

## GrovePI and Raspberry PI Introduction Labs

This worksheet is for the first two labs and is designed to get you started reading some data from the GrovePI sensors. If you find you finish the worksheet early, then feel free to start work on your project or your proposal. If this is all pretty obvious to you, then the same applies. We're around in the lab if you struggle, feel free to ask us if you have any questions.

This also contains some useful reference materials for you to use during your coursework:

**Appendix A describes the available sensors, and how to connect them to the PI.**

**Appendix B describes how to copy files onto the PI, and how to run python scripts via Secure Shell (SSH).**

We recommend you read this on a computer, so you can copy/paste the code snippets when needed.

### Part 1: My first sensor reading(s)

Open the Notepad++ text editor and write out in the following (note that the **indentation** of the program is important when you are typing it into the text editor):

```
import grovepi
import time

while True:
    #read from a digital sensor on input 4
    d= grovepi.digitalRead(4)
    #read from an analog sensor on input 0
    a= grovepi.analogRead(0)

    # output the data
    print ("D:",d)
    print ("A:",a)

    # Read roughly 10 times a second
    # - n.b. analog read takes time to do also
    time.sleep(0.1)
```

Save this file as **myfirstsensor.py** or whatever you want to call it (it must end in **.py**). In Notepad++, you must make sure that the file has 'unix line endings', go to 'edit', 'EOL-Conversion', and select 'UNIX/OS X Format'.

Connect two sensors to the GrovePI board, one digital one (e.g. motion, button, touch) to digital port 4 (D4), and one analog one to analog port 0 (A0).

Power up the raspberry PI and copy and run the script over SSH (see section "Connecting to the PI and running a Python script" for how to do this).

You should see a stream of sensor values being displayed, one pair per second. Mess with the sensors to trigger them and see how the values change.

Press Ctrl-C in order to quit the script.

## Part 2: Logging some sensor data

Enter the following into a new python script.

```
import grovepi
import time

# output a CSV header to the file
print ("time,digital,analog")
while True:
    #read from a digital sensor on input 4
    d=grovepi.digitalRead(4)

    #read from an analog sensor on input 0
    a= grovepi.analogRead(0)

    # get a timestamp so we know when it happened
    timestamp = time.time()

    #output all sensor values as comma separated values
    print ("%f,%d,%d"%(timestamp,d,a))

    # Read roughly 10 times a second
    # - n.b. analog read takes time to do also
    time.sleep(0.1)
```

Run this on the PI, but this time write the output to a file, **test.csv** and copy it back. Open the file in Notepad++ and it should look something like this:

```
time,digital,analog
1413298087.61 , 0 , 178
1413298088.64 , 0 , 137
1413298089.66 , 0 , 139
```

Or open it in Excel, have a play with it, graph a sensor reading etc.

## Part 3: Running Code in the GrovePI Emulator

The GrovePI Emulator lets you run python scripts on a PC, whilst pretending it is running on a GrovePI. You can press buttons and set sensor values in the emulator to test the logic of your code. It also lets you run against CSV files which you saved from a real raspberry PI.

First, get the grovepi emulator to a folder on your PC. It lives at:

<https://github.com/joemarshall/grovepi-emulator>

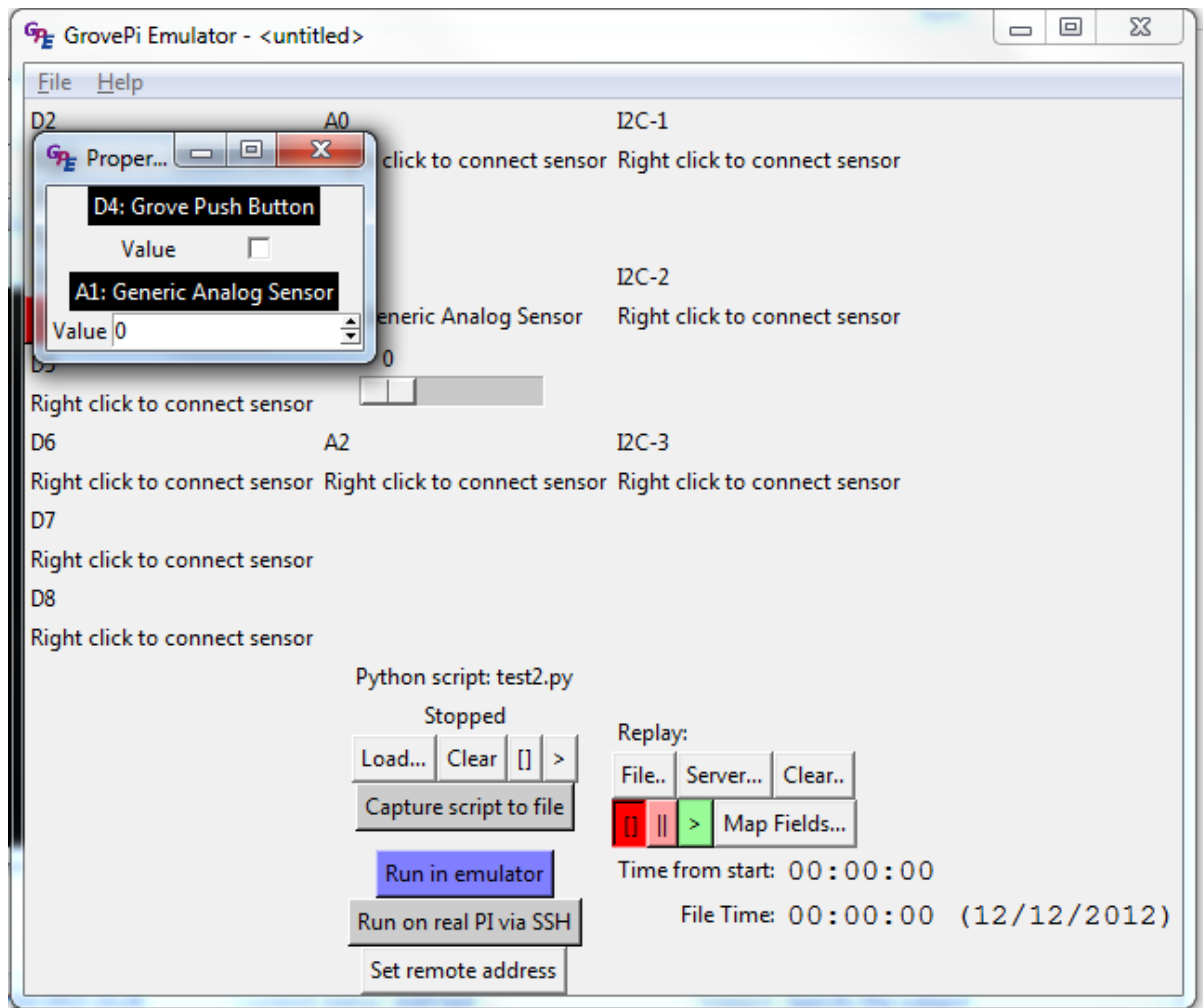
(You can download a zip file using the link below)

<https://github.com/joemarshall/grovepi-emulator/archive/master.zip>

If it is your own PC, you need Python 2.7 or later installed.

Extract all the files to a folder, and run grovepiemu.py (you should be able to just double click on the file to run it).

A window should open that looks a bit like the image below. There is also another black 'console' window which opens behind it, this is important as it is where your program output will appear.



Each column in the main window represents a set of grovepi connections, you can right click on any connection to put a virtual sensor there. Right click on D4 and put the same sensor you used in part 1 there, and do the same with connection A0.

Now, click 'Load' in the python script section, bottom of the middle column, and select the script you wrote in either part 1 or part 2. Click the 'play' button, and the script will start running. If you click on the push-button for your digital sensor, or change your analog value, you will see the values that the script is showing in the console window change. You can press the stop button on the script panel to stop your script.

So, you are now running your grovepi code on the fake sensors inside the emulator. What else can it do?

### Run code against your saved data

In Part 2, you created a .csv file whilst you messed around with the real sensors. Make sure you have this on the computer you're working on. In the emulator, there is a panel on the bottom right which controls data sources, click the 'File...' button there and select your CSV. It will ask you to select which columns go to which sensor (and choose one column for time – CSV sensor log files should always have a timestamp written into them). Then press the green play button just below it. The

time display in that panel should count up, and you should be able to see the analog value changing over time, and the digital value lighting up green or red depending on whether it is on or not.

You might find your CSV file is quite long with only a few interesting bits. If you open your CSV file, you can select just the bit you are interested in by deleting all the other lines, and replay just that. Make sure you always leave the line in at the top of a CSV file that contains the column names though.

### Quickly launch code on a real raspberry PI

If you select 'run on real PI via ssh', the grovepi emulator will upload your code to a real raspberry PI and run it. This is a lot quicker than using an ssh client manually. Click 'set remote address' to enter the IP address of your raspberry PI. You need to enter 'username@address'. On lab Pis, the username is g54mrt, so enter [g54mrt@192.168.1.1](mailto:g54mrt@192.168.1.1) or whatever the ip address is.

### Record output of your code to a csv file

If you click 'capture script to file', it will capture the output of the program to a text file.

### Pre-generate skeleton data capture code

If you set up a bunch of sensors in the emulator, then go to **File, Write Python for Current Sensors**, it will generate you a python file which will read from those sensors and then output the sensor readings as comma separated values.

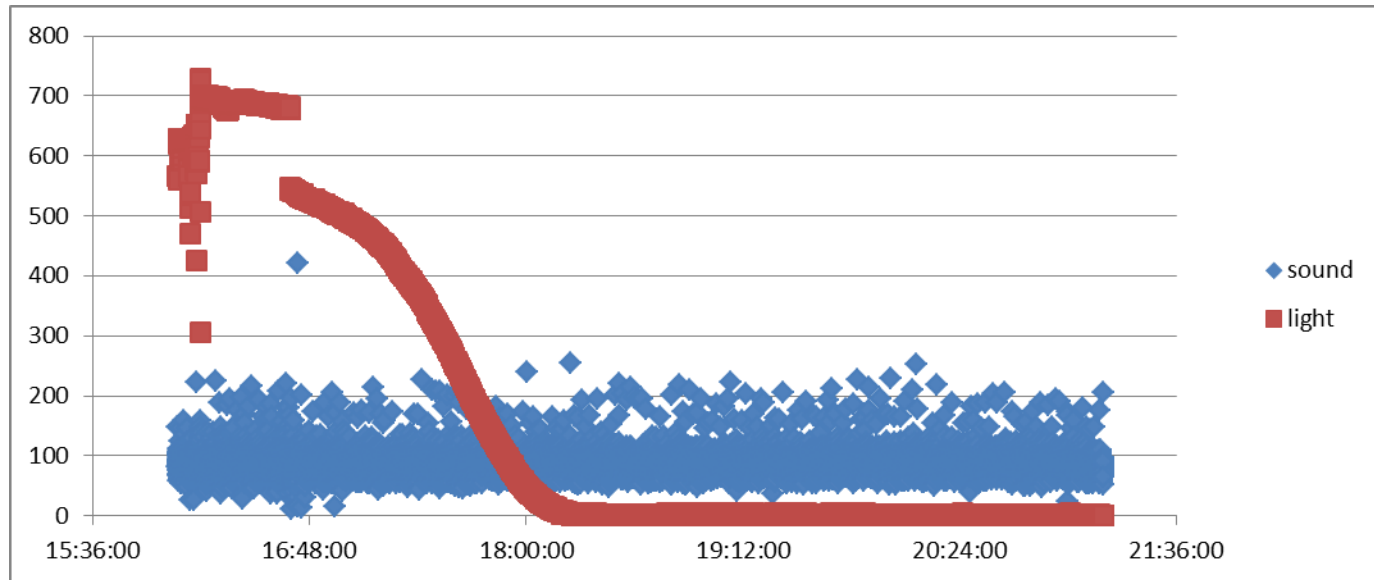
## Part 5: Open some data in Excel and graph it

You've produced some CSV files in the sections above.

Now it's time to do a couple of things with it:

- 1) Fix the dates, by adding a new column using a formula like this  
`=A13/(60*60*24)+"1/1/1970"`  
This formula converts from a python / unix timestamp as return from `time.time()` in python, into an excel time.
- 2) Make a graph of two columns (sensor streams) against each other.

e.g.



If you look at this example, you can already see the characteristics of the two sensors – light is very smooth, whereas sound is very variable and will obviously require more processing.

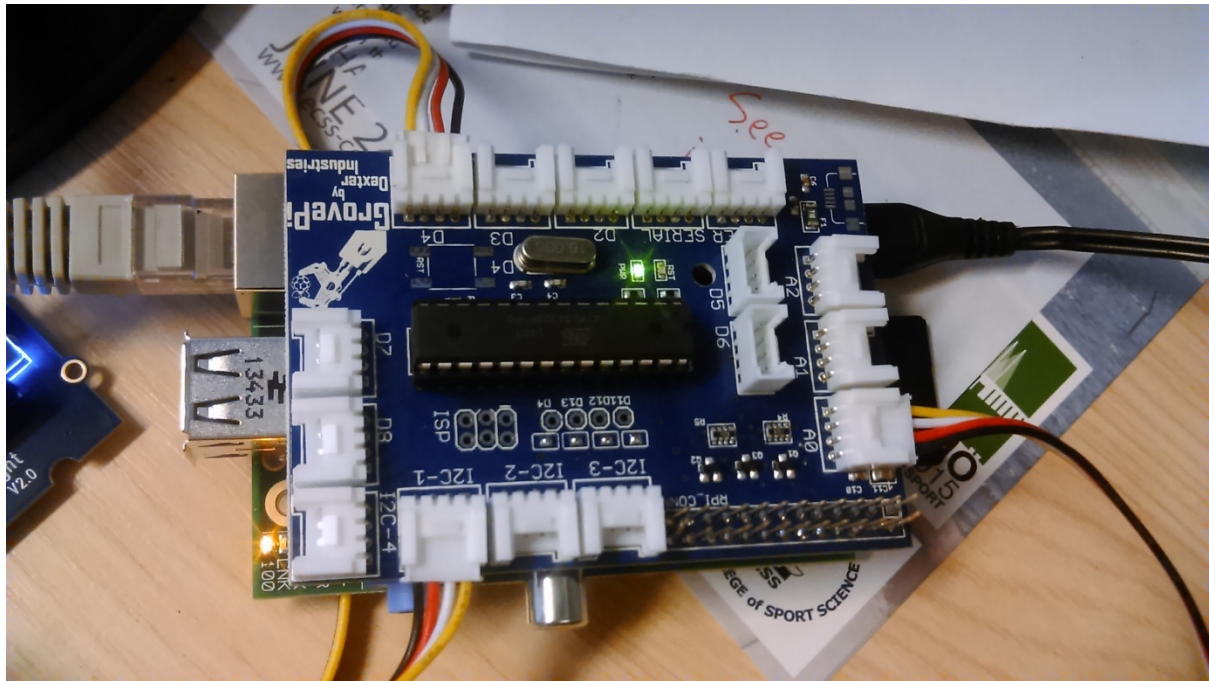
With either your data or the graph above, think about:

- What might you be able to tell from the sensor data?
- Are the sensors strongly linked or correlated?
- What do you think was happening?
- How noisy / smooth are your sensors?

## Appendix A: Connecting sensors to the PI

Sensors come in three flavours, the GrovePI board has 3 different sets of input, one for each flavour of sensors.

All sensors connect using the Grove ribbon cables to the white connectors on the GrovePI. Plug one end into the sensor, the other end into the PI.



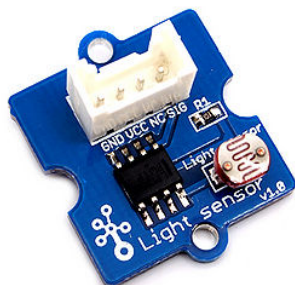
**Make sure that the PI is powered down when you plug things in**, especially to I2C, you can usually get away with plugging digital or analog sensors in while powered up, but try not to do it too often.

### Analog sensors

These connect to the ports marked A0, A1 and A3 on the GrovePI board. Analog sensors give a value over a range from 0 to 1024. Further info about all these sensors can be accessed here:

<http://wiki.seeedstudio.com/Sensor/>

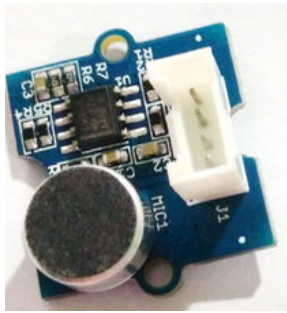
#### Light:



Measures how much light is falling on the sensor. Intensity of the light is proportional to resistance, calculated as below:

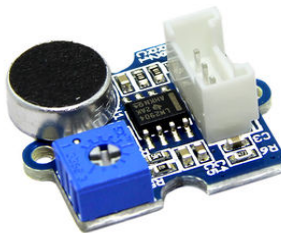
$$\text{resistance} = (\text{float})(1023 - \text{sensorValue}) * 10 / \text{sensorValue}$$

### Sound:



A microphone, digitizes sound waves. Not as useful as the loudness sensor, as it doesn't do much filtering and is a bit too noisy to use without masses of processing.

### Loudness:



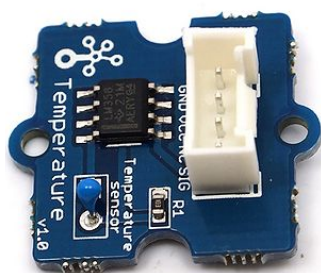
Similar to sound sensor, but has some filtering so that it responds to the amplitude of sound in a much more nice way. Still needs a bit of processing, but is quite usable (for example I used it in a room with a ticking clock, and the tick was easy to detect in the signal).

### Rotary angle sensor:



Potentiometer - measures what angle the dial is set to, gives 0 at one end and 1024 turned all the other way.

### Temperature:



Measures the temperature. Gives a value from 0..1024, which is related to resistance and then temperature by the calculation below:

```
#get the resistance of the sensor
```

```
resistance= (1023.0-float(sensorValue))*10000.0/ float(sensorValue)
```

```
# get the temperature from that (in degrees C) 3975 is a magic number
```

from the sensor datasheet

```
temperature=1/(math.log(resistance/10000)/3975.0+1/298.15)-273.15
```

## Digital sensors

These connect to ports marked D2,D3,D4,D5,D6,D7 and D8. Digital sensors are either on or off, and output either 0 or 1.

### Button:





It's a button, when it is pressed down, it outputs 1, when it is released, it outputs 0. Enough said.

#### Tilt switch:



Like a button, but turns on when it is tipped on its side.

#### Touch:



Capacitive touch sensor - like a button, but you just have to put your finger near or touch the sensor for it to output 1.

#### PIR motion sensor:



Detects movement of people (or animals, or warm objects) in the area in front of the dome.

#### Ultrasonic ranger:



Detects the closest object in front of it by using ultrasonic pulses. This is connected to a digital pin and read by sending a pulse out of the pin and then reading a response, which comes back on the same pin. Because of this, you need to use the special `grovepi.ultrasonicRead` function to get a value. The functions return the distance in cm of the closest object.

```
import grovepi
```



```
while true:
    ultrasoundVal=grovepi.ultrasonicRead(4)
    print (ultrasoundVal)
```

### Digital Temperature and Humidity:



This measures temperature and humidity. It uses a custom serial communication protocol between the grove PI and the sensor, so you can't just read it with `grovepi.digitalRead(pin)`, instead, connect it to a digital pin and run

```
import grovepi
while true:
    (temp,humidity)=grovepi.dht(pin,0)
    print (temp,humidity)
```

### I2C-based sensors

These are more 'clever' sensors, which connect using a serial bus called I2C. These require special code to talk to them. We've written code for all the sensors that we have (see lecture 4 and moodle). These modules are already on the PI, and have pydoc documentation in the code (or in html files in moodle under **Coursework Practical Resources and GrovePi software**).

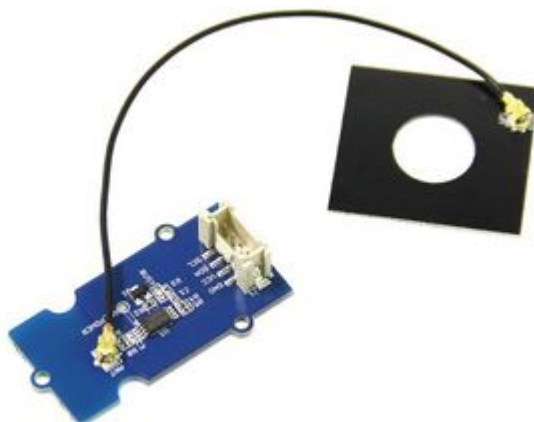
### LCD RGB Backlight



A dead handy little two line 16x2 character display, useful for status information etc. Has a multi-coloured backlight that you can control too.

```
import grovelcd
grovelcd.setText("Hello,\nWorld")
grovelcd.setRGB(0,128,255) # colour
is red,green,blue, each can be
0...255
```

### NFC tag

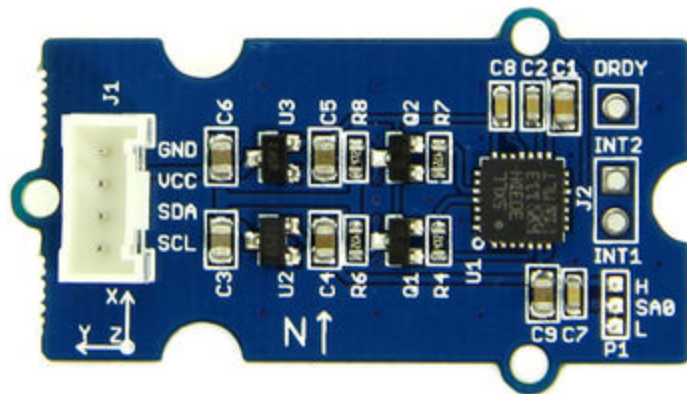


This is a programmable NFC tag. You can set the data on it using the Raspberry PI, and read data off it using a phone, or vice versa. It is a bit fiddly to use, but does the job.

```
import grovenfctag
print
(grovenfctag.readNFCDData(0,16)) #
```

```
read 16 bytes from address zero
grovenfctag.writeNFCData(0,[1,2,3,4,5])#write 1,2,3,4,5 to address zero
```

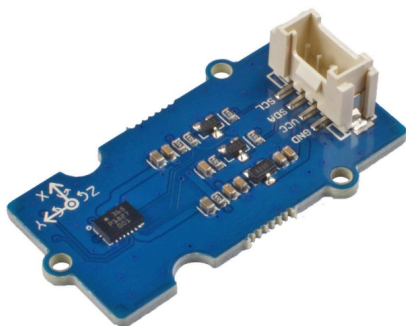
### 6 axis accel/magnetometer:



Accelerometer and magnetometer (compass) module, allows you to detect the angle something is at, or to detect bumps, vibrations etc.

```
import grove6axis
grove6axis.init6Axis() # start it up
print (grove6axis.getOrientation()) # returns orientation as yaw,pitch,roll
print (grove6axis.getAccel()) # get acceleration values
print (grove6axis.getMag()) # get magnetometer values
```

### 6 axis accel/gyro:



Similar to the above sensor but has a gyro instead of a compass, thus giving you changes in angle / angular momentum values

*NOTE the difference in appearance to other accelerometer sensor!*

```
import grovegyro
grovegyro.init6Axis() # start it up
print (grovegyro.getAccel()) # tuple w/ gravitational pull + x/y/z momentum
print (grovegyro.getGyro()) # tuple with angular momentum
```

## Grove NFC Reader

This lets you read RFID tags (like university access cards) and will tell you the unique identifier of the tags. Connect it to an i2c port on the grovepi.

You can use this identifier to distinguish between different tags, for example to make a system that can only be used by certain people scanning in with their university cards.

Use it like this:

```
import grovenfcreader
# wait for up to 5 seconds and display the ID
# of an NFC tag that is
# put in front of the reader
print (grovenfcreader.waitForTag(5))
```

It also in theory has support for talking to NFC devices like phones, transmitting files & URLs to them. If you want to try that kind of advanced use, we can help you a bit, but you will need to be a pretty competent programmer.

## Appendix B: Connecting to the PI and running a Python Script

First, write your python script and save it somewhere you can locate later on.

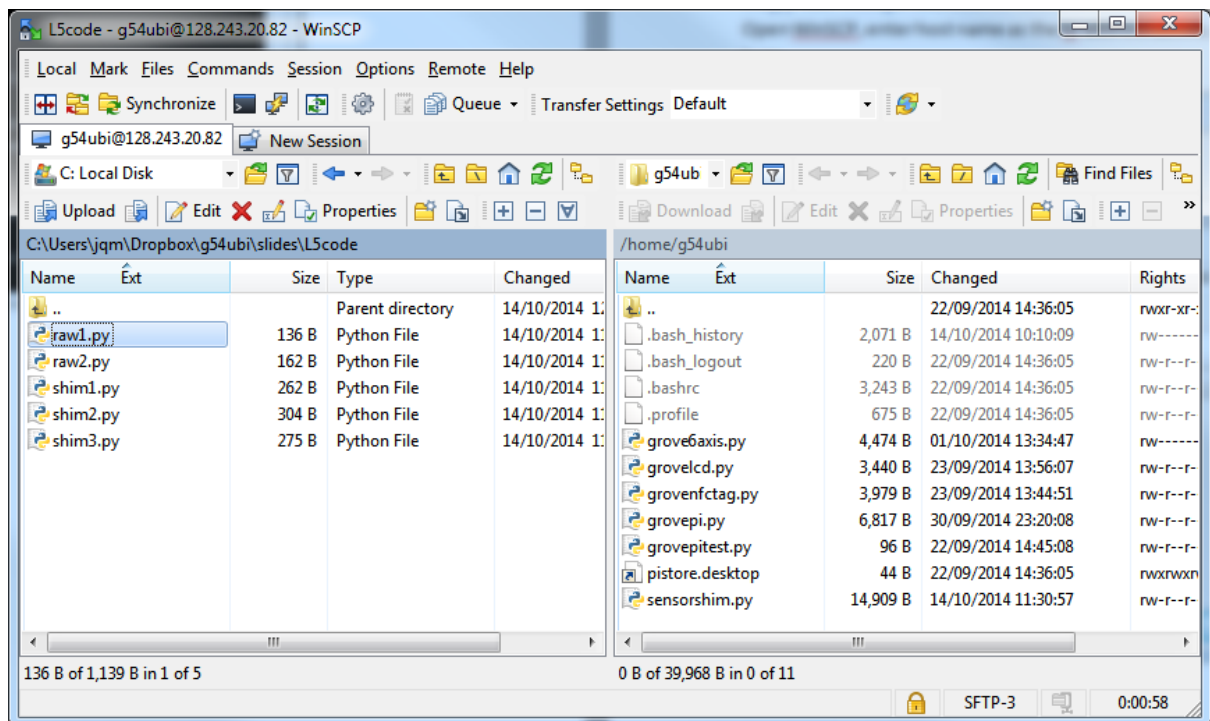
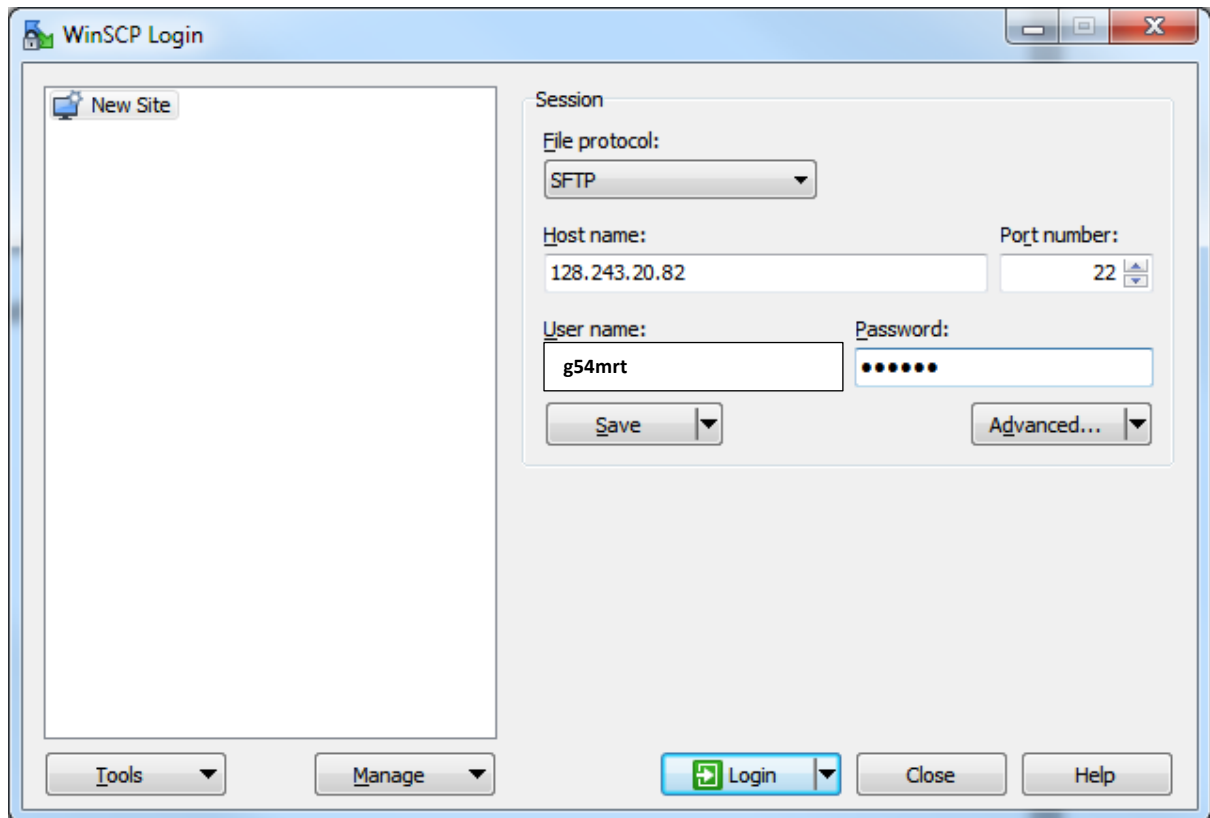
Because the PIs are shared, you might want to include your name or username in the python script filename, e.g. jm2-test.py

Start up the PI with a display connected (to grovepi port I2C-1 or I2C-2 or I2C-3). It should show the ip address of the PI and e: (wired connection) or w: (wifi connection) on the screen.



### Copy the script across to the PI

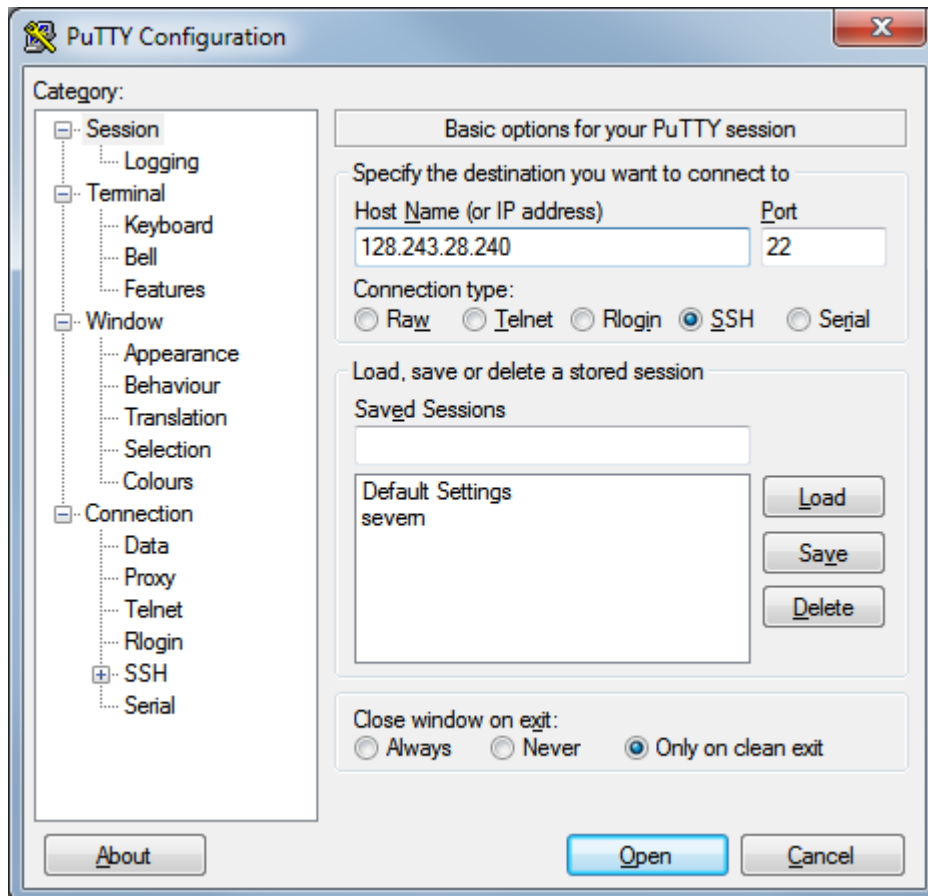
Open WinSCP, enter host name as the ip address above, username: **g54mrt**, password: **g54mrt**, click login.



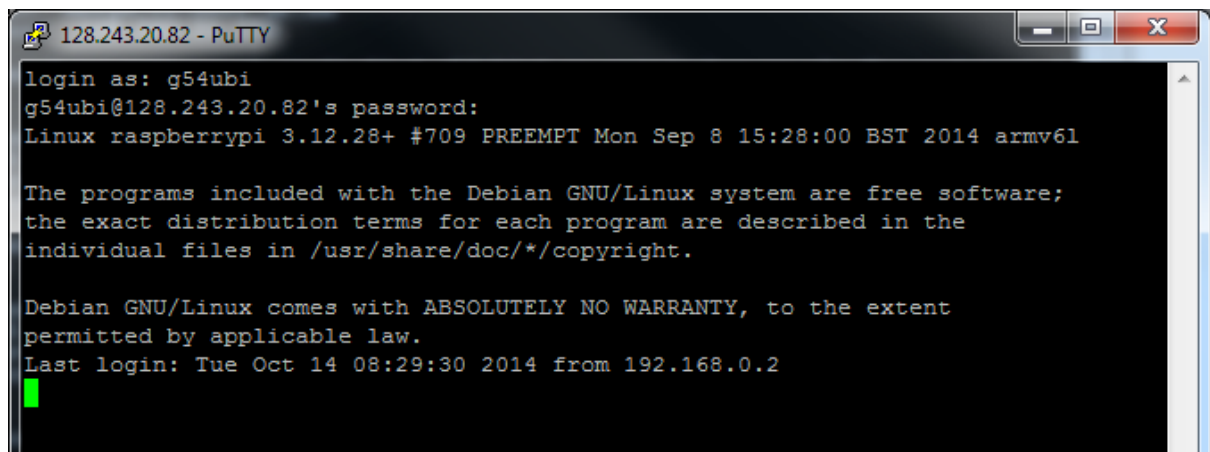
In the left hand pane, find your python script and copy it across to "home/g54mrt" on the PI

## Run the script on the PI

Open PuTTY to connect to the PI at this address.

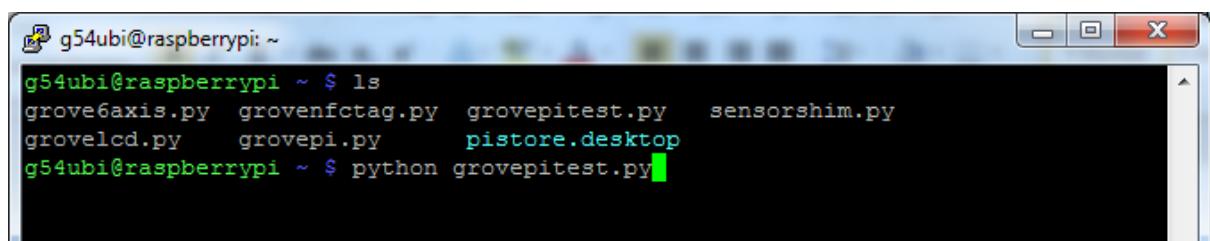


Login with username: **g54mrt** password: **g54mrt**

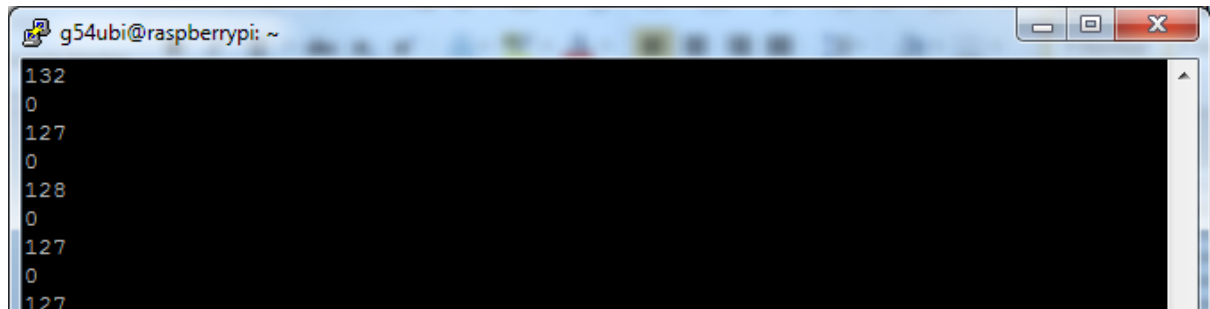


At the **g54mrt@raspberrypi\$** prompt, type

**python script.py**



Hit enter, and the output from your script appears in the terminal window. If you need to kill the script, hit **CTRL+C**.

A terminal window titled 'g54ubi@raspberrypi: ~' with a black background and white text. The output of a script is displayed as a vertical list of numbers: 132, 0, 127, 0, 128, 0, 127, 0, 127. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

To save the output to a file, use redirection like this:

```
[pi]$ python script.py > myoutputfile.txt
```

This will send the output from your script to a file called myoutputfile.txt. You won't see the output. Press Ctrl+C if you need to end the script.

You can copy your output file back across using WinSCP.

## Running scripts on the Pi while logged out / not connected

If you are wanting to run a Python script on the Pis but need them to stay running while disconnected from the network and any ssh session, you can use a program called nohup.

Note that when using nohup, you should flush after printing anything so that the file output gets written after each print statement (otherwise killing the job can result in nothing being captured):

```
import sys
while 1:
    print ("my output")
    sys.stdout.flush()
```

Here is an example:

First ssh to the Pi from a machine:

```
[my-laptop]$ ssh g54mrt@10.154.XX.XX
```

Then run your code via the nohup command on the pi

```
[pi]$ nohup python my-script.py > output.txt &
```

Your script is now running, and outputting to a file called output.txt

You should be good to disconnect.

When you come back, you need to kill the running script. First list the jobs running, then use kill to terminate the program:

```
[pi]$ jobs
```

[1]+ **Running**

```
nohup python my-script.py > output.txt &
```

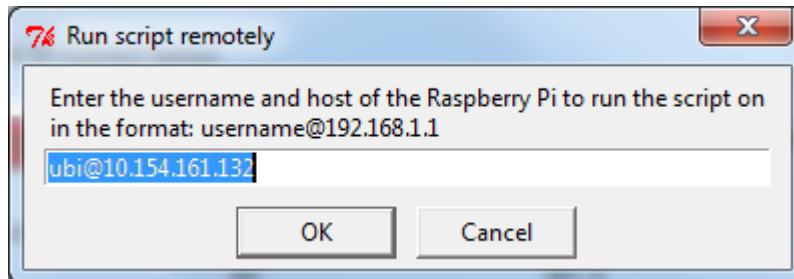
[pi]\$ **kill %1**

The output should be in output.txt

## Running scripts Quickly Using Grovepi Emulator

The grovepi emulator will automatically upload scripts to a pi and run them.

1. Launch the grovepi emulator
2. Click 'load...' under python script.
3. Click 'set remote address', and enter **g54mrt@address**, where address is the IP address shown on the PI LCD screen:



4. Click 'run on real PI via ssh'
5. Now every time you click the python script play button, it will upload the latest version of your script to the PI, then start it running.