radio.Deviation            = 35;                //Frequency deviation is 35 KHz
radio.BandWidth           = 100;               //Received bandwidth is 100 KHz
radio.SyncLength          = 3;                //Sync length is 3 bytes, the value is 0xAA2DD4
radio.SyncWord[0]          = 0xAA;
radio.SyncWord[1]          = 0x2D;
radio.SyncWord[2]          = 0xD4;
radio.vInitialize();                             //Initialize radio
radio.vGoStandby();                              // Enter standby mode
}

void loop()
{
radio.bSendMessage(str, 21);                   //Transmit a packet of message every one second
delay(1000);
}
```

➢ rfm69_Rx.ino Case explanation

```
#include <HopeDuino_RFM69.h>                 // Call the corresponding library file.
                                                 //Calling UART is added because of using UART.
#include <HopeDuino_UART.h>

rf69Class radio;                                 //Define variable radio for RF69
uartClass uart;                                //Define variable uart for UART
byte getstr[21];                                 //Define pending data buffer
//Define pending data buffer
void setup()
{
  radio.Modulation            = FSK;             //Corresponding modulation mode is FSK
radio.COB                   = RFM69       ;          //Module is RFM69
radio.Frequency             = 434000;            //Target frequency is 434MHz
radio.OutputPower           = 10+18;            //Output power is 10dBm
radio.PreambleLength      = 16;               //Preamble length is 16 bytes
radio.FixedPktLength      = true;            //Message length is fixed.
radio.PayloadLength       = 21;                 //Payload length is 21 bytes.
radio.CrcDisable            = true;                //Disable CRC is true
radio.AesOn                  = false;          //Disable AES function
radio.SymbolTime          = 416000;          //Rate is 2.4Kbps
radio.Deviation             = 35;               //Frequency deviation is 35KHz
radio.BandWidth           = 100;              //Received bandwidth is 100KHz
radio.SyncLength          = 3;               //Sync length is 3 bytes, the value is 0xAA2DD4
  radio.SyncWord[0]         = 0xAA;
  radio.SyncWord[1]         = 0x2D;
  radio.SyncWord[2]       = 0xD4;
  radio.vInitialize();                        // Initialize radio
  radio.vGoRx();                                  //enter receiving mode
  uart.vUartInit(9600, _8N1);              //Initialize UART, parameters are 9600 baud rate and 8N1 format.
}

void loop()
{
  if(radio.bGetMessage(getstr)!=0)            //Check radio whether to receive data function,
                                                  //analyze data received.
      {
      uart.vUartPutNByte(getstr, 21);             // Output the received data to PC via UART
      uart.vUartNewLine();                        //UART newline is easy to read.
      }
}
```

4. RF69 Library Function Description

"RFM69.h"and 和"RFM69.cpp"library files are stored in Arduino IDE files \ libraries \ HopeRFLib.

- **FreqStruct**

  Type: Union type

  Function: Define frequency register for RF69

  Contents: Freq, long integer, 4 bytes, frequency value;

  FreqL， byte, low 8 bit from splitting Freq value is [0:7]

  FreqM， byte, mid 8 bit from splitting Freq value is [8:15]

  FreqH， byte, high 8 bit from splitting Freq value is [16:23]

  FreqX， byte, redundancy, rounding up 4 bytes, no meaning

- **modulationType**

  Type: Enumeration type

  Function: Select modulation mode

  Contents: OOK、FSK、GFSK

  OOK——On-Off-Key is ASK modulation, a special case of ASK modulation

  FSK——Frequency-Shift-Key, relative to the ASK has a stronger anti interference effect. But the current is larger than the ASK under the same power.

  GFSK——FSK modulation with Gauss filter

- **moduleType**

  Type: Enumeration type

  Function: Select module type

  Contents: RFM65, RFM65C, RFM69, RFM69C, RFM69H, RFM69HC

- **Modulation**

  Type: modulation type

Function: Define modulation mode, select one of OOK, FSK and GFSK.

- **COB**
  Type: module type
  Function: Define module type, COB is Chip-On-Board, select one of RFM65, RFM65C, RFM69, RFM69C, RFM69H and RFM69HC.

- **Frequency**
  Type: lword type, unsigned long.
  Function: working frequency, the unit is KHz, for example: Frequency = 433920， indicates 433.92MHz.

- **SymbolTime**
  Type: lword type, unsigned long
  Function: working rate, the unit is ns, for example: SymbolTime = 416000，indicates each symbol is 416us, that is 2.4kbps.

- **Deviation**
  Type: long word type, unsigned long
  Function: frequency deviation for FSK and GFSK transmitting, the unit is KHz, for example: Deviation=45, indicates the frequency deviation is 45KHz.

- **BandWidth**
  Type: unsigned int
  Function: Target receiver bandwidth for reception, the unit is KHz, for example: BandWidth = 100， indicates the receiver bandwidth is 100KHz.

- **OutputPower**
  Type: unsigned char
  Function: output power, the unit is 0.5dBm, wherein
  For RFM69/RFM69C, set the scope of 0-31, on behalf of -18dBm ~ +13dBm,
  For RFM69H/RFM69HC, set the scope of 0-31, on behalf of -11dBm ~ +20dBm.

- **PreambleLength**
  Type: word type, unsigned int
  Function: preamble length for transmitting, the unit is byte.

- **CrcDisable**
  Type: bool type
  Function: Select whether the data package feature with CRC, set true to disable CRC function, set false to open CRC function.

- **FixedPktLength**
  Type: bool type
  Function: Define the data packet length is fixed or variable, set true to represent the fixed packet length, set false

to represent the variable length.

- **AesOn**

  Type: bool type

  Function: Define whether the data packet needs AES128 encryption, set true on behalf of the data are processed by AES128, set false on behalf of the data are not processed by AES128.

- **AfcOn**

  Type: bool type

  Function: Define whether to enable AFC functions, configure the parameter for receiving.

- **SyncLength**

  Type: byte

  Function: In wireless packet format, synchronous word length, the setting range is 1~8 bytes. Don't set to 0 bytes.

- **SyncWord[8]**

  Type: byte array

  Function: In setting packet format, synchronous word contents needs to be consistent with SyncLength settings (length).

- **PayloadLength**

  Type: byte

  Function: In fixed packet length mode, defines the length of the fixed packet

- **AesKey[16]**

  Type: byte array

  Function: After opening the AES128 encryption mode, the key content of the AES128 is defined. The key length is 128 bits, which is 16 bytes.

- **vInitialize**

  Type: function

  Input: None

  Output: None

  Function: Initialize module (chip), applicable to RFM69/C and RFM69H/HC module, call it at the start of the program. Before the call, the above related variables are set to complete. After the initialization function configuration is completed (containing call vConfig function), let the module (chip) into the Standby state, that is, non - transmitting, non - receiving, non - sleeping.

- **vConfig**

  Type: function

  Input: None

  Output: None

  Function: Configure parameters to the module (chip), suitable for the occasion needs to re configure the parameters in the working process. The same need to complete the associated variables before the call. If the associated variables set up, follow-up without modification, only to re configure the parameter, you can call it directly. If you need to switch frequency etc. in the working process, need to re modify the relevant parameters， and then call them again. After the call is completed, you need to use the mode switching function, so that let the chip work accurately in the specific mode. The mode switching functions are vGoRx、vGoStandby and vGoSleep etc.

- **vGoRx**
  Type: function
  Input: none
  Output: none
  Function: Configure module (chip) into the receiving mode

- **vGoStandby**
  Type: function
  Input: none
  Output: none
  Function: Configure module (chip) into the standby mode

- **vGoSleep**
  Type: function
  Input: none
  Output: none
  Function: Configure module (chip) into the sleep mode

- **bSendMessage**
  Type: function
  Input: **msg[ ]**， unsigned char, the calling entrance (pointer) of array to be transmitted.
  length， unsigned char, the length of array to be transmitted, the unit is byte.
  Output: bool type, true indicates the transmitting is successful, false indicates the transmitting is failure, such as: push over time, etc.
  Function: transmit the data only once (one frame), return to standby mode automatically after completion of the transmission.

- **bGetMessage**
  Type: function
  Input: msg[ ]， unsigned char, the calling entrance (pointer) of array to be received.
  Output: Returns the length of the received data, 0 indicates that the data is not received;
  Function: check whether to receive data. The object is the IO state of the chip output. If you do not receive the data, return 0; if you receive the data, return the received data length. After the completion of receiving, the module (chip) is still in the receiving state.

- **vRF69SetAesKey**
  Type: function
  Input: none

Output: none

Function: set the AesKey[16] to the relevant registers in the module (chip).

- **vTrigAfc**

  Type: function

  Input: none

  Output: noen

  Function: do an AFC adjustment manually, applicable to the AFC_ON state.

- **vDirectRx**

  Type: function

  Input: none

  Output: none

  Function: In the current parameter configuration, let the module (chip) in the DirectMode, output received data stream through the DIO2 pin, used in sensitivity tests generally.

- **vChangeFreq**

  Type: function

  Input: freq, long integer type, the target frequency to be changed.

  Output: none

  Function: By this function, you can switch the operating frequency quickly (other parameters unchanged, such as speed, bandwidth, etc., only adjust the frequency). After the switch is completed, the system continues to be in the receiving mode, which is suitable for the application of frequency hopping mechanism.

- **bReadRssi**

  Type: function

  Input: none

  Output: return the value of the currently detected RSSI

  Function: do a RSSI test manually, and return the value.

五、Pin Assignment Table:

HopeDuino	MCU	RF69

13	PB5	SCK
12	PB4	MISO
11	PB3	MOSI
10	PB2	nCS
9	PB1	POR
8	PB0	DIO0
7	PD7	DIO1（jumper）
6	PD6	DIO2（jumper）
5	PD5	DIO3（jumper）
4	PD4	DIO4（jumper）

5. Version Records

Version	Revised Contents	Date
1.0	Initial version	2016-03-29
1.1	Add watermarks, program explanations and descriptions	2016-04-06