ISSUE 13 - JUN 2013

Get printed copies at themagpi.com

# The MagPi

A Magazine for Raspberry Pi Users

RISCOS Elite

C and FORTRAN

Racing with Scratch

Parallel Calculations

Pi Matrix - Control 64 LEDs

Win a 512Mb Model B Raspberry Pi & PiAngle!

Meet Amy Mather

13 year old programmer

ISSN 2051-9990

http://www.themagpi.com

# The MagPi 13

Welcome to the 13th issue of The MagPi - when you are done reading this you will agree this is definitely not an unlucky issue!

We are so proud and thankful to be able to showcase the super smart Amy Mather as she talks to us about Conway's Game of Life, what it's like being a young hacker and what she has planned for the future.

We take your Raspberry Pi to the club scene and continue our Schism Tracker music series, ply it with plenty more of Lloyd Seaton's genius cocktails and top the night with a little Pi Matrix for good show.

If that wasn't enough, we return to the 1980s by showing you how to set up and play the seminal space trading game 'Elite' on your Raspberry Pi. We also catch up on the next installments of some of the top programming languages with C Cave, Scratch Patch and Python Pit.

For those of you who have kindly supported The MagPi Kickstarter cause, we are pleased to announce that all eight issues have been printed and distribution is now commencing! Thank you all for your patience.

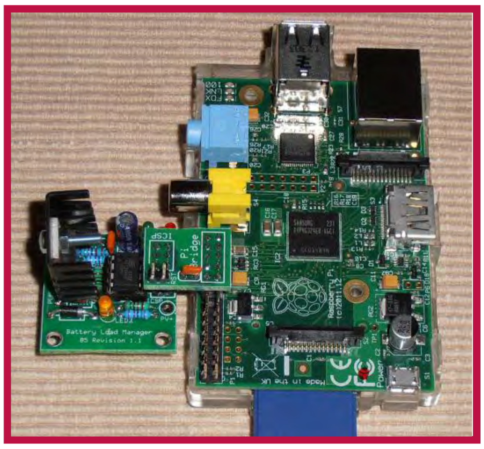
Ash Stone

Chief Editor of The MagPi

## The MagPi Team

**Ash Stone** - Chief Editor / Administration / Proof Reading
**Chris 'tzj' Stagg** - Tester
**Colin Deady** - Layout / Proof Reading
**Jason 'Jaseman' Davies** - Website
**Matt '0ther0judge0'** - Website / Administration
**Meltwater** - Proof Reading
**Aaron Shaw** - Layout / Tester / Proof Reading
**Ian McAlpine** - Issue Editor / Layout / Tester / Proof Reading
**W.H. Bell** - Layout / Administration
**Bryan Butler** - Page Design / Graphics
**James Nelson** - Proof Reading
**Adrian Harper** - Layout
**Paul Carpenter** - Tester
**Sam Marshall** - Graphics
**Dave Allan** - Tester
**Claire Price** - Layout

# Contents



Cover photo and Amy Mather article header image by **Alan O'Donohoe**
Amy Mather header image at MadLab by **Hwa Young Jung**
Amy Mather header and inline image by **Lisa Mather**

**http://www.themagpi.com**

# Amy Mather
# The future of young hackers?

**Adrian Harper**
MagPi Writer

In the unlikely case that you haven't already seen the video and don't know the name Amy Mather, I would like to introduce you to a young lady who hosted, quite possibly, the highlight presentation of the recent Manchester Jamboree, and who certainly has a bright future ahead of her in computer science.

Amy's Game of Life presentation is available on YouTube (http://tinyurl.com/amyjamboree) and I would encourage you to watch the video first and then come back to read the remainder of this MagPi article.

After watching the first time I was astounded at Amy's confidence and ability and I had to watch it a second time. I couldn't help but think that Amy is a shining example of what all our kids could be if they are given the right resources, teaching and encouragement to experiment without fear.

[MagPi] Most people have watched your presentation and have been astonished by your achievements with the Raspberry Pi, what has been the feedback from that video?

[Amy] The feedback from the Jamboree video has been amazing, with people asking me to do more talks at other conferences, interviews such as this and even a request to design a website! There have been so many lovely comments from people that I've never met - I'm really grateful to Alan O'Donohoe for giving me the opportunity to speak. I was so surprised when I was asked to talk alongside the likes of Steve Furber and Rob Bishop!

[MagPi] How did you first come across the Raspberry Pi?

[Amy] My Mum found out about the Raspberry Pi on Twitter and ordered one for me on the day of release. It took a few weeks to arrive but as soon as I got hold of it I took it to Ben Nuttall's first Manchester Raspberry Jam. There I improved upon my Scratch version of Pacman (http://bit.ly/L51Ycg) using a photograph of my face as the Pacman sprite plus I made a bat game for my little brother.

[MagPi] Before the Raspberry Pi came along, what was your experience with computers and electronics? Did you work with other platforms?

[Amy] After Alan's first Hack to the Future in February 2012 a very generous geek sent me an Arduino kit through the post. I had a school homework project to do at the time which was to make and label a volcano for geography, so I decided to incorporate my Arduino into that! When it was finished I was asked to demonstrate it at MadLab in response to a 'call for makers' for the Manchester Mini Maker Faire that was being held a couple of months later. Around the same time I attended a Manchester Girl Geeks workshop that introduced me to Codecademy. I loved it so much that I went home and carried on... I've now got 50+ badges and have worked on CSS, JavaScript, Python, PHP and HTML5 courses.

[MagPi] Obviously you are quite the accomplished programmer now. What was your first project with the Raspberry Pi?

[Amy] My first project on the Raspberry Pi was a guessing game in Python. It used the 'random' library to choose a different number each time for the answer and you had to guess which number had been picked. It told you if your answer was too big or too small and deducted points each time you got it wrong. If you got one of the best scores then it asked you for your name and saved it to a high scores list.

[MagPi] No matter the country, there are often articles in the press about the poor IT and Computer Science teaching in schools. How have you found this to be in your case? Is there much interest in gaining a deeper knowledge of computer science with your friends and classmates?

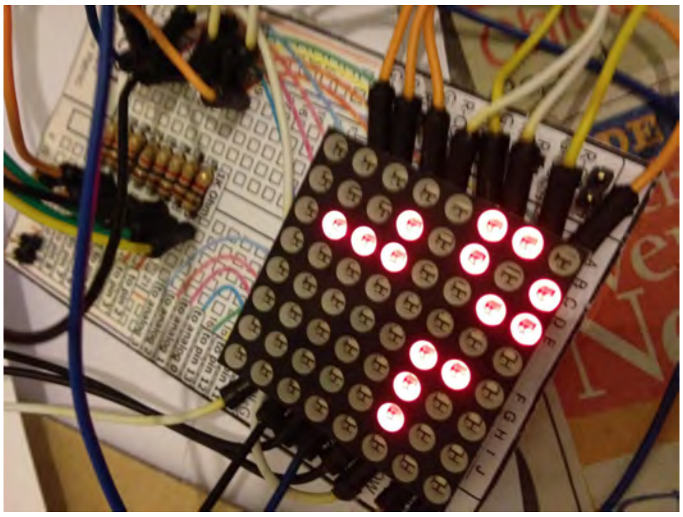
[Amy] No, most of my friends at school use a bit of HTML to update their Tumblr blogs and have done Scratch, Alice and Dreamweaver in ICT lessons, but they don't really seem to want to take it any further. I think that if the Raspberry Pi and Arduino were incorporated into ICT lessons (e.g. introduction to Python from age 11), Textiles lessons (e.g. Arduino Lilypad) or Product Design / Resistant Materials lessons (e.g. robots) then that could show them that programming can actually help you create some pretty awesome stuff. Hopefully that would be enough to inspire them to start hacking and coding their own projects outside of school.

[MagPi] Are there Raspberry Pis available in your school already?

[Amy] No, unfortunately there aren't any Raspberry Pis in my school at all, but I'd like to change that ;)

[MagPi] What more can we do as a community to get more young people, both boys and girls, into computer science? How could our readers get involved?

[Amy] I think that we need to set up more community hackspaces, or places like MadLab, that welcome kids through their doors. Readers could set up these kid friendly spaces and run events specifically for under 18's like MadLab's CoderDojos or set up code clubs in their local primary schools (http://bit.ly/HLRWwT). Your local STEMnet team (http://bit.ly/B0z2q) could help with setting this up if you're unsure.



Raspberry PI to Arduino to LED matrix. Amy is not limited to software hacking!

[MagPi] If you were able to influence the Raspberry Pi design team and introduce a new model, what changes would you bring to the table?

[Amy] I wonder if it would be a good idea to attach a 16x2 LCD screen to the board, so that once you've written your code for a project and it's on the Pi, you could then just plug in a keyboard and a terminal-like environment would come up on the LCD screen. This would mean that you could run your code without having to plug in a monitor, making it easier to incoporate into projects. I would also like to see if the design team could introduce a new board that could be used in wearable electronics, along the lines of an Arduino Lilypad but even better! I like the fact that the Lilypad is washable, lightweight and can be easily attached to fabric with conductive thread. If they could improve upon this it would be great!

[MagPi] What are the best resources out there that you can recommend for young Raspberry Pi programmers, or people in general that are giving programming a go for the first time?

[Amy] Obviously, the MagPi magazines are a great resource :) but I think that the Super Scratch Book (http://bit.ly/MVKOw8) looks like a great book to help younger users of the Raspberry Pi. For older users Simon Monk's Python book (http://bit.ly/VXJZrl) provides a great introduction to the language. There are some very helpful resources online - I really like;

| | |
|---|---|
| GeekGurlDiaries: | http://bit.ly/1088jKG |
| Adafruit: | http://bit.ly/LEQbiB |
| MakeZine: | http://bit.ly/13Xr44a |
| Instructables: | http://bit.ly/4kLr49 |
| Arduino Tutorials: | http://bit.ly/vfm8A |

The Twitter feeds @mykidcancode and @codingforkids often have great links, too.

[MagPi] Conway's Game of Life can be quite daunting. What projects would you encourage new young programmers to begin with?

[Amy] Scratch (http://bit.ly/y4Mw) is probably the best place for young programmers to begin, but if you prefer working towards goals then Codecademy (http://bit.ly/oDjwOj) will probably be more helpful. On the Codecademy website you can choose to learn a particular language or work on a project e.g. one of the first games you'll make in the JavaScript course is Fizz Buzz. Alternatively, simple guessing games or rock-paper-scissors type games are a good place to start and there are plenty of internet forums where you can search for help with any queries that you may have.

[MagPi] In your presentation you mentioned Madlab and Manchester Hackspace. Can you tell us a little about these resources and the benefits you get from going there?

[Amy] Without MadLab (http://bit.ly/rrahU) I wouldn't be talking to you today! They are AWESOME!!!! I've met so many great people there who are always willing to share their knowledge and teach me new things. At MadLab I have met: Mike Little, who was my mentor for YRS2012, Ben Nuttall who runs the Manchester Raspberry Jam (http://bit.ly/YUE2yY), Alan O'Donohoe who organised the Raspberry Jamboree, John Beckerson a senior curator from Manchester's Museum of Science and Industry who invited me to display at the Mini Maker Faire, Dr Lucie Green who is a space scientist, plus lots and lots of other very kind people who have taught me so much. I've also been to lots of workshops held by Manchester Girl Geeks (http://bit.ly/PYsHEL) and have started helping to teach at their workshops - there's just so much going on there! Bob Clough from HacMan (http://bit.ly/jGLhtz) showed me how to use Inkscape to control the laser cutter when I was making the case for my 'Conway's Game of Life on Pi'. All the people at the hackspace are happy to help people with any problems that they may have - I wish there was a MadLab/HacMan in every town to help other children discover their inner geek!

Amy shares her knowledge by teaching others.

[MagPi] Are there many other kids around your age going to the hack spaces on a regular basis? Can you tell us about the type of projects they are working on?

[Amy] At MadLab's U18 CoderDojo there's always lots of different activities going on. For example there's a MaKey MaKey kit which most of the younger children love making banana pianos with (http://bit.ly/KkiMLO), a Lego WeDo kit (http://bit.ly/12wCqup) and Bare Conductive inks were bought for the last event (http://bit.ly/16r9C3). The events are always fully booked and there are always volunteers around who can teach the children Scratch, Python, PHP, Minecraft plus a variety of other skills. At the last event, the Minecraft table all worked together making fireworks and they demonstrated their display at the end.

[MagPi] You already have access to some nice gadgets such as laser cutters and 3D printers. What technologies are on the horizon that excite you?

[Amy] I love the idea of conductive inks. I met Dr Kate Stone (http://bit.ly/5y6TPl) at the YRS Festival of Code last year and she showed me a pair of headphones that were made from paper, some electronic greetings cards that she'd made and also this jazz poster (http://bit.ly/17YXxZA). It was amazing to see what she'd achieved with the technology. My mum bought some of Bare Conductive's products at the Maker Faire UK recently and I can't wait to try out the conductive paint!

[MagPi] We're often asked by parents if their kids are old enough for a Raspberry Pi. Given your experience, what do you think is an ideal age for kids to be introduced to programming and do you think the tools available are adequate?

[Amy] I don't think that there is a minimum age; if a child can read and type, then they can learn to code!

[MagPi] As parents, what can we do, apart from buying a Raspberry Pi obviously, to spark our children's interest in computing?

[Amy] There are a couple of computing festivals in the UK soon, #Include (http://bit.ly/11AErKK) in Warwickshire that is aimed at 11-13yr olds and Silicon Dreams (http://bit.ly/XlRudv) in Leicestershire in July. Also, during the summer holiday in August, Young Rewired State (http://bit.ly/eAtCMI) is an event with centres throughout the country holding a week-long workshop, where children up to 18 years old spend 5 days working with mentors on a project, using openly available data sets. Then, at the weekend, all the teams meet up in Birmingham for the Festival of Code where they present their projects.

[MagPi] Finally, what new projects are you working on at the moment? Are you able to give us a sneak peek?

[Amy] I've been offered a touch screen and larger LED array which, when they arrive, I will add into my 'Game of Life on Pi' to show at the Manchester Mini Maker Faire this summer. Another idea is a 'T-Shirt of Life' which would be a T-shirt made from recycled materials with LEDs sewn on the front in a grid formation, which would display the Conway's Game of Life. What I'd really love to see though, is a Maker Faire exclusively for U18's - it could be called the Mini Makers Faire! It would be great if it were run like YRS with a week-long workshop beforehand using Hack Spaces and FabLabs with volunteer mentors and then at the weekend the children would present what they've made at the Mini Makers Faire. If anybody out there would like to help me do this, please get in touch!

Amy is a prime example of why we should be getting Raspberry Pis into the hands of as many kids as possible! It is young people like her that provide inspiration and drive to those within the Raspberry Pi Foundation, The MagPi, and the wider community, to keep doing what they're doing.

This certainly won't be the last time you hear of Amy in The MagPi. She is currently working on a number of articles for the magazine to help us encourage and challenge our younger readers.

**edumacation**

*n.* [ˌɛdjʊˈmäkeɪʃən]

**1.** Teaching, but fun.

**2.** An enlightening experience of imparting knowledge without the learner realising.

Need Raspberry Pi, Cases, Electronics, Kits or a bit of help with Physical Computing?

Have a chat with us at
**edu@pimoroni.com**

Yes, we do bulk discounts and POs and we have a lovely tame STEM Ambassador whose help is free.
We also don't mind if you point out our grammatical errors :-)

STEM AMBASSADORS ILLUMINATING FUTURES

**Noodles**

**HDMI 大粉 USB**

**http://shop.pimoroni.com**

Pirate Store from PIMORONI

**PI BOW**

The neat little layer case for your Raspberry Pi®

Introducing the Pibow ModelA for the low-profile Model A
A hacker's delight!

| Crystal | Toxic | Ninja | Adafruit | Rainbow |
|---|---|---|---|---|

PIMORONI

**Available From:**
**http://pibow.com/**

adafruit · maplin · cpc · MODMYPI · MCM

## The MagPi What's On Guide

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

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

## York Raspberry Jam

When: Saturday 8th June 2013 @ 10.00am
Where: National STEM Centre, University of York

This event will run from 10.00am until 5.00pm. Further information and tickets are available at http://raspberryjam.eventbrite.co.uk

## Trier Raspberry Jam

When: Saturday 15th June 2013 @ 13.00pm
Where: Trier, Germany

The event will run from 1.00pm until 5.30pm followed by a social event. Further information is available at http://www.piandmore.de/programm

## Charlotte, NC Raspberry Pi Meeting

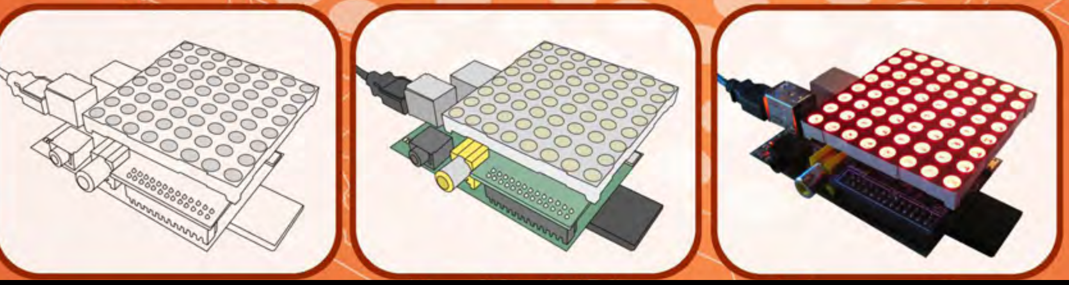
When: Thursday 13th June @ 5.30pm
Where: The Hickory Tavern - Metropolitan, Charlotte, NC 28204

The event will run from 5.30pm until 8.00pm. Further information is available at http://bdpacharlotte.org/cms/node/16

## @Bristol Raspberry Pi Boot Camp

When: Saturday 15th June 2013 @ 10.30am
Where: @Bristol Science Centre, BS1 5DB

The day will run from 10.30am until 4.30pm. Further information and tickets are available at http://www.at-bristol.org.uk/1755.html and http://pibootcampjune2013.eventbrite.com



PI MATRIX
Control an 8x8 matrix of 64 LEDs

# Part 1: Introduction to Pi Matrix and programming the I$^2$C bus

**Bruce E. Hall**
**W8BH**
Guest Writer

**DIFFICULTY : MODERATE**

## Introduction

The Pi Matrix is a fantastic tool for learning GPIO programming on the Raspberry Pi. You could hook up a few LEDs and switches, but why do that when you can program a huge matrix of 64 LEDs? If you are ready to learn about GPIO, this is the first kit you should get. Why?

- It's fun!
- It's compact; just pop it on top of the Pi
- It's clean; no need for a breadboard or wires
- It's safer; does not use unbuffered GPIO pins
- It's a great excuse to get soldering
- It's an easy way to learn about i2c on your Pi
- Did I mention that it's fun?

As a huge bonus, just remove the LED matrix and you are left with 16 new, buffered I/O pins to command. Compare that to the 8 unbuffered general I/O pins that are on the Raspberry Pi.

## Construction

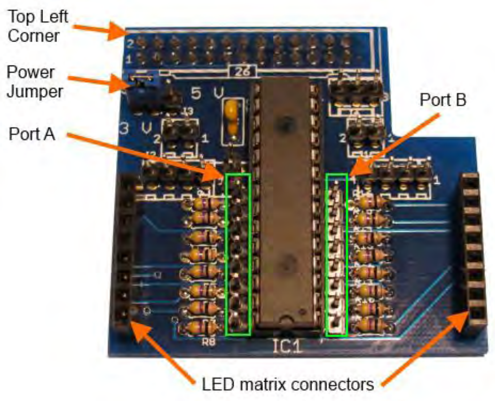
The Pi Matrix kit costs $15.99 plus shipping and is available on Ebay from http://mypishop.com and soon from http://www.modmypi.com. My bag of parts arrived safely and well packaged. This kit does not contain a huge number of parts; the PCB, the MCP23017 chip and socket, a 26-pin female connector which connects to the Raspberry Pi, two female header strips, male header pins, a header pin jumper, 16 resistors, a decoupling capacitor and of course the 8x8 LED matrix.

The kit does not come with any documentation, but simple instructions are available at http://mypishop.com/Pi%20MATRIX.html. Where the PCB has the numerals '2 1' silk-screened at the GPIO connector, consider this to be the top-left corner of the PCB. This is important because other components must be orientated correctly.

While the Pi Matrix kit is not difficult to build, some fine soldering is required. If you have not soldered before, practice on something else first.

Start by soldering the 16 resistors, followed by the socket for the MCP23017 chip. Note that the notch on the socket points downwards. Next solder the capacitor then the male header pins. There are a lot of male header pins, but you actually only need to solder the 3 header pins in the top left corner of the PCB. This is labelled 'SV1' on some boards. The other header pins are only needed when you want to do other I/O.

If you want nice, straight header pins, tack down one pin with solder, check alignment, resolder if necessary, then solder the rest when everything looks vertical and flush with the board.



Push the two female header strips onto the LED matrix, then solder them to the PCB. This ensures perfect alignment. Remove the LED matrix. Finally, solder the 26-pin female header. **NOTE:** Unlike all the other components, this header is placed on the underside of the PCB and soldered on the top side.

With the soldering complete, push the MCP23017 chip into its socket with the notch pointing DOWN, not up. Insert the header jumper onto header 'SV1' so that the leftmost and centre pins are joined. Last, insert the LED matrix making sure that pin 1 goes to the top of the left-hand header and pin 16 goes to the top of the right-hand header. The Pi Matrix won't work if you get this the wrong way round.

## Setting up I$^2$C

The Pi Matrix needs additional software installed before we can use it. In particular, we must configure the I$^2$C bus. The I$^2$C drivers are disabled by default. Let's enable them.

From the command line, enter:

```
cd /etc
sudo nano modprobe.d/raspi-blacklist.conf
```

Look for the entry `blacklist i2c-bcm2708` and add a hash '#' at the beginning of the line so it becomes `#blacklist i2c-bcm2708`. Press <Ctrl>+<X> then press <Y> and <Enter> to save and exit.

Next edit the modules file. From the command line, enter:

```
sudo nano modules
```

Add `i2c-dev` on a new line. Press <Ctrl>+<X> then press <Y> and <Enter> to save and exit.

Next install some tools and Python support. From the command line, enter:

```
sudo apt-get update
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Now add the 'pi' user to the i2c group. From the command line, enter:

```
sudo adduser pi i2c
```

Finally, shutdown your Raspberry Pi. From the command line, enter:

```
sudo halt
```

Turn off the power and seat the Pi Matrix board on the Raspberry Pi's 26-pin GPIO header. The board should lay across the top of the Pi, not away from it. Cross your fingers, turn on the power and log back into the 'pi' account.

Let's confirm that the Pi Matrix board is working. From the command line, enter:

```
sudo i2cdetect -y 1
```

**NOTE:** Use 0 instead of 1 for the bus number in the above command if you have a revision 1 Raspberry Pi. The revision 1 Pi does not have any mounting holes on the PCB; newer revisions have 2 mounting holes.

The i2cdetect command will listen for I$^2$C devices and report their addresses. If all goes well you should see all blanks except for a '20' on the second line. Congratulations! This indicates that your Raspberry Pi is communicating with the MCP23017 chip (address 0x20) on the Pi Matrix.

If you don't see the '20', remove the Pi Matrix board and start troubleshooting. Nothing that follows will work until you've established I²C communication.

## Baby steps

Now that I²C is working, it is time to talk to the LEDs! The i2cset utility in the i2c-tools package lets us send data. The format is "`i2cset -y <bus number> <chip addr> <data addr> <value>`". The '-y' option is not required but turns off scary-sounding warnings! We will send data to the Pi Matrix one byte at a time. The 'bus number' is 0 for a revision 1 Raspberry Pi and 1 for newer revisions. The 'chip addr' is 0x20, as reported previously by i2cdetect. From the command line, enter the following commands in the order shown, omitting the comments:

```
#set port A to all outputs
i2cset -y 1 0x20 0x00 0x00
#set port B to all outputs
i2cset -y 1 0x20 0x01 0x00
#set all port B (row) pins low
i2cset -y 1 0x20 0x13 0x00
#set all port A (column) pins high
ic2set -y 1 0x20 0x12 0xFF #LEDs turn on
#set all port A (column) pins low
ic2set -y 1 0x20 0x12 0x00 #LEDs turn off
```

Did you see the all of the LEDs turn on? If not, power off the Pi and reseat the Pi Matrix board and LEDs. Make sure that you have oriented them correctly. Pin 1 of the LED matrix is towards the top-left of the Pi Matrix board and the Pi Matrix board is mounted over the Raspberry Pi, not away from it.

## Matrix programming for dummies

It's time to try some real programming in Python. The simplest way to try out some Python is to use the interactive mode. From the command line, enter:

```
python
```

The Python prompt is a triple chevron '>>>'. From here we enter our Python code and the interpreter works as we go. Enter the following commands:

```
>>> import smbus
>>> addr = 0x20
>>> dirA = 0x00
>>> dirB = 0x01
>>> portA = 0x12
>>> portB = 0x13
>>> bus = smbus.SMBus(1)
```

The first line adds the I²C library for Python and the next five are constants for addressing the MCP23017 chip. The last line establishes a variable to write to the I²C bus. Enter the following to send data to the Pi Matrix board:

```
>>> bus.write_byte_data(addr,dirA,0x00)
>>> bus.write_byte_data(addr,dirB,0x00)
>>> bus.write_byte_data(addr,portB,0x00)
>>> bus.write_byte_data(addr,portA,0xFF)
```

The first two lines set all 16 I/O pins to output. The third line sets all of the row pins low and the last line sets all of the column pins high. If you didn't make any spelling mistakes, all the LEDs should be on now. To turn them off, just zero out the port A register with the following command:

```
>>> bus.write_byte_data(addr,portA,0x00)
```

Once you get that working, it isn't too hard to make an honest-to-goodness Python script. I've written a short one based entirely on the above code. Type it in using your favorite editor and save the file as hello-matrix.py. You can run it from the command line with the following:

```
chmod +x hello-matrix.py
./hello-matrix.py
```

In time-honored 'hello world' style, this script will flash the LEDs on your matrix. The LEDs might remain on, depending when you quit. Run it again or use one of the methods above to turn off the LEDs.

## Python code

```
#!/usr/bin/python
#
# hello-matrix.py
#
# Flashes all LEDs on the Pi Matrix Board

import smbus        #Python access to I2C bus
import time         #For timing delays

# Definitions for the MCP23017 chip
addr = 0x20         #I2C bus address of the 23017
dirA = 0x00         #PortA I/O direction register
dirB = 0x01         #PortB I/O direction register
portA = 0x12        #PortA data register
portB = 0x13        #PortB data register

# Program start
print "Hello, Pi Matrix! (Press <Ctrl>+<C> to stop)"
bus = smbus.SMBus(1)     #Use '1' for newer Pi's and 0 for revision 1 Pi's
bus.write_byte_data(addr, dirA, 0x00)      #all zeros = all outputs on PortA
bus.write_byte_data(addr, dirB, 0x00)      #all zeros = all outputs on PortB
bus.write_byte_data(addr, portB, 0x00)     #set row outputs low

# Loop until user stops program
while (True):
   bus.write_byte_data(addr, portA, 0xFF) #set column outputs high = LEDs on
   time.sleep(0.5)                        #keep LEDs on for 0.5 secs
   bus.write_byte_data(addr, portA, 0x00) #set column outputs low = LEDs off
   time.sleep(0.5)                        #keep LEDs off for 0.5 secs
```

## Reference

Orientation matters. With the Pi rotated so the GPIO header is at the top, SD card is on the left and USB is on the right, then:
1) LED rows go left to right, with row 1 being at the top and row 8 at the bottom.
2) LED columns go up and down, with column 1 on the left and column 8 on the right.
With this orientation, the LED matrix has the following pinout:

| | |
|---|---|
| Pin 1 = Col 4 | Pin 16 = Row 8 |
| Pin 2 = Col 2 | Pin 15 = Row 7 |
| Pin 3 = Row 2 | Pin 14 = Col 7 |
| Pin 4 = Row 3 | Pin 13 = Row 1 |
| Pin 5 = Col 1 | Pin 12 = Col 5 |
| Pin 6 = Row 5 | Pin 11 = Row 6 |
| Pin 7 = Col 3 | Pin 10 = Row 4 |
| Pin 8 = Col 6 | Pin 9 = Col 8 |

The LEDs are oriented with the anode on the columns and cathode on the rows. To light a LED you must raise the corresponding column high and take the row low.

The MCP23017 is mounted notch-down. Looking from the top, here is the data pin layout. Both 8-bit data ports are broken out on male header pins on each side of the chip:

| | |
|---|---|
| A0 (pin 21) – goes to Col 8 | B7 (pin 8) – goes to Row 8 |
| A1 (pin 22) – goes to Col 7 | B6 (pin 7) – goes to Row 7 |
| A2 (pin 23) – goes to Col 6 | B5 (pin 6) – goes to Row 6 |
| A3 (pin 24) – goes to Col 5 | B4 (pin 5) – goes to Row 5 |
| A4 (pin 25) – goes to Col 4 | B3 (pin 4) – goes to Row 4 |
| A5 (pin 26) – goes to Col 3 | B2 (pin 3) – goes to Row 3 |
| A6 (pin 27) – goes to Col 2 | B1 (pin 2) – goes to Row 2 |
| A7 (pin 28) – goes to Col 1 | B0 (pin 1) – goes to Row 1 |

Each data pin goes to a LED row or column via a current limiting resistor. If you use the male header breakouts for other projects, please note that the bit order is not the same. On the left (port A) it goes from bit 0 to bit 7, but on the right (port B) it goes in the reverse direction from bit 7 to bit 0.

## Next time...

In Part 2 of this series we will create more interesting displays on the Pi Matrix. Have fun!

# JUNE 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 a new 512MB Raspberry Pi Model B, the brilliant, new PCSL PiAngle case plus a GPIO Cobbler kit!
The second and third prize winners will each receive the superb, new PCSL PiAngle case.

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

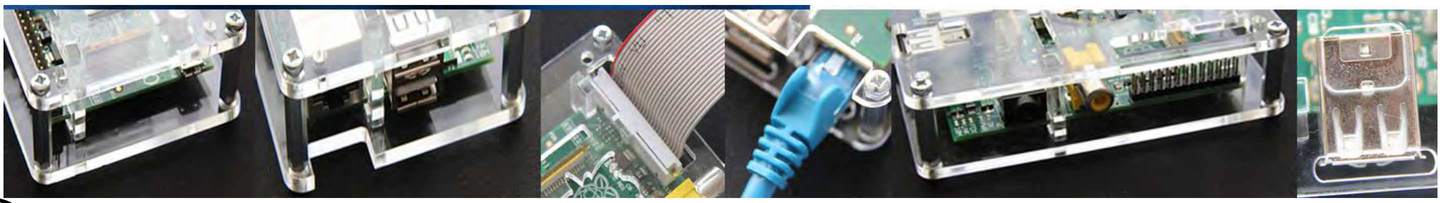
Closing date is 20th June 2013.
Winners will be notified in next month's magazine and by email. Good luck!

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

## April's Winners!

The winner of a new 512MB Raspberry Pi Model B plus PCSL Raspberry Pi case is **Harold Barker (Seagrave, UK)**.
The 2nd and 3rd prize winners of a PCSL Raspberry Pi case are **Darren Paskell (High Wycombe, UK)** and **Joseph White (Colchester, UK).**

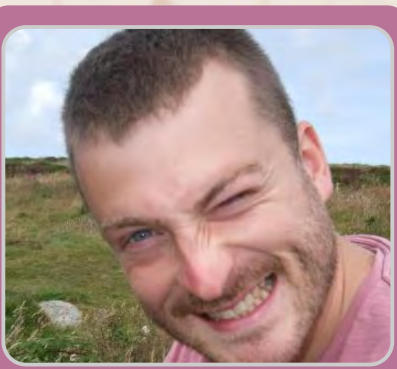
Congratulations. We will be emailing you soon with details of how to claim your prizes!



**BELLISSIMA**
by DJ Quicksilver

SchismTracker

# Creating music using the Raspberry Pi - Part 3



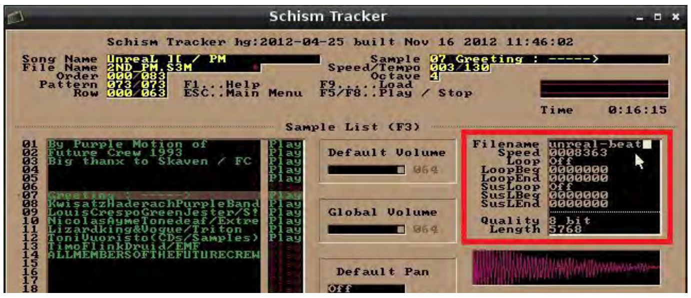
**David Honess**
Guest Writer

**DIFFICULTY : BEGINNER**

## Ripping samples

So with what you have learnt here in the previous issue of The MagPi, you are probably in a good position to start experimenting. I expect you're wondering where to get more samples and sound effects from? There are numerous solutions to this; you could search the internet for free sample packs, which can be a little hit and miss, or you can "steal" the samples that you like from other module files! This is incredibly effective because you can hear how people are using samples in their own songs first.

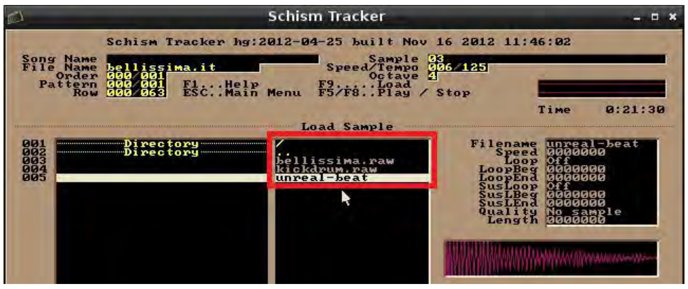
If you haven't done so already head over to http://modarchive.org and choose Music > Charts > Top Favourites. Download a few of them and start checking them out for samples that you like. F9 to load a song, F5/F8 for play/stop and then F3 to see the sample list.



```
Schism Tracker
Schism Tracker hg:2012-04-25 built Nov 16 2012 11:46:02
Song Name Unreal ][ / PM                    Sample 07 Greeting : ----->
File Name 2ND_PM.S3M                   Speed/Tempo 003/130
    Order 000/083                           Octave 4
  Pattern 073/073   F1...Help          F9.....Load
      Row 000/063   ESC..Main Menu     F5/F8..Play / Stop
                                                     Time     0:16:15
                          Sample List (F3)
01 By Purple Motion of          Play               Filename unreal-beat
02 Future Crew 1993             Play  Default Volume  Speed 0008363
03 Big thanx to Skaven / FC     Play  064              Loop Off
04                              Play               LoopBeg 0000000
05                              Play               LoopEnd 0000000
06                                                 SusLoop Off
07 Greeting : ----->            Play               SusLBeg 0000000
08 KwisatzHaderachPurpleBand    Play               SusLEnd 0000000
09 LouisCrespoGreenJester/S!    Play  Global Volume
10 NicolasAymeTonedeaf/Extre    Play  064          Quality 8 bit
11 Lizardking&Vogue/Triton      Play                Length 5768
12 ToniVuoristo(CDs/Samples)    Play
13 TimoFlinkDruid/EMF
14 ALLMEMBERSOFTHEFUTURECREW
15                                    Default Pan
16                                    Off
17
18
```

Use the cursor keys to move up or down the list and test each sample out with the Q thru P keys. When you find one that you like you can press ALT-W to save it to disk (this will save a file to the samples folder that we created earlier). If you see an error it will be because the sample has no filename on the right hand side. Just make sure you have typed a filename here (see left) and try ALT-W again.

Press F9 to load your Belissima module up again (ignore the warning). Press F3, select an empty row and press Enter. The sample that you exported should now show in the list (see below). Select it and press Enter. The sample is now loaded into your module and can be used to program a pattern in exactly the same way as demonstrated earlier.



```
Schism Tracker
Schism Tracker hg:2012-04-25 built Nov 16 2012 11:46:02
Song Name                                   Sample 03
File Name bellissima.it                Speed/Tempo 006/125
    Order 000/001                           Octave 4
  Pattern 000/001   F1...Help          F9.....Load
      Row 000/063   ESC..Main Menu     F5/F8..Play / Stop
                                                     Time     0:21:30
                          Load Sample
001      Directory          /                   Filename unreal-beat
002      Directory          ..                     Speed 0000000
003                         bellissima.raw          Loop Off
004                         kickdrum.raw         LoopBeg 0000000
005                         unreal-beat          LoopEnd 0000000
                                                 SusLoop Off
                                                 SusLBeg 0000000
                                                 SusLEnd 0000000
                                                 Quality No sample
                                                  Length 0000000
```

Obviously we're still only scratching the surface of what Schism Tracker can do here, but expect more Schism Tracker articles in the future!

Here is something for you to try on your own. Copy and paste pattern 001 (melody and kick drum) into 002. After doing that copy everything in Channel 01 to Channel 03. You will notice that the overall effect of duplicating a sample over two channels is that the sound becomes louder. See if you can integrate this into the song somehow. Remember that you have to enter new patterns into the list on the F11 screen. Press F1 for Help. Good luck and have fun!

# IT'S A SMALL WORLD AFTER ALL

**Lloyd Seaton**
Guest Writer

**These projects are for hobbyists confident with a soldering iron and are prepared to purchase their own components**

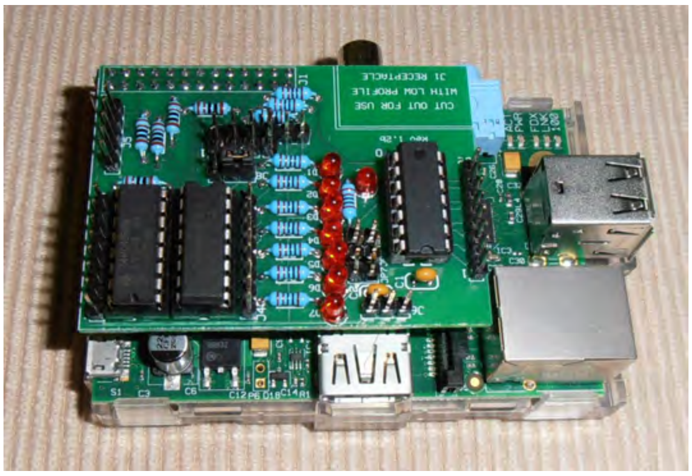
**DIFFICULTY : ADVANCED**

## Background

Regular readers of The MagPi are probably familiar with the Gertboard and many will be aware that it has an AVR microcontroller (ATmega168/328). This can be programmed via Arduino sketches to acquire analog signal measurements, drive digital outputs and read digital inputs.

Because Gertboard's ATmega chip has 14 digital I/O pins and 6 analog input pins, that can also do digital I/O, it's a fairly large integrated circuit (IC) with 28 pins overall. Less well known is that the ATmega's manufacturer, Atmel, also produces a compatible range of smaller ICs - the ATtiny family. These include microcontrollers in 14-pin and 8-pin packages for applications that require the intelligence and interface versatility of a microcontroller but don't need to interface with such a large number of circuits.

When the author set out to design a small Raspberry Pi expansion board it soon became apparent that there was not sufficient space for an ATmega chip, but an ATtiny44/84 might fit. However, the Arduino platform does not use ATtiny chips and this was initially a concern until the discovery that Arduino support for ATtiny chips was available from MIT Media Lab which integrated with Raspbian's Arduino IDE. Suddenly there was the very real possibilty that a small Raspberry Pi expansion board could include support for GPIO access **and** embrace the Arduino culture! Thanks to MIT Media Lab!



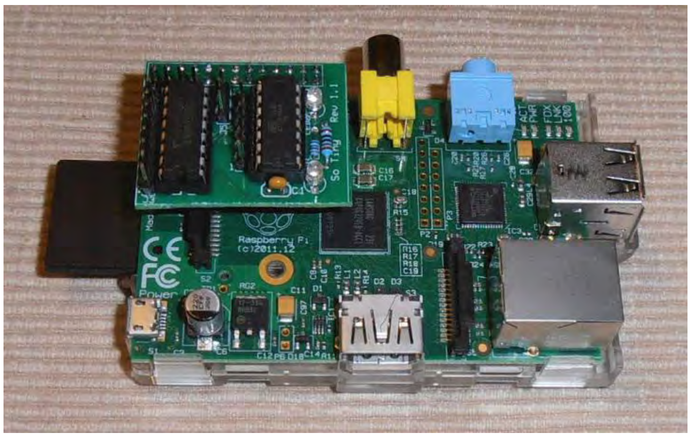
## PCB Manufacture

Previous articles in this "cocktail series" have outlined the manufacturing strategy designed to allow the manufacture of small batches of PCBs at a reasonable price. For the Tiny I/O project (above) it was decided that two PCBs should fit on each manufactured board to bring the PCB unit price down to about $23. Later, a less elaborate project, "So Tiny" was developed which fits four PCBs on each manufactured board for a PCB unit price of about $12.

## So Tiny Project

The So Tiny project is new. Consequently, it was not included in the project outlines published in issue 10 of The MagPi, so it is appropriate that it be introduced here.

- Includes ATtiny44/84 MCU
- Compatible with Raspbian Arduino IDE
- ATtiny connects to Raspberry Pi via SPI bus
- 7 Darlington outputs (6 shared +1)
- 1 indicator LED controlled by Raspberry Pi
- 2nd indicator LED (shared)
- 4 analog inputs to ATtiny
- UART and I$^2$C bus provisions

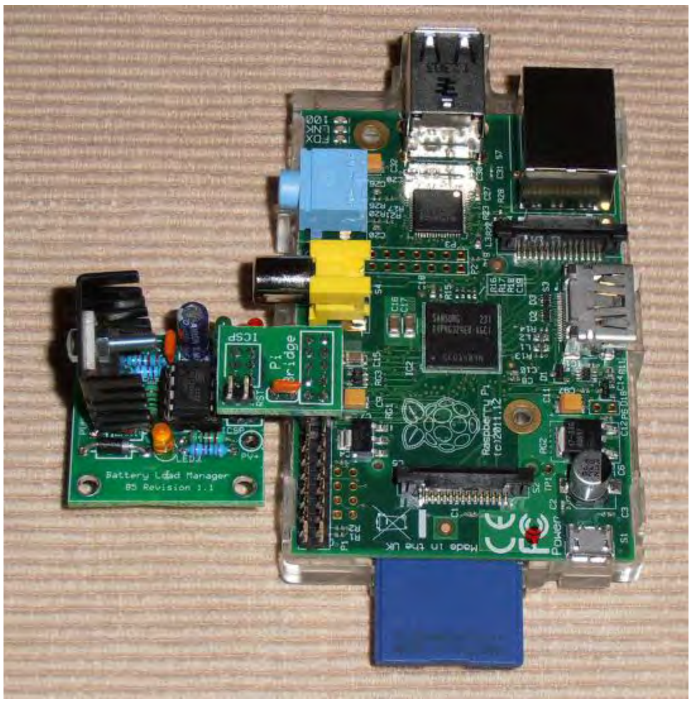


## Tiny Arduino In The Field

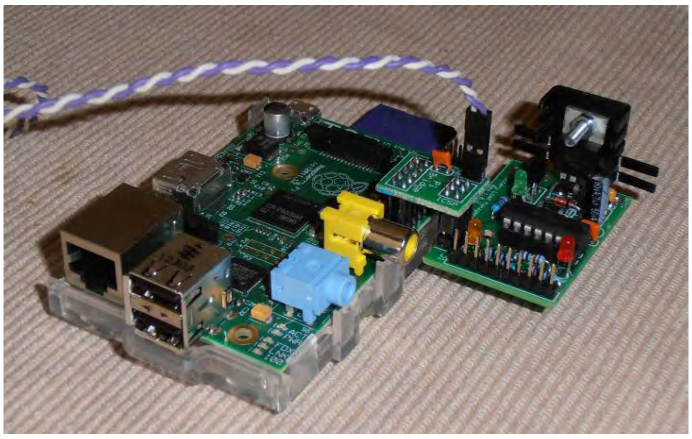
ATtiny is well suited to applications that require supervision and control of a process that needs to react to changes in its operating conditions. Such a case is the supervision of a 12V water pump to ensure that it does not pump for long periods of time (becoming hot) or in such a way that its battery suffers from excessive discharging. The Battery Load Manager 85 project uses an ATtiny45/85 MCU to control a power MOSFET to switch a high current circuit in accordance with the battery state-of-charge and a policy programmed via an Arduino sketch.

Six PCBs for this project fit on each manufactured board to achieve a PCB unit cost of <$8. An Arduino sketch has been written to implement a policy for supervision of a solar powered pump delivering drinking water for livestock. Field trials should commence soon.

The BattMan+ project, shown below, is connected to the Raspberry Pi via a Pi Bridge ICSP interconnect for programming. BattMan+ is conceptually similar to the Battery Load Manager 85 project but has additional I/O capabilities for more demanding applications.



The blog at **picocktails.blogspot.com** has a dedicated page (Issue13) with links to all of the resources related to these projects. There are design description documents with schematic diagrams, pin-out tables, suggested component suppliers and construction and operating hints. There are also links to the cocktail files, photo albums and test programs.



The information blog also has an Arduino page describing the procedure for installing ATtiny support for Arduino on Raspbian.

David Honess

Guest Writer

# Fly your Cobra MkIII trading and combat craft in this classic game

DIFFICULTY : MODERATE

Unless you're around my age, or older, two things are probably true about you. One is that you have never played Elite and the other is that the logo below probably looks like it's for a garden centre! Fear not, I am going to let you in on all these strongly guarded secrets. After reading this you will be able to go up to older gamers and watch their jaws hit the floor when you hold your own in a conversation about a seminal game from the 1980s.

ACORN

Back in 1987 Acorn Computers released the successor to the BBC Micro. It was called the Archimedes and it was tremendously popular. It had a new kind of processor called a RISC chip (Reduced Instruction Set Computer). Acorn Computers evolved over the years into what is now called ARM and you have an ARM processor in your Raspberry Pi. So you could say that the Raspberry Pi is the great-great-grandchild of the Archimedes? Up to you, but that mantle probably belongs to Acorn's earlier BBC Microcomputer.

Acorn made an operating system for their RISC chips called RISC OS which was a hugely popular, much loved, operating system. It is still being developed and in November 2012 RISC OS was released for the Raspberry Pi as an SD card image that you can download. If you're a fan of drag and drop OSes you will get on fine with it.

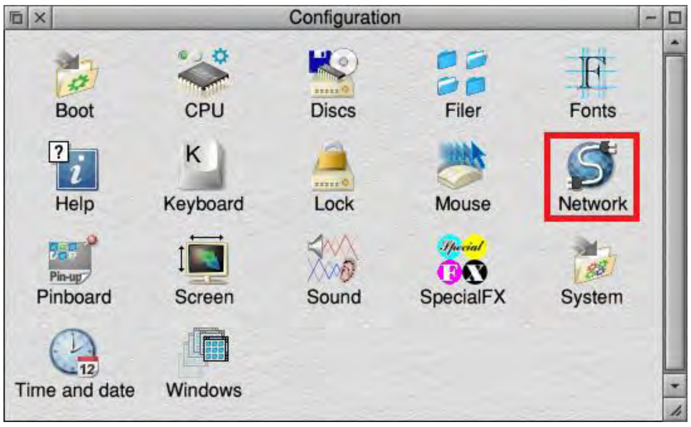
Elite was a seminal computer game that changed the entire landscape of gaming. Back in the early 1980s computer games were all about getting the highest score before your lives ran out. Elite was an open ended game that had no score, no levels and no bosses – just one enormous galaxy within which you could do whatever you wanted. This is what people call a "sandbox" game.

Elite originally came out in 1984 on the BBC Micro (above) and had wireframe graphics. By the time the Archimedes version was released in 1991 they had improved the graphics a fair bit. The Archimedes version of Elite is widely regarded as the best version of the original game, so in the instructions that follow I am going to explain firstly how to get the game running and secondly how to play it.

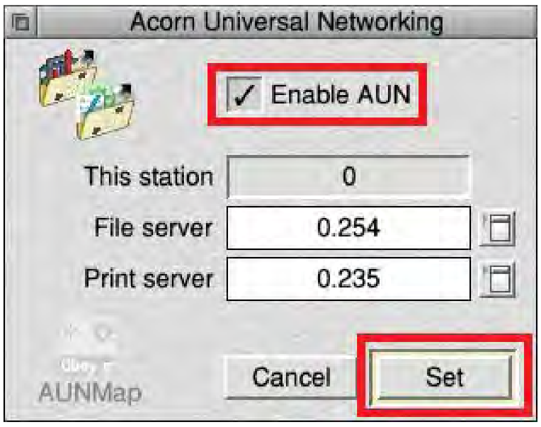
Our goal here is to run the Archimedes version of Elite on your Raspberry Pi. Unfortunately Arc Elite will not run natively on the Raspberry Pi because the Archimedes processor is 26 bit whereas the Pi processor is 32 bit. So we need to use an emulator. This is a program that will basically fool the game into thinking it's running on a real Archimedes computer and thus will run normally. We will use ArcEm which gives you a fully functional Archimedes desktop environment to work in (within which we will run the game).

The first thing we need to do though is install and boot RISC OS. This was covered in Issue 9 of The MagPi.

A quick note about the RISC OS mouse. This had three buttons. Clicking the mouse wheel on your modern mouse will do the job of the middle mouse button (button 2). It's actually the middle click that gives you context menus similar to those that you have in Windows. We'll be using those later.



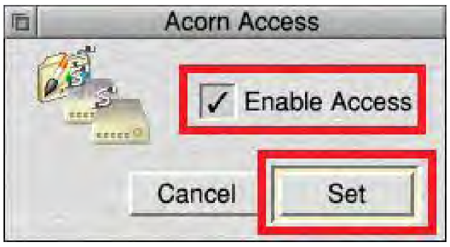
To download the necessary Archimedes emulator and files for Elite we will need to have an internet connection. Alternatively you can copy the files over from a USB memory stick. To configure network access, double-left click on the desktop icon named !Configure and then double click the Network icon on the right. From here there are just a few settings that need to be enabled for each of the three options:



Acorn Universal Networking

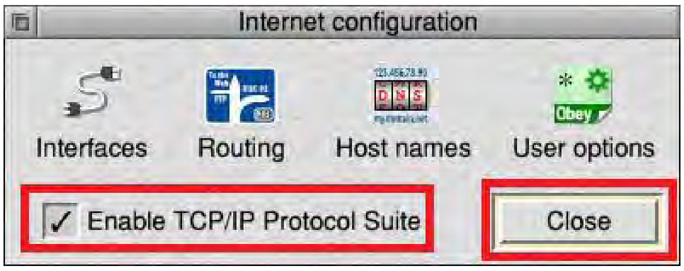
✓ Enable AUN

| | |
|---|---|
| This station | 0 |
| File server | 0.254 |
| Print server | 0.235 |



Acorn Access

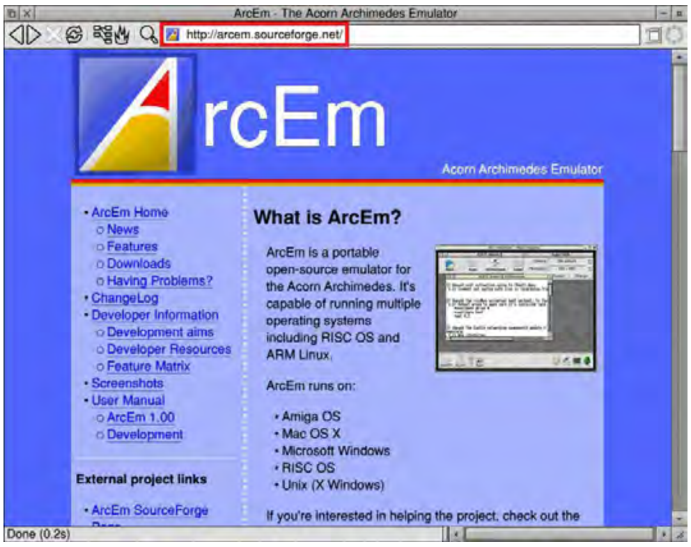
✓ Enable Access



Internet configuration

✓ Enable TCP/IP Protocol Suite

To obtain ArcEm load the !NetSurf browser, navigate to http://arcem.sourceforge.net and download the latest version via the "get it here" link. If you have trouble finding it type the following into the address bar on !NetSurf: http://sourceforge.net/projects/arcem/files/latest/download ?source=files



ArcEm - The Acorn Archimedes Emulator

http://arcem.sourceforge.net/

ArcEm

Acorn Archimedes Emulator

- ArcEm Home
  - News
  - Features
  - Downloads
  - Having Problems?
- ChangeLog
- Developer Information
  - Development aims
  - Developer Resources
  - Feature Matrix
- Screenshots
- User Manual
  - ArcEm 1.00
  - Development

External project links

- ArcEm SourceForge

What is ArcEm?

ArcEm is a portable open-source emulator for the Acorn Archimedes. It's capable of running multiple operating systems including RISC OS and ARM Linux.

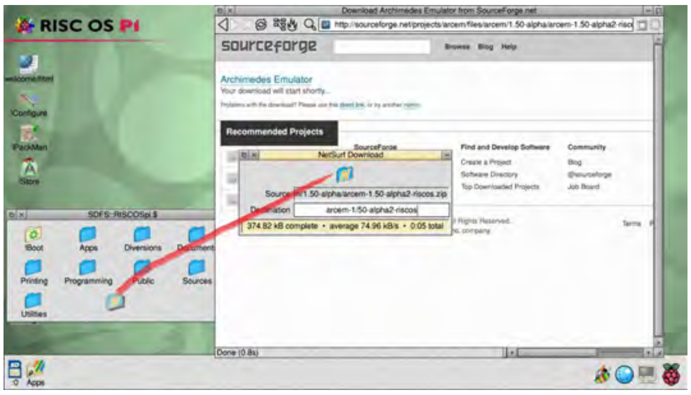
ArcEm runs on:

- Amiga OS
- Mac OS X
- Microsoft Windows
- RISC OS
- Unix (X Windows)

If you're interested in helping the project, check out the

Done (0.2s)

Note that !NetSurf may display an error saying Bad type. This is because it does not recognise the automatic download that the Sourceforge site tries to launch. Click on the direct link in Sourceforge and the download will start.



Download Archimedes Emulator from SourceForge.net

http://sourceforge.net/projects/arcem/files/arcem/1.50-alpha/arcem-1.50-alpha2-risc

Archimedes Emulator

Your download will start shortly...

NetSurf Download

| | |
|---|---|
| Source | n/1.50-alpha/arcem-1.50-alpha2-riscos.zip |
| Destination | arcem-1.50-alpha2-riscos |

374.82 kB complete • average 74.96 kB/s • 0:05 total

Tell RISC OS where you want the downloaded file to go by first double clicking the SD card icon in the bottom left, then use button 1 to drag the icon within the NetSurf Download box into the SD card window (as above).

The zig zag on the downloaded file is because it is a zip file (it has other files compressed inside it). The next thing we need to do is create a new folder to unzip the emulator into. Middle click (mouse wheel) somewhere in the SD card folder and a context menu will appear.

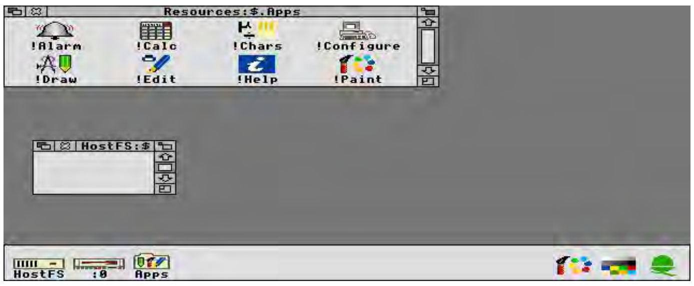
Choose New Directory > (move the mouse over the arrow), type ArcEm into the box and double click the new ArcEm folder. An empty window will display, Double click on the zip file and you will see the files inside it (this is the emulator program). Select all of the emulator program files and drag them into the ArcEm folder.

Before we can run the emulator we need to give it a ROM file. This is an image of the original Archimedes operating system that the emulator will boot up into. We will use RISC OS v3.10. Open !NetSurf, navigate to http://home.tiscali.nl/~jandboer/ and download the file called support2(1096K). We now need to get the ROM file out of the zip file. Double click the support2 file. You will see an icon called !A310Emu. It's actually a program, so don't double click it – double clicking on programs runs them. Hold down the Shift key and then double click it. You'll see another window with an OS folder. Double click the OS folder and now you'll see the file we want. It's named ro310. Drag it into the SD card window.

We now need to change a few properties on this file so that our ArcEm program will recognise it. These can each be done by middle clicking (mouse wheel) on the ro310 file. First we must remove its access protection (File 'r0310' > Access > Unprotected), then set its file type to Data (File 'ro310' > Set type > type Data) and finally rename the file to ROM (File 'ro310' > Rename > type ROM in capitals).

Open the ArcEm folder again. Hold down the Shift key and double click the !ArcEm icon. That will show the ArcEm program files. Drag the ROM file here.

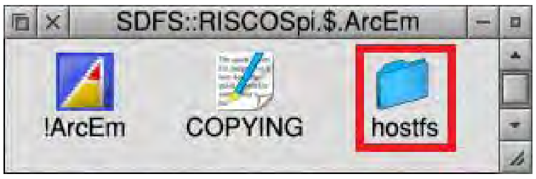
We can now run the emulator to see if it works. Double click the !ArcEm icon. The screen will go black for a moment and you will see the actual boot up sequence of the original Archimedes computer.



You will immediately notice that the screen is a little stretched. To fix this press F12 and a small command prompt will appear at the bottom of the screen. Type wimpmode 28 and press Enter twice. Now you should have a 4:3 screen ratio (640×480 at 256 colours).

You can make screen mode 28 permanent by setting it as the default screen mode in the Archimedes configuration. First double click on the Apps icon in the bottom left and then double click !Configure. It will appear in the task area in the bottom right. Click on it and then click the screen icon. Set the Default screen mode setting to 28. You can click where the current number is and edit it as text.

In the bottom left corner of the screen you'll see an icon named HostFS. This corresponds to the hostfs folder back in Raspberry Pi land that you may have noticed before (see below). Anything you put in here can then be accessed by the emulated Archimedes. Guess what we need to put in here? The game!

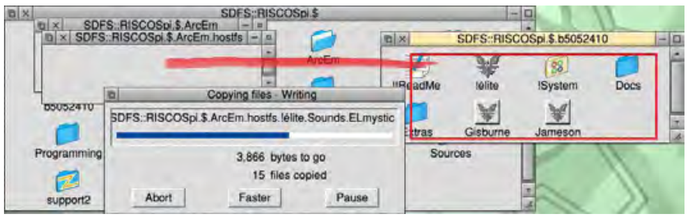


The emulator does not shut down very gracefully, if you middle click on the Acorn icon (bottom right) you can choose Exit from the context menu. But this just causes a hang. Until this bug is fixed you need to power cycle your Pi (turn it off for a second, then back on again). It is perfectly safe to do this.

When the Pi has rebooted open !NetSurf again and navigate to http://www.iancgbell.clara.net/elite/arc/
The file we want is marked **Click here to download Arc Elite (419 Kbytes)**.

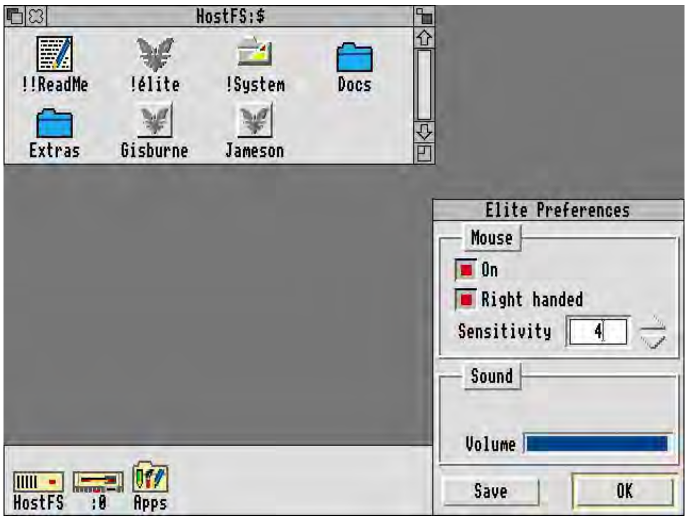
Drag the file into the SD card window, just like the other downloads we did, and you'll now see a file called **b5052410**. We need to change the type of this file to Zip so that we can extract the game. To do this middle click (mouse wheel) on the file and select File 'b5052410' > Set type and enter Zip.

Now we can extract the zip file into ArcEm's hostfs folder. Open the ArcEm folder again, then the hostfs folder. Double click the b5052410 zip file. You will now see the game files. Position the hostfs and b5052410 windows so that they are side by side (drag the title bar). Drag a box around all the game files and finally drag them all into the hostfs folder. You will then see a copying dialog box as shown below.


Copying files - Writing
SDFS::RISCOSpi.$.ArcEm.hostfs.!elite.Sounds.ELmystic
3,866 bytes to go
15 files copied

Run !ArcEm again and this time the game will be available. When you get to the Archimedes desktop, double click the HostFS icon and you should now see the game. Double click the icon named !elite to start the game. You will be asked if you have a high resolution monitor, press Y. The game will appear in the bottom right corner as an icon and probably named Jameson (the default commander name). Click this and you can then start playing!

When in the game you can press F12 to go back to the desktop. If you middle click (mouse wheel) the Jameson task icon you can get access to an Elite Preferences menu where you can configure the game settings (see below). I would suggest turning down mouse sensitivity to 1, or disabling the mouse altogether and playing the game with keys. That way the mouse works when you're in the menus and you use keys for flight. There is also a save game menu you can access from here.

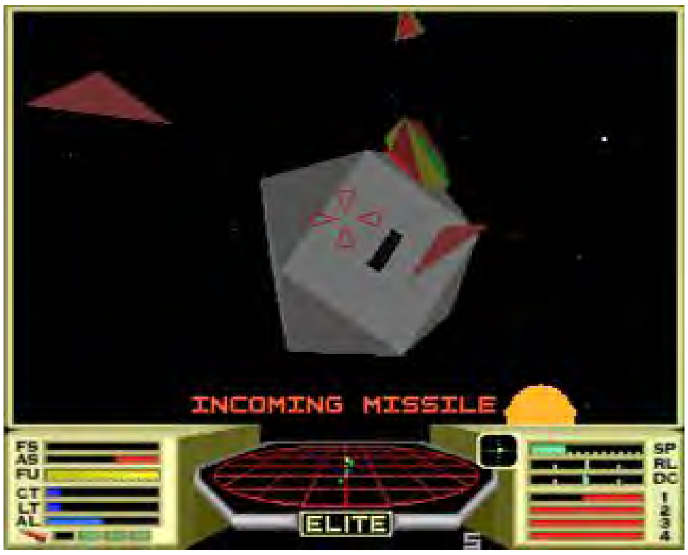


Elite Preferences

Mouse
- 🟥 On
- 🟥 Right handed
- Sensitivity 4

Sound
- Volume

In the Docs folder (above) there are some documents that come with the game that you can read. The basic idea in Elite is that you start off with a really poor space ship and by trading goods, narcotics, radioactive material etc you can earn money to buy better stuff for your ship, or buy a whole new ship.



If you want to fly using the mouse, this is a skill in itself. I recommend you press the Caps Lock key to turn on an auto centering feature which makes it 100% easier. Mouse button 1 is the fire button while buttons 2 and 3 are for speed up and slow down respectively.

On the keyboard the flight keys are S, X for pitch and <, > for roll, ? and Space bar for speed up and slow down and A is fire. To jump to another star system, press F6, choose the planet you want and then press H (for hyperspace). This will give you a 10 second count down to your jump. Manually docking with space stations is especially hard. To help, here is a video of someone doing this on the BBC version of Elite: http://youtu.be/X0czVxiEqNM

The basic idea is that you need to find the space station, get yourself lined up with its letter box shaped entrance, then fly into it as straight as you can. If you get it wrong you're dead (the game is actually that brutal). Here is how I usually do it. After you do a hyperspace jump you'll see a planet somewhere near you. Head towards it and press J to jump. If another ship gets near you a message saying MASS LOCKED will appear – you can't fire your jump drive when something else is near you. If it's a pirate you'll need to fight. Often getting them to crash into your shields is enough to kill them. Keep heading towards the planet. It should get bigger as your approach.

If you look at the game screen shot to the left, you'll notice there is a square box with a green cross hair, just below the E in MISSILE. This is used to help you locate the space station. It will show as a white dot if the station is in front of you and a red dot if behind. Try and get the white dot in the centre of the cross hair. That then means you're flying directly towards the space station. When you get near it, start slowing down and get ready to do what you saw in the video above. Fly roughly near the station, slow right down, then point towards the centre of the planet, speed up again and fly towards it for a few seconds. Slow down again, now point yourself back to the station and you should now see the entrance is pointing towards you (the entrance always points at the centre of the planet). Speed up again to about 1/3 and head in for docking. You might need to repeat this manoeuvre a few times to get yourself properly lined up. When you're happy with your alignment, you can try and match your rotation with the rotation of the space station. What I tend to do is get quite close, wait for the letter box to be horizontal and then floor it through!

To really get into the Elite world, you can read "Elite: The Dark Wheel", a short novella by Robert Holdstock. It's available at www.iancgbell.clara.net/elite/dkwheel.htm.

Watch out for the Vipers, Commander Jameson!

The MagPi wishes to thank:
Ian Bell (www.iancgbell.clara.net/elite),
David Braben (www.frontier.co.uk),
3QD Developments Ltd (www.riscos.com),
Castle Technology Ltd,
and RISC OS Open Ltd (www.riscosopen.org)
for their support with this article.

**Raspberry Pi for Dummies**
Sean McManus & Mike Cook
Wiley

Embrace the exciting new technology of Raspberry Pi!
Raspberry Pi for DUMMIES
by Sean McManus and Mike Cook
ISBN: 978-1-1185-5421-0
$24.99 / £17.99 / €21.60
WILEY

Raspberry Pi for Dummies is a great book for anyone new to the Raspberry Pi or looking to invest in purchasing one.

The book is an introduction to many aspects of the Raspberry Pi broken down into six parts, I-VI, covering everything from getting to grips with Linux and photo-editing to basic programming and an introduction to circuits and soldering. It also details great software packages for the Raspberry Pi and inspiring projects such as a weather station and a talking boat! It really gets the brain juices flowing as you realise there is no limit to what you can do with the Raspberry Pi!

The really nice thing about Raspberry Pi for Dummies is that it gives you an introduction to the many different facets of the Raspberry Pi and does so in an easy, understandable way. The chapter on circuits and soldering is a nice addition as most introduction books tend to detail the programming side of the Raspberry Pi and the potential projects without introducing the basic skills many adults and children do not have.

This book is an ideal start to any Raspberry Pi reference library.

The MagPi and Wiley are pleased to offer readers a 30% discount. To claim, order from www.wiley.com and quote promo code **VBD57**.

**Super Scratch Programming Adventure**
The LEAD Project
No Starch Press

Super Scratch Programming Adventure is an excellent book for kids and anyone who wants to learn basic programming skills as it teaches Scratch in a fun and accessible way.

It is a comic book-cum-instruction manual, which teaches Scratch through story. Our heroes are Scratchy (the cyber-space living cat), Mitch (a computer science student) and the Cosmic Defenders; Gobo, Fabu and Pele, who we must help fight the Dark Wizard and his Dark Minions. As you work through the comic you learn simple Scratch techniques to defeat the Dark Wizard resulting in some really cool mini games. At the end of the book there is a bonus section (the best part of any game), which shows you how to share your Scratch project online and how to use Scratch to write programs for the PicoBoard sensor board. (http://wiki.scratch.mit.edu/wiki/PicoBoard).

The great thing about this book is "Scratchy's Challenge!!" at the end of each stage (chapter). These challenges get you to think outside the box and come up with ways to improve what you have just created and how you would use these skills in other projects.

SUPER SCRATCH PROGRAMMING ADVENTURE!
COVERS VERSION 1.4
LEARN TO PROGRAM BY MAKING COOL GAMES!
THE LEAD PROJECT

This is a great book to start your programming adventure!

No Starch Press are offering The MagPi readers 40% off all ebooks between June 1st and June 30th. To claim, order from www.nostarch.com and quote **READMAGPI**.

# 6 - Linking to FORTRAN77

DIFFICULTY : ADVANCED

W. H. Bell

MagPi Writer

When numerical functions or algorithms are already written in another language, it can be faster to link to them rather than re-write the algorithm in C. This month's article looks at one example, linking C to FORTRAN77. Before going any further, here is a solution to the challenge problem given in Issue 9.

## Challenge solution

Start from the source code given in the Issue 9 article. Then open the file `histogram.c` file and add `#include <math.h>` at the top of the file. Then append the two functions,

```
double histMean(const Histogram *hist) {
  unsigned int i;
  double xMean = 0.;
  for (i=1;i<=hist->nBins;i++) { /* Skip underflow and overflow */
    xMean += hist->binContents[i]/hist->nBins;
  }
  return xMean;
}

double histStdDev(const Histogram *hist) {
  unsigned int i;
  double xMean, xStdDev = 0.;
  xMean = histMean(hist);
  for(i=1;i<=hist->nBins;i++) { /* Skip underflow and overflow */
    xStdDev += pow(xMean - hist->binContents[i],2)/(hist->nBins-1);
  }
  if(xStdDev > 0.) xStdDev = sqrt(xStdDev);
  return xStdDev;
}
```

Next, open the header file `histogram.h` and before `#endif` add the two function definitions,

```
/* return the mean value. */
double histMean(const Histogram *hist);

/* return the standard deviation. */
double histStdDev(const Histogram *hist);
```

These two functions can be tested by adding `printf` statements to `main.c`,

```
 printf("Mean=%f\n",histMean(&h));
 printf("StdDev=%f\n",histStdDev(&h));
```

The next part of the problem involves reusing the last example given in Issue 6. Instead of just writing the `ramUsed` value to the output text file, create a histogram before the event loop. Since the memory used is reported in bytes, the scale of the histogram will have to be adjusted to match the scale of the data. Then inside the for loop add the `fillHist` command. Finally, after the for loop add the `histMean`, `histStdDev` and `saveHist` function calls.

## Linking to FORTRAN77

FORTRAN has been around for a long time. The 77 standard was used with punch cards and for this reason the spacing at the start of a line has special meaning. If a character is present at the start of a line, the whole line is considered a comment. Source code should start on the seventh character. If a line is too long, a character six spaces from the left can be used to signal that the line is a continuation of the previous line.

Use a text editor, such as nano or emacs, to create a file called `fortran.for`. Then append,

```
      SUBROUTINE COMMONS
      IMPLICIT NONE
      INCLUDE 'FORTRAN.INC'

      INTEGER I,J

c     Print the contents of the INTEGER array.
      DO I=1,3
         DO J=1,2
            PRINT *,'COMMONS/INTARRAY(',I,',',J,')=',INTARRAY(I,J)
         ENDDO
      ENDDO

c     Print the contents of the REAL array.
      DO I=1,2
         DO J=1,3
            PRINT *,'COMMONS/REALARRAY(',I,',',J,')=',REALARRAY(I,J)
         ENDDO
      ENDDO

c     Print the string.
      PRINT *,'SOMESTRING=',SOMESTRING

      RETURN
      END
```

The subroutine commons is equivalent to a void function which has no arguments. Data is passed into the subroutine by using the common block, which is called `FORCOM`. The common block is defined in the `FORTRAN.INC` file, which is included in a similar way as C header files. The common block is a static global block of memory. This means that the location of the associated memory is fixed when the program compiles. The subroutine loops over the two 2-dimensional arrays and prints the values on the screen. The `INTEGER` type is equivalent to `int` in C. The `REAL` type is equivalent to `float` in C. The arrays `INTARRAY` and `REALARRAY` are defined as being part of the common block. Unlike C `structs`, they can be used simply by defining the common block in the same file. The order of the memory of the variables within the common block is the same as within a C `struct`. The order of the array elements is not the same as C. For a 2-dimensional array, the first and second array indices are the second and first indices in C. For a 3-dimensional array, the outer two indices are swapped with respect to C. This becomes clear if the example given in this tutorial is extended. Now add,

```
      FUNCTION CALL_BACK(A,NAME)
      IMPLICIT NONE
      REAL A, C, CALL_BACK, MULT_A
      CHARACTER*(*) NAME

c     Print the string passed into this function
      PRINT *,'CALL_BACK() NAME=',NAME

c     Call a C function mult_a_ to multiply A by some factor
      C = MULT_A(A)

c     Print the value of C returned from mult_a_
      PRINT *,'CALL_BACK() C=',C

c     Return the value of C.
      CALL_BACK = C
      RETURN
      END
```

to the end of `fortran.for`. This FORTRAN function is equivalent to a C function with two parameters and a `float` return value. Notice that the type of the two parameters is not defined within the parentheses of the function, but is instead defined afterwards. This function demonstrates how FORTRAN77 can be used to call a C function. The C function is called in the same way a FORTRAN function is called and returns a `float` value which is stored in the variable C. The value returned is then printed to the screen.

Now that `fortran.for` is complete, create a file called `FORTRAN.INC`. Then append,

```
c     A collection of variables contained in a common block
      INTEGER INTARRAY(3,2)
      REAL REALARRAY(2,3)
      CHARACTER*50 SOMESTRING
      COMMON/FORCOM/INTARRAY,REALARRAY,SOMESTRING
```

Now that the FORTRAN code has been written, open a new file called `fortran.h`. Then add,

```
#ifndef FORTRAN_H
#define FORTRAN_H

/* The C side of the fortran common block definition. */
typedef struct {
  int intarray[2][3];
  float realarray[3][2];
  char somestring[50];
} forcom;

/* The memory is allocated by the fortran code.  Therefore used extern here to cause forcom_ to
   be resolved to the FORTRAN code at link time. */
extern forcom forcom_;

/* Fortran subroutine COMMONS */
void commons_(void);

/* Fortran function CALL_BACK */
float call_back_(float *,char *, int);

/* A function called from fortran code */
float mult_a_(float *);

#endif
```

to `fortran.h`. The `struct` definition is equivalent to the FORTRAN common block. Similar to FORTRAN, a single instance `forcom_` is defined as `extern`. The use of `extern` means that while C defines this common block as a symbol it does not actually define an instance of the `struct` in memory. Instead of using an instance of the C `struct`, the program uses the FORTRAN common block `FORCOM`. If the instance was not `extern`, then the C program would define `forcom_` in a different memory location which did not match that used by the FORTRAN subroutine. Each of the variables or functions in this file are written with lower case characters and end with an underscore character. This is because FORTRAN is case insenstive when it is compiled. The FORTRAN compiler also adds a trailing underscore character to the end of the variable or function names. The function `call_back_` has one more variable than the FORTRAN version. This is because unlike C, FORTRAN keeps track of the length of the character array with one integer per string parameter. In the case of C, the string terminating character (`\0`) is used instead. This means that FORTRAN strings can be one character longer than C strings.

If the `gfortran` compiler is not already present, type:

```
sudo apt-get install -y gfortran
```

to install it. Then compile `fortran.for` by typing,

```
gfortran -c fortran.for
```

This will create a `fortran.o` file. The symbols defined in the machine code can be read using `nm`. Type,

```
nm fortran.o
```

to list the symbols. U means that the symbol is not defined, whereas T implies that the symbol has been defined. Try looking for the function names and the name of the common block variable. When C code is compiled by the gcc compiler, the resulting names have an underscore prefix. This means that the declarations in fortran.h will match the names given in fortran.for. The other symbols given in the nm output relate to system functions.

Now that the C header file has been written, create a new file called main.c. Then add,

```
#include <stdio.h>
#include "fortran.h"

/* A function to fill the FORTRAN common block. */
void fill_common(void) {
  int i,j;

  for(i=1;i<=2;i++) {
    for(j=1;j<=3;j++) {
      /* Fill the int array */
      forcom_.intarray[i-1][j-1] = i+j*10;

      /* Fill the float array */
      forcom_.realarray[j-1][i-1] = j+(float)i*0.001;
    }
  }
  /* Fill the character array */
  sprintf(forcom_.somestring,"C string");
}

/* A function provided for the FORTRAN function CALL_BACK to call. */
float mult_a_(float *a) {
  return (*a)*10.;
}

int main() {
  float a=3.14159;
  char name[]="Wrapper";
  int size;

  /* Fill common block FORCOM */
  fill_common();

  /* Print values contained in common block */
  commons_();

  /* Call the fortran function CALL_BACK(A,NAME) */
  size=sizeof(name)/sizeof(char);
  call_back_(&a,name,size);

  return 0;
}
```

When the main() function runs, the fill_common() function is used to fill the arrays of the FORTRAN common block. The for loops are declared in the same way as the FORTRAN do loops. Unlike C, FORTRAN array indices start from 1 by default. The array indices are the opposite of those used in fortran.for, the first index is the second and the second index is the first. The string "C string" is then copied into the somestring character array. The main function then calls the FORTRAN subroutine commons_ which prints the values stored in the common block. The last function call in main() is the call_back_ function. This is a FORTRAN function in fortran.for, which calls the mult_a_ function defined in the main.c file. Unlike C, FORTRAN uses pointers for all variables passed into functions. The only exception to this is the size of character arrays, which is not visible within the FORTRAN code.

The easiest way to compile the two files together and produce an executable is to use a Makefile. In the same directory as the `fortran.for`, `fortran.h`, `FORTRAN.INC`, and `main.c` files, create a new file called `Makefile`. Then add,

```
CC=gcc
F77=gfortran
TARGET=Wrap
OBJECTS=main.o fortran.o

LIBS = -lgfortran

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

clean:
	@rm -f *.o *~

veryclean: clean
	@rm $(TARGET)

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

%.o: %.for
	@echo "** Compiling FORTRAN77 Source"
	$(F77) -c $<
```

The string `-lgfortran` includes `libgfortran.a` in the link command. The space before the lines should be one tab character. More information is given in the Issue 7 article on GNU make. Once the Makefile has been saved, type `make` to compile the code. Then type `./Wrap` to run the program.

## Challenge problem

The Standard Model of particle physics can be used to produce Higgs bosons in association with W and Z vector bosons from proton-proton collisions. These processes have been studied at the Large Hadron Collider (LHC). PYTHIA is a tool which generates final states, based on theoretical calculations and phenomenological models. The challenge is to use the FORTRAN PYTHIA6 (http://pythia6.hepforge.org/) to generate ten Higgs events with associated W and Z vector bosons.

Download the latest version of PYTHIA6 and use `gunzip` to unzip it. Then create a file called `main.c` containing,

```
#include <stdio.h>
#include "pythia.h"
int main() {
  int i, mstat;
  double ecm;
  ecm=14000.0;  /* Set the centre of mass energy to be 14TeV. */
  pysubs_.msel=17;  /* Select Wh and Zh production. */
  pyinit_("CMS","p","p",&ecm,3,1,1); /* Initialise PYTHIA with the setup and beam energy */
  for(i=0;i<10;i++) { /* Generate 10 events */
    pyevnt_(); /* Generate an event with PYTHIA */
  }
  mstat=1;
  pystat_(&mstat);  /* Print a report of what was generated */
  return 0;
}
```

Create a file called `pythia.h`, which contains the C declarations of the common block PYSUBS and the function definitions needed. The FORTRAN declarations of the common block and the functions are in the PYTHIA source file. Use make to compile the programs. PYTHIA takes quite a long time to compile. The solution will be given next time.

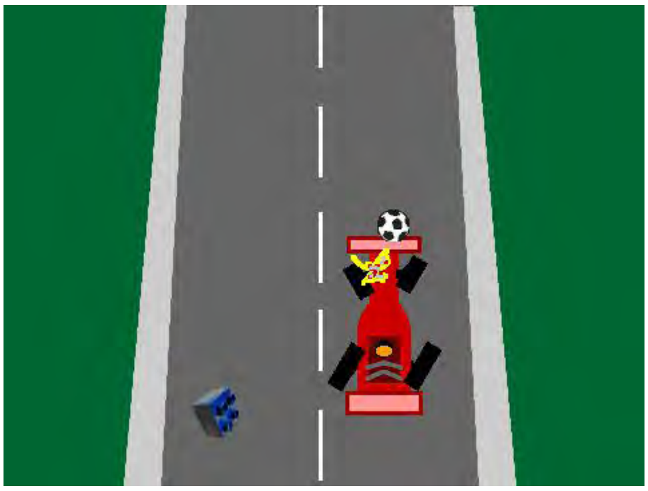
# Crash, bang, wallop!
# Create a simple racing car game

**DIFFICULTY : BEGINNER**

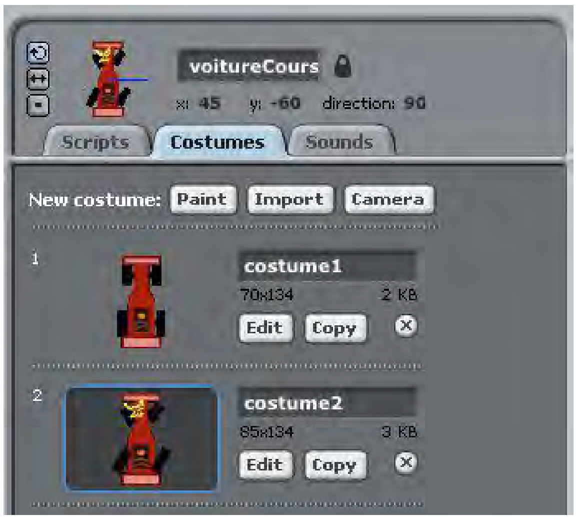
**W. H. Bell**

MagPi Writer

One way to start programming is to write simple video games. Scratch has several handy functions which speed up the construction of basic games. This article demonstrates how to produce a racing car dodging game. The game can be easily extended and the principles can be used when writing games using Python or C.



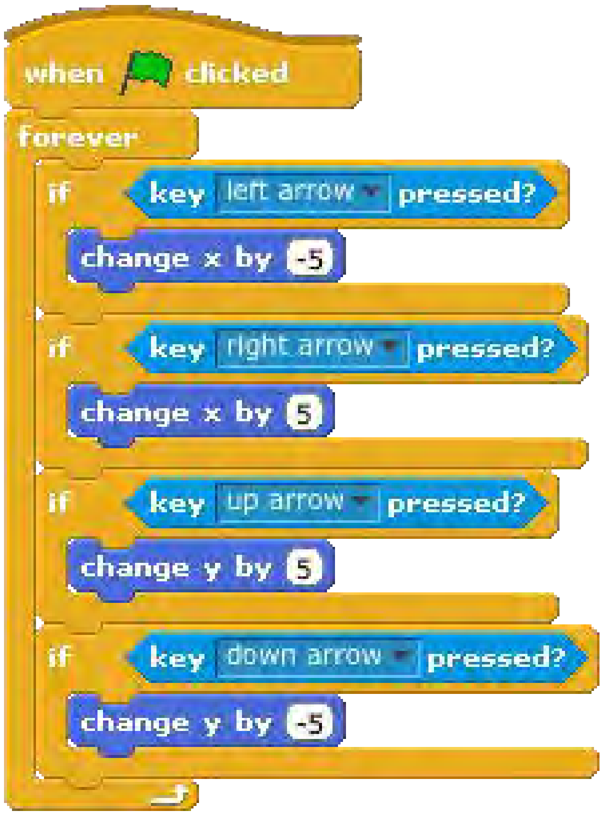
The idea of the game is to drive the car around several objects which scroll down the screen. The game ends if the car goes on the grass or touches one of the objects.



voitureCours
x: 45 y: -60 direction: 90
Scripts | Costumes | Sounds
New costume: Paint Import Camera
1 costume1 70x134 2 KB Edit Copy
2 costume2 85x134 3 KB Edit Copy

The first step to produce this game is to create or choose the sprites. The racing car was drawn with the Scratch sprite editor. To make the sprite symmetric, half was copied and then mirrored. Once the working car has been produced, go to the Costumes tab of the sprite and click on copy. Then click on Edit for the second sprite.

The second costume will be used for the broken car. The Scratch editor was used to make the car look broken. Before writing the script for the car, some objects are needed. In this case, the ball and the Lego piece were chosen from the Scratch sprite library. The library of sprites can be viewed by clicking on the folder icon, just below the stage.

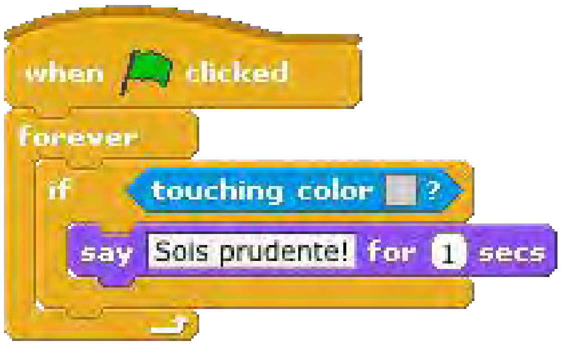


```
when [flag] clicked
forever
    if key left arrow pressed?
        change x by -5
    if key right arrow pressed?
        change x by 5
    if key up arrow pressed?
        change y by 5
    if key down arrow pressed?
        change y by -5
```

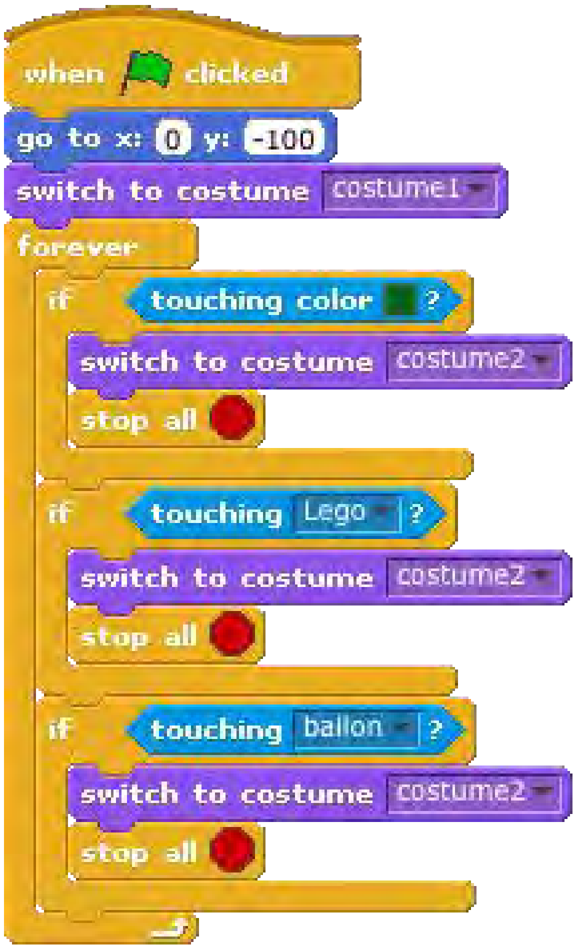
## Racing car script

The racing car is controlled by a script which has three pieces. These three pieces run in parallel, which allows the program to check the conditions in each piece at the same time. The first block checks the keyboard. If the cursor keys are pressed, the car moves left, right, up or down the screen. The centre of the stage corresponds to x=0, y=0.

The second piece of the script checks to see if the car has touched the grey colour at the side of the road. This colour has to be the same as the colour used on the stage for the program to work. When the racing car touches the edge it says "Sois prudente!" (be careful!) for one second.

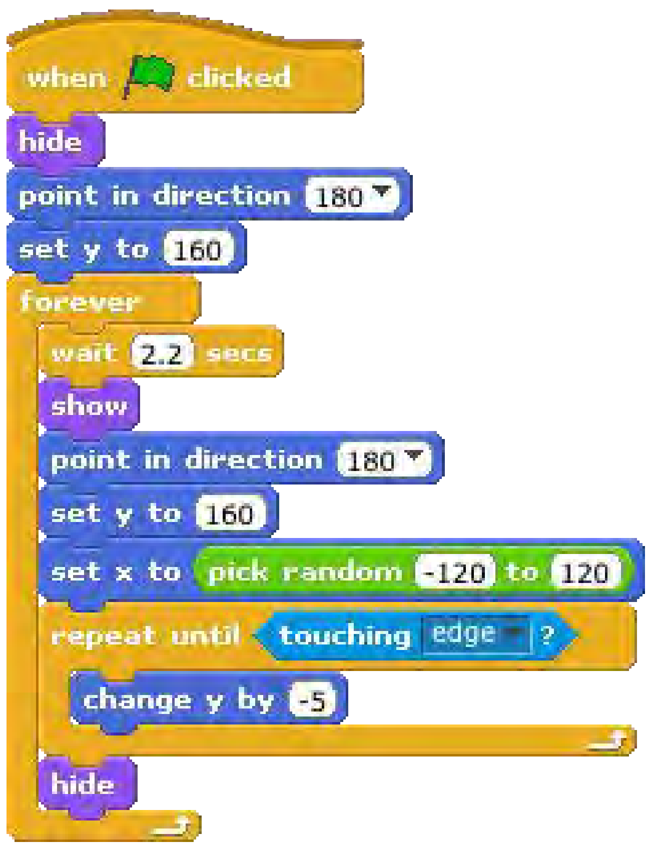


```
when [flag] clicked
forever
    if touching color [ ] ?
        say Sois prudente! for 1 secs
```



```
when [flag] clicked
go to x: 0 y: -100
switch to costume costume1
forever
    if touching color [ ] ?
        switch to costume costume2
        stop all
    if touching Lego ?
        switch to costume costume2
        stop all
    if touching ballon ?
        switch to costume costume2
        stop all
```

The last part of the script checks to see if one of the game over conditions has been reached. There are three conditions which cause the game to finish; (1) the car touches the green grass colour, (2) the car touches a Lego brick sprite, or (3) the car touches a ballon (ball) sprite. In each case the racing car sprite is changed to costume2, which is the broken car. 'Stop all' then stops the game. This script block also resets the car position to the bottom of the screen when the game starts. x=0 is the middle of the screen in the horizontal direction and y=-100 is the bottom of the screen in the vertical direction.



```
when [flag] clicked
hide
point in direction 180
set y to 160
forever
    wait 2.2 secs
    show
    point in direction 180
    set y to 160
    set x to pick random -120 to 120
    repeat until touching edge ?
        change y by -5
    hide
```

## Lego brick and ball

The lego brick and ball are controlled by the same program. To start writing the script for the Lego brick, click on the Lego brick sprite icon under the stage window. Then create the script given on the left of this page. When the script has been written, select all of it and click on copy. Then click on the ball and paste the script. For the ball, change the wait statement from 2.2 seconds to 3 seconds. As long as the two wait statements are not multiples of each other, the two objects will not arrive at the same time.

The script for the Lego brick and ball starts by hiding the sprite. The sprite is then put at the top of the screen, pointing down. The script then enters a loop. Each time the loop runs, the script waits and then shows the sprite. It is then moved to the top of the screen. The starting position is chosen at random, along the x-axis (horizontal plane). The sprite is then moved down the screen by 5 units at a time, until it touches the bottom of the stage. When the sprite touches the bottom of the stage it is hidden and the outer loop moves it back to the top.

## The stage

To complete the program, draw the stage image. Similar to the racing car, half of the stage was drawn, copied and mirrored. The green and grey colours used in the racing car script were used for the edge of the road and the grass.

## Possible extensions

There are several ways this program could be made better. The lines in the middle of the road could be changed for another image to create the illusion of movement. Sounds could be added for when the car moves or crashes. Some burning rubber marks could be added when the car slows down. Instead of a Lego brick and ball, try creating some other cars or cars that drive from the bottom of the screen to the top.

This game could be used as the basis for other games. For example, vertical scrolling space invaders is quite similar. Try swapping the racing car stage picture for a star field. Then add in some weird and wacky alien vehicles, which come racing down or drop things on the heroic space ship.

the python pit™

CLIENT SERVER
Model for using many computers

W. H. Bell

MagPi Writer

# Parallel calculations - part 2

DIFFICULTY : ADVANCED

Welcome back to the Python Pit. This article is a continuation of the tutorial started in Issue 10. If you have not already done so, it would be a good idea to read the parallel calculations article in Issue 10 before proceeding.

To test the programs in this tutorial, two short Python programs are needed. Create a file called launchBatchClient.py,

```
from FunctionCalculator import BatchCalculatorClient

if __name__ == "__main__":
  import sys
  if len(sys.argv) != 3:   # There must be three command line arguments
    sys.stderr.write("  Usage: " + sys.argv[0] + " <server> <port> \n")
    sys.exit(1)   # Exit with an error

  client = BatchCalculatorClient(sys.argv[1], sys.argv[2])
  client.loop() # Start a batch calculator client
```

Then create another file called launchBatchCalculator.py,

```
from FunctionCalculator import BatchCalculator

if __name__ == "__main__":
  calculator = BatchCalculator(socket.getfqdn(), 20000)
  calculator.initialise() # Start the server process.
  time.sleep(60) # Sleep for a minute to allow client server tests.

  # Tell any clients connected to the batch calculator
  # server to shutdown and then shutdown server.
  calculator.shutdown()
```

These two programs are for launching the client and server processes, which will allow parallel connections on several computers at once. For simplicity, save both programs in the same directory as the FunctionCalculator.py file created in the last tutorial.

Open the FunctionCalculator.py file in a text editor. Then add,

```
import socket
import time
import threading
```

to the top of the file. These import statements will be needed for the BatchCalculator, BatchCalculatorClient and other supporting classes. Before adding the main server and client processes, several supporting classes are needed. The first of these is a simple data class called StatusResult. This class contains the return values from a calculation and the associated status. Using a simple class also means that the object is mutable. Append the StatusResult class,

```
class StatusResult:
  cmd_status = 0
  result = 0.

  def __init__(self, status_val, result_val):
    self.cmd_status = status_val
    self.result = result_val

  def __str__(self):
    return "cmd_status: %d, result: %e" % (self.cmd_status, self.result)

  def __repr__(self):
    return "cmd_status: %d, result: %e" % (self.cmd_status, self.result)
```

to the end of FunctionCalculator.py. The __str__ and __repr__ return string representations of the class, which are useful for debugging.

The next supporting class to be added to FunctionCalculator.py is the BatchCalculatorThread,

```
class BatchCalculatorThread(threading.Thread):

  def __init__(self, sock):
    threading.Thread.__init__(self)
    self.sock = sock
    self.cmd = ''
    self.sr = StatusResult(0,0.)
    self.processing = threading.Event()
    self.processing.clear()
    self.setDaemon(True)

  def run(self):
    while True:
      self.processing.wait() # block until a command is given
      print "Thread \"%s\" sending \"%s\"" % (self.getName(), self.cmd)
      self.sock.send(self.cmd)
      self.sr.result = float(self.sock.recv(1020))
      self.sr.cmd_status = 2 # finished
      print "Thread \"%s\" recieved \"%s\"" % (self.getName(), self.sr.result)
      self.processing.clear() # ready for a new command

  def evaluate(self, cmd, sr):
    self.cmd = cmd
    self.sr = sr
    self.sr.cmd_status = 1 # processing
    self.processing.set()

  def processingCmd(self):
    return self.processing

  def cmd(self):
    return self.cmd

  def result(self):
    return self.result
```

This class inherits from the built in Python class Thread, such that it includes the functionality from the base class Thread. An instantiation of the BatchCalculatorThread class is given to each client which connects to the main server process in BatchCalculator, leaving the main server process to listen to other connections. The BatchCalculatorThread contains a run method which is called when the thread is started. The run method passes the waiting command to an associated client connection and then reads the result as a float. The wait and clear functions are used to stop and start the thread, such that it is only running while it has a new command to process. The other methods in the class can be executed at the same time as the run method. The evaluate method is available to pass a command to an associated client. The evaluate function uses the command status flag to block further calculation requests until the associated client has finished processing.

Now append,

```
class BatchCalculator:
  def __init__(self,host,port):
    self.host = host   # The name of this host, which is running the server
    self.port = port   # The port on this host, which is running the server
    self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.sock.settimeout(None)
    self.client_sockets = []   # A list of sockets associated with each client connection
    self.client_threads = []   # A list of BatchCalculatorThreads associated with each client
    self.shutdown_flag = False   # A flag to indicate server shutdown

  def initialise(self):
    try:
      self.sock.bind((self.host, self.port))
    except socket.error:
      return

    self.sock.listen(5)
    self.server_thread = threading.Thread(target=self.serve_forever)
    self.server_thread.setDaemon(True)
    self.server_thread.start()   # Start the Daemon function
    print "Server running on %s and listening on %d" % (self.host, self.port)

  def serve_forever(self):   # Daemon function
    while not self.shutdown_flag:   # Keep running until the shutdown flag is true
      print "Listening for a connection"
      try:
        request, client_address = self.sock.accept()   # Listen for client connection
      except socket.error:
        return
      self.client_sockets.append(request)   # Append this socket to the list
      print "Received connection from ", client_address

      client_thread = BatchCalculatorThread(request)   # Create a thread to handle the connection
      client_thread.start()   # Start the thread
      self.client_threads.append(client_thread)   # Add this running thread to the list
      print "Started an associated thread."

    if self.shutdown_flag == True:   # Tell the user the server is shutting down
      print "Listening socket shutting down"

  def disconnect(self):
    self.shutdown_flag = True # Stop listening
    print "There are %s connections active" % len(self.client_sockets)

    while len(self.client_sockets) > 0:   # Loop over clients
      sock = self.client_sockets.pop()
      print "Wrote DISCONNECT"
      sock.send("DISCONNECT") # Tell client to disconnect

  def shutdown(self):
    self.shutdown_flag = True # Stop listening
    self.sock.close()
    print "There are %s connections active" % len(self.client_sockets)

    while len(self.client_sockets) > 0: # Loop over clients
      sock = self.client_sockets.pop()
      print "Wrote EXIT"
      sock.send("EXIT") # Tell client to shutdown
```

to the end of the FunctionCalculator.py file. This class is similar to the SimpleServer class implemented in the previous tutorial and listens for connections from client processes. The class has an initialise function for starting the server, a disconnect function for telling the clients to disconnect and a shutdown function for telling the clients to shutdown. Both the disconnect and shutdown functions stop the server process, which listens for new connections. The disconnect function leaves the client processes running, such that the server program can be restarted. One more function is needed to complete this class. This is the evaluate function, needed to distribute the calculations. The evaluate function will be discussed next time.

To connect to a BatchCalculator server, a client class is needed. At the end of FunctionCalculator.py append,

```
class BatchCalculatorClient(FunctionCalculator):
  def __init__(self, host, port):
    self.host = host   # The server host name
    self.port = port   # The server port number, as a string
    self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Socket for connection
    self.connectionOpen = 0   # A socket to flag if the connection is open or not
    self.reconnectTime = 2   # The reconnect sleep time in seconds
    self.rereadTime = 2   # The re-read time in seconds

  def loop(self):
    while True:  # Keep looping until told to shutdown
      if self.connectionOpen == 0:  # If there is no connection, try to connect to the server
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        HOST, PORT = self.host, int(self.port)   # Prepare host and socket pair
        try:
          # Connect to the server
          print "Connecting"
          self.sock.connect((HOST, PORT))
        except socket.error:  # If the connection fails, close the socket and sleep
          self.sock.close()
          time.sleep(self.reconnectTime)
          print "Failed to connect"
          continue
        self.connectionOpen = 1   # Set the connection open flag

        # Read instructions from the server
        msg = self.sock.recv(1024)
        if msg is None:   # If there is no instruction yet, sleep
          time.sleep(self.rereadTime)
          continue

        msg = msg.strip()   # Remove leading and trailing characters
        if msg == "EXIT":   # Exit the while loop, if requested
          break

        if msg == "DISCONNECT":   # Disconnect from the server if requested
          self.sock.close()
          self.connectionOpen = 0
          time.sleep(self.reconnectTime)
          print "Disconnecting"
          continue

        result = "%e" % self.evaluate(msg)   # Call the evaluate method of the base class
        self.sock.send(result)   # Send the result back to the server thread
```

This class inherits from the FunctionCalculator class, such that it has an evaluate function. When the client process is started by calling the loop function, the client tries to connect to the server. If it is not successful, it sleeps and then tries again. Once it has a connection it waits for instructions from the server. If the string it receives is EXIT, the client shuts down. If the string it receives is DISCONNECT, the client closes its connection to the server and sleeps. If the string is not EXIT or DISCONNECT, it is evaluated as Python and the result is returned to the server.

This tutorial can be tested with one or multiple Raspberry Pis or computers connected to the network. Follow the instructions given in the previous tutorial, to find the IP addresses or host names, then change the address in launchBatchCalculator.py as needed. Type python launchBatchCalculator.py to start the server process. On another computer, or the same computer, type python launchBatchClient.py 192.168.1.11 20000 where the IP address should match that of the server.

The BatchCalculator evaluate function and an example calculation will be added next time.

# Feedback & Question Time

Reading The MagPi makes me happy :D Happy Birthday and thanks for all the hard work!

**Clive**

Seriously, Happy Birthday to The Mag Pi (& the team). Without your fantastic efforts the Raspberry Pi community would be a far less vibrant place.

**Pete Lomas**

Great stuff! I've been using Linux on and off for years, but I'm not really fluent in it. I find much of interest and plenty of good tips in W. H. Bell's articles in The MagPi.

**J Beale**

I'm looking forward to the next issue of The MagPi. Thanks for reducing the download file size. It's much easier to manage.

**Steve**

I thought I would just let you know how fortuitous The MagPi magazine and the Raspberry Pi was to me.

I was, and still am, recovering from a liver transplant which resulted in me being in and out of hospital with the rest of the time stuck at home. Seeing The MagPi magazine rekindled my interest in computers and the real world which I had last visited in the days of the Spectrum and BBC Micro computers. Long forgotten programming skills in Linux and an even older hobby, electronics, came tumbling back as I was able to fill in the long hours and weeks in hospital and now at home with developing the Pi in various ways.

I'm not one of the younger generation who are being targeted in the hope to engage them in what computers really are all about, but am a reconstituted student eager to use up my 'extra' time in the most useful ways possible. OK 'Skutter' may not exactly fall into that category but completing it gave me satisfaction and a lot of fun! It can been seen at http://goo.gl/WRz8h.

During this time I've also gained my amateur radio licence and am now working on ways to bring the unique features of the Pi to bear on that past-time. There are many amateur radio operators around the world who use the Raspberry Pi for a wide variety of uses related to their licence and it often features in the hobby magazines.

**Kevin Cook, M6KBD**

*[Ed: Thanks for this note Kevin. Your 'Skutter' build is very impressive.]*

One of the best magazine series I've ever read! As long as you keep making them, I'll keep reading them. Reading through an issue I always get an "Aha!"-feeling. I keep re-reading back issues to get snippets of goodies that I remember having read somewhere. Just keep soaring ;-)

**Jon Brohauge**

I only just found out The MagPi existed! Loads of back-reading to do, looks great :) Congratulations on your first year.

**Raizor**

# Project Builder?

## Connect your Raspberry Pi to the World

**Visit cpc.co.uk for full product details**

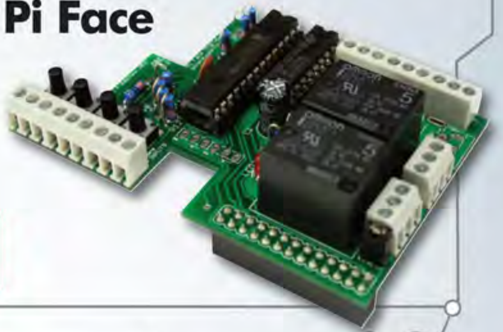
### Gertboard

**GPIO - Expansion Board**

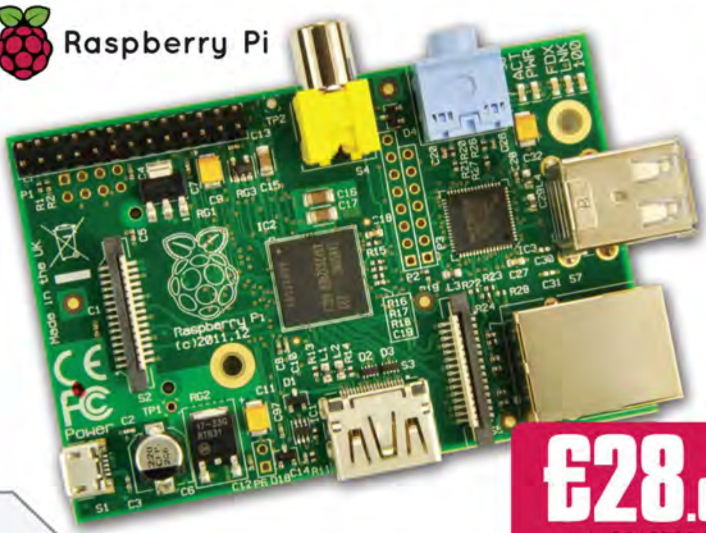
**£36**
code SC1282843

### Digital Pi Face

**GPIO - Expansion Board**



**£24**
code SC1282743

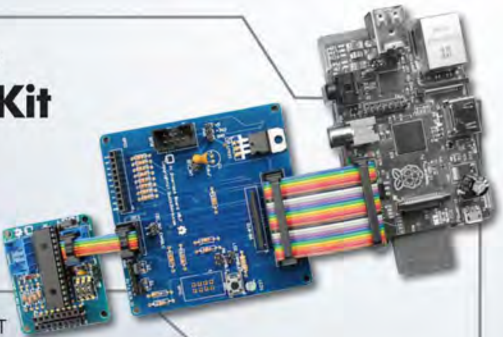


Made in the UK

**£28.07**
code SC1259043

Quick2Wire

### Combo Kit



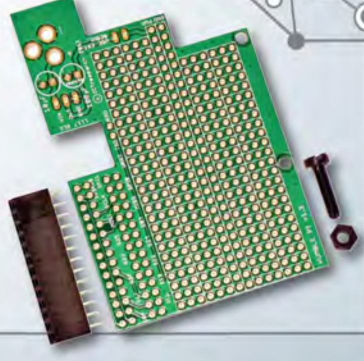
**£22.74**
code SC1294143

All prices include VAT

CISECO

### Humble Pi

**Prototyping Board**



**£4.91**
code SC1287143

### For Everything Raspberry Pi...

- Kits and Cases
- Expansion, Breakout and Prototyping Boards
- Vast Range of Components
- WiFi Dongles & HDMI to VGA Adaptors
- Video and Audio Leads
- Keyboards, Mice and Monitors
- GPIO Ribbon Cables
- Programming and User Guides

CPC

cpc.co.uk

FREE DELIVERY

Conditions apply, visit cpc.co.uk