ISSUE 18 - NOV 2013

Get printed copies at themagpi.com

The MagPi™

A Magazine for Raspberry Pi Users

Remote Sensor Monitoring
ATMEL Wireless MCUs
Raspberry Pi at CERN
BrickPi with Scratch
LOGiPi FPGA
C++ Streams
Camera GUI

Win a Raspberry Pi 16GB SD card breadboard & more

After Dark
Infrared Camera

ISSN 2051-9990

The MagPi™

http://www.themagpi.com

Welcome to issue 18 of The MagPi, a free to download community driven magazine, packed to the brim with all things Raspberry Pi.

The Raspberry Pi Foundation has just released the Pi NoiR, an adapted version of their camera with the infra-red filter removed. This month, we have a great article from Andrew Back of RS Components, where he takes you through the steps he used to create a night vision camera to catch critters in his back garden.

We introduce you to a great program called PiVision which brings an attractive GUI, allowing you to control the module through its native apps. We have a very informative article on the Raspberry Pi at CERN and we look at using Scratch with BrickPi.

We also see the first in a series of articles titled 'Project Curacao', an environmental monitoring system utilising the Raspberry Pi which will be hung unattended on a radio tower on the island nation of Curacao. Exotic!

Of course this only scratches the surface of what we have in store with more favourites on coding, reviews and new products in the world of the Raspberry Pi.

Hope you enjoy.

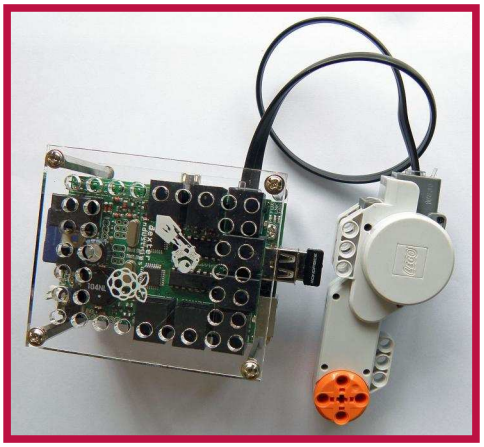
Ash Stone

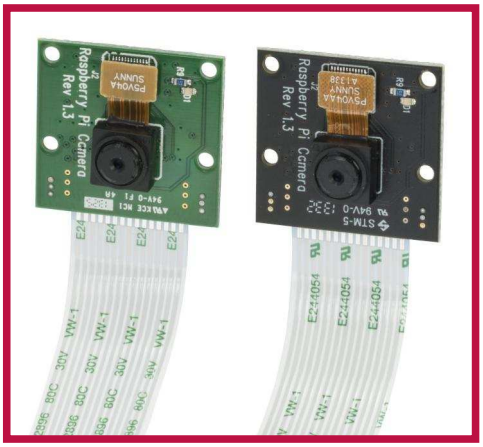
Chief Editor of The MagPi

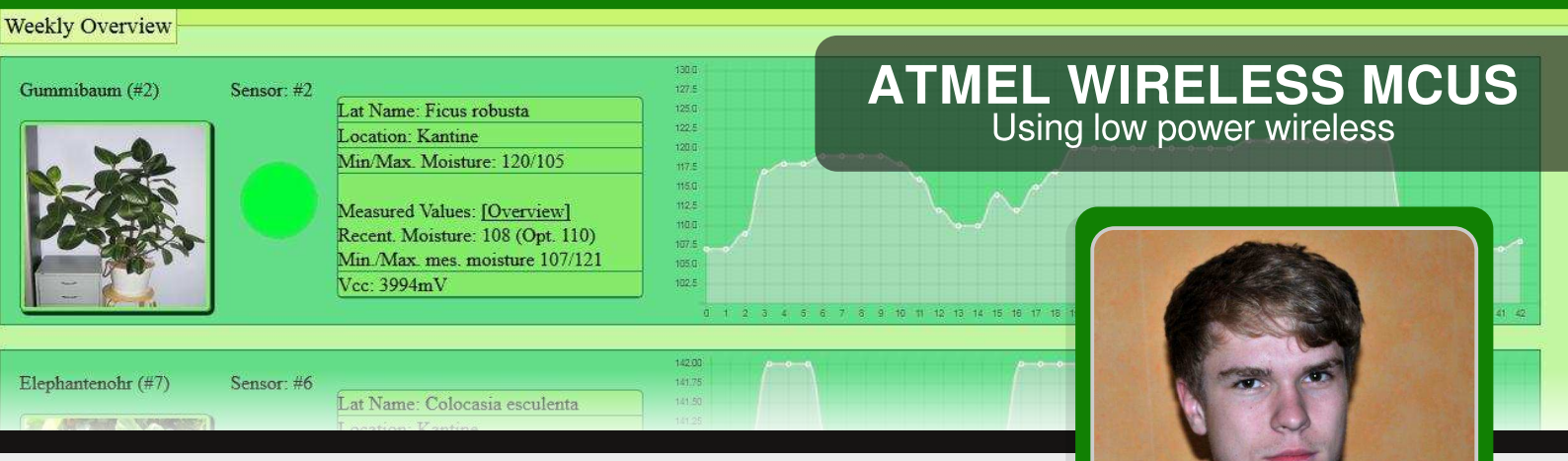
The MagPi Team

Ash Stone - Chief Editor / Administration / Proof Reading
W.H. Bell - Issue Editor / Layout / Administration
Bryan Butler - Page Design / Graphics
Ian McAlpine - Tester / Proof Reading
Matt Judge - Website / Administration
Aaron Shaw - Layout / Proof Reading
Colin Deady - Layout / Proof Reading
Claire Price - Layout / Proof Reading
Matt Weaver - Layout
Amy-Clare Martin - Layout
Gerry Fillery - Proof Reading
Paul Carpenter - Tester

Contents







Create a plant-monitoring system

Sebastian Lehmann

Guest Writer

SKILL LEVEL : ADVANCED

Introduction

This article is intended for all plant enthusiasts among us, who are a little neglectfully when it comes to watering plants. I will show you how to set up a wireless sensor network using the IEEE 802.15.4 standard and program the Raspberry Pi to work as a network coordinator/moisture web-monitoring-system which will help you taking care of your plants.

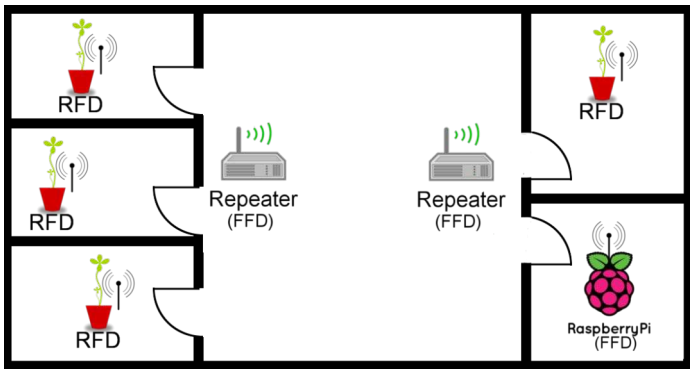
The sensor network

The idea was to stick a battery-powered sensor in each plant pot, of which I want to measure the moisture and send the recent sensor value wireless to my Pi so it can be stored and further processed. The IEEE802.15.4 standard defines a transmission protocol for Wireless Personal Area Networks. It was designed for devices with low power consumption and applications not needing high data rates. So it is just the right solution for my small sensor network.

The network topology

Using the IEEE 802.15.4 standard entails some peculiarities. The protocol just defines the two lowest layers of the OSI Model (PHY and MAC) which means that I have to take care of the network topology and manner of data sending (routing) by myself. I chose a simple tree topology, which only enables data sending in one direction, from the sensors to my Raspberry Pi. The moisture sensors (end devices) are reduced functions devices (RFDs), which are most of the time in standby mode and just wake up to measure and transmit data to the Raspberry Pi. They connect to the next full function device (FFD) within range (that can be a repeater or the Pi), and are only able to send data. The Raspberry Pi acts as the coordinator of the network and becomes thus a FFD. Its task is to open up a network with an unique PAN id on a pre-defined network channel. If new end-devices are detected, a so called short id is given to them. This is a little bit similar to assigning IP addresses via DHCP.

Since the transmission range of the RFDs is limited, I had to create some kind of repeater that works on the base of a simple flooding algorithm. Each time incoming sensor data is received, the data is broadcasted to all adjacent nodes. This could be either other repeaters or the coordinator, if it is within range. To avoid duplicate deliveries to the coordinator and to prevent packets circulating infinitely in the network, every data packet is given a consecutive number that is stored within a buffer in each FFD. If a recently received number is already in this buffer, then the data packet is automatically discarded.

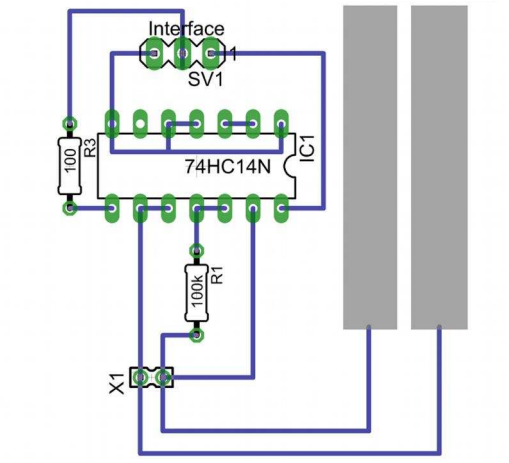


Hardware

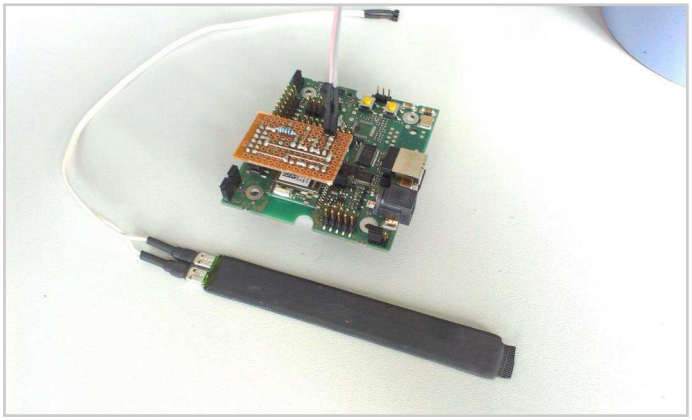
To implement the transmitters and receivers, I have chosen Atmels RF SoCs (ATmega128RFA1/ ATmega256RFR2). They combine an 8-bit microcontroller and a RF transceiver working in the 2.4 GHz ISM frequency band with enough transmission power for our application. Since they contain an AVR core, everyone who has dealt once with AVRs should quickly get into programming the devices as well. There are manufactures that produce complete radio modules containing the MCU and also an antenna. These modules are much easier to handle than an IC with QFN-package! I recommend the deRFmega128-series of the company dresden elektronik. I also used their evaluation boards to develop and test my firmware.

The moisture sensor

The moisture sensor is a relative simple DIY project. It works on the principle of capacitive measurement.



Two metal strips were combined with a Schmitt Trigger RC-oscillator circuit. These two metal strips were then put into the soil in the pot. In order to determine the moisture level you just have to determine the frequency of the circuit. Water has a much higher dielectric constant (>80) than soil (~3.9). So in wet soil the capacity of the metal strip capacitor rises noticeable. This leads to a reduction of frequency of the RC-oscillator. Enclosed by a shrink tube, the sensor might look like this:



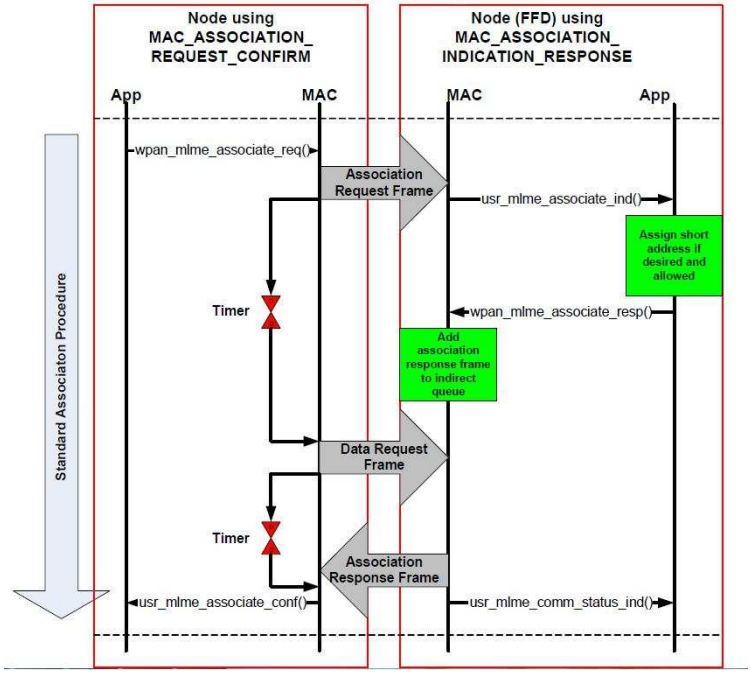
Sensor plus pluggable oscillator circuit for the evaluation board

Programming

The software for the sensors consists of 2 parts: one for measuring the frequency and analysing the sensor values and another part to transfer data to the Raspberry Pi. Measuring the frequency is not difficult. The circuit delivers a clock signal of about 120 kHz (depending of the size of the metal strips) . This can be easily read by an internal counter in the ATmega. To use the integrated transceiver for sending data based on the 802.15.4 standard, I downloaded the Atmel MAC protocol stack. The software is a little complicated and very extensive, it takes some time to get used to it. In summary, the stack is built up of 3 layers: the Mac Core Layer (MCL), Transceiver Abstraction Layer (TAL) and the Platform Abstraction Layer (PAL). Each of these layers communicates with the layer above or below. The program runs using a queue, on to which we put data that we want to transmit or from which we get error or confirm messages. Interaction with the queue usually works as follows: at first you have to make a request (e.g. to send data), which puts the instructions on the queue. After this, the request is performed and a callback function is invoked. With these confirmations, you can react on errors or

indicate successful transmissions. Explaining more details would go beyond the scope of this article. Therefore, refer to Atmels User Guide for further information:
http://www.atmel.com/Images/doc8412.pdf

With these two parts it is now possible to write a small program that wakes up each the MCU every six hours, measure the frequency, do some sensor data validation and go to suspend mode again.



Standard sequence for RFD association

Configuring the Raspberry Pi

To provide the Raspberry Pi access to the low power wireless world, we need another radio module that we can connect to it. I decided to choose the RaspBee, which is also from dresdenelektronik. The RaspBee is designed as a pluggable module that can communicate via UART with the Raspberry Pi. To handle incoming data sequences, I wrote a small Python script that runs permanently in background reading from UART and storing data into my MySQL database. Therefore we have to install the following packages:

```
sudo apt-get install -y python-serial \
python-mysqldb
```

The UART can be simply accessed with the following code:

```
import serial

baudrate = 38400
byte = 8
parity = 'N'
stopbit = 1
timeout = N
onexonoff = False
rts = False
dsr = False

ser = serial.Serial(/dev/ttyAMA0,
  baudrate,byte,parity,
  stopbit,timeout,timeout,xonoff,
  rts, self.dsr)

def getSensorString():
  string = ""
  char = ""
  while (char != "\n"):
    char = ser.read()
    if char == '\n':
      break;
    string += char
  return string
```

My output of sensor data on the console looks like this:

```
PAN-ID: babe
DEFAULT_CHANNEL: d
DEFAULT_CHANNEL_PATH: 0
Netzwerk wurde initialisiert…

##MACPLANT##REASSOCI##00212efffff001b5c##ffff
##MACPLANT##SENSORDAT##00212efffff001b44##08d
##MACPLANT##BATSTATUS##00212efffff001b7b##f9a
##MACPLANT##SENSORDAT##00212efffff001b5c##06b
##MACPLANT##SENSORDAT##00212efffff001b5c##06b
```

These are the strings transmitted from the sensors. They contain several pieces information. The identifier MACPLANT ensures our script knows that the package comes from a radio module. The type of package follows this string. This is important because the sensors do not only send moistures values, but also measures the battery voltage level. Furthermore, we need a type for handling devices connecting to the pi, which are not stored in the database yet. The last part of the string is the MAC address of the end device. This information has to be

in the payload, because otherwise the data would not be tracable once is has been re-sent by the repeater. At the end of the sequence, there is the recently measured hexadecimal voltage (in mV) or moisture value (frequency in kHz). The string can now be parsed by our python script and stored in our database.

Setting up the webserver and database

To provide the measurement data in my local network, I installed a small LAMP software bundle:

```
sudo apt-get install -y apache2  \
mysql-server php5 phpmyadmin
```

I used PhpMyAdmin for managing my database. If you prefer working with the console you can also use the MySQL client software. A ready-to-use database layout is also included in the downloadable project directory.

The web interface

Now there is just one thing left: a user interface. Based on HTML, PHP, jQuery, MySQL and the Chart.js library, it is possible to create a functional, comfortable and neat GUI. It performs the following tasks:

1. Show the last activities

It has a kind of logbook of the last activities, so that we can see the last time sensor data was received. Additionally, the system recognises new sensors that try to connect to the coordinator, critical battery-levels and low activity times.

2. Management of the plant database with the possibility to calibrate new sensors

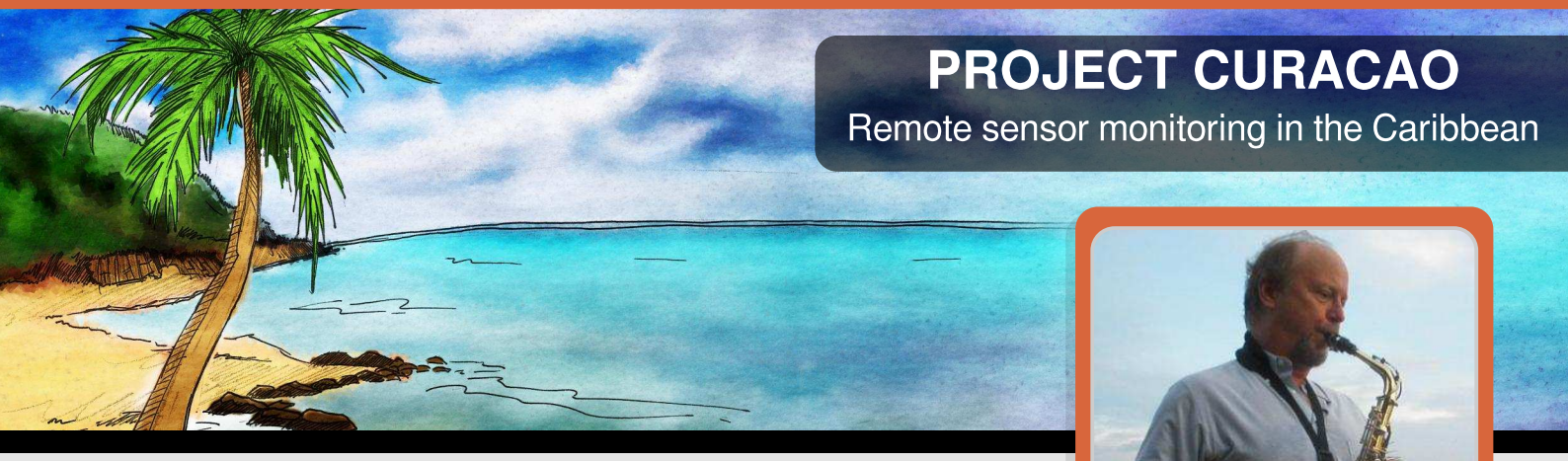
New plants can be added and assigned to a specific room. You can adjust the min and max moisture level of the soil, which has to be determined empiric before.

3. Warning at dangerously high or low moisture values.

Plants that are too dry or wet can easily be recognised the by a red or blue point in the overview. If that is not enough, you could think about other possibilities. A simple way to provide an additional alarm is to install an e-mail server that sends an alert to your e-mail account. You could also configure a Raspberry Pi to SMS gateway, to send a short text to your cellphone. Even a little sound system plugged on the Raspberry Pi that plays warning messages/sounds is possible.

Conclusion

The system has taken a little burden from me. Since I now will be warned of critical soil moistures, lifeless remains in my plant pots already became a rarity. If you have questions, are looking for detailed information or project files just visit the project website:
http://rapi-projects.de.vu/



PROJECT CURACAO

Remote sensor monitoring in the Caribbean

Part 1: An introduction and power management

John Shovic

Guest Writer

SKILL LEVEL : INTERMEDIATE

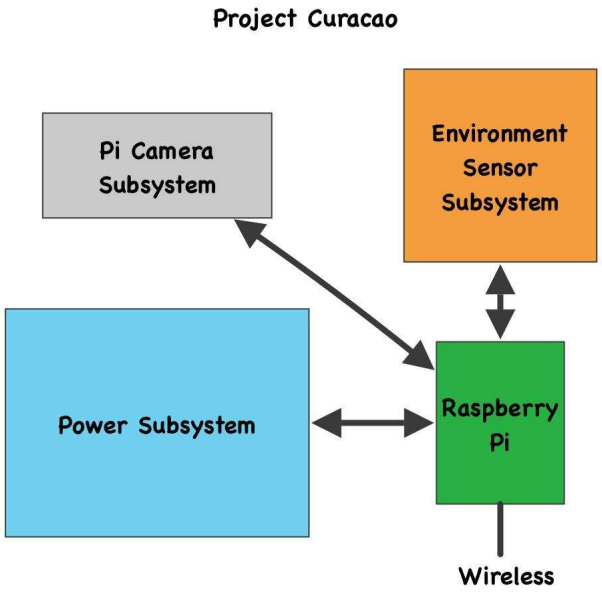
What is Project Curacao?

This series discusses the design and building of Project Curacao, a sensor filled project designed to 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. It is a harsh environment with strong tropical sun, salt spray from the ocean and unremitting heat. But it is a beautiful place to visit and a real challenge to build and install a Raspberry Pi based environmental monitoring system.

Project Curacao is designed to monitor the local environment unattended for six months. It will operate on solar power cells during this time and will communicate with the designer via an iPad App called RasPiConnect (see the XML article in Issue 17). The charging and discharging peformance of the system will be monitored as well as the degradation of the solar cells measured. The system will be designed to reboot itself via a watchdog timer and will reboot itself when power is available in the case of a low power or sun condition.

System description

Project Curacao consists of four subsystems. A Raspberry Pi Model A is the brains and the overall controller. The Power Subsystem consists of LiPo batteries and charge management. The Environment Sensor Subsystem has in-box temperature, outside temperature, luminosity, barometric pressure and humidity sensors. The Raspberry Pi Camera Subsystem contains a Raspberry Pi Camera and a servo motor controlling the cap over the camera to keep salt spray off the camera lens.

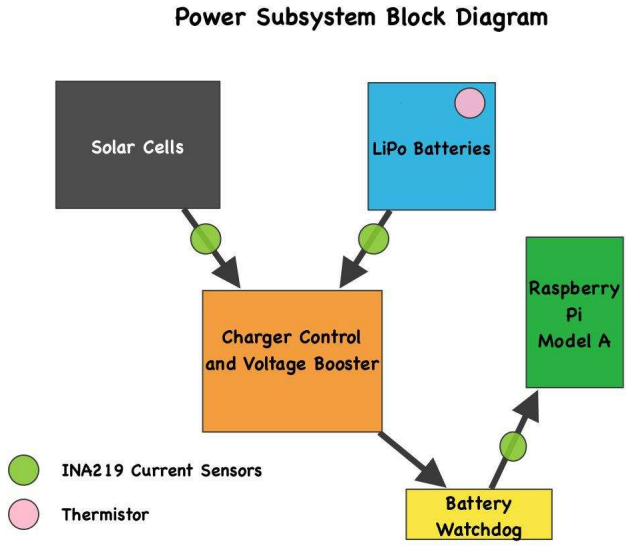


What hardware to use?

We are using the following hardware for the power subsystem project:

2 Adafruit 6V 3.4W, 530mA Solar Panels
1 Seeed Studio LiPo Rider Pro charger (includes a booster to 5V from 3.7V)

2 3300 mAh 3.7V LiPo batteries
1 Raspberry Pi Model A
Wifi
3 Adafruit INA219 Current Sensors (I2C)
1 Adafruit 12 bit A/D (I2C)



Calculating the power needed

The first criteria for designing the Power Subsystem is determining our goals. We want the Raspberry Pi to run all day and at least three hours before sunrise and three hours after sunset. Our goals and budget influence our hardware choices, so they are not totally independent.

Assume:
- 8 Hours of Sun running the cells at least at 80% of max
- Delivery of current to Raspberry Pi at 85% efficiency
- Raspberry Pi takes 350mA on average

Given these we can calculate total Raspberry Pi runtime during a typical day:

PiRunTime = (8 Hours * 80% * 1020mA) *85% / (350mA) = 15.8 hours

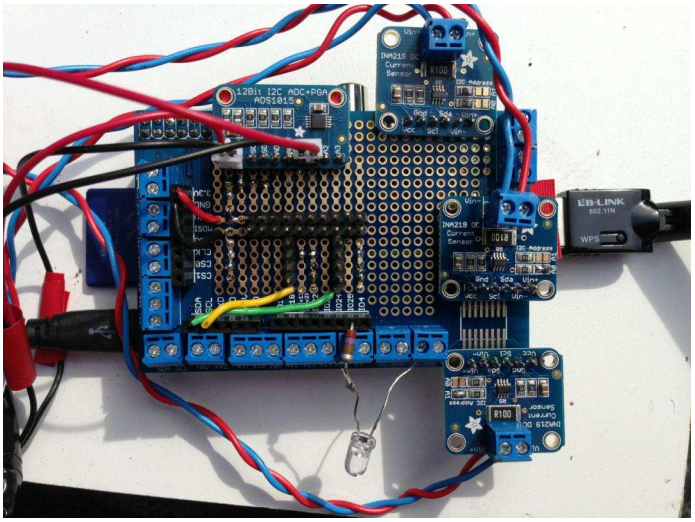
Our goal was for 14 hours, so it looks like our system will work. 6 Hours of running the Raspberry Pi will take (350mA/85%)*6 Hours = 2466mAh which is comfortably less than our 6600mAh batteries can store.

What to measure?

To be able to see what is happening in our Power Subsystem, we want to measure the currents and voltages of the solar panels, the batteries and the current into the Raspberry Pi. We also want to monitor the temperature of the batteries. All of this information will be placed in a MySQL databse for later analysis.

Putting in the sensors

The INA219 are high side current sensors. That means you connect them in-line with the + power lead. These sensors will measure both positive and negative currents. Why negative currents? The battery will receive current (negative) when charging and supply positive current when driving the load. We are connecting one in-line with the batteries, the USB cord for the Raspberry Pi and the solar panels. The INA219 also measures the voltage at the + power line which allows us to calculate power at each point (P = VI). We are also using a 10K Ohm thermistor wired to channel 0 of a 4 channel 12 bit A/D to measure the temperature of the batteries.

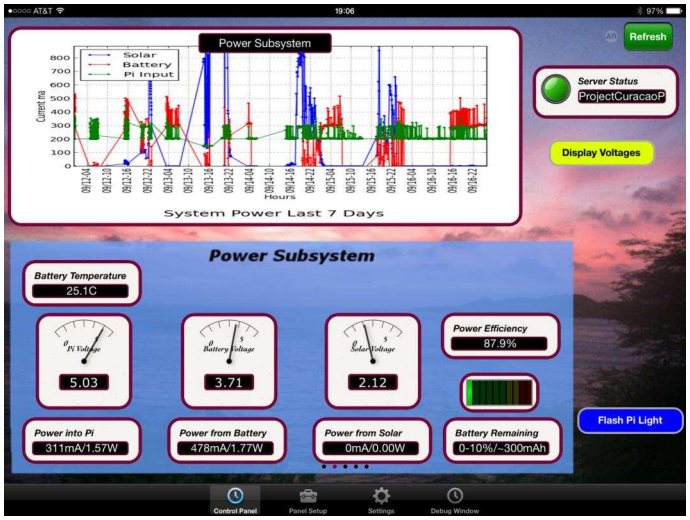


Issues with the sensors

We are using a Python scheduler (apscheduler) to gather the data from all the power subsystem sensors and put the data in a MySQL database on the Raspberry Pi. Part 4 will describe the Raspberry Pi control software. While bringing up the monitoring software for the power subsystem we discovered a problem with reading negative currents from the INA219. Reading negative currents are critical to this project when the batteries are being charged. It turned out to be a problem in both the Adafruit C++ drivers (fixed as of September 2013) and in the subfact_pi_ina219 Python drivers on GitHub. The subfact drivers fixes are on GitHub/projectcuracao.

Monitoring the sensors remotely

Project Curacao will be monitored remotely through the Internet by the use of SSH and by the use of RasPiConnect. RasPiConnect allows you to build meters, buttons, and display graphs remotely from the Raspberry Pi remotely. The screen for monitoring the power system is shown below. The software for generating this display is on github.com/projectcuracao.



The battery watchdog

After a significant period of testing, we have determined that the Raspberry Pi does not wake up very well from a discharged battery event as it is being recharged by the solar cells. What appears to be happening is the battery reaches a high enough voltage and the charger turns the power on to the Raspberry Pi. The Raspberry Pi Model A takes around 300mA and immediately pulls the battery back down to the point where the Raspberry Pi will not boot. This kind of constant boot/reboot cycle can kill SD cards after a period of time. After some thought, we determined that we could use a low power Arduino Uno with a relay as a very smart watchdog timer. This device has its own small LiPo battery, a small solar cell and it's own charger. We want this device to be independent of the main system. It will do the following four things:

1) Monitor the main system battery voltage and turn the relay on to the main system when the batteries reach 20% charge. This is in the case of a full battery discharge rate. There will be a 5% of charge hysterisis to prevent flapping. This meaning that the relay will not turn off until a 15% charge rate has been reached.

2) Monitor a GPIO pin from the Raspberry Pi, looking for transitions at least every 15 minutes. If the Raspberry Pi does not change the state of this pin every 15 minutes, the Battery Watchdog attempts to shutdown the Raspberry Pi and then cuts power for 10 seconds and re-applys the power to reboot the Raspberry Pi.

3) It will start the Raspberry Pi up approximately 3 hours before sunrise and shut the Raspberry Pi down approximately three hours after sundown. Sunrise and sundown will be approximated by the use of a photoresistor with enough hysterisis to not mistake a cloud for the sunset. It will also bring the Raspberry Pi up in the middle of the night for 30 minutes to record night time temperatures. If the main battery is fully charged, the Battery Watchdog will cancel the night time shutdown.

4) Additional GPIO pins on the Raspberry Pi will be used by the Battery Watchdog to communicate status and events. This will also be used to allow the Raspberry Pi to shut itself off in case of high temperatures.

This entire watchdog could be done in hardware with an FPGA board, but we decided to go with the parts we had on hand. A good question we have yet to answer is whether to make the power relay normally open or normally closed. If it is normally open and the Battery Watchdog fails, then the Raspberry Pi is out of action. If it is normally closed and the Watchdog fails, then the Raspberry Pi might be able to still function.

For a system designed to operate 3700 miles away from the engineer who has to fix it, a good watchdog function is critical.

What is coming Up?

Part 2 of this article will describe the environmental subsystem of Project Curacao and the use of a Raspberry Pi controlled fan. Part 3 goes through the Raspberry Pi Camera Subsystem and Part 4 describes the software system used for Project Curacao.

All of the code used in this article is posted on GitHub at github.com/projectcuracao

PiGlow

Raspberry Pi® Colour LED Plate

18-channel 8-bit PWM (0-255)
Individually addressable
6 hues + white
~300-500mcd per LED
Fits nicely inside a Pibow

http://shop.pimoroni.com/products/piglow

PIMORONI

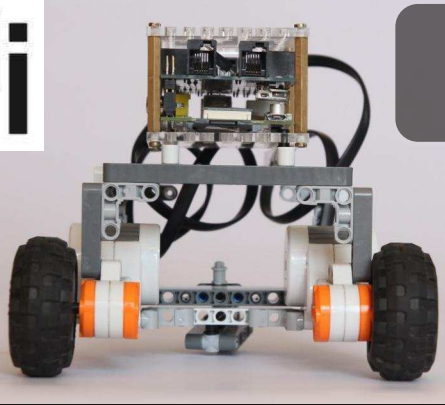
TIMBER

The neat little log cabin for your Raspberry Pi®

Made from real
spruce hardwood

Available From:
http://pibow.com/

PIMORONI



Scratch interface

John Cole

Guest Writer

SKILL LEVEL : BEGINNER

Introduction

In this tutorial we will explore a few Scratch programs for the BrickPi. There is a general discussion and set of basic software installation instructions in Issue 17 of the MagPi.

Scratch is a graphical programming langauge written by MIT, which has been discussed in many articles in previous Issues of the MagPi. Scratch programming for the BrickPi opens up a range of possibilities for younger children, due to the simplicity of the graphical language used.

Software installation

Before beginning this tutorial, the basic software installation instructions in Issue 17 should be followed to setup communication between a Raspberry Pi and the BrickPi.

Scratch can be used to access the BrickPi hardware by using Sratchpy. This package provides a library to allow Python to talk to Scratch. The Python code then talks to the BrickPi hardware. To install Scratchpy, type:

```
git clone \
https://github.com/pilliq/scratchpy.git
cd scratchpy/
sudo make install
```

Now get a copy of the Scratch example programs by using git,

```
git clone \
https://github.com/DexterInd/BrickPi_Scratch.
git
```

Now we are ready to start scratch.

Starting Scratch

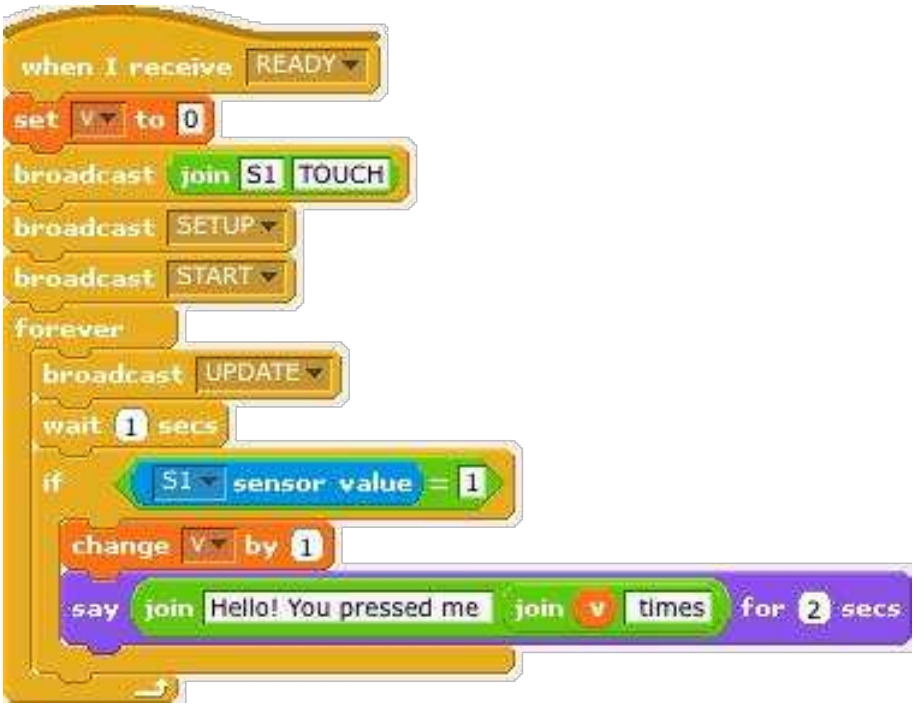
Start Scratch by clicking on the icon or by typing scratch. Then enable remote sensor connections by clicking on "Sensing", then right click on "Sensor Value" near the bottom of the Sensing tool menu and select "enable remote sensor connections". This change is needed each time Scratch is started.

The next step is to start the thin Python layer that allows Scratch to talk to the hardware. The python program is in the BrickPi_Scratch directory. Type :

```
cd BrickPi_Scratch/
sudo python BrickPiScratch.py
```

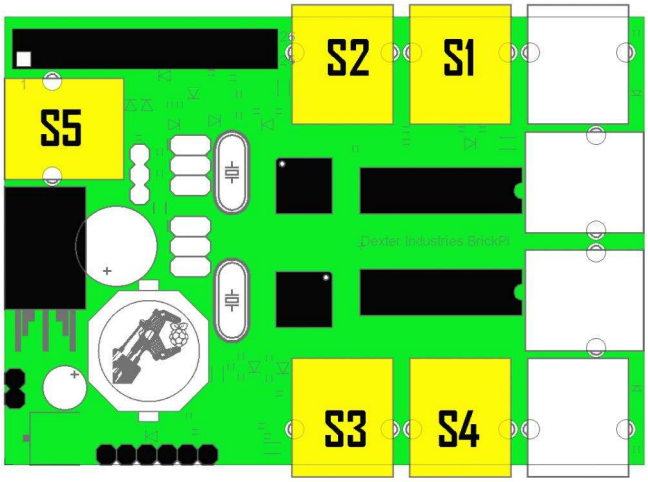
This program must be run with sudo, since by default the permissions on the device needed to access the BrickPi are only available to the root user. This Python program can be stopped by typing CTRL-C in the terminal window. Now Scratch can be used to try out some of the example programs in,

```
BrickPi_Scratch/Examples
```



Example 1: Touch Sensor

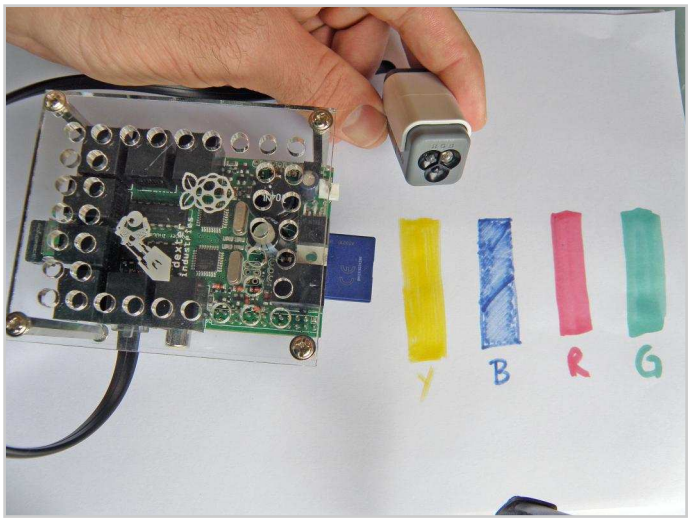
In this first example we will use the BrickPi to read a LEGO® Mindstorms touch sensor. Each time we hit the touch sensor the Scratch cat will count how many times the touch sensor has been touched. Before we begin, lets first connect a LEGO® Mindstorms Touch Sensor to Sensor Port 1 (S1) on the BrickPi.



Find S1 in the diagram, and plug your touch sensor in. Open the example program Touch Sensor test. Then start the program. You will see the Scratch cat is stationary. Now if you press the touch sensor, the Scratch Cat will tell you how many times it has been pressed. The variable v is used to count the number of times the touch sensor has been touched. The broadcast join command sets the sensor S1 as the TOUCH sensor. The next two broadcast commands setup and start the TOUCH sensor. The program then enters an infinite loop. Each time the loop runs, the broadcast UPDATE command is used to get the state of the TOUCH sensor. If the sensor is pressed, then the value of v is incremented by one. Then the cat says "Hello! You pressed me v times", where v is the value stored in the v variable.

Example 2: Color Sensor

In this example, we will use the LEGO® Mindstorms colour sensor to read the colour value on a piece of paper. This is a very simple version of what scanners and cameras do: reading color values on a surface. We can write a program that makes the Scratch cat change to be the colour the colour sensor is scanning on the paper. First, connect the colour sensor to Port S1 on the BrickPi again. Then use a clean piece of white paper and some colouring pens to colour in a yellow, blue, red and green rectangle. Here is where you can get really creative. You can use your colours to identify things with the BrickPi, or you can use colours to navigate, or you can use colours to tell between different items.



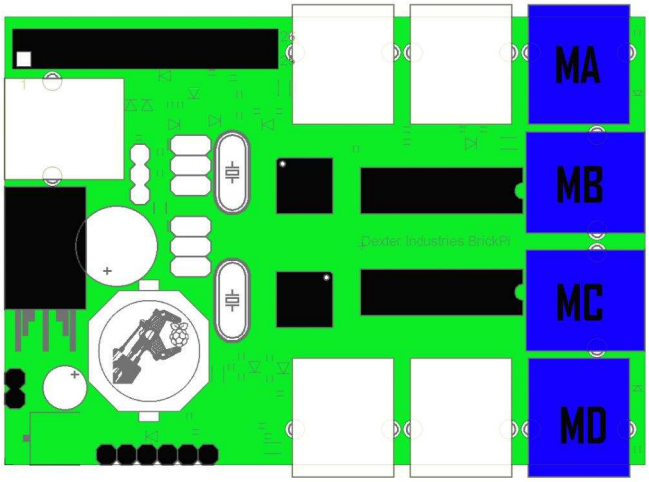
So let us take a look at how to use the colour sensor. Open the example Scratch program Color Sensor test.

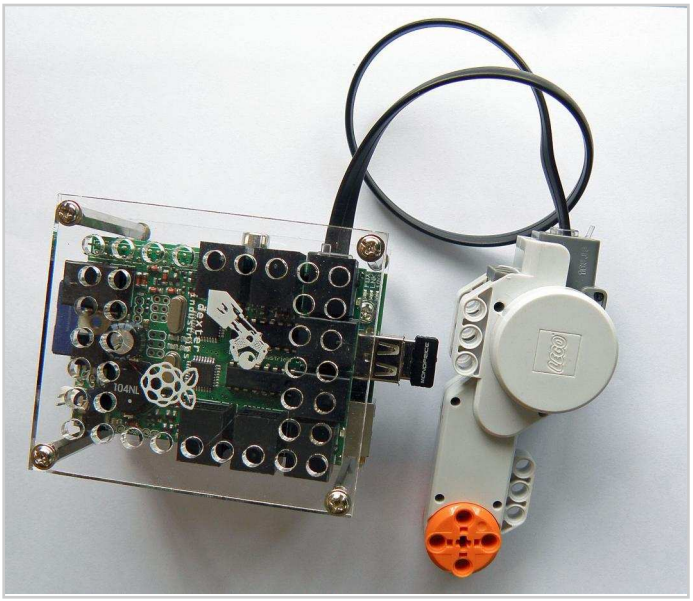
This program is slightly simpler than the first

example. The program starts by associating the S1 sensor port with COLOR. Then the connection with the COLOR sensor is established. The loop continues forever, checking for updates from the sensor. The colour value from the sensor is then used to change the costume of the Scratch cat. To avoid the Scratch cat from being constantly updated while the colour sensor remains pointing at the same colour, there is a wait statement. Now try running the program and position the colour sensor over each of the colours.

Example 3: Making a Motor Run

The fun thing about robotics is that its not just taking sensor data, its moving. In this last example we will show you how to make a motor move, and how to control it from a keyboard. In this example we will use Motor A on the BrickPi. Connect a LEGO® Mindstroms motor to motor port MA.





Now with the motor attached, let us look at an example of moving the motor. Open up the example program Motor test. Unlike the sensors, we do not have to setup anything before using the motors. We can just start using them; nothing other than the motors should go on the motor ports. So let us look at how to make a motor do just one thing.

In the Motor test program there is one spirite that has four scripts. Each of the scripts handles a separate action. If the down arrow is pressed, then the broadcast message is used to send the motor the stop command. Then the arrow sprite says "Stopped!". Pressing the up arrow causes the motor to be sent the start command. Then the arrow says "Running!". The left and right arrow keys are used to speed up or slow down the motor speed. The value of the speed variable is sent to the motor using the broadcast command and then the arrow reports the current speed. If space is pressed, then the motor will reverse direction and start turning the other way. The variable speed is set to be a negative value, but of the same size. The arrow sprite is turned around to illustrate this and the new speed is reported.

Other examples

There are a few more examples in Github that show you how to use different sensors, including the temperature sensor, the ultrasonic sensor, and the dFlex Flex sensor.

Going further

LEGO® sensors can be used within a Scratch game. For example, instead of using the keyboard to control a game try using a set of colours on a piece of paper. Rather than use the space bar, a touch sensor could be used to make a sprite jump on the screen.

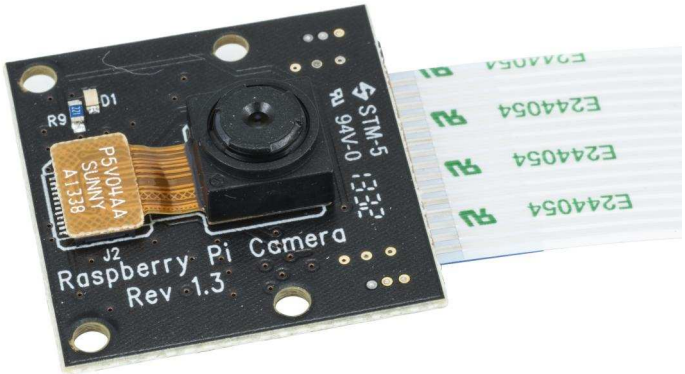
There are several ways the sensors could be used for a robotics program. For example, a robot could be used to drive forward looking for a paricular colour. Or perhaps it could be used to follow a coloured strip on the ground.

Andrew Back & Colin Deady
Guest / MagPi Writers

Introducing the Pi NoIR camera

Almost exactly 103 years after the October 1910 Photographic Journal of the Royal Photographic Society published a paper entitled "Photography by Invisible Rays" by Professor Robert Williams Wood heralding the dawn of infrared photography, the Raspberry Pi Foundation has announced the Pi NoIR, a modified Pi Camera that has had the infrared cut filter removed.

The technological evolution of photography over the last 100 years has been staggering, and coupled with the prevalence of modern digital cameras this has enabled greater photographic opportunities than ever before. The Pi NoIR opens up the world of low (visible) light, astrophotography and slightly ghostly portrait photographs for a pittance in comparison to other offerings on the market.



The CMOS sensor used in the Raspberry Pi Camera is sensitive to infrared (IR). For this reason, it is normally paired with an IR cut filter that blocks IR. This has the result that the sensor plus filter responds to light in the same way that our eyes do and colours look natural. The Pi NoIR is essentially the original Raspberry Pi camera, albeit without this infrared-blocking filter fitted (no IR, noir, get it?). This means you get the same 5 megapixel 2592x1944 pixel resolution sensor that can be controlled in the same way.

The extended spectral range of the Pi NoIR means that it can be put to use in all manner of interesting applications, for example imaging hot objects or when paired with a IR LED cluster for viewing wildlife after dark. An after dark image taken with the NoIR camera is on the cover of this issue.

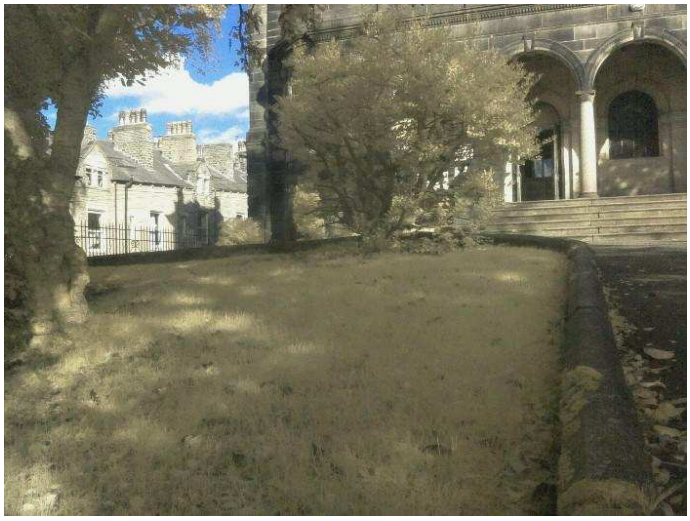
Image quality

Removing the filter leaves the camera sensitive to IR, but also adds a red/pink hue to images. Objects such as grass and leaves reflect infrared, leaving them with a brighter than normal appearance. Without the filter, the camera could

be used for hyperspectral imaging. It could be combined with structured light sources for 3D scanning. It may have some thermal imaging applications. However, since it operates in "near-infrared", it would only be able to image relatively hot objects.

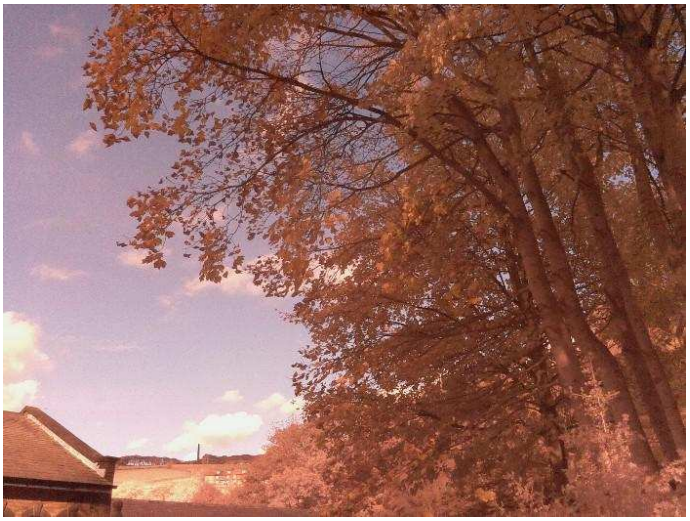
Using filters

Filters can be used with the camera to emphase particular types of objects in the field of view. Adding a blue filter to the camera removes the pink/red hue, but leaves the plants appearing brighter than normal. Blue filters can be used for detecting disease in plants, by looking for darker areas. The Pi NoIR comes with a separate blue gel filter that lets the level of photosynthesis in plants be measured.



To make use of longer wavelength light and IR only, a filter can be used to block visible light.

Firstly, a picture without a filter.



Now with a filter that blocks electromagnetic radiation below a wavelength of 850nm, noting that the human eye can see roughly in the range of 390nm to 780nm (this varies by individual).

No image processing has been applied to this image. The blue of the sky is blocked by the filter and the clouds appear to be more radiant. Some image processing can be used to enhance the result.

Why can't I just add an IR filter to the front of my Raspberry Pi Camera?

There are two types of IR filter on the market which perform the exact opposite yet are similarly named. The IR cut filter above the sensor present in most digitial cameras including the Raspberry Pi Camera blocks infrared, where-as a screw-in infrared filter that can be attached

to an SLR blocks visible light, allowing the infrared to pass through. The problem that occurs in this scenario is that while the first filter will let you just record infrared, the second will then try to block a very similar frequency range. This results in several drawbacks including potentially very dark photographs, or long exposure times and likely an inability to use the viewfinder (or live preview) to view the subject.

So why is infrared important?

It is all about information. Think of what you can see: with your eyes open you see a world of detail around you, but close your eyes and all of that information is lost. Remember, your eyes can only see a limited range of the electromagnetic spectrum. Compare this to a bee which can see into the ultraviolet (shorter wavelengths), or to a snake that can see into the infrared (longer wavelengths). The human eye is pleasantly (for us) pitched in the middle, but the world around us offers so much more light, if only we could see it. A camera that can record infrared can represent this information in colours that we can see, often referred to as false colour. This is a fantastic capability for science research.

For example, NASA's Spitzer Space Telescope, www.spitzer.caltech.edu, works in infrared capturing images that would otherwise remain unseen. This capability increases the valuable science that can be undertaken with the four Great Observatories (alongside Spitzer these include the Hubble Space Telescope, Compton Gamma Ray Observatory and Chandra X-ray Observatory).

So is the Pi NoIR a night vision camera?

like a standard digital camera the Pi NoIR needs light to be able to record an image. It does not amplify the light, but just records a different part of the electromagnetic spectrum. Infrared security cameras have infrared lights attached that are invisible to us but illuminate the scene for the camera. Wildlife films shot at night often use infrared cameras and lights in to avoid unsettling nocturnal animals. The Pi NoIR is not a night vision camera, but instead a camera that can take advantage of infrared lights for illumination.

The Pi NoIR is not purely for night work as the photographs above show. The camera will work in daylight and can produce images starkly different to the regular Raspberry Pi Camera.

So should I just buy a Pi NoIR instead of a Pi Camera?

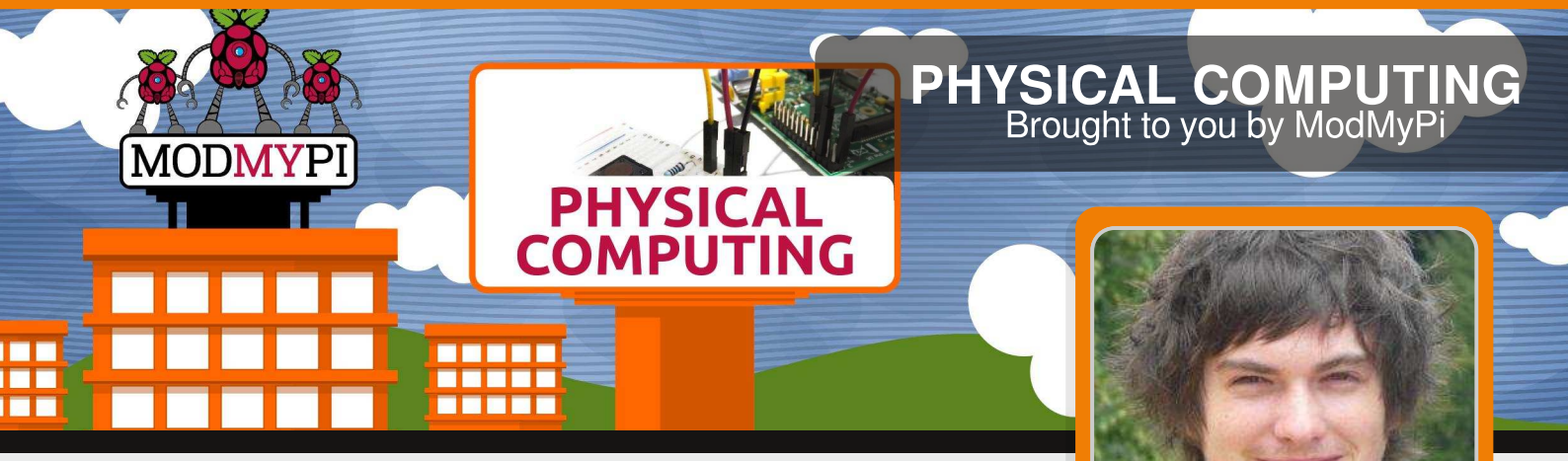
Well it depends. If you want to record infrared light then yes, a Pi NoIR ought to be considered. But if you are going to be photographing purely in the visible frequency range then the answer is less clear: if you want natural looking images then the Pi Camera is the way to go, where-as if you want images with higher (not natural) contrast, ghostly pale skin tones and the like then the Pi NoIR may be an option. Unless you have a specific project in mind for the Pi NoIR (or just want to experiment, and experimentation is a good thing) then we would suggest sticking with the Pi Camera.

The Pi NoIR is now available from RS Components for £16.80.

Raspberry Pi NoIR announcement:
www.raspberrypi.org/archives/5128

The DesignSpark blog for further information:
http://www.designspark.com/eng/blog/nocturnal-wildlife-watching-with-pi-noir
http://www.designspark.com/eng/blog/fun-with-pi-noir-and-filters

At The MagPi we are always interested to hear of new uses for the Raspberry Pi. If you are working on a particular project using the Pi NoIR then please do let us know.
Email: editor@themagpi.com



Buttons and switches with the Raspberry Pi - part 2

Jacob Marsh

ModMyPi

SKILL LEVEL : BEGINNER

Buttons and switches are a fundamental part of physical computing. This beginners tutorial is designed to teach the basics of physical operation with the Raspberry Pi using a simple momentary switch setup. In part one in Issue 17, we discussed and built our switch circuit. In this part, we will go through the steps of programming and interacting between the Pi and our components. The coding part of this tutorial is in Python, a widely used general purpose language. It is also very readable, so we can break down and explain the function of each line of code. The purpose of our code will be to read the I/O pin when the switch is pressed!

Using a switch

If you have not already done so, start X by typing startx and load the program IDLE3. Since we are starting a new project, open up a new window File>>New Window. Remember that Python is case sensitive and indentation is fundamental. Indentation, which is used to group statements, will occur automatically as you type commands, so make sure you stick to the suggested layout. The first line of our code imports the Python library for accessing the GPIO.

```
import RPi.GPIO as GPIO
```

Next, we need to set our GPIO pin numbering, as either the BOARD numbering or the BCM numbering. BOARD numbering refers to the physical pin numbering of the headers. BCM numbering refers to the channel numbers on the Broadcom chip. Either will do, but I personally prefer BCM numbering. If you're confused, use a GPIO cheat sheet to clarify which pin is which!

```
GPIO.setmode(GPIO.BCM)
```

Now you need to define the GPIO pins as either inputs or outputs. In Part 1, we set BCM Pin 17:BOARD Pin 11 (GPIO P17 [Pin 11]) as our input pin. So our next line of code tells the GPIO library to set this pin as an input.

```
GPIO.setup(17, GPIO.IN)
```

In part 1 of the tutorial, the input pin was tied high by connecting it to our 3.3V pin. The purpose of our Python program is to check to see if the input pin has been brought low e.g. when the button has been pressed. To check the high/low status of the pin we're going to use a True or False statement within an infinite loop. We need to tie

our True statement to the high value of our input pin. To do so we need to create a new variable called input_value and set it to the current value of GPIO P17 [Pin 11].

```
while True:
  input_value = GPIO.input(17)
```

The next step is to add a condition that will print something when the button is pressed. For this, we'll use a False condition. The input_value is False when the button is pressed and the associated signal is pulled low. This can be checked with a simple Python it statement.

```
  if input_value == False:
    print("Who pressed my button?")
```

When the button is pressed the program will now display the text: Who pressed my button?, feel free the change this to anything you want.

```
    while input_value == False:
      input_value = GPIO.input(17)
```

The last two lines in the above code are very important, they create a loop that tells Python to keep checking the status of GPIO P17 [Pin 11] until it's no longer low (button released). Without this, the program would loop while the button is still being pressed meaning the message will be printed multiple times on the screen before you release the button. The final program should look like this in python:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN)
while True:
  input_value = GPIO.input(17)
  if input_value == False:
    print("Who pressed my button?")
    while input_value == False:
      input_value = GPIO.input(17)
```

Save the file as button.py. In order to run the program, open a new terminal window on the Pi and type the following command:

```
sudo python button.py
```

At first nothing will happen, but if you press the button the program will print the defined message. To exit a running Python script, simply hit CTRL+C on the keyboard to terminate. If it hasn't worked don't worry. First check the circuit is connected correctly on the breadboard as defined in part 1, then that the jumper wires are connected to the correct pins on the GPIO port. If it still fails to work, double check each line of the program is correct remembering that python is case-sensitive and checking if indentation is correct. I find that typing the code out by hand will give better results than a simple copy/paste. This is a deceptively simple program that can be used for many purposes. The same code could be used to read when the pins of separate devices, such as a sensor or external micro-controller, have been pulled high or low.

Adding an LED

We'll carry on using the same breadboard as before but will require a couple extra components:

- An LED (light emitting diode), any colour you like.
- ModMyPi's Ridiculous Resistor Kit (RK995)
  * 330Ω Resistor - (Orange, Orange, Black, Black, Brown)

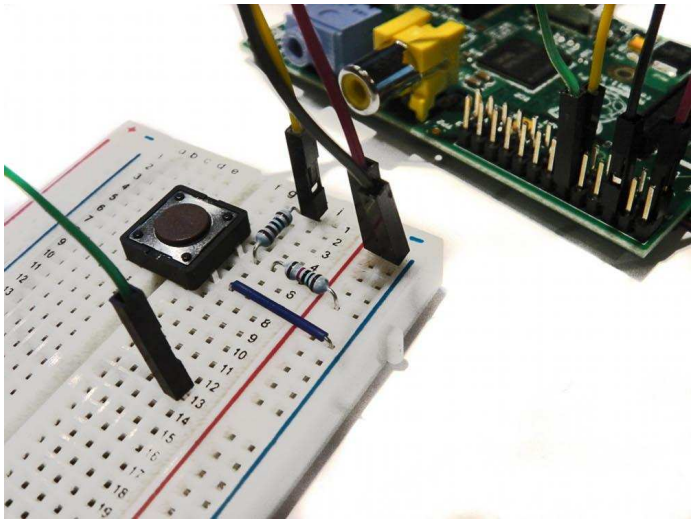
An output pin on the GPIO port can be set to either 0V (low) or 3.3V (high). Our expansion circuit will wire up an LED between an output pin and a ground pin on the Raspberry Pi's GPIO port. We'll be using Python to trigger our output pin high, causing a current to pass through the LED, lighting it up! We also add a 330Ω Resistor to limit the current that is passed through the LED.

For this exercise we will connect the LED to GPIO P18 [Pin 12] which will be defined later in

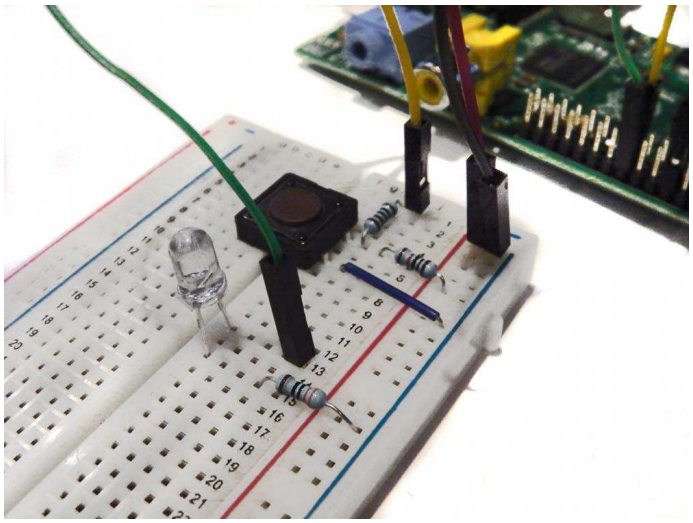
our program as an output. We will use the same ground pin GPIO GND [Pin 6] as before, which is already connected to the Negative rail (-) on our breadboard.

Building the Circuit

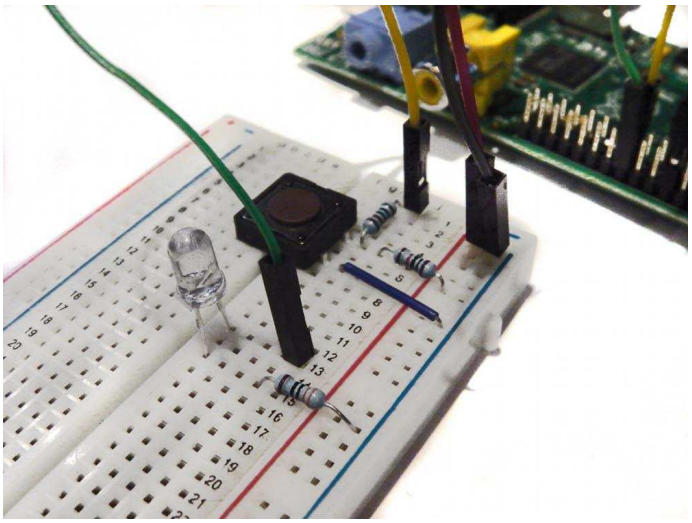
1. Connect GPIO output to breadboard. Use a green jumper wire to connect GPIO P18 [Pin12] on the Pi to an empty to row on the breadboard



2. Add an LED. Insert the long leg (anode) of the LED into the same row as the green jumper wire and insert the shorter leg (cathode) into another empty row on the breadboard. Note: Make sure the LED is inserted the correct way round as it will only allow current to pass through it in one direction. If it's wrong it won't light up!



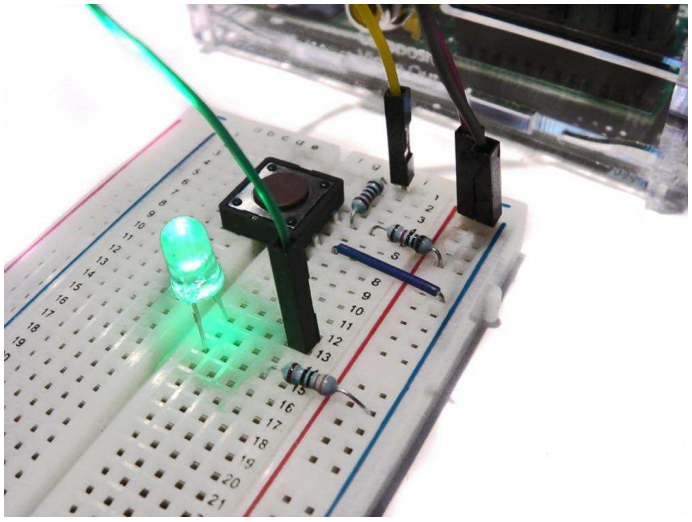
3. Add 330Ω Resistor. Connect the resistor between the cathode of the LED and the Negative rail (-) on the breadboard. This protects our LED from burning out by way of too much current.



And that's our circuit built!

Next time

Next time, we'll add a timer function to our script and a trigger function for our LED. As the script will be more complicated, it requires a proper clean-up function, so we'll code that in too!



Expand your Pi

Stackable Raspberry Pi expansion boards and accessories

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

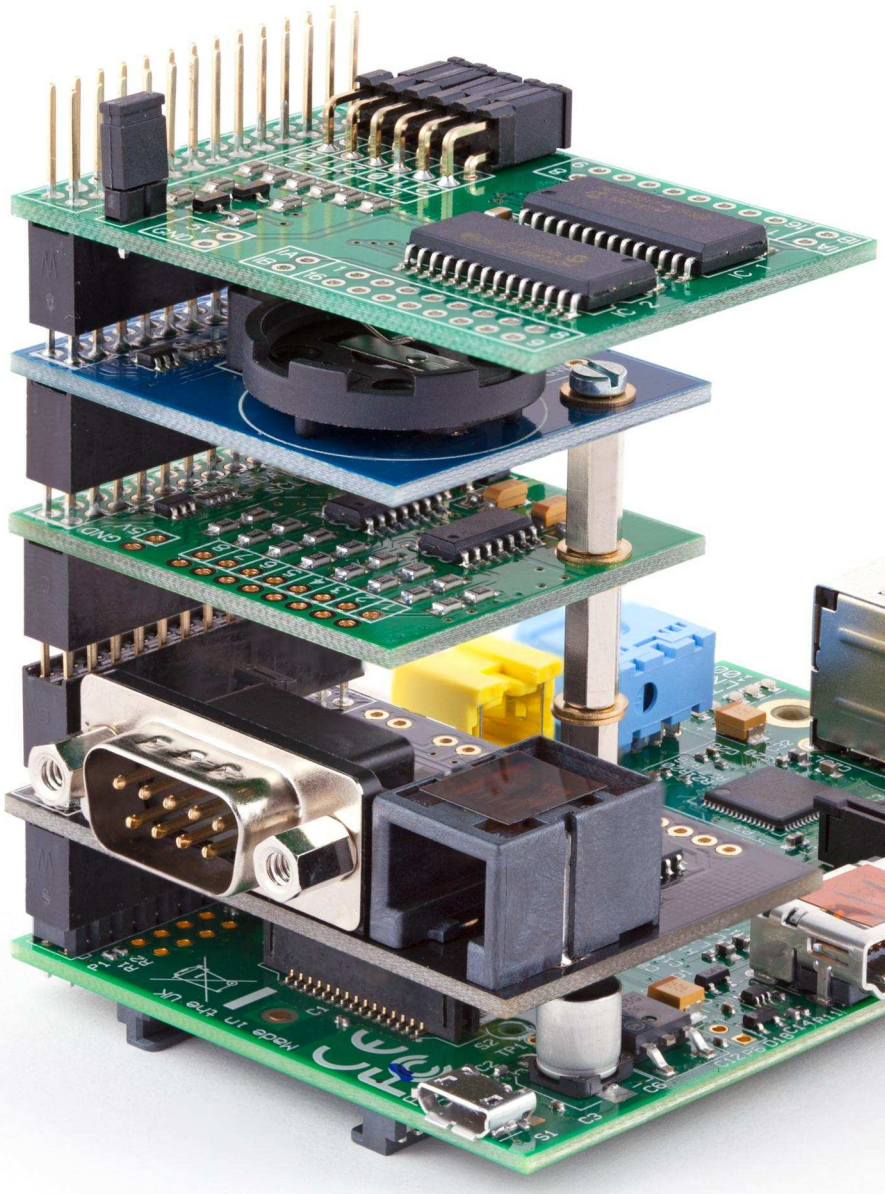
RS232 and 1-Wire® expansion board board adds a serial port to your Raspberry Pi. Ideal for the Model A to enable headless communication.

£19.99

Buffer Pi

Bidirectional Voltage-Level Translator and GPIO protection. Use 5V devices devices on your Raspberry Pi GPIO port.

£13.99



AB electronics UK

www.abelectronics.co.uk

PI VISION

A GUI for the Raspberry Pi camera

B. J. Rao

Guest Writer

Easily explore all the features of the Raspberry Pi camera

SKILL LEVEL : BEGINNER

Pi Vision is a graphical user interface (GUI) for the recently introduced Raspberry Pi camera. It allows you to control the camera's functions via its command-line applications. It also displays the commands sent. The project was written in Pascal and developed using Lazarus - a professional, free Pascal integrated development environment (IDE). It has been released as open source. Pi Vision makes it easy to control the camera plus it is also instructional and fun!

Building block

If I had to name one of the most interesting and inspiring gadgets to come out in recent years it would probably be the little Raspberry Pi. Its low cost, small foot print and the fact that it runs Linux makes it an obvious choice for all sorts of projects; a building block that sparks the imagination to do just about anything.

However, the idea that makes the Raspberry Pi so unique is that it was designed and introduced as an educational tool; something to familiarize ourselves with actual computing. Not just at a user level but at the programming and integration level as well. The recently introduced Raspberry Pi camera aligns itself with that same idea.

You probably have an i-something, a tablet device or a smart phone and you likely work and play on a Windows, Mac or even a Linux machine. But the Raspberry Pi, in all its minimalism, maximizes on the means to allow people to get started with actual computing. So much so that the Raspberry Pi is becoming a form of expression - a means of allowing so many of us to transform our imagination into a more tangible and functional form.

Native camera

Like any camera the Raspberry Pi camera has a wide spectrum of uses. It also allows many parameters to be set. But it is important to mention that the Raspberry Pi camera is not like a typical webcam.

What's the difference? Well, like the Raspberry Pi it is an educational tool. The idea is to get started with the workings and functions of a digital camera - to learn what impact each setting has to the final image. What's more, its 5MP sensor takes great HD pictures and video.

The camera is operated through terminal commands. You send commands to control it via the command prompt. The two commands to control the camera are `raspistill` and `raspivid`. Enter each command without any paramaters to see its help. Please read issues 14 and 15 of The MagPi for more details about these commands.

Using the command-line provides an understanding of how the camera is controlled. It allows you to configure commands in almost any combination you want. It also invites you to take the next step and create your own interface for it.

Capturing the interest

Most young kids, like my kids, love playing video games. Games are typically the first type of interaction with computers. But many will reach a point where they don't just want to “play” games - they also want to “make” them as well. At the very least they want to understand more about what goes on 'behind the screen'. But finding an entry point to basic computing is easier said than done.

Fortunately, the Raspberry Pi's camera is proving to be that entry point. It not only captures their interests but also seems to be maintaining it. My kids can now type in commands that do something simple but rewarding and which provide immediate visual feedback. The trick here is to get the kids to take pictures of themselves - usually while pulling funny faces. It is enough for them to type in another command and make another picture.

Pi Vision

Where does Pi Vision fit into all this? Pi Vision is a simple, open source application which attempts to leverage the entry point. The incentive of Pi Vision is to allow for easy use of the Raspberry Pi camera while also serving as an instructional tool.

Each time a picture or video is captured with Pi Vision it shows you what commands are sent to the Raspberry Pi to control and operate the camera. This assists novice users in understanding the command structure. What's more, since Pi Vision is open source you can review and copy the code plus make changes to suit your needs and interests.

In the next section of this article you will find more details about the Pi Vision open source project. For now, assuming that you already have your Raspberry Pi camera setup and running, lets install Pi Vision.

Pi Vision runs on the Raspbian OS. Start the LXDE GUI with the terminal command `startx`. Download the Pi Vision software from http://heywhatsthebigidea.net/?ddownload=1021. Right-click on the downloaded file and choose "Xarchiver". Click on the "Extract files" icon and check the option "Extract files with full path". This will create a new directory called `PiVision_RPi`.

Important! In order to run the Pi Vision application you need to first set permissions. Open the `PiVision_RPi` directory and right-click on the file `rpiCC` - the Pi Vision application. Select `Properties` and from the File Properties dialog select the `Permissions` tab. There is a small check-box which needs to be checked called “Make this file executable”.

Double-click on the Pi Vision application. The start-up page of Pi Vision includes a button called “Test Camera Preview”. This will start a 5 second 640 x 480 (VGA) camera preview in the upper left corner of your screen. Assuming that your camera is set up correctly and the preview is successful then you are now ready to start using your camera with Pi Vision.

You will notice that Pi Vision will display the commands sent in a text area in the upper region of the application. Go ahead and experiment with different settings in the Photo, Video and Settings sections to see the results. Later you can try sending the command instructions yourself. To do this take a photo or video and then select and copy the raspistill or raspivid command in the text area. Open a terminal window by selecting the "Terminal" button at the lower right region of the application. Open the "Edit" menu and choose "Paste" .

Open source

The language of choice for the Raspberry Pi is Python, and with good reason. Python offers simplicity, it is easy to understand and very easy to use. It subscribes to the novice developer, in particular, very well.

Other ways to create programs/applications are also available, but when it comes to Rapid Application Development (RAD) you will need something more advanced. For me the choice was Pascal - more specifically the Free Pascal Compiler (FPC) with the Lazarus IDE.

The Lazarus IDE is an open source development environment to aid writing applications using the Pascal language. It is very powerful and includes just about everything necessary to make really great applications very easily and with minimal effort.

Pascal is not the best language. There is no such thing as "best". The question is what does it take to satisfy the requirements and do it in a good, easy and effective way?

Pascal is a strongly typed language and offers important structured programming techniques. It was the teaching language of choice during the 80s and early 90s. Additionally, the Lazarus IDE and the FPC run on most systems; Windows, OSX, Linux and of course Raspbian. That means you can write code and develop applications for the Raspberry Pi using a powerful IDE which compiles to almost all platforms.

Not surprisingly, Pi Vision was written in Pascal using the Lazarus IDE. In fact the Lazarus IDE is part of the Raspbian OS repository. You can download and install it directly. But should you choose to do so please note that it will consume 0.5GB of storage space on your SD card.

First install Pascal by entering:

```
sudo apt-get install fpc
```

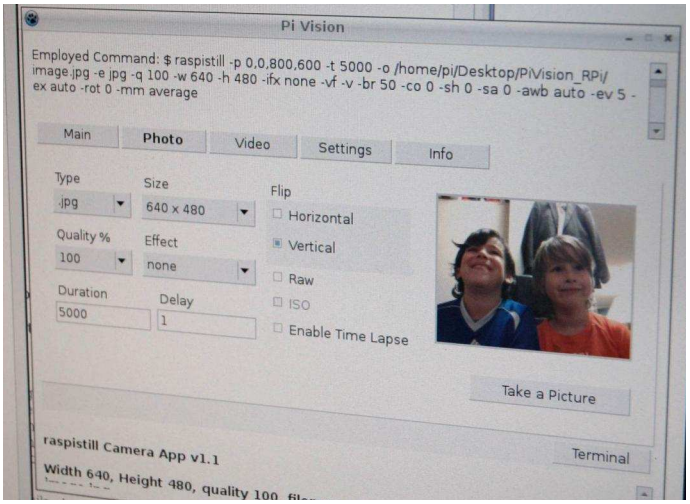
Then install Lazarus by entering:

```
sudo apt-get install lazarus
```

This can take some time to download and install. After it has completed you will find Lazarus in the ¨Programming¨ section of your LXDE application menu.

The Pi Vision project itself can be downloaded and then opened and built using Lazarus. Pi Vision is hosted on Github. Simply download the entire project, unpack it and open it using Lazarus. Full instructions can also be found inside the project.

It should be of interest to note that since you can also install Lazarus on many other operating systems such as Windows or Mac, you can open the Pi Vision project and run it on those systems as well. Obviously this would be only for review.

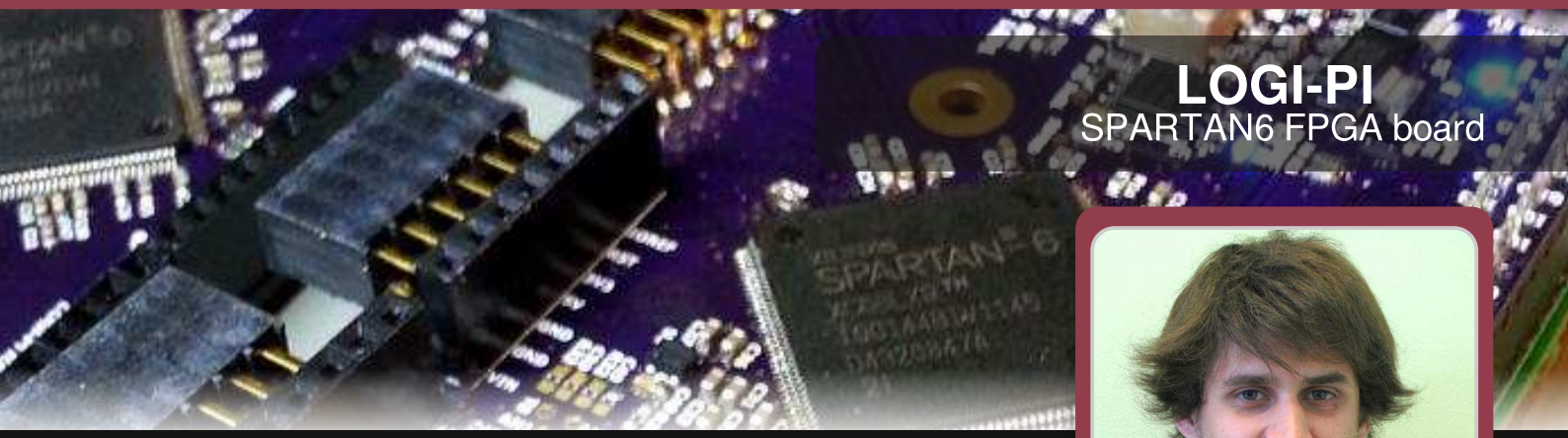


Links

Pi Vision
http://heywhatsthebigidea.net/projects/pi-vision-a-raspberry-pi-camera-controller/

Pi Vision Project Github Repository
https://github.com/local-vision/Pi-Vision

Lazarus Free Pascal Compiler
http://www.lazarus.freepascal.org/



Hardware/software co-design with LOGi-Pi

Jonathan Piat

Guest Writer

SKILL LEVEL : ADVANCED

Hardware/software co-design

Co-design consists of designing a project as a mixture of software components and hardware components. Software components usually run on processors (CPU, DSP, GPU etc.) whereas hardware components run on an FPGA (Field Programmable Gate Array) or a dedicated ASIC (Application Specific Integrated Circuit). This kind of design method is used to take advantage of the inherent parallelism between the tasks in an application and ease re-use between applications.

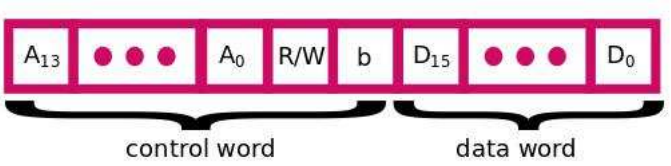
Such co-designs requires the user to :
- partition an application into hardware and software components
- map the components to the resources
- schedule the software components
- manage the communications between the different components

These steps can either be performed by co-design tools or by hand, based on the knowledge of the designer. In the following we will describe how the LOGi-Pi can be used to perform such a co-design to run real-time control oriented applications or high performance applications with the Raspberry Pi.

LOGi-Pi with Raspberry Pi

One key aspect of a co-design is the ability to communicate between the processing units of the platform. The processing units in this case are the FPGA and the Raspberry Pi. In the LOGi project we chose to use the Wishbone bus to ensure fast and reliable communication between the hardware and software components. The Raspberry Pi does not provide a Wishbone bus on its expansion, so we decided to take advantage of the SPI port and design a hardware “wrapper” component in the FPGA that transforms this serial bus into a 16 bit Wishbone master component. The use of this bus allows us to take advantage of the extensive repository of open-source HDL components hosted on opencores.org.

To handle the communication on the SPI bus side, each transaction is composed of the following information :
1) set slave select low
2) send a 16 bit command word with the 14th msb bits being the address of the access, bit 1 to indicate burst mode (1) , and bit 0 to indicate read (1) or write (0).
3) send/receive 16 bit words to/from the address set in the first transaction. If burst mode is set, the address will be increased on each subsequent access until the chip select line is set to high (end of transaction).
4) set slave select high



Such transactions allow us to take advantage of the integrated SPI controller of the Raspberry Pi, which uses a 4096 byte FIFO. This access format permit the following useful bandwidth to be reached (the 2 bit of synchro is a transfer overhead on the spi bus):

1) For a single 16 bit access :(16 bit data)/(2 bit synchro + 16 bit command + 16 bit data) => 47% of bandwidth.

2) For a 4094 byte access : (2047 * (16 bit data))/(2 bit synchro + 16 bit command + 2047 * (16 bit data)) 99.7% of the theoretical bandwidth.

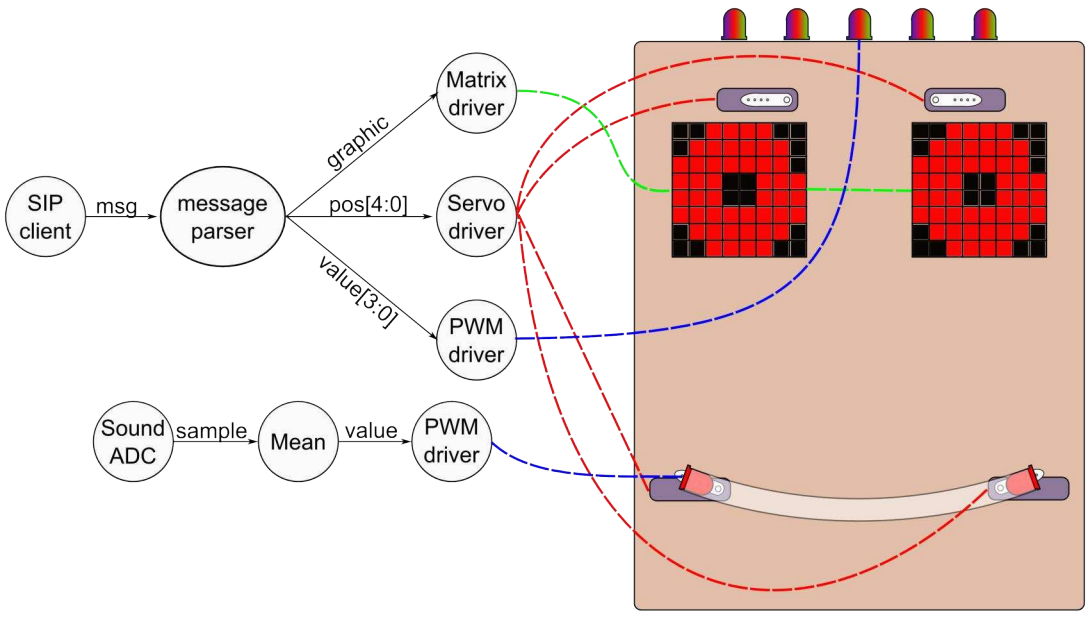
This means that for most control based applications (writing/reading register), we get half of the theoretical bandwidth, but for data based applications (writing/reading fifo/memory) we get 99% of the theoretical bandwidth. One can argue that removing the Wishbone interface in favour of an application specific data communication (formatted packet of data) on the SPI bus could enable the maximum bandwidth, but this would break the opencores.org support and would also break the generic approach we have implemented. The communication is abstracted using a dedicated C API that provides memory read/write functions and also FIFO management for communications based on a FIFO embedded into the FPGA. We also provide a library of hardware components (VHDL) that the user can integrate into designs (servo controller, PWM controller, fifo controller, pid controller, etc.).

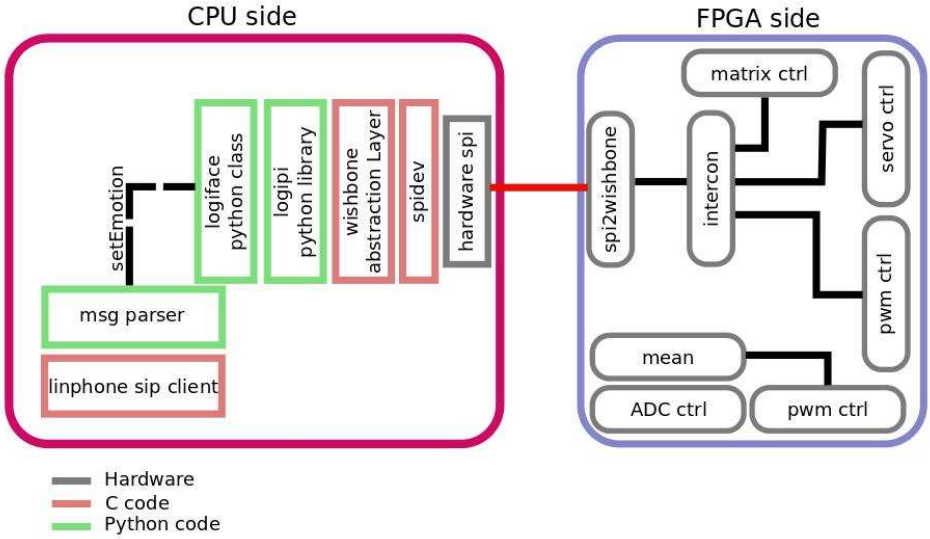
Abstracting communication layer with Python

Because the Raspberry Pi platform is targeted towards education, we decided to abstract the communication over the SPI bus using a Python library that provides easy access to the LOGi-Pi platform. This library can then be used to abstract a complete hardware design using a Python class whose methods are directly linked to the hardware components. This is achieved by writing a Python class that reflects the hardware component inside the FPGA. This class provides access to the peripheral registers or higher level functions. For example, a servo controller embedded inside the FPGA, we can design a Python class with the setPulse, setAngle method. These methods will then communicate with the hardware using the LOGi-Pi library and write the appropriate values to the corresponding peripheral address. This allows us to write the software side of the co-design in python and take advantage of the language features and libraries with high performance side being implemented in the FPGA.

Basic hardware/software co-design with LOGi-face

The LOGi-face is a demo based on the LOGi-Pi platform that acts as a telepresence device. This demo uses a set of actuators (servos, leds) to display emotions on an animatronic face. This application is composed of a set of tasks pictured in the following diagram:





The tasks are partitioned on the LOGi-Pi platform with software components running on the Raspberry Pi and hardware components on the LOGi-Pi. The choice of software components was made to take advantage of existing software libraries such as the Espeak text to speech engine and Linphone SIP client. The hardware components take care of time critical tasks such as servo driver, LED matrix controller, ADC controller and PWM controller. Further work could lead to optimize parts of the Espeak tts engine in the hardware (FPGA).

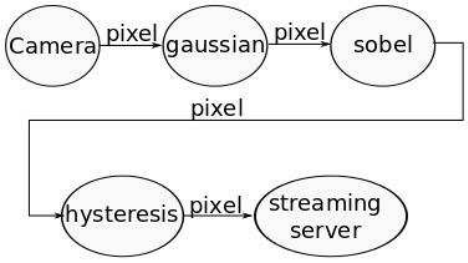
All the missing software components (emotion generation from sip clients chat session) were written in Python, to leverage the language features and ease development. The hardware side was abstracted using a dedicated Python class to abstract the peripherals inside the FPGA.

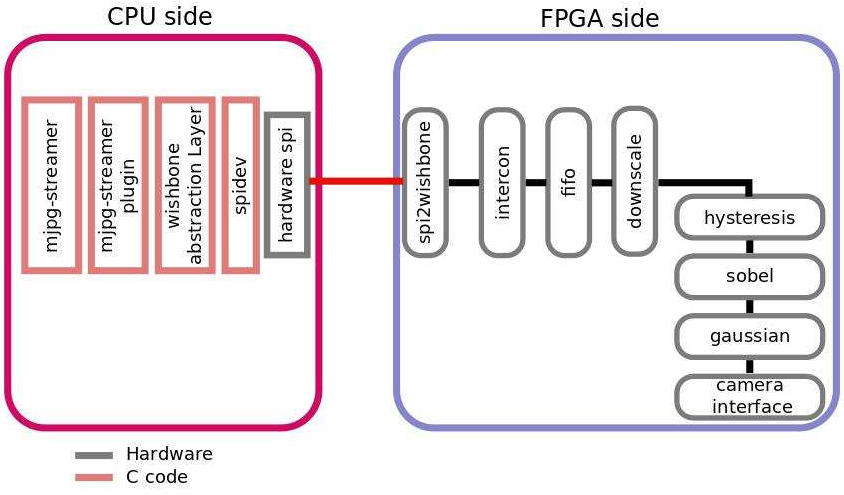
Video application using the LOGi-Pi

Hardware/software co-design tends to be relevant to high performance and time-critical applications. Image processing is the kind of application that requires high throughput, and frame accurate synchronisation, to provide relevant results for higher level filtering applications (feature tracking, visual odometry, etc.). In the following example we developed a canny edge detector leveraging the processing power of the FPGA and the flexibility of the Raspberry Pi for the encoding and networking tasks.

The hardware components for the FPGA side are extracted from a project I initiated for hardware computer vision components [https://github.com/jpiat/hard-cv]. In this application a communication bottleneck on the SPI bus forces us to send a downscaled image to the Raspberry Pi, to meet the framerate constraints. All of the image processing is performed on QVGA (320x240) frames at 30 FPS (the hardware can run flawlessly for larger format and higher frame-rate) and the result is downscaled to QQVGA (160x120). A custom MJPEG-streamer plugin reads a frame on the Raspberry Pi side, encodes it as a JPEG using libjpeg and passes the frame to MJPEG-streamer for streaming [see video : http://www.youtube.com/watch?v=0HwEcmF5wxw].

The partitioning given on the right leads to the implementation given below.





Conclusion

The LOGi-Pi is a great platform to develop co-designed applications at low cost and in a small package. The ARM11 processor has plenty of computing power, while the Spartan 6 LX9 FPGA provides enough logic for signal

processing applications. This kind of platform can easily be integrated into electrical engineering courses, because the a student can first learn software development with the Raspberry Pi (in Python for example), then use the pre-developed hardware architecture on the FPGA to learn parallelism and then learn hardware development with the LOGi-Pi. For example, the video application could be taught in the following steps :

1) Code a line detection algorithm using the OpenCV library in python

2) Take the code of the last step and replace the OpenCV calls by calls to the hardware (with the FPGA implementing Canny). Learn how to re-schedule the code to take advantage of the available parallelism.

3) Code a small part of the all Canny pipeline in hardware and use the developed architecture in the application.

4) Use the output of the architecture to drive a line following robot.

All of the code from this article can be downloaded from https://github.com/fpga-logi/Logi-projects.

KICKSTARTER

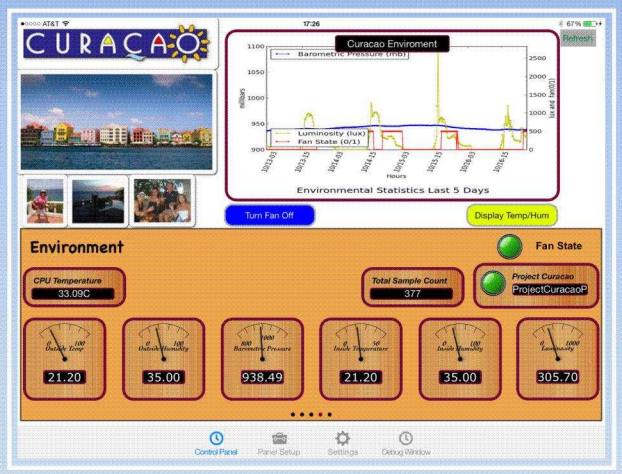
The LOGi-Team is hard at work preparing documentation, drivers, MFG logistics and support, to initiate a kickstarter campaign to launch of the LOGi FPGA boards into the wild. We plan to offer an array of peripheral boards that will allow users to learn to work with FPGAs and run some fun and interesting high performance applications, using the Raspberry Pi and the LOGi-Pi. We will be posting updates in regard to the kickstarter on the ValentFx Blog Kickstarter page[1].

If you just can't wait to get your hands on a LOGi-Pi and you have a great idea for an interesting project, see our LOGi-board competition page [2]. You can submit a project plan and we if it looks like a winner we will get a LOGi-Pi out to you right away.

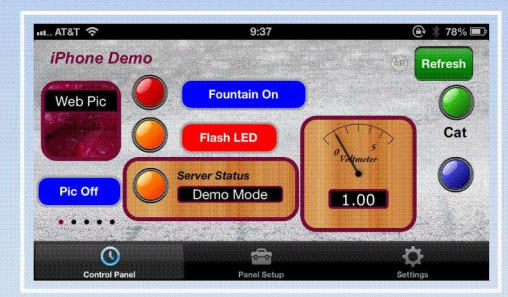
[1] http://valentfx.com/logi-blog/item/logi-kickstarter
[2] http://valentfx.com/logi-blog/item/logi-contest

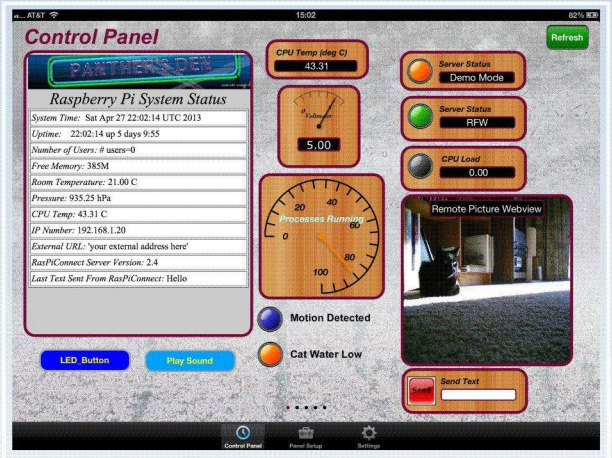
RasPiConnect

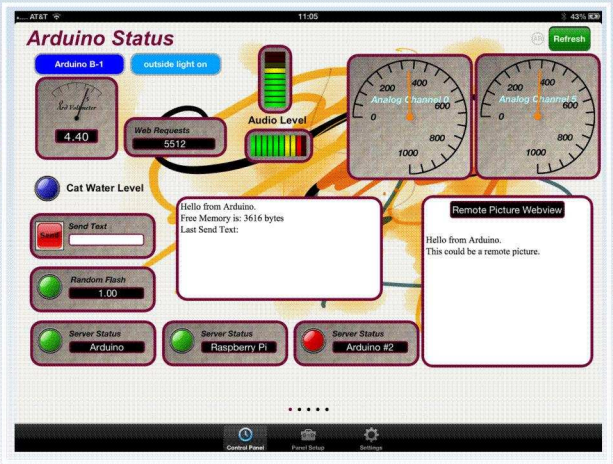
Connect your Raspberry Pi to the World!



Now Supports Arduino with in-app purchase







Available on the App Store

RasPiConnect connects your Raspberry Pi to the outside world.

It allows you to control virtually anything you connect to your Raspberry Pi from your iPad or iPhone.

- EASY to setup - no syncing required
- Buttons, gauges, webpages, webcam pictures and more!
- Build your pages on your iPad/iPhone
- Supports multiple Raspberry Pis and multiple Arduinos
- Five pages of control panels
- Unlimited Controls
- Exchange your panels with friends
- Supports any computer that supports Python (Windows, Linux, etc.)
- Now allows custom backgrounds

MiloCreek
Where Innovation Flows

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

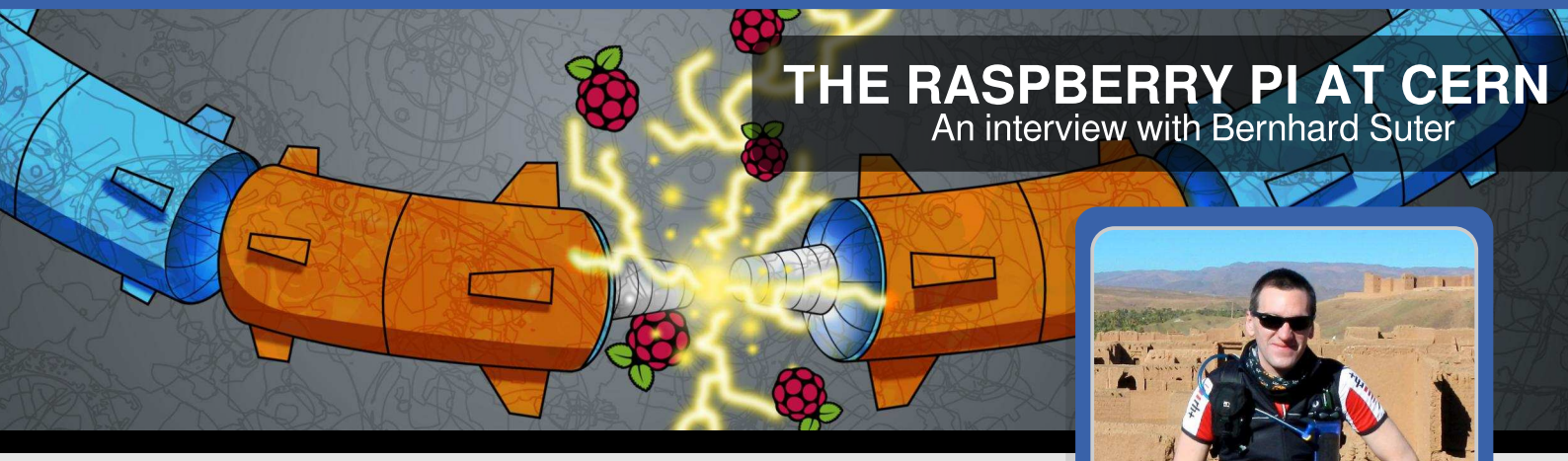
MADE FOR MAKERS

OPEN SOURCE

A UK ENTERPRISE

Find out more online @ WWW.PIBOT.ORG

FOLLOW #THEPIBOT



THE RASPBERRY PI AT CERN

An interview with Bernhard Suter

Colin Deady

MagPi Writer

On the 5th of October CERN hosted its third Raspberry Pi event. Approximately 160 people were engaged with tutorials, workshops and talks throughout the day. What is noticeable about the Raspberry Pi @ CERN events is that they are presented in both French and English, increasing accessibility to the Raspberry Pi as a result.

Bernhard Suter, a software engineer from Google was present and gave three talks at the event. Following the event The MagPi took the opportunity to speak to Bernhard about the event at CERN, the Raspberry Pi in Switzerland and some of his projects.

Why did you decide to support the Raspberry Pi @ CERN event?

The organiser contacted Google to see if anyone would be interested in events at CERN and I volunteered. I have been interested in the Raspberry Pi since before its release and was looking for events in Switzerland. I had no idea what the audience for my talks was going to be like. I knew that the Raspberry Pi attracts technical people, but I was not sure if my talks were going to be to a group of six year olds or winners of Nobel Prizes in physics, especially given that CERN is one of the heaviest users of sophisticated computer technology in Europe. I decided I was going to give a motivational talk: why computing is the basis for many exciting careers, representing a professional perspective from the industry. I also gave two technical tutorials that were more in depth for people more interested in the details.

Up until the last minute the language I would give the talks in was uncertain. The language of the Raspberry Pi itself is a challenge outside of the English speaking world. The documentation is mostly in English for example. Language can be a challenge to the adoption of the Raspberry Pi outside the English speaking world. Based on an audience survey we opted for my motivational talk on YouTube to be given twice, once in French and again in English as Geneva is a very international city.

The CERN events have been running for a while now and seem to be gaining good traction. What was your impression of the day as a whole?

My impression was that the event drew a good number of primary school age children with their parents on one hand as well as some hackers, computer users interested in a cheap embedded platform, people who were curious about the Raspberry Pi, including some science teachers from the local schools. My talk on YouTube took place in the same auditorium where the Higgs Boson discovery had been announced earlier this year (which was only a tiny bit intimidating...). I was happy about the many children in the audience, who were very engaged and asking a lot of good questions.

One room had a number of Raspberry Pi desktop setups running Scratch. The kids, who were mainly young, were clustered around the monitors at lunchtime creating a racing car game: a whole roomful of kids around these mini workstations. This was quite successful as the kids were clearly engaged. The atmosphere was fun and educational. People were working hands-on to support each other.

One really good thing was that someone from Farnell France was present selling a starter kit, the camera board and the Gertboard. People who came out of curiosity could take a Raspberry Pi home ready working. This was a good move as it meant the the barrier to entry was very low. We sometimes forget that even the Raspberry Pi can be daunting to set up if one is new to computers, for example: flashing an SD card for the first time.

There is no cottage industry of add-on boards in Switzerland unlike the UK. Therefore it is good at events to go back to the basics and show people what they can do with the Raspberry Pi. The event at CERN included a lot of introductory assistance for people.

The Python and Minecraft workshop drew a smaller crowd, but that is likely because it was later in the day at 4pm and some had already left by then. There is also always the temptation just to play Minecraft of course.

It would be good to be able to attract older kids, from secondary school, who have career choices coming up. Especially as there is the opportunity to show them the very cool and very real things they can do with computers. Compared to other areas of science and technology, where for example you couldn't build a working particle accelerator at home, in computing you can build your own very real web service or robot with just a Raspberry Pi, Linux and some other open-source tools. The message is that you can be working with the same technology set as Google, Facebook and other major IT tech companies in the world.

You also gave talks on using zero configuration networking and Python with SQLite. Why these particular topics?

Zero-conf networking is important as the future of networking is moving towards IPv6. Therefore something like Bonjour networking is very useful where the user refers to a name like rpi1.local instead of an IP address that may dynamically be changed by the network. Learning names is easier as in real life these are the things by which you identify.

The Raspberry Pi is a great embedded platform for physical computing. Therefore "just works networking" is useful for people who want to use GPIO or sensors. I've noticed that interfacing with the real world is central to many of the articles in The MagPi and not having to worry about configuring the network connection makes this more straightforward.

Programming with SQLite (http://www.sqlite.org) and Python was the other technical talk you undertook. Why do you feel Python and SQLite are such good fits for the Raspberry Pi?

Traditionally we refer to the "LAMP stack" for web applications: Linux, Apache, MySQL and PHP / Perl / Python. SQLite is an alternative to MySQL

that requires hardly any installation and no dedicated server. If you need to scale up at some point you can replace it with MySQL or PostgreSQL or even a database cluster as required. SQLite doesn't need restarts and it is easy to fix issues in the database. Extracting data for use (e.g.: in data logging) is straightforward.

In your talk you mention "Think of SQLite not as a replacement for Oracle, but as a replacement for fopen()". Would you like to elaborate on what I believe is an important point? Especially for example, for data logging, which is a common activity via the GPIO pins.

I took this straight from SQLite and I think it is a good crisp description of what it is about. For example you can use SQLite for easy storage of configuration settings that are easier to parse and update from within an application. We shouldn't ask "why should I use a database?" Instead we should ask "why shouldn't I?" Text files are easily readable for humans, but not so easy to parse for computers especially when concurrent access is needed to update the records. SQLite is a gentle introduction to databases.

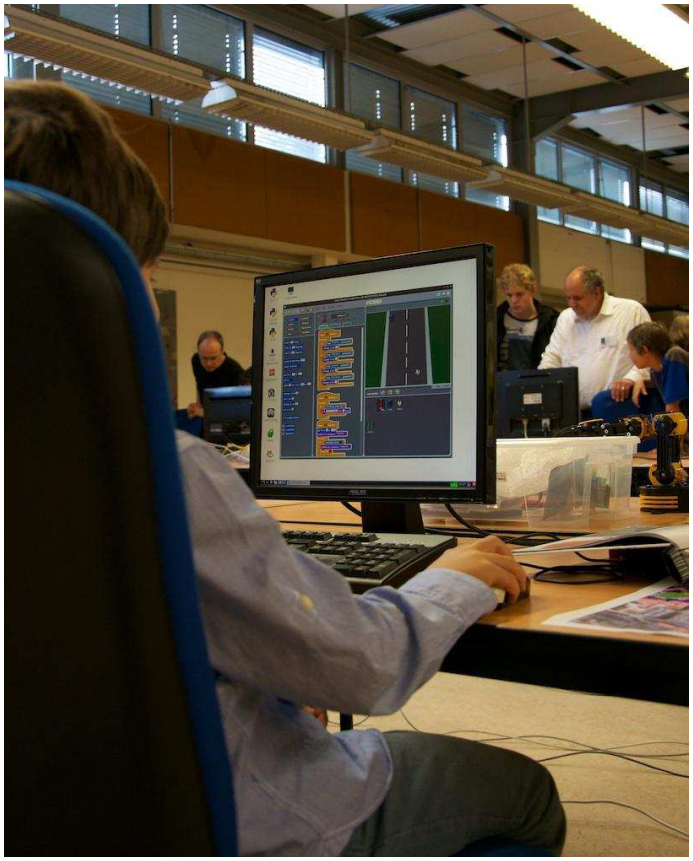
For example you could take my blog post (http://blog.kugelfish.com/2013/05/raspberry-pi-internet-access-monitor.html) on an internet connection monitor using SmokePing and write the results to a SQLite database instead. Or it would be of interest to monitor temperature evolution in various rooms in a house with visualisation layers on top. This would make for interesting graphing, for example using the public Google Charts visualisation library. Sensors would output data which is written via Python to a SQLite database and then presented via a toolkit to render the visualisation.

Python itself is a language that has upwards scalability. Python, like SQLite, is a gentle introduction to programming, but can scale to applications as large and complex as YouTube.

Were you hoping people would see the link between your two technical talks?

Yes. The other presenters, such as Pierre Freyermuth who presented how to log data from a Geiger Counter with the Raspberry Pi (http://pierremuth.wordpress.com/rpiadventure), were in fields that require sensor measurement, monitoring, time series analysis and uploading of data. Combining their data with SQLite and zero-conf networking could make this even easier on the Raspberry Pi. A couple of science teachers in the audience mentioned that they would take this further.

There is a real use case for using Raspberry Pi's in school not just to learn about computers themselves, but as a tool for science. CERN is a great example of how important computers are today in experimental science. Using a Raspberry Pi for data collection and analysis in science class would mean applying the same concepts, just at a smaller scale.



How well known is the Raspberry Pi in Switzerland and in schools in particular?

As far as I can tell, the Raspberry Pi is not really well known in Switzerland or much used in schools. Maybe this is because access to computers in schools has not really been as much a problem as what to do with them. There are negotiations going on in government to harmonise the next generation of lesson plans in computing and industry is lobbying for computer science to be a bigger piece of the curriculum. There is a danger it will be lumped in with Media Studies and would not be a technical subject. We will need students who are also familiar with the maths and science side of computing. In Media Studies it becomes more about ethics, responsible and safe

use of digital media, for example the implications of posting pictures on Facebook. There is a lot of pushing for computing as a technical subject, but the Raspberry Pi is not part of the story from what I have seen so far.

I think what is more important than teaching a particular programming language or application, it would be essential that every student is introduced to how computers work and learns the concepts of "algorithmic thinking" which are the foundations for all programming.

From what I can remember, in Switzerland, computing has never been integrated much into the school curriculum unlike for example in the UK with the BBC Micro. Computer interest was mainly happening outside of school as a hobby. But home computing mostly died out in the 1990s with the growing maturity and sophistication of PCs. Now that we are moving into the post PC world, computers are even more common, more polished. For example, my daughter was able to use an iPad by herself from a very young age. But at the same time, computers have also become more closed and less hackable and we have moved from tinkering with computers to simply using them. For example, undertaking your own game programming is intimidating as there is a feeling you have to equal the games from the big publishers, but in the 1980s you could pretty much come up with the same quality game at home as the companies producing them which was encouraging. This applies to all software. For example I wrote a text editor in assembly once as the one I had was too slow.

Despite not being very known yet, do you think that the Raspberry Pi could help computer education in Switzerland?

Certainly yes. It is true that you don't need a Raspberry Pi to program in Scratch or Python for example and in fact using existing PCs might be much easier for that. However the great advantage of a Raspberry Pi for a child is that this is a computer they can own, customise and no matter what they do, can hardly break. Even if the OS no longer boots, you can simply replace the flash card and it is working again! Unlike with other computers at school or at home, there has to be no fear of breaking things, no need to hold back on experimentation!

Being such a great device for low-cost, low-power physical computing, maybe this is where the Raspberry Pi can really shine. Maybe the Raspberry Pi isn't meant to replace a PC, but to be used for building all kinds of new devices, which can be easily connected to an existing PC as a "side-car" or plugged into a home network. This is what I had in mind with the zero-conf networking talk.

Lastly, can you tell us a bit about yourself?

Like many people involved with the Raspberry Pi, I grew up during the golden age of home computers, where programming was new and exciting enough that even non-technical magazines had articles about computers programming and printed introductions in BASIC programming. I spent long hours at friends houses who had computers and we did extensive BASIC programming. My first computer was an Amiga 500, which provided a great introduction to "real" operating systems. I taught myself C & M68k assembly, the basics of computer architecture and operating systems from magazines and books (I fondly remember the excellent "Amiga Intern" from Data Becker, a German publisher).

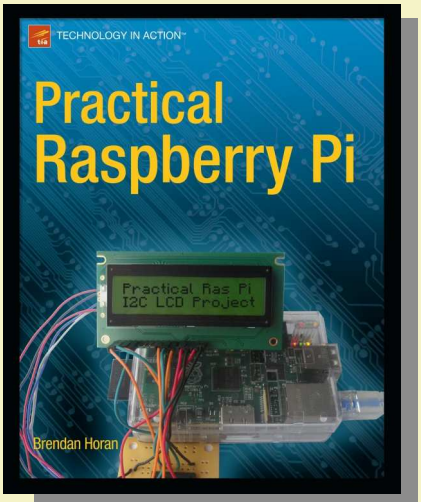
I first heard about Linux in about 1991 and after some research upgraded to a i386 PC specifically to run Linux. I was just starting university and had my first Internet access on the old VAX of the EE department, where I downloaded floppy images for one of the first Linux distributions. When I got an Internship at AT&T Bell Labs Research, I was very excited to be working along some of my heroes and demi-gods of the computer world, I guess I was something of a UNIX groupie ;-)

Throughout my career I have always been lucky to work with very bright people who were very good at what they were doing. In order to become a good software developer, it probably helps to do serious programming in extra-curricular activities long before going to university. I hope that some future entrepreneurs, innovators and computer experts will one day be able to say that it was the Raspberry Pi which sparked the launch of their fascinating career.

Bernhard has kindly made the slides from his three talks available online:

YouTube (non-technical): http://goo.gl/GktbI0
Zero-config networking: http://goo.gl/31D5Qk
SQLite & Python: http://goo.gl/jOfsss

Practical Raspberry Pi
Brendan Horan
APRESS



Practical Raspberry Pi takes you deep into the hardware of this amazing, pocket-sized computer – and shows you all that it can do!

Brendan Horan quickly gets you started with a series of fun, practical projects, all of which require only minimal programming experience. From building a temperature sensor to making a real time clock, these projects provide you with invaluable first-hand experience of the Raspberry Pi's capabilities.

Along the way you will also gain special insight into the Raspberry Pi's hardware, including how it can be so powerful and still so small and inexpensive, why it's suitable as a video player and how you can customize it for different tasks. This includes a look at running operating systems such as Android and RISC OS.

By the end of the book you will fully understand how the hardware and software go together on the Raspberry Pi. Additionally, you will be able to configure almost any operating system for the Raspberry Pi to meet your specific needs and tune it for good performance.

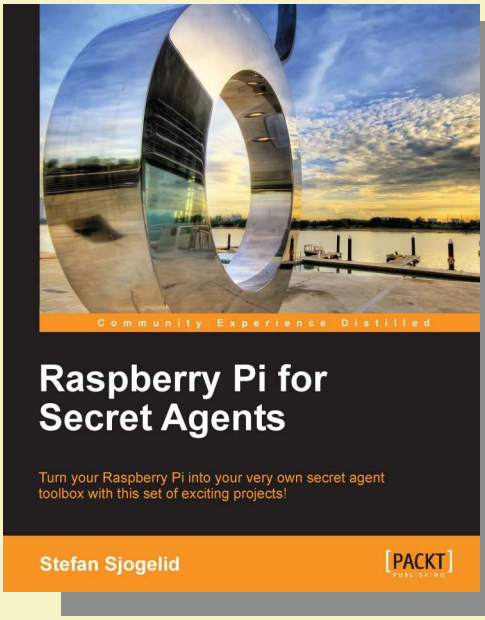
Unleash the full power of this inexpensive and fun little computer with Practical Raspberry Pi!

The MagPi and Bookaxis are pleased to offer readers a 30% discount on these two titles until November 30th. To claim, order from www.bookaxis.com/magpi and quote promo code MAGPI18.

Raspberry Pi for Secret Agents
Stefan Sjogelid
Packt Publishing

Raspberry Pi for Secret Agents starts out with the initial setup of your Raspberry Pi, guides you through a number of pranks and secret agent techniques and then shows you how to apply what you've learnt to the real world. Through a series of fun, easy-to-follow projects, you will learn how to set up audio and video surveillance, explore your Wi-Fi network, play pranks on your friends and family and even learn how to free your Raspberry Pi from the constraints of the wall socket!

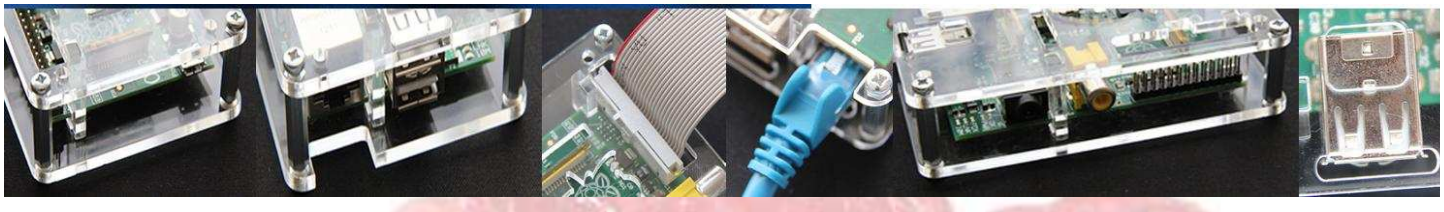
With this book you will be able to learn how to configure your operating system for maximum mischief and start exploring the audio, video and Wi-Fi projects within. Discover how to record, listen, or talk to people from a distance and how to distort your voice. You can even plug in your webcam and set up a motion detector with an alarm, or find out what the other computers on your Wi-Fi network are up to. Once you've mastered the techniques you can then combine them with a battery pack and GPS module for the ultimate off-road spy kit.



This book is for all the mischievous Raspberry Pi owners who would like to see their computer transformed into a neat spy gadget, to be used in a series of practical pranks and projects. You don't even need any previous experience with the Raspberry Pi or Linux itself to explore the exciting projects and examples.

So why not pick up a copy today?

NOVEMBER 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 new Raspberry Pi 512MB Model B, an exclusive Whiteberry PCSL case, 1A PSU, HDMI cable, 16GB NOOBS memory card, GPIO Cobbler kit, breadboard and jumper wires!

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

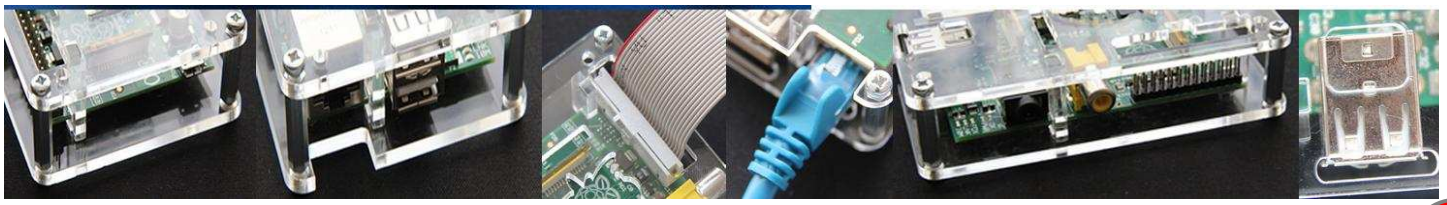
Closing date is 20th November 2013. Winners will be notified in next month's magazine and by email. Good luck!

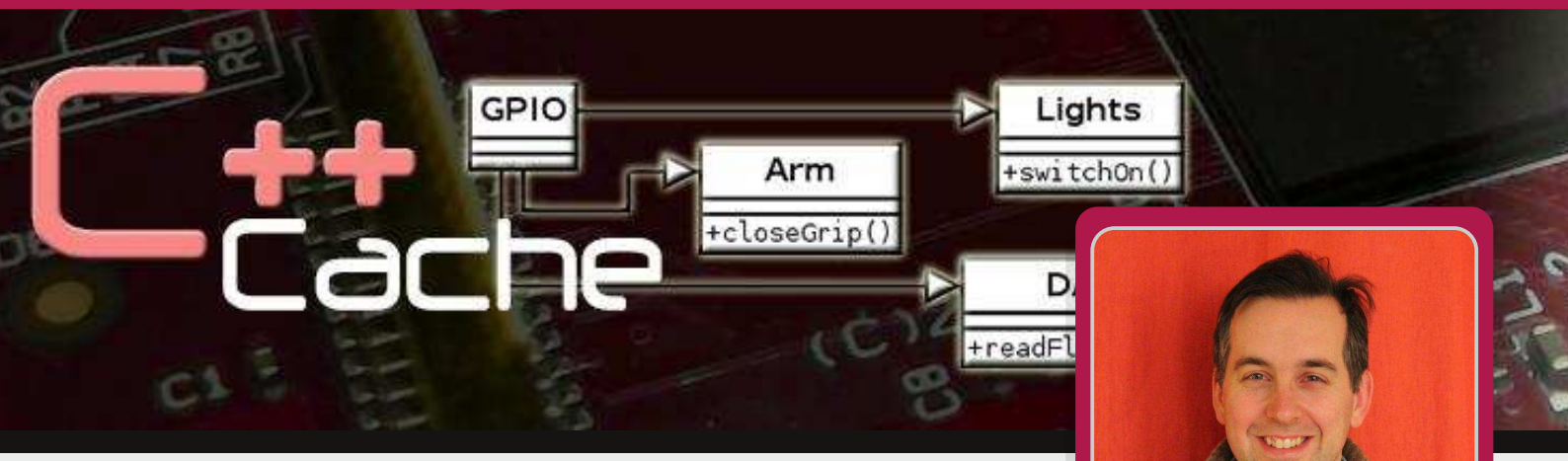
To see the large range of PCSL brand Raspberry Pi accessories visit
http://www.pcslshop.com

October'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 **Tom Cheesewright (Manchester, UK).**

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





4 - String streams

W. H. Bell

MagPi Writer

SKILL LEVEL : ADVANCED

Welcome back to the C++ Cache. In the last article in Issue 10, some more details of streams were introduced. C++ allows the use of C style input and output string operations such as `sprintf` and `sscanf`, but also provides string streams. String streams can be used in a program by including the `sstream` header file.

Input string streams

```
#include <string>
#include <sstream>
#include <iostream>

int main () {
  double d_value = 0.0;
  int i_value = 0;

  std::string str1("7.045 ");  // A space is needed
  std::string str2("12345 ");  // A space is needed

  // Very strange things happen if the trailing space is removed: the
  // next >> operation fails and no attempt to use seekg() or str()
  // will clear the problem.

  std::istringstream inStr(str1);

  inStr >> d_value; // Collect a double value
  inStr.seekg(0); // Reset the string to be re-read
  inStr >> i_value; // Collect an int value

  std::cout << "First String Stream:" << std::endl;
  std::cout << "d_value=" << d_value << std::endl;
  std::cout << "i_value=" << i_value << std::endl << std::endl;

  inStr.str(str2); // Set the string value of the stream to the second string.
```

```
  inStr >> d_value; // Collect a double value
  inStr.seekg(0); // Reset the string to be re-read
  inStr >> i_value; // Collect an int value

  std::cout << "Second String Stream:" << std::endl;
  std::cout << "d_value=" << d_value << std::endl;
  std::cout << "i_value=" << i_value << std::endl << std::endl;

  return 0;
}
```

In this example, an input string stream is used to convert a string to a double and an integer value. At the start of the program, a double and an integer variable are declared. Then two strings are instantiated and assigned two strings. The first string contains a string representation of a double number, whereas the second string contains the string representation of an integer number. A trailing space is needed to delimit the number from the end of the string. The next step is the instantiation of an input stream called `inStr`. When the `inStr` stream is instantiated, the string `str1` is assigned to it. Once a string stream is associated with a string, it can be used in the same way as other streams. The string value that is in the string associated with `inStr` is then read once as a double value and once as an integer value. In between the two stream read operations, the `seekg()` function is used to rewind the position in the stream to the start of the stream. The two numerical values are then printed on the screen. In the second part of the program, the input string stream is assigned the second string. Then it is also read as an integer and as a double value. The program converts the string to the appropriate number type, without the need to explicitly state the type needed.

Stream states and error bits

If an input string cannot be converted into the form requested the `failbit` will be set. This could happen if a program is given a dictionary word rather than a number, or the number given is too large for the type requested.

```
#include <sstream>
#include <iostream>
#include <string>

int strToLong(const std::string &inputString, long &l_value) {
  long longValue;
  std::istringstream inStr(inputString);
  inStr >> l_value;
  if ((inStr.rdstate() & std::ios_base::failbit) > 0) return 1;
  return 0;
}

int strToDouble(const std::string &inputString, double &d_value) {
  double doubleValue;
  std::istringstream inStr(inputString);
  inStr >> d_value;
  if ((inStr.rdstate() & std::ios_base::failbit) > 0) return 1;
  return 0;
}
```

```
int main() {
  std::string str = "1213333333333333222333";
  long l = 0;
  double d = 0.;
  int ret_val;
  if(strToLong(str,l) != 0) {
    std::cerr << "ERROR: strToLong failed!" << std::endl;
  }

  if(strToDouble(str,d) != 0) {
    std::cerr << "ERROR: strToDouble failed!" << std::endl;
  }

  std::cout << "l=" << l << ", d=" << d << std::endl;

  return 0;
}
```

When this program runs, the `strToLong` function call will fail since the number provided in the string is too long for the memory allocated for a long variable type. The `strToDouble` will succeed though and the number will be converted to double format. Try changing the value of `str` to a dictionary word or to a smaller number. If the number is small enough, then both functions will be successful. If a dictionary word is used, then both functions will fail.

This example program also demonstrates the use of string streams within functions. Since the string stream is instantiated each time the function is called, it is not necessary to re-wind the string stream and the string assignment is only made when the stream stream constructor is called. The functions use `const` references to the `inputString` variable, since there is no reason for the functions to update the value of `inputString`. The other parameter is given as a reference, to avoid the need to explicitly pass a pointer to the function. The state of a stream can be queried by calling the `rdstate()` function. Then to determine if a problem has occurred the `failbit()` is used with a bitwise and operator. Other valid iostates are `goodbit` (no errors), `eofbit` (end-of-file reached when reading data) and `badbit` (read or write error).

Output string streams

To finish of this month's article, here is a simple example to demonstrate an output string stream:

```
#include <string>
#include <iostream>
#include <sstream>

int main () {
  double pi = 3.141592; // A double with an approximate value
  std::string outputString;  // Instantiate a string
  std::ostringstream outStr(outputString); // Output string stream
  outStr.precision(5); // The default is 6.
  outStr << "pi=" << pi << ", "; // Append to string stream
  outStr << "sci pi=" << std::scientific << pi; // With scientific format
  std::cout << outStr.str() << std::endl; // Print the resulting string
  return 0;
}
```

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

CAS Somerset Teachers Hub Meeting

When: Wednesday 6th November 2013, 4.15pm until 6.30pm
Where: Millfield School, Room IT6, Butleigh Road, Street, Somerset, BA16 0YD, UK

The third Computing At School Teacher's meeting in Somerset. http://cassomerset-es2.eventbrite.co.uk

Dubai Tech Fest 2013

When: Friday 8th November 2013, 11.00am to 4.00pm (PST)
Where: Dubai English Speaking College, Dubai, UAE

Three day festival of technology - CPD, speakers, presentations, cyber safety, Python and Scratch, including the UAE's first Raspberry Jam. http://dubaitechfest-es2.eventbrite.com

Corso Raspberry Pi

Quando: Domenica 10 novembre 2013 dalle 09.00 alle 18.00 (GMT+01.00 Italia)
Dove: Campus La Camilla, 267 Via Dante Alighieri, 20863 Concorezzo, Italia

Esploriamolo insieme, muovendo i primi passi con linux, python, Internet e l'elettronica. Biglietto: http://corsoraspberrypi-es2.eventbrite.it

DigiMakers (formally Raspberry Pi Bootcamp)

When: Saturday 16th November 2013, 9.30am until 4.00pm
Where: At-Bristol Science Centre, Anchor Road, Harbourside, Bristol, BS1 5DB, UK

A series of community technology events aimed at children, parents and teachers. This is an introduction to 'making' in the digital world. https://www.eventbrite.com/event/8385319717

Manchester Raspberry Jam XVI (Birthday Jam)

When: Saturday 16th November, 10.00am until 5.00pm
Where: Manchester Digital Laboratory, 36-40 Edge Street, Manchester, M4 1HN, UK

Come along and bring your Raspberry Pi to life! Share ideas and see talks and demos. The Raspberry Pi Foundation will be present. There will be cake. Tickets from: http://mcrraspjam16-es2.eventbrite.co.uk

Feedback & Question Time

Guys I just wanted to drop you a quick note to say thanks for The MagPi. I got my Rasberry Pi a week or so ago and I am slowly making my way through your back catalogue, trying out as much stuff as possible.

I'm a complete Noob to programming/coding, building circuits, etc. and to be honest I'm having the most fun I've had since I used to sit typing lines of Basic into a BBC model B way back in the 80's.

My two girls (age 7 & 9) enjoy the Scratch Patch as they are just being introduced to it at school. Thanks to The MagPi they're already ahead of the game and making little routines to do various tasks and with reasonable confidence. They actually enjoy it when they go wrong and that cat does something unexpected!

So all in all, great mag, excellent content and pretty well executed. I did have one observation though, and that would be that sometimes you maybe forget that nuggets like me are reading your content. It would be great if you could sometimes explain WHY you are doing something as well as how to. For instance, i am enjoying your projects in the "In Control" section and currently waiting for some bits to build the Part 3 of the interfacing project for beginners. The layout diagrams, etc. are all brilliant and even I can follow them. It would be nice though if for instance you could explain why we are using Pin 15 and 13 on the GPIO. Would using a different pin give a different result?

Keep up the good work though and I look forward to getting through the back issues and waiting for the latest issue to hit my Newstand subscription shelf.

Many Thanks
Mike Burton
Stepping out on the Noob to Geek journey.

Great to see an article in the current issue of The MagPi (page 12, issue 17) on connecting a TMP36 temperature sensor to the Raspberry Pi via an Arduino.

I used a near identical part connected to the analogue port of the BBC Micro as part of my AO-Level Computer Science project in high school in 1988!

I inserted the sensor into a plastic tube so that the end of the sensor protruded from the end of the tube, and embedded the base of the sensor and wires in epoxy. I was then able to use the sensor to measure the temperature of liquids.

Andrew Waite

I'm slowly working through the copies of The Magpi and really enjoying it.

Stuart Smith

For 'Claire' on the back cover of the October issue of The MagPi. http://electronicsclub.info/ will prove a good starting place.

Iain Johnstone

Did you enjoy issue 18 of The MagPi? Would you like to help us create more issues? We are desperately seeking layout and proofreading volunteers. If you think you could help please contact: editor@themagpi.com. Thank you.