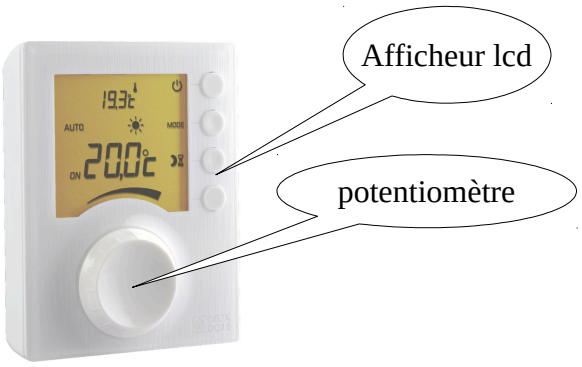
Capteurs, Arduino et domotique

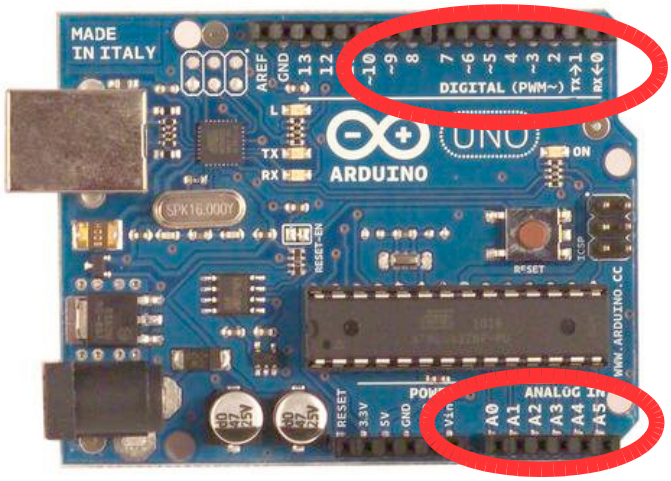
Contextualisation

On souhaite commander un store de pergola à lame inclinable en fonction de la température ou de la luminosité à l'aide d'une console de contrôle,
Pour cela, on va utiliser des capteurs , une carte arduino et des servomoteurs (pour modéliser des moteurs),

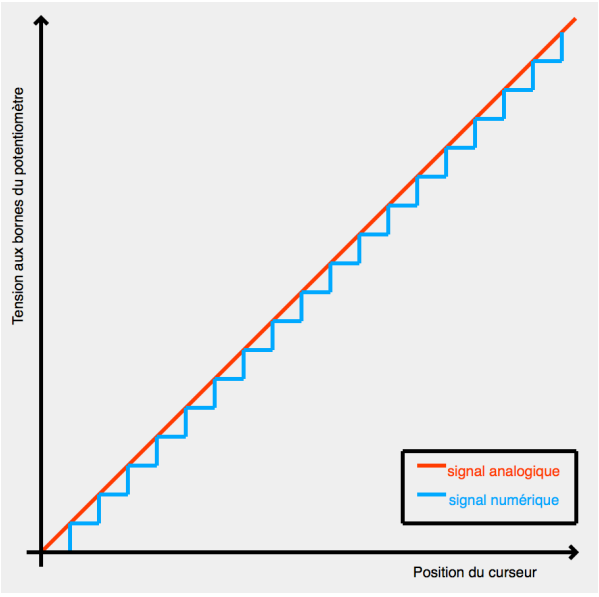


Qu'est ce qu'Arduino ?

Arduino est une carte électronique programmable en C permettant d'y brancher plusieurs types de capteurs :
- soit analogiques (valeurs continues)
- soit numériques (valeurs discrètes)
- soit logiques (valeur 1 ou 0)



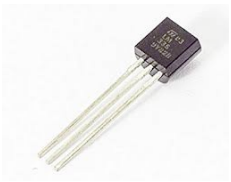
Les capteurs analogiques transforment une grandeur physique en tension comprise entre 0 et 5V.
Arduino possède donc un convertisseur analogique-numérique permettant de transformer la tension de sortie du capteur (ex potentiomètre, capteur de température, capteur de lumière) en un message binaire.
Les capteurs numériques prennent un nombre de valeurs définies. (ex humidité température, GPS...)



Capteur humidité température

Capteur GPS

Capteur sonore



Capteur de température

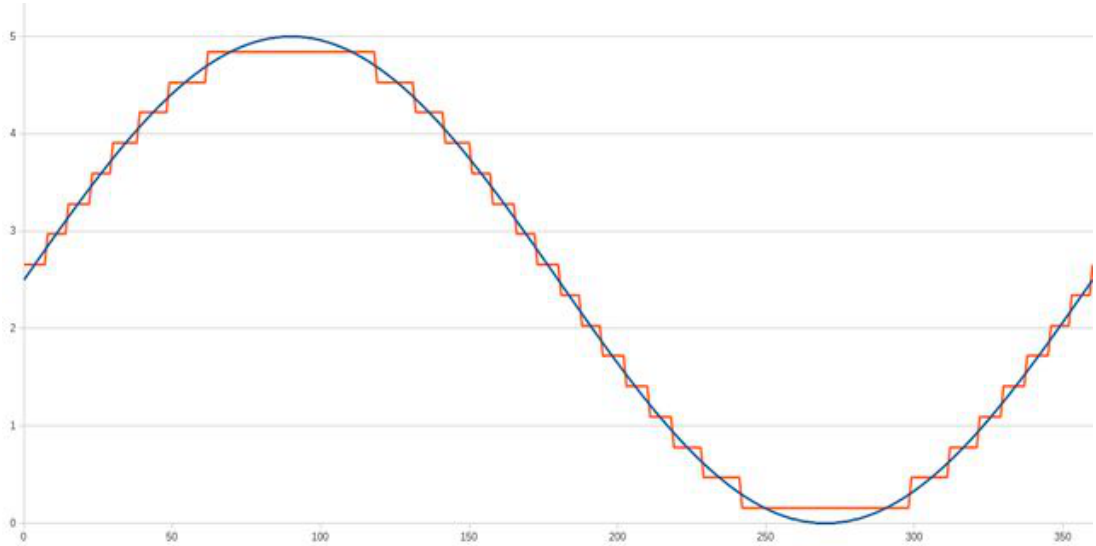
Capteur lumière

Grâce à ses entrées analogiques, la carte Arduino est capable de mesurer une tension comprise entre 0 et 5V. Par l'utilisation de la fonction AnalogRead(), il est dès lors possible d'obtenir une valeur de la tension numérisée sur 10bits comprise par conséquent entre 0 et 1023. La valeur 0 correspond donc à une tension de 0V et la valeur 1023 à 5V.

En électronique numérique, on travaille avec des bits et des octets. En analogique, on travaille avec des grandeurs physiques : tension, courant, résistance, fréquence, etc.

Pour pouvoir exploiter des mesures analogiques avec un microcontrôleur, il faut convertir la mesure analogique en une grandeur numérique. C'est justement le but des convertisseurs analogique / numérique.

Sans entrer dans les détails techniques, un convertisseur analogique / numérique permet de mesurer une

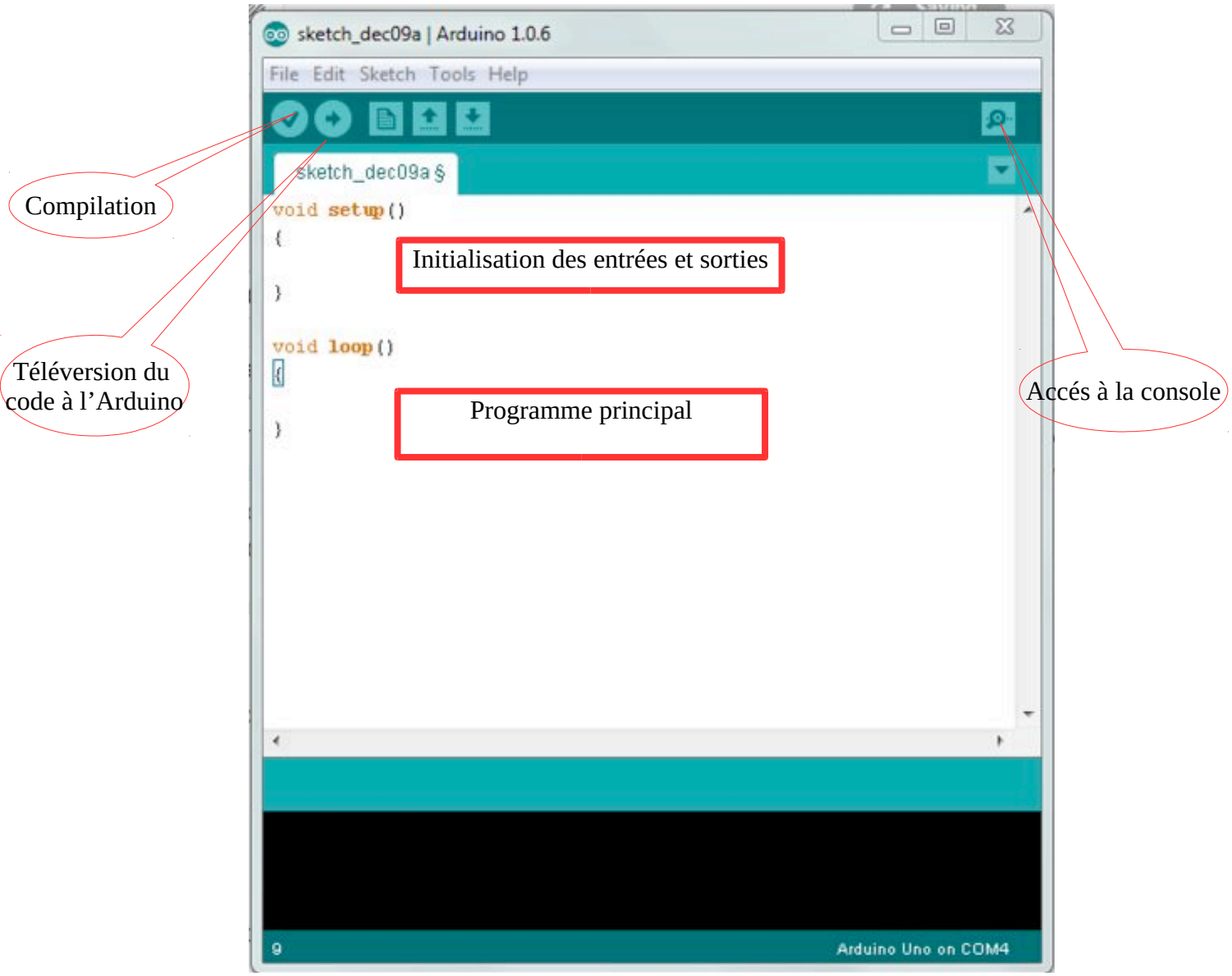


Signal analogique versus signal numérique

L'idée est simple : associer une valeur numérique (un nombre entier) pour chaque valeur analogique d'une plage de tension bien précise.

La fréquence d 'échantillonnage est de 9600 échantillons/s et le nombre de bits de la valeur numérisée est 10

Fiche technique d'utilisation de l'interface de programmation d'Arduino



Activités et exercices d'application

1. Programme de capture de l'intensité lumineuse avec une photo résistance

Le montage et les composants

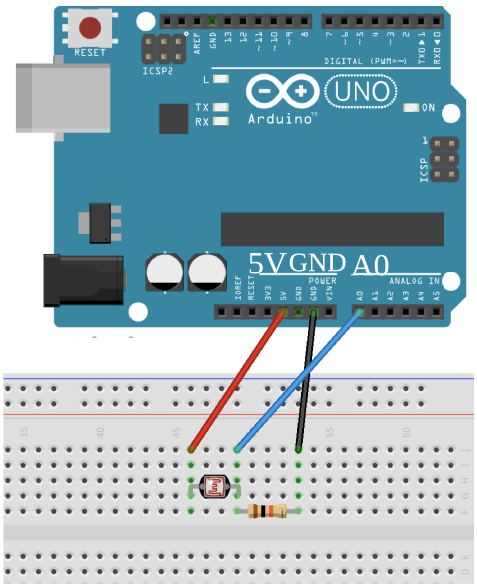
Matériel :

- Une photo résistance de 1M ohms (de diamètre 3mm ou 5mm, cela importe peu),

- Une résistance de 10K ohms (marron / noir / orange),

Une photo résistance est un composant dont la résistivité dépend de la luminosité ambiante. Pour faire simple, c'est une résistance dont la valeur change en fonction de la lumière qu'elle reçoit.

Il existe différents types de photo résistances, chacune ayant des valeurs de résistance différentes en fonction de la luminosité ambiante. Le type le plus classique de photo résistances est de 1M ohms (obscurité) / 12K ohms (pleine lumière).



La programmation d'Arduino

Le but de notre code va être de :

1. Lire la tension sur la broche `A0`,
2. Envoyer la valeur au PC (pour l'affichage),
3. Recommencer au point 1.

Pour réaliser ce morceau code, nous allons utiliser la fonction analogRead() .

```
int valeurlum=A0 ;
// Fonction setup(),
appelée au démarrage
de la carte
void setup() {
  Serial.begin(9600);
  pinMode(A0,INPUT) ;
}
```

Nous allons commencer notre programme de démonstration avec la fonction `setup()` qui va simplement initialiser la communication avec le PC.

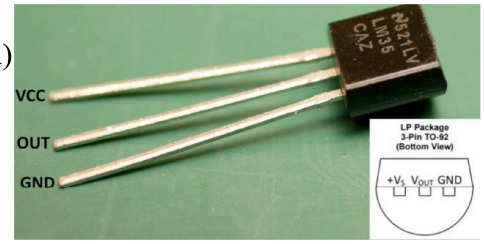
```
// Fonction loop(), appelée en
boucle tant que la carte est
alimentée
void loop() {
//Mesure la tension sur la
broche A0
  int valeurlum =
analogRead(A0);
//envoie la mesure au PC pour
affichage
  Serial.println(valeurlum);
  delay(250);
}
```

Mettre en œuvre le montage, écrire le programme et téléverser le code
Observer les valeurs sur la console en jouant sur la luminosité.

2. Programme de capture de la température

Le montage et les composants

Matériel :
- condensateur de 100 nF (optionnel)
- capteur LM35DZ

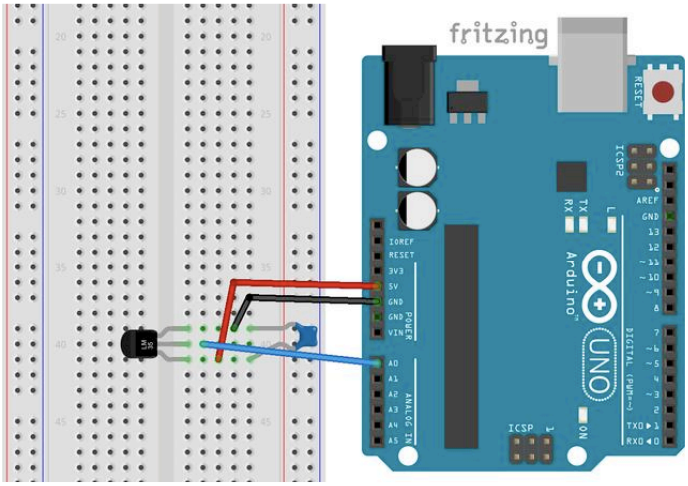


La programmation d'Arduino

Le but de notre code va être de :

1. Lire la tension sur la broche `A3`
2. Convertir la valeur mesurée en une température (pour l'affichage)
3. Envoyer la valeur au PC (pour l'affichage)

Le capteur de température_LM35DZ, capable de mesurer des températures de 0 à 100°C avec une précision de 1.5°C aux extrêmes.
La sortie analogique du capteur est proportionnelle à la température. Il suffit de mesurer la tension en sortie du capteur pour en déduire la température. Chaque degré Celsius correspond à une tension de +10mV.



4. Recommencer au point 1.

Pour réaliser ce morceau code, nous allons encore utiliser la fonction analogRead()

```
int valeur_brute=A0 ;
float  temperature_celcius=valeur_brute *
(5.0 / 1023.0 * 100.0);
void setup() {
  Serial.begin(9600);
  pinMode(A0,INPUT) ;
}
```

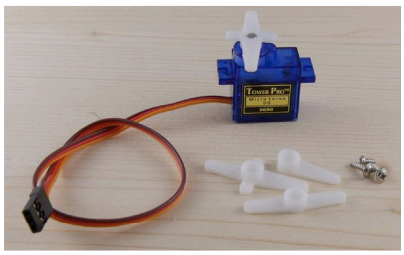
Nous allons commencer notre programme de démonstration avec la fonction `setup()`

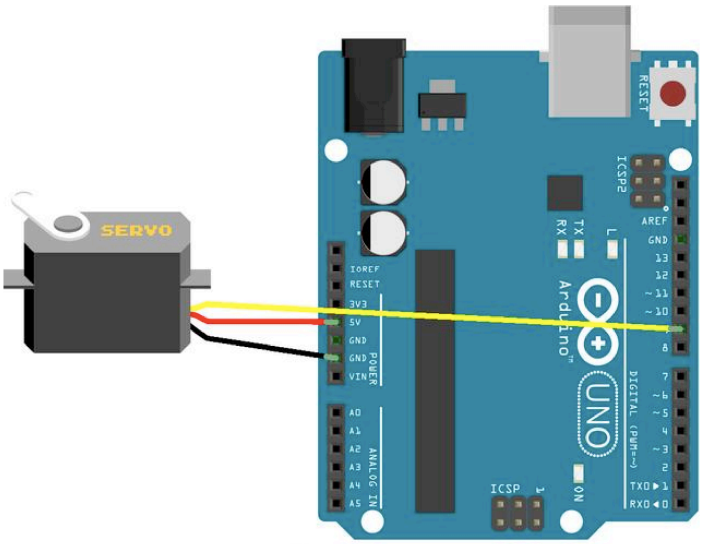
```
void loop() {
  int valeur_brute = analogRead(A0);
  Serial.println(temperature_celcius);
  delay(250);
}
```

1. Réaliser le montage.
2. Ecrire le programme et le commenter. Téléverser-le.
3. Sachant que 5V = 5000mV = 1023 en sortie de analogRead(), 10mV = 1°C, expliquer la conversion qui est faite pour obtenir la température en celcius. Indiquer le code à indiquer si on veut la température en degré fahrenheit sachant que fahrenheit = celcius * 1.8 + 32.

3. Programme pour faire fonctionner un servo moteur

Le montage et les composants	La programmation d'Arduino

 Le fonctionnement interne d'un servomoteur est simple. Un petit circuit électronique permet de contrôler un moteur à courant continu en fonction de la position d'un potentiomètre intégré au servomoteur. La sortie du moteur à courant continu est reliée mécaniquement à une série d'engrenages qui augmente la force (le couple) du servomoteur en réduisant la vitesse de rotation de celui-ci. Quand le moteur tourne, les engrenages s'animent, le bras bouge et entraine avec lui le potentiomètre. Le circuit électronique ajuste continuellement la vitesse du moteur pour que le potentiomètre (et par extension le bras) reste toujours au même endroit. Il suffit de donner une consigne au servomoteur ("reste à 45°" par exemple) et le servomoteur fera son maximum pour rester au plus près de cette consigne.	Le but de notre code va être de : Faire tourner votre moteur de 90° puis de le faire revenir à sa position (0°) initiale.
	
	Pour faire bouger notre servomoteur, nous allons devoir utiliser une bibliothèque de code, nommée "Servo". Celle-ci est fournie de base avec l'environnement de développement Arduino. Aller dans Croquis, Ajouter les bibliothèque et choisir « Servo ».
	
	`#include <Servo.h>`
	`//La bibliothèque Servo est orientée object, une variable est assignée à chaque servomoteur`
	`Servo monservomoteur ;`
	`void setup() {`
	`  monServomoteur.attach(9);`
	`// le servomoteur est sur la broche 9`
	`}`



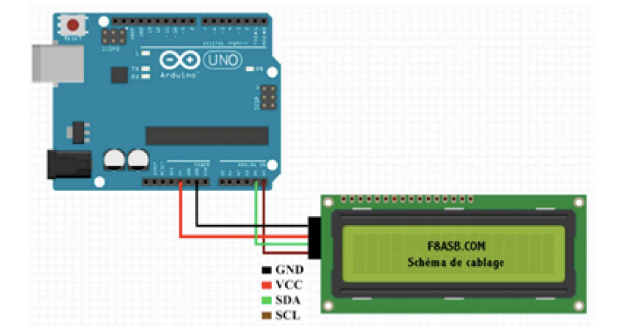
Passons à la boucle :

```
void loop() {
 monservomoteur.Write(0) ;

delay(500) ;
monservomoteur.Write(90) ;
delay(500) ;

}
```

Mettre en œuvre le montage, écrire le programme et téléverser le code.

4. Programme pour faire fonctionner un afficheur LCD

Le montage et les composants	La programmation d'Arduino


Montage à réaliser
Affichage de 2 lignes et 16 caractères.
 	
la ligne SDA (ligne de données) est sur la broche analogique 4
 	
la ligne SCL (ligne d'horloge) est sur la broche analogique 5
 	
La bibliothèque Wire,h permet de communiquer avec les composants utilisant le protocole I2C / TWI (communication série sur 2 fils).
	Inclure la bibliothèque LiquidCrystal_I2C.h (à chercher sur internet et à placer dans la library d'Arduino)
	
	`#include <LiquidCrystal_I2C.h> // télécharger sur internet la bibliothèque et l'ajouter par croquis importer ajouter`
	`#include <Wire.h> //-----Adressage matériel-----`
	
	`LiquidCrystal_I2C lcd(0x27, 16, 2); //adresse, nb caracteres, nb lignes`
	`int capteurPin = A0;`
	
	`void setup()`
	`{`
	`Serial.begin(9600);`
	`lcd.init(); // initialisation de l'afficheur`
	`}`
	
	Et la boucle principale pour l'affichage :
	
	`void loop()`
	`{`
	`lcd.backlight(); // Envoi du message`
	`lcd.setCursor(0, 0);`
	`lcd.print("Bonjour");`
	`lcd.setCursor(7,1);`
	`lcd.print("NSI");`
	`}`

5. Communication avec Python

Pourquoi ?

- ✓ pas de système d'exploitation embarqué
- ✓ peu de mémoire

- ✓ Stocker et manipuler un grand nombre de données → Python
- ✓ Python communique avec la carte Arduino.
- ✓ Données stockées dans un fichier.
- ✓ Traitement avec Python.

Comment ?

- ✓ Python communique avec le microcontrôleur
- ✓ Exécution programme en mémoire flash (le dernier programme téléversé).

DESCRIPTION | FICHE TECHNIQUE | AVIS

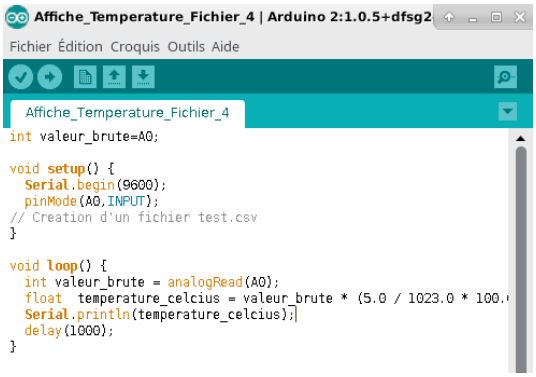
La carte Arduino Uno est basée sur un ATMega328 cadencé à 16 MHz. C'est la plus récente et la plus économique carte à microcontrôleur d'Arduino. Des connecteurs situés sur les bords extérieurs du circuit imprimé permettent d'enficher une série de modules complémentaires.

Elle peut se programmer avec le logiciel Arduino. Le contrôleur ATMega328 contient un bootloader qui permet de modifier le programme sans passer par un programmateur. Le logiciel est téléchargeable gratuitement. Cette carte est livrée sans cordon USB (voir articles conseillés).

Caractéristiques principales:

- version: Rev. 3
- alimentation:
  - via port USB ou
  - 7 à 12 V sur connecteur alim 5,5 x 2,1 mm
- microprocesseur: ATMega328
- mémoire flash: 32 kB
- mémoire SRAM: 2 kB
- mémoire EEPROM: 1 kB
- 14 broches d'E/S dont 6 PWM
- 6 entrées analogiques 10 bits
- intensité par E/S: 40 mA
- cadencement: 16 MHz
- bus série, I2C et SPI
- gestion des interruptions
- fiche USB B
- dimensions: 74 x 53 x 15 mm

Module prêt à l'emploi.
Version d'origine fabriquée en Italie.
Site officiel: www.arduino.cc



Commandes à utiliser

Récupérer la valeur du capteur

```
# Ouverture de la ligne série 9600 bauds
port = serial.Serial('/dev/ttyUSB0', 9600)

# On lit les valeurs que l'on stocke dans le tableau valeurs
temp = float(port.readline())
```

Ecrire tableau tableau_temp[…] dans un fichier temperature.csv

```
np.savetxt('temperature.csv', np.array(tableau_temp))
```

Lire le fichier temperature.csv

```
with open("temperature.csv", "r") as fichier:
    for line in fichier.readlines():
        x = round(float(line.strip()),1)
```

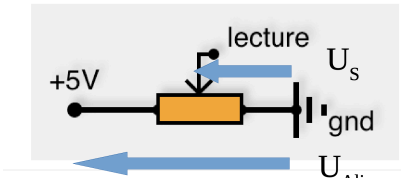
```
# Le programme permet de récupérer
# des valeurs envoyées par Serial.println() et les
# stocke finalement dans un fichier CSV
import serial
import numpy as np
import time
import matplotlib.pyplot as plt

# Ouverture de la ligne série 9600 bauds
port = serial.Serial('/dev/ttyUSB0', 9600)
tableau_temp = []
i = 0

try:
    # On lit les valeurs que l'on stocke dans le tableau valeurs
    while True:
        temp = float(port.readline())
        tableau_temp.append(temp)
        print(i,"h : ",temp)
        time.sleep(1)
        i = i+1
except KeyboardInterrupt:
    # Lorque le programme est interrompu (CTRL+C), les valeurs sont stockées
    # dans le fichier valeurs.csv
    print('Sauvegarde de temperature.csv ...')
np.savetxt('temperature.csv', np.array(tableau_temp))
```

```
Shell
========== RESTART ==========
>>> %Run Temperature_Lire.py
0 h :  25.9
1 h :  26.39
2 h :  25.9
3 h :  26.39
4 h :  26.39
5 h :  26.39
6 h :  26.39
7 h :  26.39
8 h :  26.39
9 h :  26.39
10 h :  26.39
11 h :  26.88
12 h :  26.88
13 h :  26.88
14 h :  27.37
15 h :  27.37
16 h :  26.88
17 h :  27.37
18 h :  27.37
19 h :  27.37
20 h :  27.37
```

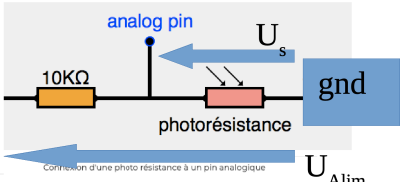
Exercices :

Premier exercice :

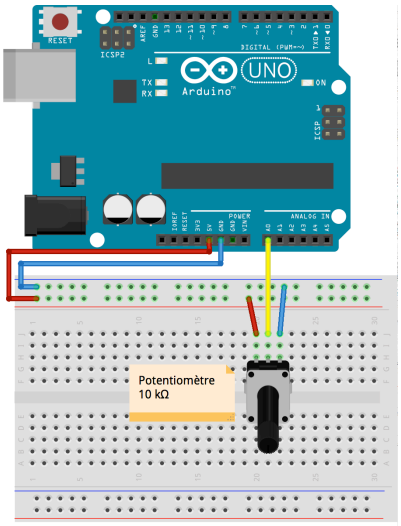
A laide de l'activité du capteur lumière, réaliser le code et le montage d'un potentiomètre (montage simple ci-contre)



Potentiomètre



Photorésistance



Deuxième exercice :

Réaliser un code afin de faire passer le servomoteur de 0 à 90 progressivement et/ou de 90 à 0.

Dans Arduino, pour créer une boucle :

```
for (initialisation; condition; incrémentation) {
//instruction(s)à exécuter;
}

Exemple :

 // boucle incrémentant la variable i de 0 à 255, de 1 en 1

   for (int i=0; i <= 255; i++){
```

Troisième exercice :

Combiner une photo résistance et un servomoteur
Cahier des charges : Ouvrir le store à l'aide du servo moteur lorsque l'intensité lumineuse est inférieure à une valeur choisie et fermer le store dans le cas contraire.

Dans Arduino, pour créer une action conditionnée :

```
if (brocheCinqEntree < 500)
{
  // action A
}
else
{
  // action B
}
```

Quatrième exercice :

Combiner le capteur température et l'afficheur afin d'afficher le temps pendant lequel la température est supérieure à une valeur que vous définirez.

```
void setup() {
    heures = 0;
    minutes = 0;
    secondes = 0;
}
void loop() {
     if(secondes == 60) // une minute est atteinte
        {
            secondes = 0; // on recompte à partir de 0
            minutes++;
        }
}

   (idem pour minutes et heures)
La fonction sprintf admet un nombre variable de paramètres. Son utilisation est
la suivante :
sprintf ( chaîne , format , param1 , param2 , ... , paramn )

sprintf(message,"duree %2d:%2d:%2d",heures,minutes,secondes);
```

Quatrième exercice :Quatrième exercice V2:

Combiner le capteur température et l'afficheur afin d'afficher le temps pendant lequel la température est supérieure à une valeur que vous définirez.

Dans Arduino, la fonction millis() permet de mesurer le temps écoulé.

```
long temps; // variable qui stocke la mesure du temps
void setup() {
temps = millis();}
void loop()
{
if ((millis()-temps>1000)}
```

Projet globalisé :

Combiner le programme température et afficheur avec le potentiomètre et le servomoteur qui permettra de fixer la température de déclenchement de la fermeture ou de l'ouverture de la pergola

Extension : Dans le cas d'ensoleillement très variable (nuage/eclaircie), proposer un cahier des charges permettant d'éviter une trop grande solicitation des servomoteur.