ISSUE 20 - FEB 2014

Get printed copies at themagpi.com

The MagPi

A Magazine for Raspberry Pi Users

PiBOX
Raspberry Pi hosting

Resolve Cable Modem 'Flap'
Environmental Monitor
Panoramic Photos
Remote Sensing
Quadcopter
Scratch I/O
Bare Metal
Algoid

Win a GEEK case, Pi NoIR camera, GPIO Cobbler, 2 SD cards & more

ISSN 2051-9990
02
9 772051 999008

The MagPi

http://www.themagpi.com

Welcome to Issue 20 of The MagPi magazine. It's a brand new year and we can't wait to see what is in store for the Raspberry Pi over the next 12 months.

After a massive response, we are pleased to write that the article series 'Bake your own Pi filling' is back by popular demand! In this article Martin Kalitis throws down the gauntlet by instructing how to create a bootable Linux SD card which can load within 10 seconds.

We have more from the Caribbean with Project Curacao. This project has been so popular with our readers that John Shovic is extending it further, in a future issue, with a conclusion presenting the project's results. In this issue John reviews the building and installation of the camera and shutter mount into the project, allowing the production of timed photos, before updating us on changes made to the project from past articles.

Deepak Patil introduces his project for panoramic photography using Pi-Pan, a robotic arm controlled by his Raspberry Pi to move his Pi Camera. Deepak looks at some of the code used to control this clever kit and discusses how to take pictures while out in your car.

We have more from Andy Baker's Quadcopter series with this issue reviewing his pre-flight checks. His article looks at controlling the movement of the Quadcopter and provides some handy questions and answers for those of you who have been building this project.

We have a great article detailing John Hobson's and Efrain Olivares' journey into managing the frustrating problem of internet dropout. We then head over to France where Yann Caron presents his development environment and language 'Algoid', before the NanoXion chaps present their Raspberry Pi colocation service.

As always, we keep you updated with the latest Raspberry Pi book reviews and upcoming events.

Let's begin...

Chief Editor of The MagPi

The MagPi Team

Ash Stone - Chief Editor / Administration / Layout
W.H. Bell - Issue Editor / Layout / Grapics / Administration
Bryan Butler - Page Design / Graphics
Ian McAlpine - Layout / Proof Reading
Matt Judge - Website / Administration
Aaron Shaw - Layout / Proof Reading
Colin Deady - Layout / Proof Reading
Chris 'tzj' Stagg - Graphics
Claire Price - Layout / Proof Reading
Amy-Clare Martin - Layout
Nigel Curtis - Proof Reading
Bulent Tuncel - Layout / Testing / Proof Reading
James Nelson - Testing / Proof Reading
Age-Jan (John) Stap - Layout

Contents

4 **PROJECT CURACAO: REMOTE SENSOR MONITORING IN THE CARIBBEAN**
Part 3: The camera subsystem

8 **PROGRAMMING FOR PI-PAN**
A pan and tilt control for your Raspberry Pi camera

10 **QUADCOPTER**
Part 2: Pre-flight checks

16 **ENVIRONMENTAL MONITOR**
Part 2: Using the Google Chart library to visualise data

19 **COMPETITION**
Win a Pi NoIR camera, GPIO breakout, gigabit hub, two SD cards and more

20 **CABLE MODEM**
Using a Raspberry Pi to automatically restore a lost internet connection

26 **RACKS OF PI**
Colocating Raspberry Pi's in France

29 **THIS MONTH'S EVENTS GUIDE**
Cambridge UK, Malvern UK, Mountain View California USA, Manchester UK, Southend-on-Sea UK

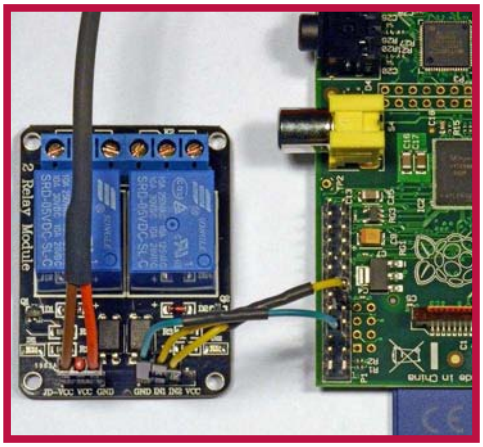
30 **ALGOID**
Programming made simple and fun

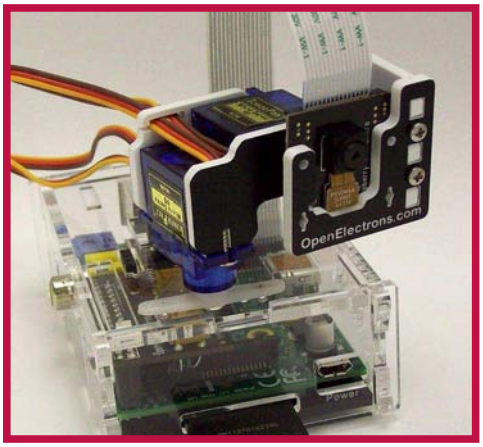
36 **THE SCRATCH PATCH**
Flexible I/O: using GPIO, SPI, files & more

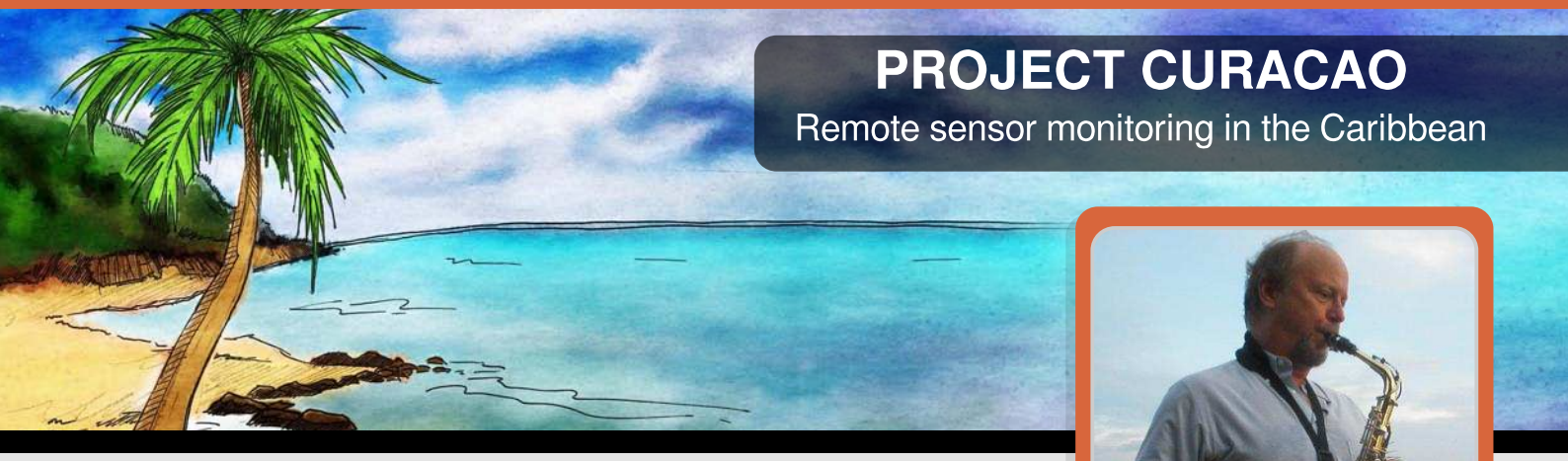
40 **MY OS: BUILD A CUSTOMISED OPERATING SYSTEM**
Part 2: Bake your own Pi filling - build tools and more

43 **BOOK REVIEWS**
Python in Easy Steps and Raspberry Pi Networking Cookbook

44 **FEEDBACK**
Have your say about The MagPi







PROJECT CURACAO

Remote sensor monitoring in the Caribbean

John Shovic

Guest Writer

Part 3: The camera subsystem

SKILL LEVEL : INTERMEDIATE

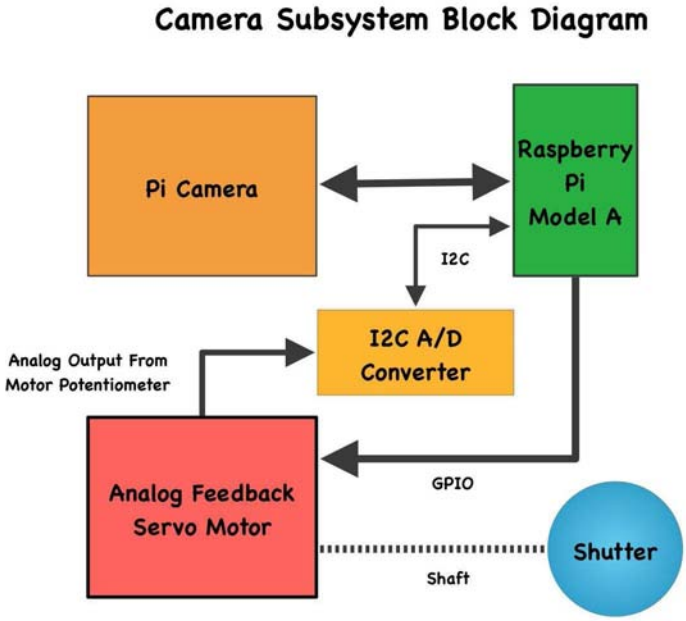
What is Project Curacao?

This is the third part of a series discussing the design and building of Project Curacao, a sensor filled project that will hang on a radio tower on the island nation of Curacao. Curacao is a desert island 12 degrees north of the equator in the Caribbean. Originally intended for four issues this series will now be extended to cover the results of the installation in a later issue of The MagPi.

Project Curacao is designed to monitor the local environment unattended for six months. It operates on solar power cells and will communicate with the designer via an iPad App called RasPiConnect. All aspects of this project are designed to be monitored remotely. Below is a picture of the tower where it will be installed in March 2014.

System description

Project Curacao consists of four subsystems. A Raspberry Pi Model A is the brains and the overall controller. The Power Subsystem was described in part 1 and the Environmental Sensor Subsystem was described in part 2. The Camera Subsystem consists of a Raspberry Pi Camera and a servo motor that opens and closes a "shutter" that is intended to keep salt spray and dirt from coating the plexiglass over the camera.

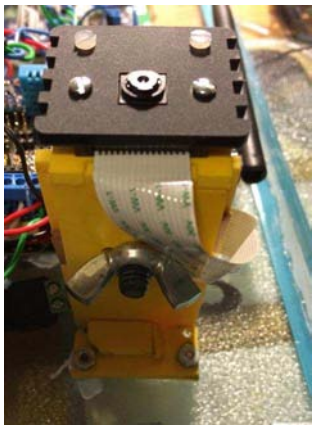


What hardware will be used?

1 Raspberry Pi Camera
1 Adafruit Analog Feedback Servo Motor
1 Adafruit ADS1015 12-Bit ADC - 4 Channel I2C
1 Tidy Cat plastic lid for shutter

Camera and Shutter Mount

We assembled the camera mount from soft plastic and superglue. It was designed to sit just under the plexiglass. Initially, we tried to mount it on top of the plexiglass, but had a very difficult time with the short fragile cable to the Raspberry Pi. The shutter and the servo motor completes the device.



Camera Software

The Pi Camera software turned out to be quite simple. The important part of the Python module to take a picture turned out to be only a few lines:

```
def  takePicture(source):
   # take picture
   print "taking picture"
   output = subprocess.check_output ("raspistill -o
/home/pi/RasPiConnectServer/static/picameraraw.jpg
-t 0",shell=True, stderr=subprocess.STDOUT )
   output = subprocess.check_output("convert
'/home/pi/RasPiConnectServer/static/picameraraw.jpg
' -pointsize 72 -fill white -gravity SouthWest
-annotate +50+100 'ProjectCuracao
%[exif:DateTimeOriginal]'
'/home/pi/RasPiConnectServer/static/picamera.jpg'",
shell=True, stderr=subprocess.STDOUT)

   pclogging.log(pclogging.INFO, __name__, source )
   print "finished taking picture"
   return
```

We started out using the picamera library (https://pypi.python.org/pypi/picamera) because it was pure Python and feature heavy. We removed it and went back to raspistill (as above) because of a GPIO line conflict with the RPIO GPIO library. We wanted to use the RPIO library because it supports DMA (Direct Memory Address - this means it works with no intervention from the CPU, thus multitasking has no effect) driven PWM (Pulse Width Modulation) for the servo motor, but it is NOT compatible with picamera.

We uninstalled the RPIO library and put the latest RPi.GPIO library back in and used their software based PWM. While it is jittery (because of the multi-tasking Linux on the Raspberry Pi), we are using a servo with a feedback potentiometer for position feedback. We check the shutter position after using the RPi.GPIO PWM library by performing an A/D conversion of the pot voltage using the ADS105. We can then see the position of the servo and run the PWM again until it is within the bounds of the shutter being open or closed. With this feedback, we can accurately position the shutter even with the jittery response due to multi-tasking.

Timed Photos

Currently, the Project Curacao main software (described in the upcoming part 4), takes a picture every hour and on demand from the control program RasPiConnect. We can even email pictures on demand using our control panel. A crontab entry transfers the latest picture to a server in the US using scp so we have a prototype external webpage showing the current status of Project Curacao. The webpage is being hosted by our good friends at MiloCreek (developer of RasPiConnect) and the address is: http://milocreek.com/projectcuracaographs.

Future functions

We are planning to add three additional features to the Camera Subsystem. These are time lapse photography, a short video on demand and a streaming video interface. A big unknown for the streaming video interface is the amount of bandwidth available for uploading data in Curacao. We know that it is an ADSL modem, but we don't know the parameters on the upload speeds.

Camera Shutter Cover Servomotor

The camera shutter cover is designed to keep the majority of dirt and sea spray off the plexiglass in front of the Pi Camera. It uses an analog feedback servo motor so we always know where the shutter is regardless of multitasking or unexpected power downs.

Control Panel

The RasPiConnect control panel shows the current luminosity and the analog position of the servo motor. It also allows you to take pictures, open and close the shutter and email the current picture. The graph shows the status of the Raspberry Pi OS and of the Project Curacao and RasPiConnectServer software.

Updates on previous articles

Relays

The fan relay and the Raspberry Pi relay have been replaced with latching relays that consume no power after they have been switched. This saves 60ma. We used http://shop.ciseco.co.uk/3v-to-5v-bistable-latching-relay-kit/. Thank you to Paul Carpenter for the excellent suggestion.

Environmental Sensors

We are adding a pitot tube with sensor to measure airspeed, using the Airspeed Sensor Kit from http://robotshop.com. Alternatively, we are looking at using the open loop voltage of the wind turbine to approximate the windspeed (see http://switchdoc.blogspot.com). It looks like it will work up to wind speeds of about 50 MPH.

Battery Watchdog System

The Battery Watch Arduino software has expanded to about 3000 lines of code. There are certainly bugs in this software. One of the unresolved issues about the architecture of Project Curacao is that the software on the Raspberry Pi can be easily updated remotely across the Internet while the software on the Battery Watchdog Arduino can not be updated. This is because there is no current way of updating the Arduino without plugging in a USB serial port. While we are investigating options (different boot loader, figuring out a way of using the USB port on the Raspberry Pi for programming) we are looking at connecting the Raspberry Pi to make our system just a bit more resilient than it currently is.

To partially address this problem we are going to build what is called a "Dead Man Switch" in software. If the Raspberry Pi has been off for two days, then the Battery Watchdog will turn it on. This helps us recover from many types of errors in the Battery Watchdog software.

We also ran into a very subtle bug using the Serial library in the Arduino. We are using two serial ports (the main USB port and a second UART talking to the Raspberry Pi). If both ports aren't set to the same baud rate, then they stop working after a while. With the baud rates matched, everything is very solid.

Power System Update

We have found an inexpensive 15 Watt wind turbine to add to the system. This could very well charge our system enough to not have to shutdown at night. More on how we are hooking up the turbine into and controlling it via the Raspberry Pi next month.

What is coming Up?

Part 4 of this article describes the software system used for Project Curacao. Part 5 will describe the process of installing the unit in Curacao and show the first weeks of results. All of the code used in this article is posted on GitHub at http://github.com/projectcuracao.

More discussion on Project Curacao at: http://switchdoc.blogspot.com

Raspberry Pi® Starter Kits

Everything but the screen

http://shop.pimoroni.com

Starter Kit £75
Deluxe Starter Kit £120

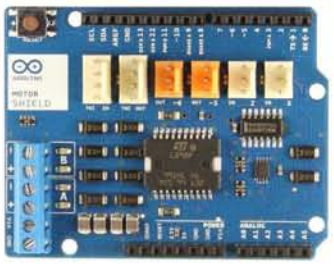
Pimoroni Gift Cards

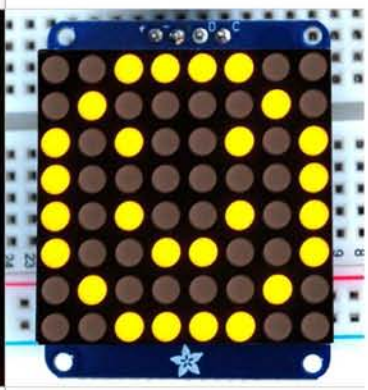
Choosing shiny things is hard!

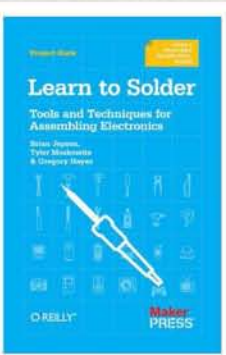
10 £25 £50

http://shop.pimoroni.com

PIMORONI







Wearables! Arduinos! Components! Dead Trees!

We now stock a range of shiny things you can make cool stuff with!

http://shop.pimoroni.com

PIMORONI

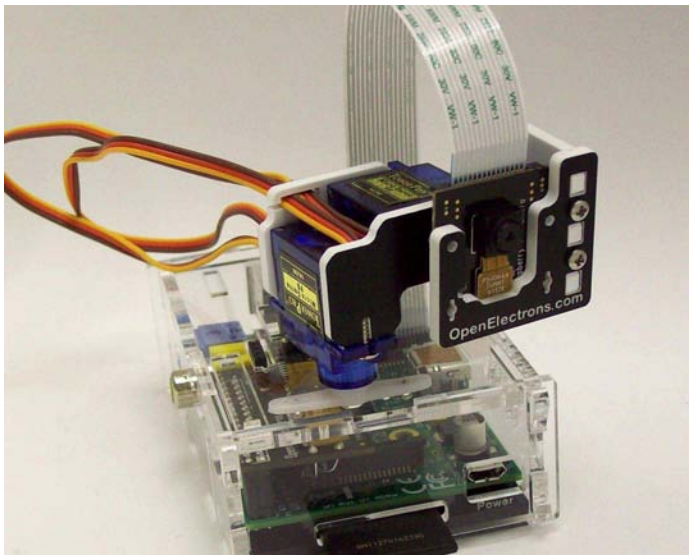
A pan and tilt control for your Raspberry Pi camera

Deepak Patil

Guest Writer

SKILL LEVEL : INTERMEDIATE

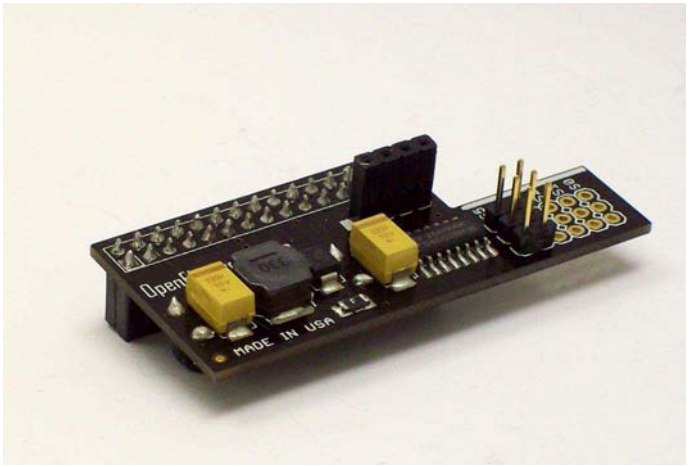
Pi-Pan[1] is a robotic arm that provides pan and tilt movements for your Raspberry Pi camera[2]. It uses standard radio control servo motors to provide X and Y axis movement.



There is a programming interface for these servos and the position of each servo can be controlled through a Python program. This gives you the ability to point your camera in a precise direction.

The Raspberry Pi camera is mounted in front of the arm assembly and its flex cable is conneted to the camera connector on your Raspberry Pi (this is the connector located between the Ethernet port and the HDMI port).

The headers of the servo motors are connected to the Pi-Pan servo controller board and this is then attached to the Raspberry Pi GPIO. The servo controller board provides an I²C interface and offers four additional channels for expansion. We will hack the servo board in a future article.



Programming the Pi-Pan

Pi-Pan uses the Servo Blaster[3] software, which implements a programming interface for the pan and tilt movements.

For example, to tilt the Pi-Pan head to position 'y', in your Python code call the function:

```
do_tilt(int y)
```

The parameter 'y' is the tilt (vertical) position of Pi-Pan ranging from 80 to 220, where:

80 = looking down
220 = looking up
150 = straight ahead (neutral position)

To pan the Pi-Pan head to position 'x', in your Python code call the function:

```
do_pan(int x)
```

The parameter 'x' is the pan (horizontal) position of Pi-Pan ranging from 50 to 250, where:

50 = looking left
250 = looking right
150 = straight ahead (neutral position)

To control individual servos, in your Python code call the function:

```
pwm(int pin, int position)
```

The parameter 'pin' is the pin number of the servo motor you want to control. Values range from S0 to S5.

The parameter 'position' is the desired destination position for the servo motor. Values range from 50 to 250.

Synchronized picture taking and camera movements with Python

To make a panoramic picture using Pi-Pan you take overlapping pictures from a fixed position, rotating the camera about a single axis, and then join the pictures using stitching software such as AutoStitch[4].

The following Python code will take pictures to make a panorama:

```
for pan in xrange(50, 226, 25):
  p.do_pan (int(pan))
  time.sleep(1)
  cmdstr = "/var/tmp/pic_" + str(pan) +
    ".jpg"
  # Take the picture.
  subprocess.call(["raspistill", "-o",
    cmdstr, "-rot", "180"])
```

This code starts the Pi-Pan servo motor at the left-most position (50) and steps through increments of 25 until it reaches the last position (225), just short of the right-most position (250).

At each step the servo stops momentarily and a picture is taken. All the pictures are stored in the `/var/tmp` folder.

The image above is an example of what can be achieved with the Pi-Pan.

Tips for shooting outdoors from a car

1) You can power the Raspberry Pi from your car's cigarette lighter port via a suitable USB adapter.
2) Before you leave home, setup a script on the Raspberry Pi such that when it is powered on it will automatically start shooting pictures. You can set this up in the `/etc/init.d` folder.
3) While shooting you can just rest the Raspberry Pi on the side view mirror and hold it steady.
4) Don't start the engine while the Raspberry Pi is running. Most cars cut off power to the cigarette lighter port while starting the engine.

References

[1] Pi-Pan: http://www.openelectrons.com/Pi-Pan
[2] Camera: http://www.raspberrypi.org/camera
[3] ServoBlaster: https://github.com/richardghirst/PiBits
[4] AutoStitch: http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html

Quadcopter pre-flight checks

Andy Baker

Guest Writer

SKILL LEVEL : ADVANCED

[Ed: This is an advanced skill level article. It is not a step-by-step guide for beginners. Quadcopters are dangerous. They can cause damage and serious injury. Be responsible, educate yourself and take precautions to protect everything and everyone in the vicinity.]

In Issue 19 I presented a high-level overview of how to build your own quadcopter using a Raspberry Pi as the flight-controller. However, there were lots of details missing in that article about tuning and testing the settings for your quadcopter. This article fills in those blanks.

Some prerequisites...

These tests cannot be powered from a wire – you need a fully charged battery attached to your quadcopter. Unlike the rest of the world where duct tape is "the force", with quadcopters it seems to be velcro! Use it generously to make sure everything is securely attached.

Connect to your quadcopter

Your quadcopter needs Wi-Fi. You need another machine (Raspberry Pi, Windows or Linux) that can open a secure shell and log into your quadcopter Raspberry Pi. I use an iPad with WebSSH or rlogin from one of my other Raspberry Pi's depending on where the testing is happening.

Get the latest code

This project is still live and the code changes regularly. Make sure you have the latest software, which is available at https://github.com/PiStuffing/Quadcopter. Copy the software into your home directory, e.g. `/home/pi`.

The command-line

The quadcopter Python code `qc.py` has various command-line parameters which control its testing and flights. Here's the complete list. We'll go through these in detail throughout the article. To run your quadcopter, at the command prompt where the code lives, type:

```
sudo python ./qc.py
```

The following is a list of the available parameters:

`-f` – fly the quad
`-v` – run the video
`-g` – calibrate gravity
`-h XXXX` – set the PWM pulse width for hover
`--tc #` – run test case number #
`--hvp XXXX` – horizontal velocity PID P gain
`--hvi XXXX` – horizontal velocity PID I gain
`--hvd XXXX` – horizontal velocity PID D gain
`--vvp XXXX` – vertical velocity PID P gain
`--vvi XXXX` – vertical velocity PID I gain
`--vvd XXXX` – vertical velocity PID D gain
`--aap XXXX` – absolute angle PID P gain
`--aai XXXX` – absolute angle PID I gain
`--aad XXXX` – absolute angle PID D gain
`--arp XXXX` – angular rate PID P gain
`--ari XXXX` – angular rate PID I gain
`--ard XXXX` – angular rate PID D gain

Countdown beeps

My quadcopter has a beeper – it counts down from 5 beeps to 1 beep before the blades spin up. I put it there to add time before take-off allowing me to abort the flight for any reason. Some of the tests we are going to do require you to start up your quadcopter and then hold it or move it in some way. You will do this during the countdown beep sequence. Therefore you MUST have a beeper, or some equivalent indicator, that take-off is imminent. The beeper can be replaced by an LED as long as you can see it clearly.

Binding the code to the hardware

I'm assuming here that you already have a fully built breadboard matching the circuit diagram / breadboard layouts provided in Issue 19.

The code refers to the propellers by their location (front-left, front-right, back-left and back-right – looking from the top) and the direction the propellers rotate (clockwise or anticlockwise). The first step of pre-flight setup is to ensure that the hardware agrees with the software.

With your battery disconnected, fit the blades to the motors to match the code: front-left and back-right blades spin anti-clockwise; the other pair clockwise. Look carefully at your set of four blades. It should be clearly visible which are designed to move clockwise and which anti-clockwise – fit them to match the code's expectations.

Plug the three motor wires into the ESC (Electronic Speed Controller) and the ESC PWM leads into the breadboard. The ESC cables are generally brown, red and orange (resistor colour coding for 1, 2, 3) or black, red and white / grey for ground, 5V and signal respectively. Connect the orange / 3 / white / grey signal wire to the following breadboard pins:

Back-left ESC plugs into BCM pin 22 / GPIO pin 15
Front-left ESC plugs into BCM pin 17 / GPIO pin 11
Front-right ESC plugs into BCM pin 18 / GPIO pin 12
Back-right ESC plugs into BCM pin 23 / GPIO pin 16

The red / 2 wire connects to 5V and the remaining brown / 1 / black wire connects to ground.

Now we need to make sure the connection to the ESCs are also in agreement with the direction each blade should rotate. For that we need to use some test code that starts up each of the motors in turn at a slow speed starting from front-left, then front-right, then back-left and ending with back-right.

Search the code for "TESTCASE 1:" for the code change used by this test.

Run the test from the command prompt:

```
sudo python ./qc.py --tc 1 -h 200
```

200 is an ESC PWM setting that should be high enough that all motors start to turn, but low enough that they spin slowly.

As each blade spins up, you should check that each spins in the correct direction. The easiest way is simply to feel the airflow from each blade as it spins – it should be blowing downward! If it feels wrong, swap any two of the three ESC wires to the motor and try again.

As mentioned, the code steps through the motors in order front-left, front-right, back-left and back right – if they don't run in that order for you, check that your ESCs are connected to the correct GPIO pins.

Hover speed

The next step is to determine an approximate hover speed. That means finding the PWM pulse width in microseconds that is required to sustain a hover. If you don't set it on the command line with the `-h` parameter then the code defaults the value to 590, which is the value that works for me. But how do you find this magic number for your quadcopter?

It involves another set of code changes – look for "TESTCASE 2:" in the code. This sets all the motors to spin for 10s, and then stop – how fast they spin is set by the command line parameter. Start up your quad with:

```
sudo python ./qc.py --tc 2 -h 100
```

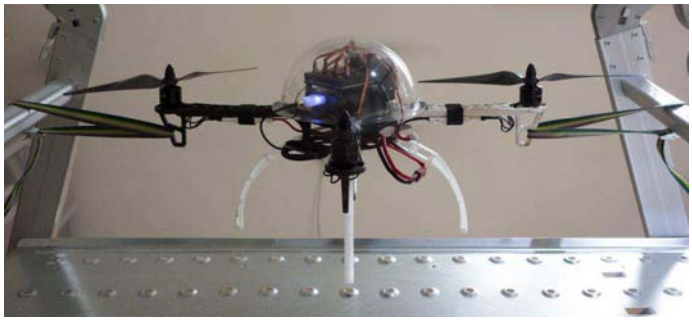
While it's beeping pick it up and and hold it above your head, making sure the propeller blades can't hit anything

(especially you!) when they start spinning. DO NOT LET GO, there's no control software running and the quad will fly around like a screaming banshee trying to cut everything in sight with its blades, especially you!

As the blades spin, focus on how the quadcopter feels. If the blades made no difference (or don't spin at all) increment the -h parameter by 100 and run the test again. As you start to feel the blades having effect, and your quadcopter is feeling lighter, be a lot more cautious. Increment -h by 25 or less until you get to a stage where your quadcopter feels weightless. At this point note the `-h` number. Find the line of code saying `cli_hover_speed = 590` and replace 590 with your magic number.

Inner PID balancing

For this test, you will need two raised platforms to balance your quadcopter between – chairs, stools, sturdy cardboard boxes will all do nicely – they just need to be the same height. They only need to be tall enough to lift your quadcopter's legs off the floor. Your quadcopter centre of gravity needs to be just below the support point – if it's above, your quadcopter will be too unstable to do PID tuning. I had to hang my quadcopter between the rungs of a step-ladder in my latest tests to achieve this. Look at http://blog.pistuffing.co.uk/?p=1314 to see how I did this previously with stools.



The code changes in "TESTCASE 3:" disable the front-right and back-left motors. Place these arms on the two platforms such that your quadcopter dangles in free space between the platforms as shown in the photo. The other two motors will spin at your newly found `cli_hover_speed`. Your quad may not balance between the platforms unless gently supported by you. Enter:

```
sudo python ./qc.py --tc 3 --arp 50
--ari 0 --ard 0
```

Between the 5 beeps and 4 beeps, your quad needs to sit on the floor as gyro calibration is underway. At 4 beeps, get your quad onto the platforms with the front-right and back-left legs balancing in between. Gently hold the nearest leg. When the blades spin up, carefully let go – what happens?

• If the arm you were supporting drops with your hand as you move it down, your angular rotation rate PID P gain (`--arp`) is not doing its job and needs to be increased significantly – try doubling it.

• If the arm seesaws increasingly meaning you need to hold the leg firmly, your PID is way too high and needs significantly lowering – try reducing by 50%

• If the arm stays where it was, wobbling slightly, but the wobble neither shrinks nor grows, you're done – lets call this K_u – the ultimate gain – in my recent retesting it came out to be `--arp 150`.

You are aiming to reach the point where your quadcopter starts seesawing – this is much easier to approach from below rather than trying to reduce the frantic wobble if you start with the P gain too high!

The next stage is to start playing with the I and D gain values. I did this months ago with days of testing, first adding I gain and reducing P gain as a result, and finally adding D gain. However, I've learnt recently from a comment on my blog that the P, I and D gains are related. Check out http://en.wikipedia.org/wiki/Ziegler-Nichols_method.

Using this method you can find K_u then K_p. The values for K_i and K_d can be estimated mathematically. To do so, you need to count how many wobbles happened during the test. For me that was 54 in a 15s test flight, so the oscillation period $T_u = 15 / 54 = 0.278$s. With these values, I tried Z-N PIDs – each worked well, but each with a minor flaw. In the end, I picked the best from each resulting in $K_p = 65$, $K_i = 360$ and $K_d = 5.2$.

I tried these values and they worked perfectly as the picture on the left shows. The left and right props are stationary, the front and back props are spinning. No blur in the photograph equals no wobbles in the hover.

With your equivalent values in place, rerun TESTCASE 3. When the test is running you should be able to let go and tap the quadcopter and both feel it resisting due to the differential gain, and returning to where it started due to its integral gain.

If you're in that position, replace the default values for `cli_arp_gain`, `cli_ari_gain` and `cli_ard_gain` in `CheckCLI()` with your own values. If the PID values don't work it is time for free-form testing:

• Oscillation is due to a too high P gain
• Lack of return to horizontal is due to a too low I gain
• The jitters are due to a too high D gain.

Enjoy the tinkering until you find your perfect tuning.

Now it's time to tune the other PIDs outdoors, but before that happens, sensor calibration is needed.

Gravity Calibration

Taking off from a sloping surface, even if it's very subtle will ultimately lead to sideways drift. That is because if the quadcopter takes off from a slope, it stays leaning throughout the flight. Leaning means some of the power that would be applied as lift is now redirected as horizontal acceleration... or that would be the case if only the gyroscope outputs were used to calculate the tilt angles. Luckily the accelerometer is also used and the result combined produces accurate values over time.

The accelerometer reads the force of gravity distribution across its X, Y and Z axes. When the quad is horizontal, its output in the X, Y and Z axes should be 0, 0, 1... or it will be after you calibrate it.

Find yourself a flat strong platform on which your quadcopter can stand. I made a custom one from some perspex with bolts at the corners for easy fine tuning but any wood / metal / glass platform propped up with cardboard / paper / anything thin will do just fine.

Place your platform on the floor. With a spirit level in both X and Y axes, get the platform as horizontal as you can by propping up the corners. Now sit the quadcopter on the platform. Leave the battery disconnected and instead boot it up from the the micro USB plug / wall wart. Then enter:

```
sudo python ./qc.py -g
```

Wait a few seconds – job done. Check the output by typing:

```
cat qcgravity.cfg
```

You should see the corrective values required by the accelerometer so it reads 0, 0, 1, when it is horizontal.

Gyro Calibration

This happens automatically for every flight and you don't need to do anything. However it is worth mentioning it in passing for completeness.

In the same way the accelerometer gives innaccurate readings without tuning, so too does the gyro. The net result of this is the quadcopter believes it is rotating on its axes even though it is not. Gyro calibration happens as the first thing when the quadcopter code starts up while it is sitting still on the ground.

The remaining PIDs

Your quadcopter is now safe for outdoor test flights. The move outdoors reduces the number of valuable things that the quadcopter can hit during these tests!

Luckily the remaining PIDs are based upon mathematical guesstimation – and if you get it wrong nothing really bad happens. It does depend somewhat on the 'gut-feel' for propeller speeds you have acquired while doing the earlier testing.

For outside testing the `--tc` parameter is not used. Instead `-f` is used to signify a preprogrammed flight pattern: a 5 second takeoff at 0.35 m/s, a 5 second hover at the resultant height of 1.75m, followed by a 5 second descent back to the ground.

Vertical velocity PID

First tuning is for the vertical velocity PID; for take-off, hover and landing. Imagine a speed of ascent of 1 m/s – that is very fast and going to need a lot of power. I guestimated (based on the experience from TESTCASE 2) that an input of 1 m/s puts out a PWM increment of 150µs to the vertical velocity PID P gain. I added another 50 as the vertical velocity PID I gain. These values just worked, so I have left them as the default values in `CheckCLI()`.

Place your quadcopter on the horizontal platform and type:

```
sudo python ./qc.py -f --vvp 150 --vvi 50
--hvp 0.0
```

It should follow the predefined flight plan as described. Do not be tempted to run this test indoors – the expected results I describe may not be what you see, and you don't want your quadcopter hitting the ceiling!

If it all starts going horribly wrong, pressing <CTRL>+C will

move on to the next step in the pre-programmed flight pattern. Hammering it hard in a blind panic will bring the quadcopter crashing to the ground – been there, done that, too many times.

If this testcase works for you then update `cli_vvp_gain`, `cli_vvi_gain` and `cli_vvd_gain` in `CheckCLI()` to use these values.

Absolute angle PID

Next the absolute angle PID – again I used the same gut feel plus guesswork algorithm. If we want to change the quadcopter from say hover to 2° and we want this to happen quickly, but not manicly, then the absolute angle PID needs a P gain of about 5 to convert from the 2° error to a 10°/s rotation speed. Again this value worked fine so I left it at that. Check yours by taking off from the ground (the horizontal take-off platform is no longer needed) by entering:

```
sudo python ./qc.py -f --aap 5 --aai 0
--aad 0 --hvp 0
```

You should see the same flight pattern, even if the ground is uneven or sloping. Again, if this works for you, update the default values for `cli_aap_gain`, `cli_aai_gain` and `cli_aad_gain` in `CheckCLI()`.

Horizontal velocity PID

Finally, the horizontal velocity. This PID takes a target of the desired speed, puts out a target acceleration which is then converted to a desired tilt angle by dividing the output by gravity g (9.81m/s^2) and taking the inverse tangent. So if we wanted to accelerate at 1g horizontally, then the quadcopter would tilt at 45°. So we start with a target to move at 1 m/s. Let's say we want this to produce 5 m/s^2 acceleration (≈ 0.5g) => tilt angle of arctan(0.5g / 1g) ≈ 26° which feels about right. So the P gain here is 1 m/s to 0.1g = 0.1. Seems like a good gut-feel starting point.

You may notice this time I haven't said "and it worked fine for me" because I am still testing it. This PID is there to prevent drift in windy conditions – the quadcopter should hover above the take-off point regardless of wind. But wet winter weather, combined with crashes in testing, are preventing me from completing this test. Stick with `--hvp 0.0` unless you are feeling confident to take on these final testing tweaks yourself.

Flight Control

Currently the quadcopter is autonomous – its flights are hard-coded rather than under the control of the user via a radio control (that's another article, once I've built and tested it). That means if you want to change a flight pattern, you need to change the code. Search for the `# Interpreter:` tag.

Currently, this code is set up to climb for 5 seconds at 0.35 m/s, hover for 5 seconds and then descend for 5 seconds at 0.35 m/s. By all means tinker with the timings and vertical velocity targets and see what effect they have.

Flying in the middle of nowhere

Throughout this article, there's an assumption that the quadcopter and the SSH client have Wi-Fi connectivity. But if you're out flying in a field then that will probably not be the case. You need to set up your quadcopter as a wireless access point (WAP). I won't explain the details of this here but just make you aware of the requirement. Check out my blog entry at http://blog.pistuffing.co.uk/?p=594, plus the others linked from there and many more like it on the Raspberry Pi forums, for details on setting up a WAP.

Q&A

If you've deviated from my design you may have questions about why your quadcopter is not working for you. Here are a few I've second guessed to guide you.

Q: How do I use my UBEC to power my Raspberry Pi?

A: My ESCs don't have an UBEC built in, so I added a DC-DC converter (switching regulator with input / output isolation) to produce 5V from the LiPo battery. That's what your UBEC does too. I suggest looking in the Raspberry Pi Forum or joining http://diydrones.com and use their forum.

Q: How do I protect against the battery going flat?

A: A really good battery can still only power a flight for under 10 minutes. Start all flights and testing with a full battery and keep in mind how long your testing has been going on. I do need to add power management to my quadcopter, but that's not very interesting so very low priority for me. Do some research, prebuilt options exist.

Q: My quadcopter does XXXX during outdoor testing. Why?

A: There are so many reasons. Luckily with each flight comes a set of diagnostic data saved as a .csv file. This is a generic spreadsheet format which Microsoft Excel and LibreOffice can understand, allowing you to diagnose the problem. Have a look in my blog http://blog.pistuffing.co.uk/?tag=statistics for the various graphs that can be plotted from the raw data.

INTRODUCING.....

THE PI BOT

From a galaxy not too far away comes the Pi Bot
On a mission to entertain earthlings in the way of the Pi

</> PROGRAM AND CONFIGURE YOUR OWN PERSONAL ROBOT

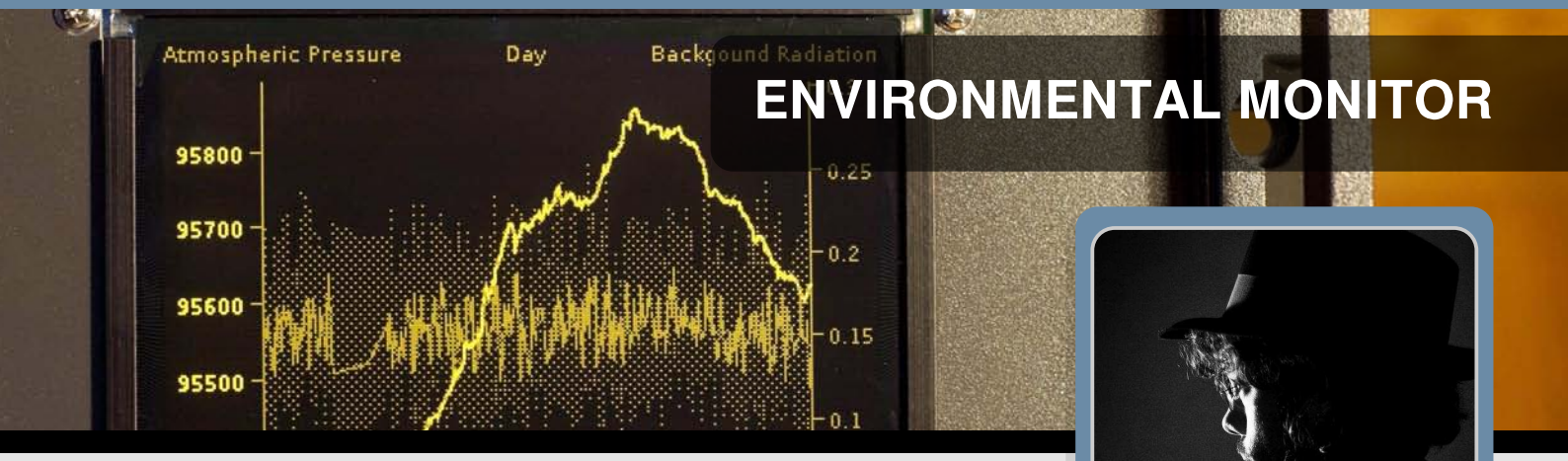
MADE FOR MAKERS

OPEN SOURCE

A UK ENTERPRISE

Find out more online @ WWW.PIBOT.ORG

FOLLOW #THEPIBOT



Publishing temperature and pressure data - Part 2

Pierre Freyermuth

Guest Writer

SKILL LEVEL : INTERMEDIATE

In the previous article, in Issue 19 of The MagPi, a Java example was given that demonstrated how to collect and log data from an I²C air pressure sensor. One convenient way to visualise and analyse the data is to use the JavaScript Google Charts library to produce a web page. Depending on the need these charts may be accessed locally, stored on the Raspberry Pi and accessed remotely, or stored and accessed remotely.

Creating chart data files

In the `DataChannel` class, a scheduled task is started to periodically average the real-time data collected. This produces several different time scale files in comma separated value format (CSV). Files containing the real time hour, day, month and year summaries are stored on the Raspberry Pi SD card.

```
private static final int MS_TO_HOUR_POINT = 15000;
Timer t = new Timer();
averagingTask = new AveragingTask();
t.schedule(averagingTask, MS_TO_HOUR_POINT, MS_TO_HOUR_POINT);
```

Uploading the files

Often a Raspberry Pi is running on a home network, behind a router that runs network address translation (NAT). This is helpful for keeping the Raspberry Pi away from the web, but is problematic if the data recorded on the Raspberry Pi needs to be viewed directly. One solution is to upload the files from the Raspberry Pi to your favourite web server using file transfer protocol (FTP). Apache provides a Java library that includes functions and classes for FTP file uploading. This library can be downloaded from http://commons.apache.org/proper/commons-net/download_net.cgi.

The `DataChannel` class uses a static method of the `UploadFTP` class to upload the files to an online web server:

```
private static final FTPClient ftp = new FTPClient();
private static final String HOSTNAME = "your_web_server.com";
public static synchronized void store(Path localFilePath) throws SocketException, IOException {
  ftp.connect(HOSTNAME);
  if (!FTPReply.isPositiveCompletion(ftp.getReplyCode())) {
    ftp.disconnect();
    return;
  }

  ftp.login("your_login", your_password");
  ftp.setFileType(FTP.BINARY_FILE_TYPE);
  ftp.enterLocalPassiveMode();

  InputStream input = new FileInputStream(localFilePath.toFile());
  ftp.storeFile("pilogger/" + localFilePath.getFileName(), input);
  ftp.disconnect();
}
```

Creating a web page

An HTML web page can be used locally or remotely to plot the data on a chart. This can be achieved with the Google JavaScript charting library Google Charts https://developers.google.com/chart/.

To import a .csv file the jquery-csv JavaScript library can be used. This library can be found at https://code.google.com/p/jquery-csv/.

JavaScript can be used to make charts on the HTML web pages and these can be made to show different time scales. To import CSV data into the `drawChartFromCSV` function, first create a data array from the input file:

```
var arrayData = $.csv.toArrays(csvString, {onParseValue: $.csv.hooks.castToScalar});
```

Then convert this data to a Google Charts data table:

```
var data = new google.visualization.DataTable();
data.addColumn('datetime', 'Date');
data.addColumn('number', 'Min');
data.addColumn('number', 'Average');
data.addColumn('number', 'Max');
data.addColumn('number', 'Max');
for (var i = 0; i < arrayData.length-1; i++) {
  var d = new Date();
  d.setTime(arrayData[i+1][0]);
  data.addRow([d, arrayData[i+1][2], arrayData[i+1][1],
    (arrayData[i+1][3] - arrayData[i+1][2]), arrayData[i+1][3] ]);
}
```

Finally a chart object can be created:

```
var chart = new google.visualization.ComboChart(document.getElementById(elementId));
chart.draw(data, options);
```

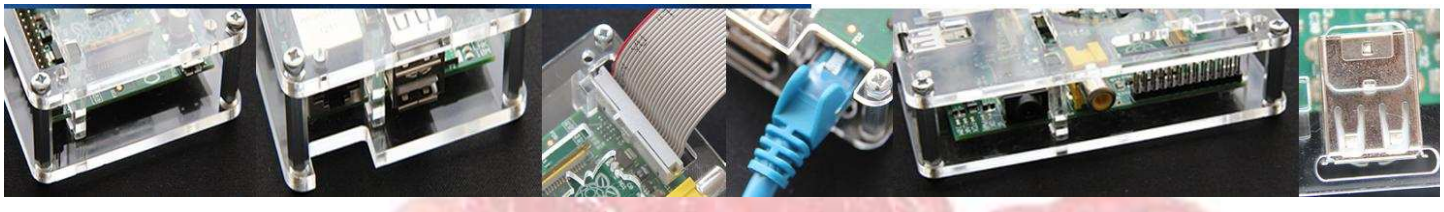
For more information look at the source code at https://code.google.com/p/pilogger/wiki/OnlinePilogger.

An example implementation can be found at : http://muth.inc.free.fr/pilogger/day.html

Outlook

You can follow the story and evolution of the system via my blog page. I have now added wireless temperature probes, and other sensors are planed: http://pierremuth.wordpress.com/rpiadventure/

FEBRUARY COMPETITION



Once again The MagPi and PC Supplies Limited are proud to announce yet another chance to win some fantastic Raspberry Pi goodies**!**

This month there is one MASSIVE prize!

The winner will receive a Cyntech Special Edition Geek case, Pi NoIR camera board, GPIO breakout board, breakout cable set, VESA mount, 16GB and 32GB SD cards and a gigabit network hub!

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

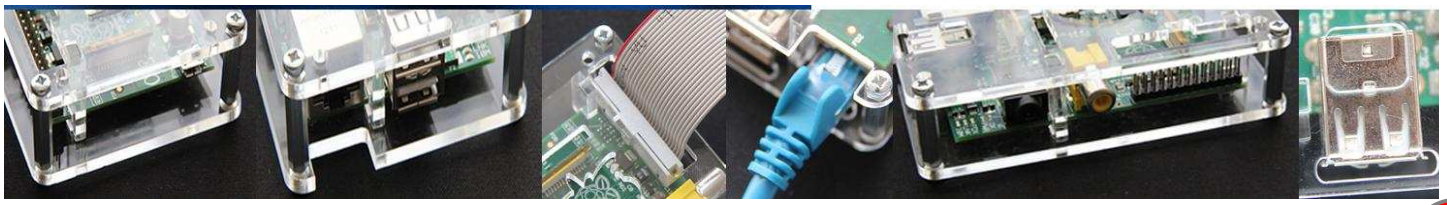
Closing date is 20th February 2014.
Winners will be notified in the next issue and by email. Good luck!

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

December's Winner!

The winner of a new 512MB Raspberry Pi Model B plus an exclusive Whiteberry PCSL case, 1A PSU, HDMI cable, 16GB NOOBS memory card, GPIO Cobbler kit, breadboard and jumper wires is **Robert Carr (Didcot, UK).**

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



How to automatically restore a lost internet connection

SKILL LEVEL : ADVANCED

John Hobson and Efrain Olivares
Guest Writers

After many visits, the cable company technician gave me his cell phone number as he left my house and said to call if anything changed. My problem with intermittent loss of internet connectivity, “modem flap” could not be solved. I was trekking to a downstairs closet to manually power-cycle my cable modem once or twice a day and the modem was getting old.

So after upgrading the cable modem, replacing all of my interior cabling, connectors and terminations and eliminating all but one professional grade splitter with no reduction in modem flaps, I had two options. One was to call my cable provider and ask them to evaluate the integrity of all of the fiber-optic cable, copper wiring, connectors and terminations between the cable head end and my residence. I would have also asked them to install an additional network node in my neighbourhood to reduce the number of “homes passed” on the network node servicing our home.[1] My other option was to try the DIY approach. It seemed prudent to choose the latter.

What I needed to do was automatically check my internet connection every 10 minutes. Then, if no internet connection was detected, power cycle my cable modem off for 15 seconds, then back on and wait 4 minutes for the modem to handshake with the cable plant. After the handshake I’d check for connectivity again. In my case the internet connection almost always returned after one power cycle, but I’d give it three chances before giving up. Then 10 minutes later the process would repeat. When I got internet back I wanted an email log of the outage time and length sent to me. I also wanted a soft, pleasing audio tone to sound when the modem power cycled to counteract the frustration I felt each time there was another modem flap.

Raspberry Pi solution

I chose the Raspberry Pi as the platform because of its size and flexibility. I had already set one up and used Python to print “Hello World” and had enough experience with electronics to use the GPIO pins to flash LEDs and trigger relays. My programming skills are, alas, not up to the task of writing the scripts for this project so I asked for the help of a fellow southern Californian, Efrain Olivares, who has been coding for many years during his career in systems management and networking.

We programmed the Raspberry Pi with a Python script to detect loss of internet connectivity. The script is executed at configurable intervals using the `cron` job scheduler contained in the Linux

based OS. `Cron` may be configured to run scripts and many other functions at specific times, dates and intervals. We chose every 10 minutes to check for internet connectivity. When internet connectivity is lost, a GPIO pin responds by triggering two relays, one to switch off power to the cable modem and the other to sound a piezo audio alarm, both for 15 seconds. After power is restored to the modem there is a 4 minute delay for it to re-connect to the cable plant and then the Raspberry Pi checks for internet availability again. If we have internet then the script ends. If not, the script executes twice more before giving up. Thanks to cron, the OS calls the script again, 10 minutes after the previous call. Both the script and `cron` function cycle at their programmed intervals until the internet connection is again detected. When internet connectivity is restored, the script adds an "internet restored" entry to a log file and emails the log to us. Fig. 1 shows the logic diagram for Efrain's code. The Python script is available for download.

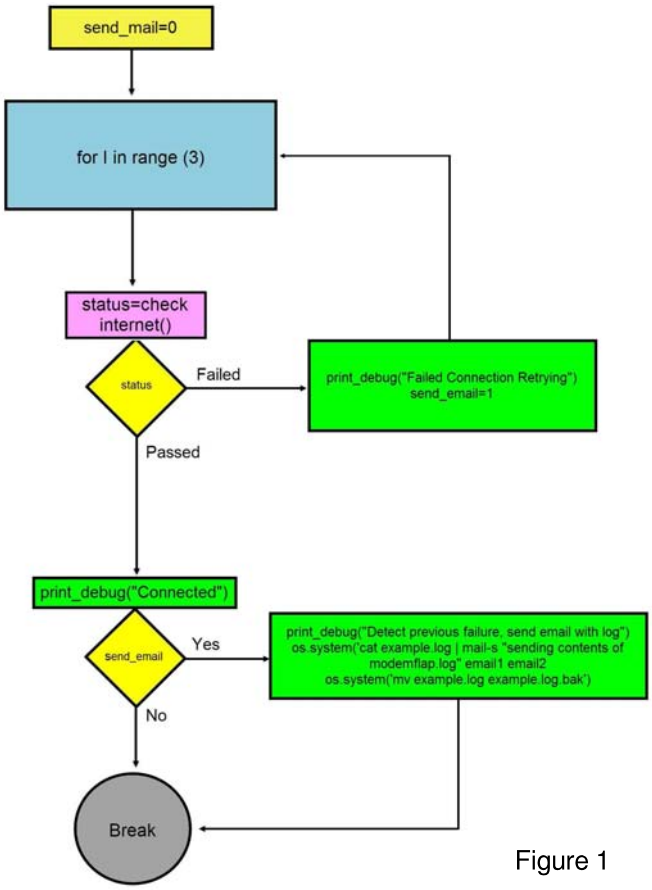


Figure 1

Construction

Much has been written regarding the susceptibility of the Raspberry Pi's GPIO pins to shorts, excess input voltages and current draw.[2] Take care when you make connections to the GPIO header and be sure to double check the pin numbers. There are no labels on the board and two different pin naming schemes. Here we use GPIO header pin # 11 for our output signal and pin # 6 as ground. I found that the female connectors on wire prototype jumpers serve as good push on connectors for the GPIO pins. Don't use either of the two Raspberry Pi power pins (#1 or #2) to supply the relay coils or piezo alarm directly as the pins can supply only 8-16mA. A powered USB hub is a good source of 5V power for both the relay board and the alarm.

The relay board includes diode protection against back current from the relay coils and an opto-isolator to keep voltages from the switched devices from returning to the Raspberry Pi if there's a component failure or misconnection. For safety reasons be sure to switch only the 12V supply to the modem, not the AC mains power supplying the brick or wall transformer. A hardware block diagram is shown in Fig. 2.

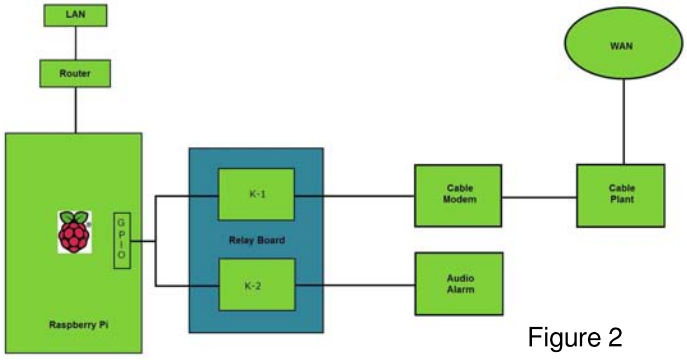


Figure 2

Building instructions

There are many excellent online and print sources for initializing your Raspberry Pi board. Here I'm assuming that the board is configured and connected to the internet.

Test your Raspberry Pi internet connection by pinging a known web site using the `ping` command. If there is no response, type `ifconfig` on the command line to see if you are connected to your DHCP router and confirm that it has assigned an IP address to your Raspberry Pi. If not connected then check the physical

connections between the Raspberry Pi board and your router and power cycle the router to see if it discovers the Raspberry Pi. Later, you may choose to create a static IP address for the Raspberry Pi so you will always know where to find it on your LAN when using SSH for remote access.

When you do establish an internet connection, upgrade and update the Raspberry Pi OS. From the command line type,

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

to download the most recent OS changes. Make a habit of updating each time you import new software into your Raspberry Pi.

Download the Python script for the project from Github http://www.github.com/n6egy/rpi.git and install it into its own directory on your Raspberry Pi. From this directory type:

```
python test_connection.py
```

If you have an internet connection the screen will display a "Connected" message indicating you've got internet access. Next, unplug your network connection and run the script again. You should see a "Not Connected" message on your monitor. Reattach the cable and run the script once more. The success message should reappear.

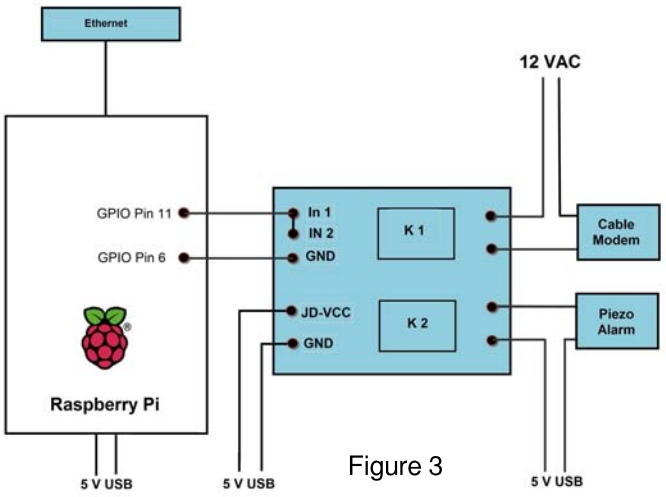


Figure 3

Now construct a "Y" wire to connect GPIO pin 11 to the two separate input pins (IN1 and IN2) on the relay board. Breadboard prototype wires with female connectors fit the pins on both boards. You'll need a soldering iron to make the splices and some heat shrink tubing for insulation. Connect GPIO pin 6 to the input ground on the relay board. (Fig. 3)

Next, cut the positive conductor of the 12V wire from the brick or wall-wart power supply to your cable modem. Strip and tin each wire and insert the ends into the screw terminals on relay #1. Insert one wire end into the centre screw of relay #1 and the other into the normally closed terminal (in this case, it's the one near the edge of the relay board).

If you want a 15 second audio alarm to sound when power is removed from the modem, then cut a length of USB cable with the Type A USB connector still attached to one end. The red and black wires in the cable connect to the positive and negative pins on the USB connector. Strip and tin the wire ends. Cut and secure the other two conductors. Connect the red wire from your piezo alarm to the centre terminal of relay #2 and the positive wire from the USB cable to the normally open relay contact. Connect the black piezo ground wire to the black USB wire. Now, when GPIO pin 11 is energised, both relays trip. The contacts on relay #1 open, removing 12V power to the cable modem and those on #2 close, sending 5V to the piezo buzzer.
(Figs. 3, 4)

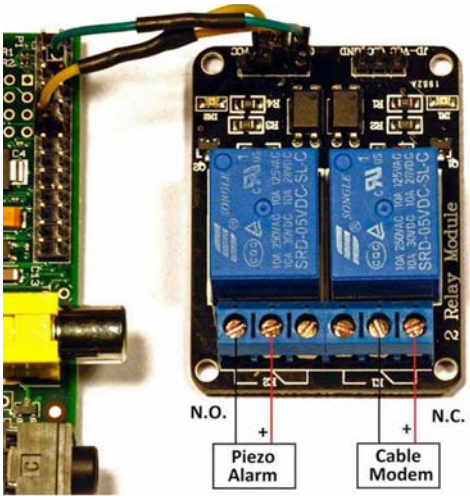


Figure 4

Finally, we must provide power to the relay board itself. Connect the red and black wires of another

USB cable to the “JD-VCC” and “GND” pins on the relay board, remembering to observe polarity of this DC voltage. I found that a standard hobby servo motor (female) connector works well on the relay board power pins. Clip and remove the centre conductor at the plug and splice the two outer conductors to the USB cable, observing correct polarity and connect. (Fig. 4)

Customise the Python script before you install the device. Add your own email address and perhaps a different title. If you like, consult the annotated script and change the duration of the modem shutdown (default is 15 seconds) and the delay in rechecking the connection (default is 240 seconds). You may also want to change the `cron` script to call the Python script at a different interval (default is 10 minutes).

Possible modifications

One modification is to use SMS or Twitter notifications rather than email. Another idea is to place an extra Raspberry Pi with an audio alarm and a blinking LED on your desktop as a remote modem flap alarm.

Remember that after you install the Raspberry Pi and it is working the way you like, nothing we’ve done here prevents it from multi-tasking. You can use it for a data hub, mini web server, or anything else the Raspberry Pi can do and it should still work to mitigate your problem with modem flap.

The Raspberry Pi has been running here for three months in the “headless” mode without problems. (Fig. 5) It is impervious to power outages as when it re-boots, the `cron` timer begins and calls the modem flap script at the programmed interval whether or not you are logged in. I’ve assigned a static IP to the Raspberry Pi so I can login via SSH or by using the PuTTY program on my Windows machine to change parameters, or add new functions.

Have fun with this practical use of the Raspberry Pi. By the way, yes I am going to send a copy of this to my cable provider!

```
15:10:02.383604: Successfully opened URL
15:10:02.396258: Connected
15:20:02.516396: Testing for internet connectivity
15:20:02.617050: Successfully opened URL
15:20:02.619565: Connected
15:30:01.989957: Testing for internet connectivity
15:30:42.055450: ERROR: Not Connected
```

Parts

Raspberry Pi (B) 512MB boards and SD cards are widely available online. As noted by many authors, the Adafruit website, amongst many retailers, provides Raspberry Pi boards as well as excellent tutorials for installing the Raspberry Pi OS and configuring the software: http://learn.adafruit.com/category/learn-raspberry-pi

Other online sources for the RaspberryPi include RS in the UK http://uk.rs-online.com and Element14 http://www.element14.com

The two channel relay board made by Sainsmart is available online at Amazon or direct from the company http://www.sainsmart.com.

The female prototype connectors are available online at Amazon as are three conductor servo connectors (also available at RC airplane and vehicle hobby sites).

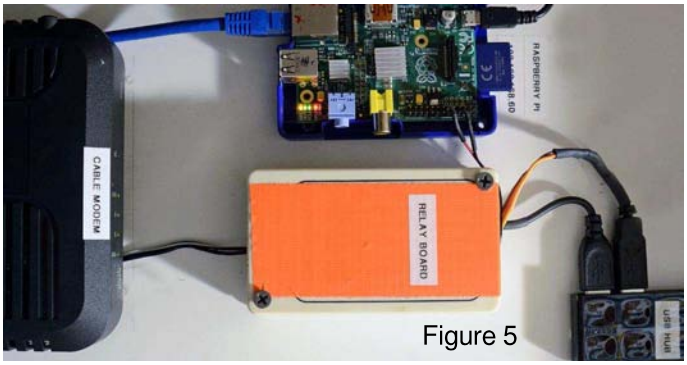


Figure 5

References

[1] Cisco Systems, Cable Modem Dropping Off-line in a 2-way Cable Network, Doc. ID.:22543, Updated Sept. 03, 2006.

[2] Ron Hackett, Have a Piece of Picaxe Pi, Nuts and Volts, Vol. 34 #8, pp 14-21 (Aug 2013). (A great introduction to working with the RaspberryPi and GPIO pins.)

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.

£12.99

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.

£16.99

RTC Pi

Real-time clock with battery backup and 5V I²C level converter for adding external 5V I²C devices to your Raspberry Pi.

£9.75

ADC Pi

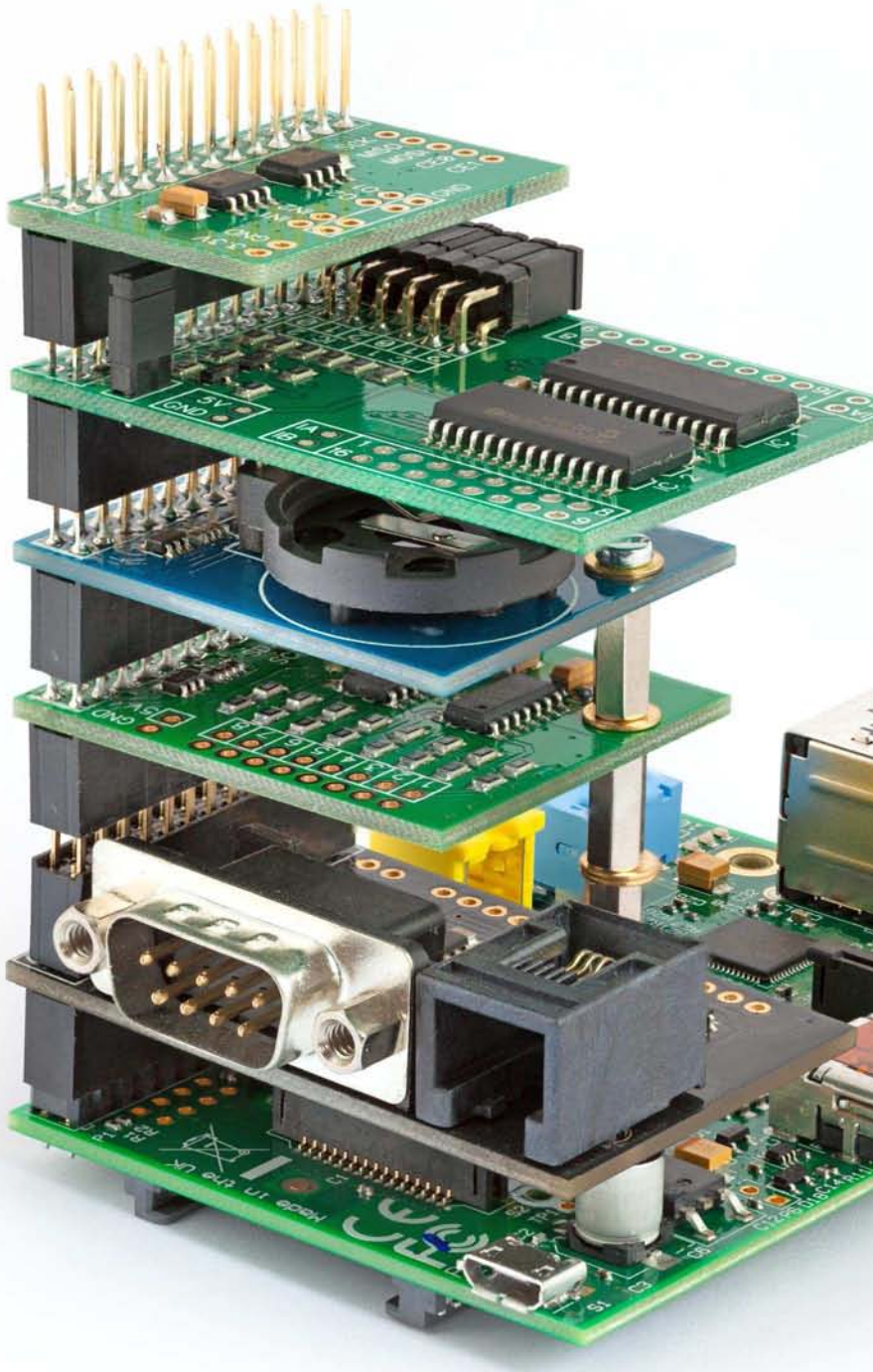
8 channel analogue to digital converter. I²C address selection allows you to add up to 32 analogue channels to your Raspberry Pi.

£17.99

Com Pi

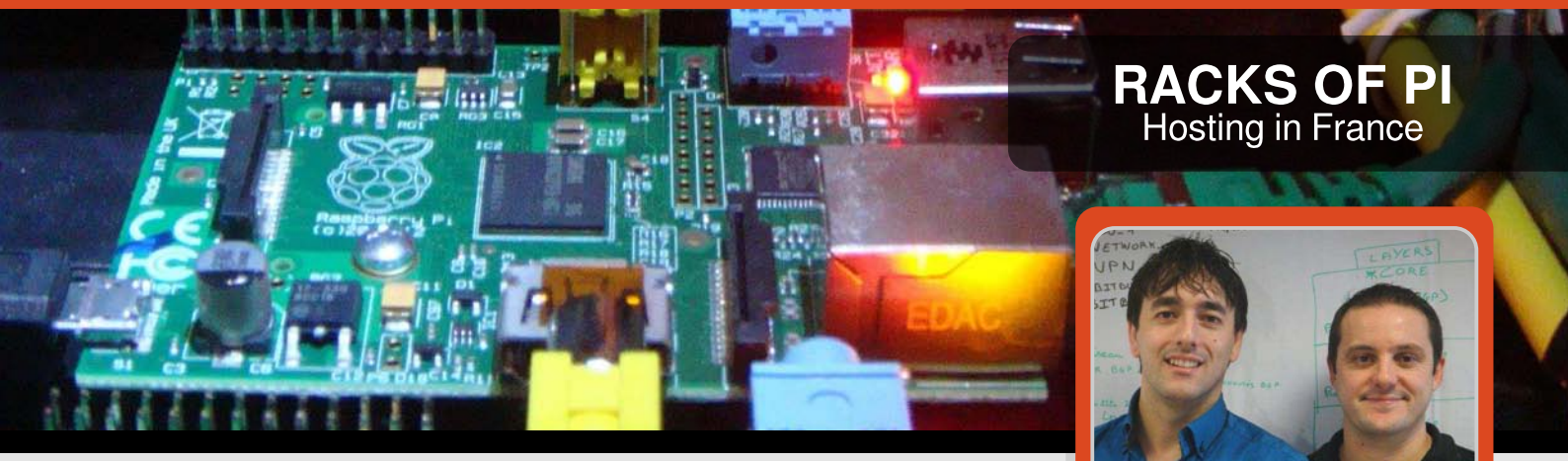
RS232 and 1-Wire® expansion board board adds a serial port to your Raspberry Pi. Ideal for the Model A to enable headless communication.

£19.99



AB electronics UK

www.abelectronics.co.uk



RACKS OF PI
Hosting in France

NanoXion's Raspberry Pi colocation service

Sebastien Fourcade & Florent Ménage
NanoXion

NanoXion is a start-up company that was inspired by the French saying "We don't have oil, but we have ideas", to provide Raspberry Pis in a professional hosting environment. The company decided to take up the challenge of using Raspberry Pis as dedicated servers.

General overview

NanoXion is based in the town of Tarbes in the South West of France. The company is a dynamic team of two talented individuals who are passionate about IT technology. NanoXion specialises in hosting solutions. The company provides all kinds of high availability professional hosting solutions, server administration, maintenance and server housing service in a datacenter.

Raspberry Pi hosting solution

More and more companies are seeking flexible and highly advanced methods to efficiently manage their IT infrastructure. However, common hosting solutions are often not well matched to the required demands of a client, especially for small businesses. The development of IT technologies should include an environmentally friendly approach. The current trend of IT technology development is towards more flexible equipment, to provide high global resource efficiency. IT management must now either fully participate in a "green" approach or risk becoming a weak link in the eco-responsibility chain of a company. Following these aims towards modernisation in the IT hosting business, NanoXion is preparing to launch its own colocation service called NX-BOX. The NX-BOX service is based on Raspberry Pi technology.

Our Vision

The NX-BOX hosting service provides an efficient, dedicated and inexpensive solution. This solution follows sustainable development principles, thanks to the low power consumption of the Raspberry Pi.

We want to stimulate digital development, by offering everyone a secure environment to develop and implement groundbreaking software in this digital world. We also believe that these compact and energy efficient devices can play a very interesting role in a professional hosting environment.

An interesting environment

The Raspberry Pi provides multipurpose solutions in our own server infrastructure, such as managing, task automation and monitoring. The performance of the Raspberry Pi is great. It is environmentally friendly thanks to low power consumption, has a very compact size and performs very well for several specific applications.

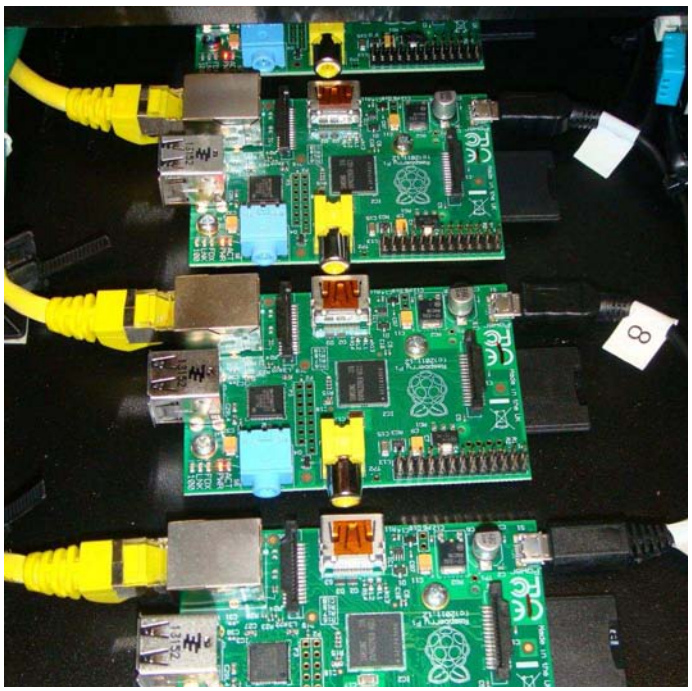
It works and rocks!

There is a lot of enthusiasm around the Raspberry Pi, which can impact society itself. A lot of communities use Raspberry Pis to develop and implement cool stuff. For four months, we have been setting up a complete and rackable solution to equip our rack-mount servers with Raspberry Pis and others compact products.

We think that a Raspberry Pi colocation service is a great choice for running specific applications and to stimulate or improve digital development, for which a "powered-up" server is not necessary.

Some technical features

Currently, NanoXion is the only company in France that provides a Raspberry Pi colocation service directly for consumers. In fact, consumers do not need to send their own Raspberry Pi because we can provide pre-configured Raspberry Pis ready for use.



We have worked out an agreement with our telecom partners to provide Raspberry Pi colocation for around €8 (£6.50) per month. This includes unlimited traffic, ability to reboot 24x7 through our manager software, native IPv6 and pre-installation of Raspberry Pi servers.

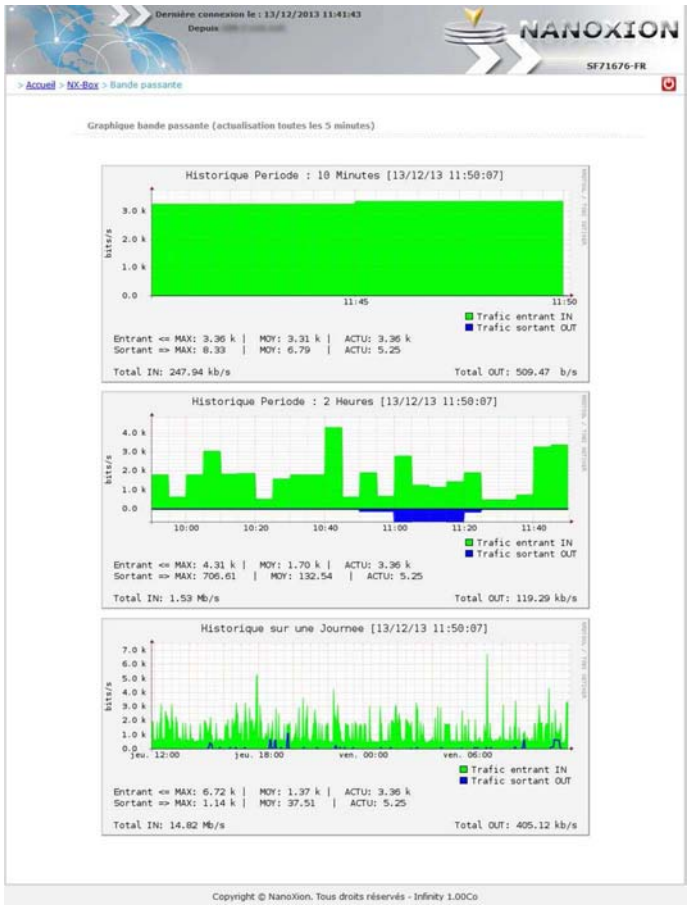
"As you see, we have several Raspberry Pis in a rack. I can tell you that we have made an unprecedented number of sketches, cutting, drilling and assembling steps for this!" says Sebastien Fourcade, NanoXion CEO.

We have developed our own monitoring tools that are not based on proprietary solutions. These tools are written in PHP, Shell,

WLanguage and other web technologies. The whole process of running Raspberry Pis in our computer racks is controlled by our own manager (codenamed "Infinity").

Power supplies have been resized to accommodate our racks. We have completely redesigned our rack infrastructure to save time and improve flexibility. Each rack has two sensors: one for temperature and one for humidity. The temperature sensors are present to make sure that the rack is functioning properly and to provide warnings of emerging risks. We are also very careful with security; tracking MAC addresses, IP addresses and network paths.

We offer our customers a control panel to manage their Raspberry Pis. This panel includes graphs of bandwidth, power management and SSH access. We provide dedicated solutions based on server-oriented Raspbian (Debian Wheezy) distributions, specifically optimised for hosting solutions.



Service level commitment

NanoXion operates with the fibre optic Public Initiative Network (PIN) implemented in the Hautes-Pyrénées. We provide a high quality service, thanks to our secure datacenter that has an uptime of 99.977% (corresponding to an unplanned downtime maximum of 2 hours a year), close proximity of the technical team, a global 24/7 technical support in case of critical issues and proactive infrastructure monitoring. Our secure and reliable professional hosting environment ensures our NX-BOX service has guaranteed quality of service and security. The entire architecture has been redesigned to achieve the best possible performance.

IPv4 vs IPv6

RIPE, the organisation in charge of delegating IP addresses in Europe, ended the distribution of new IPv4 blocks about a year ago. To cope with the imminent shortage of IPv4 addresses, NanoXion can offer your new generation dedicated Raspberry Pi server directly with IPv6.

The IPv6 protocol was developed in anticipation of this shortage, caused by an exponential increase of internet connected devices often called the "Internet of Things". Unfortunately, not all operators and manufacturers have followed recommended standards. Therefore, many of the world's users will not be ready for a full transition to IPv6 for some time. NanoXion provides customers with the choice of running servers either with a "Dual-Stack" configuration (IPv4 and IPv6 addresses) for a normal cost or with a IPv6 only configuration (no IPv4 addresses at all) at a 10% cost reduction.

Coming soon

Our Raspberry Pi colocation service is scheduled to be fully functional at the beginning of February 2014. "NX-BOX" refers to our colocation service, under which there is the "PiBOX" service. More information can be found at http://www.nx-box.net.

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

Cambridge Raspberry Jam (CAMJAM)

When: Saturday 8th February 2014, 10.30am until 12.30pm
Where: Institute of Astronomy, Madingley Road, Cambridge, CB3 0HA, UK

Morning and afternoon sessions including a programming workshop in the morning. http://www.eventbrite.co.uk/event/10043471293 and http://www.eventbrite.co.uk/event/9862701606

Malvern Raspberry Jam

When: Tuesday 11th February 2014, 2.45pm to 5.00pm
Where: Wyche Innovation Centre, Walwyb Road, Malvern, WR13 6PL, UK

An after-school community club for local students in primary and secondary education in the Malvern area who are interested in Raspberry Pi computing. http://www.eventbrite.co.uk/event/10077559251

Raspberry Jam Silicon Valley

When: Third Saturday of every month, 1.30pm until 4.30pm (PDT)
Where: Computer History Museum, 1401 North Shoreline Boulevard, Mountain View, CA 94043

Free admission, suitable for all. Will feature demonstrations of application software suitable for science, technology, enginerering and maths education. http://www.eventbrite.com/event/8469381147

OCR Raspberry Jamboree

When: Thursday 28th February to March 1st March (see web for times)
Where: Manchester Central Conference Centre, Windmill Street, Manchester, M2 3GX, UK

The second international convention sharing the educational potential of the Raspberry Pi computer. http://www.eventbrite.co.uk/event/8546357385

Southend-On-Sea Raspberry Jam

When: Saturday 1st March 2014, 10.00am until 5.00pm
Where: TAP (Old Water Works), North Road, Southend-on-Sea, SS0 7AB, UK

An open day with Raspberry Pi's. Programming and using simple electronics: LEDs, resistors and other components, driven by GPIO. http://www.eventbrite.com/event/10237355205

Learning to program video games

Yann Caron

Guest Writer

SKILL LEVEL : BEGINNER

Two years ago my 10 year old son asked me, "Daddy, what is your job?". I replied that I write computer programs which give the computer some tasks. I didn't have to wait long for the second question. "Ok, but how do you tell the computer what it has to do? How does a program work?".

That was when I decided to write Algoid to teach him what computer programming is and how to do it. Originally Algoid was designed to run on the Android platform but recently I discovered the Raspberry Pi; this low cost, credit card sized computer exactly designed for education.

What is AlgoIDE? AlgoIDE is a combination of an integrated development environment (IDE) and its own language (AL) designed together to simplify the understanding of programming!

Principles

The AL language was developed with three ideas in mind:

1) Its apprenticeship should be as progressive and memorisable as possible.
Try to explain to a kid how the "for" loop works. You will discover you need to explain variables, conditions, boolean expressions and incrementation... all just to explain a simple loop!

Algoid's syntax was designed to be as easy to memorise as possible. All its structure statements are written the same way (variables, functions, arrays, objects ...)

2) Its syntax should be as close as possible to industry standards.
Learning a language represents a large amount of time so the day the kid wants to use Java, Python or C++ they will not have to learn again unfamiliar syntax and another nuances.

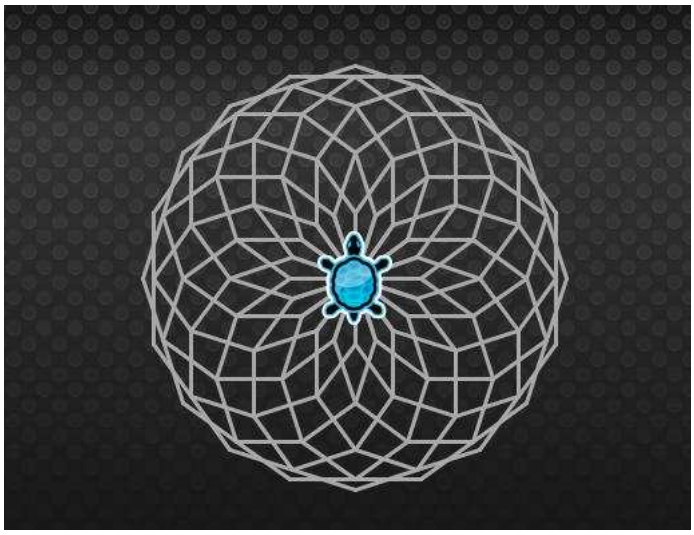
3) Its semantic should not be limited to one or two paradigms.
When I was young I was very frustrated by the BASIC language. After having understood how imperative statements and procedures work, I was limited to them. To progress I had to learn another language and another syntax.

All things considered, AlgoIDE tries to simplify programming and its understanding. My first idea was to create a step by step execution mode. With two clicks the execution slows down and highlights appear in the source code on the line being executed. This is possible with the debugging mode. We will see later how it works. AlgoIDE also has the ability to view the variables in use on its scope viewer and has its own logger.

Algoid quick tour

AlgoIDE is available on the Pi Store at http://store.raspberrypi.com/projects/algoid. It is also available on the Google Play store at https://play.google.com/store/apps/details?id=fr.cyann.algoid. Finally it is also available for Java at http://download.algoid.net.

Install it on your Raspberry Pi and launch it from the Desktop. From the `Examples` menu, choose `demo.al`. Click on the "Play" button and the demo script will run.



The program contains a function named `poly` and a loop that draws a rose window. Note that the function declaration is really special:

```
set poly = function (size, n) {
} ;
```

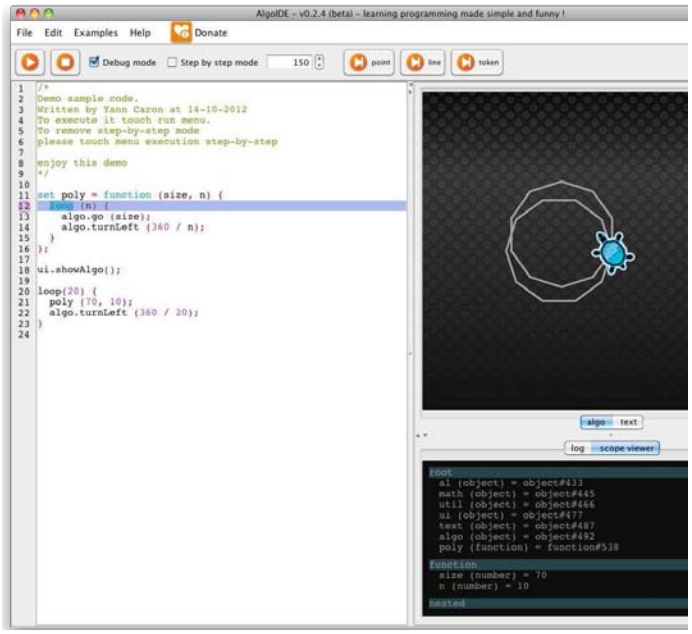
In AL everything is an expression. Example declarations are:

```
set size = 7;
set o = object () {
} ;
```

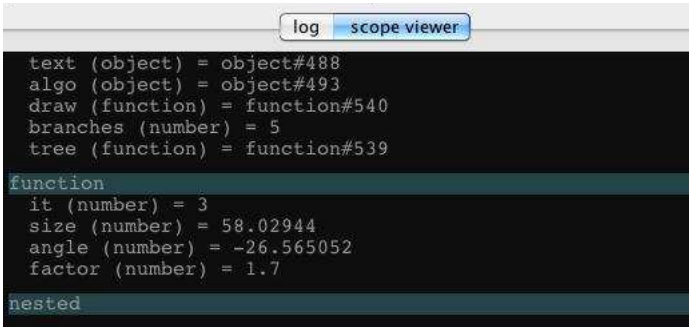
In AlgoIDE, check the "Step by step" box and click the "Play" button again. Now you can understand where the program is at and what it is doing. The number at the right of the check box allows you to change the step by step speed.

Now uncheck step by step mode and check the "Debug mode" box. Click on the number 12 in the source code margin. This will add a new breakpoint. Click on the "Play" button to run the program.

Three new buttons appear - run to the next breakpoint, run to the next line and run to the next token. The Algoid debugger considers each token as an object path and expression calculation. This is perfect to understand the precedence order for example. Try each of the three new buttons.



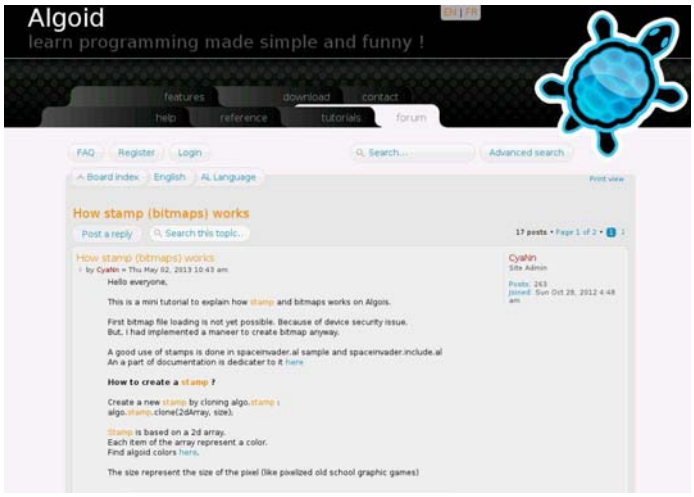
You can observe the current variables and their values in the Scope Viewer window. For example debug the binary tree program (`Examples > fractal > binary tree.al`) and set a breakpoint on line 12. Observe the value of `it` and `size` in the Scope Viewer with each iteration.



Documentation

To learn a language and how to program, the most important thing is documentation and sharing experiences. The http://www.algoid.net web site contains a complete language reference section and a continuously updated tutorial list. The tutorials

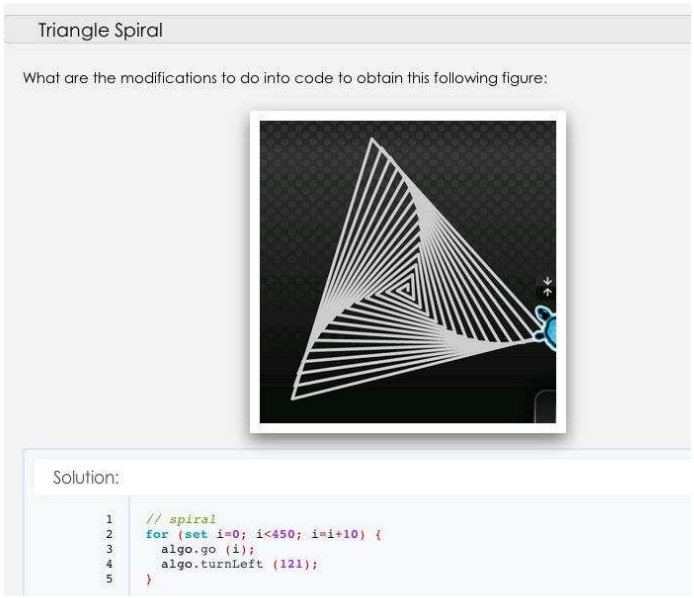
contain a lot of examples and progressive exercises to help explain each language statement and function.



Algoid is also a community federated by the forum http://forum.algoid.net. It contains a lot of discussions, problem solutions and mini-tutorials.

Another way to learn programming is with other people! Some members have created a team to develop games on Algoid. When programs are interesting they are included in the next application release. Everyone can learn from one another. Everything in the Algoid environment is based on sharing source code and knowledge.

Teaching



Algoid is also designed for teachers. As Algoid is a free tool, all help is greatly appreciated for tutorial writing, documentation corrections, etc.

Algoid gives teachers the ability to write their own learning plugins. A tutorial and an article explaining how to do that is already available. Writing a plugin in Algoid is very simple. The teacher just needs to have some basic Java and Swing development knowledge.

Plugin development is divided in two steps:

1) Develop the library with objects and functions needed for the course.
2) If necessary, develop a new Java panel (view) and bind it with the library.

A complete range of tools are available to interact with the AL language. The tutorial provides a complete Java NetBeans project starter and documentation.

This is an example of the library development:

```
// show panel method
PluginUtils.addMethod(my, "show", new PluginUtils.Behaviour() {
    @Override
    public void visite(Block<RuntimeContext> t, RuntimeContext c) {
        appContext.getIDEPanel().showPanel(myPanel.getName());
    }
});

// setter method
PluginUtils.addMethod(my, "setText", new PluginUtils.Behaviour() {
    @Override
    public void visite(Block<RuntimeContext> t, RuntimeContext c) {
        String text = PluginUtils.getParam(t, 0).getString(); // get the text parame
        myPanel.setFieldText(text);
    }
}, "text");

// event method
PluginUtils.addMethod(my, "onClick", new PluginUtils.Behaviour() {
    @Override
    public void visite(Block<RuntimeContext> t, final RuntimeContext c) {
        final FunctionInstance f = PluginUtils.getParam(t, 0).getFunction(); // get
        // register to button action listener
        myPanel.addButtonActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // create callback parameter to send
                MutableVariant text = new MutableVariant(myPanel.getFieldText());
                // call callback with parameters (if necessary)
                PluginUtils.callFunction(appContext, c, f, text);
            }
        });
    }
}, "f");
```

This is the resulting the example panel:

Once complete, just copy the `<plugin>.jar` file to the `Algoid/plugin` folder and that's it!

With this feature a lot of improvements can be added to Algoid, like GPIO communications, a true 2D game engine (planned for future development), mathematical functions, etc. Everything you want to teach or command can be added with this powerful scripting language!

Video game development

One principle of Algoid is to be fun, because when it is fun, learning becomes a pleasure. What is more fun than creating your own game?

The game above contains only 300 lines of source code and this include all the graphics and animated stars. You can play it from `Examples > games > star battle.al`.

Tutorials

In this simple tutorial, we will see how to animate a spaceship. Then features such as typing, stamps and events will be discussed.

The first steps of the program are to hide the turtle, create a spaceship and draw it.

```
algo.hide();

// ship
set ship = object () {

  // define its position
  set x = 0;
  set y = 150;

  // draw the ship
  set stamp = algo.stamp.clone (
    array {
      {-1, -1, -1, 10, -1, -1, -1},
      {-1, -1, -1, 10, -1, -1, -1},
      {-1, -1, 02, 10, 02, -1, -1},
      {10, -1, 02, 10, 02, -1, 10},
      {10, 02, 02, 10, 02, 02, 10},
      {10, 02, 10, -1, 10, 02, 10},
      {10, 02, 02, -1, 02, 02, 10},
      {10, -1, -1, -1, -1, -1, 10},
    }, 7
  );

  // draw it
  set draw = function () {
    algo.goTo (x, y);
    stamp.draw ();
  };
};
```

Note how objects are defined:

```
set ship = object () {};
```

In AL everything is an expression. An object is defined in the same way as a number or a string. Objects, functions and arrays are all declared using the same expression syntax:

```
set [name] = [something];
```

In the case of objects, parenthesis are used for inheritance and {} are used for members. Then to define members of an object, all you need to do is to define another expression within the object:

```
set o = object () {
  set attribute = 7;
  set method = function () {
    // do something
  };

  set nested = object () {
    set method = function () {
    };
  };
};
```

The semi-colons in this code are optional.

Note: In AL, objects are directly created with their declaration. An object can be used imediatelly after it is created. If some more objects of the same class are needed, it is possible to clone the initial object.

The next concept to introduce is a stamp. A stamp works by cloning (creating a new instance of an

existing object) of the `algo.stamp` object.

The syntax begins with `set`:

```
set stamp = algo.stamp.clone ();
```

A stamp contains two pieces of information: a two dimensional array of colours and the size of a pixel. In AL, an array is declared using the array keyword.

```
set a = array {1, 2, 3, 4};
```

A nested array does not require a keyword.

```
set a = array {1, {2, {3, 4}}, 5, {6, 7}}
```

This example represents a tree. It demonstrates how simple it is to define data structures in AL. The AL language is functional and uses the power of map / filter / reduce, similar to Python.

Numbers can be replaced by a function, or objects and array of functions, etc. Imagine the capabilities of this kind of flexibility.

Now the image needs to be drawn on the screen.

There are two steps to draw a stamp:

```
algo.goTo(x,y);
stamp.draw();
```

If we draw the stamp again in another position, then AL will draw an additional ship. This will create two stamps on the screen. To create an animation, we will limit the number of simultaneous stamps on the screen.

```
algo.setStack (1); // only one stamp in stack
```

The object can now be animated with an event:

```
util.pulse (function () {
  ship.draw();
}, 40);
```

AL is a complete functional language. Functions can be passed as parameters and returned as results. Events are based on a callback mechanism. If a function is passed as a parameter, then it is only called when necessary. In this case, every 40 milliseconds.

Up to this point, nothing is moving. To show some animation, let us modify the position each time like this:

```
util.pulse (function () {
  ship.y--; // Update position
  ship.draw();
}, 40);
```

Now we will add the ability to move the ship according to the mouse position with the event `algo.onMove()`.

```
algo.onMove (function (x, y) {
  ship.x = x;
});
```

Note: The mouse coordinates are given in the x and y parameters of the callback function. Thus it is possible to connect the ship x and y coordinates to the mouse x and y coordinates.

That's it! Now you are able to move a stamp in Algoid. This example can be used as the basis of a complete game.

The steps of the program are:

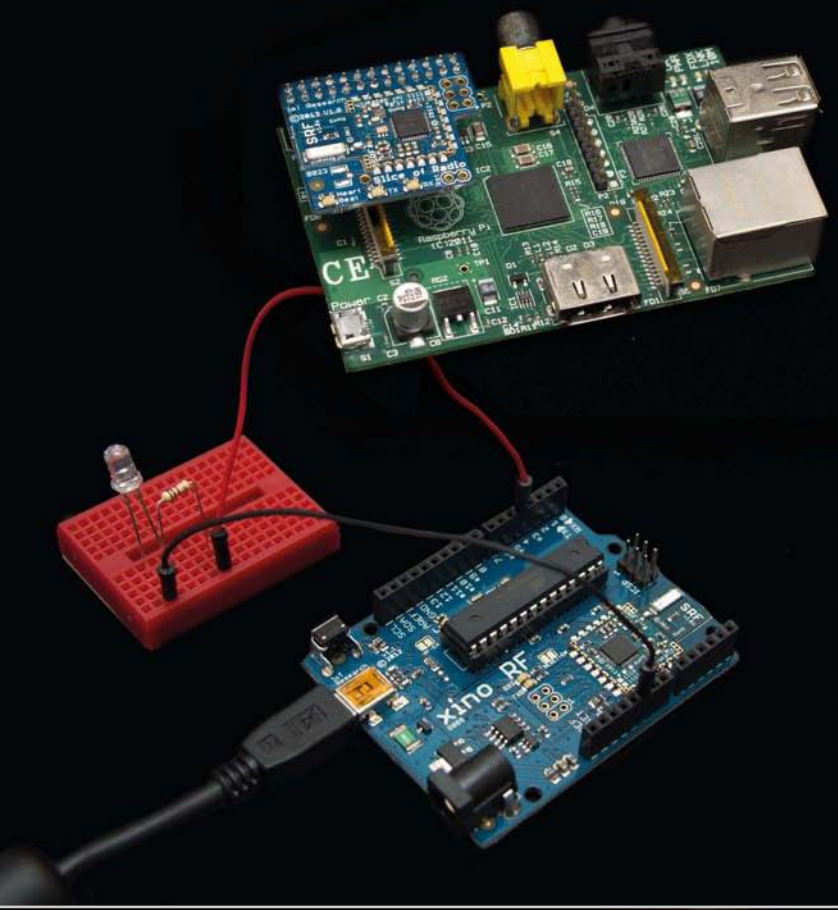
1) Define objects, their coordonates, their graphic stamp(s), their draw method etc.
2) Hide the turtle.
3) Determine how many stamp should be drawn simultaneously.
4) Create frame drawing with `util.pulse`.
5) Create behaviours with Algoid's events like `onMove`, `onClick` (or `onTouch`, `onGravity` and other sensors on Android).

Conclusion

Algoid is a great free tool, to learn or to teach programming. For further questions, suggestions or ideas, you can contact me at cyann74@gmail.com or try the forum at http://forum.algoid.net.

RasWIK

Wireless Inventor Kit for the Raspberry Pi ™



Contains 88 Parts

- Starter examples require no soldering at all
- Plug in wires and breadboard to make building easy and fast
- A pre installed and configured Raspberry Pi OS makes using your Pi a breeze
- Examples you build, can be mixed with our out the box wireless devices...how cool!
- **"It provides possibly the simplest platform for experimenting with wireless sensor networks I've ever seen."**
  (Gareth Halfacre, CustomPC, Issue 121, Oct 2013)
- Made in the UK

£49.99

www.ciseco.co.uk

Raspberry Pi is a trademark of the Raspberry Pi Foundation

CISECO
Powering creative minds

Pi NoIR - The new high definition infrared camera module from Raspberry Pi

Featuring the same 5 megapixel image sensor as the standard Raspberry Pi camera with the infrared cut-off filter removed to enable IR light frequencies, the Pi NoIR is compatible with Raspberry Pi model A & model B.

- Extended spectral range
- 5 megapixel image sensor
- Still picture resolution: 2592 x 1944
- Video resolution: 1080p 30fps

Raspberry Pi

Available now at www.rs-components.com/raspberrypi

RS

Logging data from digital & analog sensors - Part 1

W. H. Bell

MagPi Writer

SKILL LEVEL : BEGINNER

Scratch is a versatile tool that includes many concepts found in other programming languages. While Scratch provides a lot of useful syntax, being able to control input and output (I/O) devices can be problematic. Thankfully, Scratch can act as a server that sends and receives messages to clients on the same network. This means that C or Python programs can be interfaced with Scratch, to provide additional functionality to low-level hardware and other protocols.

Several schools have expressed their interest in a highly flexible Scratch interface, that allows easy addition of other I/O components. The interface needs to be flexible enough to allow teachers to add additional devices as needed. To address these requests, the RpiScratchIO https://pypi.python.org/pypi/RpiScratchIO/ Python package has been written. This article is the first in a series on RpiScratchIO and provides a basic introduction.

Installation

The first step in the installation is to install the dependencies. Open a terminal window and type,

```
sudo apt-get install -y python-setuptools python-dev
sudo easy_install pip
```

Then install RpiScratchIO and its Python dependencies by typing,

```
sudo pip install RpiScratchIO
```

To enable the SPI bus, used in the last example in this article, edit /etc/modprobe.d/raspi-blacklist.conf with `sudo nano /etc/modprobe.d/raspi-blacklist.conf` and comment out the spi kernel module, e.g:

```
#blacklist spi-bcm2708
```

Then load the SPI module with the `modprobe` command given below or reboot the Raspberry Pi to load it.

```
sudo modprobe spi-bcm2708
```

Starting Scratch & RpiScratchIO

Before starting RpiScratchIO, Scratch must be launched and setup correctly. Start Scratch from the desktop icon or from the menu. Then click on the "Sensing" menu and right click somewhere the "sensor value" text, near to the bottom of the tool palette. Then select "enable remote sensor connections". Once a program has been saved, this option is remembered as part of the program. By default in the current version of Raspbian the remote sensor connections are enabled in the menu, but the server process is not running. In this case, select "disable remote sensor connections" by right clicking on sensor value. Then click again and enable "remote sensor connections". When the server process starts, a dialog box appears to report the change. The same dialog box appears when a program that includes remote sensors is loaded, signalling that the server process has started.

To use the low-level GPIO, the RpiScratchIO program needs to be run as the root user. (For non-GPIO devices, it may be possible to run as a normal user.) To launch RpiScratchIO as the root user type:

```
sudo RpiScratchIO [<name of configuration file>]
```

If no configuration file name is given, then RpiScratchIO will try to read RpiScratchIO.cfg from the present working directory. The configuration file contains the list of devices, matching the physical wiring of components to the Raspberry Pi. If Scratch is not running or the remote sensor connections are not running in the Scratch session, then RpiScratchIO will produce an error message and exit. Once RpiScratchIO has started all of the devices listed in the configuration file will be made available within the Scratch session. To stop the RpiScratchIO client type CTRL-C. This will cleanly close any device connections.

Simple GPIO

Use nano to create a RpiScratchIO.cfg file, with the complete list of GPIO pins:

```
[DeviceTypes]
GPIO02 =
GPIO03 =
GPIO04 =
GPIO07 =
GPIO08 =
GPIO09 =
GPIO10 =
GPIO11 =
GPIO14 =
GPIO15 =
GPIO17 =
GPIO18 =
GPIO22 =
GPIO23 =
GPIO24 =
GPIO25 =
GPIO27 =
```

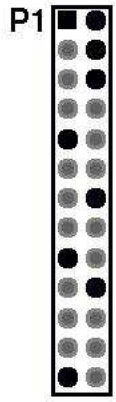
The GPIO devices does not need to be assigned a specific type of object, since RpiScratchIO will automatically recognised it as a basic GPIO connection. Therefore nothing needs to be given to the righthand side of the equals sign. More information on the configuration file format is given in the online documentation for RpiScratchIO.

Open Scratch, start the remote sensor connections and then run RpiScratchIO as the root user. This will cause the GPIO BCM numbers on the 26 pin connector of a Model B version 2 or Model A Raspberry Pi to become remote sensors. The RpiScratchIO program then waits and listens for commands from Scratch. Once RpiScratchIO is running, a broadcast message containing

```
GPIO14:write:1
```

will cause GPIO14 to be used as an output and set high (3.3V). When RpiScratchIO

receives this message, it will also set the sensor value for GPIO14 to high. Therefore, the sensor value can be used to check that the hardware was sent the output command successfully. In this example program, one sprite was created per GPIO pin to test each GPIO output in turn. The names of the sprites were chosen to match the names of the pins on a Model B version 2 or Model A Raspberry Pi board http://elinux.org/RPi_Low-level_peripherals.



Global variables

Updating a global variable that has the form "device name:channel", will call the write function of the class corresponding to the device name. If the device in question only has one channel, then the channel number can be omitted in the variable name. If the channel number is omitted, then channel number zero is used by default. Here is a simple example, that updates a single GPIO pin with a global variable:

```
[DeviceTypes]
GPIO23 =
```

The example is composed of a configuration file and a Scratch script. As mentioned in the previous sections, RpiScratchIO should be started after Scratch has been started and configured with remote sensors enabled. Try clicking on the big button to change the GPIO output voltage to 0V or 3.3V.

Triggering Scratch

Checking if a status variable or external device is set to a given value is called polling. If a switch is polled many times, then the processor will spend most of its time in an I/O loop. Alternatively, the polling frequency could be reduced. However, if the polling frequency is too slow, then the moment when the switch is closed might be missed. A much better solution is to request that the hardware tells Scratch that

something has changed. To configure a GPIO pin to do this, a `config:callback,falling` broadcast message is sent to RpiScratchIO. There are three callback states that can be chosen: rising, falling or both. When the state of the GPIO device changes, the GPIO device sends a `trig` broadcast message back to Scratch. In this example, when `GPIO23:trig` is received the dragon costume will change. The other `config:in,pullup` broadcast message is present to enable `GPIO23` as an input that uses an internal pull up resistor. If a switch connected between `GPIO23` and one of the ground (GND) pins is closed, it will cause the `GPIO23` input value to go from high (3.3V) to low (0V).

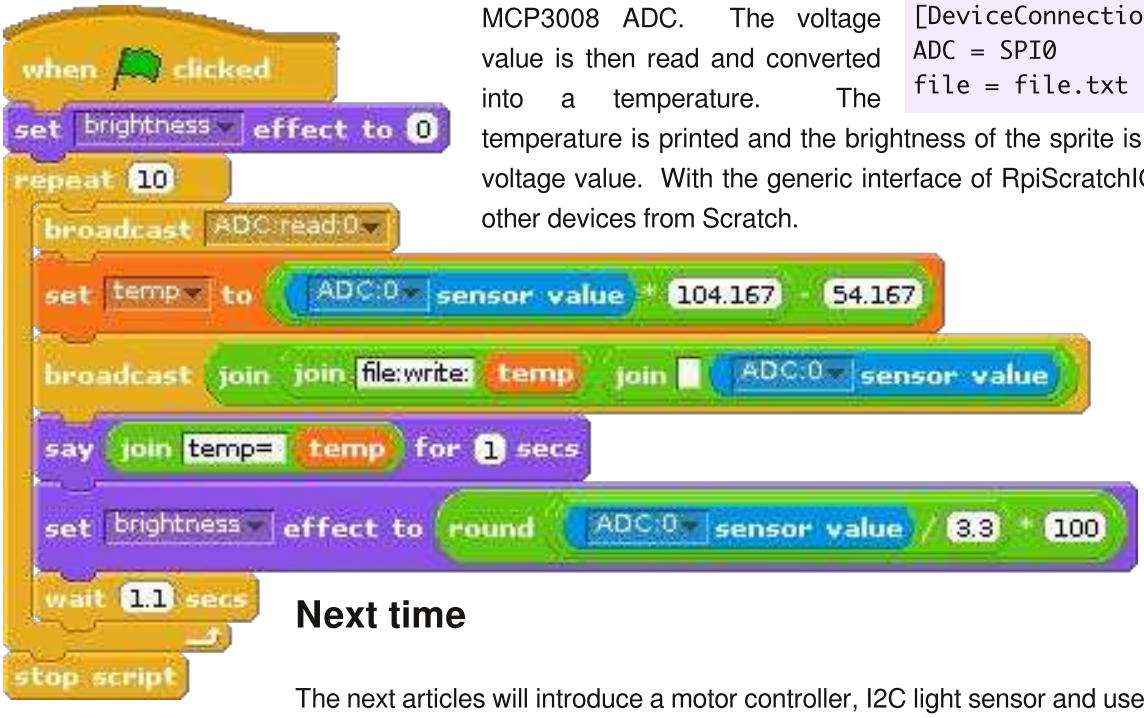
Analog sensors and files

There are lots of easy to use off-the-shelf components that can be added to the Raspberry Pi. The MCP3008 http://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf is a 10bit 8 channel SPI ADC that can be used to read the values of analog sensors. For this example, the MCP3008 was connected to a TMP36 temperature sensor, http://www.analog.com/static/imported-files/data_sheets/TMP35_36_37.pdf While temperature data can be used to make a robot sprite change colour, it is perhaps more useful to record the data for analysis afterwards. For this reason, RpiScratchIO contains a `FileConnection` device. Following previous examples, create a configuration file:

```
[DeviceTypes]
ADC = MCP3008()
file = FileConnection()

[DeviceConnections]
ADC = SPI0
file = file.txt
```

Then write a program to record the temperature values. In this example, a broadcast message is sent to RpiScratchIO, to read the first channel of the MCP3008 ADC. The voltage value is then read and converted into a temperature. The temperature is printed and the brightness of the sprite is updated using the voltage value. With the generic interface of RpiScratchIO, it is easy to use other devices from Scratch.



Next time

The next articles will introduce a motor controller, I2C light sensor and user defined devices.

Bake your own Raspberry Pi filling - Part 2

Martin Kalitis

Guest Writer

SKILL LEVEL : INTERMEDIATE

In the previous article in Issue 15 of The MagPi, we learnt a little about what it takes to build a custom Linux version for the Raspberry Pi and also configured our Ubuntu Linux desktop ready to compile software for the Raspberry Pi. With that done we can now proceed to get the necessary build tools and start down the path of compiling a small Linux operating system.

In this article we will create a bootable Linux SD card for our Raspberry Pi that will boot in around five to ten seconds. The tool that we will use to help us make our software that will be copied onto the SD card is called BuildRoot. This tool not only looks after compilation of the various source code elements, but also provides the ability to customise the software to our needs. Before we begin, ensure that we are in the home directory,

```
cd ~
```

and then download the BuildRoot tool:

```
wget \
http://buildroot.uclibc.org/downloads/
buildroot-2013.11.tar.bz2
```

Next, we need to unpack the download and change to the newly created directory that contains the Buildroot system. We do this with the following two commands:

```
tar -xjf buildroot-2013.11.tar.bz2
cd buildroot-2013.11
```

BuildRoot is capable of building Linux for many different types of hardware; we however are only interested in the Raspberry Pi. Therefore, load the default configuration for our board using the following command:

```
make rpi_defconfig
```

With the default configuration for the Raspberry Pi done we also have the ability to further customise the system we are building. This step can be simple or complex, depending on what additional pieces of software we want to be installed on the new system, along with the level of other configuration each piece of software needs in order to work correctly. Fortunately we have access to a menu system that makes the management of this process a little easier. This is started with the following command:

```
sudo apt-get install ncurses-dev
make menuconfig
```

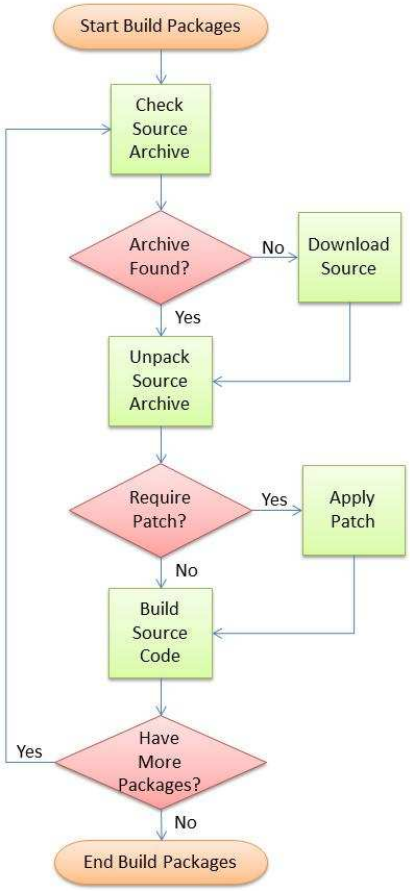
This brings up a menu that allows for control of almost every aspect of the new operating system. We will begin by customizing the message that is displayed with the login prompt.

Select System configuration from the initial menu then select System banner. Delete the existing text and add in the text you would like to see. In my example I used The Raspberry Pi Rocks. Next save this configuration by pressing <Tab> to highlight the Save option and press <Enter> to accept all the defaults.

With the additional configuration completed, we are all set up to actually download and build the software ready for the Raspberry Pi. The BuildRoot tool does this by first working out the order of which the software is to be built, known as dependency checking. Then for each of the software elements it does the following:

1. Download the source code from the internet
2. Download any patches (fixes)
3. Unzip the source code
4. Apply any patches found
5. Compile the software and once complete copy it to the correct location ready for use with the new operating system.

Note that the download process is done once. All subsequent builds that are done will skip the download as we have already downloaded the required packages.



To start this process, use the following command:

```
make
```

This will take some time to complete, since there is much to be done. Depending on the speed of the PC you are using, and the internet connection, this could even take overnight to finish. Once complete, the final task is to copy the brand new operating system across to a SD card.

The process for this begins by un-mounting the existing SD card partitions, since Ubuntu always mounts them. Next we fill the very first part of the card with zeros wiping out the boot and partition information. Then the new partition table needs to be created and formatted. The Raspberry Pi requires that the first partition, commonly known as the boot partition, is a FAT disk format. Finally the newly created binary software that was just built is copied onto the card and the card is un-mounted. A script is available which makes this

task easier. Download FlashRasPiSDCard.sh at:

```
wget http://www.themagpi.com/resources/
BareMetal/FlashRasPiSDCard.sh
```

The script is called with the name of the device that is attached to your SD card. To use it, first copy it to the buildroot-2013.11 directory. Then make sure that the correct permissions are set on the file to allow execution as follows:

```
chmod 0755 FlashRasPiSDCard.sh
```

Before we use this script, we first must discover the name of the SD card device. This is a really important step, since entering in the wrong disk device can lead to you destroying your Ubuntu Linux installation! Before inserting the SD card into your computer type:

```
df
```

This will give a list of devices that are currently mounted on your computer. The result should look something like this:

```
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda1      236179500 3842064 220333532   2% /
none                   4       0         4   0% /sys/fs/cgroup
udev             1949312       4   1949308   1% /dev
tmpfs             391772     792    390980   1% /run
none                5120       0      5120   0% /run/lock
none             1958852     152   1958700   1% /run/shm
none              102400      56    102344   1% /run/user
/dev/sdb1      240176648 4638336 223331364   3% /home
```

Note down the devices starting with /dev in the first column of the list. In this example we have /dev/sda1 and /dev/sdb1. We must not use these names with this script, since this is where your Linux system is running from. Next, insert the SD card into the card reader and type the following:

```
dmesg | tail
```

You should see something like the text below on the screen:

```
[  522.669182] sd 2:0:0:0: [sdc] 15523840 512-byte logical blocks: (7.94 GB/7.40 GiB)
[  522.671422] sd 2:0:0:0: [sdc] No Caching mode page present
[  522.671427] sd 2:0:0:0: [sdc] Assuming drive cache: write through
[  522.675174] sd 2:0:0:0: [sdc] No Caching mode page present
[  522.675178] sd 2:0:0:0: [sdc] Assuming drive cache: write through
[  522.684320]  sdc: sdc1 sdc2
```

What we are looking for is the name of the device between the square brackets; in this case it is sdc. This is the name of the device to which you will be writing the newly created Linux installation with the script file.

Now we have the name of the SD card we can write the bootable Linux system we have created. I am using /dev/sdc as my SD card location, as this is what we discovered when we plugged the card in. Notice that we have prepended the name /dev/ to the beginning of the device name, since the script requires the full path name to the SD card.

```
sudo ./FlashRasPiSDCard.sh /dev/sdc
```

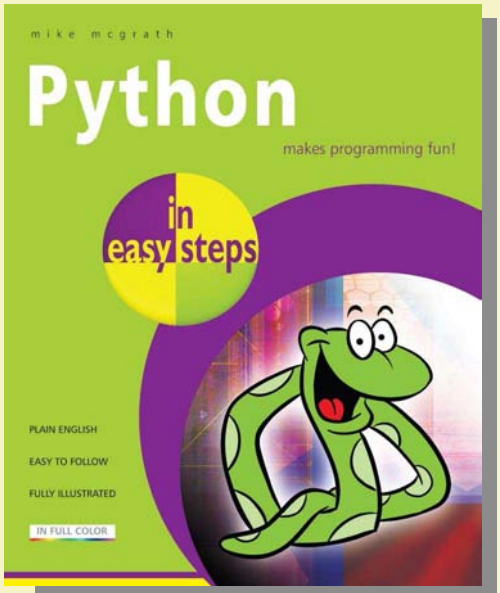
Upon running the script, you will be asked to confirm that you want to proceed. If you enter <Y> then the process continues and generates the card ready for use.

All that is left is to unmount and remove the card, plug it into your Raspberry Pi and plug in the power. When it has booted, use the username of root to log in.

Python In Easy Steps

Mike McGrath

In Easy Steps



Python in easy steps teaches you the basics of the Python programming language in simple and digestible chunks.

This friendly guide begins by explaining how to install the free Python interpreter and get started, so you can create your own executable programs from day one!

Following the useful introduction, it then covers all the Python language basics before moving on to Object Oriented Programming (OOP) and CGI scripting to handle web form data. The book concludes by showing you how to create and deploy graphical windowed applications.

Complete examples, with colourised source code that can be downloaded, are provided throughout - making your learning even easier! The book also includes useful colour icons to let you know when there is a hot tip to spice up your learning, flag something to remember, or ward you away from potential dangers.

Python in easy steps is an ideal guide to help you get to grips with Python, or as a reference guide to help you to remember the things you have learnt already. This is recommended as an indispensible guide for those exploring the Raspberry Pi.

The MagPi and Bookaxis are pleased to offer readers a 30% discount on these two titles until February 28th. To claim, order from www.bookaxis.com/magpi and quote promo code MAGPI20.

Raspberry Pi Networking Cookbook

Rick Golden

Packt Publishing

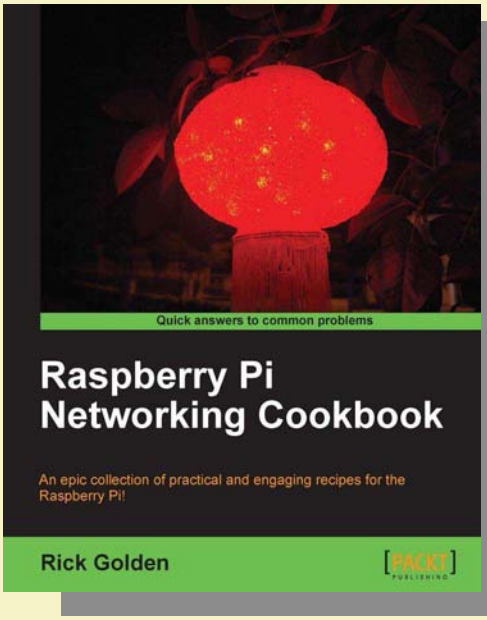
The Raspberry Pi is more than just a platform for teaching students how to program computers!

The recipes in this book show you how this inexpensive little computer can be used for a number of practical solutions that utilise existing networks on connectivity, whilst at the same time teaching you some very useful skills for any hobbyist or network administrator.

It is full of practical and engaging content designed to expand and build upon your existing skills as you work through the individual recipes contained inside.

With this book, any computer novice can learn how to become a Raspberry Pi and networking expert without any previous programming experience of knowledge required!

The book starts with instructions on setting up your Raspberry Pi and configuring it properly to ensure you are working securely. It then moves on through a number of subjects including - remote access to and from other computers, using your Raspberry Pi to securely share your documents, learning how to deploy a web server capable of serving your own content, building your own wireless access point and even a firewall!



This book can either be read from cover to cover or used as an essesntial reference companion to your Raspberry Pi... why not pick up your copy today at a considerable discount?

Feedback & Question Time

You guys are doing good work. Continue!

Victor Ramamoorthy, Ph.D

I just wanted to let you know how much I loved the article on the quadcopter (issue 19). I am a complete novice and have not gotten to the level of this project yet, but it is my ultimate goal to complete it. I just want to thank you for this article/tutorial.

My goal is to make a completely autonomous quad that can fly from one place to another with just a click of a start mission. I hope to get a round trip of one hour of flight time. With all that being said, I have found so much great information in your magazine. It's a definite read for me as soon as it comes out. Even if the projects are too advanced or not something I think I want to do, I read them to learn more about how the pi works.

Keep up the great work!
Arthur Lee Martin II

Over the last year, many changes have been made to the way you can access and enjoy The MagPi. Not only can you download the pdf or read the ISSUU version online at themagpi.com, but you can now access an HTML version too. There is also a MagPi index, which makes it even easier to find and access your favourite MagPi articles! Andrea Stagi, the genius behind the NanPy article in issue 8, has also created an android app (available from the Google Play store), which will allow even greater access to The MagPi articles on the move.

All of you should by now have received your volume 1 binder bundle of the printed magazines from the very successful Kickstarter and this year we will be announcing details of our second Kickstarter for the volume 2 binder bundle soon! We are working tirelessly to ensure we can make printed copies of the magazine available as soon as possible after it has been uploaded online meaning Pi Supply are offereing their own subscription service. You can still purchase print copies at many other stockists too.

Not only that, but The MagPi is being translated into many different languages, including French, German and Spanish! If you are able to help with translations please contact us.

We want to make The MagPi even better and can only do this with your help. We need your feedback to help us improve. If you have any comments please feel free to email us or contact us via Facebook (MagPiMagazine), Twitter (@TheMagP1) or Google+ (+The MagPi).

We are always on the look out for new, enthusiastic volunteers to help with layout and proofreading of the magazine. If you can spare a couple of hours a month to help us make The MagPi great please email The MagPi at emailthemagpi@gmail.com.