Introduction à la programmation Arduino

1. Introduction

a. Qu'est-ce qu'un microcontrôleur

Un microcontrôleur est un circuit intégré qui reprend les caractéristiques d'un ordinateur (mémoire + processeur) en version allégée. Souvent un micro contrôleur est programmé pour n'exécuter qu'une seule tâche. Il est moins performant mais nettement moins coûteux que les processeurs de PC

b. Présentation Arduino

i. Massimo Banzi https://www.ted.com/talks/massimo_banzi_how_arduino_is_open_sourcing_imagination

ii. Simone Giertz : Why you should make useless things : https://www.ted.com/talks/simone_giertz_why_you_should_make_useless_things

2. Installation drivers et IDE

a. Qu'est-ce qu'un IDE

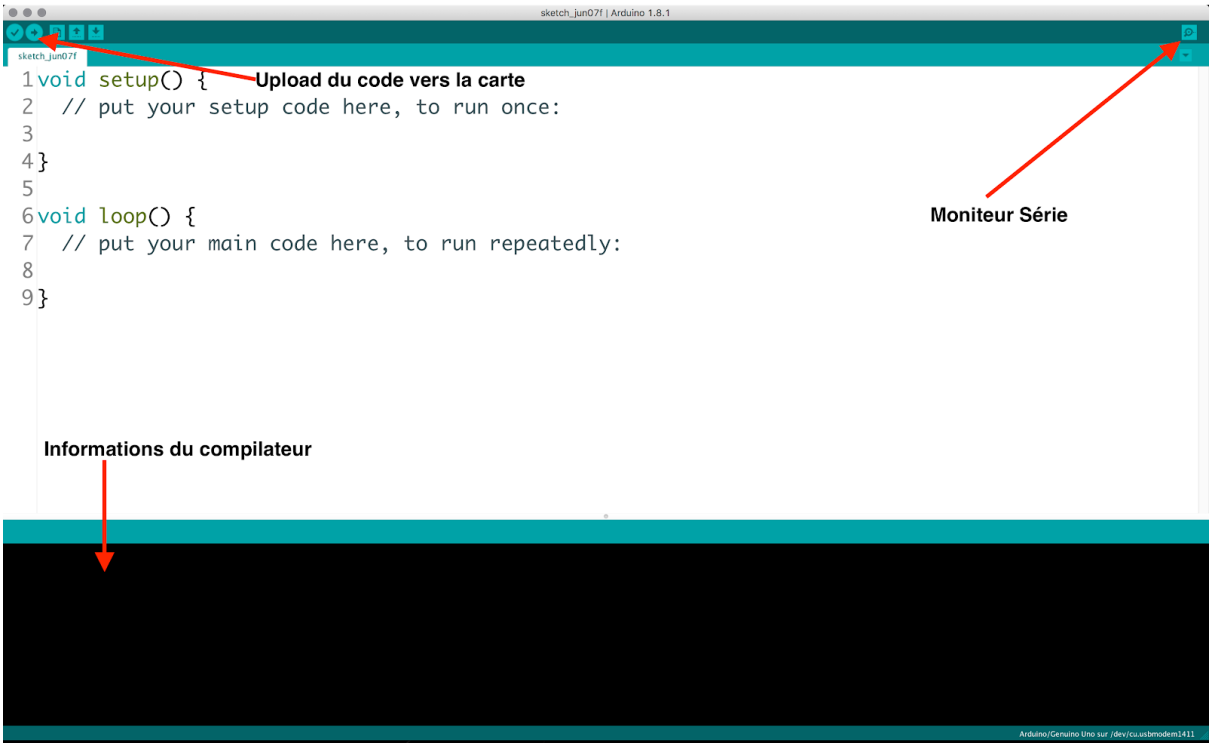
Un IDE ou Integrated Development Environment est logiciel qui contient un éditeur de texte amélioré facilitant le développement de code informatique. De plus, un ensemble d'outils facilitant le développement est souvent intégré. L'IDE Arduino comprend notamment un compilateur, la chaîne d'outils permettant de charger les programmes sur les cartes, un moniteur série, un gestionnaire de cartes et de bibliothèque…

b. Installation de l'IDE Arduino

- se rendre sur https://arduino.cc/downloads
- télécharger la dernière version du logiciel pour le système d'exploitation de votre machine.

- Ouvrir l'exécutable et suivre les instructions à l'écran

c. Présentation de l'IDE Arduino



3. Introduction à la programmation

La programmation Arduino de base se fait dans une version simplifiée du langage C++. Il est important de respecter certaines règles de syntaxe pour que le programme que vous écrivez puisse fonctionner.

a. Les instructions

Une instruction est un “ordre” que l’on donne à la machine. Dans les langages de haut niveau, les instructions sont représentées sous forme de lignes de code compréhensibles par les humains.
Les instructions d’un programme sont exécutées séquentiellement de haut en bas dans l’ordre dans lequel elles ont été écrites.
En C++, les instructions doivent se terminer par un point-virgule **(;)**
exemples :

- pinMode();
- digitalRead();

b. Les variables

Une variable permet de réserver un emplacement dans la mémoire de l’ordinateur afin d’y stocker une valeur. Une variable peut être nommée et doit avoir un type (**int**, **float**, **boolean** sont les principaux que nous utiliserons).

c. Les conditions (**if, else, else if)**

Les conditions sont des structures de contrôle permettant de prendre des décisions lors de l’exécution du programme. On utilisera les opérateurs logiques ou des valeurs booléennes pour faire des comparaisons, évaluer si une condition est remplie et agir en fonction.
Les différentes actions sont une série d’instructions comprises dans un bloc de code (délimité par les accolades **{} voir annexe et exemples pour la syntaxe**)

d. les boucles (while, for)

Les boucles sont des structures de contrôle permettant de répéter une même opération tant qu’une condition est (ou n’est pas) remplie.
Les instructions à répéter sont comprises dans un bloc de code (délimité par les accolades **{} voir annexe et exemples pour la syntaxe**)

e. les fonctions

Les fonctions sont un ensemble d'instructions ayant un objectif particulier réunies dans un même bloc de code. Les fonctions doivent être nommées, peuvent prendre des valeurs en paramètres (ou arguments) et peuvent retourner une valeur. Les instructions qui composent une fonction sont comprises dans un bloc de code (délimité par les accolades **{} voir annexe et exemples pour la syntaxe**) :

exemple :

```
int additionner(int a, int b){
  int resultat = a + b;
  return resultat;
}
```

f. les bibliothèques

Les bibliothèques (ou librairies) sont un ensemble de fonctions réunies pour réaliser une tâche particulière (ex. piloter un moteur), abstraire la complexité liée à un matériel particulier (écran LCD, capteurs ou Leds…) ou à fournir des fonctionnalités avancées (ex. une bibliothèque qui contiendrait des fonctions pour des calculs géométriques)

Structure d'un programme Arduino

void setup() : fonction exécutée par le microcontrôleur lors de sa mise sous tension. Le code contenu dans la fonction setup() n'est exécuté qu'une seule fois.

void loop() : fonction exécutée en boucle de manière continue tant que le microcontrôleur est sous tension. C'est dans cette 'boucle' que vous écrirez la plupart du code de vos projets.

4. Exemples

Digital

a. Blink
   digitalWrite
   i. changer vitesse de clignotement de la LED
   ii. LED sur breadboard
   iii. 2 LED clignotement alterné
b. button
   digitalRead
   i. allumer led si bouton est appuyé
   ii. allumer led quand le bouton est appuyé, l'éteindre au nouvel appui
   iii. allumer led si bouton appuyé 3 fois

Analogique

analogRead()

a. Potentiomètre
b. La librairie Serial
c. Phototransistor (pont diviseur de tension)

Références

Arduino MKR1000:

- https://store.arduino.cc/arduino-mkr1000-with-headers-mounted
- https://store.arduino.cc/arduino-iot-mkr1000-bundle

Arduino

- Fondations : https://www.arduino.cc/en/Tutorial/Foundations
- référence du langage : http://arduino.cc/reference/fr
- ITP/Tom Igoe (vimeo) https://vimeo.com/groups/itpcom
- IDE arduino : https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-the-arduino-desktop-ide-623be4?ref=user&ref_id=65561&offset=20

Inspiration

- https://blog.arduino.cc
- https://blog.adafruit.com
- https://hackster.io
- https://makezine.com
- https://instructables.com
- https://hackaday.com
- https://www.simonegiertz.com/

Chaînes Youtube

- Great Scott : https://www.youtube.com/channel/UC6mIxFTvXkWQVEHPsEdflzQ
- Andreas Spiess : https://www.youtube.com/channel/UCu7_D0o48KbfhpEohoP7YSQ
- Arduino : https://www.youtube.com/channel/UCUpmmT1Gm_raVpqSvQYyd2Q
- u=R/I : https://www.youtube.com/channel/UCVqx3vXNghSqUcVg2nmeqYA
- Le labo d'Heliox : https://www.youtube.com/channel/UCPFChjpOgkUqckj3378jt5w
- Les frères Poulain : https://www.youtube.com/channel/UCjED9uS41ioeFuPfbR-OBlw

Fournisseurs

fabricants + éducation

- Adafruit: https://adafruit.com
- Sparkfun: https://sparkfun.com
- Pimoroni: https://pimoroni.co.uk

Shops en ligne

- Microcontroller Hobby : https://shop.mchobby.be/
- Exp-tech : https://exp-tech.de
- Mouser : https://mouser.be (pro only ?)

ANNEXE : Memento

ARDUINO MEMENTO CODE

STRUCTURE DE BASE

int bouton1 = 14 ; // l'entrée 14 reçoit bouton1

// au début on déclare les variables qui vont être utilisées dans le programme si on en utilise

void setup() {
// ici on met le code à effectuer au démarrage, ne sera exécuté qu'une fois au début

}

void loop() {
// ici on met le code de la boucle, tourne en permanence une fois que le setup est terminé

}

Lexique et syntaxe

Instruction : un instruction est un “ordre” donné à la machine dans un langage compréhensible par elle. Dans les langage C et C++, toute instruction est terminée par un point-virgule **;**

Bloc de code : ensemble d’instructions délimité par des **{ }**

// double slash est utilisé commenter une ligne. Un commentaire n’est pas peris en compte dans le code, mais permet de décrire celui-ci
ou **/*** terminé par ***/** pour des commentaires sur plusieurs lignes

VARIABLES

Opérateurs :

= assignation
+ addition
- soustraction
***** multiplication
/ division
% modulo

COMPARATEURS

== est égal à
!= n'est pas égal à
< est inférieur à
> est supérieur à
<= est inférieur ou égal à
>= est supérieur ou égal à

TYPES de données

int	nombre entier (1,2,3,4...)
boolean	false, true (vrai ou faux)
float	nombre avec décimales (1.2, 3.98,...)
byte	nombre en binaire
char	caractère de texte à afficher (a,b,c, bonjour, résultat =, ...)

3 TYPES DE BOUCLES

WHILE

```
int var = 0;
while(var < 200){       // tant que la variable est inférieur à 200

  // fait quelque chose ici 200 fois de suite...

   var++;                // incrémente la variable, incrémenter veut dire augmenter (++)

}
```

DO WHILE

```
do                                  // faire ce qui suit...
{
   delay(50);                       // attendre la stabilisation du capteur
   x = readSensors();          // lit la valeur de la tension du capteur

} while (x < 100);                  // ...tant que x est inférieur à 100
```

FOR

```
for (int i=0; i <= 255; i++){
    analogWrite(pinPWM_3, i);
     delay(10);
}
// for (état initial, tant que i est inférieur ou égal à 255, ajouter/incrémenter de 1 jusqu'à 255) {
        écrire sur la pin 3 un PWM dont la valeur est i ;
        délais de 10 pour ralentir la procédure ;
}
```

FONCTION

```
void maFonction() {

 }
```

En général on créé les fonctions après le void loop et on l'ajoute dans le loop en mettant la valeur de retour et le nom de la fonction, dans cet exemple : maFonction() ;

CONDITION (contrôle)

```
if (x > 120){
        digitalWrite(LEDpin, HIGH);
}
else {
        digitalWrite(LEDpin, LOW);
}
```

// si x est supérieur à 120 alors on allume la led (HIGH) sur la pin LEDpin sinon elle est éteinte

Autre exemple

```
if (temperature >= 70)
{
  //couper le système car surchauffe (capteur)
}
else if (temperature >= 60 && temperature < 70)
{
  //attention sur chauffe
}
else
{
  //c'est ok on continue, condition normal
}
```

Quelques termes récurrents à arduino :

```
pinMode (pin, OUTPUT);
```
// dans le setup on définit si une pin est en entrée ou sortie

```
delay (10) ;
```
// ajoute un delay de 10 millisecondes dans la boucle

```
analogRead (pin) ;
```
// lit les données analogiques/continues sur la pin choisie

```
digitalRead (pin) ;
```
// lit la valeur HIGH ou LOW (5V ou 0V) sur la pin choisie

```
digitalWrite (pin, HIGH)
```
// écrire sur sortie digitale (pin choisie, HIGH = 3,3 ou 5 volts, LOW =0)

```
analogWrite (pin, 55)
```
// envoit un PWM sur la pin choisie avec une valeur entre 0 et 255 max.

```
serial.print (78, DEC) ;
```
// affiche 78 en décimal dans le moniteur

```
serial.println
```
// affiche des valeurs dans le moniteur

ex : *analogValue = analogRead(0);*
Serial.println(analogValue);

map(x, val min IN, val max IN, val min OUT, val max OUT);
II permet de changer les valeurs reçues et de les ajuster pour ses besoins (en midi par exemple on veut des données de 0 à 127 pour un controleur)

ex : *int y = map(x, 1, 50, 50, 1);*

Pour les librairies, veuillez consulter le site arduino, et google est aussi votre ami.

Codes couleurs résistances