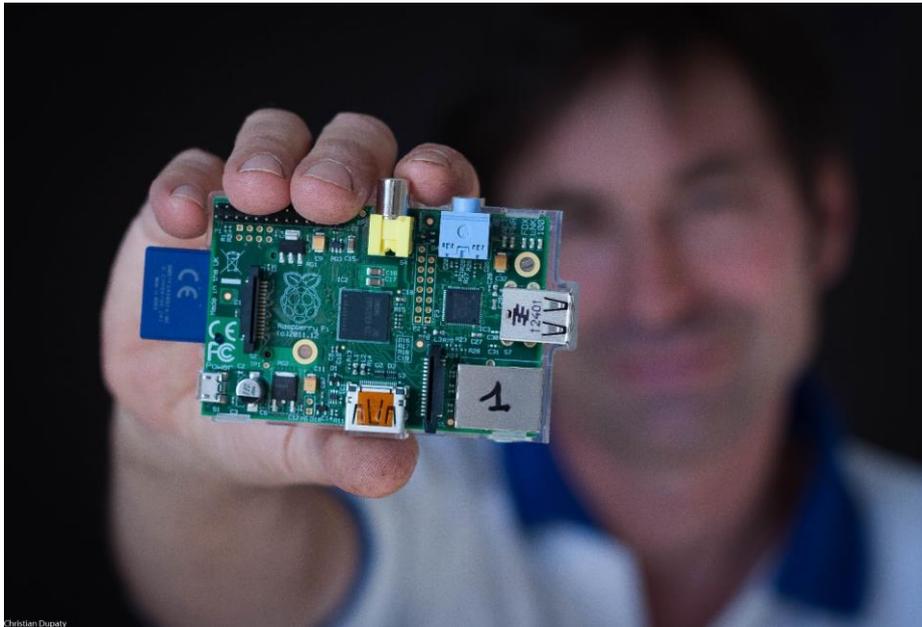


# RASPBERRY PI

## INSTALLATION-CONFIGURATION

## INTERFACES DE COMMUNICATIONS



I2C

Christian Dupaty

BTS Systèmes Numériques

Lycée Fourcade - Gardanne

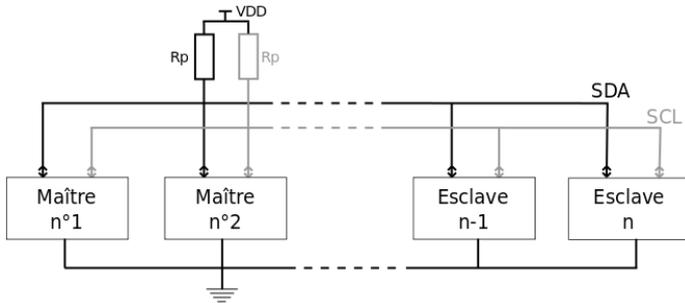
Académie d'Aix-Marseille



## 1) TP : I2C

Le bus I2C est un bus local permettant les échanges séries à courte distance entre un microcontrôleur et des périphériques (ADC, DAC, afficheur, mémoire, capteurs etc ...) ou un autre microcontrôleur, les échanges sont gérés par protocole avec adressage.

Le bus I2C est synchrone et half-duplex, sa technologie d'interfaces drains-ouverts le rend très résistant aux courts-circuits et permet une gestion simple des erreurs.



© wikipedia

Le niveau haut (1) est récessif, le (0) est dominant.

Un échange de données est toujours initié par le maître (ici Raspberry Pi).

Il commence par une Start Condition : SDA passe à 0 puis SCL passe à 0

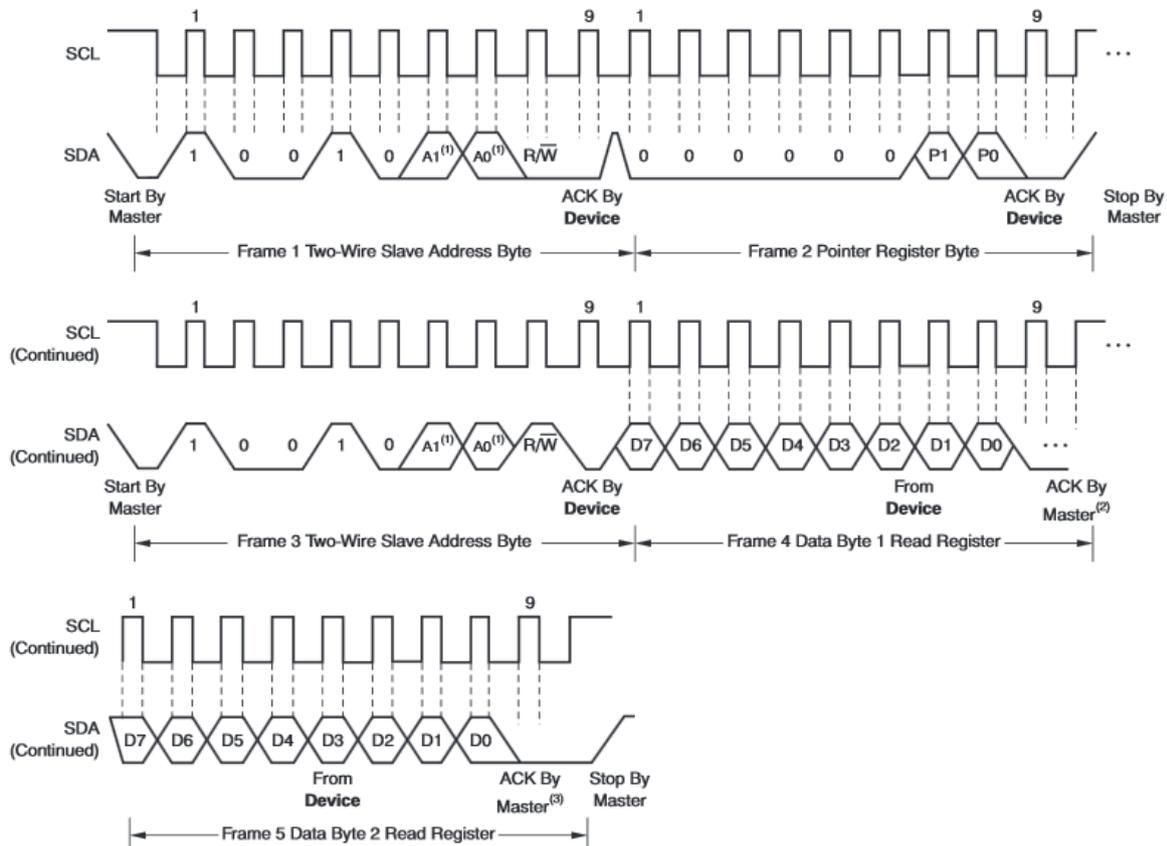
Puis l'adresse du destinataire : impaire pour une écriture, paire pour une lecture

Acquittement par le destinataire qui place un niveau bas sur la 9<sup>ième</sup> impulsion sur SCL.

Le sens de l'échange des données dépend de l'application, lecture pour un capteur, écriture pour un afficheur.

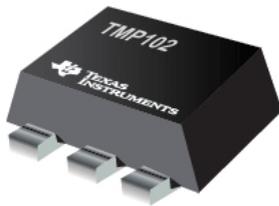
L'échange se termine par une condition stop, sda passe à 1 puis SCL passe à 1

Exemple de trame I2C, lecture d'un TMP102 (Texas-Instruments)

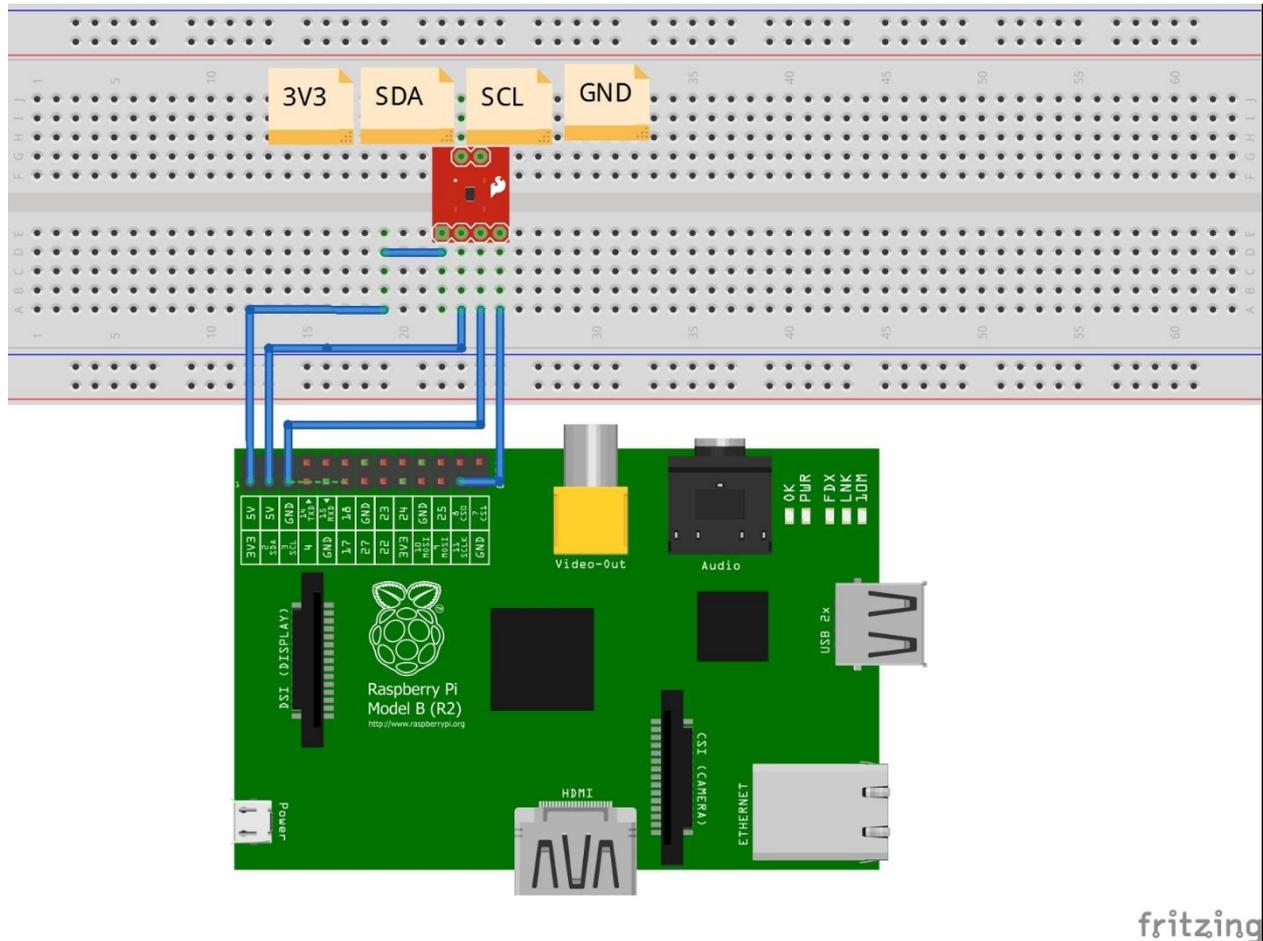




Le TP met en œuvre le capteur Texas Instrument TMP102 : <http://www.ti.com/product/tmp102>



Ce composant disponible en boîtier SOT563 est difficilement implantable sur circuit imprimé sans l'aide de machine de positionnement. Sparkfun propose une petite carte équipée d'un TMP102, des résistances de pull-up du bus I2C et d'un condensateur de découplage : <https://www.sparkfun.com/products/11931>



fritzing

**Raspberry Pi dispose d'une interface I2C et d'une bibliothèque.**

**broche 3 : SDA, broche 5 :SCL**

*(les résistances de pull-up sont déjà sur le module Sparkfun)*

**La bibliothèque Python I2C-SMBUS pour Raspberry Pi**

*I2C et SMBUS sont très similaires, les différences résident dans les fréquences d'horloge et les tensions de déclenchement.*

	I <sup>2</sup> C	SMBus
Timeout	No	Yes
Minimum Clock Speed	DC	10kHz
Maximum Clock Speed	100kHz (400kHz and 2MHz also available)	100kHz
VHIGH	0.7 × VDD, 3.0V Fixed	2.1V



<i>VLOW</i>	0.3 × VDD, 1.5V Fixed	0.8V
<i>Max I</i>	3mA	350µA
<i>Clock Nomenclature</i>	SCL	SMBCLK
<i>Data Nomenclature</i>	SDA	SMBDAT

### Programmation d'un thermomètre

Recuperation des parquets

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

### La commande modprobe

modprobe permet d'ajouter (et d'enlever) des modules dans le Noyau Linux, comme le module I2C.

```
sudo modprobe i2c-dev
sudo modprobe i2c-bcm2708
```

lsmod retourne la liste des modules actifs

modinfo nom\_du\_module retourne les informations du module nom\_du\_module

```
pi@raspberrypi ~ $ lsmod
Module                  Size Used by
i2c_bcm2708             3885  0
i2c_dev                 5590  0
aes_generic             35386  1
evdev                   9509  1
joydev                  9359  0
stmpe_ts                3184  0
fb_ili9340              3972  1
fbtft_device            22588  0
fbtft                   31031  2 fb_ili9340,fbtft_device
syscopyarea             3097  1 fbtft
sysfillrect             3264  1 fbtft
sysimgblt               2160  1 fbtft
fb_sys_fops             1389  1 fbtft
snd_bcm2835             16469  0
snd_pcm                 78862  1 snd_bcm2835
snd_page_alloc          5146  1 snd_pcm
snd_seq                 53885  0
snd_seq_device          6460  1 snd_seq
snd_timer               21043  2 snd_pcm,snd_seq
snd                     58994  5 snd_bcm2835,snd_timer,snd_pcm,snd_seq,snd_seq_de
vice
spidev                  5224  0
arc4                    1654  2
rt2800usb               15081  0
rt2800lib               56231  1 rt2800usb
rt2x00usb               11421  1 rt2800usb
rt2x00lib               42307  3 rt2x00usb,rt2800lib,rt2800usb
mac80211                276273  3 rt2x00lib,rt2x00usb,rt2800lib
cfg80211                183975  2 mac80211,rt2x00lib
rfkill                  18397  2 cfg80211
crc_ccitt               1499  1 rt2800lib
leds_gpio               2178  0
led_class               3573  2 leds_gpio,rt2x00lib
spi_bcm2708             6886  0
rpi_pwm                 6547  0
pi@raspberrypi ~ $
```



### Commandes de la librairie i2c-dev

```
sudo i2cdetect -y 1
```

```

pi@raspberrypi: ~/python
pi@raspberrypi ~/python $ sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi ~/python $

```

Lit le mot (w) ou l'octet (b) dans l'adresse 5 (rien pour un lire octet unique) du circuit à l'adresse I2C 0x1F  

```
sudo i2cget -y 1 0x48 0x5 w
```

```

pi@raspberrypi ~/python $ sudo i2cget -y 1 0x48 0x5 w
0xe006
pi@raspberrypi ~/python $

```

Ecrit 55 dans le circuit à l'adresse I2C 0x20 sur le bus 1

```
sudo i2cset -y 1 0x20 55
```

Lit le contenu en octets (b) ou mot (w) du circuit à l'adresse I2C 0x48 sur le bus 1

```
i2cdump -f 1 0x48 b
```

```

pi@raspberrypi ~/python $ sudo i2cdump -f 1 0x48 b
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-1, address 0x48, mode byte
Continue? [Y/n] Y
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  0123456789abcdef
00: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
10: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
20: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
30: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
40: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
50: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
60: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
70: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
80: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
90: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
a0: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
b0: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
c0: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
d0: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
e0: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
f0: 1a 60 4b 50 05 06 00 00 1a 60 4b 50 05 06 00 00  ?`KP??..?`KP??..
pi@raspberrypi ~/python $

```



## Utilisation de la bibliothèque Python I2C-SMBUS

Gestion capteur tmp102 en python

```
#!/usr/bin/env python
# 1 acces au bus I2C de la Raspberry Pi necessite :
#sudo modprobe i2c-dev
#sudo modprobe i2c-bcm2708
# pour verifier la presence d'un peripherique i2c :
#sudo i2cdetect -y 1

import smbus
import time

bus_pi = smbus.SMBus(1)

# I2C address for TMP102
addr = 0x48

while True:
    try:
        x = bus_pi.read_word_data(addr,0)
        msb=x>>8
        lsb=x&0x00FF
        wtemp=((lsb<<8)|msb)>>4
        print 'TMP102 I2C:  0x{0:02x} Lecture 0x{1:04x}'.format( addr,wtemp )
        print 'temperature : ',wtemp*0.0625 , 'degres C\n\r'
        time.sleep(0.5)
    except:
        print '...erreur...'
        break
```

```
pi@raspberr
TMP102 I2C:  0x48 Lecture 0x01af
temperature :  26.9375 degres C

TMP102 I2C:  0x48 Lecture 0x01af
temperature :  26.9375 degres C

TMP102 I2C:  0x48 Lecture 0x01b0
temperature :  27.0 degres C

TMP102 I2C:  0x48 Lecture 0x01af
temperature :  26.9375 degres C

TMP102 I2C:  0x48 Lecture 0x01b0
temperature :  27.0 degres C
```