https://www.halvorsen.blog

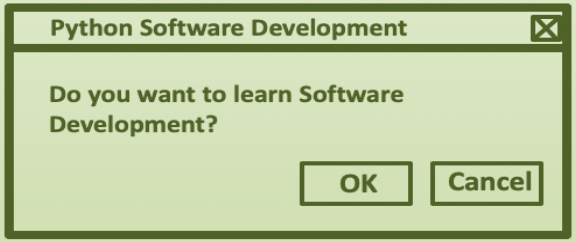
Raspberry SPI and I2C with Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

Python for Software Development

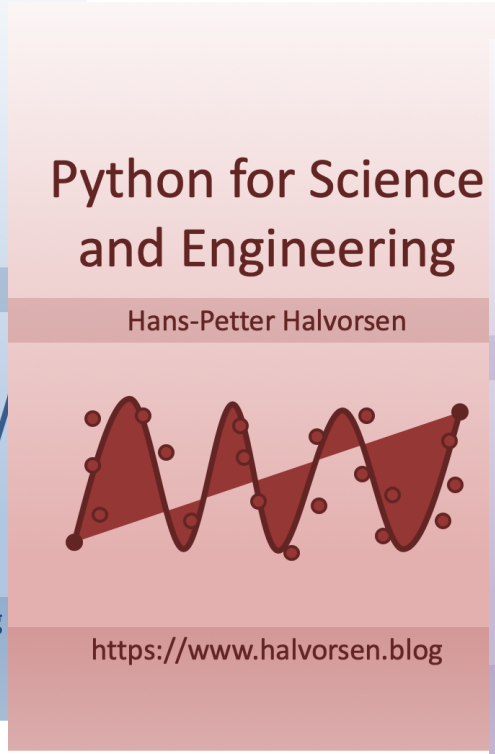
Hans-Petter Halvorsen

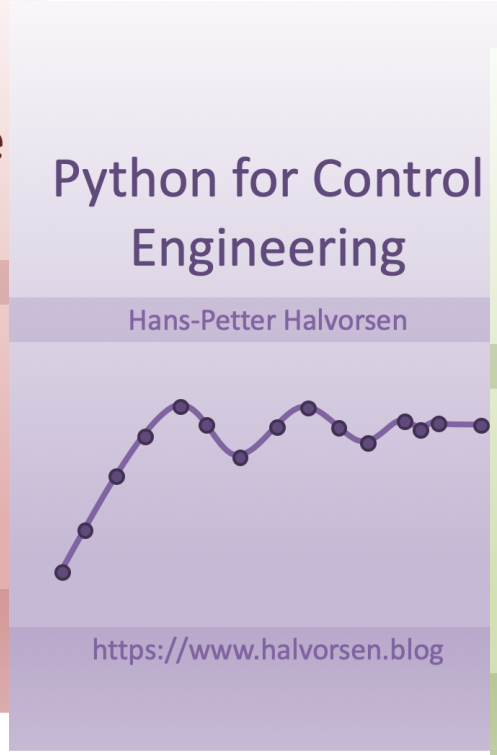


https://www.halvorsen.blog

https://www.halvorsen.blog/documents/programming/python/

Additional Python Resources



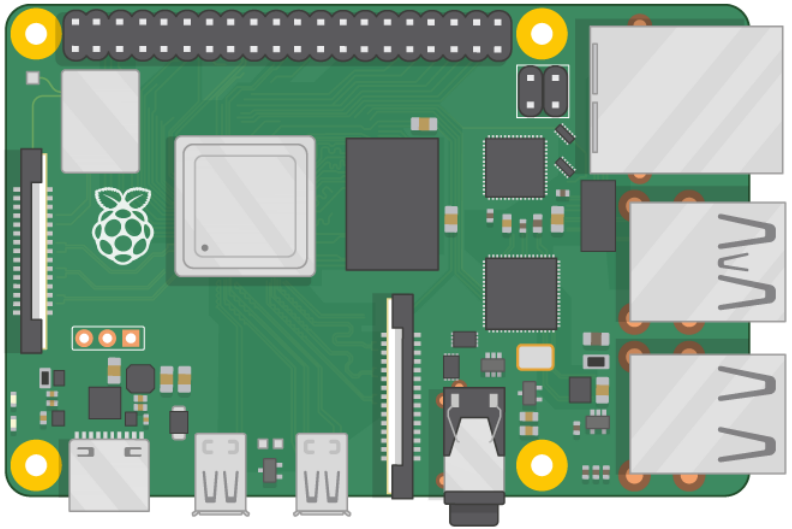


https://www.halvorsen.blog/documents/programming/python/

Contents

Raspberry Pi

Raspberry Pi is a tiny (about 9x6cm), low-cost ($35+), single-board computer that supports embedded Linux operating systems



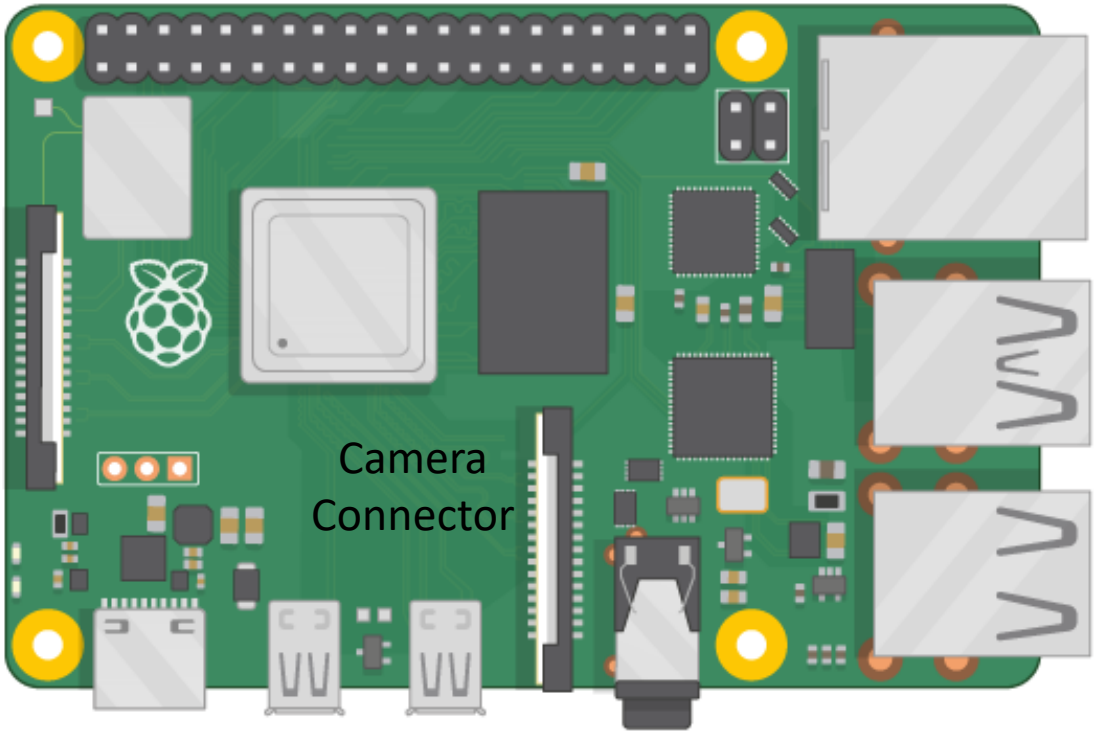
The recommended Operating System is called Raspberry Pi OS (Linux based)

https://www.raspberrypi.org

Raspberry Pi

GPIO Pins

SD Card
(the Back)



Ethernet

Camera
Connector

USB A x 4

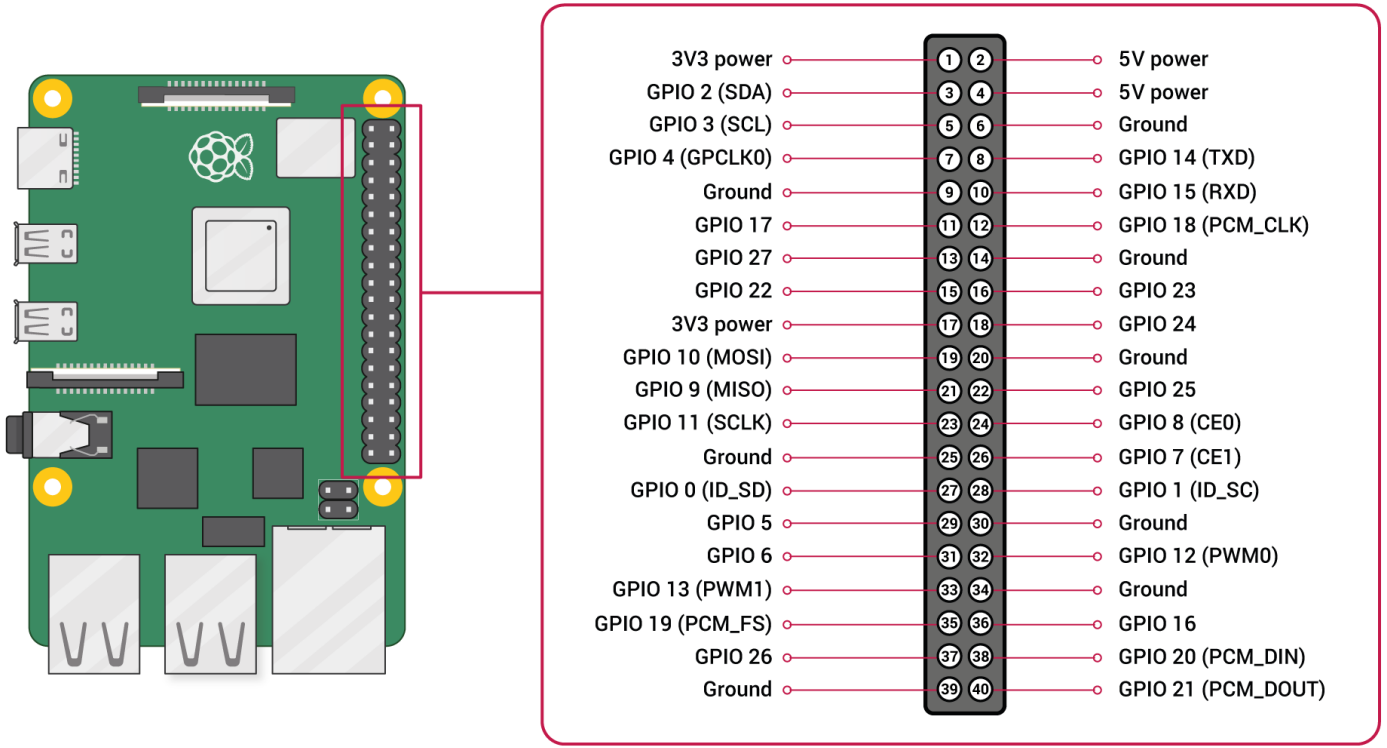
Power Supply (USB C) micro HDMI x 2

https://www.halvorsen.blog

Raspberry PI GPIO

Hans-Petter Halvorsen

GPIO



A powerful feature of the Raspberry Pi is the GPIO (general-purpose input/output) pins. The Raspberry Pi has a 40-pin GPIO header as seen in the image

GPIO Features

The GPIO pins are Digital Pins which are either True (+3.3V) or False (0V). These can be used to turn on/off LEDs, etc.

The Digital Pins can be either Output or Input.

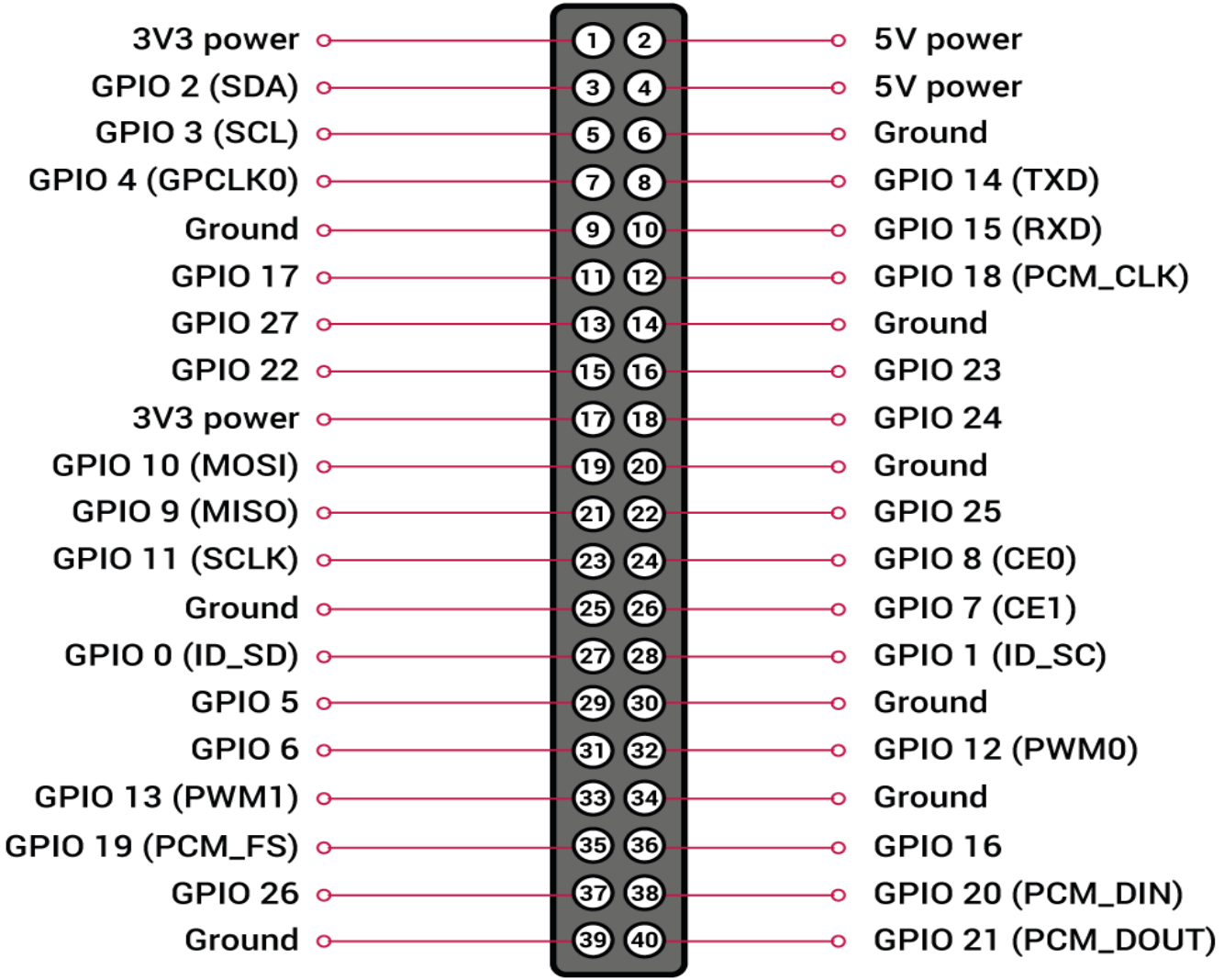
In addition, some of the pins also offer some other Features:

- PWM (Pulse Width Modulation)

Digital Buses (for reading data from Sensors, etc.):

- SPI
- I2C

GPIO


3V3 power
GPIO 2 (SDA)
GPIO 3 (SCL)
GPIO 4 (GPCLK0)
Ground
GPIO 17
GPIO 27
GPIO 22
3V3 power
GPIO 10 (MOSI)
GPIO 9 (MISO)
GPIO 11 (SCLK)
Ground
GPIO 0 (ID_SD)
GPIO 5
GPIO 6
GPIO 13 (PWM1)
GPIO 19 (PCM_FS)
GPIO 26
Ground
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
5V power
5V power
Ground
GPIO 14 (TXD)
GPIO 15 (RXD)
GPIO 18 (PCM_CLK)
Ground
GPIO 23
GPIO 24
Ground
GPIO 25
GPIO 8 (CE0)
GPIO 7 (CE1)
GPIO 1 (ID_SC)
Ground
GPIO 12 (PWM0)
Ground
GPIO 16
GPIO 20 (PCM_DIN)
GPIO 21 (PCM_DOUT)

https://www.halvorsen.blog

GPIO with Python

Hans-Petter Halvorsen

GPIO Zero

- The GPIO Zero Python Library can be used to communicate with GPIO Pins
- The GPIO Zero Python Library comes preinstalled with the Raspberry Pi OS (so no additional installation is necessary)

Resources:

- https://www.raspberrypi.org/documentation/usage/gpio/python/
- https://pypi.org/project/gpiozero/
- https://gpiozero.readthedocs.io/en/stable/
- https://gpiozero.readthedocs.io/en/stable/recipes.html

RPi.GPIO

- Rpi.GPIO is a module controlling the GPIO pins on the Raspberry Pi
- RPi.GPIO is a more “low-level“ Python Library than GPIO Zero. Actually, GPIO Zero is using RPi.GPIO
- The RPi.GPIO Python Library comes preinstalled with the Raspberry Pi OS (so no additional installation is necessary)

https://pypi.org/project/RPi.GPIO/

Digital Bus Interfaces

- SPI
- I2C

- These are synchronous serial interfaces, which means it relies on a shared clock signal to synchronize data transfer between devices

SPI vs. I2C

SPI

- 4-Wire Protocol
- SPI supports full-duplex. Data can be sent and received at the same time
- Higher data transfer rate than I2C
- Complex wiring if more than one Slave

I2C

- 2-Wire Protocol
- SPI supports only half-duplex. Data cannot be sent and received at the same time
- Lower data transfer rate than SPI
- Multiple Slaves are easier

https://www.halvorsen.blog

SPI

Serial Peripheral Interface (SPI)

Hans-Petter Halvorsen

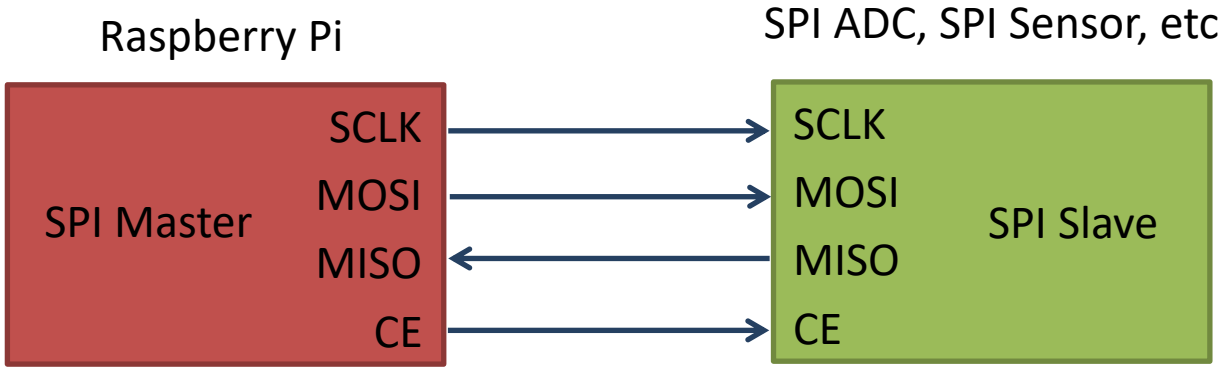
SPI

- Serial Peripheral Interface (SPI)
- 4–Wire Protocol (SCLK, CE, MOSI, MISO)
- SPI is an interface to communicate with different types of electronic components like Sensors, Analog to Digital Converts (ADC), etc. that supports the SPI interface
- Thousands of different Components and Sensors supports the SPI interface

https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/

SPI

SPI devices communicate in full duplex mode using a master-slave architecture with a single master

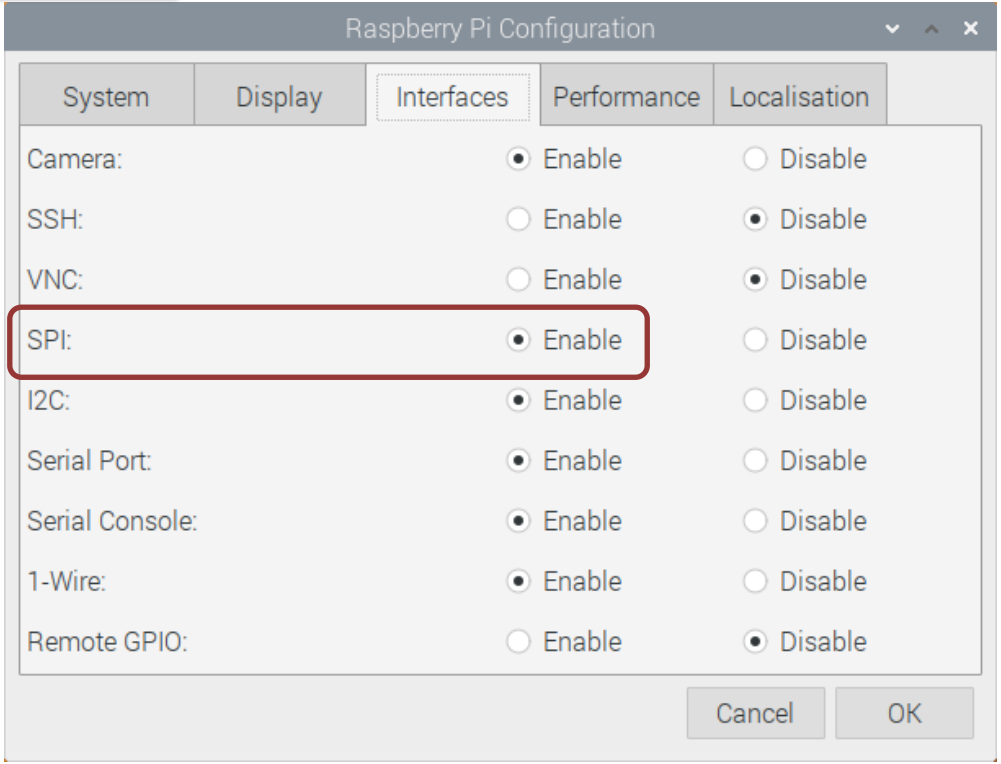


The SPI bus specifies four logic signals:

- **SCLK**: Serial Clock (output from master)
- **MOSI**: Master Out Slave In (data output from master)
- **MISO**: Master In Slave Out (data output from slave)
- **CE** (often also called SS - Slave Select): Chip Select (often active low, output from master)

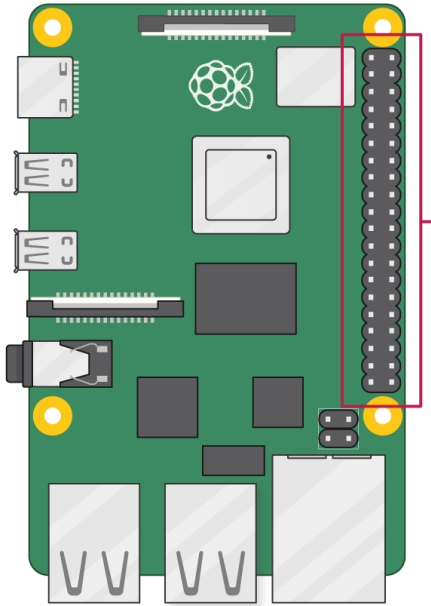
Access SPI on Raspberry Pi

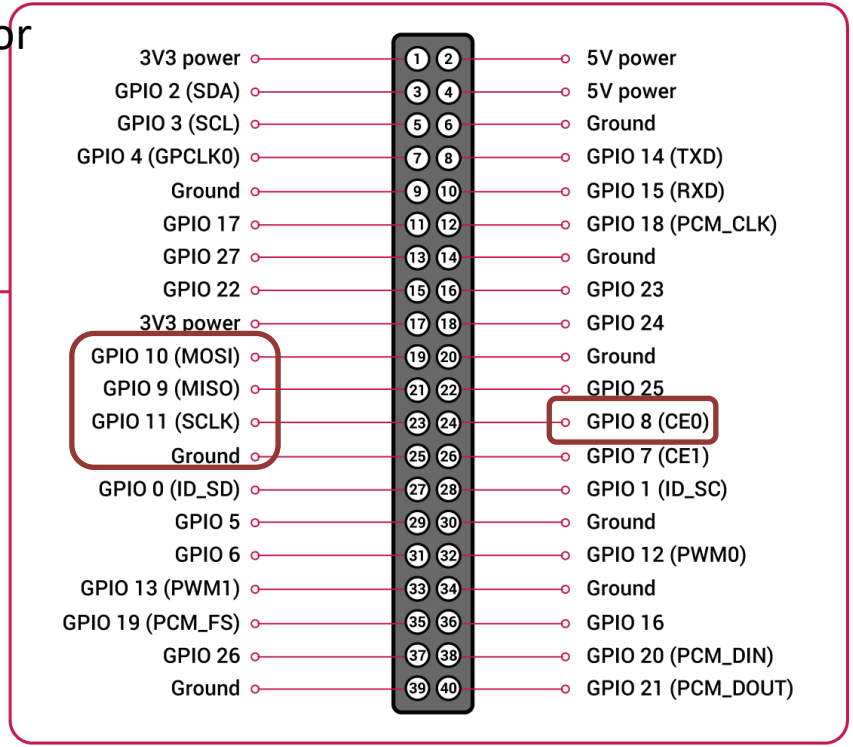
You need to Enable SPI on the Raspberry Pi



SPI Wiring on Raspberry Pi

GPIO 40 pins Connector





https://www.halvorsen.blog

ADC

Analog to Digital Converter

Hans-Petter Halvorsen

ADC

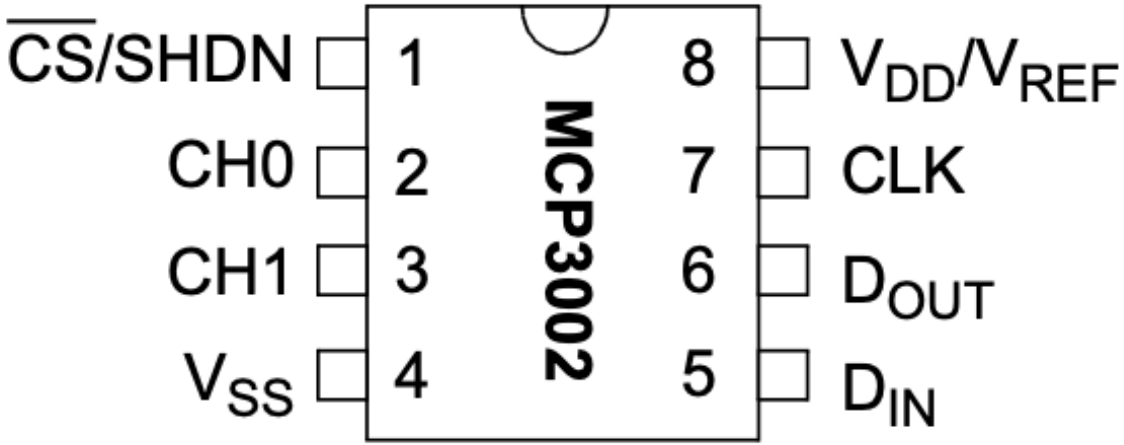
- The Raspberry Pi has only Digital pins on the GPIO connector
- If you want to use an Analog electric component or an Analog Sensor together with Raspberry Pi, you need to connect it through an external ADC chip
- ADC – Analog to Digital Converter

https://en.wikipedia.org/wiki/Analog-to-digital_converter

MCP3002 ADC chip

The MCP3002 is a 10-bit analog to digital converter with 2 channels (0-1).

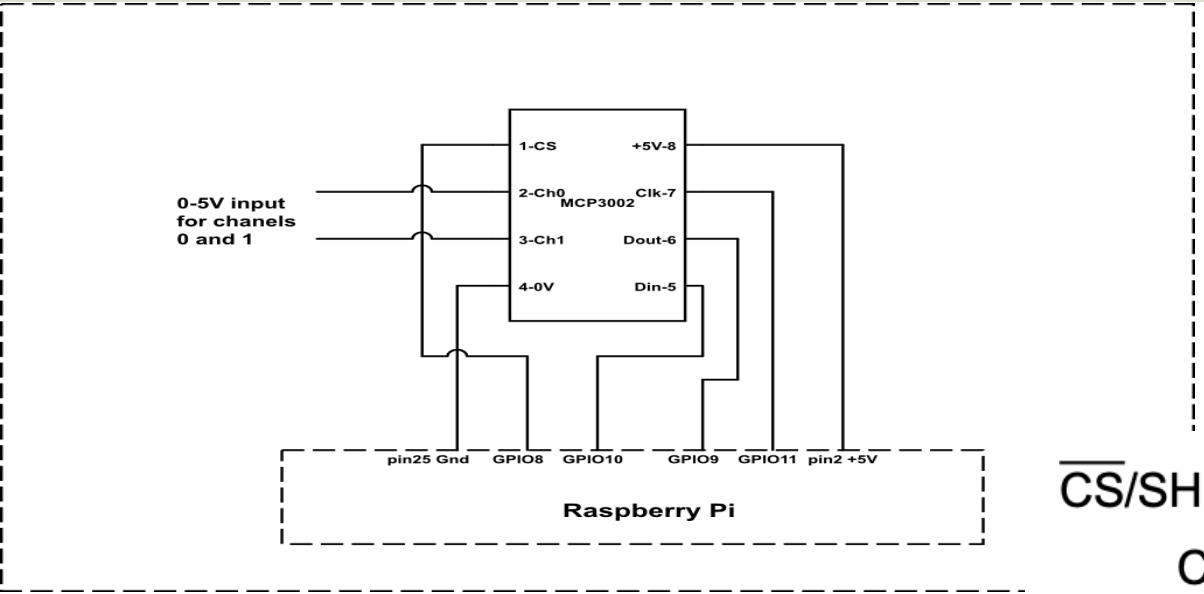
The MCP3002 uses a SPI Interface

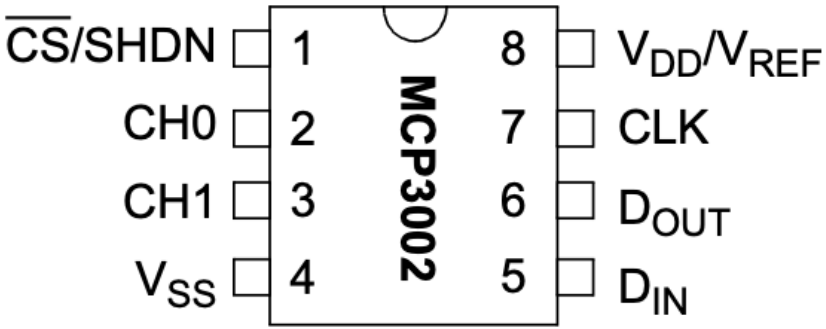


http://ww1.microchip.com/downloads/en/DeviceDoc/21294E.pdf

https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/experiment-3-spi-and-analog-input

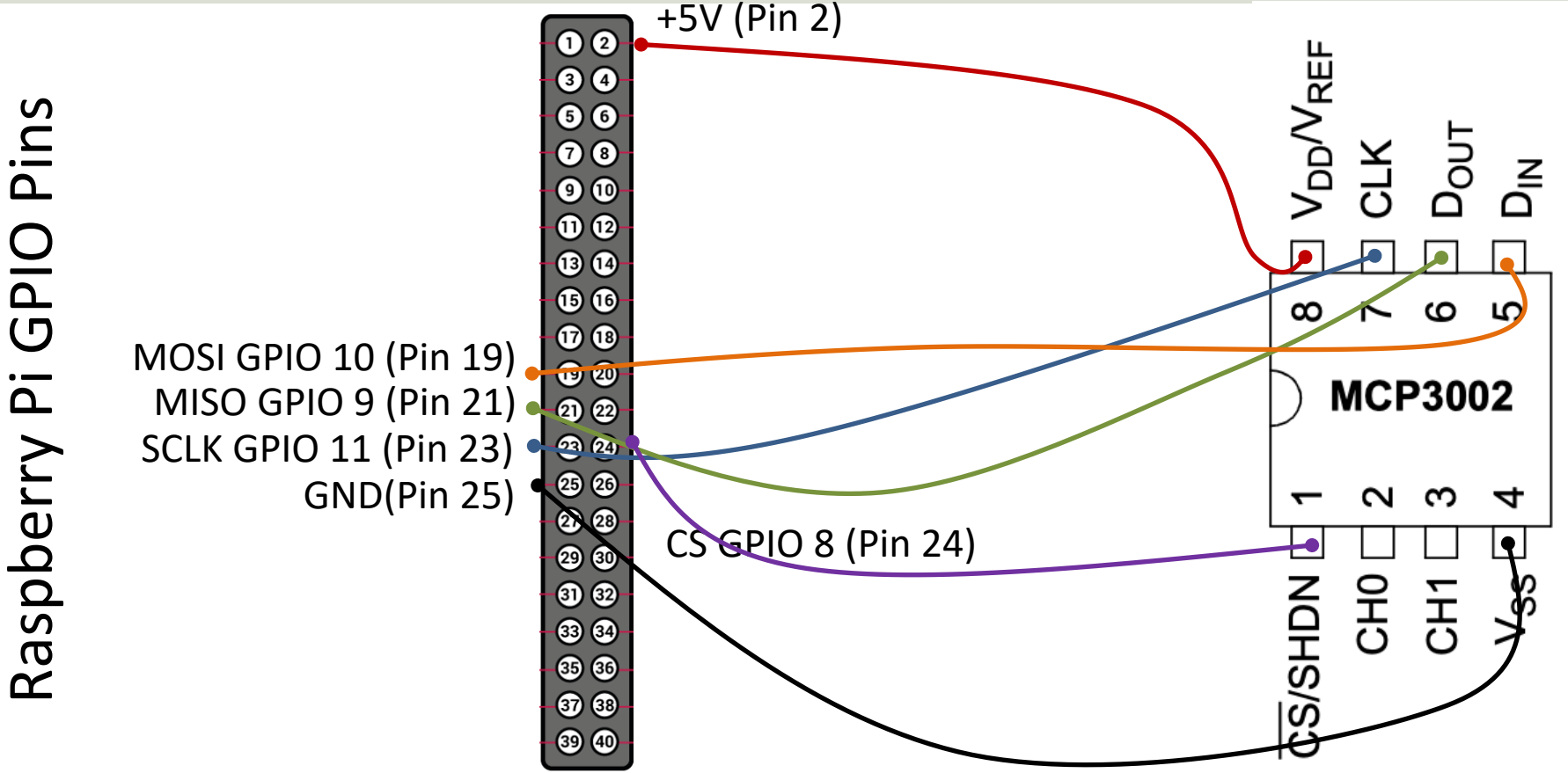
Wiring





https://sites.google.com/a/joekamphaus.net/raspberry-pi-spi-interface-to-mcp3002/

Wiring


Raspberry Pi GPIO Pins
+5V (Pin 2)
MOSI GPIO 10 (Pin 19)
MISO GPIO 9 (Pin 21)
SCLK GPIO 11 (Pin 23)
GND(Pin 25)
CS GPIO 8 (Pin 24)
MCP3002
VDD/VREF
CLK
DOUT
DIN
CS/SHDN
CH0
CH1
VSS

GPIO Zero and MCP3002

```
gpiozero.MCP3002(channel=0, differential=False, max_voltage=3.3, **spi_args)
```

channel
The channel to read data from. The MCP3008/3208/3304 have 8 channels (0-7), while the MCP3004/3204/3302 have 4 channels (0-3), the MCP3002/3202 have 2 channels (0-1), and the MCP3001/3201/3301 only have 1 channel.

differential
If True, the device is operated in differential mode. In this mode one channel (specified by the channel attribute) is read relative to the value of a second channel (implied by the chip's design).

Please refer to the device data-sheet to determine which channel is used as the relative base value (for example, when using an MCP3008 in differential mode, channel 0 is read relative to channel 1).

value
The current value read from the device, scaled to a value between 0 and 1 (or -1 to +1 for certain devices operating in differential mode).

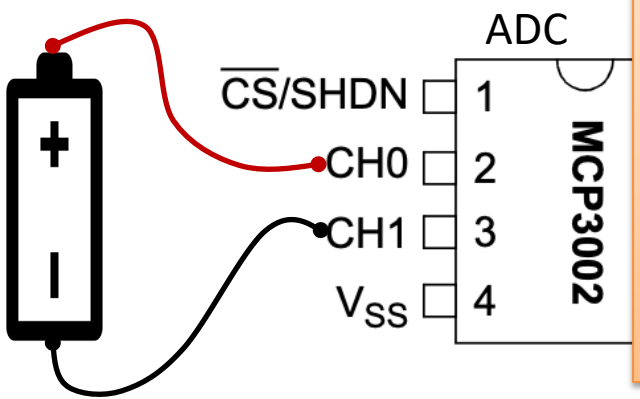
https://gpiozero.readthedocs.io/en/stable/api_spi.html

Read Data from ADC

For test purpose we start by wiring a 1.5V Battery to the CH0 (+) and CH1(-) pins on the ADC

Note! WE have set **differential=True** (meaning CH0 is “+“ and CH1 is “-“)

1.5V Battery



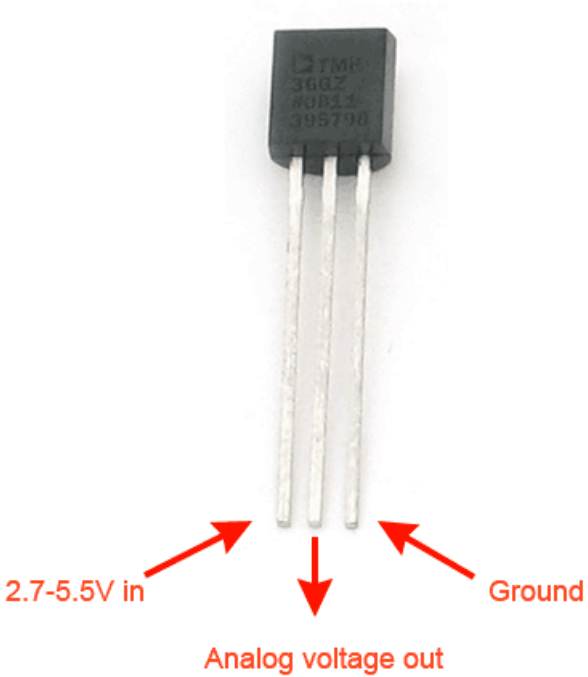

```
from gpiozero import MCP3002
from time import sleep

adc = MCP3002(channel=0, differential=True)

N = 20

for x in range(N):
    adcdata = adc.value #Value between 0 and 1
    #print(adcdata)
    voltvalue = adcdata * 5 #Value between 0 and 5V
    print(voltvalue)
    sleep(1)
```

TMP36

Temperature Sensor

Hans-Petter Halvorsen

TMP36 Temperature Sensor

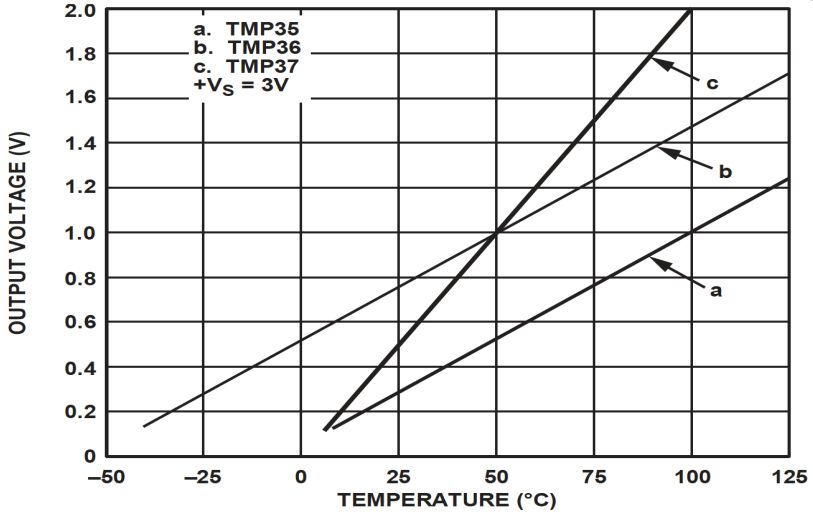


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

They use the fact as temperature increases, the voltage across a diode increases at a known rate.

https://learn.adafruit.com/tmp36-temperature-sensor

TMP36 Temperature Sensor



This gives:

$$y - 25 = \frac{50 - 25}{1 - 0.75}(x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

Convert form Voltage (V) to degrees Celsius

From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25°C)$$
$$(x_2, y_2) = (1V, 50°C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

We can find a and b using the following known formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Measure temperature with an ADC

TMP36 Temperature Sensor

Wire a TMP36 temperature sensor to the first channel of an MCP3002 analog to digital converter and the other pins to +5V and GND

```
from gpiozero import MCP3002
from time import sleep

adc = MCP3002(channel=0, differential=False)

N = 10

for x in range(N):
    adcdata = adc.value #Value between 0 and 1
    #print(adcdata)

    voltvalue = adcdata * 5 #Value between 0V and 5V
    #print(voltvalue)

    tempC = 100*voltvalue-50 #Temperature in Celsius
    tempc = round(tempC,1)
    print(tempC)

    sleep(1)
```

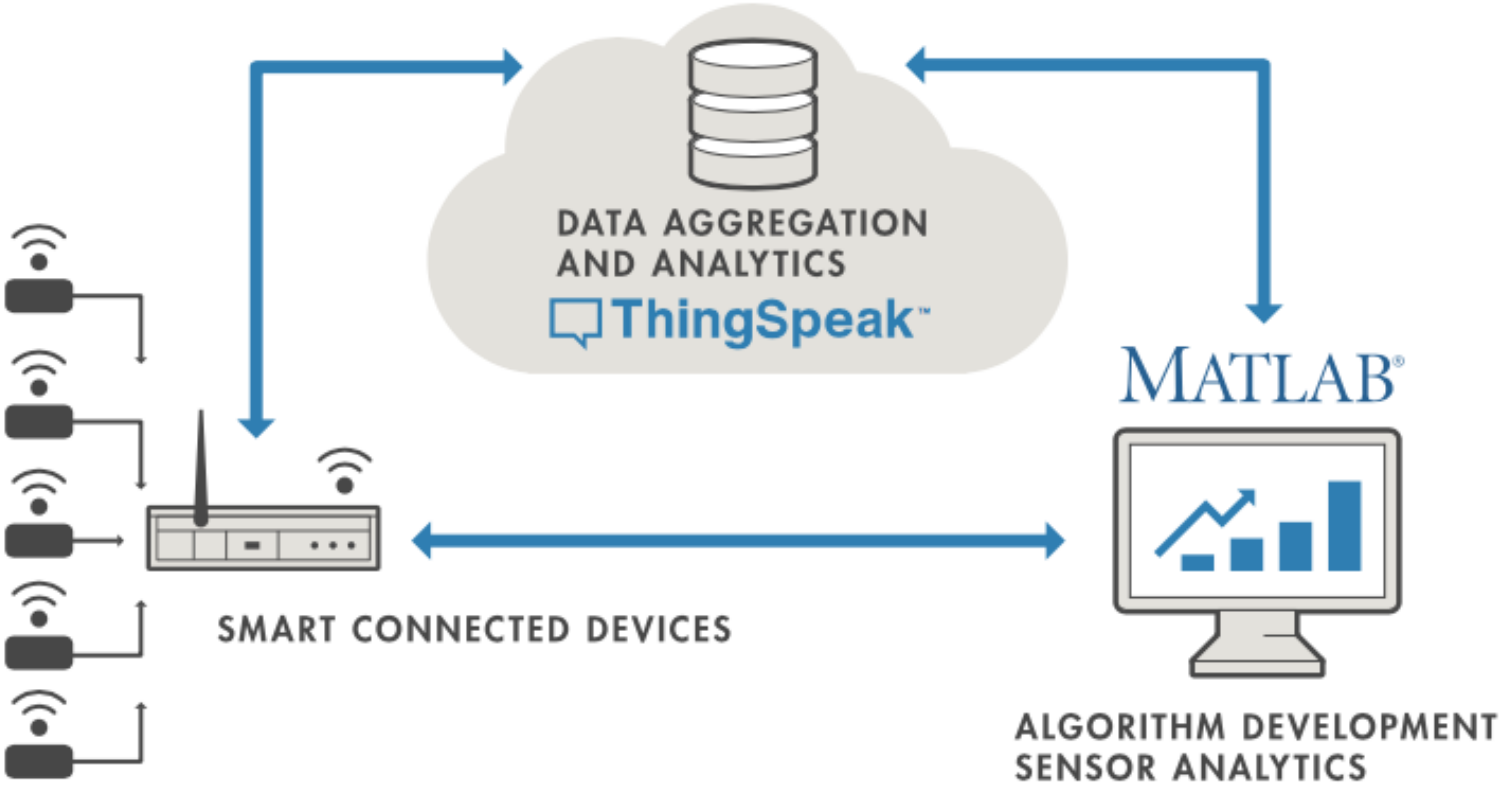
ThingSpeak

Hans-Petter Halvorsen

ThingSpeak

- ThingSpeak is an IoT analytics platform service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- The ThingSpeak service also lets you perform online analysis and act on your data. Sensor data can be sent to ThingSpeak from any hardware that can communicate using a REST API
- ThingSpeak has a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications (it also has MQTT API).
- https://thingspeak.com
- Python Library for ThingSpeak: https://pypi.org/project/thingspeak/

ThingSpeak


DATA AGGREGATION
AND ANALYTICS
ThingSpeak™
MATLAB®
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT
SENSOR ANALYTICS

ThingSpeak Write

```
import thingspeak
import time

channel_id = xxxxxx
write_key  = "xxxxxxxxxxxxxxxx"

channel = thingspeak.Channel(id=channel_id, api_key=write_key)

N = 10
for x in range(N):
      temperature = 24
      response = channel.update({'field1': temperature})
      time.sleep(15)
```

https://thingspeak.readthedocs.io/en/latest/api.html

A Free ThingSpeak Channel can only be updated every 15 sec

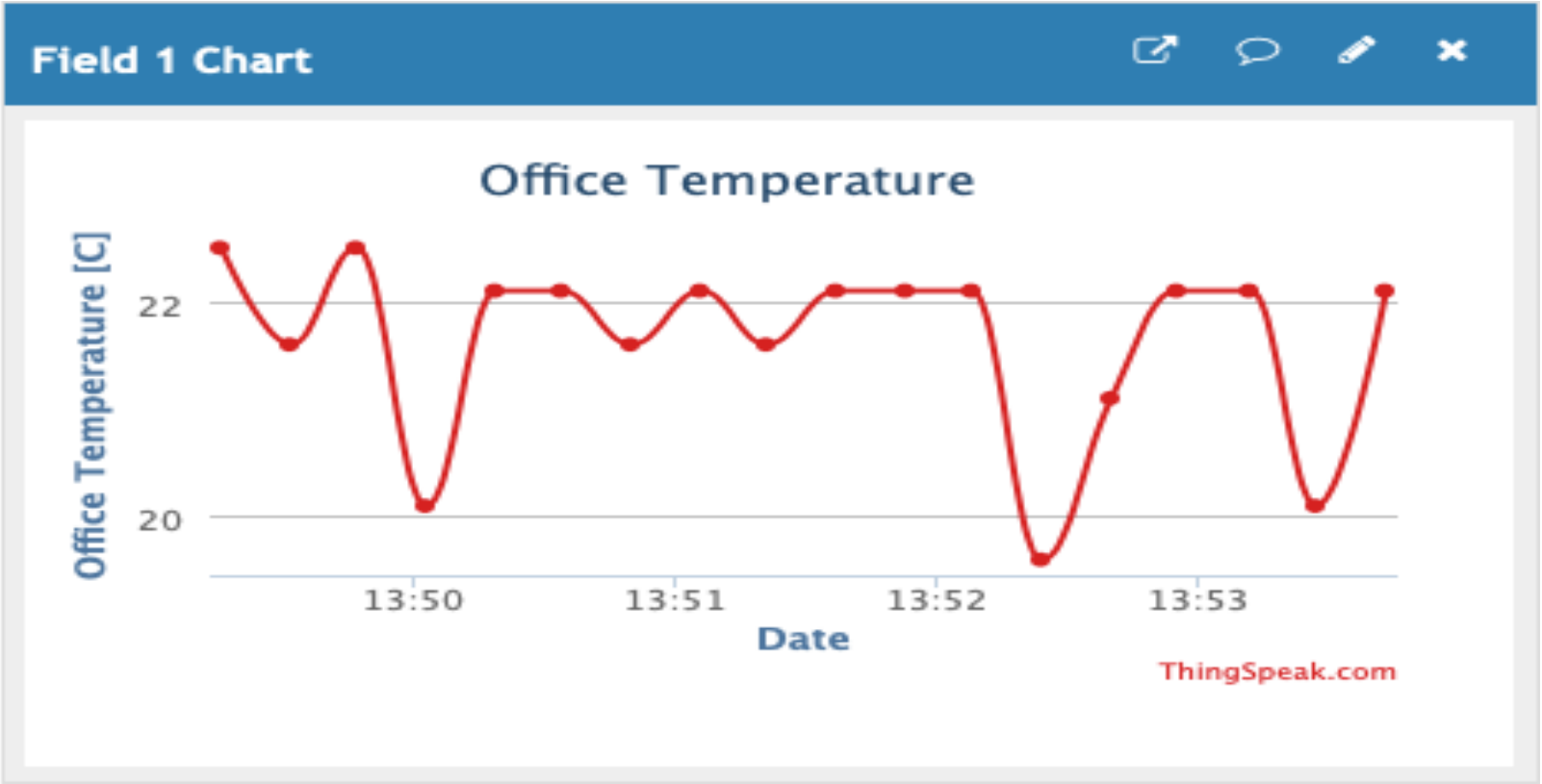
Write TMP36 Data

```
import thingspeak
import time
from gpiozero import MCP3002

adc = MCP3002(channel=0, differential=False)

channel_id = xxxxxxx
write_key  = ”xxxxxxxxxxxxxxxxx”

channel = thingspeak.Channel(id=channel_id, api_key=write_key)

N = 10
for x in range(N):
    #Get Sensor Data
    adcdata = adc.value #Scaled Value between 0 and 1
    voltvalue = adcdata * 5 # Value between 0V and 5V
    tempC = 100*voltvalue-50 # Temperature in Celsius
    tempC = round(tempC,1)
    print(tempC)

    #Write to ThingSpeak
    response = channel.update({'field1': tempC})
    time.sleep(15)
```

A Free ThingSpeak Channel can only be updated every 15 sec

Write TMP36 Data

Here we see the Temperature Data in ThingSpeak:



ThingSpeak Read

```
import thingspeak

channel_id = xxxxxx
read_key   = ”xxxxxxxxxxxxxxxx"

channel = thingspeak.Channel(id=channel_id, api_key=read_key)

#data = channel.get({})
data = channel.get_field({”field1"})

print(data)
```

https://thingspeak.readthedocs.io/en/latest/api.html

https://www.halvorsen.blog

I2C

Inter Integrated Circuit

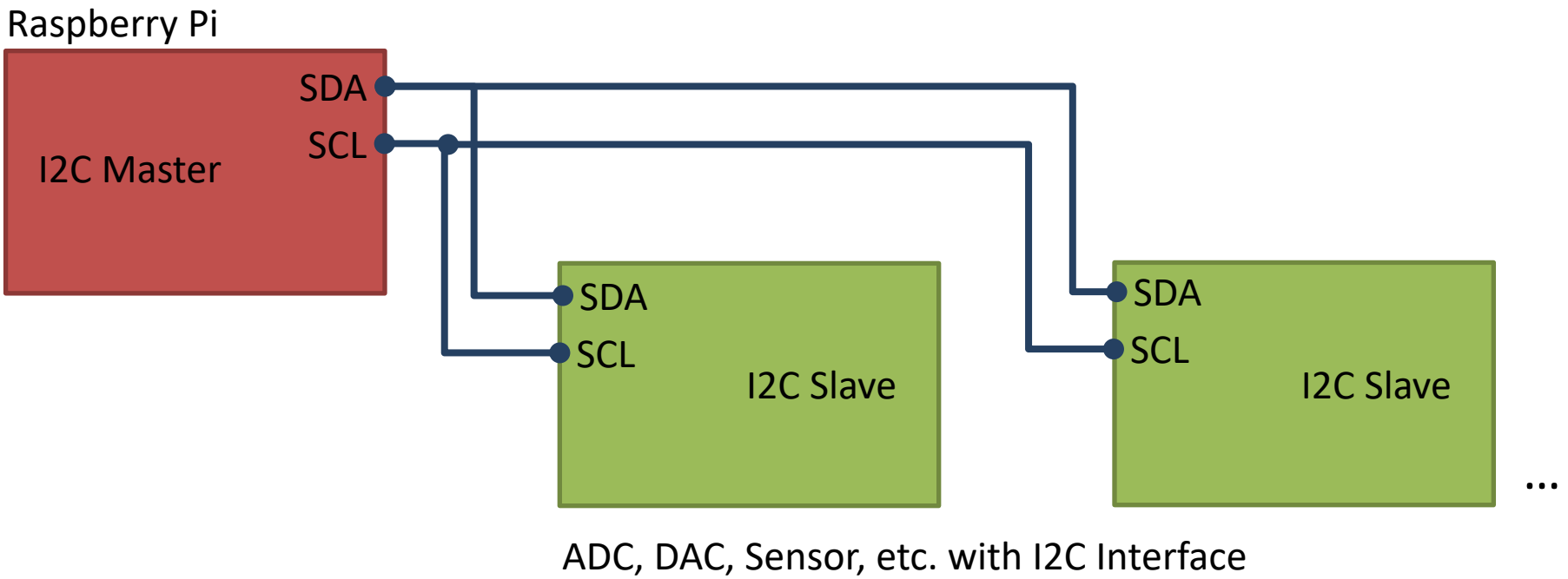
Hans-Petter Halvorsen

I2C

- I2C is a multi-drop bus
- 2-Wire Protocol (SCL + SDA)
- Multiple devices can be connected to the I2C pins on the Raspberry Pi
- Each device has its own unique I2C address

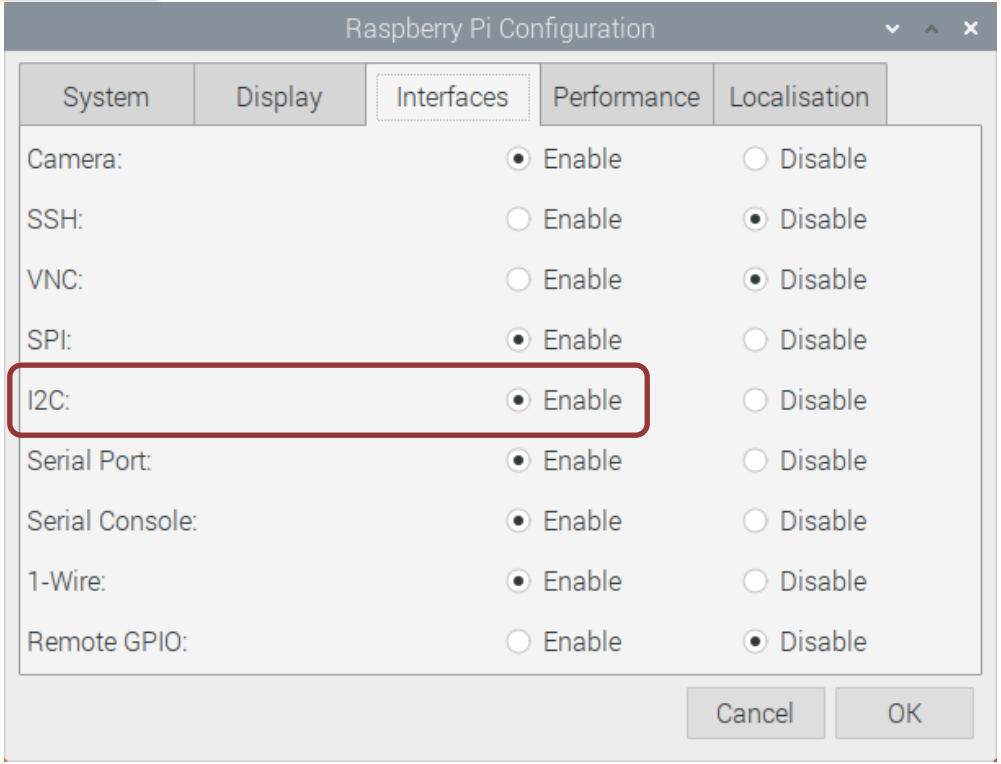
I2C

Multiple devices can be connected to the I2C pins on the Raspberry Pi
Master – Device that generates the clock and initiates communication with slaves
Slave – Device that receives the clock and responds when addressed by the master.



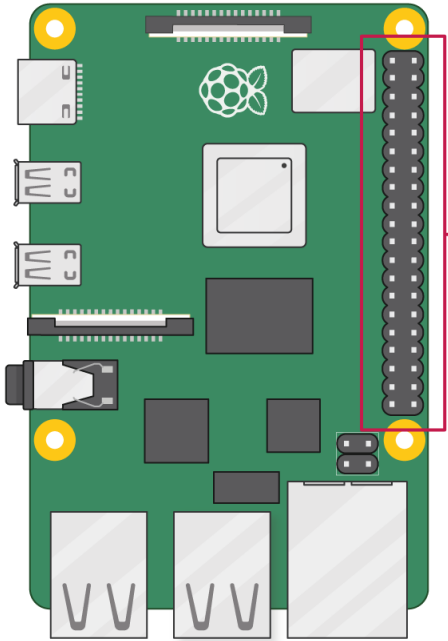
Access I2C on Raspberry Pi

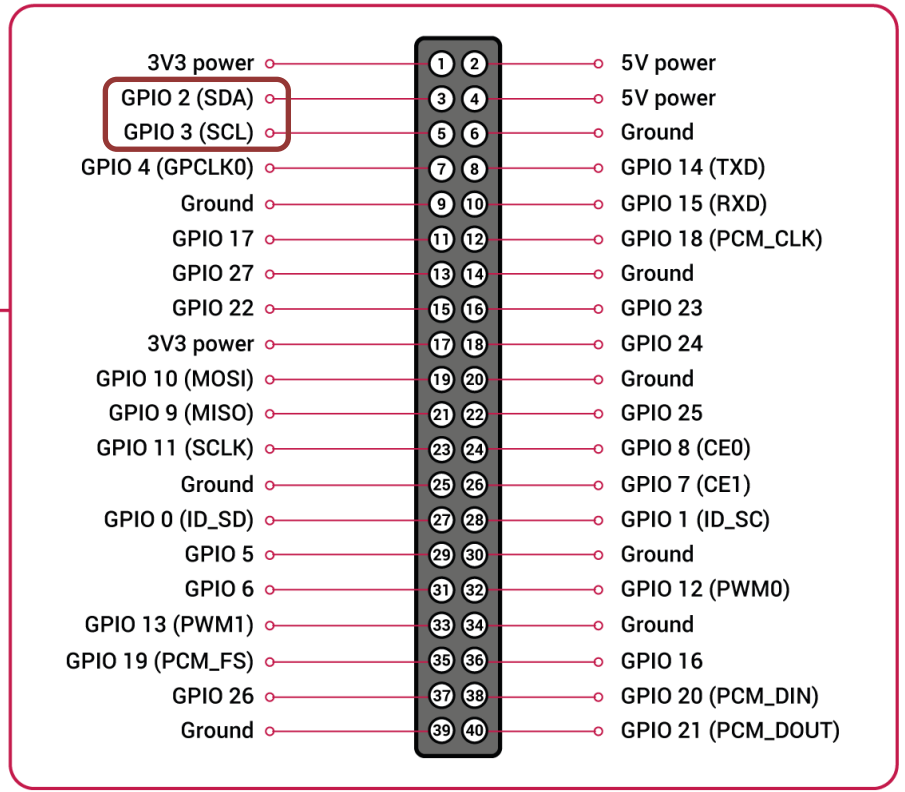
You need to Enable I2C on the Raspberry Pi



I2C Wiring on Raspberry Pi

GPIO 40 pins Connector





Note! The I2C pins include a fixed 1.8 kΩ pull-up resistor to 3.3v.

Detecting I2C Devices

Install I2C Tools on the Raspberry Pi:

```
sudo apt-get install -y i2c-tools
```

Detecting and Find the Address of the I2C Device using the `i2cdetect` command:

```
sudo i2cdetect -y 1
```

We can read and write its registers using `i2cget`, `i2cset` and `i2cdump`

Example:

```
sudo i2cget -y 1 0x48
```

Device Address

GPIO Python Libraries

- GPIO Zero
  - https://pypi.org/project/gpiozero/
- RPi.GPIO
  - https://pypi.org/project/RPi.GPIO/
- smbus (used for I2C communication)

smbus Python Library

SMBus (System Management Bus) is a subset from the I2C protocol

You can access I2C devices from Python using the smbus library:

```
import smbus
DEVICE_BUS = 1
DEVICE_ADDR = 0x15
bus = smbus.SMBus(DEVICE_BUS)

command = 0x00
value = 0x01
bus.write_byte_data(DEVICE_ADDR, command, value)

data = bus.read_byte_data(DEVICE_ADDR, command)
```

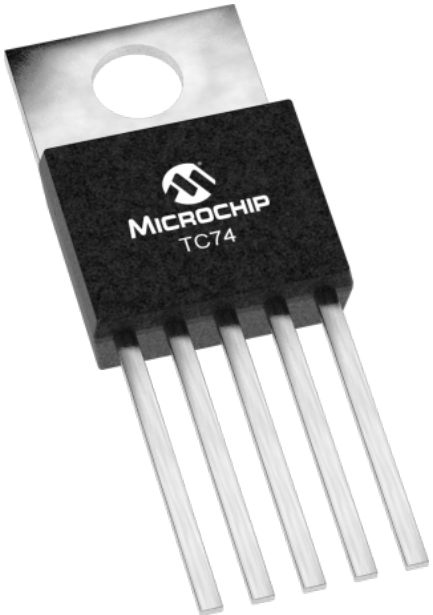
https://pinout.xyz/pinout/i2c

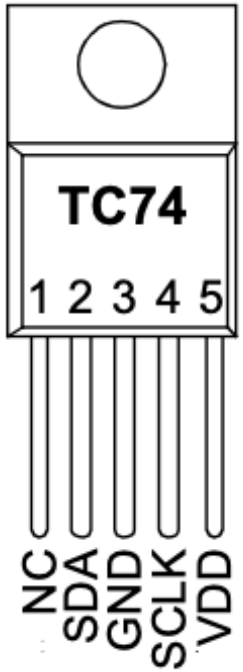
https://raspberry-projects.com/pi/programming-in-python/i2c-programming-in-python/using-the-i2c-interface-2

TC74 Temperature Sensor

SMBus/I2C Interface

TC74A0-5.0VAT



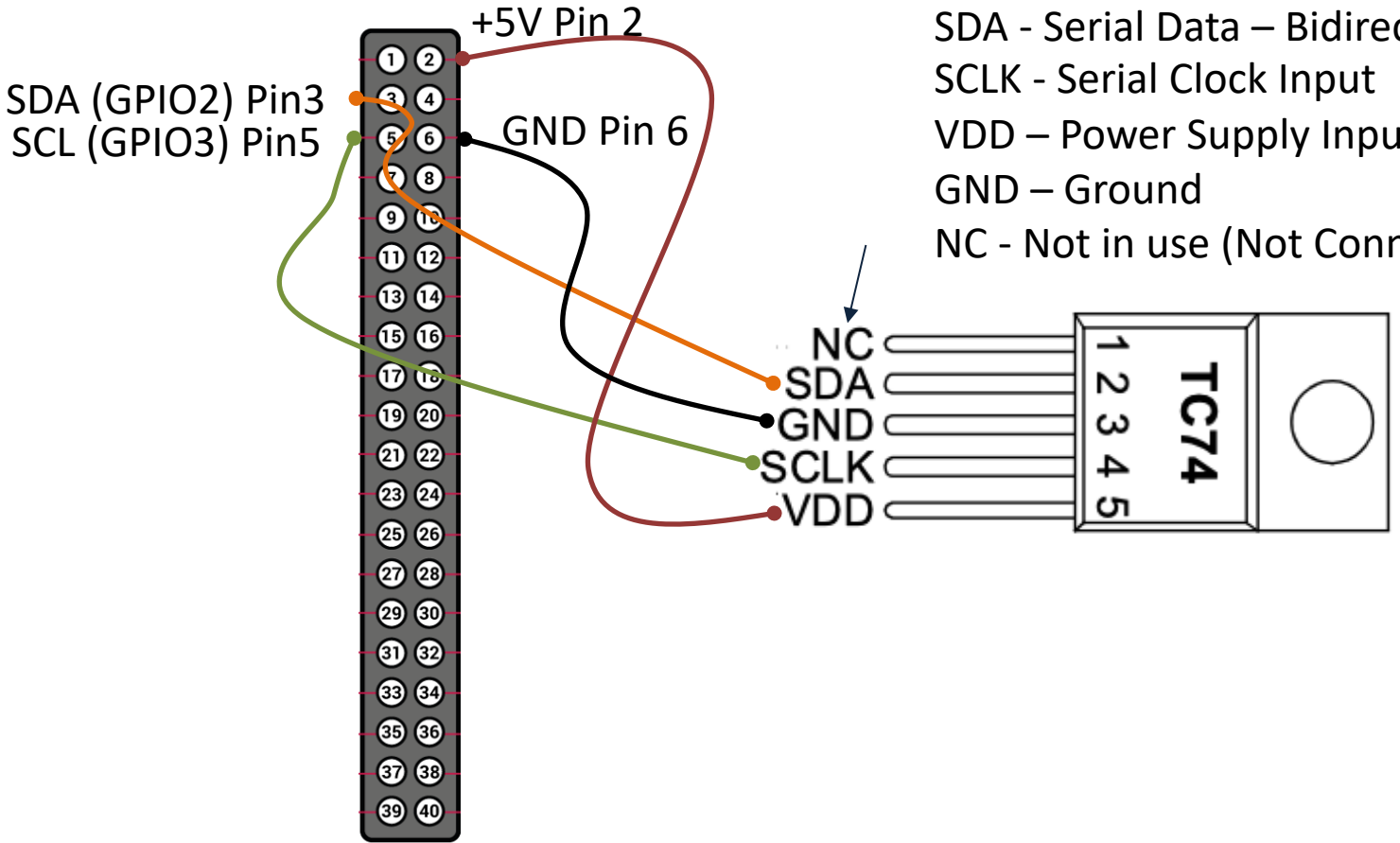


- The TC74 acquires and converts temperature information from its onboard solid-state sensor with a resolution of ±1°C.
- It stores the data in an internal register which is then read through the serial port.
- The system interface is a slave SMBus/I2C port, through which temperature data can be read at any time.

Datasheet: https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf

TC74 Wiring

Raspberry Pi GPIO Pins



TC74 Testing

Running the following in the Terminal:

```
sudo i2cdetect -y 1
```

This gives the TC74 address `0x48`

Running the following in the Terminal:

```
sudo i2cget -y 1 0x48
```

This gives the values:

```
0x16 -> 22
0x17 -> 23
0x18 -> 24
0x19 -> 25
```

(while holding my fingertips on the sensor)

TC74 Python Code Example

This code shows the basic reading of the Sensor Data.

You can add a For Loop or a While Loop for reading Sensor Data at specific intervals.

You can plot the Data using matplotlib, save data to a File or send data to a cloud service like ThingSpeak, etc.

```
import smbus

channel = 1
address = 0x48

bus = smbus.SMBus(channel)

data = bus.read_byte_data(address, 0)
print(data)
```

Or just:

```
data = bus.read_byte(address)
print(data)
```

This gives the Temperature Value in Degrees Celsius, e.g., 22

BME280

- BME280 is a Digital Humidity, Pressure and Temperature Sensor from Bosch
- The sensor provides both SPI and I2C interfaces
- Adafruit, Grove Seeed, SparkFun, etc. have breakout board bords for easy connection to Arduino, Raspberry Pi, etc.
- The Price for these breakout boards are $1-20 depending on where you buy these (ebay, Adafruit, Sparkfun, …)

BME280

- Humidity ±3% accuracy
- Barometric pressure ±1 hPa absolute accuraccy
- Temperature ±1.0°C accuracy

Datasheet:
https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

BME280

Adafruit

SparkFun

The size is about 2.5x2.5mm

So, to connect it to Raspberry Pi, you typically will use a breakout board

Grove Seeed

BME280 Python Libraries

There exists lots of BME280 libraries you can use for your BME280 Sensor

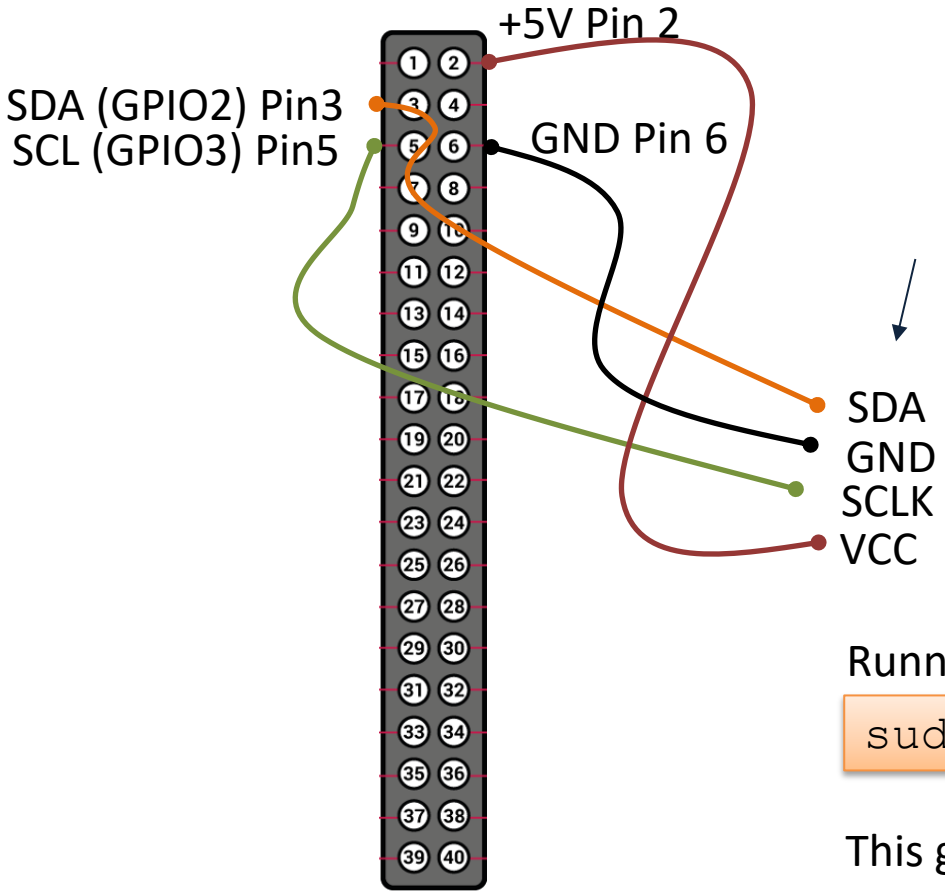
RPi.bme280:
https://pypi.org/project/RPi.bme280/

Here you find another Library:
https://www.raspberrypi-spy.co.uk/2016/07/using-bme280-i2c-temperature-pressure-sensor-in-python/

I have tested both these, and they are working fine.

BME280 Wiring

Raspberry Pi GPIO Pins



SDA - Serial Data – Bidirectional
SCLK - Serial Clock Input
VDD – Power Supply Input
GND – Ground
NC - Not in use (Not Connected)

Running the following in the Terminal:

```
sudo i2cdetect -y 1
```

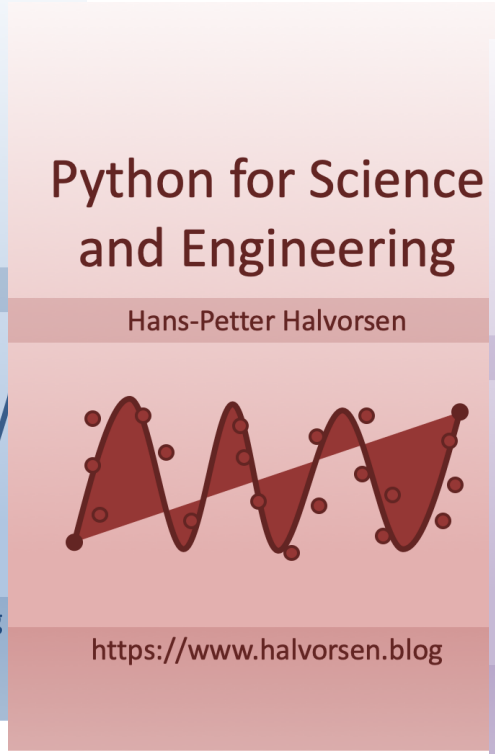
This gives the TC74 address `0x76`

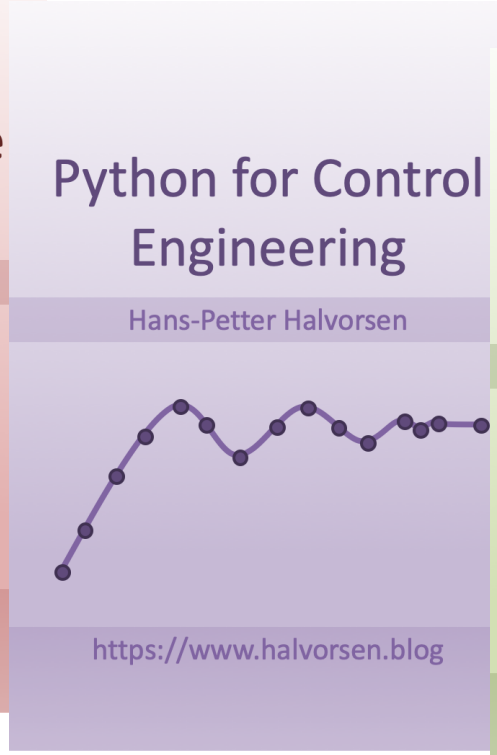
BME280 Example

```
import smbus2
import bme280

port = 1
address = 0x76
bus = smbus2.SMBus(port)

calibration_params = bme280.load_calibration_params(bus, address)

data = bme280.sample(bus, address, calibration_params)

print(data)

# Or Getting specific data:
print(data.id)
print(data.timestamp)
print(data.temperature)
print(data.pressure)
print(data.humidity)
```

https://pypi.org/project/RPi.bme280/

Additional Python Resources





https://www.halvorsen.blog/documents/programming/python/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog