ISSUE 21 - MAR 2014

BIGGEST issue yet with 48 pages!

The MagPi

A Magazine for Raspberry Pi Users

Stronghold Of The Dwarven Lords

Interview with Eben Upton
Monitoring an Aquarium
Portable Raspberry Pi
PIR Motion Detection
Weather Station
Linux Toolshed
Internet Radio

Win a GEEK case, Pi NoIR camera, GPIO Cobbler, SD cards & more

ISSN 2051-9990

The MagPi

http://www.themagpi.com

Welcome to Issue 21 of The MagPi magazine, our biggest issue yet with an incredible 48 pages! Once again we are bursting at the seams with all that is new in the Raspberry Pi world.

To mark the Raspberry Pi's second birthday, we have a MagPi exclusive four page interview with Eben Upton reviewing the past two years and what the future holds for this credit card sized beauty.

Also in this issue, Jacob, a young Raspberry Pi enthusiast, describes how he made his Raspberry Pi portable, we have more from Project Curacao plus ModMyPi give us a demo on motion sensing. We feature a great article on using the command line to troubleshoot network issues, take an in-depth look at turning your Raspberry Pi into an internet radio device, show you how to use a Raspberry Pi with a weather station and also feature how to remotely manage your home saltwater aquarium.

If that wasn't enough, we conclude with a classic game by heading deep beneath the earth, to the realm of the Dwarven Heartland, in the hope of finding gold in the text adventure "Stronghold of the Dwarven Lords".

As always, we keep you up to date with some of the latest Raspberry Pi goodies available for purchase, upcoming events, great competition prizes from PCSL and two new book reviews.

The last two weeks has been an exciting time for us at MagPi HQ. You may have seen from the Raspberry Pi Foundation site that we launched a short Kickstarter campaign to create a binder to hold all the Volume 2 magazines (issues 9 to 19). This campaign far exceeded our target and we are pleased to announce that production of the binders is now under way. Don't worry if you missed the Kickstarter. The binder and all the Volume 2 magazines will soon be available from our retailers (see page 8).

To follow our progress, like us on Facebook at http://www.facebook.com/MagPiMagazine to keep up to date and give us more of your valuable feedback.

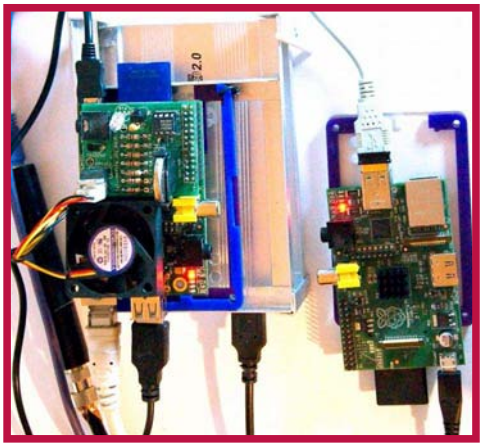
Enjoy this month's magazine!

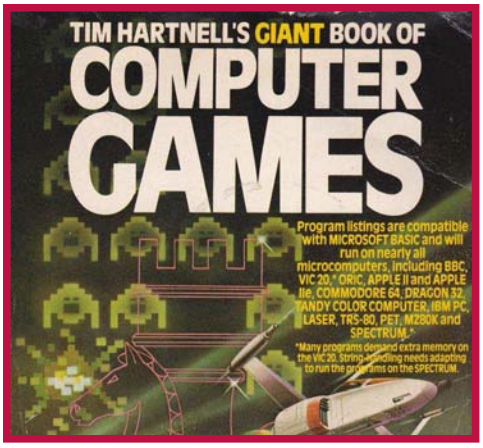
Chief Editor of The MagPi

The MagPi Team

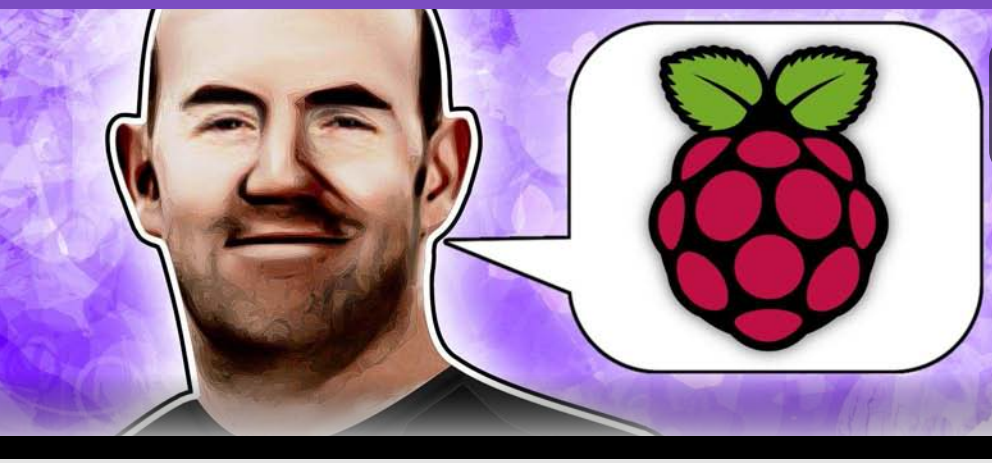
Ash Stone - Chief Editor / Administration
Ian McAlpine - Issue Editor / Layout / Testing / Proof Reading
Bryan Butler - Page Design / Graphics
W.H. Bell - Layout / Graphics / Administration
Matt Judge - Website / Administration
Aaron Shaw - Layout / Proof Reading
Colin Deady - Layout
Amy-Clare Martin - Layout
Age-Jan (John) Stap - Layout / Proof Reading
Tim Cox - Testing / Proof Reading
Sai Yamanoor - Testing
Claire Price - Proof Reading
Matthew Watson - Proof Reading
David Whale - Proof Reading

Contents





Original cover artwork by Bryan Butler



EBEN UPTON
The MagPi exclusive interview

Aaron Shaw

MagPi Writer

Eben Upton discusses the first two years of Raspberry Pi and what to expect for the future

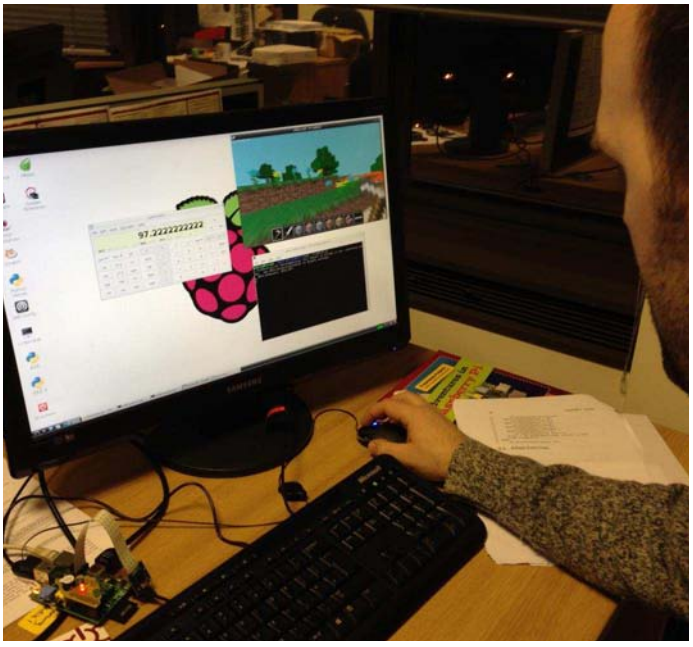
I got the chance to speak to Eben Upton via Skype in February about the last two years of Raspberry Pi and what he has in store for the future. Eben was actually on a well deserved holiday at the time, but very kindly took an hour or so out for this interview. This was only the second time I had spoken directly with Eben (the first was very brief though).

From this interview it is very easy to see why the Raspberry Pi has been so successful, with Eben at the helm. He is possibly one of the most enthusiastic people I have ever spoken to, even after what (by his own admission) have been the two busiest years of his life.

I also would like to take this opportunity to apologise to Eben for using up an hour of what little holiday time he has!

[MagPi] Since the Raspberry Pi was officially released for sale on the 29th February 2012, it has been a pretty eventful and exciting couple of years for the Raspberry Pi Foundation, the community and everyone involved with it. How would you sum up the last couple of years, and what have been your favourite moments?

[Eben] Busy I suppose. It has been good. It has been very different from what we imagined – obviously the scale has been surprising. The way that the hobbyist community and the education community have sort of reinforced each other has been a big positive surprise for us. You have got hobbyists who are interested in it because we have this not for profit educational angle and are doing cool things with the Pi, and you have the educational community who are then benefiting from the fact that the platform is now very stable and has all these lovely features, but also that there are lots of inspirational projects out there.



Eben programming Minecraft at Pi Towers

Also, because it got big and generated enough money to pay salaries, I have been able to hire some of the best engineers that I know into this, as well as an education team.

[MagPi] 2014 looks to be equally exciting, if not more so. There is the introduction of the new Computing course into the UK curriculum, from September 2014, and many other initiatives which all, at least in part, have something to do with your efforts at the Raspberry Pi Foundation. What are you looking forward to in the next year?

[Eben] One of the really nice things about the Raspberry Pi is that we can do that mix of some stuff in house and some stuff with partners – both with the technical and education sides. So I am looking forward to seeing what the education team come up with in way of resources, but also what we can achieve with partners.

One of the challenges with the 2014 curriculum is that the scale of the problem is enormous. You have 25,000 schools and the UK government hasn't gone much beyond changing the curriculum and saying "Get to it". This is something which we discovered quite early on with the production of the Raspberry Pi – making 1,000 boards is kind of hard (but tractable) but making what we make at the moment (around 5,000 boards a day) is a whole different challenge.

On some level, I sort of feel like this is kind of where we were in the 1980s. Although we look back at the 1980s as being this "golden era" of computing – but in terms of the experience of most children, if they learned to program it was because they got into it at home with their Spectrum or BBC Micro.

This kind of golden age wasn't really that golden and I think that what the government, industry and hobbyists like me are hoping can be built is something that is much better than the 1980s in terms of its breadth – and with 25,000 schools that is incredibly challenging.

[MagPi] There have been an increasing number of highly successful Kickstarter projects and start-up ventures based entirely, or partly, on the Raspberry Pi. The Telegraph recently named the Raspberry Pi accessory businesses as one of the best start up ideas for 2014. Do you also see this as a large area for growth over the next year?

[Eben] Turns out yes. Let's pretend that was the idea all along. Looking at two out of quite a large community – Jake from ModMyPi and the Pimoroni guys. They are both quite interesting as they started as casing companies for the Pi. That was one where we left all of that value out there on the table because we just didn't believe we would sell enough units to justify doing any injection moulding. We never made cases for the Pi and we never made accessories for the Pi.

Matt Timmons-Brown (aka The Raspberry Pi Guy) on work experience at Pi Towers

We made the camera board, because that requires fairly deep integration with the GPU and is not something somebody else can do – it was so much easier for us to do it. Gordon and James and I were members of the design team for the chip, so when something goes wrong we can go and look at the source code – which is an enormous advantage. A sometimes under-recognised part of the Raspberry Pi story is the amount of volunteer effort from Broadcom.

[MagPi] On top of that, there are an increasing number of children and young adults getting involved with start-up ventures. I know you actively encourage this activity at the Foundation – I guess you could say it is one of your main aims. A few startups have come about from people still in, or just out of, school. Has the Foundation thought about ways in which it can encourage young entrepreneurs who are still at school? What advice would you give to a young person looking to follow in the footsteps of inspirational young people like Ryan Walmsley, Amy Mather, Sam Nazarko, the AirPi guys and The Raspberry Pi Guy (to name just a few)?

[Eben] Entrepreneurship is a really important thing. That is the background I come from, and as a job I was interested in computing because I thought it could make me money – I don't think there is anything wrong with that. In the UK there used to be a television programme called Bergerac, set in Jersey. There was a supporting character called Charlie Hungerford who was a wheeler-dealer kind of character – and he was my childhood hero. It is this really awful thing, that I didn't idolise engineers or physicists – but a slightly dodgy, second hand car dealer.

Entrepreneurship around the Raspberry Pi is a very important thing. It is great to see guys like Ryan with the motor controller board – and of course now it is on Adafruit and he has a nice little resale business going on there. For a young guy like that to be doing a combination of technical and business stuff with the Pi is great.

The Raspberry Pi Education team at the BETT show

[MagPi] You have recently moved to a bigger office, and have taken on three new members of staff for the education team (Carrie Anne Philbin, Dave Honess and Ben Nuttall joining Clive Beale). What exciting plans have you got for them over the next year?

[Eben] The great thing about having these guys is that they are not just tech guys – they are first rate, professional teachers. You can talk with a level of authority that you just can't talk with if you didn't have teachers, and really first rate teachers, on board. It is much better than just having a bunch of random tech guys hypothesising about what it might be like to teach in a school.

[MagPi] Are there any plans for new hardware in the next year – either new add on boards or any upgraded or brand new hardware for the Raspberry Pi itself? Maybe a display for the DSI connector, or something similar?

[Eben] We are actively working on the display board and hopefully we will have some announcements on that fairly soon. We have a lovely demo that we are going to show at Electronics World in Germany, which is where we launched the Pi two years ago. The nice thing about the display connector is that you can use the HDMI output and the DSI output at the same time – so you can use two displays.

What has been interesting is that Pi NoIR has sold extremely well. Pi NoIR was a bit of a cheeky product really, in that there was no engineering effort involved. I quite liked it as it was a way of getting a new product out without using up any of my engineering resource. It just required people to make black PCBs and getting the sensor manufacturer to make one without the IR filter. It started with people out there cracking

the camera module open with a scalpel and pulling out the IR filter, and we have ended up with a product that sells thousands of units a month.

[MagPi] The actual Raspberry Pi boards themselves, are they all being made in the UK now or are RS Components still making some of them in China?

[Eben] RS are still making a few in China from time to time. A great problem to have – but Christmas was really busy and we sold so many Raspberry Pis. One of the nice things about our Chinese manufacturer, Egoman, is that they are very responsive. Because they are so close to the components (they are in Shenzhen) their lead times are much more reasonable and it is useful to have them for those surge occasions.

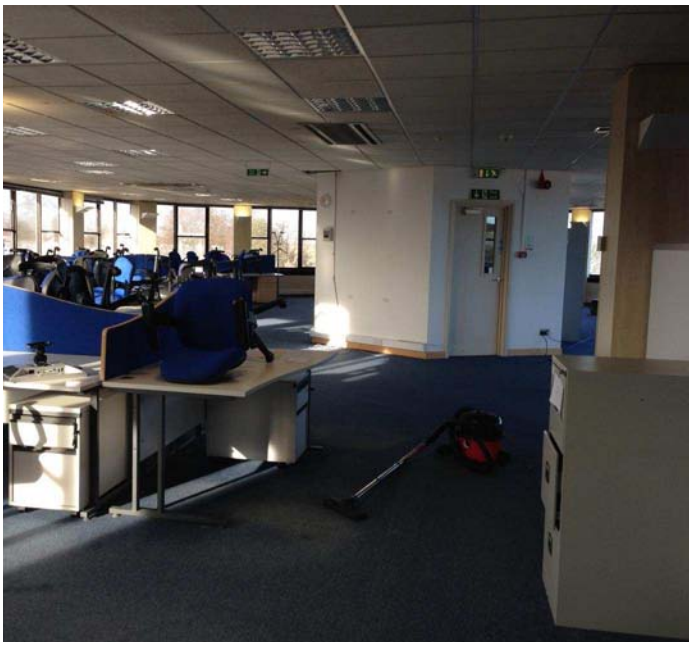
It is great that Sony actually buys in speculative stock at the moment, so that they have some additional buffer of components. However it gets to the point sometimes, like at Christmas, where all of that upside has been exhausted and you are back to waiting for 12 week lead times.

We are comfortable with the quality – they are indistinguishable from a quality perspective.

[MagPi] On a final note, there have been some pretty inspirational, awesome or just plain crazy ideas that the Raspberry Pi has made possible. Is there anything you wish someone would make with a Pi but you just haven't seen yet?

[Eben] There is one, which I think we will get eventually. Some of the more technically complicated hobbyist products, like the high altitude ballooning that Dave Akerman has been doing, I think it would be really nice if people took a look at packaging those so that they would be useful for educators. A balancing robot or a high altitude balloon project – you can create value there by helping teachers to create an assured project. That is something that the OCR collaboration generates – projects that are of known length and are known to work. If someone could launder high altitude ballooning into a really plush kit and you have everything you need, along with extension excercises, that would be fantastic.

The good thing about the whole Pi ecosystem is that there are a lot of people who could do that. I would be very surprised not to see something like that start to crawl out of the woodwork over the coming year, especially as we have the curriculum change. It is the sort of thing that would be good to have ready for the first or second year of the curriculum.



The new, larger Raspberry Pi offices (aka Pi Towers) complete with Henry the Hoover

It looks like we have an exciting year ahead of us in the Raspberry Pi world - both from an educational and technical standpoint. Here's to year 3!

On a bit of a side note, Eben also informed us that he has nearly finished writing another book titled "Learning Computer Architecture with Raspberry Pi". From the short description on the Wiley website it looks to be a fantastic read.

For those who want to get it as soon as it is released it is available to pre-order now direct from Wiley at http://goo.gl/hpd5Jf.

PRINT EDITION AVAILABLE WORLDWIDE

The MagPi will always be available for FREE from http://www.themagpi.com, The MagPi iOS and Android apps and also from the Pi Store. However, because so many readers have asked us to produce printed copies of the magazine, starting with Volume 2 (Issue 9) printed copies are now regularly available for purchase at the following Raspberry Pi retailers...

Americas

adafruit INDUSTRIES

EMEA

MODMYPI

Pi Supply

The Pi Hut

AsiaPac

AusPi Technologies Australia

Raspberry Pi® Starter Kits

Everything but the screen

http://shop.pimoroni.com

Starter Kit £75
Deluxe Starter Kit £120

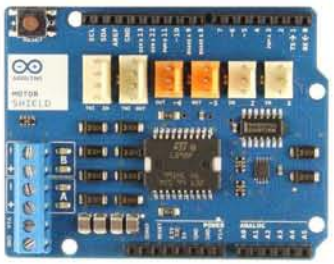
Pimoroni Gift Cards

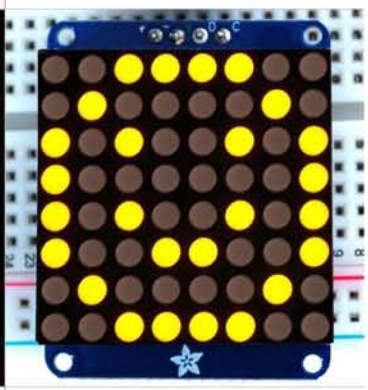
Choosing shiny things is hard!

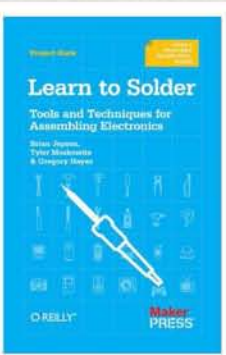
10 £25 £50

http://shop.pimoroni.com

PIMORONI







Wearables! Arduinos! Components! Dead Trees!

We now stock a range of shiny things you can make cool stuff with!

http://shop.pimoroni.com

PIMORONI

How I manage my home weather station

Neil Turner

Guest Writer

SKILL LEVEL : INTERMEDIATE

This article describes how I use my Raspberry Pi to manage my home weather station. This includes the background to the project and some examples of the information collected, as well as a few pitfalls. The complete software (with the working title of "Squall") is available from http://www.spire-three.com. It is open-source and anyone is free to build upon it.

Recording and predicting the weather

Accurate weather forecasts can save lives and money and the UK Met Office uses satellites, weather balloons, ground stations and supercomputers to produce forecasts that are accurate for several days ahead. Clearly amateurs cannot compete with these resources, but it is surprising what can be achieved with an off-the-shelf weather station and a humble Raspberry Pi. The Met Office's network of ground-based stations include Synoptic weather stations, which record the weather every hour, and Supplementary stations, which take measurements at 9 am each day.

For the Supplementary stations, the aim is to produce a continuous record of conditions at that site over a long period of time (30+ years as a minimum, but many stations have continuous records stretching back much farther). This information helps us to understand long-term changes in the climate.

In contrast, measurements taken at Synoptic stations are more frequent (hourly) and include additional parameters which provides vital information for current weather forecasts.

Modern home weather stations are largely comparable with Supplementary stations. Therefore we can use the home weather station to build-up an interesting record of our local weather, while learning a lot in the process. We may also make educated guesses about what will happen to our weather over the next twelve to twenty-four hours.

Our measurements are made electronically and thus can be taken automatically (we don't need to get up in the middle of the night and we can go on holiday knowing exactly what the weather is like at home!). We can also take readings very frequently, maybe even every five-minutes, allowing us to track changes that happen relatively quickly.

Finally, data can be transferred to a Raspberry Pi for analysis and longer-term storage.

Objectives

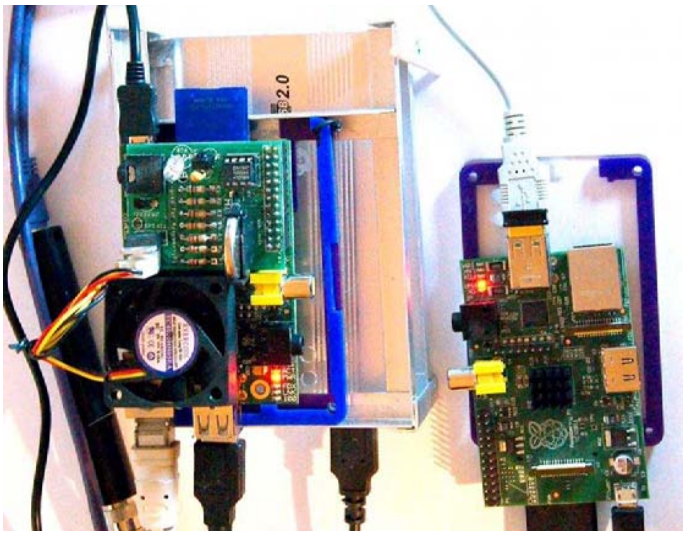
My weather station was purchased in the UK from Maplin Electronics http://www.maplin.co.uk (Code: N96FY), but the same equipment is also available under other names. It is identifiable, using the `lsusb` command, as:

```
Bus 001 Device 004: ID 1941:8021 Dream Link
WH1080 Weather Station / USB Missile Launcher!
```

My aim was to capture and store weather data so that I could look back over weather events to see what had actually happened. This would involve a lot of data suggesting that a database would be needed. I wanted to see how well MySQL, a commercial strength database, would perform on the Raspberry Pi and if it would be able to make the latest information available on my website, in near real-time (i.e. every 10-20 minutes).

Collecting raw weather data

Readings from instruments outside the house are passed to a base station inside the house using a radio link. The base station stores the data and makes it available to the Raspberry Pi via USB. When the base station's buffer is full it over-writes the oldest record. Software is supplied with the weather station but it is closed source and does not run on Linux. Fortunately, other projects have already produced interfaces for Linux and, being open-source, we can use their routines to move the weather data onto our Raspberry Pi.

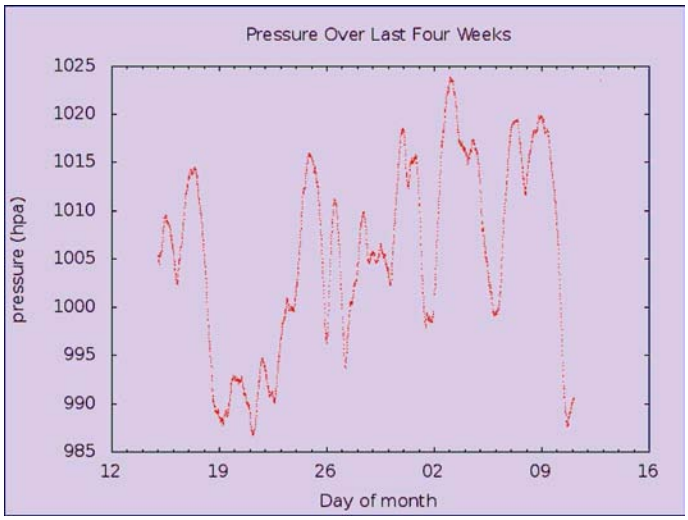


The software that I used is called pywws and was developed in Python by Jim Easterbrook. It is an excellent system and provides many facilities. However, for the purposes of Squall, I only needed the LogData function to capture the weather station data and present it as a text file on the Raspberry Pi. The pywws software is well documented with an active forum but I have also outlined my installation.

Experience

The weather station and the Raspberry Pi have worked well over the last year. The chart on the right shows data generated by my station.

The pressure measurements agree closely with those reported by the National Physical Laboratory and show a series of weather fronts moving across the UK in January/February 2013.



I live in a coastal location to the south of the country and temperatures tend to be slightly higher than the UK averages. However, the maximum and minimum temperature readings are in general agreement with monthly summaries produced by the Met Office. The above chart also shows that the weather station signal is noisy, producing an occasional spurious, unrealistically high or low reading.

Simple measures such as keeping cables short and attaching extra ferrite rings would reduce the impact of some types of interference but don't seem to have solved this problem entirely. A pragmatic approach would be to delete a record that was impossibly higher or lower than readings taken a few minutes before or after.

Installation

The base station and weather station are both powered by batteries. I would advise mounting the anemometer (wind speed) and weather vane well above ground level and in the clear. However the temperature and pressure monitors should be mounted lower, at around two metres above ground level. This would be in keeping with Met Office practice but also allows the battery to be changed more easily!

Were the objectives met?

Yes, my weather data was in good agreement with the professional results reported locally and nationally. The system worked well and I was able to transfer results automatically to my website. The MySQL database performed very well on the Raspberry Pi, giving me powerful tools to manage a small but not trivial amount of data.

Taking readings every five minutes will generate 12x24x365=105,120 records per year. This is small for MySQL but impractical to search manually. MySQL performed well on the Raspberry Pi and could be a useful option for other projects. It could also be useful for teaching SQL.

Appendix 1

Bash scripts

Squall consists of a number of bash scripts. Most of the scripts are relatively simple and cover one task by making a single call to MySQL. Most of the real work is done within the MySQL query. If you are unfamiliar with scripts it can seem quite daunting. You need to think about it as broken down into many parts. There are guides on the web and tutorials on YouTube.

In my scripts I need to work with files but I won't know the precise name of the file until I run the program as the filename will include a date. Therefore I need to work out what the correct name of the file should be and use that name. To do this we need to handle strings. Here is the `load_weather_data` script:

```
PREFIX="~/squall/data/raw/"
SUFFIX=".txt"
FILE=$(date +"%Y/%Y-%m/%Y-%m-%d")
FILENAME=$PREFIX$FILE$SUFFIX
```

First we set up some variables. `PREFIX` is the path to where the data is stored and `SUFFIX` indicates the type of data. We let the program know that a variable should be treated as a string by prefixing it with `$`.

We now have the beginning and the end of our filename. Next we generate the filename and path by calling the `date` function to get the current date. We store our files in folders using the year, month and date. An example value for `FILE` would be `"2014/2014-03/2014-03-01"`.

The final `FILENAME` is the `PREFIX`, `FILE` and `SUFFIX` added together. An example would be `"~/squall/data/raw/2014/2014-03/2014-03-01.txt"`

Appendix 2

Database and SQL

The database design is very simple, with one main table holding dates, times and values. However, values such as maximum and minimum temperature have to be calculated and I created an additional table to hold these calculated values. This is an efficient way to retrieve the information when needed but it is not best practice. To explain why this might cause problems, imagine that we find another record with a new maximum or minimum temperature. As soon as we add that new record, the previously calculated values are wrong and our database is inconsistent.

SQL commands

The majority of SQL operations create a table, add records to a table or retrieve records. It is not particularly difficult to write the SQL and there are plenty of text books, guides and tutorials available on the internet. The following example embeds SQL code from within a script:

```
mysql -hlocalhost -uroot -praspberry
  --local-infile --verbose squall <
  ~/squall/scripts/sql/load_weather_
  data.sql
```

Let's breakdown this command.

mysql Call the MySQL database manager
-hlocalhost Confirms we are working on our local machine and not remotely
-uroot Database root user, not root for the Pi
-praspberry Password for the database root user
--local-infile Warns MySQL to expect some data within a file (this is not the name of the datafile)
--verbose Display detailed messages
squall The name of the database we want to use
filename.sql A file containing SQL commands which is passed to the `mysql` command

This command invokes MySQL but the SQL is actually within the file `load_weather_data.sql`. The SQL is:

```
LOAD DATA LOCAL INFILE
  '~/squall/data/indata.dat'
  INTO TABLE rawdata
  FIELDS TERMINATED BY ','
  LINES TERMINATED BY '\n';
```

Appendix 3

Pywws

I am using the Logdata function of Pywws to transfer data from the base station. The instructions on the Pywws website are excellent. The only problem area seems to be accessing the weather station via USB. The dependencies, permissions and udev rules necessary to overcome any problems are well described in the Pywws documentation. Pywws itself is quite small so I simply installed the whole package and just used the script that I needed.

Create a work directory for Squall, then install Python and the USB libraries:

```
mkdir squall
sudo apt-get install python
sudo apt-get install python-usb
sudo apt-get install libusb-1.0-0
```

Download Pywws (`pywws-13.10-r1085.tar.gz`) from https://pypi.python.org/pypi/pywws/13.10-r1085.

Move the Pywws archive to the `squall` directory, expand it, then remove the archive file:

```
mv pywws-13.10-r1085.tar.gz ~/squall
cd ~/squall
tar zxvf pywws-13.10-r1085.tar.gz
rm pywws-13.10-r1085.tar.gz
```

Appendix 4

Connect the weather db to LibreOffice

Structured Query Language (SQL) is great for drilling down into your data but can be complicated and cannot directly produce charts or reports. The solution for many people is to connect their database to desktop tools to produce attractive graphs and reports. For example, we can connect a MySQL database on the Raspberry Pi to LibreOffice.

MySQL must listen for incoming connections (by default on port 3306) so we need to make some modifications to the MySQL configuration file. Enter:

```
sudo nano /etc/mysql/my.cnf
```

Scroll down to the `[mysqld]` section and comment out the line `skip-networking`.

The line `bind-address` will probably be set to "`127.0.0.1`". Change this to "`0.0.0.0`".

Press <CTRL>+X to close and save, then exit. Restart MySQL by entering:

```
sudo service mysql restart
```

If that doesn't work, please try the following instead:

```
sudo /etc/init.d/mysql restart
```

In order to handle these new incoming connections MySQL needs a user with the correct privileges. Open MySQL by entering:

```
mysql -hlocalhost -uroot -praspberry
```

Then create a new user called `rempi` by entering:

```
CREATE USER 'rempi'@'localhost'
  IDENTIFIED BY 'raspberry';
GRANT ALL PRIVILEGES ON squall.* TO
  rempi@192.168.1.0/24';
FLUSH PRIVILEGES;
QUIT;
```

Start LibreOffice and create a new database. Within the database wizard choose to connect to an existing database and select MySQL from the drop down list. We can connect to the MySQL database directly.

The database name is `squall` and the server field requires the IP address of the Raspberry Pi. Use the bash `ifconfig` command to find this. Port 3306 is the default and should be left unchanged.

We need to provide the name of the MySQL user created above and select to use a password.

The wizard saves the configuration and allows us to select a meaningful name for future use. The database connection can then be used with LibreOffice programs such as LibreOffice Base and LibreOffice Calc.

Grow coral, not algae, with a Raspberry Pi

Michael van den Heever
Guest Writer

SKILL LEVEL : INTERMEDIATE

What is Fish-Pi?

Fish-Pi is a soup of technologies which enable me to manage my salt water aquarium remotely via the internet. If you've ever owned a salt water aquarium you will undoubtedly know that it is a very high maintenance hobby. Any form of automation is most welcome. One of the critical aspects of salt water aquariums, and keeping corals, is light management. Too little light or too much light and you are going to grow more algae than you are corals.

There are also a few stages of lighting I control throughout the day as well as types of lighting. The amount of time certain lights are on can affect many factors within the fish tank. I needed intelligent control and not just a simple on and off timer switch. In short, I needed a Raspberry Pi.

Project goals

- Have remote control ability from anywhere in the world.
- Knowledge of time, to power up the correct light configuration at the right time of day.
- Send email notification of any power cut.
- Thermal monitoring of both the water temperature and the thermal health of the Raspberry Pi.
- Automated and manual light control for calibration.
- Visual status indication that everything is ok.
- Visual confirmation via a webcam.
- Present all information via a "Pi-powered" website.
- Finally I wanted to "learn" during this process.

Fish-Pi's features and functions

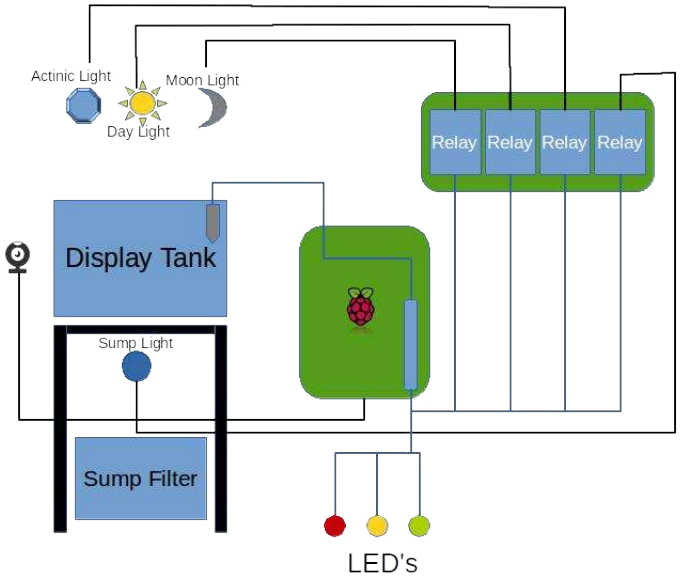
Currently, Fish-Pi has the following features. I have four types of lighting linked to my aquarium plus one temperature probe. I also use a Raspberry Pi camera to visually confirm the health and state of my tank. Various Python scripts control the state of the lighting throughout the day via a GPIO pin state check (either on or off) as well as constant water temperature metering. Finally, the temperature of the Raspberry Pi CPU is monitored.

As no display is attached to the Raspberry Pi, I have created a visual notification system using three LEDs.

RED – indicates the automatic schedule is off.
YELLOW – indicates system activity. For example, while the water temperature is being read (once every minute) the yellow LED strobes.
GREEN – indicates the automatic schedule is on.

I would like to mention at this point not to abandon simple LED communication. Developing your own visual language through strobes and flashes can be quite fun and creative. It's also a great way to visually affirm that the Raspberry Pi has not frozen up.

Visual indication



In the above diagram, at the top you can see the three light sources (comprising a total of 63W of blistering LEDs). Actinic is the blue light spectrum (30W), daylight is the white spectrum (30W) and moonlight (3W) needs no explanation. Under the aquarium I have a second tank, called a sump, which contains a sump filter that handles my water filtration. Here you will see the sump light. All lights are wired to digital relays which in turn are wired to the GPIO. Housed in the display tank is a digital thermometer which is also connected to the GPIO. A Raspberry Pi camera lets me view the tank remotely. Last, but not least, are the three status indictor LEDs.

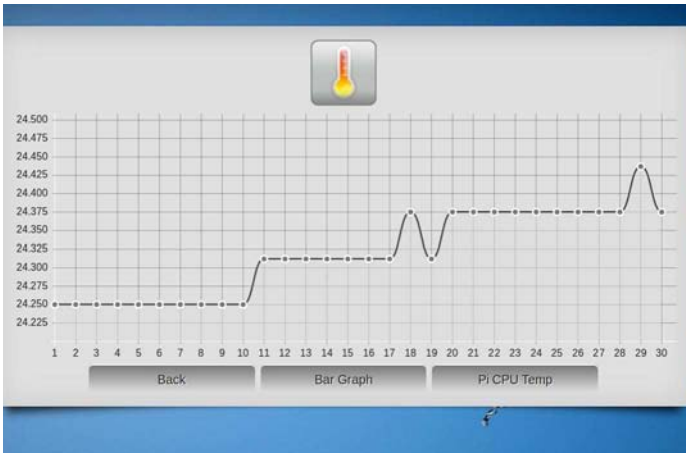
I will not be going into depth on the construction and wiring of the high power LEDs that run over my tank. Instead I will focus more on the Raspberry Pi and the various scripts. However, if anyone is interested in building high-powered LED systems using modules that can be ordered online, visit my website at http://piworx.subzerobc.com for more details.

Web interface

I put considerable effort into the graphics side of FISH-Pi, using royalty free graphics and some of my own Photoshop skills. But let me give you some advice... avoid the pretty stuff and get your framework working first. For example, get your web page buttons working correctly first, then brush them up when you are done.

In the control panel above I have buttons that control the various lights in a very simple format of ON or OFF. There is an option to cycle the lights, which basically goes into auto mode (I will elaborate on this later). At the bottom I have some orbs, which show the live status of the lights by indicating ON or OFF. This is simply achieved by checking the state of the GPIO pins and writing the value out to a text file, which is then read by the web interface.

At the top I display the current water temperature. Below that there are two icons which represent temperature data and camera access. Let's take a look at a graph I was able to create with the temperature data.



The temperature is read at one minute intervals. I take a span of 30 “reads” from the temperature database, which gives me an overview of the last half an hour. The same can be presented in a bar graph format. I have also added the Raspberry Pi CPU temperature in this section.

Hardware components

1. Raspberry Pi Model B
2. 1 x 4K7 resistor and 3 x 270R resistors
3. DS18B20 digital temperature probe
4. 3 x 3V3 LEDs (red, yellow and green)
5. 2 x dual relays
6. 30cm x 30cm 3mm acrylic sheet
7. Terminal block strip

Relay board

I would like to make a note here about the relay and also provide a picture of what it actually looks like. This is important since these are “complete” relay boards. Do not get confused with the relay itself as a part, which is just the blue block. These pre-built units just require hookup directly to the GPIO interface.

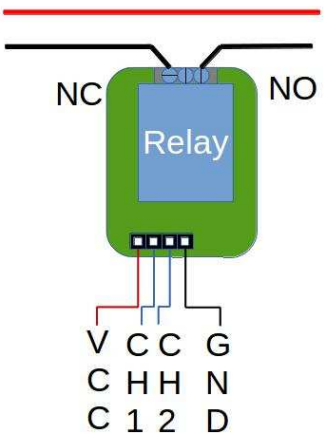
The wiring is quite simple - there are typically four pins on a dual relay.

Relay	Pi GPIO
VCC (+5V)	PIN 2
CH1 (Channel 1)	GPIO of your choice
CH2 (Channel 2)	GPIO of your choice
GND (Ground)	GPIO GND

Now hook up the relay to the device you wish to control - in my case one of my high powered LED light units. The relay is essentially going to break the circuit to the LED light unit (remember these are my high powered LED lights and not the three low power LEDs used for visual status). To keep things simple, let's take a look at the wiring for a single relay.

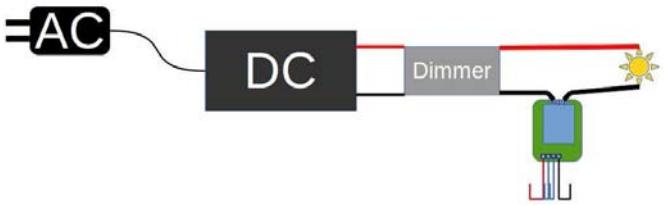
You have two choices. You can either break the power of the light system you are controlling at the AC line or at the DC line. I suggest breaking the DC line since AC voltages and currents are much more dangerous. You should seek the assistance of a qualified electrician if you intend to work on AC power.

These relays can quite comfortably switch up to 10A of AC power and up to 30A of DC power. Specs vary so be sure to buy the correct relay for the load you intend to control.



Each relay can be configured to control in one of two states - either NC (normally closed / ON) or NO (normally open / OFF). In my case all my lights have been wired to the normally open or OFF position.

The simple diagram below shows you where the relay breaks the DC line. Moving from left to right the AC from the wall plug powers the DC constant current power supply, which is rated for the LED light units. From this point on we are using DC voltages, which pass through a light dimmer control. Finally the relay breaks the DC circuit before the high power LED light unit. Although, from an efficiency perspective, it would have been better to break the AC line, I did not want to constantly spike my installation and risk damaging the power supplies.



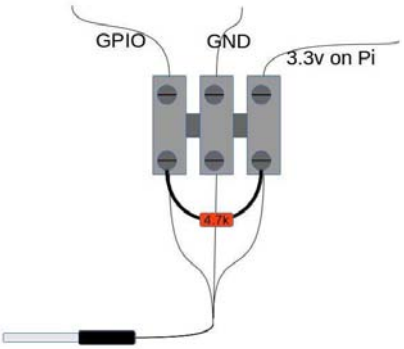
Temperature probe

I chose the DS18B20 temperature probe because it is reliable and well constructed. Even though it is made of stainless steel I would still recommend coating it in a thin film of 100% silicone. If you do not immerse it in saltwater then there is no need to do this.

I suggest you read the following article by Adafruit to understand how to use and connect the DS18B20: http://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/parts

Here is a diagram to quickly simplify the hookup. I used a terminal block to connect the wires. To protect the 4K7 resistor I placed it inside a strip of heatshrink.

There is a good reason why I didn't just solder this section up. The GPIO pin for the temperature sensor MUST be GPIO pin 4 because it is classed as a "1-wire" sensor. You are able to attach multiple "1-wire" sensors in parallel to the single GPIO pin 4 on the Raspberry Pi. The additional sensors will not need their own 4K7 resistor. The terminal block makes adding more "1-wire" sensors a very simple and neat exercise.

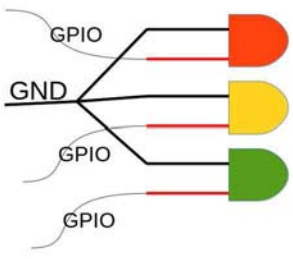


If you read the Adafruit article you will see that each "1-wire" device has its own serial number, or ID. It will register itself under its own folder on the Raspberry Pi. As a result there is no clash. Just make sure you buy digital sensors and not analogue!

Three status LEDs

Using three simple 3V3 LEDs to create a visual notification system presents no difficulty at all. The three LEDs were each wired to their own GPIO pin, making them independently addressable. I found RED, YELLOW and GREEN to be quite suitable and typical in the world of signalling.

RED is used to signal that the system is up but the schedule or cycle is disabled. GREEN is used to signal that everything is up and running. YELLOW is used to indicate activity.



For example every minute, when the temperature is read, the YELLOW LED flashes five times.

Do not forget the 270R resistors. An LED has an anode and a cathode. The resistor can be placed at either side of the LED, but the cathode (K) of the LED must connect to the ground (-) of the supply while the anode (A) connects to the positive (+).

Webcam

For the webcam I am using the Raspberry Pi camera. Note: It is important to initialise the "1-wire" modules first. In the forums some have noticed that if this is not done then the camera will freeze after taking a few shots. This is what my `/etc/modules` file contains:

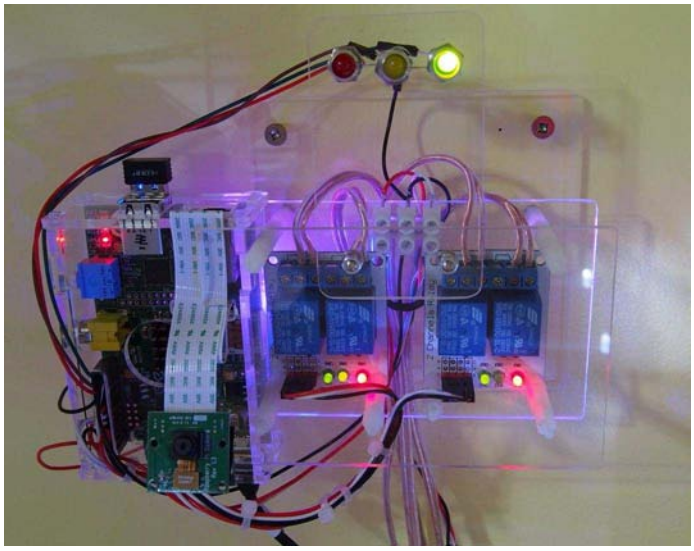
```
w1-gpio
w1-gpio pullup=1
w1-therm
#i2c-dev
#i2c-bcm2708
spi-bcm2708
snd-bcm2835
```

Some may not have experienced this, but if you do then I suggest you configure your modules to start up in the above order. (I am not using I²C hence I have left them commented out).

Putting it all together

At this point I still have the prototype body and setup. I plan to make an ornamental fish out of acrylic and produce a final case for the whole system. I will keep

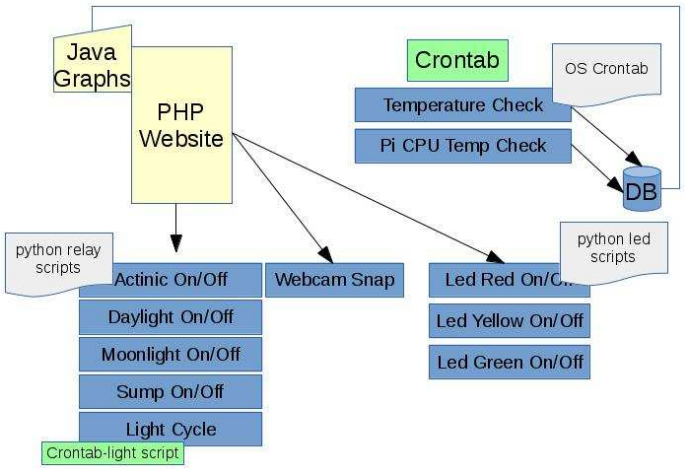
it all in a clear acrylic case so I can see all the LEDs sparkle. It also helps to show fellow inventors the components that are used.



Overall I am more than satisfied with what I was able to accomplish. This project was not too electrically complex. I had more challenges writing the code and web interface using Python, PHP, CSS and Javascript, which I will describe next.

Code overview

This section provides an overview of the code used to control and monitor my aquarium. It can be downloaded from http://piworx.subzerobc.com. I will explain the diagram below in simple terms and also provide snippets of code.



I use *apache* to provide a local web service and have installed PHP support. My PHP website executes the relevant Python scripts after a button is pressed. For example, pressing the "Daylight On" button executes the Python script `daylighton.py`, which in turn triggers the appropriate relay.

I use the Linux OS `crontab` command to schedule running the temperature reading Python scripts at one minute intervals. The two scripts `thermcheck.py` and `pi_cpu_temp.py` constantly add their results to a SQLite database. This provides the data for my Javascript Graphs module. I also have a bash script for taking pictures (`webcamshot.sh`) and a Python script to set the different LED status indicators (`status.py`). Let's get into the mechanics of the code.

The simplest script sets each status LED on or off (`led.py`). The code snippet below sets up the GPIO and defines the functions for the green LED. Similar functions are defined for the yellow and red LEDs.

```
import RPi.GPIO as GPIO # Import GPIO lib

GPIO.setwarnings(False) # Remove warnings
GPIO.setmode(GPIO.BOARD) # Use board nums
GPIO.setup(26, GPIO.OUT) # Green LED
GPIO.setup(16, GPIO.OUT) # Yellow LED
GPIO.setup(22, GPIO.OUT) # Red LED

def greenon():
    GPIO.output(26, True) #Led On
def greenoff():
    GPIO.output(26, False) #Led Off
```

Note that I am using GPIO.BOARD numbering and not GPIO.BCM. There is a big difference here so please be sure to read up on this. My git project "pipins" may be useful as it shows both the traditional GPIO.BCM and the GPIO.BOARD pin assignments (https://github.com/michaeljvdh/pipins.git).

The relay scripts do exactly what the `led.py` script does. By making a pin high or low (True or False) it will open or close the relay. Let's look at the `daylighton.py` script.

```
import RPi.GPIO as GPIO # Import GPIO lib

GPIO.setwarnings(False) # Remove warnings
GPIO.setmode(GPIO.BOARD) # Use board nums
GPIO.setup(18, GPIO.OUT)

GPIO.output(18, True)
```

The daylightoff.py script does the opposite and sets the last line as GPIO.output(18, False). This is repeated for the Actinic, moonlight and sump lights.

You may be wondering why I did not do the same as the led.py script and have all the light modes in one Python script? I made them separate because the light scripts are executed directly from the PHP website and not by another Python script. This is the simplest method for a small project.

To regularly measure the temperature I use crontab. This will execute a command at certain intervals. In this instance I needed a temperature reading taken once every minute. From the command line type:

```
crontab -e
```

This displays your crontab (schedule script). I added the following two lines at the end of the file to run the thermcheck.py and the pi_cpu_temp.py scripts. Replace xyz with your path to the scripts.

```
* * * * * sudo python /xyz/thermcheck.py
* * * * * sudo python /xyz/pi_cpu_temp.py
```

The following is the pi_cpu_temp.py script.

```
import os
# Fetch Temperature Reading
x = os.system("/opt/vc/bin/vcgencmd
  measure_temp >/xyz/pi_cpu_temp.txt")

# Clean up reading to get Celsius temp.
f = open("/xyz/pi_cpu_temp.txt")
result = f.read()
v = result.replace("\n", "")
w = v.replace("temp=", "")
x = w.replace("'C", "")
f.close()

# Write cleaned result to original file
f = open("/xyz/pi_cpu_temp.txt", "w")
result = f.write(x)
f.close()
```

The thermcheck.py script is more complex as it also writes the DS18B20 temperature reading and the Pi CPU reading to the SQLite database. Additionally, it flashes the yellow LED five times to show activity. The code is too long to include, but an important snippet to show is where to replace my temperature sensor serial value with the serial value for yours.

```
tfile = open("/sys/bus/w1/devices/28-0000
053cfcb7/w1_slave")
text = tfile.read()
tfile.close()
```

In summary the temperature readings are temporarily written to text files, which are read by the website, and they are also written to the database for use by the Javascript Graphs module.

From the web page menu you may have noticed that the lights can be either in automatic mode (cycle on) or manual mode (cycle off). The script cycleon.py controls the timed automation of the lights, along with the scripts cycleon_cron.py & cycleoff_cron.py.

The next problem I faced was how to show in the web page which lights where on and which were off. It had to be live and current. I did this by simply having the web page execute the status.py script every time it is accessed or refreshed. This script produces text files with the current state. The web page then reads these files and displays which light is on or off.

The camera was the easiest and the best part of all. I created a bash script called webcamshot.sh that uses the raspistill tool to take a picture and place it where my web page can then pick up the image.

```
#!/bin/bash
sudo raspistill -vf -hf -q 100 -n -o
  /var/www/fishpi/images/webcam/image.jpg
  -w 640 -h 480
```

Getting notified of a power failure is really easy. Simply follow online guides to configure mail on your Raspberry Pi and then add the following script to your crontab as one of the first entries. The @reboot command will automatically execute the subsequent command when a reboot occurs.

```
@reboot sudo echo "Power Failure Message"
| mail -s "PI-ALERT Possible Power
Failure" name@email.com
```

For all code, including the web page code, please visit http://piworx.subzerobc.com.

Discover new radio content across the world

Volker Ziemann

Guest Writer

SKILL LEVEL : INTERMEDIATE

A few years ago I hacked a network router to become an internet radio device by putting OpenWRT on it and adding a USB sound card. At that time this setup was the only way I could figure out how to make an intelligent box that connected to the internet at one end and output music at the other. That system, located in our kitchen, is still in use, but when we got a new stereo in the living room it was time to revisit the internet radio situation.

The Raspberry Pi has all the necessary hardware on board to enable wired or wireless internet connectivity, plus analogue and digital sound output. Having decided on the hardware, some research on the web resulted in the following strategy for the software.

- Use the music player daemon `mpd` for sound
- Control it with the music player controller software `mpc`
- Use `apache2` to serve a web interface with forms and cgi-bin functionality to select radio channels from the playlist
- Display what is currently playing
- Shutdown the Raspberry Pi gracefully via the web interface.

The last option proved to be tricky, but more on that later.

There is no hardware work involved in this approach, only some programming. The end result is a system that can be controlled from any computer, smartphone or tablet device with web access to the Raspberry Pi.

For the software approach I used Raspbian from the latest release of NOOBS. You can download this from http://www.raspberrypi.org/downloads. I assume that you already have your Raspberry Pi connected to your network, either by cable or wireless. If not, wired ethernet normally works out of the box, if you have a DHCP server on your network. Most routers offer this by default. If you have a wireless network (and a USB wireless network adapter for your Raspberry Pi) the easiest way to connect is to start the LXDE window interface by typing `startx` on the command line and then run `WiFi Config`, which you can find on the desktop.

Once you are connected to your network, you need to find out the IP address of your Raspberry Pi. From the command line, enter:

```
ifconfig
```

This will display the settings for each network adapter. Find the line that starts with `inet addr` and the four numbers that follow (e.g.

192.168.1.97) is the IP address of your Raspberry Pi. You will need this later when controlling the radio from your smartphone, etc. If you are connected by both cable and wireless, `ifconfig` will list two interfaces - `eth0` and `wlan0`. The IP address of either network interface will work for the internet radio.

Now we proceed by installing the required packages. From the command line, enter:

```
sudo apt-get update
sudo apt-get install mpd mpc apache2
```

Once installed, for good measure I suggest you shutdown then reboot the Raspberry Pi. If you do not use the HDMI cable for sound to your monitor then, while the system is shutdown, it is a good time to connect the Raspberry Pi to some active loudspeakers with a built in amplifier. Alternatively, you can connect it to the Aux/Line In input of your stereo amplifier. For this you might need a cable that has a 3.5mm jack connector attached to the Raspberry Pi audio output and RCA/phono connectors that are attached to the amplifier.

It is time to test the basic software functionality. The `mpd` program is a daemon (background process) that is started automatically at boot time. We interact with the daemon using the `mpc` program.

In order to play the radio stations we need to inform the Raspberry Pi about the channels. For this I prepared a small text file that contains the internet addresses of the radio stations I want to listen to, one station per line, with the URL and port number separated by a colon. Using either `Leafpad` in LXDE or `nano` from the command line, create a sample file with the following,

```
http://108.61.73.118:8030
http://109.123.116.202:8022
```

and save it as `playlist.txt` (the actual name does not matter - just pick any you prefer). We can now add our stations to the `mpc` playlist. From the command line, enter:

```
cat playlist.txt | mpc add
```

To check that the playlist was added, enter:

```
mpc playlist
```

This will list the stations by their URL, or by their name if they have been played.

Let's make some noise! To play the first station in the playlist enter,

```
mpc play 1
```

and to play the second station enter:

```
mpc play 2
```

Normally the play command connects to the radio station in less than two seconds and should result in some music. The first station is a classic rock station, "181.fm - The Eagle" while the second station is a classical music station, "Venice Classic Radio Italia". Look on the taskbar and notice how little CPU is being used (approximately 10%).

To display the current station plus the current artist and song being played, enter:

```
mpc current
```

To stop listening to internet radio, enter:

```
mpc stop
```

These are just the basic functions. For more details enter the following:

```
mpc help
man mpc
```

You can find a long list of internet radio stations at the following locations:

http://www.listenlive.eu,
http://www.sky.fm,
http://www.internet-radio.com and
http://www.radio-locator.com.

Check them out and listen to the diversity. To get the required URL look for the streaming file format as used by Winamp, iTunes, VLC and hardware players. This is typically a *.pls file. Download the *.pls file, open it in a text editor and view the URL. Add the URL of your favourite stations to the `playlist.txt` playlist file.

[Ed: Here are some stations from Canada, the Netherlands and the UK to get you started.]

```
http://pub7.sky.fm:80/sky_uptemposmoothjazz
http://pub6.sky.fm:80/sky_tophits
http://ice-the.musicradio.com:80/HeartLondonMP3
http://8573.live.streamtheworld.com/SKYRADIO_SC
http://8343.live.streamtheworld.com/CHQMFMAAC_SC
```

At this point you could take a shortcut and install for example "mpdroid" on your Android device or "MPoD" on your iOS device and remotely control the mpd daemon on your Raspberry Pi. You would get some functionality, but miss out on the fun of learning some HTML and cgi-bin controls of your Raspberry Pi.

HTML

At this point the basic audio functionality is established and we need to add the control interface, which is provided by the *apache2* server. Using the IP address of your Raspberry Pi that you determined earlier, using any computer on your local network, including the Raspberry Pi itself, start a web browser and enter the IP address of your Raspberry Pi into the address line. You should receive a message that says:

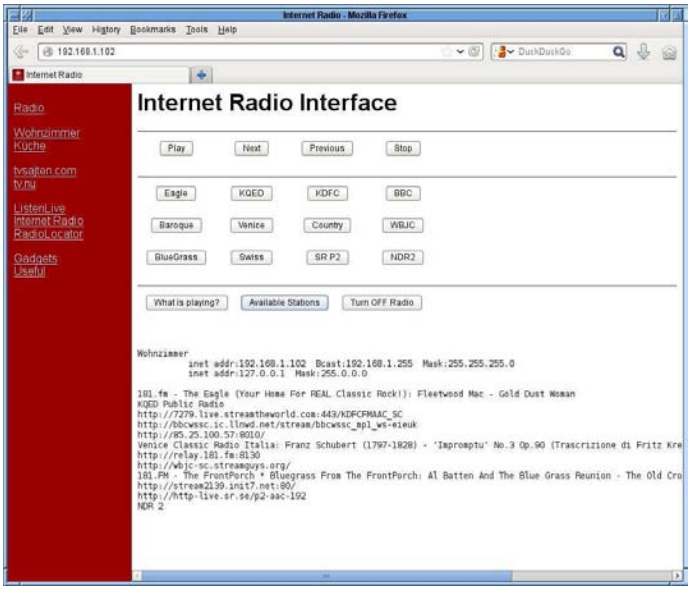
It works!

This is the default web page for this server.
The web server software is running but no content...

This means that your *apache2* web server is running, which it should do automatically after installing it - if you followed the instructions.

The web server basically dishes out web pages which are located on the Raspberry Pi. The page you were just shown is actually a file named `index.html` that resides in the directory `/var/www/` by default. You can change that location by altering the *apache2* configuration files, but I have assumed you will use the default configuration that came with the installation, as I do. Note that you need root permission to put files there; use `sudo` before each command.

We need to replace the `index.html` file with another one that provides the internet radio interface. I will show you how in a minute, but here is a screenshot of what I ended up with.



On the left hand side there is a red area with several useful links like the different internet radios scattered in my household, links to Swedish TV programmes, links to sites with internet radio stations and other useful things.

On the right hand side, in the upper half of the screen, there is the radio interface with buttons to start and stop listening plus move to the next or previous station. Below that there are twelve buttons for the twelve radio stations in my playlist file `playlist.txt`. The three buttons in the bottom row - "What is playing?", "Available Stations" and "Turn OFF Radio" - are self-explanatory.

Finally, below the buttons is a status area where the information from the button presses is displayed. In the above example I pressed the "Available Stations" button and the twelve links are displayed.

Having described the interface, we now turn to the implementation. To set up the main window we divide it into three parts; the left red sidebar, the button interface and the status display. I use HTML framesets. The `index.html` file that replaces the default one in `/var/www/` looks like:

```
<html>
<head>
  <title>Internet Radio</title>
</head>
<frameset cols="170,*" frameborder="0" framespacing="0"
    border="0">
  <frame src="indexlist.html" name="index">
  <frameset rows="50%,50%" frameborder="0"
      framespacing="0" border="0">
    <frame src="radio.html" name="main">
    <frame src="status.html" name="status">
  </frameset>
</frameset>
</html>
```

The head contains just the title information. Then there are two nested framesets. The first contains a frame with a file named `indexlist.html`, which contains the content in the red sidebar. The second frameset contains two frames, one for the radio interface, `radio.html`, and one for the status page, `status.html`. The frameset instructions contain specifications of the sub-division of the page. The first frameset sub-divides the page into two parts, where the red sidebar will be 170 pixels wide and the rest is given to the second frameset. The second frameset splits its allocated space evenly between two frames with the radio and the status. Note that the frames have names, which we will later use to direct output to the proper places.

The file `indexlist.html` file that describes the red sidebar on the left is shown here:

```
<html>
<body bgcolor=#990000 text=white link=white
    vlink=lightblue>
  <br>
  <a href="radio.html" target=main>Radio</a>
  <br>
  <br>
  <a href="http://192.168.xxx.xxx/" target=blank>Kitchen</a>
  <br>
  <a href="http://192.168.xxx.xxx/" target=blank>Living</a>
  <br>
  <br>
  <a href="http://www.listenlive.eu/" target=blank>
      ListenLive</a>
  <br>
  <a href="http://www.internet-radio.com/" target=blank>
      Internet Radio</a>
  <br>
  <a href="http://www.radio-locator.com/" target=blank>
      RadioLocator</a>
</body>
</html>
```

We see the usual header and body directives, where the red background is defined. Then there is a list of HTML anchors with the links to the respective web pages. Note that the anchors have a target directive, which directs the browser when you click the link to either a new window with `target=blank`, or to a named frame with `target=main`. The main frame is the radio area. We will use this feature later to direct output to the status section of the page.

The status page is rather trivial. It just contains the opening and closing html and body directives and some dummy text. This will be overwritten dynamically later.

CGI-BIN

The file `radio.html` contains all the fun interactive stuff. This is where the interaction with the Raspberry Pi actually takes place. The interface is built as a horizontal table sandwiched between <TR> and </TR> tags. Each entry in the table is declared between the <TD> and </TD> tags. In between these tags a <FORM> is defined that executes an action on the Raspberry Pi, as defined by the instruction `action="cgi-bin/play.sh"`.

```
<html>
<head>
  <title>Internet Radio</title>
</head>

<body>
  <h1>Internet Radio Interface</h1>
  <hr>

  <table border="0">
    <tr>
      <td width=100 align=center>
        <form action="cgi-bin/play.sh" method="POST"
            target=status>
          <input type="submit" value="Play">
        </form>
      </td>
                  :
                  :
    </tr>
  </table>

  <hr>

  <table border="0">
    <tr>
      <td width=100 align=center>
        <form action="cgi-bin/play.sh?1" method="POST"
            target=status>
          <input type="submit" value="Eagle">
        </form>
      </td>
                  :
                  :
    </tr>
  </table>
</body>
</html>
```

This uses the POST method and directs the output of whatever the play.sh script produces to the status window. To execute the form we define a button through the <INPUT> tag which displays "Play" and submits the request to execute the play.sh script on the server. See the excerpt from the radio.html file on the previous page.

The <TD> block between the <TR> tags is repeated four times to produce four rows of buttons. In between there are four blocks with the <TD> tags and the enclosed form. Notice that I supply an argument to the play.sh script by appending a question mark and a number.

You may wonder what these scripts are and where they reside on the Raspberry Pi. When the apache2 server is installed, a directory called /usr/lib/cgi-bin is automatically created and you just put the scripts in there. Note that you need root permission to put files there; use sudo.

Let's begin with the stop.sh script, because it is one of the simplest. Here is the content of the file /usr/lib/cgi-bin/stop.sh:

```
#!/bin/bash

echo "Content-type: text/plain"
echo
echo

/usr/bin/mpc stop
echo "Radio stopped..."
```

The first line with #!/bin/bash tells the operating system that the contents of this file should be treated as a sequence of bash commands, where bash is a common shell (or command interpreter) for Linux. The empty lines in the file are just for structuring and to 'guide the eye'. The three lines with echo tell the browser what content type to expect, plus two empty lines. The setup with the forms in the HTML causes the output to be piped to the status section of our web page.

The HTML interpreter of the browser that receives that information is instructed by the lines that there is just plain text following and therefore the browser has an idea how to format it. The line with /usr/bin/mpc stop executes the mpc stop command on the Raspberry Pi and whatever output it generates ends up in the status window. The last echo command writes 'Radio stopped...' to the status window.

Finally we need to make the file executable:

```
sudo chmod +x /usr/lib/cgi-bin/stop.sh
```

This changes file permissions by adding the execute bit with chmod +x.

The script files for 'Next', 'Previous' and the buttons in the bottom row all work in the same fashion. First there's the #!/bin/bash line followed by the 'Content' stuff with the three echo commands. Then there is just whatever mpc commands are required, with the output piped to the status window.

Retrieving command line arguments

The only exception is the play.sh command because I pass arguments to it from the browser. You can see from the radio.html code on the previous page that I pass the station number from the playlist in the FORM tag with action="cgi-bin/play.sh?1".

So how do we recover the string "1" when executing the file play.sh? The FORM tag with cgi-bin actually makes the arguments that follow the question mark available within the executing script, in the environment variable ${QUERY_STRING}. Here is the content of /usr/lib/cgi-bin/play.sh:

```
#!/bin/bash
echo "Content-type: text/plain"
echo
echo
if [ -z ${QUERY_STRING} ]; then
    /usr/bin/mpc play
else
    /usr/bin/mpc play "${QUERY_STRING}"
fi
```

The script tests if QUERY_STRING is empty and, depending whether it is or not, executes the mpc play command with or without the argument. Note that [is an alternate name for the test command (see man [). The -z option returns True if the length of QUERY_STRING is zero.

WARNING: Please note that I do not check the QUERY_STRING variable for illegal characters. A malicious user could hack your system in this way. I use my Raspberry Pi behind a router on a subnet only accessible by friendly users. Do not put the system on an unprotected network without adding additional validation.

Killerqueen

So far we have the basic functionality to control the mpd daemon via the mpc program using a web interface. All that remains is to turn the Raspberry Pi off using the web interface... and that proved to be tricky. Just pulling out the power cable works, but at the risk of corrupting the file system. It's also not very graceful! I wanted a button on the web interface that would cleanly shutdown the Raspberry Pi.

My initial attempts to directly call the shutdown command via cgi-bin did not work, so I resorted to a trick. From the web interface I create an empty file in the /tmp directory called killerqueen (remember that Queen song?). In the background I need another program that periodically checks the existence of the /tmp/killerqueen file and calls shutdown if it exists.

Here is the shutdown.sh script that creates the /tmp/killerqueen file if the button on the web interface is pressed:

```
#!/bin/bash
echo "Content-type: text/plain"
echo
echo

touch /tmp/killerqueen
echo "Shutting down..."
```

The file that periodically checks its existence every two seconds is called killerqueen.sh which I placed in the /usr/local/bin directory. It contains the following:

```
#!/bin/bash
rm -f /tmp/killerqueen
while [ True ]; do
    if [ -e /tmp/killerqueen ]; then
        echo "file exists, removing"
        rm -f /tmp/killerqueen
        /sbin/shutdown -h now
    fi
    sleep 2
done
```

After initially removing any forgotten /tmp/killerqueen file, it enters an infinite loop and checks the existence of the file. If it exists it removes the file and calls shutdown. Then the Raspberry Pi comes to a halt and after a few seconds you can turn the power off safely.
[Ed: If you have Raspberry Pi switches from companies like Pi Supply (UK) (http://www.pi-supply.com) or Mausberry Circuits (USA) (http://www.mausberrycircuits.com) then these will automatically power off your Raspberry Pi.]

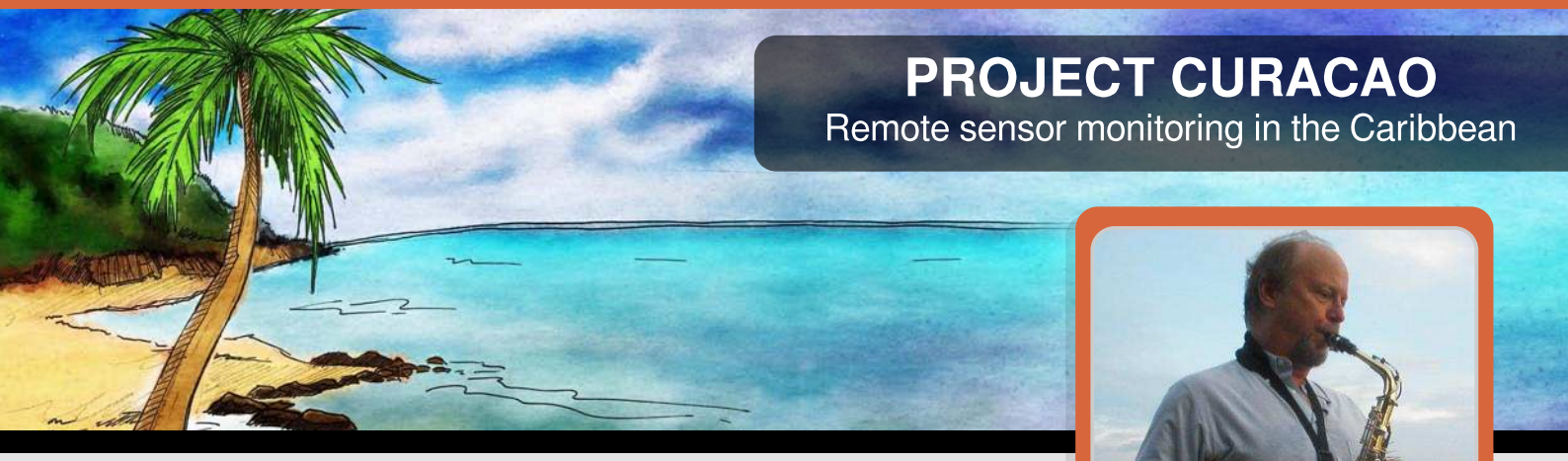
What remains is to start the killerqueen.sh script automatically when the Raspberry Pi is booted. This is easily done by placing the line,

```
/usr/local/bin/killerqueen.sh &
```

in the file /etc/init.d/rc.local. This is executed at boot time and starts the killerqueen.sh script as a background task.

We have come quite a way in the process of turning the Raspberry Pi into an internet radio. We covered the mpd and mpc combination plus setting up a web server. We created simple forms with framesets and interacted with the Raspberry Pi via cgi-bin scripts.

Please be aware that this is not the only way to achieve this functionality, but for me it was both fun and instructive. All code is available from http://ziemann.web.cern.ch/ziemann/gadget/raspiradio/raspiradio.tar.gz.



PROJECT CURACAO
Remote sensor monitoring in the Caribbean

John Shovic

Guest Writer

Part 4: The software architecture

SKILL LEVEL : INTERMEDIATE

What is Project Curacao?

This is the fourth part of a series discussing the design and build of Project Curacao, a sensor filled project that will hang on a radio tower on the island nation of Curacao. Curacao is a desert island 12 degrees north of the equator in the Caribbean. Part 5 will show the actual deployment and the first weeks of operation.

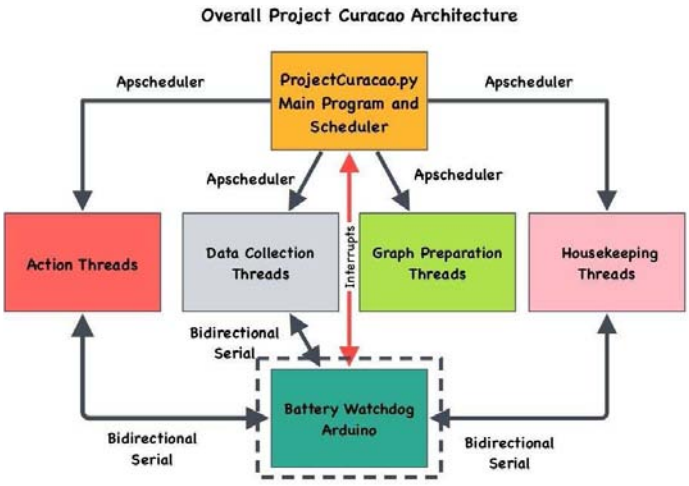
Project Curacao is designed to monitor the local environment unattended for six months. It operates on solar power cells and will communicate with the designer via an iPad app called RasPiConnect. All aspects of this project are designed to be monitored and updated remotely (with the current exception of the BatteryWatchdog Arduino).

System description

Project Curacao consists of four subsystems. A Raspberry Pi Model A is the brains and the overall controller. The Power subsystem was described in part 1, the Environmental Sensor subsystem was described in part 2 and the Camera subsystem was shown in part 3. This article goes through the software architecture of the system.

There are two computers in Project Curacao. The main "brains" of the project reside in the Raspberry Pi in a program called ProjectCuracaoMain. The ProjectCuracaoMain program contains approximately 5000 lines of code in Python. The BatteryWatchdog Arduino has 3000 lines of code written in C/C++. The third program, RasPiConnect, contains 3000 lines of code (counting only Local.py - the customized code) and is used for display, control and remote access from an iPhone or iPad.

Architecture of the Python code



The ProjectCuracaoMain Python file is not complex. It consists of some setup code and then the initialisation of the Python package `apscheduler` with the various programs to be run periodically. The scheduling code is very simple and is shown below.

```
scheduler = Scheduler()
job = scheduler.add_cron_job(powerdatacollect.datacollect5minutes, minute="*/5", args=['main', 0])
job = scheduler.add_cron_job(watchdogdatacollect.watchdogdatacollect, minute="*/5", args=['main', 10])
job = scheduler.add_cron_job(environdatacollect.environdatacollect, minute="*/15", args=['main', 70])
job = scheduler.add_cron_job(systemstatistics.systemstatistics15minutes, minute="*/15", args=['main',40])
job = scheduler.add_cron_job(doallgraphs.doallgraphs, minute="*/15", args=['main',10,100])
# camera
job = scheduler.add_cron_job(useCamera.takeSinglePicture, hour="*", args=['main',50])
# send daily picture
job = scheduler.add_cron_job(sendPictureEmail.sendPictureEmail, hour="22",minute="20", args=['main',0])
sys.stdout.write('Press Ctrl+C to exit\n')
scheduler.start()
```

The key concept of the above code is that each of the jobs is fired off in its own thread and not in the main thread of ProjectCuracaoMain. What is the advantage of this? If the thread code bombs then just the thread code stops running, leaving the main program and other threads intact.

You can also run more than one thing at a time, but we minimise that to avoid the load on the Raspberry Pi. The `apscheduler` Python package, detailed at http://pythonhosted.org/APScheduler, generates the threads for you, so it is very easy to use.

We communicate between threads through files in the "state" directory. One improvement we will be making in the thread code is to put in a lock for the BatteryWatchdog serial port to keep threads from stepping on each other when sending and receiving serial data out to the BatteryWatchdog. New to threading? An excellent tutorial is at http://www.tutorialspoint.com/python/python_multithreading.htm.

Each of the threads are simple pieces of code that actuate hardware, send email, build graphs, collect data and write to the MySQL database.

The remainder of the main program code is for setup and handling interrupts.

Doing hard time

The main software (and the BatteryWatchdog) is a realtime system. It is characterized as a "soft" realtime system because it is not guaranteed that a particular task will run at the exact time requested. Other system tasks may interrupt the software. Tasks can even be lost. This is fine for this system, but you don't want your car braking system to operate that way. "Hard" realtime systems are designed to ensure that all tasks are executed when required.

Watchdog timer

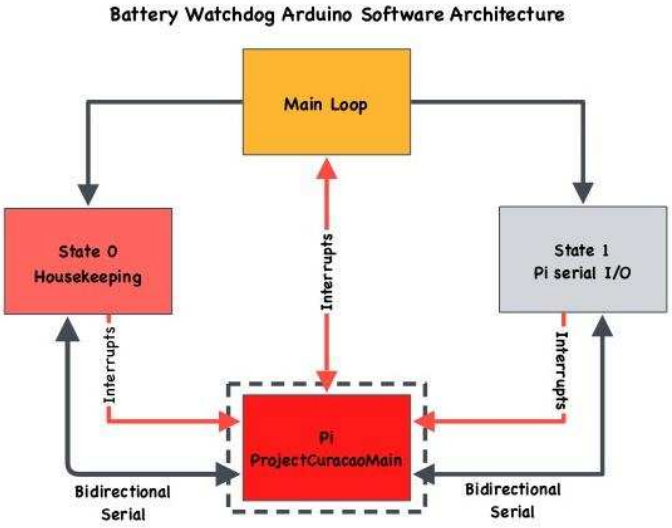
Anyone who has written complex code knows that there are still bugs remaining in the software which could prevent the software from functioning. It is also true that the Raspberry Pi OS sometimes hangs and there is always the danger of a voltage spike, lightning bolt or something causing the Raspberry Pi to stop responding. We combat this with a watchdog timer implemented with an Arduino to reboot the Raspberry Pi periodically and also when there is an issue. The Raspberry Pi sends an "I am still here" to the BatteryWatchdog every five minutes (in the sendWatchDogTimer.py program under the housekeeping directory).

If the BatteryWatchdog doesn't receive these messages for 15 minutes, then it sends an interrupt to the Raspberry Pi and tells it to shutdown (always a good idea if possible). The BatteryWatchdog waits for 5 minutes and then power cycles the Raspberry Pi. Then you have the question of "Who watches the watchdog?". This is partially answered by the use of a deadman switch in the BatteryWatchdog software, which was detailed last month. A better solution would be to implement a hardware timer that would reboot the BatteryWatchdog if it goes away.

Crontab entry

`crontab` is a utility on the Pi that will periodically call a program or a command. We only use `crontab` for one thing in Project Curacao and that is to send the graphs and current photograph to a webpage hosted by our good friends at MiloCreek (developer of RasPiConnect). You can see them at http://milocreek.com/projectcuracaographs.

Architecture of the BatteryWatchdog



The BatteryWatchdog is built on a state machine model. We have two main states in the system. State0 contains housekeeping code, the timed alarms and the watchdog code. State0 is the "normal" run state of the system.

```
void loop() {
  // state machine
  printStateVariables();

  switch (currentState)
  {
     case  STATE0:
       nextState = state0(currentState);
     break;

     case STATE1:
        nextState = state1(currentState);
     break;

     default:
     break;
  }
  checkForAlarms();

  if (nextState == currentState)
  {
    // sleep
    delay(1000);
    sleep.pwrDownMode();
    sleep.sleepDelay(sleepTime);
  }
  currentState = nextState;
}
```

When an interrupt is received from the Raspberry Pi the software transitions to State1, where the serial data from the Raspberry Pi is dealt with and data is sent to the Raspberry Pi. This is a classic software state machine design.

Note that we did not use the timed alarm Arduino library as it doesn't work if you put the Arduino to sleep (to save current). We also used an average reading technique to reduce the noise and issues in the Arduino built-in analogue-to-digital converters.

Total sensor count

We have 11 I²C devices in Project Curacao. We also use 14 GPIO pins and a total of 5 A/D pins. For a complete list of all pin functions, go to http://switchdoc.blogspot.com/2013/12/gpio-and-i2c-mappings-for-project.html.

RasPiConnect code

The RasPiConnect app can be obtained from http://www.milocreek.com. All the code is located in the Local.py file available on GitHub https://github.com/projectcuracao/RasPiConnectServer. It involves a total of 93 controls and a number of graphs. We use it on a regular basis to monitor the project.

Remote access to the system

Remote access to the system is achieved with SSH and by HTTPS via RasPiConnect. Since these ports are exposed to the internet both are password protected with 12 letter random passwords. A rule to remember is ANYTIME you expose a port to the internet, you need to use complex difficult passwords. We use https://identitysafe.norton.com/password-generator to generate these.

What is coming up?

Deployment! We are taking the finished box to Curacao on 3rd March, 2014. Part 5 will describe the process of installing the unit in Curacao and show the first weeks of results. All of the code used in this article is posted on GitHub at https://github.com/projectcuracao.

More discussion on Project Curacao at: http://switchdoc.blogspot.com

RasPiConnect

Connect your Raspberry Pi to the World!

Allows you to control virtually anything you connect to your Raspberry Pi from your iPad or iPhone.

- EASY to setup - no syncing required
- Buttons, gauges, webpages, webcam pictures and more!
- Exchange your panels with friends
- Supports multiple Raspberry Pis and multiple Arduinos
- Build your pages on your iPad/iPhone
- Ten pages of control panels
- Unlimited Controls
- Supports any computer that supports Python (windows, linux, etc.)
- *Now Allows Custom Backgrounds*

Available on the App Store

MiloCreek
Where Innovation Flows

Supports Arduino with in-app purchase

UPiS module

The **all-in-one** solution

A wealth of features in a single board:

- UPS function with Li-Po battery
- Versatile powering for RPi (7-18VDC)
- Hardware ON/OFF switch for the RPi
- Software-readable V/mA/°C sensors
- Battery backed-up real-time clock
- Timer controlled RPi ON/OFF
- RS232 and USB interfaces
- I^2C PiCO interface
- 1-wire and iButton input
- Basic I/O pin and relay outputs
- XTEA-based encryption of user software

available from distributors :

πmodules www.pimodules.com

Intelligent modules for your Raspberry Pi

RASPPISHOP · Pi Supply · MODMYPI · NETTOP · PI-SHOP.CH · www.m.nu MATTEKNIK FÖR ALLA

Making the Raspberry Pi portable

Jacob Roberts

Guest Writer

SKILL LEVEL : BEGINNER

My desktop computer recently "gave up the ghost"! To be honest I needed a new computer. Unfortunately, being 13 years old in Year 9 at school and with an income of £8 ($13) a week from a paper round, even a low-end netbook is out of the question. So I turned to my Raspberry Pi. "How hard can it be to make it portable", I thought? How wrong was I!

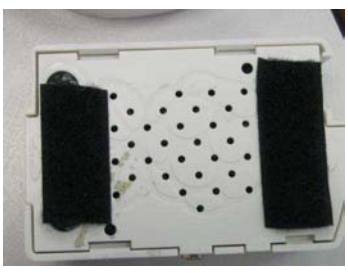
I tried following instructions from others such as bypassing regulators and modifying stuff. One fried screen and a lot of frustration later, I decided that I would design my own version.

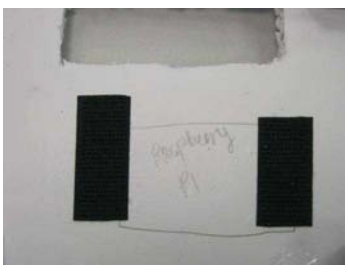
May I present the Pi Book Air, very much a portable Raspberry Pi for the rest of us.

The Pi Book Air is designed to be simple, with the only soldering being for a 9V battery connector and one switch. Here is a list of parts:

- Raspberry Pi Model B
- Raspberry Pi case
- Sheet of foam board
- Powergen 5V external battery pack
- 3" LCD TFT colour monitor
- Toggle switch
- 9V battery connector
- 9V rechargeble battery
- Rii mini wireless keyboard
- Sticky back velcro tape
- Double sided tape

The first thing I did was to prepare a rectangle of foam board 20cm x 29cm and to cut a handle 2cm from the top. Next I put the Raspberry Pi in the case and put one side of the sticky back velcro on the case and the other on the foam board just under the handle.





Next I cut one slit just under the Raspberry Pi case and another slit the width of the screen futher down. I then cut two small rectangles of foamboard and put them in the slits and stuck the screen to them using the normal tape.

Next I stuck the charger to the right of the screen with the double sided tape and connected the micro usb cable to the Raspberry Pi.

The picture shows that I used a long wire even though the charger came with a shorter cable that I managed to break! The other side of the screen I left empty but you could put a case for SDs or a USB hub there. In the top right hand corner I put a simple holder for the 9V battery.

I soldered the black wire from the 9V connector to the ground wire for the screen. I then soldered the red wire from the 9V connector to one side of the switch. The other side of the switch was soldered to the power wire for the screen. I taped the power switch to the side of the screen, pushed the connector under the screen and attached the connecter to the 9V battery. I taped any hanging wires under the main piece of foam board.

I attached the AV-cable between the Raspberry Pi composite video connector and the screen, and then pushed the excess wires under the screen.

The mini keyboard was attached just under the screen with double sided tape. If you want to use the mini keyboard for other purposes, you could of course use sticky-back velcro instead.

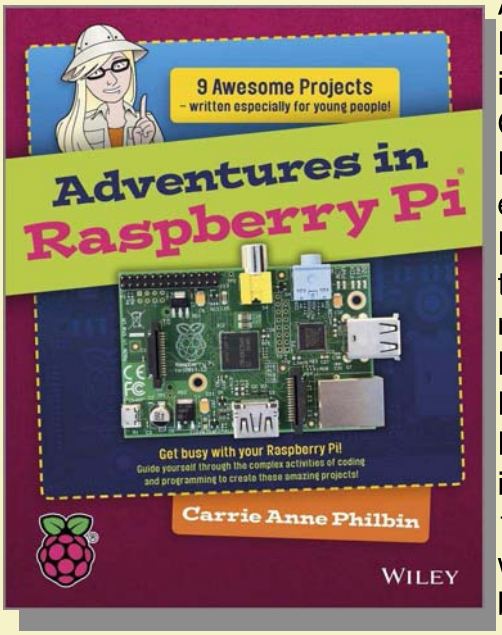
We can now attach the bluetooth dongle for the keyboared to the Raspberry Pi's USB slot.

To finish off my Pi Book Air I added some fancy graphics on top of my Raspberry Pi case.

[Ed: The MagPi applauds Jacob's creativity and ingenuity. Inspiring minds, both young and old, is exactly what the Raspberry Pi is about. If this article has inspired you, visit Adafruit at http://www.adafruit.com/products/1601 for some possible screen solutions.]

Adventures In Raspberry Pi

Carrie Anne Philbin
Wiley



Adventures in Raspberry Pi is written by Carrie Anne Philbin, an experienced High School teacher and now Education Pioneer at The Raspberry Pi Foundation. It is aimed at 11-15 year olds, who are beginning with computer programming, and takes you through a fun and varied journey of nine engaging adventures using your Raspberry Pi.

After starting with how to connect everything, the book covers a wide range of topics including programming games and activities in Scratch, programming in Python, making music with Sonic Pi and controlling hardware with GPIO pins. In the final "big adventure" you build a fully working MP3 music player, based on learning from previous adventures. This is really great fun to build and program!

To make your adventures even easier there is a collection of great video tutorials on the supporting website http://www.wiley.com/go/adventuresinrp, along with supporting code files.

This is a fun and engaging book for children, with many interesting real projects to build and extend further, that will help readers get the most from their Raspberry Pi.

The MagPi and Wiley are pleased to offer readers a 30% discount. To claim, order from www.wiley.com and quote promo code **VBF77**.

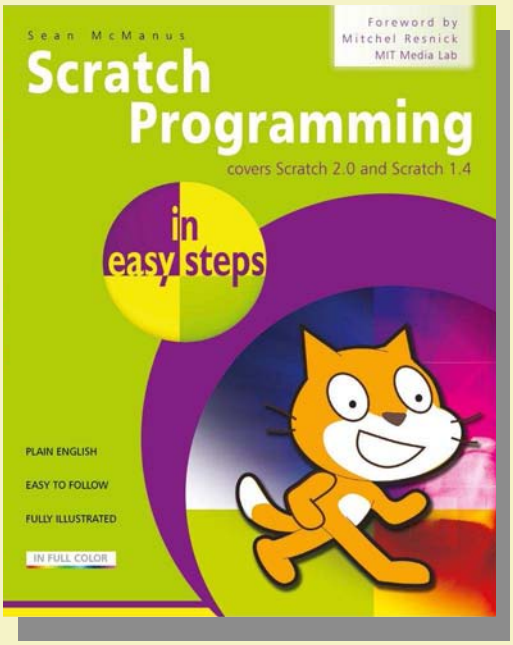
Scratch Programming In Easy Steps

Sean McManus
In Easy Steps

Scratch is a programming language that is widely used on the Raspberry Pi and in schools and colleges. Scratch's highly visual interface and drag-and-drop commands make it an ideal language for all ages to use to program. With Scratch Programming in easy steps, learning this useful programming language will be an absolute breeze.

This primer introduces you to Scratch fundamentals and then walks you through the commands to create games and animations. Learn to create games that require skill, knowledge or quick fingers, such as Spiral Rider, Space Swarm, or the classic Hangman game. Add music and special effects to your games and of course keep score.

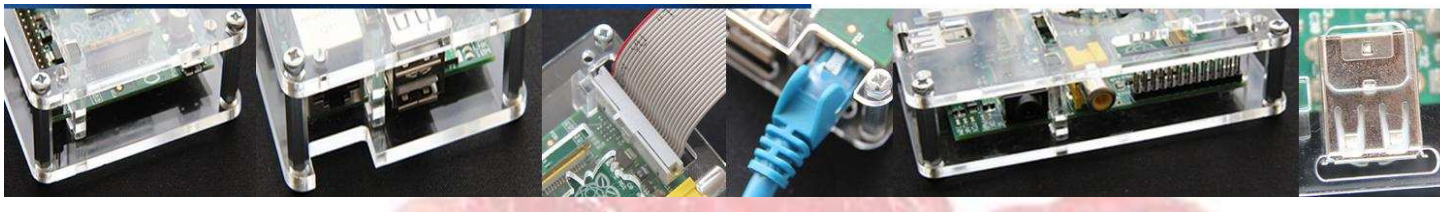
By the time you have finished with Scratch Programming in easy steps, you will be impressing your friends and family with your own computer games!



As always with the In Easy Steps books, it is written in plain English, is fully illustrated and is printed in full colour. If you are looking to start programming in Scratch then this will surely be a perfect addition to your bookshelf.

In Easy Steps are offering The MagPi readers 30% off all titles (including ebooks) between March 1st and 31st. To claim, order from www.ineasysteps.com and quote **MAGPI20**.

MARCH COMPETITION



Once again The MagPi and PC Supplies Limited are proud to announce yet another chance to win some fantastic Raspberry Pi goodies**!**

This month there are TWO winning prizes!

The first winner will receive a Pi NoIR camera board, 32GB SD card, VESA mount case and a gigabit network hub!
The second winner will receive a Cyntech Geek case, GPIO breakout board, breakout cable set and a 16GB SD card!

For a chance to win this month's competition visit **http://www.pcslshop.com/info/magpi**

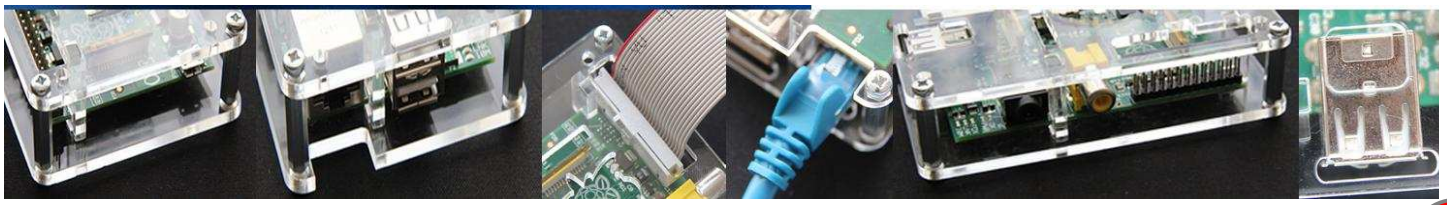
Closing date is 20th March 2014.
Winners will be notified in the next issue.

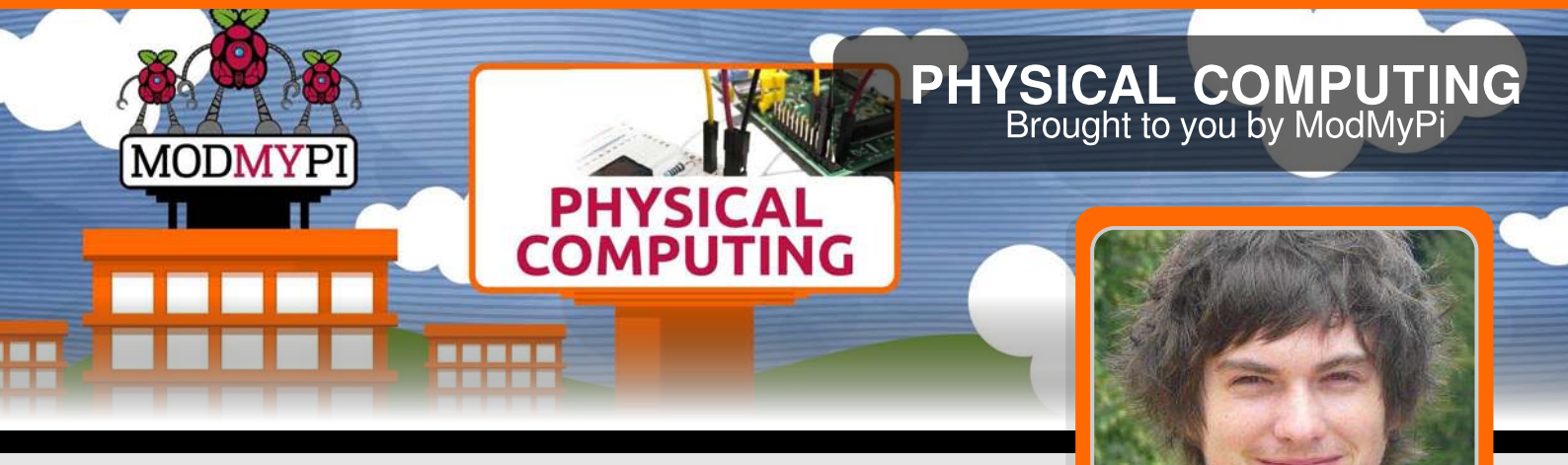
To see the large range of PCSL brand Raspberry Pi accessories visit
http://www.pcslshop.com

February's Winner!

The winner of a Cyntech Special Edition Geek case, Pi NoIR camera board, GPIO breakout board, breakout cable set, VESA mount, 16GB and 32GB SD cards and a gigabit network hub is **Robin Mackay (Edinburgh, Scotland).**

Congratulations. We will be emailing you soon with details of how to claim your prizes!





GPIO Sensing: Motion Detection - Part 1

SKILL LEVEL : BEGINNER

Jacob Marsh

ModMyPi

In previous tutorials in Issues 15 to 19 of The MagPi, the basics behind physical computing with the Raspberry Pi computer were introduced. The previous articles covered basic Python programming and Board/BCM GPIO numbering, which are both used in this tutorial.

In this tutorial, a passive infrared sensor (PIR) is used instead of a switch. The PIR is used to activate a print statement within a Python program when motion is detected.

PIR sensors

PIR sensors provide a simple way of detecting motion. Everything on the Earth emits a small amount of infrared radiation, where hotter objects emit more radiation. PIR sensors are able to detect a change in infrared levels within their detection zone. For example, when someone comes into a room the PIR detects the infrared radiation change.

Assembling the circuit

For this tutorial, the following parts are required:

* a breadboard
* 6 x male to female jumper wires
* a PIR Sensor

All of the components can be purchased from the ModMyPi online shop.

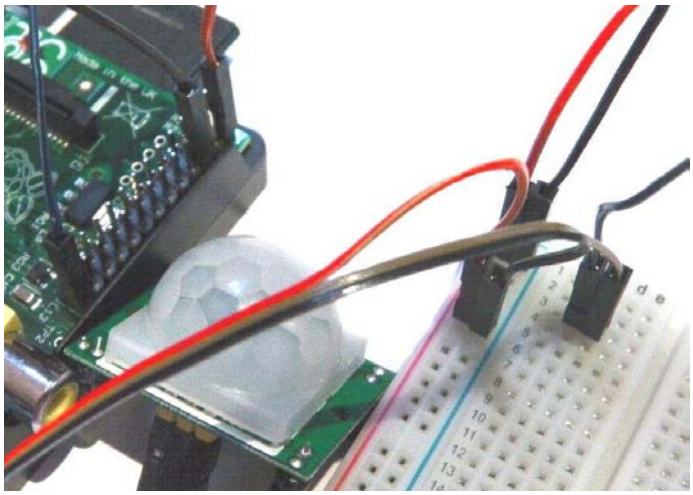
1. Use three male to female jumper wires and connect the female ends to the PIR sensor terminals. The three pins on the PIR are labelled as: Red; PIR-VCC (3-5VDC in), Brown; PIR-OUT (digital out) and Black; PIR-GND (ground).

2. Plug the male jumper wire connected to PIR-VCC into the positive rail of the breadboard, plug the PIR-GND wire into your negative rail and plug the PIR-OUT wire into any other blank rail.

3. Use a black jumper wire to connect GPIO GND [Pin 6] on the Raspberry Pi to the negative rail of the breadboard, to which the PIR-GND wire is already connected.

4. Use a red jumper wire to connect GPIO 5V [Pin 2] on the Raspberry Pi to the positive rail of the breadboard, to which the PIR-VCC wire has already been connected.

5. Connect GPIO 7 [Pin 26] to the same rail as the PIR-OUT.



Sensing with Python

The status of the PIR can be checked in a similar way as the switch mentioned in previous tutorials. However, the PIR does not need a pull-up resistor since it outputs 0V or 3V3. The GPIO 7 [Pin 26] connected to the PIR-OUT just needs to be set as an input pin.

Type in the following using the nano text editor:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
PIR = 7
GPIO.setup(PIR, GPIO.IN)
try:
  print("PIR Module Test")
  print("(CTRL+C to exit)")
  time.sleep(2)
  print "Ready"
  while True:
    if GPIO.input(PIR):
      print("Motion detected!")
    time.sleep(1)

except KeyboardInterrupt:
  print("Quitting")
  GPIO.cleanup()
```

The program includes several steps that were introduced in the previous articles. At the start there are two import statements to allow the GPIO and time functions to be used.

```
import RPi.GPIO as GPIO
import time
```

Then there is the setmode function,

```
GPIO.setmode(GPIO.BCM)
```

to configure the GPIO in BCM numbering mode. More information on the BCM numbering scheme can be found at:
http://elinux.org/RPi_Low-level_peripherals

After the GPIO setmode, a variable PIR is used to store the connection associated with the PIR digital out. This connection number is used to setup the associated pin as an input with:

```
GPIO.setup(PIR, GPIO.IN)
```

The example program includes a try-except statement, to clean up the GPIO connections when the program exits. Inside the try statement there are some print statements and a while loop. The while loop polls the GPIO input every second. If the GPIO pin is high then,

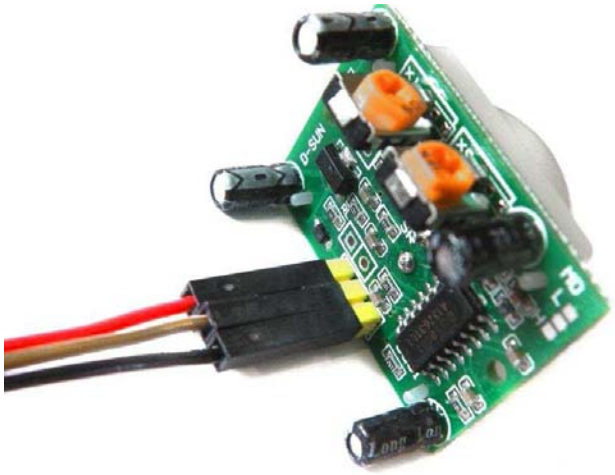
```
Motion detected!
```

will be printed on the screen.

PIR settings

On the PIR, there are two potentiometers that adjust the amount of time the PIR is high and also its sensitivity. Follow the labelling on the

board, which indicates the function of each potentiometer. If the program continues to report that motion is detected then try turning down the sensitivity of the PIR. If the potentiometer is turned clockwise then sensitivity will be increased.



GPIO callbacks

In the previous examples, GPIO polling has been used to check the state of a GPIO input pin. However, the GPIO also allows callbacks. This means that a pin can be set to call a Python function if the state of the input pin changes.

This is much better than polling since the function is called immediately, without the need for a high polling frequency.

This time when the program runs, the CPU of the Raspberry Pi waits for the signal on the PIR to rise. Once the signal rises, the function callback_up is called. The callback function is passed the channel number of the GPIO pin that went high. Possible GPIO tests are `GPIO.RISING`, `GPIO.FALLING` and `GPIO.BOTH`.

Next time

That's it, a very simple method of connecting a low cost movement sensor to the Raspberry Pi.

The next article will include a buzzer, some LEDs and a magnetic door lock. These parts will be used to build a low cost Raspberry Pi based security system!

```
import RPi.GPIO as GPIO
import time

def callback_up(channel):
  print("Up detected on channel %s" % channel)

PIR = 7
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR, GPIO.IN)

try:
  GPIO.add_event_detect(PIR, GPIO.RISING, callback=callback_up)
  while 1:
    time.sleep(100)

except KeyboardInterrupt:
  print("Cleaning up the GPIO")
  GPIO.cleanup()
```

The MagPi What's On Guide

Want to keep up to date with all things Raspberry Pi in your area?

Then this section of The MagPi is for you! We aim to list Raspberry Jam events in your area, providing you with a Raspberry Pi calendar for the month ahead.

Are you in charge of running a Raspberry Pi event? Want to publicise it?
Email us at: editor@themagpi.com

Torbay Raspberry Jam

When: Saturday 8th March 2014, 1.00pm to 3.00pm
Where: Paignton Library and Information Centre, Great Western Road, Paignton, TQ4 5AG, UK

The second Torbay Raspberry Jam. Details of the next event are at http://dcglug.drogon.net/meetings and a review of the previous event can be read at http://dcglug.drogon.net/previous-jams.

Corso Raspberry Pi

Quando: Sabato 8 marzo 2014 dalle 9.00am alle 6.00pm
Dove: Campus La Camilla, 267 Via Dante Alighieri, 20863 Concorezzo, Italia

Il corso è estremamente pratico: http://eventbrite.it/e/10566814627. Breve descrizione dell'hardware, Porta GPIO, Programmare con Python, Conversione A/D con c.i. MCP3008, Gestione LED.

PiCymru - Swansea Raspberry Jam

When: Saturday 8th March 2014, 1.30pm to 3.00pm
Where: TechHub Swansea, 11 Wind Street, Swansea, SA1 1DP, UK

The is the first event organised by PiCymru, the Raspberry Pi user group for Wales and will include talks and demos and a walkabout show and tell. http://eventbrite.co.uk/e/10323996351

DigiMakers Bristol

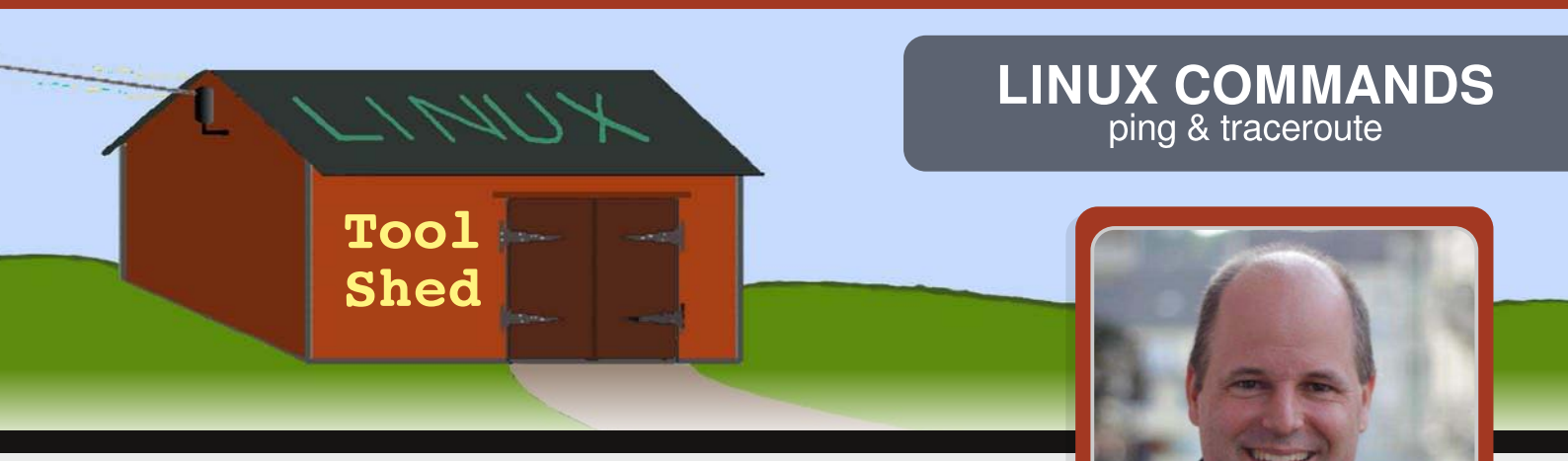
When: Saturday 22nd March 2014, 10.30am to 4.30pm
Where: @Bristol Science Centre, Anchor Road, Harbourside, Bristol, BS1 5DB, UK

A series of community technology events aimed at children (7-17), parents and teachers. This is an introduction to 'making' in the digital word. http://eventbrite.com/e/10428480867

KS2 - Codebreakers!

When: Saturday 29th March 2014, 10.15am to 12.30pm or 1.00pm to 3.15pm
Where: Camden City Learning Centre, Charrington Street, London, NW1 1RD, UK

For children 8-11 years old. Explore counting in binary to send encoded messages to each other and try to break them. http://eventbrite.co.uk/e/10615736955 and http://eventbrite.co.uk/e/10615779081



Tales from the Linux tool shed:
Give me a ping Vasili. One ping only.

Bernhard Suter

Guest Writer

SKILL LEVEL : BEGINNER

The Linux command line has a rich set of powerful tools. In this article, three commands that allow networking issues to be diagnosed are introducted.

ifconfig

For this tutorial, a Raspberry Pi is assumed to be connected to a home router. Once the Raspberry Pi is connected to the home router, `ifconfig` can be used to check the assigned IP address:

```
pi@raspberrypi ~ $ ifconfig
eth0  Link encap:Ethernet  HWaddr b8:27:eb:f9:a3:3b
      inet addr:192.168.0.18  Bcast:192.168.0.255
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:81 errors:0 dropped:0 overruns:0 frame:0
      TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:10445 (10.2 KiB)  TX bytes:8598 (8.3 KiB)
```

In this case, a Model B Raspberry Pi is connected via an ethernet cable to the router. The name `eth0` is used for the ethernet port on the Raspberry Pi.

ping

Let us picture a situation where our network connection to the internet was working fine yesterday, but suddenly seems to be down today, or at least has become very unreliable. For this example, we will assume that the IP address of our firewall/router is `192.168.0.1` (yours may be different). This will be helpful to check the local area network. We also need an external reference, such as one of the Google DNS servers `8.8.8.8`.

`ping` allows us to test the connectivity and measure delay and packet loss to any host in the network. First things first, let us see if we can still reach the router on our local area network. Open LXTerminal and enter `ping -c3 192.168.0.1`:

```
pi@raspberrypi ~ $ ping -c3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=0.854 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=0.782 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=0.778 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted 3 received 0% packet loss time 2002ms
rtt min/avg/max/mdev = 0.778/0.804/0.854/0.047 ms
```

The `ping` command works by sending probe packets to the destination and waits for them to be sent back. The command sends special echo requests and echo reply messages, as defined by the Internet Control Message Protocol (ICMP). A communication protocol is a well specified convention on how two systems can communicate with each other. In the case of ICMP, this convention is documented as an internet standard in RFC792 (http://tools.ietf.org/html/rfc792).

Since ICMP echo request/reply processing is typically part of the lowest level of networking support in each properly implemented internet host, it is a very reliable way to determine if a host can be reached over the network. However, it is worth noting that a firewall will often prevent `ping` from responding. On most Windows machines, the default firewall setup prevents ICMP echo from even getting to the machine. The default on a Raspberry Pi is that ICMP echo will get through.

The `ping` utility is nearly as old as the internet itself and is named after the eery, sharp, metallic sound of an acoustic sonar probe we might be familiar with from submarine movies, like "The Hunt for Red October".

From the results of the `ping`, we can see that our internet gateway still exists on the network and that we have a router round-trip time of less than a millisecond. Out of three probe packets that we sent, three responses were received and the fluctuation in the response time is quite low.

Instead of using `ping -c 3 192.168.0.1` we could also just use `ping 192.168.0.1`. In this case the program runs forever until interrupted by the user pressing <CTRL>+C or killing the process. This enables us to observe the state of network connectivity over time, for example as we wiggle network cables or plug and unplug devices.

`ping` allows several options. Some particularly interesting ones are:
-c count : only send <count> probes and then stop
-i interval : send a probe approximately every <interval> seconds (default 1 second)
-s size : send probe packets of size <size> (default 64 bytes)
-n : don't try to translate numeric IP addresses into hostnames

Other options can be listed by typing `man ping`.

Since the local area network is working correctly, `ping` can be used to check the external connectivity to the Google DNS servers (IP address `8.8.8.8`) with the command `ping -c 3 8.8.8.8`:

```
pi@raspberrypi ~ $ ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.

--- 8.8.8.8 ping statistics ---
3 packets transmitted 0 received 100% packet loss time 2000ms
```

Having ruled out a problem with the local area network, we now suspect that there might be a problem with the connection from our router to the internet.

There can be cases where the packet loss is somewhere in between 0% and 100%. Irregular packet loss can be caused by flaky network cables, loose connectors and unstable or overloaded gateways in the network. Even with low packet loss, high delays or high variation of delay might degrade the performance of higher level protocols, such as HTTP or SSH.

Like any good internet host, the Linux kernel in the Raspberry Pi contains a responder for ICMP echo requests and we can effectively test our own networking stack and how fast the kernel can process small packets. Enter `ping -c 5 localhost`:

```
pi@raspberrypi ~ $ ping -c 5 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_req=1 ttl=64
time=0.153 ms
64 bytes from localhost (127.0.0.1): icmp_req=2 ttl=64
time=0.155 ms
64 bytes from localhost (127.0.0.1): icmp_req=3 ttl=64
time=0.163 ms
64 bytes from localhost (127.0.0.1): icmp_req=4 ttl=64
time=0.146 ms
64 bytes from localhost (127.0.0.1): icmp_req=5 ttl=64
time=0.201 ms

--- localhost ping statistics ---
5 packets transmitted 5 received 0% packet loss time 4005ms
rtt min/avg/max/mdev = 0.146/0.163/0.201/0.024 ms
```

traceroute

While `ping` is a quick and easy way to determine if we can reach a destination or not, `traceroute` allows us to find out more about each hop our traffic is taking towards a given destination IP address. For example, `traceroute` can be used to find exactly where the traffic is lost between our router

(192.168.0.1) and the Google DNS servers (8.8.8.8) by typing `traceroute 8.8.8.8`:

```
pi@raspberrypi ~ $ traceroute 8.8.8.8
traceroute to 8.8.8.9 (8.8.8.8) 30 hops max 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.885 ms  0.800 ms  0.838 ms
 2  * * *
 3  46.140.4.109 (46.140.4.109)  35.432 ms  35.328 ms
    35.082 ms
 4  84.116.200.241 (84.116.200.241)  34.854 ms  34.642 ms
    34.429 ms
 5  ch-zrh01b-ra1-ae-1.aorta.net (84.116.134.142) 33.977 ms
    33.460 ms  33.244 ms
 6  * * *
 7  * * *
 8  * * *
```

The command prints out each hop and the time taken for the ICMP message. If the time taken exceeds the timeout limit, then three stars are printed instead. This can signal a slow network link, or a connection problem.

There appears to be a problem a few hops away from our internet connection itself. Then a few moments later, the service is restored and we can again successfully reach the destination.

```
pi@raspberrypi ~ $ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8) 30 hops max 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.965 ms  0.906 ms  1.072 ms
 2  * * *
 3  46.140.4.109 (46.140.4.109)  11.581 ms  17.977 ms
    17.604 ms
 4  84.116.200.241 (84.116.200.241)  20.844 ms  20.765 ms
    20.367 ms
 5  ch-zrh01b-ra1-ae-1.aorta.net (84.116.134.142) 15.715 ms
    15.642 ms  15.240 ms
 6  74.125.49.101 (74.125.49.101)  20.088 ms  13.548 ms
    72.14.212.210 (72.14.212.210)  13.250 ms
 7  66.249.94.52 (66.249.94.52)  18.371 ms
    72.14.232.120 (72.14.232.120)  21.770 ms
    66.249.94.52 (66.249.94.52)  20.267 ms
 8  209.85.251.248 (209.85.251.248)  20.250 ms
    209.85.251.180 (209.85.251.180)  20.315 ms
    209.85.251.248 (209.85.251.248)  19.785 ms
 9  209.85.254.118 (209.85.254.118)  20.204 ms
    209.85.254.112 (209.85.254.112)  31.628 ms
    209.85.254.116 (209.85.254.116)  19.167 ms
10  * * *
11  google-public-dns-a.google.com (8.8.8.8)  19.840 ms
    19.633 ms  19.524 ms
```

The `traceroute` command shows addresses and hostnames of all the router nodes that packets are going through from our Raspberry Pi to the destination. For this, `traceroute` takes advantage of another ICMP feature, the time-to-live expiry message. All packets that are sent through the internet have a maximum number of times that they can be passed on by routers. This value is decremented at each hop, to prevent packets from going around forever if they can't find their destination point. When a packet is discarded in the network, the router sends out an ICMP message to alert the sender that the packet has been discarded.

In order to discover a network path, `traceroute` sends out a series of packets (default of three) with a time-to-live limited to one, just to see who will send back an ICMP time exceeded message. The command then repeats this process with increasing time-to-live values until it reaches the destination.

There are many more useful commands that can be used to examine the state of the network or test its performance. However, the main advantage of tools like `ping` and `traceroute` is that they work directly with support deep in the operating system kernel. This can sometimes mean that a computer will still respond to `ping` requests, even if it otherwise appears to be completely stuck or has no networking applications running.

When network applications like SSH or web browsing are not working properly, tools like `ping` and `traceroute` are great to figure out whether there is a low-level networking problem, or the problem is maybe with the application itself.

Next time

In a future episode, we will take a closer look at layers of networking support in the Linux kernel and some tools to look at them.

Until then, can you find out where to look to find the address of your internet gateway? Hint: have a look at the `netstat` command.

Expand your Pi

Stackable Raspberry Pi expansion boards and accessories

ADC-DAC Pi

2x 12 bit analogue to digital channels and 2x 12 bit digital to analogue channels.

IO Pi

32 digital input/output channels for your Raspberry Pi. Stack up to four IO Pi boards to give you 128 I/O channels.

RTC Pi

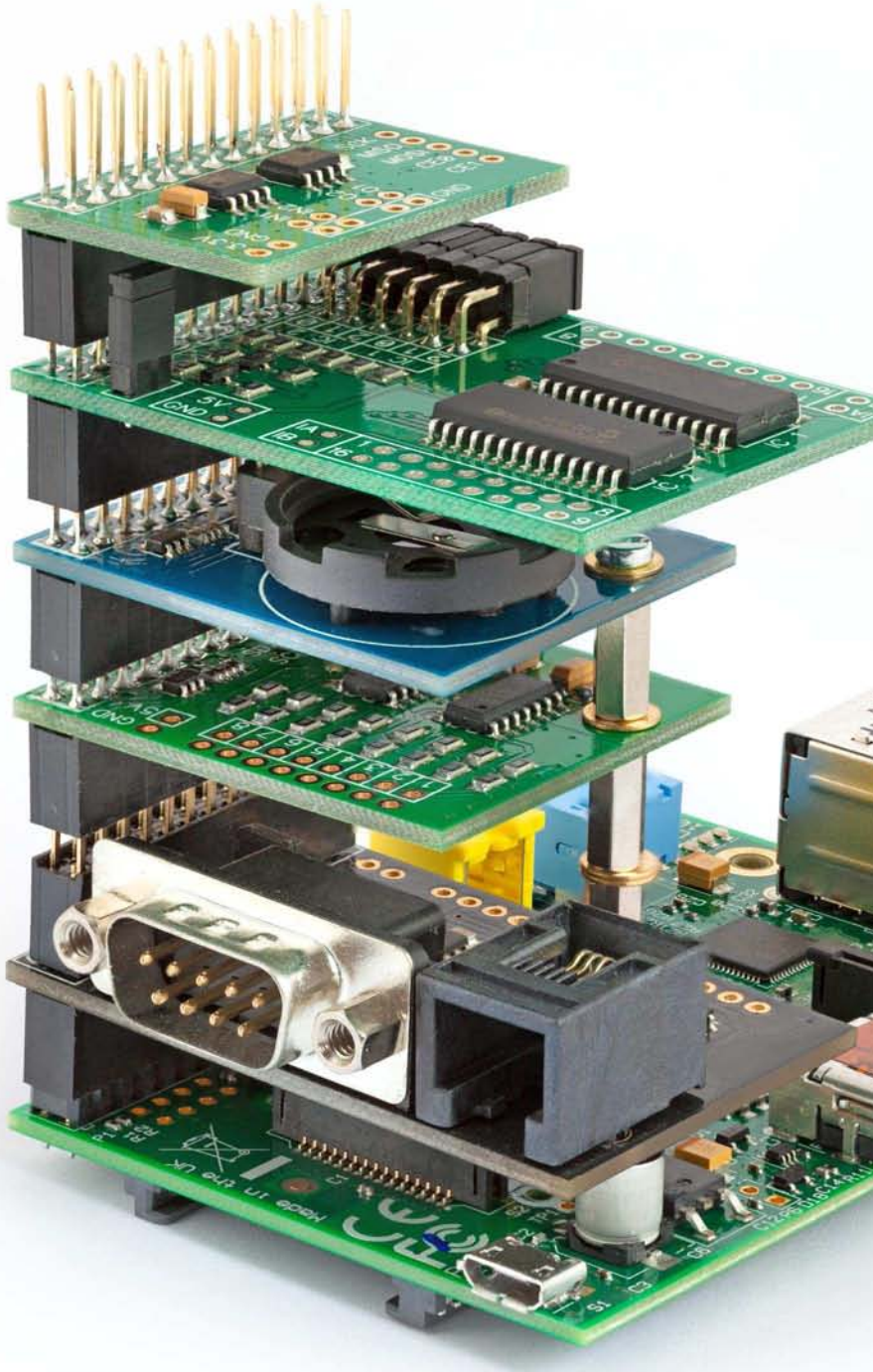
Real-time clock with battery backup and 5V I²C level converter for adding external 5V I²C devices to your Raspberry Pi.

ADC Pi

8 channel analogue to digital converter. I²C address selection allows you to add up to 32 analogue channels to your Raspberry Pi.

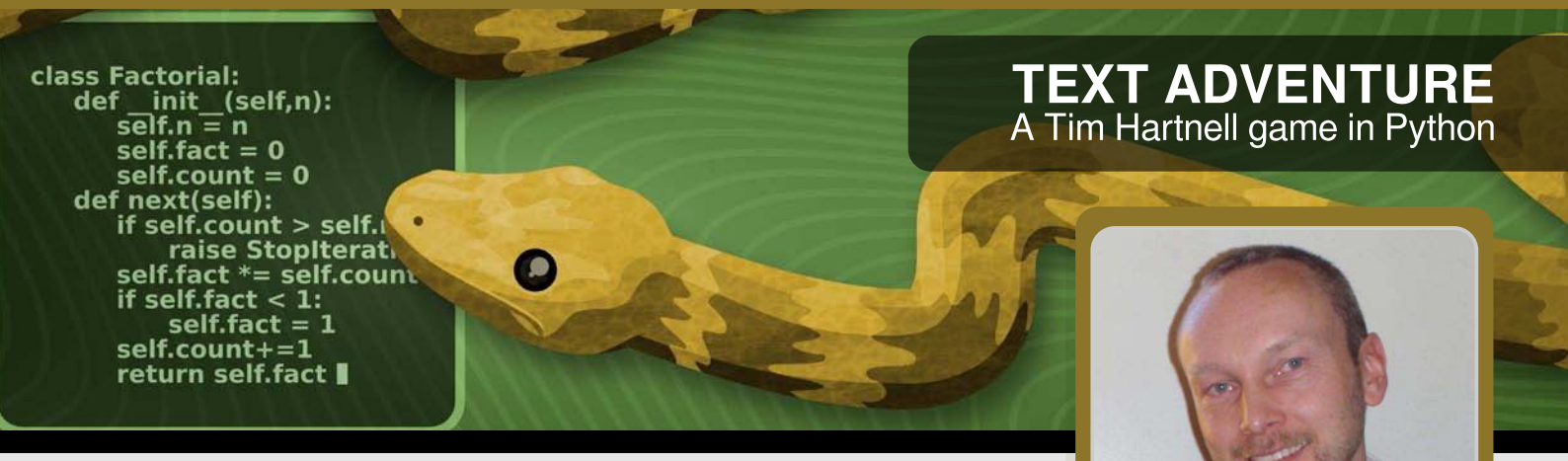
Com Pi

RS232 and 1-Wire® expansion board adds a serial port to your Raspberry Pi. Ideal for the Model A to enable headless communication.



AB electronics UK

www.abelectronics.co.uk



TEXT ADVENTURE
A Tim Hartnell game in Python

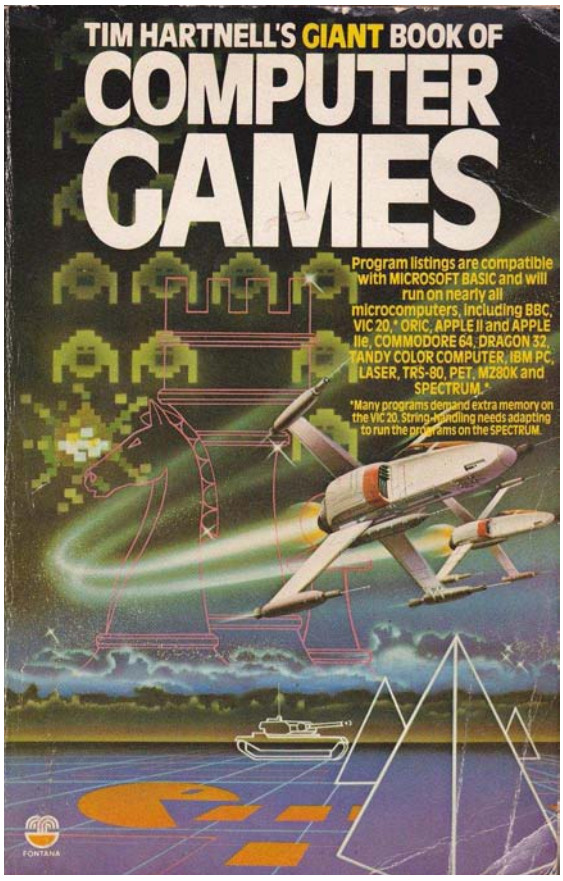
Stronghold of the Dwarven Lords

Martin Hodgson

Guest Writer

SKILL LEVEL : BEGINNER

Introduction



In the early 1980s, as early home computers boomed in popularity, a little programming knowledge was essential. My first computer was a Sinclair ZX81 with 1KB of memory. I used to dream of being able to afford a 16KB RAM expansion! To make the ZX81 do anything at all, you had to know a few BASIC commands. Thankfully, the computer's manual included a good introduction to BASIC programming. For those interested in more advanced programming, magazines like ZX Computing and Creative Computing included listings of BASIC programs. These were mostly games that could be typed in, played, saved and hacked to your hearts content. It was great fun and a fantastic way to learn how to program.

During the early '80s, the late Tim Hartnell was a prolific author of BASIC games and published many articles and over 30 books. Second-hand copies of his books are still readily available online and a few PDFs can be found on vintage computer fan websites.

Stronghold of the Dwarven Lords

Stronghold of The Dwarven Lords is a simple adventure game, from the 1984 book "Tim Hartnell's Giant Book of Computer Games". In my Python translation I try to stick as closely as possible to Tim's original BASIC version. The Python that follows is written in Python 3. Start by opening the Python IDLE3 shell., then press Ctrl+N to open a new window. I suggest typing the program in manually, just like we would have done 30 years ago. Try to work out how it works as you go along.

```
# STRONGHOLD OF THE DWARVEN LORDS v2.1
# Martin Hodgson - November 2013
# Translated from Tim Hartnell's original BASIC version...
# ...with a couple of updates. Now you can't walk through walls!

import random

# VARIABLES, LISTS
data1 = [2,2,2,3,2,4,2,5,2,6,2,7]
data2 = [3,7,4,7,5,7,5,6,5,5,5,4,5,3,6,3]
data3 = [7,3,7,4,7,5,7,6,7,7,7,8,7,9,9,8]
data4 = [9,9,10,8,10,7,10,6,10,5,10,4,8,8]
data5 = [10,3,11,3,12,3,13,3,14,3,14,2,7,10]
data6 = [6,10,5,10,4,10,3,10,2,10,2,11,2,12]
data7 = [2,13,2,14,6,11,6,12,6,13,6,14,7,12]
data8 = [14,12,8,12,8,14,9,12,9,13,9,14,10,12]
data9 = [11,9,11,10,11,11,11,12,12,9,13,9,13,10]
data10 = [13,11,13,12,13,13,13,14,14,14]
data = data1 + data2 + data3 + data4 + data5 + data6 + data7 + data8 + data9 + data10

# FUNCTIONS
def new_game(): # GOSUB 640 in original BASIC version
  global a, b, z, y, x, s, m, a, e, d
  print ("================================================================\n")
  input ("STRONGHOLD OF THE DWARVEN LORDS\nNew Game - Press Enter...")
  # Item zero and the zero at the beginning of each sub-list will be ignored...
  # ... as the BASIC program uses indices 1-15
  a = [[0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]]
  b = random.randint(1, 3)
  z, y = 14, 14
  if b == 2:
    y = 2
  if b == 3:
    z = 2
  x, s = 1, 2
  for b in (range(1, 16)):
    for c in (range(1, 16)):
      a[b].append(x)
      if random.randint(1, 10) > 8:
        a[b][c] = s
      if c<2 or c>14 or b<2 or b>14:
        a[b][c] = x
  d, e = 2, 2
  for f in (range(0, 136, 2)):
    b = data[f]
    c = data[f+1]
    a[b][c] = s
  a[z][y] = s # Makes sure the gold isn't in a wall
  m = -15
  return

def show_map(): # GOSUB 480 'help' in original BASIC version
  global b, m
  print ("\n================================================================\n")
  print ("North")
  b = 15
  while b > 0:
    strng = ""
    for c in (range(1, 16)):
      if a[b][c] == x:
        strng += ("#")
      elif b == d and c == e:
        strng += "*"
      elif a[b][c] == s:
        strng += " "
    print (strng)
```

```
    b -= 1
  print ("South")
  m += 15
  a[d][e] = s
  # Here I've omitted two lines from the BASIC version:
  # 600 FOR J = 1 TO 2000:NEXT J - Makes the prgram pause.
  # 610 CLS - Clear screen. Not possible in Python Shell?

def move(): # Lines 50 to 410 - Main game script from BASIC version
  global m, d, e
  m += 1
  print ("\n================================================================\n")
  print ("STEP NUMBER", m)
  if a[d+1][e] == s:
    print ("NORTH: OPEN")
  elif a[d+1][e] == x:
    print("NORTH: WALL")
  if a[d-1][e] == s:
    print ("SOUTH: OPEN")
  elif a[d-1][e] == x:
    print("SOUTH: WALL")
  if a[d][e+1] == s:
    print ("EAST: OPEN")
  elif a[d][e+1] == x:
    print("EAST: WALL")
  if a[d][e-1] == s:
    print ("WEST: OPEN")
  elif a[d][e-1] == x:
    print("WEST: WALL")
  print ("THE DWARVEN SOURCE BEAM READS:", (100 * abs(z-d) + 10 * abs(y-e)))
  print ("Which direction do you want to move...")
  a_string = input ("N - north, S - south, E - east, W - west, H - help ? ")
  a_string = a_string.upper() # Convert lowercase to upper case
  if a_string == "H":
    show_map()
  elif a_string == "N":
    d += 1
  elif a_string == "S":
    d -= 1
  elif a_string == "E":
    e += 1
  elif a_string == "W":
    e -= 1
  else:
    print("\nPardon? I don't understand...") # Inform the player that the command isn't recognised
  if z == d and y == e:
    win()
  if a[d][e] == x: # In the original you could walk through walls... Now you can't!
    print("\nOuch! You just walked into a wall...")
    if a_string == "N":
      d -= 1
    elif a_string == "S":
      d += 1
    elif a_string == "E":
      e -= 1
    elif a_string == "W":
      e += 1
  move()

def win():
  print ("\nYou found the Dwarven riches in just", m, "steps!\n")
  show_map()
  # This feature has been added - The original version would just END.
  new_game()
  show_map()
  move()
```

```
# MAIN PROGRAM
new_game()
show_map()
move()
```

How to play

First, read the following introduction by Tim Hartnell, whose prose revealed his great enthusiasm for game programming...

STRONGHOLD OF THE DWARVEN LORDS

Deep beneath the earth you go, far into the Dwarven Heartland. Danger is on every side as you descend, but your greed draws you on. Searching through the dusty stacks of uncatalogued manuscripts in room 546B of the British Museum, you came across a faded and almost illegible map to a Dwarven hoard of gold. Since that day you have been obsessed with the idea of finding it.

As you go down into the labyrinth you realise that the Dwarven Lords, who secreted the gold here 7389 years ago, have long since become extinct. So the main danger you face is from the layout of the cavern system itself, rather than from Guards of the Stronghold.

In STRONGHOLD OF THE DWARVEN LORDS you are in a cavern which holds the gold. Each time you play this game, the gold can be in one of three places. The only information you get as to your progress is information provided by the Dwarven Source Beam which you found as you made your way into the cave system. This gives you feedback after each move as to the location of the gold, but you need to learn how to interpret the information it gives you before you'll be able to make much use of it.

When the game starts you are shown a map of the cavern system with your position indicated as an asterisk *. Your aim is to find the hidden gold with the minimum number of steps. Each turn you are given a brief description of your surroundings and must choose the direction to take next. Once the map scrolls off the screen you are only allowed to look at it again by choosing H for help. However, each time you ask for help you are penalised 15 steps.

How the program works

For those with a basic understanding of Python, you will see that the main parts of the program are structured as follows:

- At the beginning there are a few `data` lists of numbers that are combined together to make one big list.
- There are the functions: `new_game()`, `show_map()`, `move()` and `win()`.
- Then finally there is the main program, which calls the functions in order to initiate the first game.

Tim's original variable names are not very descriptive. For me this added to the fun of working out how the program works. My code only uses basic Python constructs. The `new_game()` function is probably the most complicated part of the program; it creates a list of 15 x 15-item 'sub-lists' that represents the map. This is done by first creating a grid that is designated as 'all wall', but with a few random spaces. Then more spaces are added according to the data list. Hopefully, this will become clearer as you type in the program and play the game.

Have fun finding Tim's gold!

Feedback & Question Time

The MagPi is doing an excellent job at providing an abundance of material in various forms to us information hungry Raspberry Pi users throughout the world. The quality and quantity of articles in each issue is going from strength to strength. For all the effort that goes into producing each issue, I offer you an extremely grateful thank you.

I'm a relative newcomer to the magazine (and the Raspberry Pi in general). It was last November, when browsing through Kickstarter, that I came across the Kano kit and my interest was aroused. I had vaguely heard of the Raspberry Pi at that point but, to be honest, hadn't given it too much thought until then. It was at that time that I began to pay a little more attention to it all! The more I explored, the more I liked what I discovered. I couldn't see myself waiting until the middle of 2014 (when the Kano kit is due for dispatch to backers of the project), so I was treated to a nice Raspberry Pi setup at the end of last year.

To put you in the picture a little more, I'm of the age that remembers (and used) the Sinclair ZX80, ZX81, ZX Spectrum and so forth... Ah, those were the days! So you can understand that the more I investigated the Raspberry Pi, the more I realised the potential of this 'little beastie'. The old fever from the early 1980s was back and I was leaping around like a kid at Christmas – jumping around here, there and everywhere. There are so many roads to go down, but I've settled down a bit now and I'm currently learning the ins and outs of Python and Linux programming. I'm knitting this in with a brand-new topic as far as I'm concerned, namely electronics. Oh!... little flashing LEDs and so forth... how pretty. I'm afraid to say that the greatest majority of the blame for this obsession lies squarely at your door. I was living life in blissful ignorance until I came across your magazine ;-)

As I mentioned at the start, The MagPi offers a boat-load of material to experiment with and I take my hat off to the team for making this possible to the community at large. Your time and effort is greatly appreciated. You all deserve a big round of applause (clap, clap, clap, clap, clap...). I foresee 2014 as 'the year of The MagPi'.

Ronnie James

If you are interested in writing for The MagPi, would like to request an article or would like to join the team involved in the production, please get in touch by emailing the editor at:

editor@themagpi.com