How to receive codes from a wireless remote on your Raspberry Pi.

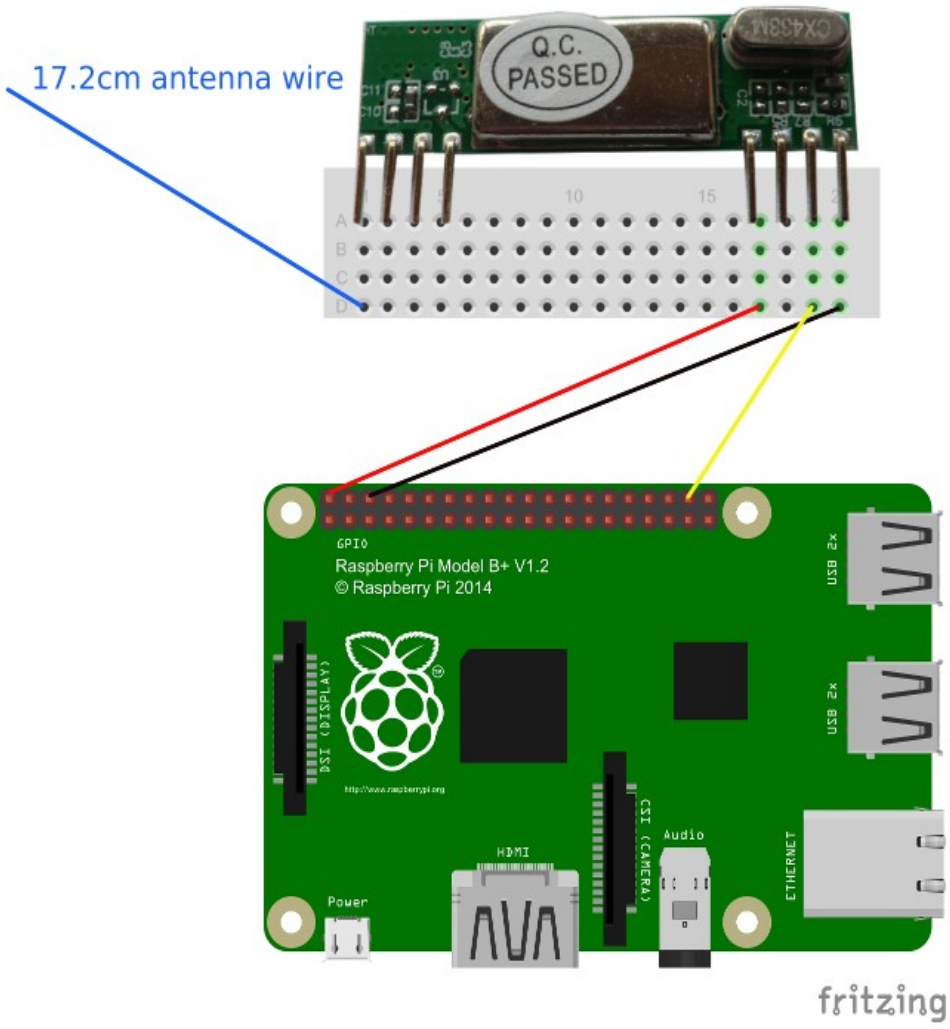
This guide will show you how to receive signals from most remote control gadgets that use the 433MHz (Europe) and 315MHz (North America) bands. The software will only receive AM signals that are transmitted using Manchester / OOK type encoding scheme.

(If you're interested in transmitting & receiving signals at 315MHz, 434MHz, 868MHz or 915MHz using FSK, GFSK, 2-FSK modulation, possibly over long distances, see our other PDF for the TI CC1101 radio modules & Raspberry Pi – www.securipi.co.uk/cc1101.pdf)

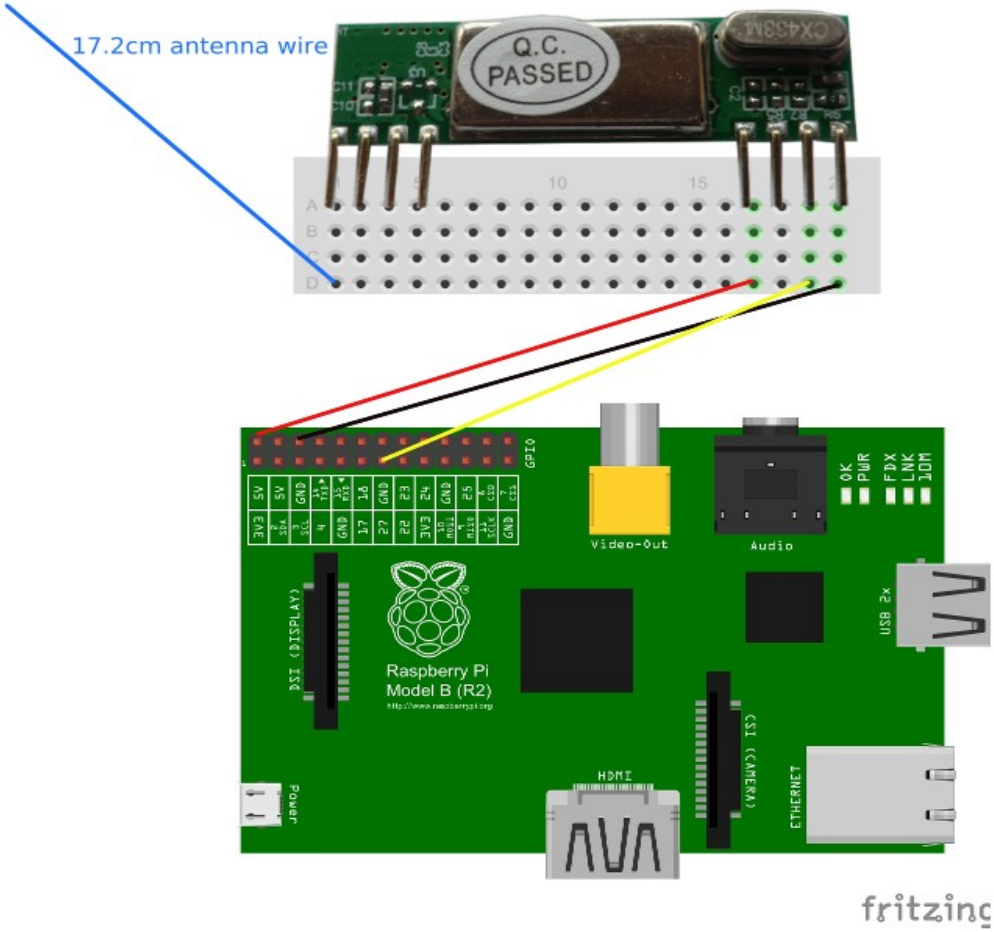
You need a suitable 433MHz or 315MHz receiver board connected to your Pi. The really cheap boards can only receive signals from up to 3 metres away. A decent crystal-controlled shielded super-heterodyne board only costs a few extra £s, and will pick up signals through walls from a 20 metre distance. http://www.ebay.co.uk/itm/161662463558

The receiver boards usually have three pins labelled 5V, GND and Data. The PiGPIO library and Python script looks for a connection on GPIO 20 by default, which is fine if you own a Pi A+, B+ or Pi2. If you have an older model B Pi, with less GPIO pins, you should edit the _433.py Python script so it looks for a connection on RX27 (GPIO27) instead of RX20 (GPIO20).

Here's how the 433Mhz receiver connects to GPIO20 on the A+, B+ or Pi 3



Here's how the receiver connects to a Pi model B on GPIO27



Firstly you need to install PiGPIO

```
wget abyz.me.uk/rpi/pigpio/pigpio.zip
unzip pigpio.zip
cd PIGPIO
make
sudo make install
```

It takes several minutes to “make” the software. Then launch the PiGPIO daemon with

```
sudo pigpiod
```

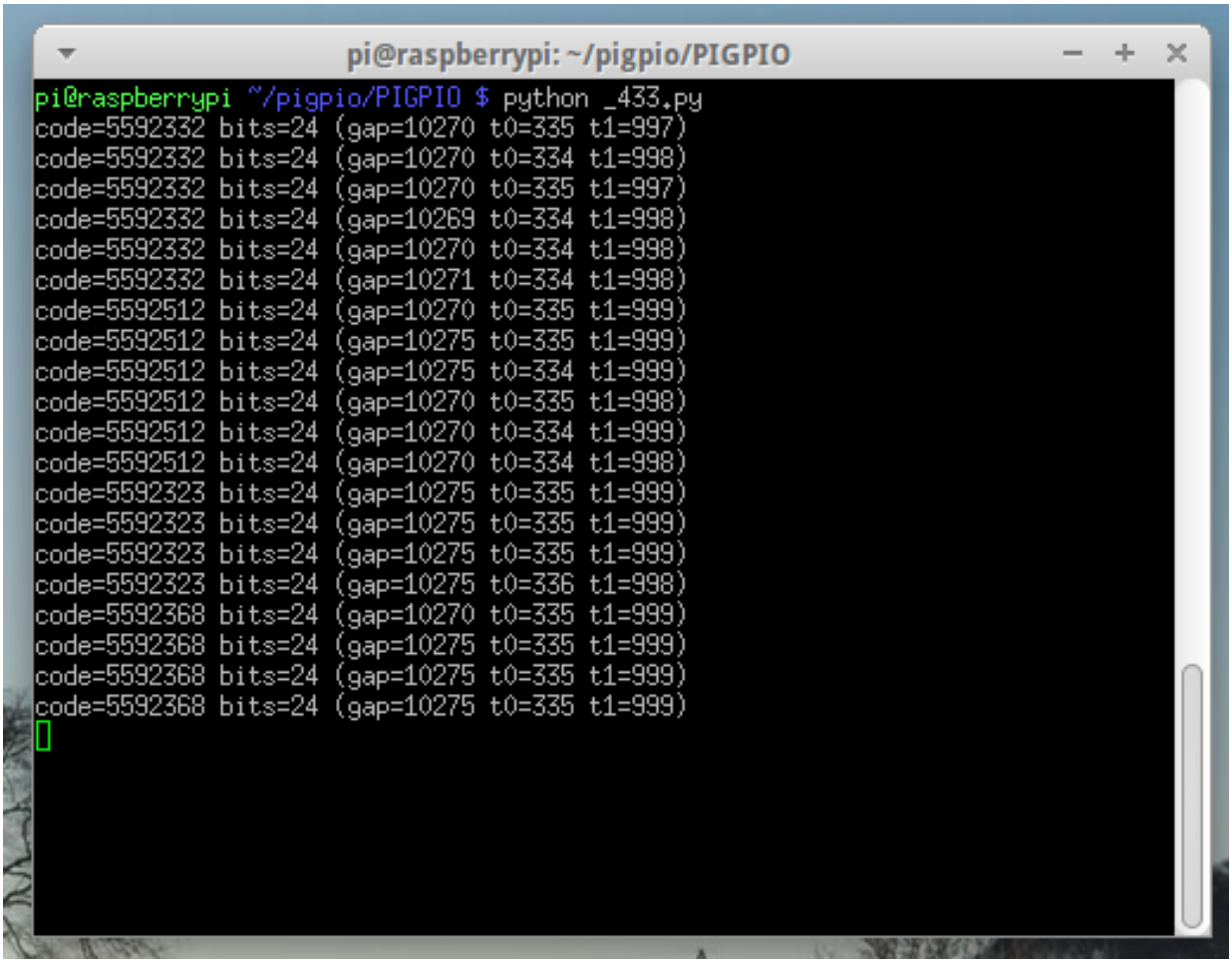
Next we need to grab the Python code for 433 decoding

```
wget abyz.me.uk/rpi/pigpio/code/_433_py.zip
```

```
unzip _433_py.zip
```

```
python _433.py
```

If you now press the buttons on your remote control you should see the codes produced on-screen.



A variety of information is produced. The gap=, t0= and t1= values are useful if you want to re-transmit the codes you've received. The only information needed for receiving is the code= part.

The _433.py program runs for 60 seconds and then quits. Most remotes transmit the same code several times in a row, as a protection against wireless interference.

The codes that start 5592 are all produced by a simple 4 button 433MHZ remote control. I've written them down, along with the button they correspond to, and will use them in a shell script that interacts with a patched version of _433.py that I've called r433.py. I only changed a couple of lines:

```
    for i in range(args-1):
        print("sending {}".format(sys.argv[i+1]))
        tx.send(int(sys.argv[i+1]))
        time.sleep(1)

    tx.cancel() # Cancel the transmitter.

time.sleep(0.5)

rx.cancel() # Cancel the receiver.

pi.stop() # Disconnect from local Pi.
```

you can pull the r433.py and 433test.sh commands down from our server with

```
wget www.securipi.co.uk/r433.py
```

and

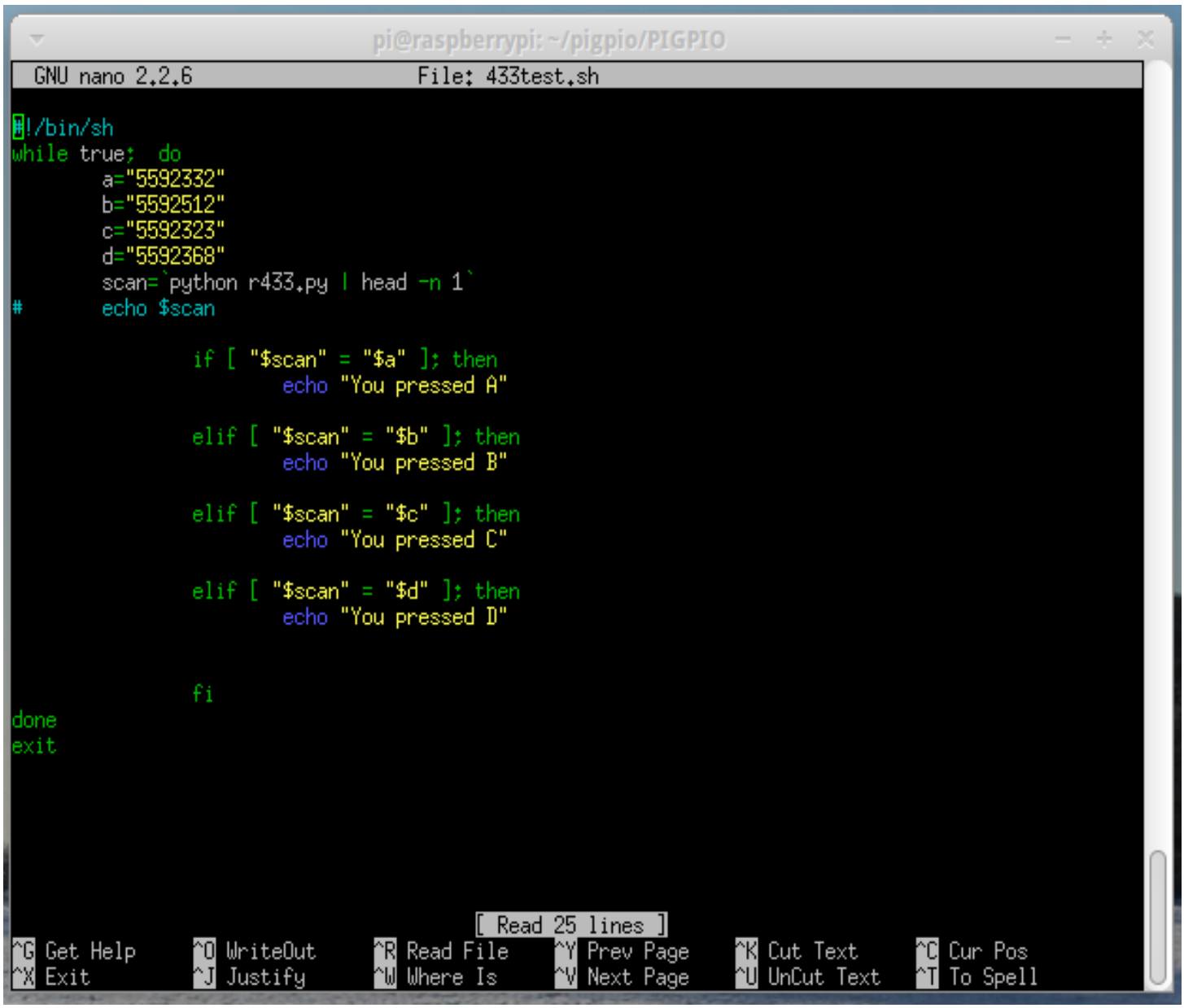
```
wget www.securipi.co.uk/433test.txt
mv 433test.txt 433test.sh
```

make the shell script executable with

```
chmod a+x 433test.sh
```

run the script with

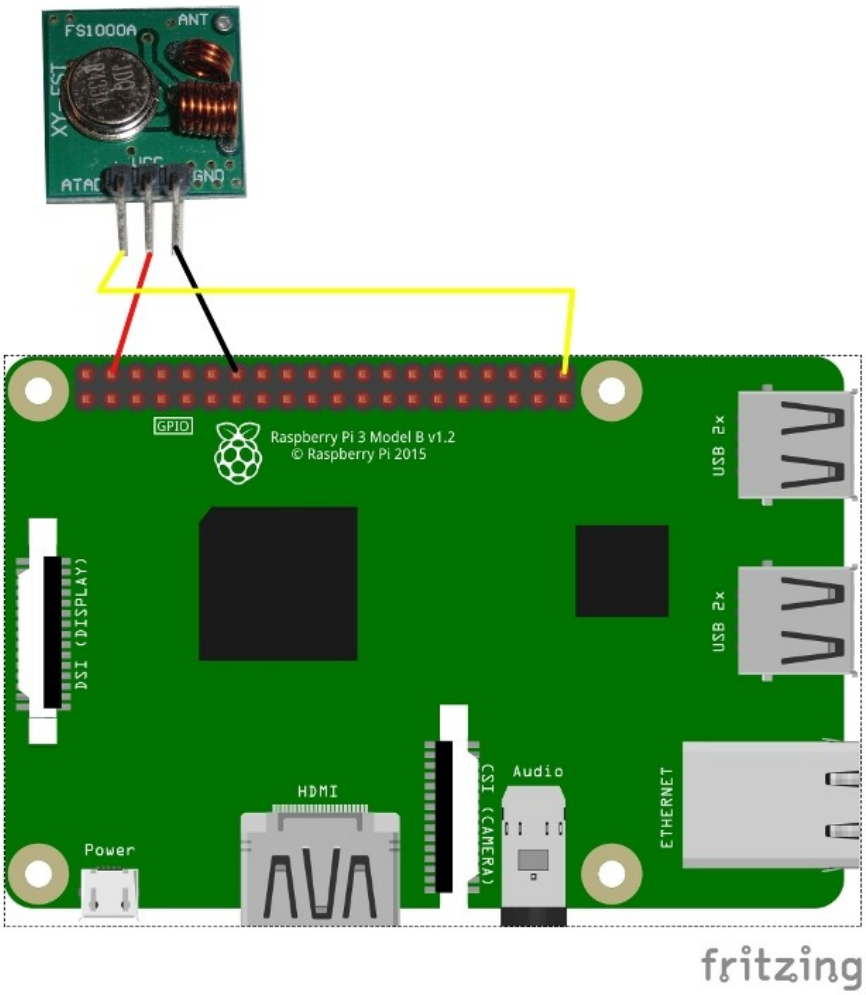
```
./433test.sh
```



Transmitting signals.

When we ran the _433.py receiver software we saw the code, bits, gap and t0 and t1 values for our remote control. If you wrote those down it's possible to use a 433Mhz transmitter module, to replay those codes using the Raspberry Pi.

The transmitter module has 3 pins: 5 volt, ground GND and Data. The data pin attaches to GPIO 21, below the receiver data pin 20. 5 volt and ground also attach to spare pins on the Pi.



We've written a small python script that displays a menu of buttons on the desktop, and currently emulates the 4 button remote control transmitter keyfob seen here:
https://www.amazon.co.uk/dp/B01DOJ9XEW

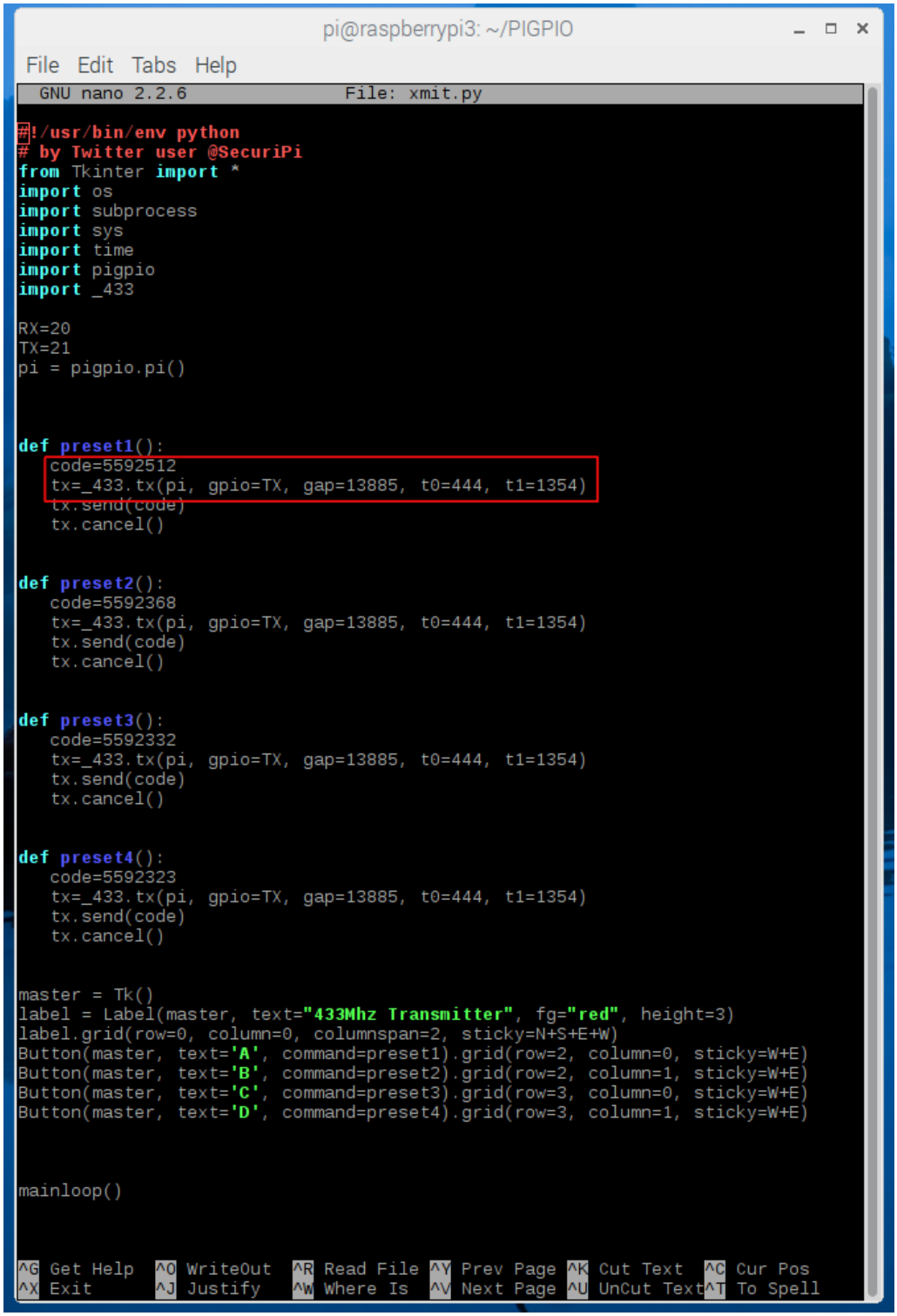
Pull the code down from our server with:

```
wget http://www.securipi.co.uk/xmit.py
```

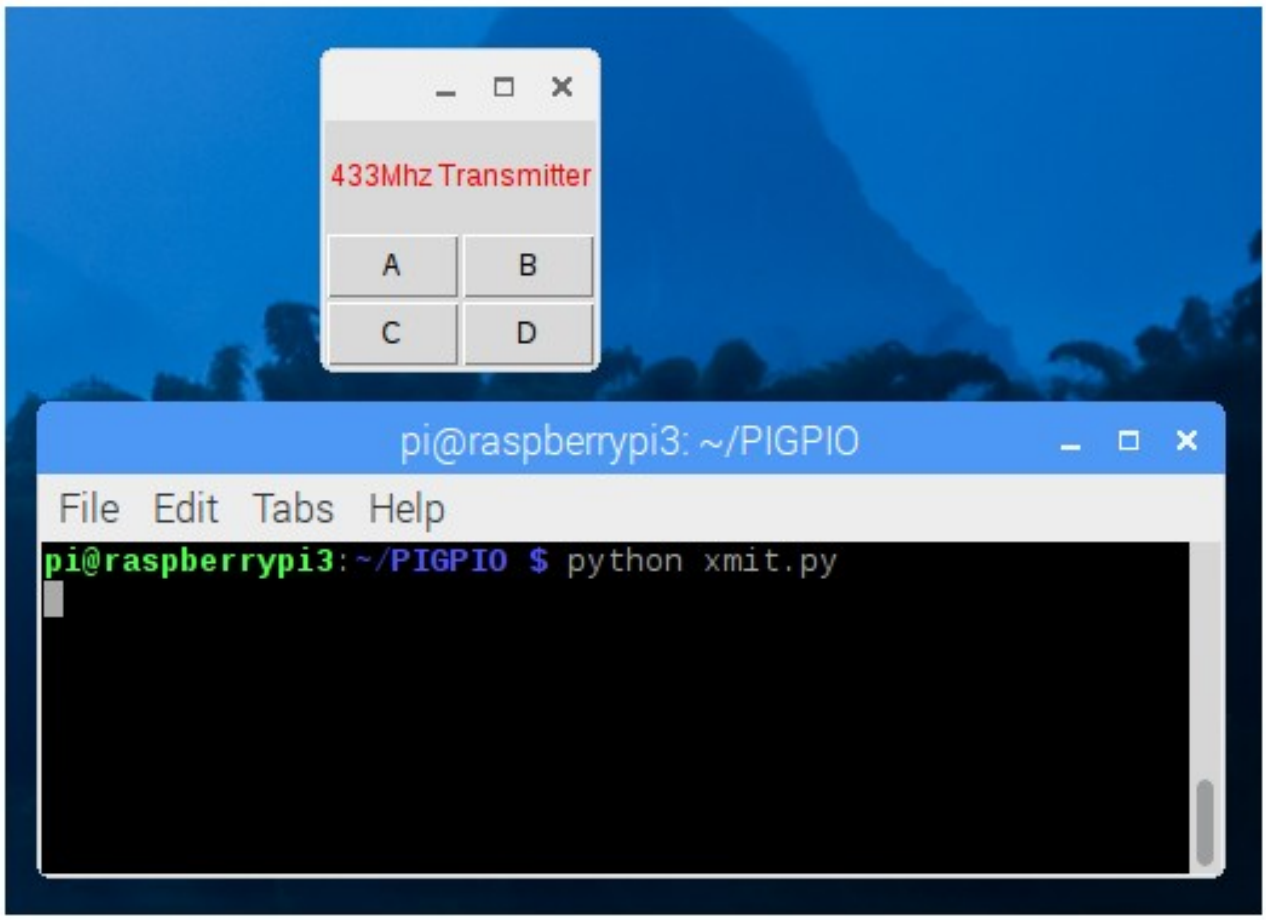
and run it on your Pi's desktop by opening a terminal window and typing

```
python xmit.py
```

You can modify the code to contain your own gadget's codes, gap, T0 and T1 settings.



This is the python script xmit.py running on the Pi's desktop. You can use it to send button codes to the receiver module that came with the 433Mhz keyfob.



Remote control mains sockets.

In this chapter we'll look at using the Raspberry Pi to automatically control a 3 pack of remote control mains sockets, that we've bought from eBay.

We've made a new receiver script called testrx1.py which prints out detected codes from 433.92Mhz wireless gadgets and also stores them in a comma-delimited file called 433log.txt.

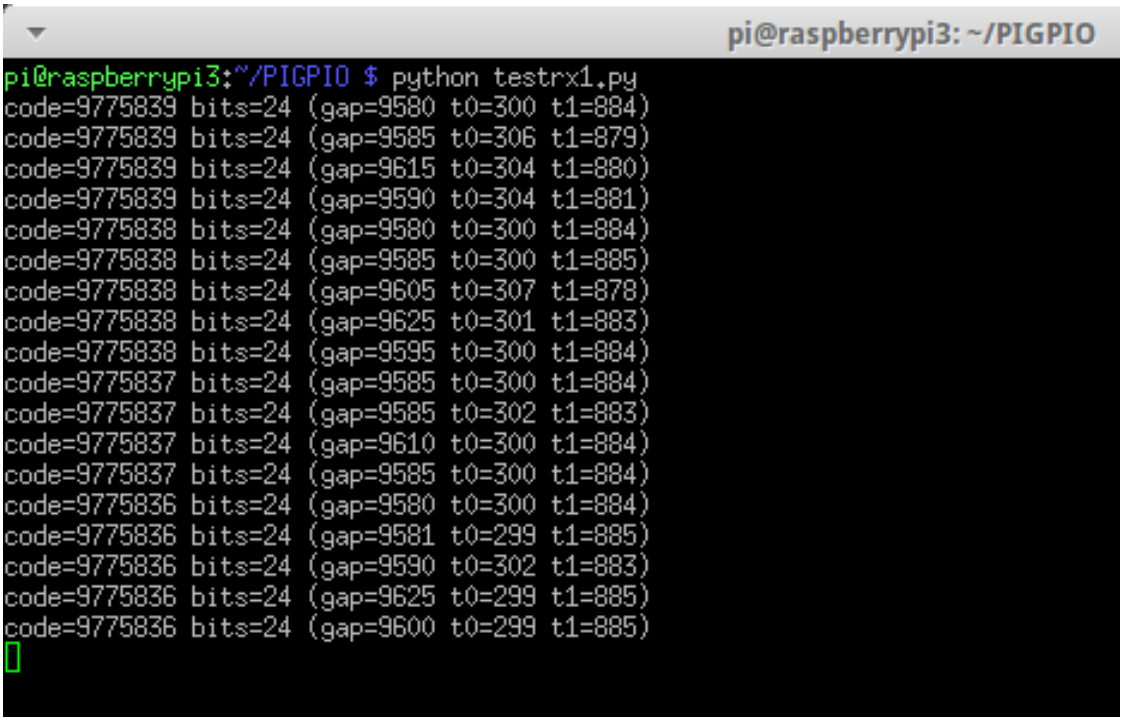
```
import sys
import time
import pigpio
import _433
from time import sleep
from signal import pause

RX=20
pi = pigpio.pi()
def rx_callback(code, bits, gap, t0, t1):
   print("code={} bits={} (gap={} t0={} t1={})".
      format(code, bits, gap, t0, t1))
   file = open("433log.txt","a")
   file.write(str(code)+',')
   file.write(str(bits)+',')
   file.write(str(gap)+',')
   file.write(str(t0)+',')
   file.write(str(t1)+'\n')
   file.close()

_433.rx(pi, gpio=RX, callback=rx_callback)
pause()
```

Assuming you've got already got the PIGPIO daemon running (sudo pigpiod), run the python

receiver script with:

```
python testrx1.py
```

When we press the on and off buttons for sockets 1 and 2 we get this output.

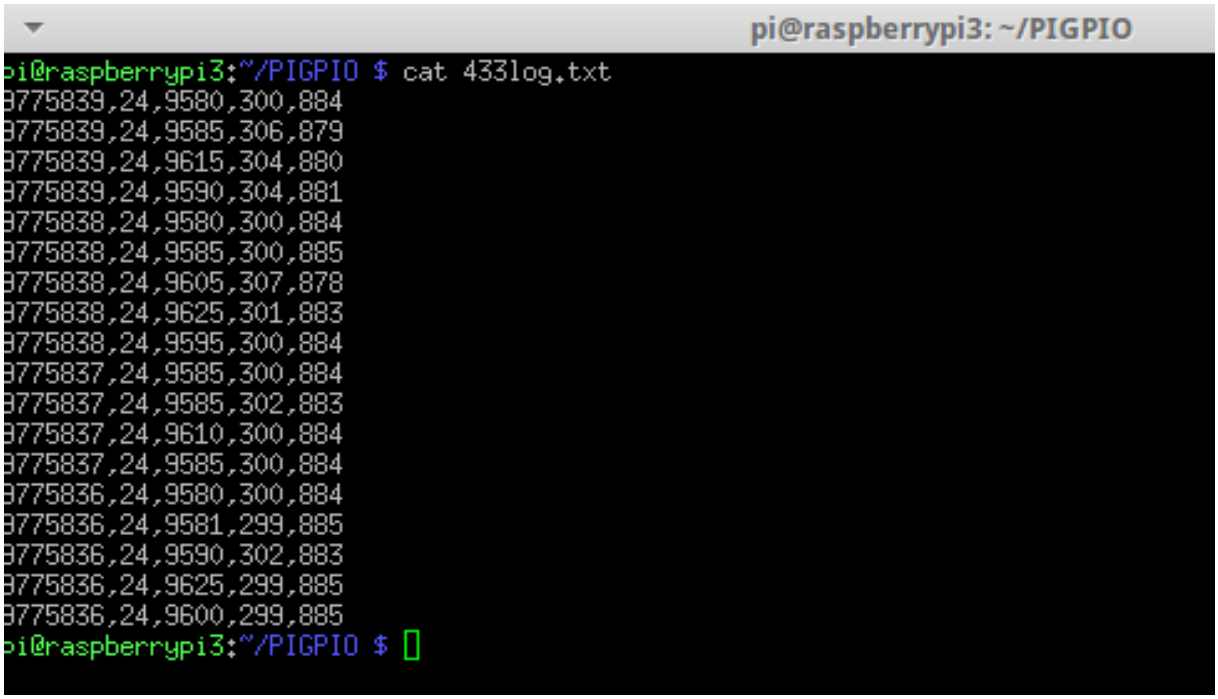


You can inspect the 433log.txt file with

```
cat 433log.txt
```

or

```
nano 433log.txt
```



On the previous two screens we could see that the last two digits of the Code would change each time we pressed a button, so:

9775839 = socket 1 on
9775838 = socket 1 off
9775837 = socket 2 on
9775836 = socket 2 off

The bits were always set to 24. The Gap, T0 and T1 values changed around a little but are basically all near enough that you could choose any set and they'll work with all the Code values.

This next script allows us to turn the sockets on and off at preset times using a Python script on the Raspberry Pi.

```
nano timer.py
```

```
import sys
import time
import pigpio
import _433
import datetime
import os
import subprocess

TX=21
pi = pigpio.pi()

def on1():
   code=9775839
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def off1():
   code=9775838
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def on2():
   code=9775837
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def off2():
   code=9775836
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

while True:
```

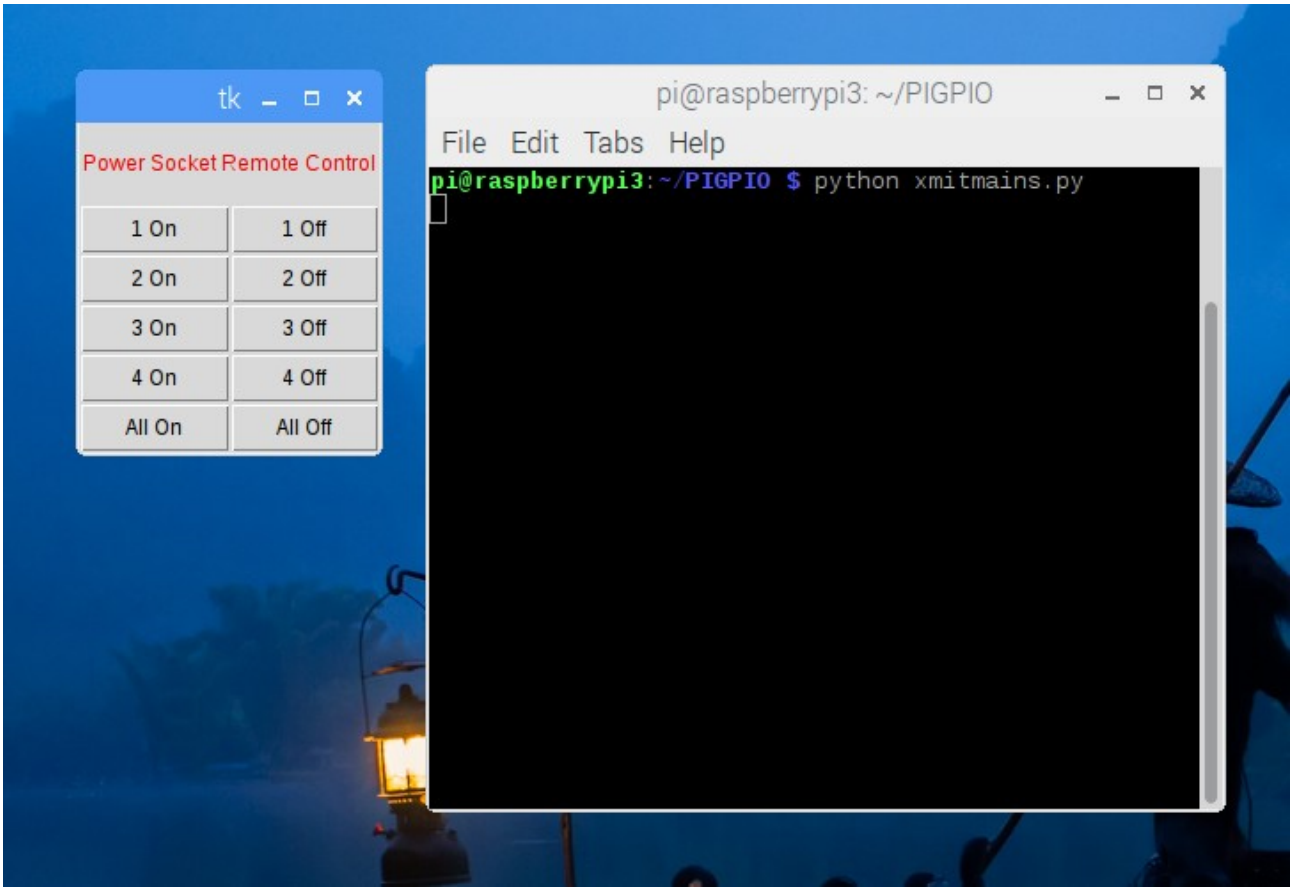
```
   now = datetime.datetime.now()
   if now.hour == 19 and now.minute == 47 and now.second == 00:
      print 'Turning socket 1 on'
      on1()
      time.sleep(2)
   elif now.hour == 19 and now.minute == 49 and now.second == 00:
      print 'Turning socket 1 off'
      off1()
      time.sleep(2)
   if now.hour == 19 and now.minute == 54 and now.second == 00:
      print 'Turning socket 2 on'
      on2()
      time.sleep(2)
   elif now.hour == 19 and now.minute == 54 and now.second == 20:
      print 'Turning socket 2 off'
      off2()
      time.sleep(2)
```

run it with :

```
python timer.py
```

The script turns socket 1 on at 19:47:00 and off again two minutes later. It also turns socket 2 on at 19:54:00 and off again 20 seconds later.

We also made a graphical desktop widget that can control the mains adapters, called xmitmains.py, which looks like this:



```
#!/usr/bin/env python
# by Twitter user @SecuriPi
from Tkinter import *
import os
import subprocess
import sys
import time
import pigpio
import _433

RX=20
TX=21
pi = pigpio.pi()


def on1():
   code=9775839
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def off1():
   code=9775838
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def on2():
   code=9775837
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def off2():
   code=9775836
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def on3():
   code=9775835
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def off3():
   code=9775834
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def on4():
   code=9775831
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def off4():
   code=9775830
```

```
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def allon():
   code=9775826
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

def alloff():
   code=9775825
   tx=_433.tx(pi, gpio=TX, gap=9580, t0=299, t1=885)
   tx.send(code)
   tx.cancel()

master = Tk()
label = Label(master, text="Power Socket Remote Control", fg="red",
height=3)
label.grid(row=0, column=0, columnspan=2, sticky=N+S+E+W)
Button(master, text='1 On', command=on1).grid(row=2, column=0, sticky=W+E)
Button(master, text='1 Off', command=off1).grid(row=2, column=1, sticky=W+E)
Button(master, text='2 On', command=on2).grid(row=3, column=0, sticky=W+E)
Button(master, text='2 Off', command=off2).grid(row=3, column=1, sticky=W+E)
Button(master, text='3 On', command=on3).grid(row=4, column=0, sticky=W+E)
Button(master, text='3 Off', command=off3).grid(row=4, column=1, sticky=W+E)
Button(master, text='4 On', command=on4).grid(row=5, column=0, sticky=W+E)
Button(master, text='4 Off', command=off4).grid(row=5, column=1, sticky=W+E)
Button(master, text='All On', command=allon).grid(row=6, column=0,
sticky=W+E)
Button(master, text='All Off', command=alloff).grid(row=6, column=1,
sticky=W+E)



mainloop()
```

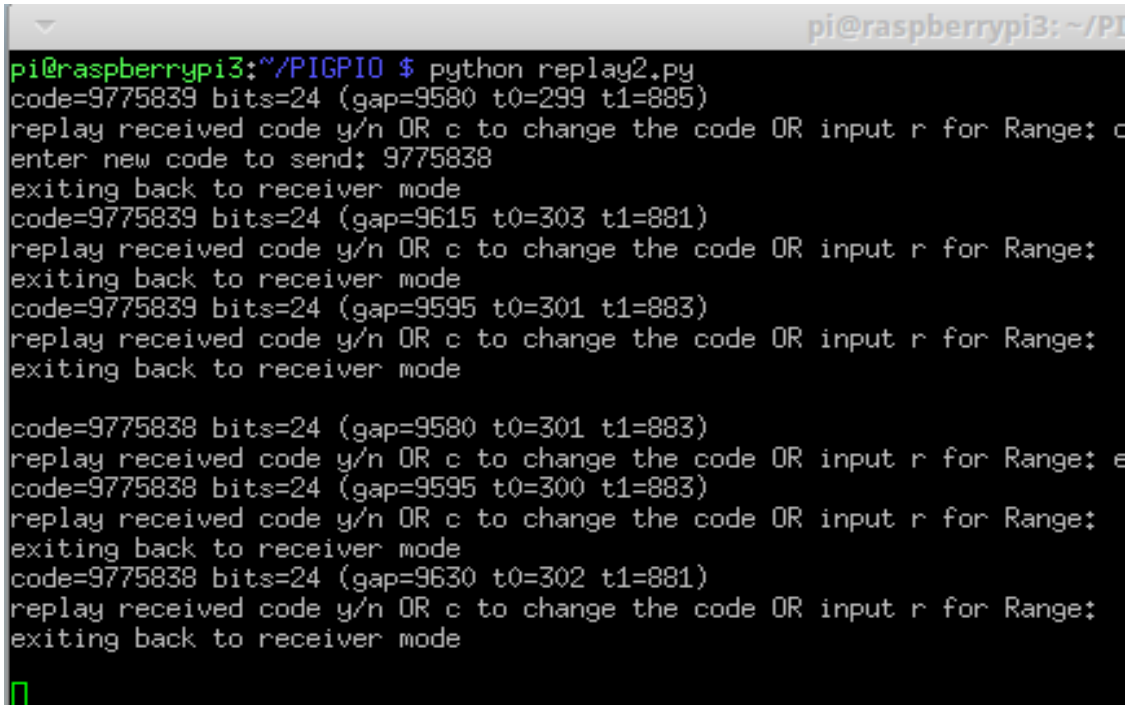
Replaying a captured code

It's possible to have the Pi listen out for wireless codes and then offer to replay the code, without having to type in all the parameters separately. I've also added the option of replaying a different code, but with the same Gap, T0 and T1 values. There's also an option to transmit codes in a range, with a step value – so I can turn my mains sockets all off or on, 1 at a time. The script is called replay2.py

```
#!/usr/bin/env python
# by Twitter user @SecuriPi
import sys
import time
import pigpio
import _433
from time import sleep
from signal import pause

TX=21
RX=20
pi = pigpio.pi()

def rx_callback(code, bits, gap, t0, t1):
   if bits == 24:
      print("code={} bits={} (gap={} t0={} t1={})".
         format(code, bits, gap, t0, t1))
      # Write info to 433log.txt file
      file = open("433log.txt","a")
      file.write(str(code)+',')
      file.write(str(bits)+',')
      file.write(str(gap)+',')
      file.write(str(t0)+',')
      file.write(str(t1)+'\n')
      file.close()
      option = raw_input('replay received code y/n OR c to change the
code OR input r for Range: ')
      if option == 'y':
         tx=_433.tx(pi, gpio=TX, gap=gap, t0=t0, t1=t1)
         tx.send(code)
         tx.cancel()
         pi.stop
      if option == 'c':
         code = input('enter new code to send: ')
         tx=_433.tx(pi, gpio=TX, gap=gap, t0=t0, t1=t1)
         tx.send(code)
         tx.cancel()
      if option == 'r':
         code = input('enter lowest code: ')
         endCode = input('enter highest code: ')
         stepCode = input('enter step value: ')
         while code <= endCode:
            tx=_433.tx(pi, gpio=TX, gap=gap, t0=t0, t1=t1)
            tx.send(code)
            tx.cancel()
            print('sending code '+(str(code)))
```

```
            code += stepCode
            time.sleep(1)
      else:
         print "exiting back to receiver mode"

_433.rx(pi, gpio=RX, callback=rx_callback)
pause()
```

You have to be careful when replaying codes to keep the receiver and transmitter modules more than 6 inches apart, otherwise the Gap, T0 and T1 values will inexplicably grow each time you replay the codes. (You can always look in the 433log.txt file for the original values of Gap, T0 and T1).



The following equipment is tested & works fine with our scripts:
Lloytron MIP range of wireless Doorbell pushes, PIR movement and Magnetic Door Sensors.
Status 3 pack mains remote controlled sockets
433Mhz version of 4 button pocket keyfob transmitter/receiver for Pi and Arduino projects.

All the scripts in this PDF also work with the 315MHz equivalents used in North America.

Notes & Useful Sales Links.

Make sure you remember to run the PIGPIO daemon each time you start the Pi (sudo pigpiod), or add it to your /etc/rc.local file, so it launches each time you power up the Pi.

Buy 433MHz if you're in the UK or Europe. Buy 315MHz if you're in North America.

You can buy the really good 433MHz receiver board from our eBay shop here:
http://www.ebay.co.uk/itm/433Mhz-Shielded-Low-Noise-Wireless-Receiver-Board-for-Raspberry-Pi-Arduino-/161662463558?roken=cUgayN

We also sell a 433Mhz transmitter/receiver pair here:
http://cgi.ebay.co.uk/ws/eBayISAPI.dll?ViewItem&item=161594608707#ht_500wt_1180
(the transmitter is great, the receiver isn't anywhere near as good as the shielded low-noise version)

You can buy the 4 button keyfob 433Mhz version here:
http://cgi.ebay.co.uk/ws/eBayISAPI.dll?ViewItem&item=131767939665#ht_609wt_1165

The 315MHz version of the shielded receiver board for North America is here in our eBay shop:
http://www.ebay.co.uk/itm/315Mhz-Shielded-Low-Noise-Wireless-Receiver-Board-for-Raspberry-Pi-Arduino-/162023269288

The 315Mhz version of the 4 button keyfob for North America is here:
http://www.ebay.co.uk/itm/4-channel-315MHz-wireless-remote-control-for-Raspberry-Pi-Arduino-UK-Stock-/161651775043

Also, our **433MHz Internet Doorbell kits for Raspberry Pi** (£20) come with the 433MHz Shielded receiver board and a wireless 433MHz door push that emails photos of callers to your phone:

Black https://www.amazon.co.uk/dp/B00UL9QBJ4
White https://www.amazon.co.uk/dp/B00UL3FRGS
No bell push https://www.amazon.co.uk/dp/B00T4DUYMS

We also do some **Home Security kits for Raspberry Pi:**

SecuriPi PIR sensor alarm kit, photos of intruders to your phone £11.99
https://www.amazon.co.uk/dp/B00GOPJJWU

Magnetic sensor window/door alarm kit for Raspberry Pi £12.99
https://www.amazon.co.uk/dp/B00DBDT6TY

48 Infra Red LED lighting kit for Raspberry Pi PiNoIR camera £15.99
https://www.amazon.co.uk/dp/B015OA6VAI

And a Home Security kit with PIR and Magnetic sensor that links supplied Arduino to your PC or Mac for £18.99 : https://www.amazon.co.uk/dp/B00P9UTF0W

See all our latest blog posts at www.securipi.co.uk
Manchester encoding info : https://en.wikipedia.org/wiki/Manchester_encoding
If you found this document useful, please follow @securipi on Twitter. Thank you.