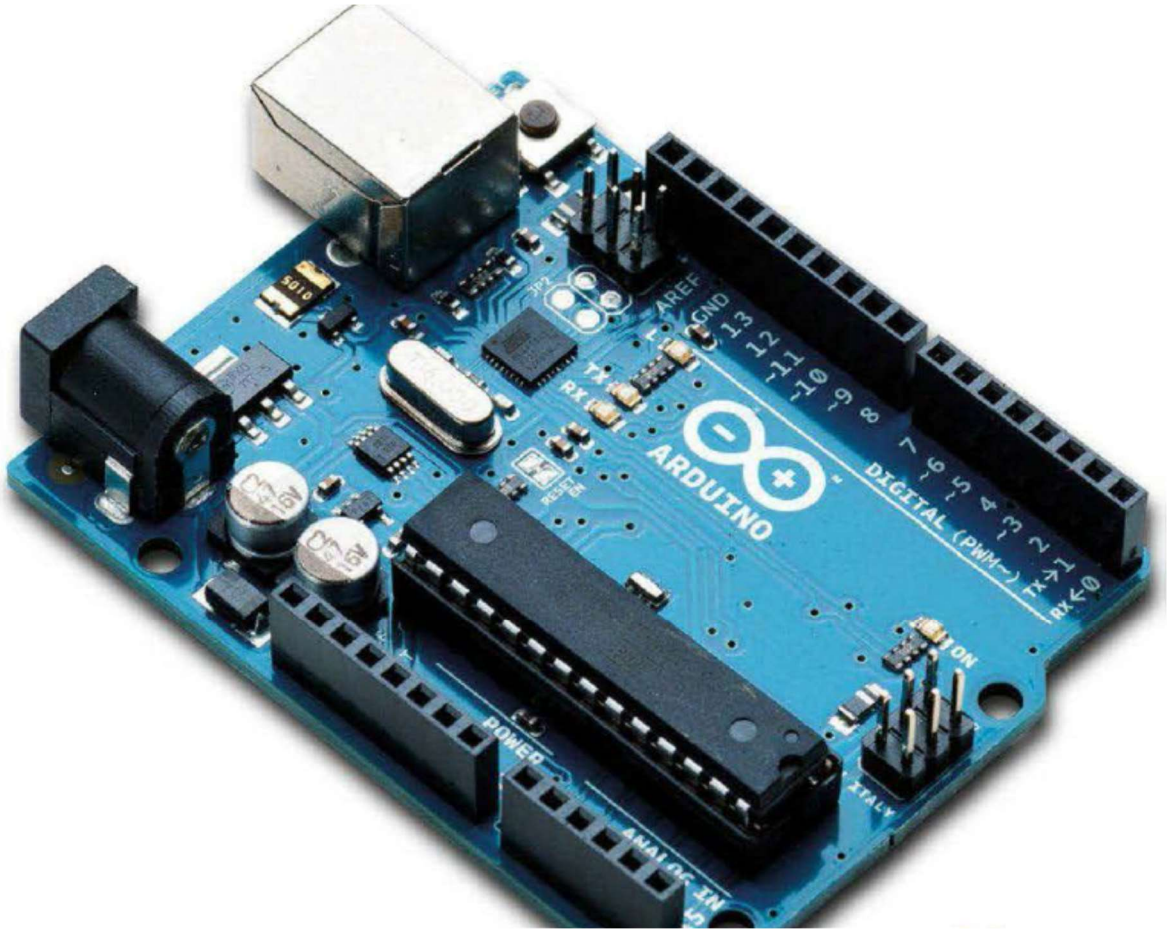
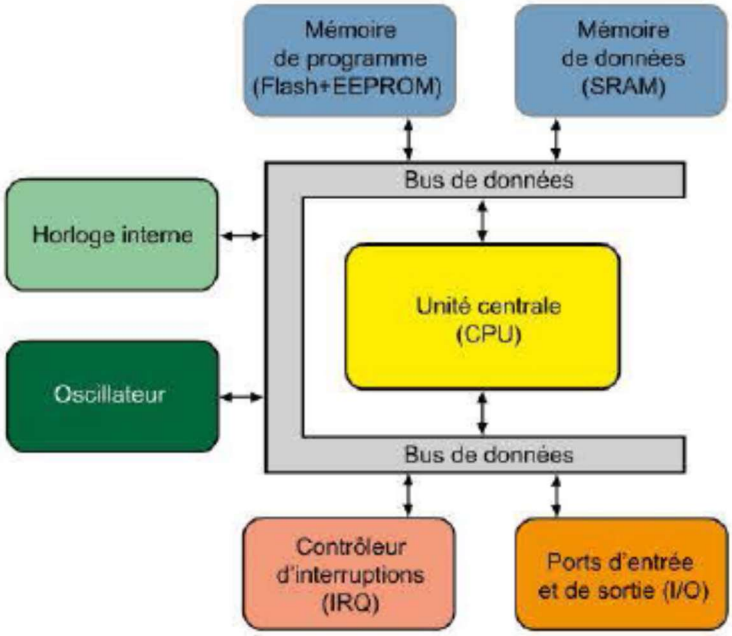
L3 AUTOMATIQUE

PFC 2023/2024

LA CARTE ARDUINO



ATmega32 de Atmel

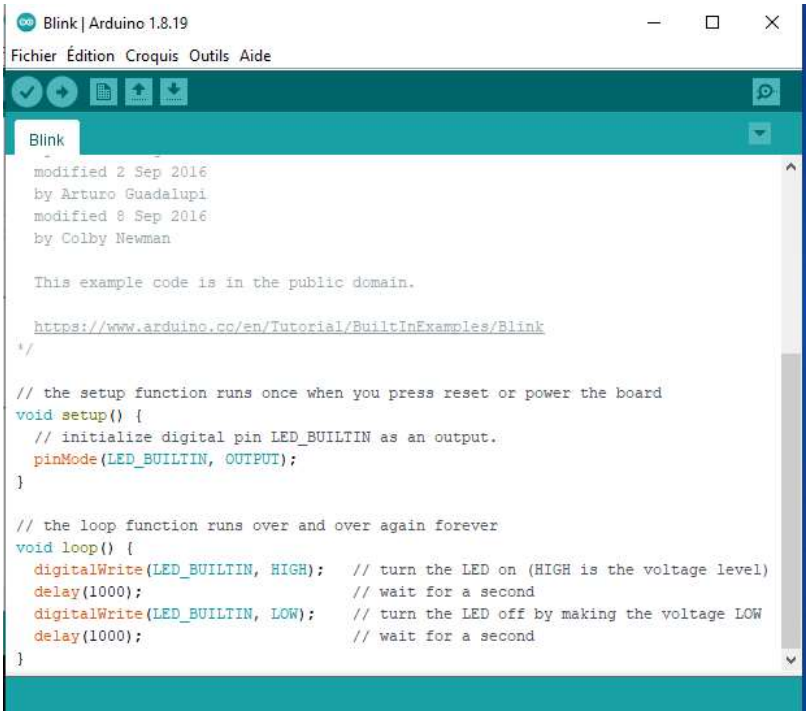


L'unité centrale (CPU)

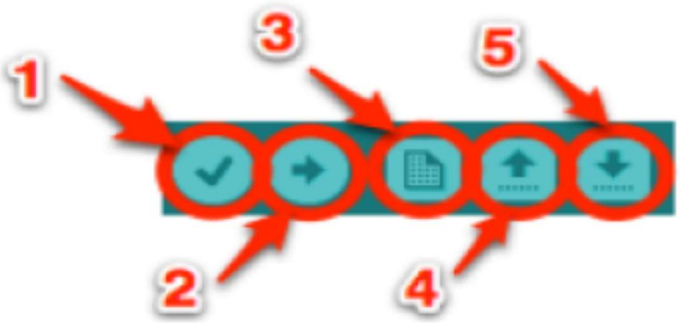
L'élément le plus important dans un microcontrôleur est l'unité centrale, appelée également CPU *(Central Processing Unit)*. Sa fonction principale consiste à décoder et à exécuter des instructions. Elle peut adresser des mémoires, gérer des entrées ou sorties et réagir à des interruptions *(interrupts)*. Une interruption (IRQ, ou *Interrupt Request*) est un signal qui demande au CPU d'interrompre un cycle de calcul en cours pour pouvoir réagir à un certain événement.

LOGICIEL DE PROGRAMMATION

Arduino IDE


Blink | Arduino 1.8.19
Fichier Édition Croquis Outils Aide
Blink
modified 2 Sep 2016
by Arturo Guadalupi
modified 8 Sep 2016
by Colby Newman
This example code is in the public domain.
https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}


code1 | Arduino 1.5.8
Menu
code1
/*
Code 1 - Edurobot.ch, destiné à l'Arduino
Objectif: faire clignoter la LED montée sur le port 13
*/
Fenêtre de programmation
//***** FONCTION SETUP = Code d'initialisation *****
// la fonction setup() est exécutée en premier et une seule fois, au démarrage d
void setup() // début de la fonction setup()
{
pinMode(13, OUTPUT); // Initialise la broche 13 comme sortie
Serial.begin(9600); // Ouvre le port série à 9600 bauds
} // fin de la fonction setup()
//***** FONCTION LOOP = Boucle sans fin = coeur du programme *****
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino
void loop() // début de la fonction loop()
Fenêtre de contrôle
1
Arduino Uno on /dev/cu.wchusbserial1410



• **Bouton 1** : ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le programme ;

• **Bouton 2 :** envoi du programme sur l'Arduino ;

• **Bouton 3 :** créer un nouveau fichier ;

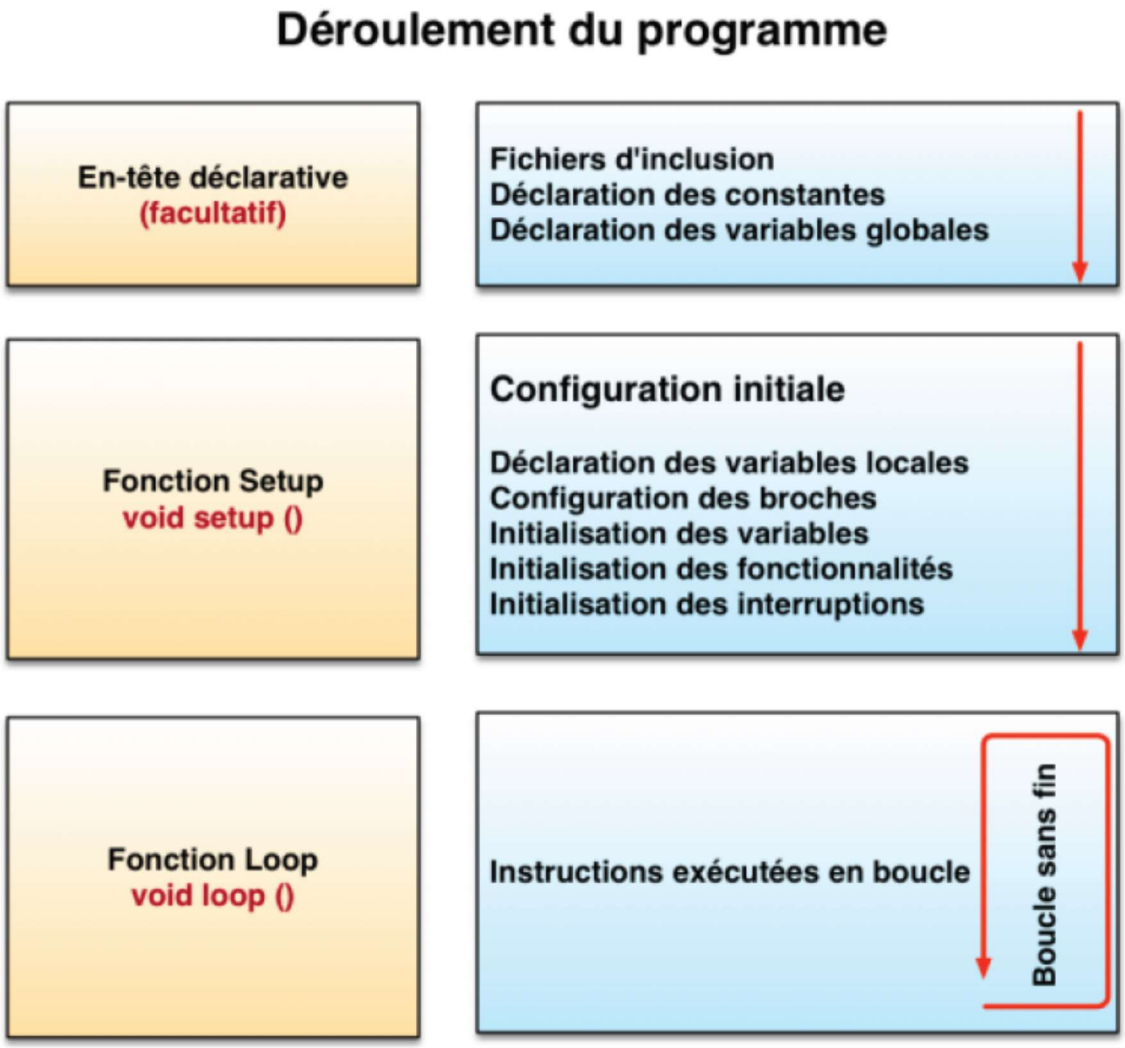
• **Bouton 4 :** ouvrir un fichier existant ;

• **Bouton 5** : enregistrer un fichier

LANGAGE DE PROGRAMMATION

Le langage de programmation utilisé est le **C++, compilé avec avr-g++, et lié à la bibliothèque de développement Arduino, permettant d'utiliser la carte et ses entrées/sorties**.

STRUCTURE D'UN PROGRAMME ARDUINO



Coloration syntaxique

Lorsque du code est écrit dans l'interface de programmation, certains mots apparaissent en différentes couleurs qui clarifient le statut des différents éléments :

En **orange**, apparaissent les mots-clés reconnus par le langage Arduino comme des **fonctions** existantes. Lorsqu'on sélectionne un mot coloré en orange et qu'on effectue un clic avec le bouton droit de la souris, l'on a la possibilité de choisir « Find in reference » : cette commande ouvre directement la documentation de la fonction sélectionnée.

En **bleu**, apparaissent les mots-clés reconnus par le langage Arduino comme des **constantes**.

En **gris**, apparaissent les **commentaires** qui ne seront **pas exécutés dans le programme**. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes. L'on peut déclarer un commentaire de deux manières différentes :

- dans une ligne de code, tout ce qui se trouve après « *//* » sera un commentaire.
- l'on peut encadrer des commentaires sur plusieurs lignes entre « */** » et « ***/** ».

La syntaxe du langage

ponctuation

Le code est structuré par une ponctuation stricte :

- **toute ligne** de code se termine par un point-virgule « **;** »
- le contenu d'une **fonction** est délimité par des accolades « **{** » et « **}** »
- les **paramètres** d'une fonction sont contenus pas des parenthèses « **(** » et « **)** ».

Une erreur fréquente consiste à oublier un de ces éléments.

Les variables

Une variable est un espace réservé dans la mémoire de l'ordinateur. C'est comme un compartiment dont la taille n'est adéquate que pour un seul type d'information. Elle est caractérisée par un nom qui permet d'y accéder facilement.

Il existe différents types de variables identifiés par un mot-clé dont les principaux sont :

- nombres entiers (int)
- nombres à virgule flottante (float)
- texte (String)
- valeurs vrai/faux (boolean).

Un nombre à décimales, par exemple *3.14159*, peut se stocker dans une variable de type float. Notez que l'on utilise un point et non une virgule pour les nombres à décimales. Dans Arduino, il est nécessaire de déclarer les variables pour leurs réserver un espace mémoire adéquat. On déclare une variable en spécifiant son type, son nom puis en lui assignant une valeur initiale (optionnel). Exemple :

```
int ma_variable = 45;
 // int est le type,  ma_variable le nom et = 45 assigne une valeur.
```

Type de variable	Type de nombre stocké	Valeurs maximales du nombre stocké	Nombre sur X bits	Nombre d'octets
int	entier	-32 768 à +32 767	16 bits	2 octets
long	entier	-2 147 483 648 à +2 147 483 647	32 bits	4 octets
char	entier	-128 à +127	8 bits	1 octet
float	décimale	-3.4×10^{38} à $+3.4 \times 10^{38}$	32 bits	4 octets
double	décimale	-3.4×10^{38} à $+3.4 \times 10^{38}$	32 bits	4 octets

Type de variable	Type de nombre stocké	Valeurs maximales du nombre stocké	Nombre sur X bits	Nombre d'octets
unsigned char	entier non négatif	0 à 255	8 bits	1 octet
unsigned int	entier non négatif	0 à 65 535	16 bits	2 octets
unsigned long	entier non négatif	0 à 4 294 967 295	32 bits	4 octets

Les fonctions

Une fonction (également désignée sous le nom de procédure ou de sous-routine) est un bloc d'instructions que l'on peut appeler à tout endroit du programme.

Le langage Arduino est constitué d'un certain nombre de fonctions, par exemple analogRead(), digitalWrite() ou delay().

Il est possible de déclarer ses propres fonctions par exemple :

```
void clignote(){
    digitalWrite (brocheLED,  HIGH) ;
    delay (1000) ;
    digitalWrite (brocheLED,  LOW) ;
    delay (1000) ;
}
```

Pour exécuter cette fonction, il suffit de taper la commande :

```
clignote();
```

On peut faire intervenir un ou des **paramètres** dans une fonction :

```
void clignote(int broche,int vitesse){
    digitalWrite (broche, HIGH) ;
    delay (1000/vitesse) ;
    digitalWrite (broche, LOW) ;
    delay (1000/vitesse) ;
}
```

Dans ce cas, l'on peut moduler leurs valeurs depuis la commande qui l'appelle :

```
clignote(5,1000); //la sortie 5 clignotera vite
clignote(3,250); //la sortie 3 clignotera lentement
```

Les structures de contrôle

Les structures de contrôle sont des blocs d'instructions qui s'exécutent en fonction du respect d'un certain nombre de conditions.

Il existe quatre types de structure :

if...else : exécute un code **si** certaines conditions sont remplies et éventuellement exécutera un autre code avec **sinon.**

exemple :

```
//si la valeur du capteur depasse le seuil
if(valeurCapteur>seuil){
    //appel de la fonction clignote
    clignote();
}
```

while : exécute un code **tant** que certaines conditions sont remplies.

exemple :

```
//tant que la valeur du capteur est supérieure à 250
while(valeurCapteur>250){
    //allume la sortie 5
    digitalWrite(5,HIGH);
    //envoi le message "0" au port serie
    Serial.println(1);
    //en boucle tant que valeurCapteur est supérieue à 250
}

Serial.println(0);

digitalWrite(5,LOW);
```

for : exécute un code **pour** un certain nombre de fois.

exemple :

```
//pour i de 0 à 255, par pas de 1
for (int i=0; i <= 255; i++){
    analogWrite(PWMpin, i);
    delay(10);
}
```

switch/case : fait un choix entre plusieurs codes **parmi une liste de possibilités**

exemple :

```
// fait un choix parmi plusieurs messages reçus
switch (message) {
    case 0: //si le message est "0"
        //allume que la sortie 3
        digitalWrite(3,HIGH);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
        break;
    case 1: //si le message est "1"
        //allume que la sortie 4
        digitalWrite(3,HIGH);
        digitalWrite(4,LOW);
        digitalWrite(5,LOW);
        break;
    case 2: //si le message est "2"
        //allume que la sortie 5
        digitalWrite(3,LOW);
        digitalWrite(4,LOW);
        digitalWrite(5,HIGH);
        break;
    }
```

Exemple

L'exemple qui suit montre l'utilisation de quelques éléments de la syntaxe du langage Arduino.

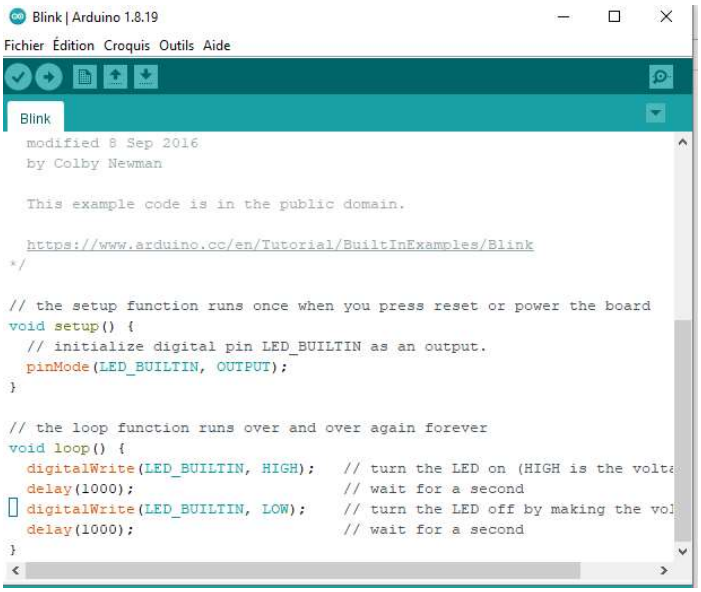
```
/*
Dans ce programme, un signal analogique provenant d'un capteur (potentiomètre)
fait varier la vitesse de clignotement d'une LED, à partir d'un certain seuil
*/
////declaration des variables
// selection de la broche sur laquelle est connectée le capteur
int brocheCapteur = 0;
// selection de la broche sur laquelle est connectée la LED
int brocheLED = 13;
// variable stockant la valeur du signal reçu du capteur
int valeurCapteur = 0;
//seuil de déclenchement
int seuil= 200;
//////////initialisation
void setup () {
    // broche du capteur configurée en entrée
    pinMode (brocheCapteur, INPUT) ;
    // broche de la LED configurée en sortie
    pinMode (brocheLED, OUTPUT) ;
}
//////////boucle principale
void loop () {
    // lecture du signal du capteur
    valeurCapteur = analogRead (brocheCapteur) ;
    //condition de declenchement
    if(valeurCapteur>seuil){
        //appel de la fonction clignote
        clignote();
    }
}
/////fonction personnalisée de clignotement
void clignote(){
```

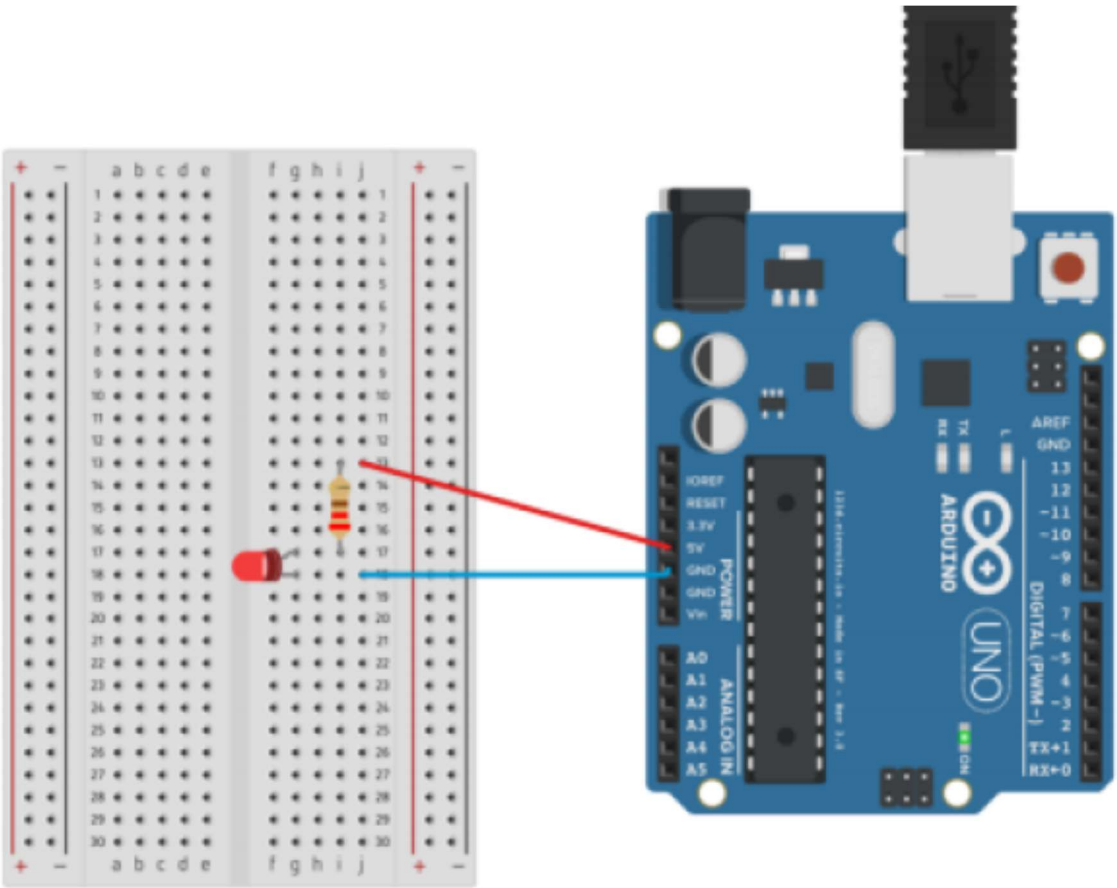
```
    // allume la LED
    digitalWrite (brocheLED, HIGH) ;
    // délai de «valeurCapteur" millisecondes
    delay (valeurCapteur) ;
    // éteint la LED
    digitalWrite (brocheLED, LOW) ;
    // delai de «valeurCapteur" millisecondes
    delay (valeurCapteur) ;
}
```

Le principe de fonctionnement est le suivant :

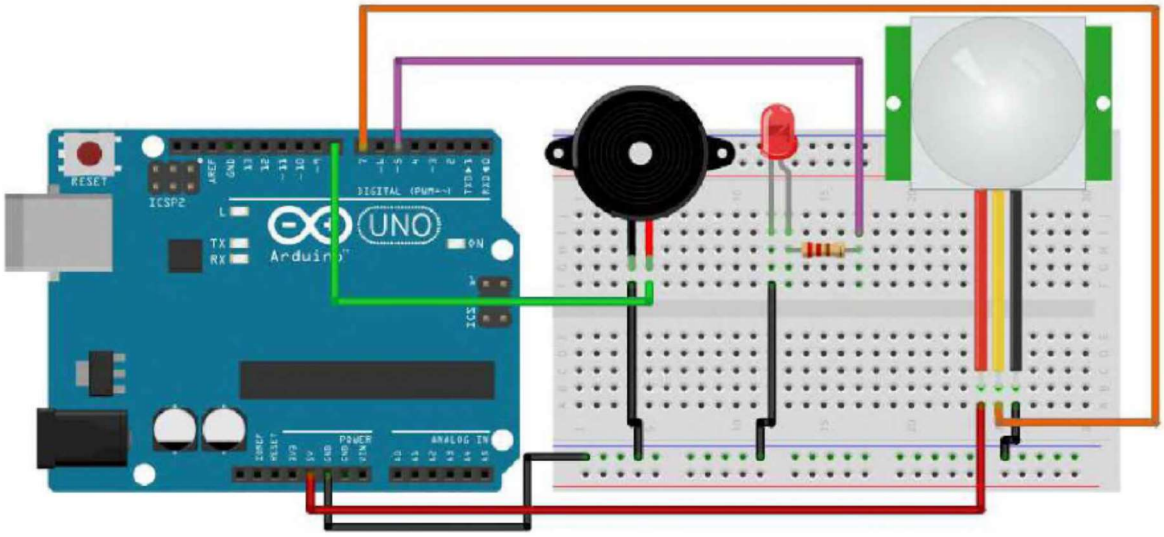
1. Le signal est lu avec la fonction « analogRead () ».
2. La valeur du signal varie en 0 et 1023.
3. Si la valeur dépasse le seuil, la LED est allumée et éteinte pendant un délai correspondant à la valeur du signal reçu du capteur.

EXEMPLE 1

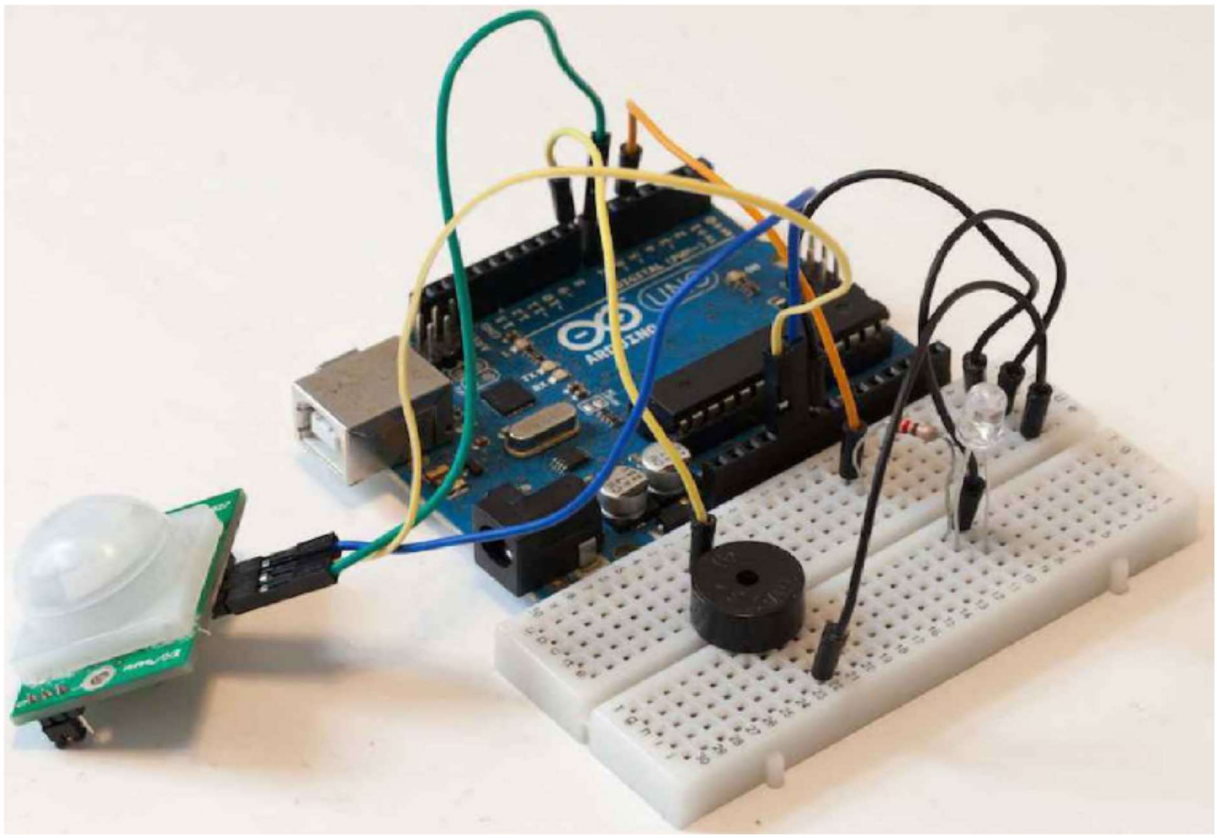

Blink | Arduino 1.8.19
Fichier Édition Croquis Outils Aide
Blink
modified 8 Sep 2016
by Colby Newman
This example code is in the public domain.
https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the volt
delay(1000); // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the vo
delay(1000); // wait for a second
}


ARDUINO
UNO
POWER
ANALOG IN
DIGITAL (PWM~)

EXEMPLE UN SYSTEME D'ALARME SIMPLE



SCHEMAS SYNOPTIQUE



Projet entièrement assemblé