LABORATOIRE D'ELECTRONIQUE APPLIQUEE Q2

Cours destiné aux étudiants de première année en Technologie de l'Informatique et orientation sécurité des systèmes

Table des matières

Laboratoire sur Raspberry

Petits rappels (Raspberry) :

- Attention : La RPI[1] est alimentée en 5V par un adaptateur en 5V DC. **MAIS ATTENTION** : les composants (ce qui est connecté sur la GPIO) travaillent en 3,3V. Ceci signifie que c'est bien cette tension qu'il faut prendre en compte pour les calculs (de résistance sur une LED, par exemple)
- Attention : Une Led se raccorde **toujours** avec une résistance en série !
- La tension d'une Led varie en fonction de la couleur. (Rappel : Led verte = 2V, 20mA)
- La Raspberry est un SoC[2]. Il faut donc installer un OS[3] sur le support du système de fichiers.
- Sur la RPI, il n'y a pas, à priori, de disque dur. L'OS et tous les autres fichiers sont sur un support mémoire : la carte micro-SD.
- Pour l'installation de l'OS sur la carte mémoire, il faut graver une image.
- Installation d'une RPI **sans** intervention sur la **configuration** des **PCs du laboratoire.**
- L'installation d'une Raspberry peut se faire avec une intervention minimale sur une configuration quelle qu'elle fut. Plus qu'une contrainte, la démarche proposée ici offre un net avantage en terme de temps et de confort. Les seules connexions à réaliser seront l'alimentation et le réseau. On peut même pousser la simplification à la seule connexion de l'alimentation, si on est dans un environnement wifi disponible.

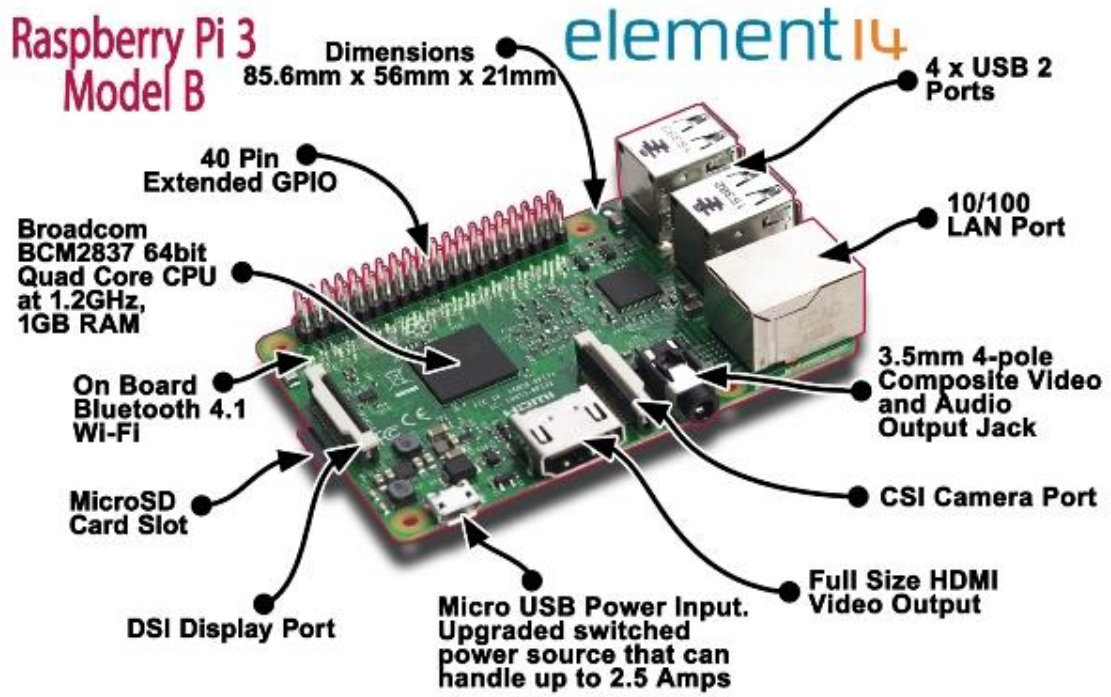


Image venant de : https://www.element14.com

[1] Raspberry
[2] SoC: System on Chip
[3] OS: Operating System

Manip 1 : Détection lumière (domotique)

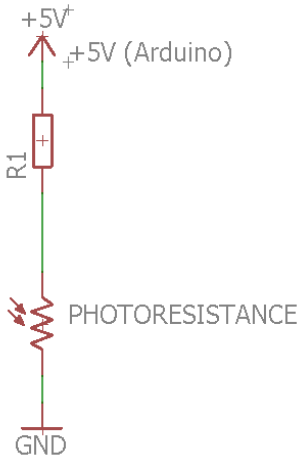
Avec les compétences que vous avez acquises sur l'Arduino, il faut réaliser un petit exercice qui consiste à détecter si nous avons de la lumière dans le local et en fonction de cette dernière, activer ou non une Led.

Arduino :

Pour ce faire, l'Arduino utilisera une photorésistance qui permet de mesurer un niveau de lumière. L'Arduino sera raccordée sur la RPI ! La programmation de l'Arduino se fera donc via l'application Arduino lancée sur la RPI et non le PC...
Je vous conseille fortement de procéder par étape :

1. Mesurer les variations de la R avec la lumière (via ohmmètre) de la photorésistance. L'Arduino ne sachant **pas mesurer ni une résistance ni un courant**, il va falloir passer à l'étape 2.
   Indiquez votre variation de résistance :
   Résistance dans le noir = ...
   Résistance à la lumière = ...

2. Réaliser un pont diviseur de tension pour avoir une **variation** de la **tension** en fonction de la lumière. Il faudra trouvez la résistance R1 à ajouter la plus adaptée pour avoir une variation de tension idéale de 0V à 5V mais ce sera impossible !! Faites des essais erreurs pour trouver la meilleure plage de tension avec des résistances R1 différentes ou utilisez la dérivée de la plage de tension aux bornes de la photorésistance par rapport à la résistance R1.
   Indiquez votre **variation de tension** : .....
   Indiquez votre **résistance R1** = ...



3. Maintenant que vous avez une variation de la tension en fonction de la lumière, vous pouvez raccorder votre montage à l'Arduino. Celui-ci doit mesurer la tension et la convertir en pourcentage de la lumière mesurée ! Affichez le pourcentage sur le moniteur série !
   Quelle fonction avez-vous utilisée pour lire la lumière ? : .....
   Quelle fonction avez-vous utilisée pour écrire le pourcentage sur le moniteur série ? : .....
   Utilisez la **fonction map** qu'Arduino a déjà réalisée pour vous !

4. Maintenant que vous avez la lumière en pourcentage, il va falloir :
   a. Eteindre la Led lorsque vous avez un pourcentage de lumière supérieur à 60%.
   b. Allumer la Led lorsque vous avez un pourcentage de lumière inférieur à 60%.

c. Utiliser un **hystérésis** pour qu'à plus ou moins 60% de lumière, il n'y ai pas de problème avec la Led.
d. Modifier l'allumage de la Led par un allumage **progressif** par rapport à la lumière. C'est à dire que lorsque le pourcentage de lumière arrive à 60% + l'hystérésis, la Led commence à s'allumer progressivement jusqu'à être complètement allumée lorsqu'on arrive à 0% de lumière et inversement !
Vous devez **afficher la progression de la Led sur le moniteur série**.
Qu'avez-vous utilisé comme fonction pour allumer/éteindre la Led progressivement ? : .....

RPI :

En envoyant des données sur le moniteur série, vous envoyez également des données sur la RPI via l'USB !

Le but de l'exercice sera d'afficher/enregister ses valeurs reçues sur la RPI toutes les secondes. Il faut, maintenant, un programme dans la Raspberry pour intercepter ces données et les sauvegarder dans un fichier. Il y a plusieurs façons de faire pour programmer sur la Raspberry, mais la plus utilisée sera d'utiliser le langage Python. C'est dans ce langage qu'est écrite la routine qui envoie l'adresse IP de votre Raspberry[4].

Pour composer ce programme, et ceci est vrai pour tous les langages de programmation, il va falloir passer trois épreuves :

- Lire des données sur un port USB
- Lire l'heure et la date
- Écrire les données dans un fichier avec l'heure en question
- Faire tourner un programme en boucle

Pour lire des informations sur un port USB, il faut ouvrir ce port. Une première restriction à laquelle nous devons faire face est qu'une ressource (port USB, fichier) ne peut être utilisée que par un process à la fois[5]. Pour identifier le port qu'il faut ouvrir, nous avons une très grande facilité.

Pour la programmation de l'Arduino, nous utilisons un IDE qui sélectionne ce port USB. Par ce biais là, il y a moyen de l'identifier. Si tel n'était pas le cas, il nous faudrait lister les ports USB et voir celui qui correspond à l'Arduino. C'est possible, mais c'est moins simple.

[4] Pour rappel, l'indentation fait partie de la syntaxe en python!
[5] pour faire court, car il y a moyen de contourner cet obstacle

Lire des données du port USB :

```
#!/usr/bin/env python
# coding: utf-8 -*-

import serial
import time

portUSB='/dev/ttyUSB0'

try:
    ser=serial.Serial(portUSB,baudrate=9600,timeout=1,writeTimeout=1)

    while True:
        ligne=ser.readline()
        heure=time.strftime('%y-%m-%d %H:%M:%S')
        print "information prise à {}:{}".format(heure, ligne)

except:
    print "Problème avec le port USB"
```

Quelques petites explications :

Les "import" ne doivent pas poser de problèmes fondamentaux. On a besoin de l'heure et d'accéder aux ports série. Donc, on charge les librairies qui conviennent.

Le "portUSB" est une variable qui doit contenir le **vrai** nom de votre port USB. (Voyer sur l'IDE de l'Arduino pour être sûr).

"try: - except:" est une façon d'intercepter des erreurs si il y en a. Si tout se passe bien, le programme sera normalement exécuté dans "try".

Si, dans la zone couverte par le "try", il y a une erreur (quelle qu'elle soit), alors une exception est générée, et la routine "except :" est exécutée. On peut donc avoir le message d'erreur lié au port USB si on a une division par "0" dans le "try".

"while True" Toujours vrais, donc on tourne en boucle.

Ligne = ser.readline() : sans commentaire!

Heure = time.strftime(...). C'est la lecture de l'heure. Cette heure est mise en forme avec la chaine de caractères qui est indiquée entre parenthèses. Celle qui est indiquée n'est pas la bonne. Quel est le format utilisé pour l'heure et la date sous Calc ? (Corrigez le format).

Ecriture des données dans un fichier

Maintenant que nous avons les données de l'Arduino, il est temps d'en faire quelque chose. Par exemple, les ranger dans un fichier.

Quel fichier et pour quoi faire ? Dans un premier temps, nous allons utiliser ces données dans un tableur (Calc). La structure d'un fichier Calc en tant que tel peut être très compliqué. Pour contourner la difficulté, nous allons utiliser un format qui permet une importation dans un tableur. C'est le format "csv" qui est, en fait, un fichier texte.

Lors de l'importation d'un fichier "csv", les données doivent être séparées par des caractères convenus à l'avance. Nous allons choisir notre séparateur en fonction des informations que nous allons exploiter.

Les informations que nous allons sauvegarder vont être de types différents, mais, nous aurons :

- Du texte
- Des dates et heures

Le texte, c'est … du texte. Mais l'heure et la date sont présentés dans les tableurs, de façon structurée. Nous aurons pour :

- La date : jj/mm/aaaa
- L'heure : hh:mm:ss

Nous voyons ici que les caractères "/" et ":" vont être *réservés* pour ces formats. Nous allons choisir le *point-virgule*[6] comme séparateur. Pour bien identifier les données, nous allons les nommer en donnant un nom aux colonnes.

[6] ";" ou "semicolon" en anglais

Voici une ébauche de programme :

```
#!/usr/bin/env python
# coding: utf-8 -*-

import serial
import time

fichierDeSortie="alarmeDatas.csv"
repDeSortie="/home/pi/tests"
modeDeFichier="a"
listeColones="Date;Heure;évènement"

portUSB='/dev/ttyUSB0'

fichier = open(repDeSortie+fichierDeSortie, 'w')
fichier.write("Date"+";"+"Heure"+";"+"Evénement")
fichier.close()

try:
    ser=serial.Serial(portUSB,baudrate=9600,timeout=1,writeTimeout=1)

    while True:
        ligne=ser.readline()
        date=time.strftime('%y-%m-%d')
        heure=time.strftime('%H:%M:%S')
        print "information prise le {} à {}:{}".format(date, heure, ligne)
        fichier = open(repDeSortie+fichierDeSortie, modeDeFichier)
        fichier.write(date+";"+heure+";"+ligne)
        fichier.close()

except:
    print "Problème avec le port USB"

```

Adaptez le programme pour exploiter les données dans Calc. Le programme ne doit sauvegarder que les données utiles[7]. (Ne pas oublier d'adapter les formats de données).

[7] On considère qu'un événement est un changement d'état d'un système.

Pense-bête :

Commandes utiles :

- "sudo" : pour exécuter une commande avec des droits d'administrateur.
- "nano" : pour éditer un fichier en mode texte. (Permet également de créer un fichier)
- "geany" : pour éditer un fichier en mode graphique. (Permet également de créer un fichier voire un dossier et de l'exécuter)
- "rm" : supprime un fichier
- "rmdir" : supprime un dossier vide
- "rm -r" : supprime un dossier non vide
- "mkdir" : créé un dossier
- "cd" : voyage dans les dossiers
- "ls" : liste les fichiers et dossiers
- "reboot" : redémarre la RPI
- "shutdown" : éteint la RPI
- "python" ou "python3" : Lance le programme python (avec la version 2 ou 3)
- "curl www.icanhazip.com" pour connaitre son adresse IP externe (IPv4)
- "scp" : transfert de fichier vers/de un ordinateur ayant un serveur SSH actif (linux)
- "winscp" : … devinez?
- "touch" : créé un fichier vide ou met à jour la date de modification du fichier
- "iwlist wlan0 scan" : liste les réseaux Wifi détecté par la RPI
- "wpa_passphrase" : permet de crypter le mot de passe du Wifi
- "soffice --calc VotreFichierALire" : permet de lire un fichier CSV avec calc