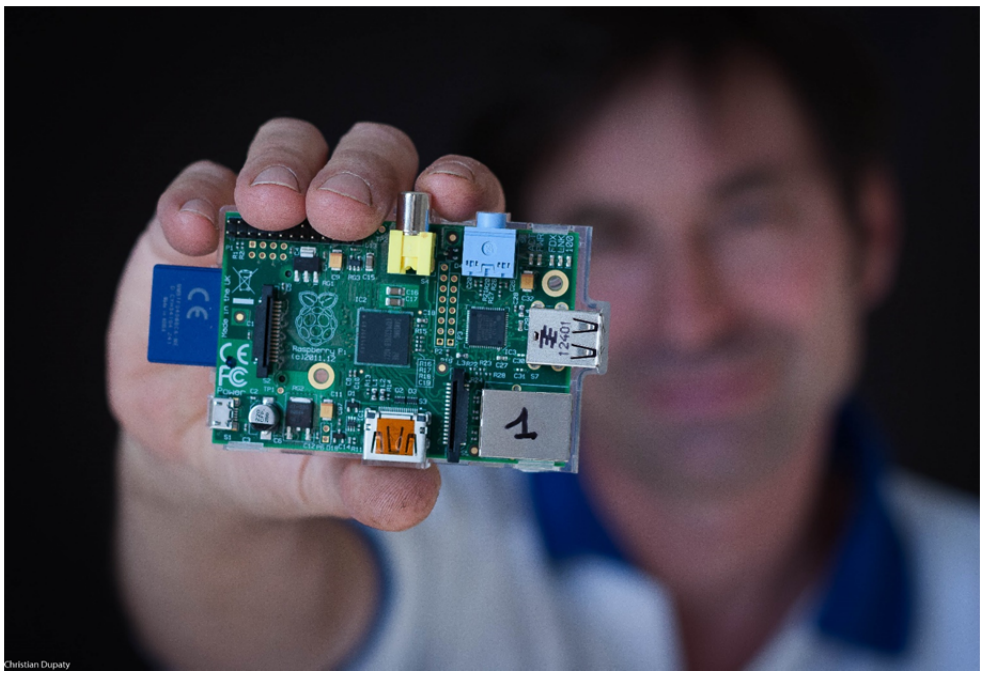
RASPBERRY PI

INSTALLATION-CONFIGURATION

INTERFACES DE COMMUNICATIONS



SPI

Christian Dupaty

BTS Systèmes Numériques

Lycée Fourcade - Gardanne

Académie d'Aix-Marseille

1) TP : SPI

A lire : http://fr.wikipedia.org/wiki/Serial_Peripheral_Interface

Le bus SPI est un bus de communication synchrone. Contrairement au bus I2C il est full-duplex (une ligne de transmission MOSI et une ligne de réception MISO). Très rapide car sans protocole logiciel, il nécessite en revanche une ligne supplémentaire /CS pour sélectionner l'esclave. Ce bus est donc très utilisé dans les communications locales pour un petit nombre de périphériques. La Raspberry Pi dispose d'un bus SPI et de deux /CS.

Un site donnant un exemple http://www.100randomtasks.com/simple-spi-on-raspberry-pi

Activer le périphérique sur la Raspberry Pi

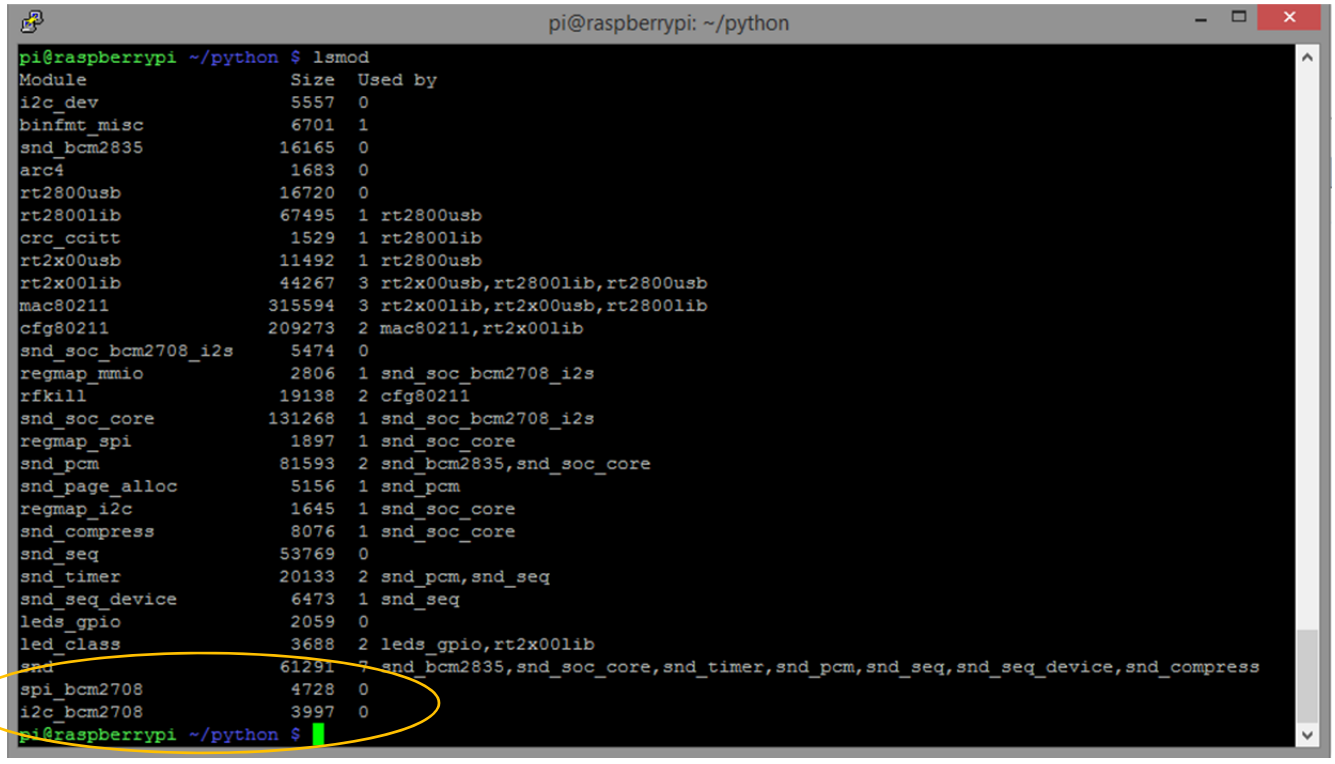
```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Retirer le # devant la ligne spi-bcm2708

(faire de même pour la ligne I2C si nécessaire)

Redemarrer : `sudo reboot now`

Après le redémarrage, taper `lsmod`, les périphériques i2c et SPI doivent apparaitre dans la liste



Installation de la bibliothèque SPI SPIdev et documentation http://tightdev.net/SpiDev_Doc.pdf

Pour installer la bibliothèque SPIDEV

```
mkdir python-spi
cd python-spi
wget https://raw.github.com/doceme/py-spidev/master/setup.py
wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
sudo python setup.py install
```

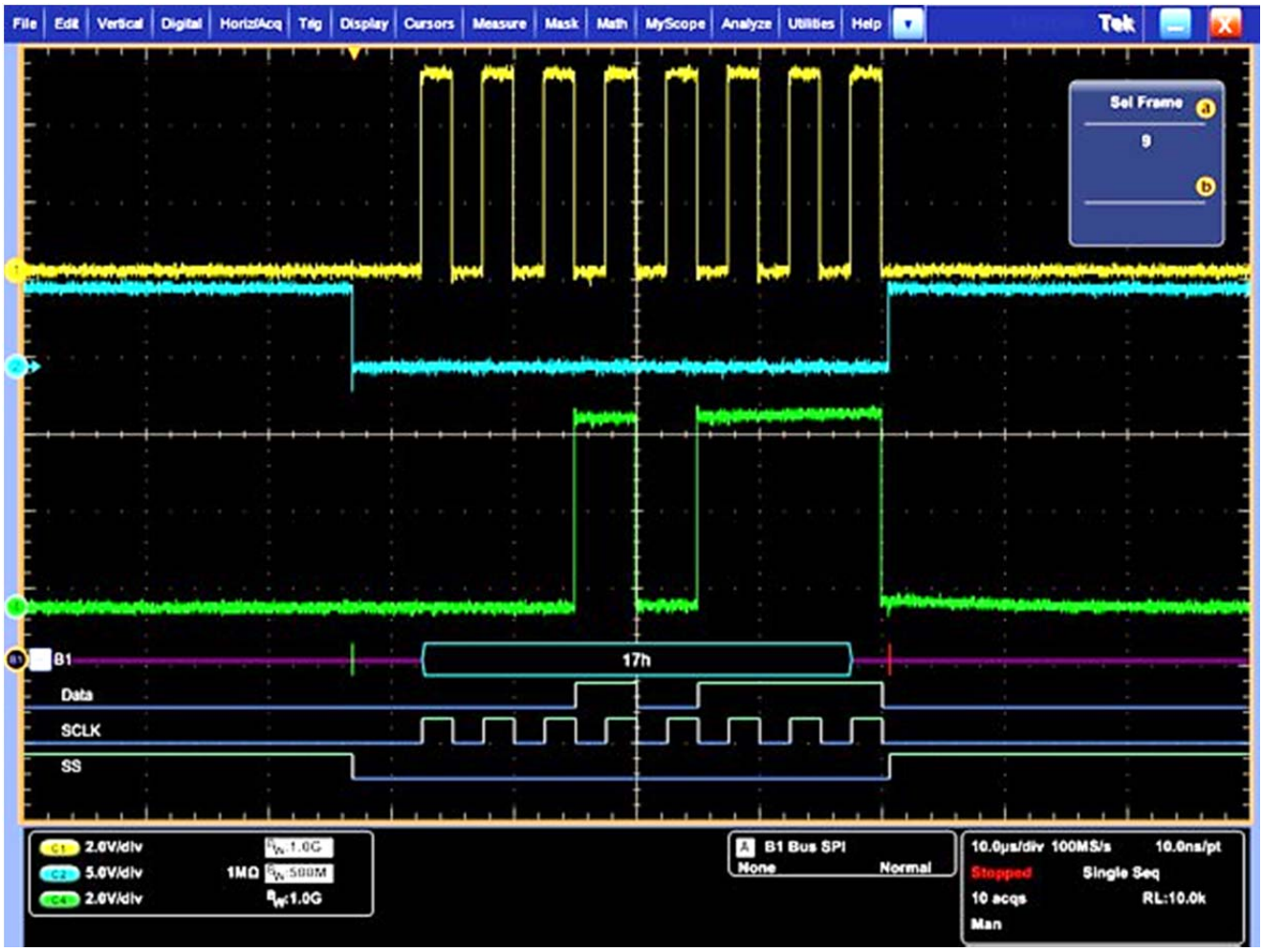
Exemple, échange d'un octet :

Le programme ci-dessous active le SPI 0 puis tous le 1/10s envoie 0x17, la réception est dans resp.
Raspberry Pi possède deux /CS pour le port SPI, dans l'exemple ci-dessous /CS1 est utilisé
Brochage sur le connecteur d'extension Raspberry Pi :
MOSI : 12, MISO : 13, SCLK : 14, CE0 : 10, CE1 : 11

```
import spidev
import time
spi = spidev.SpiDev() # crée l'objet SPI
spi.open(0, 1) # demarre spi port 0,  (CS) 1
try:
      while True:
            resp = spi.xfer2([0x17]) # transfer un octet
            time.sleep(0.1) # attend 0.1 secondes

except KeyboardInterrupt: # Ctrl+C pour quitter
spi.close() # ferme le peripherique
```

Résultat de transmission d'un octet (0x17 = 0b00010111), programme ci-dessus
En jaune l'horloge, en bleu /CS, en vert MOSI.



Fonctions de la bibliothèque SPI :

bits_per_word

Description: nombre de bits par transfert (généralement 8).
8 ou 16 ex : bits_per_word=8

close

Syntax: close()
Retournes: Rine
Description: désactive l'interface.

cshigh

Description: indique si CS est actif à l'état haut (généralement CS est actif à l'état bas)
ex : cehight=false

loop

Description: boucle pour tester l'interface. Ex loop=false

lsbfirst

Si le poids faible doit être transmis en premier
ex : lsbfirst=false

max_speed_hz

Description: Vitesse de transfert en Hz
ex : max_speed=100000

mode

Polarité du bus SPI : [CPOL|CPHA] (voir data sheet du périphérique)
x est compris entre 0b00 = 0 .. 0b11 = 3 ex : mode=0

open

Syntaxe: open(bus, device)
Description: active SPI (0 ou 1). Device est le numéro du CS (0 ou 1)

readbytes

Syntaxe: read(x)
Retourne: [values]
Description: Lit x octets dur l'esclave.

threewire

Description: Propriété des périphériques ne disposant que d'une ligne I/O (voir datasheet du périphérique)
ex : threewire=Thrue

writebytes

Syntaxe: write([values])
Retournes: rien
Description: Ecrit un octet vers l'esclave.

xfer

Syntaxe: xfer([values],tempo)

Retourne: [values]
Description : Echange les données Maitre-Esclave, CS repasse à l'état haut entre les octets. Tempo en µsec entre les octets

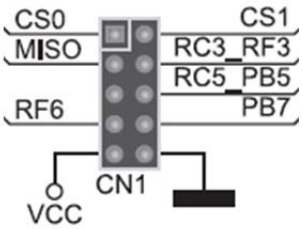
xfer2

Syntaxe: xfer2([values])
Retourne: [values]
Description: Echange les données Maitre-Esclave, CS reste à l'état bas entre les octets

Un exemple SPI : Horloge temps reel, sortie sur le module SPI « Serial 7-Segments 8 digit » MIKROELEKTRONIKA.

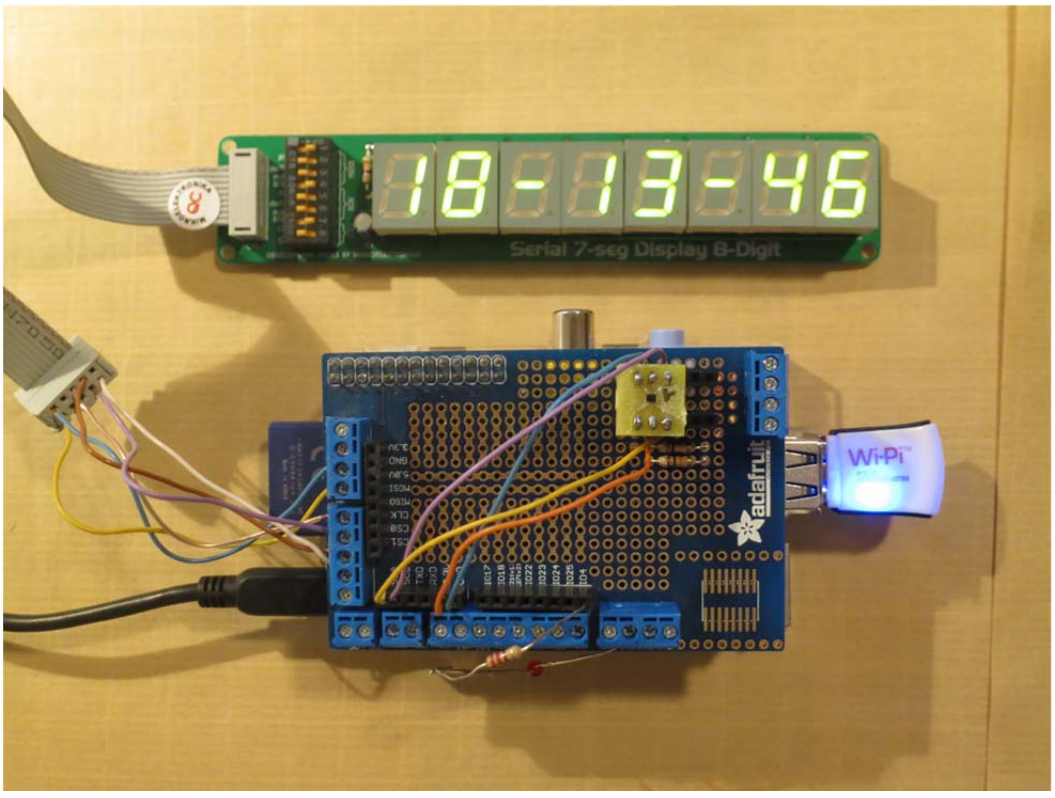
http://www.mikroe.com/add-on-boards/display/serial-7seg-8-digit/

Connexions à réaliser 7SEG-RPi:



VDD 5v GND
CS0 : RPi 10
MISO : RPi 13
MOSI : RPi 12
SCK (sur RC3_RF3) : RPi 14

La photo ci-dessous montre l'horloge temps réel. La connexion utilise la carte d'extension PiFace (http://www.piface.org.uk/)



```
#C.Dupaty 12/2013
#Test SPI : RTC avec module MIKROELEKTRONIKA "serial 7-SEG Display 8-DIGIT"

#!/usr/bin/python
# Horloge
import spidev #bibliotheque SPI
import time   # pour tempo 1s

# active le BUS SPI
spi = spidev.SpiDev()   #nouvel objet SPI
spi.open(0,0)        # sur port SPI 0 CS 0
spi.max_speed_hz = 1000000  # vitesse en Hz

global heu,min,sec,jsemaine,jour,mois,annee # variables globales

# codes 7 segments
code7seg = (0x7E,0x30,0x6D,0x79,0x33,0x5B,0x5F,0x70,0x7F,0x7B)
# tuple des jours de la semaine
nom_jour=('lundi','mardi','mercredi','jeudi','vendredi','samedi','dimanche')

# initialise le MAX7912
def Initialise():
  spi.writebytes([0x09])  #decoder BCD
  spi.writebytes([0x0])  #(0 pour ne pas decoder)

  spi.writebytes([0x0a])   # regle la luminance
  spi.writebytes([0x08])   # 17/32

  spi.writebytes([0x0b])   # affiche les 8 digits
  spi.writebytes([0x07])

  spi.writebytes([0x0c])  # power-down mode: 0. normal mode:1
  spi.writebytes([0x01])  # (0 pour shutdown)

  spi.writebytes([0x0f])  # test display: 1; EOT. display: 0
  spi.writebytes([0x00])  # (1 pour test)

# ecriture SPI
def ecr_nb(digit,val):
       spi.xfer2([digit,val])

# eteind tous les aff 7seg
def efface():
       d=8
       while(d):
           ecr_nb(d,0b00000000)
           d-=1
       ecr_nb(3,0b00000001)
       ecr_nb(6,0b00000001)

# affiche heu, min sec sur les afficheurs
def aff_heure():
       global heu,min,sec
       ecr_nb(7,code7seg[heu%10])     #heures
       ecr_nb(8,code7seg[heu/10 %10]) #dizaines d heures
       ecr_nb(4,code7seg[min%10])     #minutes
       ecr_nb(5,code7seg[min/10 %10]) #dizaines de minutes
       ecr_nb(1,code7seg[sec%10])     #secondes
       ecr_nb(2,code7seg[sec/10 %10]) #dizaines de secondes

# lit l heure systeme et la range dans les variables heu, min, sec
def lit_heure():
       global heu,min,sec,jsemaine,jour,mois,annee
       t=time.localtime()  # t est un tuple du temps local
       heu=t[3]
       min=t[4]
       sec=t[5]
       jsemaine=t[6]
       jour=t[2]
       mois=t[1]
```

```
       annee=t[0]

def RTC():
       global heu,min,sec,jsemaine,jour,mois,annee
        sec+=1         # calcul les nouvelle heure
        if sec >= 60:
          sec=0
          min+=1
          if min>=60:
            min=0
            heu+=1
            if heu >=24:
              heu=0
             lit_heure() # on recharge l heure systeme toutes les 24h
        aff_heure()     # l affiche

def main():
       global heu,min,sec,jsemaine,jour,mois,annee
       Initialise()
       efface()
        # paramatres du bus SPI
       print 'mode : ' ,spi.mode      # mode, phases SCK SDA
       print 'cshight : ' ,spi.cshigh # etat de CS
       print 'lsbfirst : ' ,spi.lsbfirst  # poids faibles emis en premier ?
       print 'threewire : ' ,spi.threewire # SDI est connectee
       print 'loop : ' ,spi.loop            # ?
       print 'bits_per_word : ' ,spi.bits_per_word # nombre de bits par mot
       print 'max_speed_hz : ' ,spi.max_speed_hz   # vitesse en Hz
       print 'C EST PARTI CTRL-C pour arreter'
       lit_heure()        # lit l heure systeme
       while True:
          RTC()           # calcule la nouvelle heure
           print '%s %2d/%2d/%4d -
%2d:%2d:%2d'%(nom_jour[jsemaine],jour,mois,annee,heu,min,sec)
          time.sleep(1)  # libere le processeur pour 1s

if __name__ == '__main__':
   main()
```