Chapter 3

DAH Projects

3.1 Project B: Building an FFT Spectrum Analyser

3.1.1 Goals of project

You will develop and build a real-time spectrum analyser. This device will sample an analog signal and perform a Fast Fourier Transform (FFT) analysis and the results can be displayed and/or written to a file. The ADC MC3208 chip, which was used during the checkpoints, could be a part of this project.

The projects have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

3.1.2 Equipment for project B

For this project you will need the following items:

- Raspberry-Pi
- Signal generator
- Analog to Digital Converter (MCP3208)
- Arduino Uno
- USB cable to connect Arduino to Raspberry-Pi.
- Microphone with preamplifier if you want to analyse acoustic signals.

Some of this equipment will be located in the red box B. You will have to share some items between Tuesday and Thursday sessions.

3.1.3 Building the FFT spectrum analyser

We suggest that you start by reading up on Fourier transformations, which you learned in a 3rd year course, and the Fast Fourier Transform as a particularly effective numerical implementation. You may also want to familiarise yourself with the Arduino micro controller, e.g. at `http://www.arduino.cc/`. It is suggested to develop and build the spectrum analyser in a modular way, e.g. to develop the FFT analysis separately from the real-time data acquisition, before combining these tasks.

There are many ways of completing this project. To give you a better idea of what may be required, take a look at the three examples below, listed in the order of increasing complexity.

1. Use the webIOPi framework. This is the simplest approach because it requires almost no new knowledge. You would use the signal generator to produce a waveform and connect the ADC (MC3208) chip to the Raspberry-PI using the SPI interface. Use the ADC chip (MCP3208) to sample the signal at regular intervals, fill an array of a fixed length (say, 1024 samples) and perform an FFT analysis of the data using a Numpy library (routine `numpy.fft.rfft()`). You would then save the absolute values of the FFT data to file and display these on a screen. This task would allow you to develop the FFT analysis and the displays. The amount of work for this is similar to a checkpoint.

   While such an FFT analyser is simple to build and control, it has severe limitations. webIOPi allows only for a very low sampling rate (as studied in Checkpoint 3). There can be missing parts in the recorded samples ("data holes") as the Raspberry-Pi is not a real-time computer. The Raspberry-Pi has a multitasking operating system, which can interrupt the data taking, especially if sampled at too large a rate.

2. Use the MCP3208 ADC chip, but control and read its output directly through the serial interface SPI using a python library called `spidev`, which is already installed on the Raspberry-Pi. The spidev library has a routine called `xfer()` that sends an array of bytes to the ADC through the SPI interface and receives ADC data in response to this request. Use `xfer()` to instruct the ADC to measure the voltage. To read the voltage from CH0 of the ADC one needs to send three bytes [6,0,0] which instructs the ADC to select channel CH0, measure the voltage and then send back another three bytes (ignore the first byte, the second and the third are the high and the low byte of the 12-bit integer output of the ADC) to the Raspberry-Pi.

```
# instruction for ADC to measure a voltage sample

import spidev
spi=spidev.SpiDev() # initialization
spi.open(0,0)       # opens the port
spi.max_speed_hz=1000000  # sets the clock speed to 1MHz = max

# recommended for VDD=3.3V
# to read ADC output
adc=spi.xfer2([6,0,0])  # read voltage from CH0 of MCP3208
# after executing this commabd, "adc" will be an array of three bytes,
# where the 2nd and 3rd will hold the output of the ADC
```

../scripts/project_B_a.py

This method is much faster than webIOPi and it is possible to record up to 5k samples per second, however real-time sampling is not possible and there can be holes in the data.

3. Use the Arduino, a real-time micro-controller, which is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals, The Arduino is based on an 8-bit 16MHz chip ATmega328 and does not have an operating system by default, so it supports real-time computing, i.e. the program does not get interrupted. The Arduino also has a built-in 10-bit ADC. For this project, the Arduino is used as an ADC and a data buffer connected to the Raspberry-Pi and it will be made available pre-programmed. From your perspective the Arduino will behave as "black box", i.e. an ADC with a buffer memory that can be accessed through an USB interface. (Note: If you are interested, you may familiarise yourself with Arduino-IDE package and upload the code, `ADCforRPI4.ino`, available on `https://github.com/fmuheim/DAH`. You will need to write python code that communicates with the Arduino using the `serial` library.

```
# open the serial port with baud rate 115200bps
import serial
ser=serial.Serial('/dev/ttyACM0',115200)

# send desired sampling frequency to Arduino and order it to begin
    sampling
ser.flushInput()  # clear the serial port buffer
ser.write(bytes([4,2]))  # the first byte can be anything between 4 and
    255.

# The larger the value the slower the sampling will be
# read data sample from Arduino ADC (1024 bytes, 8-bit resolution)
data=ser.read(1024)
```

../scripts/project_B_b.py

The 8-bit resolution simplifies data transfer and processing, since each sample equals exactly one byte of data. In fact, 8 bits correspond to the effective resolution of the ADC at the largest supported sampling frequencies where the two least significant bits (of the 10-bit ADC) are essentially noise, This method allows for fast sampling rates of up to 100k samples per second in real-time and there are no data holes.

3.1.4 Performing an FFT spectrum analysis

Performing an FFT analysis a data sample is straightforward, for example you can use a Numpy library routine, `numpy.fft.rfft()`. In all cases the data should be saved to a file and displayed on the screen. Once the FFT spectrum analyser is working and the main functionalities for saving and displaying the data spectrum is achieved, there are several possibilities to take this project forward in an open-ended way.

1. You will use FFT to analyse the spectrum of a several repetitive signal shapes, including a sinusoidal, a square and a saw waveform generated by the function generator, and

compare the amplitudes of different harmonics to mathematically predicted ones for that signal.

2. Investigate which of the many python libraries that are available for drawing on the screen best suits your needs. For example, mathplotlib can be employed to produce high-quality figures, but may be too slow to present data in real time. The `pygame` library supports drawing simple figures (points, lines, etc.) and is very fast, simple to use, and sufficient for making simple plots of the input signals and its FFT spectrum.
3. There are additional methods for generating signals. For example you can use a microphone to record acoustic signals and perform FFT analyses on these samples. Can you use it e.g. to determine the pitch of a whistled note?
4. The Arduino is used as a "black box" for this project. Learning how to program was deemed to be incompatible with the timescale of the project. However, if you have used the Arduino before or are sufficiently curious, you are welcome to investigate. There might be also other options for reading the data, and you are encouraged to investigate this.

3.1.5 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;
- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.

3.2 Project C: Building a Synthesizer

3.2.1 Goals of project

You will develop and build a small synthesizer with which you will be able to play small tunes. This project builds upon the work carried out in checkpoint 4 where you worked with an I/O expander chip. With the synthesizer you will be able to play simple tunes, but there is scope for developing the synthesizer much further.

The projects have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

3.2.2 Equipment for project C

For this project you will need the following items:

- Raspberry-Pi
- Amplified Speakers
- MCP23S17 IO Expender
- 1K Resistors
- Pushbuttons

Some of this equipment will be located in the red box C. You will have to share some items between Tuesday and Thursday sessions.

3.2.3 Building the synthesizer

The SPI I/O expander chip (MCP23S17) is supported by the webIOPi framework. In checkpoint Pi 3 you used an 8 bit I2C expander chip in a very similar way, so you can built upon this work. For this project we recommend that you use the MCP23S17 chip, which is a 16 bit SPI expander. While the two expander chips are different devices the same python methods can be applied when using webIOPi.

```
# syntax to import the MCP23S17 I/O expander chip.
# documentation on webiopi.trouch.com is not up to date.
from webiopi.devices.digital.mcp23XXX import MCP23S17
```

../scripts/project_C_c.py

1. Start your project using switch buttons to play sounds at first and then change to free wires forming touch sensors. The insulation from the end of each wire should be stripped. To be able to use the expander chip with wires as touch sensors one should utilise the positive logic (logic 1 means wire was touched).

2. You can base your project on the `Snack` audio package, which is already installed on the Raspberry-Pi. You can find a description on the following Webpage: `https://www.daniweb.com/software-development/python/code/216655/play-a-musical-note-python`. Here is an example code of how to use this package.

```
import tkinter  #This is for python3
import tkSnack


def setVolume(volume=50):
    """set the volume of the sound system"""

    if volume > 100:
        volume = 100
    elif volume < 0:
        volume = 0
    tkSnack.audio.play_gain(volume)


def playNote(freq, duration):
    """play a note of freq (hertz) for duration (seconds)"""

    snd = tkSnack.Sound()
    filt = tkSnack.Filter('generator', freq, 30000, 0.0, 'sine', int
   (11500*duration))
    snd.stop()
    snd.play(filter=filt, blocking=1)


def soundStop():
    """stop the sound the hard way"""

    try:
        root = root.destroy()
        filt = None
    except:
        pass


root = tkinter.Tk()   #This is for python3

# have to initialize the sound system, required!!
tkSnack.initializeSnack(root)
# set the volume of the sound system (0 to 100%)
setVolume(50)
# play a note of requency 440 hertz (A4) for a duration of 2 seconds
playNote(440, 2)
# optional
soundStop()

root.withdraw()
```

../scripts/project_C_a.py

Once you have set up the hardware, play a few notes and make sure you understand how the synthesizer works. In Appendix 1 of project C, shown below, you can find the frequencies for musical notes.

3.2.4 Applications using the synthesizer

1. You can play a tune and record it.
2. There are many ways to enrich your project, for example, try to implement what is called "polyphony" in the world of music synthesizers.

   Polyphony means that the synthesizer can play more than one note at the same time, i.e. this means that the synthesizer is able to detect two or more buttons being pressed at the same time. In reality the sound device (i.e. the sound chip of the Raspberry Pi) is producing multiple interleaving sounds, but our hearing sense perceives them as concurrently played sounds.
3. To play two or more sounds at the same time it is not recommended to use the `Snack` audio package. Instead you are encouraged to use pre-recorded audio samples from a piano in wav format.

   You can download a compressed file with the audio samples from the DAH DropBox page. To play an audio file (sampled note) you can use an application such as `aplay`. Below an example is given of how to play two sounds at the same time. Note the `-q` command to `aplay`, which stops it from producing unhelpful text output.

```
import os
while True:
    if ( switch1 == True  & switch2 == True ):
        os.system('aplay -q C5.wav &')
        os.system('aplay -q C5#.wav &')
```

../scripts/project_C_b.py

Capturing two or more buttons being pressed at the same time can be achieved by reading the expander as a parallel bus instead of reading the state of each push button switch in sequence. (Reading in sequence is commonly referred to as polling). When using parallel or port reading, one should save the state of the port to a variable, which can be called a "switch register". Playing the synthesizer will correspond to sweeping all bits of the switch register for finding pressed switches. After the sweep, one should set the state of the switch register to idle.

Can you find where the bottle-neck occurs in trying to achieve polyphony? Is the limitation due to hardware or software?

3.2.5 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;
- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.

Appendix 1 of Project C: Frequency Table for Notes

Frequency [Hz]	Rounded Frequency	Note	MIDI #
27.5	28	A0	21
29.1352	29	A#0	22
30.8677	31	B0	23
32.7032	33	C1	24
34.6478	35	C#1	25
36.7081	37	D1	26
38.8909	39	D#1	27
41.2034	41	E1	28
43.6535	44	F1	29
46.2493	46	F#1	30
48.9994	49	G1	31
51.9131	52	G#1	32
55	55	A1	33
58.2705	58	A#1	34
61.7354	62	B1	35
65.4064	65	C2	36
69.2957	69	C#2	37
73.4162	73	D2	38
77.7817	78	D#2	39
82.4069	82	E2	40
87.3071	87	F2	41
92.4986	92	F#2	42
97.9989	98	G2	43
103.8262	104	G#2	44
110	110	A2	45
116.5409	117	A#2	46
123.4708	123	B2	47
130.8128	131	C3	48
138.5913	139	C#3	49
146.8324	147	D3	50
155.5635	156	D#3	51
164.8138	165	E3	52
174.6141	175	F3	53
184.9972	185	F#3	54
195.9977	196	G3	55
207.6523	208	G#3	56
220	220	A3	57
233.0819	233	A#3	58
246.9417	247	B3	59
261.6256	262	C4	60
277.1826	277	C#4	61
293.6648	294	D4	62
311.127	311	D#4	63
329.6276	330	E4	64
349.2282	349	F4	65

Frequency [Hz]	Rounded Frequency	Note	MIDI #
329.6276	330	E4	64
349.2282	349	F4	65
369.9944	370	F#4	66
391.9954	392	G4	67
415.3047	415	G#4	68
440	440	A4	69
466.1638	466	A#4	70
493.8833	494	B4	71
523.2511	523	C5	72
554.3653	554	C#5	73
587.3295	587	D5	74
622.254	622	D#5	75
659.2551	659	E5	76
698.4565	698	F5	77
739.9888	740	F#5	78
783.9909	784	G5	79
830.6094	831	G#5	80
880	880	A5	81
932.3275	932	A#5	82
987.7666	988	B5	83
1046.5023	1047	C6	84
1108.7305	1109	C#6	85
1174.6591	1175	D6	86
1244.5079	1245	D#6	87
1318.5102	1319	E6	88
1396.9129	1397	F6	89
1479.9777	1480	F#6	90
1567.9817	1568	G6	91
1661.2188	1661	G#6	92
1760	1760	A6	93
1864.655	1865	A#6	94
1975.5332	1976	B6	95
2093.0045	2093	C7	96
2217.461	2217	C#7	97
2349.3181	2349	D7	98
2489.0159	2489	D#7	99
2637.0205	2637	E7	100
2793.8259	2794	F7	101
2959.9554	2960	F#7	102
3135.9635	3136	G7	103
3322.4376	3322	G#7	104
3520	3520	A7	105
3729.3101	3729	A#7	106
3951.0664	3951	B7	107
4186.009	4186	C8	108

3.3 Project D: Building an Ultrasonic Range Finder

3.3.1 Goals of project

You will develop and build an ultrasonic range finder. This device will use an ultrasound transducer as a distance sensor.

The projects have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

3.3.2 Equipment for project D

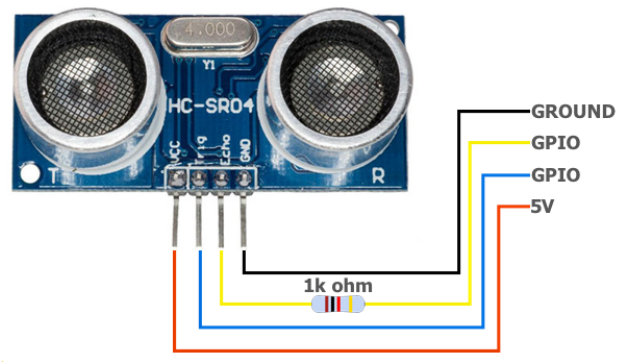
For this project you will need the following items:

- Raspberry-Pi
- Ultrasonic Transducer module HC-SR04
- 1 kΩ Resistor
- Loudspeakers

Some of this equipment will be located in the red box D. You will have to share some items between Tuesday and Thursday sessions.

3.3.3 Building the ultrasonic range finder

You will develop an ultrasonic range finder using a HC-SR04 Distance Sensor. Start by connecting the HC-SR04 to the Raspberry-Pi, as shown in the Figure below. Test the circuit using the code examples, which are provided in Appendix 1 of project D below. You will need to make a few changes in your script to get it to work.



Hints: The output of the HC-SR04 module is a logic level one of 5 V. The Raspberry-Pi works with 3.3V logic signals. A 1 kΩ resistor in series will limit the current in the circuit to prevent damaging the Raspberry Pi.

1. Work out how the HC-SR04 sensor operates. What are the "Trig" and "Echo" pins of the HC-SR04 module used for? Use the oscilloscope to capture these signals when running the code example.
2. Use these data to explain in detail how the module works and how it is possible to measure distance using ultrasonic waves.
3. Familiarise yourself with the `RPi` framework, see e.g. `https://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage`. Using the `RPi` framework complete the function `def reading(sensor):` in your script and test your circuit.
4. What are the minimum and maximum distance that your Range Finder can measure?

3.3.4 Applications using the ultrasonic range finder

Now that you have a functioning ultrasonic range finder there are several possible applications.

1. Try to understand better the properties and limitations of your device. What are the minimum and maximum size of objects the range finder is able to detect? Play with reflecting surfaces of different areas and define the angular range (span of angles) within the range finder works? Is this approach suitable for large or small objects? Is there a relation between the distance and the size of the reflection area?
2. Develop an application of your choice. An example could be an electronic "Car Parking Assistant" where sounds are created and LEDs flash (or similar) if the sensor gets too close to an object.
3. Perform a calibration of the distance scale of your application.

Note that the loudspeakers are power hungry, so they should be connected via USB to the monitor.

To play an audio file (sampled note) you can use an application such as `aplay`. Below an example is given of how to play a note using a python script.

```
import os
while True:
    if ( switch1 == True  ):
        os.system('aplay -q C5.wav &')
```

../scripts/project_D_a.py

3.3.5 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code

examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;
- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.

Appendix 1 of project D: Example Code

```
#!/usr/bin/python

def reading(sensor):

# remember to change the GPIO values below to match your sensors
# GPIO output = the pin that's connected to "Trig" on the sensor
# GPIO input = the pin that's connected to "Echo" on the sensor

    TRIG = 17
    ECHO = 27

    import time
    import RPi.GPIO as GPIO

    # Disable any warning message such as GPIO pins in use
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)

    if sensor == 0:

        # Setup the GPIO pins for TRIG and ECHO, including defining
        # if these are input or output pins

        # Insert your code here


        time.sleep(0.3)

        # sensor manual says a pulse length of 10Us will trigger the
        # sensor to transmit 8 cycles of ultrasonic burst at 40kHz and
```

```
        # wait for the reflected ultrasonic burst to be received

        # to get a pulse length of 10Us we need to start the pulse, then
        # wait for 10 microseconds, then stop the pulse. This will
        # result in the pulse length being 10Us.
        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        # listen to the input pin. 0 means nothing is happening. Once a
        # signal is received the value will be 1 so the while loop
        # stops and has the last recorded time the signal was 0
        while GPIO.input(ECHO) == 0:
          signaloff = time.time()

        # listen to the input pin. Once a signal is received, record the
        # time the signal came through
        while GPIO.input(ECHO) == 1:
          signalon = time.time()

        # work out the difference in the two recorded times above to
        # calculate the distance of an object in front of the sensor
        timepassed = signalon - signaloff

        # we now have our distance but it's not in a useful unit of
        # measurement. So now we convert this distance into centimetres
        # Define relation between "distance" and "timepassed"

        # Insert your code here

        # return the distance of an object in front of the sensor in cm
        return distance

        # we're no longer using the GPIO, so tell software we're done
        GPIO.cleanup()

    else:
        print ("Incorrect usonic() function variable.")

print (reading(0))
```

../scripts/project_D_b.py

3.4 Project E: Building a Precision Refrigerator

3.4.1 Goals of project

The course organiser would like to celebrate the DAH course with a glass of fine wine, chilled to exactly the right temperature. You will design and build a thermostat to precisely control the temperature of a fluid. Regrettably, fine wine could not be made available for teaching purposes and it will be replaced by a beaker of tap water for this project. You will use a Peltier thermoelectric device to cool the liquid. The temperature of the liquid will be monitored using 1-wire temperature sensors, which were used during the checkpoints.

The projects have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

3.4.2 Equipment for project E

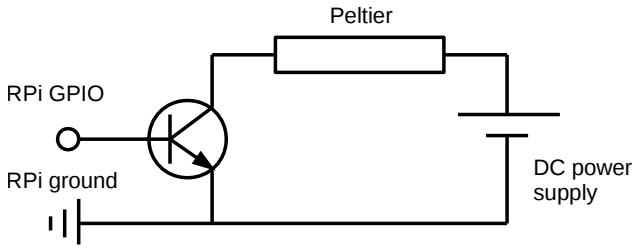
For this project you will need the following items:

- Raspberry-Pi
- DC power supply
- Peltier thermoelectric device
- 2x temperature sensor S18B20
- Power transistor Darlington TO-220
- Heat sinks
- Beaker

Some of this equipment will be located in the red box E. You will have to share some items between Tuesday and Thursday sessions.

3.4.3 Building the refrigerator

You will need to build a circuit to supply power to the Peltier element as the Raspberry-Pi cannot supply enough current itself. You can use the power supplies available in the lab, to produce a DC voltage and control it using a power transistor connected to a GPIO pin on the Raspberry-Pi. Use a 1 A, 1 V supply for the Peltier.



Example circuit diagram for Peltier control.

Research Peltier devices: note that one side gets hot as the other side gets cold, so you will need to place the hot side in contact with a heat sink. Do not power the device for more than a few seconds without this heat sink, or you may damage it! You may also need a heat sink for the transistor.

The temperature of the liquid must be monitored, so you will need to find a way to get the 1-wire sensors near to it. You will be provided with a waterproof temperature sensor, with part number DS18B20, so you will need to adjust your code appropriately if reading it with webIOPi.

3.4.4 Monitoring and controlling the temperature

Aim first to be able to precisely control the liquid at a few centigrade below room temperature before trying to find out how cold you can make the liquid. The Peltier device will take a long time to cool the liquid much below room temperature, so don't waste time waiting for this.

During Checkpoint 5 you already learned how to read information from the 1-wire temperature sensors, but you should now consider this in more detail. How precisely can you measure the temperature of the liquid? Are there systematic effects that you can account for by calibrating your sensors? What will you do about temperature gradients across your refrigerator when the Peltier element is switched on? Remember that you have additional (but not waterproof!) sensors that you used in Checkpoint 5 — these might be useful, and you have already studied their performance.

Your project should include a way of displaying the current temperature of the liquid, and previous measurements. You should also include a visual indication of the state of the Peltier, rather than just poking it to see if it feels cold.

The circuit to power and control your Peltier device is simple, but you should think carefully about when your python code should switch the cooling on and off. Rather than having a simple threshold temperature for turning the cooling on and off, you may want to use a hysteresis loop. Rapidly toggling power to the Peltier in an uncontrolled fashion is unlikely to give good performance.

You could enhance the scope of your project in the following ways.

1. Try varying the cooling power of your Peltier using pulse-width modulation. Here you would use your Raspberry Pi GPIO pin to create a square wave to toggle the power

transistor on and off. You then adjust what percentage of time is spent in the on or off state: the 'duty cycle.'

2. Write an interface that allows the user to start and stop the temperature control programme, show the status and temperature of the Peltier, and change the temperature value of the thermostat.
3. Consider running the Peltier as a heating element. What changes are required to the sensor, the circuit and the control programs?

3.4.5 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;
- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.

3.5 Project F1: Make Accurate Measurements of Particle Masses

3.5.1 Goals of project

You will use LHCb data on the invariant mass of particle candidates that you were introduced to during a checkpoint. You will analyse this in a much more sophisticated way and closer to the actual analysis performed leading to its publication. You will use the maximum likelihood process to fit different mass model shapes to the data. From this you will determine the parameters of the mass model for the three signal peaks, and their errors. You will start with a very simple Gaussian mass model. You will then improve this and use a more sophisticated model.

The projects have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

3.5.2 Equipment for project F1

Note: this is a data analysis project, which does not require use of the Raspberry Pi. This project is best carried out using the Physics CPlab computers. No other equipment is needed. There is a CPlab computer available on every desk in the DAH laboratory. You may also use your own laptop computer, but you will need to be able to install python and python packages on your own.

3.5.3 Detailed project description

You were previously introduced to the LHCb Upsilon data. 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 Υ(1S), Υ(2S) and Υ(3S) mesons where the Υ(1S) meson is the ground state and the Υ(2S) and Υ(3S) states are radial excitations (for LHCb paper, see DOI: 10.1007/JHEP06(2013)064).

During checkpoint 6, you performed some very simple "peak finding". In this project you are going to do the analysis much like it would actually be carried out in a particle physics experiment.

Download the files ups-15.bin and ups-15-small.bin from the DAH Dropbox, These files contain the data recorded by LHCb in 2015 and a subset with a factor 5 less data. The files are written in binary format and contain five observables for a large number of muon pairs

- invariant mass of muon pair in GeV/c^2;
- transverse momentum $p_\perp$ of muon pair in GeV/c;

- rapidity η of muon pair;
- momentum p of muon pair in GeV/c;
- transverse momentum $p_{\perp,1}$ of first muon in GeV/c;
- transverse momentum $p_{\perp,2}$ of second muon in GeV/c.

Write a python script that reads the data from this file, see below. Plot histograms of all six variables, choosing a reasonable bin width. Always label plots correctly with title and axes and save these to a file.

```
import numpy as np

# import data
# xmass = np.loadtxt(sys.argv[1])
f = open("datafiles/ups-15-small.bin","r")
datalist = np.fromfile(f,dtype=np.float32)

# number of events
nevent = len(datalist)/6
xdata = np.split(datalist,nevent)
print(xdata[0])

# make list of invariant mass of events
xmass = []
for i in range(0,nevent):
    xmass.append(xdata[i][0]/1000.)
    if i < 10:
        print(xmass[i])
```

../scripts/project_F_a.py

1. Consider first the Upsilon(1S) (Υ(1S)) particle, which is the particle with the lowest mass, i.e. the left most peak in the plot. Construct a composite probability density function (PDF) for the invariant mass of the muon pairs, which contains two components:

   - A Gaussian shape to fit the Υ(1S) mass peak;
   - A shallow falling exponential to fit the background shape of the mass spectrum underneath and around the peak.

2. Use this PDF in a Maximum Likelihood fit to determine the parameters of the PDF. Note hat it is essential that the composite PDF remains normalised to 1 over the range of the fit.

   Determine the Υ(1S) meson mass and yield, and all other parameters, and their and errors.

   You should be able to obtain the parameter errors directly from the minimization engine of your choice (scipy.optimize.minimise, scipy.optimise.curve_fit, lmfit, see `https://lmfit.github.io/lmfit-py/` or Minuit). Depending on your choice you will be able to chose different minimising methods. It would be good to show that you understand these by obtaining them yourself from the parameters of the Gaussian signal fit - this is described in the data handling lectures.

   Plot the fitted signal shape on top of the data.

3. Now consider the entire mass range, and perform a simultaneous fit for all three Upsilon peaks, and the underlying background. Again you should always report the parameter values, and their errors. Plot the fitted signal shape on top of the data.
4. The results so far probably look "quite good" by eye. i.e. the signal shape plotted on top of the data probably looks like it fits quite well. However this can be misleading when performing a precision measurement. You should make a plot of what are called the "residuals". A residual is the difference between the data in the binned histogram and the best-fit mass model value for the centre of that bin. Describe what you see.
5. There are several ways to enhance the scope of the project. For example, if the single Gaussian mass model does not fit the data perfectly, one can try other mass models, i.e. by using a signal PDF that goes beyond a single Gaussian function. Examples are:
   - A PDF comprising a function which is the sum of two Gaussian functions (i.e. one narrow and one wide Gaussian function to fit a single Upsilon mass peak)
   - A Crystal Ball function, which incorporates a non-Gaussian tail at the lower end of the mass peak. The functional shape is described elsewhere, e.g. see: `http://en.wikipedia.org/wiki/Crystal_Ball_function`.
     You could implement each of these functions in your PDF and see how much better they are at describing the data.
6. Make 2-dimensional plots of the additional observables vs the invariant mass of the muon pair. Find out if you can improve the purity of your signal sample. The purity is defined as the ratio of signal events over all events in a region.
7. As on open ended activity, you will see in the paper that the analysis is also done by dividing the data up into bins of transverse momentum (pT) and rapidity (eta). You can explore doing this analysis yourself. It is somewhat more complex as you have to think about which are common parameters (e.g. masses) and which are not (e.g. background fractions).
8. Read the publication and see what is said about systematic errors. Make a reasonable attempt at determining some systematic errors on the masses.

3.5.4 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;

- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.

3.6 Project G: Building a Remote Sensing System

3.6.1 Goals of project

You will develop a remote sensing system, similar to a weather station. A WiFi micro-controller will be used to acquire temperature and humidity data and publish these on a embedded web server. You will use the Raspberry-Pi to download these data and build a monitoring system.

The projects have an open-ended aspect and are an opportunity where you can show your own initiative and demonstrate your experimental and computational skills.

3.6.2 Equipment for project G

For this project you will need the following items:

- Raspberry-Pi
- Adafruit HUZZAH ESP8266 WiFi micro-controller
- DHT22 Temperature Humidity Sensor
- DC power supply
- USB to TTL UART 6PIN CP2102 Module Serial Converter
- Adafruit RGB 16x2 LCD and Keypad Kit
- Windows PC in laboratory
- WiFi hotspot

Some of this equipment will be located in the red box G. You will have to share some items between Tuesday and Thursday sessions.

3.6.3 Building the Remote Sensing System

You will develop a remote sensing system, similar to a weather station. For this you need to put together different elements, including connecting a temperature and humidity sensor to a micro-controller, programming the micro-controller, connecting an LCD display to the Raspberry-Pi, and remotely connecting the sensing system to the Raspberry-Pi via a WiFi hotspot. It is suggested that you familiarise yourself with all the components by consulting the corresponding web pages and reading their technical specifications. A micro-controller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals, so it is more limited than a Raspberry-Pi. Here we will use the Adafruit HUZZAH ESP8266 WiFi micro-controller, see `https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/overview`.

1. Mount the Huzzah ESP8266 micro-controller on a breadboard. Use a DC power supply to provide the +5V. Please note that it is always recommended to switch off the power supply when connecting devices or changing the wiring. Connect the 3.3V output and GND to the rails on the breakout board. Then use the USB to Serial cable - Receive, Transmit and Ground Lines (Rx,TX, GND) - to connect the ESP8266 with to the Windows PC, located on the desk in the right-hand corner of the DAH laboratory. When connected, reset theESP8266 by pushing the reset button while holding the GIPO0 button. This will make the micro-controller ready to boot and a dimmed red LED will show.

2. To program the ESP8266 you need to start the Arduino-IDE package, which is installed on the Windows PC. Using the instructions on `https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-arduino-ide`, go to Arduino → File → Preferences and copy the following link into the Additional Board Manager URLs field:
   `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
   Then you invoke the Board Manager (Tools → Board → Boards Manager), search for say "ESP" and install "esp8266 by ESP8266 Community". If successful, the ESP8266 should appear under Tools → Board. Check that the CPU Frequency is set correctly (80 MHz) on the micro-controller and set upload baud rate (115200) and the matching port of your USB to serial cable (e.g. COM4).

   In addition you need to install two libraries. Use the library manager (Sketch → Include Libraries → Library Manager), first search for say "Adafruit Unified" and install "Adafruit Unified Sensor library by Adafruit" then search for say "DHT" and install "DHT sensor library by Adafruit".

3. As a first programming exercise, perform the "Blink Test". Consult the instructions on `.../using-arduino-ide`. Copy the code into an Arduino IDE sketch, compile the code (Sketch → Verify / Compile) and upload (⇒) to the micro-controller. The sketch will start immediately - you'll see the LED blinking. Hooray!

4. The next step is to connect the temperature and humidity sensor DHT22, see `http://www.adafruit.com/products/385`, to the micro-controller. The DHT22 requires three lines (3.3V, GND and data). Use GPIO #2 as serial input on the ESP8266. Remember to switch off the power supply during this process.

5. You are now ready to program the temperature and humidity web server. Download the example code into the Arduino-IDE application, using the instructions from `https://learn.adafruit.com/esp8266-temperature-slash-humidity-webserver/code`. You will need to change this code by adding the name and password of the WiFi hotspot.

```
const char* ssid     = "DAH_ADHOC";
const char* password = "dahlab15";
```

../scripts/wifi.ino

   Compile the code (Sketch → Verify / Compile) and upload it to the micro-controller. Recall to put first the micro-controller ready into boot state.

6. In order to connect the ESP8266 micro-controller to the WiFi hotspot, which is located on the desk in the left corner of the laboratory, switch on the WiFi hotspot, if necessary,

push the reset button on the micro-controller and open a Serial Monitor on the Arduino-IDE (→ Tools → Serial monitor). The ESP8266 & DHT22 sensor system should start to work and you should obtain the IP address of the HTTP server, see:

```
Working to connect ..................................................

DHT Weather Reading Server
IP Address: 192.168.1.2
HTTP server  started
```

7. The final step is to read the DHT22 & ESP8266 remotely with the on-board WiFi adapter of the Raspberry-Pi. To enable WiFi on the Raspberry-Pi, execute the following command:

   ```
   studentnn@dahpimm  ~ $ wpa_cli enable wlan0
   ```

   Check the network connection and select the WiFi hotspot. You might have to type in the password. Using the Chromium web browser, you can now read the sensor by connecting to the correct IP address, e.g. `http://192.168.1.2`. Find out how to read temperature and humidity from the DHT22 Adafruit webpage, `https://learn.adafruit.com/esp8266-temperature-slash-humidity-webserver/using-the-webserver`.

3.6.4 Using the Remote Sensing System

Reading temperature and humidity using a web browser is straightforward, but you want to go beyond single measurements. Write a python script which samples a series of measurements and displays these. For this you need to parse the web server data. Use the instructions at https://docs.python.org/3/howto/urllib2.html to write a script that will regularly read the temperature and humidity from the remote sensor and print the data onto the screen and/or file. What would be a reasonable update frequency? Further possible applications could include the following:

1. Write a well structured python script in which reading the temperature and humidity sensor are functions or classes. Convert temperature values from Fahrenheit into Celsius.
2. Write a script which displays an animated series of measurements in real time. Use a hot air-blower/hair-dryer to vary the temperature. Avoid applying the heat-source for too long to prevent the DHT22 from starting to melt. Plot the temperature versus humidity and explain what you observe.
3. Display the temperature and humidity on an LCD display. Connect the LCD display to top of the Raspberry-Pi. Make sure that you use the provided GPIO port Extender plug to keep the LCD board from touching the USB and Ethernet ports of the Raspberry-Pi. Download the python script `char_lcd_plate.py` from `https://github.com/fmuheim/DAH`. Run this script to understand how to program the LCD display. Consulting the instructions at

`https://learn.adafruit.com/adafruit-16x2-character-lcd-plus-keypad-for-raspberry-p` `usage`, write a python script which regularly reads and updates temperature (in Celsius) and humidity on the LCD display.

3.6.5 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;
- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.

3.7 Project X: Develop a Project or Suggest Your Own Project

3.7.1 Goals of project

We have a few possible projects where the project descriptions and goals are not advanced enough to be included in the list of projects A to G, see the list below. If you are interested in one of these, please let us know. In addition, if you have an idea of what you want to do with the Raspberry-Pi, please tell us and we will discuss it. Please note that we will have to make a decision if your own project or one from the list below is feasible within the time scale of the DAH course.

3.7.2 List of possible projects

We give here a list of projects using the Raspberry-Pi and/or the Arduino, where more development work is required and little or no project description exists.

- Building an oscilloscope using the Arduino micro-controller as an ADC and the Raspberry-Pi as the DAQ to display waveforms.
- Building a motion sensor using an accelerometer connected to an ADC and read out by the Raspberry Pi.
- Build a CCTV-like imaging capturing triggered by a motions sensor, for info see `http://www.raspberrypi.org/learning/python-picamera-setup/`.
- Control switches, LEDs and relays using the PiFace Digital expansion boards, see `http://www.piface.org.uk/products/piface_digital/`.
- Controlling a toy train set-up in the laboratory, for info see the following web link: `http://www.mathworks.co.uk/company/newsletters/articles/adding-fun-to-first-year.html?nocookie=true`. You will need to bring your own toy train set-up.
- Suggest your own project.

3.7.3 Project planning

The project descriptions are generally significantly less detailed than what was made available for the checkpoints. Any material covered during checkpoints including python code examples are assumed to be known. Only essential and new information is provided and you are expected to take care of the details. Python code snippets are provided where necessary, but you will have to understand yourself what they do. It is recommend that you google for information about your project on the web, including data sheets of components and python libraries, if applicable. Python scripts should be well structured, either using functions or classes.

The timeline will vary between different projects, but in general, it is recommended that you plan your work as follows:

- weeks 8 & 9: Building your gadget and/or writing the required code;
- week 10: Analysis of data or equivalent;
- week 11: Finish writing of project report and prepare submission.

Note that you are advised to start writing your report as the project progresses.

For guidance on report writing, how the projects will be assessed, plagiarism and the submission deadline, please consult the DAH course booklet, available on Learn.