ISSUE 04 - AUG 2012

Win a MagPi Case in our competition (Page 11)

# The MagPi

*A Magazine for Raspberry Pi Users*

# EBEN & LIZ: The Interview



http://www.themagpi.com

# The MagPi

*Welcome to Issue 4,*

*This month the team are very proud to annouce that for the first time, The MagPi has been created entirely using Raspberry Pi computers.*

*As promised, we bring you our interview with Eben & Liz from the Raspberry Pi Foundation.*

*We have some new articles, as well as our popular series, covering hardware projects, programming and other general operating tips.*

*We have received a good response to our request for volunteers. There are now a few new additions to the team, however we are still looking for more help, so please do come forth!*

*Ash Stone*

*Chief Editor of The Magpi*


**TEAM:**

**Ash Stone**
*Chief Editor / Administrator / Header*
**Jason 'Jaseman' Davies**
*Writer / Website / Page Designs*
**Meltwater**
*Writer / Photographer / Page Designs*
**Chris 'tzj' Stagg**
*Writer / Photographer / Page Designs*
**Bobby 'bredman' Redmond**
*Writer / Page Designs*
**Darren Grant**
*Writer / Page Designs*
**Lix**
*Page Designs / Graphics*

**0The0Judge0 (Matt)**
*Administrator*
**Antiloquax**
*Writer*
**W.H. Bell & D Shepley**
*Writers*
**Rob McDougall**
*Writer*
**Colin Norris**
*Editor / Graphics (C Cave Header)*
**Andrius Grigaliunas**
*Photographer*

# Contents

# IN CONTROL

## INTERFACING PROJECTS FOR BEGINNERS

PART 3

BY DARREN GRANT

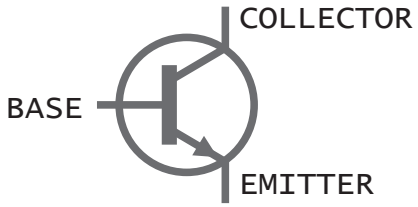
**We have already used the Raspberry Pi GPIO outputs to drive light emitting diodes, but it is not possible to directly drive anything larger because of power limitations. In this part we look at using a transistor to control devices that require more power.**

So what do we mean when we talk about the Raspberry Pi GPIO ports being low powered? Power is measured in Watts and is calculated from the available Voltage multiplied by the available current. By default the Raspberry Pi GPIO provides an output of 3.3 Volts and up to 8 milliamperes. This means that our available power output is limited to 3.3V x 0.008A = 0.0264W or 26 milliwatts. While it is possible to increase the output current to 16mA using a software setting this still only gives us an absolute maximum of 0.05W for a single GPIO pin, it is however not advisable to run at this level for any sustained period of time especially if multiple IOs are in use. So I would advise you to consider the default values the maximum.

26mW is a tiny amount of power that is inadequate for the requirements of higher powered devices such as motors and filament lamps that typically have multi-watt power requirements. However, it is possible to use the outputs to act as control signals for electronically activated switches, making it possible to control virtually anything with the right equipment. There are a number of different components that can be used as power switching devices, the one chosen will depend on the application and equipment to be controlled.

### The Transistor

We are going to be looking at a low power transistor switching circuit. With the vast number of transistors available from an electronics catalogue you may be thinking that they are complicated. However transistors can be used in two ways, either as an amplifier or as a switching device. For our purposes, using them as an electronic switch simplifies things.




NPN Transistor Symbol

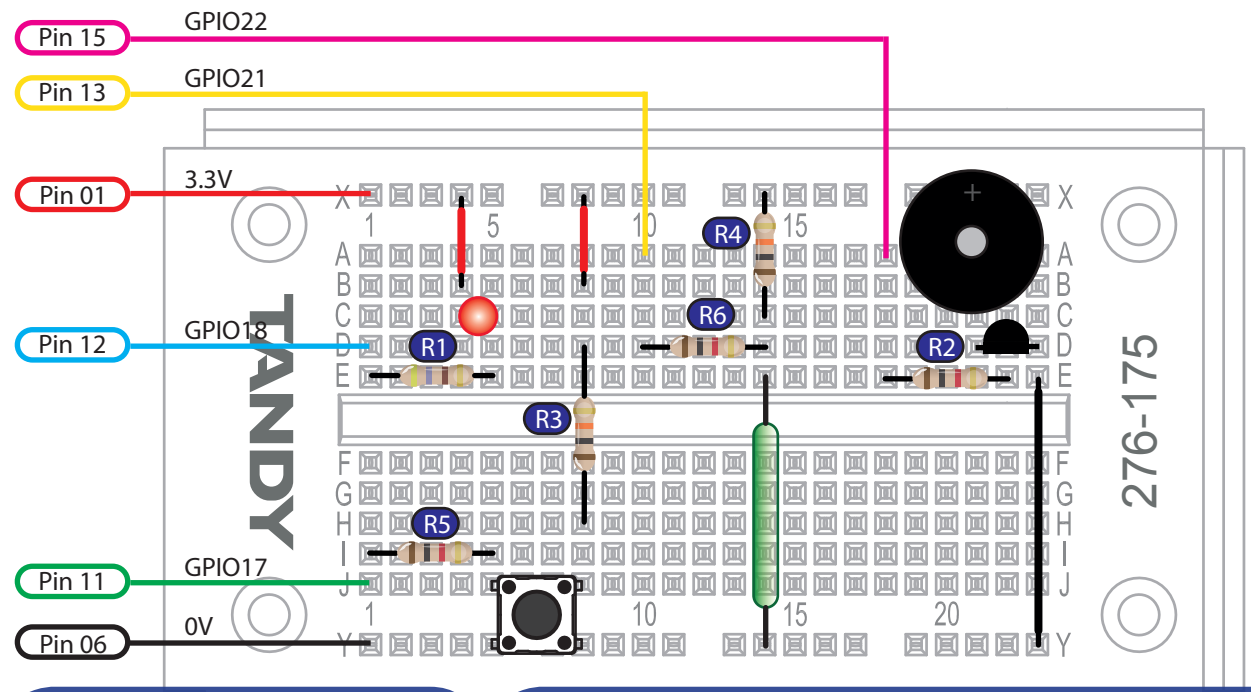
A transistor has three terminals called Collector, Base & Emitter. When a small current is applied to the Base terminal it allows a much larger current to pass between the collector and emitter terminals.

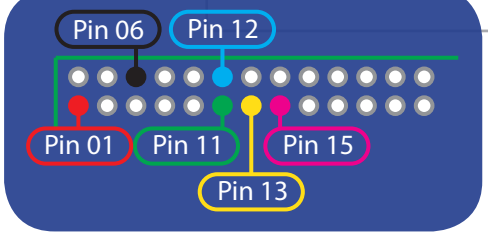
The amount of current that can be switched depends on the transistor chosen and the amount of current available to the base terminal. This is known as the DC current gain shown as $h_{FE}$ in a transistor data sheet. For the PN2222A transistor the $h_{FE}$ Value is 100, to switch a 150mA load we need only apply 1.5mA to the base, well within the capability of the Raspberry Pi GPIO and gives us the ability to switch up to 0.5W.

A typical transistor is a relatively low powered device that on its own could not be used for example to switch a 50W car headlamp but could easily switch a bank of high brightness LEDs or a small buzzer.

Transistors can only be used for DC Voltages, so can **NOT** be used for switching AC mains electricity. A key advantage of a transistor is that it as it has no moving parts it can be switched very quickly, thousands of times a second if required and has a very long operational life compared to a mechanical relay.

**IMPORTANT**
Before connecting anything to the Raspberry Pi please be aware that incorrect connections could cause damage. Please take care.



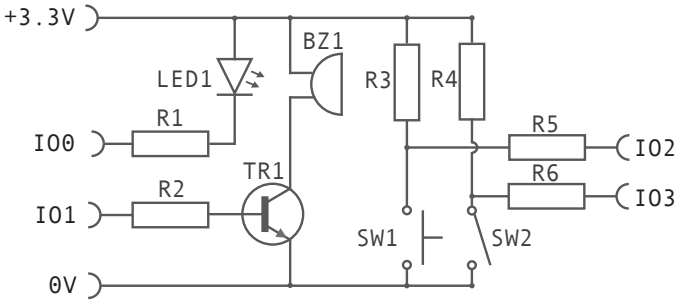

SW1 – Tactile Switch J5, J8, Y5, Y8
SW2 – Reed Switch
R1 – 470Ω Resistor D5, H5
R2 – 1kΩ Resistor I1, I5
R3 – 10kΩ Resistor D8, H8
R4 – 10KΩ Resistor
R5 – 1kΩ Resistor
R6 – 1kΩ Resistor
LED1 – Red LED C4, C5
TR1 – 2N2222 D21, D22, D23
BZ1 – Mini 3.3V Buzzer X21, B21
Wire link X4, B4
Wire link X8, B8
Wire link E23, Y23

## Alarm System Circuit

To demonstrate the use of a transistor, we will construct an alarm system that will sound a small buzzer when activated.

## Circuit Description

We will be using two outputs and two inputs. The first output is the familiar LED circuit that will be used to indicate when the alarm is armed. The second output is connected via resistor R2 to the base of the transistor, the resistor is necessary to limit the amount of current supplied to the base of the transistor so that there is just enough to switch it on. The collector of the transistor is connected to a buzzer so that it will sound when the output connected to the base of the transistor is set high. Although we could use the transistor to switch a higher voltage for example 5V, to keep things simple we are using a 3.3V buzzer so that the whole




circuit can be powered from a single voltage.

The first input is the same small push-button switch as in previous experiments, this will be used to arm and disarm the alarm. The second input is a magnetically operated reed switch. A magnet would be attached to a door and the switch to the door frame so that the switch will be opened and closed when the door is opened and closed. If you don't intend to use the circuit on a real door and prefer not to buy a reed switch for the purpose of experiment-

```
import RPi.GPIO as GPIO
import time
GPIO.setup (11, GPIO.IN)
GPIO.setup (12, GPIO.OUT)
GPIO.setup (13, GPIO.IN)
GPIO.setup (15, GPIO.OUT)

while True:
      if not GPIO.input(11):
            if GPIO.input(13):
                  print “The door is open - please close the door and try again.”
                  GPIO.output(15, True)
                        time.sleep(.3)
                        GPIO.output(15, False)
                  flash = 3
                  while flash > 0:
                              GPIO.output(12, True)
                              time.sleep(.3)
                              GPIO.output(12, False)
                              time.sleep(.3)
                              flash -= 1
            else:
                  active = ‘true’
                  activated = ‘false’
                  time.sleep(.1)
                  if GPIO.input(11):
                        print “Alarm Armed”
                        while active == ‘true’:
                              GPIO.output(12,False)
                              if not GPIO.input(11):
                                    time.sleep(.1)
                                    if GPIO.input(11):
                                          print “Alarm Disarmed”
                                          time.sleep(.1)
                                          active = ‘false’
                              if GPIO.input(13):
                                    print “**** Alarm !!! ****”
                                    activated = ‘true’
                                    GPIO.output(15, True)
                                          time.sleep(10)
                                          GPIO.output(15, False)
                                    while activated == ‘true’:
                                          if not GPIO.input(11):
                                                time.sleep(.1)
                                                if GPIO.input(11):
                                                      print “Alarm Disarmed”
                                                      time.sleep(.1)
                                                      active = ‘false’
                                                      activated = ‘false’
                                          else:
                                                GPIO.output(12, True)
                                                      time.sleep(.3)
                                                      GPIO.output(12, False)
                                                      time.sleep(.3)


      else:
            GPIO.output(12,True)
```

ing feel free to modify the circuit slightly to use another small tactile switch or whatever sort of switch you have to hand in place of the reed switch.

The buzzer and transistor must be inserted into the breadboard the correct way round. The buzzer used here has a longer leg that indicates the positive terminal that should be connected to the top. The transistor should be placed with the front (flat side) facing towards the bottom of the breadboard as shown in the illustration, the legs will need to be spread out to fit the holes in the breadboard.

**Program Description**

The main loop of the program waits for GPIO Pin 11 to go LOW when the button is pressed. We then check to see if the door is closed by checking the status of the reed switch on pin 13. If the door is open we briefly turn on the buzzer to create a short beep and flash the LED to indicate that we cannot arm the alarm. Then we return to the main loop waiting for the button to be pressed again.

If the door is closed we turn the LED on and start a loop that constantly monitors the status of the switches. If the push-button is pressed we end the monitoring loop, switching off the LED and returning to the main loop waiting for the alarm to be armed again. If however the reed switch is triggered by the door being opened we sound the buzzer and flash the LED. To avoid causing a nuisance with the buzzer it will stop after 10 seconds but the LED will remain flashing to show that the alarm was set-off until the button is pressed again to reset it. NOTE: You will need to press and hold the button to reset the alarm after activation.

**Conclusion**

We have covered how to control high power devices using a transistor. There are many other options available for switching higher power loads such as adding a second 'Power Transistor' to the circuit to create what is known as a darlington pair. Unfortunately there is not enough space in this section to cover darlington pairs and other switching methods.

If you have questions or ideas I would encourage you to get involved on the official Raspberry Pi forum to share ideas with others about your projects. ●

**Shopping List**

COMPONENTS
1x 3mm Red LED (standard brightness)
1x 470Ω Resistor
3x 1kΩ Resistor
2x 10kΩ Resistor
1x Miniature PCB Tactile Switch
1x Reed Switch + Magnet
1x 3.3V Mini Buzzer
1x 2N2222 Transistor

ACCESSORIES
1x Breadboard
6x Male-female jumper wires
1x Selection of short jumper wires

TOOLS
If you don't have them already then a set of small long nosed pliers for bending component leads and wire cutters will make construction easier.

# 3-Axis Accelerometer

**DFFICULTY: ADVANCED**

Here is a 3-axis accelerometer solution for just a few pounds. Accelerometers measure the force of acceleration, allowing them to sense movement, speed and direction. It's a great way to get some real world data and opens up a host of experiments.

**Materials:**
RaspberryPI (internet-connected)
MMA7455 Accelerometer (Available at Farnell)
Thin Single Core Wire (e.g. enamelled 0.1mm)
2.54mm dual row header socket (16 pin)
Glue
Solder
Flux (optional)
Small piece of wood/plastic to attach the MMA7455 to for ease of use.

**Tools:**
Scalpal
Hooked nose twezers
Soldering iron
x10 Microscope (optional)

For sources of enamelled wire try small transformers, loudspeaker / earphone coils, small motor windings.

A liquid flux pen helps to make a clean joint and prevent solder bridges. If you have a bridge between pads/legs with dry solder, flux can help to increase surface tension and pull solder to a pin.

## The construction:

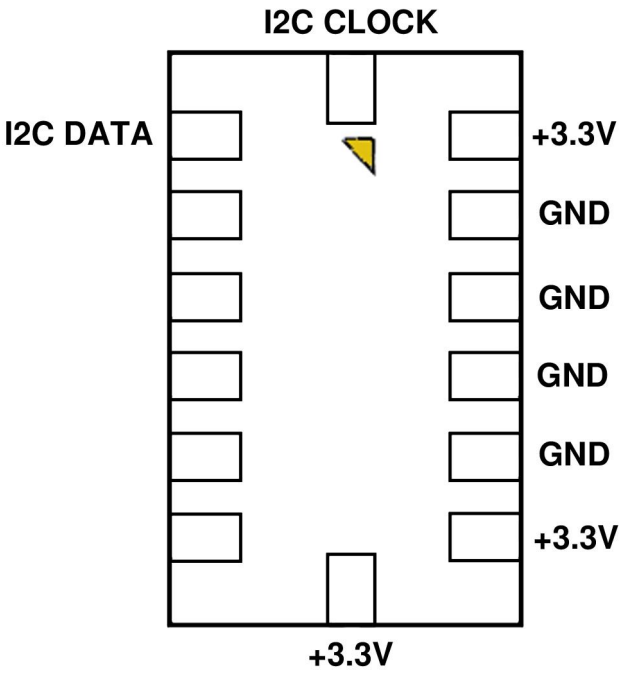
1. With a cotton bud and Isopropyl alcohol or nail polish remover clean the opposite side to the contacts of the accellerometer and the side of the small piece of wood/plastic you wish to attach the accelerometer to.

2. Attach the two cleaned surfaces using the glue.

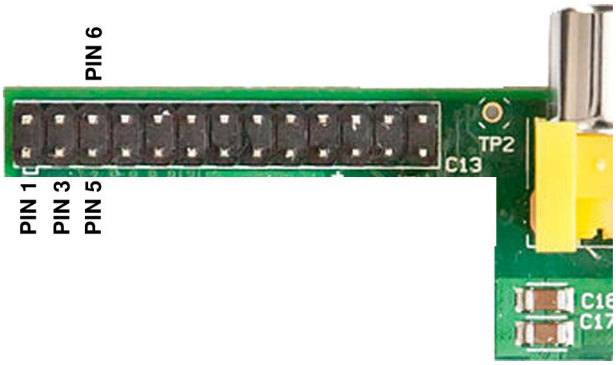
NOTE: To aid soldering, fix the whole assembly to a solid surface. I use blue tack.

3. The following table, and images, show which contact of the MMA7455 to attach to which pin of the PI GPIO via the 2.54mm header:

| MMA7455 | PI GPIO |
|---|---|
| I2C_CLOCK | PIN 5 |
| I2C_DATA | PIN 3 |
| +3V3 | PIN 1 |
| GND | PIN 6 |

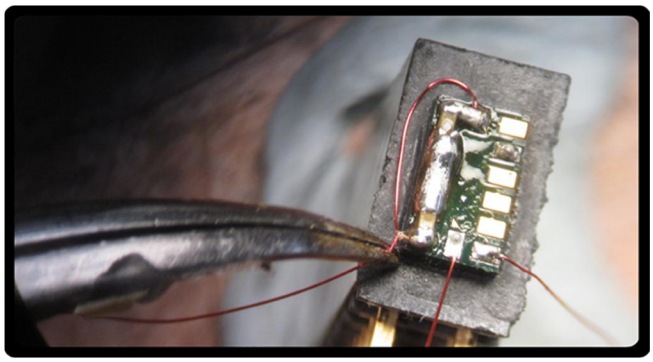



MMA7455 (BOTTOM)




**Raspbeery Pi GPIO**

The method suggested for soldering the wires is to add solder to the iron tip (wetting the tip). Then take a length of thin wire and pre-tin the end 2mm by dipping the wire end into the solder just added to the soldering iron tip which will burn off the enamel and tin coat the copper.



When under a 10X scope I have used worn tips (like old nails) because the magnification allows you to position a bad tip well. Despite this I have a file which I use to shape my tip to a fine point. This is done when the Iron is cold/off.

The downside is it corrodes fast and if you file the tip you must apply solder as it heats up otherwise it will oxidise and solder tip wont wet. A small tin of tip cleaner also helps to keep the tip oxide free.

## On the PI

The following **commands** were used on a fresh install of the debian6-19-04-2012.img:

login
"**sudo bash**"
"**wget http://www.frank-buss.de/raspberrypi/w1-test**"
"**bash w1-test**"
"**reboot**" and login
"**sudo apt-get install i2c-tools**"
"**sudo apt-get install python-smbus**"
"**sudo bash**"
"**startx**"
open lxterminal
"**leafpad /etc/modules**"
add a new line
**i2c-dev**
save and exit
"**reboot**"
login
"**ls /dev**" there should be an i2c-0 and an i2c-1 in the list
use the code on the following page and save it as MMA7455.py
open lxterminal
change directory to where this file was saved
"**python MMA7455.py**"

This should open a window and give a graph of the accelerometer value.

The python program plots a graph of the x-axis acceleration over the range -2 to +2 g

NOTE: The code is only measuring in one direction so you may need to change the position of the accelerometer for the best results.

To test the accelerometer I attatched several rubber bands together and taped an unopened tin of beans to the one end and the other to the underside/edge of a shelf. The accelerometer was blue tacked to the side of the tin on the extended wires. The tin could be pulled down and released to give a decaying sine wave on the python window.

***Continues on the next page***

```
# I2C writes =D0=0
# MMA7455 I2C address 1D (3A ,3B) write , read
# AN3745 app note for calibration
# byte read , write 1D , write address, read 1D ,DATA
# Byte write, write 1D , write address, write data.
# addresses,
# 06 XOUT8
# 07 YOUT8
# 08 ZOUT8
# 09 STATUS  D0 1=data ready
# 0A detection source
# 0F who am i
# 16 Mode Control  x1000101 measure 2gmode
# 18 Control1  D7 filter 0=62Hz,1=125Hz other 0
# 19 Control2  default 0

#!/usr/bin/python
import smbus
import time
#import graphics
import pygame

# Define a class called Accel
class Accel():
b = smbus.SMBus(0)
      # Read the value
def getValue(self):
            #address 1D, register 0x16 sets mode
self.b.write_byte_data(0x1D,0x16,0x45)

            # point at reg 6
self.b.write_byte_data(0x1D,0x06,00)

            # Read the x value
l = self.b.read_byte_data(0x1D,00)
return l

screen = pygame.display.set_mode((1000,300))
      # Create an object called MMA7455
MMA7455 = Accel()

# Get the value from it
for a in range(1000):
Aval= MMA7455.getValue()
print Aval,a
Aval=Aval+128
if (Aval >255):
Aval=Aval-255
screen.set_at((a,Aval),(255,255,255))
pygame.display.flip()
```

Hope you give it a go

***Article by Rob McDougall***

# 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 R-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

## Bristol Raspberry Jam

Where: The Bristol and Bath Science Park
When: 20th August
Tickets: Free at:
http://raspberrybristol.eventbrite.com

Join the Bristol Raspberry Jam and demonstrate any projects you are working on and express ideas you have for the future of the community with other fans and guest speakers from all over the country!

## The MagPi COMPETITION

This month The MagPi are proud to announce our first competition in partnership with PC Supplies Limited! PC Supplies are offering a fantastic set of goodies to get you up and running including the following:

Limited Edition MagPi Case
HDMI Cable
Power Supply
Audio Cable
Video Cable
GPIO Cable
Network Cable
32GB SDHC Card with Raspbian


PC SUPPLIES LIMITED

To enter visit
http://www.pcslshop.com/info/magpi

Closing date is 20/8/2012.

Winner will be notified by email. Good luck!

# Kernow Pi Launch

## *How Raspberry Pi (and lots of other goodies) are being brought into Cornish classrooms.*

***Article by: Ed Walsh, Lead Consultant for Science, Cornwall learning.***

ICT, as a school subject, doesn't have a great name in some quarters. It's not that pupils don't need to find out how to word process or prepare a presentation but it might not always, shall we say, float the boat for all teenagers.

We might consider it worthy, rather than creative. Raspberry Pi, on the other hand, is pretty cool. How does this view sit with schools? Are we going to see the curriculum turned upside down and a revolution in attitudes towards what pupils should learn about?

Cornwall has a thriving software industry. After all, with superfast broadband, if you can work anywhere, why not work there? However there was a growing sense of frustration amongst senior managers that the education system wasn't coming up with the goods.

'Many pupils leave school having been trained in word processing and other office applications" said UKNetWeb CEO Toby Parkins "but this is only half the story – and not a very interesting half. Digital technology supports a whole range of products and services."

Now, this view isn't new. However the launch of Raspberry Pi has provided a whole range of people with a rallying point. An open invitation to secondary schools resulted in dozens of people turning up to find out how they could learn what the kit does and how it can be used in the classroom. So why should schools buy into the Raspberry Pi dream?

One of the great things is that pretty quickly pupils can get to the 'take off' point, whereby they're not dependent upon being fed more information by a teacher, but can explore ideas themselves and collaborate in overcoming problems. Pupils can get really involved. These are also critical skills for the economy.

That often doesn't mean much when you're 13 - after all, how many of us, when young teenagers, took an interest in certain things because they'd help to build the economic base of the country? – but it's true. Furthermore, getting to be good at programming means getting good at logical thought, structured reasoning and problem solving. Even if you leave programming behind, those are skills with a real currency. "Computers really become interesting when you can use them to control other things" said Packet Ship CEO Paul Clark. "Years ago many young people learned programming skills on machines such as the BBC and Sinclair computers. We've lost that now and we have a chance to redress this."

So what do schools need to make this a reality? If we could drop off a box of Raspberry Pi computers at the Reception desk of a typical school and come back in a month's time, what would have happened? Well, the first thing to say is there's a lot of interest, a lot of enthusiasm and a desire to make things happen. However there are also barriers and we need to understand this. ICT teachers tend to come from a wide range of backgrounds. For someone whose heart is in, say, control technology or systems engineering, Raspberry Pi may be a dream come true; an idea whose time has come. However, if your background is in, for example, commerce and business the confidence (and interest) levels may be a bit lower. Some teachers need some support. There are also technical issues; it's no good being afraid of wires and expertise in this area is crucial. Some school network systems are designed with security higher on the agenda than access. There's also the need to be able to pass around great ideas for classroom activities and how to use them.

What's a good way of providing support then, especially as schools are at significantly different stages of development in their IT capability and teacher experience? Well, half a dozen schools have stepped forward to take up a pathfinder role.
We're setting up a support website for these people with sample teaching materials, teacher guidance, the opportunity to have technical queries answered by experts (thanks to industry support) and a means to share ideas and experiences.

We'll get together maybe once a term but lots of development will be online. The starting point will be using Raspberry Pi itself so the teachers have joined the queue to buy their own, by which time we should have some activities for them to try out. Then it'll be getting to grips with the techy side; inevitably going to be some issues there, but, as Toby said, "we're in the answers business. That's basically what we do in our business all day – find solutions to problems."

The next stage will be into the classroom and using them with pupils. We'll need ways to get pupils learning the basics pretty quickly and then starting to come up with their own ideas. It's like scaffolding – starting with lots of support and gradually removing it.

So what will it look like when it's all working? Well, for my money, pupils will be using digital technologies to solve all kinds of problems and making some of their ideas into realities. Earlier this year I met Emily Miller, the sixth form student who won the Intermediate Young Engineer award with her cot monitor. Now, this doesn't use a Raspberry Pi – it predates the release. However what really impressed me was her assertion that "being able to imagine something….and then make it a reality, now that's cool." We've now got a significant number of people with a range of skills and a desire to make some changes in the classroom. Toby Parkins: "Cornwall is well placed to play a leading role in this exciting future but pupils have to have the skills to make the most of it. Our aim is to reverse the Cornish brain drain, with students feeling that they have to move out of the county to access opportunities."

Watch this space.

MagPi Exclusive

# INTERVIEW

***This month, Eben and Liz Upton from the Raspberry Pi Foundation, kindly put aside some time for an informal interview with the MagPi team. Despite the couple having not eaten and coming directly out of another interview, and after a long day, they were as bouncy, fun and energetic as ever.***

Q1: **Do you feel that you have already achieved your goals, and if not, what still needs to be done?**

Eben: We have not achieved our goals. Specifically, we have achieved NONE of our formal goals, which are all around education. It's a long-duration plan! I will know we achieved our goals when I am drawing a pension and the country still has an economy that can pay for it!

Liz: Blimey, no, the competition [see the Raspberry Pi Homepage for details of the Summer Code Contest] is a really good start, but we're not going to be any way into achieving our goals for a year - and we won't know if we have for another 5 years or so. There's also a load of supporting stuff out there as you guys know, we can't believe how helpful the MagPi material is. CAS are currently doing a free manual for teachers (or students, or anyone else who is interested) - I was just talking to them today about illustrations. And it's been completely brilliant to watch the software community grow so fast; we still can't believe it.

Q2: **What is your attitude towards Commercial Pi Projects? Some projects might get very large, and need a large quantity of Pi's. I've searched the forum, but haven't found conclusive proof that you approve of commercial projects?**

Liz: We totally approve of them.

Eben: More than that; we think they're completely vital to the future success of Raspberry Pi. People shouldn't think that just because we're a charity we're not in favour of people making money. You should totally quote Peter Mandelson. "I am intensely relaxed about people getting filthy rich."

**MagPi: [Note - following this interview, the order limits have been lifted and multiple Raspberry Pis can now be ordered.]**

Q3: **With the benefit of hindsight, is there anything you would have done differently were you to design the model B or in your approach the project in general?**

Eben: I think we might have tried to be more ready before the news got out! The USB port would have been better off inline - in the grand scheme of things, that's not a disaster. It would have been nice if we could have started at the first availability of the chip rather than 9 months after.

Liz: But on the whole, we're really happy with how things have worked out. For a tiny charity with no full-time workers (except me), I think we've done amazing things.

Eben: Liz is warned that she may not form a union.

Q4: **When is the camera module hoped for? What would you envisage the next project to be?**

Liz: the next thing after that will be a DSI to LVDS bridge, that display project and the camera project are the only two Foundation projects. We're hoping for the camera in the next 3 months or so, we're knee-deep in active technical and commercial discussions, but the display timing is less well defined.

Eben: But you know what we're like with the ability to hit date targets...

Q5: **Do you have any more projects in the pipeline now that the Raspberry Pi is out there?**

Eben: For the love of God, no! I plan on a quiet and easy death some time, but that's as far as I've got.

Liz: Eben is pointing at our giant year planner and saying: "We should keep that when we're done. It will remind us never to do a hardware start up again."

**MagPi: We hope you're collecting air miles as well!**

Liz: You would not believe the air travel. It makes us feel very guilty about the...

Eben: Carbon pigs!

Liz: Eben is referring to a website we typed our energy choices into. It gave you a pig for every bad thing you did to the environment. They ran out of pigs!

Eben: So for an experiment, we reduced our air travel on it by a factor of 10 and they still ran out of pigs.

**MagPi: No plans for world domination then?**

Eben: Well, we could talk about some of the other plans, right...

Liz: *Cough*, I'm saying nothing. *Lips sealed*....

Eben: All I'm saying is: don't you think school textbooks are a bit of a racket?

***(continued over page...)***

**MagPi: Interesting...**

Q6: **We saw Eben was the king of rap, what music do you both like?**

Eben: Our Joint favourite band probably The Killers, Liz loves Muse and I like Biffy Clyro. We do have a sneaky admiration for arrogant trance though.

Liz: No we don't!

Eben: You bought that Titanium song!

Liz: We saw the Killers in a hometown gig in Vegas on our wedding anniversary a few years ago - best anniversary EVER, and we had steak. And I used to be a choral scholar at St Martin in the Fields (Eben is laughing hilariously at my characterisation of that as "pro chorister").

**MagPi: They will probably have an increase in sales if we print that, number one in August!**

Eben: And we listen to a *lot* of classical music, usually when plonking trolls on the forums. Bach is an inspiration to us all, we seem to believe that doing awesome stuff has to come with pain; with being hard and angst. But Bach was...well, Bach - and a totally normal guy. Who liked coffee, jokes about flatulence (as did Mozart) and taking the mick out of his orchestra.

Q7: **Do you know of the current USB / Ethernet issue? This is where low powered devices knock out the Ethernet connection when running lxde/gdm, which in turn causes the lxterminal to hang. It could be a Ethernet driver issue... Would this be sorted before the educational release?**

Eben: There's work going on literally as we speak (Dom's working on it in the evenings) to improve the USB stack. Interestingly, the Model A is really useful in that respect, because it allows you to insert a USB protocol analyser between 2835 and the hub.

Q8: **Could a spare USB2 link on the Model B LAN/IO chip be taken to the mini-USB power plug for the RasPi to then be used as a *client* USB device for direct connection to a host such as a PC? (USB3 could supply the higher current and are backwards compatible with USB2)**

Eben: There is no spare USB2 link! However...it may be feasible to use the Model A as a USB device; since there's no hub in the way.

Q9: **Is it possible with the current model B board to emulate the Sega Megadrive/Super Nintendo consoles on Debian squeeze?**

Liz: Absolutely. I've seen SNES emulators in the wild; don't think we've seen any Megadrive emulators yet, but it's perfectly doable. Although I have no idea why you'd want something that

can play Echo the Dolphin. Sonic all the way!

Q10: **When people look back on your life, do you think Raspberry Pi will be the achievement that stands out?**

Eben: No. It will be for the unification of general relativity and quantum mechanics. For somebody who bailed out of Physics for CompSci in his second year at university, I remain very optimistic.

Liz: Did I mention I make a *mean* flapjack?

Q11: **We heard the next Mars rover will have a Raspberry Pi on board [Joke] ;)**

Eben: Pi on Mars. God, that'd be cool. I am retiring to Mars with Elon Musk, I have a big boy-crush on him.

Liz: Eben is forbidden to go to Mars if it becomes possible if he does not take me.

Q12: **Is the foundation working on any education packs for schools (I know you said CAS are)?**

Liz: Not directly. That kind of stuff is done by our partners and the community (there are only a very few of us with very limited time). I do have a meeting with an exam board on Thursday which is hoping to use Raspberry Pi in its computing provision. We do as much support as we can, but we hope to see materials come from the people we're working with; rather than from us directly.

Q13: **What can the community do to make the Rasp-Pi easy for a middle school science teacher to pick up, giving real hands-on for kids?**

Eben: Really careful, worked examples with material for the teacher and the student.

Liz: I totally agree; a lot of the materials we see are either for the teacher or the student, it's incredibly helpful for teachers to have both.

Q14: **How many Raspberry Pis have you sold so far?**

Liz: There are more than 200k in the wild now, and there are orders for...well more than that, but Eben is saying I can't say how many.

Q15: **Are you surprised by the variation of projects that people are using the Pi for? (Beyond programming and interfacing) and which is your favourite?**

Eben: No, and FishPi and rockets! Obviously.

Liz: Yes (I have a lower expectation from human-kind than Eben does...). I am a total sucker for the pro kitchen and brewery applications we're seeing. A wall of sous vide cookers! Fridge temp control! Sealing beef under vacuum and BEER, of course!

***(continued over page...)***

Q16: **What is the lowest minimum amps required to run PI?**

Liz: Minimum amps for a model B is about 0.7 Amps

Eben: A model A is much less than that, around 0.2 Amps I think.

Q17: **What do you imagine will be the effect of the Raspberry Pi on the Arduino?**

Eben: It's a $25 host PC for Arduino

Liz: We think it's a good thing for Arduino. It should mean that their sales increase; we're very keen to see people use them together. We wish people would stop seeing Raspberry Pi as some sort of existential threat to Arduino. It really isn't.

Q18: **A majority of Raspberry Pi owners / buyers seem to be doing this to get a very cheap 1080p capable media player rather than a programming / experimenting computer that the Raspberry Pi was originally meant to be. What are your feelings about this?**

Eben: It's great news. We're not some bunch of use-case fascists. We don't care what people do with it as long as people do *stuff* with it. You've got to remember that our goal for the UK is to get 1000 new CS students per year. That's a really modest goal...

Liz: It doesn't matter if most of them end up as media players. Some of those media players are going to end up in the hands of kids who find they can do other stuff with them. That number can be tiny, but every one of those kids is a win. So the more end up as media players, the more end up in living rooms, the more kids see them, the more kids experiment - it's all good for the project. We're already seeing a lot of people on Twitter saying "I saw one at my friend's house; I'm going to order one." It's exactly what we were hoping for.

Q19: **What have you gained from the experience of being involved in this project and what has surprised / pleased / disappointed you? (Apart from the TROLLS)**

Eben: I've learned that open source is actually real. The bazaar pitch is real; if you make something like Raspberry Pi, then you can rely on people chipping in; that's been amazing.

Liz: I'm also amazed at how plastic kids can be. It's brilliant watching them with a Raspberry Pi and seeing them just get on with it.

***Big Thanks to Eben & Liz for supporting us!***

***The MagPi Team***

# The MagPi Letter of the Month!

Hi

First, congratulations on producing an excellent resource for the RPi community, you should be justly proud of yourselves for coming up with such a professional and useful publication.

Now, my comment is with regard to the article on the GPIO by Mike G8NXD in issue 3 - he warns about the lack of protection offered by the GPIO connectors, rightly so. I feel that it is important to point out that the GPIO pins only offer 3.3 volt CONTROL signals, and aren't intended to drive ANY sort of load.

If you want to connect something external to a GPIO pin, even something like a LED, it ideally needs to be connected via some sort of buffer, so that the GPIO signal is used just to switch the buffer, not to take any of the load. This becomes more important as you add extra LEDs, especially if they are the high-brightness type.

A simple buffer can be made from a cheap NPN transistor and a couple of resistors, and should only cost a few pence - for example, plastic-packaged versions of the 2N2222A NPN transistor may be bought from Ebay for 5p each - a small price to pay to protect your RPi!

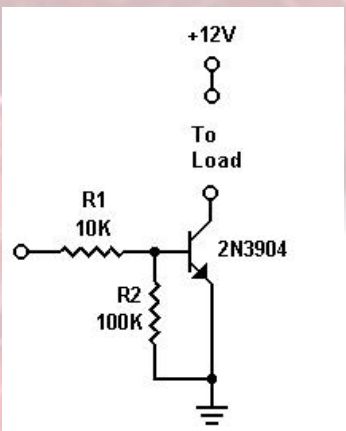


Figure 2

A good article on transistors as switches may be found here http://www.rason.org/Projects/transwit/transwit.htm - I've copied the following diagram from there, which gives the general idea :

This example shows a 12v supply and a 2N3904 transistor, but a 2N2222 can be substituted. For operation from 5v, I'd use lower resistor values, 3K3 for R1, 33K for R2. This will happily switch a 30mA LED, in fact the 2N2222 will handle loads up to 600mA although you'd need to calculate different values for R1 and R2 to match the load - full details in the quoted article.

Finally, don't take the power from the GPIO pins, use a separate source - however, you will need to connect the GPIO ground to the buffer ground and, of course, the required GPIO control signal pin to the buffer input.

Hope this is useful, and saves someone from the expense - and grief - of 'Magic Smoke'!

Best Regards

J Ellerington

(Figure 2 taken from http://www.rason.org/Projects/transwit/transwit.htm)

# HOW TO CUSTOMISE YOUR LXDE MENU

***Arranging the menus in LXDE is a little more of a challenge than on Windows...***

As you start to install more and more packages the menu becomes unwieldy.

Before starting to alter the menu, we need to understand the locations of certain configuration files.

## .desktop files

.desktop files are similar to shortcuts in Windows. They are generally stored in /usr/share/applications. These are just text files that can be edited.

Open LXTerminal and type:

```
cd /usr/share/applications
ls
```

You will see a list of .desktop files that correlate with items on your menu.

Let's take a peek inside one of them. Type:

```
sudo leafpad gpicview.desktop
```

You will see a list of variables. One of the variables is called Name=Image Viewer. This is the name that will appear on the Menu. This tells us that this is the shortcut file for the Image Viewer application (which is actually called gpicview). There are also translated versions of the name listed below. These will be the displayed name if you have altered your LXDE language settings. Other important variables are Categories, Exec and Icon. These define the category that the application falls under, the command that is run when you select the menu item, and the name of it's icon image. Let's close Leafpad for now.

## .directory files

.directory files are branches of your menu. These are generally stored in /usr/share/desktop-directories.

In the LX Terminal type:

```
cd /usr/share/desktop-directories
ls
```

The files that are named lxde-something.directory are the ones of interest.

Let's have a look in one of them:

```
sudo leafpad lxde-education.directory
```

You'll notice this is pretty similar to the format of the .desktop files. There are variables for Name in various languages, the icon image name and Type=Directory. Close Leafpad.

Let's create our own directory file:

```
sudo leafpad lxde-magpi.directory
```

This gives us a new empty text file. Type the following into it:

```
[Desktop Entry]
Name=The MagPi
Name[en_GB]=The MagPi
Comment=Desktop accessories
Comment[en_GB]=Desktop accessories
Icon=applications-accessories
Type=Directory
```

Click File > Save, and then close Leafpad.

## lxde-applications.menu file

The lxde-applications.menu file can be found in /etc/xdg/menus. This is an XML formatted text file that contains information about how menu items should be displayed.

At the LXTerminal, type:

```
cd /etc/xdg/menus
sudo leafpad lxde-applications.menu
```

Near the top of the file you should see a line that says:

```
<!-- Accessories submenu -->
```

Make a gap just above that and add the following section:

```
<!-- MagPi submenu -->
<Menu>
<Name>The MagPi</Name>
<Directory>lxde-
magpi.directory</Directory>
<Include>
<Filename>gpicview.desktop</Filename>
</Include>
</Menu> <!-- End The MagPi -->
```

Click File > Save.

Have a look on your LXDE Menu. You should see a new branch entry called 'The MagPi' with the Image Viewer listed within.

Now edit the accessories section as shown here:

```
<!-- Accessories submenu -->
<Menu>
<Name>Accessories</Name>
<Directory>lxde-
utility.directory</Directory>
<Include>
<And>
<Category>Utility</Category>
<!-- Accessibility spec must have either
the Utility or Settings category, and we
display an accessibility submenu already
for the ones that do not have Settings,
so don't display accessibility
applications here -->

<Not><Category>Accessibility</Category><
/Not>
<Not><Category>System</Category></Not>
<Not><Filename>gpicview.desktop</Filenam
e></Not>
<Not><Filename>pcmanfm.desktop</Filename
></Not>
</And>
</Include>
</Menu> <!-- End Accessories -->
```

In the <Include><And><Not><Filename></Filename></Not></And></Include> section of the Accessories Menu we have declared that the gpicview.desktop (Image Viewer) and the pcmanfm.desktop (File Manager) should not be displayed on the Accessories branch of the menu.

Save the Leafpad file and have a look on the menu to confirm that both Image Viewer and File Manager are now gone from the Accessories menu.

Let's put the File Manager into our MagPi menu branch, by editing the <!-- MagPi submenu --> section as shown:

```
<!-- MagPi submenu -->
<Menu>
<Name>The MagPi</Name>
<Directory>lxde-
magpi.directory</Directory>
<Include>
<Filename>gpicview.desktop</Filename>
<Filename>pcmanfm.desktop</Filename>
</Include>
</Menu> <!-- End The MagPi -->
```

With some detective work you should be able to identify which .desktop files correspond to the items on the menu.

Using the methods demonstrated above, you should be able to add, remove or move items around on your LXDE menu, as well as adding new branches to the menu.

**A WORD OF CAUTION: Incorrect XML code could leave you with a corrupted or missing LXDE menu**

***Article by Jaseman***

# Command Line Clinic


## In this month's article, we will see how to edit configuration files. Configuration files allow you to change the behaviour of a Linux computer.

One of the most powerful aspects of Linux is the fact that you have the ability to configure the behaviour of any part of the operating system. Some of the most common parameters can be configured by using graphical tools. However, there are thousands of configurable parameters available and these are not all covered by the graphical tools, so it is very likely that you will need to use the command line to change a parameter. If you come from a Windows background, you can think of editing configuration files as being very similar to editing the Windows registry.

In general, the steps are always the same. You will need to find the correct configuration file, edit the file, and tell the operating system to reload the configuration file.

The first step is to find the correct configuration file. For most common tasks, you will find the configuration files in the /etc directory. It is very unlikely that you will pick a file at random and to start editing, it is much more likely that you will follow a guide or tutorial and this will tell you exactly which file you should be editing. So we need a simple example.

As an example, let us configure Linux to automatically load the sound module. This is very useful because it means that programs can send sounds to the monitor. By default, te sound module is not loaded in most Raspberry Pi distributions, and a lot of users would prefer if it was loaded automatically.

To load the sound module, we need to edit the file which configures which kernel modules should be automatically loaded. This file is called /etc/modules.

Before you edit a file, it is a good idea to make a backup just in case you make a mistake. To make a copy of the file /etc/modules, you would use the command

```
sudo cp /etc/modules /etc/modules.old
```

This copies the file /etc/modules to a file called /etc/modules.old. Note that the sudo command executes the cp command as a

## What text belongs in each file?

Each configuration file is different and there are special rules on what is expected in each file. Usually you will be following a guide which will tell you exactly what you should type. But if you want to learn more about each configuration file, you can read the relevant manual page.

To do this, enter the man command followed by the name of the file. For example, to see the manual for the /etc/modules file, enter the command

```
man modules
```

Press the up/down arrows to read the manual, press q to quit.

superuser, this is needed because an average user does not have the right to change anything in the /etc directory.

If you make a real mess of the original file, you can copy the backup file to the original to undo any damage by using the command

```
sudo cp /etc/modules.old /etc/modules
```

Now it is time to actually edit the configuration file. There are lots of command-line editors available, we will use the nano editor which is suitable for beginners. To edit the file use the command

```
sudo nano /etc/modules
```

You will find yourself in a simple text editor, you can move around with the arrow keys. As an example, we will add the sound module snd_bcm2835 to the file so that sound will be enabled automatically. In this file, the order of the entries is not important, so we can add the following text to the end of the file

```
# Enable sound
snd_bcm2835
```

Note that in some Linux distributions, you may find that snd_bcm2835 has already been enabled, in this case you will not be able to follow this example.

In most configuration files, a line starting with # is a comment. It is a good idea to add a comment decribing any changes that you make. It can be very easy to forget the changes that you have made, so try to make the comment as meaningful as possible. If you are following an online guide, it is a good idea to put the web address of the guide into the comment so that you can find why you made the change.

To save the file, press the ctrl and o keys. To quit the editor, press the ctrl and x keys.

After saving the file, it is a good idea to look at the contents of the file to make sure that the editing was successful. Use the cat or the less command. If using the less command, use the q key to quit.

```
less /etc/modules
```

Just editing the configuration file is not enough, you must tell the operating system to read the new configuration file. If you are following a guide, the guide will tell you if you need to issue a command, usually to restart a program or to restart the complete system. In this example, we need to restart the complete system to re-read the /etc/modules file, so we need to use the command

```
sudo reboot
```

That's it, you have now successfully changed a Linux configuration file. The same general principles can be used to edit any configuration file.

## A quick way to type long filenames

You may find that you need to type some very long filenames. These can be difficult to type and Linux is not tolerant if you make a spelling mistake. Luckily, there is some help available.

When you are halfway through typing a filename, you can press the tab key to automatically complete the end of the filename. If the filename is not completed automatically, press the tab key again to see the choices available, and keep typing.

To see how this works, try typing

```
sudo nano /etc/mod
```

and then press the tab key twice. Then press the u key and press the tab key again.

# Tutorial 2 - Variables, conditions and loops.

Before introducing this month's tutorial, let us take a quick look at the solution to last month's challenge problem:

## Challenge solution

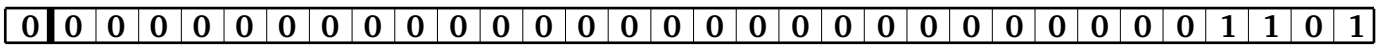
```
#include <stdio.h>
int main()
{
  int i = 100, j = 9, k;
  i = i/10;
  k = i - j;
  printf("Well done this program compiles.\n");
  printf("%d- %d = %d\n",i,j,k);
  return 0;
}
```

Did you get them all right? Rather than trying to find all of the errors by eye, it is often a good idea to use the compiler to find them. If programs have been well designed, the compiler can be a helpful tool to spot typing mistakes.

Programming languages offer different ways of writing similar structures. The languages Scratch, Python, and C, all allow variables, logic conditions, and loops to be declared.

### *Variables*

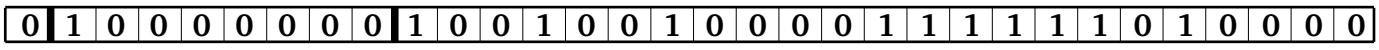
There are two categories of numeric variables in C, (i) whole numbers (e.g. `int`) and (ii) floating point numbers (e.g. `float`). When a variable is declared, a block of memory is allocated to the variable. The amount of memory allocated to an `int` and `float` is the same, but the way the number is stored is different. An `int` is stored in memory as a sign bit and a 31bit value. For example,




is 13 in decimal. From the right to the left, the bits correspond to $2^0$, $2^1$,..., $2^{30}$. The sign bit is on the far left. The number thirteen is formed from the sum of the bits. The largest number which can be stored in an int is $2^{31}$-1, which is 2147483647. If one more is added to this number, the number becomes -2147483648. This happens because the sign bit is set. Here is a simple test program,

```
#include <stdio.h>
int main()
{
  int i = 2147483647;   /* Assign (2^31)-1 to i */
  i++;   /* Increment i by one */
  printf("%d\n",i);   /* Print the value of i */
  return 0;   /* Return success to the operating system. */
}
```

An `int` cannot be used to store a fraction of a number. Fractional or floating-point numbers can be stored in a `float`. Unlike an int, a float has three pieces: a sign, an exponent and a mantissa. For example,




is 3.14159 in decimal. The sign bit is on the far left, the exponent bits are in the middle and the mantissa bits are on the far right. The number is formed by the multiplication: `sign x` $2^{\text{exponent}}$ `x manissa`

The value of the exponent is stored in the same way as the int. However, the meaning of the mantissa bits is different: from left to right the bits correspond to $2^0$,$2^{-1}$,$2^{-2}$,$2^{-3}$, etc.. If a floating-point number becomes too big, it will become not-a-number or infinity. If the precision bits in the mantissa are all used additional precision will be lost. This loss of precision information is called a floating-point error. In the example,

```
#include <stdio.h>
int main()
{
  float f = 3.14159;   /* Assign the approximate value of PI. */
  f+=0.0000001;   /* Add a very small number to it. */
  printf("%f\n",f);   /* Print the value stored in f. */
  return 0;   /* Return success to the operating system */
}
```

the addition of 0.0000001 to the float f is lost.

Right, lets get back to some simple mathematics. The division of `float` and `int` variables behaves differently. An integer division will always be rounded down to the nearest `int`, whereas a `float` will be rounded to the precision of the mantissa.

```
#include <stdio.h>
int main()
{
  int i = 3;   /* Declare an int and assign it the value three. */
  float f = 3.;   /* Declare a float and assign it the value three. */
  i /= 2;   /* Divide i by two and assign the result to i.*/
  f /= 2.;   /* Divide f by two and assign the result to f. */
  printf("int (3/2) = %d, float (3./2.) = %f\n",i,f);   /* Print the result. */
  return 0; /* Return success to the operating system. */
}
```

The result of this program is

```
 int (3/2) = 1, float (3./2.) = 1.500000
```

Characters are stored in a similar way as an `int`. A `char` is a signed 8-bit number. It can be used to print the associated ASCII character or the position of the ASCII character within the table.

```
#include <stdio.h>
int main()
{
  char c = 'a';   /* Declare a character c and assign it 'a' */
  printf("The character \'%c\' has the value %d\n",c,c);   /* Print the value. */
  return 0;   /* Return success to the operating system. */
}
```

A string is represented in memory as a series of sequential characters. A series of sequential values of the same type is called an array.

## *Arrays*

An array of ten integers can be declared by,

```
 int array[10];
```

Each element an array can be accessed using the corresponding index. Similar to many other languages, these indices start from zero and have values up to N-1 where N is the size of the array. When an array is declared, a sequential block of memory is allocated. The values in each array index may or may not be zero. Therefore, it is important to initialise each array element with a value. Arrays may be declared and assigned values at the same time,

```
int array[10] = {1,2,3,4,5,6,7,8,9,10}
```

or each element of the array can be initialised individually.

```
#include <stdio.h>
int main()
{
  /* Create an array with four element and assign values. */
  float fourVector[4]={1.0, 2.0, 0.5, 2.292};
  /* Print the number of elements in the array. */
  printf("There are %ld elements\n",sizeof(fourVector)/sizeof(float));
  printf("fourVector[3]=%f\n",fourVector[3]); /* Print the fourth element.*/
  return 0;   /* Return success to the operating system. */
}
```

In this example, the `sizeof` command returns the size of the array as a whole. Therefore, dividing this by `sizeof(float)` returns the number of elements in the array.

A string of nine characters can be stored in a character array of ten characters,

```
char str[10];
```

The tenth element is used to store the string terminating character `'\0'`. An array of strings can be stored as an array of an array of characters. Here is a simple string example,

```
#include <stdio.h>
int main()
{
  char str[50]="A test string";   /* Create a 50 character string */
  printf("\"%s\"\n",str);   /* Print the string. */
  printf("The third character = \'%c\'\n",str[2]);
  return 0;   /* Return success to the operating system. */
}
```

## *Conditions*

Many different conditions can be expressed in the C language. These conditions can be used within if-else statements, switch, or within loops. A summary of logic conditions is provided in the table below,

| Condition | Meaning | Condition | Meaning |
|---|---|---|---|
| a == b | a 'equal' b | ! a | 'not' a |
| a != b | a 'not equal' b | a \| \| b | a 'or' b |
| a > b | a 'greater than' b | a ^ b | a 'exclusive or' b |
| a <= b | a 'less than or equal' b | a && b | a 'and' b |

These conditions can be combined together to produce more complicated conditions. When several conditions are combined together it is important to use parentheses, to set the order in which the conditions are tested. For example, `(a || b) && c` is not the same as `a || (b && c)`, since the logic condition within the parentheses is evaluated first.

Conditions can be used within `if else` statements to choose a particular outcome,

```
#include <stdio.h>
int main()
{
  int apples = 20;  /* Start with twenty apples. */
  float cost, costPerApple = 0.20;  /* Set the price. */
  printf("How many apples would you like to buy?\n");  /* Ask the customer */
  scanf("%d",&apples); /* Read the number of apples needed. */
  if(apples > 20) {   /* Check if there are enough apples left. */
    printf("Sorry we only have 20 apples left.\n"); /* Apologise. */
  }
  else {
    cost = costPerApple*apples;   /* Caculate the total cost. */
    printf("That will be %.2f pounds please.\n",cost);   /* Ask for the money. */
  }
  return 0;
}
```

## *Loops*

Repetitive operations, such as accessing array elements or performing the same mathematic operation several times can be more efficiently implemented using loops. C offers three main loop types: `for`, `while`, and `do while`. Starting with the `for` loop,

```
#include <stdio.h>
int main()
{
  int i;
  for(i=0;i<10;i++) { /* Loop from zero to nine. */
    printf(" i = %d\n",i);  /* Print the value of i. */
  }
  return 0;
}
```

Between the paratheses of the for loop there are three sections: `(initialisation; condition; incremention)`. The initialisation happens once at the start of the loop. Then the loop continues until the condition is false. Each loop executes the compound statement within the `{}` brackets and then executes the incremention arguments.

## Challenge problem

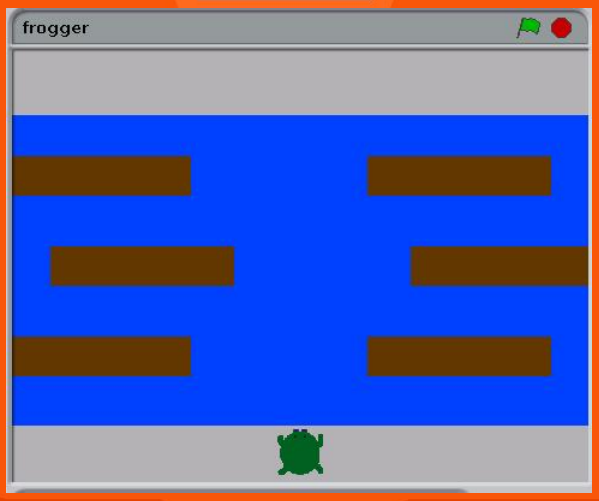
Write a program to calculate the triangular number series. Ask the user how many terms in the series should be calculated. Then use a for loop to calculate each term up to this limit. Triangular numbers can be calculated user a sum. For example,

```
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
```

where the triangular number series is 1,3,6... Therefore, try using a counter and sum variable within the loop. The solution to the problem will be given next time.

***Article by W. H. Bell & D. Shepley***

# THE SCRATCH PATCH




In this article we will make a very simple game in Scratch.

I hope many MagPi readers will enter the Raspberry Pi foundation's Summer Programming competition. Even if you haven't made many games in Scratch before, you'll soon be Scratching with the best of them!

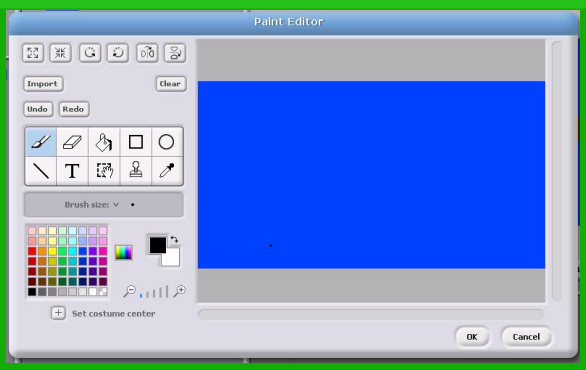
You can download this project from:
http://scratch.mit.edu/forums/
(my username is "racypy").

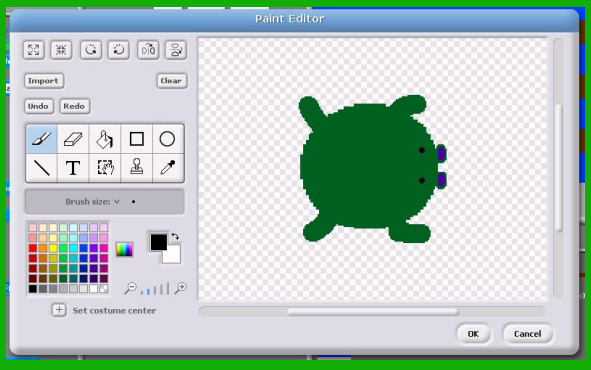
The first thing to do is delete the cat sprite, because we won't be using that. Right click on it and choose "delete".

Next, we need to create a background. To do this, click on the white square in the "stage" area on the bottom right of the screen.

Then click on "backgrounds" in the centre panel and use the built-in paint application to make something like this.

Next you need a sprite that looks something like a frog and six identical "logs". Click on the star and paint-brush in the "New Sprite" area, to create your sprites.



## Frog Scripts

These colours should match those of your logs and the banks of the river.

**Scratch On!**

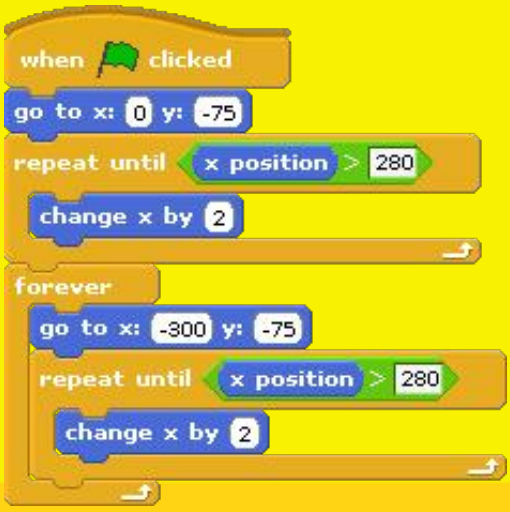


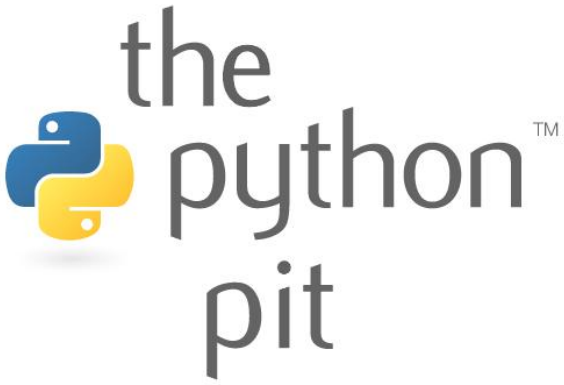
These are the scripts I used for my frog. The numbers referring to the positions of things may be different depending on how you drew the background. I've used some sounds I found online, but you could use the ones that come with Scratch - you'll need to import them first. Click on "Sounds" in the centre window.

Now you'll need six logs, each with a script. You make the logs by choosing a new sprite and drawing it, just as we did with the frog. Below you can see the first pair of scripts. You'll need to experiment with changing the y co-ordinates for the other four logs until you get them just right.

I hope you find this example helpful and go on to make some fun games yourself.

### Log Scripts (you'll need one for each log)




**This month's example game demonstrates how to capture and use keydown/keyup events as well as introducing music and sound. It also uses some of the techniques we have shown in previous examples. You will need to download some .png images and .wav audio files.**

arm_linux.png, fish.png, burp. wav, toy.wav and tune.wav are available in zip file which can be downloaded from the following link:
http://www.themagpi.com/files/issue4/fish.zip

or alternatively from:
http://ubuntuone.com/1Fz4NY0P5l9DUv07YxkHId

```
# Fish Game

# By antiloquax & Jaseman - 18th July 2012

import pygame, random
from pygame.locals import *
pygame.init()
clock = pygame.time.Clock()

screen = pygame.display.set_mode([600,400])
pygame.display.set_caption("Fish Game")

music = pygame.mixer.Sound("tune.wav")
music.play(-1)
toy = pygame.mixer.Sound("toy.wav")
burp  = pygame.mixer.Sound("burp.wav")

tux = pygame.image.load("arm_linux.png").convert_alpha()
fish = pygame.image.load("fish.png").convert_alpha()

tx=280; ty=180; txd=0; tyd=0
bx=600; by=-15
fx=600; fy=random.randint(0,370)
bkcol=[127,212,255]
lives=3; score=0; run = True;

# Draw the surfaces for Tux, Fish and Ball
tuxsurf = pygame.Surface((60,70))
tuxsurf.fill(bkcol)
tuxsurf.set_colorkey(bkcol)
tuxsurf.blit(tux,[0,0])
fishsurf = pygame.Surface((35,30))
fishsurf.fill(bkcol)
fishsurf.set_colorkey(bkcol)
fishsurf.blit(fish,[0,0])
ballsurf = pygame.Surface((60,60))
ballsurf.fill(bkcol)
ballsurf.set_colorkey(bkcol)
ballsurf.set_alpha(128)
pygame.draw.circle(ballsurf,[255,255,255],[30,30],30)
```

```
while run:
      for event in pygame.event.get():
            if event.type == pygame.QUIT: run = False
            if event.type == pygame.KEYDOWN:
                  if event.key == pygame.K_LEFT: txd=-4
                  if event.key == pygame.K_RIGHT: txd=4
                  if event.key == pygame.K_UP: tyd=-4
                  if event.key == pygame.K_DOWN: tyd=4
            if event.type == pygame.KEYUP:
                  txd=0; tyd=0
      tx+=txd; ty+=tyd

      # This part stops Tux from leaving the edges of the screen
      if tx<= 0: tx=0
      if tx>=540: tx=540
      if ty<=0: ty=0
      if ty>=330: ty=330

      # Make the ball chase Tux
      if bx>=tx: bx=bx-1
      else: bx=bx+1
      if by>=ty: by=by-1
      else: by=by+1
      fx=fx-4
      if fx<=-10: fx=600; fy=random.randint(0,370)

      # Collision Detection (Tux & Fish, Tux & Ball)
      if fx<=tx+50 and fx>=tx and fy>=ty-30 and fy<=ty+70:
            toy.play(); fx=600;fy=random.randint(0,370); score +=1
      if bx<=tx+40 and bx>=tx-40 and by>=ty-50 and by<=ty+60:
            burp.play(); bx=600; by=-15; lives -=1; tx=280; ty=180

      screen.fill(bkcol)
      screen.blit(tuxsurf,[tx,ty])
      screen.blit(fishsurf,[fx,fy])
      screen.blit(ballsurf,[bx,by])
      font = pygame.font.Font(None, 20)
      text = font.render("Score: "+str(score), 1, (0,0,0))
      screen.blit(text, [5,5])
      text = font.render("Lives: "+str(lives), 1, (0,0,0))
      screen.blit(text, [80,5])

      if lives <= 0:
            font = pygame.font.Font(None, 120)
            text = font.render("GAME OVER!", 1, (255,0,0))
            screen.blit(text, [36,150])
            pygame.display.update();pygame.time.wait(4000)
            lives=3; score=0; tx=280; ty=180

      pygame.display.update(); clock.tick(100)
pygame.quit()
```

See if you can figure out which numbers to alter to make Tux move faster, or the Ball or Fish. You might also want to try using your own sound effects or graphics.

PYTHON VERSION: 2.6.6 / 3.2.2
PYGAME VERSION: 1.9.2a0
O.S.: Debian 6 / Win7

TESTED!

# Feedback

*'Thanks for your most excellent magazine. Even though I'm an old greybeard I'm very much enjoying MagPi - it is a great source on getting the most out of my R.Pi, like setting up sound and getting rid of the overscan - took me ages to hunt down on the web but was all there on one page.'*

**Michael Field**

*'I've seen the past few magazines you've put together and think its a great idea for the Raspberry Pi Community, very beneficial for new users. There is a lot of useful content there, but I do think the magazine would benefit from a consistent look and feel throughout.'*

**Alex Dato**

*'I just wanted to say thanks. This magazine has some really good articles. Please keep up the good work.'*

**Tim Bradley**

*'As a virtual beginner re. Linux systems and programming your magazine has been really helpful, not to mention a good read, I just wanted to say thanks and keep up the good work. If you'd like any submissions from complete beginners, just let me know!'*

**Andrew Kilpatrick**

*'My name is Callum, im 16, I live in Scotland and im a computer enthusiast.*

*I am writing to you to thank you very much... the magazine makes the start up of anyone's raspberry pi as simple as using a computer everyday'*

**Callum Galpin**

*'Just want to say thanks to everyone who puts this magazine together for the community.'*

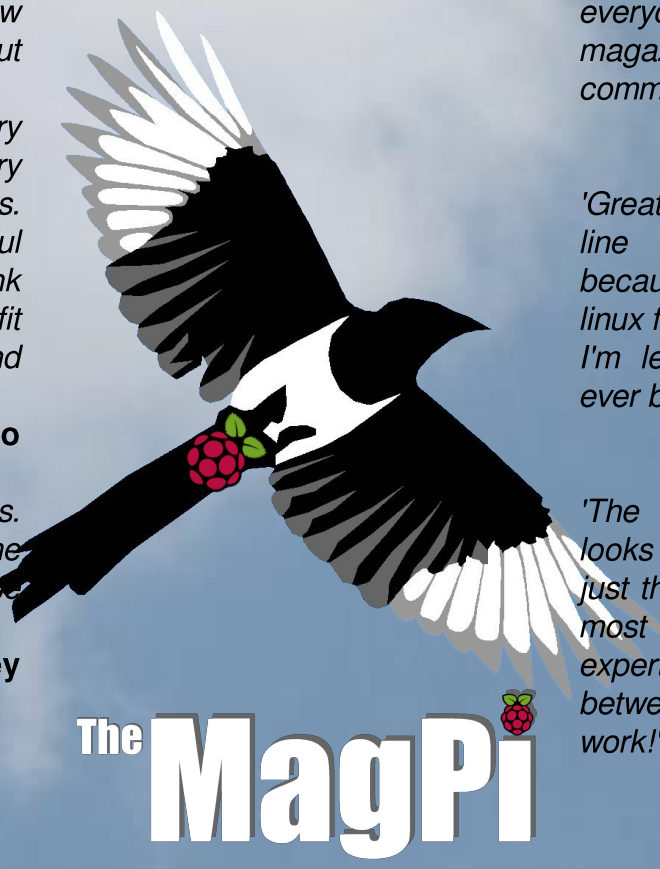
**Crenn**

*'Great issue! The command line tutorials are useful, because I've been using linux for a few years now, but I'm learning things no one ever bothered to mention!'*

**Ben Wiley**

*'The MagPi is brilliant. It looks good and is pitched at just the right level to interest most people - Beginners, experts and everyone in between. Keep up the good work!'*

**Matt Hawkins**



The MagPi

**editor@themagpi.com**