ISSUE 09 - FEB 2013

Get printed copies of issues 1-8 plus binder at themagpi.com

The MagPi™

A Magazine for Raspberry Pi Users

Meet Ladyada

An interview with the Founder of Adafruit Industries

This Issue...

- WebIOPi Framework
- Backup SD cards
- Vala & LedBorg
- Scratch GPIO
- RISCOS

ISSN 2051-9990

The MagPi™

http://www.themagpi.com

The MagPi

Welcome to Issue Nine,

Many thanks to all of you who generously supported our kickstarter project. The kickstarter raised over four times the target amount, with 616 backers. After a small delay, we are gearing up for printing. The folder design for Volume 1 (Issues 1 to 8) has been finalised. If you missed out on the kickstarter, then Volume 1 can still be ordered from our online shop www.themagpi.com/shop/

This month we are very pleased to present an interview with engineer and founder of Adafruit Industries, Limor Fried. She has been an encouragement to many, providing tutorials, answering questions and supplying parts to build all sorts of interesting projects.

This issue also contains hardware articles, setup instructions for RISCOS and Arch Linux operating systems, and our regular programming articles on Scratch, Python and C.

Ash Stone

Chief Editor of The MagPi

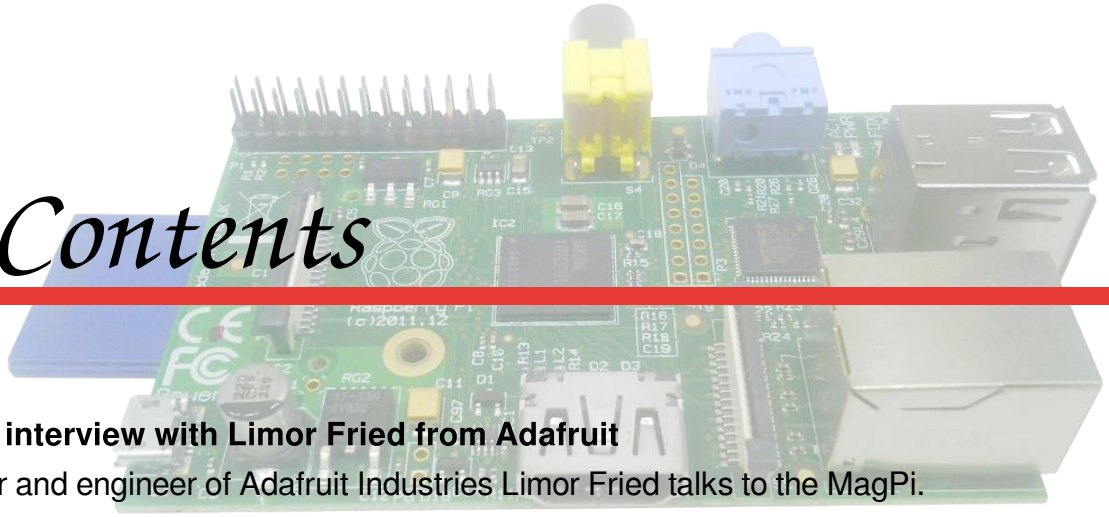
MagPi team

Ash Stone - *Chief Editor / Administrator*
Tim 'meltwater' Cox - *Writer / Page Designs / Admin.*
Chris 'tzj' Stagg - *Writer / Photographer / Page Designs*
Colin Deady - *Writer / Page Designs*
Jason 'Jaseman' Davies - *Website / Page Designs*
Matt '0the0judge0' - *Website / Administrator*
Aaron Shaw - *Writer / Page Designs / Graphics*
Ian McAlpine - *Writer / Page Designs / Graphics*
Sam Marshall - *Page Designs / Graphics*
W. H. Bell - *Writer / Page Designs*
Bryan Butler - *Page Designs / Graphics*
Colin Norris - *Graphics*
Mark Robson - *Proof-reading*
Alex Baker - *Proof-reading*
Richard Wenner - *Proof-reading*
Steve Drew - *Proof-reading*
Chosp - *Proof-reading*
Benjamin Donald-Wilson - *Proof-reading*
Mike Richards - *Proof-reading*

Guest writers

Alex Eames
Eric PTAK
Norman Dunbar
Pete Nowosad
Ross Taylor
Simon Walters

Cover: Limor Fried, engineer and founder of Adafruit Industries.



Contents

adafruit INTERVIEW

She is an open source hardware advocate, founder of Adafruit and was voted "Entrepreneur of 2012". Who is Limor Fried?

[MagPi] First our congratulations on being awarded Entrepreneur magazine's "Entrepreneur of 2012" last month. Do you think this is the first sign of mainstream acceptance of "hacking", in its true definition and the "maker" movement in general?

[Limor] Thank you so much for the kind words. The Raspberry Pi community deserves a lot of thanks as well, all of the voting was via the internet and the Pi community really rallied for us! I believe the maker movement is past the "is this a real thing?" stage. About 6 years ago I was invited to a conference about the new maker movement that had just started to happen and a very large company made a point to say Adafruit was not a real company. It's been a challenge every day to prove a great business can support a great cause like open-source. Being awarded Entrepreneur magazine's "Entrepreneur of the year" means there are less barriers for someone starting out now. They don't need to hear something is not possible or not real, they can see there are unlimited opportunities for making and sharing - and running a successful business.

[MagPi] We're getting a little ahead of ourselves. Let's take a step back. You are the founder and engineer of Adafruit, the New York based company that you formed in 2005 after you graduated from the Massachusetts Institute of Technology (MIT) with a master's degree in Electrical Engineering and Computer Science. What inspired you to start your own company rather than "cut your teeth" with an industry employer?

[Limor] Running your own company isn't for everyone, I wasn't even sure it was for me at first, but the freedom and flexibility to pursue whatever you want and work on the important things is seductive and rewarding. There's a ton of risk of course, but the biggest risk is regret later if you don't at least try. There's never been a better time to run a company that celebrates smart people, smart communities and learning. The demand for efforts like the Raspberry Pi has totaly changed Adafruit. Anyone can learn to design electronics, write code and have multiple ways to get the products in the hands of customers. One of the things about running a company is you can take on some projects that at first do not seem to have impact on the bottom line, but you can take the risk. Many of the projects we do at Adafruit would never be approved by a big company solely focused on a few products. We have over 1,200 products and some of them are purely experimental.

[MagPi] Your nickname is "Ladyada" which I am assuming has some relation to Lady Ada Lovelace, the world's first computer programmer? Of course the link to Adafruit is more obvious, but what was the inspiration

behind the name? Maybe you had a premonition that you would be working with raspberries one day?

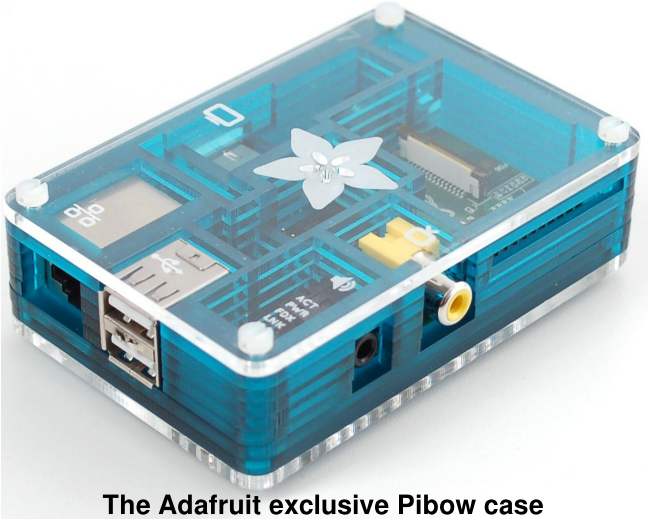
[Limor] When I was younger all I did was play around with Linux, installing it on anything I could find and exploring all the things that made it work. In my hacker days, actually I'm still in those days now :) ..., my nickname was Ada. I was always programming, building, reverse engineering so the Ladyada name has stuck with me from the start. Looking back at all the Linux hacking it seems like I was in training to work on the Raspberry Pi. For the younger folks out there who like spending time hacking away, it really can end up being a fantastic adventure and a career!

[MagPi] We have all heard of open source software and it is used every day by millions of people, including 1 million plus Raspberry Pi users. But you are heavily involved in the open source hardware community. Indeed all Adafruit products are made available as open source hardware with free download of schematics, PCB layouts, firmware and software. Arduino is a popular example of open source hardware but can you tell us more about what you think is the future of open source hardware?

[Limor] I like to skip to the end of the story, all hardware is copyable. There are trademarks for logos and names and patents for some things, but if it's made out of physical bits and it's interesting, someone is going to copy it. So I've always worked back from that, if someone is going to copy something of mine I should do my best to make it educational, fun and help society. In a world where companies like Apple and Samsung are suing each other it's pretty clear that progress stops when you think you can stop copying. One way to look at it is recipes, we can all make any dish we like at home, but we go to restaurants for an experience. That's how I look at hardware, you're not just buying the physical bits from Adafruit, you're getting the service, support and community of makers. In the future every hardware company will need to be a cause and a business, Raspberry Pi and Arduino are great examples.

[MagPi] What got you interested in the Raspberry Pi? Looking at www.adafruit.com I can see that the Raspberry Pi section is one of the largest.

[Limor] At Adafruit we have a big goal and mission; to teach kids programming and making... and it's actually teach everyone, not just kids. We think everyone should be able to use a low-cost educational computer to learn electronics and of course learn a computer language. We struggled with how we would be able to start this endeavor and that's when the Raspberry Pi was announced. It became so popular so quickly that it really kept us on our toes meeting demand. Every single thing we design or curate in the store is tested by me. For the Raspberry Pi we knew it would be important to have the best educational resources in addition to the best support. Six months later, the Pi section is one of our largest and the Pi tutorials are the most viewed on the Adafruit Learning System http://learn.adafruit.com/category/raspberry-pi.



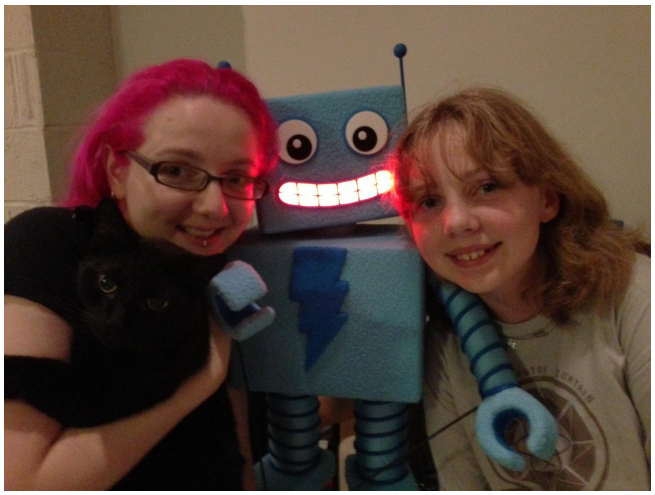
The Adafruit exclusive Pibow case

But wait, there's more! We also knew we'd need a great web-based way to teach, so we've invested a lot of time and resources in building our own integrated development environment (IDE), complete with step debugging and visualizer. The Raspberry Pi WebIDE is by far the easiest way to run code on your Raspberry Pi. Just connect your Pi to your local network. Then log on to the WebIDE in your web browser to edit Python, Ruby, JavaScript, or anything and easily send it over to your Pi. Also, your code will be versioned in a local git repository, and pushed remotely out to bitbucket so you can access it from anywhere, and any time. Watch the video at http://youtu.be/8NoiBBgaKCI

Continued over page...

Lastly, there is our Pi product line up. We take the best of Adafruit design and apply it to helping people get the most out of their Pi. Everything is fully documented and supported for makers of all ages... and we are just getting warmed up as they say!

[MagPi] Women have always been sparsely represented in the technology industry and unfortunately this is likely to remain while social and educational bias continue to influence young girls. Of the fifteen finalists for “Entrepreneur of 2012”, you were the only female. But the Arduino introduced electronics and interfacing to artists and creators while the Raspberry Pi is empowering educationalists to engage both boys and girls in fun and interesting ways. My two young daughters each have their own Raspberry Pi. They create their own programs using Scratch or copy Scratch programs that we regularly print in The MagPi and then modify them. You are a great role model for young women today, and I recently discovered 11 year old "super awesome Sylvia" (www.sylviashow.com), but what more can be done to encourage girls to study engineering topics?



Limor, Mosfet, AdaBot and Sylvia

[Limor] One of my favorite quotes is "We are what we celebrate" by Dean Kamen. There are many women in tech but they're not celebrated as much or in the same way as their male counterparts. Journalists and conference organizers can put a well deserved spotlight on many amazing women; they are out there. If a young girl growing up does not see women in a particular field, or is not able to say "I want to do that, I want to be like her" then we'll never see more women in that field. It's really up to each of us to find people in the community and elevate them. There's been a lot of progress, and there's still a lot of work to be done. With a $35 Pi on the market all of us can give the gift of engineering to a young kid. You never know what will spark someone on a path of engineering, but I think we have more opportunities, in fact I think we have a million more than we did last year.

[MagPi] Our younger readers might be interested in Mosfet. What can you tell us about Mosfet? Maybe Mosfet and Liz’s Mooncake should Skype?

[Limor] Mosfet is my black cat of about 10 years now. He's grown up at Adafruit and he's on our weekly live engineering show "Ask an Engineer" at http://www.adafruit.com/ask. There is something special about cats and engineers, we are not sure what it is exactly, but the best engineers we know have cats - so much so we started "Cats of Engineering" at http://www.adafruit.com/catsofengineering.

In January, MAKE Magazine had their second annual worldwide Maker meet up over Google+ Hangouts. Eben & Liz's cat was able to see Mosfet as we all talked about the 1 million milestone for the Pi and beyond. I'm not sure what will happen with the internet 10, 50 and 100 years from now, but as far as I can tell, it was meant for cats to communicate with each other :)

[MagPi] The Adafruit website has a very broad range of intriguing products. I’m particularly interested in the 2x16 negative RGB display plate for the Raspberry Pi and the MintyBoost looks like it could easily power a Model A Raspberry Pi. Then there’s the whole range of FLORA wearable electronics too. What products currently have the greatest interest among your customers and which products excite you currently?

[Limor] The two big product lines are anything Raspberry Pi and our new wearable platform FLORA. People want to learn how to program, make some cool projects and also jump in to the new frontier of wearable electronics. Right now the products that are really exciting to me are the ones that inspire the community to do cool projects and share. I like the Pi based Wi-Fi radios and I really

enjoy watching the projects come in that use FLORA in ways we didn't think about. I'm working on a Pi game controller, a touch screen and for FLORA an even smaller version for super-tiny wearable applications.

[MagPi] I'm constantly amazed by the projects that folks create with their Raspberry Pi. I either read about them at www.raspberrypi.org or I see them on your "Show & Tell" show or we publish them in The MagPi. What projects have you seen built using products that you sell that caused a "Wow" moment?

[Limor] One of my favorite projects is one that's in progress. A very prolific maker named Kris in our community is making a Raspberry Pi based snow-blower robot. He's printing out all the robotic parts, controlling servos, vision and more, all with a Pi. It's a functional robot that is built on open-source and we're all learning and sharing together. To think someone could make such a complex project at home in one's spare time, and at a reasonable cost is just "Wow!". [Ed: I saw Kris on the 19 January episode of "Show & Tell". His work is impressive.]

[MagPi] You talked earlier about the Adafruit Learning System, which contains over 100 tutorials and videos including a series for the Raspberry Pi. I personally had no idea just how extensive this educational resource actually is and I will be returning to learn more plus also to download the Raspberry Pi WebIDE. What are your plans to further develop the Adafruit Learning System?

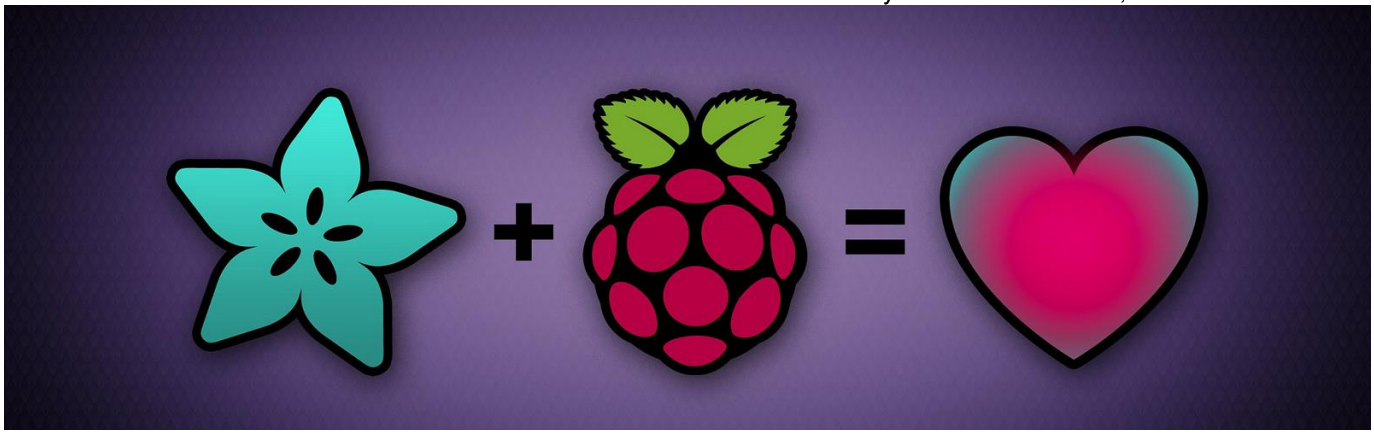
[Limor] Thank you, the team has done an amazing job making the best learning system online. Justin, Tyler, Daigo and my team completely took tutorials to the next level for us. The Adafruit Learning System is just one part of the giant task of education for everyone. We're starting with programming and engineering, but we're not going to stop there. We think there's unlimited potential to maximize all of our potential through sharing. We will be adding our badging system, more user features and more ways to seamlessly use and share knowledge. We have PDFs you can print and share, the system works great on devices and we'll be adding more video and more interactivity (sensor logging and more) very soon.

[MagPi] I keep reading that 2013 is the year of the entrepreneur. Maybe this is influenced by the global economy but we certainly live in interesting times. Products such as the Raspberry Pi and Arduino are empowering a new generation and enabling new capabilities plus 3D printers, Hacker Spaces and Maker Faires are all making technology more accessible to everyone in a spirit of community, education and sharing. I won't ask you what is the next big thing(!) but having been a major influencer in this community and watched it evolve over that past 7 years what trends do you see looking forward?

[Limor] You can make any future prediction come true if you're willing to do the hard work, so at Adafruit we try to predict the future by building it. In the next 7 years more people will be able to do anything they can think of, so what does that mean? I think we need to pick some big problems to solve. If a $35 computer can help teach, could it also diagnose and help to heal? Could it help democracy and freedom around the world? Could this type of technology and level of access for all bring us together so we can all move forward? I think so, and that's where I'll be spending my days and nights.

Article by Ian McAlpine

Photos by John De Cristofaro, Adafruit and its staff.



Tutorial: Remote controlled robot cam

WebIOPi is a REST framework which allows you to control Raspberry Pi's GPIO from a browser. It's written in Javascript for the client and in Python for the server.

You can fully customize and easily build your own web app. You can even use all the power of WebIOPi directly in your own Python script and register your functions so you can call them from the web app. WebIOPi also includes some other features like software PWM for all GPIO.

Installation

Installation on the Raspberry Pi is really easy, as it only requires Python. On Raspbian Wheezy, you can use the PiStore to download and install WebIOPi. You can also install it using a terminal or a SSH connection. Check the project page for the latest version, then type:

```
$ wget
http://webiopi.googlecode.com/files/
WebIOPi-0.5.3.tar.gz
$ tar xvzf WebIOPi-0.5.3.tar.gz
$ cd WebIOPi-0.5.3
$ sudo ./setup.sh
```

You should see some compile and install process, to finally get a success output with usage instructions:

```
WebIOPi successfully installed
* To start WebIOPi with python:
sudo python -m webiopi
* To start WebIOPi with python3:
 sudo python3 -m webiopi

* To start WebIOPi at boot:
sudo update-rc.d webiopi defaults
* To start WebIOPi service:
sudo /etc/init.d/webiopi start
* Look in /home/pi/webiopi/examples
for Python library usage examples
```

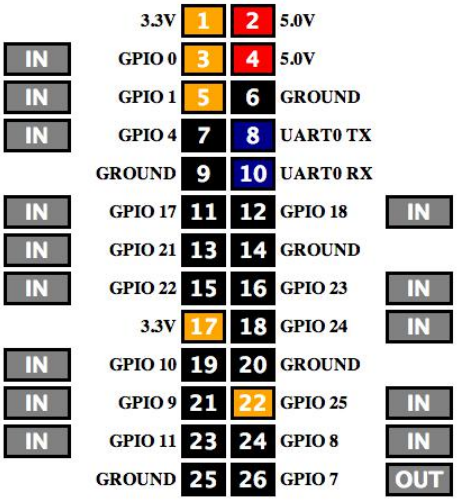
You will have a line for each installed Python version you can use to launch WebIOPi.

It's time to start WebIOPi, for example with Python 2.X:

```
$ sudo python -m webiopi
WebIOPi/Python2/0.5.3   Started   at
http://[IP]:8000/webiopi/
```

First Use

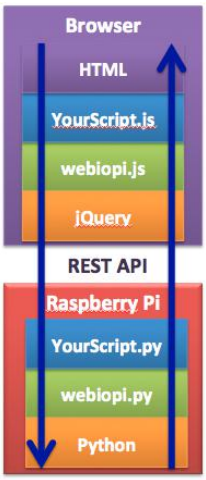
Open a browser from any device of your network, and point it to the given URL: `http://[IP]:8000/webiopi/` or you can use localhost if you are connected to your Pi with a keyboard and a display plugged into it. You will then be asked to log in, **default user is webiopi and password is raspberry.** You should see the default header app:



With the default header app, you can toggle GPIO functions between input and output, and toggle pin states. Just click on the IN/OUT buttons and on each pin to change their state when set as output.

All GPIO can be directly used with the REST API. For instance, to setup GPIO 23 as an output, just make an HTTP POST request on /GPIO/23/function/out then to output a logical 1, make POST on /GPIO/23/value/1. To retrieve states, make HTTP GET on /GPIO/23/function and /GPIO/23/value.

The included Javascript library allows GPIO changes without caring about REST calls.

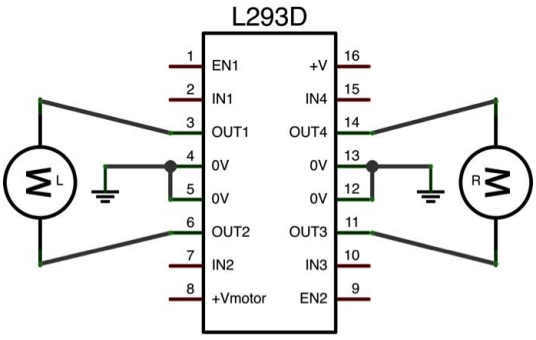


Robot Cam

The next parts of this article are about a robotised webcam you can control from any web browser. You will need:

- Chassis
- Raspberry Pi with WebIOPi
- Operational USB Webcam
- Operational USB WiFi adapter
- L293 H-Bridge
- 2 sets of motors, reducers and wheels
- Battery and power regulation
- Electronic prototyping parts

From the electronic point of view, L293 contains an electronic circuitry similar to the Skutter's H-Bridge from the December MagPi issue. L293 adds an enable input that can be used with a PWM signal to limit the speed. It also has two power inputs, one for the logic (+V=5V), and one that suits the motors (+Vmotor<36V).

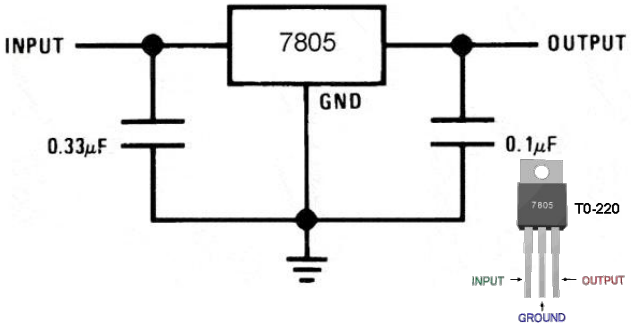


Motor rotation is controlled by IN* and EN*:

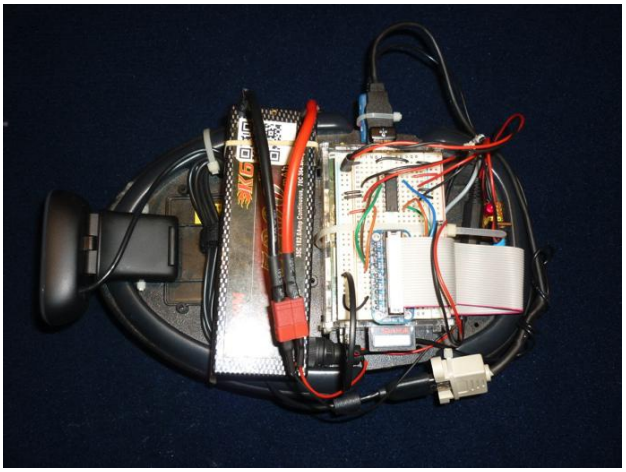
IN1	IN2	EN1	MOTOR L
LOW	LOW	X	STOP
X	X	LOW	STOP
HIGH	LOW	HIGH	FORWARD
LOW	HIGH	HIGH	BACKWARD

IN3	IN4	EN2	MOTOR R
LOW	LOW	X	STOP
X	X	LOW	STOP
HIGH	LOW	HIGH	FORWARD
LOW	HIGH	HIGH	BACKWARD

We can connect +V to Pi's +5V and IN*/EN* to GPIOs. +Vmotor will be connected to the battery or a dedicated regulator. You will need at least one 5V regulator to power the Pi with a battery through the micro USB plug. You can use a 7805 with two capacitors to do that:



With an input voltage higher than 7V, you will have a 5V regulated output voltage for the Pi.



To control the H-Bridge with WebIOPi and create an interface, we have to write a few Python lines for the server side, and some Javascript for the client.

Writing the Python Script

Start by creating a new Python file with your favourite editor. You will have to import webiopi then instantiate a Server.

One parameter is required: the port the server has to bind to. You can also change the

Continued over page...

default login and password. The server will start and run in its own thread until the end of the script. We need to add a loop to keep the server running. We use the webiopi.runLoop() function for that. It will sleep the main thread until CTRL-C is pressed. We can also pass a function to the loop.

```
import webiopi

# Instantiate WebIOPi server
# It starts immediately
server = webiopi.Server(
            port=8000,
            login="cambot",
            password="cambot")

# Run the default loop
webiopi.runLoop()

# Cleanly stop the server
server.stop()
```

The previous script simply starts the WebIOPi server. We can use the default web app to interact with GPIOs, the REST API or the Javascript library. We could directly control H-Bridge lines from Javascript, but we'll add go_forward and stop REST macros to decrease latency.

To continue, we need a GPIO library. We can use RPi.GPIO or the integrated GPIO library, which is a fork from RPi.GPIO. An integrated library removes many sanity checks to give full access from the server and to give more functionality.

Right after the import section:

```
# Integrated GPIO lib
GPIO = webiopi.GPIO
```

We add variables to ease H-Bridge control:

```
# Left motor GPIOs
L1=9  # L293 IN1 on GPIO 9
L2=10 # L293 IN2 on GPIO 10
LS=11 # L293 EN1 on GPIO 11

# Right motor GPIOs
R1=23 # L293 IN3 on GPIO 23
R2=24 # L293 IN4 on GPIO 24
RS=25 # L293 EN2 on GPIO 25
```

Before the Server call, we write functions for both left and right motors then wrap them into go_forward and stop macros:

```
# Left motor functions
def left_stop():
    GPIO.output(L1, GPIO.LOW)
    GPIO.output(L2, GPIO.LOW)

def left_forward():
    GPIO.output(L1, GPIO.HIGH)
    GPIO.output(L2, GPIO.LOW)

# Right motor functions
def right_stop():
    GPIO.output(R1, GPIO.LOW)
    GPIO.output(R2, GPIO.LOW)

def right_forward():
    GPIO.output(R1, GPIO.HIGH)
    GPIO.output(R2, GPIO.LOW)

# Set the motors speed
def set_speed(speed):
    GPIO.pulseRatio(LS, speed)
    GPIO.pulseRatio(RS, speed)

# Movement functions
def go_forward():
    left_forward()
    right_forward()

def stop():
    left_stop()
    right_stop()
```

Then, and always before the Server call, we initialise GPIO:

```
# Setup GPIOs
GPIO.setFunction(LS, GPIO.PWM)
GPIO.setFunction(L1, GPIO.OUT)
GPIO.setFunction(L2, GPIO.OUT)

GPIO.setFunction(RS, GPIO.PWM)
GPIO.setFunction(R1, GPIO.OUT)
GPIO.setFunction(R2, GPIO.OUT)

set_speed(0.5)
stop()
```

Finally, we have to register macros on the server to add it to the REST API. This will allow them to be called from the web app:

```
server.addMacro(go_forward)
server.addMacro(stop)
```

Article by Eric PTAK

This will be continued next time, but if you want to get a head start visit http://files.trouch.com/webiopi/cambot.zip

BACKUP YOUR RASPBERRY PI

Learn how to backup your Raspberry Pi SD card with optional compression plus splitting files to burn to DVD.

In this 2 part article I shall explain how you can easily backup your Raspberry Pi SD card to a different computer, to a USB device, to CDs or DVDs and also how to restore from these. All of this is simple on a Linux computer. Windows users requiring a backup to CDs or DVDs would be better off using a live Linux distribution such as the excellent Linux Mint.

Why Backup?

You should always have one or more backup copies of your SD card. Get into the habit of making a daily backup before it is too late... especially if you frequently edit files or add software.

SD cards are much more robust than hard drives, but they can still easily fail with total loss of data. I have had two SD cards become corrupted and completely lock up my Pi. The only recourse was to pull the power lead out and unfortunately in both cases I was left with an SD card where the "/" partition could not be read, so I had no operating system.

Luckily I had a backup. It was a backup of a 4Gb SD card but I had corrupted my new 8Gb one. Fortunately you can restore a 4Gb image to a bigger SD card and I will explain how to perform this task later in this article.

Windows Users

These instructions are intended for Linux only. For Windows users you can make a complete backup image of your SD card using the same Win32DiskImager program that you used to create your Raspbian SD card. Insert your SD card into an SD card reader in your Windows machine and start Win32DiskImager. Enter a name for your backup image and click on `Read`. When the backup is complete, the file size of the backup image file should be very similar to the size of the SD card.

A Few Caveats

Because Windows is the most common operating system, many vendors supply USB thumb drives and USB hard drives formatted with the FAT32 file system. Note that FAT32 cannot cope with any files that are bigger than 4Gb. If your SD card image is larger than this, you will need to split the file into 4Gb chunks.

You could reformat the USB device with NTFS, but if you never intend to use the device with any Windows computers, then you can easily format it with a Linux file system such as EXT4.

Equally, whether on Linux or Windows, if you are using the 32-bit version then once again a file size limitation will be encountered. The maximum size for a single file on a 32-bit operating system is 4Gb. If your backup file is larger than this you will have to split it into 4Gb chunks. Compressing and splitting backups is covered in detail below.

Getting root Access

The remainder of this article assumes that you have root access to the computer where you will be saving the backup. To get root access you can either prefix every command with `sudo`, or you can run the following command to start a root shell:

```
$ sudo sh
```

Find Your SD Card

Plug your SD card into your Linux computer, not your Raspberry Pi. If your computer has a built-in card slot, you most likely have the card on /dev/mmcblk0. If you have it in a USB adaptor of some kind, it's most likely going to be /dev/sdx where 'x' is the digit representing the highest device of this kind.

```
Disk /dev/mmcblk0: 7948 MB, 7948206080 bytes
4 heads, 16 sectors/track, 242560 cylinders, total 15523840 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000dbfc6

    Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880    15523839     7700480   83  Linux
```

To find out, run the `fdisk -l` command. The tail end of the output from this command can be seen at the top of this page. I can recognise the card by the size, 8Gb (or 7948Mb as listed above) which is the only device or partition that I have of this size. Also shown are the two partitions on the device and seeing that one of them is small and W95 FAT32 (LBA) and the other is Linux, then I am certain that this is my SD card.

In the above, the suffixes 'p1' and 'p2' in the Device Boot column indicate the partitions on the card. The card itself is the device named /dev/mmcblk0 and we need the device name, not the partition names.

Make a Backup

To make a backup switch to your backup directory, where you intend to keep the copy of your SD card, and run the following command, typed all on one line:

```
$ dd if=/dev/mmcblk0 of=Rpi_8gb_backup.
img bs=2M
```

The output from the above command will be similar to the following:

```
3790+0 records in
3790+0 records out
7948206080 bytes (7.9 GB) copied,
1369.42 s, 5.8 MB/s
```

That's it. The whole 8Gb SD card has been (slowly) copied to my computer. How do I know it worked? Type `ls -l -h` and you will see the new file name and size.

Compressing the Backup

Once created and checked, the backup image can be compressed to save space. This is a simple matter of using the `gzip` command:

```
$ gzip -9 Rpi_8gb_backup.img
```

This command attempts to perform maximum compression. This uses a lot of CPU but will result in the smallest output file. You may adjust the trade off between CPU and file sizes by changing the '-9' option to a smaller number. The quickest compression will be obtained with the '-1' option at the expense of the file size being larger.

The use of `gzip` in this manner requires that you have enough space to save both the full size image file and the compressed image for as long as the `gzip` command is running. Once complete, the full sized file will be deleted leaving only a compressed file, named as per the original file name but with an additional '.gz' extension. In this example my compressed image file would be Rpi_8gb_backup.img.gz.

Compression on the Fly

If you only wish to make a backup or if you don't have enough available space to hold both the full sized image and the compressed image, then you may wish to consider compressing on the fly.

Unless otherwise instructed, the `dd` command defaults its output file to be the console. You can use this feature to pipe the output through `gzip` and create a compressed image file in a single step. You will not need as much disc space because the full sized image file is never created, only the smaller compressed one. The command to do this is:

```
$ dd if=/dev/mmcblk0 bs=2M | \
gzip -9 - > Rpi_8gb_backup.img.gz
```

The trailing "\" must be at the very end of the line, just before you press the Return key. It indicates to the shell that the command is not yet complete and more text will follow.

Continued over page...

Because you are redirecting the output from `gzip` to a file, it is your responsibility to supply the file name and the '.gz' extension. You must also include the hyphen after the '-9' in the `gzip` command. This tells `gzip` to write its output to the console which, in this case, has been redirected to a file named Rpi_8gb_backup.img.gz.

Splitting the Backup

As mentioned previously, some file systems cannot cope with files larger than a specific size. FAT32, for example, has a 4Gb limit and 32-bit operating systems can also only cope with 4Gb for each individual file. If you intend to create or copy your backups with these types of systems or devices, then you must split the image files into appropriate sized chunks.

Once again, a pipe comes to the rescue. The `split` utility in Linux is designed to allow a large file to be split into a number of smaller files, each of a given size. If we wish to create a compressed backup of an SD card, ready for burning onto one or more CDs or DVDs, then the following command will backup the card, compress the backup on the fly and then split the compressed image into a number of 2Gb files ready to be burned onto DVDs.

```
$ dd if=/dev/mmcblk0 bs=2M | \
gzip -9 - | \
split --bytes=2G - \
Rpi_8gb_backup.img.gz.part_
```

Remember to press the Return key immediately after typing the "\" on each line.

The `split` command, similar to `gzip`, requires a hyphen as the input file name. In the above there is no input file name as we are reading from a pipe, so the hyphen tells `split` to read the input from the console which, in this case, is the piped output from `gzip`.

The final parameter to `split` defines the root part of the output file names. Each one will be called 'Rpi_8gb_backup.img.gz.part_xx'. The 'xx' part will be 'aa', 'ab', 'ac' and so on.

It is possible that compressing the SD image will make the file small enough to fit within the file system limits of your chosen output device. In this case, you may rename the single part file back to the original name.

Restoring Backup Files

Having the backup image compressed or split into a number of sections means that restoring the backup takes a little more work. Once again the process is relatively simple and involves piping a number of commands together to form a chain of utilities, with the final output going into the `dd` command. This then writes the data back to the SD card.

The simplest case is when the image file has been compressed but not split. To restore this directly to the SD card, without having to create a full sized uncompressed image, type the following command:

```
$ gunzip Rpi_8gb_backup.img.gz -c | \
dd of=/dev/mmcblk0 bs=2M
```

The '-c' in the `gunzip` command above is required to tell `gunzip` to write the decompresssed data to the console. The `dd` command reads its input, unless otherwise specified, from the console so the output from `gunzip` passes straight into `dd` and out to the SD card.

If the backup is split into chunks, then we need to concatenate these together in the correct order and pipe that to the above command pipeline, as follows:

```
$ cat Rpi_8gb_backup.img.gz.part_* | \
gunzip -c | \
dd of=/dev/mmcblk0 bs=2M
```

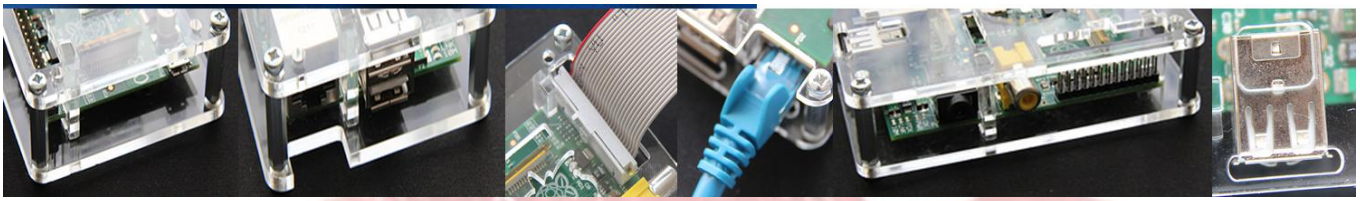
The list of files to be joined back together again must be specified in the correct order. However, using a wild card to the `cat` command causes them to be read in alphbetical order, which is exactly how we want them to be.

Coming in Part 2

In part 2 of this article, I will show you how you can check that the backup was successful and how you can use the backup file as a pseudo hard disc. I will also show how you can modify the files within it - all without needing to restore the image onto an SD card.

Article by Norman Dunbar

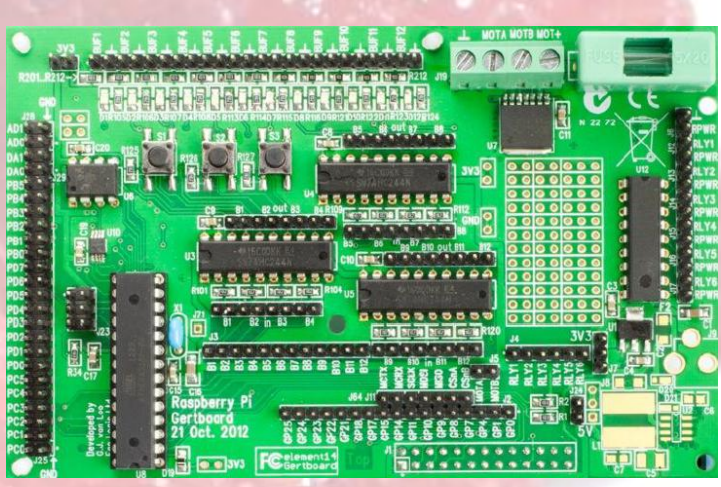
FEBRUARY COMPETITION



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 three prizes!

The first prize winner will receive one of the new pre-assembled Gertboards! The second and third prize winners will receive a raspberry coloured PCSL case.



For a chance to take part in this month's competition visit:
http://www.pcslshop.com/info/magpi

Closing date is 20th February 2013. Winners will be notified in next month's magazine and by email. Good luck!

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

December's Winners!

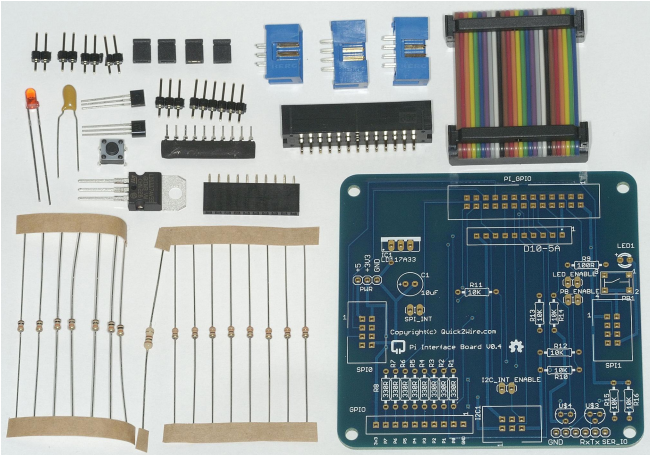
There were over 540 entries! It was a difficult choice with so many great submissions**.** The winner of the 512Mb Raspberry Pi Model B with 1A 5V power supply and PCSL Raspberry Pi case is **Fiona Parker (Ilkley, UK)**.
The 2nd and 3rd prize winners of the PCSL Raspberry Pi case are **Damien Plunkett (Flagstaff, USA)**, and **Nico van der Dussen (Pretoria, South Africa).**
Congratulations. We will be emailing you soon with details of how to claim your prizes!

Pi Interface Board Review
Quick2Wire

The Quick2Wire Pi Interface Board is a new interface board for the Raspberry Pi which will go on sale in February 2013.

I managed to get hold of one of the beta kits for review and also had a chat with Romilly Cocking, Director of Quick2Wire.

So, What's In The Kit?



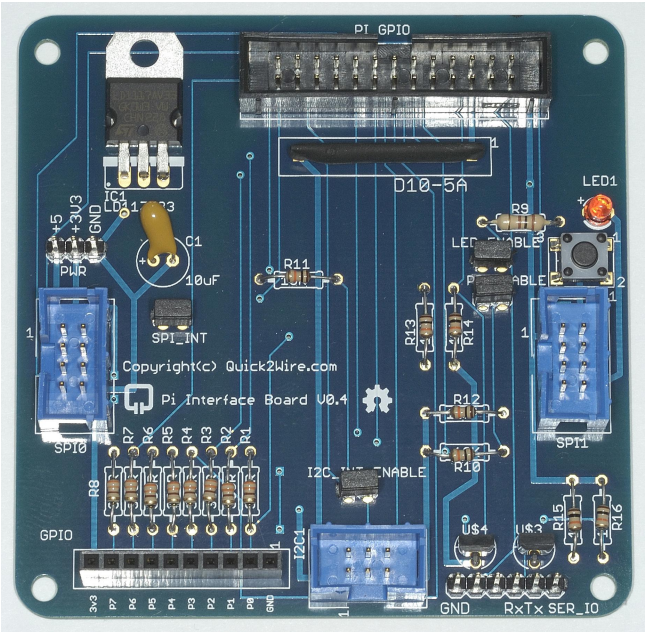
1 x printed circuit board (PCB)
1 x multi-coloured ribbon cable
4 x jumpers
11 x assorted headers
2 x transistors (FETs)
1 x tantalum capacitor
1 x light emitting diode (LED)
1 x push-button
1 x voltage regulator (3V3)
1 x diode array
16 x resistors

Interface Build

The build was a pleasure. I'd been teaching my son to solder the day before, so I'd left the leaded solder out and the leaded bit on the iron. I decided to go with those for this build.

The instructions are clear and the build is quite easy. You could probably build it without the instructions because the excellent silk screen layout, printed on the board, shows you what goes where. The only potential pitfalls are the orientation of the diode array and tantalum capacitor. They both have to be the right way round. The instructions explain how to make that happen.

It took me about 20 minutes to do all the resistors (I was fussy about aligning them). Overall, it took 53 minutes from start to finish and it was an enjoyable experience. This is what it looks like finished.



Software Installation

The next step was to fully update Raspbian and install some software to test the interface. The instructions were excellent and software installation worked flawlessly. The longest part was doing the Raspbian package updates (using `sudo apt-get update` and `sudo apt-get upgrade`).

"The thing which stands out here, compared to other Python based GPIO drivers, is that, as long as you are in the "gpio" group there's no need to be a

sudo user to run the commands. This is great!"

Testing Procedures

The following describes the test procedures from the provided manual. Test commands are given for switching the LED on/off and reading the button's status (pressed/unpressed). These worked perfectly (although the LED on mine is not very bright).

Then you are invited to test your 5V FTDI interface (a type of cable you can use to log in directly with no network connection) if you have one. Mine worked perfectly. I was able to get a command line terminal login through the serial port. And that completed the structured test procedures. Everything passed.

Testing the Quick2Wire Python Libraries

Digging a bit deeper than the test manual instructions, I downloaded the Python Quick2Wire libraries from GitHub https://github.com/quick2wire and went through two test programs I found there – flashing the led and using the button to control it. These weren't yet documented.

It's worth mentioning that you can use this software library, to control GPIO ports, regardless of what hardware you have (apart from the Pi itself). So, theoretically, I could write a third Python version of the Gertboard test suite to go with the WiringPi for Python and RPi.GPIO versions.

RGB LED

So then I wrote a little script to blink a three colour (Red, Green, Blue – RGB) LED different colour combinations using P7, P6 & P5 (Quick2Wire numbering scheme) as outputs. That also worked perfectly. Here's a link to a video guided tour of the Quick2Wire board with test programs in action. http://youtu.be/JJuKtsdAerU

We Interrupt this Article...

Another area where the Quick2Wire Python libraries stand out (which I haven't explored yet) is epoll support. This allows you to control things using interrupts instead of continual polling. This means that you can have your program respond to a change, instead of using up loads of processor power continually checking the GPIO ports' states. This is an important development.

Conclusion

The Quick2Wire Pi Interface is a very nice board with a lot of promise and what looks like being a very good Python Applications Programming Interface (API) behind it. It's going to be priced cheaper than you'd be able make one yourself. Being ultra-fussy, I'd prefer a more visible (brighter) LED. But that's really the only thing I'd change, which is saying a lot!

The Future?

There's a lot more Quick2Wire add-on boards in the pipeline, including an analog to digital converter (ADC), motor controllers, lcds, servo controllers etc. It's going to be a modular system, easy to build and easy to use. There's even talk of a Scratch interface at some point down the line.

All the hardware and software is open source. The emphasis is on software libraries and much needed free teaching resources. It looks very much to me as if Quick2Wire is one to watch. Their web site is at Quick2Wire.com.

Article by Alex Eames

Alex Eames runs the RasPi.tv blog, where he's often up to something educational, fun, innovative or just plain silly with a Raspberry Pi. He also wrote the Python port of the Gertboard software.

DID YOU KNOW?

The Quick2Wire team consists of nine people, spanning a 7 hour time zone. They live in Chicago, London, Bristol and the Pyrenees.

They have never all been in one location!

Introducing RISC OS on the Raspberry Pi

History

The reduced instruction set computing (RISC) operating system (OS) for Acorn RISC machine (ARM) based computers and emulators has been around since 1987 (originally under the guise of Arthur), almost for as long as the ARM chip itself. The first ARM 2 based computer was eponymously named the Acorn Archimedes after the famous ancient Greek inventor. At the time it represented a revolutionary leap forward on the then ubiquitous 6502 based BBC micro being sold into the home and educational market. Following on from this the more powerful RISC PC was released in 1994 based on the StrongARM chip running at 300Mhz, though production ceased around a decade ago.

In 1998 Acorn broke up and Castle Technology bought the rights to RISC OS from Pace Technology and released the Ioynix PC. From 2006 RISC OS Ltd (ROOL) has taken over RISC OS development via a shared source initiative (SSI) and a few variants now exist that run on the RISC PC emulator for Windows and Unix (RPCEmu), the beagle board, the panda board, ARMini, and of particular interest here, the Raspberry Pi.

In many ways the Raspberry Pi and RISC OS are ideally suited as partners. Key of course is that the Pi contains at its heart an ARM 11 chip. Also however, thanks to its legacy, RISC OS is undemanding on system resources and works efficiently even when CPU power and memory are in short supply. This can be largely attributed to the fact that the majority of the operating system is coded directly in ARM assembler by clever programmers. In addition, although all essential functionality is provided, system extensions and libraries are loadable as modules on an as required basis.

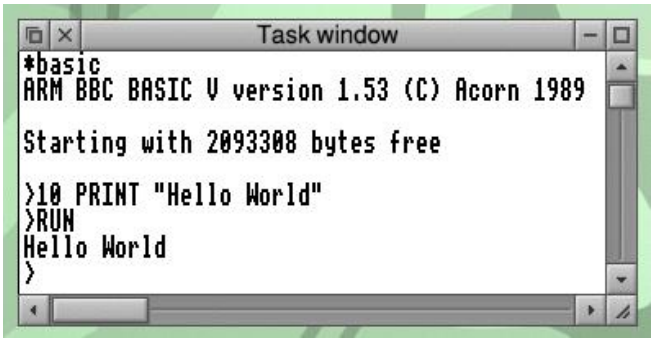
Booting

RISC OS boots straight into a windows, icons, menus, pointer (WIMP) desktop with a nice Raspberry Pi background (see screen shot) from which applications can be launched. A strip at the bottom of the screen known as the icon bar holds devices at the left and running applications at the right. Clicking on a device icon (e.g. hard drive, SD card, RAM disc) opens a filer window which can be used for browsing and launching different types of file such as BASIC programs, modules and applications. Similarly left clicking on an application typically opens a window for the user to interact with it, or clicking with the middle button brings up a menu from which configuration options can be set and actions performed. Task windows open up a command line interface (CLI) from which many common tasks can be executed and these can be grouped and packaged into Obey files for convenience.

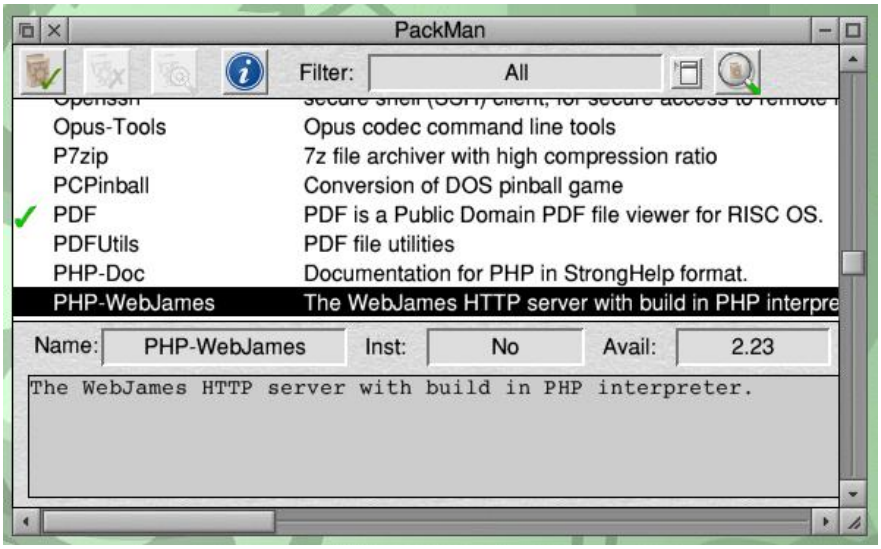
WIMP based applications co-operate through the software interrupt (SWI) based WIMP application programmers interface (API) which is documented in the Programmer Reference Manuals (PRMs). There are available from the the foundation.riscos.com web site and run to five large volumes effectively constituting the equivalent of the bible for RISCOS application developers.

Files can be pinned to the desktop for easy access and wallpapers and screen savers can be easily configured, so anyone familiar with Windows or Unix will quickly feel at home.

RISC OS comes with an internet browser called !NetSurf, though at present network access must be provided via an ethernet cable. ARM BBC Basic can be started from a Task Window (press ctrl F12) from the * command prompt by typing Basic.



Bundled applications include a text editor !Edit, a drawing program !Draw and a painting application !Paint, however a plethora of third party applications are available including games, music, DTP and art packages. Many of the most used of these are freely installable using the supplied package manager application (!PackMan) which is styled on the lines of Linux Update Manager.



RISC OS Open have joined forces with some of the leading software developers in the RISC OS community and sell at a large discount the Nut Pi, a package of flagship RISC OS software specifically for RISC OS Pi.

A tasks window allows memory to be allocated between various parts of the system and user applications, and a number of different screen modes of different pixel counts and number of colours (up to 24 bit colour depth) are supported.

Installation

To install RISC OS on the Raspberry Pi visit the www.raspberry.org/downloads page and follow the download instructions. The download zip contains a disc image file that can be written to an SD card using the freely available Windows based Win32DiskImage application or the Unix dd tool in the same way as the Raspbian wheezy distribution.

My particular interest in RISC OS on the Raspberry Pi is as a host for the Charm set of development tools and demos targetted at the educational and enthusiast sector, for which I am the author. A GPL licensed release is bundled with the distro, however the latest release is available from www.charm.qu-bit.co.uk which is optionally recompilable to utilise the VFP co-processor for floating point operations. See the article on Charm in the next edition of the MagPi for more information.

In summary RISC OS on the Raspberry Pi is easy to install, quick to boot at under 20 seconds, responsive, intuitive and easy to pick up so why not give it a whirl?

Article by Pete Nowosad

Installing & configuring

Learn how to install Arch Linux, a barebones rolling LINUX distribution.

Many people think of Linux as an operating system, but in fact it's actually just a kernel, the base. To make it into a proper operating system, you need to add a little bit more. As Linux is free and open-source, many people have done this, and each one has done it slightly differently, creating many different 'distributions', or 'distros' for short.

The main Raspberry Pi distribution offered at http://www.raspberrypi.org/downloads is Raspbian, a version of Debian. However, if you scroll down a bit more, you'll see some others, including one called Arch Linux.

So what's the difference between Raspbian and Arch? The main difference is the package managers and the way updates are managed.

Debian, and therefore Raspbian, are very strict on package updates. They have to go through testing, so the maintainers can be sure they are stable and work before releasing them to the regular users. This is good because it means software is almost guaranteed to work, but not so good as it means updates take a little while to reach users.

Arch Linux is different in this respect, releasing updates as soon as possible. For this reason it is called a 'rolling release' distro, since you only have to install it once and then whenever a package gets an update you can upgrade it there and then. This allows users to get updates more quickly, although it means software is more unstable. In case of trouble, you can simply image the SD card again.

The other major difference between the two is that Raspbian comes completely ready, while Arch comes with the bare essentials, allowing users to pick what they want to install. This makes setup harder for newcomers though, but this guide should help ease the process.

So, if continuous updates and your Pi the way you want it sound good, why not have a go at installing Arch? **You will need an internet connection for your Pi though**, so if your internet is particularly slow or you have a low download limit, it may be best to stick with Raspbian.

First download the latest image from http://goo.gl/az5T4

Then flash the image to the SD card, using `Win32DiskImager` or `dd`. More information can be found in Issue 2 of the MagPi. With that done, we can get on with the setup.

First boot

The first boot might take a little while longer, just wait until it's done. Once you get to the login screen, use the user name `root` and the password `root`.

You will then have a terminal open. You will notice if you try `startx`, it will not work. That gives you an idea of how little Arch comes with. Don't worry though, we'll get to installing a graphical user interface.

Before you begin doing anything, you may want to change the keyboard layout to your country. This is done by typing:

```
loadkeys country code
```

I'm in England, so my country code would be `uk`. A full list can be found here: http://goo.gl/xZh9N. This is only temporary, we will set it permanently later on.

Editing language settings

The default hostname for the system is alarmpi. I personally don't like this, and would rather it was something else. If you feel the same, you can change the hostname of the system by typing:

```
echo hostname > /etc/hostname
```

where `hostname` is the new hostname you want. I've used `raspberrypi`. It will not be effective until after a reboot.

Next we will change the language and the timezone. To look at the available timezone areas, type:

```
ls /usr/share/zoneinfo
```

Choose which area suits you best (for me it would be Europe) and then type:

```
ls /usr/share/zoneinfo/area
```

to see the specific timezones. Pick one, and type:

```
ln -s /usr/share/zoneinfo/area/timezone
/etc/localtime
```

all on one line. My choice was London, so area would be Europe and timezone London. If you get an error saying “File exists”, type:

```
rm /etc/localtime
```

Then type the previous command again.

Now to edit the locale. This is used to determine the country you live in so things like dates can be displayed correctly. To do this, we need to edit some files. Type the following:

```
nano  /etc/locale.gen
```

find your country and remove the '#' symbol in front of it. For example, mine would be en_GB. When you are done, use Ctrl+O to save and Ctrl+X to exit. Then type:

```
locale-gen
```

Now, we need to make another file, so type:

```
nano /etc/locale.conf
```

and edit it to the same language and country code as before.

Finally we need to set the console's keymap so it fits the country's keyboard all the time. For this type,

```
nano /etc/vconsole.conf
```

and change KEYMAP to the country code you used with the loadkeys command previously.

All the language settings are set now, so if you want you can reboot to see the changes, by typing:

```
reboot
```

Using pacman

Arch Linux's package manager is called pacman, and we use that to install packages. To install a package, type:

```
pacman -S <package name>
```

Try it with the package sudo, because we'll need it later.

As Arch is a 'rolling release', quite a lot of updates have come out since the image was released, so to upgrade all your packages type:

```
pacman -Syu
```

These should both work fine straight away with the most recent image, though you need an internet connection. Because of how quickly updates come, it's recommended you run a full upgrade regularly, once a week or maybe even once a day to keep on top of it all.

Should you want to remove a package, you can do that with:

```
pacman -R <package name>
```

And to see a list of installed packages, type:

```
pacman -Q
```

Adding a new user

It is vitally important that we make a new user for our system, as logging in as root has security issues. To add a new user, enter:

```
adduser
```

and follow the instructions to add a new user.

Next, we need to add them to the sudoers list so they can still install programs, but in a more secure way. If you haven't already, install sudo. To add the user to the `sudoers` file, type:

```
export EDITOR = nano && visudo
```

This will allow you to edit the `sudoers` file with the familiar `nano` editor. Find the line that says `root ALL=(ALL)  ALL` and copy it onto a different line, replacing root with the user name of the new user. If you would like to have it so sudo does not ask for your password, like on the default Raspbian install, put `NOPASSWD:` in front of the final ALL.

Finally, we can change the password for the root account, using the command:

```
passwd root
```

Then be sure to pick a secure password.

And with that, we're done with the basic setup! Type:

```
logout
```

to log out of root and login as the new user you set up.

Setting up a graphical user interface

This final part is optional, but if you'd prefer more than just a command line you should do it. To install a graphical interface, simply type:

```
pacman -S gamin dbus xorg-server xorg-xinit xorg-server-utils mesa xf86-video-fbdev xf86-video-vesa xfce4
```

all on one line. Once the installation is finished, type:

```
cp /etc/skel/.xinitrc ~/.xinitrc
```

and then:

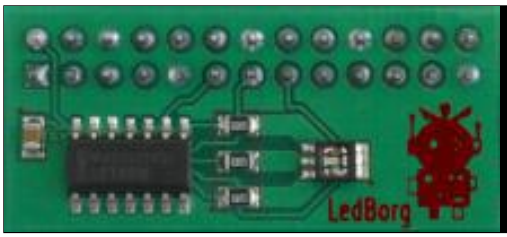
```
echo "exec startxfce4" >> ~/.xinitrc
```

and finally:

```
startx
```

Your graphical environment should then start. Congrats, you now have a working Arch Linux system on your Raspberry Pi! You may want to edit the `config.txt`, but this process is the same as Raspbian. Have fun!

Article by Alex Kerr



Introducing Vala

Writing a simple web controller for the LedBorg RPi add-on

The Vala language is, as programming languages go, very much a new kid on the block and is still under many programmers' radar. We will use Vala to communicate with LedBorg from www.piborg.com/ledborg over the internet. [Ed: also in this issue we will see how we can perform a similar activity with Python in The Python Pit.]

Vala is a C# style language built on the GLib object system providing easy access to the base GNOME libraries and subsystems. The compiler, valac, turns the Vala code into C, and in turn triggers the system's C compiler to produce native code. This means that, unlike C#, there is no .NET framework or virtual machine. Effectively, it is a higher-level way of writing C apps. The project's home page is https://live.gnome.org/Vala .

Although Vala is based around the GLib and GNOME libraries, it's not only for developing GNOME desktop apps and is also good for writing console-based apps and services.

LedBorg

LedBorg is pre-assembled, it sits on the GPIO pins and has an ultra-bright RGB LED. Each channel - red, green and blue, has three intensities: off, half-brightness, and full-brightness.

The LedBorg driver creates the device file /dev/ledborg. This is a great example of the UNIX philosophy at work with devices exposed as files instead of mysterious APIs. We can read and write to /dev/ledborg just like we would with any other file, for example:

```
echo "202" > /dev/ledborg
```

The three digits control R, G and B, with each set to either 0, 1 or 2, corresponding to the three intensities, eg: '202' makes the LedBorg light up bright purple (red+blue).

Network Control

I decided that an easy way to control LedBorg remotely was to use the well-known HTTP protocol. Using LibSoup in Vala, it is easy to set up a light-weight HTTP server that can respond to requests. Testing would be straightforward from any web browser on the network.

The server takes GET requests in the following URL format:
/?action=SetColour&red=x&green=y&blue=z
where x, y, and z are integers between 0 and 2.

For the ease of getting started, the program also responds to all requests with a very minimal HTML form containing drop-down selectors for the three colours, and a submit button.

The Code

To try out this code, you will need to have the following packages installed plus dependencies. Assuming Raspbian/Debian is the running OS:

```
$ sudo apt-get install valac \
  libsoup2.4-dev
```

After entering the code, below, and saving as LedBorgSimpleServer.vala it can then be compiled with the following command:

```
$ valac --pkg libsoup-2.4 --pkg \
  gio-2.0 --pkg posix --thread -v \
  LedBorgSimpleServer.vala
```

You may want to enter this command line in a text file and save it as compile.sh - then make it executable by running

```
$ chmod +x compile.sh
```

You can re-compile with

```
$ ./compile.sh
```

The -v flag generates verbose output, giving an idea of what it is doing. If you want to see the generated C code, add the -C flag to get LedBorgSimpleServer.c

Run the program with ./LedBorgSimpleServer

and navigate to your Pi's IP address in a browser, adding :9999 to specify the port number, eg: http://192.168.1.69:9999 .

The code is missing some robustness: we are not checking for the presence of the red, green and blue GET parameters, nor are we validating their values. Nor is feedback in the HTML sent back to the client, to say whether the operation was successful, what colour has been set, or if the device wasn't found.

These additions can be an exercise for the reader!


```
// LedBorgSimpleServer.vala

// the namespaces we'll be using
using GLib;
using Soup;

// our main class
public class LedBorgSimpleServer :
GLib.Object {
  // define the port number to listen on
  static const int LISTEN_PORT = 9999;

  // define the device file to write to
  static const string DEVICE =
    "/dev/ledborg";

  // the method executed when run
  public static int main (string[] args)
{
    // set up http server
    var server = new Soup.Server(
      Soup.SERVER_PORT, LISTEN_PORT);

    // handle requests from the client
    server.add_handler("/",
      default_handler);

    // get the running http server
    server.run();

    return 0;
  }

  // default http handler
  public static void
default_handler(Soup.Server server,
Soup.Message msg, string path,
GLib.HashTable<string, string>? query,
Soup.ClientContext client)
  {
    // action a request
    if(query != null)
    {
      // check parameter to be sure
      if(query["action"] == "SetColour")
      {
        // get RGB from url params
        string red = query["red"];
        string green = query["green"];
        string blue = query["blue"];

        /* build our RGB colour string
        Each 0, 1 or 2:
        off, half or full brightnesss */
        string colour = red + green +
          blue;

        // do colour change
        do_colour_change(colour);
      }
    }

    // build the html for the client
    string html = """
<html>
  <head>
    <title>LedBorgSimpleServer</title>
  </head>
  <body>
    <form method="get" action="/">
     Red:<select name="red">
      <option value="0">Off</option>
      <option value="1">1/2</option>
      <option value="2">Full</option>
     </select>
     Green:<select name="green">
      <option value="0">Off</option>
       <option value="1">1/2</option>
       <option value="2">Full</option>
     </select>
     Blue:<select name="blue">
      <option value="0">Off</option>
      <option value="1">1/2</option>
      <option value="2">Full</option>
     </select>
     <input type="submit" name="action"
value="SetColour" />
    </form>
  </body>
</html>
    """;

    // send the html back to the client
    msg.set_status_full(
      Soup.KnownStatusCode.OK, "OK");
    msg.set_response("text/html",
      Soup.MemoryUse.COPY, html.data);
  }

  // do the colour change
  public static void
    do_colour_change(string colour)
  {
    /* Here we use posix file handling
    to write to the file instead of
    vala's gio file handling, as we
    don't want the safety of
    gio getting in the way when
    operating in /dev */
    // open the file for writing
    Posix.FILE f = Posix.FILE.open(
      DEVICE, "w");

    // write the colour string to file
    f.puts(colour);
  }
}
```

Article by Ross Taylor

The MagPi What's On Guide

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

Then this new section of the MagPi is for you! We aim to list Raspberry Jam events in your area, providing you with a RPi 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

Preston Raspberry Jam

When: Saturday 9th February 2013 @ 10:00am
Where: Accrington Academy, Queens Road West, BB5 4FF, UK

The meeting will run from 10:00am until 5:00pm and is hosted by TechWizZ
Tickets and further information are available at http://techwizz.eventbrite.co.uk

New York City Raspberry Jam

When: Thursday February 21st 2013 @ 7:00pm
Where: Two Sigma, 16th Floor, 100 Avenue of the Americas, New York, NY, USA

Bring your projects, ideas and see what others are doing. There will even be a demo of Adafruit's WebIDE. Further information is available at http://www.meetup.com/NYC-Raspberry-Jam

Sheffield Raspberry Jam

When: Last Wednesday of the month @ 6:30pm
Where: 1st Floor, The Workstation, Grinders Hill / Brown St., Sheffield, S1 2BX, UK

The meetings are hosted by GISThub. Doors open at 6:20pm and the meeting runs from 6:30pm until 8:30pm. Further information is available at http://sheffieldraspi1210.eventbrite.com

CAS York Hub Meeting

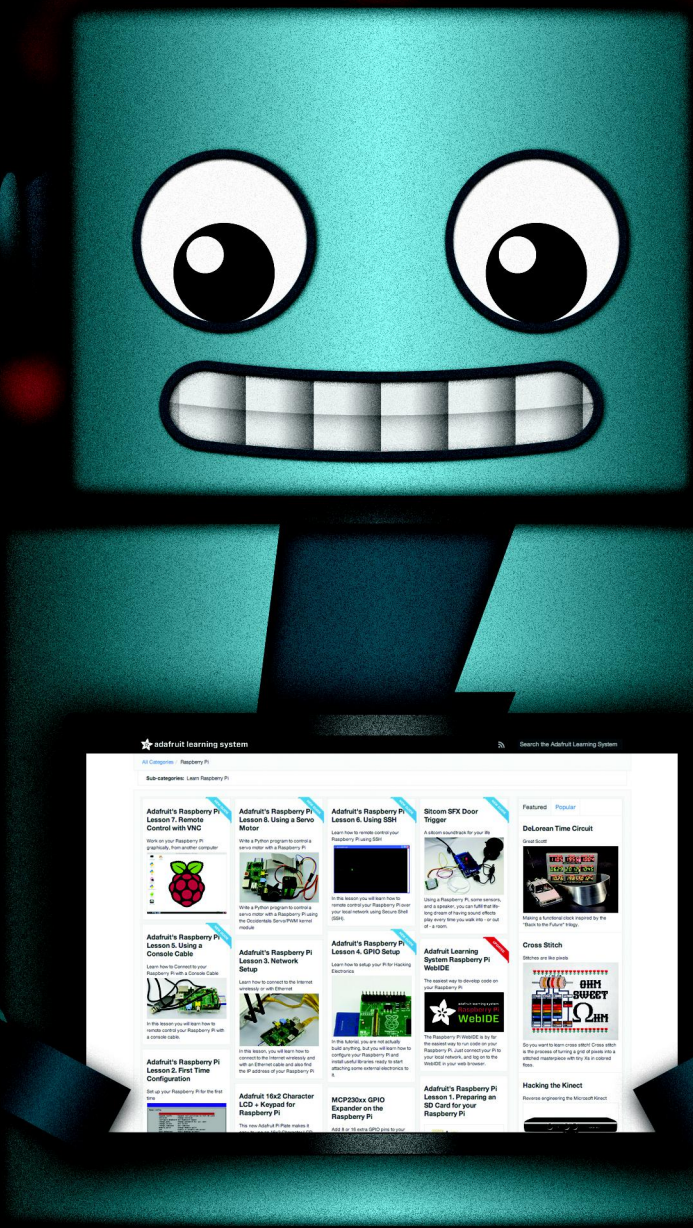
When: Wednesday 13th February 2013 @ 4:30pm
Where: National STEM Centre, University of York, YO10 5DD, UK

The meeting will run from 4:30pm until 7:30pm and is entitled "Engaging Pupils with Raspberry Pi".
Tickets and further information are available at http://rcasstem.eventbrite.co.uk

adafruit learning system

Raspberry Pi WebIDE

BUILD AMAZING THINGS & LEARN HOW TO PROGRAM WITH THE RASPBERRY PI



adafruit

HTTP://LEARN.ADAFRUIT.COM/CATEGORY/RASPBERRY-PI

Tutorial 5 - Structs, header files and data analysis.

Welcome back to the C cave. This tutorial includes an example program built from several C source files and compiled with a Makefile. Before continuing, how did you get on with the previous challenge problem?

Challenge solution

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/sysinfo.h>

int main() {
  int i = 0;
  float ramUsed;
  char gpCmdOne[250], gpCmdTwo[250], systemCmd[1000];
  FILE *cmdPtr = 0, *outPtr = 0;
  char c, fileName[100], *strPtr = 0;
  struct sysinfo info; /* A sysinfo struct to hold the status. */

  cmdPtr = popen("hostname","r"); /* Run hostname command. */
  if(!cmdPtr) return 1;
  strPtr = &fileName[0]; /* Get a pointer to the string. */
  while((c=fgetc(cmdPtr)) != EOF) { /* Reach each character. */
    *strPtr = c; /* Set the character value. */
    strPtr++; /* Move to the next array position. */
  }
  pclose(cmdPtr); /* Close the hostname file. */
  strPtr--; /* Move backwards one array element to overwrite the new line. */
  sprintf(strPtr,"-data.txt"); /* Append the suffix. */
  printf("%s\n",fileName);
  outPtr = fopen(fileName,"w"); /* Open the output file. */
  if(!outPtr) return 1; /* If the output file cannot be opened return error */
  for(i=0;i<60;i++) {
    sysinfo(&info);  /* Get the system information */
    ramUsed = info.totalram - info.freeram;
    ramUsed /= 10240.0;
    fprintf(outPtr,"%d %f %d\n", i, ramUsed, info.loads[0]); /* Write ram used. */
    usleep(500000); /* Sleep for 1/2 a second. */
  }
  fclose(outPtr); /* Close the output file. */

  /* Now plot the data */
  sprintf(gpCmdOne, "plot \'%s\' using 1:2 title \'%s\'", fileName, "Ram used");
  sprintf(gpCmdTwo, ", \'%s\' using 1:3 title \'%s\'\n", fileName, "Load");

  /* Create the full command, including the pipe to gnuplot */
  sprintf(systemCmd,"echo \"%s%s\" | gnuplot -persist",gpCmdOne,gpCmdTwo);

  system(systemCmd); /* Execute the system command. */
  return 0; /* Return success to the system. */
}
```

The solution includes functions and techniques discussed in previous tutorials. There are more simple ways to form the file name from the host name. C provides a `string.h` header file which includes the declaration of several useful functions for string operations. The full list of functions can be viewed by typing `man string`. Strings can be concatenated by using `strcat`,

```
char fileName[100]="myHost", suffix[10]="-data.txt"
strcat(fileName,suffix);  /* Append suffix to fileName, result in fileName. */
```

The host name can be read using `fgets` rather than `fgetc`,

```
fgets(fileName,100,cmdPtr); /* Read until EOF or newline or 99 characters. */
```

where 100 is the size of the `fileName` character array. Lastly, the host name can also be read using the `gethostname` function from `unistd.h`,

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>

int main() {
  char fileName[100], suffix[10]="-data.txt";
  gethostname(fileName,100);
  strcat(fileName, suffix); /* Append the suffix to the fileName */
  printf("%s\n",fileName);
  return 0;
}
```

Structs

Structs were introduced quickly in the last article in issue 6, to allow the use of system information. Structs occupy a continuous block of memory, similar to FORTRAN common blocks. The syntax of their usage is very similar to C++ classes with public data members. A `struct` is defined with a name and compound definition of variables. These variables can also be structs. Starting with a simple `struct`,

```
struct dataPoint {
  unsigned int timeSeconds;
  float value;
};
```

The `int timeSeconds` is defined first in memory and then the `float value`. The size of a `struct` in memory is the sum of the sizes of the variables. The definition of a `struct` should be made before its use and is typically found in a header file, but can also be written in the same file before its usage. To test this simple `struct`,

```
int main() {
  /* Declare a variable of "struct dataPoint" type. */
  struct dataPoint s;
  /* Assign the struct s some starting values. */
  s.timeSeconds = 60;
  s.value = 100.0;
  /* Print the size and memory locations */
  printf("sizeof(s) = %ld\n", sizeof(s));
  printf("&(s.timeSeconds) = %p\n", &(s.timeSeconds));
  printf("&(s.value) = %p\n", &(s.value));
  printf("sizeof(unsigned int) = %ld\n", sizeof(unsigned int));
  printf("sizeof(float) = %ld\n", sizeof(float));
  return 0;
}
```

where the program assigns values and prints the memory locations to demonstrate the memory structure.

When structs are passed to functions, by default a local copy of the struct is made within the function. This means that when the function finishes the value of the struct in the function which called it is unchanged. This is the same behaviour as if a basic variable had been passed to the function,

Continued over page...

```
void printDataPoint(struct dataPoint dp) {
  printf("timeSeconds = %d, value = %f\n", dp.timeSeconds, dp.value);
}
```

To modify the values of a struct within a function and retain these values, pointers can be used:

```
void clearDataPoint(struct dataPoint *dp) {
  dp->timeSeconds = 0;
  dp->value = 0.;
}
```

where the `dp->timeSeconds` syntax is equivalent to `(*dp).timeSeconds`. Other than the short hand "->" syntax, the behaviour is exactly the same as that of simple variables discussed in Issue 5.

Header files

To illustrate the use of header files, the next example defines a histogram data structure and functions. Histograms can be very useful to monitor long term experiments, provide summary figures or be used for data storage themselves.

Header files should be included to use functions within standard libraries or functions implemented within other C source files. These files contain the declaration of functions and data structures, but do not contain the implementation of the functions. The implementation is given within `.c` files, which are complied and built into static or dynamic libraries or directly linked to. Similar to standard header files, additional header files can be written to contain function definitions and data structures.

```
#ifndef HISTOGRAM_H
#define HISTOGRAM_H
#define MAX_BINS 1000
/* Define a data structure to hold histogram data. */
struct histogram {
  unsigned int nBins;
  float xMin;
  float xMax;
  float binContents[MAX_BINS];
};
/* Define the struct as a new type, as a shorthand. */
typedef struct histogram Histogram;
/* Fill a histogram. */
int fillHist(Histogram *, float value, float weight);
/* save a histogram to a file. */
int saveHist(Histogram *, FILE *);
#endif
```

is a header file called `histogram.h`, which defines a struct and declares functions, but does not implement the functions it defines. In the case that the header file is included inside another header file, the header file might be included in a program more than once. To prevent this double declaration, a precompiler `ifndef` case is used. This case is true the first time the header file is included and false for additional include statements. The define statements define values which are replaced when the precompiler runs, which is just before the compilation of the code. Since dynamic memory allocation has not been discussed yet, a fixed size array is used for the `binContents`. Lastly `typedef` is used to simplify the Histogram variable declaration. The header file must be included in a program before the struct or functions are used,

```
#include "histogram.h"
#include <stdio.h>
#include <stdlib.h>
int main() {
  unsigned int i;
  Histogram h; /* Create a histogram struct */
  initHist(&h,10,0.,10.); /* Initialise the histogram */
  for(i=0;i<1000;i++) { /* Generate 1000 random points */
    fillHist(&h,10*(float)rand()/RAND_MAX,1.0); /* Histogram each value. */
```

```
  }
  saveHist(&h,"Hist.txt"); /* Save the histogram. */
  return 0;
}
```

This program histograms random numbers, which are generated between zero and one. The program cannot be run without implementing functions defined in the `histogram.h` header file. This implementation should be given in a `.c` file,

```
#include "histogram.h"
#include <stdio.h>
int initHist(Histogram *hist, unsigned int nBins, float xMin, float xMax) {
  unsigned int i;
  if((hist->nBins+2) >= MAX_BINS) {
    printf("Error: too many bins requested.\n");
    return 1; /* An error has occured. */
  }
  hist->nBins = nBins;
  hist->xMin = xMin;
  hist->xMax = xMax;
  for(i=0;i<(hist->nBins+2);i++) hist->binContents[i] = 0.;
}
int fillHist(Histogram *hist, float value, float weight) {
  unsigned int ibin;
  float binSize;
  if(value < hist->xMin) ibin = 0; /* Underflow */
  else if(value >= hist->xMax) ibin = hist->nBins+1; /* Overflow */
  else { /* Find the appropriate bin. */
    ibin = 1;
    binSize = (hist->xMax - hist->xMin)/hist->nBins;
    while(value >= (ibin*binSize + hist->xMin) &&
          ibin < hist->nBins && ibin < MAX_BINS) {
      ibin++;
    }
  }
  if(ibin >= MAX_BINS) { /* Stay within the array */
    printf("Error: ibin = %u is out of range\n",ibin);
    return 1; /* An error has occured. */
  }
  hist->binContents[ibin] += weight; /* Add the weight */
  return 0;
}
int saveHist(Histogram *hist, const char *fileName) {
  FILE *outputFile = 0;
  unsigned int ibin;
  float binSize;
  outputFile = fopen(fileName, "w"); /* Open the output file. */
  if(!outputFile) { /* If the file is not open.*/
    printf ("Error: could not open %s for writing\n",fileName);
    return 1; /* An error has occured. */
  }
  binSize = (hist->xMax - hist->xMin)/hist->nBins;
  /* Write the bin centres and their contents to file. */
  ibin=0;
  while (ibin < (hist->nBins+2) && ibin < MAX_BINS) {
    fprintf(outputFile,"%lf %lf\n",
            binSize*((double)ibin+0.5) + hist->xMin,
            hist->binContents[ibin]);
    ibin++;
  }
  fclose(outputFile); /* Close output file. */
  return 0;
}
```

Continued over page...

Rather than type gcc several times, a Makefile can be used to build the source files and produce the executable,

```
CC = gcc
TARGET = hist
OBJECTS = $(patsubst %.c,%.o, $(wildcard *.c))

$(TARGET): $(OBJECTS)
	@echo "** Linking Executable"
	$(CC) $(OBJECTS) -o $(TARGET)

clean:
	@rm -f *.o *~

veryclean: clean
	@rm -f $(TARGET)

%.o: %.c
	@echo "** Compiling C Source"
	$(CC) -c $<
```

where more information on make is given in issue 7. Put the two .c files in the same directory as the histogram.h and Makefile. Then type make to build the executable. When the program runs it produces a text file which contains the sum of the weights within the underflow, bins and overflow. The format is chosen to allow the histogram to be plotted with gnuplot,

```
gnuplot
plot 'Hist.txt' with boxes
```

Similar to the previous tutorial, this plotting command could be added to a program also. The plot can be saved as a png file by typing,

```
set term png
set output 'Hist.png'
replot
```

after the original plotting command.

Challenge problem

Use the previous article and this article to histogram the system load for 30 minutes. Add functions to the histogram.h and histogram.c files to calculate the mean and standard deviation of the distribution. The mean of a histogram can be calculated from,

```
xMean = 0.;
for (i=1;i<=nBins;i++) { /* Skip underflow and overflow */
  xMean += binContents[i]/nBins;
}
```

The standard deviation of a histogram is given by,

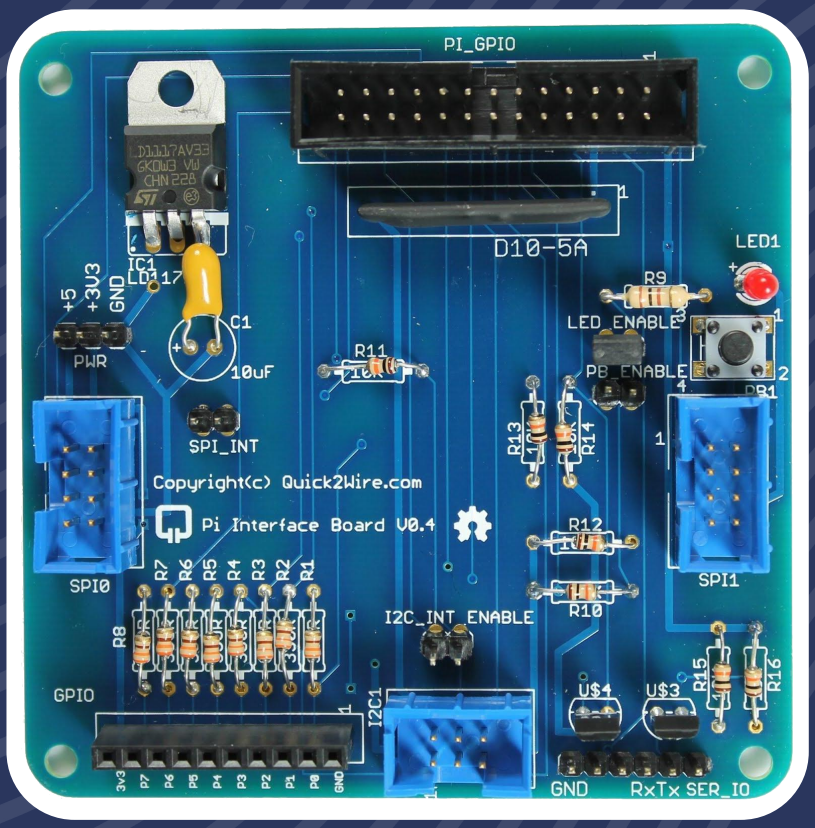
```
xStdDev = 0.;
for(i=1;i<=nBins;i++) { /* Skip underflow and overflow */
  xStdDev += pow(xMean - binContents[i],2)/(nBins-1);
}
if(xStdDev > 0.) xStdDev = sqrt(xStdDev);
```

The solution to the problem will be given next time.

Article by W. H. Bell

Quick2Wire

SAFE AND SIMPLE CONNECTION TO YOUR RASPBERRY PI



HARDWARE

- Interface board
- I²C Port Extender
- Analogue board

SOFTWARE

- For GPIO, I²C, SPI
- Device Libraries
- Examples

SUPPORT

- Articles
- Tutorials
- Forum

Interface board: £13.86 | Port Extender: £9.80 | Combo: £22.66 (save £1.00)

Prices include UK VAT but exclude Postage and Packaging: from £2.70

Find out more at quick2wire.com

THE SCRATCH PATCH

Scratch Controlling GPIO

This month's article intends to make it as SIMPLE AS PI to get up and running with GPIO in Scratch and allow your Raspberry Pi to control some lights and respond to switches and sensors.

Whilst the Raspberry Pi is a great tool for the creation of software, using languages such as Scratch, Python, C etc., the best way to make it really come alive and to add even more enjoyment to this cheap, credit card sized computer is to start playing around with hardware hacking and physical computing. This involves using the Raspberry Pi to control things like LEDs, and respond to switches and sensors. More often than not it also includes knowledge and learning of both hardware and software in a very interesting practical environment – not just coding for the sake of coding but, for example, creating robots and programming them to do cool things!

This article is based on a post on the Cymplecy blog by Simon Walters (http://wp.me/p2C0q1-27), a primary school ICT teaching assistant and general Scratch guru!

Minimum Requirements – a Raspberry Pi with Raspbian installed and a working internet connection, a breadboard, some Light Emitting Diodes (LEDs), some resistors and some wire connectors. Total cost £5-£10 (not including the Pi).

How to get a Rapsberry Pi to control the GPIO Pins from Scratch

Your RaspberryPi needs to be connected to the internet to install the software but internet is not needed to run ScratchGPIO. Copy the text below (starting at sudo and ending at gpio.sh) and paste that into an LX Terminal window and run it to download the installer**:**

```
sudo wget http://db.tt/jCIVXBJE -O /boot/install_scratch_gpio.sh
```

And then type, and run: `sudo /boot/install_scratch_gpio.sh`

This will install all the necessary extra software and some simple examples. (If you do not have internet on your Pi then put your SD card into a card reader and try using your browser to right-click and save the script direct to your SD card and then put it back into your Pi and run the second instruction)

Connecting Components Up

Extreme care should be taken when connecting hardware to the GPIO pins as it can damage your Pi – only do this if you're confident of your ability to follow these instructions correctly**.** At a minimum you should get a breadboard and use some female-male 0.1 leads (available from RS/CPC or your local Maplin). Check out some GPIO pin guides to make sure you know what pins do what.

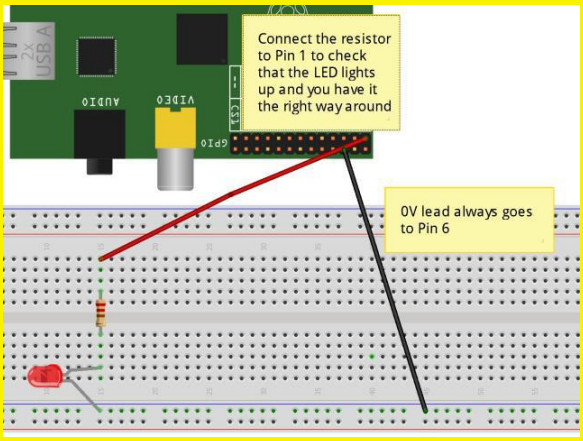


Figure 1 - LED Test

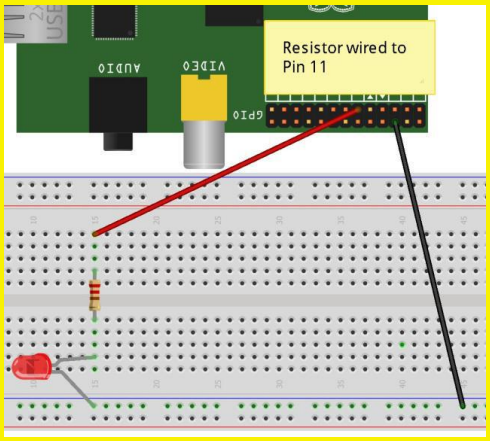


Figure 2 - GPIO Test

As in Figure 1 above, wire up Pin 1 (3.3V) to (at least) a 330ohm resistor – connect that resistor to the long lead (this is the positive lead) of an LED and then connect the other end of the LED to Pin 6 (Ground). This should cause the LED to light up. If it doesn't work try reversing your LED, as you probably have the polarities reversed. Once working, you can now move the red (or whatever colour you have used) lead from Pin 1 to Pin 11, as in Figure 2 above.

You should now run the special Scratch icon (Scratch GPIO) on your desktop. This is actually a completely normal version of Scratch, it just runs a little Python background program to allow Scratch to talk to the GPIO. If you have any Python programs accessing the GPIO running already this could cause your Pi to crash when opening Scratch GPIO. To avoid this, open an LX Terminal window and run: `sudo killall python`

To test out the GPIO control in Scratch, click File>Open and select blink11 from /home/pi/Scratch. Once the program has loaded, click on the Green Flag and your LED should now blink on for 1 second and off for 2 seconds – see troubleshooting on the Cymplecy blog if this doesn't happen.

What more can I do with Scratch and the GPIO?

Next time we will be looking at more exciting ways to use the GPIO with Scratch.

Article by Simon Walters and Aaron Shaw

the python pit

In this month's Python Pit we show you how to control your Raspberry Pi with your smartphone by using the web.py framework from www.webpy.org.

This article provides an alternative to using Vala, also covered in this issue.

With web.py you can present web forms on client browsers (we are using Chrome on Android or Safari on iOS) and accept the form content via HTTP Post back to the web server (your Raspberry Pi). The values returned are fed directly to your Python script, which in turn can then execute any commands that you require. Web.py contains an in-built webserver meaning that it is not necessary to install Apache or Lighttpd.

By way of an example we will use your smartphone's browser connected via WiFi to your local area network and in turn your Pi to perform basic remote control of LedBorg (www.piborg.com). LedBorg is a pre-assembled LED array capable of outputting 27 colours. However for this tutorial we will just demonstrate the principle with red, green and "black" (all LEDs off). The technique described below works equally well for any other Pi remote control project. Hence replace the LedBorg specific code to meet your own requirements.

We can install web.py from the command line via PIP (see The Python Pit in issue 8 for instructions on installing PIP):

```
sudo pip install web.py
```

Next create a directory (pyserver in this example) on your Pi to act as the root of the web server that web.py will start. Inside create two other directories and and save the following files within:

```
pyserver/ledborg.py
pyserver/templates/index.html
pyserver/static/styles.css
```

We will concentrate on the first two files: ledborg.py contains our python code and index.html the template web page that will be called when the program executes. The stylesheet, styles.css is optional and changes the usual drab grey buttons found on web forms to be coloured and larger, as seen in the screenshot. The styles were generated at www.cssbuttongenerator.com.

Run the program on your Pi and navigate to the Pi's IP address on your smartphone, appending the port number 8080. This will present you with the web form, enabling commands to be sent to your Python script when you tap the buttons. In the example screenshot the smartphone connects to the Pi via http://192.168.1.69:8080 - you can determine your IP address by executing ifconfig at the command line.

ledborg.py

```
# web.py: controlling your Pi from a smartphone
# using LedBorg (www.piborg.com) as an example
# ColinD 27/12/2012

import web
from web import form
```

PYTHON VERSION: 2.7.3rc2
PYGAME VERSION: 1.9.2a0
O.S.: Debian 7

TESTED!

```
# Define the pages (index) for the site
urls = ('/', 'index')
render = web.template.render('templates')

app = web.application(urls, globals())

# Define the buttons that should be shown on the form
my_form = form.Form(
 form.Button("btn", id="btnR", value="R", html="Red", class_="btnRed"),
 form.Button("btn", id="btnG", value="G", html="Green", class_="btnGreen"),
 form.Button("btn", id="btnO", value="0", html="-Off-", class_="btnOff")
)

# define what happens when the index page is called
class index:
    # GET us used when the page is first requested
    def GET(self):
        form = my_form()
        return render.index(form, "Raspberry Pi Python Remote Control")

    # POST is called when a web form is submitted
    def POST(self):
        # get the data submitted from the web form
        userData = web.input()

        # Determine which colour LedBorg should display
        if userData.btn == "R":
            print "RED"
            lbColour = "200" #Rgb
        elif userData.btn == "G":
            print "GREEN"
            lbColour = "020" # rGb
        elif userData.btn == "0":
            lbColour = "000"
            print "Turn LedBorg Off"
        else:
            print "Do nothing else - assume something fishy is going on..."

        # write the colour value to LedBorg (see www.piborg.com)
        LedBorg = open('/dev/ledborg', 'w')
        LedBorg.write(lbColour)
        print lbColour
        del LedBorg

        # reload the web form ready for the next user input
        raise web.seeother('/')
# run
if __name__ == '__main__':
    app.run()
```

templates/index.html

```
$def with (form, title)
<!doctype html>
<html>
    <head>
        <title>$title</title>
        <link rel="stylesheet" type="text/css" href="/static/styles.css">
    </head>
    <body>
        <br />
        <form class="form" method="post">
        $:form.render()
        </form>
    </body>
</html>
```

Try adding a Random button to output any one of the 27 possible colours from LedBorg shown in the image at the bottom of the previous page.

Web.py supports several other form elements including drop-down lists and checkboxes. Full details can be found on at www.webpy.org.

the python pit

Feedback & Question Time

Great job in making possibly the most useful magazine I have ever read! I'm only 14 years old and your mag has inspired me to start programming. Thanks!

J Forster

Great job to you all for the 2012 issues. I am one of your 600+ backers with the Kickstarter to get a physical copy of them. I was wondering if you had considered doing a similar 2013 type of fund raiser - or offering a subscription service through your website?

T Giles

[**Ed:** Yes we are planning a subscription service. Look out for a survey soon on different options]

Congratulations on the magazine! As an RPi user of a certain "Vintage" it's nice to "Tinker" with a cheap, relatively simple, relatively powerful, open source piece of hardware again!

P Welsh

Your publication is outstanding! The variety of well composed articles each MagPi issue contains are incredibly helpful. Please keep up the good work!

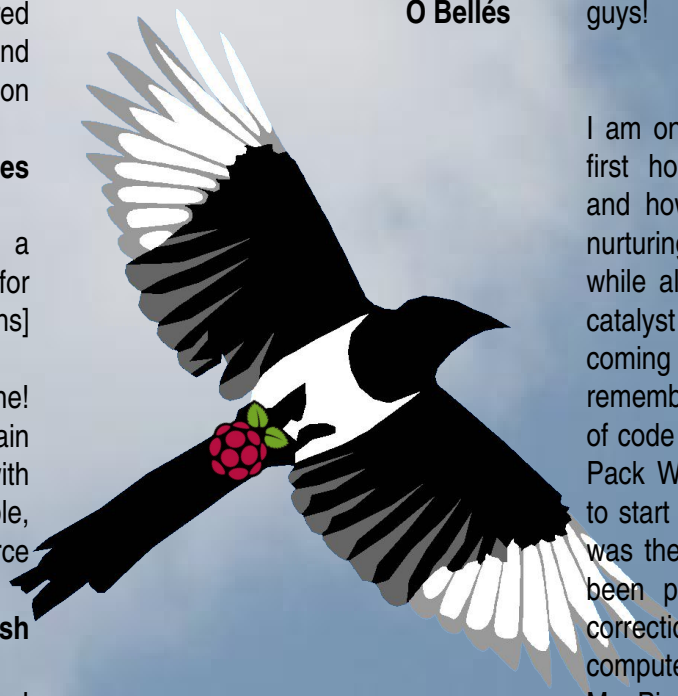
T Gomez

I am glad you are making a physical copy of these. I have yet to find a device I want to read soft copies on so this makes you even more awesome. You will go nicely on the shelf next to my physical copies of Linux Format.

Yakko TDI

Thank you for this amazing magazine .

O Bellés



As someone who cut hit computing teeth on a Sinclair MK14, I think the Pi is amazing.
I have read most issues of MagPi at least twice and am very impressed by the breadth and depth of the articles you have published. As a parent I am very impressed that most of the articles published are not game related. Please keep up the good work guys!

J Ainhirn-Williams

I am one who can remember the first home computer magazines and how important they were in nurturing our programming skills while also acting as an essential catalyst to the creative enthusiasm coming through schools. I can remember typing in line after line of code into a ZX81 only for 'Ram Pack Wobble' to leave us having to start all over again. Then there was the computer listing that had been put through full text auto correction and justification! But, computer magazines like The MagPi really are an essential ingredient in sparking young programmers off to achieve greater things. So keep up the good words.

D Lockwood

The MagPi™

editor@themagpi.com