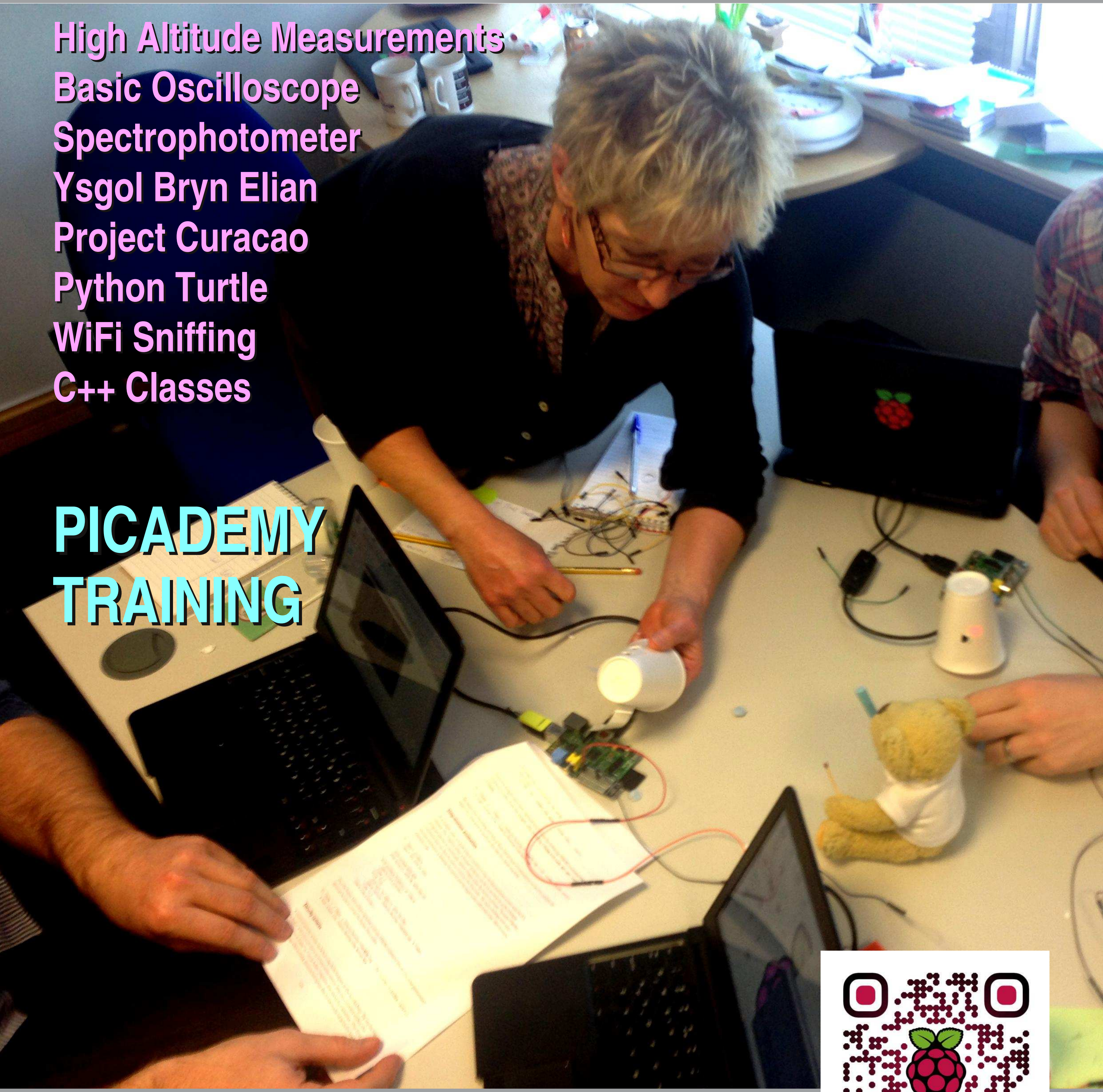
ISSUE 24 - JUN 2014

Get printed copies at themagpi.com

The MagPi

A Magazine for Raspberry Pi Users



High Altitude Measurements
Basic Oscilloscope
Spectrophotometer
Ysgol Bryn Elian
Project Curacao
Python Turtle
WiFi Sniffing
C++ Classes

PICADEMY TRAINING

ISSN 2051-9990

http://www.themagpi.com

Welcome to issue 24 of The MagPi magazine.

This month's issue is packed cover to cover with something for just about everyone!

We kick off with Daniel Pelikan's 'Raspberry Pi Scope', an article which describes in detail how to use the Raspberry Pi as a 10 MSPS scope. We follow this with the concluding part of Michael Petersen's Weather Balloon series, where he looks at the code used in their Multi-Sensor Array before moving onto a great article looking how to build a spectrophotometer using the Raspberry Pi and Wolfram language.

John Shovic shares with us his penultimate article, part five, of Project Curacao. John describes actually deploying the monitor and also reveals some results from within the first 8 weeks of it's data collection. He finishes the article with some suggestions of future upgrades and we look forward to part six to see how this project has evolved further.

We have a MagPi exclusive interview with Carrie Anne Philbin from the foundation on the first Picademy at Raspberry Towers and we pay homage to 1980's LOGO with an article looking at the use of Python with the module 'Turtle' to produce similar graphics. Allen Heard, Head of IT at Ysgol Bryn Elian High School describes his fantastic Tech-Dojo events inspiring tomorrow's programmers and we start a new series looking at improving our understanding of Wi-Fi in Richard Wenner's education article 'Raspberry Spy'. We finish off by returning to C++ cache looking at object-object communication.

We had better begin...

Ash Stone

Chief Editor of The MagPi

The MagPi Team

Ash Stone - Chief Editor / Administration / Layout
W.H. Bell - Issue Editor / Layout / Administration
Bryan Butler - Page Design / Graphics
Ian McAlpine - Layout / Testing / Proof Reading
Matt Judge - Website / Administration
Aaron Shaw - Admin / Proof Reading
Nick Hitch - Admin / Layout
Colin Deady - Layout / Testing / Proof Reading
Tim Cox - Proof Reading
Chris Stagg - Testing / Proof Reading
Paul Carpenter - Testing
Dougie Lawson - Testing / Proof Reading
Nick Liversidge - Proof Reading

Contents

4 **PISCOPE**
Using the Raspbery Pi as a scope

12 **STUDYING ATMOSPHERIC POLLUTION WITH A MULTI-SENSOR ARRAY**
Part 2: Implementing the code

18 **PROJECT CURACAO**
Part 5: Deployment and results

22 **WOLFRAM ANALYSIS: MEASURING LIGHT ABSORPTION**
DIY chemistry lab: building a spectrophotometer

26 **PICADEMY**
An interview with Carrie Anne Philbin

32 **PYTHON AND TURTLE GRAPHICS**
Bringing Testudines out of the '80s

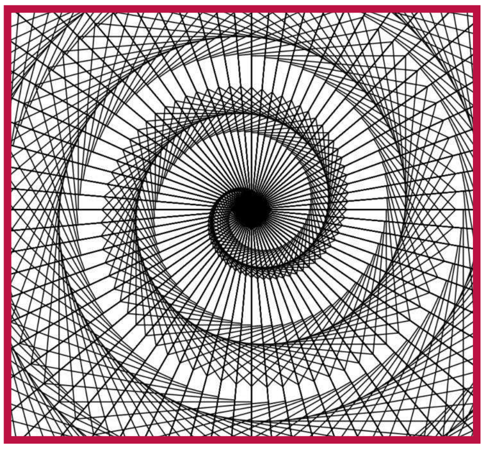
34 **SCHOOL REPORT: DISCUSSION OF MAKING SESSIONS**
Tech-Dojo with the Raspberry Pi

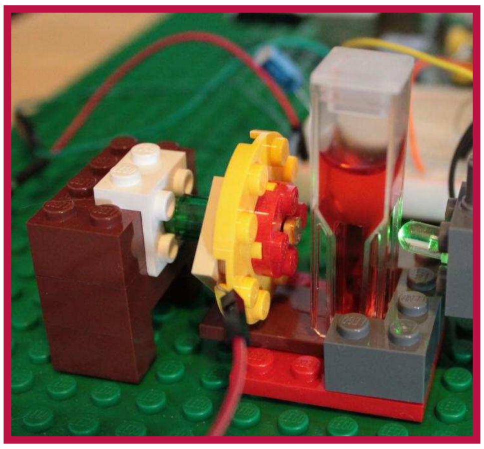
36 **PACKET SNIFFING**
Raspberry Spy Part 1: understanding Wi-Fi networks

42 **C++ CACHE**
Part 6: Communication between objects

46 **THIS MONTH'S EVENTS GUIDE**
Bristol UK, Southend on Sea UK, Merthyr Tydfil UK, Keighley UK, Trier Germany

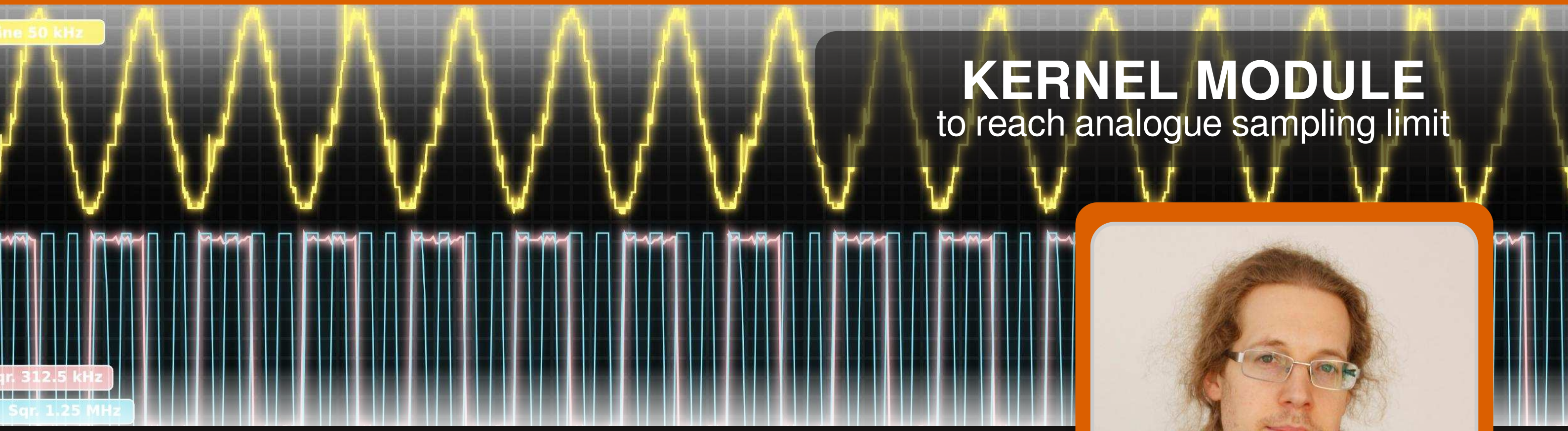
48 **FEEDBACK**
Have your say about The MagPi





Cover and Picademy photos courtesy of Allen Heard

http://www.themagpi.com



KERNEL MODULE
to reach analogue sampling limit

Building an oscilloscope with a Raspberry Pi

SKILL LEVEL : ADVANCED

Daniel Pelikan

Guest Writer

An oscilloscope can be very useful to analyse a circuit or experiment. Slow signals can be sampled with a sound card and xoscope http://xoscope.sourceforge.net/ However, sound cards cannot run at sampling frequencies above 100kHz.

To achieve higher sampling rates, I tried using an Arduino. With the Arduino internal Analog Digital Converter (ADC), I was able to reach 1,000,000 samples every second (1MSPS). Next, I tried using the Arduino with an external ADC, which reached around 5MSPS. However, this was still too slow for my application and I searched for a cheap way to reach a higher sampling speed.

I decided to try the Raspberry Pi, which provides 17 General Purpose Input Output (GPIO) pins that can be used to interface with ADC chips. Although we could use an ADC that connects via SPI or I²C to the Raspberry Pi, these protocols result in sampling rates that are as slow as a sound card readout or slower. One needs to use an ADC that has a parallel data output, which can be connected to a Raspberry Pi.

Parallel ADC and realtime readout

A parallel ADC can be used to take a sample on the rising edge of a clock signal and output the sample on the data pins on the falling edge. The aim is to clock the ADC at our required sample rate and read all of the data pins between each sample.

The Raspberry Pi is a general purpose computer that can run a Linux operation system. However, Linux operating systems do not normally run processes in realtime. This is because the operating system listens for inputs from other devices, rather than just processing one command at a time. When reading an external ADC, one needs to make sure that the time between each sample point is the same. Without a realtime operating system, this is not guaranteed.

After a lot of tests and a lot of reading about interrupts and process control in Linux, I decided to write a Linux kernel module to try to solve the realtime readout problem.

A Linux kernel module

Writing a Linux kernel module provides the possibility to perform low level hardware operations. We need to run with the highest possible priority, reading the GPIO register with the system interrupts disabled for as short a time as possible.

Compiling a kernel module

The text that follows assumes that a Raspberry Pi is being used to compile the Linux kernel. However, another Linux PC can be used to perform cross-compilation to speed up the process: http://elinux.org/RPi_Kernel_Compilation

To set up the build environment correctly, copy the Bash script from the top of the next page into a file, make it executable and then run it.

Before proceeding, it may be useful to read over some Linux kernel development documentation:
http://www.tldp.org/LDP/lkmpg/2.6/html/lkmpg.html

To read and write registers on the Raspberry Pi, we need to know their addresses in memory. This information can be found in the BCM2835-ARM-Peripherals documentation:
http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

```
#!/bin/bash
# A script to setup the Raspberry Pi for a kernel build
FV=`zgrep "* firmware as of" /usr/share/doc/raspberrypi-bootloader/changelog.Debian.gz | head -1 | awk '{ print $5 }'`
mkdir -p k_tmp/linux
wget https://raw.github.com/raspberrypi/firmware/$FV/extra/git_hash -O k_tmp/git_hash
wget https://raw.github.com/raspberrypi/firmware/$FV/extra/Module.symvers -O k_tmp/Module.symvers
HASH=`cat k_tmp/git_hash`
wget -c https://github.com/raspberrypi/linux/tarball/$HASH -O k_tmp/linux.tar.gz
cd k_tmp
tar -xzf linux.tar.gz
KV=`uname -r`

sudo mv raspberrypi-linux* /usr/src/linux-source-$KV
sudo ln -s /usr/src/linux-source-$KV /lib/modules/$KV/build
sudo cp Module.symvers /usr/src/linux-source-$KV/
sudo zcat /proc/config.gz > /usr/src/linux-source-$KV/.config
cd /usr/src/linux-source-$KV/
sudo make oldconfig
sudo make prepare
sudo make scripts
```

Bash script

Writing the kernel module

Create a C file called `Scope-drv.c` that contains:

Scope-drv.c (1/10)

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
#include <linux/time.h>
#include <linux/io.h>
#include <linux/vmalloc.h>

int init_module(void);
void cleanup_module(void);
static int device_open(struct inode *, struct file *);
static int device_release(struct inode *, struct file *);
static ssize_t device_read(struct file *, char *, size_t,
  loff_t *);
static ssize_t device_write(struct file *, const char *,
  size_t, loff_t *);

#define SUCCESS 0
#define DEVICE_NAME "chardev" /* Device name */
#define BUF_LEN 80 /* Maximum length of device message */
```

To set address values and allow simpler register manipulation in the C code, append the preprocessor macros at the bottom of this page to the C source file. More information on these macros can be found at: http://www.pieter-jan.com/node/15

The next piece of code that should be added to the C source file defines the GPIO connections that are used to connect to the ADC. For this article, a six bit ADC was used. Therefore, six GPIO connections are needed per ADC:

Scope-drv.c (3/10)

```
/* Number of samples to capture */
#define SAMPLE_SIZE      10000

/* Define GPIO Pins */
/* ADC 1 */
#define BIT0_PIN 7
#define BIT1_PIN 8
#define BIT2_PIN 9
#define BIT3_PIN 10
#define BIT4_PIN 11
#define BIT5_PIN 25

/* ADC 2 */
#define BIT0_PIN2 17
#define BIT1_PIN2 18
#define BIT2_PIN2 22
#define BIT3_PIN2 23
#define BIT4_PIN2 24
#define BIT5_PIN2 27
```

The numbering scheme used follows the BCM numbering scheme given at: http://elinux.org/RPi_Low-level_peripherals

Scope-drv.c (2/10)

```
/* Settings and macros for the GPIO connections */
#define BCM2708_PERI_BASE 0x20000000
#define GPIO_BASE (BCM2708_PERI_BASE + 0x200000) /* GPIO controller */

#define INP_GPIO(g)  *(gpio.addr + ((g)/10)) &= ~(7<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio.addr + (((g)/10))) |= (((a)<=3?(a) + 4:(a)==4?3:2)<<(((g)%10)*3))


/* GPIO clock */
#define CLOCK_BASE              (BCM2708_PERI_BASE + 0x00101000)
#define GZ_CLK_BUSY (1 << 7)
```

Now add the remaining function and variable definitions given below to the C file.

Scope-drv.c (4/10)

```
struct bcm2835_peripheral {
  unsigned long addr_p;
  int mem_fd;
  void *map;
  volatile unsigned int *addr;
};

static int map_peripheral(struct bcm2835_peripheral *p);
static void unmap_peripheral(struct bcm2835_peripheral *p);
static void readScope(void); /* Read a sample */

static int Major; /* Major number set for device driver */
static int Device_Open = 0; /* Store device status */
static char msg[BUF_LEN];
static char *msg_Ptr;

static unsigned char *buf_p;

static struct file_operations fops = {
  .read = device_read,
  .write = device_write,
  .open = device_open,
  .release = device_release
};

static struct bcm2835_peripheral myclock = {CLOCK_BASE};
static struct bcm2835_peripheral gpio = {GPIO_BASE};
struct DataStruct{
  uint32_t Buffer[SAMPLE_SIZE];
  uint32_t time;
};

struct DataStruct dataStruct;

static unsigned char *ScopeBufferStart;
static unsigned char *ScopeBufferStop;
```

The first part of this code defines a struct to hold the address information. A pointer of this struct type is passed to the `map_peripheral()` and `unmap_peripheral()` functions, which are used to map the hardware registers into memory and release the mapping.

The `myclock` and `gpio` structs are assigned the register addresses of the GPIO and clock pins, such that they might be used later. The `DataStruct` is defined to hold the time and voltage data read from the ADC. The time information is needed in order to calculate the time between each sample. In addition two pointers `ScopeBufferStart` and `ScopeBufferStop` are defined for later use.

Now that all of the function declarations have been made, the implementation of each function must be added to complete the kernel module.

Memory mapping functions

```
static int map_peripheral(struct bcm2835_peripheral *p){
  p->addr=(uint32_t *)ioremap(GPIO_BASE, 41*4);
  return 0;
}
```

```
static void unmap_peripheral(struct bcm2835_peripheral *p){
  iounmap(p->addr);//unmap the address
}
```

Scope-drv.c (5/10)

Add these two functions to the end of the C source file.

The readScope function

The `readScope()` function is responsible for the ADC readout. Add the code below to the end of the C file.

```
static void readScope(){
  int counter=0;
  struct timespec ts_start,ts_stop;

  /* disable IRQ */
  local_irq_disable();
  local_fiq_disable();

  getnstimeofday(&ts_start); /* Get start time in ns */

  /* take samples */
  while(counter<SAMPLE_SIZE){
    dataStruct.Buffer[counter++]= *(gpio.addr + 13);
  }

  getnstimeofday(&ts_stop); /* Get stop time in ns */

  /* enable IRQ */
  local_fiq_enable();
  local_irq_enable();

  /* Store the time difference in ns */
  dataStruct.time =
    timespec_to_ns(&ts_stop) - timespec_to_ns(&ts_start);

  buf_p = (unsigned char*)&dataStruct;
  ScopeBufferStart = (unsigned char*)&dataStruct;

  ScopeBufferStop = ScopeBufferStart+
    sizeof(struct DataStruct);
}
```

Scope-drv.c (6/10)

The first action in this function is to disable all interrupts to provide realtime readout. It is very important that the time while the interrupts are disabled is minimised, since the interrupts are needed for many other Linux processes such as the network connections and other file operations. Reading 10,000 samples takes approximately 1ms, which is small enough not to cause interrupt related problems.

Before reading out the ADC, the current time in nano seconds is stored. Then the full GPIO register is read out 10,000 times and the data are saved in `dataStruct`. After the readout, the current time is requested again and the interrupts are enabled again. The time between each sample point is calculated from the time difference divided by 10,000.

The init_module function

In order to make a kernel module work, the module needs some special entry functions. One of these functions is the `init_module()`, which is called when the kernel module is loaded. Add the C code below to the end of the C source file.

Scope-drv.c (7/10)

```
int init_module(void){
  struct bcm2835_peripheral *p=&myclock;
  int speed_id = 6; /* 1 for 19MHz or 6 for 500 MHz */

  Major = register_chrdev(0, DEVICE_NAME, &fops);
  if(Major < 0){
    printk(KERN_ALERT "Reg. char dev fail %d\n",Major);
    return Major;
  }
  printk(KERN_INFO "Major number %d.\n", Major);
  printk(KERN_INFO "created a dev file with\n");
  printk(KERN_INFO "'mknod /dev/%s c %d 0'.\n",
    DEVICE_NAME,Major);

  /* Map GPIO */
  if(map_peripheral(&gpio) == -1){
    printk(KERN_ALERT "Failed to map the GPIO\n");
    return -1;
  }

  /* Define input ADC connections */
  INP_GPIO(BIT0_PIN);
  INP_GPIO(BIT1_PIN);
  INP_GPIO(BIT2_PIN);
  INP_GPIO(BIT3_PIN);
  INP_GPIO(BIT4_PIN);
  INP_GPIO(BIT5_PIN);

  INP_GPIO(BIT0_PIN2);
  INP_GPIO(BIT1_PIN2);
  INP_GPIO(BIT2_PIN2);
  INP_GPIO(BIT3_PIN2);
  INP_GPIO(BIT4_PIN2);
  INP_GPIO(BIT5_PIN2);

  /* Set a clock signal on Pin 4 */
  p->addr=(uint32_t *)ioremap(CLOCK_BASE, 41*4);

  INP_GPIO(4);
  SET_GPIO_ALT(4,0);
  *(myclock.addr+28)=0x5A000000 | speed_id;

  /* Wait until clock is no longer busy (BUSY flag) */
  while(*(myclock.addr+28) & GZ_CLK_BUSY) {};

  /* Set divider to divide by 50, to reach 10MHz. */
  *(myclock.addr+29)= 0x5A000000 | (0x32 << 12) | 0;

  /* Turn the clock on */
  *(myclock.addr+28)=0x5A000010 | speed_id;

  return SUCCESS;
}
```

This function registers the new device (`/dev/chardev`), maps the GPIO address and configures the input connections. It then sets the clock to run at 500MHz and uses a divider to provide a 10MHz clock signal on GPIO pin 4.

The device file `/dev/chardev` provides a mechanism for an external program to communicate with the kernel module. The 10MHz clock signal on GPIO Pin 4 is used to drive the ADC clock for sampling. More details on setting the clock can be found in chapter 6.3 General Purpose GPIO Clocks in http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

The GPIO bit samples may not be synchronised with the clock. This can cause bits to be read from the same sample twice or be missed. This can be improved by setting the clock as close as possible to the frequency of the GPIO readout.

Clean up and device functions

Add the C code below to the end of the `Scope-drv.c` file.

Scope-drv.c (8/10)

```
void cleanup_module(void){
  unregister_chrdev(Major, DEVICE_NAME);
  unmap_peripheral(&gpio);
  unmap_peripheral(&myclock);
}
```

This function is called when the kernel module is unloaded. It removes the device file and unmaps the GPIO and clock.

The implementation of four more functions need to be added to the C file, to handle connections to the device file associated with the kernel module:

```
static int device_open(struct inode *inode,
  struct file *file){
  static int counter = 0;
  if(Device_Open) return -EBUSY;
  Device_Open++;
  sprintf(msg,"Called device_open %d times\n",counter++);
  msg_Ptr = msg;
  readScope(); /* Read ADC samples into memory. */
  try_module_get(THIS_MODULE);
  return SUCCESS;
}

static int device_release(struct inode *inode,
  struct file *file){
  Device_Open--; /* We're now ready for our next caller */
  module_put(THIS_MODULE);
  return 0;
}

static ssize_t device_read(struct file *filp,char *buffer,
  size_t length,loff_t * offset){
  int bytes_read = 0; /* To count bytes read. */
  if(*msg_Ptr == 0) return 0;

  /* Protect against going outside the buffer. */
  while(length && buf_p<ScopeBufferStop){
    if(0!=put_user(*(buf_p++), buffer++))
      printk(KERN_INFO "Problem with copy\n");
    length--;
    bytes_read++;
  }
  return bytes_read;
}
```

Scope-drv.c (9/10)

```
static ssize_t device_write(struct file *filp,
  const char *buff, size_t len, loff_t * off) {
  printk(KERN_ALERT "This operation isn't supported.\n");
  return -EINVAL;
}
```
Scope-drv.c (10/10)

The `device_open()` function is called when the device file associated with the kernel module is opened. Opening the device file causes the ADC to be read out 10,000 times, where the results are saved in memory. The `device_release()` function is called when the device file is closed. The `device_read()` function is called when a process reads from the device file. This function returns the measurements that were made when the device file was opened. The last function `device_write()` is needed to handle the case when a process tries to write to the device file.

Building and loading the module

Create a Makefile in the same directory as the `Scope-drv.c` file. Then add

```
obj-m += Scope-drv.o

all:
  make -C /lib/modules/$(shell uname -r)/build \
M=$(PWD) modules

clean:
  make -C /lib/modules/$(shell uname -r)/build \
M=$(PWD) clean
```

where the indents should be a single tab. (More information on Makefiles is given in Issue 7 of The MagPi.)

The kernel module can now be compiled on a Raspberry Pi by typing

```
make
```

Once the module has been successfully compiled, load the module by typing:

```
sudo insmod ./Scope-drv.ko
```

Then assign the device file, by typing:

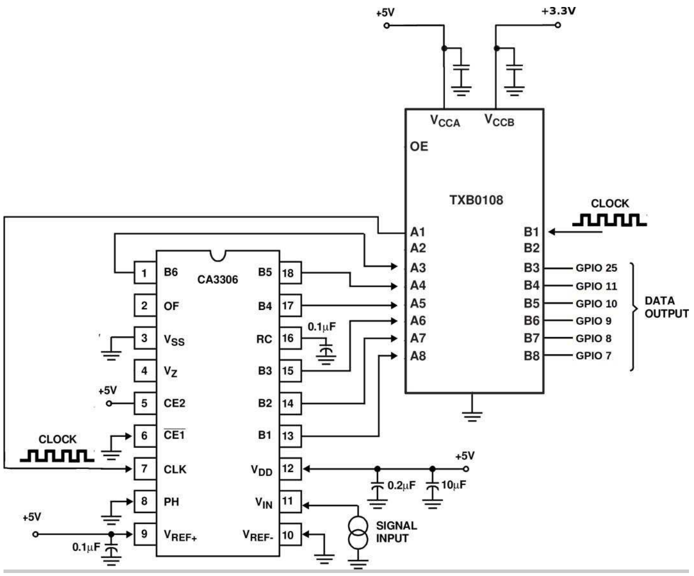
```
sudo mknod /dev/chardev c 248 0
```

Connecting the ADC

Now that the kernel module has been described, an ADC is needed to provide the input data. For this article, a CA3306 ADC from Intersil was used. This is a 6-bit 15 MSPS ADC with a parallel read out. This ADC is very cheap and fast. Many other ADC chips with parallel readout could be used, although it is necessary to check the datasheet for connection details and clock speed settings, etc..

For the selected ADC, 6-bit implies that between the ground level (0V) and the reference voltage (5V) there are 64 divisions to represent the signal. This is quite course, but is enough for simple applications.

The selected ADC operates with 5V logic, but the Raspberry Pi uses 3V3 logic. Therefore, a level converter is needed to protect the Raspberry Pi from being damaged. The simplest way to achieve this is to use a dedicated level converter, such as the TXB0108 from Texas Instruments. To ensure that stable readings are obtained from the ADC, it is recommended that a separate 5V supply is used as your VREF+ and VDD supply. This prevents voltage drops that can occur if the power supply is shared with the Raspberry Pi. However, a common ground (GND) connection should be used for the external



supply, ADC and Raspberry Pi.

Data acquisition

Once the ADC has been connected and the kernel module has been loaded, data can be read from the ADC by connecting to the device file associated with the kernel module. To connect to the kernel module, another program is needed. This program could be written in several different programming languages. For this article, C++ was chosen. Create a new file called `readout.cpp` and add the C++ given below and on the next page.

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <bitset>

typedef unsigned int uint32_t;

/* To match kernel module data structure */
const int DataPointsRPi=10000;
struct DataStructRPi{
  uint32_t Buffer[DataPointsRPi];
  uint32_t time;
};
```

```
int main(){
  //Read the RPi
  struct DataStructRPi dataStruct;
  unsigned char *ScopeBufferStart;
  unsigned char *ScopeBufferStop;
  unsigned char *buf_p;

  buf_p=(unsigned char*)&dataStruct;
  ScopeBufferStart=(unsigned char*)&dataStruct;
  ScopeBufferStop=ScopeBufferStart+
    sizeof(struct DataStructRPi);

  std::string line;
  std::ifstream myfile ("/dev/chardev");
  if(myfile.is_open()){
    while(std::getline(myfile,line)){
      for(int i=0;i<line.size();i++){
        if(buf_p>ScopeBufferStop)
          std::cerr<<"buf_p out of range!"<<std::endl;
        *(buf_p)=line[i];
        buf_p++;
      }
    }
    myfile.close();
  }
  else std::cerr<<"Unable to open file"<<std::endl;

  // Now convert data for text output

  // Time in nano seconds
  double time=dataStruct.time/(double)DataPointsRPi;

  for(int i=0;i<DataPointsRPi;i++){
    int valueADC1=0;//ADC 1
    // Move the bits to the right position
    int tmp = dataStruct.Buffer[i] & (0b11111<<(7));
    valueADC1=tmp>>(7);
    tmp = dataStruct.Buffer[i] & (0b1<<(25));
    valueADC1+=(tmp>>(20));

    int valueADC2=0;//ADC2
    tmp = dataStruct.Buffer[i] & (0b11<<(17));
    valueADC2=tmp>>17;
    tmp=dataStruct.Buffer[i] & (0b111<<(22));
    valueADC2+=(tmp>>20);
    tmp=dataStruct.Buffer[i] & (0b1<<27);
    valueADC2+=(tmp>>22);

    // Print the values of the time and both ADCs
    std::cout<<i*time<<"\t"<<valueADC1*(5./63.)
             <<"\t"<<valueADC2*(5./63.)<<std::endl;
  }
  return 0;
}
```

This program includes the definition of the data struct that matches the version in the kernel module.

The `main()` function connects to the `/dev/chardev` device, which causes the kernel module to readout the ADC and store the values. Then the data are read from the memory buffer and copied into the local buffer within the `main()` function. Finally, the data are converted into a time in nano seconds and voltage values. The time and two voltage values are then printed in columns.

The voltage values read by the ADCs are encoded as six bits. The bits are decoded using bit shift operations and bitwise and operations.

To compile the data acquisition program, type:

```
g++ -o readout readout.cpp
```

Then run the program by typing:

```
./readout > data.txt
```

The data file can be displayed using gnuplot. Install gnuplot by typing:

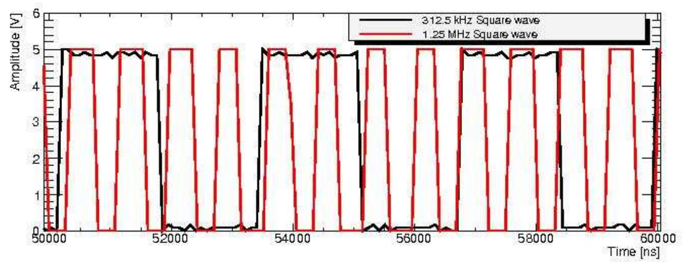
```
sudo apt-get install -y gnuplot-x11
```

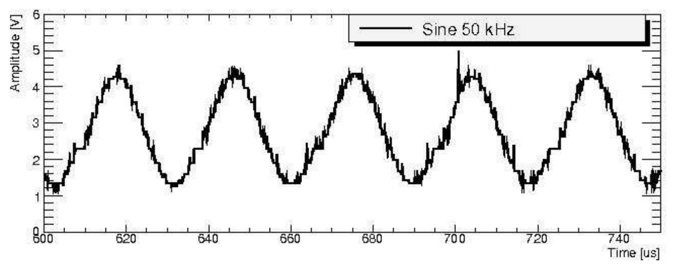
Then type gnuplot and enter the macro given below:

```
set key inside right top
set title "ADC readout"
set xlabel "Time   [ns]"
set ylabel "Voltage   [V]"
plot "data.txt" using 1:2 title 'ADC1' with lines, \
     "data.txt" using 1:3 title 'ADC2' with lines
```

More information on gnuplot can be found at:
http://www.gnuplot.info/

gnuplot could also be run directly from the readout program as discussed in The C Cave article in Issue 6 of The MagPi. Alternatively, gnuplot can be used within a Bash script as described in the Bash Gaffer Tape article in Issue 12 of The MagPi.





Expand your Pi

Stackable Raspberry Pi expansion boards and accessories

ADC-DAC Pi

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

IO Pi

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

RTC Pi

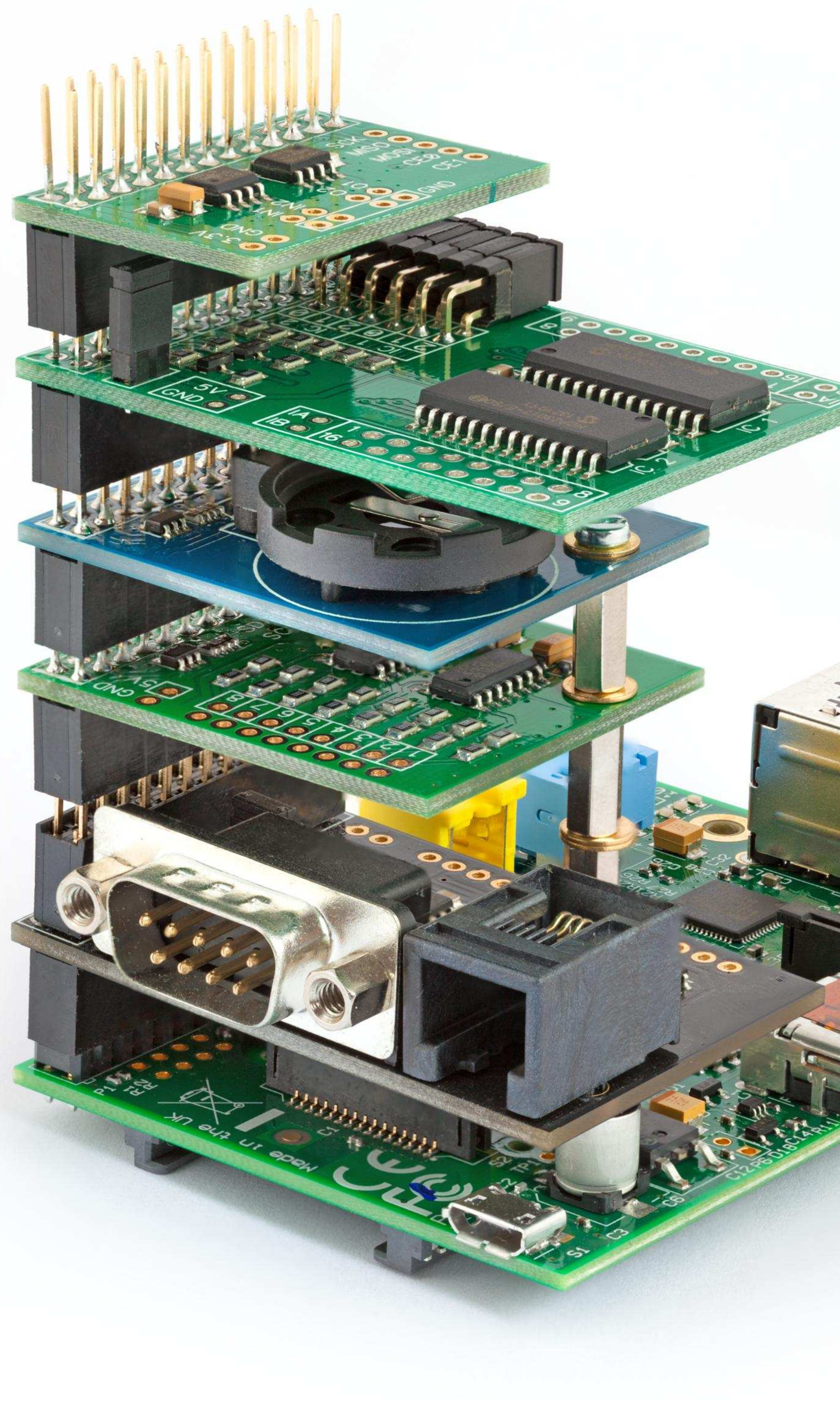
Real-time clock with battery backup and 5V I^2C level converter for adding external 5V I^2C devices to your Raspberry Pi.

ADC Pi

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

Com Pi

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



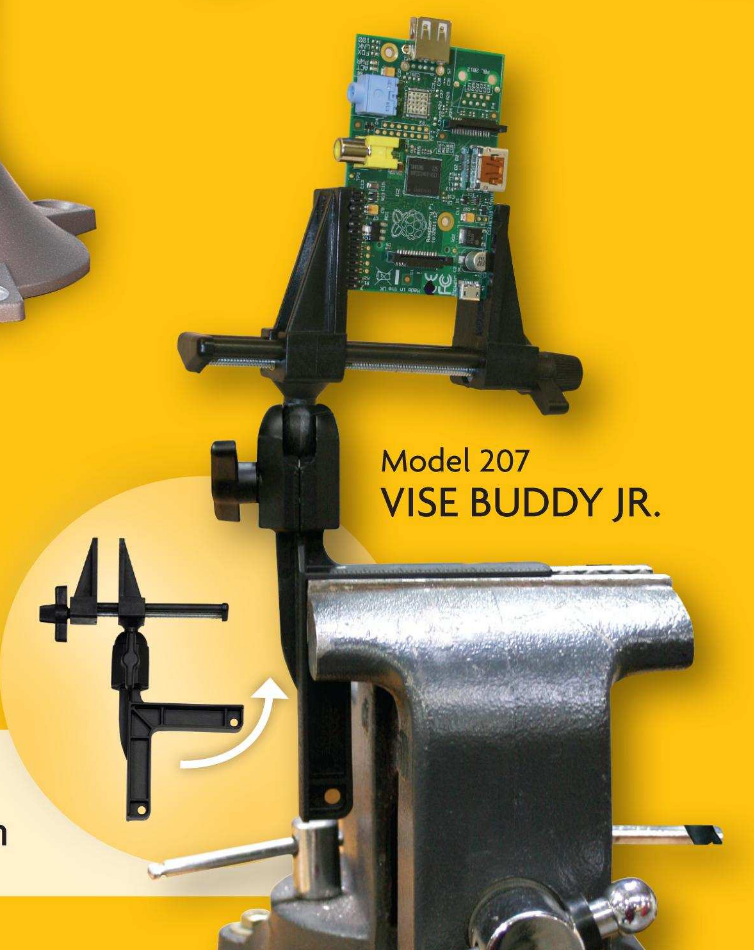
AB electronics UK   www.abelectronics.co.uk

VISIT US ON

WORLD'S MOST VERSATILE CIRCUIT BOARD HOLDERS

Model 209
VACUUM
BASE PV JR.

Model 201
PV JR.



Model 207
VISE BUDDY JR.

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

PANAVISE®
Innovative Holding Solutions

www.panavise.com

PIMO RONI

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

STUDYING ATMOSPHERIC POLLUTION

A Multi-Sensor Array for Atmospheric Science

Michael Petersen

Guest Writer

Part 2: Implementing the Code

SKILL LEVEL : ADVANCED

Introduction

This two part series describes the design and construction of a Multi-Sensor Array (MSA) for studying atmospheric pollution in an urbanised mountain basin; specifically, the region above Salt Lake City, Utah (USA). The MSA is flown on research balloons by HARBOR, an undergraduate research group at Weber State University in Ogden, Utah. The MSA produces a column of measurements from ground level (approx. 1km ASL) to the lower stratosphere (approx. 35 km ASL).

During flight, the system is exposed to pressures as low as 0.75 mmHg, where heat exchange becomes difficult even at temperatures of -50°C. Jet stream winds can exceed speeds of 200km/h, applying punishing shock and vibration forces to our electronic equipment. In this extreme environment, the MSA must continue to gather scientific data for a minimum of 4 hours.

The first part of this series in Issue 23 of the MagPi focused on the hardware design of the MSA system. This second part describes how the software was designed and implemented.

Library installation

The MSA software is written completely in C. We chose C primary because we were already familiar with it, and because it is a fairly common language for embedded systems. We made use of a fantastic C library for the BCM2835 ARM chip used on the Raspberry Pi. The Library is maintained by Mike McCauley at: http://www.airspayce.com/mikem/bcm2835

The BCM2835 C library provides functions for communicating with I^{2}C devices, accessing GPIO pins, and generating Pulse Width Modulation (PWM) signals to control motors and heaters.

The most recent version of the library is 1.36. To install the library onto the Raspberry Pi, connect to the Internet and enter the following:

```
wget http://www.airspayce.com/mikem/bcm2835/bc
m2835-1.36.tar.gz
tar zxvf bcm2835-1.36.tar.gz
cd bcm2835-1.32
```

Before compiling the library, it is necessary to modify the source code if it is going to be installed on a version 1 Model B Raspberry Pi. For other Raspberry Pi models, no changes are needed.

For a version 1 Model B Raspberry Pi, you must edit the source code by entering:

```
cd src
nano bcm2835.c
```

Now, go to line 30 and uncomment the following portion:

```
#define I2C_V1
```

This compiles the code to function with the correct I²C bus. Then, save and close the `bcm2835.c` file.

Now compile the source code,

```
./configure
make
sudo make check
sudo make install
```

To include this library when compiling in C, use:

```
gcc -o name name.c -l bcm2835
```

MSA software overview

The MSA operational flight program is made up of three basic threads, 'Main', 'Sample Rate Timer' and 'Data logging scheduler'

Main thread

Upon start-up, the main thread automatically begins and sets up POSIX threads for the timer and scheduler, initiates the I²C bus, initialises the GPIO pins, creates data files, and monitors the mission pull pin. Once the mission pin is pulled, a mission timer begins and the current time is stamped on each line in the data files. The USB, Ethernet, and HDMI chips are disabled to conserve power and the main loop begins. In the main loop, the MSA simply waits for an event to initiate a safe shutdown; such as when the shutdown button is pressed, the mission countdown timer has elapsed, or the battery voltage drops below a preset threshold for more than one minute.

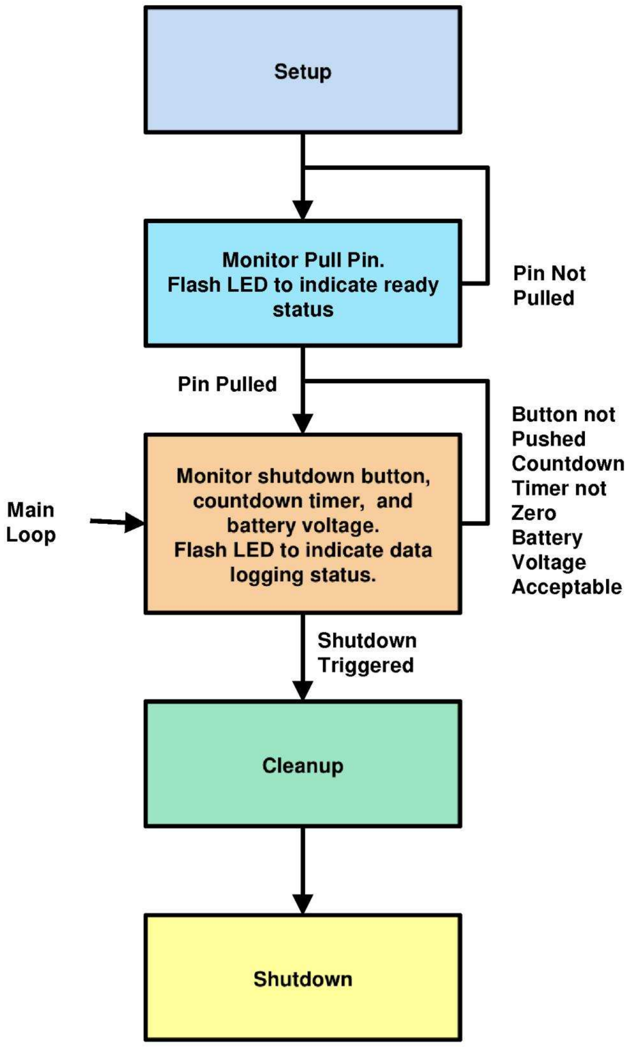

Setup
Monitor Pull Pin. Flash LED to indicate ready status
Pin Not Pulled
Pin Pulled
Main Loop
Monitor shutdown button, countdown timer, and battery voltage. Flash LED to indicate data logging status.
Button not Pushed Countdown Timer not Zero Battery Voltage Acceptable
Shutdown Triggered
Cleanup
Shutdown

Figure 1: The Main thread

Shutdown is initiated in C by the following command:

```
System("shutdown -h -P now");
```

Sample rate timing thread

The MSA code is written without any interrupts. It was a challenge to create a timer that could ensure that measurements would be taken at regular intervals. Fortunately, the C compiler that comes standard with Linux distributions is equipped with POSIX thread capabilities. We made use of parallel programming, to create the timing and scheduling threads.

The sample rate timing thread is designed to post a semaphore at a given time interval, in our case 100 ms, to set the base data sample rate at 10Hz. It makes use of the `nanosleep()` function, to create a 100 millisecond timer for reading sensors.

```
void *schedule_timer(void *arg) {
  int milisec = 100;
  struct timespec req= {0};
  req.tv_sec = 0;
  req.tv_nsec = milisec * 1000000l;
  while(1) {
    nanosleep(&req, null);
    sem_post(&sem);
  } // end while loop
} // end schedule_timer
```

After 100 milliseconds, the timer releases the semaphore to the data logging schedule thread.

Data logging schedule thread

The data logging schedule thread takes the semaphore, whenever it is available, and begins a read cycle. The MSA has two main reading cycles:

1. Flight dynamics data:
  • accelerometers
  • gyroscopes
  • magnetometer

2. Environmental data:
  • temperature
  • humidity
  • pressure
  • other data

The scheduling thread also maintains a five minute countdown timer. When the timer elapses, the MSA code saves all of the data files and the timer is reset.

Flight dynamic sensors are read every cycle, at a rate of 10Hz. Environmental data is read at a much slower rate, due to the relatively long conversion time of the LTC2495 ADC (approximately 160ms). Every 200ms, the scheduling thread determines whether to set, clear, or read an ADC channel. Nested switch statements determine which channel to read (0 to 15) and which function to call (`set`, `clear`, or `read`).

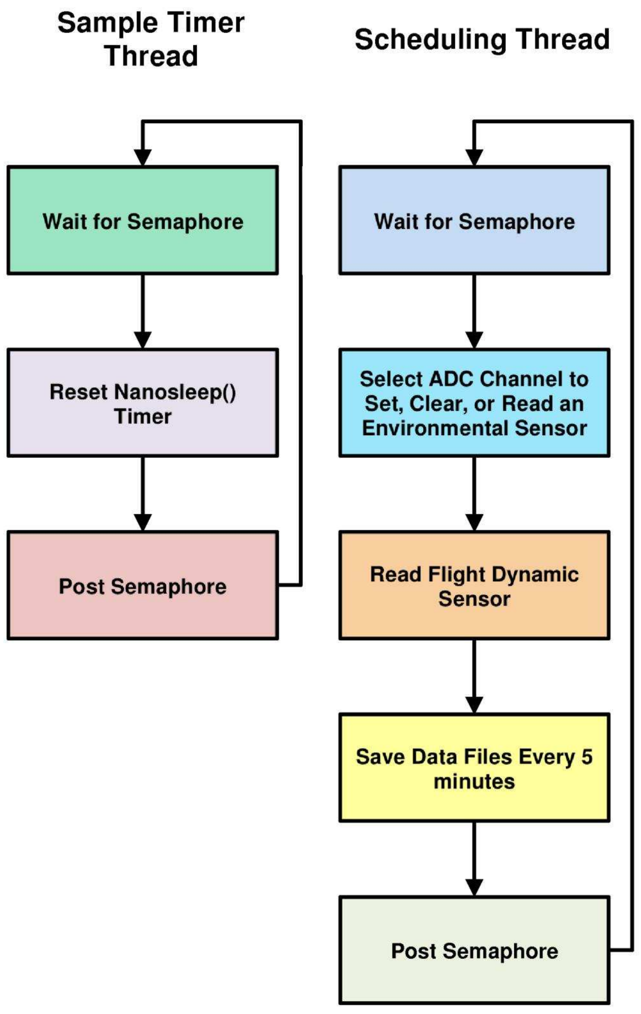


Figure 2: Timer and scheduling threads

It takes approximately 10 seconds to read all of the ADC channels:

200 msec x 3 cycles x 16 channels = 9.6 sec

which seems like a lot of time. This is fine, however, since most environmental parameters change relatively slowly as a function of altitude. A typical ascent rate is only 4.1 meters per second, which means that the balloon travels about 40 meters per read cycle. One read every 40 meters provides adequate resolution for temperature, pressure, humidity and even gas composition when viewed over 30,000 meters. If greater resolution is necessary an additional ADC can always be added.

Reading I^{2}C sensors

The MSA sensors are divided into two different groups: (1) analog and (2) digital sensors.

The analog sensors are all wired through an I^{2}C ready LTC2495 ADC. The digital sensors for the MSA were all chosen to be I^{2}C accessible. With all of the sensors communicating over the I^{2}C bus, we were able to free up the remaining GPIOs for other purposes.

Reading analog sensors

Before reading any I^{2}C sensors, the bus must be initialised and set to a desired speed. In our case, we designed the bus to run at high-speed (400KHz). The following code below shows how we set up the I^{2}C bus. Note that function calls prefaced with bcm2835 are part of the bcm2835 C library previously mentioned.

```
bcm2835_i2c_begin();
bcm2835_i2c_setclockdivider(bcm2835_i2c_clock
_divider_626);
```

To read data from the ADC it is first necessary to select an input channel, set the gain, and choose whether it is single ended or differential input. We used only single ended sensors for the MSA, so setup was the same for each channel. The following function sets the channel to read the pressure sensor on the ADC which is located at the I^{2}C address 0x14.

```
unsigned char chan_0[] = {0xb0, 0x80};
// 0xb080 selects adc channel 0,
// single ended input, +in, gain x 1

unsigned char ltc2495 = 0x14;
bcm2835_i2c_setslaveaddress(ltc2495);
bcm2835_i2c_write(chan_0, sizeof(chan_0));
```

The first read from the ADC is generally unusable and has to be cleared; this is because the initial data has leftover information from a previous sample or a different sensor. The following function clears the channel by storing it a junk register and ignoring it:

```
unsigned char junk[3];
bcm2835_i2c_setslaveaddress(ltc2495);
bcm2835_i2c_read(junk, sizeof(junk));
```

We can now read the ADC. The following example code reads the pressure sensor on channel 0:

```
bcm2835_i2c_setslaveaddress(ltc2495);
unsigned char p_buf[3];
bcm2835_i2c_read(p_buf, sizeof(p_buf));

// mask the first bit (unused)
p_buf[0] = p_buf & 0x7f;

pressure = (p_buf[0] * 0x10000) + (p_buf[1] *
0x100) + (p_buf[2]);
pressure = pressure/0x40;

// multiply by the step size to get voltage
v_pressure = pressure * 0.000025177;
```

The actual value is then processed from the signal voltage according the datasheet for a particular sensor.

Reading digital sensors

The digital sensors are a little more straight forward, because they don’t have to be reset every read cycle. However, some of them require burst reads (I^{2}C reads with repeated starts and no stop) to send the address of a specific register to be read and an address to the destination where the data will be saved. Burst reads are also used to ensure that multiple axis data comes from the same sample.

Sensors like the HMC5883 3-axis magnetometer must be set up initially, but keep their settings for successive reads. The following code sets the sample rate of an HMC5883 to 8 averaged samples at rate of 15Hz. It is also set to measure continuously with a gain of 5:

```
unsigned char hmc5883 = 0x1e;
bcm2835_i2c_setslaveaddress(hmc5883);

Unsigned char gain_5[] = {0x01,0xa0};
Unsigned char cont_mode[] = {0x02, 0x00};
Unsigned char samp_rate_8_15[] = {0x00, 0x70};
Bcm2835_i2c_write(samp_rate_8_15,
  sizeof(samp_rate_8_15));
Bcm2835_i2c_write(gain_5, sizeof(gain_5));
Bcm2835_i2c_write(cont_mode,
  sizeof(cont_mode));
```

A read is completed as follows:

```
unsigned char mag_reg[] = {0x03};
unsigned char mag_buf[6];
bcm2835_i2c_read_register_rs(mag_reg, mag_buf,
  sizeof(mag_buf));
```

The data buffer now holds 6 bytes of data, 2 bytes for each axis, which are stored as two's compliment values. As before, the actual value is then processed according to the datasheet for the sensor being used. A complete version of the code discussed can be found at: https://github.com/mikeanthoney/RPi_atmosphere

Data Outputs

The MSA program stores all of the data in .CSV files that are easy to use in LibreOffice or Excel. A typical output looks something like the following:

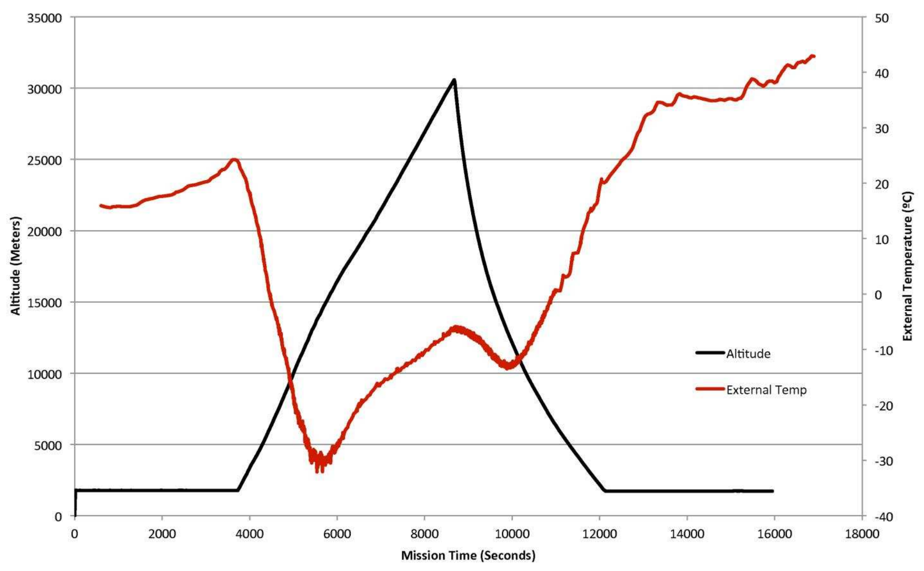
```
Date: 11/09/13
time,reg_temp,xtemp,itemp,ipressure,ihumidity
,dust_v,xhumidity,batt_voltage,
12:59:18,37.12,23.81,33.02,768.89,22.78,
0.0000,41.57,7.53,
12:59:23,37.00,23.81,33.02,768.85,22.55,
0.0000,38.78,7.53,
12:59:28,36.88,23.88,33.29,768.83,22.32,
0.0000,40.37,7.53,
. . . . . .
```

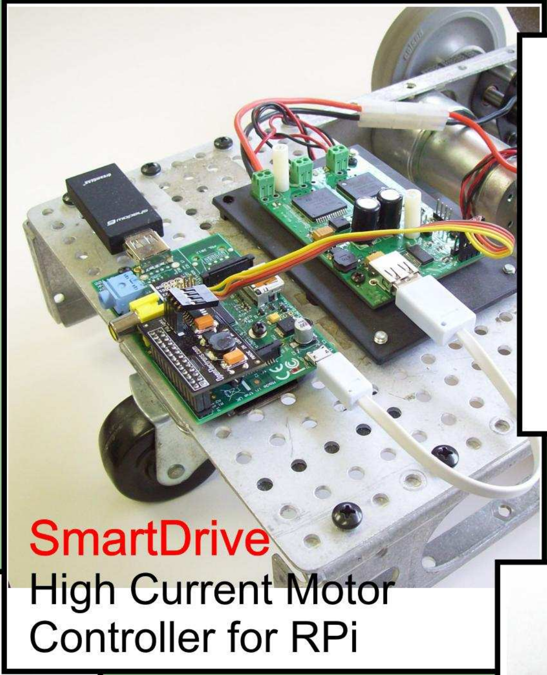
The chart below was created using MSA data from a test flight last July above Duchesne, Utah. The mission lasted 4.5 hours and reached an altitude of just over 30km.

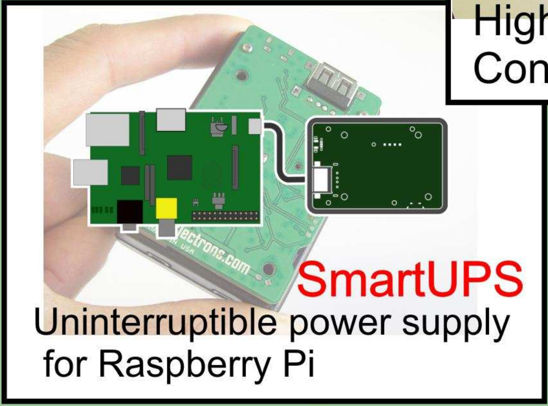
Notice that the relatively low sample rate of the temperature sensor (0.2Hz) still provided adequate resolution over the course of the mission.

Coming up

We are currently working on a balloon borne optical sensor for measuring PM2.5 aerosols. The new sensor will require a vacuum pump to maintain a steady flow rate of 3LPM. A feedback loop will be designed to control a heater for maintaining a relative humidity of less than 50%RH. The Raspberry Pi will generate a PWM signal to regulate both temperature and flow rate. We have already begun prototyping critical components.





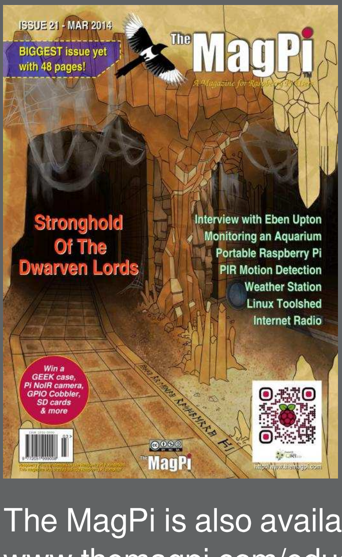


OpenElectrons.com

The MagPi print edition

Experience hundreds of pages of Raspberry Pi hardware projects, software tutorials, reviews of the latest expansion boards and interviews with the makers and creators involved.

Originally funded through Kickstarter, The MagPi printed bundles of Volumes 1 and 2 bring all the information of this peer-reviewed magazine within arm's reach.



Volume 1: Issues 1-8
Volume 2: Issues 9-19

Available worldwide fom these resellers:

swag.raspberrypi.org
www.modmypi.com
www.pi-supply.com
thepihut.com
www.adafruit.com (USA)
www.buyraspberrypi.com.au (Australia)

The MagPi is also available on special offer to schools for a limited time only:
www.themagpi.com/education

PROJECT CURACAO

Remote sensor monitoring in the Caribbean

John Shovic

Guest Writer

Part 5: Deployment and Results

SKILL LEVEL : INTERMEDIATE

What is Project Curacao?

This is the fifth part of a series discussing the design and building of Project Curacao, a sensor filled project that hangs on a radio tower on the island nation of Curacao. Curacao is a desert island 12 degrees north of the equator in the Caribbean. Part 6 in the fall will show the upgraded sensor suite and replacement of the wind turbine.

Project Curacao is designed to monitor the local environment unattended for six months. It operates on solar power cells and communicates with the designer via an iPad App called RasPiConnect. All aspects of this project are designed to be monitored and updated remotely (with the current exception of the Arduino Battery Watchdog).

System description

Project Curacao consists of four subsystems. A Raspberry Pi Model A is the brains and the overall controller. The Power Subsystem was described in part 1, the Environmental Sensor Subsystem in part 2 and the Camera Subsystem was shown in part 3. Part 4 described the software running the Raspberry Pi and the Arduino Battery Watchdog.

Results of shipping

The box arrived in our luggage on March 3, 2014. Airport security (TSA) inspected the box twice, but aside from picking up the box with the outside temperature sensor (naturally the most expensive sensor in the box), pulling the wires off, it survived.

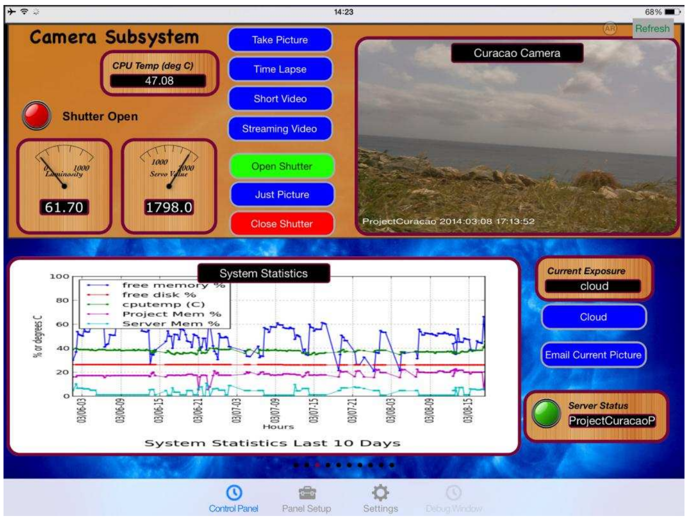
One of the key things to remember about shipping a piece of equipment through airport security is to take out the batteries and put them in your carry on luggage. The other item is to make sure you provide a written explanation of what it is and what it does with the box in the luggage. We wrote a short letter about Project Curacao and included reprints of the MagPi series of articles.

The temperature sensor was fixable. There were also a couple of loose grounds to be tightened. A group of sensors were giving odd readings so I knew to look for a loose ground. One bad sensor might be the sensor, where-as a bad group of sensors was likely a ground or power.

Remember, you can always trust your mother, but you can never trust your ground.

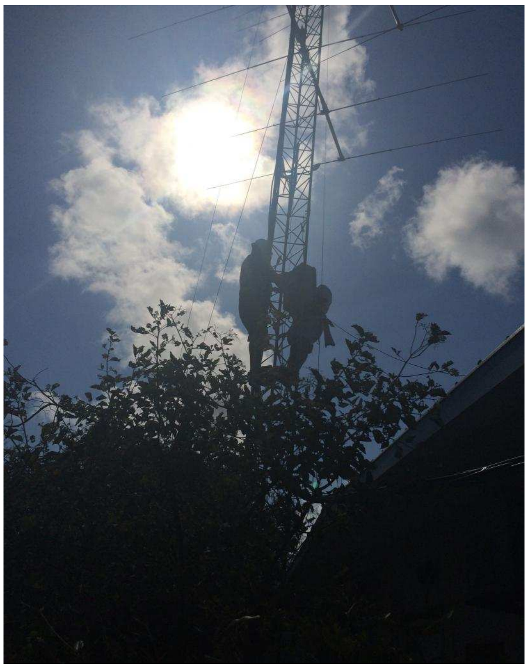
We put the box outside for the first time and the solar cells worked perfectly. The wind turbine needs to be up on the tower to really roar. The wind turbine only generated about 40ma on the ground but should do better up on the tower.

RasPiConnect connected to the box and generated the first Caribbean picture:

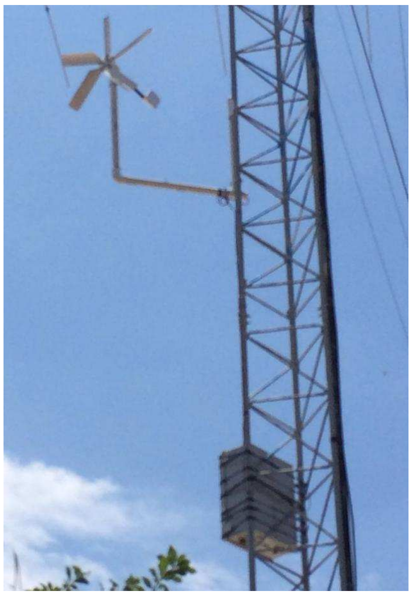


Deployment

We spent ten days finishing off some software and preparing to put the box up the tower. Understand, however, this was in addition to going to the beach, dancing, coffee with the neighbors and some very nice parties. Eventually, we did put up the box. The box and wind turbine was deployed on the tower on March 13, 2014.



Geoff Howard, all round good guy and expert tower climber, was invaluable. This is dangerous work and safety belts and harnesses were used.



The resulting installation of the box and the wind turbine is above. The angle of the box is not what was designed however. The tower mounting platform that we carried down to Curacao did not fit. It was designed for a slightly smaller tower, so we had to strap the box flush to the tower and perpendicular to the ground. This had a significant effect on the amount of solar power generated. Since the latitude is about 12 degrees, the solar cells should be pointing due south around 78 degrees in the current configuration. While the box is designed to adjust for the amount of power generated, the output was far less than designed.

The problem with this was not with the Raspberry Pi, it was with the Arduino Battery Watchdog. We adjusted the run time of the Pi with the Arduino to fit the available power. However, the Arduino is meant to run 100% of the time and the behavior of the Arduino is ill-defined in very low battery conditions. It would glitch the real time clock which gave some very random behavior.

Note on the RasPiConnect (www.milocreek.com) screen below you can see the Arduino battery voltage going down and down and down until we turned it off for a few days and moved it to the roof.

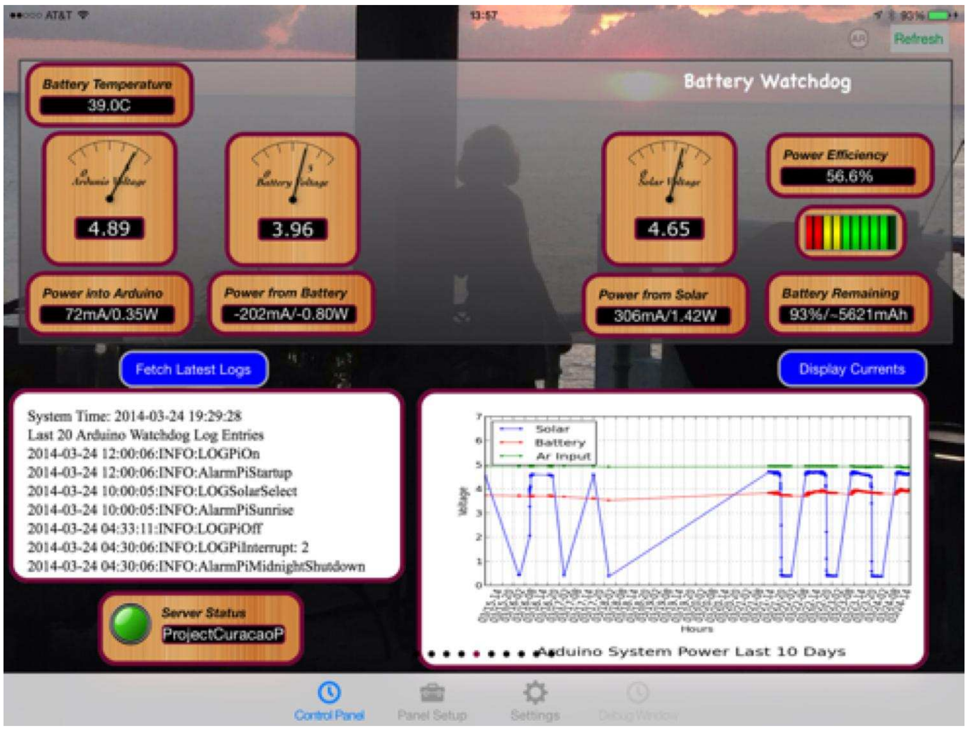
We were hoping the wind turbine would make up the difference but were disappointed on two counts discussed below.

We mounted a mirror at right angles below the solar cell hoping to improve the amount of power generated, but it wasn't significantly increased.

After the wind turbine event described below, we moved the box down to the roof and increased the angle to increase the solar power.

Wind turbine results

Based on our measured 50W wind turbine measured curves (given on http://switchdoc.com), we did not expect too much from this turbine at wind speeds of 15 MPH.



```
System Time: 2014-03-24 19:29:28
Last 20 Arduino Watchdog Log Entries
2014-03-24 12:00:06:INFO:LOGPiOn
2014-03-24 12:00:06:INFO:AlarmPiStartup
2014-03-24 10:00:05:INFO:LOGSolarSelect
2014-03-24 10:00:05:INFO:AlarmPiSunrise
2014-03-24 04:33:11:INFO:LOGPiOff
2014-03-24 04:30:06:INFO:LOGPiInterrupt: 2
2014-03-24 04:30:06:INFO:AlarmPiMidnightShutdown
```

It would only contribute a trickle charge during the night to the project. Based on the limited data we received from the box, it would only add an hour of runtime every day.

Following is an analysis of the wind data:

Total Number of Data Points: 49
Number of Unloaded 50W Wind Turbine Data Points: 37
Average Windspeed: 7.1 MPH
Low: 0.0 MPH
High: 16 MPH

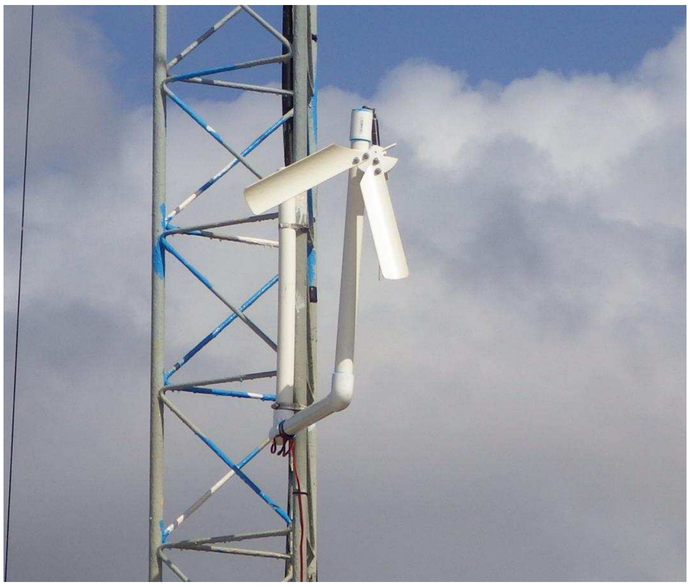
Number of Current Samples (Loaded 50W wind turbine): 12
Average Current Delivered to Battery: 22ma
Low Current Delivered to Battery: 0ma
High Current Delivered to Battery: 138.6ma

Max Power from Turbine to Battery: 0.5W

This 50W wind turbine would need about 25 or 30 MPH winds to run this project.

First full week of operations - death of a wind turbine

We lost the wind turbine during the first week of operation to a wind storm, but the turbine was a last minute addition and so not very important to the overall success of the project. This occurred less han a week after we departed: we got an email saying that there had been a wind storm overnight with gusts about 35 MPH and that the wind turbine was destroyed.



The wind turbine worked as predicted by our models. We switched on the turbine at night to provide a trickle charge to the main computer. We got about 60 - 90ma of current at 15MPH. Nowhere close to the 200-300ma it takes to run the Pi. Note that the turbine popped out of the stand and thoroughly destroyed itself. We think that is was a sympathetic vibration with a particular wind speed (much like the "Galloping Gertie" Tacoma Narrows bridge [Ed: https://en.wikipedia.org/wiki/Tacoma_Narrows_Bridge_(1940)]) since we observed that the stand was flexing in the wind. We will either prevent the next turbine from popping out or stiffen the stand to prevent the flexing or possibly both.

The first 8 Weeks of Operation

We recently hit eight weeks of Project Curacao running in the warm and windy climate. The project has been fairly robust to this point. We lost one sensor the other week. The light color sensor (BM017 / tsc3574) went dead and no longer responds to I2C requests although other sensors on the same bus do continue to respond. We have had no problems with the Arduino Battery Watchdog yet. Since we fixed the solar power problems, the Arduino is rock solid.

Here's an example email that came in from the Project Curacao box telling me the environmental monitoring system has turned the fan on because of high temperatures inside the box (degrees C):

```
From: ********@gmail.com
Subject: ProjectCuracao Fan ON(TMP)
Date: April17,2014at12:08PM

Fan turning ON: State: ot:36.50 it:38.10
oh:34.50 ih:49.40 sv:4.77
```

The box also emails bootup and shutdown messages as well as a picture from the Caribbean once a day at 15:20 Pacific time.

September Upgrades

We are planning a maintenance trip to Curacao. The planned upgrades are:

1) Replacement of wind turbine (possibly with a different type of turbine).

2) Stiffening the turbine mount to avoid another "Galloping Gertie".

3) Addition of a vibration sensor on the turbine mount to monitor the flexing.

4) Adding a loose strap over the top of the turbine to allow turning of the turbine but not "popping out" of the mount.

5) Adding wind speed, wind direction and rainfall sensors.

6) Modification of the Arduino Battery Watchdog to record wind turbine current and voltage even when the Raspberry Pi is off.

7) Modification of the angle of the solar panels - maybe with a sun tracking system

You can see the live hourly data and picture from Project Curacao at: http://milocreek.com/projectcuracaographs.

What's Next?

Our final article, Part 6 in Autumn 2014, will describe the upgrades and the required changes to the Project Curacao system. This has been a large project and we want to thank all of the people and companies that helped make this possible.

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

DIY chemistry lab: Building a spectrophotometer

SKILL LEVEL : INTERMEDIATE

Robert J. LeSuer

Guest Writer

This article describes how to build a simple photometer that can be used, for example, to quantify the amount of food coloring in beverages. It uses the Wolfram Language to collect, visualise and analyse the results.

Introduction

I was in the beverage isle of the supermarket one day and was looking at the soft drinks. One of the drinks, a "Watermelon Punch" had listed on it Red 40 as one of the ingredients. Red 40, also known as Allura Red AC or E129, was phased out of the UK by the Food Standards Agency in 2009 because some research showed that it may induce hyperactivity in children. The food dye is still allowed in the USA market, but if questions have been raised about the safety of the dye, I would like to know how much of it I'm consuming.

I am an analytical chemistry professor and one of the fun parts about my job is that I get to teach students how to play with instruments: we use them, we break them, we repair them and sometimes we even build them. One major area of analytical chemistry is called spectroscopy, which uses light (or more broadly, the electromagnetic spectrum) to explore the properties of atoms, molecules and materials. A spectroscopic instrument commonly found in chemistry labs is the visible spectrophotometer, which helps a scientist learn how materials interact with light of different colors.

My goal is to create a simple Raspberry Pi based spectrophotometer that can help me quantify the amount of Red 40 in the "Watermelon Punch". Since I may want to turn this project into a laboratory experiment for my chemistry students, I will keep the hardware requirements simple and the programming accessible to students who may never have programmed before.

Supplies

All of the electronic parts should be available at your local electronics store or an online retailer. I use either Newark, http://www.newark.com, or Adafruit, http://www.adafruit.com:

- Green LED (20 mA max)
- CdS photoresistor [Adafruit #161, Newark #95F9039]
- 100 uF electrolytic capacitor

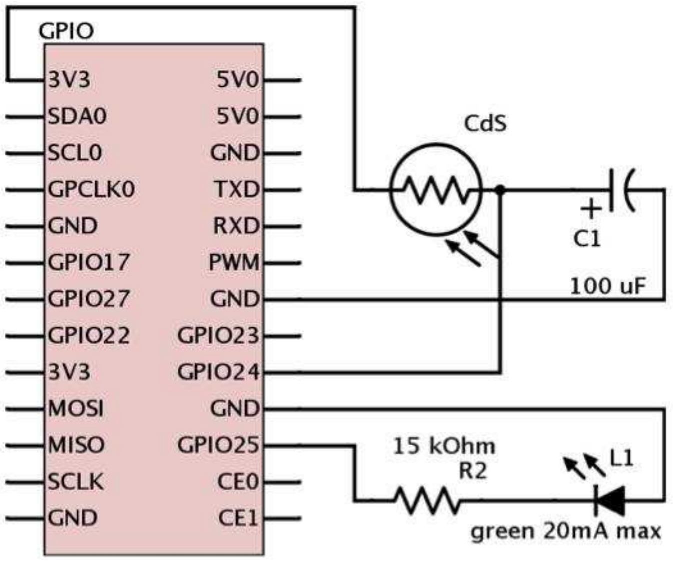
The science equipment I borrowed from my personal lab, but retailers such as Edmund Scientifics, http://www.scientificsonline.com, and the Labware section of Amazon, http://goo.gl/qyfqiy are good sources for the following items:

• Spectrophotometer cuvettes
• Graduated cylinder (10 to 25 mL)

The soft drink I used is Snapple "Watermelon Punch" and to create my standard solutions I used McCormick red egg dye (also known as food colouring).

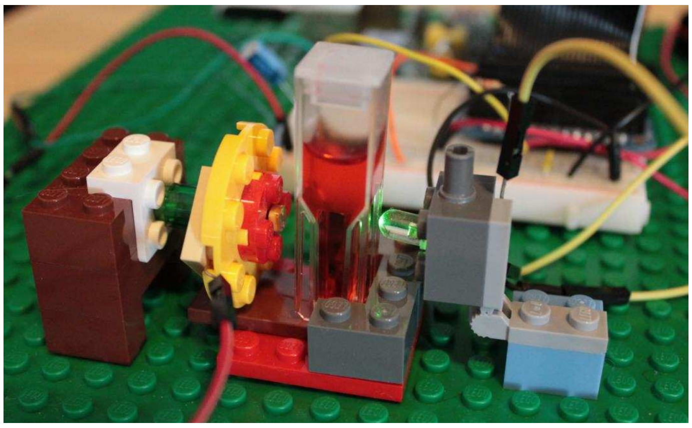
Instrument design

The basic parts of a spectrometer are the source, the sample and the detector. For this project I am only interested in measuring one colour (red) so I can simplify the source by using a green LED. The detector used in this experiment is a CdS (cadmium sulphide) photocell that has a resistance dependent on the amount of light shining on it.



Because the Raspberry Pi doesn't have an analogue input, I used some suggestions from Adafruit, http://learn.adafruit.com/basic-resistor-sensor-reading-on-raspberry-pi, for making analogue measurements. I connect the photocell to a capacitor and "ping" the circuit - essentially measuring how long it takes for the capacitor to charge. Since the charging time is influenced by the resistance in the circuit, this time will be related to the amount of light hitting the photocell.

To keep everything in one place, I used some LEGO® - which makes for a relatively cheap and robust optical bench. What is not shown, because it is a boring photo, is that the whole circuit is placed under a box. We only want our detector sensing light from the LED, thus we need to block out light from all other sources.



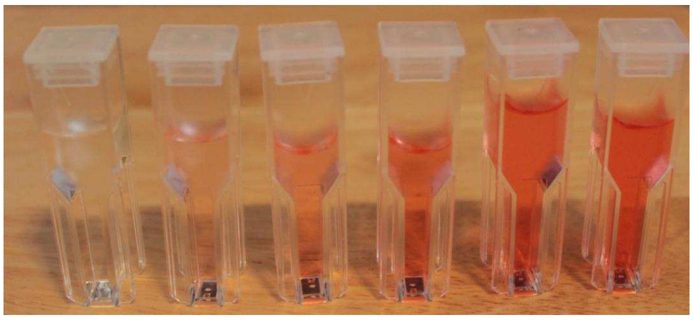
The science

White light, such as that from the Sun, is composed of all the colours of the rainbow. Objects have colours based on how the molecules that make up those objects interact with the light. Typically, materials can do one of three things with light: absorb it, reflect it, or let it pass through (transmit it).

When we see an object, say a red-coloured soft drink, the molecules that make up the soft drink absorb all the colours except for red, which it transmits and reflects. If there are a lot of molecules that absorb those colours, then the drink will appear dark red; conversely, if there are only a small number of molecules that absorb, then the drink will appear light red. There is a relationship, called Beer's Law, which shows that the amount of light absorbed is proportional to the concentration of the absorbing species, and it is this law that makes spectrophotometry so powerful.

To perform this type of analysis, I need a calibration curve, which shows the relationship between the instrument output and the concentration of Red 40 dissolved in water. I took one drop of commercial Red 40 egg dye and diluted it to 10 mL. Since one drop is approximately 50 uL (0.050 mL), this was a 200x dilution. The colour of the "Watermelon Punch" looks like it falls between 1000x and 5000x dilutions of the dye.

One way to make these solutions is to take 1 mL of the 200x solution and dilute it to 5 mL (making a 200 x 5 or 1000x dilution), then repeat this step four more times with larger final volumes.



Below is a chart summarizing the process. Each solution is made by diluting 1 mL of the 200x stock solution.

Final Volume (mL)	Dilution Factor	1 / Dilution Factor
5	1000x	0.00100
10	2000x	0.00050
15	3000x	0.00033
20	4000x	0.00025
25	5000x	0.00020

I used a slightly different dilution scheme than the one described here; however, the results are the same. We also need the reading from a cuvette with just water to account for light absorbed in the absence of dye. This is called a "background" reading. Ideally, when the detector output is plotted verses the inverse of the dilution factor, we should observe a linear relationship.

The software

Now that the Wolfram Language is available to all owners of the Raspberry Pi, we have at our disposal a comprehensive system for data acquisition, analysis and visualisation. I will describe one way to use Mathematica as the front end of the spectrophotometer. We need several functions to (a) discharge the capacitor, (b) read the status of the GPIO pin connected to our detector, and (c) time how long it takes for the capacitor to charge. For this project I wanted to do all the programming in Mathematica, which at the moment is not very fast when it comes to reading and writing to the GPIO. This is done through the functions `DeviceRead` and `DeviceWrite`. Here is the function `short` which is used to discharge the capacitor:

```
short[] := Module[{},
  DeviceConfigure["GPIO",24 -> "Output"];
  DeviceWrite["GPIO", {24->0}];
  Pause[2];
  DeviceConfigure["GPIO",24 -> "Input"];
]
```

`Module` is normally used to define local variables, although we have none in this function - so it is being used to keep the code tidy. The `short` function first configures GPIO pin 24 as an output pin and a 0 is written to it. After a two second delay, the pin is reconfigured as an input. This function serves to discharge the capacitor and prepare the detector for making a measurement.

In the `readpin` function we could use Mathematica's `DeviceRead` function. However, `DeviceRead` is a little too slow for the capacitor I am using. I circumvent this problem by reading the state of our detector pin a different way:

```
readpin[] := Module[{pin, out},
  pin = OpenRead["/sys/class/gpio/gpio24/
value"];
  out = Read[pin];
  Close[pin];
  out
]
```

Here, there are two local variables, `pin` and `out`; `pin` is used to open a file stream to the kernel treepath for the GPIO pins and the value is stored in `out`. In the Wolfram Language, output is suppressed by the ";" at the end of a line, so the last statement in `readpin` makes the function return the value of `out`.

The last function wraps together turning the source on and off, discharging the capacitor and timing how long it takes to charge:

```
 measure[] := Module[{np = 1200,data = {},
tth},
   DeviceWrite["GPIO", {25 -> 1}];
   short[];
   data = AbsoluteTiming[Table[readpin[],
{np}]];
   tth = Quiet@If[NumberQ[#], #, np]/np *
First@data &[Position[Last@data, 1, 1,
1][[1,1]]];
   DeviceWrite["GPIO", {25 -> 0}];
   tth
 ]
```

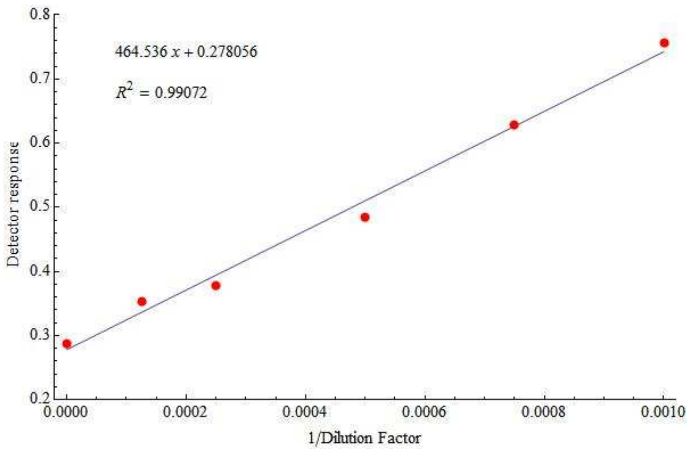
The three local variables in `measure` are: np (number of points), or the number of times the GPIO pin will be read; `data`, which is a place to store the pin values; `tth`, or the time needed for the GPIO to go high. The `DeviceWrite` function turns on the LED source and then the detector is initialized with `short`. The data is then collected along with the time needed for that operation.

Perhaps the most confusing line of code is next, which searches the data for the first instance of a "1" and determines how long it took for the GPIO pin to go high. Occasionally, no "1" will be found, indicating that insufficient light is reaching the detector. A warning would be thrown if `Quiet` was not applied to the command.

If you are trying this setup on your own, the value of np will probably need to be adjusted based on how well you block out stray light and the size of your capacitor. Mathematica has a bunch of tools to create lists of data, average multiple datapoints, plot the data and perform a least squares fit. I don't have enough space in this article to go through the details; however the on-line documentation, http://reference.wolfram.com/language, has plenty of examples.

The analysis

By measuring each sample five times and plotting the average output versus the inverse of the dilution factor, I obtained the results shown on the right. The linear relationship means I can be fairly confident in the results of the unknown measurement. The equation for a straight line is $\gamma = m\chi + c$; so by rearranging I can obtain an equation for finding the "dilution factor equivalent" for the soft drink.
These results indicate that the amount of Red 40



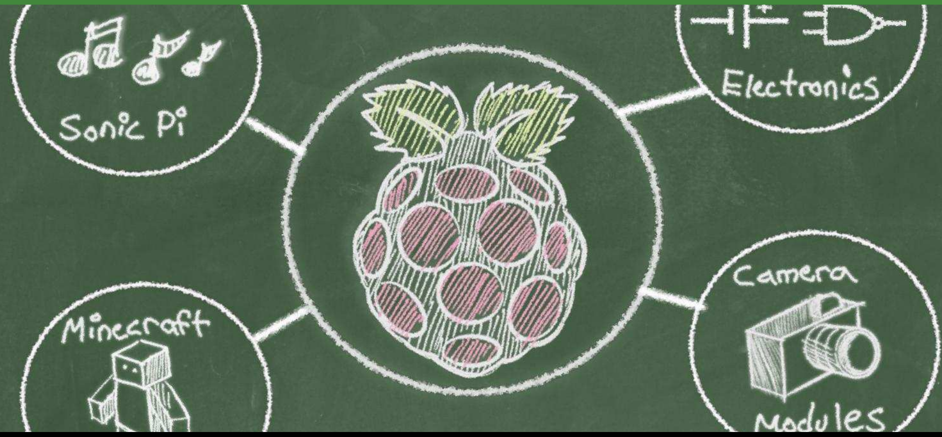
in "Watermelon Punch" is equivalent to taking one drop of the McCormick red dye solution and diluting it by a factor of 3000. In order to get a more usable value, I analyzed the commercial dye in my lab and found it to contain approximately 11 grams of dye per litre of solution. A little bit of math and we find that this is about 3.7 mg of dye per litre of soft drink. Using a commercial spectrometer, I obtained 4.6 mg of dye per litre, so the home-made spectrophotometer agrees reasonably well.

Presently the recommended acceptable daily allowance in the USA of Red 40 is 7 mg/kg body weight per day. I would have to drink a large amount of this beverage before reaching my daily limit of Red 40.

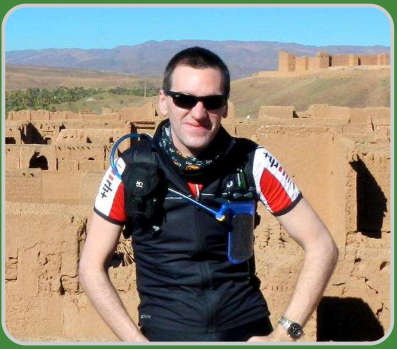
Where to go from here?

I only measured one of many different soft drinks - which leaves many other samples to probe. Similar tests can be used for other food colorings which would require the use of different LEDs. There are also improvements to be made to the spectrometer itself: an ADC can replace the resistor/capacitor detector, an autosampler can be added and multiple LEDs can be incorporated into the design to allow for multi-dye analysis.

The door to your Raspberry Pi driven home-chemistry lab is wide open and ready for discovery.



An interview with Carrie Anne Philbin



Colin Deady & Tim Cox
MagPi Writers

On May 14th The Raspberry Pi Foundation hosted the first Picademy. Over two days 24 teachers attended this free event and got first hand experience with the Raspberry Pi and its potential as a great teaching tool. Afterward, The MagPi talked with Carrie Anne Philbin, Education Pioneer at the Foundation about the event and the future of computing education in the UK.

MP: How many applicants were there for the Picademy and how were they selected?

Carrie Anne: Roughly 100 individuals applied for Picademy. A mix of primary and secondary teachers from Computing, Art and Science were selected. Some had prior experience of Raspberry Pi and some were completely new to it.We believe that computing is a cross curricular activity and that Raspberry Pi can help teachers develop that aspect in their teaching.

Four of the 24 were pre-selected so that they could act as lead learners, teachers who are already using Raspberry Pi. I think that mix was really important and integral to what we're trying to do with Picademy because getting those individuals together makes something exciting happen.

That mix of people and the opportunity to network by trying different ideas and thinking about how they can apply it to their classroom is really important. This has sparked ideas about how it can be cross-curricula as well.

MP: How many teachers do you see graduating from Picademy a year? Do you foresee a butterfly effect of graduates training others?

Carrie Anne: We are initially looking to run on average 6 Picademies per year with 20 to 24 teachers per event, with the next two taking place before September. Raspberry Pi Certified Educators are expected to help lead training of other teachers in their area, be enthusiasts for Raspberry Pi in their community, and to create teaching resources which will be made available via our website. Having only run one Picademy, we are still collecting data on how effective the butterfly effect is, but we hope to grow the education community across the country and then eventually the world.

"The greater emphasis on programming [in the new curriculum] made me worry about whether we had sufficient subject knowledge as a staff, or if we have the resources to enable this to be taught properly.

Picademy helped to put me in contact with teachers who are already doing amazing things in the classroom, and who generously share their ideas, expertise and planning!"

- Stacey Ramm

MP: The projects undertaken at Picademy really caught our attention, especially the bullet time Babbage Bear. How did that come about?

Carrie Anne: That was an idea that came up over dinner on the first night. An art teacher suggested Babbage could hold the chalk and a lit match during the sequence. It was really important to me that the teachers that we selected were not all the same teacher. You could run a Picademy that's all for primary, Key Stage 2 computing teachers, but that doesn't make any sense to me. It needs to be this mixed bag of people for interesting things to happen, and I'm hoping that the next Picademy will also get teachers from other subjects applying.

"Other teachers should not think that because they did not attend they are unable to use a Raspberry Pi in a classroom or get help on where to start. The first graduates from Picademy are happy to help; there are websites you can check and people you can tweet, so you should have a go. What's the worst that could happen?"

- Sway Grantham

MP: Teachers are clearly going to be the driving force for improvements in computer science in schools. A common question from the Raspberry Pi community is “what can we do to help?”

Carrie Anne: I believe in the collaboration between teachers, academics and industry experts in furthering the mission to improve how computer science is taught in schools. The community was a topic covered during Picademy by Matthew Manning from Raspberry Pi 4 Beginners, and Alex Eames from Raspi.Tv. Raspberry Jams were also high on the agenda with graduates being asked to form panel sessions at future Jams in their area. Code Clubs are also a great way for the community to help teachers in schools as both parties have an opportunity to learn from each other.

MP: What support is there to Picademy graduates post-certification?

Carrie Anne: We have given all graduates from Picademy a ‘Raspberry Pi Certified Educator’ badge and tag to use on their blogs and other social media, as well as a special tag on our forum. We’ve added a Picademy sub-forum so that graduates have a common place to discuss ideas and seek support from our community where it is needed. Although it is still early days, we are toying with the idea of creating more levels to keep our certified educators on their toes.

MP: With the emphasis on coding coming from government is there a concern that broader scientific literacy, coupled with appropriate critical thinking skills may be put to one side in preference to learn by rote computer programming? How does Picademy specifically aim to address this?

Carrie Anne: Picademy is not your regular run of the mill teacher training course; it is not us standing at the front delivering training about the Raspberry for two whole days.

Instead, it's about asking what ideas the teachers have, writing that down, working through project ideas and thinking where we can get engineers in to help them develop those.

The emphasis is on the attendees to take an active, hands on role in their learning. Developing their ideas during the two days is actively encouraged. We also reflect on how the attendees approached different tasks over the two days, and suggest ways they can take these approaches back to their classrooms to avoid rote computing programming. Creativity, critical thinking, problem solving, and collaboration are actively encouraged during Picademy.

"Something I wasn't expecting at all: one of our members had no computing experience whatsoever and said that when the new Computing curriculum was announced and they saw what was on it that they were going to quit teaching. But having spent two days with us had completely changed their mind."

- Carrie Anne

MP: How can Picademy support new teachers who likely graduated from school and university without themselves having been taught computer science at school?

Carrie Anne: Any UK teacher can apply for Picademy. We support all teachers at all points in their careers, and from any subject area. We also plan to work directly with teacher training centres to help them add Raspberry Pi to their training programs in the future.

MP: A mention of the use of GitHub for lesson plans caught our attention when reading reviews of Picademy from those that attended. Do you see collaborative toosl being important for teachers to share their ideas?

Carrie Anne: We are very excited about using Github and the introduction of GitHub Eduation [Ed: https://education.github.com]. Not only to host our resources but also as a tool for teachers to be able to share code with their classes, track changes that their students make, and for wider collaboration.

MP: There are many ways to engage with teachers. Why did you decide on the small group format for Picademy?

Carrie Anne: When I first started at the Foundation in January I went to BETT. While last year I found teachers were asking why bother with Raspberry Pi in the classroom, this year the questions were very different: how can we use Raspberry Pi in our classrooms? We talked about the resources we were creating, but then teachers would ask us when we would be providing Continuing Professional Development? CPD is something that teachers are supposed to do each year. Clive and I saw online that there are third party companies making quite a lot of money delivering CPD to teachers with Raspberry Pi but they are doing things like how to set up, how to plug it in, ie: really basic stuff. I'm self taught with Raspberry Pi, as are a lot of teachers I know. So I thought why should someone be selling our training when we could be providing it for free and do a much better job of it? In 2012 I became a Google Certified Teacher and I learnt how to think about teaching in a different way. I thought about that training I'd had and how I could apply it to Raspberry Pi.

And that's where Raspberry Pi Certified Educators came from. People would apply, we'd have them here for two days and they would then become certified.

To have our own group of trained teachers who we can call on around the country means we can point schools to those teachers and they can provide training or generally be a point of contact in the community.

"In utilising the Pi for things like weather stations, monitoring equipment etc, teachers would see an immediate difference in engagement, motivation and interest, not to mention the fact that they would be providing rich, contemporary learning experience for the young people whose education is entrusted in them."

- Allen Heard

MP: Taking new skills into the classroom now is great, but how does this become a longterm (five to ten years) investment in teaching?

Carrie Anne: The next six or so years are going to be interesting. The lessons secondary school teachers are delivering will be redundant as a new cohort arrives in year seven with major computational thinking skills. This transition is a long term process. Schools and teachers should see this period as an opportunity to change more than just their own skills, and also to consider the approaches they take to teaching and learning: to embrace technology and think cross curricular. Every adult is a lifelong learner, and teachers in particular consistently look to improve their practice. Research is required right now that will help inform the teaching practises of others over the next ten years.

The biggest fear I have is that instead of death by Powerpoint it is going to be death by Scratch because all primary schools are going to be using Scratch and then the students could go up to secondary school and do it all over again. So the question about five or ten years from now is really important because it is an ongoing process. It will be interesting seeing the primary kids going up to secondary school and how that transition is dealt with.

MP: An important point here seems to be the time to attend Picademy. How does that work in practice given pressures of the school term?

Carrie Anne: I agree. We put this event on in the school holidays. We are planning the next two towards the end of Summer term because after the exams teachers have more time to attend events. We are considering creating a fund for some schools. For example, if a teacher really wants to come but their school won't let them because it involves taking two days out of school, then the school can apply for a fund so we can cover the costs of their supply cover. This is just an idea at the moment.

MP: Going forwards are you hoping for more involvement from industry in Picademy to help teachers?

Carrie Anne: I'm a huge advocate for that as it was through going to Raspberry Jams and working with industry experts that I improved my knowledge and skills. I do envisage in the future that perhaps we can have some more industry experts come in and talk about what they do. We did give a presentation on the wider community including The MagPi, Raspberry Jams and the various YouTube channels for Raspberry Pi that people are creating.

We've set up a sub-forum for Picademy teachers and hope that they will engage with our community who are very helpful, especially if the teachers have a specific problem. Teachers can be really scared of joining forums because I think maybe about five years ago there was a lot of backlash if you asked a really NOOB question about something. Where-as now I think the Raspberry Pi community is such a fantastic one that it's changed my mind about that and we're trying to explain that to teachers. I think the

Picademy teachers lead this on the forum and create a much better way of industry people and teachers being able to collaborate together.

MP: Yes, we've noticed that the Raspberry Pi forums are very helpful.

Carrie Anne: Teachers are professionals who have gone through a lot of training, but may not have a solid computing background. However they are life long learners bred to ask questions, and when you ask something you are just looking for an answer.

It's really about building the confidence of teachers to ask a question. The first thing we started with at Picademy was that there are no stupid questions: if you have a question, ask it. Someone in the room will find an answer for you, and if no-one there could we'd go and find an engineer and get them in to help. How can you learn if you can't feel confident to ask a question? I think this is really important.

"I wanted to find out exciting (yet easy to set up) ways of using Raspberry Pi within the classroom. I was not disappointed.
The greatest thing is that I was shown the Raspberry is not just about textual programming and Linux commands. This is really important if it is to work within education."

- Ed Dourass

MP: We've found a number of teachers coming to us at events unsure where to start or who to ask about Raspberry Pi. There's a definite need for core help, like that provided by Picademy.

Carrie Anne: Absolutely. I think teaching and learning in the profession is really beginning to change. I always try to explain to teachers that you don't have to be the expert in the classroom, especially with a subject like ours. It's ok to learn with your students. As long as you can point them in the direction of where there may be an answer then that's good. You can't be an expert in everything, certainly not in the field of computing. The new programme of study is not just coding but is a whole range of different aspects of the subject.

"The teachers who attended the event are all committed to sharing their knowledge and supporting teachers who are finding the teaching of computing challenging. Picademy will help to build a valuable self-support network of teachers"

- Graham Hastings

MP: How do you see teachers being able to get budgets for the hardware needed on top of the Raspberry Pi's themselves? This is a concern teachers have raised with us at the Digimakers event.

Carrie Anne: Having the lead teachers in the room meant we could talk about how we managed to get funds at school: through crowd funding and going on Twitter. When I was teaching I asked on Twitter if anyone had ten monitors they were getting rid of and literally within an hour someone from a local office said they were getting rid of a load. I think teachers are open to looking for different solutions to this. Crowd funding is quite an interesting one. I think a lot of parents would happily put a fiver or tenner into a campaign to buy some kit.

Industry experts are asking all the time: "How can I help education?" There you go: donate a bit of money, kit or time.

MP: Do you hope that rather than a reboot of teaching computing in 1980s that this is a reboot and improve?

Carrie Anne: Exactly. It's an iterative process, and we're keen to explain this to teachers. What I did with Sonic Pi in the beginning was different to how Sonic Pi is now and where we're taking it in the future.

http://www.raspberrypi.org/picademy

Uninterruptible Power Supply for the Raspberry Pi

CW2.

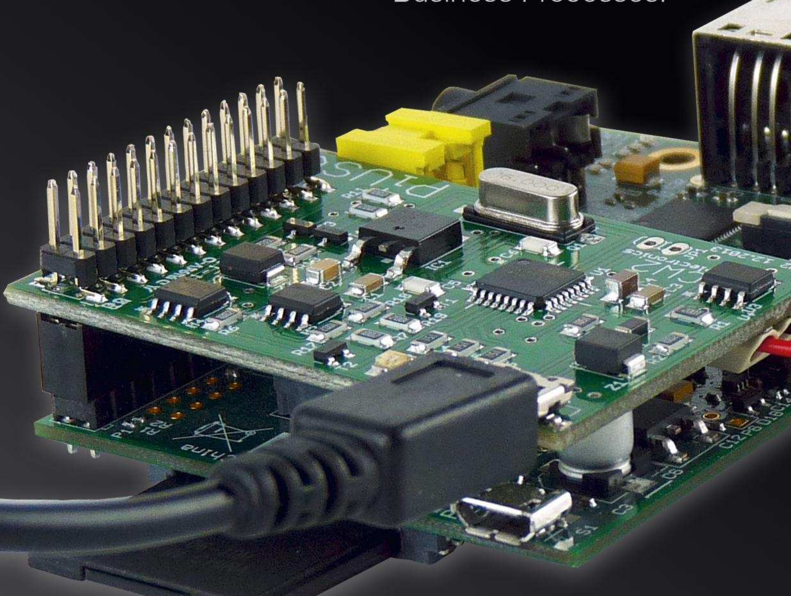
Innovation accelerating Business Processes.

Run your Pi anywhere & protect your SD Card from curruption created by loss of external power.

- Safe shutdown when power is off
- Corrupted SD-cards are a thing of the past
- Automatic restart of the Raspberry Pi after power failures

PI USV comes with a 6 AA-Batteryholder

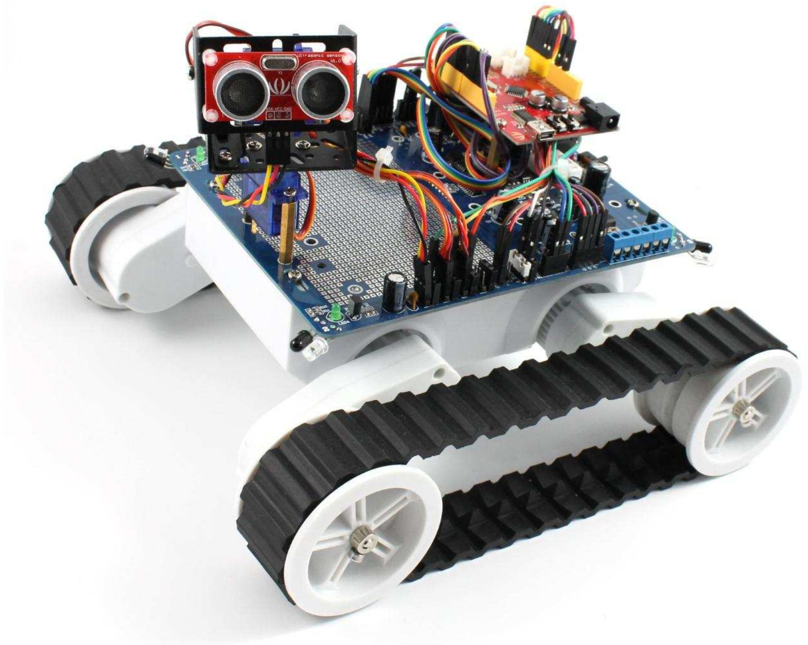
Buy now:
www.piups.net/distributors

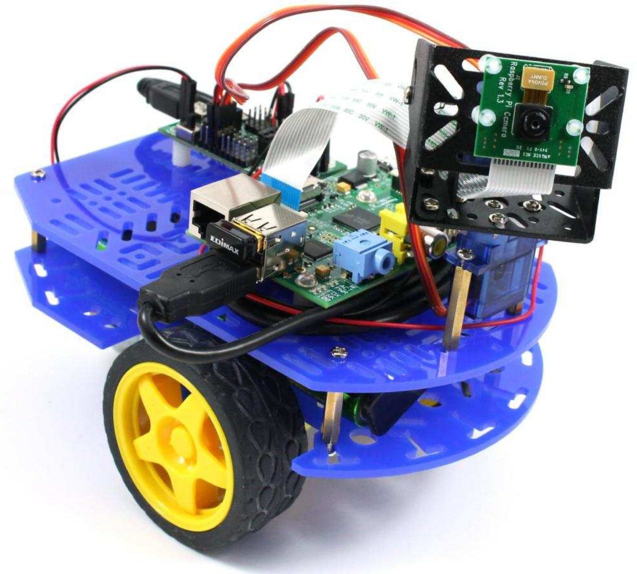


Picture shows the Pi UPS connecting to the Raspberry Pi

CW2. GmbH & Co. KG
Email: sales@cw2group.com • www.piups.net

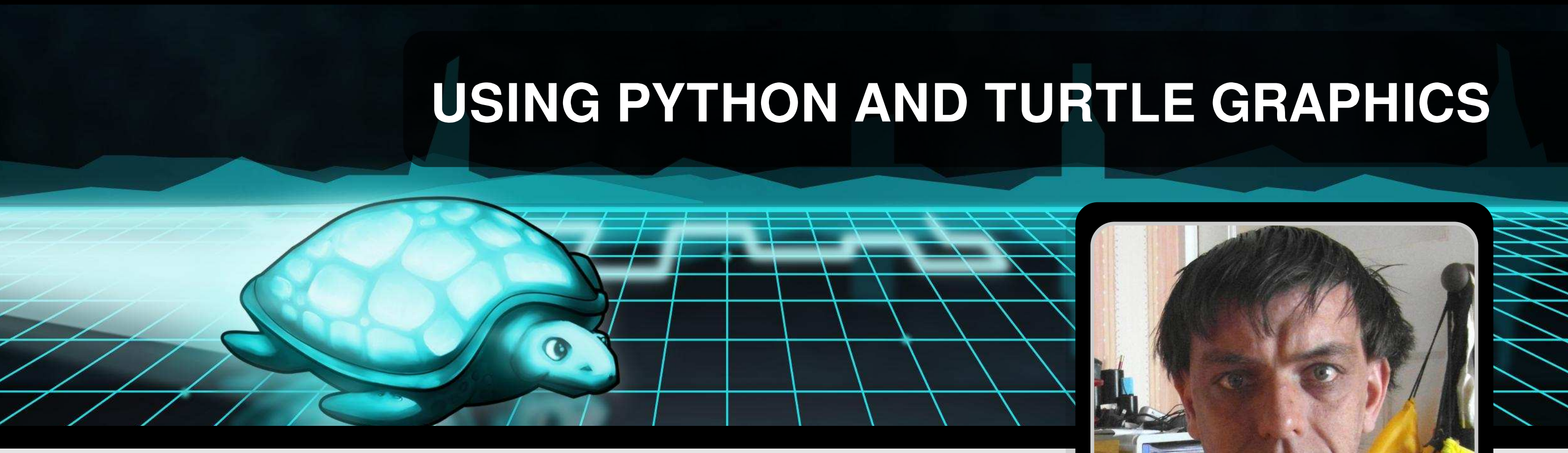
DAWN ROBOTICS





RASPBERRY PI AND ARDUINO ROBOT KITS

www.dawnrobotics.co.uk



Bringing Testudines out of the '80s

SKILL LEVEL : BEGINNER

Paul Sutton

Guest Writer

A bit of history

Back in the 1980s a computer language called LOGO was designed and aimed at schools for teaching simple programming: you had a cursor and what could be described as an imaginary pen. The software could move this pen around and draw pictures. As with a real pen it could be lifted up, repositioned, or the colour changed.

Many schools had a robot connected to a computer. The turtle's commands controlled the robot: FD would drive forwards, LT to turn left. PD (pen down) would lower an attached pen onto the paper (or floor...) and the turtle would draw as it drove. PU (pen up) likewise would cease drawing. For example:

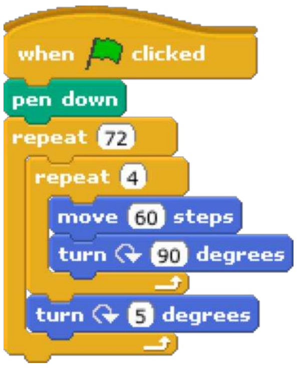
```
PD
REPEAT 4 [FD 50 LT 90]
PU
RT 360
```

would draw each side of a square approximately 5cm long. As a turtle was wired with a ribbon cable to its computer the final command is essential as it effectively unwinds the turtl'es cable by rotating a full circle clockwise.

Back to the present day

Fast forward to today and Python as well as other modern languages such as Ruby and Scratch (see below) have similar capabilities.In Python a module named Turtle allows you to draw shapes on a canvas using simple commands similar to LOGO.

For this introduction you will already need to be familiar with basic loops and straightforward programming in Python. If you are relatively new to Python but are familiar with Scratch then taking a look at creating turtle graphics in Scratch first may be of use. This also illustrates the sort of thing you could do with turtle graphics.



```
when [flag] clicked
pen down
repeat 72
    repeat 4
        move 60 steps
        turn ↻ 90 degrees
    turn ↻ 5 degrees
```

Within Scratch create the program to the left.

For our Python version we are fortunate that the turtle module is part of the standard installation. To begin open idle or nano and enter the following:

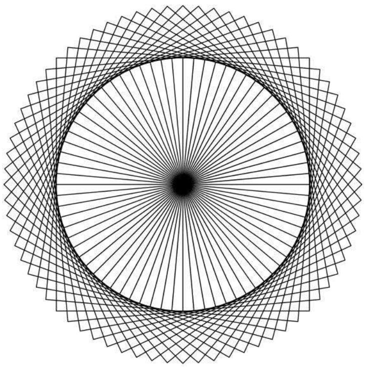
```
#!/usr/bin/env python
import turtle
import time

for n in range(0, 4):
    turtle.forward(50)
    turtle.left(90)
time.sleep(5)
```

We import 2 modules for this to work. `turtle` is essential and the second, `time`, keeps the image on screen for a few seconds, else Python will remove the canvas from the screen.

When run we create a loop that repeats 4 times. Within this loop we go forward 50 pixels, and turn left 90 degrees. Because this is repeated 4 times we end up with a square.

If we now repeat the same loop we can produce an interesting pattern similar to the one below:

```
for x in range(0,72):
    turtle.left(5)
    for n in range(0,4):
        turtle.forward(150)
        turtle.left(90)
```



This time we repeat the loop 72 times: there are 360 degrees in a circle: each time we repeat we are moving 5 degrees, so 360 / 5 = 72. Each new square drawn is 5 degrees anticlockwise from the previous one.

There is no reason to stick to drawing squares. As long as you know how many degrees are in the shape you want then it can be drawn. For example, an equalaterial triangle has 3 sides and 120 degrees. for this the loop is repeated 3 times and each turn is 120 degrees:

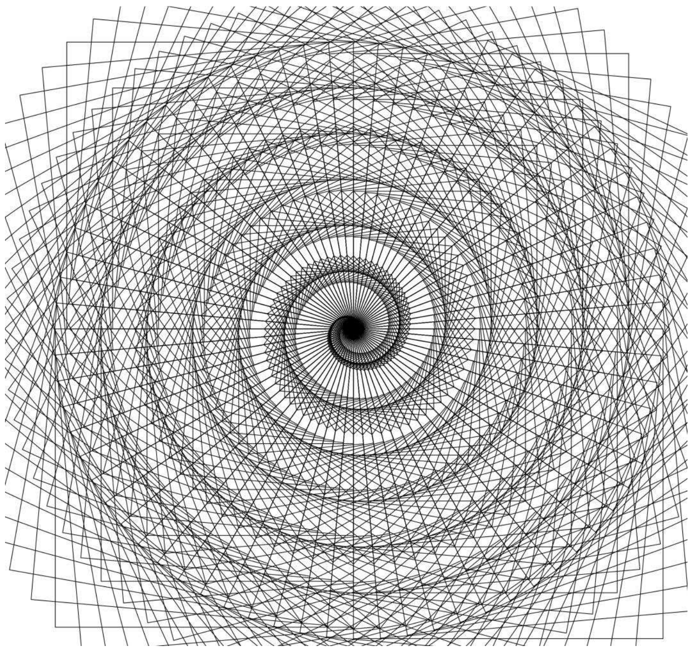
```
for x in range(0,72):
    turtle.left(5)
    for n in range(0,3):
        turtle.forward(150)
        turtle.left(120)
```

We can export the final pattern to a vector (EPS) file to keep for later, and require the user to click on the image to exit the program and save the image:

```
import turtle
import time

#set file name
fname="dial.eps"

for x in range(0,72):
    turtle.left(5)
    for n in range(0,4):
        turtle.forward(150)
        turtle.left(90)

ts = turtle.getscreen()
ts.getcanvas().postscript(file=fname)

print "Saved image to: ", fname
print "All done. Click image to exit."

turtle.exitonclick()
```

You can do some other clever things too. For example, start drawing a square 50x50 pixels, and then in each interation increase by 1 until it reaches 200x200 (or what ever value you want). If you start off with 50, then its not too small to see. But you can start with 1. you end up with something that looks like this.

```
for x in range(50,200):
    turtle.left(5)
    for n in range(0,4):
    turtle.forward(x)
    turtle.left(90)
```



This article just scratches the surface of what turtle can do. Try out different values, have a look at the documentation for other commands you can use such as turtle.speed, and have fun.

Further reading

http://en.wikipedia.org/wiki/Logo_(programming_language)
http://en.wikipedia.org/wiki/Turtle_graphics
http://docs.python.org/2/library/turtle.html

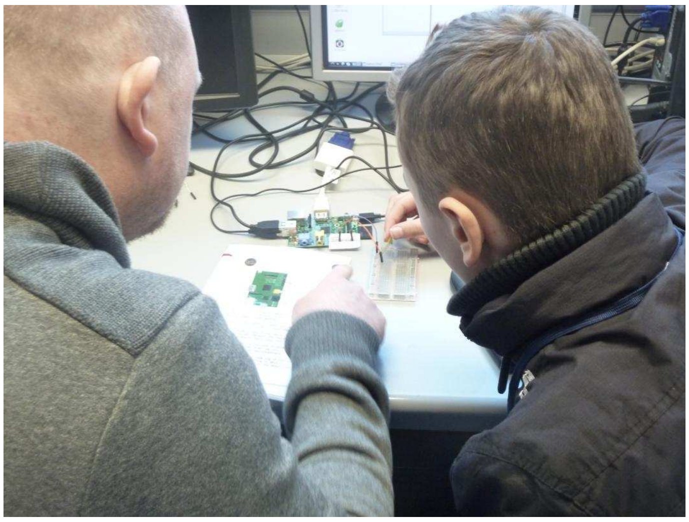
Ysgol BrynElian High School

Allen Heard

Guest Writer

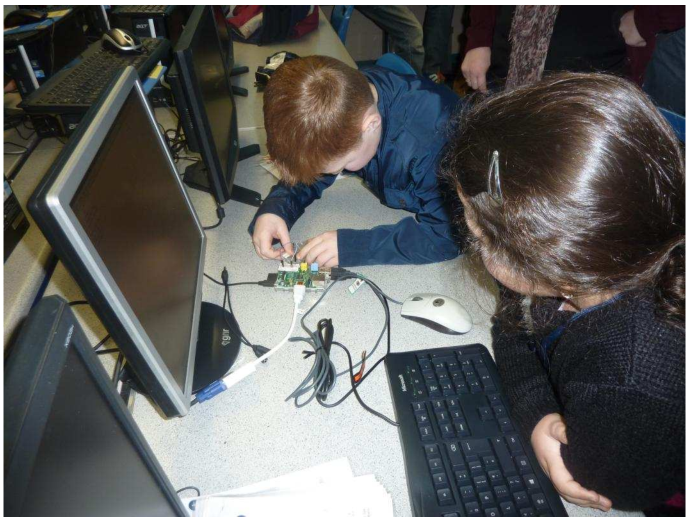
Tech-Dojo with the Raspberry Pi

As soon as the Raspberry Pi was launched, I was keen to get started with this awesome innovation. The Raspberry Pi plays a key role in our school's Tech-Dojo events that have seen over 150 children, parents and teachers attend fun packed days on a Saturday. These days have included learning about coding, electronics, Minecraft and App building.



As a result of the Tech-Dojo events, the school has won an award for digital projects that engage with the local community through the annual North Wales 14-19 e-learning competition and I have recently been awarded a Silver Pearson Teaching award for Outstanding use of Technology in Education.

Digital Literacy is almost a given in today's society, however computing is less so. The aim of our sector-leading Tech-Dojo sessions is to introduce the latest technology to the local community. This includes current students, primary children, home educated children, parents and even other teachers through continuing professional development (CPD). Our sessions introduce them to the possibilities of computing through a range of fun activities. We use Raspberry Pi's extensively, to introduce visual and text based coding. We add electronics into the mix, such that participants can control lights and code working circuits. These events are promoted through local primary schools, Twitter, and the Tech-Dojo App for smart phones.



The Tech-Dojo sessions have seen a wide range of attendees, including children from schools across North Wales, as well as teachers looking for CPD. Members of the local educational authority (LEA) have also attended, to experience the learning atmosphere for themselves. These sessions are ultimately aimed at raising awareness of the possibilities of using the Raspberry Pi, Scratch, Minecraft, and App building tools, to help children to realise the possibilities of being skilled in these areas. In doing so, we begin to raise the aspirations of the young people in our community.

We pride ourselves on having staff with the expertise, motivation, and enthusiasm in the field of computing, to be able to make a difference to the community through our inclusive attitude towards engaging with those around us. This includes children moving from primary to secondary schools, as well as secondary pupils. For us, it is all about getting technology into children's hands and helping them to realise their potential. We also aim to facilitate sessions, to include home educated children in the community.

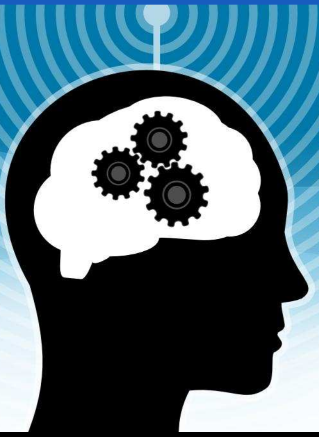
Using our innovative practice developed here at Ysgol Bryn Elian, we aim to deliver first class sessions that inform and educate. Our goal is to reach as many people as possible, to raise aspirations across the region, in addition to preparing schools for changes in the educational landscape through sector-leading practice.

To date we have run two highly successful Tech Dojo's, with over 150 children from 20 schools across North Wales taking part. These fun-packed days have been supercharged with technology, to ensure all are engaged, having fun and most of all are learning! Impact of these events is of paramount importance to us. Therefore, we seek feedback on every session. The positive comments have been staggering!

We try to provide different content at each event. For example, we already have six new activities planned for the next Tech-Dojo (September 2014), where we aim to create an intruder alarm, use motors to create optical illusions and create the classic wire-buzz game, all using a Raspberry Pi that the children will code themselves!

Computing plays an essential role in our daily lives. In response, our school embedded computing within its KS3 curriculum even before it was on the Government's educational agenda. The school has delivered taster sessions to children from local primary schools, using Pygame to make Carrie Anne Philbin's Space Invaders graphics and produced Scratch code to re-create popular games such as Angry Birds and Flappy Bird. These activities really get the children engaged in algorithm design and computational thinking. We are now seeing the fruits of our labour. The school is right at the front of this exciting and innovative area of the curriculum, offering CPD to local Primary Schools in their bid to improve skills prior to impending changes on the educational landscape. The future looks bright for technological innovation across North Wales.

As well as being a teacher, Allen is a certified educator for the Raspberry Pi and was part of the first Picademy. He is now working with the Raspberry Pi Foundation, to produce teaching materials. He also plans to offer Raspberry Pi CPD training across four education authorities in North Wales during the summer.



Richard Wenner

Guest Writer

Raspberry Spy Part 1: Understanding Wi-Fi

SKILL LEVEL : INTERMEDIATE

Introduction

Understanding network protocols and associated security is a highly useful skill in today's world of internet technology. While many network protocols are common to both wired and wireless networks, wireless (Wi-Fi) networks pose additional challenges.

In this series of articles, a Wi-Fi network will be needed. This could be provided by a home router or wireless access point. Make sure that you either own the wireless access point or have explicit permission from the owner to perform some packet sniffing experiments. Do not use the techniques discussed in this series on other networks, unless you have explicit permission from the wireless access point owner or Wi-Fi system administrator.

Equipment needed and configuration

For this series a Raspberry Pi Model B, a known working USB Wi-Fi dongle, an inquisitive mind and optionally an empty Pingles® tube are all that is needed. Having a Wi-Fi dongle that works when plugged directly into the Raspberry Pi USB port is useful for portable operation.

Start from the most recent Raspbian image. Make sure that the image is up to date with:

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

Do not forget to use `passwd` to change the default password for the 'pi' user account to something less well known! For this series it is preferable to boot to the console plus the SSH daemon also needs to be enabled. Both of these can be set with:

```
sudo raspi-config
```

Many of the network tools that are used in this series require root user privileges. As typing `sudo` in front of each command can become tiresome, after logging on to the 'pi' user account a new shell can be started as root by typing:

```
sudo -s
```

To exit the root shell, type:

```
exit
```

Be careful not to remove or overwrite important system files while using the root shell.

Wi-Fi sniffing

It is relatively straight-forward to sniff your Wi-Fi network, or a network that you have explicit permission to sniff. All that is needed is a USB Wi-Fi dongle, which can be either connected directly to the Raspberry Pi or via a powered USB hub. Although these dongles are small and low cost, they are packed with technology.

Twenty five years ago, when I first started designing this type of equipment, it was difficult to fit the electronics into something the size of a very large box of matches. Today everything fits onto a board almost the same size as the USB connector!

If the Wi-Fi dongle is plugged directly into your Raspberry Pi, keep the keyboard connected but you can remove any mouse. We are not going to use the GUI interface, so a mouse is not necessary.

Testing a Wi-Fi dongle

All Wi-Fi dongles are not the same. They differ in performance based upon the chipset used in their design. You can assess details about the capability of your Wi-Fi dongle by installing a small testing program with,

```
apt-get install iw
```

and then entering:

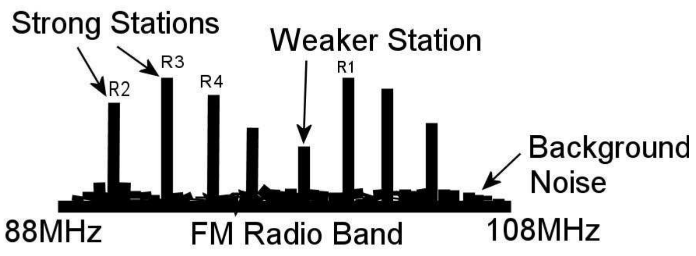
```
iw list | less
```

Piping to `less` with the | character allows you to see all of the output. Scroll up and down using the arrow keys and press <Q> when finished. The output displays a good deal of information about the capabilities of your Wi-Fi dongle. It also displays other information you may want to question later, but for the moment the important section is 'Supported interface modes:'. 'Monitor' must be in this list to continue, otherwise you will need another dongle. See http://elinux.org/RPi_USB_Wi-Fi_Adapters for known working and not working Wi-Fi dongles.

'Monitor' mode (rfmon) is the useful ability that allows us to observe everything that is being transmitted on the Wi-Fi bands. It can be used to sniff 802.11a/b/g/n traffic.

Hiding secret radio signals

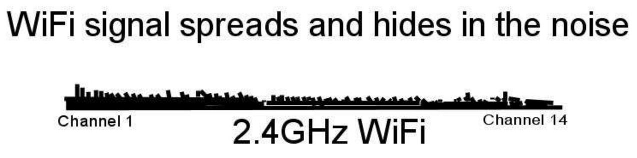
If you wore a special pair of glasses that could see radio stations, the FM radio band would look like this...



Each of the stations would appear as individual peaks in power that stay where they are. You tune your radio to receive each station.

Wi-Fi uses a different type of radio signal. It employs techniques that make it difficult to receive; in fact such techniques were used by the military to hide radio transmissions. One technique keeps changing frequency so that it 'pops up' all over the band. You can only receive it if you keep re-tuning your receiver, but this means knowing where the signal is going to be! To compound the problem, this re-tuning has to be done thousands of times a second (so has to be done electronically)! This type of signal is called 'Frequency Hopping Spread Spectrum' (FHSS) and is a patent jointly held by the 1930's Hollywood actress Hedy Lamarr and composer George Antheil.

A second type of encoding takes the information to be sent and combines it with a repeating, high-speed, pseudo-random digital key. This key also has the effect of spreading the radio signal across the band and hiding it in the background radio 'noise'. The signal can only be recovered by a receiver with a synchronised copy of the same key. The original, hidden information 'pops out' of the noise once the key is applied. This process is known as 'Direct Sequence Spread Spectrum' (DSSS). Both FHSS and DSSS techniques are still used to hide clandestine radio stations.



Kismet monitoring software

Kismet is the wireless network detector of choice we are going to install. Downloading and installing it will take up to an hour as neither `apt-get` nor `aptitude` will install the latest version of Kismet reliably. So we need to install it manually... but this is a good exercise!

First we need to install some supporting code. Enter as one continuous line or package by package, as shown:

```
apt-get install libncurses5-dev
apt-get install libpcap-dev
apt-get install libpcre3-dev
apt-get install libnl-3-dev
apt-get install libnl-genl-3-dev
```

Next we need to download the Kismet code. Enter all of the following on a single line:

```
wget http://www.kismetwireless.net/code/
kismet-2013-03-R1b.tar.gz
```

The downloaded file needs decompressing:

```
tar xvf kismet-2013-03-R1b.tar.gz
```

The decompression causes a long list to stream down the screen. Move into the newly created directory,

```
cd kismet-2013-03-R1b
```

and type the following on a single line:

```
./configure --prefix=/usr --sysconfdir=
/etc --with-suidgroup=pi
```

(Note: Every option is separated by a space and then two hyphens).

This causes another long list to stream down the screen. We now compile the application. Enter:

```
make
```

Things now really slow down. This stage can take up to an hour. Finally, to install enter:

```
make suidinstall
```

This allows the 'pi' user to run Kismet if required.

Kismet requires a place to store details. Enter:

```
mkdir /root/kismet_logs
```

Kismet configuration

The final few lines that appear following the build refer you to the README file. This is worth reading. Enter:

```
cat README | less
```

Kismet retains its configuration details in the file `/etc/kismet.conf`. This file should be edited and the following lines changed, if necessary:

```
nano /etc/kismet.conf
```

In nano press <CTRL>-W to find the `logprefix`, `ncsource` and `gps` parameters and uncomment or edit the lines to read:

```
 logprefix=/root/kismet_logs
 ncsource=wlan0:forcevap=false,validatefcs
=true
 gps=false
```

Yes, that final line does switch off any GPS (Global Positioning System) connection. You might ask why you would want to map Wi-Fi details to physical locations? It's an excellent question, but this is exactly what Google did as they roamed the streets photographing for Street View... and forgot to tell us!

Wireshark support

Kismet works in partnership with Wireshark, a network protocol analyser used for network troubleshooting, analysis and education. Let's install that now:

```
apt-get install wireshark
```

Wi-Fi dongle configuration

This part of the set-up has been left until now, in case you have been using your dongle for internet access. Now is the point where we break any existing link and reconfigure the Wi-Fi dongle to work specifically with Kismet. Enter:

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

Comment out the following lines, by adding a # character at the front, so they look like:

```
 #allow-hotplug wlan0
 #iface wlan0 inet manual
 #wps-roam /etc/wpa_supplicant/wpa_supplic
ant.conf
 #iface default inet dhcp
```

We now edit the file `ifplugd`. This is used to automatically configure the ethernet device when a cable is connected and unconfigure the device when the cable is unplugged. We want to change it so that it ignores the Wi-Fi dongle. Enter:

```
nano /etc/default/ifplugd
```

Change the following lines from,

```
INTERFACES=”auto”
HOTPLUG_INTERFACES=”all”
```

to:

```
INTERFACES=”eth0”
HOTPLUG_INTERFACES=””
```

After saving the changes, reboot your Raspberry Pi by entering:

```
shutdown -r now
```

Using Kismet

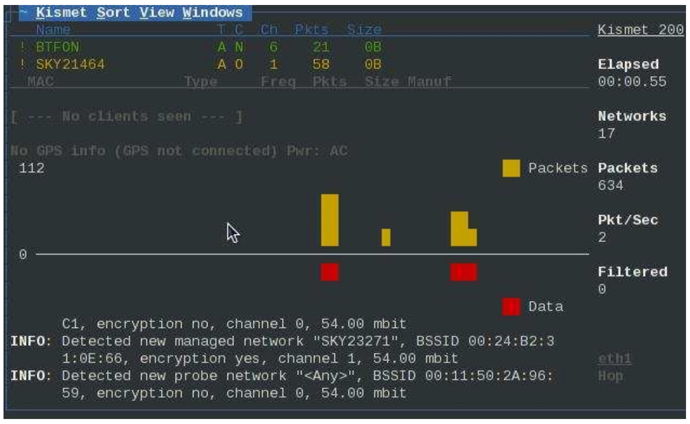
Once the Raspberry Pi reboots, log in as the 'pi' user and enter:

```
sudo -s
kismet
```

The text screen will be transformed and the Kismet server console will appear. Once all appears to be ok you can press <Tab> to select Close console window then press <Enter>.

[Ed: You can tell Kismet the chipset of the Wi-Fi dongle you are using, but by default it will try to determine this automatically. I tested two dongles. The RT5370 based dongle was successfully discovered but the RTL8188CUS based dongle was not.]

A series of questions will be asked. The first is a warning about running as root. Press <Tab> to select OK then press <Enter>. Press <Enter> again over the Yes option to automatically start the Kismet server. Also choose Start when offered the logging and console options. If all is well the screen should burst into life, displaying networks, power levels and a rather clever steaming screen of data packets.



```
~ Kismet Sort View Windows
  Name                 T C  Ch  Pkts  Size                      Kismet 200
! BTFON                A N  6    21    0B
! SKY21464             A O  1    58    0B                       Elapsed
 MAC              Type      Freq Pkts  Size Manuf               00:00.55

[ --- No clients seen --- ]                                     Networks
                                                                17
No GPS info (GPS not connected) Pwr: AC
 112                                               Packets      Packets
                                                                634

                                                                Pkt/Sec
                                                                2
 0
                                                                Filtered
                                                                0
                                                   Data
      C1, encryption no, channel 0, 54.00 mbit
INFO: Detected new managed network "SKY23271", BSSID 00:24:B2:3
      1:0E:66, encryption yes, channel 1, 54.00 mbit            eth1
INFO: Detected new probe network "<Any>", BSSID 00:11:50:2A:96:  Hop
      59, encryption no, channel 0, 54.00 mbit
```

Press the <~> key (known as tilde) to reveal the pull down menus. These provide you with access to a visual representation of your local Wi-Fi world. What you see should only raise more questions, so explore!

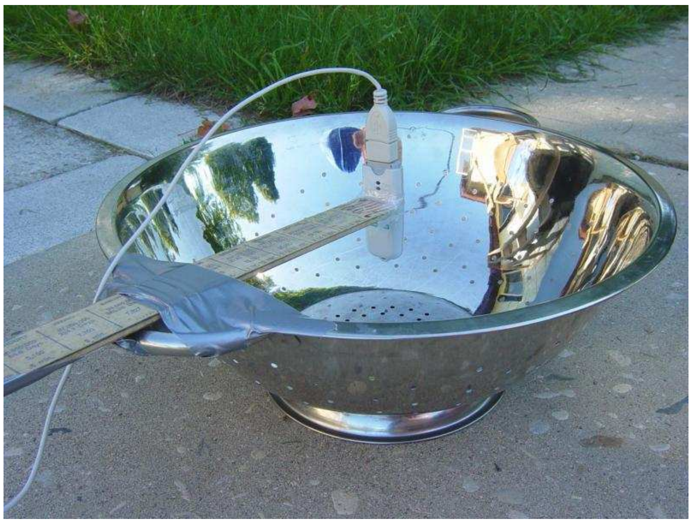
To cleanly shutdown the Kismet client and server, press <~> then <Enter> and then press <Q>. You may want to investigate the contents of the log file located in `/root/kismet_logs`. We will make more of this in a coming article.

Greater range

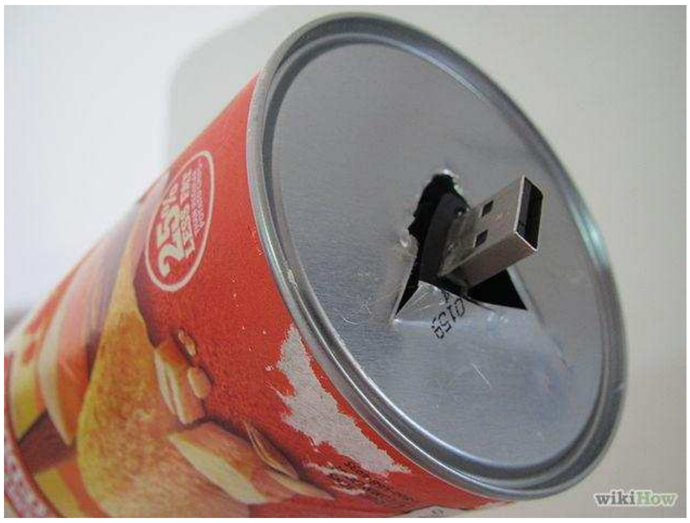
The light from a bulb or LED is boosted into a beam by the reflector inside a car headlight or bicycle torch.

The only difference between light waves and radio waves is the frequency. (We have two limited bandwidth receivers embedded in the front of our heads, each with its own reflector – the retina).

Similarly we can boost the signal received by a Wi-Fi dongle by carefully placing it at the focal point of a bowl, just like the LNB on a satellite dish. Ideally the bowl will have a parabolic shape. At http://karlherrick.com there are examples based on a wok and a spaghetti strainer, as shown below. This will allow you to sniff your Wi-Fi networks from up to a kilometre away (line of sight).



There is also an article on wikiHow at http://www.wikihow.com/Increase-the-Range-of-Your-Wifi which uses a Wi-Fi dongle and a Pringles® container to provide a significant increase in range.



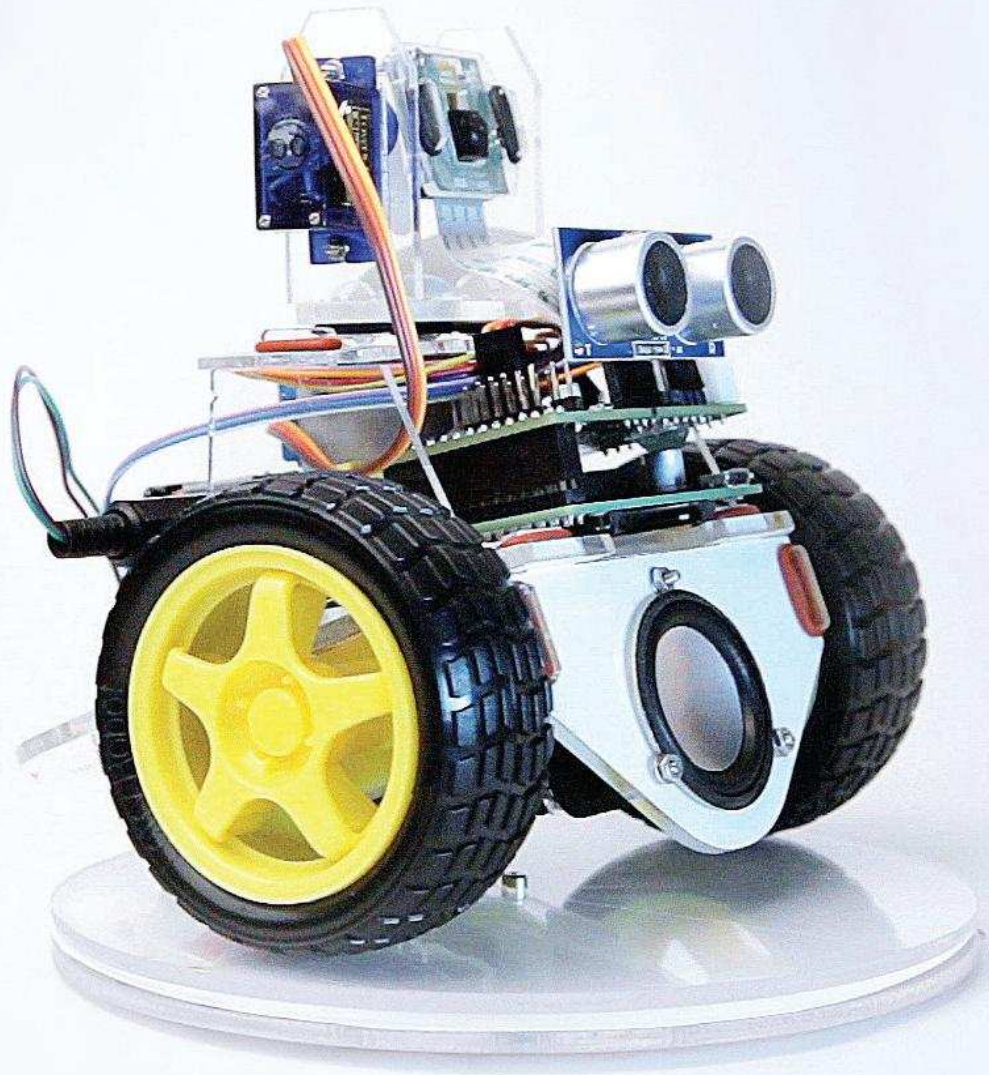
Dossier

We have covered a large amount in this article and you should have a very effective device for monitoring Wi-Fi signals. In the next article we will learn how to analyse the data we have collected.

GIVE YOUR RASPBERRY PI ROBOT POWERS!

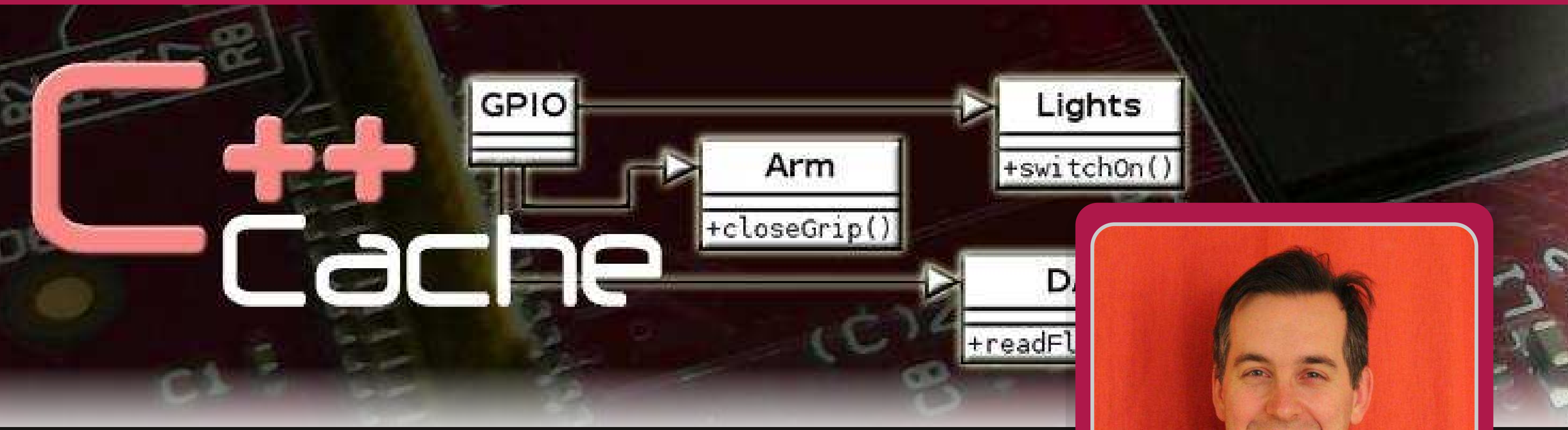
B π T

www.PiBot.org



Now introducing...
The PiBot Workshop Experience

More info and Signup @
www.pibot.org/workshops



6 - Communication between objects

SKILL LEVEL : ADVANCED

W. H. Bell

MagPi Writer

This month's article is a continuation of the introduction to C++ classes given in Issue 23 of The MagPi. In this article communication between objects is introduced.

Object-object communication

Objects are instances of a class. This means that if two objects of a class are created, they occupy two different memory locations. In some cases, it can be very useful to define a class with one or more data members that are objects of a different class. A class that contains other objects as data members can easily call the member functions of the objects it contains, in a similar manner as the `main()` function called the member functions of a simple class in Issue 23. However, for an object that is assigned as a data member to call the member functions of the object that contains it, it should call the correct instance in memory. This can be achieved by using the `this` pointer to pass a pointer to the data member object.

Communication between two objects

Graphical user interfaces can include of a frame class that contains other objects. This frame class can then create other objects that it owns. When the frame class is deleted, the objects that it contains should also be deleted. The frame class will typically have a size in number of pixels and be able to receive commands from the user. For a given frame class, there might be a class that describes a control panel. In this example, when the frame class received a command or is resized it should pass this information on to the control panel class. When a button on the control panel is clicked with the mouse pointer, it could need to access some of the information in the frame class or cause the frame class to exit.

The hypothetical frame and control panel example can be simplified to a class that contains a pointer to another class. In this simplified example, the container class is called `Parent` and the class it contains a pointer to is called `Child`. Open a text editor such as `nano` or `emacs` and create a new file called `Child.h`. Then add the C++ code at the top of the next page. As previously discussed, the precompiler `#ifndef`, `#define`, `#endif` syntax is present to prevent the class from being defined twice if the header file is included twice. As a convention, the name of the file in upper case is typically used for this check.

```
#ifndef CHILD_H
#define CHILD_H

class Parent;  // Forward declaration to reduce precompile time.

class Child {
public:
  Child(Parent *);  // Construct a Child with a Parent pointer
  void run(void); // A member function to call the Parent class

private:
  Parent *m_parent; // A data member to store a pointer to the Child's Parent.
};

#endif
```

The `Child` class has a simple constructor that accepts a pointer of `Parent` class type, a public member function `run()` that takes no arguments and one data member to store a pointer to the `Parent` object.

Create another file called `Child.cpp` and add:

```
#include <iostream>
#include "Parent.h"
#include "Child.h"

using namespace std;

Child::Child(Parent *parent):
  m_parent(parent) {
}

void Child::run() {
  cout << "Parent frame dimensions = {"
       << m_parent->x() << ", "
       << m_parent->y() << "}" << endl;
}
```

This file contains the implementation of the constructor and the `run()` member function. The constructor sets the value of the private data member pointer, using the input `Parent` pointer. The member function prints the x and y values of the `Parent` object on the screen.

Now create a fill called `Parent.h` and add:

```
#ifndef PARENT_H
#define PARENT_H

class Child;  // Forward declaration to reduce precompile time.

class Parent {
public:
  Parent(unsigned int x, unsigned int y); //  Construct a Parent frame with some dimensions.
  ~Parent(void); // Destructor to clean up the particle pointer
  void run(void); // Call a member function in the Child class
  unsigned int x(void) { return m_x; } // Get the x dimension
  unsigned int y(void) { return m_y; } // Get the y dimension
  Child* child(void) { return m_child; }  // Get the child pointer

private:
  Child *m_child; // A data member to store a pointer to the Child class
  unsigned int m_x;  // A data member to store the x dimension
  unsigned int m_y;  // A data member to store the y dimension
};

#endif
```

This class contains a constructor that takes x and y dimensions, a destructor to delete the Child pointer, a run() member function to create a Child object and call its run() function, accessor functions to return the values of m_x, m_y and the m_child, and private data members that contain a Child pointer and dimensions x and y. Since the Parent class declaration includes a pointer of Child type, the Child class must be known to the compiler beforehand. If the private data member contained a Child object (rather than a pointer to a Child object), it would be necessary to include the Child header file. However, since a pointer is used it is enough to tell the compiler that Child is a class. It is a good idea to do this where possible, since it speeds up the precompilation process.

Create another file called Parent.cpp and append:

```
#include "Parent.h"
#include "Child.h"

Parent::Parent(unsigned int x, unsigned int y):
  m_child(0),
  m_x(x),
  m_y(y) {
}

Parent::~Parent() {
  if(m_child) delete m_child;
}

void Parent::run() {
  if(!m_child) { // Only create a child if there isn't one already
    m_child = new Child(this);
  }
  m_child->run();
}
```

This is the implementation of the Parent class. It contains the implementation of the constructor, destructor and run() member function. It is not necessary to implement the accessor functions, since they are already implemented in the Parent.h header file. In the Parent constructor, the private data members are initialised in the same order as they are defined in the Parent.h file. If they were not in the same order, a warning message would be reported during compilation. The Parent destructor checks to see if the m_child pointer is null. If it is not null, it deletes the object associated with the pointer. If the pointer is null, the run() function creates a new Child pointer on the heap. Then the run() function calls the run() function in the Child class. Notice that the Child class is instantiated by calling the Child class constructor that takes a Parent pointer as an argument. In this case, the this pointer is used to pass a pointer to the current Parent object to the Child object.

This example program has two more files that are needed to get it working. First, create a file called main.cpp and add the main() function given below:

```
#include "Parent.h"

int main() {
  Parent *parent = new Parent(200, 150);
  parent->run();
  delete parent;
  return 0;
}
```

The main() function creates an instantiation of the Parent class on the heap, calls its run() member function and then deletes the Parent object. The Parent constructor is passed the x and y dimensions as 200 and 150. Similar to the Child.cpp and Parent.cpp files, the header file that contains the class declaration is included to allow the code to be compiled.

The last thing needed to build the example code in a straight forward manner is the Makefile. Create a file called Makefile in the same directory as the other source files (Child.h, Child.cpp, Parent.h, Parent.cpp, main.cpp) and add to it:

```
CC=g++
TARGET=pc
OBJECTS=main.o Parent.o Child.o

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

clean:
	@rm -f *.o *~

veryclean: clean
	@rm -f $(TARGET)

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

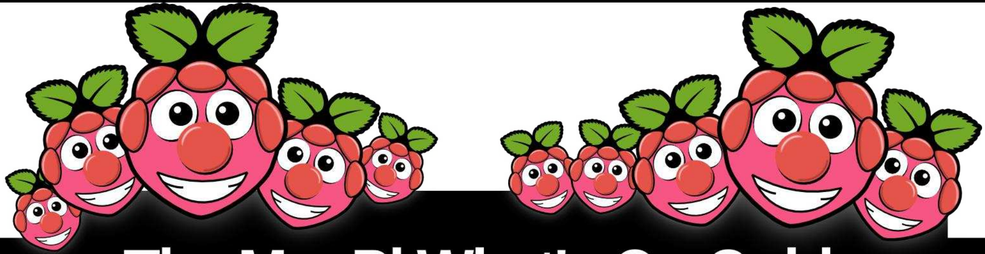
where the indents shown should be a single tab. Then type

```
make
```

to build the executable. Once the executable has compiled, type

```
./pc
```

to run the example code. As an exercise, create a Parent object inside the Child class and access its run() member function instead. The values will not be the same as the other Parent object returns, since the two objects are separate instances in memory.



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

Bristol Digimakers

When: Saturday 14th June 2014, 10.30am to 4.30pm
Where: At-Bristol, Anchor Road, Harbourside, Bristol, BS1 5DB, UK

A series of community technology events aimed at children (7-17), parents and teachers. An introduction to 'making' in the digital world. http://www.eventbrite.com/e/11031145453

Southend Raspberry Jam

When: Saturday 21st June 2014, 10.00am to 5.00pm
Where: Temporary Arts Project, North Road, Southend on Sea, Essex, SS0 7AB, UK

There will be talks, show and tell, workshops, videos, robots, flashing leds and lots of fun.
http://soslug.org

The Professional Academy | Raspberry Pi

When: Tuesday 24th June, Tuesday 1st July, Friday 4th July
Where: Crownford House, Swan Street, Merthyr Tydfil, Wales, UK
Develop skills to program the Raspberry Pi, including the ability to interface with un-manned aerial vehicles. Course fee is fully funded for those who live or work in certain areas of Wales.
http://professionalacademy.southwales.ac.uk/raspberrypi

Keighley Raspberry Jam

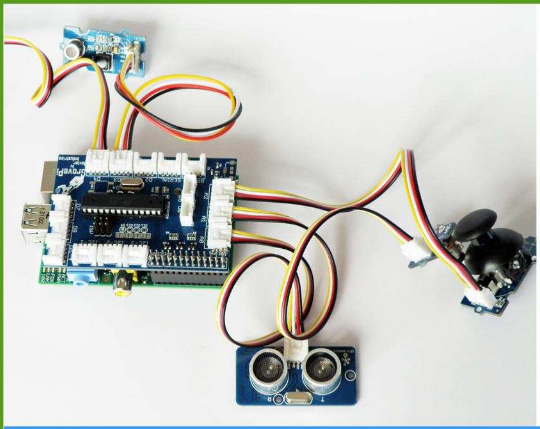
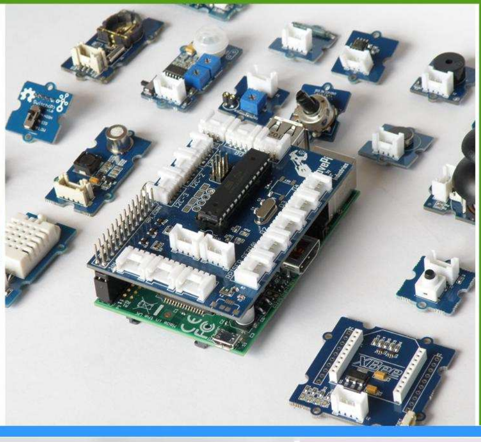
When: Saturday 28 June 2014, 1.00pm to 4.00pm
Where: Fab Lab Airedale, Unit 24, Dalton Mills,Dalton Lane, Keighley, BD21 4JH, UK

The first free Jam organised by HashBang Studio, to introduce the Raspberry Pi and bring people together - everybody welcome! Visit http://www.hashbangstudio.com for more information.

Pi and More 5 Raspberry Jam

When: Saturday 28th June 2014 from 8.00am CEST
Where: Universität Trier, Campus II, Hörsaalzentrum, 21 Behringstraße, 54296 Trier, Deutschland

There will be talks, workshops, and projects using the
Raspberry Pi or other embedded systems. http://www.piandmore.de

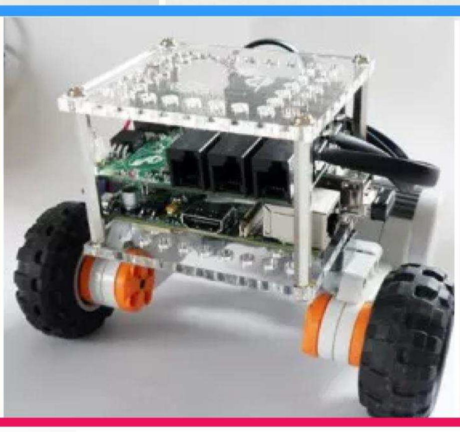
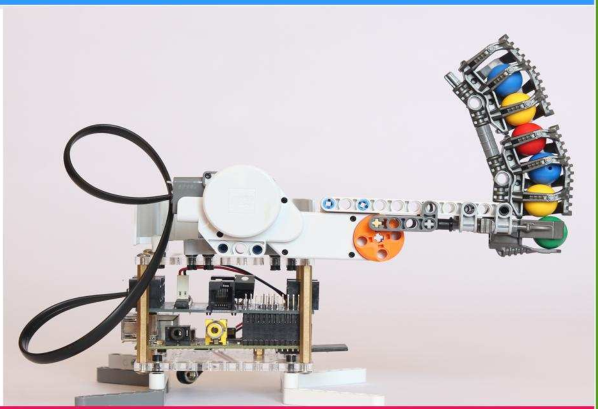



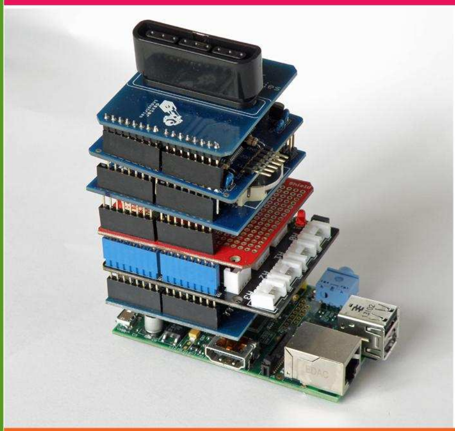
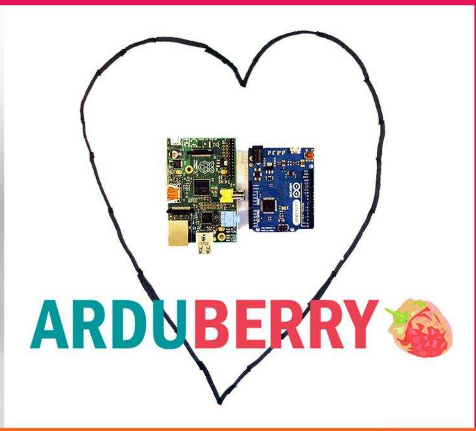
GrovePi

Connect Hundreds of Sensors to your Raspberry Pi

BrickPi

Turn your Raspberry Pi into a LEGO© Robot

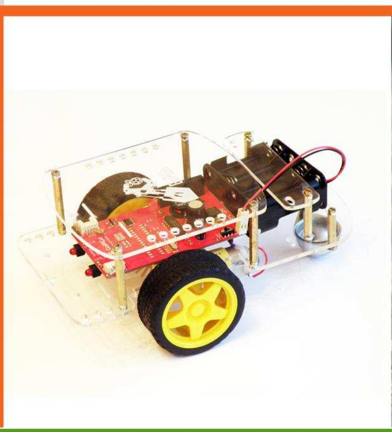
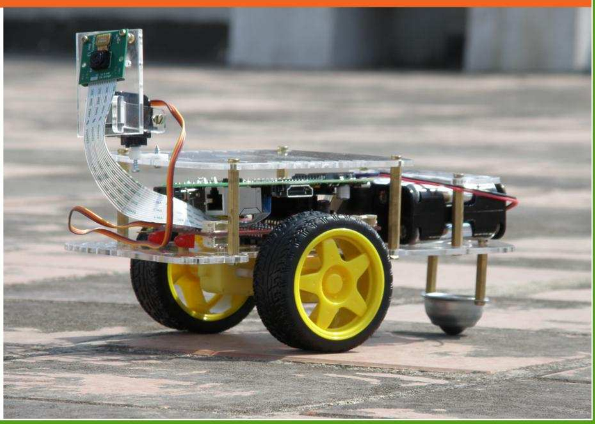



Arduberry

Unite the Raspberry Pi and Arduino

GoPiGo

Turn Your Raspberry Pi into a Robot

dexterindustries.com

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

Feedback & Question Time

The MagPi was recently at PiCymru Raspberry Jam in Cardiff where we received a very warm welcome. It was really encouraging to see all of the great things people of all ages are doing with their Raspberry Pi's. Here's some feedback we received:

The MagPi has given us some great ideas to try out at school.
A Teacher

A robot arm... that grabs sweets. Brilliant!
Anon

Is The MagPi suitable for young children? [Ed: Yes, absolutely - we cover a wide range of topics of varying complexity and it is a great project resource to work through with your kids].
Anon

Thanks for Volumes 1 and 2. I'm looking forward to Volume 3 in print as well.
Anon

Our recent Kickstarter campaign for Volume 2 also generated some great feedback:

I really love The MagPi and hope that I get the skills and find the time to build something amazing that can be published as well.
Henner

Keep up the good work, and make sure you keep the simple type in code each issue.
Allan

Can't wait to get my copy. Printed versions are so much easier to use.
Roger

I buy this for my son...he loves his RasPi and *really* loves your publication!
Daniel

Love the mag - long may it continue!
Paul

I like that you have a wide range of technical-ness in the articles
Rebecca

This is the first time I'm going to read The MagPi. I'm hoping to find lots of cool stuff and learn a lot from it.
Victor

I really enjoy the magazine with my 10 year old daughter!
David

Loving The MagPi magazine. Is it possible to submit articles to The MagPi to be considered for publication?
Darren

If you are interested in writing for The MagPi, would like to request an article or would like to join the team involved in the production of the magazine, then please get in touch by emailing the editor at:

editor@themagpi.com