Systèmes embarqués

Introduction à la programmation microcontrôleurs
et à l'impression 3D.

Pape Abdoulaye BARRO, Ph.D.

Enseignant-chercheur
UFR des Sciences et Technologies
Département Informatique

E-LabTP, Laboratoire des TP à Distance, UFR-SET,
Marconi-Lab, Laboratoire de Télécommunications, ICTP, Italie

Email: pape.abdoulaye.barro@gmail.com

- Généralités
- Architecture et familles de microcontrôleur
- Capteurs et actionneurs
- Programmation des microcontrôleurs
  - Programmation sous Arduino
    - Etudes de cas
  - **Programmation sous Raspberry Pi**
    - Etudes de cas
- Initiation à l'impression 3D

PLAN

Programmation des microcontrôleurs

La programmation sous Raspberry Pi

La programmation sous Raspberry Pi

Généralités

- Dans ce chapitre, nous allons utiliser un nano ordinateur avec un écran hautement défini pour traiter les données collectées par des capteurs: Raspberry Pi .

> Raspberry Pi est un nano-ordinateur, de la taille d'une carte de crédit, équipé d'un microprocesseur ARM, de la mémoire RAM, d'une carte vidéo, d'une carte Ethernet, du Wi-Fi et du Bluetooth, d'un connecteur USB-2 et d'un connecteur HDMI.

- Il a fait son apparition (en publique) en 2012 avec comme objectif principal d'offrir aux étudiants et autres intéressés, un outil expansible et très accessible leur permettant d'apprendre plus efficacement l'informatique en général et la programmation en particulier.
- En 2006, les premiers prototypes sont développés sur des ***microcontrôleurs Atmel*** ATmega 644.

La programmation sous Raspberry Pi

Caractéristiques

- Les Raspis sont très attrayants grâce à leurs périphériques de bas niveau intégrés.
  - La carte comporte des broches d'entrée/sortie à usage général (GPIO), dont certaines prennent en charge les communications I2C, SPI et UART (de type RS-232);
  - De plus, un bus audio spécifique (I2S) est disponible, ainsi qu'une interface CSI haute vitesse pour connecter la caméra Pi sur mesure, et une interface DSI pour connecter les panneaux LCD;
  - Certaines de ces caractéristiques peuvent être utilisées pour mettre en œuvre la même fonctionnalité que sur l'Arduino;
  - Il peut être utiliser également comme système informatique hôte standardisé (Unité central).

La programmation sous Raspberry Pi

Installation et configuration

- Raspberry Pi est livrée sans système d'exploitation. Il fonctionne avec plusieurs variantes de Linux, notamment, Debian et autres logiciels compatibles.
- Par défaut, il utilise Raspbian qui est un système d'exploitation optimisé et gratuit basé sur Debian. Il existe plusieurs versions de raspbian :
  - Wheezy du Debian 7;
  - Jessie du Debian 8;
  - Stretch du Debian 9;
  - Buster du Debian 10

La programmation sous Raspberry Pi

Installation et configuration

Installation

- Avant de procéder à l'installation du Raspberry Pi, il faut disposer, d'une SD carte d'au moins 16Go, d'une image de raspbian et d'un installateur/Extracteur (logiciel) d'image.
  - Pour Raspbian, une archive (.zip) est disponible sur:

    https://www.raspberrypi.com/software/operating-systems/
  - Pour le logiciel, il s'agit de Win32DiskImager (il en existe d'autre) accessible sur: https://sourceforge.net/projects/win32diskimager/
  - Il est également possible d'utiliser un installateur qui se chargera d'extraire une image et puis de l'installer sur la carte. Il est disponible sur le site:

    https://www.raspberrypi.com/software/
- Sur un ordinateur, charger Raspbian sur la carte SD, éjecter et insérer la à son emplacement sur le Raspberry et puis brancher la carte.

La programmation sous Raspberry Pi

Installation et configuration

Configuration

- Au démarrage, vous devez renseigner le login (*pi*) et le mot de passe (*raspberry*). À noté que le clavier est en *QWERTY*.

La programmation sous Raspberry Pi

Installation et configuration

Configuration

- Il existe deux manière d'accéder au menu de configuration du raspberry Pi:
  - Soit en ligne de commande en tapant au console: sudo raspi-config
  - Ou directement dans le Menu principal, puis Préférences, ensuite Configuration du Raspberry Pi.
  - Vous pouvez configurer:
    - Le clavier
    - La localisation
    - Le fuseau horaire
    - La prise en main à distance via SSH
    - L'utilisation des ports GPIO
    - La Caméra
    - …
  - Si vous désirez changer le mot de passe taper sur le terminal: sudo passwd
- Redémarrer ensuite la carte en tapant: sudo reboot

La programmation sous Raspberry Pi

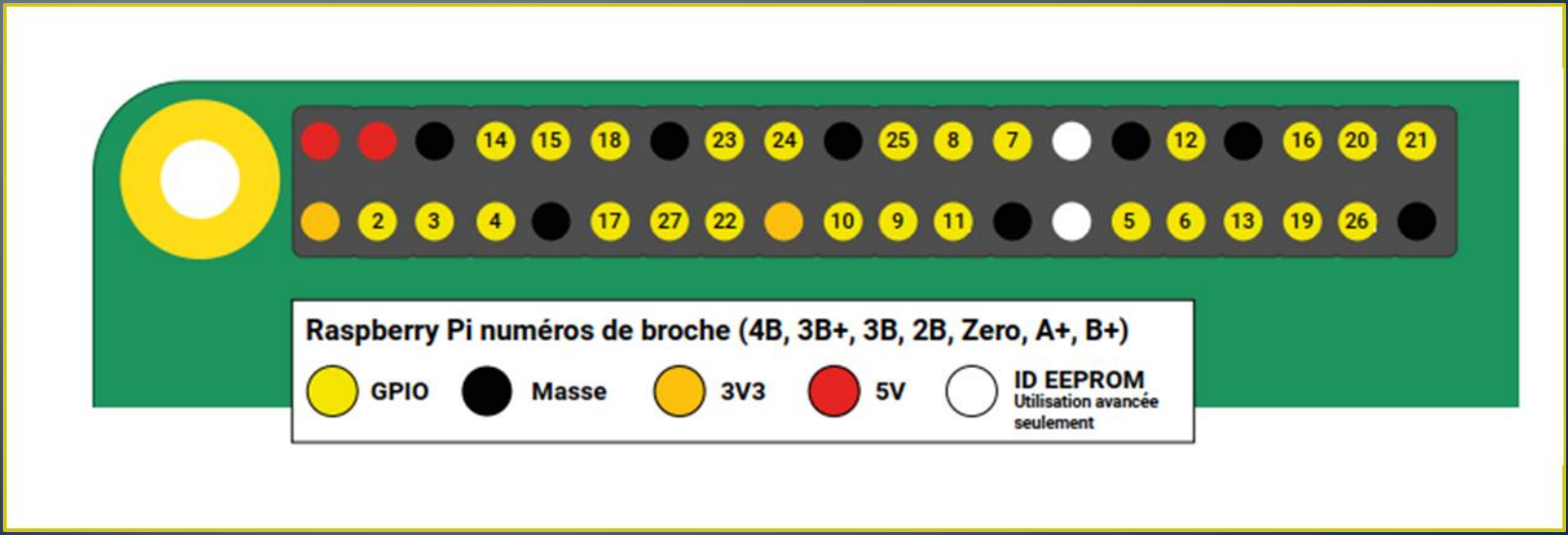
GPIO avec Python

- Le port GPIO (*general-purpose input/output*) vous permet de connecter des périphéries comme des LED et des interrupteurs au Raspberry Pi et de pouvoir les commander via un programme que vous devez créer. Les broches peuvent être utilisées aussi bien en entrée qu'en sortie.
- Pour utiliser les ports GPIO, il faut installer le module *RPi.GPIO* à l'aide de la ligne de commande :
  - sudo apt-get install RPi.GPIO
- Pour les utiliser avec un programme Python, il faut installer la bibliothèque Python-GPIO:
  - sudo apt-get update
  - sudo apt-get install python-dev
  - sudo apt-get install python-rpi.gpio

La programmation sous Raspberry Pi

GPIO avec Python

- Le connecteur GPIO est constitué de 40 broches mâles dont certaines sont destinées aux projets d'informatique physique, d'autres sont consacrées à l'alimentation, d'autres encore sont réservées pour la communication. Ci-dessous, une image du connecteur.

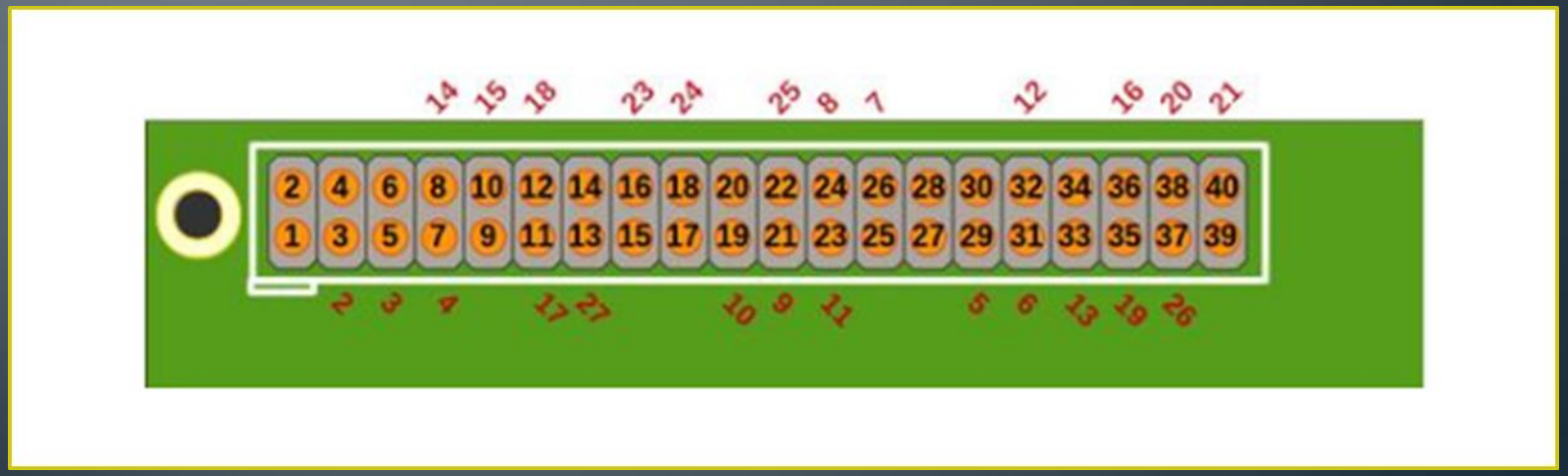


La programmation sous Raspberry Pi

GPIO avec Python

Numérotation des ports GPIO

- La bibliothèque *RPi.GPIO* supporte différentes méthodes pour accéder aux ports, nous en distinguons deux(2):
  - le mode *BCM (en rouge)* qui correspond à la numérotation électronique de la puce;
  - le mode alternatif *BOARD (en noire)* qui correspond à celle de la sérigraphie du connecteur de la carte.



La programmation sous Raspberry Pi

GPIO avec Python

- Pour pouvoir utiliser les GPIO avec Python, il faut utiliser un IDE de programmation python sur Raspbian (nous utilisons *Thonny*).
- On peut parcourir quelques notions de base en Python sans entré dans les détails (puisse qu'on s'adresse à un publique déjà informé). Nous allons voir les types, les variables, les conditions, les boucles, les fonctions et les bibliothèques.
  - Un type caractérise le contenu d'une variable. Il peut s'agir d'un chiffre, d'un caractère, d'un texte, d'une valeur de type vrai ou faux, d'un tableau de valeurs, etc.
    - Python est un langages à typage dynamique,
      - ce qui veut dire qu'il n'est pas nécessaire de déclarer les variables avant de pouvoir leur affecter une valeur.
      - Le type de données peut aussi changer en cours de l'exécution du programme.
      - La fonction type() permet de connaitre le type de la valeur d'une variable. Par exemple, si a=10, alors type(a) donnera *int*.
      - Les types sont: int pour les nombres entiers, float pour les nombres à virgules flottante, str pour les chaines de caractères, bool pour les booléens, list pour une collection d'éléments séparés par des virgules, complex pour les nombres complexes, etc. .
      - À partir des types de base, il est possible d'en élaborer de nouveaux comme :
        - Le tuple qui est une collection ordonnée de plusieurs éléments. Par exemple a=(3, 7, 10);
        - Le dictionnaire qui est un rassemblement d'éléments identifiables par une clé. Par exemple d={"x": 4, "y": 2};

La programmation sous Raspberry Pi

GPIO avec Python

- Une variable est une zone mémoire dans laquelle une valeur est stockée. En Python, la **déclaration** d'une variable et son **initialisation** se font en même temps. Par exemple, a=2. Il faut cependant respecter les règles usuelles suivantes:
  - Le nom doit commencer par une lettre ou par un underscore ;
  - Le nom d'une variable ne doit contenir que des caractères alphanumériques courants;
  - On ne peut pas utiliser certains mots réservés.
- On utilise une structure conditionnelle pour faire exécuter un bout de code si certaines conditions sont remplies. Les structures conditionnelles en Python sont:
  - La condition if ("si") ;
  - La condition if…else ("si…sinon") ;
  - La condition if…elif…else ("si…sinon si… sinon") .

  Ici, les opérateurs de comparaisons et les opérateurs logiques (and, or, not) sont largement utilisés.
- Une boucle est utilisée pour exécuter en plusieurs fois un bloc d'instructions tant qu'une condition donnée est vérifiée. Nous avons accès à deux boucles en Python :
  - La boucle while ("tant que…") dont le nombre d'itération n'est pas connu à l'avance;
  - La boucle for ("pour…") dont le nombre d'itération est bien connu. Avec la boucle for, on peut utiliser la fonction range() pour définir une plage de valeurs. Exemple:
    - range(5) permet de générer les valeurs 0, 1, 2, 3 et 4;
    - range(5, 10) permet de générer les nombres 5, 6, 7, 8 et 9;
    - range(6, 10, 2) permet de générer les nombres entre 6 et 10 par pas de 2 (6, 8 et 10);
  - L'instruction break permet de stopper l'exécution d'une boucle lorsqu'une certaine condition est vérifiée.
  - L'instruction continue permet elle d'ignorer l'itération actuelle de la boucle et de passer directement à l'itération suivante.

La programmation sous Raspberry Pi

GPIO avec Python

- Lorsqu'on a à ré-écrire plusieurs fois un bout de code dans un programme, avec éventuellement des changements de données, il est préférable de le regrouper sous un nom unique appelé fonction. Pour déclarer une fonction en Python, la syntaxe est la suivante:

```
def nom_fonction(<paramètres>):
    instructions
```

Il existe des fonctions prédéfinies en Python. Exemple print(), input(), etc. .

- Une bibliothèque est un ensemble de modules (classes, fonctions,…) ajoutant des possibilités étendues à Python. Pour utiliser les fonctionnalités d'une bibliothèque, il va falloir l'inclure dans le programme en question avec le mot clé *import* comme suit:

```
import nom_de_la_bibliothèque
```

La programmation sous Raspberry Pi

GPIO avec Python

- L'utilisation des GPIOs nécessite l'inclusion de la bibliothèque dans le fichier (d'extension *.py*) en créant un objet *GPIO* comme suit:
  - *import RPi.GPIO as GPIO*

- On peut également importer la bibliothèque *time* propre à python, dans laquelle est contenue la fonction *sleep* avec laquelle vous pouvez définir facilement des temps d'attente dans un programme, comme suit:
  - *import time*
  - time.sleep(5) *//attente de 5s*

La programmation sous Raspberry Pi

GPIO avec Python

- Il va falloir définir le mode de numérisation également:
  - GPIO.setmode(GPIO.BOARD) // mode board
  - GPIO.setmode(GPIO.BCM) // mode bcm
- Pour déclarer et initialiser une entrées-sorties (E/S), il suffit de préciser son numéro, son mode d'utilisation (entrée ou sortie) et éventuellement son état initial (*mode sortie uniquement*) comme suit:
  - GPIO.setup(6, GPIO.IN) // broche 6 comme entrée numérique
  - GPIO.setup(8, GPIO.OUT) // broche 8 comme sortie numérique
  - GPIO.setup(8, GPIO.OUT, initial=GPIO.HIGH) // broche 8 est une sortie initialisée a l'état haut.
- l'état des E/S, le module *RPi.GPIO* accepte des variables dédiées (*GPIO.HIGH* ou *GPIO.LOW*), des entiers (*1* ou *0*) ou des booléens (*True* ou False).

La programmation sous Raspberry Pi

GPIO avec Python

- Pour changer l'état d'une sortie numérique, on procède comme suit:
  - GPIO.output(8, GPIO.LOW) // la broche 8 est mise à BAS
- Il est possible de connaitre l'état d'une entrée en utilisant la fonction input.
  - GPIO.input(6) // on interroge la broche 6 pour connaitre son état
- Pour connaitre la configuration d'une entrée/sortie numérique, on utilise la fonction *gpio_funtion*. Les valeurs renvoyées sont alors GPIO.INPUT, GPIO.OUTPUT, GPIO.SPI, GPIO.I2C, GPIO.HARD_PWM, GPIO.SERIAL ou GPIO.UNKNOWN
  - state=GPIO.gpio_funtion(8)
  - print(state) // affiché dans la console la configuration de la broche
- A la fin du programme, il est conseillé d'effectuer une purge des ressources en utilisant la fonction *cleanup*.
  - GPIO.cleanup() // libère toutes les entrées/sortie utilisées

La programmation sous Raspberry Pi

GPIO avec Python: exemple de Blink

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.OUT)

nbreRepetition = input("donner le nombre de fois que la LED doit clignoter?\n")

i=0
while i < nbreRepetition :
    GPIO.output(21, True)
    time.sleep(1)
    GPIO.output(21, False)
    time.sleep(1)
    i = i+1

GPIO.cleanup()
```

La programmation sous Raspberry Pi

Lancement du programme au démarrage

- En informatique physique, on aura besoin d'exécuter un programme au démarrage de la carte. Or sur Raspberry pi, ce mécanisme n'est pas directement pris en compte. Ce sera donc au programmeur de le gérer.
- La méthode la plus simple de lancer un programme au démarrage de la Raspberry Pi est d'utiliser le fichier rc.local qui se trouve dans /etc.
  - Ce script est sensé contenir des lignes de commandes qui seront exécutées juste avant que la Raspberry Pi n'ai fini de booter. Le script se termine par exit 0, indiquant la fin des exécutions. Il va falloir donc ajouter votre programme juste avant.
  - Si, par exemple, votre programme est nommé test.py et se trouvant dans /home/pi, alors on écrit: /usr/bin/python3 /home/pi/test.py
  - Il va falloir faire très attention car votre programme doit rendre la main au script pour que le Raspberry Pi puisse finir de booter. Votre programme doit de ce fait, terminer son exécution oubien s'il doit tourner en boucle infinie, on doit alors le lancer en tache de fond en ajoutant un & après la commande comme suit: /usr/bin/python3 /home/pi/test.py &

À suivre

Feedback sur:

pape.abdoulaye.barro@gmail.com