Introduction à Arduino avec la carte Circuit Playground Classic d'Adafruit

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

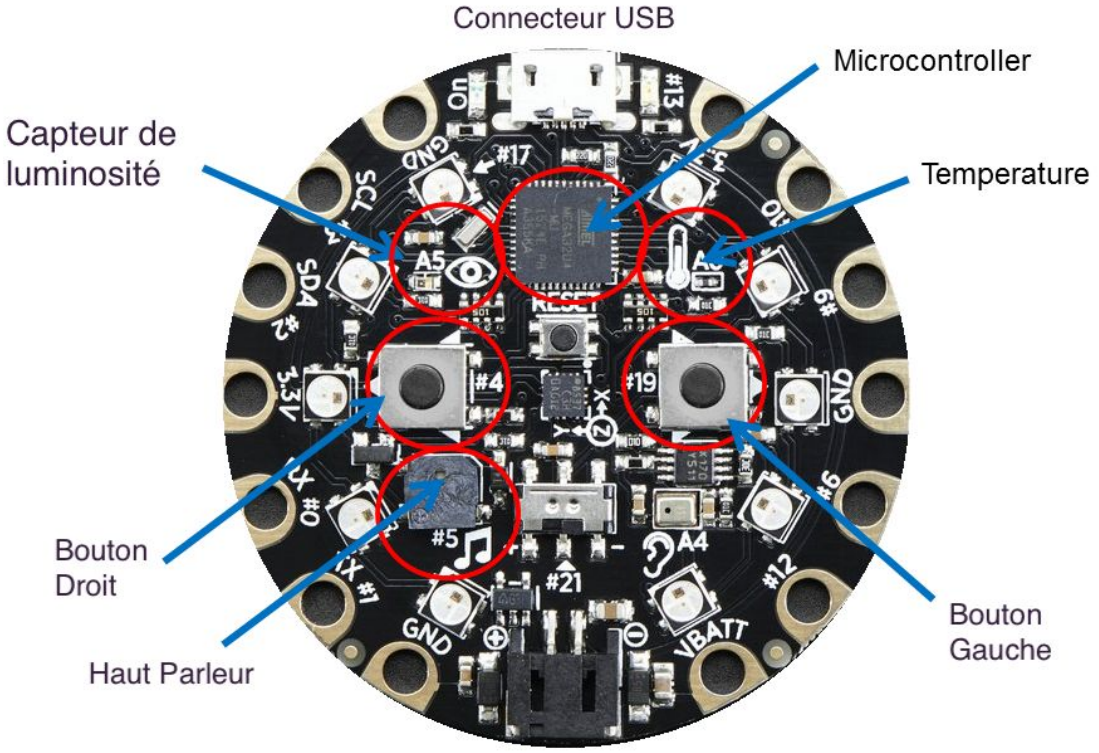
i. Massimo Banzi https://www.ted.com/talks/massimo_banzi_how_arduino_is_open_sourcing_imagination

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

c. Arduino vs Circuit Playground Classic

	Arduino UNO	Circuit Playground
processeur	AVR ATmega328p (16Mhz)	AVR ATmega32u4 (8Mhz)
architecture	8bits	8bits
mémoire	32KB flash + 2kb SRAM + 1KB EEPROM	32KB flash + 2KB RAM
tension de fonctionnement	5v	3.3v
entrée/sorties digitales	14	8
entrées analogiques	6	4
sorties PWM	6	4
entrées capacitives (tactiles)	-	8
SPI/UART/i2C	oui	oui

capteurs intégrés	-	7
actuateurs intégrés	1 (led sur pin 13)	3 (dont 10 leds RGB)



d. Présentation Circuit Playground

Circuit Playground Classic vs Circuit Playground Express
Le circuit playground express est une version plus ou moins similaire du circuit playground, à la grande différence que cette dernière est développée pour pouvoir faire tourner du code **Circuit Python** (une version simplifiée de MicroPython). Le Circuit Playground Express est également programmable via l'interface Graphique **MakeCode.**
Ce workshop se base sur la version Classic de la carte.

i. Le bouton RESET:

Un double appui permet de passer le Circuit Playground en mode *bootloader* (durant lequel il est possible de charger du code sur la

carte). Cela se fait normalement automatiquement, mais il arrive parfois que le microcontrôleur ne réponde pas.

ii. Pads : digital in/out | analog In | pwm | capacitif

https://learn.adafruit.com/circuit-playground-lesson-number-0/alligator-pads-pinout#pad-usage-pinout-17-7

iii. Capteurs

1. push buttons
2. switch
3. temperature
4. luminosité
5. acceleromètre
6. micro
7. capacitif

iv. Actuateurs

1. LED
2. Leds RGB (Neopixels)
3. Speaker

v. HID (Human Interface Device - Interface Homme-Machine)

À l'inverse de l'Arduino Uno, par exemple, le Circuit Playground peut être reconnu par les ordinateurs comme un périphérique USB. En utilisant les bibliothèques prévues, il est donc possible d'utiliser le Circuit Playground comme clavier ou comme souris. (très utile lorsque l'on veut créer rapidement un controlêur pour une application web [musique, jeux...])

e. Programme du workshop

i. JOUR 1 : introduction à la programmation
ii. JOUR 2: réalisation d'un projet
    1. Lampion RGB (http://danthegeek.com/2017/08/16/party-light-with-adafruits-circuit-playground/)
    2. Jeu Simon (https://learn.adafruit.com/circuit-playground-simple-simon/overview)
    3. Votre idée ?

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…

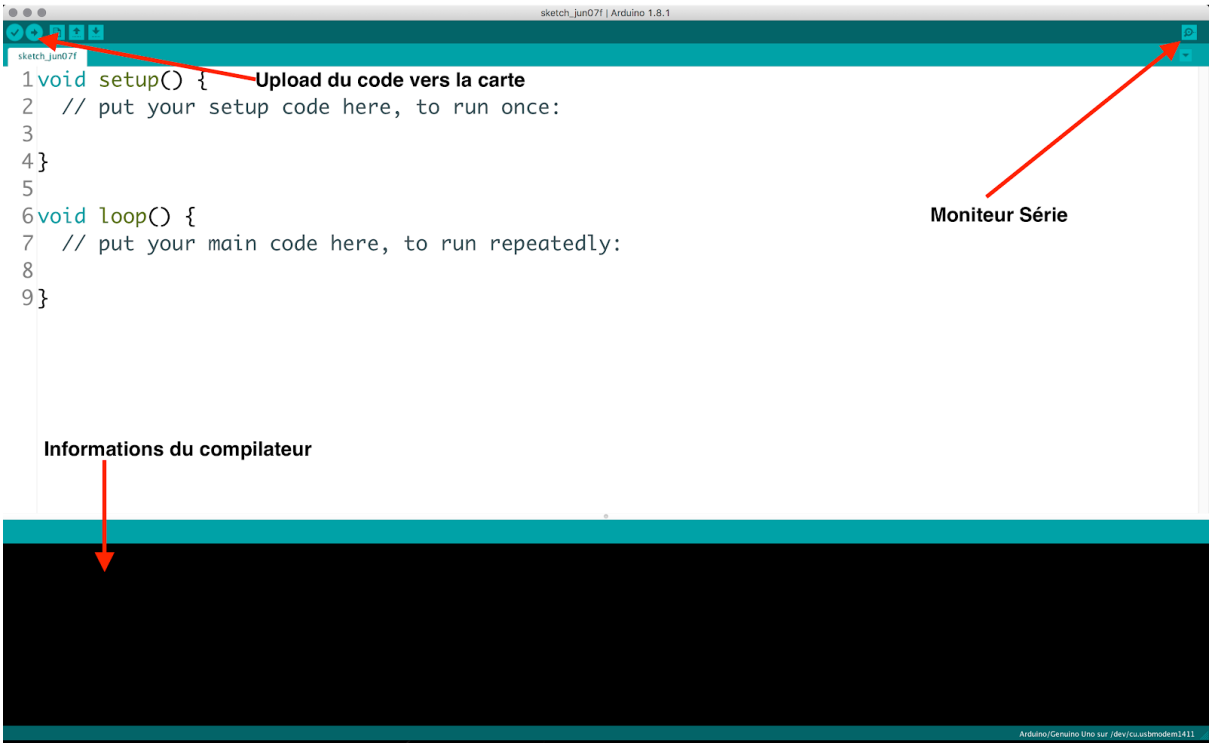
b. Les drivers Circuit Playground

(Uniquement pour les utilisateurs sous Windows)
https://learn.adafruit.com/introducing-circuit-playground/windows-driver-installation
https://github.com/adafruit/Adafruit_Windows_Drivers/releases/latest

c. 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

d. Présentation de l’IDE Arduino



e. Configuration de la carte, du port et utilisation de la librairie Circuit Playground

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 => Arduino way + CircuitPlayground way [structure, instructions, variables]
   digitalWrite | CircuitPlayground.redLED(true)
   i. changer vitesse de clignotement de la LED
b. button => Arduino way + CP way
   digitalRead | CircuitPlayground.leftButton() / CircuitPlayground.rightButton()
   i. allumer led si bouton est appuyé
   ii. allumer led si deux boutons sont appuyés
   iii. allumer led si bouton appuyé 3 fois
c. switch => arduino way + CP way
   i. clignoter LED si switch fermé, led fixe si éteint
   ii. changer la direction de rotation d'un pixel selon l'état du switch
d. pixels => CP (boucles)
   Neopixels => utilisation

Analogique

analogRead() |

a. light sensor => Arduino way + CP way
   i. faire varier la luminosité des pixels en fonction du capteur de luminosité (fonction map)
b. Accelerometer (libraries)
   i. déplacer un pixel en fonction de la position du circuit playground

Tactile (entrées capacitives)

CircuitPlayground.readCap(pin#)

a. allumer tous les pixels au moment de l'appui sur un pad
b. allumer le pixel en vis à vis du pad

HID

En utilisant les librairies **keyboard.h** ou **mouse.h**, il est possible de faire passer le circuit playground pour un clavier ou une souris et d'utiliser les différents capteurs comme source pour envoyer des données vers l'ordinateur.
(*dans le programme Arduino, voir exemples > 09.USB)*

a. utiliser la fonctionnalité tactile des pads pour simuler l'appui sur les flèches (HAUT, BAS, GAUCHE, DROITE)

b. Utiliser l’accéléromètre pour simuler les déplacements de la souris

Références

Adafruit Circuit Playground (en anglais):

- présentation : https://learn.adafruit.com/introducing-circuit-playground
- Leçon #0 : https://learn.adafruit.com/circuit-playground-lesson-number-0
- digital input : https://learn.adafruit.com/circuit-playground-digital-input
- analogique : https://learn.adafruit.com/circuit-playground-analog-input
- acceleromètre : https://learn.adafruit.com/circuit-playgrounds-motion-sensor
- Inspiration/Tutos : https://learn.adafruit.com/category/classic
- Liste des fonctions fournies par la librairie : https://caternuson.github.io/Adafruit_CircuitPlayground/

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

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/UCVqx3vXNghSqUcVg2nmegYA
- 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.

List des méthodes de la classe Adafruit_CircuitPlayground

- **Red LED**
  - void redLED(boolean v)
- **Slide Switch**
  - boolean slideSwitch(void)
- **Push Buttons**
  - Left : boolean leftButton(void)
  - Right : boolean rightButton(void)
- **Light Sensor**
  - uint16_t lightSensor(void)
- **Neopixels (led RGB)**
  - void clearPixels(void)
  - void setPixelColor(uint8_t p, uint32_t c)
  - void setPixelColor(uint8_t p, uint8_t r, uint8_t g, uint8_t b)
  - void setBrightness(uint16_t b)
  - uint32_t colorWheel(uint8_t x)
  - Basic color detection:
    - void senseColor(uint8_t& red, uint8_t& green, uint8_t& blue)
    - uint32_t senseColor()
- **Capacitive touch**
  - uint16_t readCap(uint8_t p, uint8_t samples=10)
- **Temperature sensor**
  - Celsius : float temperature(void)
  - Farenheit float temperatureF(void)
- **Microphone**
  - uint16_t soundSensor(void)
- **Buzzer**
  - void playTone(uint16_t freq, uint16_t time, boolean wait=true)
- **Accelerometer**
  - float motionX(void)
  - float motionY(void)
  - float motionZ(void)