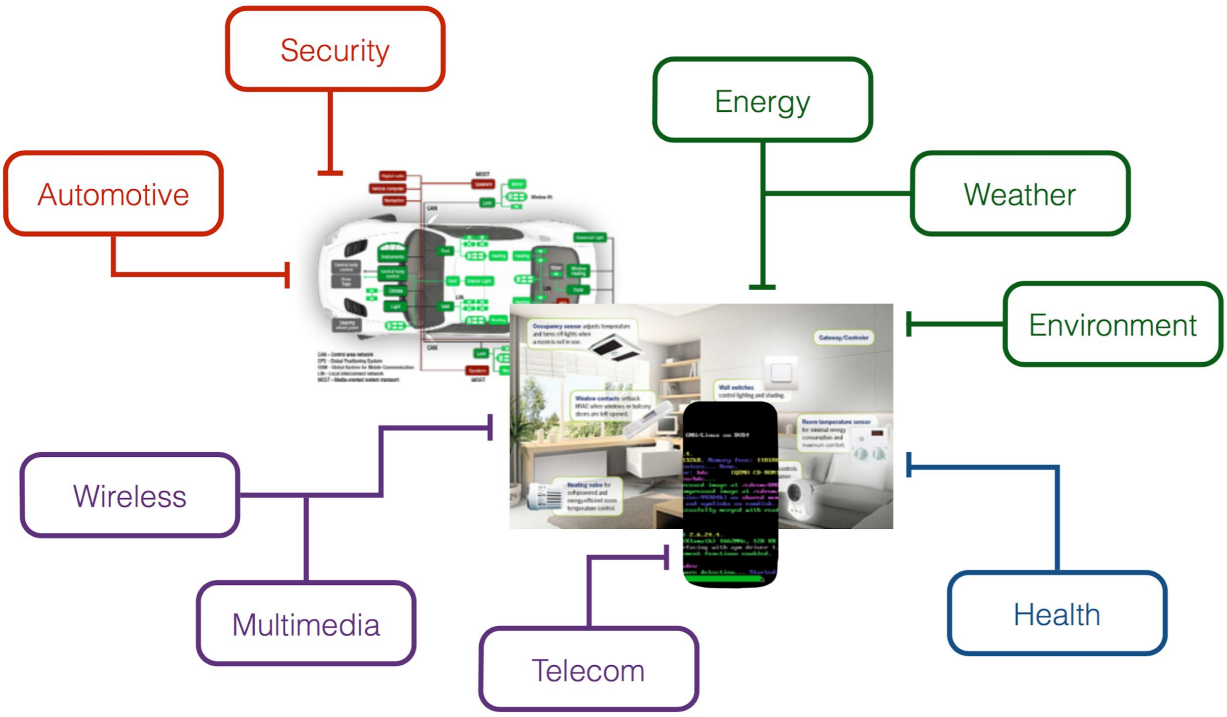
Présentation Générale

Module IOC — MU4IN109
Franck Wajsbürt

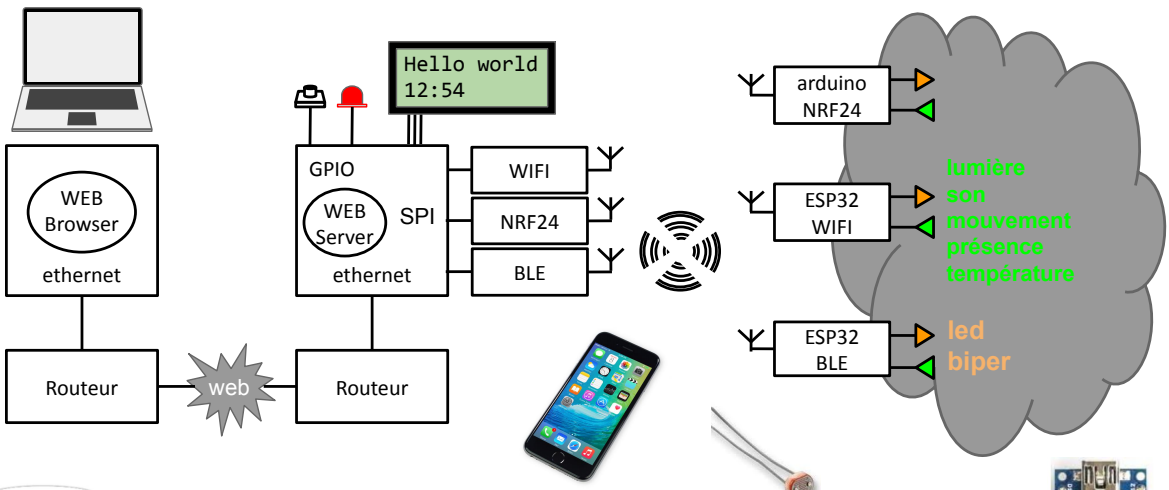
Domaine du cours

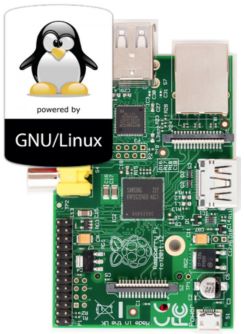


Plan

- Plateforme cible et Thèmes abordés
- Informations sur les unités de calculs utilisées
- Périphériques passifs avec un peu d'électronique
- Conversion Analogique ↔ Numérique

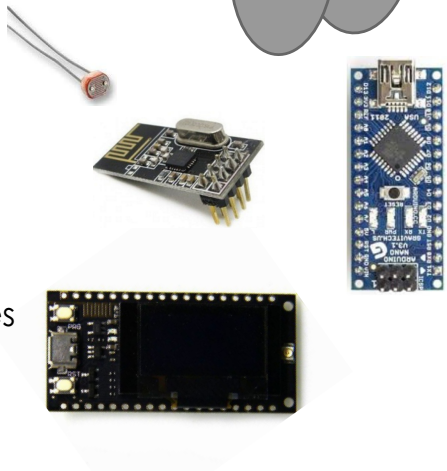
Plateforme cible



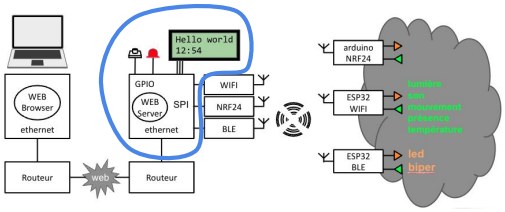


Thèmes abordés

- Pilotes Linux
- protocole filaire
- protocole radio
- server web
- Systèmes embarqués
- Base de données
- Publish & Subscribe

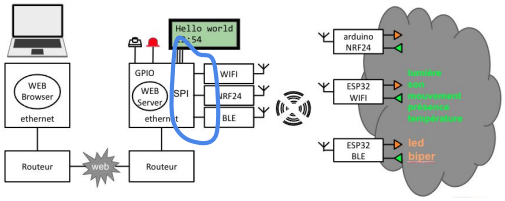


Pilotes Linux



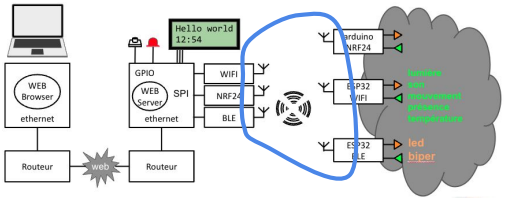
- Objectif
  - Comment interagir avec le matériel depuis une application utilisateur ?
- Matériels
  - Raspberrypi 1 (il n'y en a que 7…)
  - LED, bouton-poussoir et afficheur LCD
  - Programmation depuis un PC du réseau (cross-compilation)
- Étapes
  - Sans changer le noyau, mais en donnant des droits à l'application pour l'accès aux registres des périphériques
  - En créant un driver (pilote) caractère dans Linux pour que l'application accède au device (périphérique) avec l'API POSIX standard (open, read, write, close, etc.)
  - D'abord ce sera avec les LEDs et le BP, puis plus complexe le LCD

Protocoles filaires



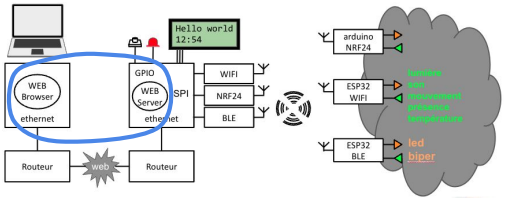
- Objectif
  - Les devices sont en général assez complexes et le processeur central du système ne les commande pas dans le détail. Les devices dispose de leur propre processeur, parfois plus complexe que le processeur de l'application. Entre les devices et le processeur central, on utilise des fils avec des protocoles. On verra quelques protocoles.
- Matériels
  - Raspberrypi 1 pour le LCD
  - ESP32 pour RS323, I2C et SPI
  - Programmation depuis un PC du réseau (cross-compilation)
- Étapes
  - En fait, hormis pour le LCD, on ne verra pas le détail des signaux parce qu'on utilisera des API qui vont abstraire la communication, mais pour comprendre certains paramètres, il faut comprendre le principe de fonctionnement.

Protocoles radio



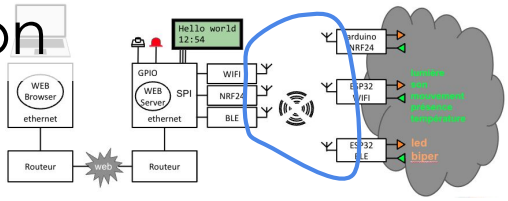
- Objectif
  - Les systèmes communicants sont presque toujours des systèmes distribués qui utilisent des liaisons sans fil entre le processeur central et les capteurs/actionneurs. On verra quelques protocoles.
- Matériels
  - Raspberrypi 3 pour la station de base
  - ESP32 pour les capteurs/actionneurs
  - Protocole WIFI, peut-être BLE et LORA,
- Étapes
  - Etant donné la complexité des couches protocolaires, il n'est pas envisageable de commander directement le matériel. On utilisera donc des API.
  - Pour les plus énervés, on pourrait contrôler le protocole NRF24 avec les Raspberrypi 1 et des Arduino Atmel

Serveur WEB + BDD



- Objectif
  - Un système embarqué doit souvent être accessible à distance, que ce soit pour l'interroger ou pour le contrôler. La méthode la plus simple consiste à le connecter à Internet et à utiliser un navigateur WEB.
    Cela nécessite de créer un serveur pour répondre aux requêtes du navigateur, permettant d'accéder indirectement à l'application et, souvent, à une base de données (BDD ou DB) pour gérer un historique..
- Matériels
  - PC Linux
  - Raspberrypi 1 puis 3 pour la station de base
  - ESP32 pour les capteurs/actionneurs
- Étapes
  - On va d'abord créer un couple client-serveur sur un seul PC avec une application TCP-IP puis un serveur HTTP (pour le WEB)
  - Puis, sur une Raspberrypi 1 pour contrôler les LED, BP, et LCD
  - Enfin, on mettra le serveur WEB sur les Raspberrypi 3 avec une BDD

Protocole de communication



- Objectif
  - Sur Internet les protocoles de communications dépendent des applications qui communiquent : HTTP pour les clients-serveur WEB, SMTP pour le mail, mais aussi DNS, FTP, SSH, etc.
    Il y a aussi des protocoles spécialisés pour les échanges avec les capteurs/actionneurs, tels que MQTT. Nous verrons que MQTT repose sur un principe très simple de publish/subscribe.
- Matériels
  - PC Linux
  - Raspberrypi 3 pour la station de base
  - ESP32 pour les capteurs/actionneurs
- Étapes
  - On ne verra que le protocole MQTT/WiFi, mais si certains veulent utiliser NRF24, il faudra créer son propre protocole (en s'inspirant de MQTT)

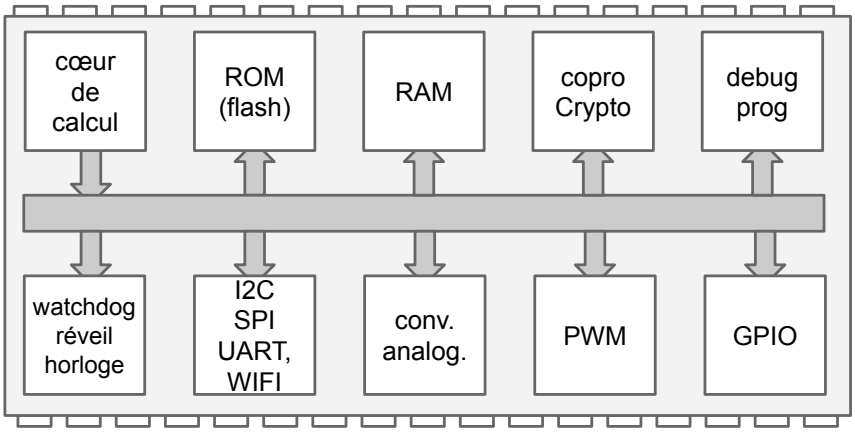
Unités de calculs

nano-computer

- Les réseaux de capteurs sont constitués de "*nano-computers*" construits autour d'un SoC interconnectés en filaire ou par radio.
- Un *nano-computer* est un ordinateur complet de la taille d'une carte de crédit ou d'un ticket de métro, avec :
  - un ou plusieurs processeur(s)
  - ram + rom (flash)
  - nombreux contrôleurs d'entrées-sorties
  - périphériques d'entrées-sorties basiques (BP, LED, écran, HP)
  - connecteurs
  - chargeur de batterie
  - etc….
- Exemples utilisés en TP
  - Raspberry PI 1 et 3,
  - TTGO LoRa 32
  - *Arduino nano,*

Que contient un SoC ?

Un SoC ou System-On-Chip rassemble au moins 1 cœur de calcul des contrôleurs de périphériques et de la mémoire.
(un MCU pour Micro Controller Unit est un *petit* SoC)

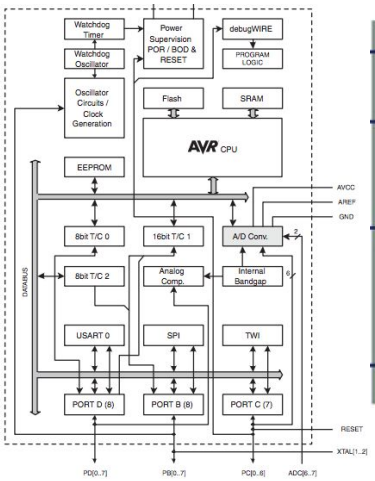


Peu de composants autour sont nécessaires pour réaliser un *nanocomputer*

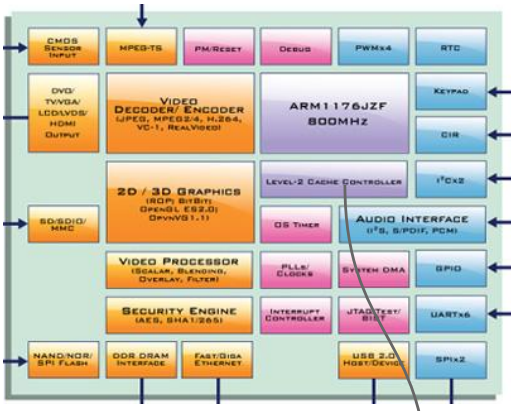
Un SoC contient beaucoup de choses !

Les documentations font des centaines de pages mais la plupart des fonctionnalités sont abstraites par des drivers ou des API spécifiques.

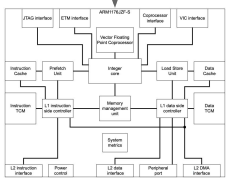
Nous allons utiliser le C / C++ avec Linux/Raspberry et Arduino/ESP32

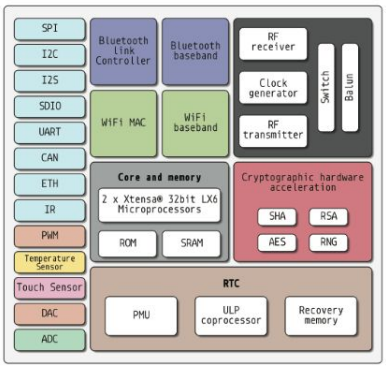


Atmega328p
(Arduino)



BCM2835
(Raspberry-Pi 1)



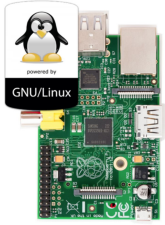


XTENSA LX6
(ESP32)

Raspberry Pi

- Université de Cambridge
- Basé sur des SoC Broadcom / ARM
- Supporte OS généralistes : Linux, Windows IOT, Android, RiscOS
- Raspberry Pi 1 est sortie en février 2012, la Rpi4 en 2019
- Modèles utilisés en TP :

- Raspberry Pi 1B (2012)



  - BCM2835 (1 core) + GPU
  - ARM V6 32 bits
  - 700 MHz
  - 256 MB
  - SDcard
  - 2 USB
  - ethernet 10/100 MHz
  - HDMI FullHD + son
  - 8 GPIO
  - I2C, 2 SPI, UART, PWM
  - 200mA (alim 1 A)

- Raspberry Pi 3 (2016)

  - BCM2837 (4 cores) + GPU
  - ARM V8 64 bits
  - 1200 MHz
  - 1024 MB
  - SDcard
  - 4 USB
  - ethernet 10/100 MHz,
  - HDMI FullHD + son
  - 17 GPIO
  - I2C, 2 SPI, UART, PWM
  - 800mA (alim 2.5 A)
  - WIFI 802.11ac, Bluetooth 4.1

Arduino[1]

- Interaction Design Institute Ivrea (Italie)
- Intègre initialement des SoC (System On Chip) Atmel 8 bits
  mais aussi des 32 bits : Xtensa, ARM Cortex-M0/M3 et Intel Quark x86
- Basé sur le projet Wiring 2003[2] (Hernando Barragán)
- Exécutif spécifique (Inspiré de Processing[3] MIT 2001 *pour Graphiste*)
- Projet Open Source pour le code et le matériel ⇒ beaucoup de clones

- Arduino Uno (2005)
  - Atmega328p
  - 5V
  - core 8 bits
  - 16 MHz
  - Flash 32 kB + EEPROM 1 kB + SRam 2 kB
  - 14 GPIO dont 6 PWM
  - I2C, SPI, UART,
  - 8 Entrées Analogiques 10 bits
  - 20 mA + 40 mA par I/O
  - LED + 1 bouton reset

1. https://www.wikiwand.com/fr/Arduino
2. https://www.wikiwand.com/fr/Wiring
3. https://www.wikiwand.com/fr/Processing

ESP32

- Espressif Systems (Shanghai)
- *nanocomputers* pour réseaux de capteur radio
- Intègre des SoC Xtensa dual-core (or single-core) 32-bit LX6 (2016)
- Version précédente : ESP8266 (2014)
- Plusieurs Exécutifs spécifiques SDK en C++, LUA, Python, C, Go
- Supporte Arduino et aussi Free RTOS

- Module TTGO-Lora32-Oled (Wemos)
  - XTENSA LX6
  - 3.3V
  - dual-core 32 bits
  - 240 MHz
  - Flash 32 MB
  - SRam 512 kB
  - 28 GPIO (multi-fonctions)
  - I2C, SPI, UART, 18 Entrées Analogiques (12 bits)
  - 50 mA
  - LED + 1 bouton reset
  - WIFI 802.11 bgn, Bluetooth 4.2,
  - LoRa 868 MHz (SX1276)

ESP32-C3

- Espressif Systems (Shanghai)
- *nanocomputers* pour réseaux de capteur radio
- Intègre un RISC-V !
- Supporte ESP-IDF, Arduino et micro-python
- Module ESP32-C3-0.42LCD
  - RISC-V 32 bits
  - 3.3V
  - 240 MHz
  - Flash 4 MB
  - SRam 400 kB
  - 11 GPIO (multi-fonctions)
  - I2C, SPI, UART, 4 Entrées Analogiques
  - LED RGB + 1 bouton reset
  - WIFI 802.11 bgn, Bluetooth 5 LE,

Périphériques

Que connecte-t-on à un SoC ?

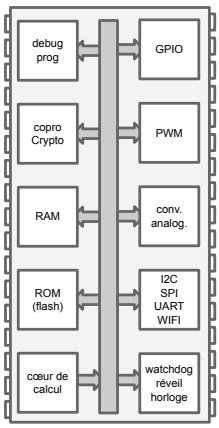
Les périphériques qui vont permettre de se connecter à l'environnement.

Entrées capteurs

lumière, micro, hygromètre, mouvement, détecteur de gaz, proximité, torsion, pression, thermomètre, accéléromètre, GPS, ...

IHM

clavier, télécommande, surface tactile, voix, ...



Sorties actionneurs

moteurs CC ou PàP, relais, HP, leds, ...

IHM

écran LCD, buzzer, voix, ...

- Il existe une très grande variété de capteurs et d'actionneurs mais le nombre d'interface est plus réduit en passant par des protocoles de communication standard : rs232, CAN, I2C, SPI, USB, WIFI, BLE, LoRa
- Les capteurs et actionneurs contiennent souvent des µC pour réduire le travail de l'application ou réduire la quantité d'informations à échanger.

Deux types de périphériques

- Périphériques passifs

  Lecture / écriture directe d'une grandeur électrique numérique [0, 1] ou analogique (OV à 3.3V)

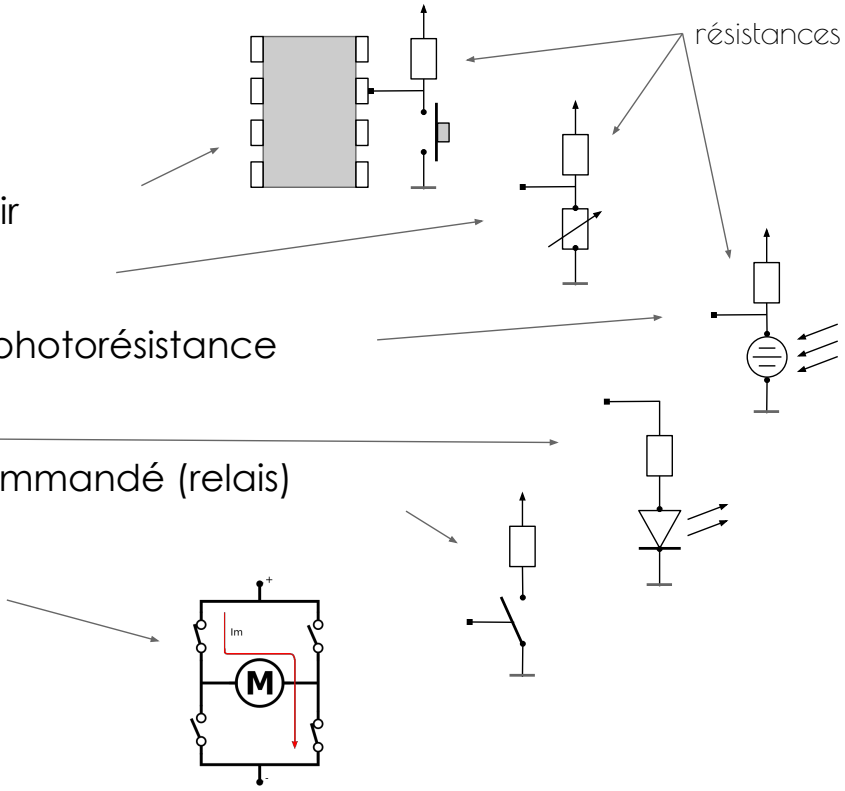
- Périphériques actifs

  Un contrôleur de périphérique offrant une abstraction sous forme de registres de contrôle accessibles par un protocole de communication avec ou sans fils.

  En général, ces périphériques intègrent ...
  un micro-contrôleur !

Périphériques passifs basiques

- bouton poussoir
- potentiomètre
- thermistance, photorésistance
- LED
- interrupteur commandé (relais)
- moteur(s)

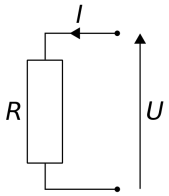


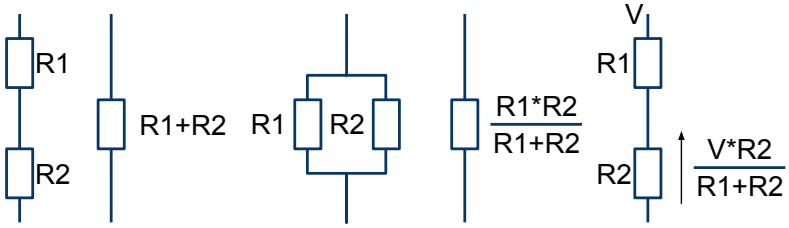
Résistance et capacité

Résistance

loi d'Ohm

U = R I



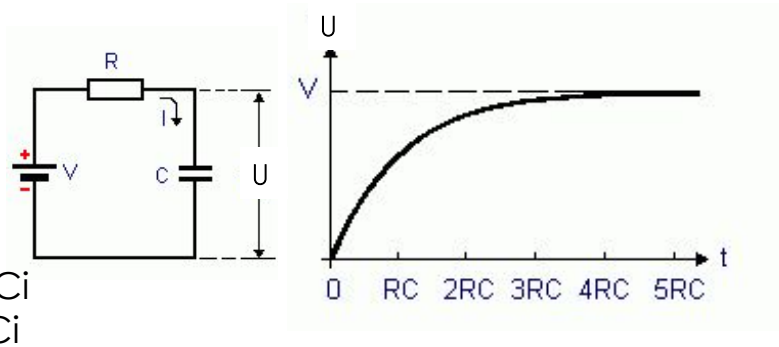


Capacité

Q = CU

I = C dU/dt

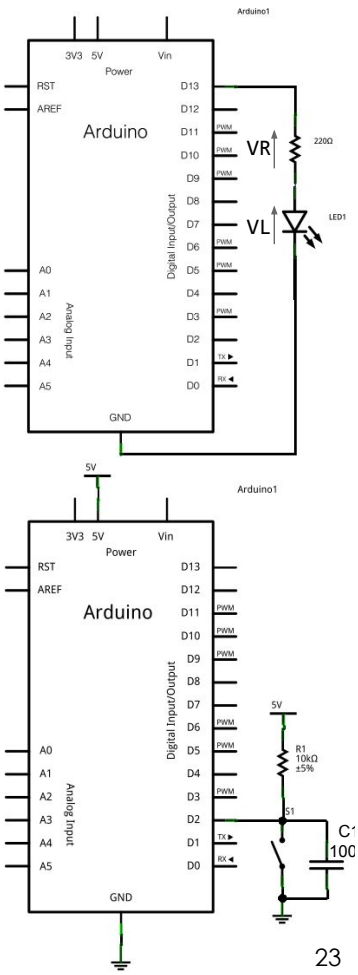
charge d'une capacité



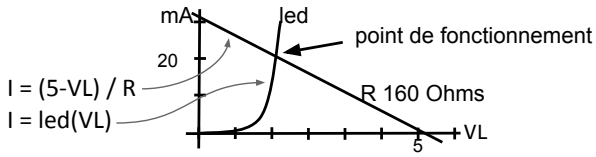
composition

- parallèle C = Σ Ci
- série C = Σ 1 / Ci

Entrées-Sorties basiques

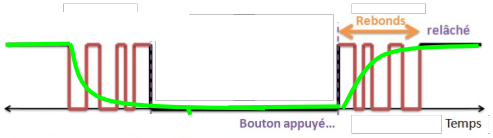


- Allumer une LED
  - Il faut que D13 soit programmé en sortie (5V)
  - loi d'Ohm : U = RI (ou R = U/I)
  - La tension aux bornes d'une LED allumée est ~ 1.8V




```
si VL = 1.8V ⇒ I = 20mA
⇒ VR = 5-1.8 = 3.2V
⇒ R = 3.2/20 = 160
```


- lire un bouton poussoir
  - Lorsque l'interrupteur est ouvert D2 est à 5V
  - Lorsqu'il se ferme la capacité se décharge à 66% en RC secondes (résitance*capacité=temps)
  - en R1*C1 = $10^4 * 10^{-7} = 10^{-3}$ = 1 ms (66%)



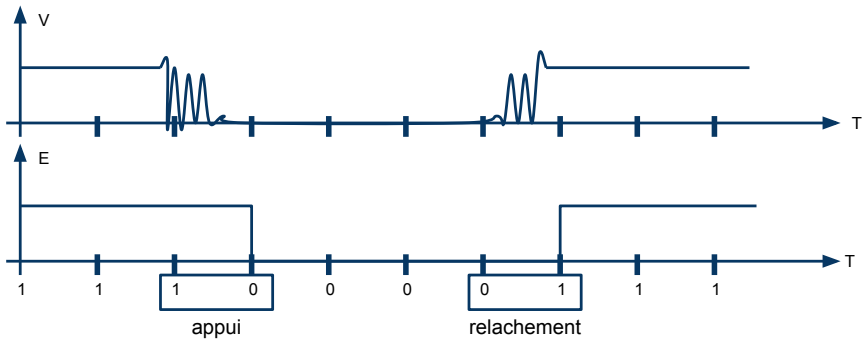
http://eskimon.fr/ est une aide précieuse

Traitement des rebonds

Il suffit d'échantillonner à une période supérieure aux rebonds, mais inférieure à la durée d'appui

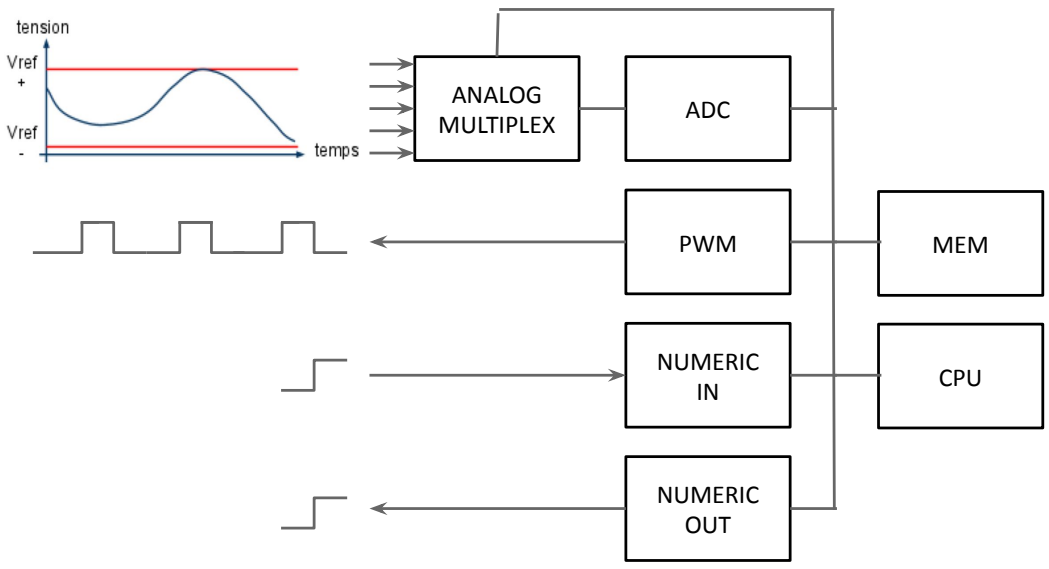
soit
- B : une variable
- E : la valeur du bouton poussoir
- appui : un drapeau à 1 si appui
- relache : un drapeau à 1 si relachement

faire périodiquement
  B = (B << 1) & E
  si (B & 3) == 2 **alors**
    appui = 1
  finsi
  si (B & 3) == 1 **alors**
    relache == 1
  finsi
finfaire



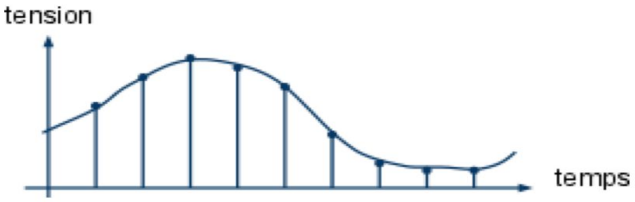
Conversion Analogique

Place de l'analogique



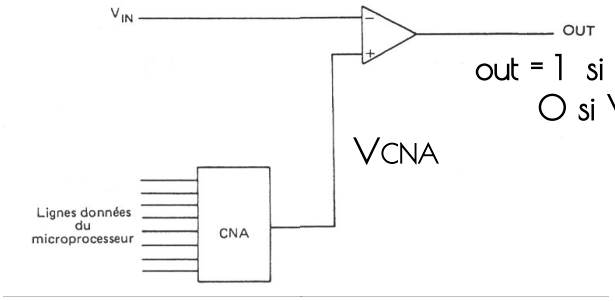
Convertisseur Analogique Numérique 1/2

Signal analogique 1 fil → CAN → Signal numérique N bits

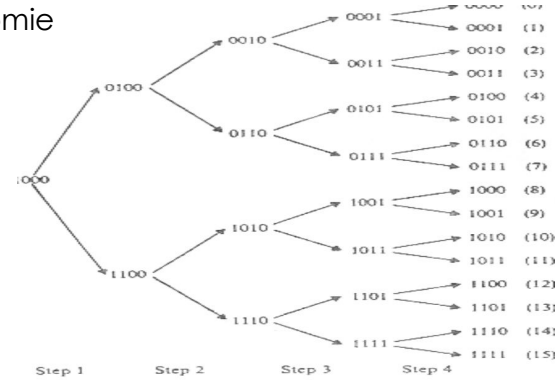


La conversion impose une quantification qui va dépendre du nombre de bits du signal numérique 8, 10, ou 12 bits

Recherche par dichotomie



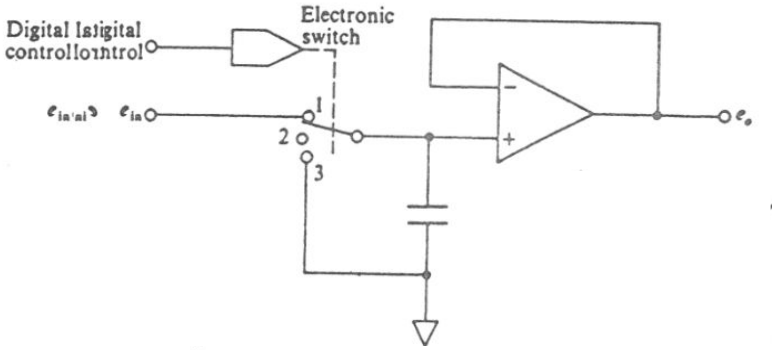
out = 1 si Vin > Vr
0 si Vin <= Vr

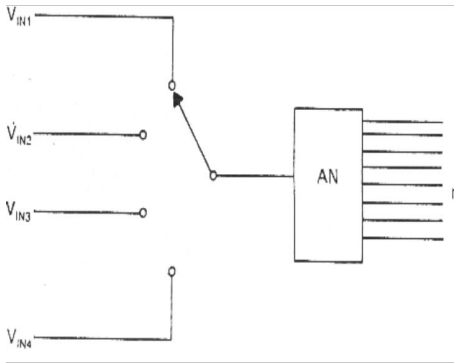


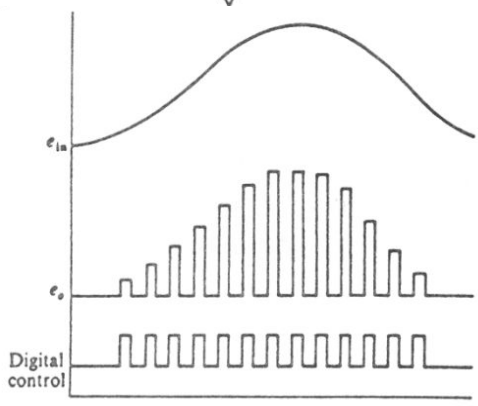
Convertisseur Analogique Numérique 2/2

Avant le convertisseur :

- échantillonneur-bloqueur
- précédé d'un multiplexeur analogique







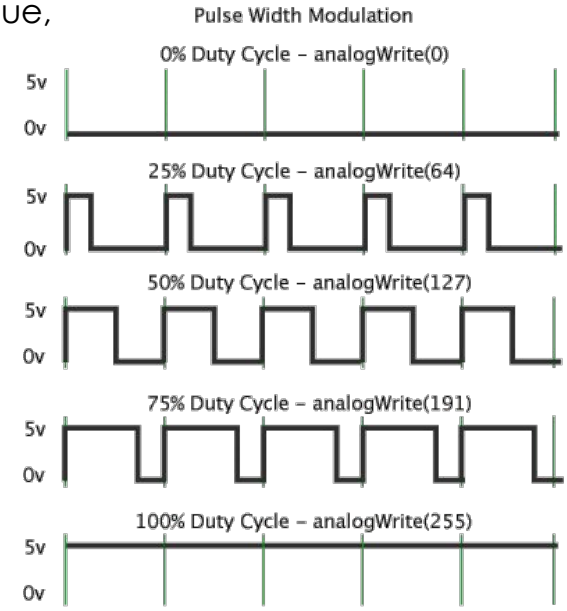
Conversion Numérique Analogique PWM

Sans Convertisseur Numérique Analogique, on peut produire un signal modulé en largeur d'impulsion : PWM

La valeur moyenne du signal est proportionnel à la largeur d'impulsion

C'est ce propose l'Arduino.
L'ESP32 propose 2 vrais Convertisseurs Numérique Analogique

http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.AnalogWrite



Conclusion

- Pendant cette UE, vous allez programmer des petits matériels, mais sans faire de montages électroniques.
  Si certains souhaitent le faire, c'est possible :-), sinon, ce qui a été vu dans ce cours sera la seule partie électronique
- Vous verrez quelques briques technologiques :
  API, composants, langages, méthodes, protocoles, etc.
  C'est vaste et on ne fera souvent qu'effleurer le sujet :-(
  libre à vous d'aller plus loin, si cela vous intéresse.
- La semaine prochaine, nous verrons comment contrôler les ports d'entrée-sortie d'une raspberry-pi sans driver.