

Wireless doorbell & PIR receiver project.

This kit will allow your Raspberry Pi to listen out for the 433MHz radio signal transmitted by a variety of Lloytron MIP wireless doorbells (tested with latest MIP3 door push 24/11/2022), PIR sensors & Magnetic door sensors. It will also detect signals from Driveway Patrol & Digiteck driveway alarms.

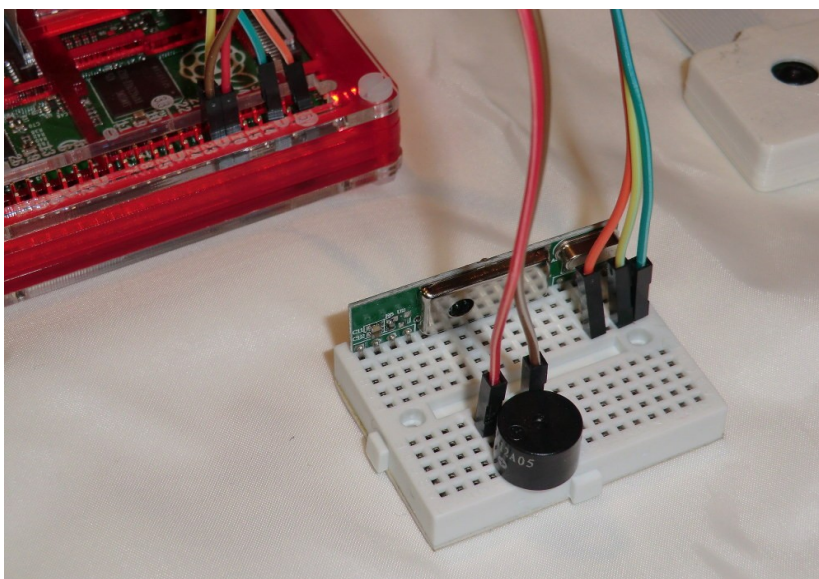
The Pi can differentiate between the unique codes from each device & send you an email stating which has been triggered. If you have a Raspberry Pi camera or USB webcam pointing at the doorway, you can get a photo of the visitor emailed to your mobile phone.



Kit includes: 433MHz receiver board , mini breadboard, three IDC connection wires (all pictured), 5 volt buzzer & 2 connection wires, DVD containing instructions PDF, various shell scripts.

<https://www.amazon.co.uk/dp/B00UL9QBJ4> & <https://www.ebay.co.uk/itm/131419470115>

Doesn't include: Raspberry Pi, red Pi case, Lloytron doorbell pack or Driveway Alarm.

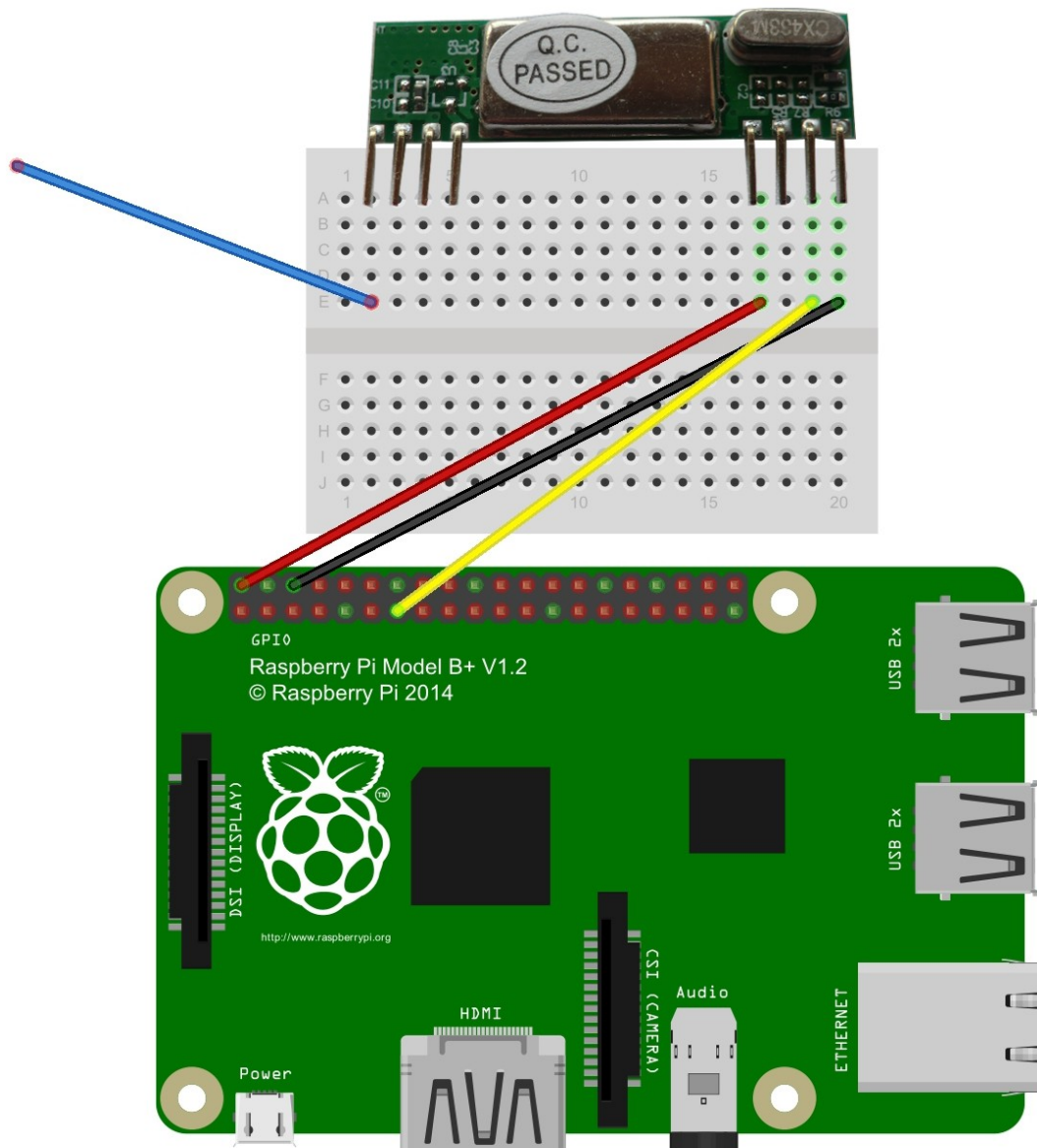


Here's how the 433MHz receiver attaches to a Raspberry Pi 4, 3 or Pi Zero WH.

The blue wire in the picture is an optional antenna wire, which should be 17.3cm long & can be coiled around a biro so it takes up less space. We didn't need to use the external antenna wire in any of our tests. You can expect the radio module to detect the door bell from 20 metres away.

The colours of all the wires we've sent you will be different, just make sure connect the correct points together.

The red wire connects to 5 volt + on the Pi, the black wire connects to GND & the yellow wire connects to GPIO 27 (which in WiringPi-speak is confusingly referred to as GPIO 2)



fritzing

Install WiringPi library

The WiringPi library now comes included with more modern versions of Raspberry PI OS. Issue these commands at the \$ terminal prompt to check you have it:

```
gpio -v
gpio readall
```

You'll now see a list of all the GPIO assignments for Wiring Pi. I've highlighted our GPIO pin 27.

```
pi@raspberrypi: ~/433Utils/RPi_utils
pi@raspberrypi ~/433Utils/RPi_utils $ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2   | 8   | SDA,1    | IN    | 1 | 3   | 4   |      | 5v       |     |     | |
| 3   | 9   | SCL,1    | IN    | 1 | 5   | 6   |      | 5v       |     |     |
| 4   | 7   | GPIO, 7   | IN    | 1 | 7   | 8   | 1   | ALT0     | TxD  | 15  | 14  |
|     |     | 0v       |       |   | 9   | 10  | 1   | ALT0     | RxD  | 16  | 15  |
| 17  | 0   | GPIO, 0   | IN    | 0 | 11  | 12  | 0   | IN       | GPIO, 1 | 1  | 18  |
| 27  | 2   | GPIO, 2   | IN    | 0 | 13  | 14  |     |          | 0v     |     |     |
| 22  | 3   | GPIO, 3   | IN    | 0 | 15  | 16  | 0   | IN       | GPIO, 4 | 4  | 23  |
|     |     | 3,3v     |       |   | 17  | 18  | 0   | IN       | GPIO, 5 | 5  | 24  |
| 10  | 12  | MOSI     | IN    | 0 | 19  | 20  |     |          | 0v     |     |     |
| 9   | 13  | MISO     | IN    | 0 | 21  | 22  | 0   | IN       | GPIO, 6 | 6  | 25  |
| 11  | 14  | SCLK     | IN    | 0 | 23  | 24  | 1   | IN       | CE0    | 10  | 8   |
|     |     | 0v       |       |   | 25  | 26  | 1   | IN       | CE1    | 11  | 7   |
| 0   | 30  | SDA,0    | IN    | 1 | 27  | 28  | 1   | IN       | SCL,0  | 31  | 1   |
| 5   | 21  | GPIO,21  | IN    | 1 | 29  | 30  |     |          | 0v     |     |     |
| 6   | 22  | GPIO,22  | IN    | 1 | 31  | 32  | 0   | IN       | GPIO,26 | 26  | 12  |
| 13  | 23  | GPIO,23  | IN    | 0 | 33  | 34  |     |          | 0v     |     |     |
| 19  | 24  | GPIO,24  | IN    | 0 | 35  | 36  | 0   | IN       | GPIO,27 | 27  | 16  |
| 26  | 25  | GPIO,25  | IN    | 0 | 37  | 38  | 0   | IN       | GPIO,28 | 28  | 20  |
|     |     | 0v       |       |   | 39  | 40  | 0   | IN       | GPIO,29 | 29  | 21  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
pi@raspberrypi ~/433Utils/RPi_utils $
pi@raspberrypi ~/433Utils/RPi_utils $
pi@raspberrypi ~/433Utils/RPi_utils $
```

On a Raspberry Pi 4 I had to issue these commands to make gpio readall work correctly:

```
wget https://www.securipi.co.uk/wiringpi-2.61-1-armhf.deb
sudo dpkg -i wiringpi-2.61-1-armhf.deb
```

Then do this again:

```
gpio -v
gpio readall
```

Install 433Utils.

Next, we need to install a set of 433Mhz radio utilities written in the C programming language. We use C for this part, because Python or a shell script are too slow. When entering the commands below, remember that Linux & the Raspberry Pi are case-sensitive, so 433Utils isn't the same as 433utils.

```
git clone --recursive https://github.com/ninjablocks/433Utils
cd 433Utils/RPi_utils
```

With the utils installed, we now need to patch them with some code from our website, so we can make RFSniffer interact with our shell scripts. First move the RFSniffer.cpp source code to a different filename & then download our 433.zip file. Finally compile everything with Make.

```
mv RFSniffer.cpp oldRFSniffer.cpp
wget http://www.securipi.co.uk/433.zip
tar xvf 433.zip
chmod a+x *.sh
make
ls -al
```

You should now see a list of files. The test.sh file will let you figure out the codes being sent by your various Lloytron devices and driveway alarms.

```
sudo ./test.sh
```

Now press your doorbell or activate your PIR. If the code being transmitted matches the valid string in the test.sh file, then you get a good read message and the numeric code of your device is displayed, but if the code doesn't match you get a BAD READ message, and the numeric code received is displayed.

Write down the numerical codes for each of your devices on a piece a paper for use later on.

You can stop the test.sh script from running by holding down Ctrl-C on the keyboard. You can edit the test.sh script, so it says your doorbell or PIR is valid like this:

```
nano test.sh
```

look for the line that says valid="457624" & change the number to your code.

Ctrl-O + Enter to save the file & then Ctrl-X to exit. Now run the test.sh file again

```
sudo ./test.sh
```

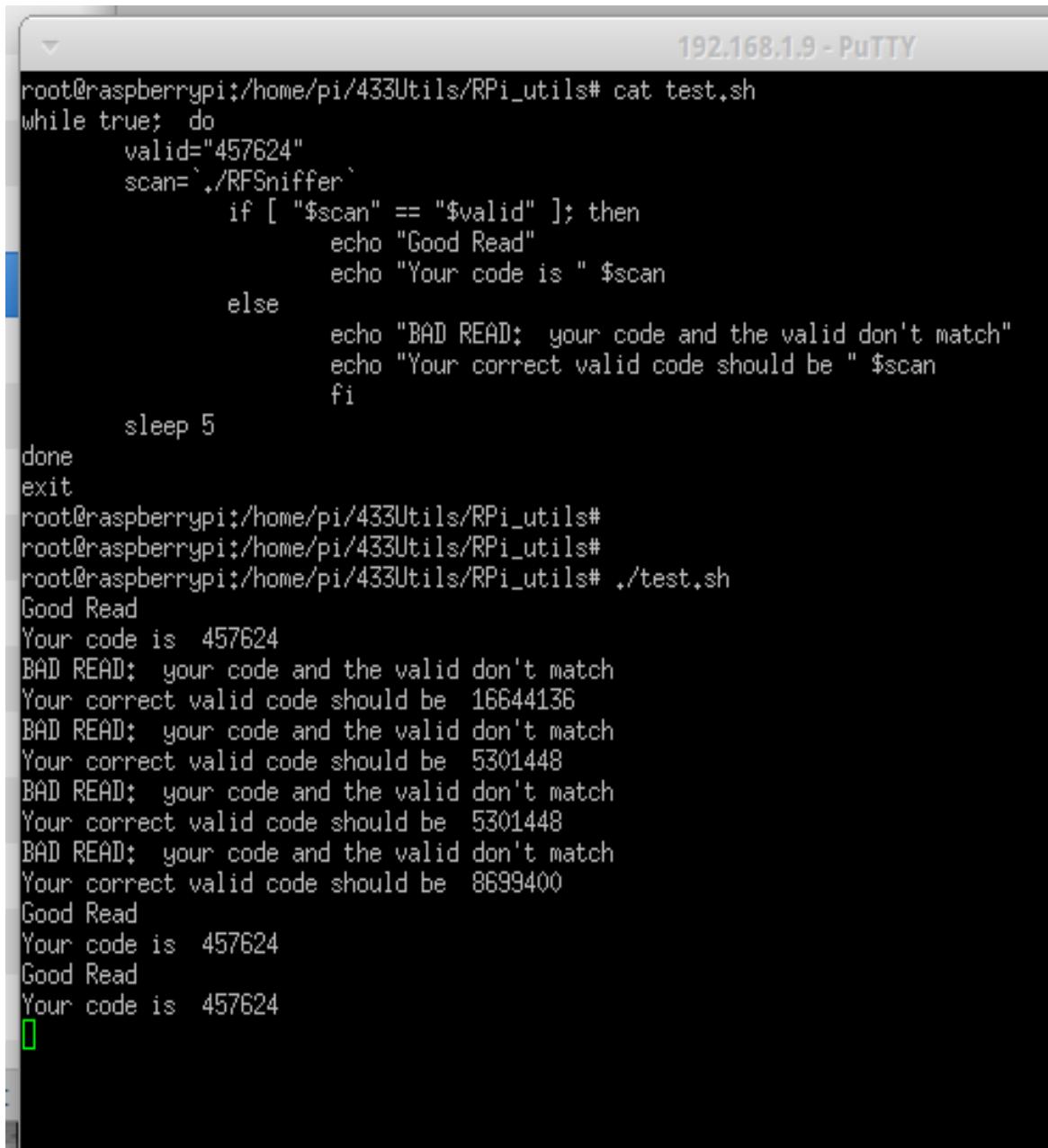
Now, when the doorbell or PIR is activated, you should see your code as the valid one.

Ctrl-C to quit the test.sh script.

Photo of test.sh script & the output it produces:

- | | |
|-----------------------------------|-----------------------------|
| 1. Lloytron Doorbell 1 | 457624 (current valid code) |
| 2. Lloytron Doorbell 2 | 16644136 |
| 3. Lloytron Magnetic door sensor | 5301448 |
| 4. Lloytron PIR movement detector | 8699400 |

Each of our Lloytron devices always send the same code. Your devices will have different codes.

A screenshot of a terminal window titled "192.168.1.9 - PuTTY". The terminal shows the execution of a script named "test.sh". The script is a while loop that runs indefinitely. It sets a "valid" code to "457624" and uses a command "scan=`./RFSniffer`" to get a new code. It then compares the scanned code to the valid code. If they match, it prints "Good Read". If not, it prints "BAD READ: your code and the valid don't match" followed by "Your correct valid code should be " and the valid code. The output shows the script running and detecting several "BAD READ" errors with various codes (16644136, 5301448, 8699400) before finally detecting a "Good Read" with the code 457624. The prompt at the bottom is a green cursor.

```
root@raspberrypi:/home/pi/433Utils/RPi_utils# cat test.sh
while true; do
    valid="457624"
    scan=`./RFSniffer`
    if [ "$scan" == "$valid" ]; then
        echo "Good Read"
        echo "Your code is " $scan
    else
        echo "BAD READ: your code and the valid don't match"
        echo "Your correct valid code should be " $scan
    fi
    sleep 5
done
exit
root@raspberrypi:/home/pi/433Utils/RPi_utils#
root@raspberrypi:/home/pi/433Utils/RPi_utils# ./test.sh
Good Read
Your code is 457624
BAD READ: your code and the valid don't match
Your correct valid code should be 16644136
BAD READ: your code and the valid don't match
Your correct valid code should be 5301448
BAD READ: your code and the valid don't match
Your correct valid code should be 5301448
BAD READ: your code and the valid don't match
Your correct valid code should be 8699400
Good Read
Your code is 457624
Good Read
Your code is 457624
█
```

If you don't have the Lloytron wireless MIP doorbells & sensors, but a different make or model and don't see a code displayed, then your doorbell or PIR isn't compatible with this kit, sorry.

If you do have the Lloytron MIP sensors and aren't seeing a code produced, then it's likely the receiver module is wired incorrectly or your doorbell/PIR isn't close enough to the Pi (initially test everything in the same room).

A word about the way folders & files are organised on a Raspberry Pi.

If you've just ran the ./test.sh file on the previous page and decide to power off & come back to your Raspberry Pi tomorrow, you'll notice that when the Pi next powers up you won't be able to find the test.sh file again easily. This is because we installed it into the folder /home/pi/433Utils/RPi_utils.

So next time you power up your Pi, & login in as user Pi, do this:

```
cd ~/433Utils/RPi_utils  
ls -al
```

and you'll see the list of files for this project.

Type in

```
cd ..
```

and you'll move up one folder level to folder /home/pi/433Utils

type in `cd ..` again and you'll be in folder /home/pi

to print the current working directory, type in

```
pwd
```

If you want to make your Pi always go into folder /home/pi/433Utils/RPi_utils on bootup, type in

```
nano /home/pi/.profile
```

make the last line say

```
cd ~/433Utils/RPi_utils
```

Ctrl-O to WriteOut and then Ctrl-X to Exit

When you reboot your Pi you should automatically go to folder /home/pi/433Utils/RPi_utils.

How to make an executable script.

We've assumed you're running the latest version of the Raspbian OS for the Raspberry Pi. (There's an ISO image of it on the DVD and instructions for making a bootable SD card at the end of this document).

When you see us change font to **courier**, we are indicating a command you type into the command line. For example:

```
nano test2.sh
```

Will open a basic text editor & allow you to enter a series of commands, that form the basis of all our scripts. Enter the following text:

```
echo "hello world"
```

Then save the file & exit. To make the command executable type:

```
chmod u+x test2.sh
```

and then to run the script

```
./test2.sh
```

Some scripts will need to be run as root (the highest security level). To do that type:

```
sudo ./test2.sh
```

To see a list of files in the current folder type

```
ls -al
```

If you have any problem running a script, then you've either made a typo, forgotten to make it executable with **chmod**, or need to put a **sudo** in front of it.

If you get bored of entering **sudo** before each command, you can switch to root by entering:

```
sudo su
```

You'll notice the prompt then changes from \$ to #. You can exit root by pressing Ctrl-D. While you're in root mode it's a good idea to change the default passwords for root & user pi, like this:

```
passwd  
passwd pi
```

Sending command line email Raspberry Pi Bullseye (Updated November 2022)

Bullseye doesn't support the old SSMTP method, you need to use MSMTTP to send email instead. Before we can send command line emails we need to install & configure several mail applications from the internet. You also need to have made an **extra Gmail account** for just the Pi to use, and configured a 16 character app password in the google account settings screen (see page 9/10).

```
sudo apt install msmtpp msmtpp-mta mailutils mpack ca-certificates
```

Setup default settings for MSMTTP.

```
cd /etc
sudo touch msmtprc
sudo nano msmtprc
```

```
account default
host smtp.gmail.com
port 587
logfile /tmp/msmtpp.log
tls on
tls_starttls on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
auth login
user your-new-pi-gmail-account@gmail.com
password your-16-character-app-password
from First Last Name
account account2
```

save file & exit.

```
cd ~
```

To go back to your home folder.

Send a test email with this command, substituting the email address below with your own.

```
echo -e "Subject: Test Mail\r\n\r\nThis is a test mail" |msmtpp --
debug --from=default -t you@yourdomain.co.uk
```

That goes all on one line. If it worked you can remove the --debug option next time.

You can also use it like this

```
echo -e "Motion Detected" | msmtpp you@yourdomain.co.uk
```

This is how you send a photo attachment as an email:

```
mpack -s "alarm photo" /home/pi/image.jpg you@yourdomain.co.uk
```

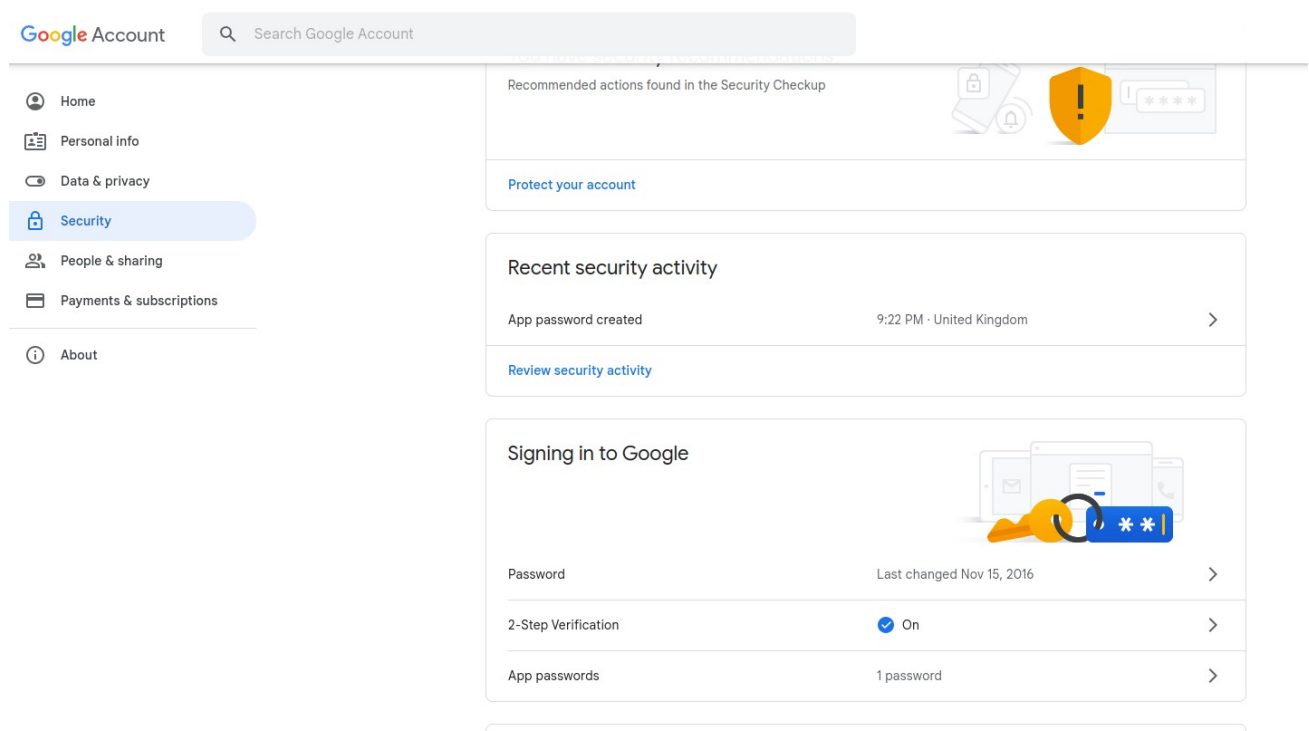

Setting up a spare Gmail account, just for your Pi to use (updated Nov 2022)

The simplest way of sending emails & photos as attachments from your Pi, is to setup a new Gmail account for the Pi to use, even if you already have an existing Gmail account you use on your phone or PC. For one thing, it gets you 15GB of new cloud storage for your alert photos & secondly it removes the complication of generating application specific passwords for other apps on your existing Gmail account.

You need to create the new Gmail account in the web browser on your PC or Mac @

<https://accounts.google.com/SignUp?service=mail>

Then setup 2-Step Verification with your phone and then you'll be able to set an App password:



The app password is 16 characters long with **no spaces**, and this is what we use on the Raspberry Pi along with the new gmail address in the `/etc/msmtprc` file, when setting the configuration.

[← App passwords](#)

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Name	Created	Last used	
Raspberry Pi	9:22 PM	9:25 PM	

Select the app and device you want to generate the app password for.

Raspberry Pi 2

×

GENERATE

And

Google Account

[← App passwords](#)

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Generated app password

Email

securesally@gmail.com

Password

••••••••••

Your app password for your device

umlo inkw imdj wptz

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

DONE

Now we have a working email account, just for the Pi to use when sending emails. Any photos sent from the Pi will be backed-up in the Sent folder & you only need delete old photos if you get near to the 15GB limit. Emails from the Pi can be sent to any other email address on your phone or PC.

Send an email when the doorbell or PIR is triggered.

This script builds on test.sh by adding basic email capability & event logging to a text file.

```
nano alarm1.sh
```

Then enter the following commands into the nano editor:

```
#!/bin/sh
while true; do
    valid="457624"
    scan=`./RFSniffer`
    if [ "$scan" = "$valid" ]; then
        d=`date +%d%m%y`
        t=`date +%T`
        echo "Alarm triggered $t $d" | mail -s "Alarm" youremailaddress@gmail.com
        echo "Alarm triggered $t $d"
        echo "Alarm triggered $t" >> log$d.txt
    else
        echo "bad read your code and valid dont match"
        echo $scan
    fi
    sleep 20
done
exit 0
```

Save the file & exit.

Once again, make the script executable with

```
chmod u+x alarm1.sh
```

Assuming that worked okay, you can now run the alarm1.sh script.

```
sudo ./alarm1.sh
```

When the drive alarm is triggered you should see “Alarm Triggered” appear on the display, a record will also be added to the date stamped text file. Now check your emails & you should have a record of the alarm being triggered. You can quit the alarm1.sh script with Ctrl-C.

You can check the contents of a time stamped log file with:

```
cat log060613.txt
```

You can see a list of all your log files with:

```
ls -al log*.txt
```

If you have lots of entries in your log file, you can pause the display with:

```
cat log060613.txt | more
```

or you can open the file in the editor with:

```
nano log060613.txt
```

If you notice the time & date aren't set correctly, you can run:

```
sudo dpkg-reconfigure tzdata
```

Setup the Raspberry Pi camera module.

Install latest version of Raspbian to an SD card.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
raspi-config
```

Enable camera support in raspi-config & reboot.

Check the camera's working with these two test commands:

```
raspistill -o image.jpg
```

```
raspivid -o video.h264 -t 10000
```

```
mpack -s "alarm photo" /home/pi/image.jpg you@yourdomain.co.uk
```

The standard photo resolution was too big to view on our mobile phone screen, so in the script below we've reduced the size to something easier to view on a small screen. The video that gets recorded to your SD card is still HD.

The video files captured by the script are typically too large to send as email attachments. If you want to view them remotely, it's best to login into the Pi using Filezilla in SFTP mode.

```
nano drivealarm-rpicam.sh
```

Enter the script:

```
#!/bin/sh
while true; do
    valid="457624"
    scan=`./RFSniffer`
    if [ "$scan" = "$valid" ]; then
        d=`date +%d%m%y`
        t=`date +%T`
        raspistill -o $t$d.jpg -w 1024 -h 768 -q 30 -hf
        echo "Driveway alarm $t $d" | mail -s "Driveway alarm" you@gmail.com
        echo "Driveway alarm $t $d"
        echo "Driveway alarm $t" >> log$d.txt
        raspivid -o $t$d.h264 -t 10000
        mpack -s "door open photo" $t$d.jpg you@gmail.com
    else
        echo "bad read your code and valid dont match"
        echo $scan
    fi
    sleep 20
done
exit 0
```

Save the file (Ctrl-O) & exit (Ctrl-X).

```
chmod u+x drivealarm-rpicam.sh
```

```
sudo ./drivealarm-rpicam.sh
```

The script takes a photo & video when the PIR is triggered & sends the photo as an attachment to the Gmail account specified in the script. It then records 10 seconds of HD quality video to the memory card. The time of the event is also recorded to a date-stamped log file.

Here's an analysis of what happens on the raspistill command line:

```
raspistill -o $t$d.jpg -w 1024 -h 768 -q 30 -hf
```

Output a still photo to a file whose name is the current time+date.jpg. Make the photo measure 1024 pixels wide (-w) by 768 pixels high (-h). Set the quality (-q) to 30, which is quite low but reduces the size of file we then send by email. Finally, horizontal flip the picture (-hf) because our test picture was the wrong way around.

Text on photo.

How to overlay text on a photo from the Raspberry Pi Camera module.

You might want to overlay text on a photo to provide a time & date stamp, before emailing the photo to your phone.

Firstly, we need to install the Imagemagick library:

```
sudo apt-get install imagemagick
```

Next, we need to take a test photo & then use the `convert` command to overlay the text.

```
raspistill -o 1.jpg -w 1024 -h 768 -q 30
```

```
d=`date +%d%m%y`
```

```
t=`date +%T`
```

```
convert -pointsize 20 -fill yellow -draw 'text 850,30 "'$t'  
'$d'' 1.jpg 2.jpg
```

As you can see in the example above, the top-left of the photo is co-ordinate 0,0 and the bottom-right would be 1024,768. The file 2.jpg is now time & date stamped.



If you notice the time & date aren't set correctly, you can run:

```
sudo dpkg-reconfigure tzdata
```

The script below will email you a time & date stamped photo:

```
while true; do
    valid="457624"
    scan=`./RFSniffer`
        if [ "$scan" = "$valid" ]; then
    d=`date +%d%m%y`
    t=`date +%T`
    raspistill -o 1.jpg -w 1024 -h 768 -q 30
    raspivid -o $d$t.h264 -t 10000
    convert -pointsize 20 -fill yellow -draw 'text 850,30 "'$t' '$d'"" 1.jpg $d$t.jpg
    mpack -s "alarm photo" /home/pi/$d$t.jpg you@youremailaddress.co.uk

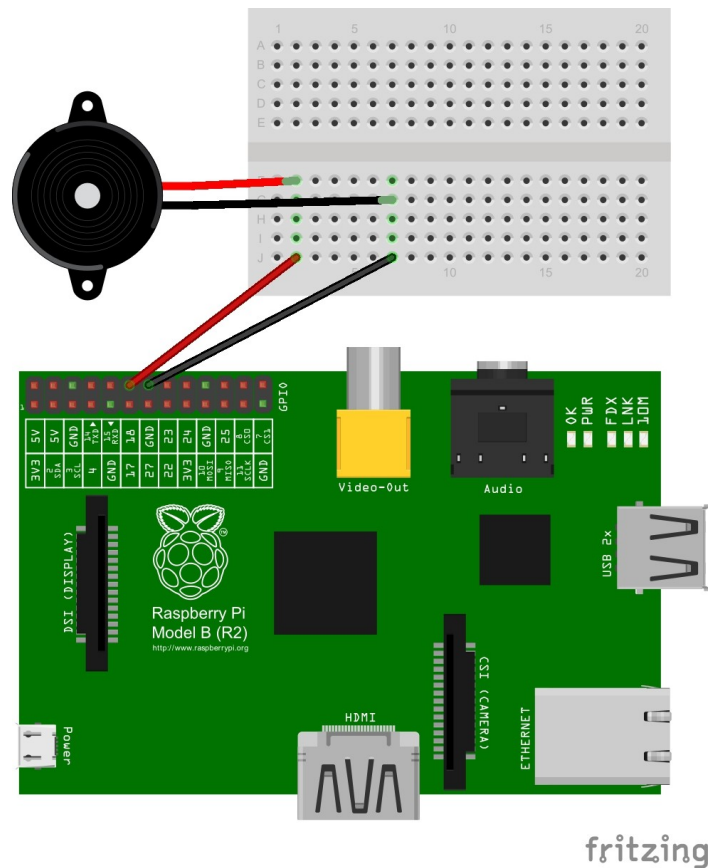
        else
            echo "bad read your code and valid dont match"
            echo $scan
        fi
    sleep 20
done
exit 0
```

You'll also find the script on our DVD, it's called drivealarm-rpicamtd.sh

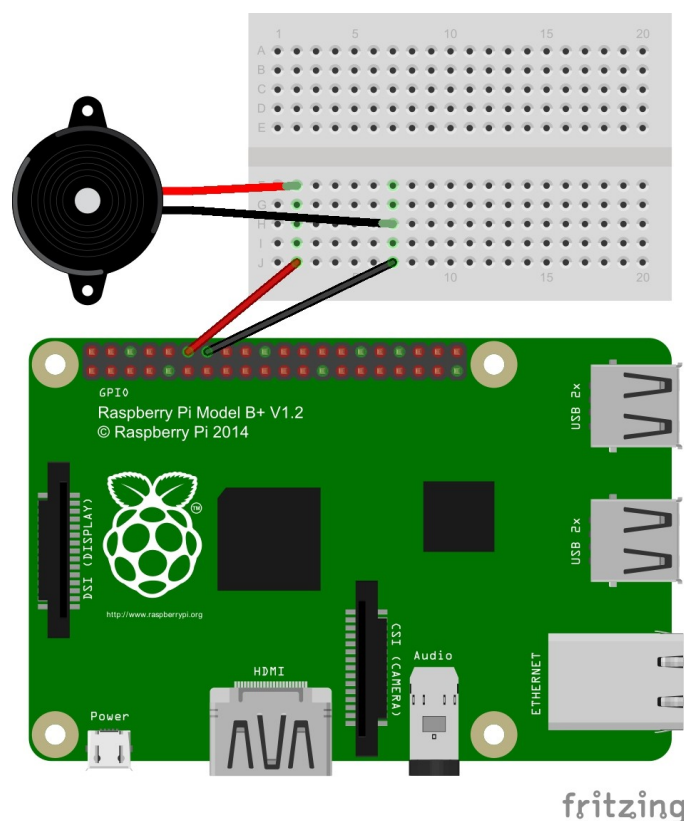
How to attach a buzzer to the Raspberry Pi.

Cheap 5 volt buzzer modules are available on eBay for around £1 and connect to a GPIO pin & GND on the Raspberry Pi. If you set the GPIO pin high, the buzzer emits a noise, and when you set the GPIO pin back to 0 it stops. Make sure you wire the buzzer the right way around (shorter leg to gnd, longer leg to GPIO pin 18). The wires we've supplied maybe different colours.

Here's how you connect it to the Model B Raspberry Pi.



And the Pi 4 , Pi 3 and Pi Zero WH.



Here's a script called `buzzer.sh`, that will turn the buzzer on & off twice. It gets pulled down from our server with the other scripts.

```
#!/bin/sh

echo "18" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio18/direction

    echo "1" > /sys/class/gpio/gpio18/value
    sleep .5
    echo "0" > /sys/class/gpio/gpio18/value
    sleep .5
    echo "1" > /sys/class/gpio/gpio18/value
    sleep .5
    echo "0" > /sys/class/gpio/gpio18/value

echo "18" > /sys/class/gpio/unexport
```

There are also three other individual buzzer patterns, in `buzzer2.sh`, `buzzer3.sh` & `buzzer4.sh`. The last two scripts use a loop to achieve more complicated patterns. If you have a combination of doorbells & PIR sensors, it's good to have a different noise for each.

You can test the buzzer, by typing in:

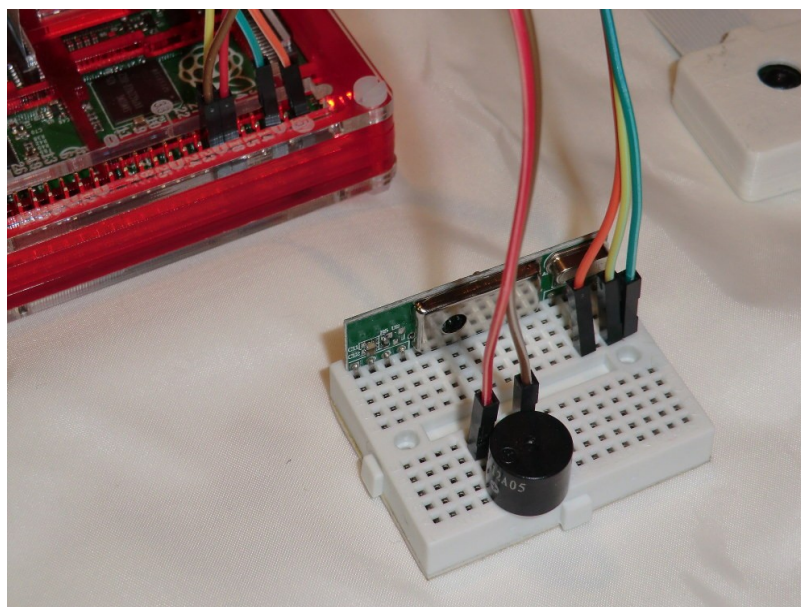
```
sudo ./buzzer.sh
```

add the line

```
./buzzer.sh &
```

to an existing script to sound the buzzer. The & ampersand character makes the buzzer command run in the background, so it doesn't delay the rest of the script.

As you can see, the buzzer & the 433MHz receiver fit comfortably on the same mini breadboard. Remember the longer leg on the buzzer is the + side, shorter leg is GND.



How to play an MP3 file when the doorbell is pressed.

```
#!/bin/sh
while true; do
    valid="457624"
    scan=`./RFSniffer`
    if [ "$scan" = "$valid" ]; then
        omxplayer -o local Doorbell.mp3
    else
        echo "bad read your code and valid dont match"
        echo $scan
    fi
    sleep 20
done
exit 0
```

If you have a set of speakers plugged into the 3.5mm headphone output on the Pi, the script listed above will play an MP3 file when the doorbell is pressed. The line highlighted in red can be placed in any of the other scripts.

You'll find a number of MP3 sounds to experiment with on the DVD. Others free MP3 samples can be found through Google. <https://www.securipi.co.uk/doorbell-mp3s.zip>

To play an MP3 file to the 3.5mm headphone output

```
omxplayer -o local Doorbell.mp3
```

and to play an MP3 file through the HDMI port on your TV

```
omxplayer -o hdmi Doorbell.mp3
```

How to make your Pi automatically run our scripts at startup.

Here's how to make our scripts auto-run in the background each time you power up the Raspberry Pi, even when you don't have a screen & keyboard attached & no user is logged in. We've assumed you've already gone through all the previous chapters and have the scripts working correctly.

```
sudo nano /etc/rc.local
```

Scroll down to the bottom of the file & **above** the last line that says **exit 0** type in these lines :

```
cd /home/pi/433Utils/RPi_utils  
./drivealarm-rpicamtd.sh &
```

Then save the file, exit & reboot your Pi.

```
sudo reboot
```

Now, when you press the doorbell or trigger the driveway alarm you should get a photo sent to your phone's email account & if you cut the power to the Pi & let it reboot, the script should start again automatically.

If you write any of your own scripts, & want them to execute at startup time from rc.local, then it's important to make the 1st line read

```
#!/bin/sh
```

Also, when launching a script using rc.local, it doesn't know that your script is in folder /home/pi or /home/pi/433Utils/RPi_utils, which is why we first change the folder using the **cd** command. By changing folder first & then running the file, references to other files in the script will still be found.

Night & day photo emailer script for PiNoIR camera.

```
#!/bin/sh

while true; do
    valid="457624"
    scan=`./RFSniffer`
    HOUR="$(date +%H)"

    if [ "$scan" = "$valid" -a $HOUR -ge 21 -o "$scan" = "$valid" -a $HOUR -lt 07 ]; then
        echo "taking a Night-time photo"
        d=`date +%d%m%y`
        t=`date +%T`
        raspistill -ex night -o 1.jpg -w 1024 -h 768 -q 30
        raspivid -ex night -o $d$t.h264 -t 50000
        convert -pointsize 20 -fill yellow -draw 'text 850,30 "'$t' '$d'"" 1.jpg $d$t.jpg
        mpack -s "Night-time alarm photo" $d$t.jpg youremailaddress@gmail.com
        rm 1.jpg
    elif [ "$scan" = "$valid" ]; then
        echo "taking a Daytime photo"
        ./buzzer.sh
        d=`date +%d%m%y`
        t=`date +%T`
        raspistill -o 1.jpg -w 1024 -h 768 -q 30
        raspivid -o $d$t.h264 -t 10000
        convert -pointsize 20 -fill yellow -draw 'text 850,30 "'$t' '$d'"" 1.jpg $d$t.jpg
        mpack -s "Daytime alarm photo" $d$t.jpg youremailaddress@gmail.com
        rm 1.jpg
    else
        echo "BAD READ: your code and the valid don't match"
        echo "Your correct valid code should be " $scan
    fi
    sleep 5
done
exit 0
```

The script drivealarm-night.sh is pulled down from our server with the other scripts. It's really only useful if you have the PiNoIR camera module without the infrared filter on it.

If the hour is greater than or equal to 21 or less than 07 (so between 21:00 & 6:59) it takes a photo & video using night settings, otherwise it takes the photo in regular mode. You can adjust the start & end hours in the first IF statement, to suit the time of day it gets dark & light.

The night settings on the Raspberry PiNoIR camera basically use a longer exposure time, so while you get much more light in the photo than if you'd used the regular Pi camera, you'll notice that anyone moving in darkness will appear to leave a streak, and you won't be able to make out faces unless they stand completely still for several seconds. If you've got plenty of night IR illumination then experiment with regular exposure at night, rather than night exposure.

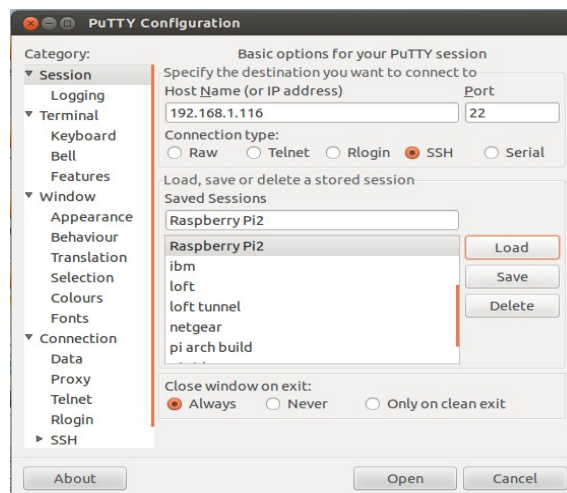
Connect to your Pi remotely from your PC.

It's a pain leaving a keyboard, mouse & monitor connected to your Pi when running the driveway alarm, so we login remotely using the free Putty terminal emulation package. Putty is available for PC, Mac & Linux from <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> This gives you a remote text terminal window on your PC, where you can issue commands as if you were sat in front of the Pi.

When you first install Raspbian Wheezy onto an SD card & start your Pi, you'll notice the **raspi-config** command will give you the option to turn on SSH – this is the service we use to login remotely. When you boot up the Pi, you'll also see the IP Address allocated to your Pi, above the Login & Password prompt – it will look something like 192.168.1.149 ← *note down the address you see and enter it into the Putty software on your PC.*

If you don't see the ip address you can use the command:

```
sudo ifconfig
```



On the PC running Putty, make sure the IP address is entered, you are set to Port 22 & the SSH radio button is selected, choose Open. Login as *pi* & password *raspberrypi*. Once logged in use **passwd** to change the easily guessed default to something else. We suggest you also do **sudo su** and **passwd** again, to change the root password, then do CTRL-D to drop back to user Pi.

You can make the scripts you've created run in background on the Pi & then logout from Putty. You do this by adding an ampersand character '&' to the end. To run scanner.sh in background type:

```
./scanner.sh &
```

Make a note of the process number displayed (say 2501 for example below) & then logout using CTRL-D. You should now receive emails from your Pi every time the alarm is tripped.

When you log back in you can kill the background process by rebooting the Pi, or by doing:

```
kill -9 2501
```

How to transfer files from your Pi to PC using Filezilla.

On your Raspberry Pi type in

```
ifconfig
```

and make a note of your inet addr of eth0 interface, it will be something like 192.168.1.135

On your PC, Mac or Linux PC download & install Filezilla from
<https://filezilla-project.org/download.php>

Launch Filezilla & go to File → Site Manager → New Site → name it Raspberry Pi.

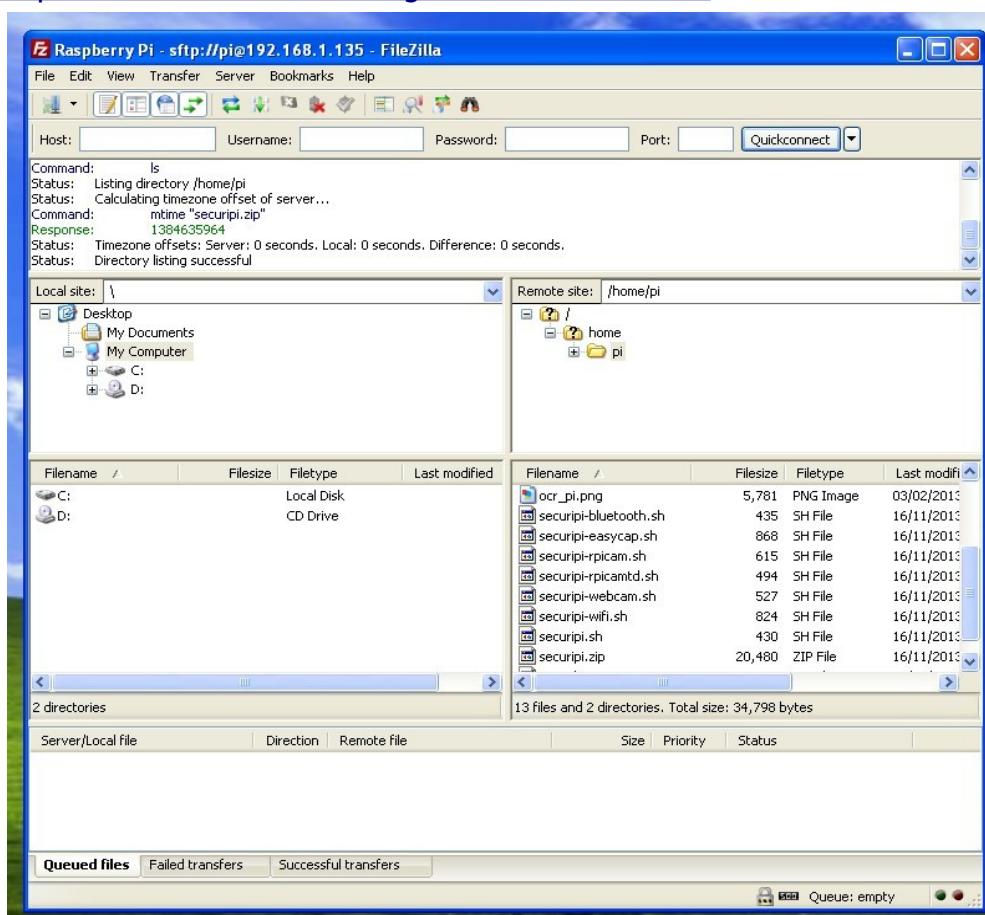
Then enter the following, remembering your IP address won't be the same as mine:

Host : 192.168.1.135
Port : 22
Protocol : SFTP
Logon Type : Normal
Login : pi
Password : raspberry

Then click OK.

File → Site Manager → Raspberry Pi → Connect → tick always trust this host tickbox & OK.
List of files on the Pi appears in the right hand pane. You can drag videos captured on your Pi onto your PC's Desktop (left hand pane).

Once you have the H.264 video files on your PC, you can play them using the free VLC Media Player. <http://www.videolan.org/vlc/index.html>



Extra Resources.

You might find this 433.92 MHz radio PDF, with more Python-based code, useful too:

<https://securipi.co.uk/remote-433-receivers.pdf>

It works with the radio receiver you've already purchased from us.

if you're interested in other radio frequencies and longer distance communications:

<http://www.securipi.co.uk/cc1101.pdf>

Happy to offer support to customers who've bought 433MHz stuff from our eBay or Amazon shops.

<https://www.ebay.co.uk/str/convertstuffuk>

<https://www.amazon.co.uk/sp?ie=UTF8&seller=A3FJQLQ9748AAR>

Email tim@trcomputers.co.uk

If you've bought an RXB6 433.92MHz module elsewhere and aren't getting the 20 metres reception range it's likely you bought one with a crystal marked 13.52127 on it. Get another with the crystal marked 433 and you'll get the good range.