Chapitre 2 (suite)

Arduino

Plan du chapitre 2

- Les fiches
- Les fonctions principales
- Le moniteur série
- Ecriture et lecture analogique
- PWM : Pulse Width Modulation

Liaison fiches microcontrôleur

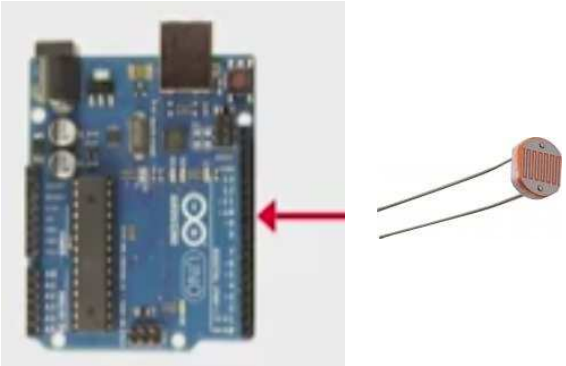
- Les fiches sont reliées au microcontrôleur
- qui va leur envoyer des potentiels électriques
- en suivant le croquis qu'on lui a écrit
- => fiches = interface du microcontrôleur

Fiches d'entrée et de sortie

⌘Pins d'entrée :

⌘Le microcontrôleur (le croquis) envoie des potentiels électriques

⌘Pin d'entrée :



⌘Le microcontrôleur (le croquis) reçoit des potentiels électriques

Entrées sorties d'une fiche

- Si on veut utiliser une pin en entrée ou sortie, il faut l'indiquer
- Souvent (mais pas obligatoirement) dans la fonction `setup()`
- La fonction pour cela est `void pinMode(pin, mode)` avec :
  - `pin` : le numéro de la fiche
  - `mode` : de valeur `INPUT`, `OUTPUT`, or `INPUT_PULLUP` (avec polarité inversée)

  configure la fiche spécifiée pour fonctionner comme une entrée ou une sortie
- biblio : `https://www.arduino.cc/en/Reference/PinMode`
- `pin` peut valoir :
  - 0 à 13 pour les pins digitales
  - A0 à A5 (de type `int`) pour les pins analogiques (en entrée seulement)
- Remarque : on écrit directement `A0` comment argument (`#define` …)

Lecture et écriture digitale

- La fonction `int digitalRead(pin)` avec :
  - `pin` : le numéro de la fiche

  retourne l'état d'un fiche. Ce peut être `LOW` (0 volt) ou `HIGH` (5 volts sur une Arduino UNO)
- biblio : `https://www.arduino.cc/en/Reference/DigitalRead`
- La fonction `void digitalWrite(pin, value)` avec :
  - `pin` : le numéro de la fiche
  - `value` : de valeur `LOW` ou `HIGH`

  affectue la valeur `LOW` (0 volt) ou `HIGH` à la fiche `pin`
- Remarque : les pins analogiques peuvent servir de pins de sorties ... digitales c'est à dire on peut écrire `digitalWrite(A0, HIGH);`
- biblio : `https://www.arduino.cc/en/Reference/DigitalWrite`

Lecture d'une fiche analogique

- La fonction `int analogRead(pin)` avec :
  - `pin` : le numéro de la fiche. Celle-ci doit être une pin analogue (`A0` à `A5`)

  retourne l'état d'un fiche analogique. C'est un entier de valeur `0` (pour `0` volt) à `1023` (pour `5` volts)
- Par exemple :

```
int pinVal;
pinVal = analogRead(A3);
```

- biblio : `https://www.arduino.cc/en/Reference/AnalogRead`

La fonction de pause

- La fonction `void delay(msec)` avec :
  - `msec` : nombre de millisecondes

  fait une pause à l'exécution du croquis de `msec` millisecondes
- Par exemple, le code :

```
digitalWrite(3, HIGH);
delay(1000);
digitalWrite(3, LOW);
```

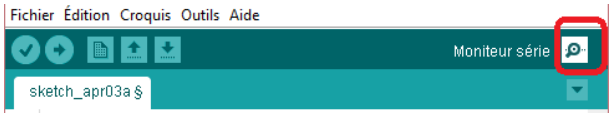
  permet d'envoyer 5 volts sur la pin 13 pendant 1 seconde
- biblio : `https://www.arduino.cc/en/Reference/Delay`

Arduino : pour commencer

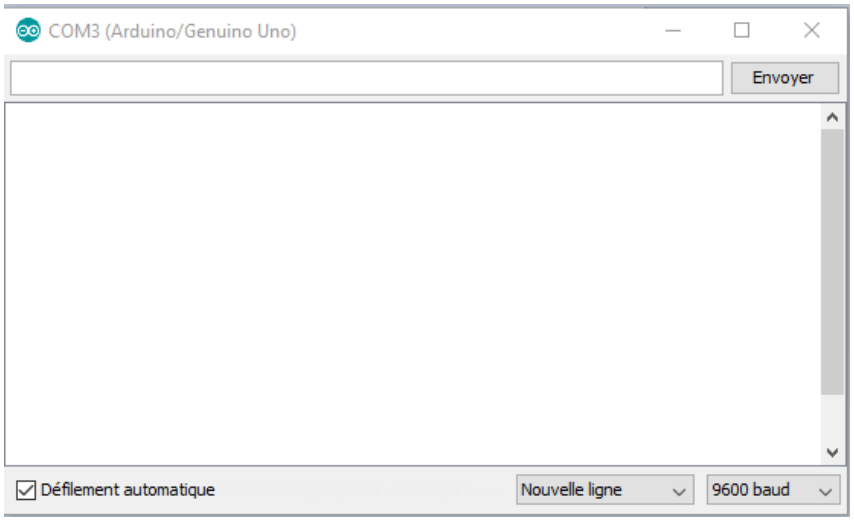
- La doc de référence d'Arduino : https://www.arduino.cc/en/Reference/HomePage
- https://www.arduino.cc/en/Hacking/BuildProcess
- https://www.arduino.cc/en/Reference/Setup
- https://www.arduino.cc/en/Reference/Loop
- https://www.arduino.cc/en/Reference/PinMode
- https://www.arduino.cc/en/Reference/DigitalWrite
- https://www.arduino.cc/en/Reference/DigitalRead
- https://www.arduino.cc/en/Reference/AnalogRead

Le "moniteur série"

- Lorsque la carte Arduino est connectée au PC, ils peuvent communiquer entre eux (par le protocole série UART)
- Dans l'IDE cliquer sur le bouton Moniteur série

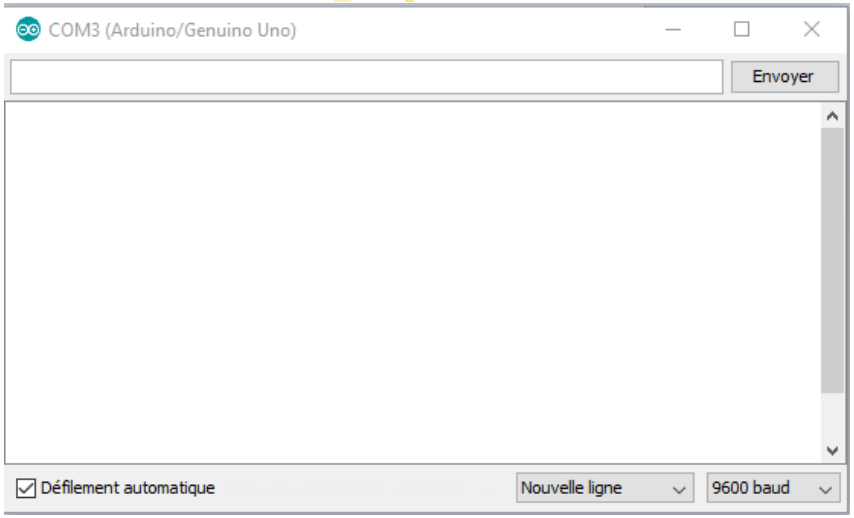


- Une fenêtre pour la communication est alors ouverte :



Communication grâce au moniteur série

- Avec la fenêtre :



- des données envoyées par la carte Arduino peuvent être affichées
- on peut envoyer des données à la carte Arduino par l'intermédiaire du clavier du PC
- Cela peut servir à déboguer

Communication carte Arduino vers PC

- L'émetteur et le récepteur doit connaître la vitesse de transfert = durée pour transmettre un bit qui restera à 0 (potentiel 0) ou 1 (potentiel n volts) pendant cette durée
- `Serial.begin(9600)` dans `setup()`
- 9600 baud => 104 microsecondes pour transmettre un bit
- `Serial.print(texte)` ou `Serial.println(texte)` écrit dans la console Moniteur série du texte (et revient à la ligne pour `println()`). Par exemple :

```
Serial.println("entrée du while");
```

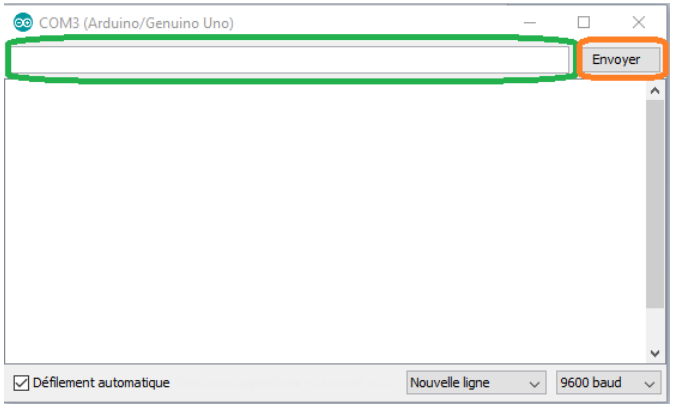
- Pour faire imprimer un `int`, utiliser `Serial.write(unEntier)`. Par exemple :

```
int n = 43;
Serial.write(n);
```

, idem pour un `float` ou un `double`

Communication PC vers carte Arduino (1/)

- Des données peuvent être envoyées du PC vers la carte à l'aide de la zone de texte en haut du moniteur série et cliquer le bouton Envoyer



- En fait les envois du PC vers la carte sont bufferisés
- La carte Arduino lit les valeurs envoyées par la fonction `int Serial.read()`
- Cette fonction retourne un octet (codé en `int`) si un octet est disponible, `-1` sinon
- On a donc :

```
int unOctetLu = Serial.read();
```

Communication PC vers carte Arduino (2/)

- On plus lire plusieurs octets en les déposant dans un tableau de caractères par :

```
char buffer[16];
Serial.readBytes(buffer, 16);
```

- Dans le croquis, avant de faire des lectures, il bon d'écrire `Serial.available()` pour savoir si le buffer a des données

Fonctions statiques de classes

- En fait, les appels `Serial.XXX(…)` sont des appels de fonctions statiques de la classe `Serial`
- Les notions de classes, fonctions statiques (= de classes) sont des notions du langage C++
- Elles sont utilisées de la même manière qu'une fonction quelconque
- Mais appartiennent à la classe (= au module) `Serial`

Exercice

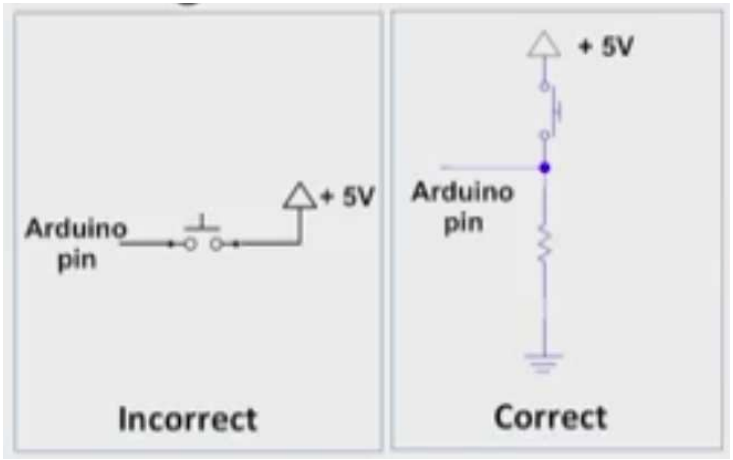
- Dialogue entre le PC et la carte Arduino grâce à la console série.
  Programmer un interrupteur numérique !

Obtenir des infos par la carte

- Les microcontrôleurs sont sensibles aux tensions électriques
- Les capteurs convertissent leurs informations (chaleur, flexion, humidité, luminosité, etc.) en potentiels électriques (parfois indirectement cf. potentiomètre)
- Ces tensions sont amenées dans les fiches par des câbles
- Lire une tension de la fiche digitale `pin` est obtenu par `int digitalRead(pin)`
- La valeur retournée est `HIGH` ou `LOW`
- Lire une tension de la fiche analogique `pin` est obtenu par `int analogRead(pin)`
- La valeur retournée est un int entre `0` et `1023`

Encore de l'électricité

- Bien utiliser les résistances
- Comparaison de 2 circuits : Lire un pushbutton



- Il faut bien relier la fiche à la terre pour qu'elle se décharge et ait un potentiel à `0` : lorsque le bouton est ouvert, la fiche est reliée à la terre de potentiel `0`
- Dans le circuit à gauche elle peut rester à un potentiel non nul (inertie électrique, électricité statique ?)

Capteur (sensor)

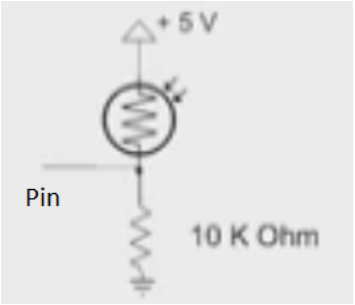
- = détecteur
- pour les entrées
- Ce sont souvent des capteurs résistifs : ils changent leur résistance suivant certaines informations : ~ potentiomètre

- Photorésistance, thermomètre, flex resistor ~ potentiomètre

La photorésistance

- Quand la luminosité augmente, la résistance diminue
- Une photorésistance :

- Si on fait le circuit :



- Quand la luminosité augmente la fiche Pin aura un potentiel plus grand, n'est ce pas ?
  (U = RI dans chaque portion du circuit et U + U' = constant = 5 volts)

Pulse Width Modulation (PWM)

- Les potentiels qui sortent d'un microcontrôleur sont soit hauts (5 ou 3,5 volts) soit 0
- Comment faire des valeurs continues sur un seul signal ?
- La solution : faire varier le temps pendant lequel le signal est haut
- Définition : le duty cycle est le pourcentage du temps pendant lequel le signal est haut sur une période
- Accroitre le duty cycle augmente la tension finale
-

- Cette technique est la Pulse Width Modulation (PWM)

Fonction d'écriture analogique

- `void analogWrite(pin, value)` génère un signal PWM
- Peut être utilisé pour faire varier la luminosité d'une LED, la vitesse d'un moteur, etc.
- La fiche `pin` va générer un signal rectangulaire avec un duty cycle adapté à `value` (jusqu'au prochain appel à `analogWrite()`, `digitalRead()` ou `digitalWrite()` sur la même fiche)
- `value` doit avoir une valeur entre `0` et `255` (`0` pour `0%` de duty cycle, `255` = `100%` de duty cycle)
- Cette fonction fonctionne sur les fiches 3, 5, 6, 9 et 11 de la carte Arduino. Voir le symbole ~
- On n'a pas besoin d'appeler `pinMode()` sur une fiche en sortie qui utilise `analogWrite()`
- La fonction `analogWrite()` n'a rien a voir avec les fiches analogiques ou avec la fonction `analogRead()`
- biblio : `https://www.arduino.cc/en/Reference/AnalogWrite`

Fonction d'écriture analogique : un exemple

```
int brillance = 0;
int increment = 5;

void loop(){
     analogWrite(led, brillance);
     brillance += increment;
     if (brillance == 0 || brillance == 255) {
          increment = - increment;
     }
     delay(30);
}
```

- Que fait cet exemple ?
- Boucle sur allume et éteint progressivement une led

Exercice

- Un détecteur de lumière gère une diode qui s'allume en fonction de la luminosité

Fin