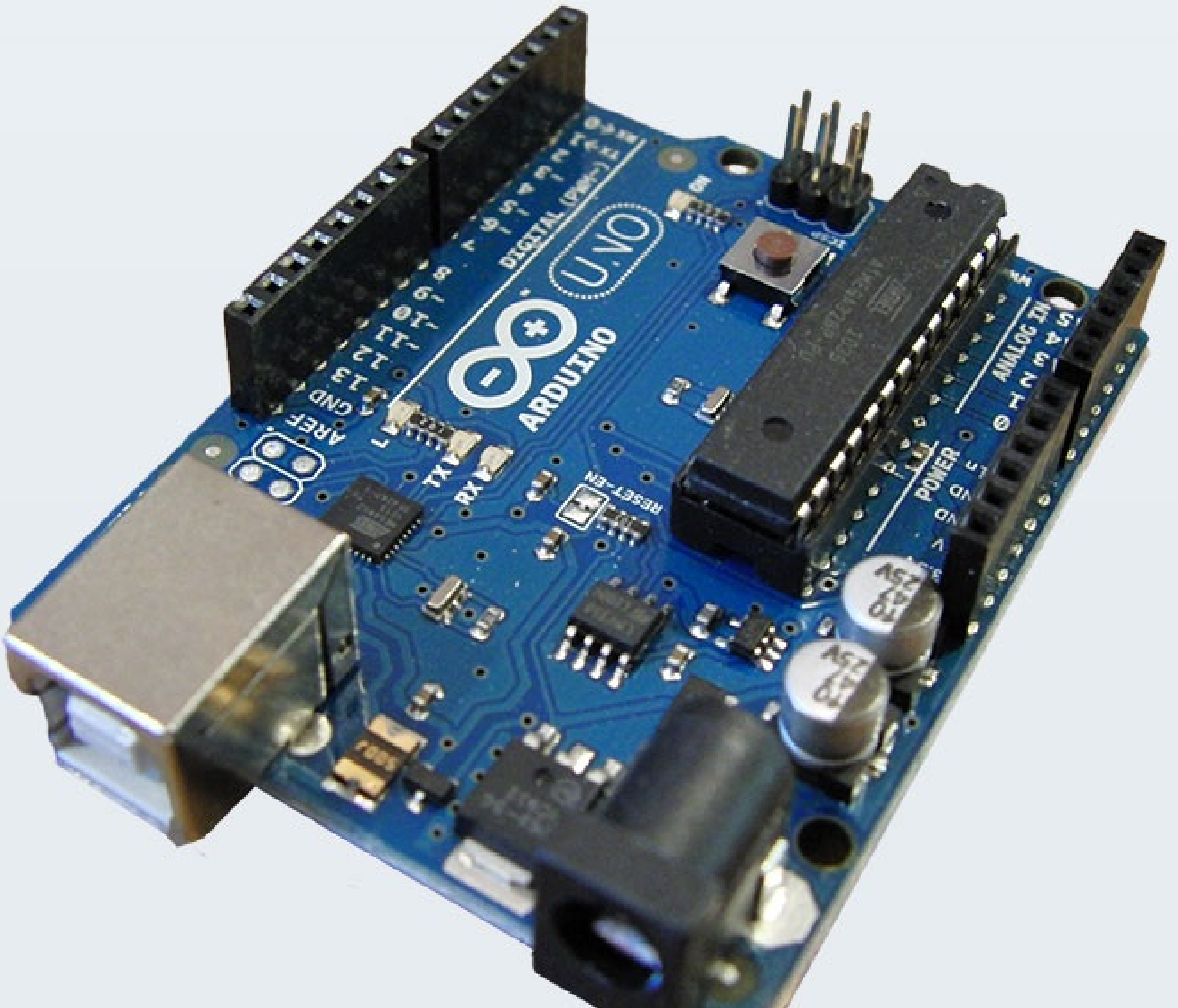
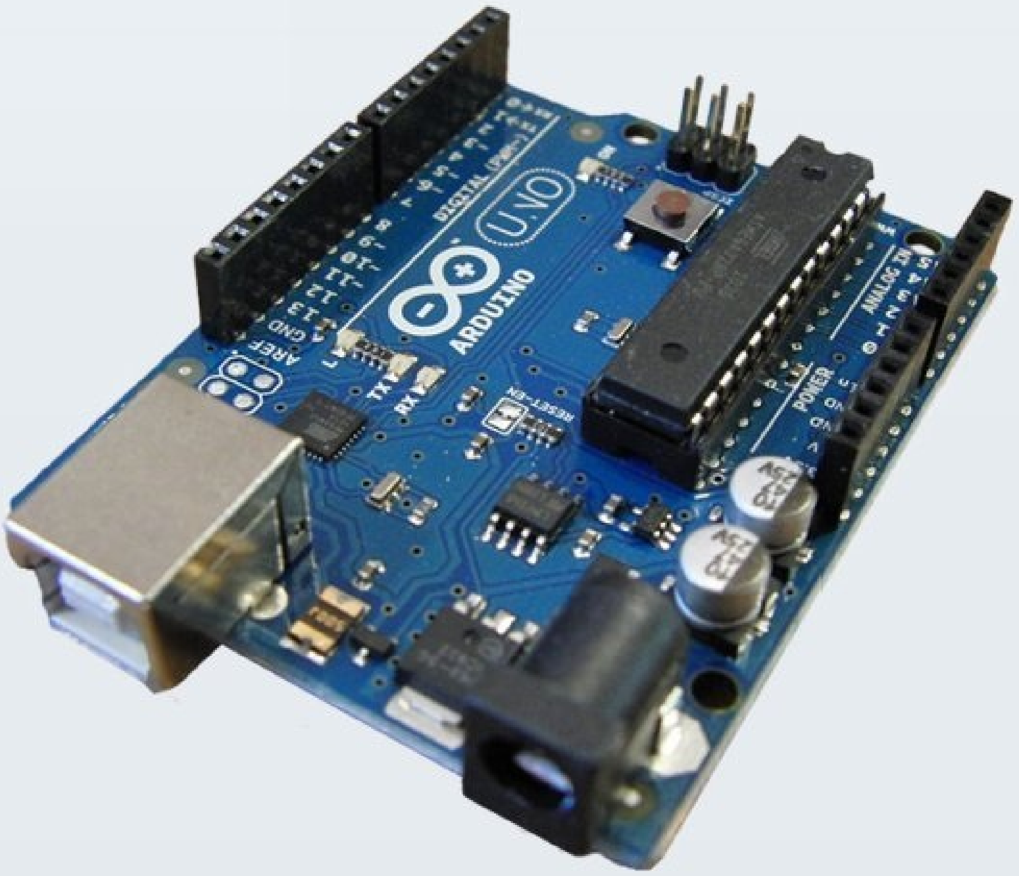
Home Security Projects for ARDUINO



Tim Rustige

Home Security Projects for ARDUINO



Tim Rustige

Home Security Projects for Arduino

Tim Rustige

Home Security Projects for Arduino

© Tim Rustige

First edition: June 2015

Published by TR Computers Limited.

Table of Contents

Introduction.

Our kit allows you to connect PIR motion sensors & magnetic door sensors to the USB port on your PC or Mac - via an Arduino board - and then receive alarm photos on your smartphone, when they're triggered. Our full project kit contains everything you'll need: Arduino compatible board, PIR sensor, magnetic door sensor, PDF instructions, Arduino software and Python software. We also show you how to use the inexpensive NRF24L01+ wireless boards for communications.

The Arduino compatible boards we use are the Uno R3 & Pro Nano clones with USB ports. You write a small program - called a Sketch - on your PC and then upload it to the Arduino board using the Arduino development software on your PC. The uploaded program then runs on the Arduino board each time you power it up. If you want to use the board for a different project, you just reprogram it with a new Sketch via the USB port.

The boards have digital and analogue input/outputs: there are 13 digital pins, which are like switches that can be on or off. So we can set one of the pins as an input & connect a PIR module to it, and also set another of the pins as an output and connect an LED to it (the LED we use is actually hard-wired onto the board). We can then write a Sketch (the Arduino name for a program) that senses the PIR motion sensor has been tripped and turns on the LED for 30 seconds.

There are also six analogue pins which can detect an input value between 0-1023. These are useful if you want to connect a sensor that might have a range of values, like a temperature, moisture or light sensor. Towards the end of this document we show you how to read a light sensor.

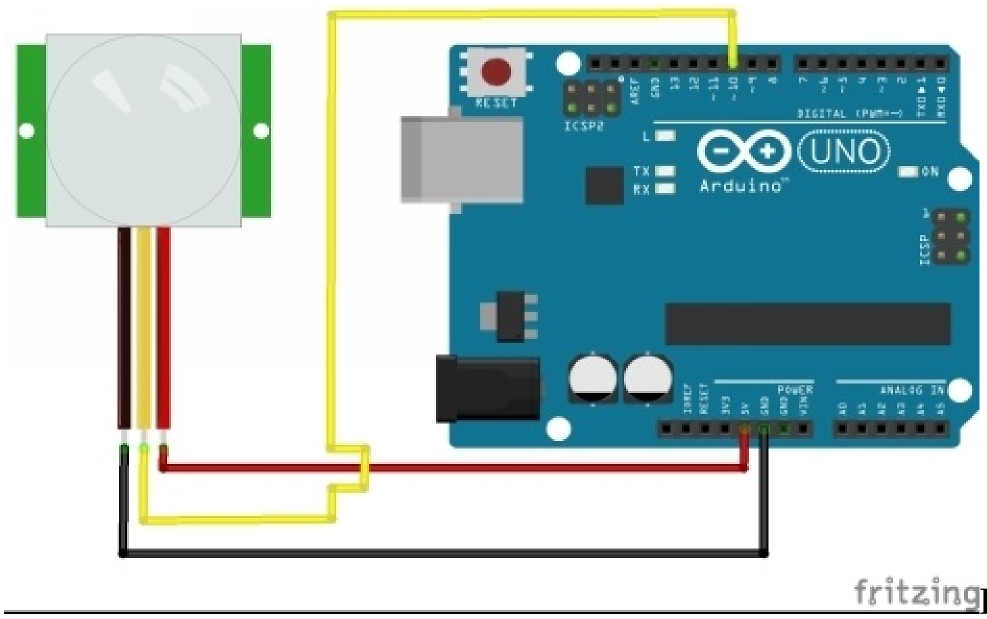
The boards connect to the PC using the USB port. The USB port supplies the 5 volt power required by the Arduino, and also the interface for reprogramming the board with a new Sketch. The board can talk back to the PC over a USB serial interface, which allows us to send a “motion detected” message back to the PC when the PIR is triggered.

On the PC we're going to use the programming language Python to read the “motion detected” message and then send an email alert to our phone, using a Gmail account. We're using Python because it's available for Windows, Mac & Linux PCs. We've chosen Python 2.7.8, because some of the library functions we need aren't yet available for the newer version Python 3.4.1

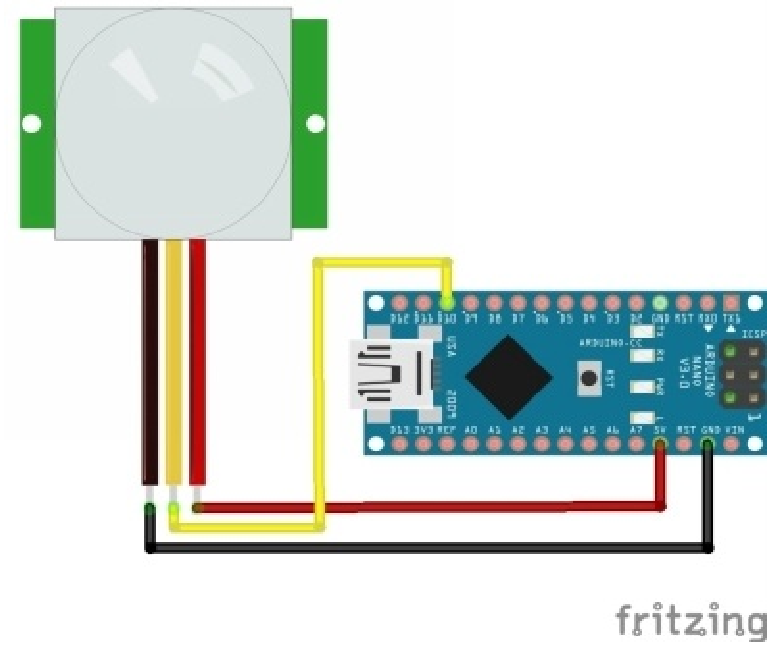
USB Arduino models can be powered by a PC, a USB mains adapter or a USB battery pack.

Connecting a PIR module to the Arduino.

Here’s how to connect the PIR module to the Uno R3 compatible board. If you lift the dome off the PIR you’ll see the pins are labelled. (Don’t worry about wire colours, just make sure you attach the three points together correctly.)



Here’s how to connect the smaller Pro Nano board: GND to GND , OUT to D10 & VCC to 5V

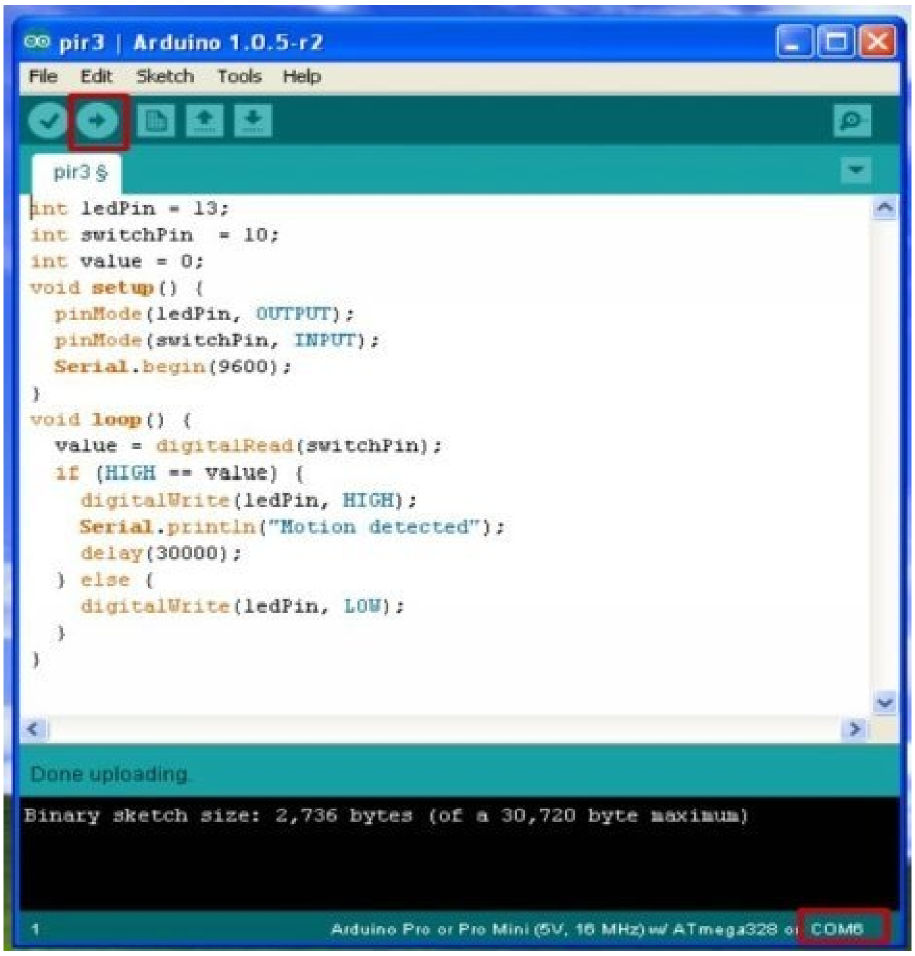


Setup the Arduino software.

The latest version of the Arduino 1.0.6 software suite for Windows, Mac OSX & Linux can be downloaded from http://arduino.cc/en/Main/Software

Once installed to your Windows PC, launch the software from the Desktop shortcut.

It will look like this:



```
int ledPin = 13;
int switchPin  = 10;
int value = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(switchPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  value = digitalRead(switchPin);
  if (HIGH == value) {
    digitalWrite(ledPin, HIGH);
    Serial.println("Motion detected");
    delay(30000);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Done uploading.

```
Binary sketch size: 2,736 bytes (of a 30,720 byte maximum)
```

1 Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328 o COM8

Go to File → Open & select the sketch called pir. Insert USB lead into the Arduino & your PC.

Go to Tools → Serial Port & select the serial port that appeared when you attached the Arduino.

(make a note of the port number, COM8 in the red box, you’ll need it later for the Python scripts)

Go to Tools → Board & select either Arduino Uno ATmega328 (the larger board) or Arduino Nano w/ ATmega328 (the smaller board).

You can now program the Arduino by clicking on the circle with a right-pointing arrow in it.

Setup Python on your PC

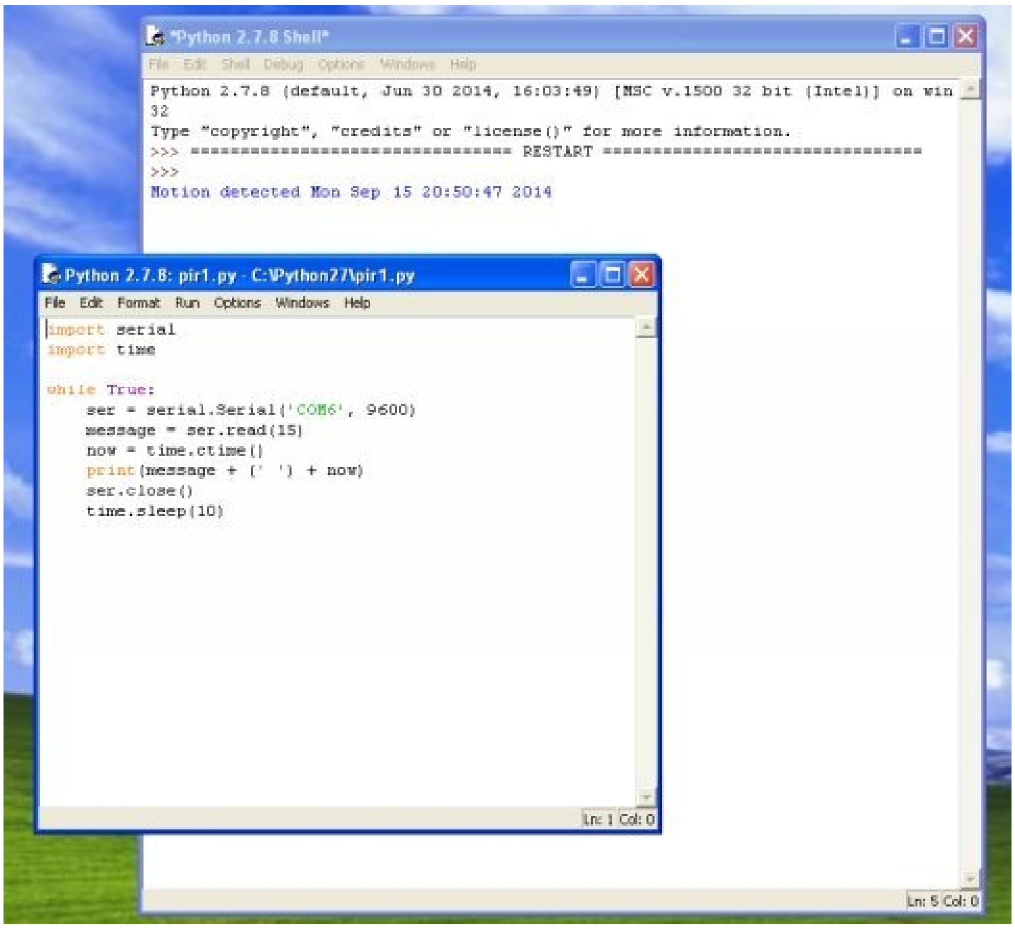
As we mentioned in the introduction, we use the Python programming language on the PC to read a serial input from the USB cable attached to the Arduino. We'll be looking out for a "Motion Detected" message from the Arduino & connected PIR module. When we see that message we'll perform an action. There are various example scripts, each slightly more complex & useful.

We're going to use Python version 2.7.8 which is available for Windows, Mac & Linux here:

https://www.python.org/download/releases/2.7.8/

Please don't be tempted to download the 64 bit or the newer 3.4.1 version, as the libraries we're using won't work with it. Python 2.7.8 32 bit for Windows XP, Vista 7 & 8.1 is also on the DVD.

Here's what the Python IDLE (GUI) looks like:


Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.8: pir1.py - C:\Python27\pir1.py
File Edit Format Run Options Windows Help
Ln: 1 Col: 0
Ln: 5 Col: 0

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Motion detected Mon Sep 15 20:50:47 2014
```

```
import serial
import time

while True:
    ser = serial.Serial('COM6', 9600)
    message = ser.read(15)
    now = time.ctime()
    print(message + (' ') + now)
    ser.close()
    time.sleep(10)
```

We also need to install the PySerial library pyserial-2.7.win32.exe (md5) for Python 2.x (2.4…2.7)

from https://pypi.python.org/pypi/pyserial for Windows installs. This is also on the DVD.

Python on Windows PCs.

Launch Python 2.7 → IDLE (Python GUI) from Windows Program menu. File → Open → pir1.py

New window opens → select Run → Run Module

If you get an error, then you'll need to change the serial COM port number in the pir1.py script to match the configuration on your PC. You can determine which USB serial COM port number has been allocated to your Arduino in the bottom right of the Arduino programming software screen.

Move in front of the PIR module & you should see the "Motion Detected" message in the Python Shell screen.

Here's what the contents of Python script Pir1.py look like in case you want to type it yourself:

```
import serial
import time

while True:
   ser = serial.Serial('COM6', 9600)
   message = ser.read(15)
   now = time.ctime()
   print(message + (' ') + now)
   ser.close()
   time.sleep(10)
```

The first two lines of the program setup the time & USB serial port functions. Then we have the main loop of the program, which continues forever, or until you close the Idle Python GUI window.

The first line of the main loop sets up the serial port, which you will almost certainly have to change from COM6 to something else, like COM8 or COM10, depending what you found when you ran the Arduino software.

You can also see which COM port has been allocated to your Arduino board in Windows XP under Control Panel → System → Hardware → Device Manager → Ports (COM & LPT).

Try inserting the USB device into a different port on your PC if you don't see it listed.

The next line reads 15 characters from the serial port into the variable message . The next line reads the current time & date into the variable now . Then the variables get printed to the screen, the serial port gets closed & we wait ten seconds while the PIR module resets.

Next, we’ll add email capabilities..

Setting up a spare Gmail account.

The simplest way of sending emails & photos as attachments from your PC, is to setup a new Gmail account for the PC to use, even if you already have an existing Gmail account. 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.

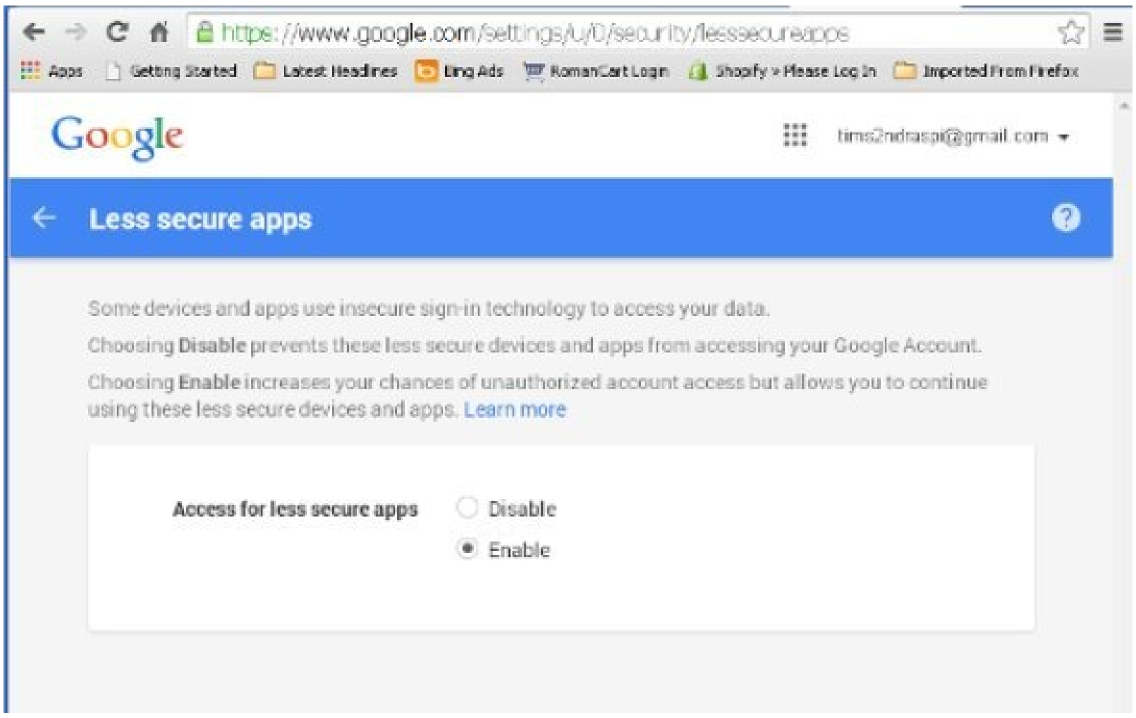
You need to create the new Gmail account in the web browser on your PC or Mac @

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

and note down the login & password for later.

Next, you need to set the new Gmail account to Enable “less secure apps”. While logged in go to:

https://www.google.com/settings/u/0/security/lesssecureapps


https://www.google.com/settings/u/0/security/lesssecureapps

Apps | Getting Started | Latest Headlines | Bing Ads | RomanCart Login | Shopify » Please Log In | Imported From Firefox

Google

tims2ndraspi@gmail.com

Less secure apps

Some devices and apps use insecure sign-in technology to access your data.

Choosing **Disable** prevents these less secure devices and apps from accessing your Google Account.

Choosing **Enable** increases your chances of unauthorized account access but allows you to continue using these less secure devices and apps. Learn more

Access for less secure apps Disable [✓ Enable]

Now we have a working email account, just for the PC to use when sending emails from Python. Any photos sent from the PC 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 PC running Python can be sent to any other email address on your phone or a different PC.

Python script with email function.

First, go to the website gmail.com & create a new account, just for the Arduino to use. Do this even if you already have a Gmail account you use every day. Use that new login & password below.

Launch Python 2.7 → IDLE (Python GUI) from Windows Program menu. File → Open → pir2.py

You'll need to change the text I've highlighted in red.

```
import serial
import time
import smtplib

TO = ‘your-phones-existing-email-adress@gmail.com‘
GMAIL_USER = ‘your-arduinos-new-email@gmail.com‘
GMAIL_PASS = ‘hard-to-guess-password‘

SUBJECT = ‘Intruder’
TEXT = ‘PIR Triggered’

def send_email():
    print(“Sending Arduino Email”)
    smtpserver = smtplib.SMTP(“smtp.gmail.com”,587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = ‘To:’ + TO + ‘\n’ + ‘From: ‘ + GMAIL_USER
    header = header + ‘\n’ + ‘Subject:’ + SUBJECT + ‘ ‘ + now + ‘\n’
    print header
    msg = header + ‘\n’ + TEXT + ‘ \n\n’
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()

while True:
    ser = serial.Serial(‘COM6‘, 9600)
    message = ser.read(15)
    now = time.ctime()
    print(message + (‘ ‘) + now)
    send_email()
    ser.close()
    time.sleep(10)
```

Change the TO = email address to the one you want to receive the alert, typically the email address you use on your smartphone. Change the GMAIL_USER and GMAIL_PASS values to the ones for the brand new gmail account you just made. Also remember that your serial port probably won’t be the same as my COM6.

select Run → Run Module & you should receive an email on your phone when the PIR is triggered.

How to grab an image from a webcam in Python.

To capture an image from a webcam in Microsoft Windows, we need to install Python libraries VideoCapture.py and PIL. You need to download VideoCapture-0.9-5.zip from http://videocapture.sourceforge.net/ and Python Imaging Library 1.1.7 for Python 2.7 from http://www.pythonware.com/products/pil/index.htm.

VideoCapture.py comes as a compressed file with the folders \Lib & \DLLs inside, the contents need to be manually cut & pasted into your c:\Python27 folders \lib & \DLLs

If you've downloaded VideoCapture.py from the internet , don't copy the files from the folder "DLLs (for 64-bit Python)" - even if you're using 64 bit Windows, use the 32 bit DLLs instead.

VideoCapture.py & PIL are also on the DVD. Install PIL, as you would any other Windows application, once the file \DLLs\vidcap.pyd has been copied to folder c:\Python27\DLLs & the five files inside \Lib folder to c:\Python27\Lib folder.

If you're using Windows 8, you may find you need to first copy those five \Lib files to c:\Python27 & then copy them from there to the folder c:\Python27\Lib.

We tested the image capture using an old Sony PS2 Eyetoy webcam on Windows XP, and a brand new Logitech C270 webcam on Windows 7 64 bit & Windows 8.1. - other models may or may not work correctly. (Note: We're using 32 bit versions of Python & all libraries on 32 & 64 bit Windows – they work fine)

The script vidcaptest.py shown below will capture a photo called image.jpg to the current folder on your hard drive, which is hopefully c:\Python27. (Don't try and run vidcaptest.py from the DVD!)

```
from VideoCapture import Device
cam = Device()
cam.setResolution(640,480)
cam.saveSnapshot(‘image.jpg’,timestamp=1,boldfont=0,textpos=‘bc’)
```

If the photo captured correctly, you can now move on to the next script.

Capture a photo & email it as an attachment.

Once you’ve confirmed that image.jpg got captured correctly, you can use the pir3.py script below to capture a photo from the webcam and email it to your phone when the PIR is triggered.

Parts of the script that you need to alter are highlighted in red.

```
#!/usr/bin/python
import serial
import time
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
from VideoCapture import Device
import os

gmail_user = “your-arduino@gmail.com”
gmail_pwd = “hard-to-guess-password”
cam = Device()

def mail(to, subject, text, attach):
    msg = MIMEMultipart()

    msg[‘From’] = gmail_user
    msg[‘To’] = to
    msg[‘Subject’] = subject

    msg.attach(MIMEText(text))

    part = MIMEBase(‘application’, ‘octet-stream’)
    part.set_payload(open(attach, ‘rb’).read())
    Encoders.encode_base64(part)
    part.add_header(‘Content-Disposition’,
    ‘attachment; filename=”%s”’ % os.path.basename(attach))
    msg.attach(part)

    mailServer = smtplib.SMTP("smtp.gmail.com", 587)
    mailServer.ehlo()
    mailServer.starttls()
    mailServer.ehlo()
    mailServer.login(gmail_user, gmail_pwd)

    mailServer.sendmail(gmail_user, to, msg.as_string())
    mailServer.close()

while True:
    ser = serial.Serial('COM8', 9600)
    message = ser.read(15)
    now = time.ctime()
    print(message + (' ') + now)
    cam.setResolution(640,480)
    cam.saveSnapshot('image3.jpg',timestamp=1,boldfont=0,textpos='bc')
    mail("your-phones-email-account@gmail.com",
    "PIR motion alarm",
    "This email was sent with python",
    "image.jpg")
    ser.close()
    time.sleep(10)

```

Connect a magnetic door sensor to the Arduino

Components required:

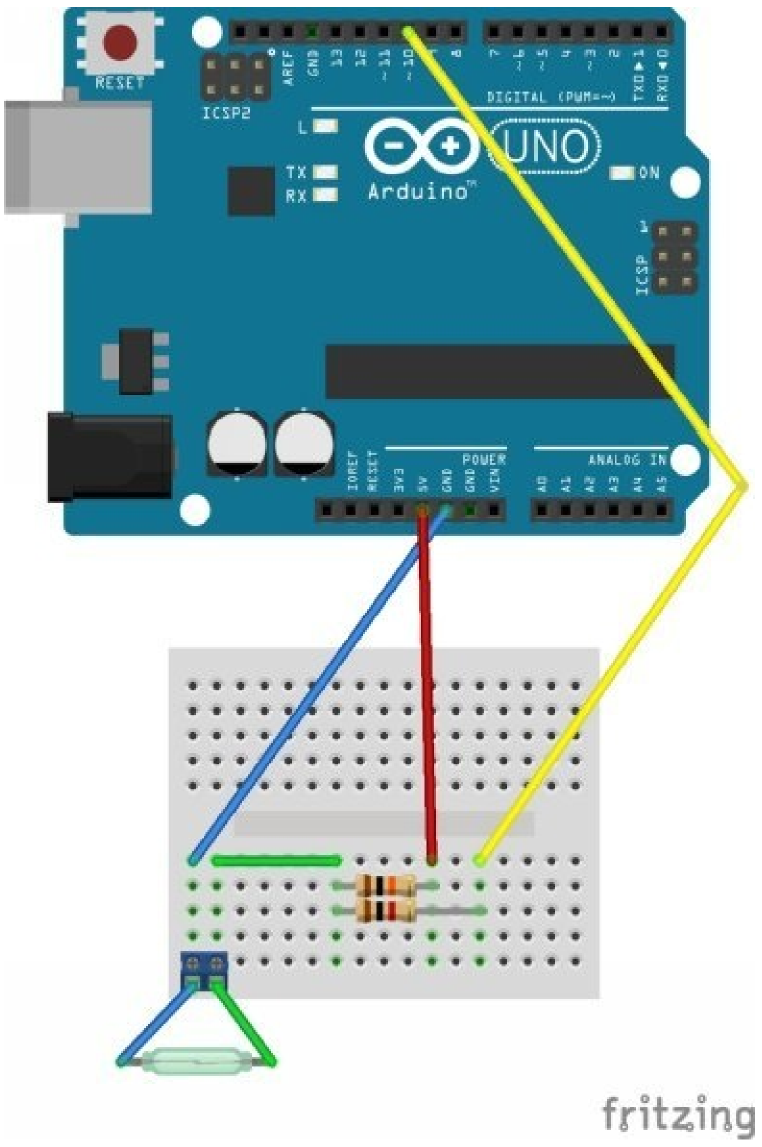
Mini Breadboard

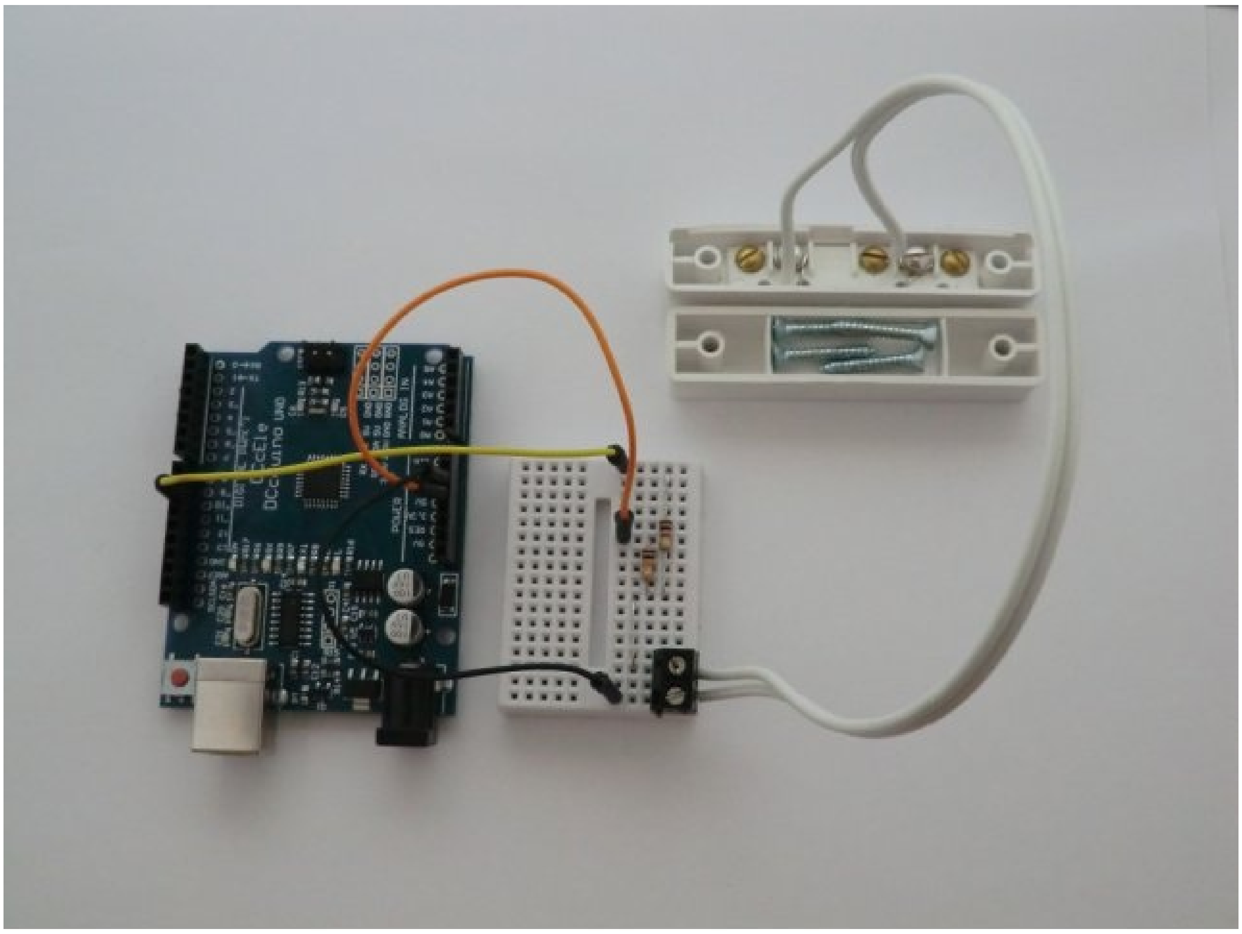
1K resistor (red, black, brown)

10K resistor (orange, black, brown)

Magnetic reed switch, link wire & terminal block.

3x male to male IDC cables (colour not important).





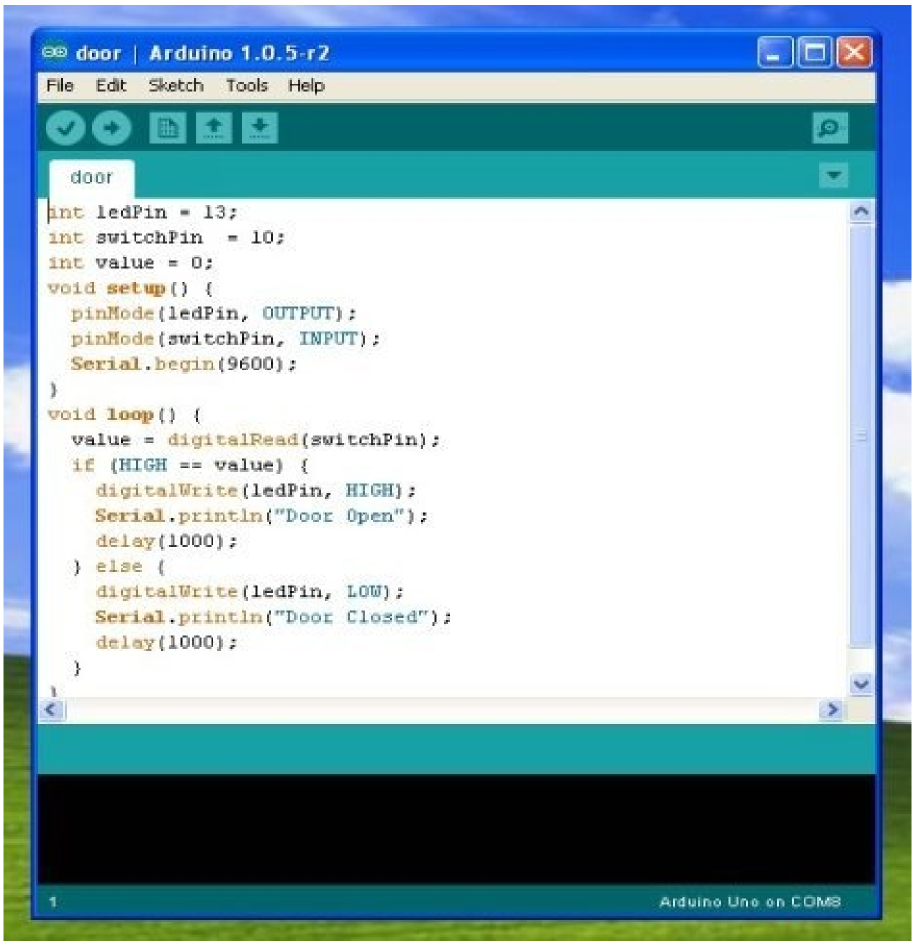
The colour of the IDC male to male leads we've supplied aren't important, just make sure you join the correct points together.

Setup the Arduino software for door sensor.

The latest version of the Arduino 1.0.6 software suite for Windows, Mac OSX & Linux can be downloaded from http://arduino.cc/en/Main/Software

Once installed to your Windows PC, launch the software from the Desktop shortcut.

It will look like this:



```
int ledPin = 13;
int switchPin  = 10;
int value = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(switchPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  value = digitalRead(switchPin);
  if (HIGH == value) {
    digitalWrite(ledPin, HIGH);
    Serial.println("Door Open");
    delay(1000);
  } else {
    digitalWrite(ledPin, LOW);
    Serial.println("Door Closed");
    delay(1000);
  }
}
```

Go to File → Open & select the script called door. Insert the USB lead into the Arduino & your PC.

Go to Tools → Serial Port & select the serial port that appeared when you attached the Arduino.

(make a note of the port number - COM8 bottom right - you'll need it later for the Python scripts)

Go to Tools → Board & select either Arduino Uno ATmega328 (the larger board) or Arduino Nano ATmega328 (the smaller board).

You can now program the Arduino by clicking on the circle with a right-pointing arrow in it.

Setup Python with door sensor on your PC

As we mentioned in the introduction, we’ll use the Python programming language on the PC, to read a serial input from the USB cable attached to the Arduino. We’ll be looking out for a “Door OPENED” message from the Arduino & connected door sensor. When we see that message we’ll perform an action. There are various example scripts, each slightly more complex & useful.

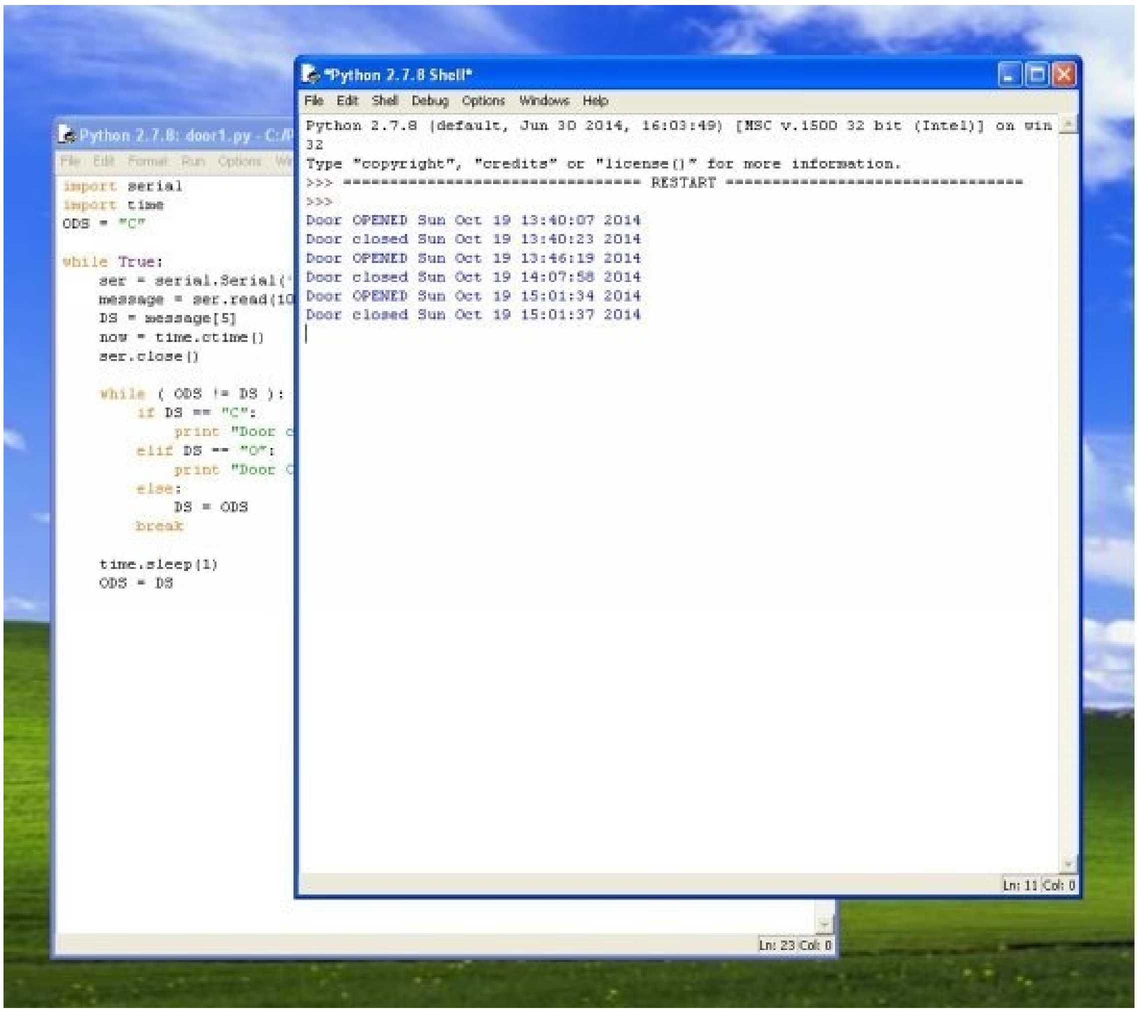
We’re going to use Python version 2.7.8 which is available for Windows, Mac & Linux here:

https://www.python.org/download/releases/2.7.8/

Please don’t be tempted to download the 64 bit or the newer 3.4.1 version, as the libraries we’re using won’t work with it. Python 2.7.8 32 bit for Windows XP, Vista 7 & 8 is also on the DVD.

We also need to install the PySerial library pyserial-2.7.win32.exe (md5) for Python 2.x (2.4…2.7) from https://pypi.python.org/pypi/pyserial for Windows installs. This is also on the DVD.

Here’s what the Python IDLE (GUI) looks like:


Python 2.7.8: door1.py - C:/
File Edit Format Run Options Wi
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
Ln: 11 Col: 0
Ln: 23 Col: 0

```
import serial
import time
ODS = "C"

while True:
    ser = serial.Serial('
    message = ser.read(10
    DS = message[5]
    now = time.ctime()
    ser.close()

    while ( ODS != DS ):
        if DS == "C":
            print "Door c
        elif DS == "O":
            print "Door O
        else:
            DS = ODS
        break

    time.sleep(1)
    ODS = DS
```

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Door OPENED Sun Oct 19 13:40:07 2014
Door closed Sun Oct 19 13:40:23 2014
Door OPENED Sun Oct 19 13:46:19 2014
Door closed Sun Oct 19 14:07:58 2014
Door OPENED Sun Oct 19 15:01:34 2014
Door closed Sun Oct 19 15:01:37 2014
```

Python door sensor example on Windows PCs.

Launch Python 2.7 → IDLE (Python GUI) from Windows Program menu. File → Open → door1.py

New window opens → select Run → Run Module

(If you get an error, then you'll need to change the serial COM port number in the door1.py script to match the configuration on your PC – it's in ***red*** *text in the listing below. You can determine which USB serial COM port number has been allocated to your Arduino in the bottom right of the Arduino programming software screen. Or, you can see which COM port has been allocated to your Arduino board in Windows XP under Control Panel → System → Hardware → Device Manager → Ports COM & LPT. Try inserting the USB device into a different port on your PC if you don't see it listed.)*

Move the magnet away from the wired door sensor and you should see the "Door OPENED" message in the Python Shell screen. Move it back & you should see "Door closed".

Contents & explanation of Python script Door1.py, in case you want to type it yourself:

```
import serial
import time
ODS = "C"

while True:
   ser = serial.Serial('COM8', 9600)
   message = ser.read(10)
   DS = message[5]
   now = time.ctime()
   ser.close()

   while ( ODS != DS ):
   if DS == "C":
   print "Door closed " + now
   elif DS == "O":
   print "Door OPENED " + now
   else:
```

```
   DS = ODS
   break

   time.sleep(0.1)
   ODS = DS
```

The first two lines of the program give us access to the time & USB serial port functions in Python. The variable ODS stands for “OLD DOOR STATE” and the starting value is “C” for closed.

The main While loop repeatedly reads 10 characters from the USB/serial port into the variable message . The 5thcharacter of the message variable gets loaded into DS & should either read “C” for closed or “O” for open (it will occasionally read something else, but we ignore anything that isn’t “O” or “C”). We get the current date & time into the variable now . Then we close the serial port.

The secondary loop compares the door state we just read in (either “C” or “O”) to the old door state, & if they’re different, it means someone has opened or closed the door… So the script prints “Door OPENED” & the time & date to the screen, or “Door closed”. If the value in DS isn’t “O” or “C” (which it occasionally will be due to errors reading the USB/serial port) then we set DS back to the old door state ODS.

We then drop back out to the main loop and sleep for 0.1 of a second. Lastly, whatever is held in the variable DS now gets shifted into ODS (the Old Door State).

Python door alarm script with email function.

First, go to the website gmail.com & create a new account, just for the Arduino to use. Do this even if you already have a Gmail account you use every day. Enter those new account details below.

Run Python 2.7 → IDLE (Python GUI) from Windows Program menu. File → Open → door2.py

You'll need to change the text I've highlighted in **red**.

```
import serial
import time
import smtplib

TO = 'your-phones-email-address@gmail.com'
GMAIL_USER = 'your-arduino@gmail.com'
GMAIL_PASS = 'password'

NOW = " "
ODS = "C"
message = " "

def send_email():
    print("Sending Email")
    smtpserver = smtplib.SMTP("smtp.gmail.com",587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = 'To:' + TO + '\n' + 'From: ' + GMAIL_USER
    header = header + '\n' + 'Subject:' + message + ' ' + NOW + '\n'
    print header
    msg = header + '\n' + message + '\n' + NOW + ' \n\n'
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()


while True:
    ser = serial.Serial('COM8', 9600)
```

```
message = ser.read(11)
DS = message[5]
NOW = time.ctime()
ser.close()

while ( ODS != DS ):
if DS == “C”:
print “Door closed ” + NOW
send_email()
elif DS == “O”:
print “Door OPENED ” + NOW
send_email()
else:
DS = ODS
break

time.sleep(0.1)
ODS = DS
```

Change the TO = email address to the one you want to receive the alert, typically the email address you use on your smartphone. Change the GMAIL_USER and GMAIL_PASS values to the ones for the brand new gmail account you just made. Also remember that your serial port will probably not be the same as my COM8.

select Run → Run Module & you should receive an email on your phone when the door sensor is triggered.

Next, I’ll show you how to grab a photo from a webcam, when the door sensor is triggered.

How to grab an image from a webcam in Python.

To capture an image from a webcam in Microsoft Windows, we need to install Python libraries VideoCapture.py and PIL. You need to download VideoCapture-0.9-5.zip from http://videocapture.sourceforge.net/ and Python Imaging Library 1.1.7 for Python 2.7 from http://www.pythonware.com/products/pil/index.htm.

VideoCapture.py files inside the folders \Lib & \DLLs need to be manually cut & pasted into your c:\Python27 folders \lib & \DLLs

Don't copy the files from the folder "DLLs (for 64-bit Python)" if you've downloaded VideoCapture.py from the internet, even if you're using 64 bit Windows.

VideoCapture.py & PIL are also on the DVD. Install PIL, as you would any other Windows application, once the file \DLLs\vidcap.pyd has been copied to folder c:\Python27\DLLs & the five files inside \Lib folder to c:\Python27\Lib folder.

If you're using Windows 8, you may find you need to first copy those five \Lib files to c:\Python27 & then copy them from there to the folder c:\Python27\Lib.

We tested the image capture using an old Sony PS2 Eyetoy webcam on Windows XP, and a brand new Logitech C270 webcam on Windows 7 64 bit & Windows 8.1. - other models may or may not work correctly. (Note: We're using 32 bit versions of Python & all libraries on 32 & 64 bit Windows – they work fine)

The script vidcaptest.py shown below will capture a photo called image.jpg to the current folder on your hard drive, which is hopefully c:\Python27. (Don't try and run vidcaptest.py from the DVD!)

```
from VideoCapture import Device
cam = Device()
cam.setResolution(640,480)
cam.saveSnapshot('image.jpg',timestamp=1,boldfont=0,textpos='bc')
```

If the photo captured correctly, you can now move on to the next script.

Capture a photo & email it as an attachment.

Once you’ve confirmed that image.jpg got captured correctly, you can use the door3.py script below to capture a photo from the webcam and email it to your phone when the door is opened. You should have already created a new gmail.com account solely for use with the Arduino.

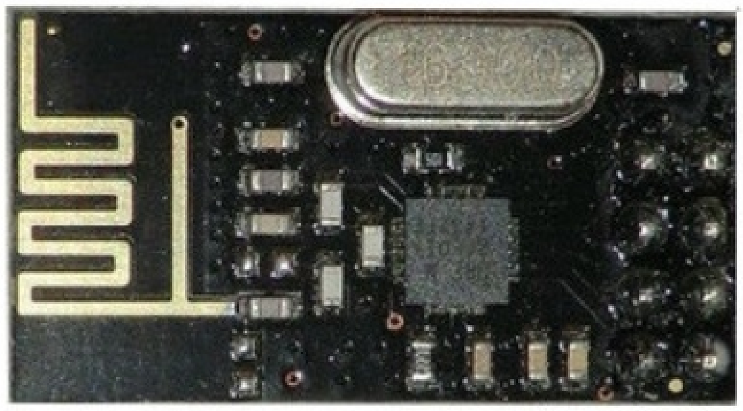
Parts of the script that you need to alter are highlighted in red.

```
#!/usr/bin/python
import serial
import time
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
from VideoCapture import Device
import os

gmail_user = “your-arduino@gmail.com”
gmail_pwd = “hard-to-guess-password”
cam = Device()

NOW = ” “
ODS = “C”
message = ” “

def mail(to, subject, text, attach):
    msg = MIMEMultipart()

    msg[‘From’] = gmail_user
    msg[‘To’] = to
    msg[‘Subject’] = subject

    msg.attach(MIMEText(text))

    part = MIMEBase(‘application’, ‘octet-stream’)
    part.set_payload(open(attach, ‘rb’).read())
    Encoders.encode_base64(part)
    part.add_header(‘Content-Disposition’,
    ‘attachment; filename=”%s”’ % os.path.basename(attach))
    msg.attach(part)

    mailServer = smtplib.SMTP(“smtp.gmail.com”, 587)
    mailServer.ehlo()
    mailServer.starttls()
    mailServer.ehlo()
    mailServer.login(gmail_user, gmail_pwd)
    mailServer.sendmail(gmail_user, to, msg.as_string())
    mailServer.close()


while True:
    ser = serial.Serial(‘COM9‘, 9600)
    message = ser.read(11)
    DS = message[5]
    NOW = time.ctime()
    ser.close()

    while ( ODS != DS ):
    if DS == “C”:
    print “Door closed ” + NOW
    cam.setResolution(640,480)
    cam.saveSnapshot(‘image3.jpg’,timestamp=1,boldfont=0,textpos=‘bc’)
    mail(“your-phones-email-address@gmail.com“,
    “Door Closed”,
    “Door Closed ” + NOW,
    “image3.jpg”)
    elif DS == “O”:
    print “Door OPENED ” + NOW
    cam.setResolution(640,480)
    cam.saveSnapshot(‘image3.jpg’,timestamp=1,boldfont=0,textpos=‘bc’)
    mail(“your-phones-email-address@gmail.com“,
    “Door OPENED”,
    “Door OPENED ” + NOW,
    “image3.jpg”)
    else:
    DS = ODS
    break

    time.sleep(0.1)
    ODS = DS

```

Part 2 – Going Wireless

While the examples already given work great with an Arduino connected to the USB port on your PC, Mac or Raspberry Pi. You might want to have a number of different sensors, in various places around your home, reporting back wirelessly to the Arduino attached to your PC.

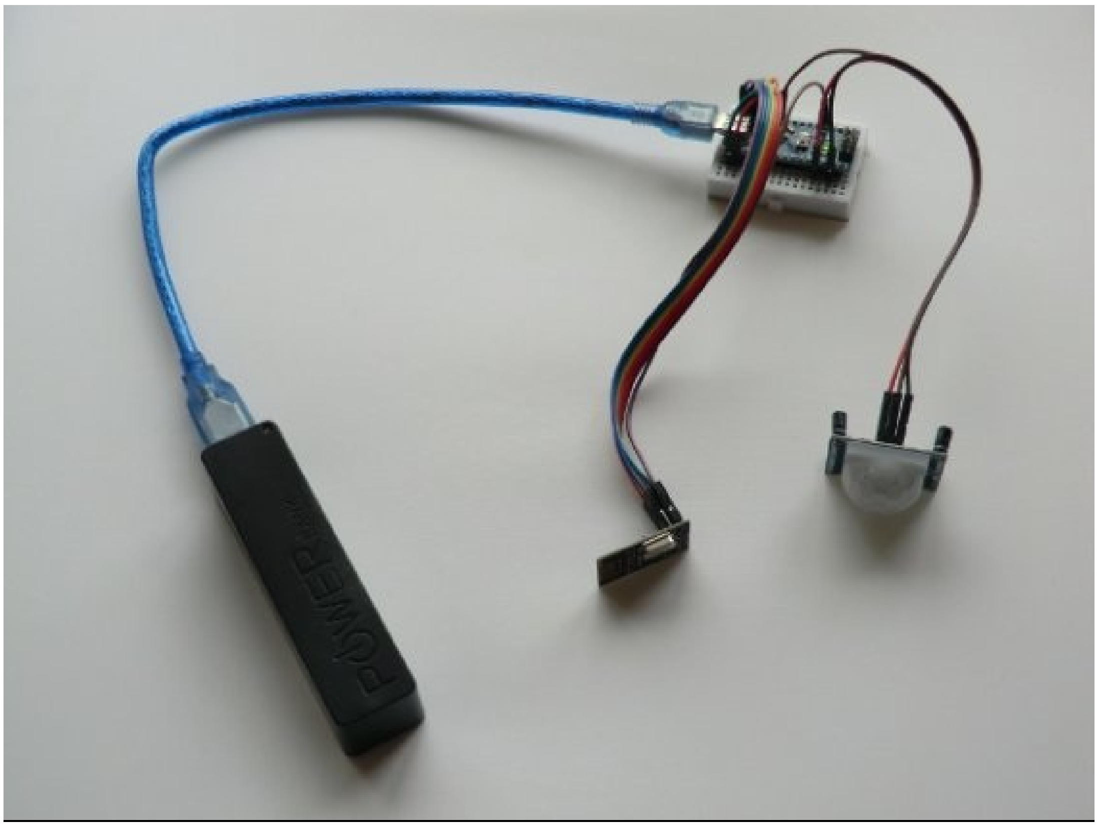
As the Arduino Uno R3 and Nano clone boards can be bought from China for around £3/$5 each, we can make complete wireless PIR sensors for around £10/$15 each. We're going to use the inexpensive NRF24L01+ boards, which cost around £1/$1.50 each on eBay, to communicate wirelessly. Try and buy the ones pictured as we know they work correctly.

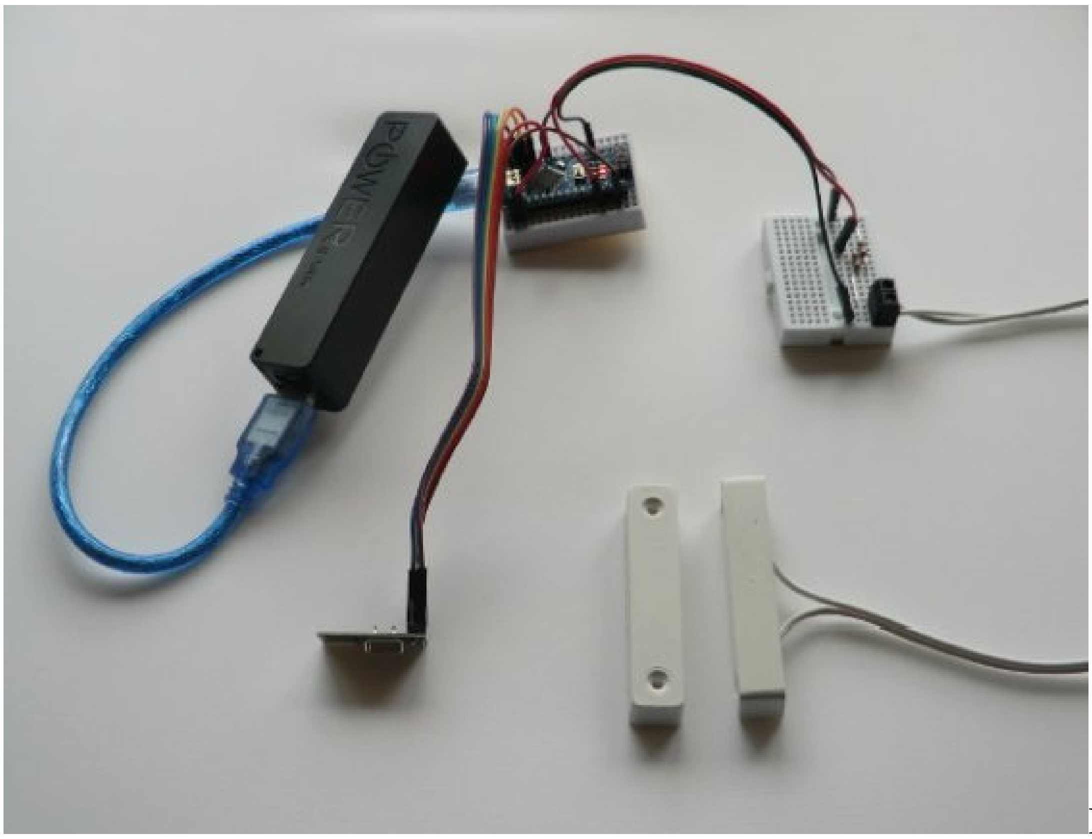


The NRF24L01+ boards transmit and receive in the same licence-free 2.4GHz band used by WiFi, but aren't WiFi compatible. You can still get interference from other WiFi devices though, so don't set up your project next to your WiFi router.

The operating range of the standard boards with an antenna printed on the PCB is around 20 metres, but there is also a slightly more expensive version with an external antenna, that can manage 100 metres.

Here's what a fully assembled wireless PIR sensor looks like. Powered by a USB battery pack.



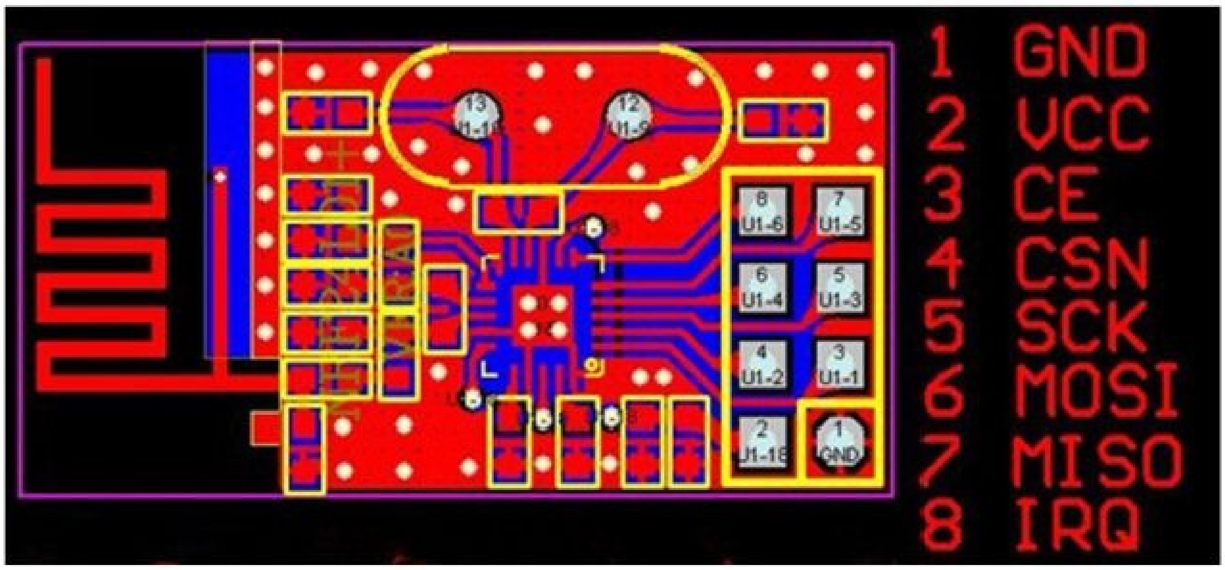


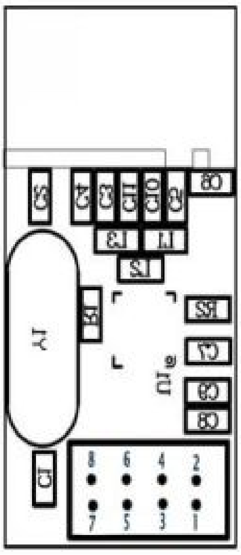
Here’s what a fully assembled magnetic door sensor looks like.

How to connect the NRF24L01+ to Arduino

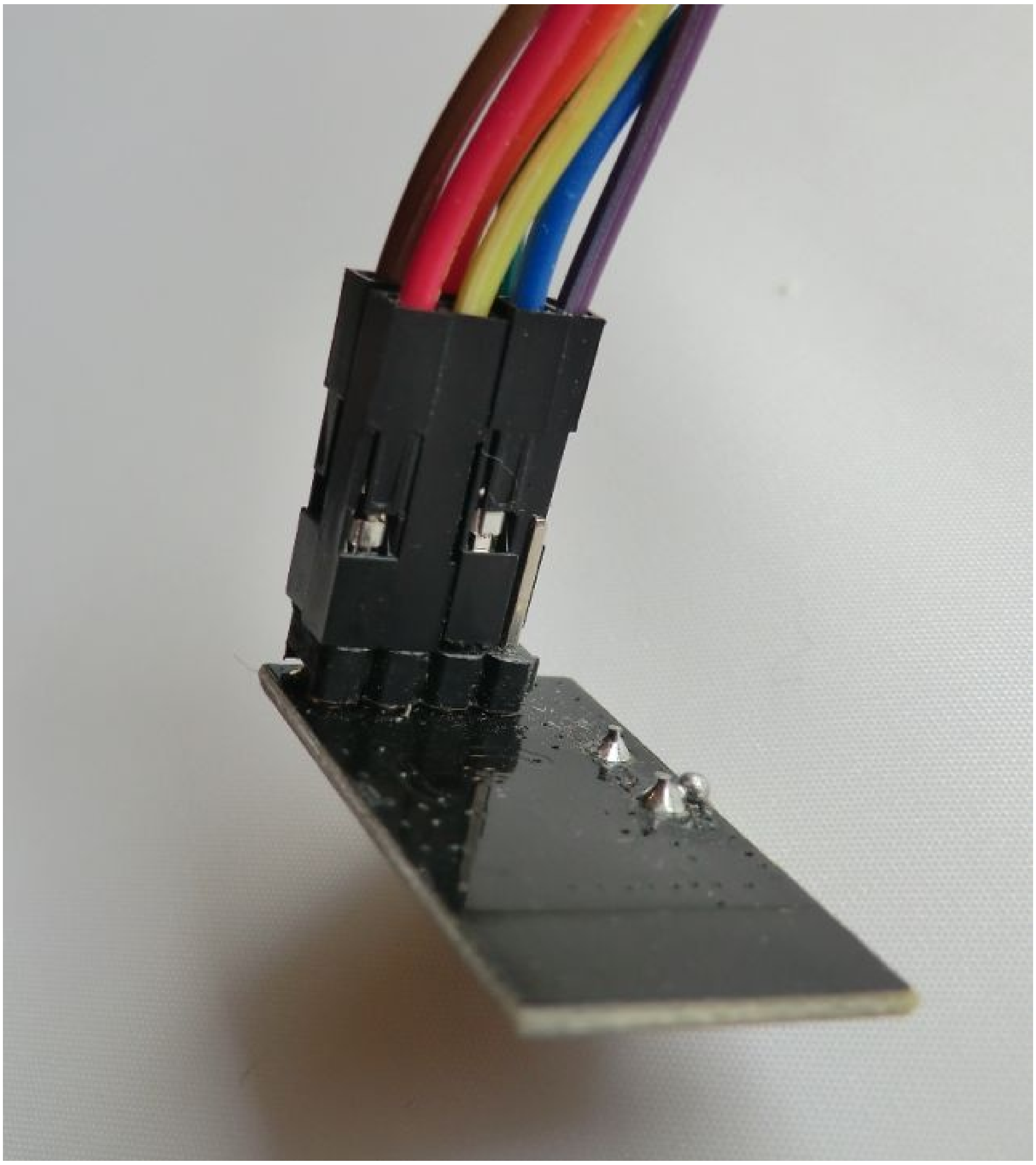
Signal	**RF Module pin**	**Wire Colour**	**Arduino Pin**
GND	1	Brown	GND
VCC	2	Red	3.3V
CE	3	Orange	9
CSN	4	Yellow	10
SCK	5	Green	13
MOSI	6	Blue	11
MISO	7	Violet	12

Top view:





Bottom view:

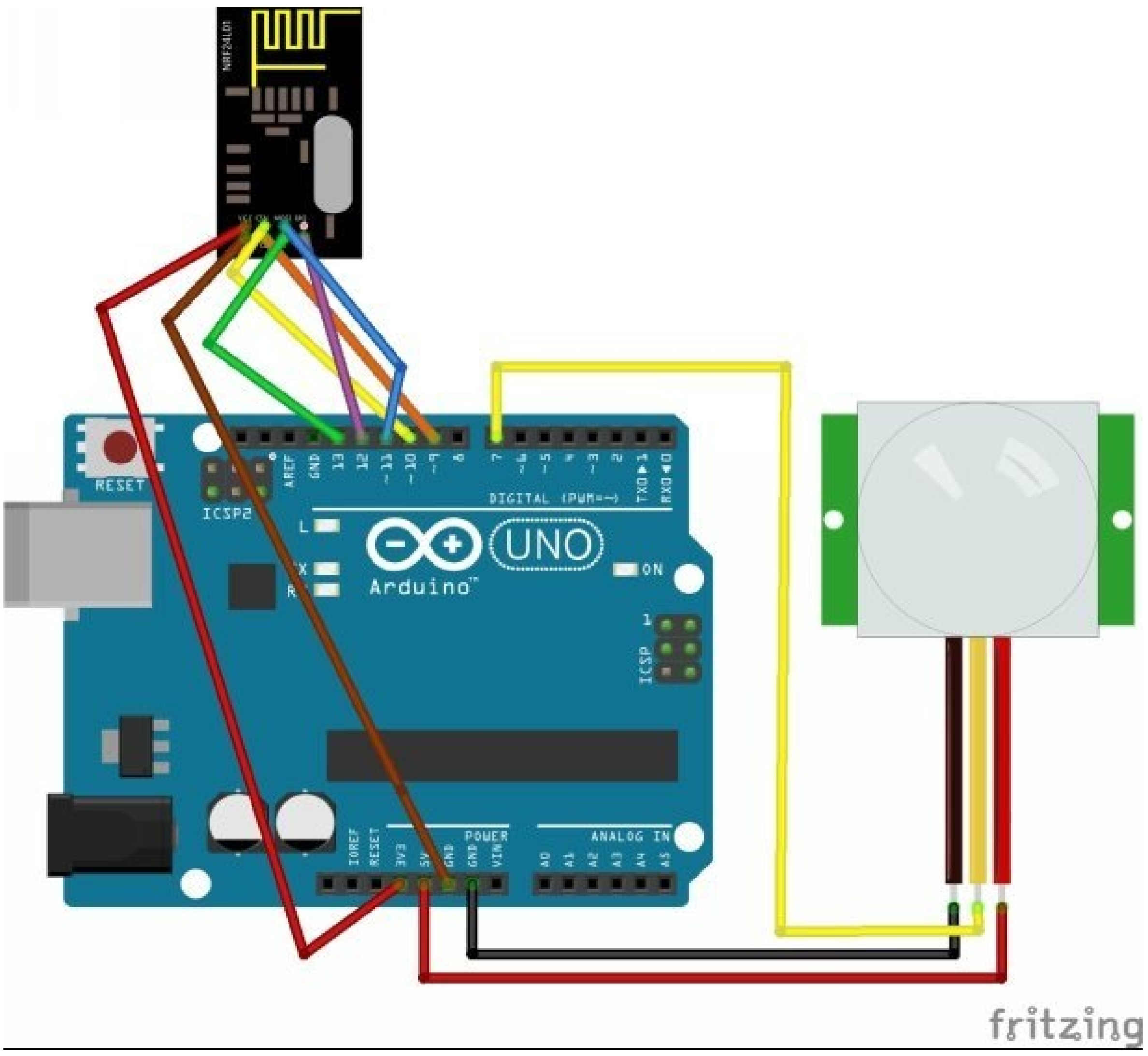


Cabling diagram for Arduino Uno R3 wireless PIR

The link cables are standard 20cm long 40 way male to female DuPont breadboard cables available from eBay. The brown, red, orange, yellow, green, blue, violet cables are all next to each other in the 40 way cable we used, and can be separated into a 7 wire bunch.

Remember the NRF24L01+ boards are powered by 3.3V, not 5V. The red wire from the NRF24L01+ connects to 3V3 on the Arduino.

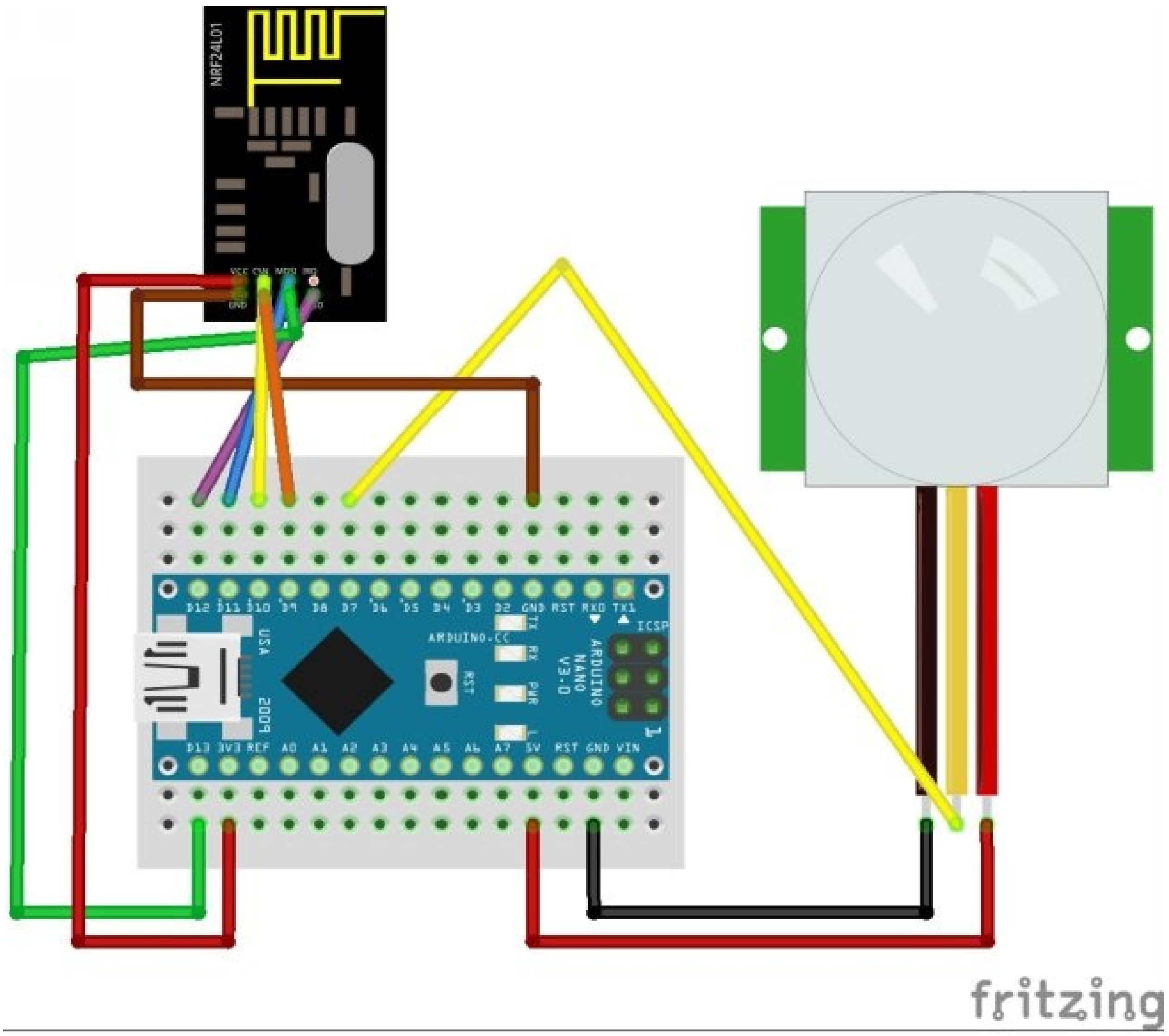
Here’s a Fritzing wiring diagram for an Arduino Uno R3 board:



Cabling diagram for an Arduino Nano wireless PIR.

The link cables are standard 20cm long 40 way male to female DuPont breadboard cables available from eBay. The brown, red, orange, yellow, green, blue, violet cables are all next to each other in the 40 way cable we used, and can be separated into a 7 wire bunch.

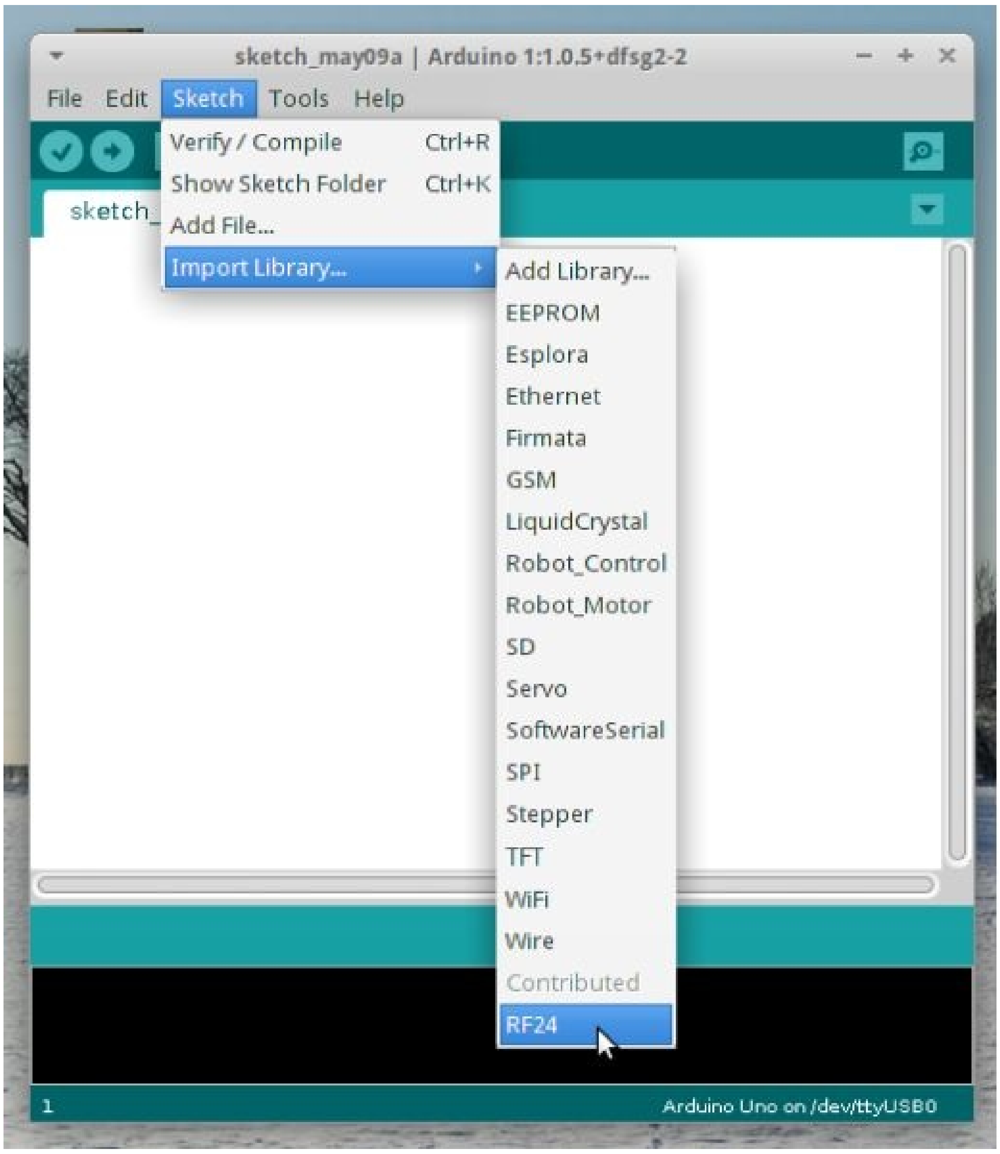
Here's a Fritzing wiring diagram for an Arduino Nano USB board on a mini breadboard:


NRF24L01
VCC CSN MOSI IRQ
GND
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RXD TX1
ARDUINO.CC
ARDUINO NANO V3.0
ICSP
TX RX PWR L
RST
USA 2009
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
fritzing

Installing ManiacBug's RF24 library for Arduino.

To make it simple for the Arduino to talk to the NRF24L01+ module, we need to install ManiacBug's RF24 library.

Go to the web page https://github.com/maniacbug/RF24 and click on the Download ZIP button on the right-hand side of the page. Rename the file RF24-master.zip to RF24.zip. Open the zip file and extract the RF24-Master folder. Rename the extracted folder RF24-Master to RF24.



Start up you Arduino IDE software & go to Sketch → Import Library → Add Library and select the unzipped & renamed RF24 folder. On the latest Arduino 1.6.4 on Windows you go to Sketch → Include Library → Add .ZIP Library → select the extracted RF24 folder (not the zip file).

Upload PIR transmitter code to the Arduino.

Once you have the RF24 library installed and all the wires connected up, it just remains to upload the file *transmitpir.ino* to the Arduino (We showed you how to do this in the earlier examples).

```
//Import needed libraries
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Declare Constants and Pin Numbers
#define CE_PIN   9
#define CSN_PIN 10

// Define the transmit pipe
const uint64_t pipe = 0xE8E8F0F0E1LL;

//Create Radio
RF24 radio(CE_PIN, CSN_PIN);

//Declare variables
int switchPin  = 7;
int value = 0;
int msg[1];

//setup
void setup() {
  pinMode(switchPin, INPUT);
  Serial.begin(9600);

}

//Main Loop
void loop() {
//read PIR status
  value = digitalRead(switchPin);
  if (HIGH == value) {
    //print motion detected to serial terminal
    Serial.println(“Motion detected”);
    // Prepare the radio module to transmit
    radio.begin();
    radio.openWritingPipe(pipe);
    radio.powerDown();
    delay(1000);
    radio.powerUp();
    // Send code for the sensor, use 0000 through to 8888
    String theMessage = “4444“;
    int messageSize = theMessage.length();
    for (int i = 0; i < messageSize; i++) {
    int charToSend[1];
    charToSend[0] = theMessage.charAt(i);
    radio.write(charToSend,1);
    }
    //terminate message is a 9
    msg[0] = 9;
    radio.write(msg,1);
    //kill power to radio
    radio.powerDown();
    delay(2000);
  }
}
```

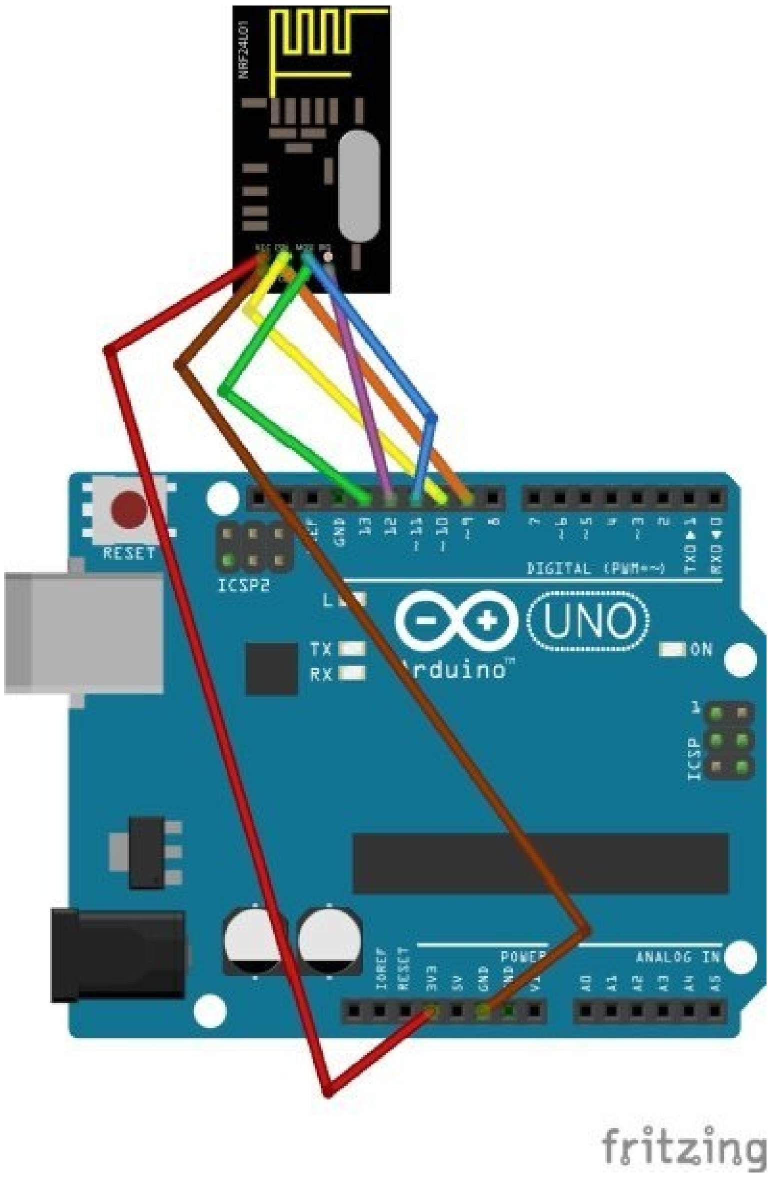
Once the transmitter sketch has been uploaded to the Arduino, it will run whenever you connect power. We used a 5V USB power bank to power our first transmitter node.

When anyone moves in front of the PIR, the Arduino prints “Motion Detected” to the Serial Monitor, sends a 4444 code message over the wireless module, and terminates the message by sending a number 9. If you have several Arduinos setup with PIR modules you can reuse the same transmitter program in each Arduino, but just change the transmit string to 3333 or 5555, so you can distinguish between them at the receiver. Don’t use the string 9999 though, as it will confuse the software.

To test the transmitter with the Arduino connected to the USB port on your PC, open up the Tools → Serial Monitor option in the Arduino IDE software and you should see a “Motion Detected” message each time you move near the PIR module.

To further test the transmitter, we need a 2nd Arduino running the receiver code attached to our PC.

Connect up a 2nd Arduino with a NRF24L01, just like we did for the transmitter unit, but this time there’s no need to attach a PIR sensor. We’ll then program the 2nd Arduino with the receiver code.


NRF24L01
RESET
ICSP2
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing

Upload receiver code to a 2nd Arduino.

Once you have the RF24 library installed and the transmitter setup, it just remains to upload the receiver code to a 2nd Arduino. (We already showed you how to do this in the earlier non-wireless examples) File is called *receiver.ino* on our DVD.

```
//setup required libraries
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SPI.h>

// setup variables and radio
int msg[1];
RF24 radio(9,10);
const uint64_t pipe = 0xE8E8F0F0E1LL;
int lastmsg = 1;
String theMessage = ””;

// setup radio
void setup(void){
 Serial.begin(9600);
 radio.begin();
 radio.openReadingPipe(1,pipe);
 radio.startListening();
}

// main loop
void loop(void){
 if (radio.available()){
   bool done = false;
   done = radio.read(msg, 1);
   char theChar = msg[0];
   if (msg[0] != 9){
   theMessage.concat(theChar);
   }
   else {
   Serial.println(theMessage);
```

```
    theMessage= ””;
    }
    }
}

```

If you have the receiver Arduino connected by a USB port to your PC, you can open the serial monitor (Tools → Serial Monitor) & see messages coming from the transmitter when the PIR module is triggered. If you’ve setup several Arduinos & PIRs as transmitters, then you’ll see messages like 3333, 4444, or 5555 depending which PIR has been triggered.

Occasionally you’ll see the message received as 444 instead of 4444, but we still have enough information to know which sensor has been triggered (see “Something Else” chapter near the end of the book for more on this). In the next example i’ll show you how to send an email alert to your phone, via the receiver Arduino connected to a PC, when the PIR has been triggered.

Setting up a spare Gmail account.

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

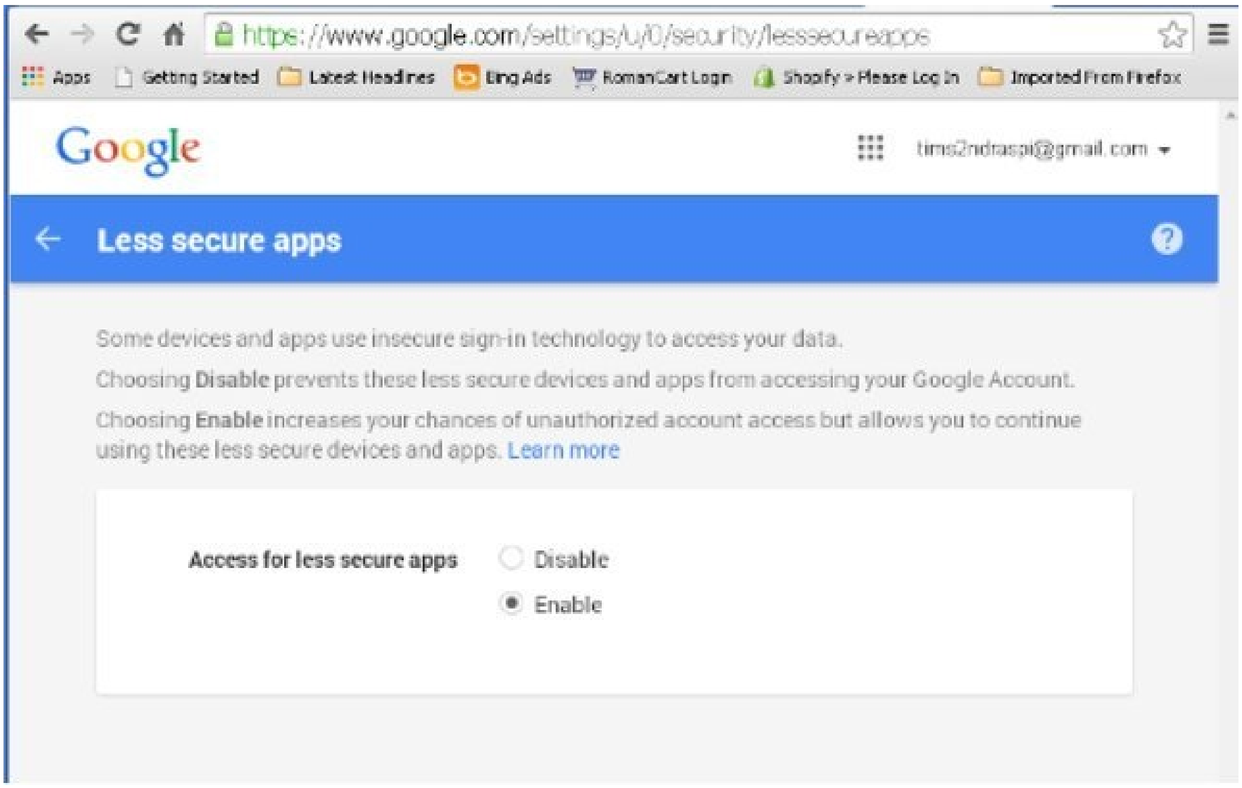
You need to create the new Gmail account in the web browser on your PC or Mac @

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

and note down the login & password for later.

Next, you need to set the new Gmail account to Enable “less secure apps”. While logged in go to:

https://www.google.com/settings/u/0/security/lesssecureapps


https://www.google.com/settings/u/0/security/lesssecureapps

Less secure apps

Some devices and apps use insecure sign-in technology to access your data.

Choosing **Disable** prevents these less secure devices and apps from accessing your Google Account.

Choosing **Enable** increases your chances of unauthorized account access but allows you to continue using these less secure devices and apps. Learn more

Access for less secure apps Disable [✓ Enable]

Now we have a working email account, just for the PC to use when sending emails from Python. Emails from the PC running Python can be sent to any other email address on your phone or a different PC.

Python script with email function to run on your PC.

First, go to the website gmail.com & create a new account, just for the Arduino to use. Do this even if you already have a Gmail account you use every day. Make sure you enabled “less secure apps” access to the gmail account. Use that new login & password below.

Launch Python 2.7 → IDLE (Python GUI) from Windows Program menu. File → Open → wirelesspir.py

You’ll need to change the text I’ve highlighted in red.

```
import serial
import time
import smtplib

TO = ‘your-phones-existing-email-adress@gmail.com‘
GMAIL_USER = ‘your-arduinos-new-email@gmail.com‘
GMAIL_PASS = ‘hard-to-guess-password‘

SUBJECT = ‘Intruder’
TEXT = ‘PIR Triggered’

def send_email():
    print(“Sending Arduino Email”)
    smtpserver = smtplib.SMTP(“smtp.gmail.com”,587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = ‘To:’ + TO + ‘\n’ + ‘From: ‘ + GMAIL_USER
    header = header + ‘\n’ + ‘Subject:’ + SUBJECT + ‘ ‘ + now + ‘\n’
    print header
    msg = header + ‘\n’ + TEXT + ‘ \n\n’
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()

while True:
    ser = serial.Serial(‘COM6‘, 9600)
    message = ser.read(4)
    now = time.ctime()
    print(message + (‘ ‘) + now)
    send_email()
    ser.close()
    time.sleep(10)

```

Change the TO = email address to the one you want to receive the alert, typically the email address you use on your smartphone. Change the GMAIL_USER and GMAIL_PASS values to the ones for the brand new gmail account you just made. Your serial port probably won’t be on COM6.

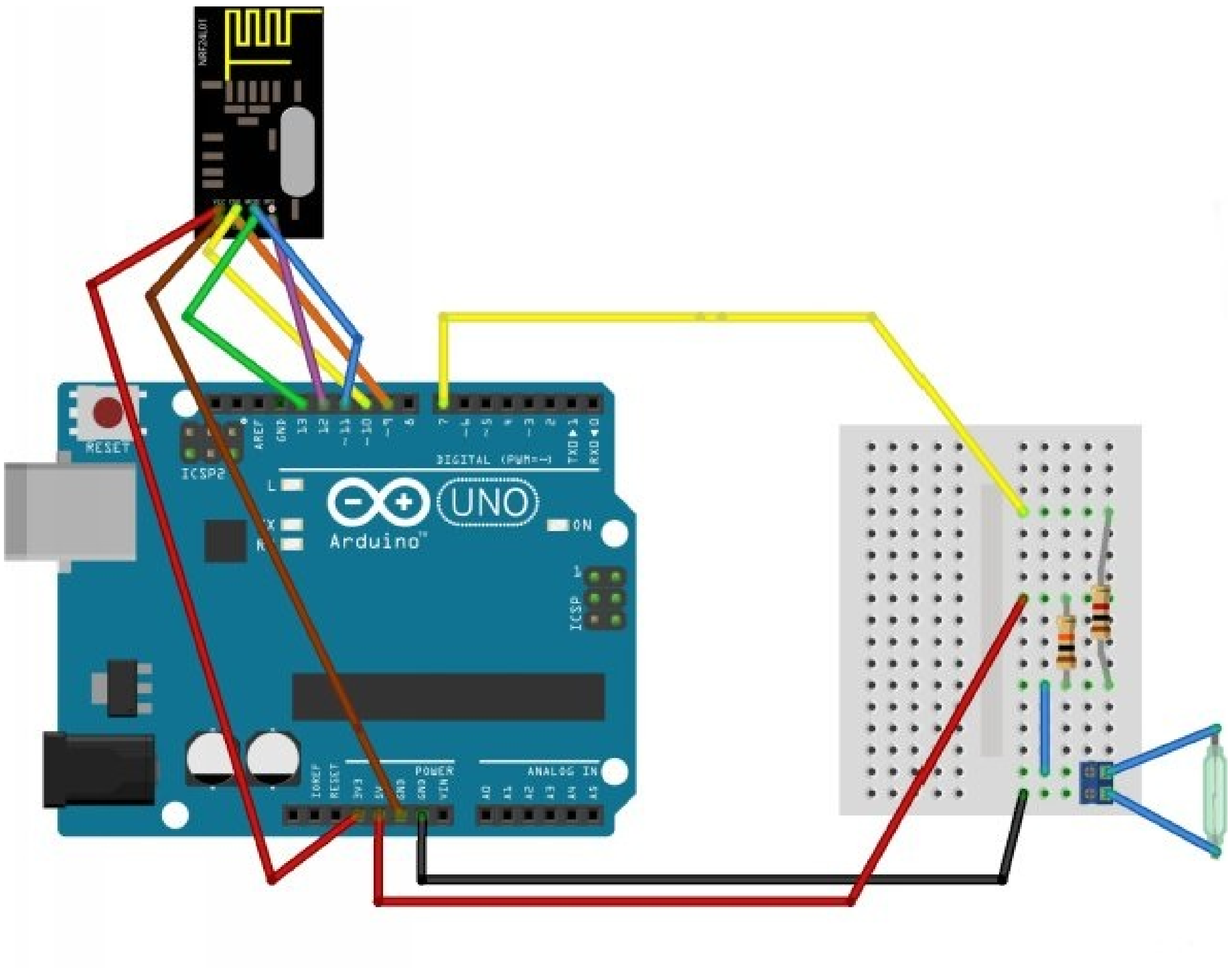
select Run → Run Module & you should receive an email on your phone when the PIR is triggered.

Arduino Uno R3 wireless magnetic door sensor

The link cables to the NRF24L01+ are standard 20cm long 40 way male to female DuPont breadboard cables available from eBay. The brown, red, orange, yellow, green, blue, violet cables are all next to each other in the 40 way cable we used, and can be separated into a 7 wire bunch.

The mini breadboard used for the magnetic door circuit uses a 1k (brown, black, red) & a 10k (brown, black, orange) resistor to provide a stable pull up/down voltage divider circuit.

Here’s a Fritzing wiring diagram for an Arduino Uno R3 board:

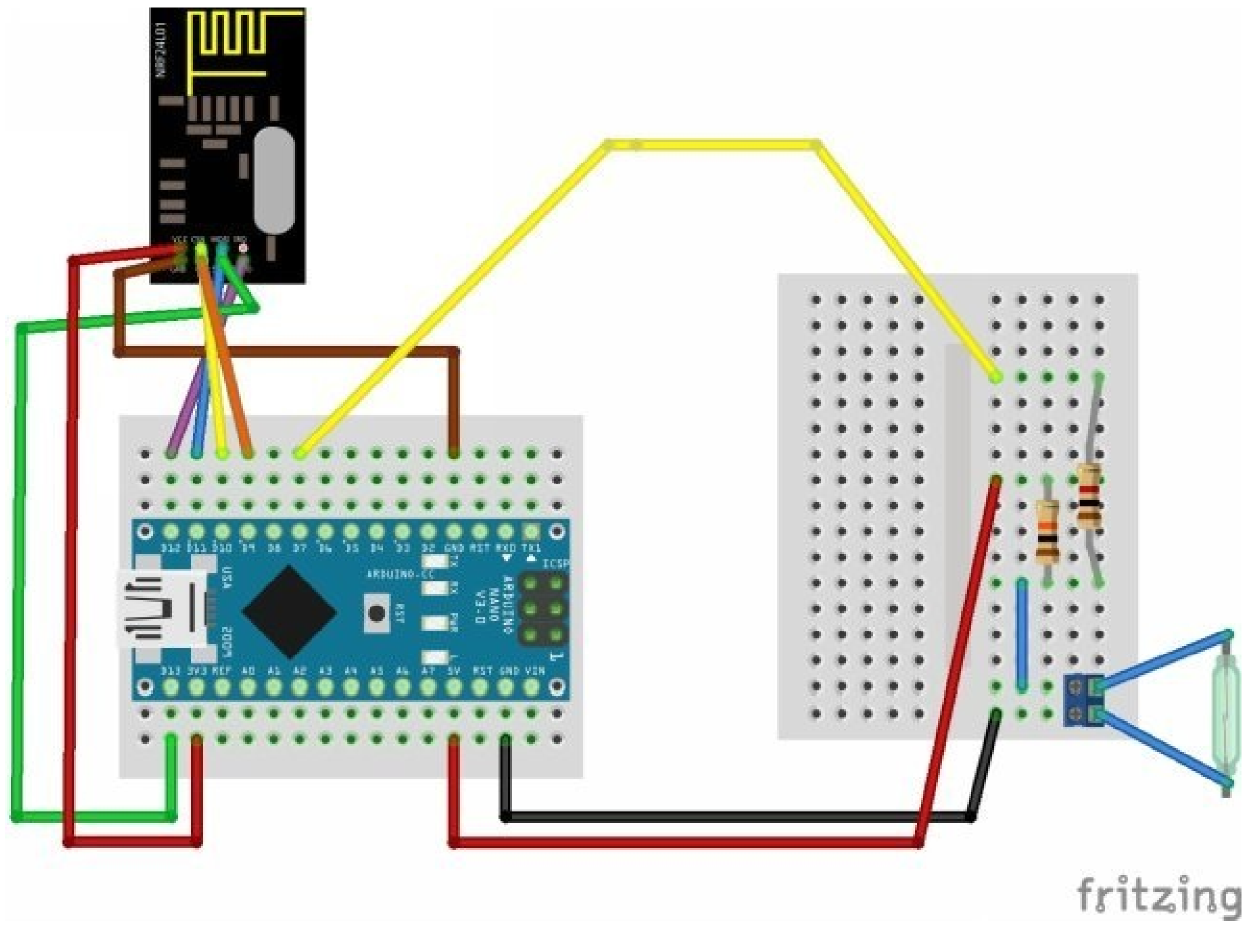


Arduino Nano with wireless magnetic door sensor

The link cables to the NRF24L01+ are standard 20cm long 40 way male to female DuPont breadboard cables available from eBay. The brown, red, orange, yellow, green, blue, violet cables are all next to each other in the 40 way cable we used, and can be separated into a 7 wire bunch.

The mini breadboard used for the magnetic door circuit uses a 1k & a 10k resistor to provide a stable pull up/down voltage divider circuit.

Here’s a Fritzing wiring diagram for an Arduino Nano USB board:



Upload transmitter code to the Arduino for magdoor.

Once you have the RF24 library installed and all the wires connected up, it just remains to upload the wireless magdoor transmitter code to the Arduino over USB. (we've showed you how in previous examples). Here's the magdoor alarm transmitter sketch, called *transmitdoor.ino*

```
//Import needed libraries
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Declare Constants and Pin Numbers
#define CE_PIN   9
#define CSN_PIN 10

// Define the transmit pipe
const uint64_t pipe = 0xE8E8F0F0E1LL;

// Create Radio
RF24 radio(CE_PIN, CSN_PIN);

//Declare variables - ds is door state and ods is old door state
int switchPin = 7;
int ds = 0;
int msg[1];
int ods = 0;


//setup
void setup() {
  pinMode(switchPin, INPUT);
  Serial.begin(9600);

}


//Main Loop
void loop() {
```

```
//read Door status
 ds = digitalRead(switchPin);
 if (HIGH == ds && ds != ods) {
    //print door opened to serial terminal
    Serial.println(“Door Opened”);
    String theMessage = “5555”;
    // Prepare the radio module to transmit
    radio.begin();
    radio.openWritingPipe(pipe);
    radio.powerDown();
    delay(1000);
    radio.powerUp();
    // Send code for the sensor, use 0000 through to 8888
    int messageSize = theMessage.length();
    for (int i = 0; i < messageSize; i++) {
    int charToSend[1];
    charToSend[0] = theMessage.charAt(i);
    radio.write(charToSend,1);
    }
    //terminate message is a 9
    msg[0] = 9;
    radio.write(msg,1);
    //kill power to radio
    radio.powerDown();
    delay(2000);
    ods = ds;
    }

    else if (HIGH != ds && ds != ods) {
    Serial.println(“Door Closed”);
    String theMessage = “6666”;
    // Prepare the radio module to transmit
    radio.begin();
    radio.openWritingPipe(pipe);
    radio.powerDown();
    delay(1000);
    radio.powerUp();
    // Send code for the sensor, use 0000 through to 8888
    int messageSize = theMessage.length();
    for (int i = 0; i < messageSize; i++) {
    int charToSend[1];
    charToSend[0] = theMessage.charAt(i);
    radio.write(charToSend,1);
    }
    //terminate message is a 9
    msg[0] = 9;
    radio.write(msg,1);
    //kill power to radio
    radio.powerDown();
    delay(2000);
    ods = ds;
    }

    else {
    ods = ds;
    }


}
```

If the door is opened the transmitter node sends a 5555 code over the NRF24L01+ link, and when the door is closed again the transmitter sends a 6666 code. The variables ds (door state) and ods (old door state) keep track of whether the magnetic door sensor reads 0 or 1, and if that state has just changed.

On the Arduino receiver side, we can use the same sketch we used for the PIR receiver (as we’re still just receiving 4 digit codes). We just need a different Python script running on the PC/Mac to interpret the fact the door has been opened or closed.

Python code for wireless magnetic door sensor.

Run this code on your PC or Mac running Python 2.7.8 IDLE.

```
import serial
import time

while True:
    ser = serial.Serial(‘COM8’, 9600)
    code = ser.read(4)
    now = time.ctime()
    if code == “4444”:
    message = “PIR triggered”

    elif code == “5555”:
    message = “Door Opened”

    elif code == “6666”:
    message = “Door Closed”

    else:
    message = “Something Else”

    print(message + (‘ ‘) + now)
    ser.close()
    time.sleep(1)


```

We’ve tested the code above with an Arduino Nano clone attached to a PIR sensor, a different Arduino Nano clone attached to a magnetic door sensor, and a Arduino Uno R3 clone running the NRF24L01+ receiver code attached to a PC.

The next example brings the magnetic door sensor & PIR sensor receiver codes together in a script that sends an email from your PC when they’re triggered.

Python code for wireless PIR & door sensor to email.

Run the code *wirelessdoor.py* on your PC or Mac running Python 2.7.8 IDLE.

```
import serial
import time
import smtplib

TO = ‘your-phones-email-address@gmail.com’
GMAIL_USER = ‘gmail-account-you-made-just-for-this@gmail.com’
GMAIL_PASS = ‘hard-to-guess-password’

NOW = ” “
message = ” “

def send_email():
    print(“Sending Email”)
    smtpserver = smtplib.SMTP(“smtp.gmail.com”,587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = ‘To:’ + TO + ‘\n’ + ‘From: ‘ + GMAIL_USER
    header = header + ‘\n’ + ‘Subject:’ + message + ‘ ‘ + NOW + ‘\n’
    print header
    msg = header + ‘\n’ + message + ‘\n’ + NOW + ‘ \n\n’
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()


while True:
    ser = serial.Serial(‘COM8’, 9600)
    code = ser.read(4)
    NOW = time.ctime()
    if code == “4444”:
    message = “PIR triggered”

    elif code == “5555”:
```

```
message = “Door Opened”

elif code == “6666”:
message = “Door Closed”

else:
message = “Something Else”

print(message + (‘ ‘) + NOW)
print(code)
send_email()
ser.close()
time.sleep(1)

```

Connect a LDR light sensor to the Arduino

Components required:

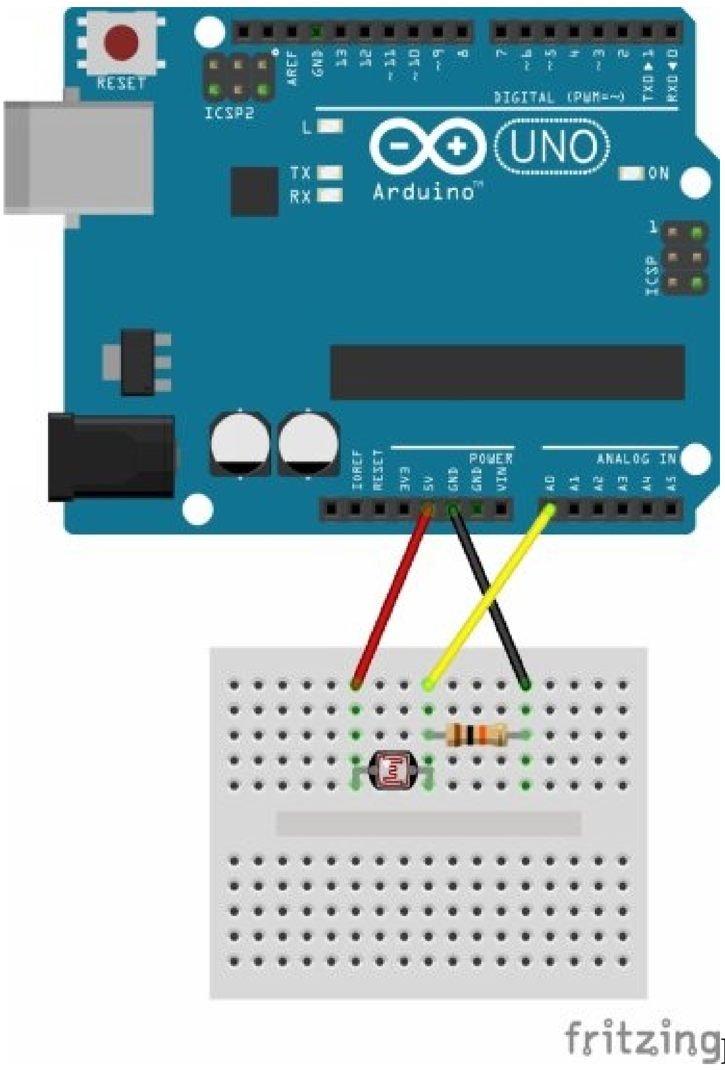
Mini Breadboard

10K resistor (brown, black, orange)

LDR – light dependent resistor

3x male to male IDC connecting cables

The LDR (light dependent resistor) varies its resistance depending how much light is falling on its face. We can read this value on the Arduino & use it to improvise a wardrobe, cupboard or drawer alarm. If the cupboard is opened, more light will be detected and that can trigger an email to our phone.



Here's how it's connected: (the colours of the wires you use isn't important, just the positions)

Arduino analog input pins.

The digital input pins on the Arduino detect whether a pin is high or low, or if you prefer set to 1 or 0. Any of the digital pins will only ever read high or low, but if we want to read from a sensor that can have a wide range of possible values we use an analogue input pin. There are six different analog input pins available.

Examples of sensors you can attach to the analog pins are: light sensors, humidity sensors, temperature sensors, gas sensors, alcohol sensors & water level sensors. The sensors connected to an analog input on the Arduino will give a value between 0 and 1023.

You can use the Arduino Sketches and Python scripts we've provided below to read any analog sensor connected to an Arduino & get it to send the value to a PC. Just Google your sensor type and see what supporting resistors etc are recommended for connecting it to an Arduino.

Upload the sketch LDR2 to the Arduino & then open Tools → Serial Monitor to see the values scroll up the screen. Cover the sensor with your finger & you'll see the value go below 200.

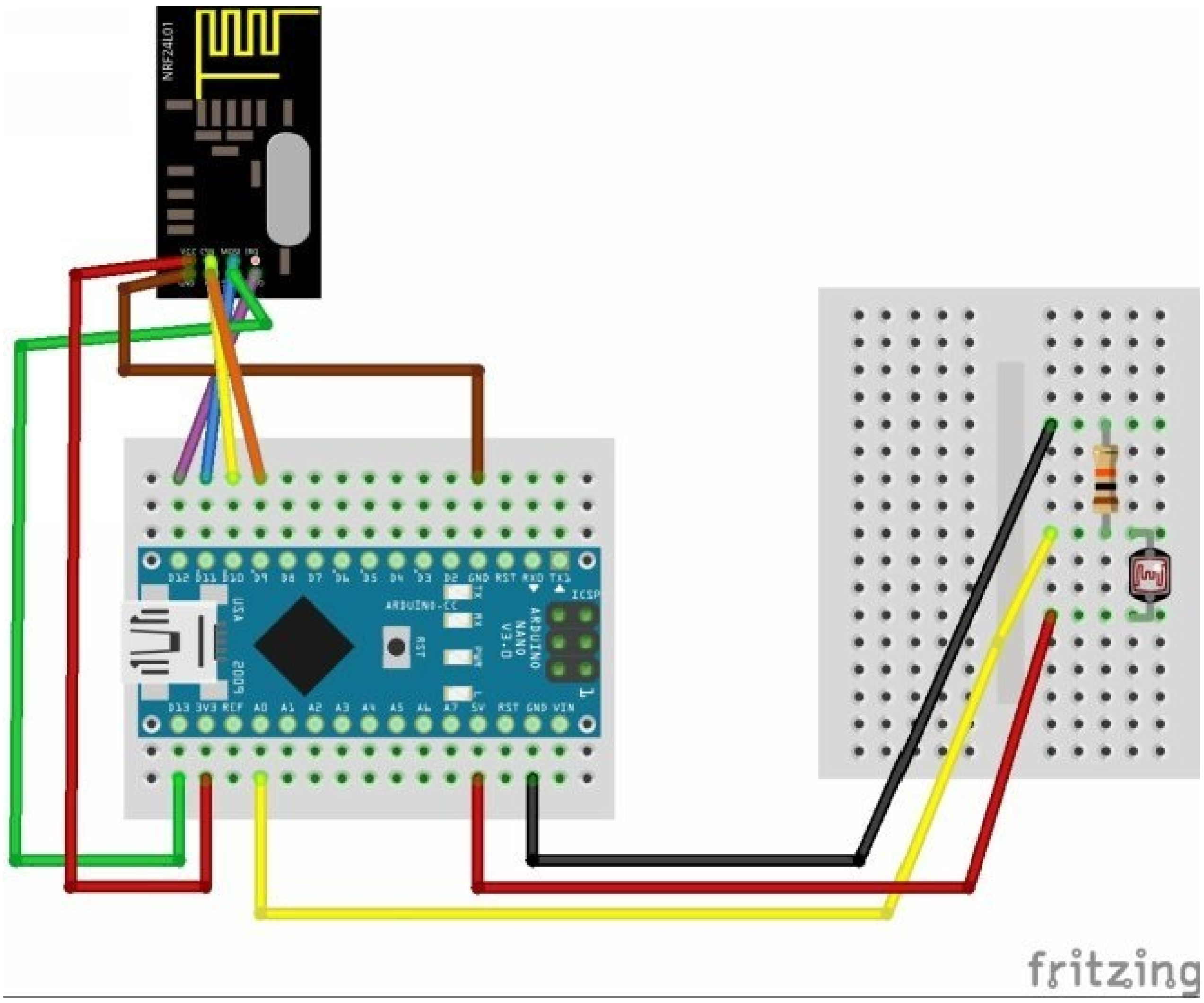
```

// sketch to read light value from LDR sensor - over 200 is light
// prints value to the serial terminal

int sensorPin = A0;    // select the input pin for the LDR sensor
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  Serial.begin(9600);
}

void loop() {
 // read the value from the sensor:
 sensorValue = analogRead(sensorPin);
 Serial.println(sensorValue);
 delay(500);
}
```

As a value below 200 can be consider darkness, we can use a trigger value of more than 200 to send an email saying our drawer or cupboard door has been opened. Use the sketch above to figure out the trigger value needed for your particular situation. In the next example we’ll connect the LDR sensor circuit to an Arduino Nano & NRF24L01+ board to make a wireless sensor version.

Make a wireless Arduino LDR light sensor.

Here's the Fritzing wiring diagram showing how to connect an Arduino Nano to the NRF24L01+ board and the light dependent resistor circuit. (You don't need to use the same colour wires as us, just connect the correct points together).

The idea behind this project is that you can place the Arduino (powered by a USB battery pack) inside a drawer, cupboard or wardrobe, and it will wirelessly send a trigger code to your PC, which in turn will email your phone when the light level increases & email you again when it goes back to darkness.


NRF24L01
VCC CSN MOSI IRQ
GND CE SCK MISO
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RXD TX1
ARDUINO-CC
ARDUINO NANO V3.0
ICSP
TX RX PWR L
RST
USA 2009
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
fritzing

Sketch code for wireless Arduino LDR light sensor.

Upload the following Sketch to your Arduino board (called *transmitldr.ino* on our DVD/Server).

```
//Import needed libraries
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//Declare Constants and Pin Numbers
#define CE_PIN   9
#define CSN_PIN 10

// Define the transmit pipe
const uint64_t pipe = 0xE8E8F0F0E1LL;

// Create Radio
RF24 radio(CE_PIN, CSN_PIN);

//Declare variables
// ds is drawer state and ods is old drawer state
int ds = 0;
int msg[1];
int ods = 0;
int sensorPin = A0;    // select the input pin for the LDR sensor


//setup
void setup() {
    Serial.begin(9600);

}


//Main Loop
void loop() {
```

```
//read Drawer status
 ds = analogRead(sensorPin);
 if (ds > 200 && ods <= 200)  {
   //print door opened to serial terminal
   Serial.println("Drawer Opened");
   Serial.println(ds);
   Serial.println(ods);
   String theMessage = "7777";
   // Prepare the radio module to transmit
   radio.begin();
   radio.openWritingPipe(pipe);
   radio.powerDown();
   delay(1000);
   radio.powerUp();
   // Send code for the sensor, use 0000 through to 8888
   int messageSize = theMessage.length();
   for (int i = 0; i < messageSize; i++) {
   int charToSend[1];
   charToSend[0] = theMessage.charAt(i);
   radio.write(charToSend,1);
   }
   //terminate message is a 9
   msg[0] = 9;
   radio.write(msg,1);
   //kill power to radio
   radio.powerDown();
   delay(500);
   ods = ds;
   }

   else if (ds < 200 && ods >= 200) {
   Serial.println("Drawer Closed");
   Serial.println(ds);
   Serial.println(ods);
   String theMessage = "8888";
   // Prepare the radio module to transmit
   radio.begin();
   radio.openWritingPipe(pipe);
```

```
    radio.powerDown();
    delay(1000);
    radio.powerUp();
    // Send code for the sensor, use 0000 through to 8888
    int messageSize = theMessage.length();
    for (int i = 0; i < messageSize; i++) {
    int charToSend[1];
    charToSend[0] = theMessage.charAt(i);
    radio.write(charToSend,1);
    }
    //terminate message is a 9
    msg[0] = 9;
    radio.write(msg,1);
    //kill power to radio
    radio.powerDown();
    delay(500);
    ods = ds;
    }

    else {
    ods = ds;
    }
}

```

When the Sketch is running on your Arduino, open the Tools → Serial Monitor screen, and you’ll see the message “Drawer Opened” appear when the light level goes above 200, along with the contents of the DS & ODS variables (I’ve left them in for troubleshooting purposes) the radio module then transmits code 7777.

When the light level drops below 200, you’ll see the “Drawer Closed” message, along with the DS & ODS values. The radio module then sends the 8888 code.

We use the 2nd Arduino & NRF24L01+ board receiver setup from the previous experiments to receive the codes on our PC or Mac. Next I’ll show you the Python code to run on the PC.

Python code for wireless Arduino LDR light sensor.

Run the Python example *wirelessldr.py* on your PC or Mac running Python 2.7.8 IDLE. It detects the 7777 or 8888 codes coming over the USB connection from the receiver Arduino & sends an email alert.

Remember to change the items highlighted in red to suit your setup.

```
import serial
import time
import smtplib

TO = ‘your-mobile-phones-email@gmail.com’
GMAIL_USER = ‘new-mail-account-you-just-made@gmail.com’
GMAIL_PASS = ‘hard-to-guess-password’

NOW = ” “
ODS = “C”
message = ” “

def send_email():
    print(“Sending Email”)
    smtpserver = smtplib.SMTP(“smtp.gmail.com”,587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = ‘To:’ + TO + ‘\n’ + ‘From: ‘ + GMAIL_USER
    header = header + ‘\n’ + ‘Subject:’ + message + ‘ ‘ + NOW + ‘\n’
    print header
    msg = header + ‘\n’ + message + ‘\n’ + NOW + ‘ \n\n’
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()


while True:
```

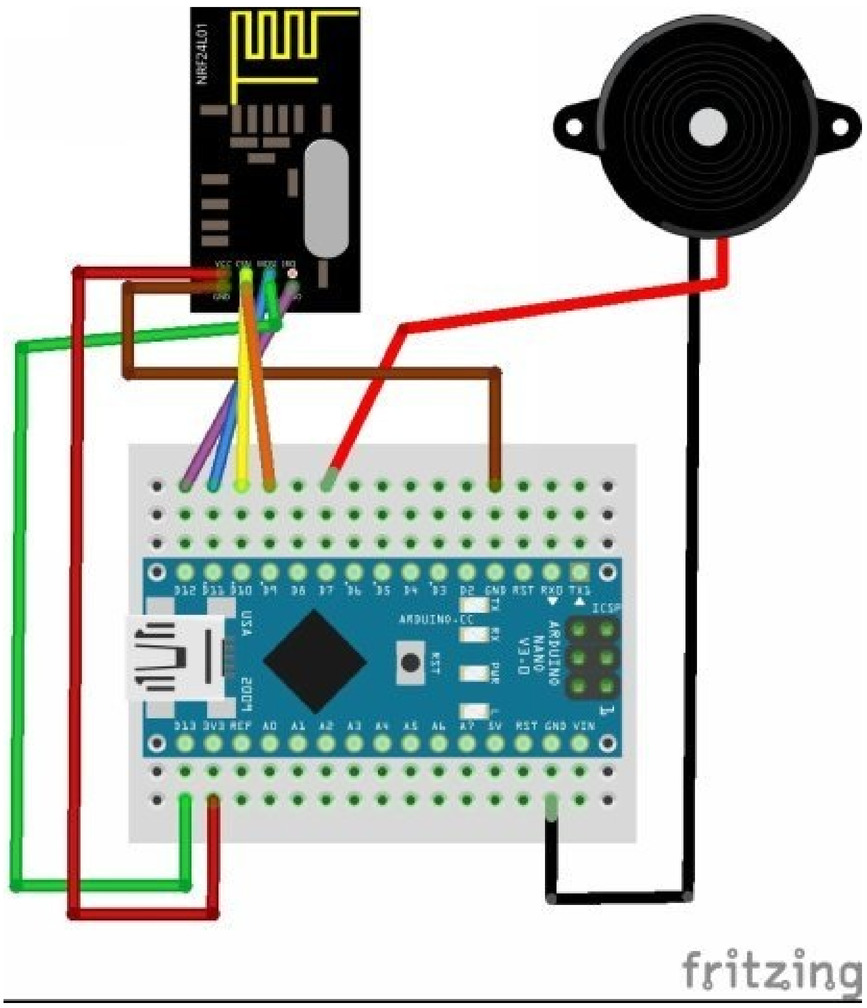
```
ser = serial.Serial(‘COM8‘, 9600)
code = ser.read(4)
NOW = time.ctime()
if code == “7777”:
message = “Drawer Opened”

elif code == “8888”:
message = “Drawer Closed”

else:
message = “Something Else”

print(message + (‘ ‘) + NOW)
print(code)
send_email()
ser.close()
time.sleep(1)
```

Make a standalone battery powered beeping receiver.

If your wireless transmitter is outside somewhere & you don’t have access to a PC to attach your receiver to, then it can be handy to have a battery powered Arduino receiver that can give you an indication that the alarm has been triggered.

In this example we’ll use a 5 volt beeper to sound for half a second when a valid code 4 digit alarm code is received.

Here’s how to wire everything together:



Arduino & NRF24L01 receiver portable beeper Sketch.

Upload sketch *receiverbeep.ino* to your Arduino. The sketch expects the beeper + side to be connected to digital pin 7 & the beeper’s - to GND.

```
//setup required libraries
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SPI.h>
// setup variables and radio
int beepPin = 7;
int msg[1];
RF24 radio(9,10);
const uint64_t pipe = 0xE8E8F0F0E1LL;
int lastmsg = 1;
String theMessage = ””;

// setup radio
void setup(void){
 Serial.begin(9600);
 pinMode(beepPin, OUTPUT);
 radio.begin();
 radio.openReadingPipe(1,pipe);
 radio.startListening();
}

// main loop
void loop(void){
 if (radio.available()){
  bool done = false;
   done = radio.read(msg, 1);
   char theChar = msg[0];
   if (msg[0] != 9){
    theMessage.concat(theChar);
    }
    else {
     Serial.println(theMessage);
     digitalWrite(beepPin, HIGH);
     delay(500);
     digitalWrite(beepPin, LOW);
     theMessage= ””;
    }
  }
}

```

You can now run the Arduino receiver unit from a portable USB power without being attached to a PC. When a valid 4 digit code is received the beeper will sound for half a second - delay(500).

You could change the sketch around so that different 4 digit codes give different duration beeps.

Setting up Arduino IDE software on a Raspberry Pi.

Although we've said you need to connect the Arduino to the USB port on a PC or Mac, it's also possible to connect the Arduino board to a Raspberry Pi's USB port.

If you want to program the Arduino boards from the Raspberry Pi, rather than a PC or Mac, then you need to install the Arduino software. Open the Terminal on your Pi & type in:

```
sudo apt-get update
sudo apt-get install arduino
```

Currently that installs Arduino IDE 1.0.1, which works fine (current PC/Mac version is 1.6), but then you also need to manually install the RF24 library.

Download ManiacBug's RF24 library from https://github.com/maniacbug/RF24 by clicking on the Download ZIP button near the lower-right side of the page. Unzip the folder to your Desktop and rename the folder RF24-master as RF24. Quit the Arduino software if it's already running.

Move the RF24 library to the correct location on the Pi, with:

```
sudo cp /home/pi/Desktop/RF24* /usr/share/arduino/libraries -r
```

Then start the Arduino software at Menu → Electronics → Arduino IDE and go to Sketch → Import Library and you should see RF24 listed.

Next you need to download our Arduino Sketches & Unzip them

```
wget http://www.securipi.co.uk/arduino-sketches.zip
```

Unpack them with

```
unzip arduino-sketches.zip
```

You can now upload any of the Arduino sketches to your Uno or Nano board. For example receiverbeep.ino

Go to File → Open and select the Sketch to upload to the board.

Go to Tools → Board and select either Uno or Arduino Nano w/ATmega328

Then Tools → Serial Port and select /dev/ttyUSB0

Then click the Blue Right Arrow icon (underneath Edit) to upload your Sketch.

Using the Python scripts on a Raspberry Pi

You can use the Raspberry Pi desktop or command line to run our Python 2 scripts. (don't use Python 3).

You need Python 2.7.3 installed to run these scripts, the default on my Pi. Check your version with:

```
python -V
```

Download them to your Pi's command line with

```
wget http://www.securipi.co.uk/arduinopython.zip
```

Unpack them with

```
unzip arduinopython.zip
```

See a list of them with

```
ls -al
```

Show the USB serial ports in your Pi with

```
ls -l /dev/ttyUSB*
```

Lets assume your Arduino is on /dev/ttyUSB0.

Edit *wirelesstest.py* using nano and replace the reference to COM8 with /dev/ttyUSB0

```
nano wirelesstest.py
```

Save the file (CTRL-O) & exit (CTRL-X) nano.

Run the Python script

```
python wirelesstest.py
```

Make the same changes to python scripts *wirelesspir.py*, *wirelessdoor.py* & *wirelessldr.py*

If you notice the time on the Raspberry Pi is incorrect, change it with

```
sudo dpkg-reconfigure tzdata
```

Python script to capture from Pi Camera module.

This script is in our download as wirelesspirpi2.py. Make sure you change the email addresses highlighted in red. Launch it with

```
python wirelesspirpi2.py
```

Make sure you've enabled "less secure apps" in the Gmail settings too.

```
#!/usr/bin/python
# make a new gmail acount and enter the correct details below
# change the email addresses we've used to match your sender & recipient
# this script works with Raspberry Pi Camera plugged into a Raspberry Pi

import serial
import time
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
import os

gmail_user = "your-new-email-address@gmail.com"
gmail_pwd = "hard-to-guess-password-goes-here"

def mail(to, subject, text, attach):
  msg = MIMEMultipart()

  msg['From'] = gmail_user
  msg['To'] = to
  msg['Subject'] = subject

  msg.attach(MIMEText(text))

  part = MIMEBase('application', 'octet-stream')
  part.set_payload(open(attach, 'rb').read())
  Encoders.encode_base64(part)
  part.add_header('Content-Disposition',
        ‘attachment; filename=”%s”’ % os.path.basename(attach))
  msg.attach(part)

  mailServer = smtplib.SMTP(“smtp.gmail.com”, 587)
  mailServer.ehlo()
  mailServer.starttls()
  mailServer.ehlo()
  mailServer.login(gmail_user, gmail_pwd)
  mailServer.sendmail(gmail_user, to, msg.as_string())
  mailServer.close()

while True:
   ser = serial.Serial(‘/dev/ttyUSB0’, 9600)
   message = ser.read(4)
   now = time.ctime()
   print(message + (‘ ‘) + now)
   os.system(‘raspistill -o image.jpg -hf’)
   mail(“you@your-phones-email-address.com“,
  “PIR motion alarm”,
  “This is an email sent with Python from your Raspberry Pi”,
  “image.jpg”)
   os.system(‘rm image.jpg’)
   ser.close()
   time.sleep(10)
```

Something Else

I said earlier in the instructions that sometimes a four digit code (like 5555) might be received as only three digits (like 555) at the receiver, and that we could account for that in the software if you find you're receiving lots of "Something Else" messages.

All we need to do is only use the 1st digit of the 4 digit message, which in this example will be 5. So look at the Python code examples that run on your PC, and any line that looks like

```
elif code == "5555":
```

Can be changed to

```
elif code[0] == "5":
```

Here's a full example. You still need to modify the items in red text. The lines in green have been modified.

```
import serial
import time
import smtplib

TO = 'your-phones-email-address@gmail.com'
GMAIL_USER = 'your-new-gmail-account@gmail.com'
GMAIL_PASS = 'hard-to-guess-password-goes-here'

NOW = " "
message = " "

def send_email():
    print("Sending Email")
    smtpserver = smtplib.SMTP("smtp.gmail.com",587)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = ‘To:’ + TO + ‘\n’ + ‘From: ‘ + GMAIL_USER
    header = header + ‘\n’ + ‘Subject:’ + message + ‘ ‘ + NOW + ‘\n’
    print header
    msg = header + ‘\n’ + message + ‘\n’ + NOW + ‘ \n\n’
    smtpserver.sendmail(GMAIL_USER, TO, msg)
    smtpserver.close()


while True:
    ser = serial.Serial(‘/dev/ttyUSB0‘, 9600)
    code = ser.read(4)
    NOW = time.ctime()
    if code[0] == “4”:
        message = “PIR triggered”

    elif code[0] == “5”:
        message = “Door Opened”

    elif code[0] == “6”:
        message = “Door Closed”

    else:
        message = “Something Else”

    print(message + (‘ ‘) + NOW)
    print(code)
    print(code[0])
    send_email()
    ser.close()
    time.sleep(1)
```

Useful links:

https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo