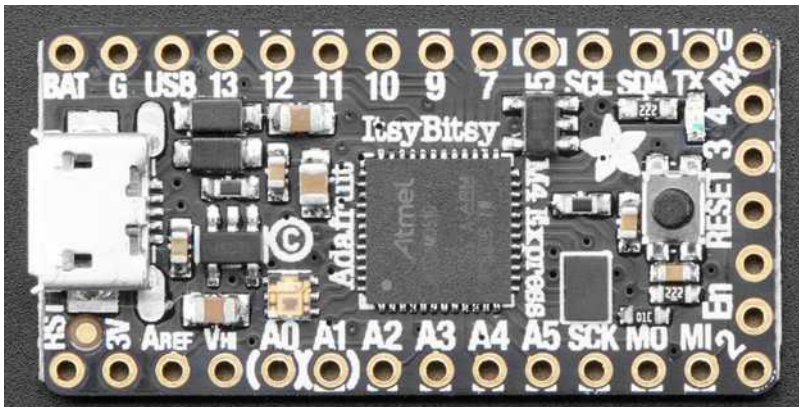
Cours sur les capteurs

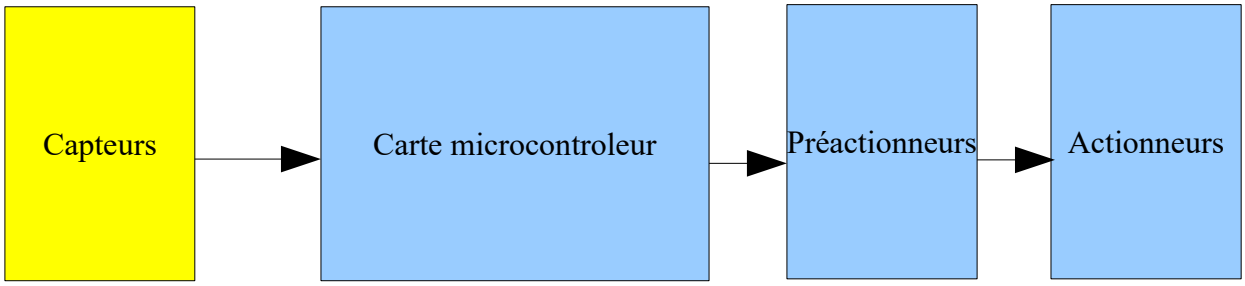
Christophe LIGERET et Benjamin MOUSSET
Juillet 2019

Introduction

Pour illustrer la séquence sur les capteurs, nous allons utiliser le microtrôlleur ATSAMD51 : par exemple l'ItsyBitsy d'Adafruit : https://www.adafruit.com/product/3800



Cette carte électronique coûte environ 15 $ est construite autour du microcontrôleur Microchip ATSAMD51 Cortex M4.
Cette carte se programme aussi bien en C avec Arduino qu'en Python avec Mu Editor par exemple.
Le processeur tourne à 120 MHz (il y a mieux : pour le même prix, une carte TeenSy 4.0 doté du microcontrolleur ARM Cortex-M7 fonctionne à 600 MHz et possède 1024K de RAM) ; possède une mémoire RAM de 192 KB et une mémoire Flask de 512 KB. Cette carte a 23 entrées / sorties (0 – 3,3V) dont 7 entrées analogiques ; 2 sorties analogiques et 18 configurables en mode PWM ; une sortie « haute tension » 0 – 5V pour piloter des leds Neopixel par exemple.
Ce microcontroleur est alimenté par une tension VCC = 3,3V (d'autres circuits fonctionnent avec du 5V : Arduino...). Dans ce document, on prendra Vcc = 3,3V.



Les capteurs sont des dispositifs électriques / électroniques destinés à convertir une information physique en une information exploitable par le microprocesseur.

Par exemple, un simple détecteur mécanique avec contact électrique va transformer un mouvement mécanique en signal électrique.

Une cellule photoélectrique va transformer un flux de lumière en courant électrique.

Une cellule photorésistive va transformer un flux de lumière en résistance électrique.

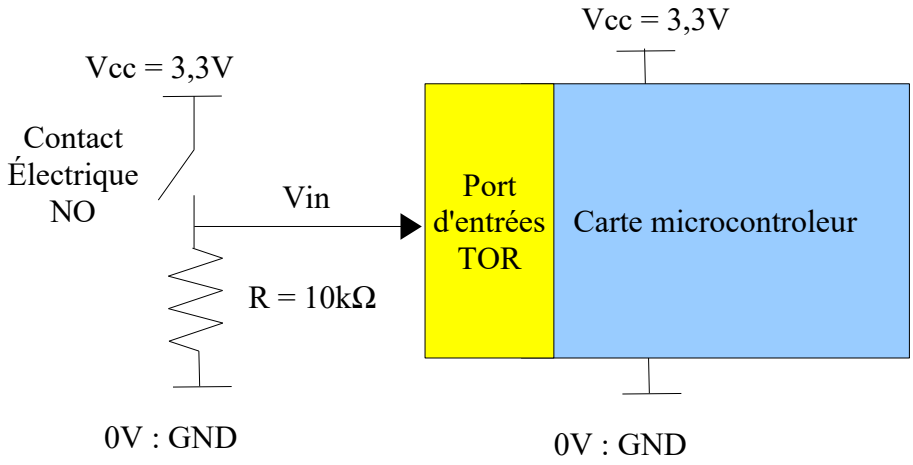
De même, une thermistance (capteur de température) va transformer une température en résistance...

Les grandeurs électriques sont « lues » par le microprosseur via une interface d'entrée : nours étudierons dans ce document 3 familles de capteurs :
- les capteurs tout ou rien qui délivrent un signal au microprocesseur 0V ou 3,3V
- les capteurs analogiques qui délivrent un signal au microprocesseur entre 0V et 3,3V
- les capteurs numériques qui dialoguent avec le microprocesseur via un bus de données (série) : UART ; I2C ou SPI

I / Capteurs tout ou rien

Un capteur tout ou rien délivre à l'unité de traitement une variable binaire. Cette variable binaire est codée avec 2 niveaux de tension : 0 ou False : 0V et 1 ou True : 3,3V (Vcc).

Exemple avec un capteur mécanique équipé d'un contact électrique : 1 NO monté sur Vcc



ATTENTION : il faut veiller à ce que la tension d'entrée sur le port d'entrée du microcontroleur soit toujours comprise entre 0V et Vcc (ici entre 0V et 3,3V) car sinon, on risque de détériorer le microcontroleur. En réalité, il y a une toute petite tolérance d'environ +/- 0,3V. Si on applique une tension de 5V sur une entrée tolérant une tension entre 0V et 3,3V, on rique de la griller : lire attentivement la documentation technique !

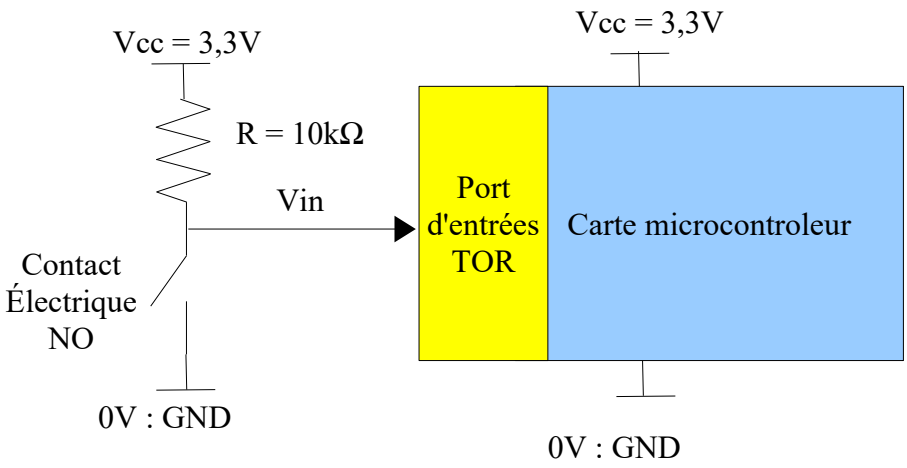
Fonctionnement du dispositif : lorsqu'on active le capteur, le contact se ferme et le signal Vin passe à 3,3V
Le microcontroleur lit alors un « 1 » logique : il renvoie **`True`** en Python.
Lorsqu'on désactive le capteur, le contact s'ouvre et le signal Vin passe à 0V (via la résistance de pull-down : R = 10 kΩ) : le microcontroleur lit alors un « 0 » logique : il renvoie **`False`** en Python.

ATTENTION : il est obligatoire de placer une résistance de pull-down pour forcer le 0V lorsque le contact est fermé, car sinon, l'entrée du microcontroleur est du type « haute impédance » et risque d'^étre flottante : elle lira alors aléatoirement des 0 ou des 1.

```
# Exemple de programme de lecture du capteur TOR
import time
import board
from digitalio import DigitalInOut, Direction, Pull
led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

# For Itsy M4 Express : other controllers : see datasheet
capteur =  DigitalInOut(board.D5)
capteur.direction = Direction.INPUT
capteur.pull = None

while True:
  etatCapteur = capteur.value
  if  etatCapteur == True:
    print(« Capteur activé »)
  else :
    print(« Capteur au repos »)
  time.sleep(1.0)
```

Remarque : dans certains microcontroleur, la résistance de pull-down y est intégrée : on peut l'activer par l'instruction : `capteur.pull = Pull.DOWN`
Ceci permet de faire l'économie de la résistance R = 10 kΩ et de brancher directement le contact entre Vcc et Vin.

Exemple avec un capteur mécanique équipé d'un contact électrique : 1 NO monté sur GND



Fonctionnement du dispositif : lorsqu'on active le capteur, le contact se ferme et le signal Vin passe à 0V
Le microcontroleur lit alors un « 0 » logique : il renvoie **`False`** en Python.
Lorsqu'on désactive le capteur, le contact s'ouvre et le signal Vin passe à 3,3V (via la résistance de pull-up : R = 10 kΩ) : le microcontroleur lit alors un « 1 » logique : il renvoie **`True`** en Python.

ATTENTION : là encore, il est obligatoire de placer une résistance de pull-up pour forcer le 3,3V lorsque le contact est fermé, car sinon, l'entrée du microcontroleur est du type « haute impédance » et risque d'^étre flottante : elle lira alors aléatoirement des 0 ou des 1.

```
# Exemple de programme de lecture du capteur TOR
import time
import board
from digitalio import DigitalInOut, Direction, Pull
led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

# For Itsy M4 Express : other controllers : see datasheet
capteur =  DigitalInOut(board.D5)
capteur.direction = Direction.INPUT
capteur.pull = None

while True:
  etatCapteur = capteur.value
  if  etatCapteur == True:
    print(« Capteur au repos »)
  else :
    print(« Capteur activé »)
  time.sleep(1.0)
```

Remarque : dans certains microcontroleur, la résistance de pull-up y est intégrée : on peut l'activer par l'instruction : `capteur.pull = Pull.UP`
Ceci permet de faire l'économie de la résistance R = 10 kΩ et de brancher directement le contact entre Vcc et Vin.

Capteur tout ou rien et interruptions
Les deux programmes présentés ci-dessus permettent de lire l'état du capteur TOR, mais, si on le déclenche très brièvement, il y a un risque que le microprocesseur ne le voit pas : on effectue ici un « pooling » du capteur une fois par seconde environ. Si entre 2 lectures, il se passe quelquechose, on ne le verra pas.

Pour remédier à ce problème, on peut utliser des interruptions : on déclenche l'exécution d'une partie du code Python sur un événement : par exemple, lorsque le capteur s'active ou se désactive.

Malheureusement, pour l'instant, Circuitpython ne prend pas en charge les interruptions : il faut alors faire preuve d'expertise et lire / écrire directement dans les registres du microcontroleur.

Exercices autour de capteurs tout ou rien (TOR)

Exercice 1 : lire attentivement ce programme ci-dessous

```
import board
from digitalio import DigitalInOut, Direction, Pull
led13 = DigitalInOut(board.D13) # Led 13 intégrée à la carte ItsyBitsy M4
led13.direction = Direction.OUTPUT # la led 13 est sur la patte n°13
capteur = DigitalInOut(board.D2) # le capteur est cablé entre la patte 2 et la
# masse GND
capteur.direction = Direction.INPUT # le capteur est une entrée TOR
capteur.pull = Pull.UP # le capteur est une entrée TOR PULLUP : 3,3V au repos

while True:
  if capteur.value == True: # le capteur est au repos (3,3V)
    led13.value = False # On éteint la led 13
    print(« La led 13 est éteinte »)
  else: # le capteur est au activé (0V)
    led13.value = True # On allume la led 13
    print(« La led 13 est allumée »)
```

1°) Que fait exactement le programme ci-dessus ?

2°) On remplace maintenant le code dans la boucle while par :

```
import time
while True:
  if capteur.value == False: # le capteur est au activé (0V)
    led13.value = True # On allume la led 13
    print(« La led 13 est allumée »)
    time.sleep(5.0)
  else: # le bouton poussoir est au repos (3,3V)
    led13.value = False # On éteint la led 13
    print(« La led 13 est éteinte »)
```

Que fait ce nouveau programme ? A quelle application de la vie de tous les jours pourrait-il être utile ?

Exercice 2 : lire attentivement ce programme ci-dessous

```
import board
from digitalio import DigitalInOut, Direction, Pull
led13 = DigitalInOut(board.D13) # Led 13 intégrée à la carte ItsyBitsy M4
led13.direction = Direction.OUTPUT # la led 13 est sur la patte n°13
capteur1 = DigitalInOut(board.D2) # le capteur est entre la patte 2 et GND
capteur1.direction = Direction.INPUT # le capteur est une entrée TOR
capteur1.pull = Pull.UP # le capteur est une entrée TOR PULLUP : 3,3V au repos
capteur2 = DigitalInOut(board.D3) # le capteur est entre la patte 3 et GND
capteur2.direction = Direction.INPUT # le capteur est une entrée TOR
capteur2.pull = Pull.UP # le capteur est une entrée TOR PULLUP : 3,3V au repos

while True:
  if capteur1.value == False: # le capteur 1 est activé (0V)
    led13.value = True # On allume la led 13
  if capteur2.value == False: # le capteur 2 est activé (0V)
    led13.value = False # On éteint la led 13
```

1°) Que fait exactement le programme ci-dessus ?

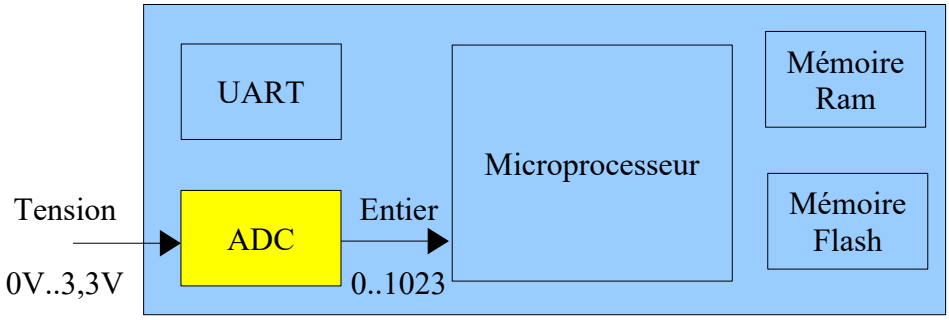
2° Faire une recherche sur Internet sur les bascules SET / RESET.

Exercice 3 : On considère un dispositif composé de 3 capteurs : c1 ; c2 et c3 et d'une sortie (led13).
Coder en langage Python un programme qui calcule : led13 = (c1 + c2) . (c3 + !c2)

Source : https://learn.adafruit.com/circuitpython-essentials/circuitpython-digital-in-out

II / Capteurs analogiques

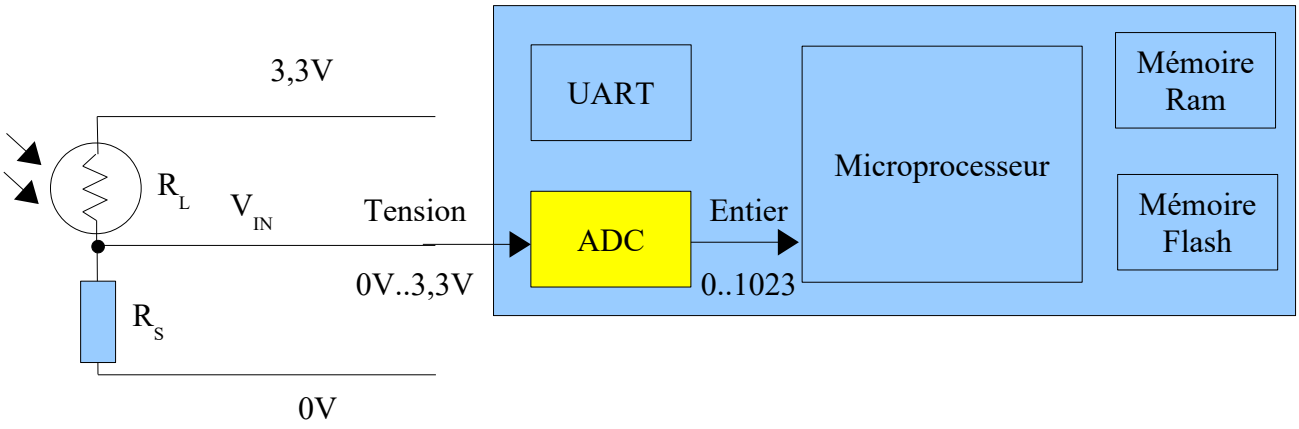
Un ADC pour Analog to Digital Converter est un convertisseur analogique numérique qui permet de convertir une tension électrique (entre 0V et 3,3V) en un nombre entier (généralement entre 0 et 1023)
On supposera que le nombre en entrée du convertisseur analogique numérique est linéaire en fonction de la tension électrique en entrée.



microcontrôleur

Certains capteurs (température, luminosité, pression, force...) sont de type analogique et délivrent un signal électrique « continu », image de la grandeur physique mesurée.

Exemple avec un capteur de lumière analogique

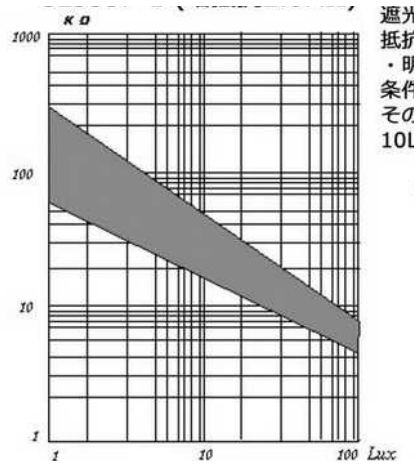


Lorsque la photorésistance est plongée dans l'obscurité, sa résistance est très élevée : Vin est environ égal à 0V. Par contre, lorsque la photorésistance est soumise à un éclairement intense, sa résistance est très faible : Vin est environ égal à 3,3V. La tension Vin varie continuement entre 0V et 3,3V.

Photorésistance GL5516

On peut modéliser la résistance de cette photorésistance par :
$R = e^{-0,651 \ln(E)} + 4,605$ où
R est la résistance en $k\Omega$ et E est l'éclairement en lux.

Réciproquement : $E = e^{-1,537 \ln(R)} + 7,079$



Caractéristique électrique de la photorésistance GL5516

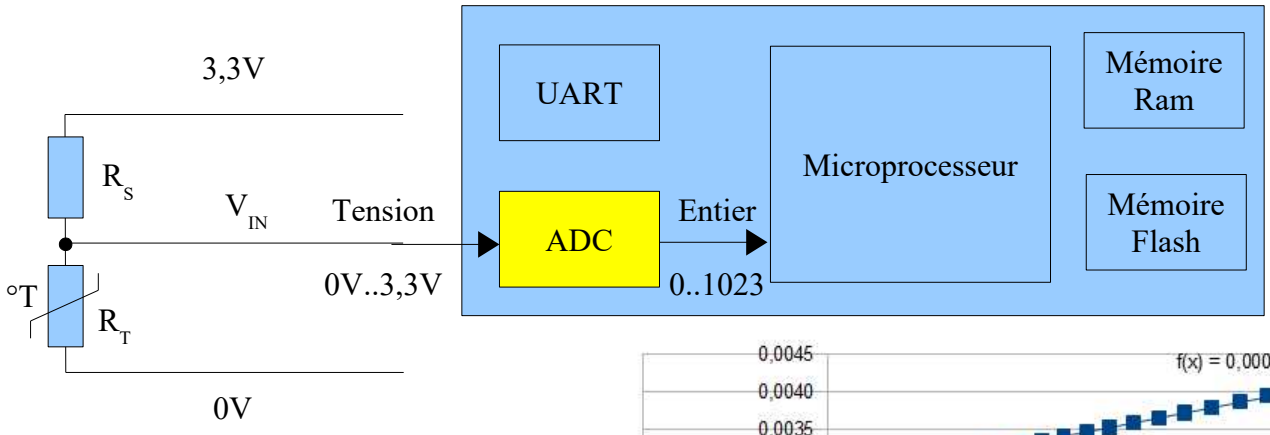
On prendra dans cet exemple RS = 10 $k\Omega$

Notre microcontroleur mesure sur sa patte d'entrée analogique une tension entre 0 et 3,3V. Cette tension va nous permettre de trouver la valeur de la photorésistance : en effet, l'ensemble {photorésistance + RS} forme un pont diviseur de tension: $VIN = \frac{RS}{RS+R} Vcc$ d'où $R = RS\frac{Vcc-Vin}{Vin}$.

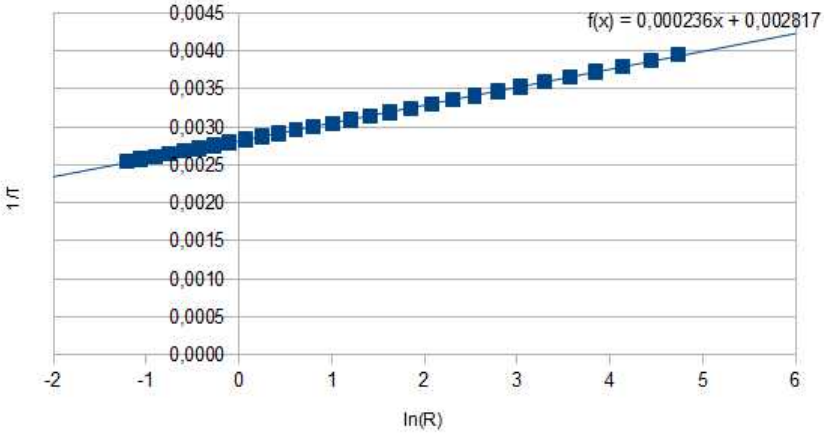
Ainsi, on peut proposer le code ci-dessous pour afficher la valeur de l'éclairement en lux :

```
# Affichage de l'éclairement en lux
import time
import board
from analogio import AnalogIn
from math import exp,log
RS = 10

photoresistance = AnalogIn(board.A1)

def lireTensionVin():
  return (photoresistance.value * 3.3) / 65536

while True:
  Vin = lireTensionVin()
  R = RS*(3.3 - Vin)/Vin
  E = exp(-1.537*log(R)+7.079)
  message = «L'éclairement est égal à»+str(E)+«lux»
  print(message)
  time.sleep(1.0)
```

Exemple avec un capteur de température analogique



Thermistance NTC
TDK B57164K103J K164
10 kΩ



On admettra que $T_{\circ C} = \frac{1}{0.002817 + 0.000236 \times \ln\left(R_S \times \frac{V_{IN}}{V_{DD} - V_{IN}}\right)} - 273.15$ et on prendra RS = 10 $k\Omega$

Ainsi, on peut proposer le code ci-dessous pour afficher la valeur de l'éclairement en lux :

```
# Affichage de la température en °C
import time
import board
from analogio import AnalogIn
from math import log
RS = 10

thermistance = AnalogIn(board.A1)

def lireTensionVin():
  return (thermistance.value * 3.3) / 65536

while True:
  Vin =  lireTensionVin()
  T = 1/(0.002817+0.000236*log(RS*Vin/(3.3-Vin))) - 273.15
  message = «La température est égale à»+str(E)+«°C»
  print(message)
  time.sleep(1.0)
```

Exemple avec un capteur de pression analogique

On considère le capteur de pression décrit ci-dessous :

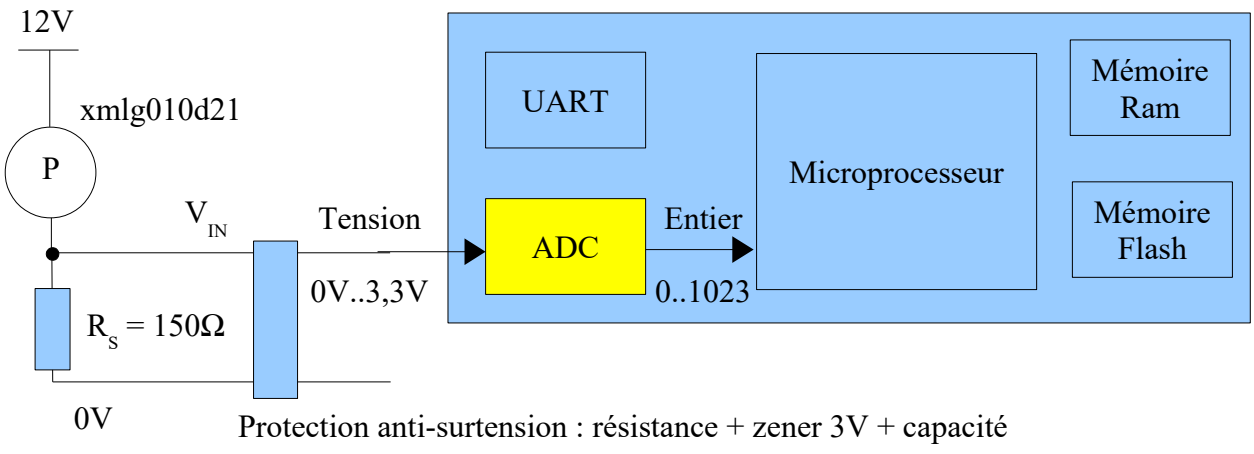
Product datasheet
Characteristics

xmlg010d21
pressure sensor XMLG - 0..10 bar - G 1/4A (male) - 24 V - 4..20 mA

Main

Range of product	OsiSense XM
Product or component type	Electronic pressure sensors
Pressure sensor type	Pressure transmitter
Pressure sensor name	XMLG
Pressure sensor size	10 bar
Fluid connection type	G 1/4A (male) conforming to ISO 228
Controlled fluid	Air (-15...125 °C) Corrosive fluid (-15...125 °C) Hydraulic oil (-15...125 °C) Fresh water (0...125 °C)
Type of output signal	Analogue
Analogue output function	4...20 mA, 2 wires
Electrical connection	3 pins 1 male connector M12
[Us] rated supply voltage	12/24 V DC, voltage limits: 8...33 V

La sortie de ce capteur est branchée sur la masse au travers une résistance de 150 Ω



Protection anti-surtension : résistance + zener 3V + capacité

Ainsi, on peut proposer le code ci-dessous pour afficher la valeur de la pression en bar :

```
# Affichage de la pression en bar
import time
import board
from analogio import AnalogIn

capteurPression = AnalogIn(board.A1)

def lireTensionVin():
  return (capteurPression.value * 3.3) / 65536

while True:
  Vin =  lireTensionVin()
  courant = Vin/150
  pression = 0.625*courant-2.5
  message = «La pression est égale à»+str(pression)+«Bar»
  print(message)
  time.sleep(1.0)
```

Exercice 1 :
On souhaite réaliser un système d'éclairage automatique qui se déclenche sur un seuil de 20 lux.
On enverra des messages à la console du type « éclairage ON » ou « éclairage OFF »
Complétez le programme ci-dessous :

```
# Système d'éclairage automatique
import ............
import ............
from analogio import ............
from math import ...........
RS = 10

photoresistance = ............(............ . A1)

def lireTensionVin():
  return (photoresistance.value * 3.3) / 65536

while True:
  Vin = lireTensionVin()
  R = RS*(3.3 – Vin)/Vin
  E = exp(-1.537*log(R)+7.079)
  ......................
    ......................
  ......................
    ......................
  time.sleep(1.0)
```

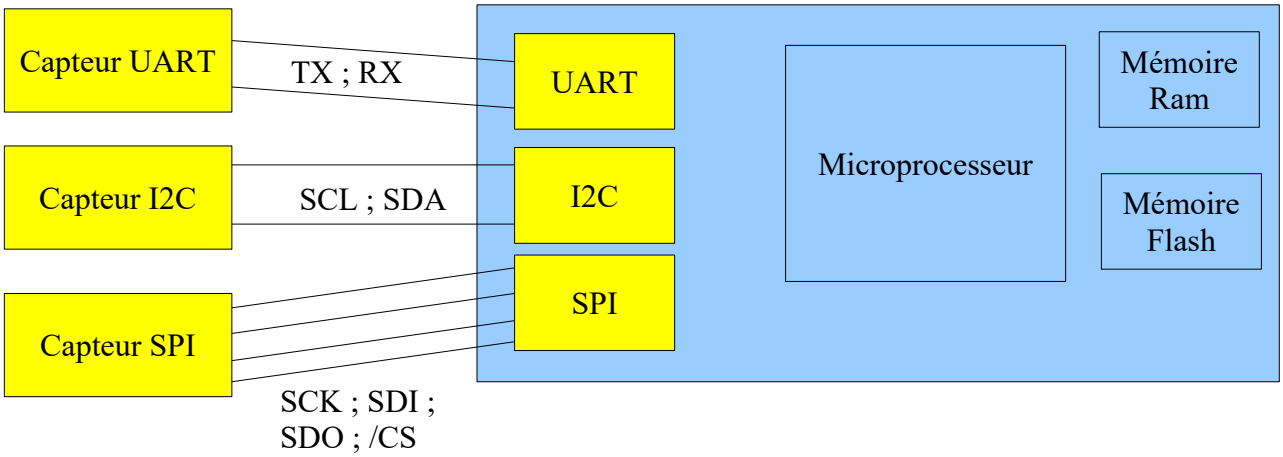
Exercice 2 : reprendre l'exercice 1 avec les spécifications suivantes pour éviter les instabilités :
- l'allumage de l'éclairage se fait lorsque l'intensité lumineuse est inférieure à 20 lux.
- l'extinction de l'éclairage se fait lorsque l'intensité lumineuse est supférieure à 30 lux.

Exercice 3 : reprendre les exercices 1 et 2 avec les spécifications suivantes pour éviter les cycles courts :
- l'allumage de l'éclairage se fait lorsque l'intensité lumineuse est inférieure à 20 lux avec une durée minimale de 30 secondes.
- l'extinction de l'éclairage se fait lorsque l'intensité lumineuse est supférieure à 30 lux avec une durée minimale de 10 secondes..

Source : https://learn.adafruit.com/circuitpython-essentials/circuitpython-analog-in

III / Capteurs numériques

Nous présentons dans cette partie des capteurs numériques : le transfert d'informations entre le capteur et le microcontroleur se fait au moyen d'un bus de commuication série : I2C : SPI ; UART …
On rencontre des capteurs numériques lorqu'il y a un prétraitement réalisé à l'intérieur du capteur (le capteur n'est pas une simple résistance variable comme on a peu le voir ci-dessus).



Ci-dessus : présentation des ports UART ; I2C et SPI présents sur la plupart des microcontroleurs

Description des ports UART ; I2C et SPI

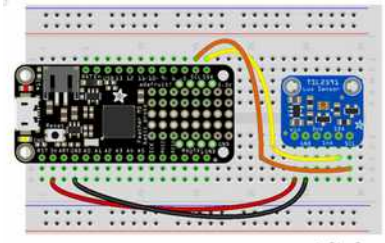
Le port UART (Universal Asynchronous Receiver Transmitter : port série « classique ») est doté de minimum 2 signaux : un signal TX sur lequel le microcontroleur envoie des messages au capteur et un signal RX sur lequel le capteur envoie des mesures au microcontroleur. Il peut avoir une longueur de quelques mètres selon le débit utilisé (généralement 9600 baud). Avec des drivers dédiés, la longueur de la connection peut atteindre plusieurs kilomètres.

Le port I2C (Inter Ingrated Cirduit : bus de communication entre circuit intégrés) est un port courte portée (quelques cm) qui permet de faire communiquer différents circuits intégrés entre eux : microcontroleur ; mémoire ; capteurs embarqués... Il est doté de deux lignes : une pour l'horloge de synchronisation : SCL et une pour les données : SDA qui voyagent aussi bien dans le sens microcontrôleur vers capteur que l'inverse. Le bus I2C est fondé sur la notion d'adresse ; de maître et d'esclave : le maître est le microcontrôleur (adresse 0) qui interroge successivement les différents circuits reliés sur SCL et SDA. Chaque esclave est identifié par une adresse et est capable de reconnaître un message qui lui est dédié. La vitesse de transfert des informations est de l'ordre de 400 kbit/s.

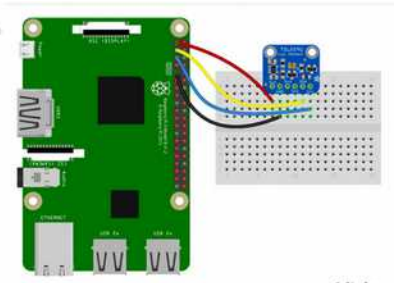
Le port SPI ressemble un peu au bus I2C mais il y a 4 signaux et pas de notion d'adresse : une horloge de synchronisartion SCK ; une ligne SDO sur laquelle le microcontroleur émet des messages vers le(s) destinataire(s) ; un signal SDI sur lequel le capteur émet des messages vers le microcontroleur et un fil /CS qui permet d'adresser physiquement le composant avec lequel on souhaite commuiquer. Ce bus est de courte portée (quelques cm) mais permet des débit importants : 1 Mbit/s .

Exemple d'un capteur de luminosité numérique

Nous allons étudier un montage avec le circuit TSL2591 (High Dynamic Range Digital Light Sensor)



Connection avec un ATSAMD51

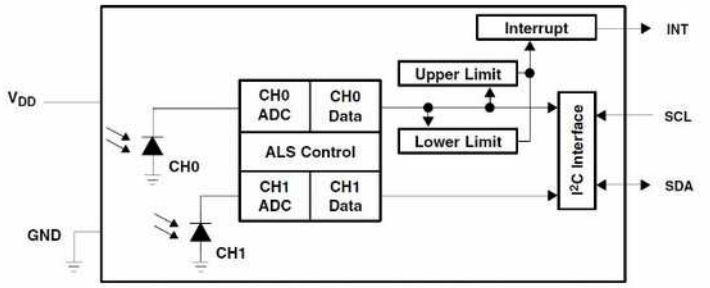


Connection avec un Raspberry Pi

TSL2591

Ci-dessus : le branchement entre le microcontroleur et le capteur TSL2591 se fait au moyen d'un bus I2C : on relie SCL et SDA (fils vert et jaune). De plus, on relie l'alimentation 3,3V entre le microcontroleur et le capteur : 0V : noir et 3,3V : rouge.

Comme le TSL2591 est un capteur « intelligent », il faut tout d'abord télécharger une bibliothèque permettant de communiquer avec lui en Python : la bibliothèque adafruit_tsl2591 est disponible à l'adresse :
https://github.com/adafruit/Adafruit_CircuitPython_TSL2591
Ce capteur permet de déterminer l'intensité lumineuse qu'il reçoit en lux ; l'intensité lumineuse dans le spectre visible et également l'intensité lumineuse infrarouge.



Ci-dessus : schéma interne du capteur TSL2591 : on remarque les 2 photodiodes permettant de « voir » dans le visible et l'infrarouge

Exemple de programme avec le TSL2591 :

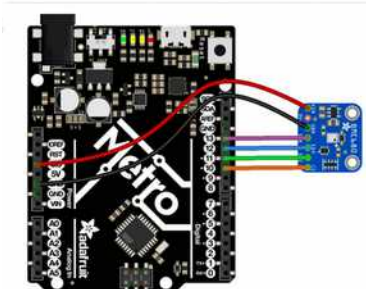
```
import board
import busio
import adafruit_tsl2591
i2c = busio.I2C(board.SCL, board.SDA)
capteurLumiere = adafruit_tsl2591.TSL2591(i2c)

intensiteLumiereLux = capteurLumiere.lux
intensiteLumiereVisible = capteurLumiere.visible
intensiteLumiereInfrarouge =  capteurLumiere.infrared

print('Intensité lumineuse: {0}lux'.format(intensiteLumiereLux))
print('Intensité lumière visible: {0}'.format(intensiteLumiereVisible))
print('Intensité lumière infrarouge: {0}'.format(intensiteLumiereInfrarouge))
```

Exercice :
A parti de données ci-dessus, propoer un programme en langage Python qui revoie les 4 messages suivants aucune lumière détectée ; lumière visible détectée ; lumière IR détectée ; lumière visible + IR détectée

Sources : https://learn.adafruit.com/adafruit-tsl2591/python-circuitpython
https://learn.adafruit.com/circuitpython-essentials/circuitpython-i2c

Exemple d'un capteur numérique de température ; humidité ; pression et qualité de l'air

Un des avantage des capteurs numériques est qu'il est possible d'intégrer plusieurs capteurs dans un seul boîtier et en plus, faire de la fusion de capteurs pour obtenir des informations complémentaires : par exemple, avec la température et la pression, le capteur peut déterminer l'altitude.
De même, avec la température et l'hygrométrie, on peut calculer le point de rosée (utilisée en air conditionné)



Ci-dessus : capteur BME680 (avec carte Adafruit) : capteur de temperature ; humidité ; pression et qualité de l'air ; à gauche : bus SPI ; au centre la carte Adafruit et à droite le capteur BME680

Avant de se lancer dans un programme Python, il sera nécessaire de télécharger la bibliothèque `adafruit_bme680` à l'adresse : https://github.com/adafruit/Adafruit_CircuitPython_BME680

Exemple de programme avec le BME680 :

```
import time
import board
from busio import I2C
import adafruit_bme680

capteurBME680 = adafruit_bme680.Adafruit_BME680_SPI(board.SCK, MISO=board.MISO)
cs = digitalio.DigitalInOut(board.D2)

# Pression de référence lorsqu'on est au niveau de la mer
bme680.sea_level_pressure = 1013.25

while True:
  temperature = capteurBME680.temperature
  pression = capteurBME680.pressure
  humidite = capteurBME680.humidity
  qualiteAir = capteurBME680.gas
  altitude = capteurBME680.altitude # grâce à la fusion de T et P, on peut trouver h

  print("\nTemperature: %0.1f C" % temperature)
  print("Pression: %0.3f hPa" % pression)
  print("Humidité: %0.1f %%" % humidite)
  print("Pollution: %d ohm" % qualiteAir)
  print("Altitude = %0.2f meters" % altitude)
  time.sleep(1)
```

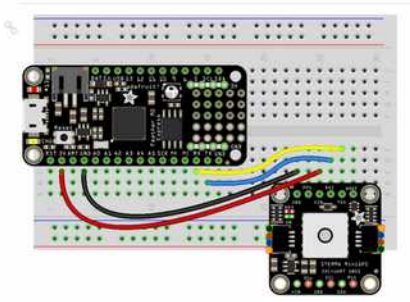
Exercice 1 : proposer un programme qui allume la led 13 lorsque la température dépasse 30°C

Exercice 2 : proposer un programme qui calcule le point de rosée.
https://fr.wikipedia.org/wiki/Point_de_rosée

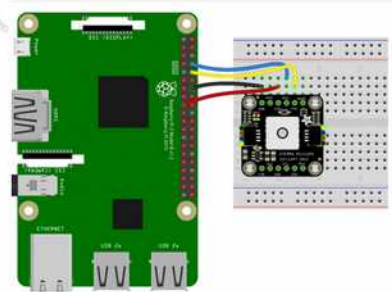
Sources : https://learn.adafruit.com/adafruit-bme680-humidity-temperature-barometic-pressure-voc-gas
https://learn.adafruit.com/circuitpython-basics-i2c-and-spi

Exemple d'un capteur numérique : module GPS

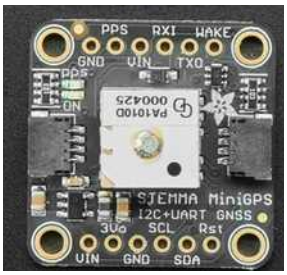
Le capteur numérique présenté ici est un capteur de position GPS construit autour du circuit MTK3333.
C'est un récepteur GPS qui peut faire l'acquisition de 33 satellites de manière simultanée.
Il peut calculer jusqu'à 10 positions par secondes.
Ce capteur est comptible avec les systèmes GPS (Américain) ; GLONASS (Russe) ; GALILEO (Européen) et QZSS (Japonais).



Connection avec un ATSAMD51



Connection avec un Raspberry Pi



Adafruit Mini GPS PA1010D Module

La connection entre le microcontroleur et le module GPS se fait au moyen de 4 fils : 0V ; 3,3V ; TX et RX.

Pour pouvoir programme le module GPS PA1010D d'Adafruit, il est nécesssaire de télécharger la bibliothèque `adafruit_gps` qui est disponible à l'adresse :
https://github.com/adafruit/Adafruit_CircuitPython_GPS

Exemple de programme avec le module PA1010D :

```
import time
import board
import busio
import adafruit_gps

uart = busio.UART(board.TX, board.RX, baudrate=9600, timeout=10)
gps = adafruit_gps.GPS(uart, debug=False)  # On utilise le port UART

# On active le GPS
gps.send_command(b"PMTK314,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0")

# On demande au GPS d'envoyer des données toutes les secondes
gps.send_command(b"PMTK220,1000")

dateDerniereAcquisition = time.monotonic()
while True:
  gps.update()
  dateAcquisitionActuelle = time.monotonic()
  if dateAcquisitionActuelle - dateDerniereAcquisition >= 1.0:
    dateDerniereAcquisition = dateAcquisitionActuelle
  if not gps.has_fix: # Le GPS n'est pas prêt
    print("Waiting for fix...")
    continue
    print("Latitude: {0:.6f} degrees".format(gps.latitude))
    print("Longitude: {0:.6f} degrees".format(gps.longitude))

if gps.satellites is not None:
  print("Nombre de satellites: {}".format(gps.satellites))
if gps.altitude_m is not None:
  print("Altitude: {} meters".format(gps.altitude_m))
```

Sources : https://learn.adafruit.com/adafruit-mini-gps-pa1010d-module
https://learn.adafruit.com/circuitpython-essentials/circuitpython-uart-serial