Arduino

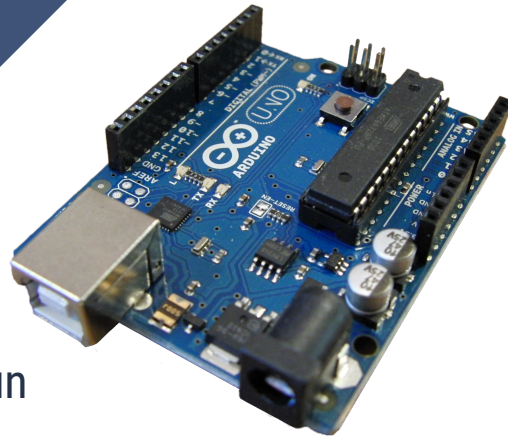
Vincent DEJONGHE
Amélie LAPORTE
Adrien MENDES SANTOS
Rodolphe TWOREK

université de BORDEAUX

Suivez le cours sur:

https://anthagonas.github.io/arduino/

Qu’est ce que Arduino?



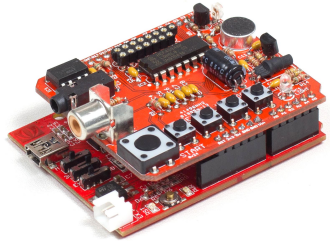
Un circuit imprimé en matériel libre sur lequel se trouve un microcontrôleur.

Une petite unité de calcul accompagné de mémoire, de ports d’entrée/sortie et de périphériques permettant d’interagir avec son environnement.

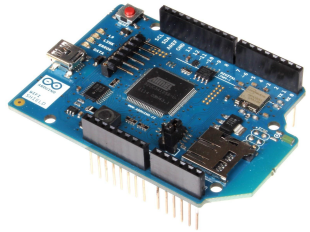
Les atouts

- Shields:

Plutôt que de devoir souder et créer les fonctionnalités que l'on veut donner à son arduino, il est possible d'ajouter directement un shield.



= Une carte que l'on ajouter directement à son Arduino qui est livrée avec une bibliothèque permettant de le contrôler.



- IDE Arduino:

Gratuit et téléchargeable en ligne

- Langage de programmation: basé sur C++

Programmer avec Arduino

Setup: Toutes les opérations nécessaires à la configuration de la carte.

Loop: Exécute en boucle après l'exécution de la fonction setup.

```
void setup(){
}

void loop(){
}
```

Les variables

Nom	Contenu	Plage de valeurs
char	Entier ou caractère	(0 -> 255) -128 -> 127
int	Entier	(0 -> 65 535) -32 768 -> 32 767
long	Entier	(0 -> 4 294 697 295) -2 147 483 648 -> 2 147 483 647
Float / double	Nombre à virgule Flottante	-3,4028235 E+38 -> 3,4028235E+38
string	Chaîne de caractères (Objet)	Aucune
Booléen	Booléen	true/false

Les opérateurs

=	affectation
==	comparaison
!=	différence
<, >	Inférieur, supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
&&	et
||	ou
!	non

Mathématique

+	plus
-	moins
*	multiplication
/	division
%	modulo

Les boucles

If:

```
if(<valeur booléenne>){
<instruction>;
}else{
<instruction>;
}
```

Switch:

```
Switch(<variable>){
Case <valeur>:
<instruction>;
Break;
default:
<instruction>;
}
```

Les boucles

While:

```
while(<valeur booleenne>)
{
<instruction>;
}
```

For:

```
for(<initialisation>;<valeur booléenne>;<évolution>)
{
<instruction>;
}
```

Fonctions , structures et classes

Fonction:

```
<type de retour> <nom>(<paramètre>)
{
<instruction>;
}
```

Structure :

```
struct <nom> {
<type> <nom du champ>;
}
```

Classe:

```
class
<nom> {
public:
<attributs et champ publics>;
private:
<attributs et champ privés>;
protected:
<attribut et champ privés>;
}
```

Fonctions de bases

Les fonctions les plus utilisée sont les fonctions d'entrée/sorties.

Ce sont elles qui permettent d'envoyer ou de mesurer une tension sur une des broches de la carte.

Il est nécessaire de définir la direction des broches utilisées via la fonction **pinMode** en lui donnant d'une part, la broche concernée, et d'autre part, la direction.

```
void setup() {

pinMode(1,OUTPUT); //
Broche 1 en sortie

 pinMode(2,INPUT ) ; //
Broche 2 en entrée

}
```

Entrées / Sorties

Toutes les broches sont capables d’écrire et de lire des données numériques (c'est-à-dire des 0 (0V) ou des 1 (5V)).

Il existe un type de branche, les broches PWM permettant de moduler le temps passé à l’état 1 (5V).

Ces fonctionnalités sur les broches d’entrées sorties sont utilisables par le biais de quatre fonctions.

Entrées / Sorties

Ces fonctionnalités sur les broches d'entrées sorties sont utilisables par le biais de deux fonctions :

- **digitalRead(pin):** mesure une donnée numérique sur une des broches, la broche en question doit être réglée en entrée.

- **digitalWrite(pin,value):** écrit une donnée numérique sur une des broches, la broche concernée doit être réglée en sortie. Le paramètre value doit être égal à HIGH (état 1 soit 5V) ou LOW (état 0 soit 0V).

La gestion du temps

Fonctions permettant de gérer le temps.

- **delay et delayMicroseconds :**

Pause. Insère une pause suivant le paramètre passé, bloque le microcontrôleur, on ne peut alors plus effectuer aucune action.

- **millis et micros :**

Mesure du temps. Permet d'incrémenter un intervalle pour une action donné. Ces fonctions incrémente une variable.

Les interruptions

Parfois nécessaire, d'attendre un événement externe (appui sur un bouton, données d'un capteur, etc.) pour effectuer une action.

Utilisation des fonctions d'interruptions :

- **attachInterrupt et detachInterrupt**

Ce mécanisme interrompt le code exécuté, il est prioritaire par rapport au reste du code.

Les interruptions - Suite

4 types d'événements :

LOW: Lorsque la broche est à l'état 0 (0V).

RISING: Lorsque la broche passe de l'état 0 (0V) à l'état 1 (5V) (front montant).

FALLING: Lorsque la broche passe de l'état 1 (5V) à l'état 0 (0V) (front descendant).

CHANGE: Lorsque la broche change d'état (front montant et front descendant).

Les interruptions - Exemple

```
volatile boolean etat=false;
void appuiBouton(){
    etat=!etat; //Changement d'état
}
void setup(){
    pinMode(2,INPUT); //Broche 2 en entrée
    attachInterrupt(0,appuiBouton,RISING); // On attache à
l’interruption 0 (broche 2) la fonction appuiBouton sur un front montant
}
```