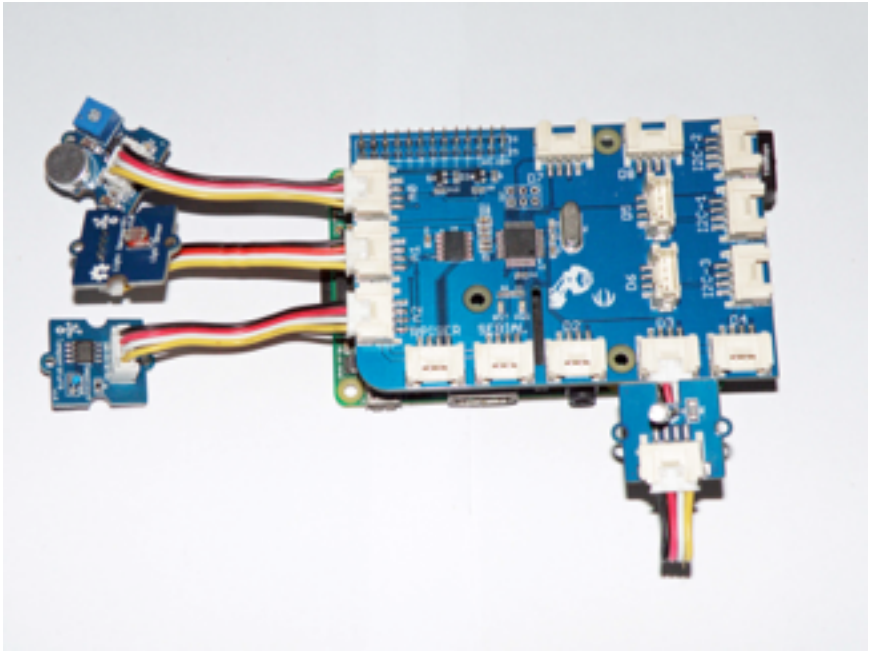
Raspberry + GrovePi et les Entrées/Sorties (E/S)

Module F.A.S. Polytech Montpellier - IG3

Composants & Logiciels nécessaires

- La pile matérielle :

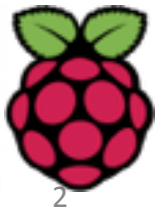


- LED (DEL) / Bouton / Capteurs

capteurs Grove

GrovePi+ Bridge

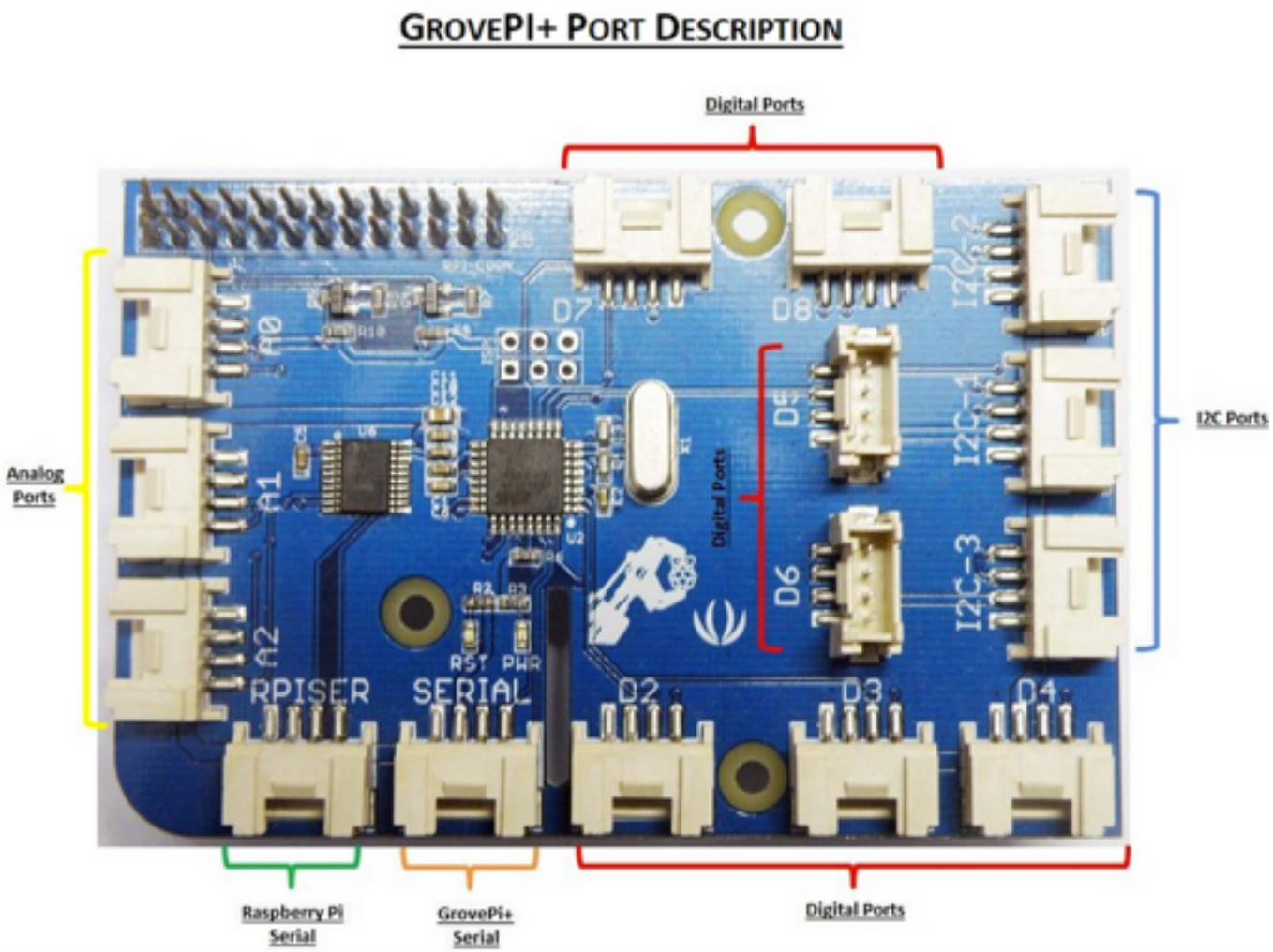
ssh, scp



Périphériques de la carte (E/S)

- Les E/S avec des capteurs sur une Raspberry de base se font essentiellement par l'intermédiaires de broches (*pin*) regroupées dans un bornier.
- Mais nous utiliserons une carte (*bridge GrovePi*) en intermédiaire ayant des connecteurs plus simples (de type *Grove*)
- La carte dispose d'un **convertisseur Analog to Digital** (A/D ou *ADC*) disposant de 6 canaux.
- Elle a aussi un **port Série** (UART)

E/S sur le Bridge GrovePi+



Rappels

Ampères Volts Ohms

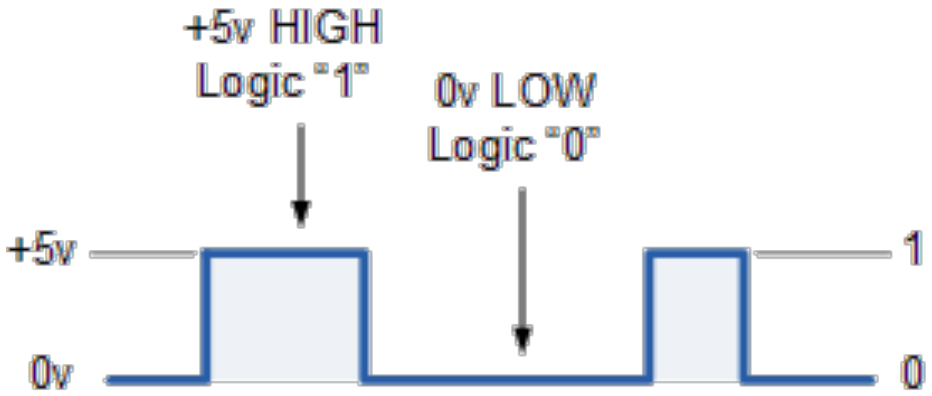
- Notions d'intensité, de tension et de résistance

Illustration de la tension et de l'intensité

Illustration de la résistance

E/S numériques

Niveaux logiques
d'une E/S numérique



- Valeur logique HIGH = vrai = ON = +5V *(3.3V pour le raspberry sans le bridge GrovePi)*
- Valeur logique LOW = faux = OFF = 0V

E/S numériques

- les connecteurs D2 à D8 sont numériques (‘***digital***’) et accepte des E/S de valeurs 0 ou 1 (sur un 1 bit donc)
- chaque connecteur peut être utilisée en entrée ou en sortie (on peut alterner sur la même) :

```
pinmode(2,OUTPUT)
pinmode(3,INPUT)
```

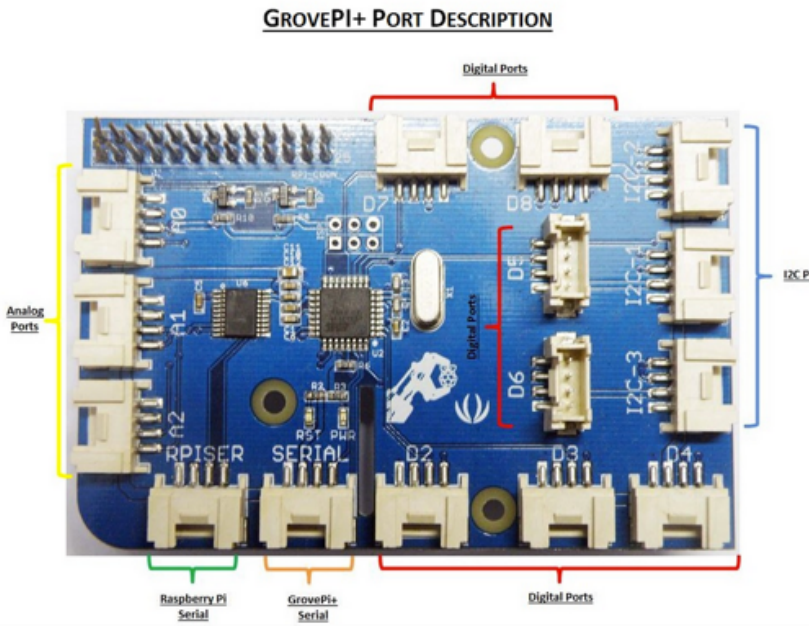
- Effectuer une sortie :

```
digitalWrite(2,HIGH)
```

- Effectuer une lecture :

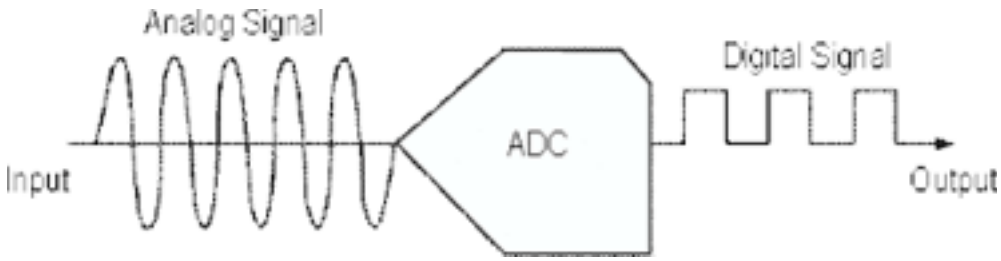
```
int val = digitalRead(8)
```

Note : en lecture, attention à la tension résiduelle (0-1V = LOW ; 3,5-5V = HIGH ; entre les deux le GrovePi peut renvoyer n’importe quoi)



Entrées Analogiques

https://www.dexterindustries.com/GrovePi/engineering/port-description/



- Connecteurs A0,A1,A2
- Ils utilisent le convertisseur A/D values qui renvoie des valeurs de 0 à 1023 (codées sur 10 bits donc)
- On ne peut utiliser analogRead() qu'avec A0,A1,A2 (aussi connus comme les connecteurs 14,15 et 16).
- Effectuer une lecture :

```
int v = analogRead(A0)
```

- Tension effectivement lue : `U = v*5.0/1024`

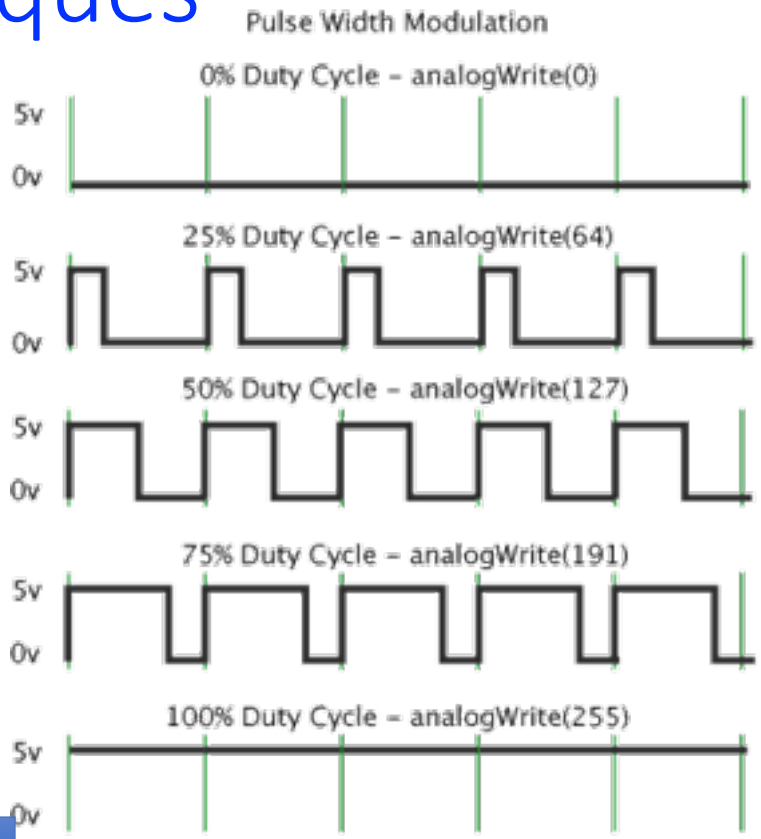
Sorties Analogiques

Données numériques sur Raspberry

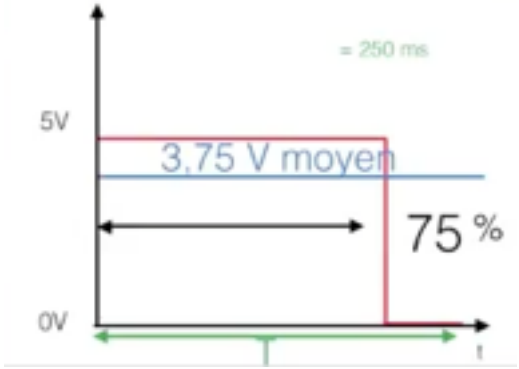
-> Pour sortir un signal analogique, on utilise la technique PWM (Pulse With Modulation) :

- signal digital (0 ou 5V)
- alternance d'états
- tension moyenne contrôlée par le temps cumulé où la tension est à 5V

- « Rapport cyclique » (*duty cycle*) = % du temps passé en haut sur une période



Exemple :



Valeur moyenne

5V * 0,75 = 3,75V

source : http://www.plaisirarduino.fr

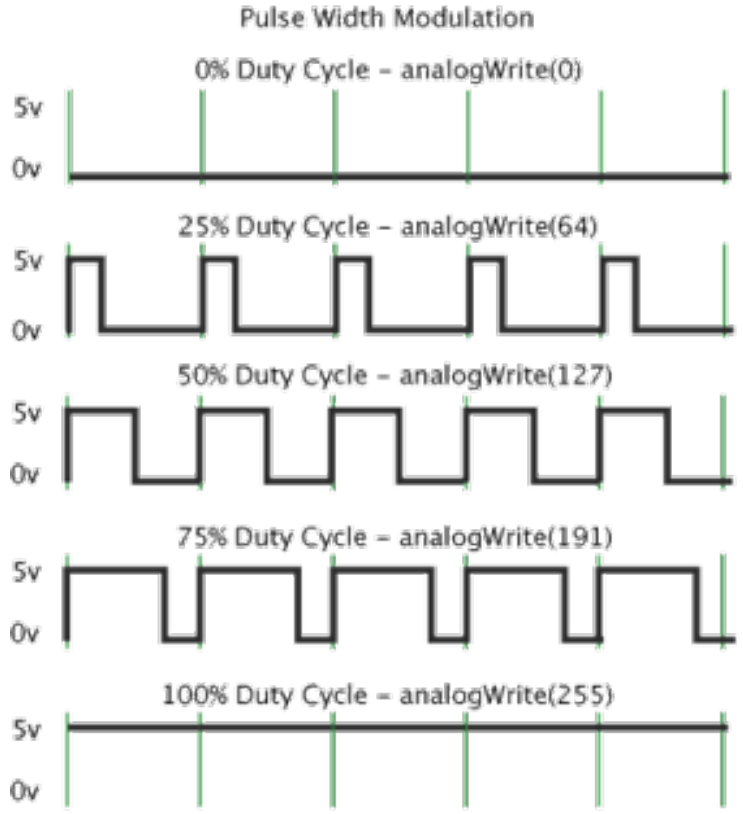
Sorties Analogiques

- Envoyer une tension : `analogWrite(num,val)`
- où val est codé sur 8 bits : valeurs entre 0 (=0V) et 255 (=5V)
- Etrangement (voir doc) il faut utiliser les connecteurs D3,D5,D6

Sorties Analogiques

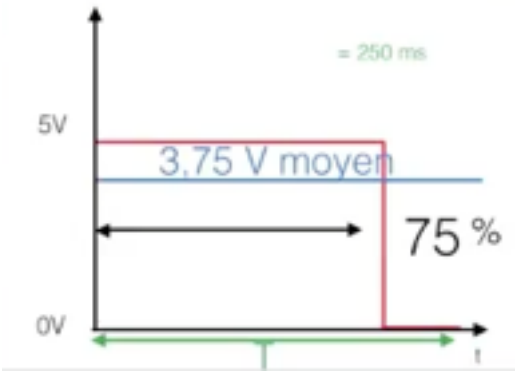
Fonctionnement de PWM (suite) :

- sur Arduino la période est de 2ms ou 1ms suivant la broche
- Le rapport cyclique est encodé sur 8 bits (valeur de 0 à 255)
- Valeur = rapport cyclique * 255



Exemple :

75% * 255 = 191



pour appliquer sur A0 une tension de 3,75V en sortie on utilise : analogwrite(D3,191)

Valeur moyenne

5V * 0,75 = 3,75V

source : http://www.plaisirarduino.fr

Côté programmation

Programmable en C, C++, **Python**, Java, Swift, …

The python library (Available at github) for grove pi has three types of functions:

1. **Basic Arduino Functions**
   - digitalRead
   - digitalWrite
   - pinMode
   - analogRead
   - analogWrite
2. **Grove Specific Functions**
   - temp
   - ultrasonicRead
   - acc_xyz
   - rtc_getTime
   - dht
3. **Private Functions for Communication**
   - write_i2c_block
   - read_i2c_block
   - read_i2c_byte

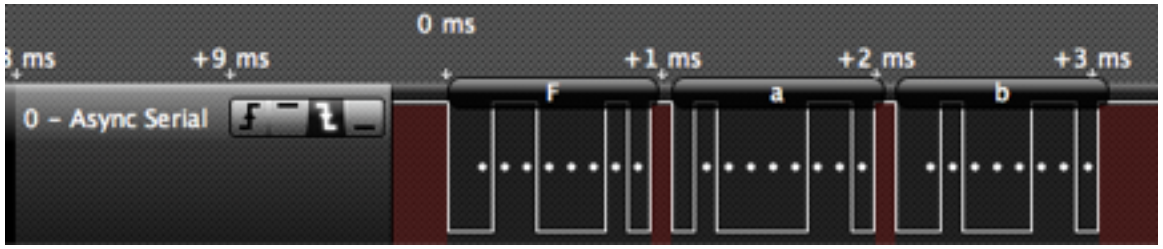
Utiliser la bibliothèque grovepi.py (voir TP)

Annexe

Compléments

E/S Séries

Par exemple à 9600 bits/sec (1 bit toutes les 0.1 ms) :



- Au repos le signal est à 1
- Pour chaque octet envoyé on envoie un bit 0 pour dire start et un bit 0 en fin pour dire stop
- Exemple : `Serial.print("Fab");` la lettre « F » soit 0**x**46 donc 0**b**01000110 :

A : pas de signal
B : start bit
C : début de l'octet envoyé
(bits les moins significatifs d'abord)

D : bit stop
E : bit start pour le caractère suivant (le « a » de « Fab »)

source https://arduino.stackexchange.com/questions/1569/what-is-serial-begin9600

E/S Séries

Lecture sur le port série :

```
while (Serial.available()>0) { # il y a des octets à lire
    char c = Serial.read()
}
```

Dernières remarques :

- comme chaque communiquant sur le port série calcule le temps par division depuis l'horloge de son processeur, il peut y avoir de petites erreurs d'arrondis, donc des valeurs incorrectes par moment
- Si on commence à lire de la donnée au milieu d'une transmission, on peut interpréter un 0 comme un bit start et se tromper sur toutes les valeurs suivantes :(
- Il est donc de bon ton de laisser un `delay(ms)` dans le programme entre deux transmissions pour que cette situation ne se produise pas (on laisse le temps de passer au moins 10 bits voir plus pour être sûr).

NB : Lire plusieurs caractères peut être géré par une boucle (voir par ex https://forum.arduino.cc/index.php?topic=288234.0)