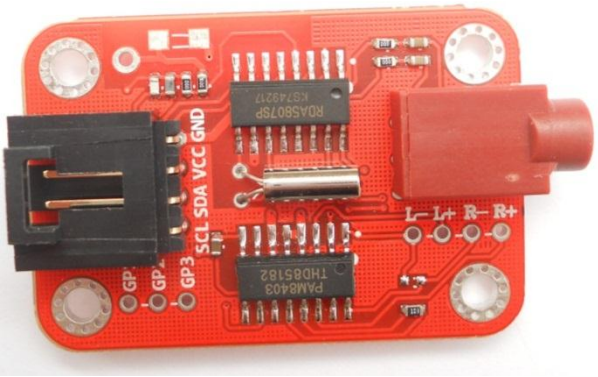
FM Radio Receiver Module

Introduction

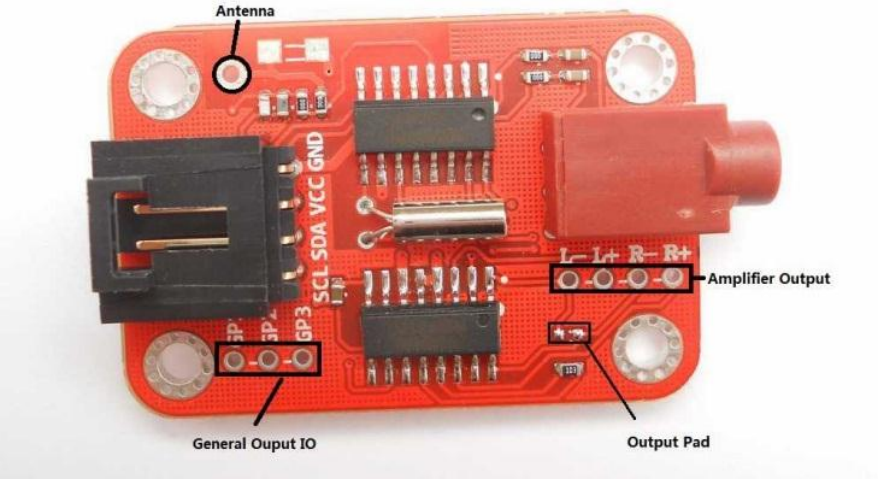
Wanna make a radio yourself with Arduino? Here comes the Radio Receiver module which makes it very easy to make your own radio.

This module communicates with Arduino or other MCU through I2C interface. All commands are sent through I2C interface. We also create a library for Arduino to control it.



Interface

The anti-reverse 4pin header is I2C interface. And the headphone jack is the read one.



I2C

*VCC is 5V. TTL level of I2C interface is also 5V. With anti-reverse header, never have to worry about connecting in the wrong way if you have **4pin anti-reverse cable**.*

Headphone Jack

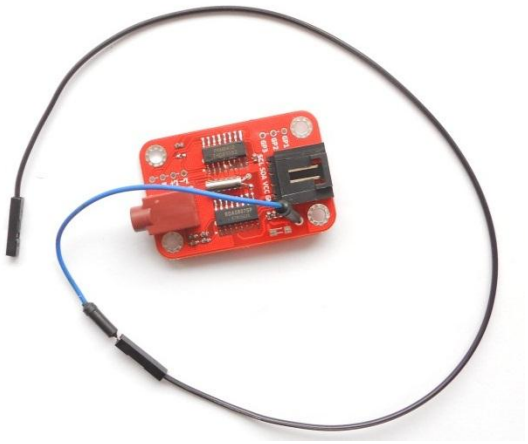
Standard 3.5mm earphone jack.

Antenna

On the picture, the hole marked with Antenna is where you connect or solder the antenna. Any metal line could serves as an antenna. Here is an instruction on how to calculate the length antenna. Theoretically the antenna length for 100Mhz is 75cm.

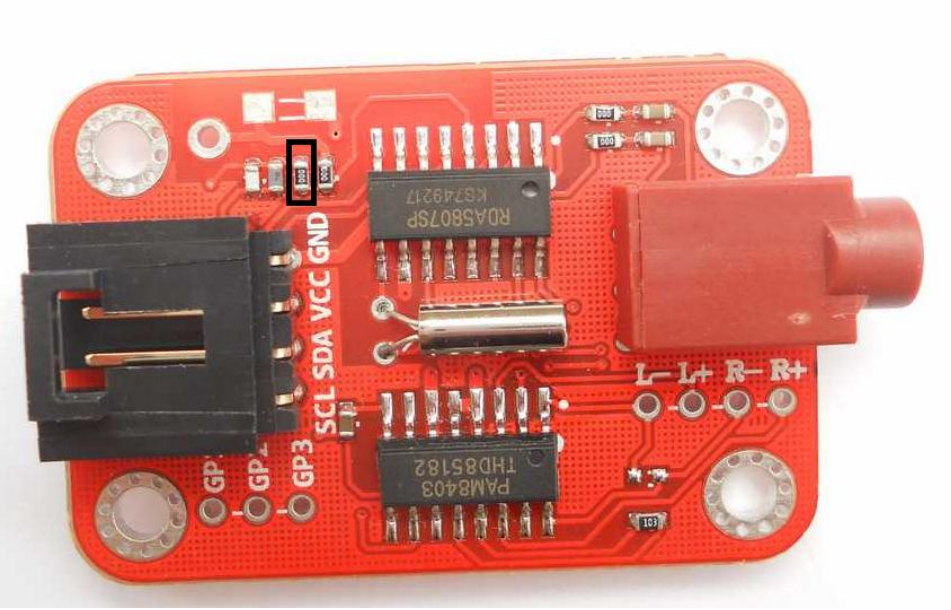
We also connect the antenna to the headphone jack. When you use headphone jack, the line could serve as antenna. It works in the same with those tiny radio devices which only work while you plug in the earphone. If you don't use the headphone and connect loud speaker to the Amplifier Output (shown above in the picture), you have to connect external antenna.

In the test we made a very simple antenna:



Note that plugging in headphone into the jack will disable the Amplifier Output. So you couldn't use headphone line as an antenna while you want the sound from the Amplifier Output.

This module's performance depends much on the antenna. So a good antenna will contribute much to its performance. If you to improve with a good external antenna and want to disable the headphone jack antenna, you could remove the following resistor:



Amplifier Output

There are 4 pads which you could connect your speaker on. L- and R- are not connected together. But you could connect them together while wiring, or bridge the Output Pad to connect them together. The amplifier could drive up to 3W speakers. In the test,

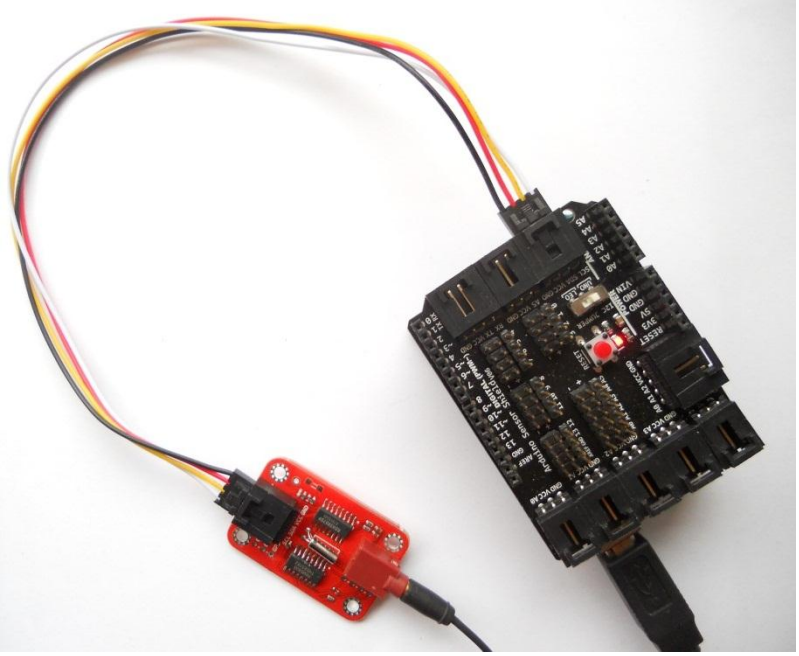
*we tried 5W and it works well (however, we don't recommend to do so). Also, please note that the working jack will disable Amplifier Output. Recommended speaker: **2W3.5ohm Small Speaker***

General output IO

Those ports could be controlled by commands received from I2C. Users could control them output HIGH or LOW. So they might be useful in some application. For example, users could connect LED on it to indicate state.

Connection

If you have our Arduino Sensor Shield V6, you could plug and play as the following picture:



If you don't have sensor shield, you could wire them in the following way:

Arduino		FM Module
GND	----------	GND
5V	----------	VCC
SDA	----------	SDA
SCL	----------	SCL

Code

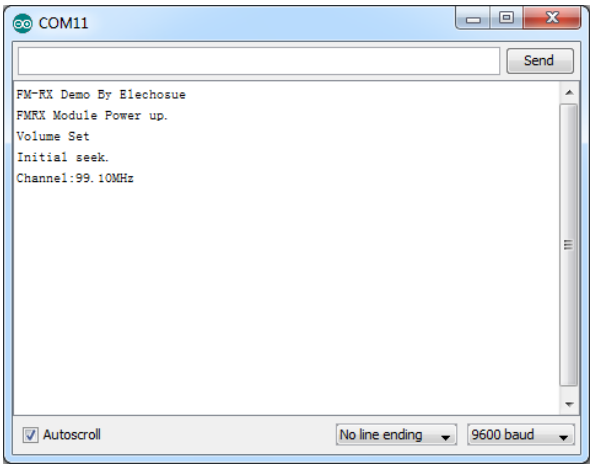
We supply library and example code. The following is example code. For better understanding, I add comment aside the code in blue.

```
/** include library */
#include <FMRX.h>
float channel;
void setup(void)
{
  Serial.begin(9600);
  Serial.print("FM-RX Demo By Elechosue\r\n");
```

```
  /** I2C initial */
  i2c_init();

  fmrx_power();
  fmrx_read_reg(fmrx_reg_r);
  Serial.print("FMRX Module Power up.\r\n");

  /** set volume */
  fmrx_set_volume(15);
  Serial.println("Volume Set");

  /** Receiving signal strength threshold. Those signal whose strength is under this value will be ignored.
        Range:0-127
*/
  fmrx_set_rssi(15);

  /**
    Select a band, parameter:
    BAND_EU:      87-108MHz, Europe
    BAND_US:      87-108MHz, USA
    BAND_JP:      76-91MHz, JAPAN
    BAND_JP_WIDE:  76-108MHz
  */
  fmrx_select_band(BAND_EU);

  channel=fmrx_seek(SEEK_DOWN);
  Serial.println("Initial seek.");
  Serial.print("Channel:");
  Serial.print(channel, 2);
  Serial.println("MHz");
}

void loop(void)
{
  static u8 vol;
  if(Serial.available()>0){
    switch(Serial.read()){

/*
  If received character '1', seek for music station upward
*/
case '1':
        Serial.println("Wait..");
        channel = fmrx_seek(SEEK_DOWN); // seek for radio station
        Serial.println("Seek down.");
        Serial.print("Channel:");
        Serial.print(channel, 2);
        Serial.println("MHz");
        break;

/*
  If received character '2', seek for music station downward
*/
case '2':
        Serial.println("Wait..");
        channel = fmrx_seek(SEEK_UP); // seek for radio station
        Serial.println("Seek up.");
        Serial.print("Channel:");
        Serial.print(channel, 2);
        Serial.println("MHz");
        break;

/*
  If received character '3', turn up volume by 1. Volume value: 0~15
*/
case '3':
        Serial.println("Wait..");
        if(vol < 0x0F){
          vol++;
        }
        fmrx_set_volume(vol); // set the volume
        Serial.print("Volume+:");
```

```
      Serial.println(vol);
      break;

/*
  If received character '4', turn down volume by 1. Volume value: 0~15
*/
case '4':
      Serial.println("Wait..");
      if(vol > 0){
        vol--;
      }
      fmrx_set_volume(vol); // set the volume
      Serial.print("Volume-:");
      Serial.println(vol);
      break;

/**
   If received '&', set new channel. Input data must start with '&' and followed by 4 numbers, the first 3 is the integer part
    (Unit: MHz), the last one is the decimal part. And the channel must between 76MHz and 108Mhz.(eg: &0756 for 75.6MHz, and &0666 is out of range)

  */

case '&':
      u8 i,buf[4];
      float ch;
      i=0;
      delay(30);
      while(Serial.available()&&i<4){
        buf[i]=Serial.read();
        if (buf[i]<= '9' && buf[i]>= '0') {
        i++;}
        else{
        i=0;
        break;
        }
      }
      if (i==4){
        ch = (buf[0]-'0')*100+(buf[1]-'0')*10+(buf[2]-'0')*1+0.1*(buf[3]-'0');
        Serial.println(fmrx_set_freq(ch),2); //set new frequency and print new frequency
      }else{
        Serial.println("Input Error.");
      }
      /** dummy useless character */
      while(Serial.available()){
        Serial.read();
      }
      break;
  }
 }
}
```

Upload the example sketch to Arduino and open the Serial monitor:



You could send command through serial monitor. The following commands are supported:

'1': seek for music station upward

'2': seek for music station downward

'3': turn up volume by 1. Volume value: 0~15

'4': turn down volume by 1. Volume value: 0~15

'&XXXX': set new frequency. Input data must start with '&' and followed by 4 numbers, the first 3 is the integer part (Unit: MHz), the last one is the decimal part. And the channel must between 76MHz and 108Mhz. (eg: &0756 for 75.6MHz, and &0666 is out of range)

Where to buy

FM Radio Receiver Module -- Arduino compatible

Disclaimer and Revisions

The information in this document may change without notice. If you have any problem about it, please visit www.elechouse.com or email to

Revision History

Rev.	Date	Author	Description
A	Oct. 31st , 2012	Wilson	Initial version