Decoding and sending 433MHz RF codes with Arduino and rc-switch

Posted on 14 July 2013

In this tutorial I'll show you how to use an Arduino to decode signals from RF remotes, and re-send them to remotely control some mains switches and a garage door.

April 2017: Updated links for rc-switch, which has moved from code.google.com to github.com

Note: This guide was written for Australia, where it's legal (http://www.acma.gov.au/theACMA/spectrum-at-434-mhz-for-low-powered-devices) to operate low powered devices (25mW) in the 433MHz band without a licence. **Check what's legal in your own country**. If you're transmitting to (or on the same frequency as) a garage door opener / RC toy etc that you bought in your own country, it should be fine, provided you stick to the power limit.

Stuff you need

1. Arduino - I'm using an Arduino Uno Rev3 (https://store.arduino.cc/product/A000066).
2. 433.92Mhz RF Transmitter and Receiver Pair - I got mine from eBay (http://www.ebay.com.au/itm/ws/eBayISAPI.dll?ViewItem&item=400480013376) for the ridiculously cheap price of $1.45:
    - Transmitter Model No: MX-FS-03V (marked XD-FST)
    - Receiver Model No: MX-05V (marked XD-RF-5V)
    - They work just fine - the main problem is that there is no datasheet or

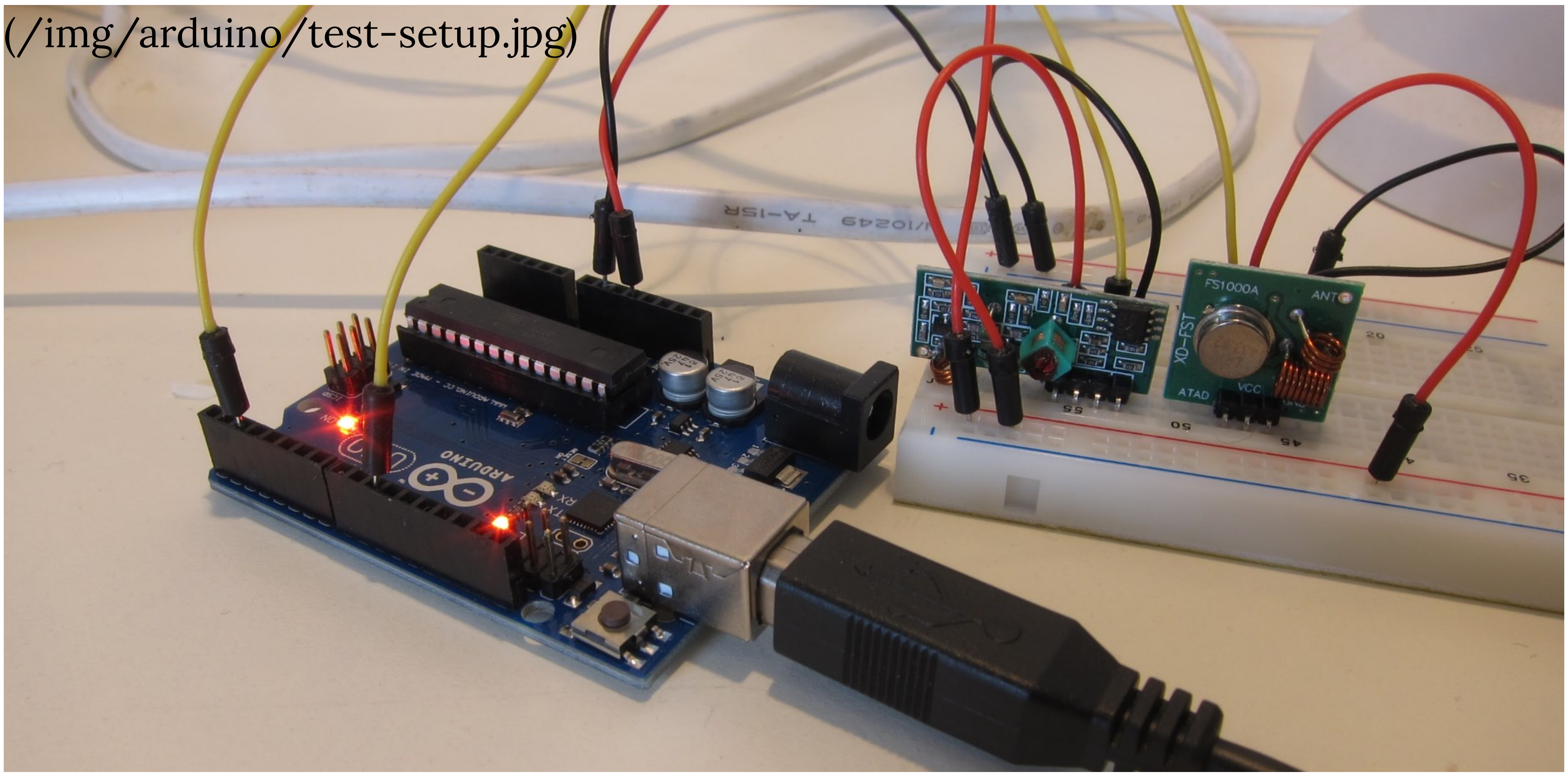
documentation available. Some basic specs are available on hobbycomponents.com (http://forum.hobbycomponents.com/viewtopic.php?f=39&t=1324) but that's about it. Similar modules should also be available at your local electronics shop e.g. Jaycar.
3. Breadboard and jumpers - also available from eBay or Jaycar.
4. rc-switch (https://github.com/sui77/rc-switch/)
5. A remote-controlled garage door, and/or:
6. Some remote controlled mains switches. Mine are "PowerTran" model A0342.
7. A basic knowledge of Arduino and C++

Connect transmitter and receiver to Arduino

The rc-switch wiki has connection diagrams for both the transmitter (https://github.com/sui77/rc-switch/wiki/HowTo_Send) and receiver (https://github.com/sui77/rc-switch/wiki/HowTo_Receive).

Mine looked like this:

(/img/arduino/test-setup.jpg)



Identify your remote controls

Most RF controlled devices should work with rc-switch, but if you want you can open up the remote and check that the encoder chipset is on the list of compatible chipsets (https://github.com/sui77/rc-switch/wiki).

I've got two remote controls - one for my garage door (EV1527 (http://aitendo3.sakura.ne.jp/aitendo_data/product_img/wireless/315MHz-2012/RX315-HT48R/EV1527.pdf) chipset), and one for my RC mains switches (LX2262A-R4 (http://en.chiptrue.com/images/LX2262_en.pdf) chipset).

Here's some pictures of them:

(/img/arduino/garage-opener.jpg)

(/img/arduino/garage-opener-inside.jpg)

(/img/arduino/ms-front.jpg)

(/img/arduino/ms-rear.jpg)

(/img/arduino/ms-remote.jpg)

(/img/arduino/ms-remote-inside.jpg)

Decode signals from your remote controls

rc-switch has built-in functions that sends codewords for certain natively supported devices (https://github.com/sui77/rc-switch/wiki/HowTo_OperateLowCostOutlets) - so if you have one of these devices (I don't) you may be able to skip this step.

Open the rc-switch `ReceiveDemo_Advanced` example sketch. Upload it and open the serial monitor.

Hold your remote near your receiver module and press a button. The Arduino should decode the signal and print the results in the serial monitor. This is what I got for my remote-controlled mains switch when I press the button to turn channel 5 on:

```
Decimal: 3932352 (24Bit)
Binary: 001111000000000011000000
Tri-State: 011000001000
PulseLength: 232 microseconds
Protocol: 1
Raw data: 7244,232,692,124,792,592,328,596,324,596,328,596,324,140,784
,144,788,120,792,136,780,136,788,140,788,128,784,144,796,124,780,140,7
84,596,336,588,968,96,36,104,908,132,1412,68,248,64,28,484,56,
```

The LX2262A-R4 (http://en.chiptrue.com/images/LX2262_en.pdf) uses a 12 tri-state (http://blog.sui.li/2011/04/12/163/) bit codeword comprising 8 address bits followed by 4 data bits. For the tri-state codeword above - `011000001000` - the address is `01100000` (channel 5) and the data/command is `1000` (turn on).

My mains switches can have up to 8 addresses with a separate on and off command for each. By pressing every button and decoding the signals I worked out the codes for all the addresses and commands:

```
Address Bits: 8
Channel 1 = 01110000
Channel 2 = 00110000
Channel 3 = 01010000
Channel 4 = 00010000
Channel 5 = 01100000
Channel 6 = 00100000
Channel 7 = 01000000
Channel 8 = 00000000

Data Bits: 4
Turn On  = 1000
Turn Off = 0000
```

I suspect the address codewords will be the same for all devices of the same make & model - if anyone can confirm this please let me know.

The EV1527 (http://aitendo3.sakura.ne.jp/aitendo_data/product_img/wireless/315MHz-2012/RX315-HT48R/EV1527.pdf) chipset in my garage door remote uses a 24-bit codeword comprising 20 address bits followed by 4 data bits. The codes I got from my garage door remote are:

```
Button 1:
Decimal: 8571080 (24Bit)
Binary: 100000101100100011001000
Tri-State: not applicable
PulseLength: 321 microseconds
Protocol: 1
Raw data: 9964,956,332,312,976,312,976,308,980,304,980,308,980,952,340
,304,980,956,336,188,908,276,728,264,124,168,308,60,24,60,236,88,88,20
4,88,76,80,56,1020,284,440,56,24,40,100,84,12,36,56,

Button    Address              Data
Button 1: 10000010110010001100 1000
Button 2: 10000010110010001100 0100
Button 3: 10000010110010001100 0010
Button 4: 10000010110010001100 0001
```

Write code for your device

rc-switch has built-in functions that sends codewords for certain natively supported devices (https://github.com/sui77/rc-switch/wiki/HowTo_OperateLowCostOutlets) - so If you have one of these devices (I don't) you should be able to use the `RCSwitch::switchOn()` and `RCSwitch::switchOff()` methods in the TypeX example sketches.

If not, you'll need to manually set the PulseLength and Protocol and send raw codes using the `RCSwitch::send()` or `RCSwitch::sendTriState()` methods, as shown below.

Code for the RC mains switch

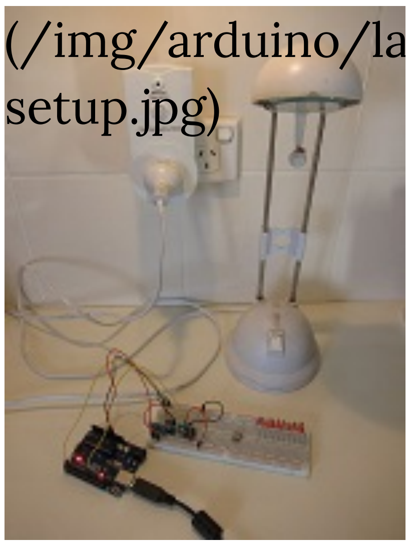
The following code - based on the `SendDemo` sketch - switches one of my remote controlled mains switches on and off every 1 second. Note the pulse length has to be manually set because it differs from the default pulse length for Protocol 1. I've created a function `command()` which accepts channel number and on/off as integer arguments and looks up the corresponding address and data commands specific to my device. For your device you could create a similar function, or just send the raw codes.

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {

 Serial.begin(9600);

 // Transmitter is connected to Arduino Pin #10
 mySwitch.enableTransmit(10);

 // Set Protocol (default is 1, will work for most outlets)
 mySwitch.setProtocol(1);

 // Set pulse length
 // NB Pulse length must be set AFTER Protocol,
 // because setProtocol(1) also sets pulse length = 350
 mySwitch.setPulseLength(232);

 // Optional set number of transmission repetitions.
```

```
  // Mine seem to work with 2, yours may need more
  mySwitch.setRepeatTransmit(2);
}

// Switch channel 8 on and off every 1 second
void loop() {
  command(8, 1);
  delay(1000);
  command(8, 0);
  delay(1000);
}

void command(int nAddress, int nData) {

  // List of device addresses - may be different for your devices
  char* addressCodes[8] = { "01110000", "00110000", "01010000", "00010
000", "01100000", "00100000", "01000000", "00000000" };

  // List of commands - may be different for your devices
  char* dataCodes[2] = { "0000", "1000" };

  // Concatenate the Address and Data codes into a single codeword
  char sendCode[13] = "";
  strcat(sendCode, addressCodes[nAddress-1]);
  strcat(sendCode, dataCodes[nData]);

  // Send the code
  mySwitch.sendTriState(sendCode);
}
```

Here's what my test setup looked like:

(/img/arduino/la
setup.jpg)



Here is a video of the Arduino remotely switching my lamp on and off every 1 second:

Code for the garage door opener

Here is the code which opens and closes my garage door (simulates button 1) every 10 seconds. It also flashes the on-board LED to indicate a command has been sent.

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {

 Serial.begin(9600);

 // Transmitter is connected to Arduino Pin #10
 mySwitch.enableTransmit(10);

 // Optional set pulse length.
 mySwitch.setPulseLength(321);

 // set protocol (default is 1, will work for most outlets)
 // mySwitch.setProtocol(2);

 // Optional set number of transmission repetitions.
 // mySwitch.setRepeatTransmit(15);

 pinMode(13,OUTPUT);

}

void loop() {
 mySwitch.send("100000101100100011001000");
 digitalWrite(13,HIGH);
 delay(500);
 digitalWrite(13,LOW);
 delay(10000);
}
```

Antenna (or not)

The transmitter module seems to have a range of several metres without an antenna. If you require more range, you can add an external antenna by soldering a length of insulated wire to the “ANT” via on the transmitter. Recommended length is $^1/_4$ wavelength, which is approx 17cm @ 433MHz.

Future work and applications

Things I plan to do in future include:

1. Include control of IR devices e.g. TV and air-conditioning remotes.
2. Switch appliances, lights etc on/off via a webpage or phone app.
3. Switch lights, TV, radio etc on and off on a schedule whilst I'm holidays so it looks like I'm still at home. To improve the illusion, the schedule could be varied randomly.
4. Automatically turn off appliances & lights if unintentionally left on.
5. ~~Play pranks on people by randomly opening and closing their garage doors.~~ You shouldn't do this, it's naughty.
6. Port to Raspberry Pi (rc-switch has a rpi port).

References

1. rc-switch - Arduino lib to operate 433/315Mhz devices like power outlet sockets (https://github.com/sui77/rc-switch/)
2. Similar blog posts which taught me what I needed to know:
    - Xose Pérez - Decoding 433MHz RF data from wireless switches (http://tinkerman.eldiariblau.net/decoding-433mhz-rf-data-from-wireless-switches/)
    - Xose Pérez - Decoding 433MHz RF data from wireless switches. The data (http://tinkerman.eldiariblau.net/decoding-433mhz-rf-data-from-wireless-switches-the-data/)
    - Suat Özgür - Low cost RC power sockets (radio outlets) + Arduino (https://sui77.wordpress.com/2011/04/12/163/)
3. Datasheets
    - 433MHz Wireless Modules MX-FS-03V & MX-05 (HCMODU0007) (http://forum.hobbycomponents.com/viewtopic.php?f=39&t=1324)
    - Garage Door Remote - LX2262 (http://en.chiptrue.com/images/LX2262_en.pdf)
    - RC Mains Switches - EV1527 (http://aitendo3.sakura.ne.jp/aitendo_data/product_img/wireless/315MHz-2012/RX315-HT48R/EV1527.pdf)

Questions and Comments

If you have any questions or suggestions on how I can improve this post, please leave a comment 🙂

← **PREVIOUS POST (HTTPS://WWW.LIWEN.ID.AU/DRIP-FREE/)**

NEXT POST → (HTTPS://WWW.LIWEN.ID.AU/PAY-WITH-POINTS/)