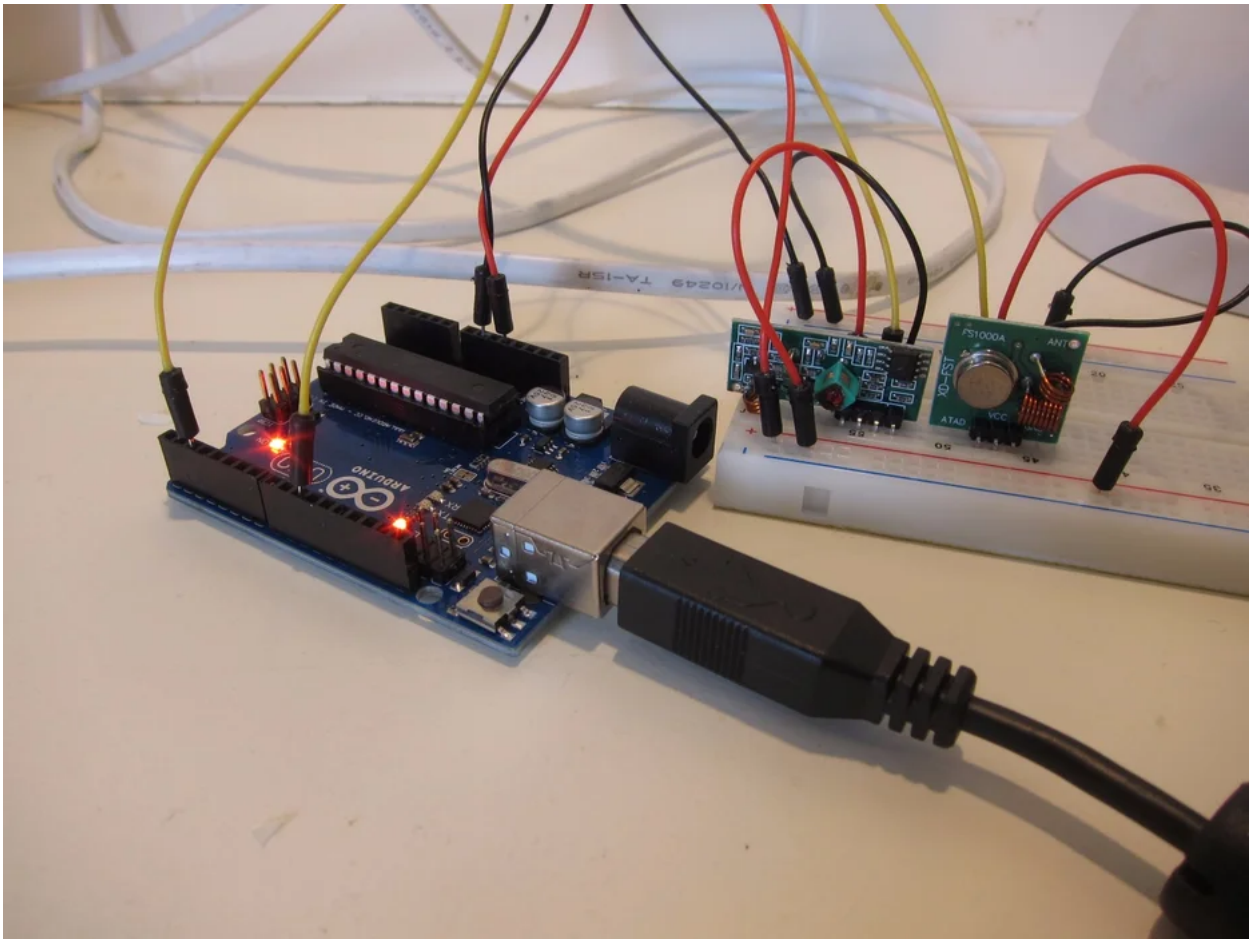
Decoding and Sending 433MHz RF Codes With Arduino and Rc-switch

By liwenyip in CircuitsArduino

Introduction: Decoding and Sending 433MHz RF Codes With Arduino and Rc-switch



Originally published at www.liwen.id.au/arduino-rf-codes

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.

Note: This guide was written for Australia, where it's legal 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.

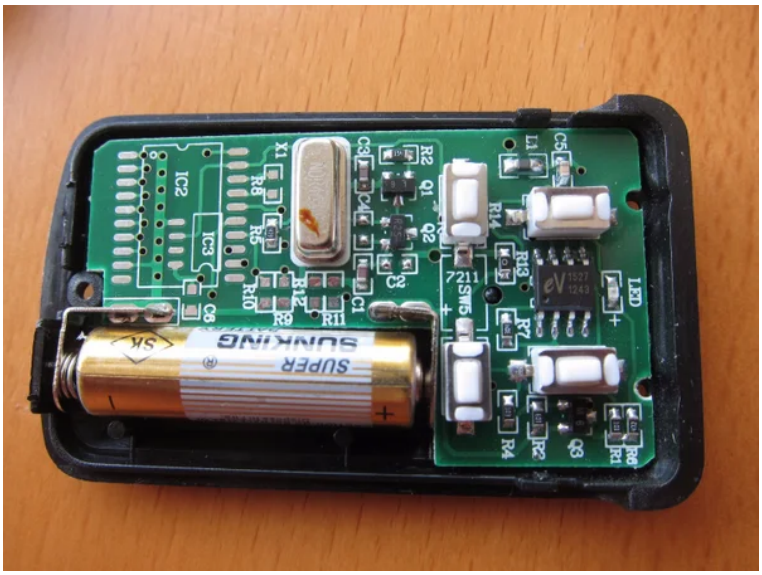
Step 1: Stuff You Need

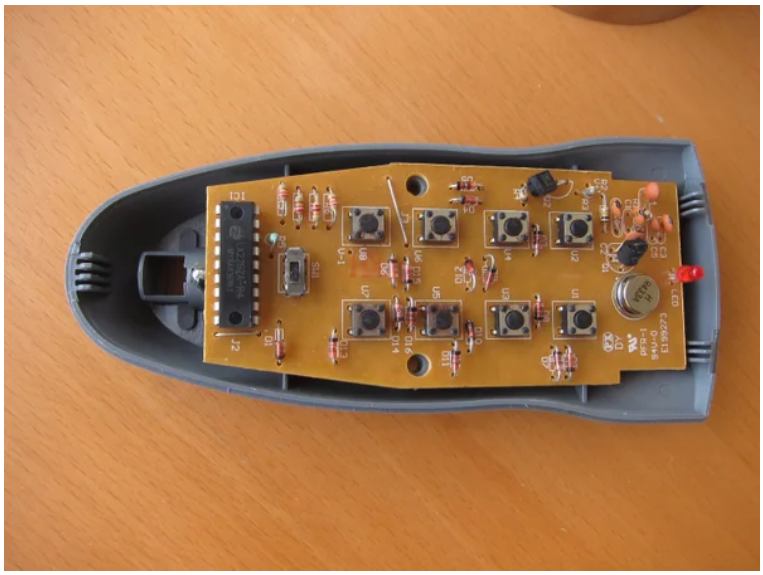
1. Arduino - I'm using an Arduino Uno Rev3.
2. 433.92Mhz RF Transmitter and Receiver Pair - I got mine from eBay for the ridiculously cheap price of $1.45:
   Transmitter Model No: MX-FS-03V (marked XD-FST)
   Receiver Model No: MX-05V (marked XD-RF-5V)
   They work just fine - the main problem is that there is no datasheet or documentation available. Some basic specs are available on hobbycomponents.com 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
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++

Step 2: Connect Transmitter and Receiver to Arduino

The rc-switch wiki has connection diagrams for both the transmitter and receiver.

Step 3: 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.

I've got two remote controls – one for my garage door (EV1527 chipset), and one for my RC mains switches (LX2262A-R4 chipset).

Step 4: Decode Signals From Your Remote Controls

rc-switch has built-in functions that sends codewords for certain natively supported devices – 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: 7244,232,692,124,792,592,328,596,324,596,328,596,324,140,784,144,788,120,792,136,780,136,788,140,788,128,784,144,796,124,780,140,784,596
```

The LX2262A-R4 uses a 12 tri-state 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 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: 9964,956,332,312,976,312,976,308,980,304,980,308,980,952,340,304,980,956,336,188,908,276,728,264,124,168,308,60,24,60,236,88,88,204,88,7

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

Step 5: Write Code for Your Device

rc-switch has built-in functions that sends codewords for certain natively supported devices - 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.

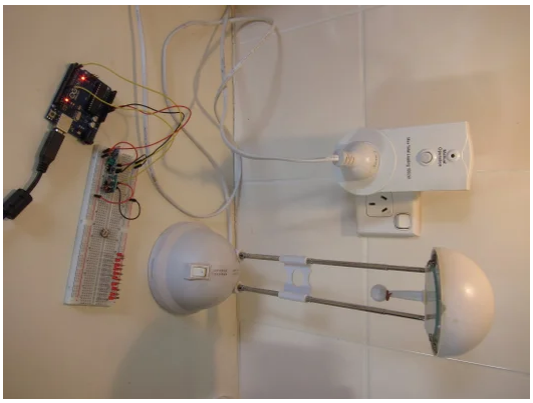
The following code – based on the "SendDemo" sketch – switches one of my remote controlled mains switches on and off every 1 second. Note the pulse length has to be manually set because it differs from the default pulse length for Protocol 1. I've created a function – command() – which accepts channel 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);

 // 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);
}
```

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);
}
```

Step 6: It Works!



Step 7: Notes

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.

Step 8: 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).

Step 9: References

1. https://code.google.com/p/rc-switch/
2. Similar blog posts which taught me what I needed to know:
    1. http://aitendo3.sakura.ne.jp/aitendo_data/product_img/wireless/315MHz-2012/RX315-HT48R/EV1527.pdf
    2. http://en.chiptrue.com/images/LX2262_en.pdf
    3. http://forum.hobbycomponents.com/viewtopic.php?f=39&t=1324
3. Datasheets
    1. http://blog.sui.li/2011/04/12/163/
    2. http://tinkerman.eldiariblau.net/decoding-433mhz-rf-data-from-wireless-switches-the-data/
    3. http://tinkerman.eldiariblau.net/decoding-433mhz-rf-data-from-wireless-switches/