ISSUE 11 - APR 2013

Get printed copies at themagpi.com

The MagPi™

A Magazine for Raspberry Pi Users

This Issue...

Control Your Heating System
Make A Wireless Pi-Point
Print From Your Pi
Create An Intranet
...and much more

Win a limited edition BLUE Raspberry Pi!

MINECRAFT PI EDITION

How to Install, Play and Program

ISSN 2051-9990

Raspberry Pi is a trademark of The Raspberry Pi Foundation.
This magazine was created using a Raspberry Pi computer.

CC BY NC SA

The MagPi™

http://www.themagpi.com

Welcome to issue 11,

The Raspberry Pi computer opens up many opportunities. It can be used as a web server, print server and as a wireless access point. If the problem can be solved with a low power Linux computer it can be solved with a Raspberry Pi.

This month we present some more articles which demonstrate how versatile a Raspberry Pi computer is. There are hardware and automation projects and several programming articles to choose from. We are pleased to provide a first assembly article and hope to see some bare-metal coding in the future.

Lastly, thank you to all of those who have ordered Volume 1 (Issues 1-8) as part of the Kickstarter project and through PayPal. We are very grateful for your continued patience. We are also pleased to announce that the binders and stickers have been printed and several of the issues from volume one have been submitted for printing. We hope to be shipping Volume 1 very soon.

If you can dedicate some time to help with article layout or testing, please email editor@themagpi.com.

Chief Editor of The MagPi

The MagPi Team

Ash Stone - Chief Editor / Administration
W.H. Bell - Layout / Graphics / Administration
Matthew Judge - Website / Administration / Tester
Chris 'tzj' Stagg - Administration / Tester / Graphics
Tim 'meltwater' Cox - Administration
Lix - Administration
Ian McAlpine - Layout / Tester / Graphics
Isa McKenty - Layout / Graphics / Tester
Sam Marshall - Page Design / Layout
Bryan Butler - Page Design & Theme / Graphics / Tester
Aaron Shaw - Layout / Graphics / Tester / Proof Reading
Nick Wreck - Layout / Graphics / Tester
Colin Deady - Layout / Graphics
Mark Tranter - Layout / Graphics
Courtney Blush - Layout / Tester / Proof Reading
Steve Drew - Layout
Adrian Harper - Tester / Proof Reading
Paul Carpenter - Tester
Phil Tesseyman - Tester
Shelton Caruthers - Tester / Proof Reading
Mark Robson - Proof Reading

Contents

RPi HEAT

Control and Monitor Your Home Heating System with a Raspberry Pi and a Smartphone

Nikolaos Tsipas

Guest Writer

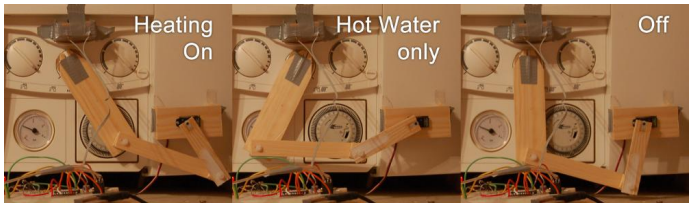
DIFFICULTY : ADVANCED

Introduction

In this article, we explore the possibilities of using a Raspberry Pi for the remote control and monitoring of a home heating system using: servo control, analog to digital conversion, C programming and web development.

Servo Control

We decided to use a servo device to control the on/off switch of the boiler. We could have used relay switches for the same purpose, but tried to stick to the principle of making no modifications to the boiler (as it's owned by the landlord :D). The following image shows how the servo is mounted and the various states of control.

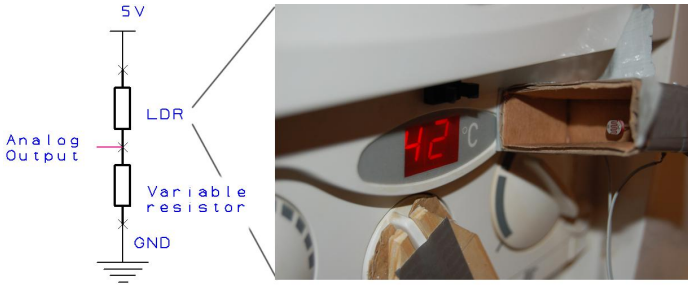


Our mini servo has reduced power requirements, so we could power it directly from the 5V pin of the Raspberry Pi. To create the needed pulse for servo control, we utilized wiringPi library's softPWM implementation. Small glitches exist, but the accuracy is sufficient for this kind of control. The code below uses GPIO pin 0 as a PWM signal output to control the servo. We assume that wiringPi library is already installed. The "control" variable accepts integer values in the range of 180 to 194. The input to that variable will control the three positions of the switch. This will eventually be controlled from a web based user interface.

```
pinMode(0, OUTPUT);
digitalWrite(0, LOW);
softPwmCreate(0,0,200);
softPwmWrite(0,control);
```

Sensors and ADC

Our servo device can now control the state of the boiler from our Raspberry Pi. However, when we remotely control the system, we cannot be sure that the servo worked as it should and that the boiler is in the state we want. For this reason, we need a way to get feedback from the Raspberry Pi regarding the boiler's state. The most straight-forward way to achieve this is to check the status of the LED indicator on the boiler. When the LED is on, the boiler is turned on, and the LED is off, when the boiler is off. A simple LDR (Light Dependent Resistor) light sensor in a dark box, mounted on the boiler's two 7 segment LED displays should do the job. A LDR sensor is a variable resistor whose resistance depends on the amount of light arriving on it.



We also use two LM35 temperature sensors with analogue output; one for air temperature, mounted on the breadboard and one for water temperature, mounted on a radiator.

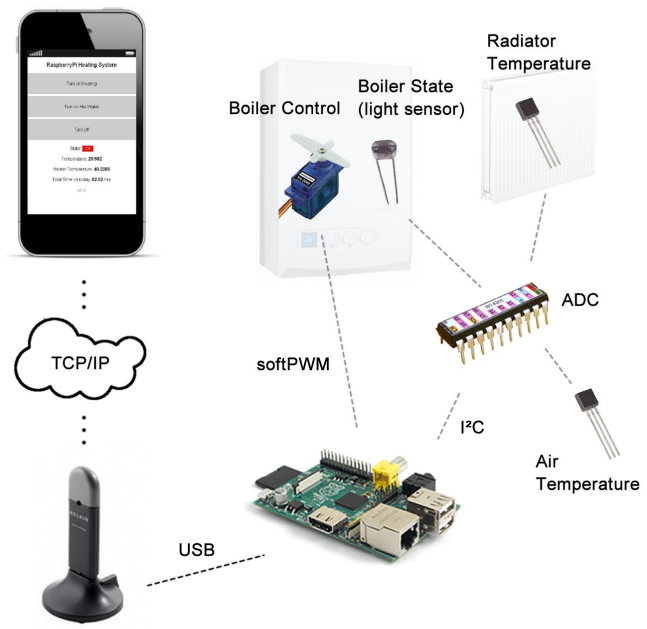
You'll need an analogue-to-digital converter (ADC) when using analogue sensors. We chose a BV4205, a 10 channel I²C enabled ic. The analogue output from the three sensors gets converted to digital and is fed to the Raspberry Pi through the I²C bus. However, due to known issues with the I²C bus on the Raspberry Pi, a "patch" to slow down the clock of the I²C bus had to be applied before using the BV4205. Alternatively, a microcontroller (e.g. atmega) could be used for the purpose of an ADC.

Below, we use the most basic parts of the code needed for communications over the I²C bus to read the analogue input of a sensor connected to channel 7. Similar code can be used for interaction with any I²C compatible device.

```
//Initialize I2C bus
int fd;
char *fileName = "/dev/i2c-1";
//Note: When using a Rev 1 Pi, use
"/dev/i2c-0"
int  address = 0x31;
unsigned char buf[10];
fd = open(fileName, O_RDWR);
ioctl(fd, I2C_SLAVE, address);
//Select ADC channel 7
buf[0] = 1;
buf[1] = 7;
write(fd, buf, 2);
//Start conversion
buf[0] = 2;
write(fd, buf, 1);
//Fetch result of conversion
buf[0] = 4;
write(fd, buf, 1);
read(fd, buf, 2);
unsigned short packedword;
packedword = (buf[0] <<8) | buf[1];
```

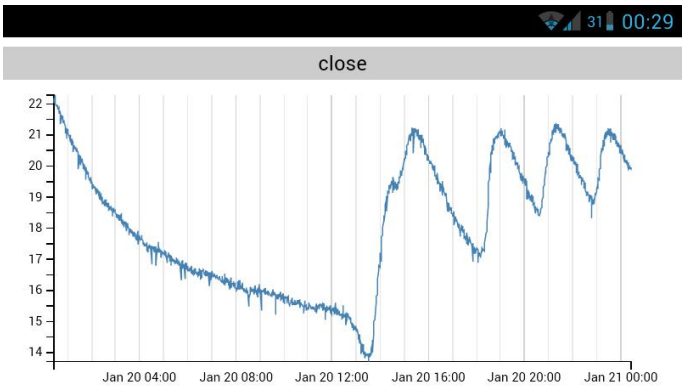
User Interface

The following schematic shows the setup and how the devices connect and interact. A simple web interface provides easy access from web enabled devices. The web interface is mainly



based on jQuery, PHP and MySQL.

Using PHP's exec command, we get access to our two main executables that control the servo's position and read sensor output. Ajax has been used extensively in order to achieve real-time monitoring of the boiler's state and sensor readings. Furthermore, we calculate how long the system was on each day (in minutes) by analysing the light sensor's output. The user is also presented with an svg (d3.js library) graph of the air temperature for the last 24 hours with time resolution of 1 minute.



By exploiting data from the sensors, we could further extend the system and make it more intelligent by fully automating on and off events based on sensor output and specific usage profiles. For more information (code, schematics, pictures) about the project visit: http://projectedneuralactivity.blogspot.co.uk/

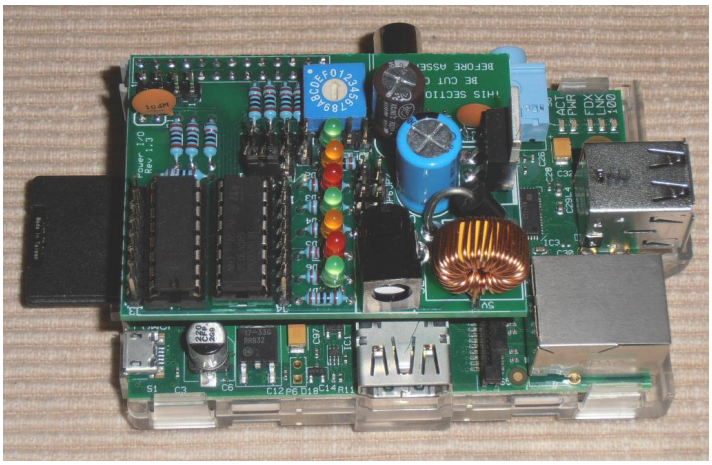
POWER AND I/O EXPANSION BOARD

Lloyd Seaton

Guest Writer

This is a constructional project for the hobbyist who is confident with a soldering iron, likes to have options and is prepared to purchase their own components.

DIFFICULTY : ADVANCED



Functionality

In spite of its small size, Power I/O endows the Raspberry Pi with substantial GPIO interfacing capabilities and flexibile power supply arrangements:

- 7 digital outputs with high current drivers
- 7 indicator LEDs associated with the above
- 5 digital inputs with 16V tolerance
- UART and I2C bus provisions
- Jumper options for flexibility
- Hexadecimal rotary switch
- Switching mode power supply (7 ~ 20VDC)

Form Factor

As can be seen above, Power I/O sits directly above the Raspberry Pi and integrates neatly, so that the two units together have a minimal footprint and a tidy appearance.

Provision is made for the printed circuit board (PCB) to have a cut-out to clear the Raspberry Pi's video connector if the constructor chooses to use a low profile receptacle (socket) to connect with the Raspberry Pi. This unit has instead been constructed using the preferred 3M high profile socket so that no cut-out was required and there is plenty of clearance between the two printed circuit assemblies (PCAs) to eliminate any risk of an electrical short or restriction of air circulation around the Broadcom SoC or ethernet chip.

"Include, omit or vary components to match your individual preferences."

PCB Manufacturing

The first thing you must decide is how to procure your printed circuit boards (PCBs) and there are several ways, depending on your affiliations and the interests of you and your affiliates. This project's PCB is represented in 4 different cocktail files from which you can choose.

The file, 2xPiPower+IO.pcb, simply has twin copies of this project so that, if you use this file to order your PCBs through the ExpressPCB.com MiniBoard Pro service, each of the 3

manufactured PCBs will yield 2 PCBs for this project, to be separated by hacksaw. Therefore, the yield from an order will be 6 PCBs for this project.

The file, MegaPowerIO.pcb, has 1 copy of this project's PCB together with 1 copy of a PCB for the MegaPower project. Therefore, a MiniBoard Pro order through ExpressPCB.com will yield 3 PCBs for this project and 3 PCBs for the MegaPower project, after separation.

The file, PowerQuartet.pcb, has 1 copy each of PCBs for: this project, Battery Load Manager 85, Battery Load Manager+ and Pi Bridge ICSP Interconnect. Therefore, a MiniBoard Pro order through ExpressPCB.com will yield 3 PCBs for each of these projects, after separation.

The file, TinyPowerIO.pcb, has 1 copy of this project's PCB and a copy of a PCB for the Tiny I/O project. Therefore, a MiniBoard Pro order through ExpressPCB.com will yield 3 PCBs for this project and 3 PCBs for the Tiny I/O project, after separation.

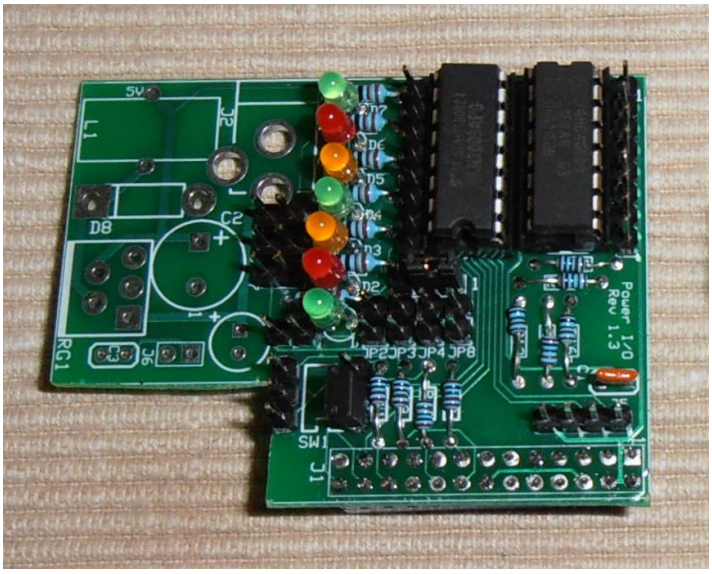
All of the projects in the above cocktail files were outlined in the Issue 10 article, "Try a Cocktail of Projects". Extensive documentation, photos and test programs for all of these projects are available on the Internet via the information blog at **http://picocktails.blogspot.com**.

Options

An important advantage of constructing your own expansion board is that you can optionally include, omit or vary components to match your individual preferences.

"This project calls for reasonable proficiency at soldering"

The unit pictured to the right has been deliberately assembled without the switching power supply, but, if circumstances change, the power supply components can be added later. This unit has a cut-out section at the lower left, necessitated by the use of a low-profile socket instead of the preferred high-profile 3M socket.



A hexadecimally encoded switch (SW1) can be included if the intended application appreciates its functional or educational value but it has been omitted from the unit pictured above for budgetary or functional reasons, replaced by a pair of 3-pin headers that can be jumpered to match user requirements.

Construction

This project calls for reasonable proficiency at soldering as the component density is relatively high, but an elaborate tool kit is not required. Your greatest construction assets will be planning, care, and patience.

At **http://picocktails.blogspot.com**, there is a dedicated page (Issue 11) with links to all of the resources related to this project. There is a design description document with schematic diagram, pin-out tables, suggested component suppliers, construction and operating hints. There are also links to the cocktail files, photo albums, and a Python test program. The blog page also provides a mechanism for publication of any other information that may be required for the project's on-going support.

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

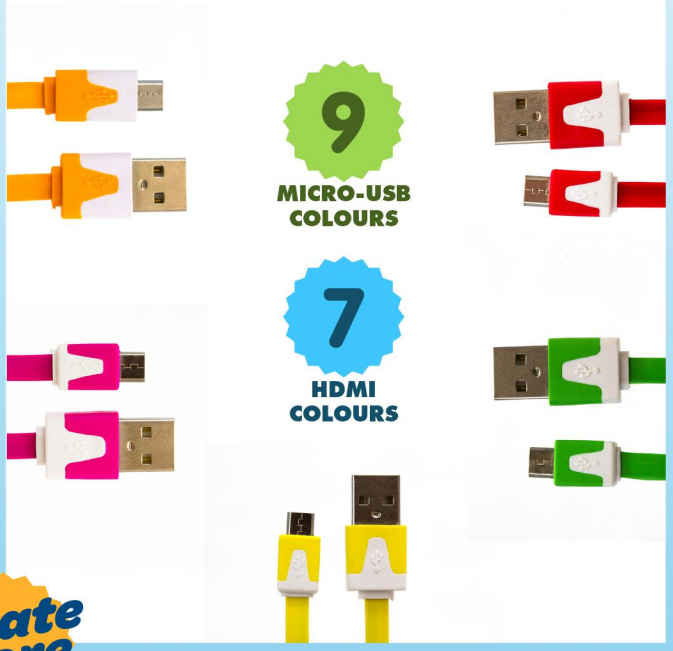
Yes, we do bulk discounts and POs and we have a lovely tame STEM Ambassador whose help is free.
We also don't mind if you point out our grammatical errors :-)

STEM AMBASSADORS
ILLUMINATING FUTURES

Noodles

HDMI 大粉 USB

http://shop.pimoroni.com



Pirate Store from PIMORONI

PI BOW

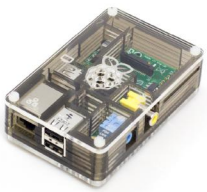
The neat little layer case for your Raspberry Pi®

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

Crystal

Toxic

Ninja



Adafruit

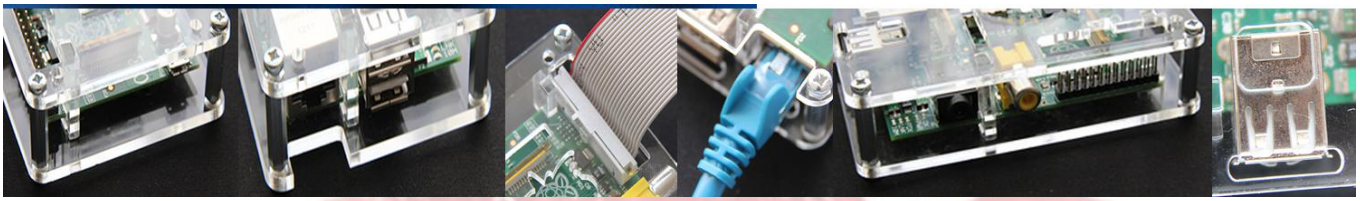
Rainbow

PIMORONI

Available From:
http://pibow.com/

adafruit · maplin · CPC · MODMYPI · MCM

APRIL 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 plus a PCSL Raspberry Pi case!
The second and third prize winners will receive a PCSL Raspberry Pi case.

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

Closing date is 20th April 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

March's Winners!

The winner of the new 512MB Raspberry Pi Model B is **Jonathan Watters (Enfield, UK)**. The 2nd and 3rd prize winners of the PCSL LCD VESA mount case are **Dennis Gray (Logan, Utah, USA)** and **Richard Stevenson (Portsmouth, UK).**

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

Turn your Raspberry Pi into a Wireless Pi-Point

Guy Eastwood

Guest Writer

DIFFICULTY : ADVANCED

How to convert your Raspberry Pi into a Wireless Access Point using a simple WiFi USB dongle.

You will need

- A Raspberry Pi model B (of course!)
- An SD Card for your Pi – I use 4GB for this but whatever you have should be good
- A USB WiFi Dongle – I used a ZyXEL Communications Corp. ZyAIR G-202 802.11bg

Setting up your SD Card

First install Raspbian from the Raspberry Pi site at the URL below. At the time of writing the current version is 'Wheezy':

```
http://www.raspberrypi.org/downloads
```

Install the image to your SD card as explained here:

```
http://elinux.org/RPi_Easy_SD_Card_Setup
```

Logging into your Pi

Log in to your Pi – I setup mine via SSH but no reason why you can't do it via a keyboard and screen if you have it connected that way. The default login is username 'pi' with password 'raspberry'. You'll be wanting to change these later for security. If not logging in via SSH make sure you have a network connection with internet access.

If you're using SSH then you'll need to locate your Pi's IP address on your LAN using (on Linux systems, at least):

```
$ sudo nmap -sP 192.168.0.0/24
```

Which will list all IP's on your network (if your network is 192.168.1.x then change the network range in the command to 192.168.1.0/24 and so on). You'll get a list along the lines of...

```
Nmap scan report for UNKNOWN
(192.168.0.54)
Host is up (0.65s latency).
MAC Address: 00:24:2C:12:62:55 (Hon Hai
Precision Ind. Co.)
Nmap scan report for UNKNOWN
(192.168.0.55)
Host is up (0.23s latency).
MAC Address: B8:27:EB:9E:CA:4D
(Raspberry Pi Foundation)
Nmap scan report for UNKNOWN
(192.168.0.57)
Host is up (0.0036s latency).
MAC Address: 00:A0:DE:86:D3:6B (Yamaha)
```

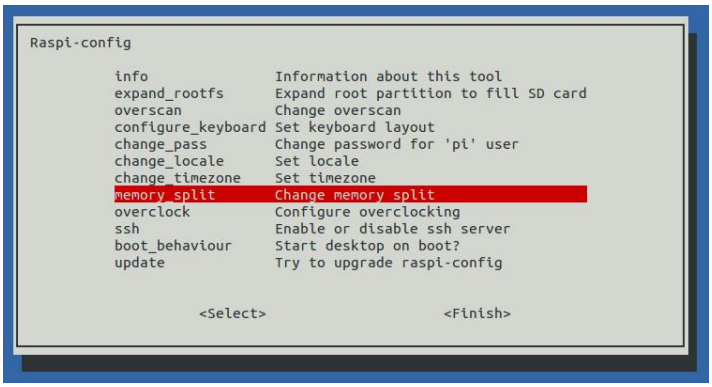
In this case the Pi is at 192.168.0.55 so I'd login with ssh pi@192.168.0.55.

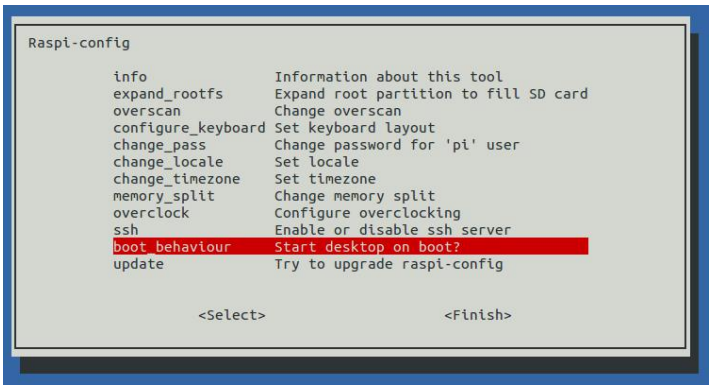
Configuring your Pi

Firstly run:

```
$ sudo raspi-config
```

if you haven't yet, to setup your Pi (you'll likely want to at least change your memory split to 16 and turn off 'Start desktop on boot?' which you'll not be needing) then reboot and set a root password using `sudo passwd` and enter a decent (i.e. not 'password'!) password for the root user. From this point on I'll assume you're logged in as root. I'm also assuming you know how to use vi or vim console text editors (or maybe an alternative like nano).





Installing the necessary software packages

I prefer using aptitude for package installs and will be using it throughout this article – if you're happier using apt-get then you'll be wanting to alter some of the installation commands to suit. Install aptitude with:

```
$ apt-get install aptitude
```

Bring your Pi up to date using:

```
$ aptitude update; aptitude safe-upgrade
```

This may take a little while and also depend on your internet connection speed.

You'll need to install these packages:

```
$ aptitude install rfkill zd1211-
firmware hostapd hostap-utils iw dnsmasq
```

These are:
rfkill -- Wireless utility
zd1211-firmware -- Software for dealing with **zd1211** based Wireless hardware
hostapd -- The hostap wireless access point daemon
hostap-utils – Tools that go with hostap
iw – Wireless config utility
dnsmasq – a DHCP and DNS utility
You may also want to add 'vim' to that list which is a nicer console editor than the default 'vi'.

The zd1211-firmware package is hardware-specific but is a very common implementation for wifi dongles. Your dongle will also need to support 'AP' mode which you can verify by typing:

```
$ iw list
```

and under the 'Supported interface modes' you will hopefully see 'AP' listed along with others like 'managed' and 'monitor'. If not you're out of luck with your particular dongle, so if you have another Linux machine handy you may want to plug the dongle into that and check it with ‘iw list’ before you start!

Configuring the wireless interface

Your wireless interface must be set statically for hostap, edit your /etc/network/interfaces file to look like this:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
iface wlan0 inet static
address 192.168.1.1
netmask 255.255.255.0
```

Then restart your wireless interface using:

```
$ ifdown wlan0; ifup wlan0
```

which should hopefully go smoothly without errors. The address of 192.168.1.1 should not be the same as the network connected to eth0, my main LAN is 192.168.0.0/24 and the wireless interface should be a different network. This adds a simple layer of security in case you're configuring an open access point as I was and allows you to firewall one network from the other far more easily.

Configuring the Access Point

Now to configure hostap. Edit /etc/hostapd/hostap.conf (it may not already exist but this will create it, anyway) to look like this:

```
interface=wlan0
driver=nl80211
ssid=test
channel=1
```

The settings are pretty obvious, 'driver' being the only exception, just leave it as it is but change the other values as you see fit, though these should do for an initial test. One thing to bear in mind is that hostap seems to be very literal about reading its configuration file so make sure you have no trailing spaces on the end of any lines! Restart the hostapd service with:

```
$ service hostapd restart
```

All being well you should now see 'test' as a wireless network, though a little more work needs to be done to connect to it yet.

Configuring the DHCP server

The final step is to configure dnsmasq so you can obtain an IP address from your new Pi-Point. Edit your /etc/dnsmasq.conf file to look like this:

```
# Never forward plain names (without a
#dot or domain part)
domain-needed

# Only listen for DHCP on wlan0
interface=wlan0

# create a domain if you want, comment
#it out otherwise
#domain=Pi-Point.co.uk

# Create a dhcp range on your /24 wlan0
#network with 12 hour lease time
dhcp-range=
192.168.1.5,192.168.1.254,255.255.255.0,
12h

# Send an empty WPAD option. This may be
#REQUIRED to get windows 7 to behave.
#dhcp-option=252,"\n"
```

Remember to change the dhcp-range option for the network IP range you're using. If you're having problems with Windows 7 machines then try uncommenting the last line. Restart dnsmasq by typing:

```
$ service dnsmasq restart
```

for the configuration to take effect. Now you should be able to connect to your new Pi-Point and get a proper IP address from it.

Now your Pi-Point is up and running there's a few things to take care of. You'll probably have noticed that you can't load Google or anything else. That's because although you can connect to the Pi-Point and get an IP address from it, it's not yet routing your requests across your network and out onto the internet. We'll address this next.

Configure the network routing

The first thing you need to do is to allow forwarding in the Pi-Point, this is achieved by issuing the command:

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

to turn forwarding on (N.B. you can also permanently set the ip_forward value as a one-off in /etc/sysctl.conf by uncommenting the appropriate line). Now your Pi-Point is forwarding

packets you need it to provide NAT between your wifi network and your main network which is attached to the internet:

```
$ iptables -t nat -A POSTROUTING -j
MASQUERADE
```

You should now have a working Pi-Point which will allow users to connect wirelessly and use the internet via your internet connection!

Making sure it reboots as a working Access Point

To ensure your Pi-Point works from a reboot – remember you've done all this inside your current login session with console commands - you'll need to create a run file to turn on forwarding, NAT and run hostap at boot time. Create a file /etc/init.d/pipoint with these contents, keep all the commented lines, too, as these are used by the startup processes:

```
#!/bin/sh
# Configure Wifi Access Point.
#
### BEGIN INIT INFO
# Provides:          WifiAP
# Required-Start:    $remote_fs $syslog
$time
# Required-Stop:     $remote_fs $syslog
$time
# Should-Start:      $network $named
slapd autofs ypbind nscd nslcd
# Should-Stop:       $network $named
slapd autofs ypbind nscd nslcd
# Default-Start:     2
# Default-Stop:
# Short-Description: Wifi Access Point
configuration

# Description:       Sets forwarding,
starts hostap, enables NAT in iptables
### END INIT INFO

# turn on forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# enable NAT
iptables -t nat -A POSTROUTING -j
MASQUERADE

# start the access point
hostapd -B /etc/hostapd/hostapd.conf
```

Next make the script executable with:

```
$ chmod +x /etc/init.d/pipoint
```

Next add the script to the startup sequence using:

```
$ update-rc.d pipoint start 99 2
```

This should ensure your Pi-Point will reboot as a functioning wifi access point.

Where next?

You could try installing SQUID proxy to reduce bandwidth from wifi users across your network or even use a Pi-Point in place of buying a wifi router to create a private gaming network unconnected to your main LAN which game players (e.g. Minecraft) could connect to as an isolated wireless network.

About the Author

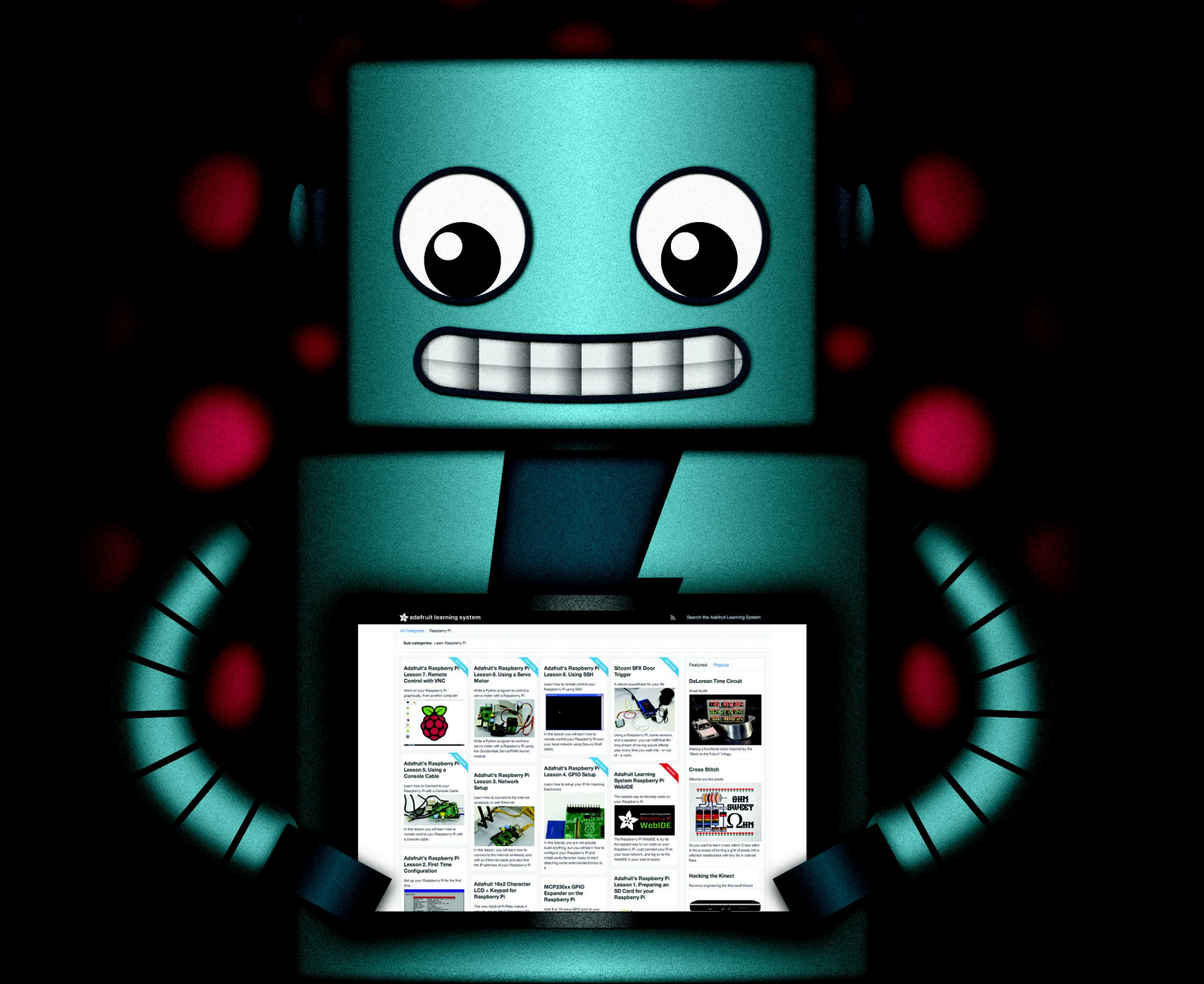
Guy Eastwood is a web development professional (most of the time) these days and Open Source advocate, having cut his teeth on pretty much everything from ZX81 BASIC to Sharc Assembler DSP audio plugins for sound cards. He fills spare time mucking about with Android stuff, Raspberry Pi configurations, Playstation3 and doing as he's told (promptly) by his culinary genius wife.

For further details visit http://www.pi-point.co.uk

adafruit learning system

Raspberry Pi WebIDE

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



adafruit

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

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

Yorkshire RISC OS Show

When: Saturday 20th April 2013 @ 10:30am
Where: The Cedar Court Hotel, Wakefield, West Yorkshire, WF4 3QZ

This show will run from 10:30am until 4:30pm. Further information is available at http://www.wakefieldshow.org.uk/index.php

Bermuda Raspberry Jam

When: First Tuesday of each month @ 6:00pm
Where: Admiralty House, Pembroke, Bermuda

The meeting will run from 6:00pm until 8:00pm. Further information is available at http://bermudaraspberryusergroup.teamsnap.com or email rpi@sainib.com

Geneva Raspberry Pi at Fêtons LINUX

When: Saturday 27th April 2013 @ 1:00pm
Where: HEPIA, rue de la Prairie 4, Genève, Suisse

The meeting will run from 1:00pm till 7pm.
Further information is available at http://www.fetons-linux.ch/

Bristol Raspberry Pi @Bristol Boot Camp

When: Saturday 20th April 2013 @ 10:30am
Where: @Bristol, Anchor Road, Bristol, BS1 5DB

The event will run from 10:30am till 4:30pm. Pre-registration required.
Further information is available at https://bcsbristolbootcamp.eventbrite.com/

Minecraft: Pi Edition

DIFFICULTY : INTERMEDIATE

Martin O'Hanlon

Guest Writer

The Game

Minecraft is a game which has achieved monumental success - almost 30 million copies, across all its versions, have been sold; not bad for a game which doesn't really have a point! It is classified as an Indie Sandbox game, but if it does have a point, it is to make stuff (and people have really made stuff), from fully functioning computers to scale models of the starship Enterprise.

Now Minecraft has come to the Pi and the two best things about Minecraft: Pi Edition are it is free and comes with an API - two things you do not get with any other version of Minecraft (certainly not yet anyway).

Installing

If you haven't installed Minecraft already head over to pi.minecraft.net to download and view the latest installation instructions At the time of writing this was the current method:

1) Download & Extract

```
  cd ~
  wget
https://s3.amazonaws.com/assets.minecraft.net/p
i/minecraft-pi-0.1.1.tar.gz
  tar -zxvf minecraft-pi-0.1.1.tar.gz
```

2) Run

```
  cd mcpi
  ./minecraft-pi
```

Playing

The Pi edition of Minecraft is a cut down version of the pocket edition and currently only offers one playing mode, Classic, which is all about exploring and building. Click 'create world' and start exploring and building.

You control your player with the mouse and keyboard:

Moving - the mouse changes where you look, while the arrow keys move you forward, backward, left and right.

Up and down - pressing space will make you jump while you are walking. Double tapping will start you flying. While you are flying, space will make you move up and shift will make you go down.

Inventory - you can cycle through your inventory using the mouse scroll wheel (or keys 1-8) and change the blocks in your immediate inventory (the strip along the bottom) by pressing E which brings up all the blocks you can use.

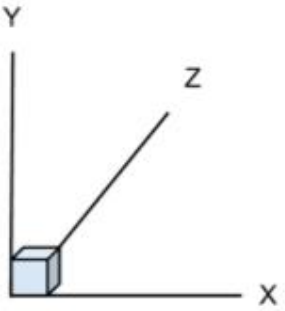
Destroy blocks - press (hold) the left mouse button.

Place blocks - press the right mouse button.

Back to Menu - press escape.

The API

The API allows you to write programs which control, alter and interact with the Minecraft world, unlocking a whole load of Minecraft hacking. How about creating massive structures at the click of a button, a bridge which automatically appears under your feet allowing you to walk across massive chasms, a game of minesweeper or a huge clock?



Minecraft is a world of cubes or blocks, all with a relative size of 1m x 1m x 1m. Every block has a position in the world of x,y,z. x and z are the horizontal positions and y is the vertical.

The API works by changing the 'server', which runs underneath the game, allowing you to interact with these blocks and the player, such as:

Get the player's position.

Change (or set) the player's position.

Get the type of **block**.

Change a **block**.

Change the **camera angle**.

Post messages to the player.

Libraries

You can interact with the server directly by sending a message to it, but the nice people at Mojang also provide libraries for Python and Java which simplify and standardise using the API.
The libraries are installed along with the game in the ~/mcpi/api/java and ~/mcpi/api/python directories.
The following example is written in Python and uses Mojang's Python API library. The first task is to create a directory for your API program and copy the Python library to it.

API Example

1) Create Directory

```
mkdir ~/minecraft-magpi
```

2) Copy Python Library
We will be using the python api library supplied with the game.

```
cp -r ~/mcpi/api/python/mcpi ~/minecraft-magpi/minecraft
```

3) Create program
Open Idle (or your favourite editor) and create a program file called `minecraft-magpi.py` in the `~/minecraft-magpi` directory.
We are going to need 3 modules: the minecraft.py and block.py modules from the minecraft directory containing the library we just copied and the standard time module so we can introduce delays into our program.

```
import minecraft.minecraft as minecraft
import minecraft.block as block
import time
```

Next, we need to use the Minecraft class in the Python library to create a connection to the game's server. We will use this object to interact with the game and provide access to all the functions. When your program runs the following statement Minecraft will have to be running and you will need to be in a game, otherwise you will get errors.

```
mc = minecraft.Minecraft.create()
```

Using our minecraft object, mc, we can then interact with the game and send the player a message. We will also put a delay in using the `time.sleep()` function otherwise the whole program will run too fast for us to see what is going on.

```
mc.postToChat("Hello Minecraft World")
time.sleep(5)
```

Using the program built so far, you can test to make sure everything is working. Load up Minecraft and create a (or enter an existing) world. Press ALT+TAB to switch to your editor. If you're using Idle select 'Run Module' from the Run menu. If all has been setup correctly you will see the "Hello Minecraft World" message in the game.

Interacting with the player is done through the `player` class of the `mc` object allowing us to find and change the position of the player. The next block of code finds the player's position using the `getPos()` command, which returns an object of x,y,z coordinates. The `setPos()` command is then used to move the player 50 blocks up, by adding 50 to the player's y coordinate. We then add a delay, so there is enough time for your player to fall down to the ground!

```
  playerPos = mc.player.getPos()
  mc.player.setPos(playerPos.x, playerPos.y +
50, playerPos.z)
  mc.postToChat("Don't look down")
  time.sleep(5)
```

You can use the position of the player as a starting point for interacting with blocks. This way you can find out what block the player is standing on or place blocks around the player. There is however a challenge as the x,y,z coordinates returned by the `getPos()` function are decimals (aka floats) as the player can be in the middle of a block. To interact with blocks we need to use whole numbers (aka integers), so we need to use the function `getTilePos()`, which returns the block (or tile) he's standing on.

The code below gets the player's tile position. It then calls the Minecraft API's `getBlock()` function to find out the type of block the player is standing on (by subtracting 1 from the y coordinate) before using `setBlock()` to create blocks of the same type the player is standing on around him. So if your player is standing on DIRT, he will end up with DIRT surrounding him, however if he is standing on STONE, STONE will appear.

```
  playerTilePos = mc.player.getTilePos()
  blockBelowPlayerType =
mc.getBlock(playerTilePos.x, playerTilePos.y -
1, playerTilePos.z)
  mc.setBlock(playerTilePos.x + 1,
playerTilePos.y + 1, playerTilePos.z,
blockBelowPlayerType)
  mc.setBlock(playerTilePos.x, playerTilePos.y
+ 1, playerTilePos.z + 1, blockBelowPlayerType)
  mc.setBlock(playerTilePos.x - 1,
playerTilePos.y + 1, playerTilePos.z,
blockBelowPlayerType)
  mc.setBlock(playerTilePos.x, playerTilePos.y
+ 1, playerTilePos.z - 1, blockBelowPlayerType)
  mc.postToChat("Trapped you")
  time.sleep(5)
```

We have now trapped our player within 4 blocks (providing he doesn't break out!), in order to set him free we need to remove a block. Removing blocks is done using `setBlock()`, but rather than making the block solid like WOOD or STONE we set it to AIR.

```
  mc.setBlock(playerTilePos.x + 1,
playerTilePos.y + 1, playerTilePos.z,
block.AIR)
  mc.postToChat("Be free")
  time.sleep(5)
```

A full list of all the available blocks can be found in either the Minecraft API specification or in the block.py module in the Python API library `~/mcpi/api/python/mcpi/block.py`.

The API also allows you to set many blocks at a time, allowing you to create cuboids very quickly using the `setBlocks()` command. It works by specifying 2 sets of x,y,z coordinates between which it then fills the gap with a certain block you pass as the final parameter. The code below will create a diamond floor underneath our player 50 blocks (across) x 1 block (up) x 50 blocks (along), with our player in the middle (i.e. 25 behind and to the left, 25 in front and to the right).

```
  mc.setBlocks(playerTilePos.x - 25,
playerTilePos.y - 1, playerTilePos.z - 25,
playerTilePos.x + 25, playerTilePos.y -1,
playerTilePos.z + 25, block.DIAMOND_BLOCK)
  mc.postToChat("Now thats a big diamond
floor!")
```

To recap on the functions covered in this article:

postToChat(message) - communicate with the player(s) in the game.

getBlock(x, y, z) - get a block type for a specific position.

setBlock(x, y, z, blockType, blockData) - set (change) a block to a specific blockType.

setBlocks(x1, y1, z1, x2, y2, z2, blockType, blockData) - set lots of blocks all at the same time by providing 2 sets of co-ordinates (x, y, z) and fill the gap between with a blockType.

player.getPos() - get the precise position of a player.

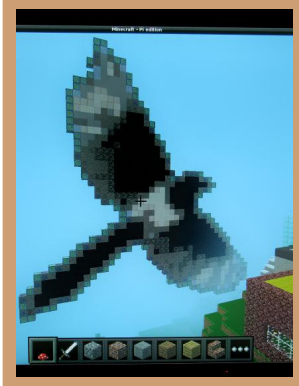
player.setPos(x, y, z) - set (change) the players position.

player.getTilePos() - get the position of the block where the player current.

There are few other functions available in the API which should be explored. Using only the small number discussed in this article it is possible with some imagination and a small amount of programming knowledge to create some quite fantastic constructions, tools and utilities.

I love Minecraft. It is a tremendously creative game and with the Pi edition's API it opens up new level of creativity and will hopefully encourage more people to try their hand at programming.

Creating the cover image
from blocks to virtuality



Ian McAlpine created the impressive Minecraft magpie on this month's cover. The MagPi asked Ian how he achieved this.

With the Minecraft API installed (see main article text) I decided I wanted to create a large copy of the magazine's logo that would be easily recognisable in the game world, using Python. This was my very first Python program. Coding the magpie rather than building manually gave me the freedom to place the magpie anywhere in the Minecraft world. The whole thing was very experimental, balancing effort in terms of pixelation while still retaining an obviously recognisable image.

Having identified that I was limited to 8 shades of black/grey/white and 4 shades of blue in terms of Minecraft blocks, that partially determined the pixelation. Using Gimp on my Raspberry Pi, I pixelated The MagPi magpie image to 43x44 blocks. This seemed to be the right balance... but still meant manually mapping 1892 squares to Minecraft blocks! It could be made smaller but the pixelation would be more coarse and the magpie less recognisable.

In Python, the image is represented by a list of lists (ie a 2-dimensional array). Each cell in the array was given a character; 1-8 for black, white and shades of grey plus A-D for the shades of blue. 0 represented the "Air" block type. Five other variables were used; 3 for the x, y, z coordinates, one for the angle in the vertical plane (0, 90, 180 or 270 degrees) and a variable to indicate if the magpie should be plotted or deleted. A nested FOR-loop iterates through the 2-D array and plots the blocks.

Because some block types cannot be placed in free space, it is important that you build from the bottom up. When deleting (ie replacing all blocks with block type "Air") you need to delete from the top down. If you delete from the bottom up, some bricks will fall before they can be deleted thus leaving a pile of rubble on the ground!

You should take a backup copy of your world before creating large structures like this as you will likely use the wrong coordinates. My 10 year old daughter Emily was very unhappy when I obliterated her village! Take a copy of the hidden .minecraft directory in your home directory (use ls -A to see all hidden files in a folder) plus also the games directory in the mcpi directory.

Printing with CUPS

Stewart Watkiss

Guest Writer

1 - Configuring CUPS

DIFFICULTY : BEGINNER

This is a two part tutorial on configuring printing on the Raspberry Pi. In this article I am going to explain some of the essentials of setting up printing and adding your printer in Raspbian. This could be useful for printing from your Raspberry Pi or to turn your USB printer into a network enabled printer. Next month we will look at how we can print from the command-line and from within your own Python programs.

The standard method of printing on Linux is with the Common Unix Printing System™ better known as CUPS. Developed by Apple Inc., CUPS is open source software using standard based printing protocols.

The job of converting an application print request to something that your printer can understand is a memory hungry task. I therefore recommend using a 512MB model B for printing directly from the Pi. Even with a 512MB Raspberry Pi you can still run into memory issues. If you have problems with printing then it is worth trying with as few applications running as possible, before looking at other causes.

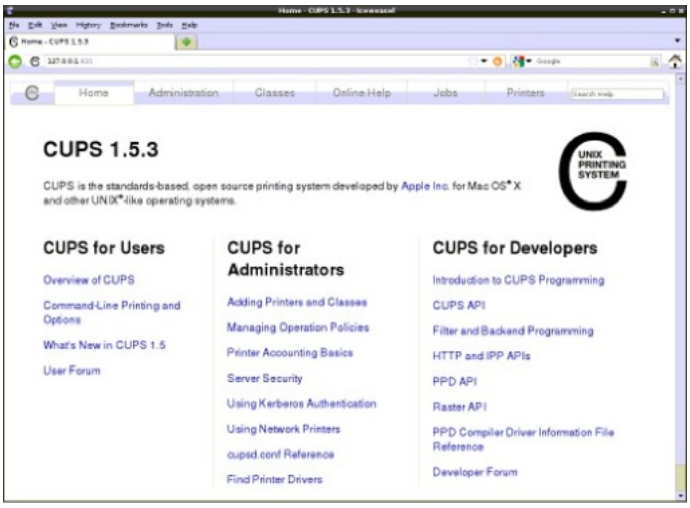
First we need to install CUPS from the software repositories:

```
sudo apt-get install cups
```

CUPS uses the lpadmin group to determine who is authorised to administer the printers. To add a printer you will need to add your user to the lpadmin group:

```
sudo usermod -a -G lpadmin pi
```

We are now going to use the CUPS web based interface which is provided by default. All the functionality is built into CUPS, so we do not need to worry about enabling a web server. Just point your browser at http://127.0.0.1:631 and use your normal username and password when prompted during configuration or when administering the printers. Choose the Administration tab and then Add Printer.



CUPS will search for printers that are physically connected to the Raspberry Pi or that it can see on the local network. In my case it found my network printer "Epson Stylus Photo PX720WD". You will also see some options for HP printers as these are often shown even if there are no HP printers connected.

After selecting the appropriate printer you can customise the name and location of the printers. These can normally be left at their defaults, although you could change the name to one that is easier to remember. If you are likely to be printing from the command line then it can also be useful to make the printer name shorter and avoid any spaces.

An option you may like to enable is “Share this printer” which can be used if you would like to use the Raspberry Pi as a print server for other computers. This will allow you to print to that printer from other computers on the same local network.

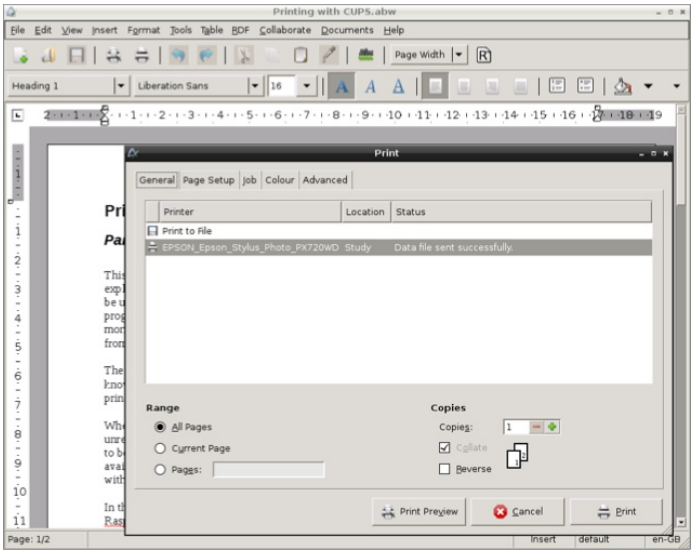
The next page will show available printer drivers and attempt to find the closest match. If you have a common printer then there is a good chance of finding an exact match, but if not then it may be possible to select a different printer with similar functionality. Some printer manufacturers do provide Linux drivers that could be installed instead, although it depends upon the manufacturer. If CUPS is not able to find a suitable match you should try the website of your printer manufacturer to see if they have any Linux drivers and follow their instructions.

After selecting an appropriate printer you can choose options for paper size and then the printer drivers will be installed. You can now view the printer through the Printers tab, where you can change any parameters and print a test page. I also suggest selecting the printer as the Server default printer through that page, as that will make it easier to print using the command line tools.

The printer should now be available in any printer enabled applications (eg. office tools / web browsers) and printing should be similar to that implemented on other operating systems. If you need any help then online help is included within the CUPS web page :
http://127.0.0.1:631/help/

On my Raspberry Pi I have installed AbiWord which is a lightweight word processor that runs well on the Raspberry Pi. You can see the standard print dialog in the screenshot. You can install AbiWord with:

```
sudo apt-get install abiword
```



Next month we will look at printing from the command line and how to print from your own Python applications.

Quick2Wire ▸ Win three full sets of Quick2Wire kits

Quick2Wire Limited are giving away three full sets of Quick2Wire kits (Interface kit, I2C Port Expander kit, I2C Analogue kit), to be awarded on the basis of projects that you submit.

Competition Rules

Your project must run on the Raspberry Pi, and must be an original work by you. You must agree to release the design of your hardware, and your associated software as Open Source. Hardware must be licensed under the **CC 3.0** share-alike license; your software and any libraries that you use must be licensed under the **MIT** license or **LGPL.**

The project does not have to be complete, but a significant part of it must be demonstrably working.
The winning projects will be chosen by a panel of judges (to be announced) on the basis of

- Educational value
- Utility
- Time and money required (less is better :) and
- Technical merit.

We'll also take your age into account if you are under 18.

The decision of the judges will be final.

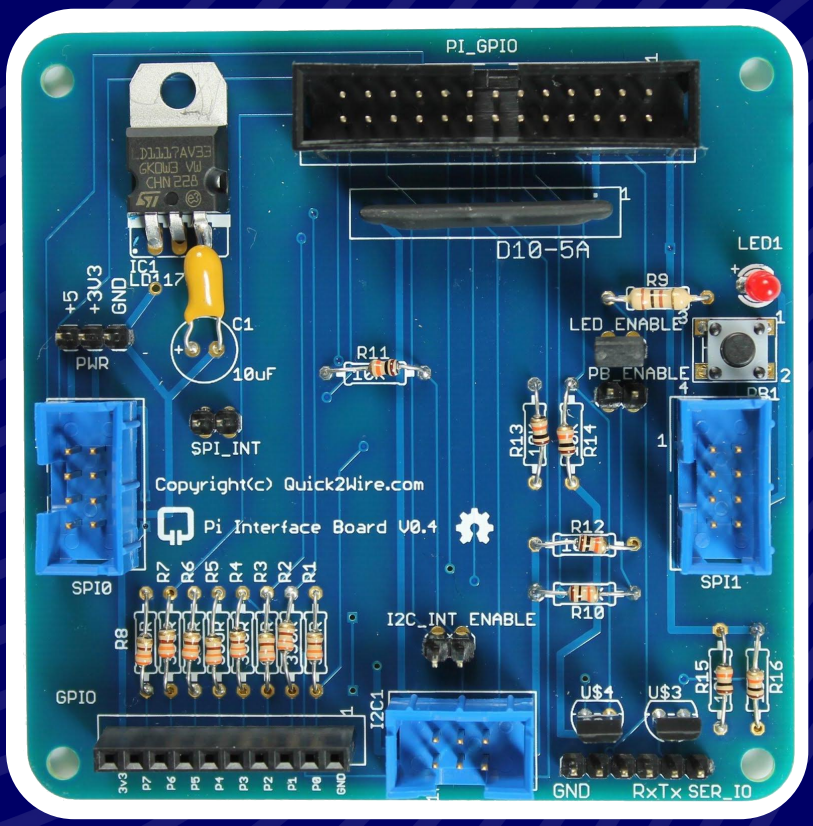
Entries must be received by **8 AM** UK time on **Monday 15th April**, and the winner will be announced on **1st May.**

How to enter

For more details on how to enter the competition please visit http://quick2wire.com/2013/03/quick2wire-competition-3/

Quick2Wire

SAFE AND SIMPLE CONNECTION TO YOUR RASPBERRY PI



HARDWARE

- Interface board
- I²C Port Extender
- Analogue board

SOFTWARE

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

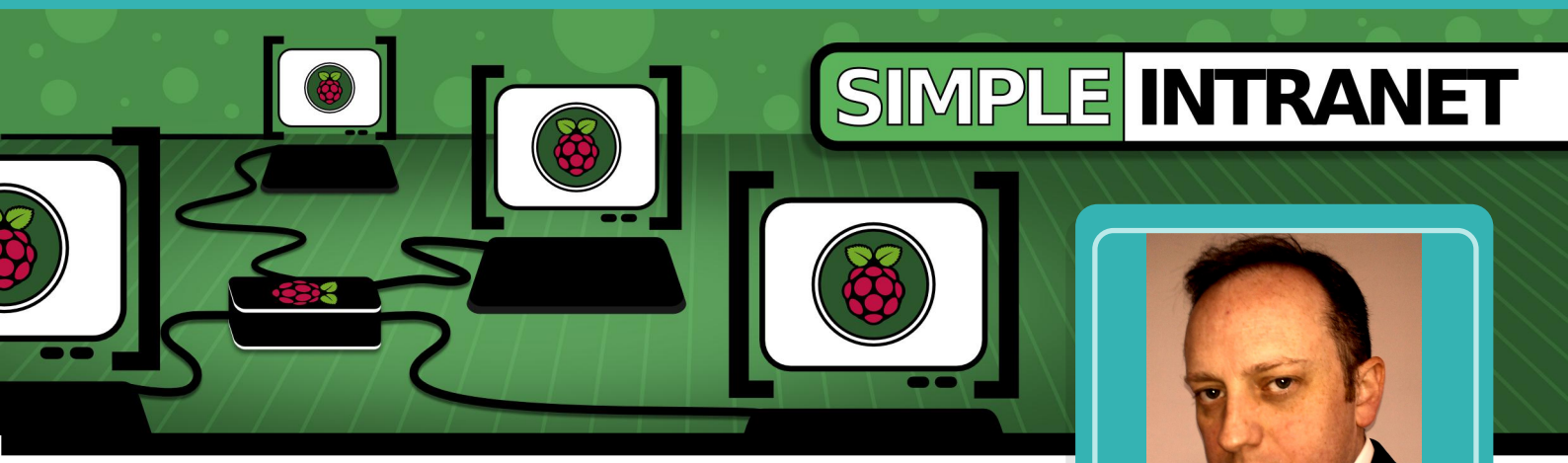
SUPPORT

- Articles
- Tutorials
- Forum

Interface board: £13.86 | Port Extender: £9.80 | Analogue board: £16.48

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

Find out more at quick2wire.com



Setting up a simple intranet

Jason "Jaseman" Davies
MagPi Writer

DIFFICULTY : MEDIUM

In this tutorial I will be showing you how to set up a very basic intranet that is hosted from a Raspberry Pi.

What is an 'Intranet'?

An intranet is similar to a website, however it is not accessible outside of your local area network (LAN). For example, if you set up an intranet at home, only the computers that are connected directly to your router/modem at your house can access it. If it is at work, only computers within your office can see it.

It is possible to create a Virtual Private Network (VPN) link to gain access from outside of your LAN. In this case a computer that has internet access can connect in to your LAN and then access the intranet site. We won't be covering VPN connections today though.

So without further ado, let's get started.

1. Changing your Raspberry Pi's hostname

You will want to change the hostname (Or network name) of your Raspberry Pi to match the name of your intranet. To keep it simple, lets just call it 'mynet'.

From the command line type:

```
sudo nano /etc/hostname
```

change the name (By default this is 'raspberry') to 'mynet'. Press CTRL+X to save the changes (Y to confirm). Then type:

```
sudo nano /etc/hosts
```

Change any references of 'raspberry' to 'mynet'. Again Press CTRL+X to save the changes (Y to confirm). Then restart your Pi by typing:

```
sudo reboot
```

2. Installing Apache2 Webserver

To be able to host a webpage on your intranet, you need to install something called 'Apache 2 Webserver'. To do this, go to the command line and type:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install apache2
```

We have to specify the ServerName for Apache2:

```
sudo nano /etc/apache2/apache2.conf
```

At the bottom of the file add:

```
ServerName mynet
```

Then press CTRL+X to save the changes (Y to confirm). Now restart apache,

```
sudo service apache2 restart
```

Note: the update and upgrade might take some time, but worthwhile to make sure you have the latest versions.

3. Check that Apache2 is working

Using another PC, tablet, netbook or smartphone on your network or wifi, open a web browser and visit http://mynet. If all is well you should see the default webpage for Apache2

You might find that your device cannot resolve the domain name. In which case you can access the page using the ip address of the Pi. To find out the Pi's ip address, type the following at the command line of the pi:

```
sudo ifconfig
```

Look for the inet addr for eth0 - it might be something like 192.168.1.75

Try browsing to http://192.168.1.75 (Or whatever the ip number is of your Pi)

4. Changing the default webpage

For this, it is helpful if you have some basic knowledge of the HTML language. There are lots of guides on the internet to get you started.

On the Raspberry Pi, go to the command line and type:

```
cd /var/www
ls
```

You should see a file listed called index.html. This is the file that opens when you visit http://mynet

To change the webpage we can use any text editor. I'm going to use nano:

```
sudo nano index.html
```

Alter the contents of index.html so that it looks like this:

```
<html>
<head>
<title>My Intranet</title>
</head>
<body>
<center>
<h1>Welcome to my intranet</h1><p>
</body>
</html>
```

Refresh or revisit http://mynet from your other computer to see the changes.

5. Adding sub-pages

Edit the index.html file again by typing:

```
sudo nano /var/www/index.html
```

Change it as follows:

```
<html>
<head>
<title>My Intranet</title>
</head>
<body bgcolor="green">
<center>
<h1>Welcome to my intranet</h1>
<p>Click <a href="page2.html">here</a>
for page 2.</p>
</body>
</html>
```

Then we will create page2.html. Type:

```
sudo nano /var/www/page2.html
```

and then add,

```
<html>
<head>
<title>Page 2</title>
</head>
<body bgcolor="red">
<center>
<h1>This is page 2</h1>
<p>Click <a href="index.html">here</a> to
return to the main page.</p>
</body>
</html>
```

After saving this file, refresh your browser and try using the links to switch between the index.html page and page2.html.

RISC OS

Raspberry Pi: Assembly Language

Bruce Smith

Guest Writer

DIFFICULTY : ADVANCED

Learn how to program your favourite computer in its native machine code by using Assembly Language.

One of the questions I get asked most of all is, “What’s the difference between assembly language and machine code?” The answer I give is simple enough, “Assembly language creates machine code.” This series will teach you the ins and outs of assembler and will show you how to create your own machine code programs to run on your Raspberry Pi.

Machine code is just that, the code that runs your machine. Every computer uses machine code; however, there are different types of machine code for different types of computers. The type is determined by the microprocessor the computer uses, and on the Raspberry Pi this is ARM.

In its purest form machine code is a sequence of numbers. Each number and the order of the numbers have meaning to your ARM chip. No matter what operating system you have running on your Raspberry Pi or what software you use on it, it is using machine code. So why not learn more about it?

Assembly language provides a form of shorthand that makes it easier to create machine code. It uses mnemonics to represent machine code operations – in the same way we use them to shorten SMS messages and Tweets. A standard assembler mnemonic will generally have three letters. What do you think these two mnemonics mean?

```
ADD
SUB
```

They are the mnemonics for addition and subtraction.

```
MOV
```

is the mnemonic for the move command. It takes information from one place and places it in another place. How easy is that?

Assembling a Machine Code Program

One of the easiest ways — but certainly not the only way — of writing an assembler program is to use BBC BASIC. This is available as part of the RISC OS operating system which is available in Raspberry Pi flavour from the Downloads page on the Raspberry Pi website. So it’s free and readily available.

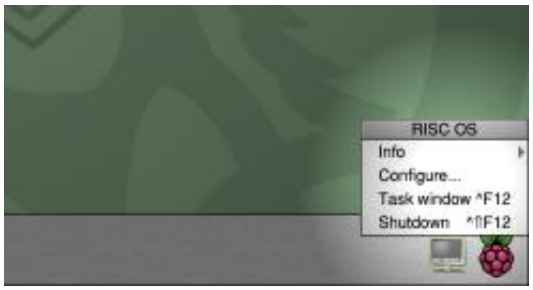
With RISC OS running, there will be a raspberry icon in the bottom right hand corner of the screen (the task bar). Move the pointer over it and from the pop-up menu select ‘Task Window’

(See image at right).

A new window will be displayed showing an "*" as a prompt. In this window type:

```
BASIC
```

And press the RETURN key. This will start BBC BASIC.



Select 'Task window' from the Raspberry menu to access BBC BASIC in RISC OS.

You can then enter the program shown in PANEL 1. When you have entered it, type:

```
RUN
```

and the program will be assembled. You will get an assembly listing which will look something like this:

```
00008FAC
00008FAC                        .START
00008FAC     E3A0002A           MOV R0, #ASC ("*")
00008FB0     EF000000           SWI "OS_WriteC"
00008FB4     E1A0F00E           MOV R15, R14

PROGRAM ASSEMBLED AT: &008FAC
PROGRAM SIZE IS: 12 BYTES LONG
```

If you get any error messages check the line indicated carefully. Programs are case sensitive so ensure you have entered it exactly as shown.

The assembly listing has several columns. The first is the memory address where each instruction is placed (and this will most likely be different on your Raspberry Pi). The second column contains the machine code. The number sequence:

```
E3A0002A
```

is the machine code for:

```
MOV R0, #ASC("*")
```

The third column is where labels are placed. Finally the fourth column is the assembly language for each line.

The first line shows how we can integrate BASIC commands in the assembler. We did not have to look-up the ASCII code for an asterisk; we used the ASC function of BASIC to do it for us. You can use all BASIC functions in this way. This integration with BASIC can be as simple or complex as required. If it is a legitimate function, it will assemble correctly. Something like this is acceptable and will assemble:

```
MOV R0, #INT(SIN(DEG 60)*100)
```

Try it!

Running Machine Code

To execute the machine code we use the CALL statement thus:

```
CALL START
```

Here 'START' is the variable label which signifies where the machine code begins and which the BASIC program defined in line 50. When you CALL the program, you will see an asterisk printed on the screen. This machine code could also have been executed by typing:

```
CALL CODE%
```

or:

```
CALL &8FAC
```

In this last case use the value printed out on the screen by the 'Program Assembled at:' statement, as it may be different for you.

Saving Machine Code

We used the SAVE command to save the assembly language program. To save machine code we use the *SAVE command and we need to know its start addresses and length of the machine code. These should have been printed out by lines 100 and 110 of the program. The command *SAVE has this format:

```
*SAVE <filename> <startaddr> + <length>
```

For the example above, the *SAVE command would look something like this:

```
*SAVE asterx 8FAC +12
```

You will need to use the numbers your program printed out.

The *SAVE command does not know it is saving machine code, what it does is save an area of memory. The address of the last byte of memory saved is given by adding the length to the start.

Once you have the machine code file saved, you have something you can execute directly from file. This can be performed using the *RUN command thus:

```
*RUN asterx
```

The program will load into the address where it was originally assembled and then run. The result will be the asterisk as before.

If you are running machine code you can omit the RUN portion and just use the filename thus:

```
*asterx
```

ARM Locations

The program uses an area of memory that is inside the ARM chip called a register (R0) to hold the ASCII value of the asterisk in line 60 and they use a register again in line 80 to return to BASIC. We'll examine this in depth next time.

BREAK OUT PANEL 1: Program and Description

```
10 REM >ASTERISK
20 DIM CODE% (100)
30 P%=CODE%
40 [
50 .START
60 MOV R0, #ASC ("*")
70 SWI "OS_WriteC"
80 MOV R15, R14
90 ]
100 PRINT "PROGRAM ASSEMBLED AT: &"; ~CODE%
110 PRINT "PROGRAM SIZE IS: "; P%-START; "
BYTES LONG"
```

Line 10 is a REMark statement. If you type:

```
SAVE
```

RISC OS will use the text after the '>' in a REM statement in the first line as a filename. So this program would be saved as 'ASTERISK'.

Line 20 reserves 100 bytes of space pointed to by the variable CODE%.

The variable P% is used by BBC BASIC to select where in memory the machine code will be stored. Line 30 points P% at the space reserved by CODE%

Opening and closing square brackets (lines 40 and 90) are used to mark the start and end of the assembly language program. Anything inside these lines is assumed to be assembly language.

Line 50 is a label called 'START'. Labels are always preceded with a full-stop.

Lines 60,70 and 80 are the assembly language instructions to print an asterisk on the screen. The ASCII code for an asterisk is loaded into Register (line 60) and then printed on the screen (line 70). Line 80 returns control back to BASIC.

Line 100 and 110 print out where the program was assembled and how long it is.

Tip Panel

If you are having trouble getting at certain characters on your keyboard in RISC OS, try setting your keyboard driver to the USA by typing:

```
*KEYBOARD USA
```

About the Author

Bruce Smith is an award winning author. His book, Raspberry Pi Assembly Language Beginners, is now available from Amazon.

Check him and his book out at www.brucesmith.info for more Raspberry Pi resources.

An introduction to Charm Data Types

DIFFICULTY : ADVANCED

Peter Nowosad

Guest Writer

An Introduction to Charm Data Types

Computers can accurately and quickly manage, manipulate and transform information in accordance with sets of coded instructions packaged into operating systems and programs. In this article of the Charm series I am going to concentrate on how Charm defines and handles different types of data used to represent that information.

Basic Data Types

Currently Charm defines four intrinsic types of data as shown in the following table:

Keyword	Size (bytes)	Use	Example
boolean	1	Logical flags	true
char	1	Characters	'C'
int	4	Integers	-321
real	8(d) or 12 (e)	Real numbers	1.23e-7

`boolean` values represent flags that are either `true` or `false`, and strictly speaking only require one bit of data. It is possible to combine up to 8 boolean flags in a `char` or 32 flags in an `int` using the Charm bit operators. For convenience and speed however, the compiler does not attempt to pack `boolean` definitions together but stores each in its own byte location which is the smallest unit of memory the ARM CPU can directly address.

`char` values represent single characters using the ubiquitous ASCII code. Some characters are non-printing and for historical reasons these are known as control characters, the most common being space (32), tab (9), linefeed (10) and carriage return (13).

`int` values represent signed 32 bit numbers in the range - 2147483648 to 2147483647 with negative numbers having the top bit set.

When handling `real` numbers Charm will use IEEE double precision floating point if VFP support is enabled, or extended precision if using the standard RISCOS floating point emulator. Note that at the cost of a few trailing digits, floating point operations complete an order of magnitude faster with VFP support as they are performed in hardware rather than software.

Constants

Charm allows constant values of basic types to be named using the `const` keyword. By convention the source code in the Charm distribution uses upper case with an

underscore delimeter between words for constant names. The type of the constant is inferred from the value it is assigned (using BNF notation) as follows:

- `[true | false]` - `boolean`
- `'?'` - `char` e.g. `'Q'`
- `[+ | -] [0 - 9]+` - `int` e.g. -123
- `[+ | -] [0 - 9]+ . [0 - 9]+ [e [+ | -] [0 - 9]+]` - `real` e.g 2.345e-6

The `hex` and `bin` prefixes introduce constants that are coded in hexadecimal (`[0 - 9, A - F]+`) or binary (`[0 , 1]+`) number bases e.g. `hex 20` or `bin 100000` for the space character.

Non printing characters which are not part of the Charm language per se can be included as character data by escaping them using the `'\'` character e.g.

- `'\t'` - tab
- `'\n'` - carriage return
- `'\\'` - backslash

Note that constants may contain expressions and these may reference other constants as long as the result of the expression can be resolved at compile time e.g.

```
const  ROWS = 4, COLUMNS = 6,
CELLS = ROWS * COLUMNS;
```

It is also possible to declare lists of constants using the `list` keyword. In this case numbers are implicitly assigned according to position in the list starting from 1 and going up to the number of list elements e.g.

```
list RED, GREEN, BLUE;
```

assigns values 1, 2 and 3 to the constants RED, GREEN and BLUE respectively. This is particularly useful when defining an enumeration defining a finite set of distinct but associated values that can for instance be used as case statement labels.

Variables

A Charm variable must be declared before it can be used. In the declaration the variable is given both a name (conventionally lower case with an underscore delimiter between words and starting with a `z` for a pointer) and a type. Variables can be declared to be of either a basic or more complex type, with complex data types being derived from basic types in a number of ways:

- by homogeneous aggregation using the `array` keyword.
- by hetrogenous aggregation using the `record` keyword or using `module` level `dynamic` data declarations.
- by using the `ref` keyword prefix to create a reference (or pointer) to a variable of a given type.

Arrays contain data collections accessed using an index that runs from 0 to one less than the size of the array using square bracket notation e.g. `variable [ index ]`. The index must resolve to type integer.

Records contain a collection of typed fields each of which is given a different name within the record name space so that it can be accessed in the form `<record name>.<field name>` where by convention record name is in camel case. Dynamic module data can be access in a similar fashion as `<ref module name>.<variable name>` where the module name reference may be `this` if inside a `dynamic` procedure.

`ref` type pointer variables are automatically dereferenced by Charm when the variable to which they point is accessed. Variables of complex data type are always passed by reference to procedures though they can be declared directly as static or stack based variables. While the `proc` keyword introduces procedural code, it is possible to declare variables of type `ref proc` that point to procedures and to execute those procedures via their pointers if the correct signature is adhered to. Note that since procedures can

optionally return a value of a specified type using the `return` keyword in which case calls to them can be substituted in expressions for a variable of that type.

Variables of type `ref` can be assigned the special value `nil` which means that they are not currently pointing anywhere. When set to this value an attempt to use them to access data will result in a runtime exception, however they can still safely be compared with the `nil` value to see whether they contain a valid reference.

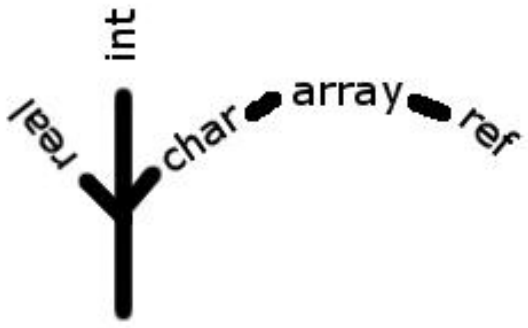
Note that when comparing pointers the usual = and # comparison operators become `:=:` and `:#:` since using the first form will cause the pointers to be dereferenced before the comparison is made.

Strings

Now let's take a closer look at a particularly prevelant type within Charm, namely `ref array char` to point to a string:

```
array [] char hello = ('H', 'e',
'l', 'l', 'o', 0);
ref array char hello_string = hello;
```

where the relevant portion of the Charm compiler type tree is:



We can output this to the screen on a line of its own using the standard `vdu` stream of the `Out` library module as follows:

```
Out.vdu.str (hello_string).nl ();
```

A few points to note are:

• All strings are terminated by a zero character.

• The length of the string array is implicitly calculated by the compiler, however if it were to be explicitly entered it would be `[6]` to accommodate the terminating zero.

• Characters can be extracted from the string using an array index starting from 0 and running to one less than the string length, for instance `hello_string[1]` contains `'e'`.

Since strings are used so much, the compiler associates the type of a string literal such as `"Hello"` with type `ref array char` and automatically adds the trailing zero, so more concisely we can write:

```
ref array char hello_string = "Hello";
```

In fact one of the signatures of the Charm `~start` procedure which defines the entry point for a Charm program takes two parameters, the first of which is an integer specifying the number of command line arguments being passed in and the second of which is of type `ref array ref array char` which while at first sight looking somewhat daunting is simply a pointer to an array of strings (one per argument).

The story on strings does not end there, as the Charm libraries contain a `String` module that can hold a dynamic instance of a modifiable string. This is held in the heap, can grow as required, and is automatically released by the `String` destructor when the `String` object goes out of 'scope' (see below).

Type Conversions

In general the compiler will not allow variables of different types to be combined in assignments or expressions since this frequently this makes no sense e.g. assigning a `real` to a `boolean`. The three arithmetic types `char`, `int` and `real` can however be mixed, and the compiler will generate a result of the widest width (greatest number of bits) of the operands. You may also mix these types on the left and right of assignment statements, however watch out for loss of precision if the left hand side is narrower than the result being assigned to it.

Charm does permit a couple of powerful casting operations through the `address (variable)` construct which can convert a variable of any type to an integer and the `[int]` construct which can convert an integer to any type. With great power comes great responsibility however, so these are used but rarely and then with care. You can see an example in the Charm `debug` library module which casts the heap address to an array of bytes so that their contents can be dumped to a file.

Scope

Location	Lifetime	Construction
static	program	type v = value;
heap	variable	type v := new V;
stack	scoped	type v := value;

The scope of a variable relates to from where it is accessible and is associated with its lifecycle. A variable can reside in one of the following places:

Variables declared statically exist for the lifetime of the program and are initialised at compile time, by default to zero or `nil` for references. Variables declared on the heap are created by the `new` operator and released using the `delete` operator when no longer needed. Variables on the stack exist until their containing scope is released. Typically this is when a procedure exits, or execution of a block of code between { and } delimiters finishes.

The specially named `~startup` and `~shutdown` procedures can assist with setting up static variables, and the `~new` and `~delete` procedures can help with setting up heap and stack based variables when more than a single compile or run time assignment is required.

Importantly for information hiding, by default constants, variables and procedures are only accessible from within the module in which they are declared. By using the `export` keyword however, it is possible to make these definitions publicly available to other modules. Referencing modules are required to explicitly state their dependencies using the `import` keyword to state each module whose public definitions they wish to use. Such references must then be qualified with the name of the imported module e.g. `Out.vdu.str` refers to the `str` procedure of the `vdu` instance of the `Out` library module used to output strings to the screen.

Next Time

Next time I intend to cover some of the coding side of Charm including examples of sequence, selection and iteration under the umbrella of writing expressions, assignments and flow of control statements.

Exercise

Finally for this month's exercise I suggest replacing the default `!NewProject MyProject` module with the following code to output square roots for the first dozen integers calculated using the Newton Raphson method:

```
import lib.Out;

module MyProject
{
  proc sqrt (int n) real
  {
    real v := 1;
    for int i := 0 step inc (i)
    while i < 5 v :=
    (v + n / v) / 2;
    return v;
  }

  export proc start (ref array ref
  array char argv)
  {
    for int i := 1 step inc (i)
    while i <= 12
    Out.vdu.str ("sqrt ").num_fld
    (i, 2).str (" = ").float_fld
    (sqrt (i), 10).nl ();
  }
}
```

CONSOLE COLOURS

ADDING COLOUR
to the terminal

W. H. Bell
MagPi Writer

Colours using escape sequences

DIFFICULTY : BEGINNER

Sometimes plain text on a terminal background is just not exciting enough. While there is a full GNU library called ncurses, this article takes a look at how to use simple escape sequences to control colours from several different languages.

Escape sequences affect the colour and formatting when text is printed to a terminal window. This means that if some text is printed to a file and then viewed with less, the the result will not have the same formatting. The effect of escape sequences is dependent on the terminal type. Some terminals support 256 colour, whereas others only support 8bit colour codes. Since 8bit colour is most likely to be available, this article just discusses 8bit codes.

In the case of emergencies, red blinking text can come in handy. For example, using C

```
#include <stdio.h>
int main() {
  char redBoldBlinking[] = "\033[0;1;5;31m";
  char defaultConsole[] = "\033[0m";
  printf("%s",redBoldBlinking);
  printf("NEED CHOCOLATE!\n");
  printf("%s",defaultConsole);
  return 0;
}
```

or Python,

```
redBoldBlinking = '\033[0;1;5;31m'
defaultConsole = '\033[0m'
print redBoldBlinking+"NEED
CHOCOLATE!"+defaultConsole
```

or Bash,

```
redBoldBlinking='\033[0;1;5;31m'
defaultConsole='\033[0m'
printf $redBoldBlinking
echo "NEED CHOCOLATE!"
printf $defaultConsole
```

escape sequences can be used to control the colour and formatting of text printed on a terminal screen. This technique can be used in the same way with other languages too. The character `\033[` is the Control Sequence Introducer (CSI). The CSI character can be used to pass several different commands to the terminal window, such as clear the screen, bell and reposition the cursor. The examples in this article use `\033[a;b;c;dm`, where `a=0` resets the terminal, `b=1` turns on bold, `c=5` turns on blinking, `d=31` sets the foreground colour to red and the sequence is terminated by `m`. The `defaultConsole` has a single value `0`, which is to reset the text to its default colour. Escape sequences can be used to turn on just one colour or text type.

Colours	Styles
0 - black	0 - normal display
1 - red	1 - bold on
2 - green	2 - faint
3 - yellow	3 - standout
4 - blue	4 - underline
5 - magenta	5 - blink
6 - cyan	7 - reverse video
7 - white	

Add 30 to the colour value to control the foreground and 40 to the colour value to control the background.

The Raspberry Pi 1st Anniversary Competition

WIN a Limited Edition **BluePi!**

Want to get your hands on a limited edition blue Raspberry Pi from RS Components? As part of the celebrations to mark the 1st Anniversary of Raspberry Pi, The MagPi is pleased to announce a competition where ten lucky winners will be able to do just that. Only 1,000 Raspberry Pi Model B, Revision 2 boards have been produced by RS in an exclusive blue colour. Each limited edition board is stamped with a serial number and is supplied in a special presentation box together with a crystal blue user case and a certificate of authenticity signed by Eben Upton.

To be in with a chance of owning one of these blue beauties, all you have to do is visit **www.themagpi.com/bluepi** and answer the seven questions displayed on the right. All correct entries will be put in to a prize draw, with the first ten drawn winning a Blue Raspberry Pi. Winners will be notified by email. For full terms and conditions visit the competition page link below.

1. On what date did the Raspberry Pi Foundation launch the Raspberry Pi?
a) February 28th 2012 b) February 29th 2012 c) March 1st 2012

2. How long did the Raspberry Pi take to develop?
a) 1 year b) 4 years c) 6 years

3. What was the driving factor when developing the Model B board?
a) Price b) Size c) Design

4. Which TV network helped champion the Pi story in the early days?
a) ABC b) BBC c) ITV

5. Which number is closest to the number of components on the Raspberry Pi board?
a) 137 b) 167 c) 187

6. Approximately, how many Raspberry Pis are in circulation?
a) 50,000 b) 1 million c) 3 million

7. How many Raspberry Pis did the Foundation originally anticipate to sell?
a) 10,000 b) 100,000 c) 1,000,000

*All questions relate to the Model B Raspberry Pi

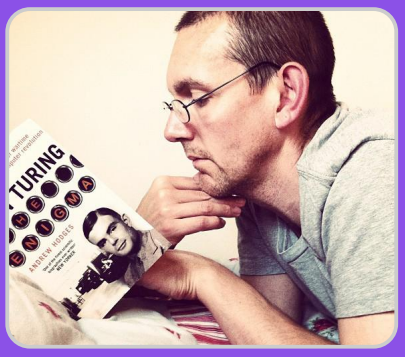
Enter at **www.themagpi.com/bluepi**

More details will be available on the competition page

The MagPi™

King of the Heap (Sort)

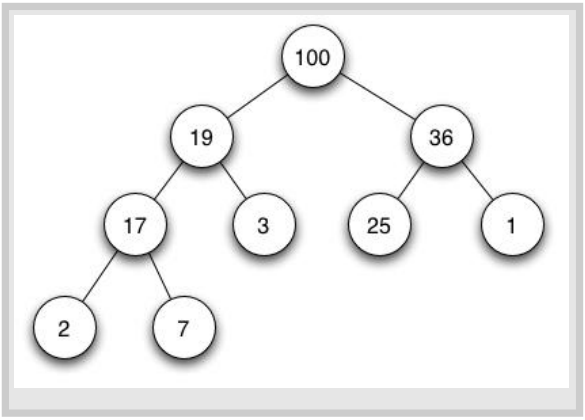
DIFFICULTY : EASY



antiloquax
evil genius

In this month's article we are going to implement Heap Sort.

The big idea here is the "Heap".
If you have a set of numbers, you can arrange them in a kind of tree structure.



Rules of the Heap

- each node can have up to two child nodes.
- each child must be less than or equal to its root.

Scratch On!

Here's a Heap in a Scratch list. The first item is the biggest number.
Its "children" are at positions 2 and 3.
In fact, the children of a node at index i are at index 2i and 2i+1.

Heap Sort is much faster than Bubble Sort (which we looked at in Issue 6) because, on average, it makes a lot less swaps.

You can "heapify" a list (yes that is a real word!) quite quickly.
Then you know the number at the top of the heap must be the biggest.
You then put this at the end of your list and heapify the values above it.
If you keep repeating this, the list will be sorted.

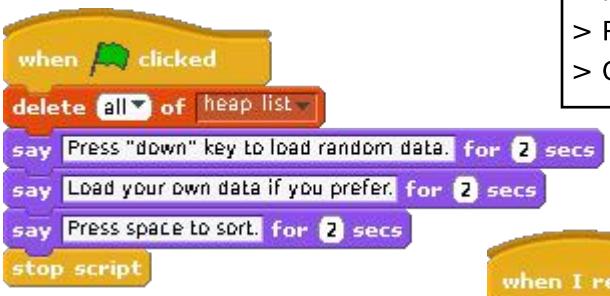
Speed Test (50 numbers):	Bubble Sort made 637 swaps and took 52s. Heap Sort made 245 swaps and took 37s.

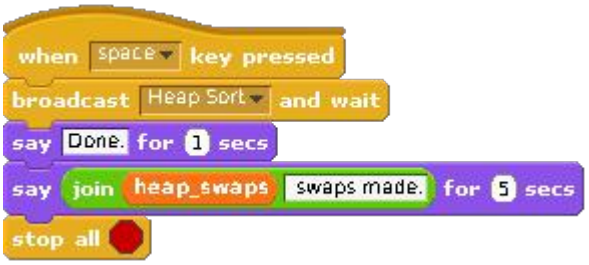
To import your own list of numbers:
> Make a text file with a number on each line.
> Right-click your list (on the stage).
> Choose "import" and select your file!

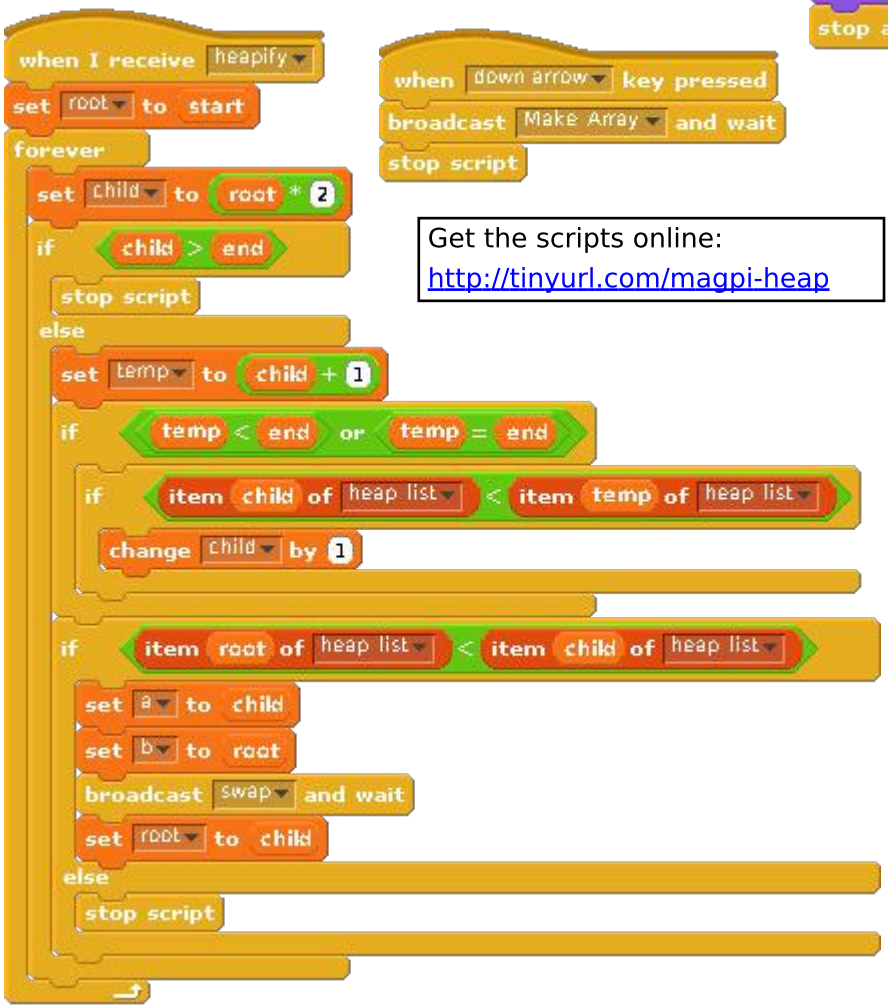
my_list.txt

File Edit

```
1 6
2 1
3 9
4 2
5 0
6 4
7 8
8 5
9 3
10 7
```







Get the scripts online:
http://tinyurl.com/magpi-heap

Feedback & Question Time

I have cerebral palsy which means that my movements are not very accurate!! So using a soldering iron is not possible. It also makes using windows difficult and I use Linux for that reason. So I love the Raspberry Pi, but the only problem is when I try to get add-ons for it, it is not always clear from the advertisement whether the add-on comes soldered together or not. Indeed on a couple of occasions the picture was of the assembled part, but when I ordered it, the part that arrived needed soldering, rendering it useless for me!! Could I please use the pages of your magazine to plead with suppliers to make clear whether their parts need soldering or not?

Paula Thomas

[ed: This is a very valid point, and one which should be addressed by kit suppliers.]

I think that your magazine is pretty awesome.

Michel Bargone

The magazine is excellent, and I enjoy them all very much. You are doing a great job!! I'm very impressed. Keep up the good work.

John Franklin

Congratulations for an excellent magazine.

Steve

Congrats on a great mag... Can you issue an upgrade allowing the deletion of separate issues [in the newsstand app]?

Alister Lockhart

[ed: This update will be made to the newsstand app - available on iTunes, if you don't already have it (search for The MagPi)]

I just want to say how much I enjoy reading the MagPi. An interesting and well produced magazine. Keep up the very good work.

Nigel Trewartha

I've only got to play around with a Raspberry Pi just recently, even though I've had a model B sitting on my desk since he middle of last year, and I'm impressed with The MagPi magazine that I downloaded and read on the iPad - it certainly takes me back to my early days of hacking around with an Acorn Electron and various Spectrums etc.

Adrian Harper

I found The MagPi only through google and I must say it is really very interesting. My question is, will there be more ones [issues] in german and also is there any way we can support you except with money (I did Donate a bit...)

Paolo Pietroluongo

[ed: There is constant work going on translations of the magazine. If you would like to help translate the magazine to an existing, or new, language, please get in touch with us.]

The MagPi is a trademark of The MagPi Ltd. Raspberry Pi is a trademark of the Raspberry Pi Foundation. The MagPi magazine is collaboratively produced by an independent group of Raspberry Pi owners, and is not affiliated in any way with the Raspberry Pi Foundation. It is prohibited to commercially produce this magazine without authorization from The MagPi Ltd. Printing for non commercial purposes is agreeable under the Creative Commons license below. The MagPi does not accept ownership or responsibility for the content or opinions expressed in any of the articles included in this issue. All articles are checked and tested before the release deadline is met but some faults may remain. The reader is responsible for all consequences, both to software and hardware, following the implementation of any of the advice or code printed. The MagPi does not claim to own any copyright licenses and all content of the articles are submitted with the responsibility lying with that of the article writer. This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit:

http://creativecommons.org/licenses/by-nc-sa/3.0/

Alternatively, send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.