DTU

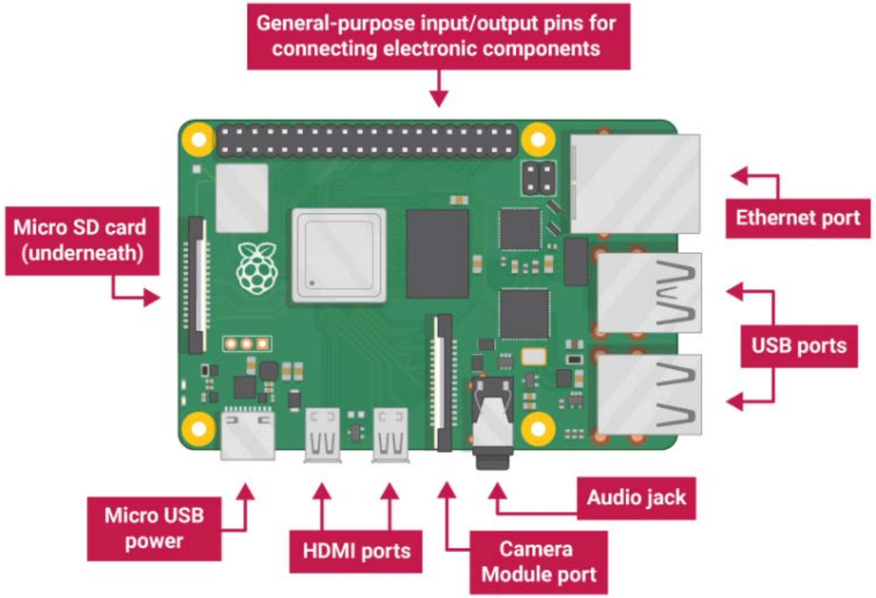
DTU Skylab Digital – Introduction Workshop

Getting Better with Raspberry Pi

101 Summary

- Used the Pi as a "desktop PC"
- Used the terminal and CLI
- Wrote a couple of Python programs
- Read buttons
- Wrote to LEDs

- Started a Python program from bootup
- Installed and used libraries
- Worked with conditions and loops
- Used the Python Interpreter and CLI


General-purpose input/output pins for connecting electronic components
Ethernet port
Micro SD card (underneath)
USB ports
Micro USB power
HDMI ports
Audio jack
Camera Module port

Agenda

- Get connected to the Pi over SSH and establish FTP connection

- Set up an i2c OLED display
- AD Conversion
- Analogue Sensors
- Smoothing
- Threads

- Digital Sensors
- Multiplexing
-
- Web Services and APIs
- Google Text To Speech
- API example – download song data from Genius.com
-
- Using the Pi Camera
- Overview of useful libraries

SSH Connection

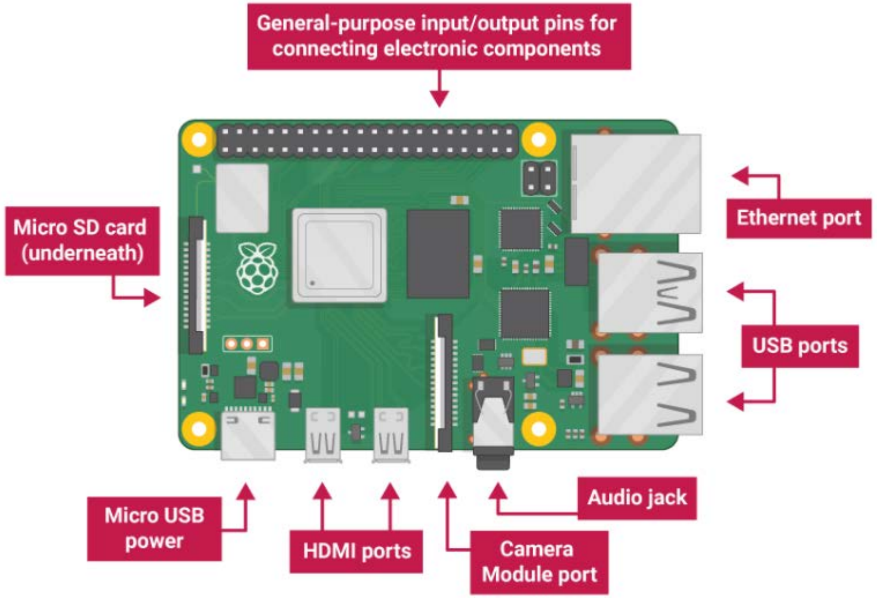
What we Need:

- The Pi’s IP address (run these commands on Pi)
  - ***hostname –I***
  - ***ifconfig***

- A PC **on the same network** with a tty client installed
  - user: pi
  - password: raspberry

- The Port Number (22)

- TASK: Establish an SSH connection to the Pi and make a folder for today


General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Micro USB power
HDMI ports
Camera Module port
Audio jack

The SSH interface is the terminal and CLI. Remember these commands: cd, ls, mkdir, rm, sudo …,

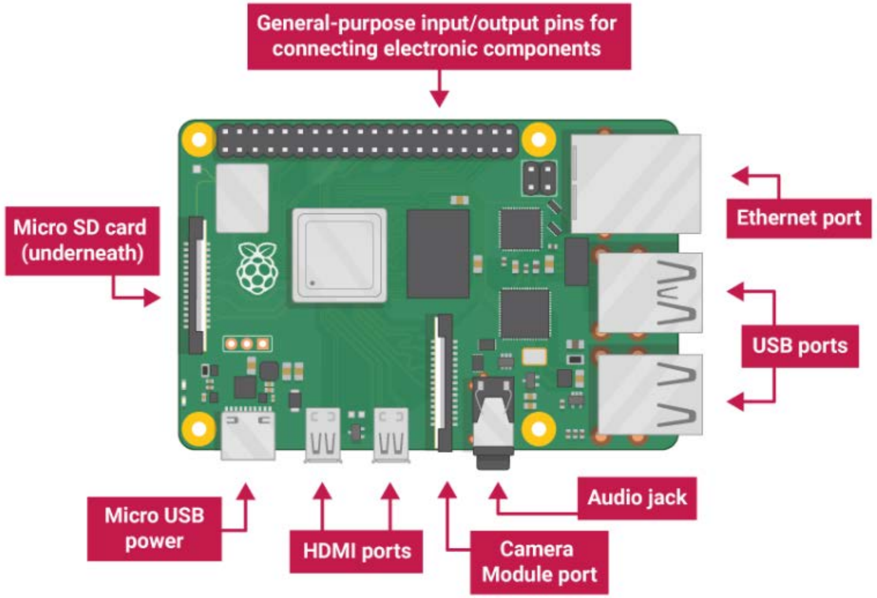
FTP Connection

Use the same credentials, IP and Port.

– Create a Python folder on your local HD

– Set up the FTP and save the site

– Set the view to HD/Python Pi/Python

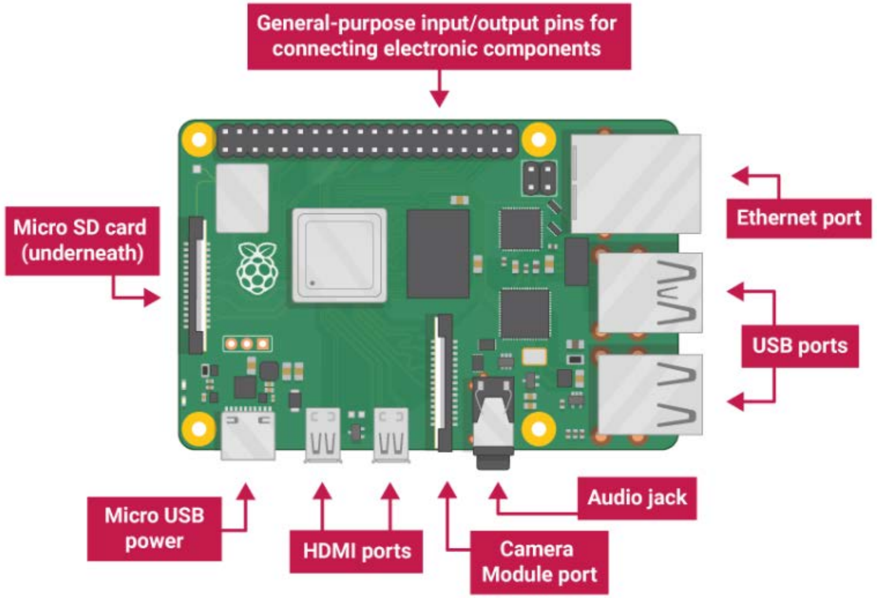

General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Micro USB power
HDMI ports
Camera Module port
Audio jack

- Install e.g. Sublime Text if you do not have an IDE
- Useful: Have Python installed on client PC for experimentation

- TASK: Write something to the terminal using Python
  - – Write code in text editor and save as .py
  - – Upload to Pi by FTP
  - – Execute on CLI over SSH – ***python myfile.py***

Our Development Environment

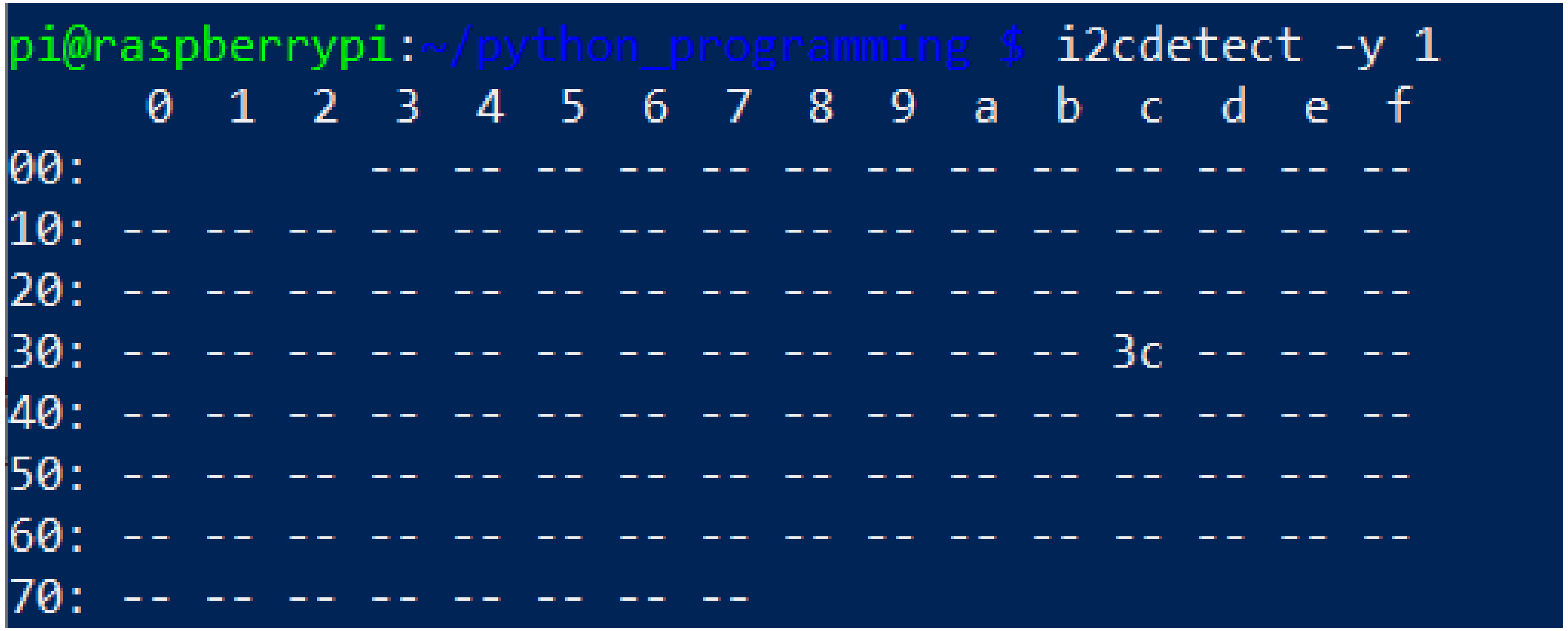
- Write in Text Editor
- Upload using FTP Client
- Run on CLI of Pi over SSH


General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Micro USB power
HDMI ports
Camera Module port
Audio jack

- Good to have Python installed locally
  - **But** libraries not 1:1 on Pi/Win
  - **But** GPIO read/write will throw errors (***try, except***)
  - **But** different camera interface etc.
- Use a TE with syntax highlighting and autocomplete
- Use a Python IDE, e.g. PyCharm, Spyder
- Fix the Pi’s IP address (if you have a permanent installation)
- Change the default user/pass

Set up your display

- Plug in the display to the i2c Clock and Data lines, and +5v and GND
- Find the address using the `i2cdetect` command



```
pi@raspberrypi:~/python_programming $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

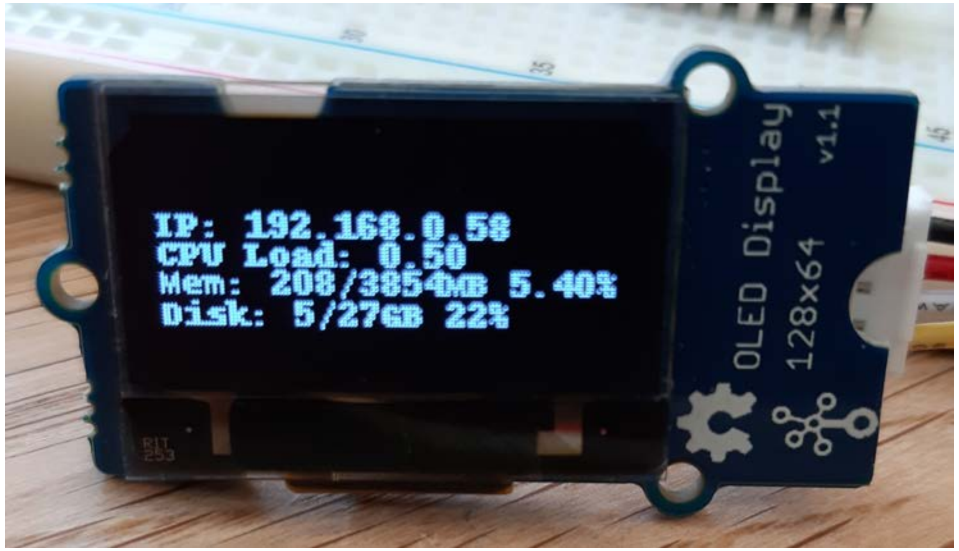
Set up your display

https://github.com/bencahillDTU/Raspberry_Pi_102

```
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
cd Adafruit_Python_SSD1306
sudo python setup.py install

git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
cd Adafruit_Python_GPIO
sudo python3 setup.py install
```

Set up your display

- Test the OLED by cd’ing to SSD 1306 ***examples*** and running a couple
- Change some of the values and prepare for reading some sensors
- Try to experiment with the PIL library
  - Draw shapes
  - Import an image
  - Scroll some text
- See my ***expand.py*** example

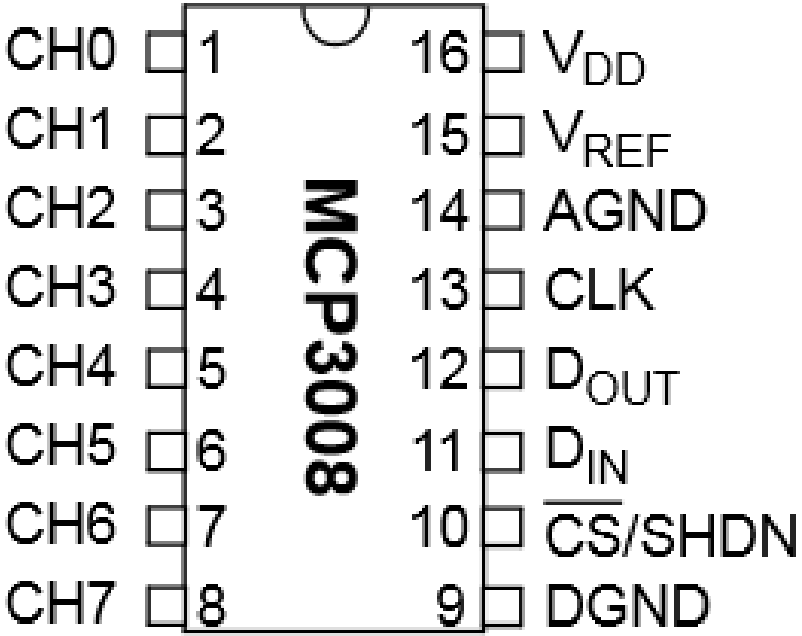


```
IP: 192.168.0.58
CPU Load: 0.50
Mem: 208/3854MB 5.40%
Disk: 5/27GB 22%
```

AD Conversion and Analogue Sensors

- Interface that we can query
- 8 Channels
- 10 bit (1024 levels)
- SPI Interface

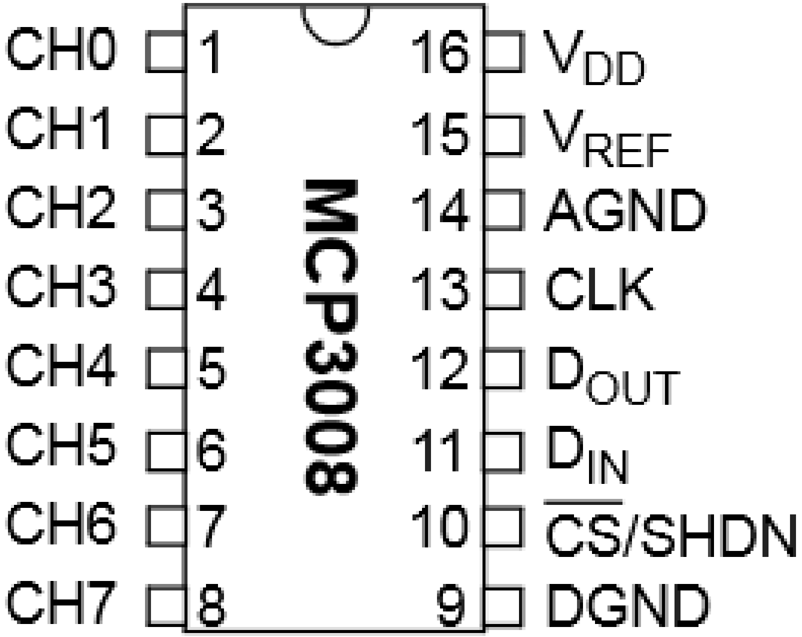
- Left side: INPUT
- Right: Logic and Output



- Connect the IC to the breadboard across the groove

AD Conversion and Analogue Sensors

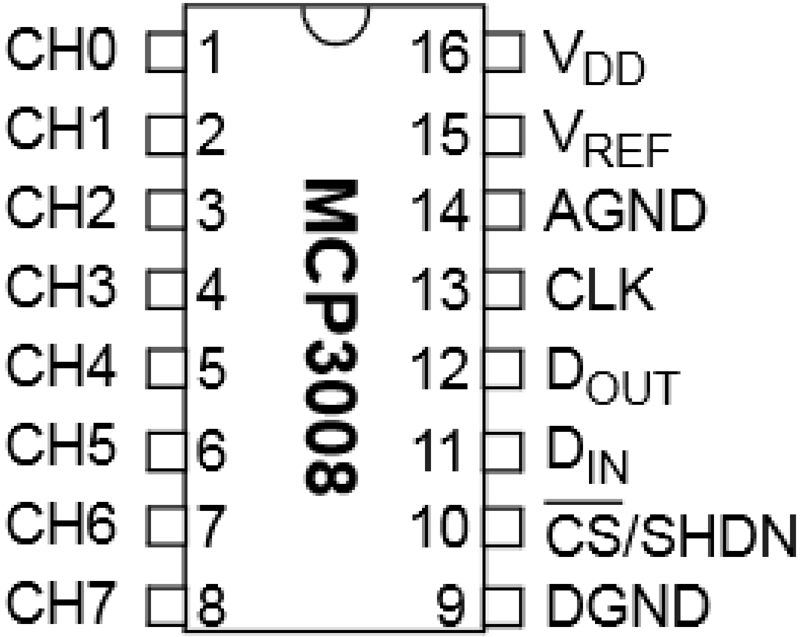
GPIO	MCP3008
Pin 1 (3.3V)	Pin 16 (VDD)
Pin 1 (3.3V)	Pin 15 (VREF)
Pin 6 (GND)	Pin 14 (AGND)
Pin 23 (SCLK)	Pin 13 (CLK)
Pin 21 (MISO)	Pin 12 (DOUT)
Pin 19 (MOSI)	Pin 11 (DIN)
Pin 24 (CE0)	Pin 10 (CS/SHDN)
Pin 6 (GND)	Pin 9 (DGND)



AD Conversion and Analogue Sensors

- Connect a potentiometer
- 1 to 3.3v
- 2 to CH0
- 3 to ground

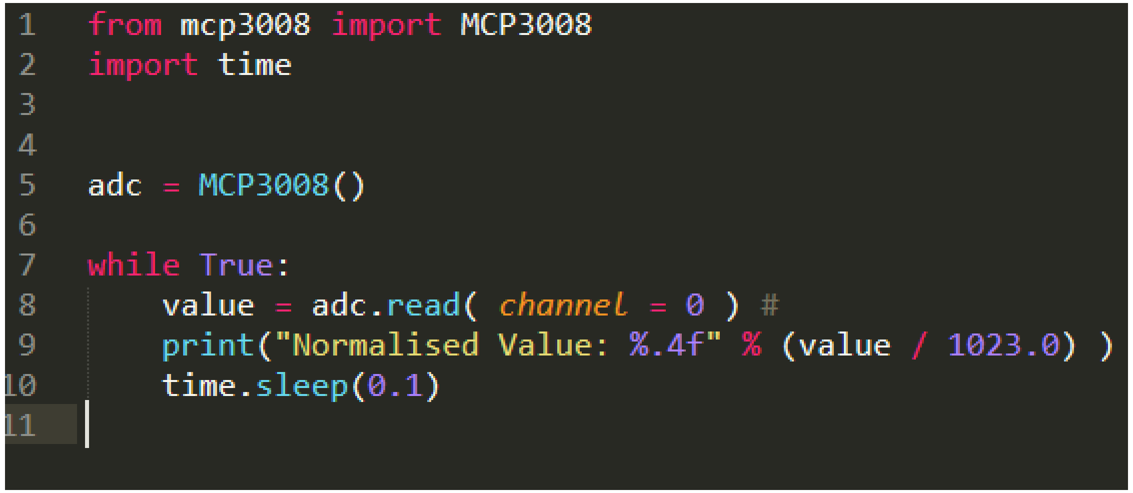
- Connect an LDR
- 1 to 3.3v
- 2 to CH1
- 2 to GND through 10k resistor



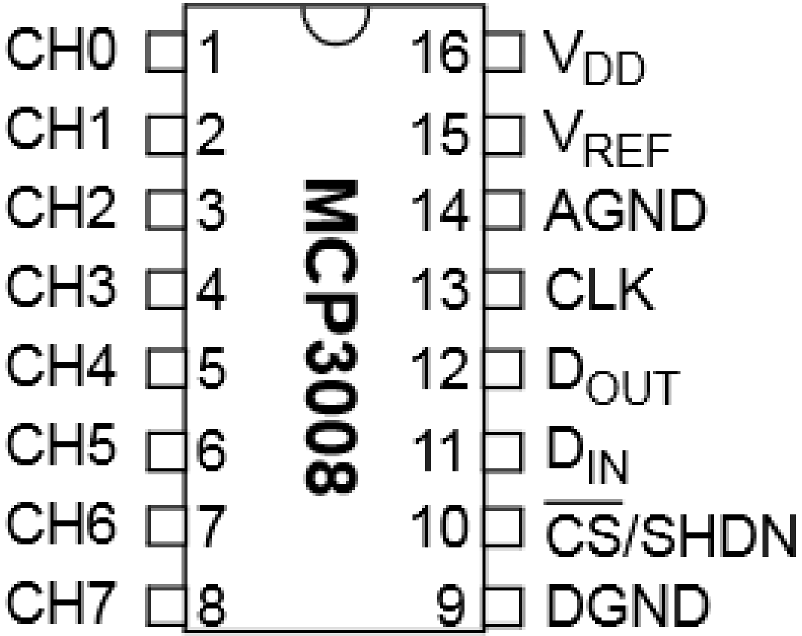
AD Conversion and Analogue Sensors

- Install the library
- Download the mcp3008.py class file
- Open it and look

- Usage



```python
from mcp3008 import MCP3008
import time


adc = MCP3008()

while True:
    value = adc.read( channel = 0 ) #
    print("Normalised Value: %.4f" % (value / 1023.0) )
    time.sleep(0.1)

```



AD Conversion and Analogue Sensors

- TASK:
  - Map the pot value to the width of rectangle on the display
  - OR
  - Display the value

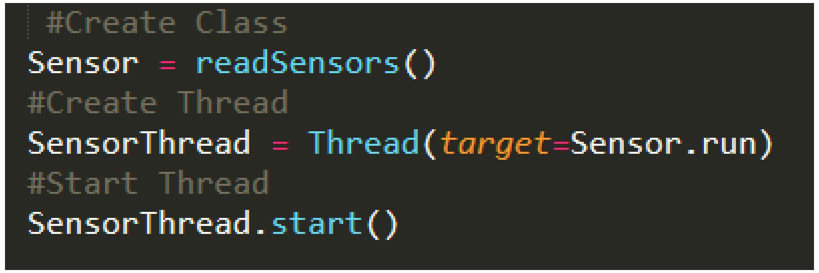
- The potentiometer is stable
- The LDR less so
- Try another analogue sensor (temperature)

- Smooth the data
- Over x amount of samples
- HINT use the ***sum()*** list method
- How is the performance?!

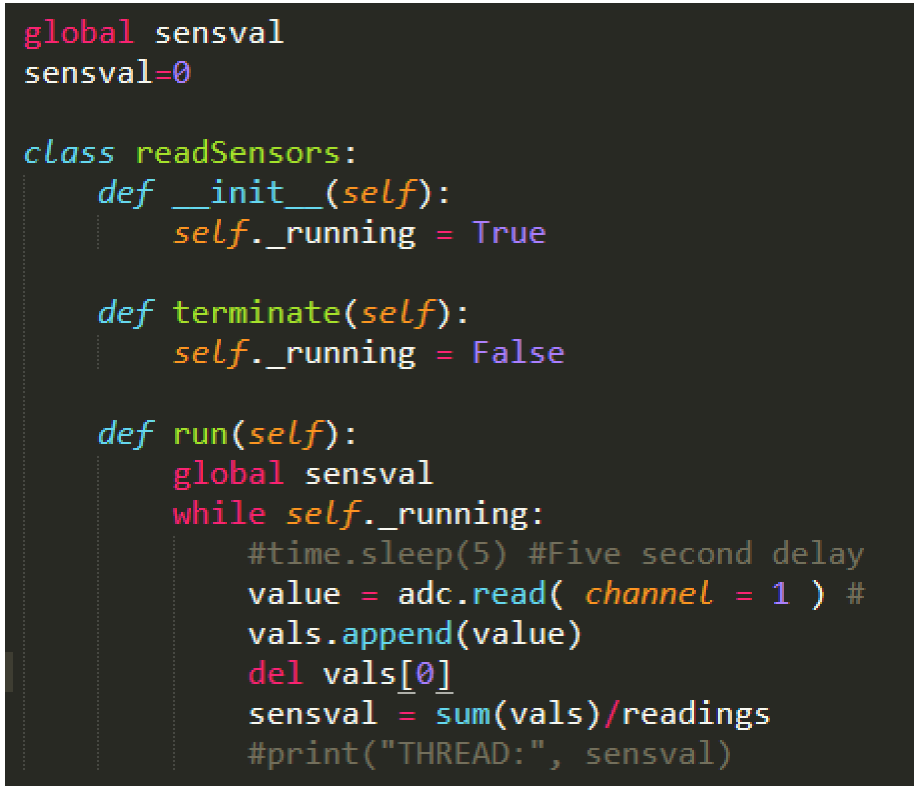
Threads

- TASK:
  - `from threading import Thread`
  - Create a new class to handle the sensor reading
  - Update the display in the main body of the program
  - Use a global variable to hold the sensor data

Initialise:



```
#Create Class
Sensor = readSensors()
#Create Thread
SensorThread = Thread(target=Sensor.run)
#Start Thread
SensorThread.start()
```



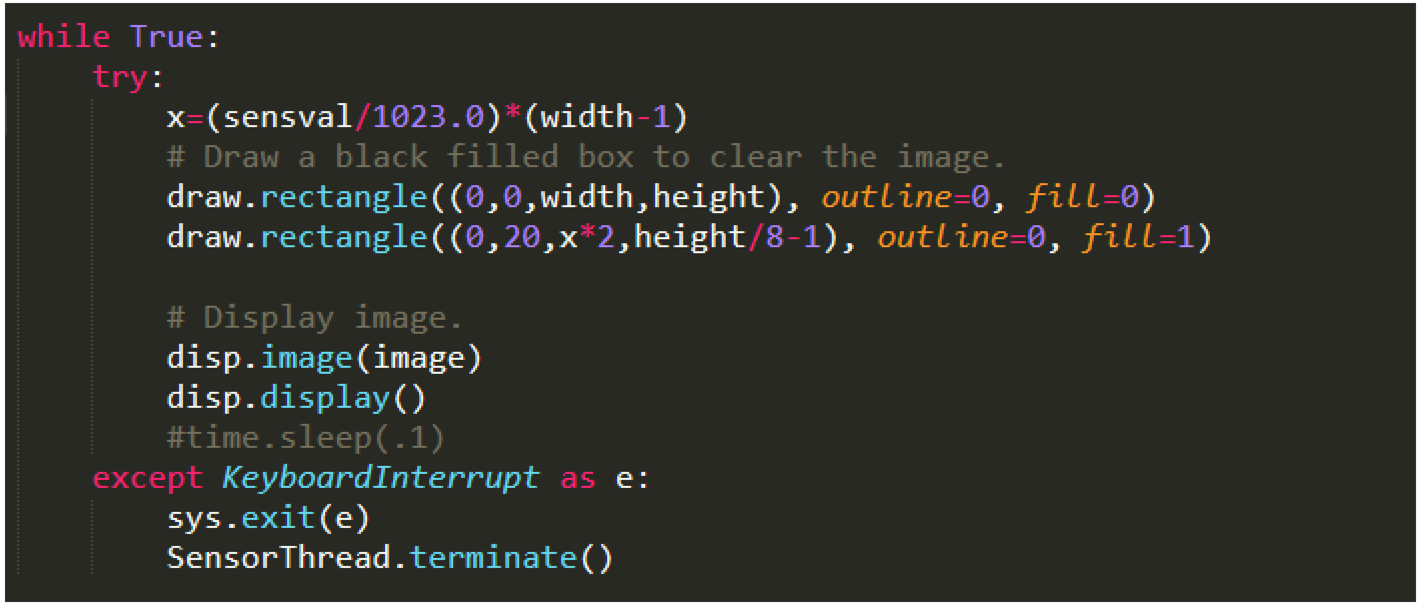
```
global sensval
sensval=0

class readSensors:
    def __init__(self):
        self._running = True

    def terminate(self):
        self._running = False

    def run(self):
        global sensval
        while self._running:
            #time.sleep(5) #Five second delay
            value = adc.read( channel = 1 ) #
            vals.append(value)
            del vals[0]
            sensval = sum(vals)/readings
            #print("THREAD:", sensval)
```

Threads will run unless you terminate them!

- Catch the keyboard interrupt
- Safely Exit
- Terminate the thread(s)

- How’s the performance?!
- Increase the number of samples

- TASK:
  - Read a couple of sensors
  - Into a ***global*** list
  - Make a small dashboard



```
while True:
    try:
        x=(sensval/1023.0)*(width-1)
        # Draw a black filled box to clear the image.
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        draw.rectangle((0,20,x*2,height/8-1), outline=0, fill=1)

        # Display image.
        disp.image(image)
        disp.display()
        #time.sleep(.1)
    except KeyboardInterrupt as e:
        sys.exit(e)
        SensorThread.terminate()
```

Summary of AD Conversion with MCP3008

- Allows us to sample 8 inputs
- At 10 bit resolution
- SPI Interface
- Very fast conversion rate (200kHz @5v)

- Threaded to improve performance
- Faster readings necessitate smoothing

- Cheap
- Easy to implement in a design
- Robust, “old” tech
- 3208 is 12 bit

Multiplexing (MUX)

- Digital “switches”
- One to many (8:1 = 1:8)
- Uses binary logic to change switch position

- WHY?
- Expands I/O cheaply
- Can be used for input OR output
- Works well with e.g. 7-segment displays

- Examples:
  - – “knobby”´interfaces
  - – Anything with lots of LEDs

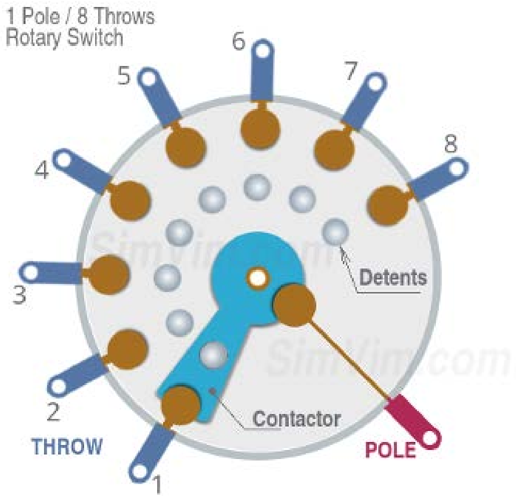
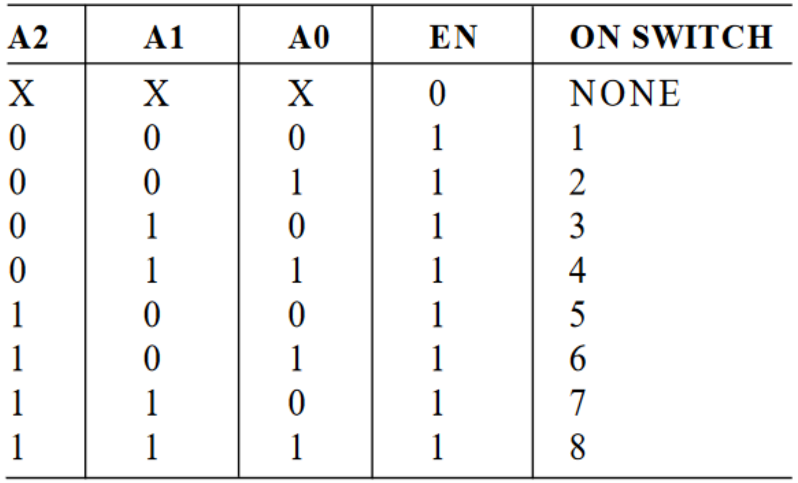


Table I. ADG608 Truth Table



A2	A1	A0	EN	ON SWITCH
X	X	X	0	NONE
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8

X = Don’t Care

Multiplexing (MUX)

- A0, A1, A2 are the logic
- S1…S8 are the poles
- D is the single pole
- EN is enable
- Vss should be at ground
- Vdd should be at +5v

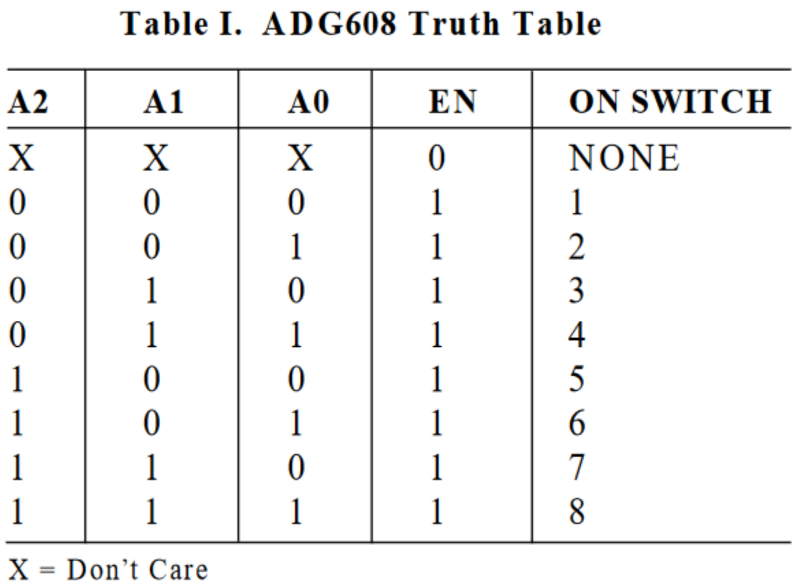
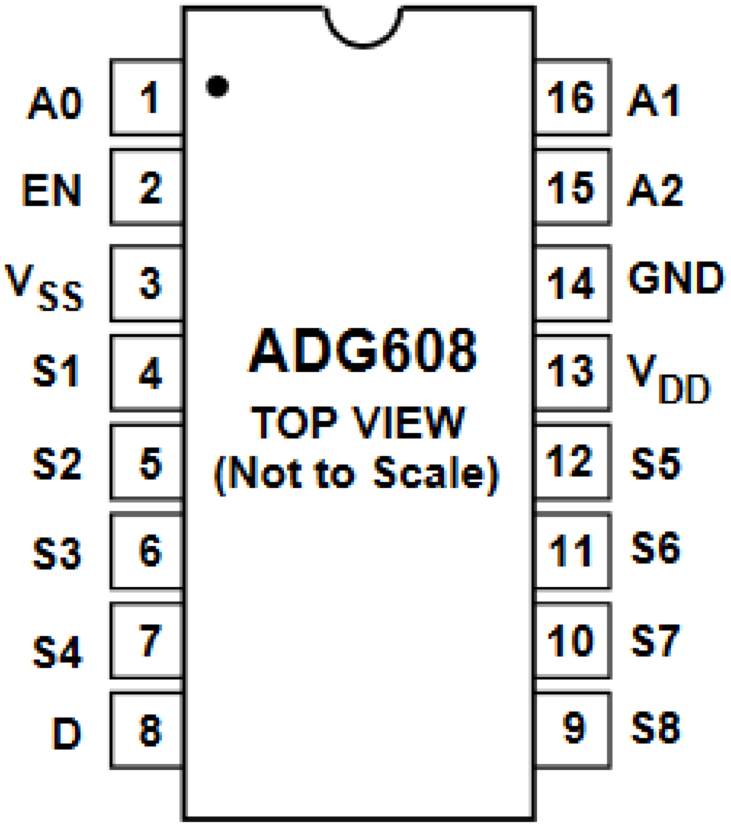


Table I. ADG608 Truth Table

A2	A1	A0	EN	ON SWITCH
X	X	X	0	NONE
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8

X = Don't Care



Multiplexing (MUX)

– TASK:
- Write to 8 LEDS
- LED + to Sx pin
- LED – to ground through small resistor
- Ax pins to Pi GPIO
- pinmode(output)

- Let’s see how fast we can do this.
- Get 8 LEDs or a segmented display
- Design a sequence
- Iterate through the logic

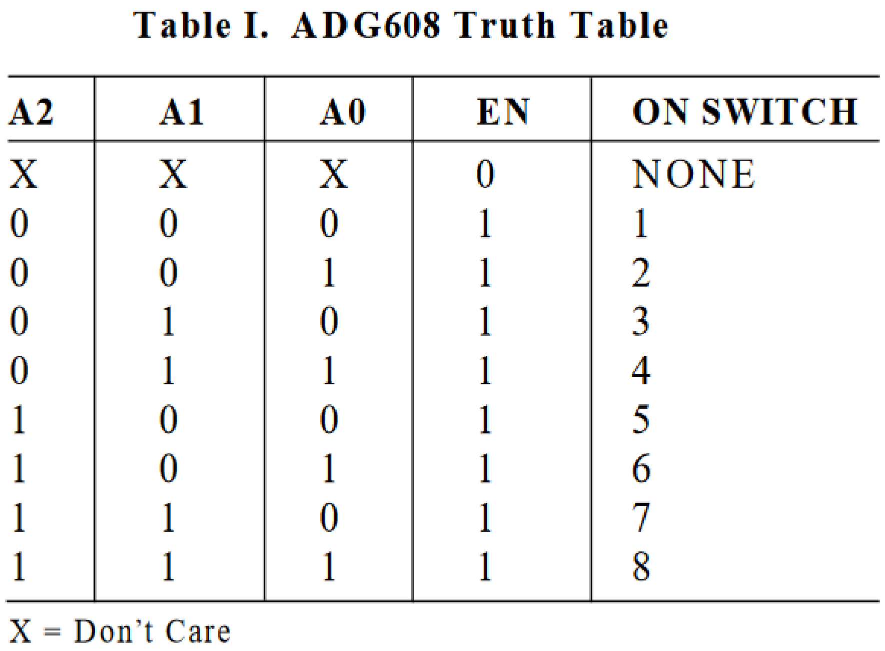
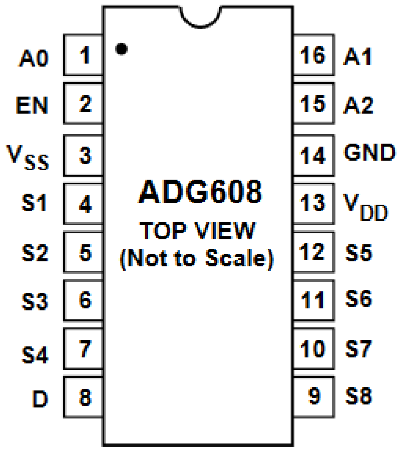


Table I. ADG608 Truth Table

A2	A1	A0	EN	ON SWITCH
X	X	X	0	NONE
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8

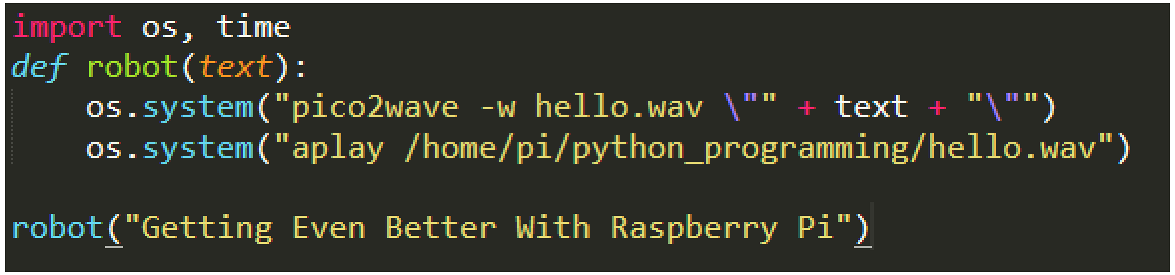
X = Don’t Care



CLI tools

- You can call CLI tools from Python
- The ***os*** library
- When there is no Python interface
- Install TTS

- Call the CLI command
- Note the escape characters



```
import os, time
def robot(text):
    os.system("pico2wave -w hello.wav \"" + text + "\"")
    os.system("aplay /home/pi/python_programming/hello.wav")

robot("Getting Even Better With Raspberry Pi")
```

CLI tools

- Transfer data from the Pi to a remote machine

- Use scp to transfer FROM the pi

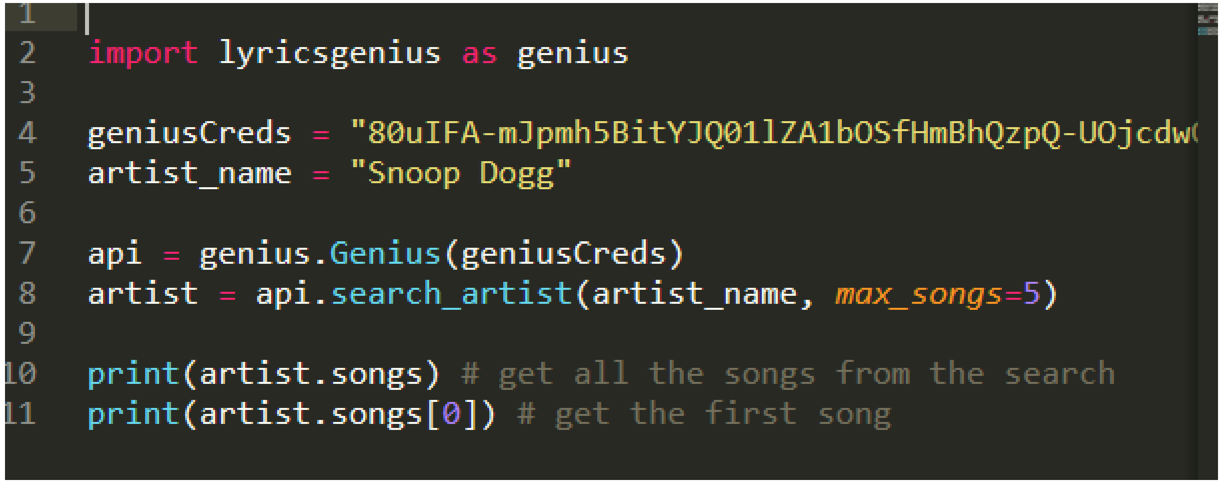
-
  ```
  scp  pi@192.168.0.58:~/copy.py copy_copy.py
  ```

- Use OpenSSH
- https://winaero.com/enable-openssh-server-windows-10/

APIs

- Interface with another application
- Get data or use services

- Genius API
- Get data about music.



```
import lyricsgenius as genius

geniusCreds = "80uIFA-mJpmh5BitYJQ01lZA1bOSfHmBhQzpQ-UOjcdw
artist_name = "Snoop Dogg"

api = genius.Genius(geniusCreds)
artist = api.search_artist(artist_name, max_songs=5)

print(artist.songs) # get all the songs from the search
print(artist.songs[0]) # get the first song
```

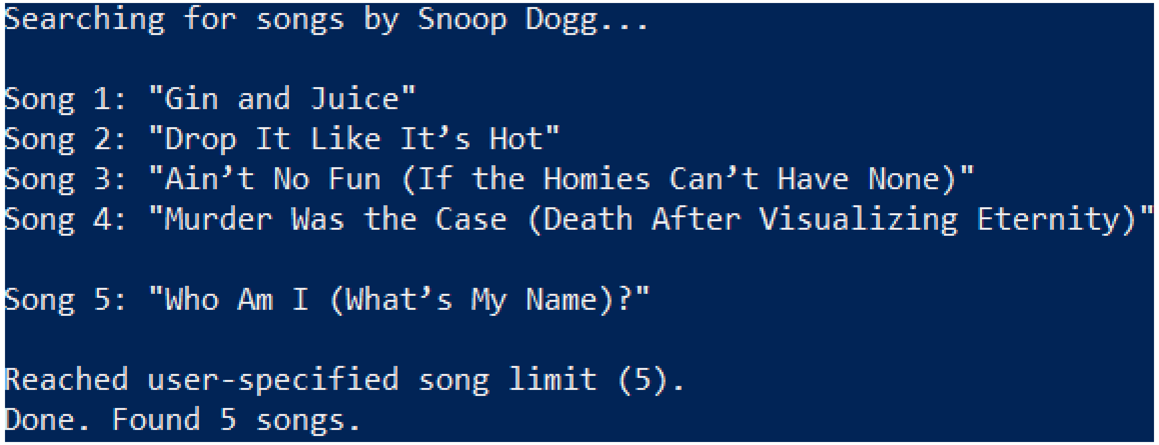
APIs

- Sign up for account
- Install dependencies
- Get an API key
- Use it in your Python program

- https://docs.genius.com/

- Let’s try it.

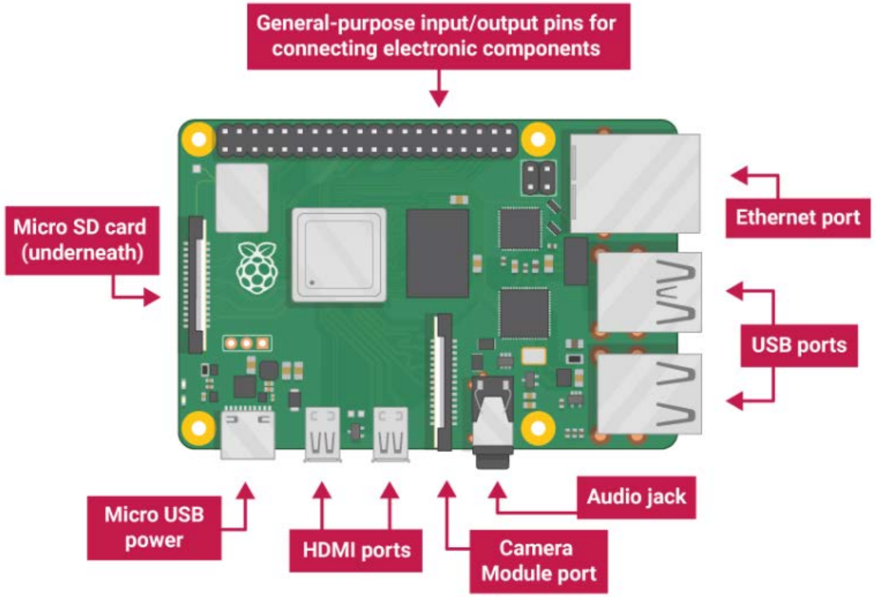
- Takes time
- So makes sense to store any data locally
- And retrieve it later



```
Searching for songs by Snoop Dogg...

Song 1: "Gin and Juice"
Song 2: "Drop It Like It’s Hot"
Song 3: "Ain’t No Fun (If the Homies Can’t Have None)"
Song 4: "Murder Was the Case (Death After Visualizing Eternity)"

Song 5: "Who Am I (What’s My Name)?"

Reached user-specified song limit (5).
Done. Found 5 songs.
```

What We Have Learned

- How to do AD Conversion and sample sensors
- How to smooth the data and visualise it
- How to create threads to handle sensor reading
- How to expand the I/O with a MUX
- How to use CLI tools in Python
- How to get data from the Pi to a local machine
- How to use an API


General-purpose input/output pins for connecting electronic components
Ethernet port
Micro SD card (underneath)
USB ports
Audio jack
Micro USB power
HDMI ports
Camera Module port

It’s Ten to Four

- If you would like to continue
  - Email me
  - Buy yourself a Pi4
  - Find a project for ECTS on a course and we can support the tech side

- Before we leave
  - Delete the directory you made – ***rm mydir***
  - Shutdown the Pi
  - PSU and Pi/Screen back in box
  - Tidy up components and stuff
  - Put the rc.local file back to normal!

DTU Skylab Digital – Introduction Workshop

Thanks for today!

Feedback/Followup/Questions: benca@dtu.dk

skylab.dtu.dk