Data Acquisition and Handling (DAH)

Franz Muheim (Course organiser) <f.muheim@ed.ac.uk>
Ben Wynne <b.m.wynne@ed.ac.uk>
Giorgos Sidiropoulos <g.sidiropoulos@ed.ac.uk>

Senior Honours
Semester 1, 2017/18

Edinburgh, September 7th, 2017

Chapter 1

DAH: Course Overview

1.1 Introduction

Data Acquisition and Handling (DAH) is a Senior Honours course, which was introduced in 2014 during a review of whole degree programme. DAH will introduce you to methods and tools of modern Data Acquisition and Handling, including Analog and Digital electronics, reading out sensors (detectors), handling and interpreting data. This course replaces JH Electronics Methods. Note that some parts of Electronics Methods (digital and analog electronics) are now taught in 2nd year Practical Physics. The DAH course will focus on data acquisition and data analysis.

1.2 Schedule

In week 1 (Tuesday 19 and Thursday 21 September 2017) there will be lectures introducing the DAH course. Laboratory work commences in week 1 (Tuesday 19 September 2017) and finishes at the end of week 11 (Thursday 30 November 2017). The laboratory sessions will take place on Tuesday and Thursday afternoons from 14:00 to 17:00 and you need to attend one of these afternoons. Please sign up for one of the afternoons using the online sign-up tool. The laboratory sessions will be held in JCMB 3301.

- Tuesday 19 September, 2 to 3 pm, JCMB Lecture Theatre LTC:
  Introduction to course and Lecture 1.
- Thursday 21 September, 2 to 3 pm, room JCMB Lecture Theatre LTC:
  Lecture 2.
- week 1, Tuesday 19 or Thursday 21 September, 3 to 5 pm, JCMB room 3301:
  laboratory work.
- weeks 2 to 11: Tuesday or Thursday 2 to 5 pm, JCMB room 3301:
  laboratory work.

1.3 Syllabus

The outline syllabus is as follows:

- Analogue signal processing. Treatment of noise. Filtering. Buffering using sample and hold;
- Analogue to digital conversion. Sampling rates. Characteristics & errors;
- Digital to analogue signal conversion;
- Communication protocols (Bus standards). Input/Output (I/O);
- Digital signal processing. Triggering. Fourier transforms;
- Data acquisition using a Raspberry Pi and Arduino and python;
- Advanced data analysis. Multi-parameter likelihood fits;
- Practical examples, e.g. temperature sensors, ultrasound sensors, FFT spectrum analyser, synthesizer, digital signal generators, motion sensors, remote sensing, image processing, with CCDs.

1.4 Learning Outcome

On completion of this course, the student will be able to:

1. Understand core concepts of data acquisition, data handling and data analysis in physical sciences;
2. Apply standard practical laboratory techniques (e.g. routine handling of data acquisition equipment and writing short, procedural computer programs) as directed in a script to achieve a stated goal;
3. Apply advanced practical laboratory techniques (e.g. handling of complex data acquisition equipment, and writing data acquisition computer programs) with limited direction to achieve a stated or open-ended goal;
4. Apply advanced data handling and data analysis techniques (e.g., data selection and representation, multi-parameter likelihood fits and writing data analysis computer programs) with limited direction to achieve a stated or open-ended goal;
5. Present a record of an experiment or computation in an appropriate, clear and logical written form (e.g. laboratory notebook, laboratory report, fully documented computer code), augmented with figures graphs, audio or movies where appropriate.

1.5 Laboratory work

The laboratory sessions of the DAH course will take place in JCMB 3301 on Tuesday and Thursday afternoons from 14:00 to 17:00. You will need to sign up for one of the afternoons using the online sign-up tool, linked to Learn. In the laboratory you will work in pairs, so if you have a partner you will need to sign-up for the same afternoon.

The laboratory work consists of checkpoints and projects, which are described in detail in Chapters 2 and 3. You will work with a Raspberry-Pi, which is a credit-card sized computer that plugs into a computer/TV screen and a keyboard. A manual will be provided. To control the Raspberry-Pi you will write python code. Example python scripts and code snippets will be provided on github, see `https://github.com/fmuheim/DAH`.

During weeks 1 to 6 of the DAH course you will need to complete six checkpoints. In each checkpoint you will learn a specific aspect of data acquisition or data handling and complete a prescribed number of tasks. You will work in pairs during the checkpoints. Each pair will have their own set of kit, however the Raspberry-Pi's will be shared between the Tuesday and Thursday afternoon sessions. Each pair will be given a yellow box with the required kit in which you can preserve your work for use in the following week. The boxes should be labelled with your names.

You should maintain a clear record of your work in a laboratory notebook as you work through the checkpoints. This must include diagrams of the circuits used. Python code written on the Raspberry-Pi must include explanatory comments and at each check point you need to demonstrate that your code compiles and runs correctly. Each partner will need to maintain their own notebook to demonstrate that a checkpoint has been completed.

In weeks 7 to 11 of the DAH course, you will carry out a project during the laboratory sessions. You will continue to attend during the same afternoon as for the checkpoints. The projects will build upon what you have learned during the checkpoints, but you will also encounter new material. While the checkpoints concentrated on specific data acquisition techniques and data handling methods the projects will allow you to progress towards building a small DAQ and/or data analysis system. The projects will have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

You will be able to choose from a list of projects, but due to the availability of equipment, the number of spaces for each project will be limited. A signup sheet will be provided.

The equipment specific to each DAH project will be available in red boxes. Some of the parts, including Arduinos and loudspeakers, as well as the Raspberry-Pi's, will be shared between Tuesday and Thursday afternoon sessions. In addition, each pair will continue to use the yellow box in which you can preserve your work for use in the following week.

For the DAH project you will continue to work in pairs. Throughout the project, each of you should maintain a clear record of your work in your laboratory notebook. As an example, diagrams of built circuits must be included. Python code written on the Raspberry-Pi must include explanatory comments. Each partner will be required to submit an individual report for the project.

1.6 Assessment

Data Acquisition and Handling is a continuously assessed course. The overall DAH assessment will be made from three parts. The sum of the marks achieved while carrying out the checkpoints will count for 30% of the total course mark. The marks obtained for the DAH project will count for 60% of the total course mark. Finally, there will be a quiz/hand-in which will count for 10% of the total course.

1.6.1 Assessment of checkpoints

There are six checkpoints and checkpoints 2 to 6 will be assessed during the laboratory hours. While working in pairs, each student will be assessed. The marks awarded by one of the demonstrators need not to be the same for both students. The assessment will be performed when you decide that you have completed the tasks for a checkpoint or parts thereof as allowed by the marking scheme. For each checkpoint a total score of between 8 and 10 marks will be awarded. You will only be awarded marks if you can demonstrate that the relevant circuit functions correctly and that your python code achieves the requested results. You will need to take care that you can demonstrate all parts of a checkpoint and not just the last sub-point, i.e. you are required to have separate codes for each task. In total up to 45 marks will be available for the checkpoints. You should be able to complete each checkpoint should in one afternoon. The deadlines for assessing checkpoints are as follows.

- The deadline for checkpoint 2 is week 3, for checkpoint 3 it is week 4, for checkpoint 4 it is week 5, for checkpoint 5 it is week 6 and for checkpoint 6 it is week 7.

The overall laboratory assessment will be made from the sum of marks of the check points, which constitutes 30% of the total course mark.

1.6.2 Project assessment

Your DAH project will be assessed through the submitted material. This includes your project report and your DAH software (e.g. Python scripts) and, if deemed useful, supplementary material. Guidance on how to prepare these items is given below. The project will be marked according to the University Common Marking Scheme.

Report Preparation

For how to write a proper report we refer you to the workshop slides on report writing in the Senior Honours (SH) Projects course, which are available at `https://www.wiki.ed.ac.uk/display/SP/SH+Projects`. The basic layout of a DAH report will be similar to an SH project report with the main difference being that a good DAH report will be shorter and should typically be approximately 7 pages long. It is expected that the report is typed. Most students use LaTeX or Word, either is fine.

When planning and writing a report, you need to be selective about what to include in your report, it should be a concise technical document. However, it also needs to contain all the information required for the reader to understand what was achieved, i.e. with your report you need to be able to demonstrate to what extend the project was carried out successfully. It is often useful to include circuit diagrams, pictures of the setup, or plots of measurements. A good report would allow a fellow student to be able to reproduce your work. Students are advised to start writing their report as the project progresses. Experience shows that report writing usually will take longer than anticipated.

Each partner is required to submit an individual report for the project. If you are working with a partner, this report must make it clear which parts of the project were carried out together, which parts are only your work, and which parts were only carried out by your partner. The report must contain a signed declaration, which will be available on Learn.

Programming Code: e.g. Python

Programming code for the DAH project, e.g. using Python, written on the Raspberry-Pi or on another computer, must include explanatory comments. When reading a (python) script, a reader should easily be able to understand what the script will do. All code written (in python or another programming language) for the DAH project will need to be submitted using the "Assessment" tool on Learn. The files should be bundled up in a .zip or .tar file. A README file should be included. Submission details will be provided.

Supplementary Material

You are encouraged to submit supplementary material if you consider the material as a part of the project that does not fit into the report format. This could include your laboratory notebook, output files produced by running a python script, short videos on a USB drive or a link to a webpage. If you have questions about the suitability of material, please consult with the Course Organiser. All such supplementary material must be clearly listed in an appendix to the report and referred to in the main text.

Submission Deadline:

The assessed material for the DAH projects will need to be submitted by **12.00 NOON on Friday, 1st December 2017**. By the deadline you must have submitted

- an electronic version of your project report to Turnitin via Learn; and have handed in the following to the Teaching Office in JCMB (Room 4315):
- a signed "Own Work Declaration" form;
- a hardcopy of your project report;
- any supplementary material.

The marks obtained for the DAH project will count for 60% of the total course mark. Reports submitted after the deadline will receive a penalty of 5% (equivalent to 3 marks out of 60) for each calendar day by which the deadline is exceeded. Students who, for good reason, find they are unable to meet the deadline, should contact the DAH Project Organiser and Course Secretary before the deadline.

1.6.3 Quiz

Midway during the course, you will be need to submit a quiz/hand-in on questions about data acquisition and handling material. You will be given two weeks to solve these questions on your own time, i.e. you should not use laboratory hours to solve the quiz questions. The exact deadline for handing in the quiz will be announced on Learn, it will be around the end of week 7 of the semester. The quiz will count for 10% of the total course mark.

1.7 Plagiarism:

The University regulations on plagiarism apply, see Section 27 of the Taught Assessment Regulations, available online at `http://www.ed.ac.uk/schools-departments/academic-services/policies-regulations/regulations/assessment`.

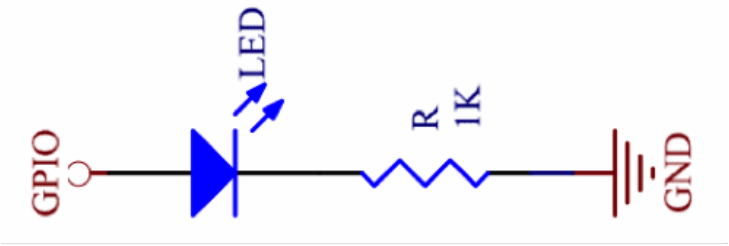
Chapter 2

DAH Checkpoints

Important note: Please consult the DAH manual to familiarise yourself with the equipment, including the Raspberry-Pi, LEDs, temperature sensors, ADCs, DACs, I/O, switches, breadboard and connectors. The manual contains detailed instructions on how to operate the Raspberry-Pi. It is suggested to use the chromium web browser. To copy python code snippets (see below) into your python scripts, download these files from github, see `https://github.com/fmuheim/DAH`. Data sheets for all electronic elements are available from Dropbox, see: `https://www.dropbox.com/sh/gfnisnh4ntnum1d/AAAWtNL_AhcpR8PZ_QmqZpsja?n=112609310`. The DAH manual also provides information on how to start and stop the webIOPi web server.

2.1 Checkpoint 1: LEDs and Switches

1.1. Control an LED with the Raspberry-Pi by completing the following steps. Connect the Raspberry-Pi to a breadboard and start the webIOPi web server. Start the chromium web browser and go to the webIOPi header webpage. Set GPIO 24 to OUT, connect this output to an LED with a 1 kOhm resistor in series to ground, and switch the LED on and off using the web interface. Repeat the exercise with negative logic (active low) by connecting the LED with a 1 kOhm resistor in series to 3.3 V. Draw a schematic diagram for this circuit, see example below.



1.2. Using positive logic (active high) connect a push-button Switch between 3.3V and GPIO23 and, with a 1kOhm resistor in series, to ground and explain what happens on the webIOPI header webpage. Draw a schematic diagram for this circuit. Afterwards stop the webIOPi web server.

1.3. Connect an LED with a 1 kOhm resistor as in checkpoint 1.1 above. Start python interactively to set GPIO 24. Using the code templates, import the webiopi framework, make a GPIO object and turn the LED on and off.

```
studentnn@dahpimm  ~ $ sudo python3
```

```
# Import webIOPi
import webiopi

# Make a GPIO object
GPIO = webiopi.GPIO

# Set which GPIO pin is connected to the LED
LED0 = 24

# Setup GPIOs
GPIO.setFunction(LED0, GPIO.OUT) # Set Pin as output
GPIO.digitalWrite(LED0, GPIO.HIGH) # Turn on the LED
GPIO.digitalWrite(LED0, GPIO.LOW) # Turn off the LED
```

../scripts/checkpoint_1a.py

Write the first Python script to blink an LED for either positive or negative logic by adding a "while loop".

```
# Loop for ever
while True:

# Toggle LED after time in seconds as defined inside sleep method
    value = not GPIO.digitalRead(LED0)
    GPIO.digitalWrite(LED0, value)
    webiopi.sleep(2)
```

../scripts/checkpoint_1b.py

1.4. Connect the push button switch using using negative logic (active low) and the LED circuit with positive logic (active high). Write a python script to toggle the LED status every time the push button switch is pushed.

```
# Imports
import webiopi

# Set which GPIO pin is connected to the switch
#    Insert your code here

# Read the switch and if it is pressed toggle the state of the LED
if (GPIO.digitalRead(SWITCH0) == GPIO.LOW):

# Read the value of the LED, invert it and save it to variable value
# Write variable value to the LED
#    Insert your code here
```

../scripts/checkpoint_1c.py

2.2 Checkpoint 2: ADC, DAC and SPI BUS

Most experimental observables are continuous: their values can vary by arbitrarily small amounts. However, we record measurements as discrete values: a number with some range of uncertainty. Creating a numerical (digital) measurement from a continuous (analogue) signal is called digitisation, and is performed by an Analogue to Digital Converter (ADC). The digital information can then be analysed with a computer.

In this checkpoint we will use an ADC to read information from a light sensor into the Raspberry Pi. We will also perform the opposite task, varying the brightness of an LED by converting a numerical output from the Raspberry Pi into the corresponding voltage level using a Digital to Analogue Converter (DAC).

2.1. Connect an ADC MCP3208 chip to the Raspberry Pi using the SPI Interface on pin GPIO 8 (SPI_CE0). Make sure that all required connections between the MCP3208 and the Raspberry-Pi are made (see pin-out sheet). Use a multimeter to check that power (VDD) and ground (AGND and DGND) are correctly connected. Explain what all connections are for.

Connect a Light Dependent Resistor (LDR) and a 4.7 kΩ resistor as a voltage divider between 3.3 and 0 V, using an ADC input channel to measure the voltage in the middle. Use python interactively to read the voltages of all eight ADC channels. Try reading a specific ADC channel, and experiment with all the other methods given below. You are encouraged to consult the following webpage `http://webiopi.trouch.com/`(→ Tutorials → Using Devices → Analogue).

Explain how the ADC works and what the meaning of the return values of each method is. What is the primary ADC output and how is the voltage output calculated from this? Cover the LDR with your hand, and explain how the ADC readings change.

```
# Import ADC chip libraries
from webiopi.devices.analog.mcp3x0x import MCP3208

# Define ADC on Chip Enable 0 (CE0/GPIO8)
ADC0 = MCP3208(chip=0)

# Read all ADC channels in Volts.
print ( ADC0.analogReadAllVolt() )

# Play with the following methods
ADC0.analogCount()
ADC0.analogResolution()
ADC0.analogMaximum()
ADC0.analogReference()
ADC0.analogRead(channel)
ADC0.analogReadFloat(channel)
ADC0.analogReadVolt(channel)
ADC0.analogReadAll()
ADC0.analogReadAllFloat()
ADC0.analogReadAllVolt()
```

../scripts/checkpoint_2a.py

[3 marks]

2.2. Leave the ADC in place, but also connect the DAC MCP4922 chip to the Raspberry-Pi using the other SPI Interface on GPIO 7 (SPI_CE1). Make sure that all required connections between the MCP4922 and the Raspberry-Pi are made (see pin-out sheet). Use a multimeter to check that power (VDD) and ground (AVSS) are correctly connected. Explain how the DAC works and what all connections are for.

Use python interactively to set a value — e.g. 1.3 V — to output VOUTA of the DAC. Measure this voltage with a multimeter.

```
# Import DAC chip libraries
from webiopi.devices.analog.mcp492X import MCP492X

# Define DAC on Chip Enable 1 (CE1/GPIO7)
DAC1 = MCP492X(chip=1, channelCount=2, vref=3.3)

# Output 1.3V on channel 0 of DAC1
print ('output 1.3V on channel 0 of DAC1')
DAC1.analogWriteVolt(0, 1.3)

# Print value of channel 0 of DAC1
print ('value of register for channel 0 of DAC1')
print (DAC1.analogReadVolt(0))
```

../scripts/checkpoint_2b.py

[2 marks]

2.3. Connect one output of the DAC to an LED. Write a python script that varies the brightness of the LED by setting a series of different values for the output voltage of the DAC. Now arrange the circuit so that the LED is next to the LDR, and the change in brightness can be measured. The laboratory is quite bright relative to an LED, so you might need to cover the breadboard to show a convincing change. Modify your script to read the ADC input each time you set the DAC output. Write the DAC setting and measured ADC values at each step to an output file with comments such that the content of the file will explain your work.

[4 marks]

2.3 Checkpoint 3: Generating and Sampling Analogue Signals

3.1. Connect an ADC chip (MCP3208) to the Raspberry Pi, as in Checkpoint 2. Verify that the ADC works with a DC voltage produced with a potentiometer or an LDR in series with a 4.7 kΩ resistor

Use the signal generator to produce a repetitive signal, e.g. a sinusoidal waveform. Display the output on the oscilloscope. Set the amplitude of the signal such that the waveform can be read by the ADC chip, which can sample between 0 V and VREF = 3.3 V. Set the frequency to 10 Hz.

Connect the output of the signal generator to an ADC input channel. Using python in interactive mode read a few samples of the ADC output. Comment on what you measure. [Caveat: Don't connect a signal with a voltage outside the range of the ADC chip, which could destroy it and the Raspberry Pi.]

[1 marks]

3.2. Measure the waveform produced by the signal generator by writing a python script that records 100 ADC samples and displays these on a graph. Always label plots correctly with title and axes and save these to a file (pdf format). [Hint: Use the pylab interface for plotting graphs. Example files are available on github.]

[2 marks]

3.3. Calibrate the voltage scale of the ADC output with respect to the voltage scale displayed on the oscilloscope by using a square waveform that closely matches the ADC input range. First connect the signal from the signal generator to the oscilloscope. Read the input voltage for the high and low sections of the square waveform off the oscilloscope screen. For this use the trigger threshold dial to determine these voltage levels as precisely as possible. Then connect the signal to an ADC input channel. Write a python script that takes 100 ADC samples and writes these into a file, then expand the script to determine the average ADC voltages for the high and low sections of the square waveform and record the results. Reduce the amplitude of the input waveform by a factor of two and repeat above procedure. Plot the four calibration measurements, i.e. the measured ADC voltages (from the two sets of measurements of the high and low sections) versus the four input voltages (measured with the oscilloscope) on a graph and comment.

[2 marks]

3.4. What is the maximum signal frequency with which you can properly record a given repetitive signal? First, consider which would be the best waveform for this investigation. What is the sampling frequency? Explain what happens when a signal is undersampled. Make a plot with a waveform that is undersampled.

[2 marks]

3.5. Connect a DAC chip (MCP4922) to the Raspberry Pi, as in Checkpoint 2. Using a python script, generate a sinusoidal waveform on the DAC output and plot the waveform to a graph.

Use the ADC chip (MCP3208) to digitise the waveform generated by the DAC. Write a python script that takes 100 DAC samples and 100 ADC measurements and plot these on the same graph. Take into account the limitations encountered in 3.4.

[3 marks]

2.4 Checkpoint 4: Input/Output I/O and I2C BUS

4.1. Input/Output (I/O) Expander chips enable the user to connect many devices having the same or similar functions. With the Raspberry Pi this can be achieved using the I2C bus. Connect the PCF8574AN chip (I2C BUS Expander) to the Raspberry-Pi. Make sure that all required connections between the PCF8574AN chip and the Raspberry-Pi are made, see pin-out sheet. Explain how the I/O Expander works and what the SDA, SCL and A0, A1, A2 address lines are.

Using negative logic connect an LED to output P0 of the PCF8574AN Expander chip. Write a python script to blink the LED using negative logic, see checkpoint 1 for setting up a while loop. Why is negative logic necessary?

You may consult the webIOPi webpage `http://webiopi.trouch.com/` (→ Tutorials → Using Devices → Digital) to find information on the GPIO expander chip.

```
# Imports
import webiopi
from webiopi.devices.digital.pcf8574 import PCF8574A
mcp = PCF8574A(slave=0x38)

# Retrieve GPIO lib
GPIO = webiopi.GPIO

LED0 = 0 # Set which PCF8574 GPIO pin is connected to the LED (negative
    logic)

# Setup GPIOs
mcp.setFunction(LED0, GPIO.OUT) #Set Pin as output

# Turn on the LED for the first time
mcp.digitalWrite(LED0, GPIO.LOW)

# Loop for ever
#    Insert your code here
# Include a delay
webiopi.sleep(0.10)
```

../scripts/checkpoint_4a.py

[3 marks]

4.2. Connect an additional 3 LEDs to outputs P1, P2 and P3 of the PCF8574AN Expander chip. Write a python script which turns the LEDs on and off in a predefined pattern, such as a running light. Consider P0 to P3 as default, but you are encouraged to play with other patterns.

[2 marks]

4.3. Consult the webpage `http://webiopi.trouch.com/` (→ Tutorials → Using Devices → Digital) for the GPIO Expander. Use the portWrite(value) method to manipulate all four LEDs at the same time. Connect a push button switch to pin P4 of the expander chip. Write a python script such that an LED pattern toggles every time the button is pushed.

```
# Loop for ever
while True:
    # Read the switch and if it is pressed toggle the state of the LED
    if (mcp.digitalRead(SWITCH0) == GPIO.LOW):
        # Insert your code here

        # dummy write to reset switch register
        mcp.digitalWrite(SWITCH0, GPIO.HIGH)
```

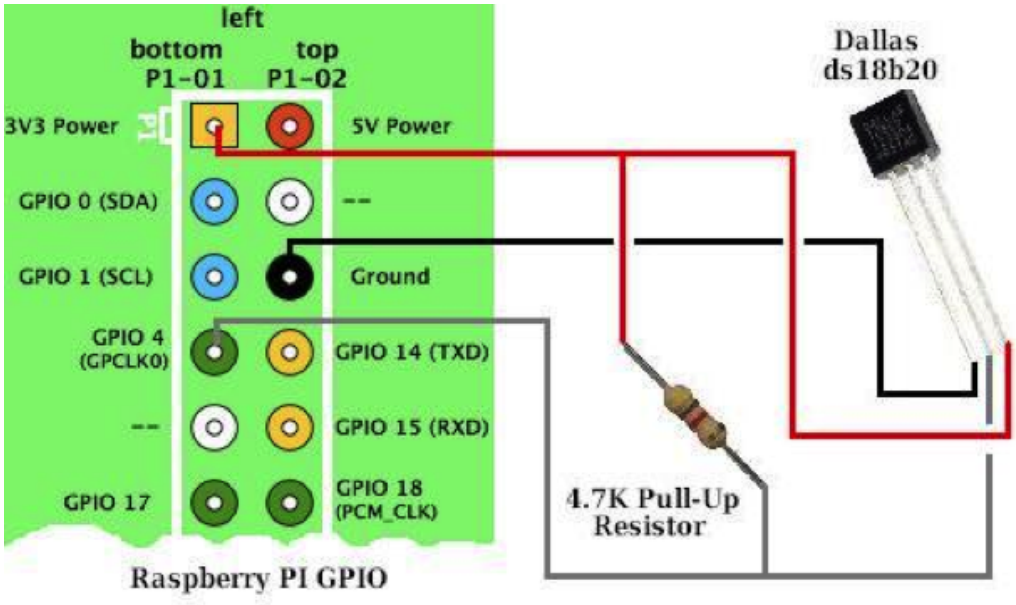
../scripts/checkpoint_4b.py

[3 marks]

2.5 Checkpoint 5: Temperature Sensors

5.1. We will be using DS18B20 temperature sensors for this checkpoint. Take a look at the datasheet here: http://www.adafruit.com/datasheets/DS18B20.pdf or download it from the DAH Dropbox.

Connect a DS182B0 temperature sensor to your Raspberry pi (look at the flat front of the sensor to get it the right way around):



What is the interface between the DS18B20 temperature sensor and the Raspberry Pi? Explain how it works.

Locate the sensor output by finding the file that has the serial number of your sensor:

```
studentnn@dahpimm ~ $  cd /sys/bus/w1/devices
studentnn@dahpimm /sys/bus/w1/devices $ ls
10-00080265b6d6 w1_bus_master1
```

where n = 1 to 50 and mm = 01 to 22. Note that your temperature sensor won't be called 10-00080265b6d6, this is just an example.

Now read the sensor output, i.e. the raw temperature measurement:

```
studentnn@dahpimm /sys/bus/w1/devices ~ $ cd 10-00080265b6d6
studentnn@dahpimm /sys/bus/w1/devices/10-00080265b6d6 $ cat w1_slave
30 00 4b 46 ff ff 0d 10 29 : crc=29 YES
30 00 4b 46 ff ff 0d 10 29 t=23937
```

This should be interpreted as 23.937 centigrade (degree Celsius).

WebIOPi provides a simple way to access the temperature sensor data in python. It is best to test this by running python in interactive mode first.

```
studentnn@dahpimm ~ $ python3
```

```
# Import
from webiopi.devices.sensor.onewiretemp import DS18S20

# Readout temperature sensor
tmp0 = DS18S20(slave="10-00080265b6d6")
tmp0.getCelsius()
```

../scripts/checkpoint_5a.py

[2 marks]

5.2. Measure temperature with the DS18B20 sensor versus time. Choose a sensible time interval. Write a python script to make a graph of 50 temperature measurements versus time. Always label plots correctly with title and axes and save these to a file.

As a second step the graph should update itself as each temperature measurement is made. Write a python script for this purpose. Once this is working, play with it by touching the temperature sensor with you fingers. Describe what is happening and sketch it in your lab book.

The following code example shows how to display a plot that updates regularly:

```
import pylab
import matplotlib.animation as animation
import datetime

# Empty arrays of time and measurement values to plot
timeValues = [ ]
measurements = [ ]

# Set up the plot object
plotFigure = pylab.figure()

# The function to call each time the plot is updated
def updatePlot( i ):

    timeValues.append( datetime.datetime.now() ) # Store the current time
    measurements.append( MEASUREMENT )           # Store the measurement
    plotFigure.clear()                           # Clear the old plot
    pylab.plot( timeValues, measurements )       # Make the new plot


# Make the animated plot
ani = animation.FuncAnimation( plotFigure, updatePlot, interval=1000 )
pylab.show()
```

../scripts/checkpoint_5b.py

[3 marks]

5.3. Add another temperature sensor to your circuit by connecting it in parallel with the existing one. You don't need to make separate connections to the Raspberry Pi: your new sensor can share these with the existing one (Just make sure that you connect it the right way around).

Find its serial number in the w1/devices folder like before. Now ensure that you can read out your two temperature sensors simultaneously in python. You can test this by running python in interactive mode first. Note that your temperature sensors will have different serial numbers.

```
studentnn@dahpimm  ~ $ python3
```

```
# Import
from webiopi.devices.sensor.onewiretemp import DS18S20

# Readout temperature sensor
tmp0 = DS18S20(slave="10-00080265b6d6")
tmp1 = DS18S20(slave="10-000802cb3b5c")
print ( str( tmp0.getCelsius() ) + ", " + str( tmp1.getCelsius() ) )
```

../scripts/checkpoint_5c.py

What is the smallest change in temperature that a sensor can report? Explain this with reference to the datasheet, and how the temperature information is encoded.

Investigate the accuracy of the sensor readings by looking at the stability of the measurement with time, and by comparing the outputs of your two sensors. You can probably assume that the ambient temperature in the lab is constant, but shielding your sensors from breezes may help.

Modify your graphing code from part 5.2 to make histograms of the temperature measurements of the two sensors.

You can use the pylab histogram command:

```
# Pylab makes graph plotting very easy:
import pylab

# Make a histogram with NumberOfBins bins
# in the range binMinimum to binMaximum
pylab.hist( someData, bins=NumberOfBins, range=[binMinimum, binMaximum] )
```

../scripts/checkpoint_5d.py

Make also a graph of the difference between the temperature of the two sensors. Determine the RMS value of a set of temperature difference measurements between the two sensors. Explain your results. Are these consistent with the datasheet?

[3 marks]

2.6 Checkpoint 6: Data Handling and Analysis

6.1. This is a data handling exercise and the Raspberry Pi is not required. Thus this checkpoint is best carried out using the Physics CPlab computers. There is a CPlab computer available on every desk in the DAH laboratory. You can use the following python libraries.

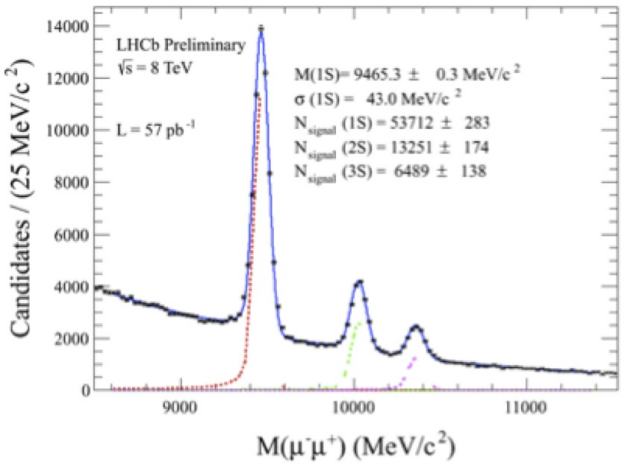
```
# Import
# pylab has a LOT of useful things in it.
import pylab

# numpy is the fundamental package for scientific computing with Python.
import numpy as np

# Make a histogram with arrays of nr. of entries and of bin edges
entries, binedges, patches = pylab.hist(xmass, bins = Nbins, range = [
    binMin,binMax])
```

../scripts/checkpoint_6a.py

The LHCb experiment at the Large Hadron Collider at CERN has recorded a sample of muon pairs with invariant masses in the range of 8.5 to 11 GeV/c^2. Three clear peaks are observed in this mass spectrum. These correspond to the production of Upsilon mesons, which are bound states of a b and a anti-b quark. These states are known as the $\Upsilon(1S)$, $\Upsilon(2S)$ and $\Upsilon(3S)$ mesons where the $\Upsilon(1S)$ meson is the ground state and the $\Upsilon(2S)$ and $\Upsilon(3S)$ states are radial excitations (for LHCb paper, see DOI: 10.1007/JHEP06(2013)064).



Download the file upsilons-mass-xaa.txt from the DAH Dropbox, which contains the invariant masses of a large number of muon pairs in units of GeV/c^2 in text format. Write a python script that reads the data from this file and plots a histogram of all the masses, choosing a reasonable bin width. Always label plots correctly with title and axes and save these to a file. [Hint: The bin width should be chosen such that each of the three peaks is clearly resolved and represented by a sufficient number of bins for analysis.]

[2 marks]

6.2. Determine the masses of the three particles by determining the bins with the highest number of entries in the peak regions. Divide the histogram into three peak regions and write a local peak finding method for this part. What are the mass differences between the Υ(2S) and Υ(3S) states with respect to the Υ(1S) meson?

[2 marks]

6.3. Determine the mass of the Υ(1S) meson and its statistical uncertainty. This can be achieved by several methods. First by looking at the mass spectrum, choose a suitable region around the Υ(1S) mass peak. Calculate the mean, the unbiased variance and standard deviation for the events in this region. Use these values to determine the standard deviation of the mean. Comment on possible biases for this method.

The mass peaks corresponding to the three Υ mesons can be described reasonably well by a Gaussian function, $f(x) = \frac{N}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-/\mu)^2}{2\sigma^2}\right)$ where x is the invariant mass of the muon pairs, μ is the mass of the Υ(1S) meson, σ is the Gaussian width (mass resolution) and N is the total number of signal events. By inspecting visually the muon-pair mass spectrum, determine the Full Width Half Maximum (FWHM) of the Υ(1S) mass peak. Assuming a Gaussian signal shape estimate the mass resolution σ from the FWHM. Compare this result for the mass resolution of the Υ(1S) peak with the standard deviation in the signal region determined above and comment.

[3 marks]

In the muon-pair mass spectrum define a signal region of width ± 150 MeV/c^2 around the Υ(1S) peak position and determine the number of events N in this region. Define an upper and lower sideband region where there are only background events. These sidebands should each be half as wide as the signal region and located at masses equidistant from the Υ(1S) peak position. Assuming that the background is falling linearly with the muon-pair mass, determine the number of background events B in the signal region (below the Υ(1S) mass peak). Perform either a linear least squares fit in the sideband regions or use the sideband subtraction method for this. Determine the number of signal events S in the signal region.

Alternatively, if you know how to perform a fit you may choose fitting the Υ(1S) peak in the mass spectrum for this part.

Compare your mass measurement with the Particle Data Group (`pdg.lbl.gov`) and comment. [Hint: The PDG lists the properties of particles. Select "pdgLive - Interactive Listings" followed by "Mesons b anti-b" to find the Υ(1S) meson.]

[3 marks]