**Summary**

**Introduction**

**Linux and Raspbian and Basic functions**

**Your Bible**

**Setting up the Raspberry**

**Connecting to the Wifi**

**Installing Apache, MySQL, PHPmyAdmin**

**Providing a remote acess to the Raspberry data**

**Setting a static IP address**

**Setting a remote connexion with SSH and Putty**

**Sending mail with IP address : ssmtp**

**http://www.havetheknowhow.com/Configure-the-server/Install-ssmtp.html**

**Use of sensors/camera/devices**

**SHT15**

**Wind Vane/Anemometer**

**Camera for timelapse**

**Real time clock : Preventing Time Drifting**

**Automatically recover the data from the Raspberry**

**Storing Database in a drive**

**First method : Setting the USB drive as the MySQL database directory**

**Second method : Copying database into a csv file**

**The Code**

**Perspectives**

**Providing a remote access to the Raspberry**

**Summary**

**Introduction**

**Linux and Raspbian and Basic functions**

Raspberry is operated by Linux. Before you go further, you need to know the basics about this langage. To be honest, I didn’t, so my advice is : take some time to play with the terminal, and make sure you are familiar with the basics commands. I won’t list all of them, but I will rather point you the concept you should understand:

- directories : this is essential. Whenever you want to access something, or execute a script, or save something somewhere, you need to know precisely where you are in the system. Especially, from the Terminal, you generally operate in the directory /home/pi. So If you want to execute a script somewhere else, you need to provide the path to the other directory. That is very important. It is also very useful to understand the directory and how the folders are organised. Sometimes, you will spend days wondering why a script doesn’t work whereas the only reason is that you are not executing it in the right directory.

- shebangs : this concept is linked with directories. The shebangs are in the beginning of each script, and enable you to call your script wherever they are provided you give the right path.

- permissions: some directories won’t be accessible for everyone, but only for the root user. You need to ask yourself the question of the permission each time you create a script. Sometime you won’t manage to run a script. Try again with root permission.

- scripts : most of my scripts are bash script (.sh) or python script (.py).

Classic command :

*sudo* gives you the root permission

*nano* open the editor. For example, if I want to create the script test.sh in the Desktop, I will run ‘nano /home/pi/Desktop/test.sh’

*shutdown –h now*

*cd* changes the directory

*reboot*

*python* this is when you want to run a python script. For example, if I want to run the script test.py written in the Desktop, I will run ‘python /home/pi/Desktop/test.py’.

IMPORTANT : most of my python codes are written in Python 3, so instead of ‘python’ you need to run the command ‘python3’

*apt-get update* This is a very useful command. Most of the time, when the installation of a software won’t work, running this piece of code will get you out of trouble. For example, if I want to update Apache, I will run ‘apt-get update Apache’. You need to have an Internet access to run this command.

Anyway, I could go on like this forever, but it would be useless, because every information you need is on the Internet. I knew nothing about Linux before I begun, and I think I got the job done. So let’s get started.

**Your Bible**

I will refer to this document quite a lot, because it explains many things about the Raspberry. From now on, it will be your bible. Cheers.

**Setting up the Raspberry**

As I just said, you need to do a certain amount of steps when you are brought a new Raspberry : installing Raspbian, giving password, etc… I won’t list the first steps, you can follow the bible, it will guide you gently through the first steps of setting up the Raspberry.

**Connecting to the Wifi**

At some point, you need have an Internet access, especially to install many softwares (Apache, PHPmyAdmin, etc…) and to run updates when you are in troubles. I don’t even remember by which miracle I figured it out, but trust me, that’s how you connect to the wifi with the raspberry.

First run sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf and enter the following lines :

network={

ssid="eduroam"

scan\_ssid=1

key\_mgmt=WPA-EAP

eap=PEAP

identity="uqrbart1@uq.edu.au"

password="Australia57"

id\_str="home"

}

This will enable you to connect to the wifi of UQ. You’re welcome.

**Installing Apache, MySQL, PHPmyAdmin**

We need to store the data that the Weather Station is going to record. Eventually, those data will be stored in a USB drive, but first we will write them into a MySQL database. Basically, you will writes data inside the raspberry in the form of tables, and you can access those data through the Internet, provided you know the IP address of the Raspberry. I’ll talk again about that.

But first you need to install those software into the Raspberry. Once again, you can follow the Bible. You need to make sure you have an Internet connexion before going there, because you are going to download those softwares directly from the web

**Use of sensors/camera/devices**

**SHT15**

**Wind Vane/Anemometer**

**Camera for timelapse**

**Real time clock : Preventing Time Drifting**

**Automatically recover the data from the Raspberry**

**Copying database into a csv file**

**The Global Code**

**IMPORTANT :**

**- Each USB drive has a different name for the Raspberry when connected. IT means that if you decide to use another USB to store the data, you will have to change the directories in the python codes, or no data will be written on the new USB. There probably is a way to automatically name the directory in function of the USB connected.**

**- Erase the MySQL databse once you are sure you have every information backed up in the USB. Otherwise, next time you use the weather station, the CSVexport and CSVbackup scripts will store useless data in the USB.**

**Set up a static IP address**

Summary: The IP address of the Raspberry is dynamically assigned by the DHCP each time you boot up, and it is likely to change. To remotely access your raspberry, you need to make sure you have a static IP address.

**http://www.suntimebox.com/raspberry-pi-tutorial-course/week-3/day-5/**

Issues: the static address given to the raspberry might be assigned to another device while the Raspberry is out of the network. I still need to figure out how to prevent this from happening.

Moreover, this only works when connecting to the Ethernet network. I still haven’t figured out how to set a static IP address with the Wifi.

**Sensor SHT15**

Follow the instructions provided by this site to install the python package that you will run to make the sensor work.

<https://pypi.python.org/pypi/Pi-Sht1x/1.0.1>

The global file will be loaded in the directory /usr/local/lib/python3.4. In this files, all the function that will be called in the python codes are defined, that’s why you need to take care of this library. You can run the example code /usr/local/lib/python3.4/dist-packages/examples to get used to how the sensor works.

IMPORTANT : **this package is written in Python 3**, so every time you want to run your code, you need to type “*sudo python3*” instead of “*sudo python*”.

Moreover, the python3 does not work very well with MySQL, especially, you won’t be able to import MySQLdb. To actually write your data into MySQL database, you rather need to call the function pymysql by running the code “import pymysql”.

Then, instead of connecting to MySQLdb, you can run “conn = pymysql.connect(your\_code)”.

You’ll see that I use this connector in every one of my python script. If you want more information about this connector, you can go there:

<http://stackoverflow.com/questions/4960048/python-3-and-mysql>

After having downloaded the python files to the raspberry, you need to connect the SHT15 to the GPIO of the raspberry. Here are the PINs I use to connect the sensor:

GND – Pin 6

VCC – Pin 1

Data – Pin 12

SCK – Pin 16

The code I use to run the SHT15 is strongly extracted from the example code I talked about earlier. Here is the code :

**Anemometer**

**Connected to Ground and GPIO 5 (pin 6 and 29)**

**https://www.raspberrypi.org/learning/weather-station-guide/anemometer.md**

**Wind vane**

**Storing Database into a Drive**

**Copying database to the drive**

Instead of writing the mysql databse directly in the drive, it is far easier to copy regularly the database into the drive from the original databse directory. This way, you can even directly copy the data into a CSV file.

Here is the simple script I use :

**Back Up structure**

There is two types of data we need to store. First, all the data from the sensors: SHT15, wind vane, anemometer. These data will be stored in the MySQL database, so the challenge is to back up this database in the USB drive.

The second type of data is the pictures taken for the Timelapse.

These data will be written simultaneously, so you need to pay attention to the fact that at some point, several script will be run in background. Therefore there is a need to make sure that the directory is the same in all the script for the operation to work simultaneously.

**MySQL back up : crontab**

The data from the sensors will be written from the startup script, but the back up will be run thanks to crontab.

**Timelapse : Raspberry script, On StartuP**

Crontab only allow operations for a one minute minimum. As you want to take pictures every X seconds, crontab is not the right solution.

**Adding Real Time Clock**

The Raspberry has no internal clock. To set up the clock, you need to connect to a network that will give you automatically the UTC. Yet we might want to operate the weather station out of the network. Having a precise time even without Internet is crucial since we are going to take measures and timelapses that need to be precisely timed.

No worries, let’s just add a Real Time Clock module to the Raspberry.

First, you need to wire the RTC module, follow the easy steps here:

<https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/wiring-the-rtc>

Then, you need to configure the Raspberry to detect the RTC:

<https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/set-rtc-time>

NOTE : for a reason I still ignore, the UTC given by the network is 2 hours in advance (thanks IT guys). That is important because you give the time to your clock thanks to the network. To make sure the clock is giving you the right time (and not the false UQ network time), you need first to manually set up the Raspberry time with this command

*dpkg-reconfigure tzdata*

You will be proposed several UTC, check the Brisbane one. Once your Raspberry is manually set up to Brisbane UTC, you need to give this time to your clock once for all, by running

*hwclock –w*

Check if it’s working with the command

*hwclock –r*

From now on, the RTC will always give this time. Good job.

**Looping with Python**

<http://stackoverflow.com/questions/474528/what-is-the-best-way-to-repeatedly-execute-a-function-every-x-seconds-in-python>

Timers : To run things in loop, I choose to use python codes. I named these python codes “Timer1”, “Timer2”, etc… These codes will be run at startup thanks to the file /etc/rc.local.

These codes execute other python codes. Make sure you previously run the chmod +x command to change the permissions on the python codes. Also, the shebangs are very important when you call a python script inside another python script.