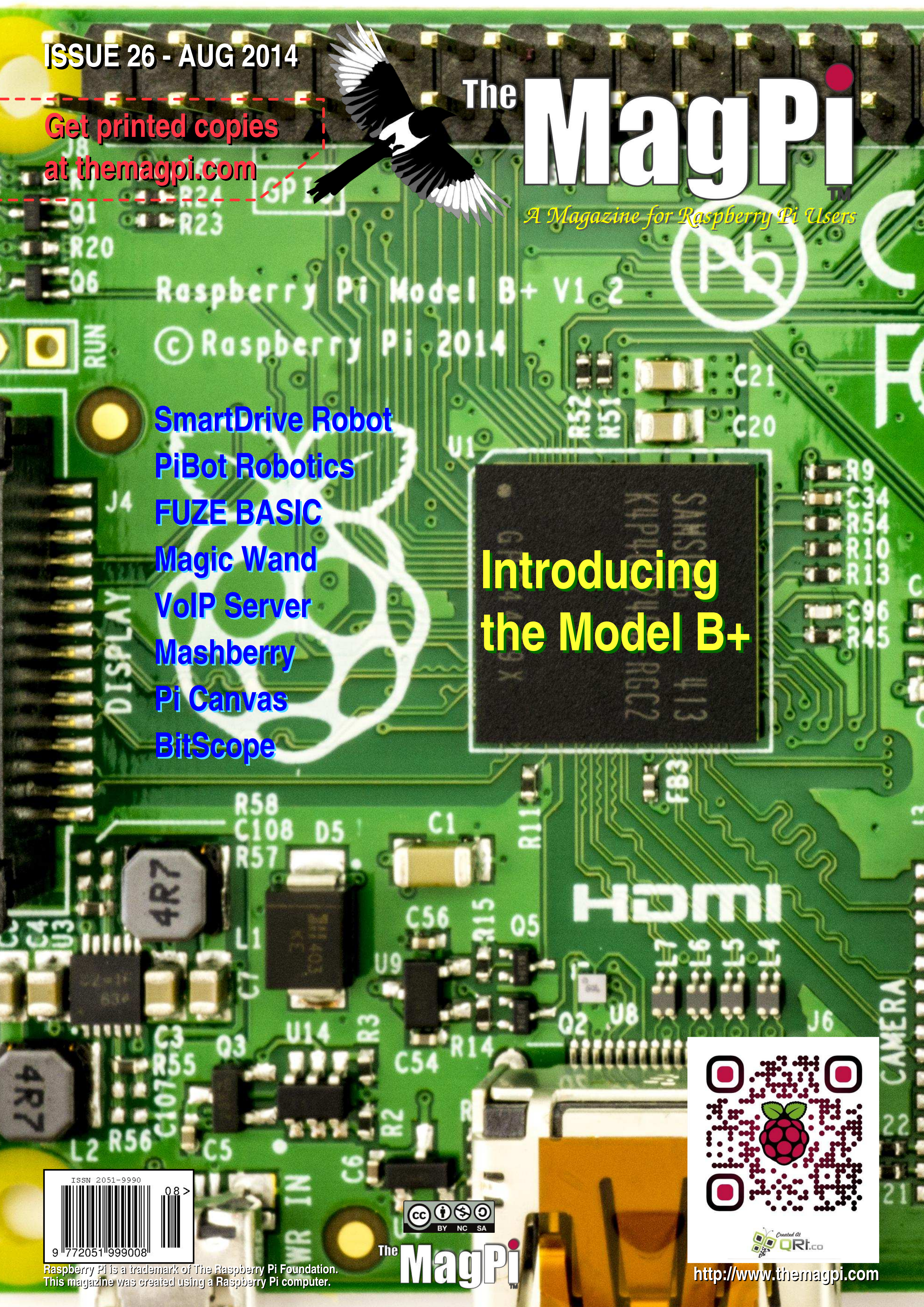
ISSUE 26 - AUG 2014

Get printed copies at themagpi.com

The MagPi

A Magazine for Raspberry Pi Users



- SmartDrive Robot
- PiBot Robotics
- FUZE BASIC
- Magic Wand
- VoIP Server
- Mashberry
- Pi Canvas
- BitScope

Introducing the Model B+

ISSN 2051-9990

http://www.themagpi.com

Welcome to Issue 26 of The MagPi magazine.

This month's MagPi contains another great selection of hardware, software and programming articles. Michael Giles explains how to turn your Raspberry Pi into a magic wand with a fun hardware project that demonstrates the persistence of vision effect, while John Mahorney from Suncoast Science Center in Florida shows how a Raspberry Pi is used to display dynamic art. Big news this month is the launch of the Model B+ and MagPi writer Aaron Shaw has all the details for you on page 22.

Robotics is always a popular theme and Harry Gee continues on from last month's article and shows how to add speech, voice control and facial recognition to your robot. Additionally, Rishi Deshpande describes how to use the SmartDrive controller board to easily control high current motors.

Another popular topic is beer and Sebastian Duell explains his hardware and software "Mashberry" project which he uses to optimise the mashing process. Karl-Ludwig Butte continues his series on using a Raspberry Pi and BitScope for electronic measurement plus we have an interesting article by Walberto Abad on how to set up your own VoIP telephone system.

Last, but definitely not least, Jon Silvera continues his series on learning how to program with FUZE BASIC. This is an exciting article with lots to discover. I certainly don't remember ever being able to control sprites this easily with BASIC!

We try to make sure each issue of The MagPi contains something of interest for everyone, regardless of age and skill level. Have we got the balance right? Let us know by sending an email to editor@themagpi.com or comment on our Facebook page at http://www.facebook.com/MagPiMagazine.

Finally, a big thank you to all the volunteers who work hard behind the scenes to produce The MagPi for your enjoyment and education.

Let's get started...

Ash Stone

Chief Editor of The MagPi

The MagPi Team

Ash Stone - Chief Editor / Administration
Ian McAlpine - Issue Editor / Layout / Proof Reading
W.H. Bell - Administration
Bryan Butler - Page Design / Graphics
Matt Judge - Website / Administration
Aaron Shaw - Layout
Nick Hitch - Administration
Colin Deady - Layout / Testing / Proof Reading
Dougie Lawson - Testing
Nick Liversidge - Layout / Proof Reading
Age-Jan (John) Stap - Layout
Claire Price - Proof Reading
Rita Smith - Proof Reading

Contents

4 **MAGIC WAND**
Persistence of vision: build a magic wand with an accelerometer

8 **PI CANVAS DIGITAL ART DISPLAY**
Display dynamic art using a Raspberry Pi

12 **SMARTDRIVE ROBOT**
Coding a remote-controlled robot with the SmartDrive add-on board

14 **MASHBERRY**
Homebrewing with the Raspberry Pi

18 **PIBOT**
Part 2: Add the power of speech, hearing and vision to your robot

22 **RASPBERRY PI MODEL B+**
All the details on the latest addition to the Raspberry Pi range

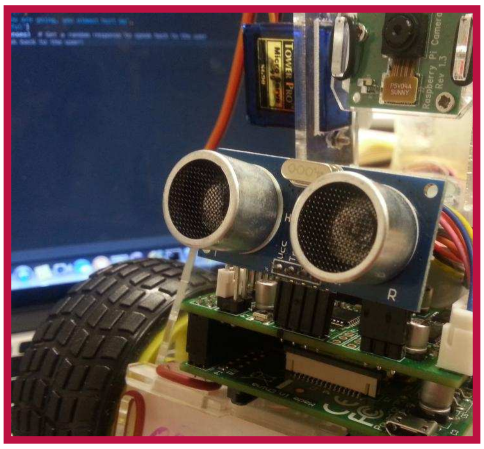
26 **BITSCOPE**
Part 2: Electronic measurement with the BitScope oscilloscope add-on board

30 **VOICE OVER IP SERVER**
Using Asterisk to implement a low cost telephone system

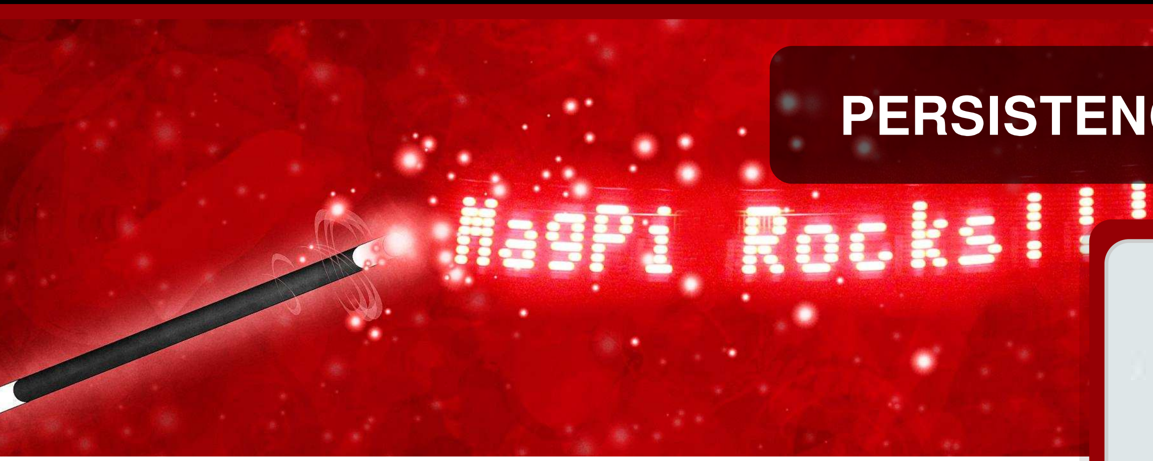
34 **THIS MONTH'S EVENTS GUIDE**
Southend UK, Mountain View CA, Malvern UK, Manchester UK, Winchester UK

36 **FUZE BASIC**
Part 2: Variables, procedures and sprites

44 **HAVE YOUR SAY**
Send us your feedback and article ideas



http://www.themagpi.com



Build a magic wand with an accelerometer

SKILL LEVEL : INTERMEDIATE

Michael Giles

Guest Writer

Persistence of Vision displays create an image by quickly displaying one column of pixels at a time. When the device moves rapidly along a linear path the human eye can view the image as a whole as it is built up column by column. When I say rapid I mean rapid, at least one twenty-fifth of a second. At this speed an afterimage is seen in the retina and the viewer perceives the rapid succession of LED blinks as a full image.

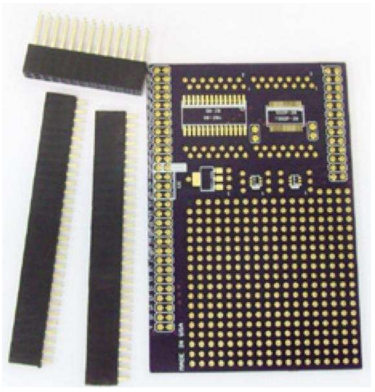
Operation

The magic wand displays five columns of pixels for each specific letter in a user defined string. The accelerometer is used to determine which direction the wand is swinging to avoid displaying the string in reverse.

Project parts list

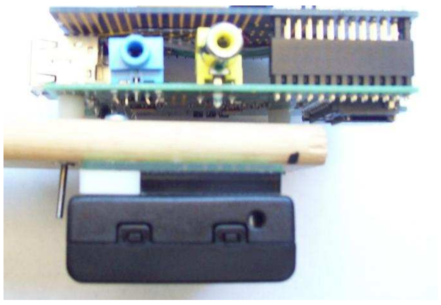
1x Raspberry Pi
1x Pi Prototyping kit (OpenElectrons.com)
1x LSM303 breakout (OpenElectrons.com)
1x SmartUPS (OpenElectrons.com)
1x PCF8574 chip, manufacturer part PCF8574ADW
1x 10uF through-hole capacitor, ESK106M050AC3AA
2x 82K through-hole resistors, 271-82K-RC
8x 4mm flat top red diff LEDs, HLMPM201

Obtaining parts



To build the circuit I used the Pi Prototyping Kit from OpenElectrons.com. It has multiple integrated circuit footprints and unwired through-holes for great prototyping flexibility. For the accelerometer I used the LSM303 breakout board, also from OpenElectrons.com.

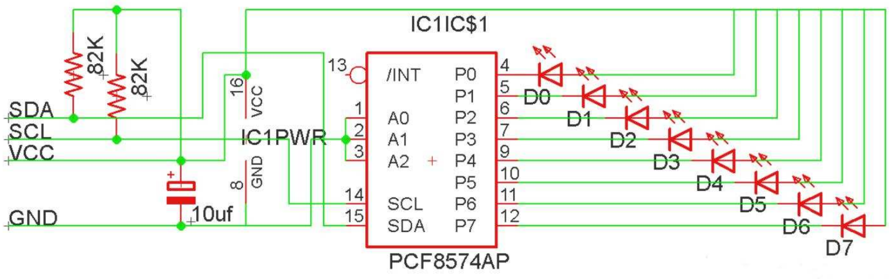
Magic wand assembly

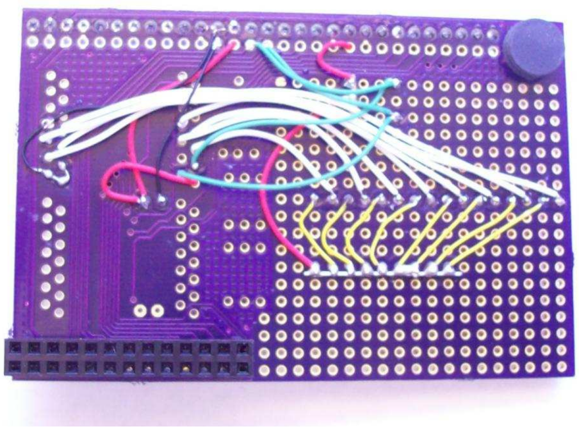


The circuit build for the magic wand required soldering of through-hole components as well as the surface mount PCF8574 chip. The datasheet showed the In/Out ports of P0-P7 to which I connected the eight LEDs. In the first build I placed a bussed resistor array between the 5V power and the LEDs to limit the current, but when I tested I realized the lights were extremely difficult to see in the daylight. I then shorted the resistor array and the LEDs became much more visible.

Connecting the LSM303 breakout was by far the simplest part. The Pi Prototyping board has two I^{2}C female connections for a quick plug in for breakout boards.

The full schematic for the project is shown below.





Apparatus assembly

In order to move the LEDs fast enough to see the image, I attached the Raspberry Pi along with the Pi Prototyping board to a long flat wooden stick using screws and spacers. Wood was used to avoid any shorting that may occur. On the back of the stick I attached the SmartUPS to power the Raspberry Pi and make my device more mobile. The SmartUPS is powered by three AA batteries and, though it has several functions, is only used for power in this project. On the other end of the wooden stick I drilled a hole and attached a rod to be used as a handle. This allowed me to easily swing the magic wand around in a circle.

Installing packages

For this program I used the OpenElectrons_LSM303 package installed using the pip package manager. If you do not have pip you can get it by opening a terminal window and typing:

```
sudo easy_install pip
```

To install the OpenElectrons_LSM303 package type:

```
sudo pip install OpenElectrons_LSM303
```

This command will install the package as well as the OpenElectrons_i2c package needed for I^{2}C functions.

Programming

In order to generate the ASCII characters, I first created a look up table. The table is just a dictionary in which each character is a list containing five hexadecimal values. Each value generates one vertical line of pixels. The example below shows the hexadecimal for the letters B, C and D. A full alphabet, with numbers and symbols can be downloaded in the Python sample programs available from: http://www.openelectrons.com/pages/63

```
#5x7 ascii characters lookup table
lookuptable = {
...
    'B' : [0x7f,0x49,0x49,0x49,0x36],
    'C' : [0x3e,0x41,0x41,0x41,0x22],
    'D' : [0x7f,0x41,0x41,0x41,0x3e],
...
}
```

With the look up table created, I then started writing the program by importing any needed files and defining my variables. Note that only ACCEL is imported from the OpenElectrons_LSM303 file. Because the LSM303 chip contains a magnetometer, as well as an accelerometer, the library contains two separate classes. Also, notice the 'str' variable. This is the string the magic wand will display.

```
import time
import os, sys
```

```
from OpenElectrons_i2c import
 OpenElectrons_i2c
from OpenElectrons_LSM303 import ACCEL

test = 1
oe = OpenElectrons_i2c(0x38)
lsm = ACCEL()
str = "MagPi Rocks!!!"
length = len(str)
index = 0
test = 1
g = 9.81
t = .05
print str
```

The `while` loop starts by turning off the LEDs. Next it reads the accelerometer value along the X-axis then quickly reads the value again. With these two values a simplified calculation of acceleration can be performed.

```
while test == 1:
  #turn leds off
  oe.simpleWriteByte(0xff)
  #get first value
  array = lsm.readArray(lsm.ACCEL_X_Y_Z | 0x80, 6)
  aclraw = lsm.accl(array, 0)
  time.sleep(t)
  #get second value
  array = lsm.readArray(lsm.ACCEL_X_Y_Z | 0x80, 6)
  aclraw2 = lsm.accl(array, 0)
  #divide values to compensate for gravity
  #and subtract to find delta
  acl = (aclraw2/g) - (aclraw/g)
  #filter approximate still values
  if acl <= 30 and acl >= -30:
      acl = 0
```

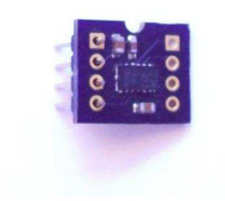
The following `if` statement and `while` loop actually light the LEDS. The `while` loop accesses the look up table for every letter in the string one at time. This allows any message inserted into the 'str' variable to be displayed without any other changes in the program.

```
#if moving from right to left display string
  if (acl) > 0:
    time.sleep(.15)
    while index < length:
      for number in lookuptable[str[index]]:
        oe.simpleWriteByte(~number)
        index = index + 1
        #turn off leds for a short time to
        #account for letter spacing
```

```
        oe.simpleWriteByte(0xff)
        time.sleep(.0015)
    #reset string
    index = 0
```

A link to the code in its entirety is shown below. Have fun creating images and messages with your own magic wand.

LSM303 Magnetometer

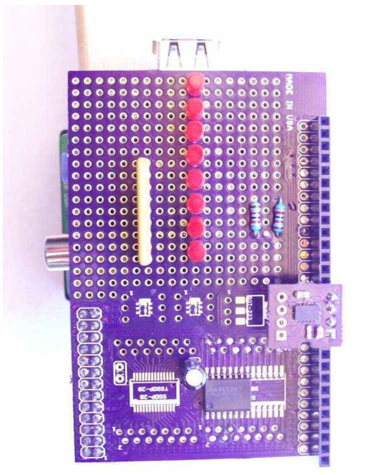


When creating your own magic wand you may come across various issues with certain wand designs. If the wand changes direction at any given time, you will encounter a rapid deceleration. This causes the static and dynamic accelerometer readings to clash, resulting in unwanted input data. If you are a mathematician or are extremely knowledgeable about advance physics concepts, this may be a fun project for you. But for everyone else, you may want to use the LSM303 magnetometer. The code has a few differences, but once you figure out the proper magnetic readings it is very easy. The magnetometer reads the Earth's magnetic field so readings will be different depending on direction and location.

Useful links

OpenElectrons.com full magic wand project kit and program:
http://www.openelectrons.com/pages/63

OpenElectrons.com SmartUPS:
http://www.openelectrons.com/pages/33



VISIT US ON

WORLD'S MOST VERSATILE CIRCUIT BOARD HOLDERS

Model 209
VACUUM BASE PV JR.

Model 201
PV JR.

Model 207
VISE BUDDY JR.

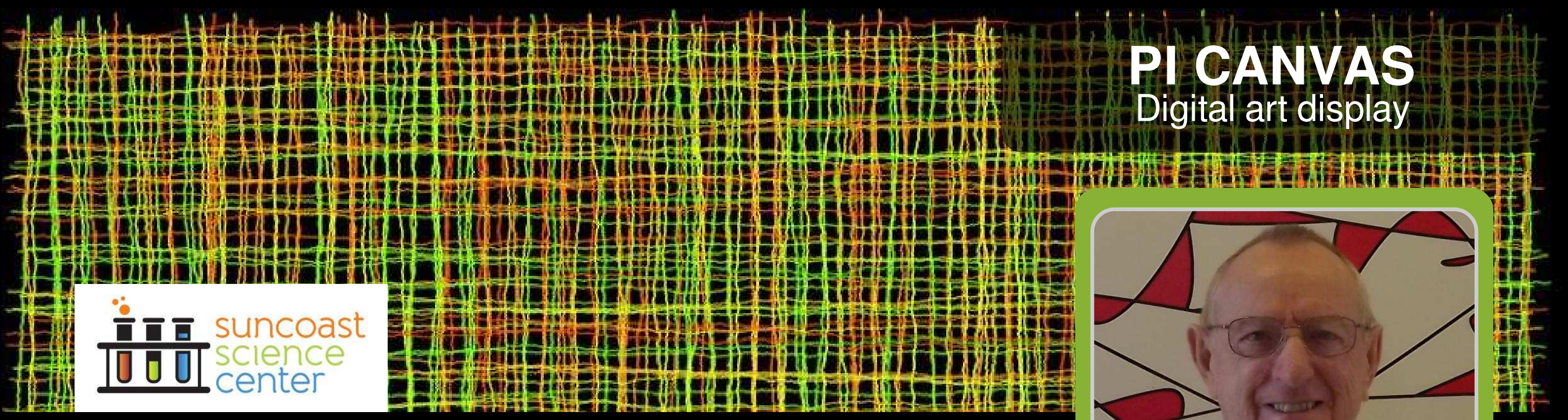
- Work-holding tools for electronics projects
- Circuit board holders make soldering easy & fun
- Versatile hobby vises for any project

PANAVISE®
Innovative Holding Solutions

www.panavise.com

PIMO RONI

PANAVISE IS AVAILABLE AT
http://shop.pimoroni.com



John Mahorney

Guest Writer

How to display dynamic art using a Raspberry Pi

SKILL LEVEL : BEGINNER

In the past digital artists have been limited to static images (prints) when creating wall art. The Pi Canvas allows digital art to be created and displayed on a wall just like a framed print. However, the art is now dynamic which opens up new creative opportunities.

In this article I will describe how you can make a Pi Canvas using your Raspberry Pi to produce a display of dynamic art - art which can optionally interact with viewers.

What is Pi Canvas?

Pi Canvas is a Raspberry Pi mounted on an HDTV that has a USB connector to power the Raspberry Pi. The Pi Canvas has no keyboard, no mouse, nor does it respond to the TV remote control. It is simple to use. Just hang it on the wall, plug it in, press the power button and let it do its thing.

The Pi Canvas can operate 24/7 or be powered on and off as required. It can also be made to interact with its environment through electronic sensors (e.g. ultrasonic or infrared) which opens up even more creative opportunities.

How to make a Pi Canvas?

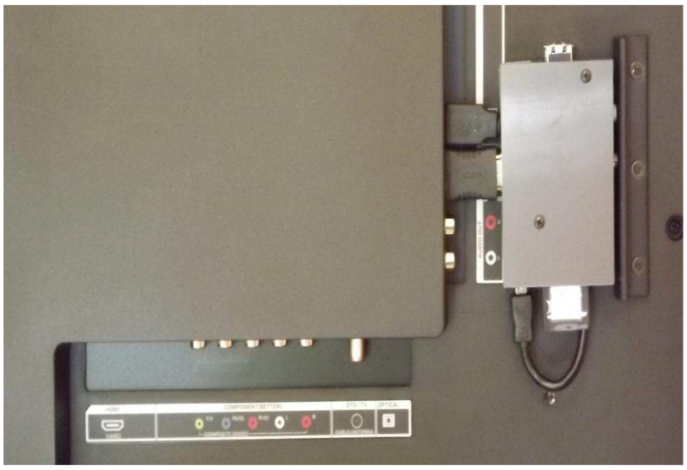
The hardware and software used are all popular, open source and well documented. Basic knowledge of the hardware and software is not covered here as there are many excellent tutorials available online.

The Pi Canvas was developed in the Faulhaber Fab Lab at the Suncoast Science Center in Sarasota, Florida. It supports **STEAM** education (Science + Technology + Engineering + Art + Mathematics) for all ages. For more details, please visit http://www.suncoastscience.org.

Pi Canvas hardware

The Pi Canvas hardware requirements are simple:

• Raspberry Pi (model A or B)
• USB <-> micro USB cable (for power)
• HDMI male to male coupler
• HDTV (e.g. VIZIO model E390-A1)



While any HDTV can be used, the example HDTV is particularly good as it has a USB connector sufficient for powering the Raspberry Pi. It also has a recessed area on the back for attaching the Raspberry Pi, which allows for flush wall mounting. Finally it has a narrow, clean bezel which makes a nice frame.

Pi Canvas software

Just like the hardware, the Pi Canvas software requirements are simple and familiar to many:

• Latest Raspbian OS
• Chromium web browser
• HTML5 Canvas element
• JavaScript

Chromium has a kiosk mode for full screen operation with no browser decoration or controls. The HTML5 Canvas element offers good graphics capabilities for art and is fast. Finally JavaScript is an easy but capable language for browsers. This software combination runs very well on a Raspberry Pi.

Example JavaScript tutorials can be found at http://www.w3schools.com/js/. You should also visit http://www.html5canvastutorials.com.

Pi Canvas configuration

In addition to the normal Raspberry Pi setup, the following steps are required to make a Pi Canvas with the listed hardware and software. You will need separate power for the Raspberry Pi because the HDTV does not supply adequate power for this step.

Enter the following on the command line:

```
sudo apt-get update
sudo raspi-config
```

When the Raspberry Pi Software Configuration Tool appears, enable the option to boot to desktop.

Next we will install the `chromium` package plus some other utilities. The `unclutter` package removes the mouse cursor from the screen after some inactivity.

```
sudo apt-get install chromium
```

```
sudo apt-get install unclutter
sudo apt-get install x11-xserver-utils
```

Your dynamic artwork is a web page. On the next page there is some example artwork code. Enter this code into a text editor and save it as `sample.html`. Put this sample file into the `/home/pi/` folder.

When the Raspberry Pi is powered on, we want to automatically start Chromium with the artwork running. On the command line enter:

```
sudo nano /etc/xdg/lxsession/LXDE/autostart
```

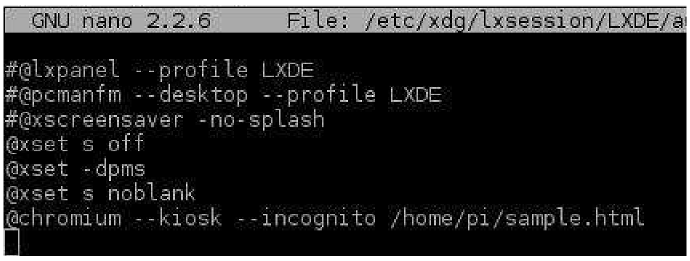
Comment out the following lines by prefixing them with a #:

```
#@lxpanel --profile LXDE
#@pcmanfm --desktop --profile LXDE
#@xscreensaver -no-splash
```

Then add the following lines:

```
@xset s off
@xset -dpms
@xset s noblank
@chromium --kiosk --incognito /home/pi/
  sample.html
```

In nano you should see the following:



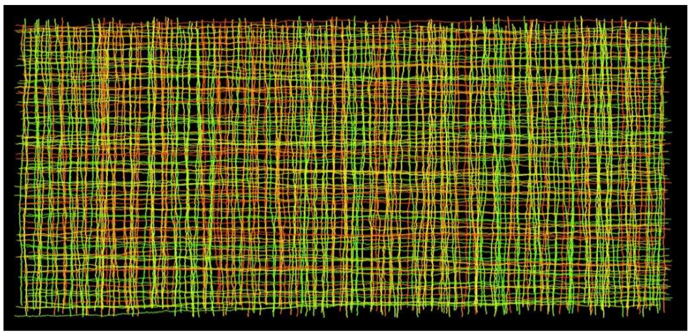
```
  GNU nano 2.2.6        File: /etc/xdg/lxsession/LXDE/a

#@lxpanel --profile LXDE
#@pcmanfm --desktop --profile LXDE
#@xscreensaver -no-splash
@xset s off
@xset -dpms
@xset s noblank
@chromium --kiosk --incognito /home/pi/sample.html
```

To save your changes press `<Ctrl>+X`, then `Y` then `<Enter>`.

When you restart your Raspberry Pi, the sample artwork should automatically appear. To open a second login press `<Ctrl>+<Alt>+F2`. To return back to the art display press `<Ctrl>+<Alt>+F7`.

Pi Canvas artwork

The Suncoast Science Center has a display area for showcasing inventions of all kinds created in the Fab Lab. Here is a screen shot of a Pi Canvas on display in the Fab Lab display area. The artwork is titled *"Loose Weave Digital Fabric"*. In April it won an award at the Art Center Sarasota “One World” exhibition.



The artwork creates a new digital fabric every ten minutes. Drawing the digital fabric is an important visual aspect of the artwork and takes about five minutes.

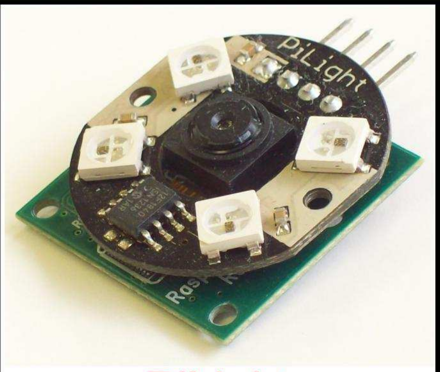
The fabric colour is random and the thread colours are random within +/- 45 degrees of the fabric colour on the HSLA (Hue - Saturation - Lightness - Alpha) colour wheel.

Pi Canvas sample artwork code

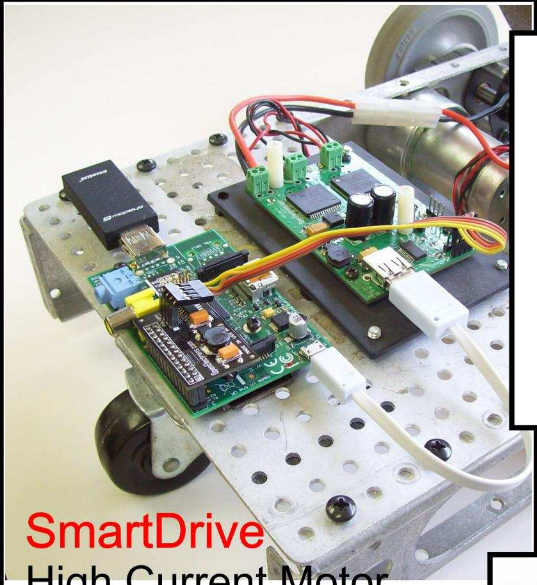
Here is a sample artwork file which draws a "starburst" approximately every 10 seconds. The hues remain within 30 degrees of the initial, random hue on the HSLA colour wheel.

What happens if you change the values of dhue or tension?

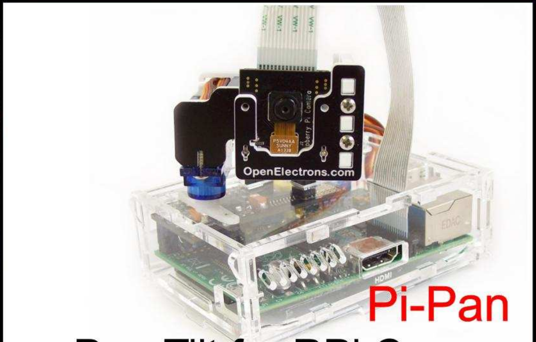
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Starburst</title>
<script type="application/javascript">
var interval = 10;
var canvasWidth = 800;
var canvasHeight = 600;
var x = 0;
var y = 0;
var count = 0;
var hold = 1000;
var hue = Math.random() * 360;
var dhue = 30;
var tension = 1;
var fiberHue = hue + Math.sin(Math.
 random() * Math.PI * 2) * dhue;
var fiberColor  = "hsla(" + fiberHue +
 ", 100%, 50%, 1)";

setInterval(eachTick, interval);

function initialize() {
  var cc=document.getElementById(
   "canvas1" ).getContext("2d");
  cc.save();
  cc.translate(400, 300);
  hue = Math.random() * 360;
}

function eachTick() {
  count = count + 1;
  if(count > 0 && count <= 360){
    draw();
  }
  if(count > 465){
    var cc=document.getElementById("
     canvas1").getContext("2d");
    cc.restore();
    cc.clearRect(0,0,canvasWidth,
     canvasHeight);
    initialize();
    count = 0;
  }
}

function draw() {
  var cc=document.getElementById(
   "canvas1").getContext("2d");
  fiberHue = hue + Math.sin(Math.random()
   * Math.PI * 2) * dhue;
  fiberColor  = "hsla(" + fiberHue +
   ", 100%, 50%, 1)";
  cc.rotate(Math.random() * Math.PI*2);

  for(var j=0; j < 200; j++){
    x = x + Math.random() * 2;
    y = y +  Math.sin(Math.random() *
     Math.PI * 2) * tension;
    cc.fillStyle = fiberColor;
    cc.fillRect(x,y,2,1);
  }

  x = 0;
  y = 0;
}
</script>
</head>

<body onload="initialize()" style=
 "margin:0px; border:0px; padding:0px;
 background-color:#000000;
 overflow:hidden;">
<canvas id="canvas1" width="800"
 height="600"/>
</body>
</html>
```



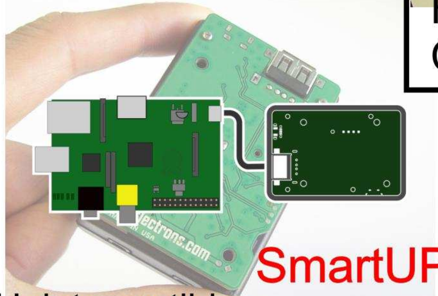
PiLight
Light for RPi Camera
(Infrared and Regular)



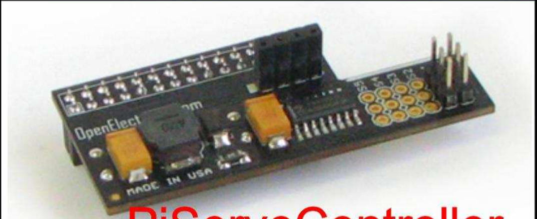
SmartDrive
High Current Motor
Controller for RPi



Pi-Pan
a Pan-Tilt for RPi Camera



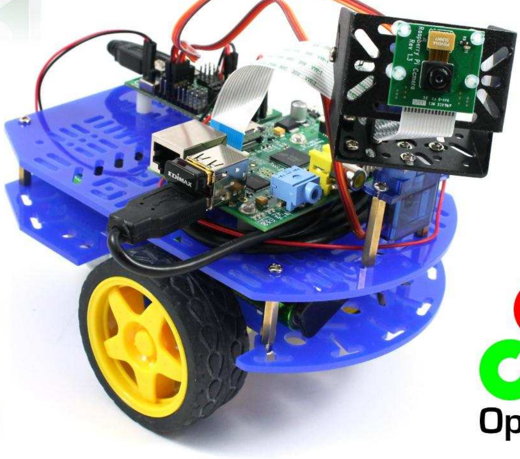
SmartUPS
Uninterruptible power supply
for Raspberry Pi



PiServoController
6 Channel Servo Controller
for Raspberry Pi

OpenElectrons.com

DAWN ROBOTICS

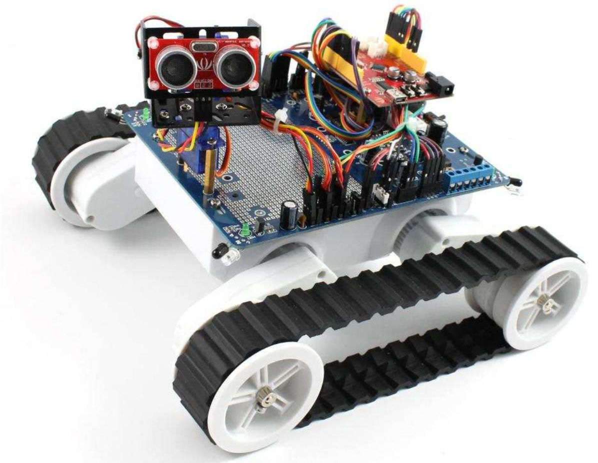


python powered

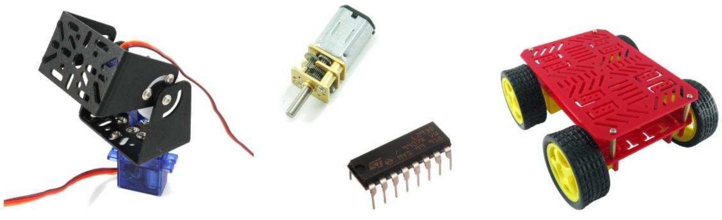
OpenCV

Raspberry Pi Camera Robot Kit

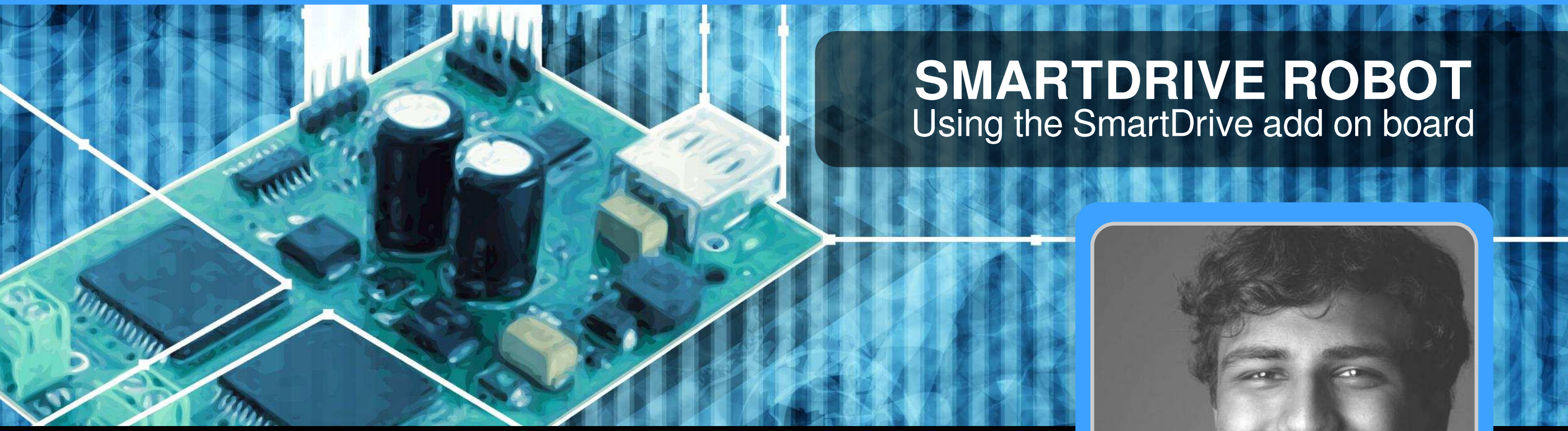
- Drive over WiFi using a Tablet, Smartphone or PC
- Programmable using Python and OpenCV for Computer Vision
- All software Open Source!



More robot kits and parts at
www.dawnrobotics.co.uk



Find lots of robot
and Raspberry Pi tutorials at
blog.dawnrobotics.co.uk



SMARTDRIVE ROBOT
Using the SmartDrive add on board

How to control a robot with a joystick

Rishi Deshpande

Guest Writer

SKILL LEVEL : BEGINNER

The Raspberry Pi is amazing in the sense that it can be used to create programs for almost any purpose. I've seen people create media centres, cloud storage devices, weather stations, video game emulators and various other implementations on this little computer.

However, I wanted to start off with something that didn't require lots of software maintenance and decided upon creating a remote-controlled robot. The hardware specifics of this robot will vary from user to user, therefore I want to focus on the coding required to get the robot to move.

I use a joystick to control my robot. To control the motors I use a SmartDrive controller. This is a motor driver that allows you to control up to two high current motors with the Raspberry Pi.

When starting this program, it is important to import the following modules:

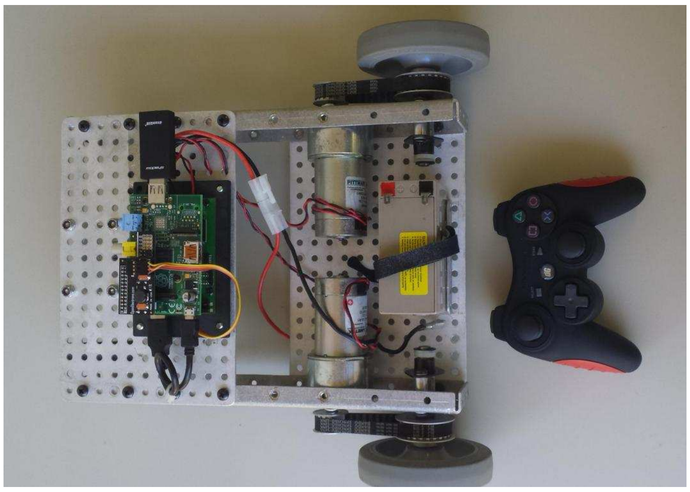
```
import pygame
import sys, os
from SmartDrive import SmartDrive
```

The `pygame` module is used to create a joystick object, the `sys` module is to be able to quit the program when prompted and the `os` module is to read the string environment.

This is the code to create the joystick object:

```
try:
  j = pygame.joystick.Joystick(0)
  j.init()
  print 'Enabled joystick:'+ j.get_name()
except pygame.error:
  print 'no joystick found.'
```

A `try-except` block is recommended because it is important to be able to catch the exception of a `pygame.error` for debugging purposes.



When creating the code to actually move the robot, it is helpful to create a separate function for that purpose.

The function is as follows:

```
def move(motor, speed):
  direction = 1
  if(speed < 0):
    direction = 0
    speed = speed * -1
  if speed > 100:
    speed = 100
  SmartDrive.SmartDrive_Run_Unlimited(
   motor, direction, speed)
```

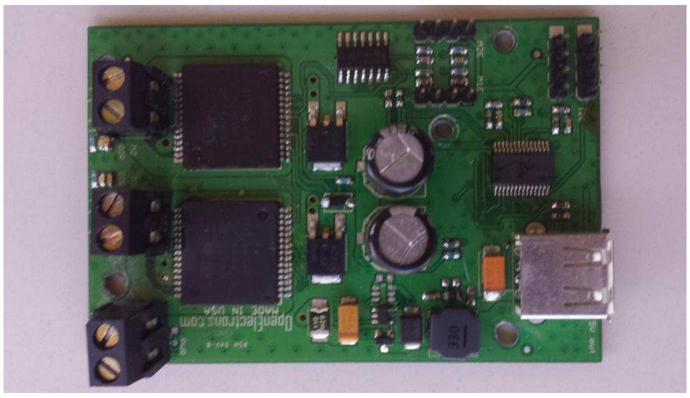
When the `direction` is 0, the motor will run backwards and when the `direction` is 1, the motor will run forwards. The SmartDrive speed ranges from 0-100 therefore it is important to keep within this scale. This function can now be called whenever prompted.

When starting on the main loop of the program, please note that scaling for the joystick axis is required. The `get_axis` function from pygame ranges from -1 to 1, with 0 being centered. Therefore we need to scale by 100 to be able to use the axis values for the speed parameter in the above `move` function. The main loop code is as follows:

```
while True:
  pygame.event.get()
  x1 = 100 * j.get_axis(0)
  y1 = 100 * j.get_axis(1)
  if j.get_button(1):
    move(1, 0)
    move(2, 0)
    sys.exit(1)
  lMotor = x1 + y1
  rMotor = x1 - y1
  move(1, lMotor)
  move(2, -rMotor)
```

The `get_button` function is used to quit the program and stop the robot from running.

I have listed all of the electrical parts, motors and other hardware that I have used for this project. I would also recommend checking out your local scrapyard for any motors that you would like to use instead of buying them. My recommendation would be to look for window motors, or electric toy car motors. I decided to use the SmartDrive because it is capable of supporting up to 300W per motor, though I only used motors that ran at 22W each! There is definitely more headroom to use the SmartDrive for more serious projects.



As mentioned previously, the SmartDrive allows the Raspberry Pi to control up to two high current motors. The SmartDrive is controlled by the Raspberry Pi using the I^{2}C interface.

The programming interface (API) is coded in Python and contains various different functions that allow you to control the motors in a variety of different ways. Two motors can be connected to the SmartDrive the two black screw terminals labeled M1 and M2. (The third black screw terminal is for power.) It is also possible to program the SmartDrive with the C language.

There are functions that allow you to run the motor in terms of degrees turned, time run in seconds and even for a certain number of rotations. It also supports rotary encoders. A nice bonus feature of the SmartDrive is that it is capable of providing a 5V output which can be used to power the Raspberry Pi without using another power source.

Parts List:
1. SmartDrive:
http://www.openelectrons.com/pages/34
2. Tempest TR1.3-12 Battery (12V):
http://www.tempestbatteries.com/html/tr1.3-12.html
3. DreamGrear Shadow USB wireless joystick:
http://www.dreamgear.net/shadow-6-wireless-controller-for-ps3-1.html
4. 2x Pittman GM9234E765-R1 motors:
http://www.gearseds.com/competition_motor.html

Sebastian Duell
Guest Writer

Homebrewing with the Raspberry Pi

SKILL LEVEL : ADVANCED

Introduction

When you are into homebrewing, you are faced with different problems. Brewing equipment, preferably, should be low priced. When doing all-grain brewing, there is also the need for a system that can control the temperature of the mash at different points and at different times.

There are professional brewing controllers available, but these are very expensive. So the idea was to build a cheap brewing-controller mainly from standard components. The controller should have a graphic display, a web interface for configuration and a recipe management system to make brewing different sorts of beer easier. The Raspberry Pi seemed to be ideal.

Brewing beer

Beer is made from malt, hops, water and yeast. To get a beer out of these ingredients, several processing steps are needed. These steps are in general:

• Mashing
• Lautering
• Hop boiling
• Fermentation

The first step in brewing is the mashing. During this step the starch of the malt is converted to sugar and extracted from the malt so the wort is obtained. (Wort is the liquid extracted from the mashing process during the brewing of beer.)

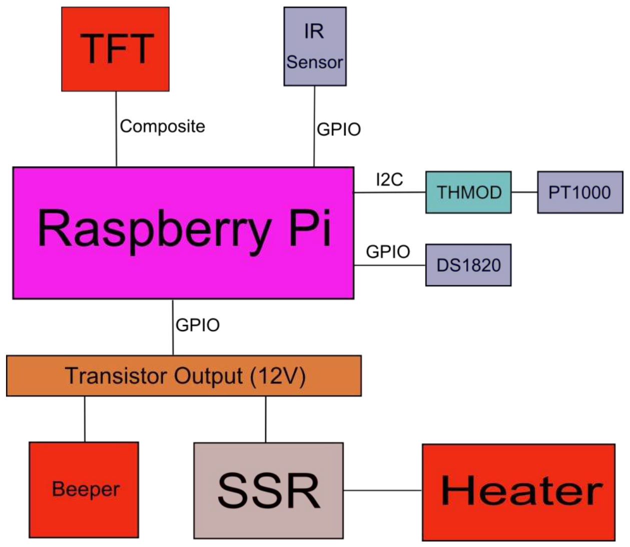
The conversion of the starch to sugar is done by the enzymes contained in the malt. To optimise the conversion to sugar by the enzymes, multiple resting periods at different temperatures are needed. These temperatures have to be held for specific times. That's the point where MashBerry is used.

After mashing, the malt is separated from the wort in a process called lautering. It is basically a kind of filtering. After the lautering, the wort is boiled with hops. In this step, the beer gets hop flavours and bitterness.

Additionally, some chemical processes take place while boiling. These processes, for example, will clean the wort from the proteins. After boiling, the wort is cooled down and yeast is added to ferment the beer (in this step, the yeast is producing the alcohol).

After fermentation the beer is bottled and aged for several weeks or months, depending on the type of beer. Then it's ready to drink.

Hardware



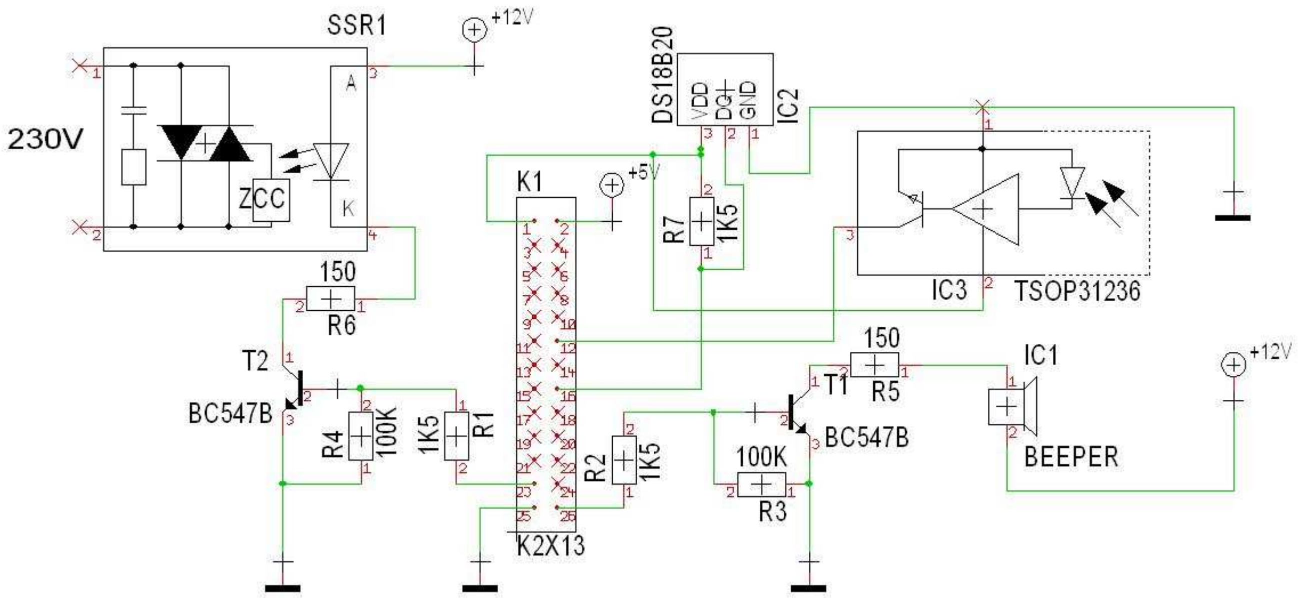
To use a Raspberry Pi as a brewing controller, two main interfaces had to be added. The first interface is a temperature sensor for measuring the temperature of the mash. There are two possibilities for temperature sensors: a cheap DS1820 sensor or a more accurate PT1000 sensor with a Hygrosens I^{2}C converter module.

The Hygrosens module (THMOD-I2C-R2) is a small module with an I^{2}C interface that directly converts the value read from a PT1000 sensor to a usable temperature. To use this converter an I^{2}C level shifter between 5V and 3V3 is needed to connect it to the Raspberry Pi.

As an alternative, a DS1820 sensor can be used. There are sensors with the appropriate metal housing and temperature range (>100°C) available. These sensors can be connected to the Raspberry Pi using only a single resistor.

To control the electric heater of the mash container, a solid state relay (SSR) is used. The SSR is driven by a simple transistor circuit connected to the GPIO of the Raspberry Pi. Such a circuit also drives the piezo beeper. The SSR is housed in an external box including power plugs and filters.

For visualization a 3" TFT display is connected to the Raspberry Pi's composite video port. An IR receiver can be connected to the GPIO to enable MashBerry to receive IR codes from a common IR remote control.



Software

The MashBerry software is built using the Qt framework. The heart of it is the PID controller, which is used to control the temperature of the mash. The PID controller uses the temperatures of the temperature sensor as an input and outputs a power value between 0-100%.

A PID controller is a controller with a feedback loop. Each time the PID algorithm is triggered the temperature error between the set point and the actual temperature is calculated.

With this error, the old output value and the PID parameters (K_p, K_i, K_d), a new output value is calculated using the three values called the proportion, the integral and the derivative (PID).

The PID controller has the advantage (if properly tuned) to reach a nearly constant temperature.

The power value from the PID is then fed into a kernel driver which controls the GPIO in real-time. This driver is basically a hack of the system's timer interrupt, where the switching of the GPIO takes place.

The driver does the switching from 0-100% in two seconds. One percent of output power results in 20ms of on-time for the SSR. At 50Hz mains frequency this is one AC cycle per percent. With this method the power of the heater can be controlled very accurately.

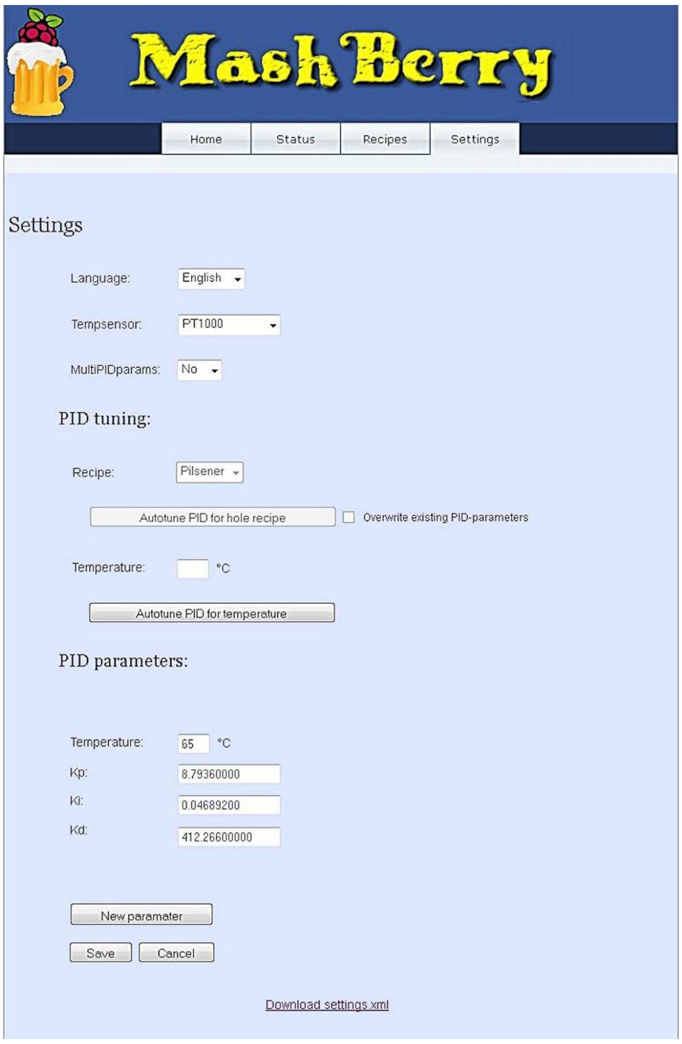
The PID parameters can also be autotuned.

More info about PID controllers can be found at http://en.wikipedia.org/wiki/PID_controller.

The MashBerry application runs on a Linux system without X11, using the embedded version of Qt4 which runs directly on the framebuffer. So very little resources are used.

The software can be downloaded from http://sebastian-duell.de/en/mashberry/downloads.html

There is also a complete SD card image available. It is based on Raspbian and includes the MashBerry application, the framebuffer version of the Qt libraries and a modified Linux kernel, which is capable of driving the SSR relay via the GPIO in real-time.



Settings

Language: English

Tempsensor: PT1000

MultiPIDparams: No

PID tuning:

Recipe: Pilsener

☐ Overwrite existing PID-parameters

Temperature: °C

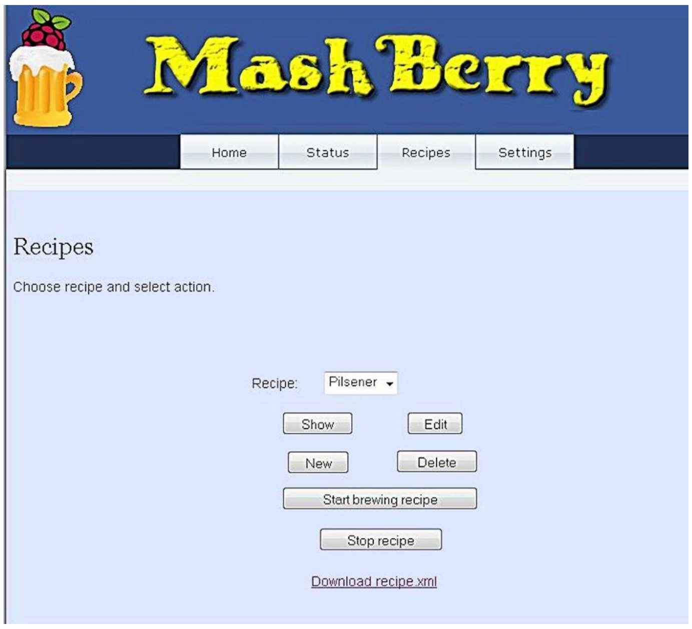
PID parameters:

Temperature: 65 °C

Kp: 8.79360000

Ki: 0.04689200

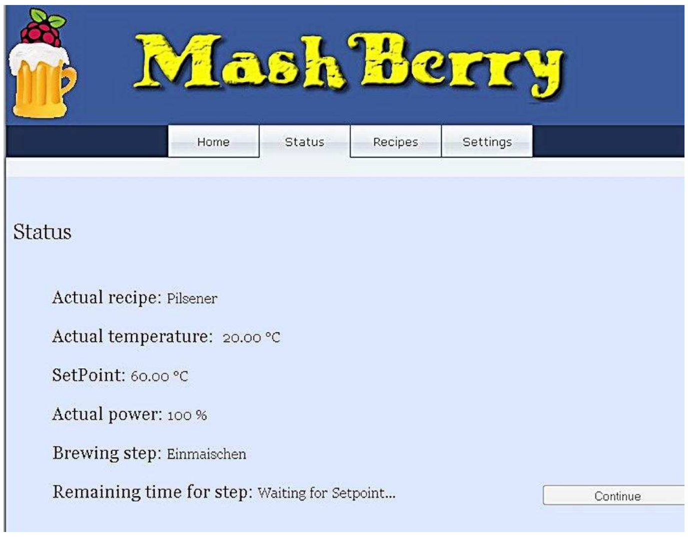
Kd: 412.26600000



Recipes

Choose recipe and select action.

Recipe: Pilsener



Status

Actual recipe: Pilsener

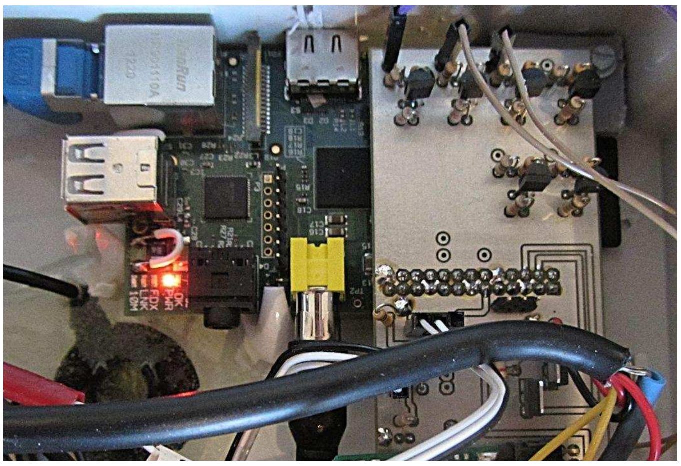
Actual temperature: 20.00 °C

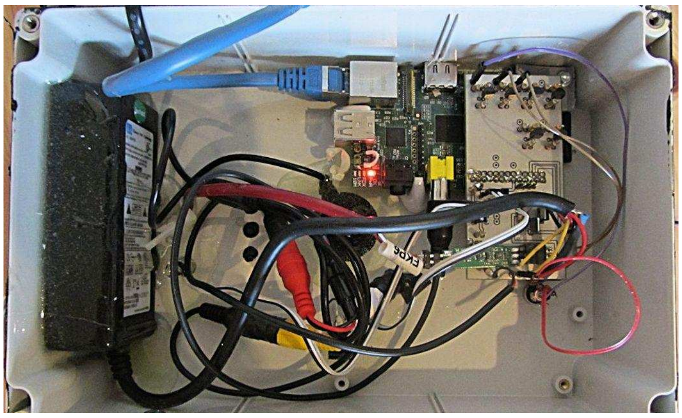
SetPoint: 60.00 °C

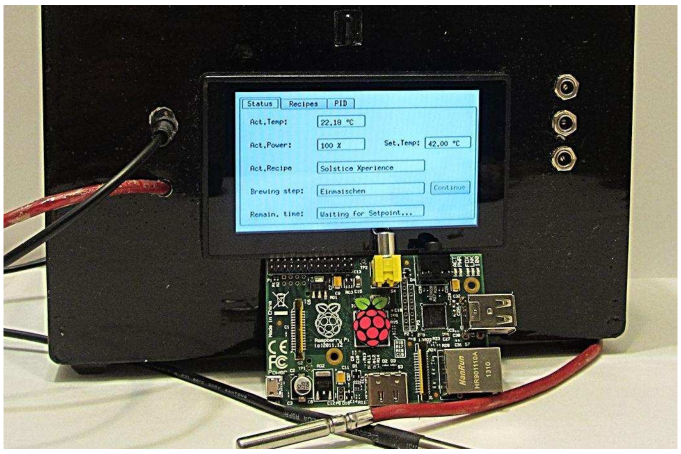
Actual power: 100 %

Brewing step: Einmaischen

Remaining time for step: Waiting for Setpoint...







Act.Temp: 22.18 °C

Act.Power: 100 %

Set.Temp: 42.00 °C

Act.Recipe Solstice Xperience

Brewing step: Einmaischen

Remain. time: Waiting for Setpoint...

Building a MashBerry

To build a MashBerry you need the following parts:

• Raspberry Pi and SD card
• Power supply with 5V and 12V
• Housing
• 2x BC547B transistor
• 2x 150R resistor
• 2x 100K resistor
• 3x 1K5 resistor
• DS18B20 temperature sensor
• 12V Piezo beeper
• 26-pin header for Raspberry Pi connector
• Solid state relay suitable for your power needs

Optional:
• TSOP31236 IR-receiver
• 3" TFT Display

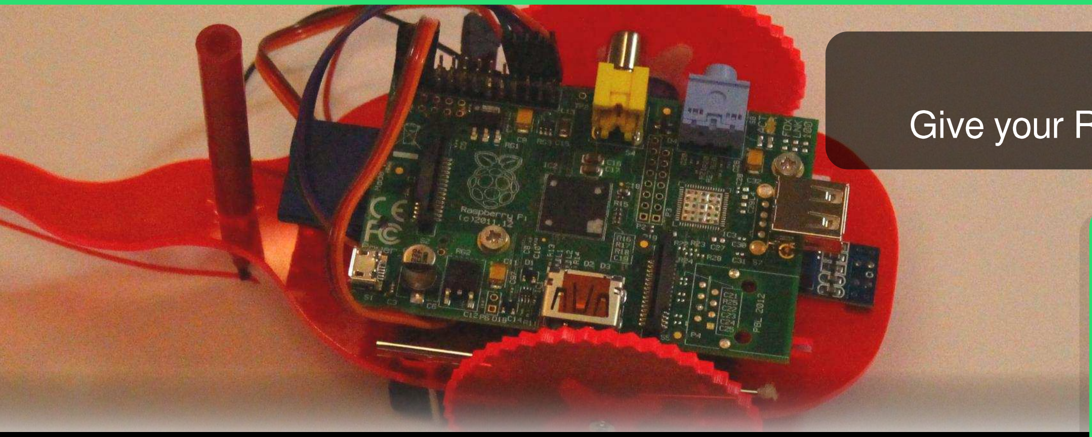
Reference

Helpful information on the beer brewing process can be found at http://www.brewwiki.com.

Brewing is the production of beer through the fermentation of extracts from malted grains - traditionally barley or wheat. Malted grains are made by allowing grains to germinate and then drying them in kilns. The malting process develops enzymes necessary for converting complex starches into sugars.

Mashing is a step in the brewing process that combines crushed malts with hot water in a mash tun to convert complex starches into simple sugars that are more readily fermented. There are many variations of mashing, but the single infusion mash is easily done with home equipment and suitable for most popular beer styles.

Lautering is a process in brewing beer in which the mash is separated into the clear liquid wort and the residual grain. Lautering usually consists of 3 steps: mashout, recirculation and sparging.



Harry Gee

Guest Writer

Add the power of speech, hearing and vision to your robot - Part 2

SKILL LEVEL : INTERMEDIATE

Introduction

Part 1 of this article in Issue 25 provided an introduction to robotics and gave practical tips for how you can build a fun little robot with your Raspberry Pi. In this second part we will cover some more advanced explorations into Raspberry Pi robotics and demonstrate how the Raspberry Pi can be used to create some impressive robotic behaviours and systems.

Three areas will be introduced that each have immense value for useful robotics. These are text to speech, voice recognition and computer vision. Each of these relate to an aspect of human centric abilities - speech, hearing and vision. We saw in Part 1 how robotics is about making computing real world and also that useful robots can sense, process and then act in the world intelligently. In each of our chosen areas we will introduce the technology and give a simple example of how it can be used to do something interesting in a robotic application.

The first two areas, text-to-speech and voice recognition, are both to do with sound. With a low cost microphone, a speaker and a Raspberry Pi, a world of possibilities opens up and this has been a reason why I've included them in my PiBot project that I have been developing.

A speaker gives the robot the power to talk, make interesting sounds and play music. Adding a microphone adds the ability of voice recognition as well as sensing the robot's acoustic environment.

First let's cover making a Raspberry Pi robot talk using text-to-speech.

Power of Speech

The best text-to-speech (or TTS) solution I have found for the Raspberry Pi is eSpeak. It has a good range of voices and is not too resource intensive (meaning it leaves processing power for other things too!). As well as being lightweight, eSpeak provides a simple and straight-forward command line interface that can be easily integrated into Python, as well as other languages. It even allows us to record straight to a WAV sound file with a simple option in the command line. Best of all, it has a Stephen Hawking-esque sound that gives it a fitting dose of panache. It is also good fun finding out what it mispronounces!

To get started, the best thing to do is to just try installing eSpeak and see if it works "out of the box". If you are not using HDMI audio, do not forget to have an amplified speaker or headphones plugged in to your Raspberry Pi. For help in setting up audio, or for debugging any audio problems you have, please see this great post on Raspberry Pi Spy (http://www.raspberrypi-spy.co.uk/2013/06/raspberry-pi-command-line-audio/).

To install eSpeak, on the command line enter:

```
sudo apt-get install espeak
```

This will install the `espeak` and `espeak-data` packages. Try issuing a command straight to eSpeak:

```
espeak "Hello, can we be friends?"
```

The first time eSpeak runs it will probably have a short delay before speaking, which seems to disappear on subsequent executions. You will also probably get quite a long list of warnings about "ALSA lib", "server sockets" and the "jack server". These are harmless and can be ignored. The important thing is that it speaks to you!

More details of TTS and working with eSpeak can be found at http://espeak.sourceforge.net.

Now that we can give a Raspberry Pi robot the power of speech, what do you think it could be used for? Maybe it could let you know whenever you have new email and read it out aloud. I'm sure you can thinking of several other things.

As a simple code example let's consider our PiBot's speech being triggered every time it bumps into something. Here we have added a hardware switch that gets triggered everytime it bumps into anything. From our PiBot Python library we have exposed this event as a function called `PiBot.isBumped`.

```
from espeak import espeak
import PiBot
import random

def bumpReply():
  if PiBot.isBumped():
    responses = ['Ouch, that hurt!',
     'Watch where you are going!',
     'Ouch, be careful!']
    speak = random.choice(responses)
    espeak.synth(speak)
```

Power of Hearing

One very promising project I have discovered for robotics is called Jasper. This project claims that you can control anything with your voice and their website goes on to explain that, "*Jasper is an open source platform for developing always-on, voice controlled applications*".

For years 'voice control' has been an aspirational technology for the world's most advanced (and expensive) robots and it is remarkable that this free software now makes this achievable with an inexpensive Raspberry Pi robot.

Jasper works by identifying specific spoken words (trigger words) that then can activate an action (e.g. execute a Python function). The spoken words that you want to use as triggers are given to Jasper through a string list. Each Python script that you want to use with Jasper needs to contain a string list called `WORDS`, an `isValid()` function and a `handle()` function. The `isValid()` function relates to words being recognised and the `handle()` function relates to actions that occur.

The `WORDS` string array holds the words that you want to extract from the speech. As an example, let's choose the single word "dance". We will declare it like this:

```
WORDS = [“dance”]
```

We will also want to set the priority this script has over other scripts. The higher the number, the more important and further in front of other scripts with similar words it will be. We will set ours at 10 for now. If there is another script with priority less than 10, and it has “dance” in its `WORDS` array, then our script will be used because it has a higher priority.

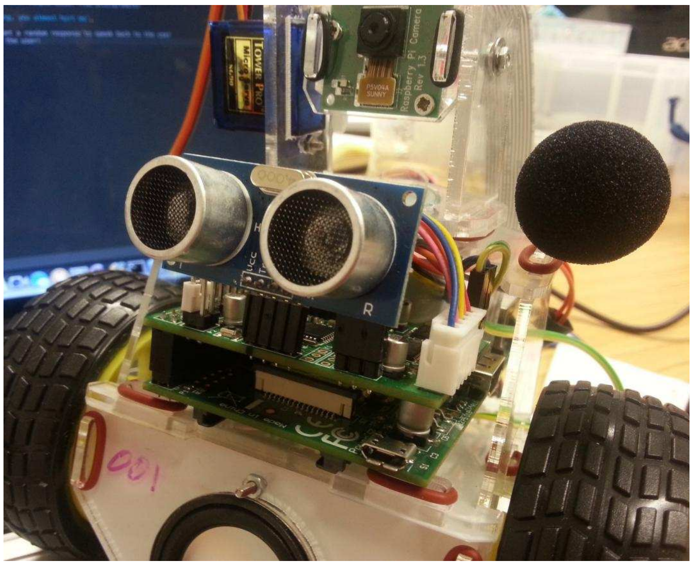
```
PRIOIRTY = 10
```

The `isValid()` function checks the transcripted text input from Jasper's audio recognition engine, to determine if this is the correct script. This will check the input from the user and return `true` if this script is related to the input text.

```
def isValid(text):
  return bool(re.search(r'\bdance\b',
   text, re.IGNORECASE))
```

The `handle()` function will basically perform an action in relation to the input. Here is where Jasper will respond to the input. You will need to pass `text`, `mic` and `profile` as variables which give you more options with Jasper. In this example I get Jasper to acknowledge that it is going to dance and then call a function from our PiBot script to get Jasper dancing!

```
def handle(text, mic, profile):
  mic.say(“Yeah, sure, watch these
   moves.”)
  PiBot.dance
```



Here is the full script:

```
__author__ = 'alexgray'

import PiBot
import re

WORDS = ["dance"]
PRIORITY = 10

def is_valid(text):
  """
  Return True if input relates to "dance"

  Arguments:
  text -- user-input, typically
          transcribed speech
  """

  return bool(re.search(r'\bdance\b',
   text, re.IGNORECASE))

def handle(text, mic, profile):
  """
  Makes PiBot dance

  Arguments:
  text -- user-input, typically
          transcribed speech
  mic -- used to interact with the user
         (for both input and output)
  profile -- contains information related
             to the user (e.g. phone #)
  """

  line = "Watch these moves!"
  mic.say(line)
  PiBot.dance
```

More details on Jasper can be found at http://jasperproject.github.io.

Power of Vision

The third area of advanced Raspberry Pi robotics is computer vision. With the Raspberry Pi’s HD camera module you now have the power to capture visual data. Recently the excellent open source computer vision library OpenCV was implemeted on the Raspberry Pi. This now gives great processing capabilities for analysing visual data and opens up a world of possibilities and useful applications. Face recognition, blob tracking, motion detection and gesture mapping are all possible using OpenCV on the Raspberry Pi. It is of course very exciting to implement these things on a robot. First of all though we will need to install OpenCV on our Raspberry Pi.

We followed a guide from Adafruit to get OpenCV working with our project. You can read it at https://learn.adafruit.com/raspberry-pi-face-recognition-treasure-box. This also contains links to the image capture, training and configuration scripts mentioned below.

In order to be able to recognise a face we need a number of pictures of that person. We can do that with the Python script `capture-positives.py`. This accesses the Raspberry Pi camera so needs to run as root:

```
sudo python capture-positives.py
```

Multiple images of the same face should be taken from different angles. We use these images to train the face recognition model. This will take some minutes to finish. Enter:

```
python train.py
```

After this part we will have to adjust the `config.py` script in order to configure our servo motor's movement. The various options are detailed in the above link and will vary depending on your application.

The final Python script that we need initialises the Raspberry Pi camera and performs the facial recognition. This version of the code is adapted from the `box.py` script from Tony Dicola's OpenCV Facial Recognition project on GitHub (https://github.com/tdicola/pi-facerec-box). This script detects a single face and is the code we need to run our Raspberry Pi in face recognition mode.

```
__author__ = 'alexgray adapted from
 original code by Tony Dicola'

##Full source:
https://github.com/tdicola/pi-facerec-box

import cv2
import config
import face

def init_camera():
  camera = config.get_camera()
  return camera

def face_detect(camera):
  ## Get image from camera
  image = camera.read()

  ## Convert image to grayscale
  image = cv2.cvtColor(image,
   cv2.COLOR_RGB2GRAY)

  ## Get coordinates of single face in
      captured image
  ## Coords will mean a face was detected
  result = face.detect_single(image)

  ## If no face return False, else True
  if result is None:
    return False
  else:
    return True

def main():
  ## Initialise the camera
  camera = init_camera()

  while True:
    # If we see a face, is it recognised?
    if face_detect(camera):
      print("Hi there,nice to meet you!")
```

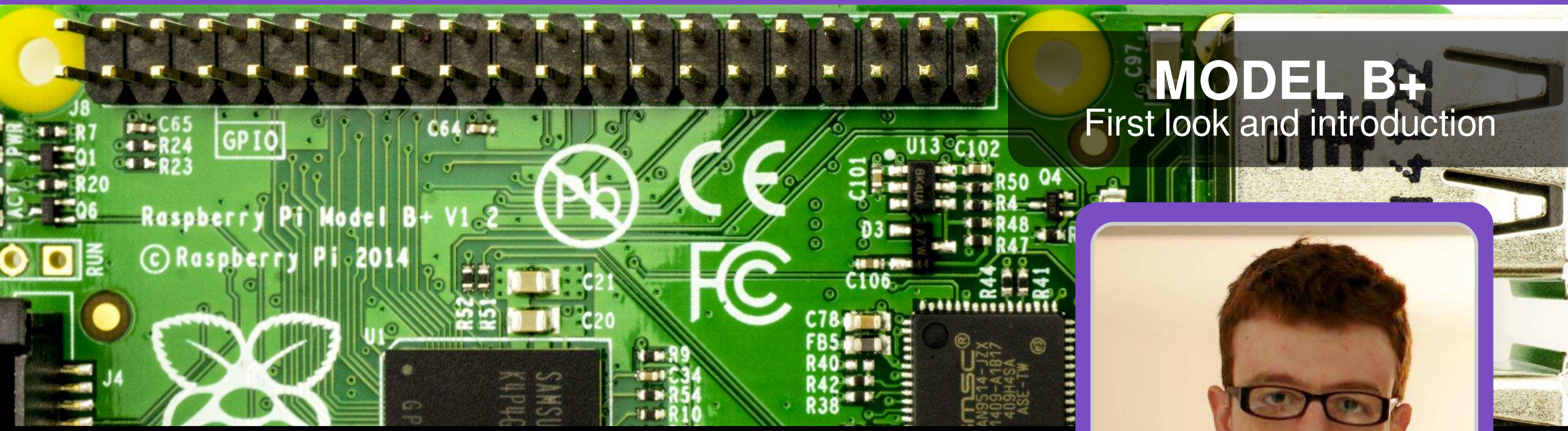
Robotic Future

These topics are quite advanced so if you've managed to follow this article completely then you are doing very well!

There are lots of resources online if you want to explore any of these topics further. In particular a number of detailed articles are found on the PiBot website at http://www.pibot.org/how-to and we will be adding more as our adventures into Raspberry Pi robotics continue!

We are now working on computer vision, voice recognition and text-to-speech for our PiBot robot and all this code will be open source and shared with the community too. I don't know about you but we are excited about making use of the incredible software that is now available on the Raspberry Pi for robotics. Thanks to the Raspberry Pi and its community we can now make a robot that is able to hear you, recognise your face and speak to you as well! Exciting times indeed!

Thanks to Aldi Rina, Alex Gray and Steph Tyszka from PiBot for their contributions to this article.



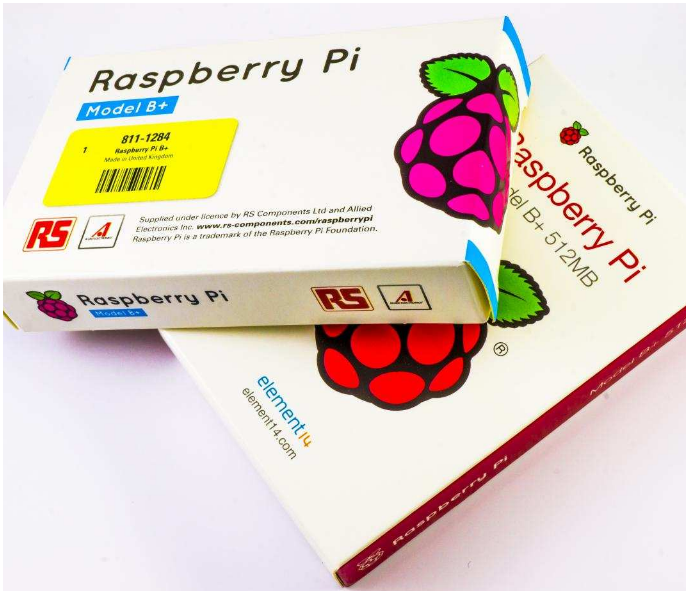
Aaron Shaw

MagPi Writer

A brief introduction to the latest Raspberry Pi hardware release

The first Model B Raspberry Pi was originally launched for sale on the 29th February 2012. In two short years our favourite little computer has racked up some pretty serious sales with over 3 million units sold to date and there are no signs of the rate of sale tailing off anytime soon. The Raspberry Pi is here to stay - that is now a well known fact. Both the Foundation and the community that surrounds the Raspberry Pi have produced some incredible software and hardware and has achieved some pretty amazing things - balloon flights to space, visits to Buckingham Palace and perhaps most importantly (and most relevantly to the Foundation's charitable goals) inspiring a huge number of people to get involved with computing, electronics and STEM.

As you probably already know, in the early hours (UK time) on the 14th July 2014 the Raspberry Pi Foundation announced the latest hardware upgrade - the Raspberry Pi Model B+. This represents the first major hardware change to the Raspberry Pi board since the upgrade of the Model B hardware to 512 MB of RAM in October 2012. There had been some minor changes to various components in the interim period (likely due to pricing and supply issues or similar) as well as the release of both the Camera Module, Pi NoIR and the Compute Module. Additionally, as some of the more eagle eyed of you may have noticed, more recently (around April 2014) the USB hub and LAN chip on the Model B had also been upgraded from a LAN9512 to a LAN9514 chip as well as a general redesign of the PCB. Looking at the documentation that comes in the box with the Raspberry Pi, it looks like the change may have been related to FCC class B device certification. However, the change in LAN/USB chip is likely to have also been a hardware trial of sorts before the more obvious changes were made in the Model B+, which actually make use of the additional functionality present in the upgraded LAN/USB chip.



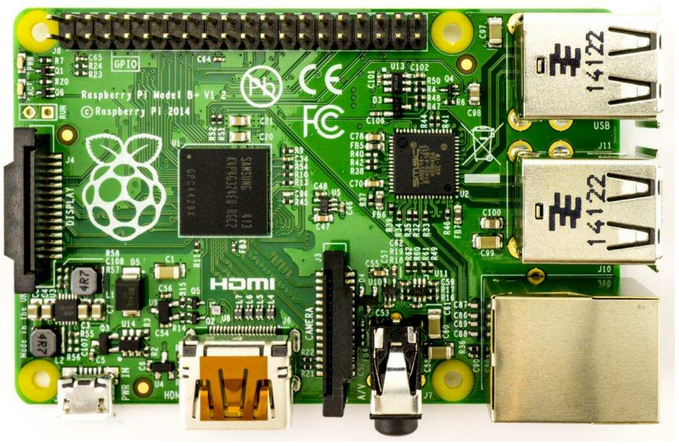
New Raspberry Pi packaging

Where to buy?

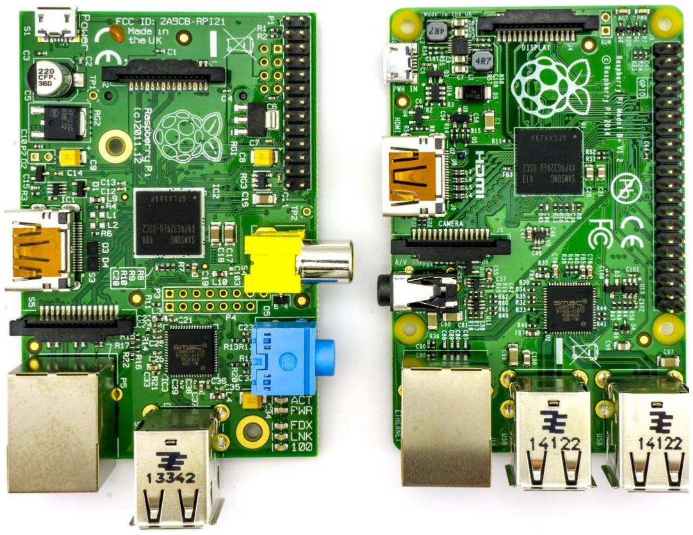
As per usual, the Raspberry Pi Model B+ is available to purchase from the two main distributors - RS Components and Farnell element 14 and their subsidiaries. It is also already in stock, along with a selection of accessorries, at a number of independent retailers across the world.

First impressions

Whether the Model B+ board that you purchase has originated from an RS Components or Farnell batch, the first thing you are likely to notice is the fantastic new packaging designs as can be seen on the picture on the bottom of the last page. My personal preference is the Farnell packaging, however they are both a significant improvement on the original packaging (basically a plain white box) and the product already looks far more professional with that simple step.

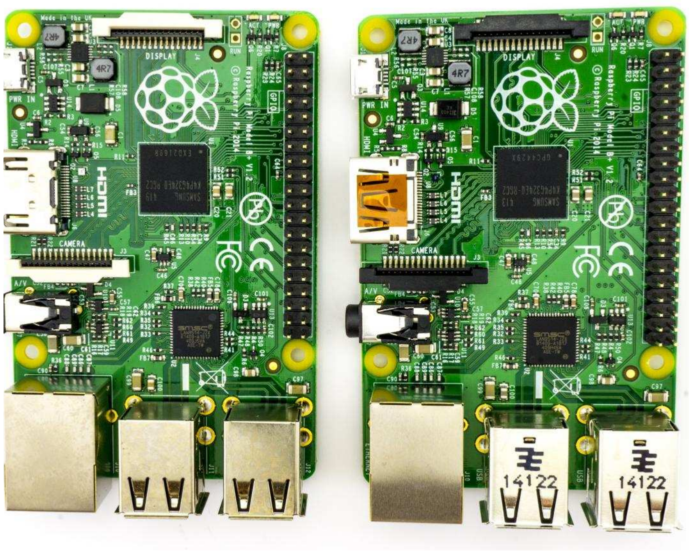


Opening the box and taking a look at the new board the changes are fairly obvious, and this is even more apparent if you look at the picture to the right comparing the new Model B+ with the old Model B. The most important changes, in my opinion, are the inclusion of a 40 way GPIO header instead of 26 on the Model B, four USB ports instead of two and the change to using a microSD card instead of a full size one. As mentioned above, this change to 4 USB ports is possible due to the change to a new LAN/USB chip (LAN9514). This is the little black chip just behind the USB ports. Further changes include the relocation of the 3.5mm jack, removal of the RCA video output jack (the composite video signal has been relocated to the fourth pole on the 3.5mm jack) and the addition of four squarely positioned mounting holes. The new microSD card holder is now of the push-push type so it is held securely in place (when compared to the friction fit of the old one) and also gives a nice "click" when it is correctly inserted!



Comparison between Model B and Model B+ boards

You will probably also notice from the comparison picture that the USB ports no longer hang off the edge as far and are now lined up with the Ethernet port. There has also been a general tidy up of the design and the major ports (excluding the GPIO and DSI connectors) now all lie on just 2 sides of the board. Obviously some



Comparison between RS Components (left) and Farnell element14 (right) boards of the connectors have had to be shifted around slightly to accomodate this. This should allow for some much neater cabling for home cinema and professional installations where neat installation is very important. For people who like neatness and aesthetics, the round edges on the new PCB are probably also a welcome addition! Looking at the top of the board there are now only two LEDs on the top (power and activity), and they have been relocated to the same side as the DSI port. The Ethernet connection and activity lights are still present but are now located inside the Ethernet jack itself which both looks great and saves space.

Looking at the comparison of the Model B+ boards from the different suppliers on the previous page, there are some small differences in component selection (see the USB, HDMI, CSI and DSI ports). I am not sure if this is down to the manufacturers sourcing components of their own or whether this is just a coincidence. In either case, the boards are both made in the Sony factory in Wales and look fantastic.

There are some less noticeable changes "under the hood" with the Model B+ getting an updated power circuit that replaces linear regulators with switched ones in order to reduce power consumption by up to 1 Watt. The audio circuit also now incorporates a dedicated low-noise power supply which should improve the quality somewhat and the USB ports can now be fed with up to 1.2 Amps as well as featuring better overcurrent and hotplug behaviour (plugging and unplugging devices while the Raspberry Pi is running).

Getting started

Hopefully you, like me, are fortunate enough to use an SD card with the Raspberry Pi that is actually a microSD card in a holder. Personally I use the official SD card with the Raspberry Pi logo silk screened on the holder. This means I did not have to purchase a new SD card in order to get started with the Model B+. However due to the new firmware needed, it is not necessarily quite as easy as just swapping the SD card out of the old Model B and into the Model B+. There are a few steps you may need to undertake first. This is especially important if you have not updated your operating system in a while.

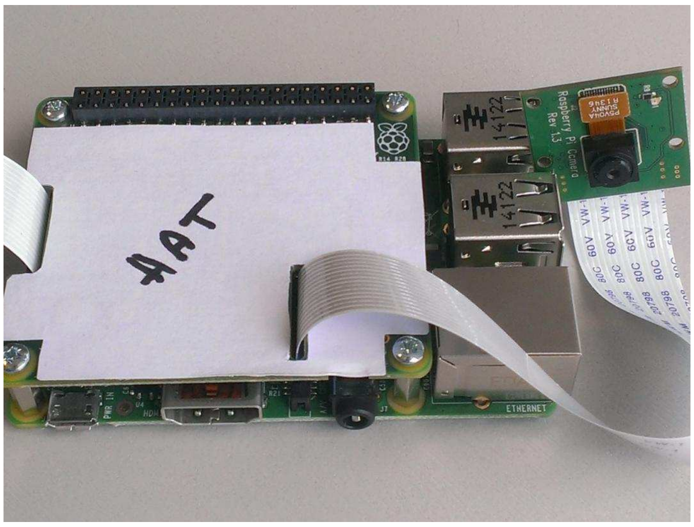
Open an LX Terminal window and type the following to make sure the software on your SD card is up to date:

```
sudo apt-get update
sudo apt-get upgrade
```

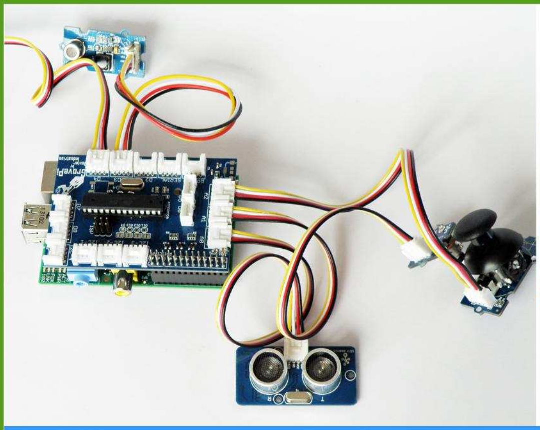
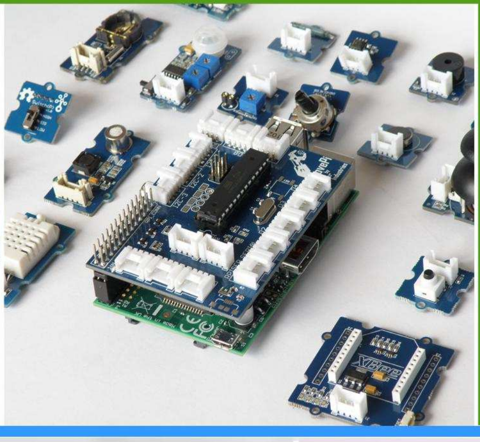
Raspberry Pi HATs

With the update to the Model B+, the Raspberry Pi Foundation also recently announced the specification for what they are calling HATs (Hardware Attached on Top). The idea is to have a more regulated framework for add on boards as well as just having a nice name to use for them (similar to Beaglebone Capes and Arduino Shields). The specification outlines a mechanical shape, as can be seen in the picture below, which makes use of the 40 pin header as well as all four of the mounting holes.

All HATs must also have an EEPROM on board which will contain code that allows the Raspberry Pi to automatically identify the add on board and set up the GPIO pins and Linux drivers necessary to get you up and running quickly - which will be extremely useful in schools and clubs with children and beginners.



Model B+ Raspberry Pi with mechanical sample of HAT board

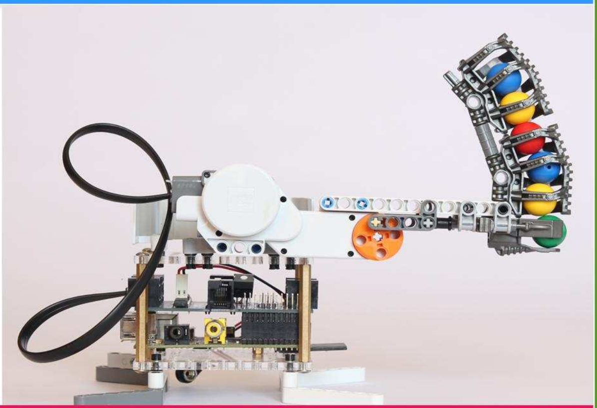
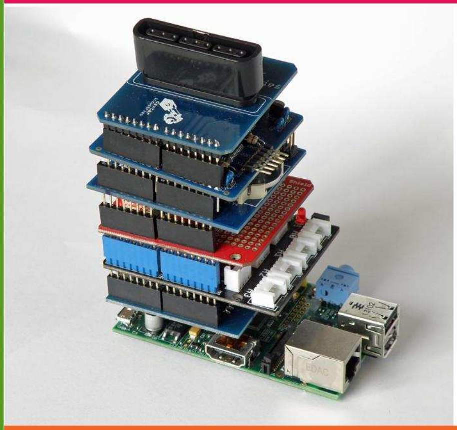
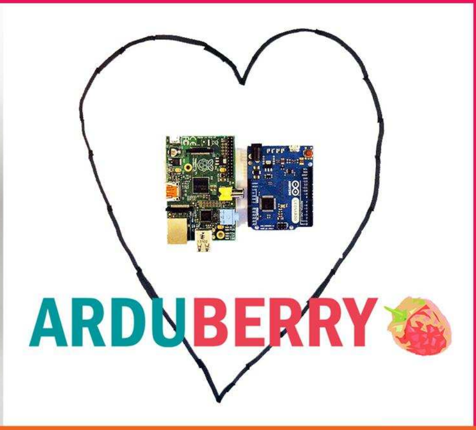



GrovePi

Connect Hundreds of Sensors to your Raspberry Pi

BrickPi

Turn your Raspberry Pi into a LEGO© Robot

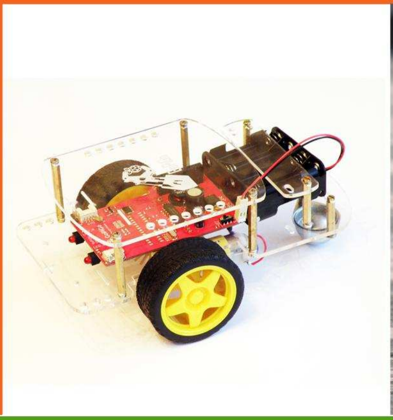
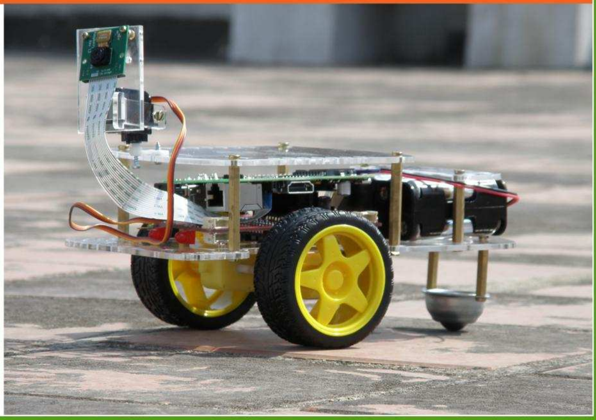




Arduberry

Unite the Raspberry Pi and Arduino

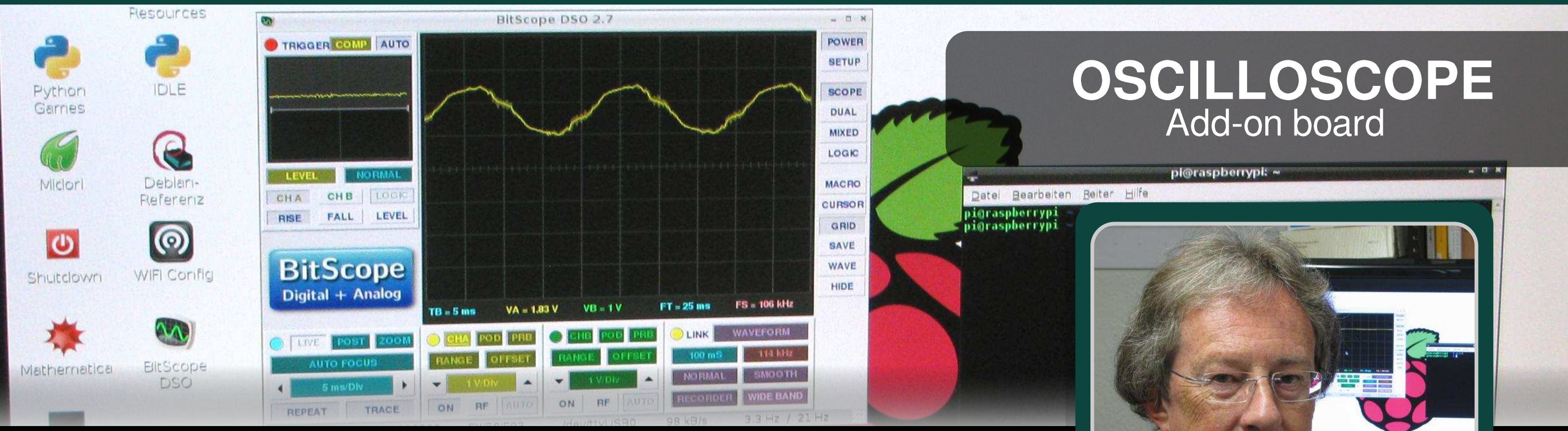
GoPiGo

Turn Your Raspberry Pi into a Robot

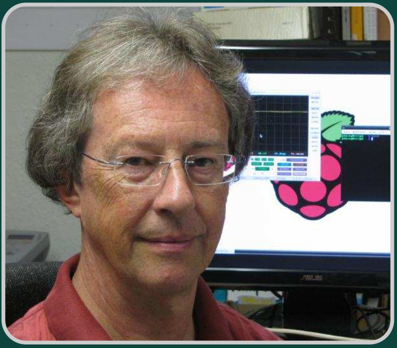
dexterindustries.com

MagPi Readers! Use the code "MagPi14" for a 10% discount in our store.



OSCILLOSCOPE

Add-on board



Karl-Ludwig Butte

Guest Writer

Electronic measurement with BitScope - Part 2

SKILL LEVEL : INTERMEDIATE

In last months article, we equipped the Raspberry Pi with a BitScope Micro add-on board and installed the BitScope digital storage oscilloscope (DSO) software. Now we are ready to delve into the fascinating field of electronic measurement with a fully-fledged DSO.

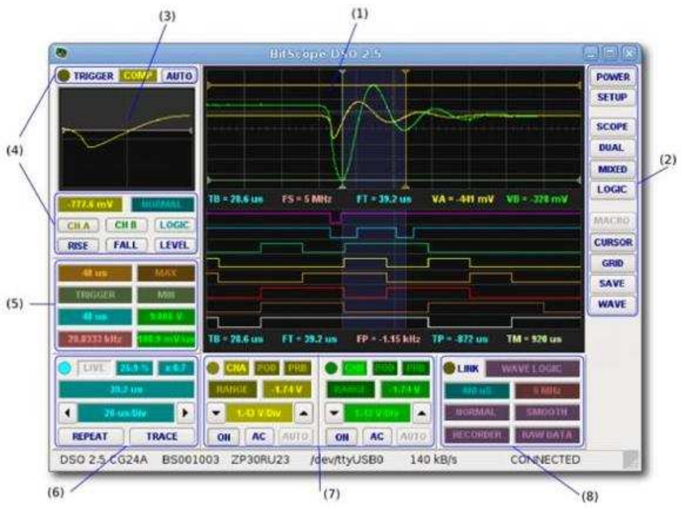


Fig. 1: BitScope DSO software main screen elements (photo courtesy by BitScope Designs)

In Part 1 you learnt that an oscilloscope measures electrical voltages. So let's start by measuring the voltage of the Raspberry Pi itself.

Identify the channel control pad for Channel A (7) and set it to 2V per Div. This means that one square of the y-axis on the main screen represents 2V. With 8 squares stacked up on the Y-axis we are able to measure a maximum of 16V. Now connect the red test lead to pin CHA and the black test lead to pin GND, opposite pin CHA on the BitScope Micro. Fig. 2 shows the pin layout of the BitScope Micro to help you to select the right pins.

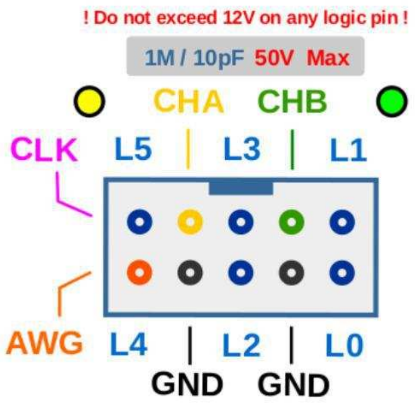


Fig. 2: Pin layout of the BitScope Micro (photo courtesy by BitScope Designs)

Press the top of the black test lead down, so that the metal gripper appears at the opposite side and connect to pin 6 of the GPIO on the Raspberry Pi. Do the same with the red test lead but connect it to pin 2 of the GPIO. Fig. 3 shows the setup.

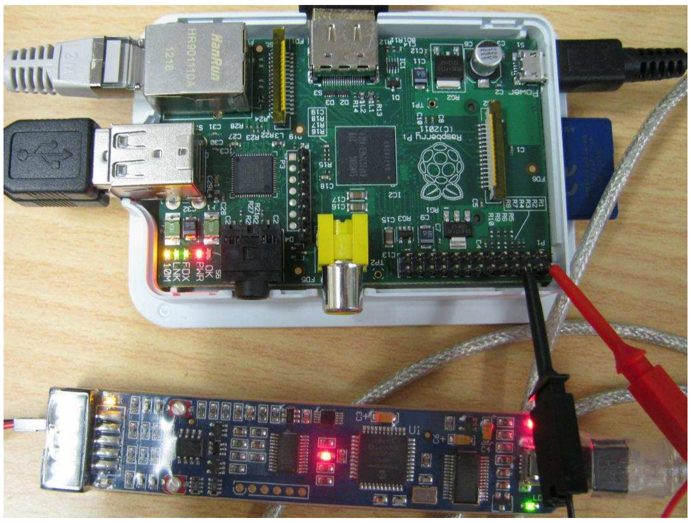


Fig. 3: Measuring +5V on the GPIO pins 2 and 6 of the Raspberry Pi

Look at the main screen of the BitScope DSO (1). The x-axis is the line in the middle and has small vertical lines to sub-divide each square. This is our 0V line. But our yellow beam is in the middle of the third square above the x-axis. Because we know there is a voltage of 5V between pin 2 (the plus pole) and pin 6 (the minus pole), and we set the channel control to 2V per Div, so the horizontal line indicates exactly this voltage. In Fig. 4 I have inserted an extra scale in red on the y-axis to help you interpret the screen.

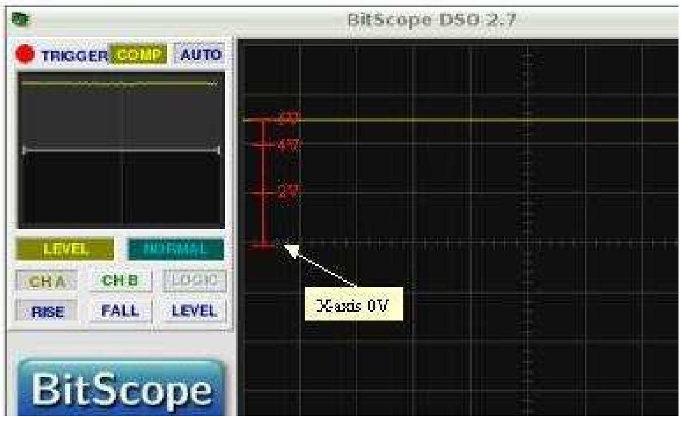


Fig. 4: Measuring +5V between the Rasperry Pi GPIO pins 2 and 6.

You may get a reference measurement with your multimeter, if you like. Could we measure other and higher voltages than 5V? Sure, but what about our input range? We have to change the input range with the channel control for Channel A (7) to an appropriate value. While working with an oscilloscope you should always be aware about the settings of the instrument and the expected voltages in the circuit.

Is this NE555 timer IC still working?

In our next experiment we want to find out if a NE555 timer IC is still functional. This IC is often used when a clock signal is required. For this experiment we need:

- 2x 1K resistors (R1, R2)
- 0.1 µF capacitor (C1)
- 10 nF capacitor (C2)
- NE555 timer IC (IC1)
- small breadboard
- test leads provided with the BitScope Micro

Fig. 5 shows the circuit diagram and Fig.6 shows you how to implement this circuit on a breadboard.

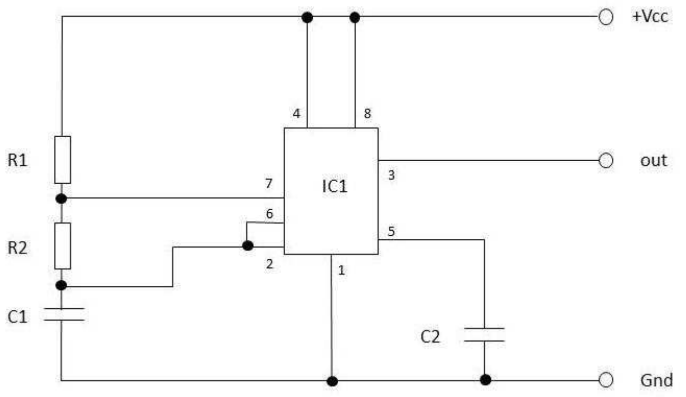


Fig. 5: Circuit diagram of the clock generator

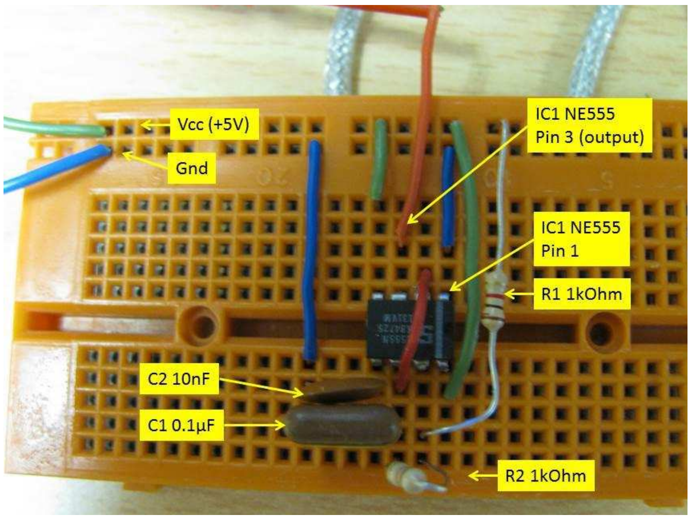


Fig. 6: Implementation on a breadboard

When you have finished building the clock generator on the breadboard, set the time base control (6) to 50 µsec per Div and the channel control pad for Channel A (7) to 2V per Div. Additionally, set the trigger controls (4) to MEAN.

Our Raspberry Pi has enough power to supply the +5V the clock generator needs. Therefore connect a blue test lead from pin 6 of the Raspberry Pi GPIO (Gnd) to the Gnd connection of the breadboard and a green test lead from pin 2 of the Raspberry Pi GPIO (+5V) to the Vcc connection of the breadboard.

Two more test lead connections are needed. Connect pin 3 of the NE555 with CHA of the BitScope Micro and connect Gnd of the breadboard with Gnd of the BitScope Micro, opposite CHA.

Phew! Things can get complicated fast. Look at Fig. 7 and make sure you have all the connections right.

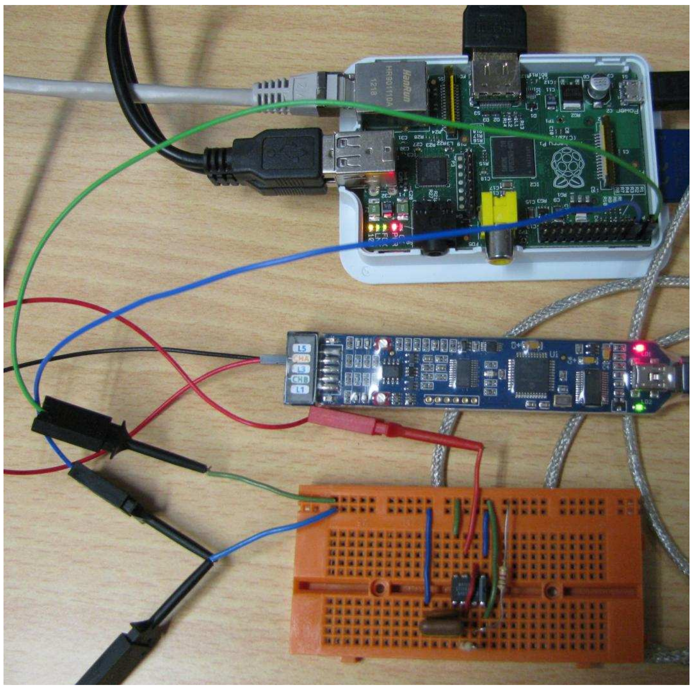


Fig. 7: Connecting the clock generator to the Raspberry Pi and BitScope Micro

Look at the main screen of the BitScope DSO software (1) and you should see a similar square wave, as shown in Fig. 8.

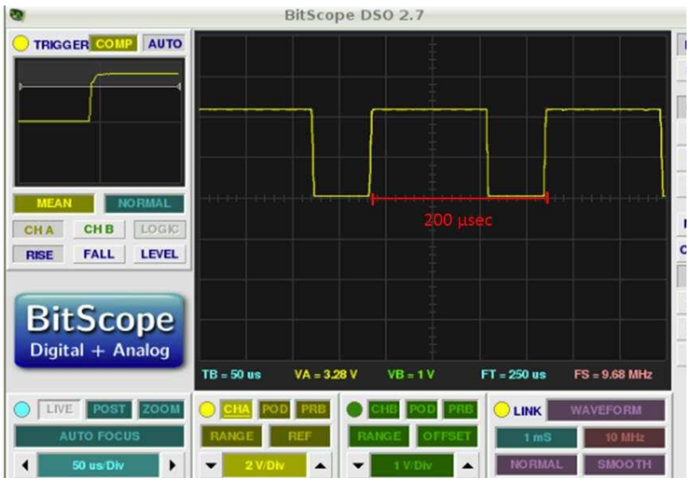


Fig. 8: Output of the clock generator circuit

This square wave proves that our NE555 is still functional and could be used for our next electronics project. If you do not see a square wave, check your circuit for any errors. If all connections between the different parts and the Raspberry Pi are ok and you still don't get a square wave it can be assumed that the NE555, or one of the other components, is defective.

If your clock generator is operational you may try to find out the frequency with which your clock generator is running. For the answer you should look at Fig. 8 (or the measurement on your own screen) and remember what I wrote about measuring frequencies in Part 1 last month. As an aid, I have drawn a line in red on the output diagram showing that the period of the square wave is 200 µsec.

In this second part we measured voltage and frequency. In the next part there will be some more experiments with the clock generator and how to put an oscilloscope to good use.

The BitScope Micro add-on board is available from BitScope Designs in Australia (http://www.bitscope.com), in Germany from BUTTE publishing company (http://www.butte-verlag.de), in the UK from Pimoroni (http://shop.pimoroni.com) and in Switzerland from (http://www.pi-shop.ch). A German translation of this article is available at http://www.butte-verlag.de.

Expand your Pi

Stackable Raspberry Pi expansion boards and accessories

ADC-DAC Pi

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

IO Pi

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

RTC Pi

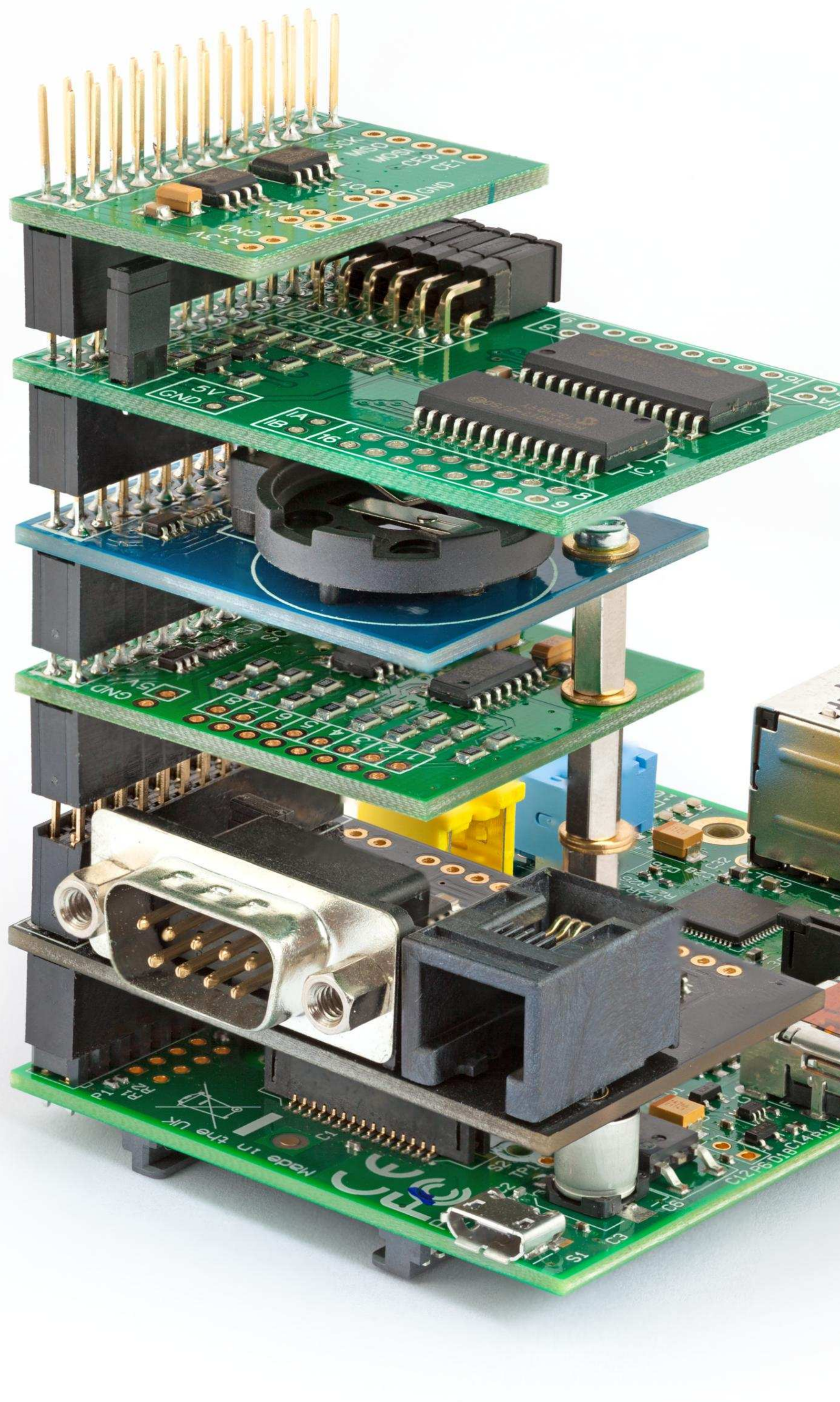
Real-time clock with battery backup and 5V I^{2}C level converter for adding external 5V I^{2}C devices to your Raspberry Pi.

ADC Pi

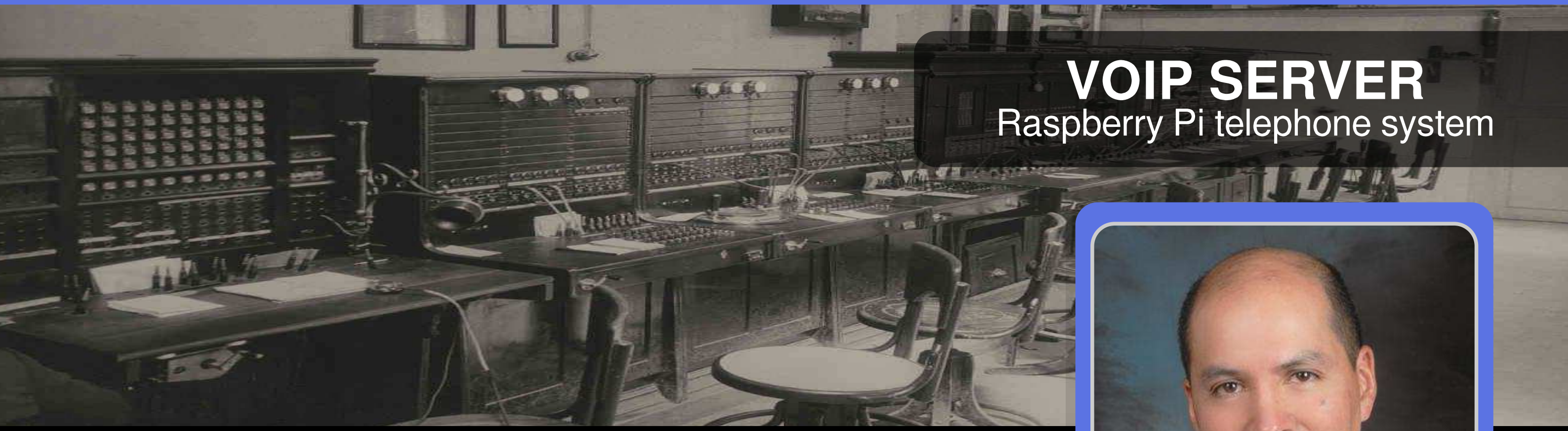
8 channel analogue to digital converter. I^{2}C address selection allows you to add up to 32 analogue channels to your Raspberry Pi.

Com Pi

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



AB electronics UK   www.abelectronics.co.uk



Using Asterisk to implement a low cost telephone system

SKILL LEVEL : INTERMEDIATE

Walberto Abad

Guest Writer

After investigating a number of technology solutions that provide VoIP (Voice-over-Internet Protocol) and IP telephony services, including support for the new trend of UC (Unified Communications) for small businesses, I personally concluded that the Raspberry Pi is able to deliver a totally viable and very low cost solution. When compared to the $100's needed to invest in a dedicated server for a VoIP/UC solution, the cost of a Raspberry Pi and accessories is unmatched.

The Raspberry Pi solution is based on Raspbian running the Asterisk VoIP/UC software. This open source solution provides a high degree of configuration and of course can be used for a multitude of solutions and applications in different areas.

This article demonstrates that VoIP/UC solutions are not high risk and do not require high implementation costs.

Introduction

Telephony has evolved rapidly over the past few decades, migrating from analogue communications to digital communications and IP telephony based on VoIP. This also enables Unified Communications - the integration of real-time communication services such as instant messaging (chat), telephony, data sharing, video conferencing, speech recognition, etc. with non-real-time communication services such as voicemail, email, SMS and fax. UC is not necessarily a single product, but a set of products that provides a consistent, unified, user-interface and user-experience across multiple devices and media-types (http://en.wikipedia.org/wiki/Unified_communications)

VoIP is the transmission of voice over the internet using protocols such as SIP (Session Initiation Protocol) and RTP (Real-time Transport Protocol), among others.

Baseline

To implement a VoIP/UC solution, the system must meet various industry standards plus the network equipment must be able to differentiate and prioritise voice and video applications over other types of data usage.

Basic Components

The hardware and software requirements are simple. You probably just need to download the software.

Hardware:
• Raspberry Pi Model B/B+
• 4 GB SD card (minimum)
• 1A power supply

• Network cable
• Optional SIP phone or SIP adapter (this article uses the Dlink DPH-150SE)

Software:
• Raspbian
• Asterisk communications software
• LinPhone soft phone (supports iOS, Android, Blackberry, Linux, Windows and OSX). You can download this from http://www.linphone.org.

Installation

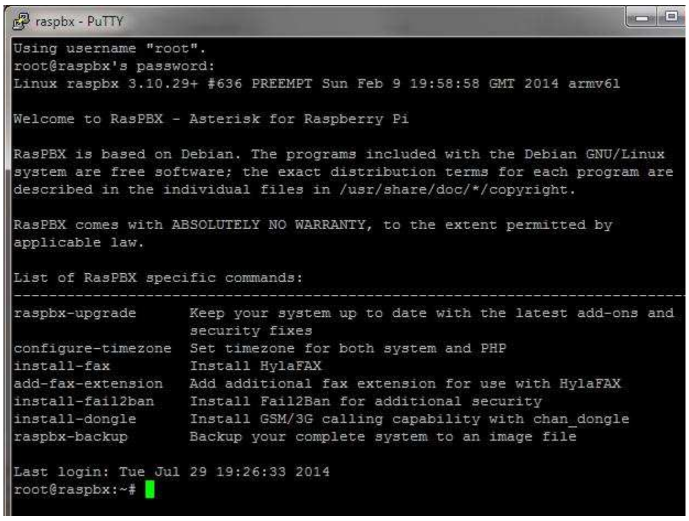
For the initial setup you may need to use a USB keyboard and mouse with the Raspberry Pi, plus a connection to a monitor. Once configured, the Raspberry Pi will run "headless".

The best and easiest way to get the Asterisk software is to download the latest SD card image at http://www.raspberry-asterisk.org/downloads. This contains Raspbian with the Asterisk communication software and FreePBX GUI pre-installed. The image is written to the SD card following the steps at http://www.raspberrypi.org/documentation/installation/installing-images/.

When the system starts, login as `root` with the password `raspberry`. If you wish, you can do this remotely. On Windows install the PuTTY SSH client and connect using `root@raspbx`. On an Apple Mac, simply open the Terminal and enter `ssh root@raspbx.local`. Later you will want to disable root login via SSH as this is a security weakness. Once logged in, the first command you want to run is:

```
raspbx-upgrade
```

This will update all the software to the latest version, including Raspbian and the kernel.



```
Using username "root".
root@raspbx's password:
Linux raspbx 3.10.29+ #636 PREEMPT Sun Feb 9 19:58:58 GMT 2014 armv6l

Welcome to RasPBX - Asterisk for Raspberry Pi

RasPBX is based on Debian. The programs included with the Debian GNU/Linux
system are free software; the exact distribution terms for each program are
described in the individual files in /usr/share/doc/*/copyright.

RasPBX comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

List of RasPBX specific commands:
-------------------------------------------------------------------------------
raspbx-upgrade      Keep your system up to date with the latest add-ons and
                    security fixes
configure-timezone  Set timezone for both system and PHP
install-fax         Install HylaFAX
add-fax-extension   Add additional fax extension for use with HylaFAX
install-fail2ban    Install Fail2Ban for additional security
install-dongle      Install GSM/3G calling capability with chan_dongle
raspbx-backup       Backup your complete system to an image file

Last login: Tue Jul 29 19:26:33 2014
root@raspbx:~#
```

The next thing you need to do is set up a static IP address. You need to specify the static IP address you want to use, the network mask and the gateway of your router or cable modem. The,

```
ifconfig
```

command will provide your current IP address and the network mask. Your new static IP will have the same first three octets as your current IP. The last octet must be outside the range that your router uses for dynamic IP addresses. To find the gateway address, enter:

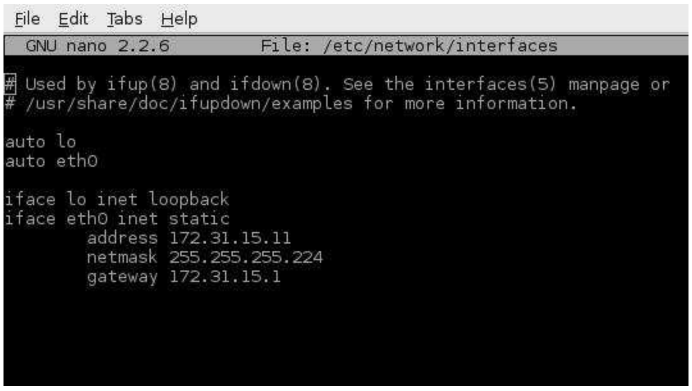
```
netstat -r
```

Edit the `interfaces` file with the command:

```
nano /etc/network/interfaces
```

Your `interfaces` file will look something like the screenshot below.

Note that you need to replace the word `dhcp` on the `eth0` line with the word `static`. Also be sure to press the <Tab> key once to get the desired indendation.



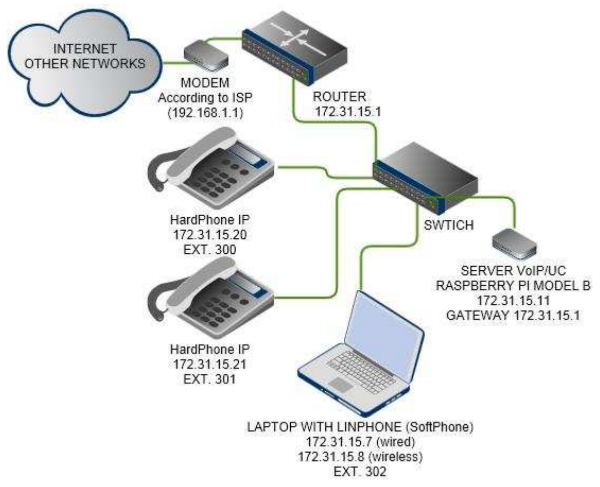
```
  GNU nano 2.2.6          File: /etc/network/interfaces

# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.

auto lo
auto eth0

iface lo inet loopback
iface eth0 inet static
        address 172.31.15.11
        netmask 255.255.255.224
        gateway 172.31.15.1
```

Once saved, reboot to use your new network settings. From now on you can use either the static IP or the `raspbx` hostname. For example, when using PuTTY to connect with the static IP address above, I can now use `root@172.31.15.11`.

Asterisk configuration

We are now going to use the FreePBX graphical user interface to configure the Asterisk software. This helps to make the process simple and easy. The FreePBX software came pre-installed with the Asterisk image.

An example architecture diagram is shown below.

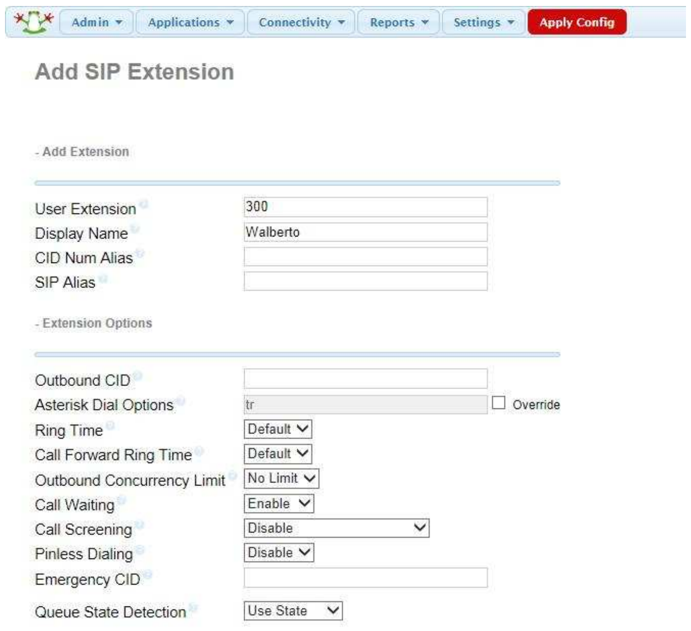


To start FreePBX open a web browser and enter `http://raspbx` or your static IP. (For Apple Mac you will enter `http://raspbx.local`). This will open the FreePBX Administration page.

There are three options:
1) **FreePBX Administration** is used to configure Asterisk
2) **User Control Panel** is for users to adjust their personal settings
3) **Get Support** opens the FreePBX website.

Click on `FreePBX Administration`. The default login is `admin`, with the password `admin`. The Applications menu has various options including Extensions, Conferences and Ring Groups. Click on `Extensions`.

As no extensions exist, you will add a new extension. For the Device option choose `Generic SIP Device` then click on `Submit` to go to the next page. There are many options but we will just set the User Extension to `300`, the Display Name to `Walberto` and the secret option to `ext300`. Click on `Submit` to add the extension.



Add SIP Extension

- Add Extension

User Extension	300
Display Name	Walberto
CID Num Alias	
SIP Alias	

- Extension Options

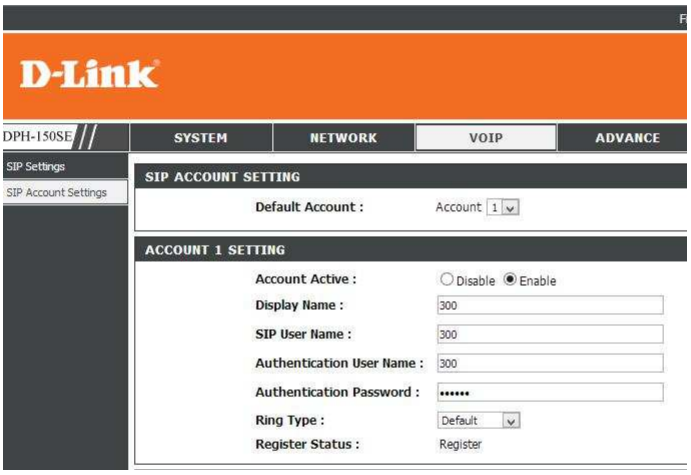
Outbound CID	
Asterisk Dial Options	tr ☐ Override
Ring Time	Default
Call Forward Ring Time	Default
Outbound Concurrency Limit	No Limit
Call Waiting	Enable
Call Screening	Disable
Pinless Dialing	Disable
Emergency CID	
Queue State Detection	Use State

On the right side of the screen, click on `300` to view the extension you just added. Verify the port option is set to `5060`. Click on `Submit`, then click on the red `Apply Config` button to save your changes.

Repeat this procedure for the other extensions that you want to create. I created extensions 301 and 302.

We now configure the IP phone extensions. This will vary by device but we will use the Dlink DPH-150SE as an example. The important settings are to disable the DHCP option, verify the SIP Phone Port Number is `5060`, the Registrar Server is the IP address of your Raspberry Pi and in the Others section we enable the `Register with Proxy` option.

For the SIP Account Settings option we enter the details we previously used when adding extensions using FreePBX. The Authentication User Name is the extension number and the Authentication Password is the secret entry (i.e. `ext300`).


D-Link

DPH-150SE | SYSTEM | NETWORK | VOIP | ADVANCE

SIP Settings
SIP Account Settings

SIP ACCOUNT SETTING

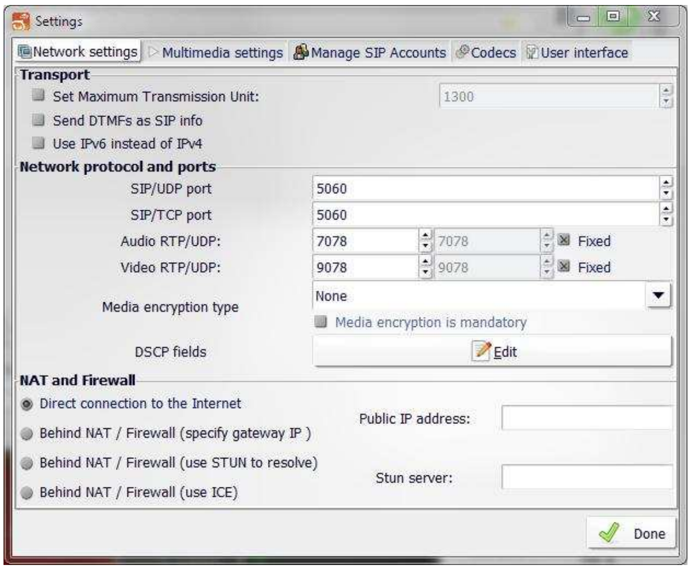
Default Account :	Account 1

ACCOUNT 1 SETTING

Account Active :	Disable [✓ Enable]
Display Name :	300
SIP User Name :	300
Authentication User Name :	300
Authentication Password :	••••••
Ring Type :	Default
Register Status :	Register

Softphone configuration

Start Linphone and from the `Options` menu choose `Preferences`. Confirm the Network settings are as shown below.


Settings

Network settings | Multimedia settings | Manage SIP Accounts | Codecs | User interface

Transport

Set Maximum Transmission Unit:	1300
Send DTMFs as SIP info	
Use IPv6 instead of IPv4	

Network protocol and ports

SIP/UDP port	5060		
SIP/TCP port	5060		
Audio RTP/UDP:	7078	7078	[✓ Fixed]
Video RTP/UDP:	9078	9078	[✓ Fixed]
Media encryption type	None		
	Media encryption is mandatory		
DSCP fields	Edit		

NAT and Firewall

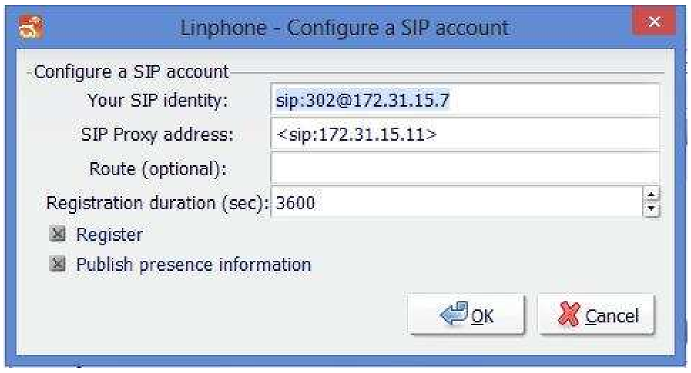
[✓ Direct connection to the Internet]
Behind NAT / Firewall (specify gateway IP)
Behind NAT / Firewall (use STUN to resolve)
Behind NAT / Firewall (use ICE)

Public IP address:
Stun server:

Done

In the Multimedia settings, verify that `Echo cancellation` is enabled. In Manage SIP Accounts enter your display name. In my example the soft phone is extension 302 so the username is `302`. The resulting SIP address is `<sip:302@172.31.15.7>`. Click the `Add` button to register the account with Asterisk.

Enter your SIP identity from above and the SIP Proxy address (i.e. the IP address of your Raspberry Pi). See the screenshot at the top of the next column for details.


Linphone - Configure a SIP account

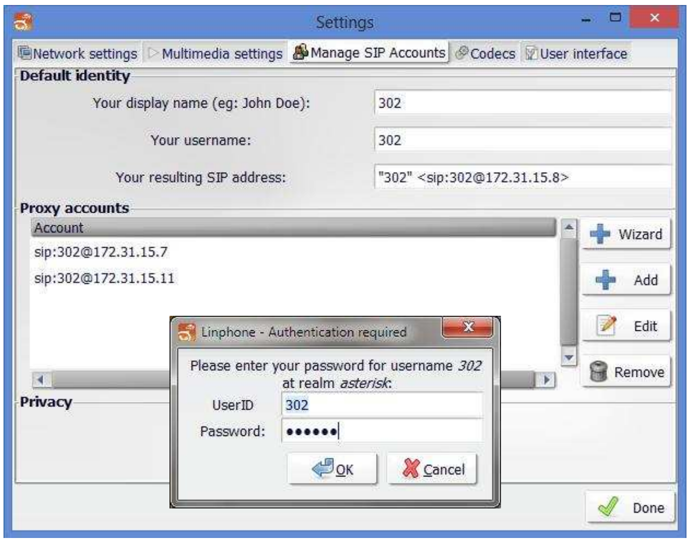
Configure a SIP account

Your SIP identity:	sip:302@172.31.15.7
SIP Proxy address:	<sip:172.31.15.11>
Route (optional):	
Registration duration (sec):	3600

[✓ Register]
[✓ Publish presence information]

OK | Cancel

You will then be asked for a password. For extension 302 I set this to be `ext302`. Click `OK` and the registration should be confirmed.


Settings

Network settings | Multimedia settings | Manage SIP Accounts | Codecs | User interface

Default identity

Your display name (eg: John Doe):	302
Your username:	302
Your resulting SIP address:	"302" <sip:302@172.31.15.8>

Proxy accounts

Account
sip:302@172.31.15.7
sip:302@172.31.15.11

Wizard | Add | Edit | Remove

Privacy

Linphone - Authentication required

Please enter your password for username *302* at realm *asterisk*:

UserID	302
Password:	••••••

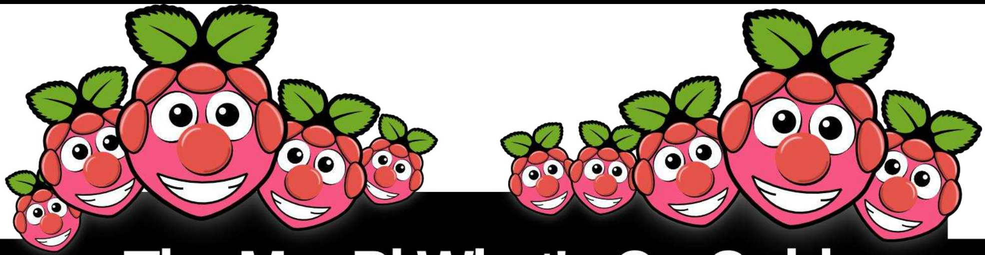
OK | Cancel

Done

With FreePBX and Asterisk you can implement various services like conference rooms, IVR (Interactive Voice Response), call groups, plus incoming and outgoing calls via the normal PSTN, SIP trunk lines or the internet.

The Future

The development of communications using VoIP and the internet is driving the convergence of Unified Communications systems into a single system and environment. FreePBX and Asterisk is a superb example of how sophisticated communication systems can be implemented for very low cost.



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

Southend Raspberry Jam

When: Saturday 16th August 2014, 10.00am to 5.00pm
Where: Tickfield Centre, Tickfield Avenue, Southend-on-Sea, SS2 6LL, UK

There will be talks, show and tell, workshops, videos, robots and lots of fun. Learn how to code in Scratch, Blockly and Python. http://soslug.org/node/2023

Raspberry Jam Silicon Valley

When: Saturday 16th August 2014, 1.30pm to 4.30pm PDT
Where: Computer History Museum, 1401 N. Shoreline Blvd., Mountain View, CA 94043, USA

Open to all and free to attend whether you are new or experienced with the Raspberry Pi.
http://www.eventbrite.com/e/8469381147

Malvern Raspberry Jam

When: Wednesday 20th August 2014, 3.45pm to 5.00pm (Student) and 7.30pm to 9.00pm (Adult)
Where: Wyche Innovation Centre, Walwyn Road, Upper Colwall, Malvern, WR13 6PL, UK

Come to be inspired, make friends and collaborate on new ideas. Student Jam: http://www.eventbrite.com/e/10077559251 Adult Jam: http://www.eventbrite.com/e/11053058997

Manchester Raspberry Jam 22

When: Saturday 30th August 2014, 10.00am to 5.00pm
Where: The Shed, John Dalton West, Chester Street, Manchester, M1 5GD, UK

Everyone welcome. Opening talk around 10am, followed by hacking on whatever you want.
http://mcrraspjam.org.uk/next-event/ and http://www.eventbrite.co.uk/e/12458883857

Soton, Winchester and Basingstoke Raspberry Pi Meeting

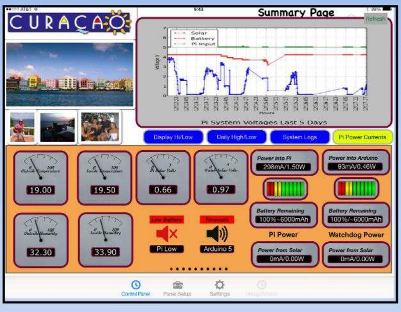
When: Wednesday 3rd September 2014, 8.00pm to 10.00pm
Where: The Roebuck PH, Stockbridge Road, Winchester, SO23 7BZ, UK

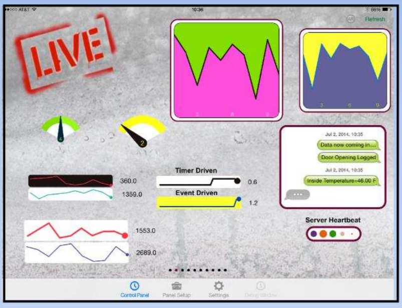
Relaxed evening Q&A event over a beer with Raspberry Pi's running, no need to register. Contact Dougie Lawson: dl1ims@gmail.com

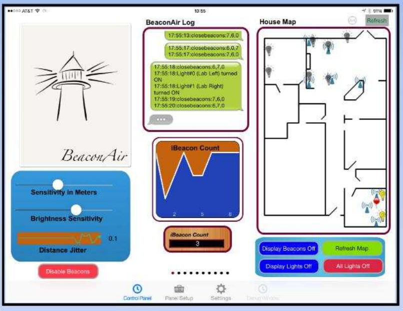
RasPiConnect

Connect your Raspberry Pi to the World!

Build Your Own Contol Panel!







- ***New Live Controls***
- EASY to setup - no syncing required
- Exchange your panels with friends
- Supports multiple Raspberry Pis and multiple Arduinos
- Build your pages on your iPad/iPhone
- Ten pages of control panels
- Supports any computer that supports Python (windows, linux, etc.)

Visit milocreek.com for more info!

Available on the App Store

MiloCreek
Where Innovation Flows

Supports Arduino
with in-app purchase

Plusberry Pi

Now
live on IndieGogo

Beautiful design

Integrated powered hub

2.5" HDD bay

Power switch

www.PlusberryPi.com

Part 2: Variables, procedures and sprites

Jon Silvera

Guest Writer

SKILL LEVEL : BEGINNER

A few years ago I brought my old BBC Micro down from the loft to show my kids how I started in computing. To my surprise my two girls Molly and Gracie, and my son David were all intrigued by the BASIC prompt and the Syntax Error response returned by about every input.

They wanted to know more so we spent a few days learning a few BASIC programming commands and playing some classic games. It got me thinking that wouldn't it be great to bring back a computer in the same vein... something that brought access to programming right to the forefront just like it was back in the 1980's.

The Raspberry Pi turned out to be the answer as it retains many attributes of the BBC Micro such as accessible GPIO ports. The only downside was that it did not have a version of BASIC directly suited to our needs. Enter Gordon Henderson, the author of the WiringPi libraries, who developed a version of BASIC called RTB (Return To BASIC). If you have programmed in BBC BASIC (arguably one of the best ever versions of BASIC) then you will be very familiar with RTB. It is designed specifically to support the Raspberry Pi and its GPIO.

A deal was struck between FUZE and Gordon to produce FUZE BASIC, which includes a vast array of enhancements. These tailor FUZE BASIC so it is more in line with the requirements of the newly revised UK IT curriculum.

At FUZE we focus on FUZE BASIC to deliver a learning experience far more accessible to a broader age range and ability group than more complex languages. Quite simply, BASIC is easier to pick up and learn than just about any other language ever devised. You do not need to be adept at maths, you do not need to understand the operating system to any great extent and you certainly don't need to have programmed before.

While more experienced programmers might scoff at us BASIC students, I assure you that BASIC has something for everyone. Even the most adept coders will find BASIC a great platform to test out ideas and experiment.

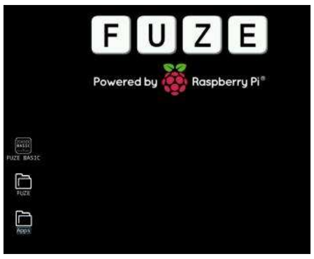
Getting FUZE BASIC

I am very pleased to announce that FUZE BASIC is available for FREE! Visit http://www.fuze.co.uk click on `Resources` and then download the latest

FUZE boot image from the FUZE BASIC Updates tab. The FUZE boot image is the same as the Raspbian boot image, but configured with the latest version of FUZE BASIC.

You will need to unzip the boot image file and install it onto an SD card. **Please note that you need a minimum 8GB SD card.** You also need software to install the image onto the SD card. Windows users can use Win32DiskImager, but full installation details for Linux, MacOS and Windows are available from the official Raspberry Pi website at http://www.raspberrypi.org/documentation/installation/installing-images/.

Next boot your Raspberry Pi with your new FUZE image. There isn't anything particularly different with the boot image compared to the standard Raspbian image, except for a new Desktop background image and a FUZE BASIC icon.



First, open the Fuze folder and then open the Programs folder. Inside this, create a new folder called MagPi.

We are going to create a game with this tutorial so the next thing is to download the graphics.

Please go back to http://www.fuze.co.uk and to the Resources page. Click on the Tutorials tab and download the six sprites contained in 'The MagPi Tutorial' section. These sprites are the player's rocket ship, the enemy rocks and the ever important bullet. Download and copy these six files into the MagPi folder we created earlier.

On the FUZE desktop, double click the FUZE BASIC icon to start FUZE BASIC.

The welcome screen will appear and you will be presented with the Ready> prompt.

Let's familiarise ourselves with the environment. The Ready> prompt means you are in Direct mode. Type in Hello and press <Enter>. You will get the message, “Equals expected”. That is exactly what should happen. The computer has no idea what Hello means. Instead, enter:

```
Number1 = 10
Number2 = 10
Answer = Number1 + Number2
```

I suspect there are many of you who are already more than comfortable with what is going on here, but we need to explain to the newcomers.

Variables

The words Number1, Number2 and Answer are just names. They are called variables. Variables are tags we store values in. We could have used any name but generally it is best to use names that make sense in the context of the program. If we wrote a program using variables like N1 and N2 and A then when we come back to the program later we will have no idea what everything means. However, variable names like ShipX and ShipY are more obvious. Try and make this a habit. You will appreciate it later.

So, we stored the number 10 in the variable Number1 and 10 in the variable Number2. We then said that the variable Answer equals Number1 + Number2.

At this point you should know what the value of Answer is. Do you? I hope so or we're in big trouble! Enter:

```
PRINT Answer
```

You should see 20.

If anything else whatsoever happens then something has not gone to plan and you should go back and check where you went wrong.

Direct mode and Edit mode

We are currently in Direct mode. This is where we can enter commands and expect an immediate response. For example we can check variables and enter simple instructions. It's not programming though is it? Press F2 to enter the FUZE BASIC Editor.

Useful commands/shortcuts in Direct mode

EXIT	Exit and return to the desktop
DIR	List files in the current folder
CD name	Change to folder name
CD ..	Go back one folder
LOAD name	Load program name
SAVE name	Save program name
NEW or F12	Clear program from memory
F2	Open the editor
RUN or F3	Run the current program

You will see a blank screen with a green flashing cursor and a dotted line across the bottom. This is the Editor environment. Here we can enter a list of program instructions that can be saved and executed (RUN) whenever we want.

Press F2 again. This will take you back to Direct mode. Actually it will ask you for a file name. In this first case don't bother, just press F2 again and it will put you in Direct mode again. One last time, press F2 again and you will be back in the Editor. You get the idea - F2 takes you between the Editor and Direct mode.

Hello MagPi

In the Editor enter the following program:

```
CYCLE
PRINT "Hello MagPi"
REPEAT
```

You don't actually need to worry about capitals or lower case when entering commands. It is a good habit in FUZE BASIC to type commands in capitals, but it is not essential. However the names we give to variables, as we did above with Number1 and Answer etc., are set in stone. If we give a variable the name NUMBER1 then we must refer to it as such every time in our program. Variable names are case-sensitive. If we expect the variable numBER1 to return the same result then we are in for a big surprise. The variable numBER1 has not been defined so will generate an error.

Enough of the dull stuff! You should at this point be in the Editor with the program as listed on the left. Press F3. If at this point the program has not been saved then it will ask you to do so. Just enter a name like Hello and press <Enter>. The program should then run. "Hello MagPi" should display in a never ending list down the screen.

To stop it, press the <Esc> key.

Press F2 to go back to the Editor and change the program so that it looks like the following:

```
CYCLE
PRINT "Hello MagPi ";
REPEAT
```

The only difference is that we have added a space in between MagPi and the end quotation mark and added a semi-colon to the end of the PRINT line. The semi-colon tells BASIC to display the next item immediately after the last one and not on a new line. The space just puts a gap in between. Press F3 to run the program again. This time instead of a long list of "Hello MagPi" going down the screen, this time it displays "Hello MagPi " across the screen.

More about the Editor

Again, press <Esc> to exit the program and then F12 to wipe the current program from memory. You should have a blank screen in the Editor. If not then try pressing F2 and F12 until you get there. When you are in Direct mode you can type,

```
EXIT
```

to exit the program.

Right now we need to be in Direct mode with no program in memory.

If you type,

```
NEW
```

in Direct mode it will clear the memory so when you go into the Editor it will be blank. In direct mode enter:

```
DIR
```

Among other things, you should see a directory called `MagPi`, if you did everything above. Enter:

```
CD MagPi
```

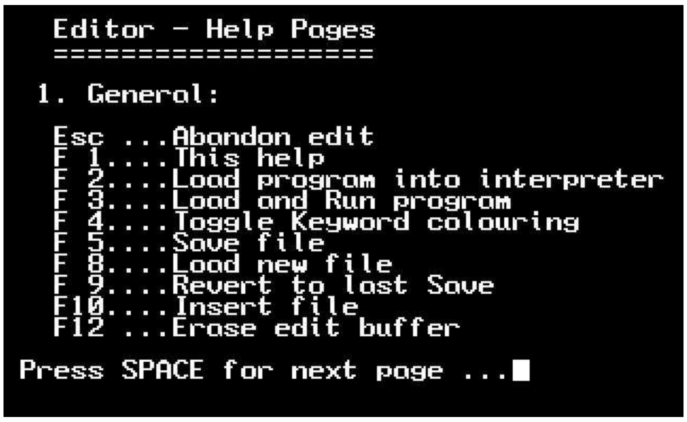
This will put us in the same directory, or folder, where we saved the sprites earlier. When we create our program we want it to be in the same folder as the sprites.

Press `F2` to go to the Editor and enter the following program:

```
// MagPi Game
PROC Setup

CYCLE
REPEAT

END

DEF PROC Setup
ENDPROC
```

Press F3 to run the program. The first time, it will ask you for a file name. Enter `MagPi` and press `<Enter>`. The FUZE BASIC Editor automatically adds the file extension `.fuze` to file names. When you press `<Enter>` the program will run but nothing of any interest will happen as we haven't done anything of interest yet! If you have entered anything incorrectly you may get an error in which case F2 will take you back to the Editor. If all is well, the screen will just go blank as the program is in an infinite loop (`CYCLE` / `REPEAT`).

Press the `<Esc>` key and then `F2` to return to the Editor. Note, you can get help at any time in the Editor by pressing `F1`.



```
  Editor - Help Pages
  ===================

 1. General:

  Esc ...Abandon edit
  F 1....This help
  F 2....Load program into interpreter
  F 3....Load and Run program
  F 4....Toggle Keyword colouring
  F 5....Save file
  F 8....Load new file
  F 9....Revert to last Save
  F10....Insert file
  F12 ...Erase edit buffer

 Press SPACE for next page ...
```

This is the basic structure of our program. It is important as we progress to try and build some good habits. When naming variables a popular method is called Camel Case. ThisIsCamelCase. The reason we use it is because we are not allowed to use spaces in variable names. Camel Case (notice the humps) makes things readable at a glance.

As you write larger programs another variable issue will raise its head. Short, non-related variable names WILL cause you grief later. A 200+ line program will be very difficult to understand if you have used variable names like `pbx` instead of `PlayerBulletX` and just `X` instead of `PlayerX`.

Long names take more time when editing, but they will save hours later when debugging. Also consider at some point your code might be scrutinised by someone else. You do want to make your program legible!

Time to play a game

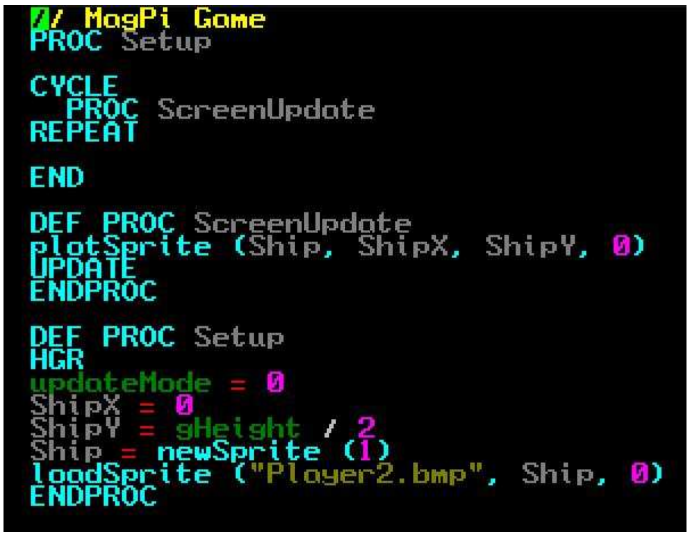
We are going to store our variables in a `PROC`edure called `Setup`, along with our sprites and sound files. This keeps our program organised. The `CYCLE` and `REPEAT` commands define our main program loop. This is where all the action will happen.

Now we need to load the sprites so we can start having some fun. Open the Editor by pressing `F2`, if you're not already in it.

Edit the `MagPi` code so it becomes:

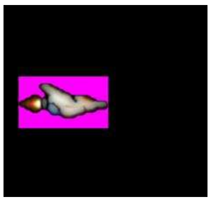
```
// Magpi Game
PROC Setup

CYCLE
  PROC ScreenUpdate
REPEAT

END

DEF PROC ScreenUpdate
  plotSprite (Ship, ShipX, ShipY, 0)
  UPDATE
ENDPROC

DEF PROC Setup
  HGR
  updateMode = 0
  ShipX = 0
  ShipY = gHeight / 2
  Ship = newSprite (1)
  loadSprite ("Player2.bmp", Ship, 0)
ENDPROC
```

It should look something like this in the Editor.



```
// MagPi Game
PROC Setup

CYCLE
  PROC ScreenUpdate
REPEAT

END

DEF PROC ScreenUpdate
plotSprite (Ship, ShipX, ShipY, 0)
UPDATE
ENDPROC

DEF PROC Setup
HGR
updateMode = 0
ShipX = 0
ShipY = gHeight / 2
Ship = newSprite (1)
loadSprite ("Player2.bmp", Ship, 0)
ENDPROC
```

When you run the program you might be surprised to see a bright pink (magenta) box surrounding our space ship. Don't worry, there's nothing wrong... we just need to set this colour to be transparent. FUZE BASIC will not draw any colour that is specified as the transparent colour.

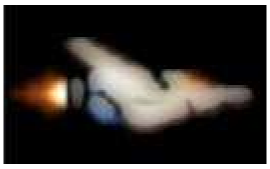


We need this because a sprite graphic is simply a box and everything in the box is drawn. So if we have a white background in our sprite it will display a white box, which is no good on our black space background. We can stop this by specifying a single colour to be transparent so it is not drawn.

Add the `setSpriteTrans` command directly below the `loadSprite` command, as shown below, and then press F3 to run the program again:

```
  loadSprite ("Player2.bmp", Ship, 0)
  setSpriteTrans (Ship, 255, 0, 255)
```

That's much better!



Program explanation

Now that we have something more substantial, let's take a proper look at what is going on. We'll explain each section of the code line by line.

```
// MagPi Game
```

Anything displayed after the // one the same line is ignored. This allows us to add comments to make the program easy to understand.

```
PROC Setup
```

Tells the program to jump to the procedure called `Setup`, run whatever is there and return when it comes to the `ENDPROC` command.

```
CYCLE
  PROC ScreenUpdate
REPEAT
```

These three lines define our main program loop. Whatever is between the `CYCLE` / `REPEAT` loop will be repeated indefinitely.

```
END
```

`END` signifies the end of the program. When the program executes the command it will return to Direct mode. In our program this can only happen when `<Esc>` is pressed.

```
DEF PROC ScreenUpdate
```

This starts the definition of the ScreenUpdate procedure. It will update everything on the screen

```
plotSprite (Ship, ShipX, ShipY, 0)
```

The plotSprite command draws a sprite (Ship) at screen coordinates X (ShipX) and Y (ShipY) using version 0. Having different versions of a sprite enables animation.

```
UPDATE
```

When we draw graphics to the screen they are actually drawn to a background screen. The UPDATE command copies the background screen to the main screen. This keeps everything running smoothly and simplifies game programming.

```
ENDPROC
```

Return back to where the procedure was called.

```
DEF PROC Setup
```

The Setup procedure is deliberately placed at the end of the program. It is a good habit to keep all the main setup commands in one place so it is easy to find them. Also this will usually end up being quite a big routine so you don't want it getting in the way at the beginning.

```
HGR
```

This initialises high-resolution graphics mode.

```
updateMode = 0
```

Sets the screen update system to manual.

```
ShipX = 0
ShipY = gHeight / 2
```

ShipX and ShipY are used to store the X and Y coordinates of the player's ship. ShipY takes a system constant called gHeight, which is the pixel height of the screen, and divides it by 2 to determine the vertical middle of the screen.

```
Ship = newSprite (1)
```

This creates a sprite ID called Ship with room for just one version of the graphic. The version count starts from 0. Later we will increase the number of versions so we can animate the sprite.

```
loadSprite ("Player2.bmp", Ship, 0)
```

The loadSprite command assigns the named graphic image to a sprite ID (Ship) and stores it as version 0.

```
setSpriteTrans (Ship, 255, 0, 255)
```

This specifies the transparent colour of the sprite ID (Ship) to bright pink using red (255), green (0) and blue (255) values between 0 and 255.

```
ENDPROC
```

Return back to where the procedure was called.

Add controls and movement

We are now making progress but unfortunately we do not have space for much more this month. Let's add one more piece to our game to make it feel like we are really getting somewhere.

We are going to add a new procedure called CheckControls which will check for the Up, Down, Left and Right arrow keys being pressed and correspondingly change the position of the rocket.

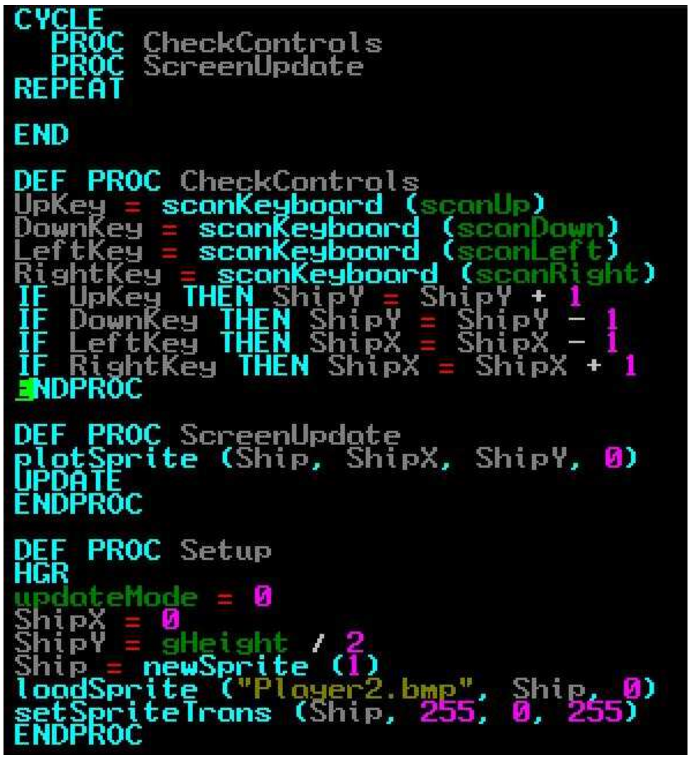
First we add the call to the CheckControls procedure to our main program loop:

```
CYCLE
  PROC CheckControls
  PROC ScreenUpdate
REPEAT
```

Now add the definition of the CheckControls procedure to your program. It does not matter where you place this procedure but we will place it before the definition of the ScreenUpdate procedure:

```
DEF PROC CheckControls
  UpKey = scanKeyboard (scanUp)
  DownKey = scanKeyboard (scanDown)
  LeftKey = scanKeyboard (scanLeft)
  RightKey = scanKeyboard (scanRight)

  IF UpKey THEN ShipY = ShipY + 1
  IF DownKey THEN ShipY = ShipY - 1
  IF LeftKey THEN ShipX = ShipX - 1
  IF RightKey THEN ShipX = ShipX + 1
ENDPROC
```

Your new code should look something like this in the Editor.



```
CYCLE
  PROC CheckControls
  PROC ScreenUpdate
REPEAT

END

DEF PROC CheckControls
UpKey = scanKeyboard (scanUp)
DownKey = scanKeyboard (scanDown)
LeftKey = scanKeyboard (scanLeft)
RightKey = scanKeyboard (scanRight)
IF UpKey THEN ShipY = ShipY + 1
IF DownKey THEN ShipY = ShipY - 1
IF LeftKey THEN ShipX = ShipX - 1
IF RightKey THEN ShipX = ShipX + 1
ENDPROC

DEF PROC ScreenUpdate
plotSprite (Ship, ShipX, ShipY, 0)
UPDATE
ENDPROC

DEF PROC Setup
HGR
updateMode = 0
ShipX = 0
ShipY = gHeight / 2
Ship = newSprite (1)
loadSprite ("Player2.bmp", Ship, 0)
setSpriteTrans (Ship, 255, 0, 255)
ENDPROC
```

Run the program by pressing F3. I'm going to be very disappointed if you have not worked out what will happen when you press the cursor keys.

Coming up...

Next time we will learn more BASIC commands plus add a few enemies and some fire power to our game.

FUZE BASIC

FUZE®

COMPETITION
TEASER

In our October issue, the folks at FUZE are planning to run a FUZE BASIC programming competition, with an incredible **£500** of prizes!

First prize is the amazing FUZE T2-R kit, worth £230. Not only does this have everything you need to maximise your enjoyment of the Raspberry Pi, it also includes an OWI programmable robotic arm kit!

Second prize is the superb FUZE T2-A kit, worth £180. Again, this contains everything you need including a Raspberry Pi Model B, solderless breadboard, various electronic components, SD card with FUZE BASIC, printed Programmer's Reference Guide and much more!

Third prize is the excellent FUZE T2-C kit for your Raspberry Pi. Worth £80, this kit contains the FUZE case, keyboard, integrated USB hub, FUZE I/O board and power supply.

Details of the prizes can be found at http://www.fuze.co.uk/products.

Over the coming months you will learn everything that you need, but if you want to give yourself a head start then download the FUZE BASIC Programmer's Reference Guide from http://www.fuze.co.uk/resources-2/.

FUZE®

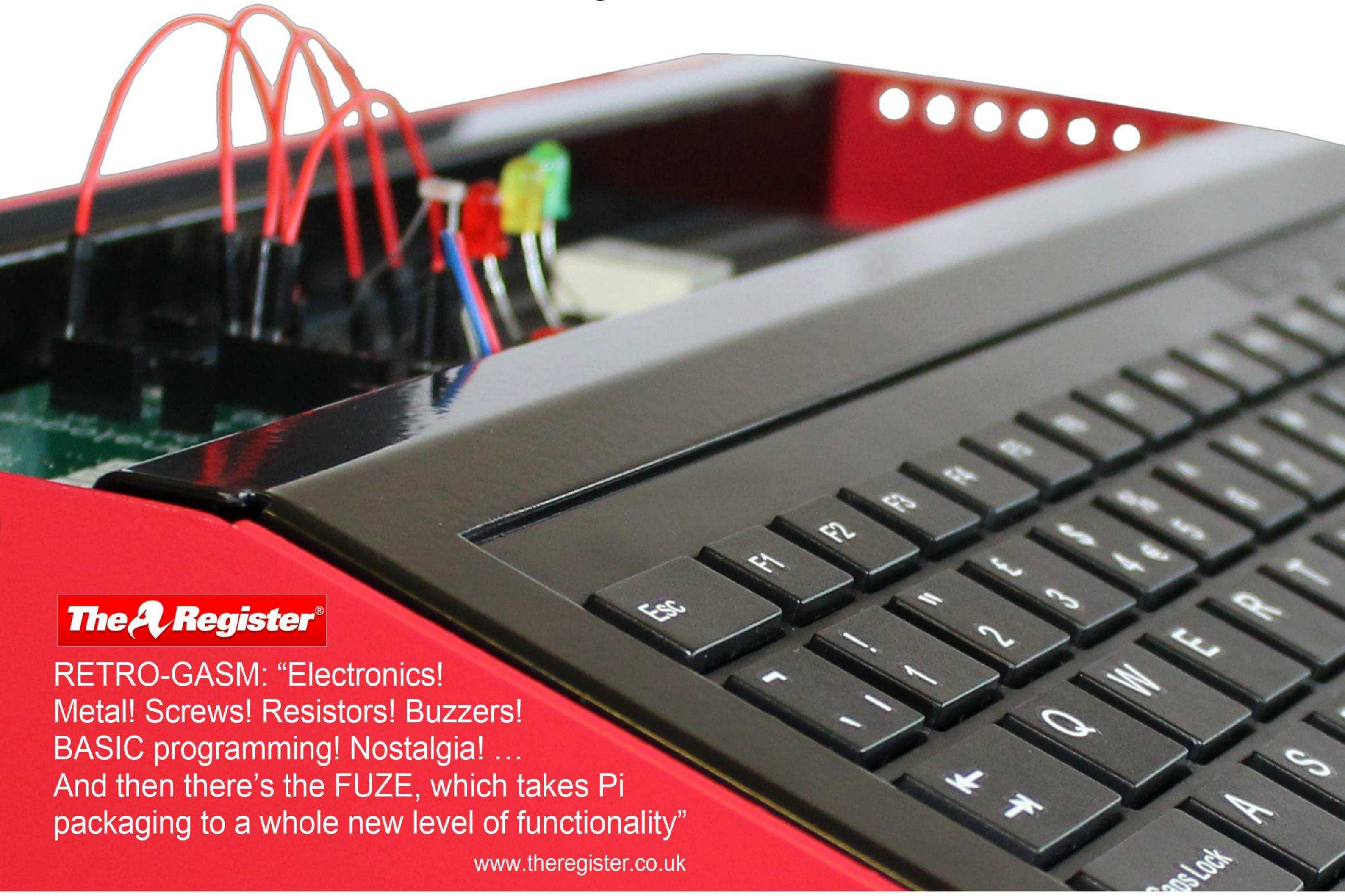
"The FUZE is what the Raspberry Pi was designed for"

THE NATIONAL MUSEUM OF COMPUTING

FUZE Type II

The Ultimate Case

for Raspberry Pi B & the new B+



The Register®

RETRO-GASM: "Electronics! Metal! Screws! Resistors! Buzzers! BASIC programming! Nostalgia! … And then there's the FUZE, which takes Pi packaging to a whole new level of functionality"

www.theregister.co.uk

Prices start from £89.99
See FUZE website for details

- **Protect your Pi from physical & static damage**
- **UK keyboard* & 4 Extra USB ports**
- **FUZE I/O Board with GPIO pass-thru**
- **Clearly labeled input output ports**
- **2 Amp power supply and on/off switch!**
- **Adds analogue ports, 4 in & 1 out**

FUZE Technology Ltd - www.fuze.co.uk
+44 (0) 1844 239 432 - contact@fuze.co.uk

* USA & German keyboard layouts are also available

PRINT EDITION AVAILABLE WORLDWIDE

The MagPi is available for FREE from http://www.themagpi.com, from The MagPi iOS and Android apps and also from the Pi Store. However, because so many readers have asked us to produce printed copies of the magazine, we are pleased to announce that printed copies are now regularly available for purchase at the following Raspberry Pi retailers...

Americas	EMEA	AsiaPac
adafruit INDUSTRIES	MODMYPI; AUTHENTIC SWAG EDUCATING SINCE '12; The Pi Hut; Pi Supply	AusPi Technologies Australia

Have Your Say...

The MagPi is produced by the Raspberry Pi community, for the Raspberry Pi community. Each month we aim to educate and entertain you with exciting projects for every skill level. We are always looking for new ideas, opinions and feedback, to help us continue to produce the kind of magazine you want to read.

Please send your feedback to editor@themagpi.com, or post to our Facebook page at http://www.facebook.com/MagPiMagazine, or send a tweet to @TheMagP1. Please send your article ideas to articles@themagpi.com. We look forward to reading your comments.