

# Document pour l'initiation à l'électronique et Arduino

Romain THOMAS

24 août 2019

# Sommaire

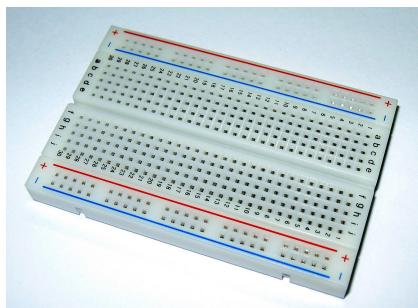
<b>1 Bases de l'électronique</b>	<b>2</b>
1.1 La breadboard . . . . .	2
1.2 La LED . . . . .	2
1.3 L'interrupteur . . . . .	2
1.4 La diode et le sens du courant . . . . .	3
1.5 La LED RGB (Red, Green, Blue) . . . . .	3
1.6 Le moteur . . . . .	3
1.7 Le transistor MOSFET . . . . .	4
<b>2 Débuter avec Arduino</b>	<b>5</b>
2.1 Présentation de la carte Arduino nano . . . . .	5
2.2 Résumé des différentes boucles, conditions et fonctions . . . . .	5
2.2.1 Créer un programme Arduino . . . . .	5
2.2.2 Interagir avec les pin de l'Arduino . . . . .	6
2.2.3 Les variables . . . . .	6
2.2.4 Les conditions . . . . .	6
2.2.5 Les boucles . . . . .	7
2.3 Allumer une LED et téléverser le programme sur la carte . . . . .	7
2.4 Utiliser un bouton poussoir et une LED . . . . .	8
2.5 Utiliser la transmission série . . . . .	8
2.6 Le PWM avec la fonction analogWrite . . . . .	9
2.7 Utiliser un moteur . . . . .	10
2.8 Utiliser un moteur en PWM . . . . .	10
2.9 La LED RGB . . . . .	11
2.10 Le sonar HC-SR04 . . . . .	12
2.11 Le pont en H L293D . . . . .	13
<b>3 Utiliser des bibliothèques</b>	<b>15</b>
3.1 La bibliothèque Motor . . . . .	15
<b>A Les logiciels pour Arduino</b>	<b>17</b>
A.1 Les logiciels à installer . . . . .	17
A.2 Ajouter une bibliothèque avec Arduino IDE . . . . .	17
A.2.1 Ajouter une bibliothèque pour tout les projets . . . . .	17
A.2.2 Ajouter une bibliothèque pour un seul projet . . . . .	18
A.3 Ajouter une bibliothèque avec PlatformIO . . . . .	18
<b>B Électronique et programmation</b>	<b>19</b>
B.1 Le code couleur des résistances . . . . .	19
B.2 Les types de variables . . . . .	20

# Chapitre 1

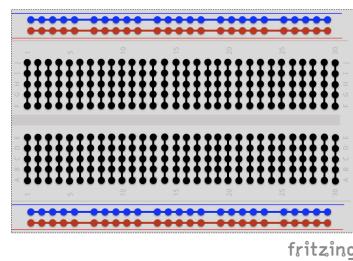
## Bases de l'électronique

## 1.1 La breadboard

La breadboard sert à réaliser des montages électroniques afin de les tester. La figure 1.2 montre comment les emplacements sont reliés à l'intérieur.



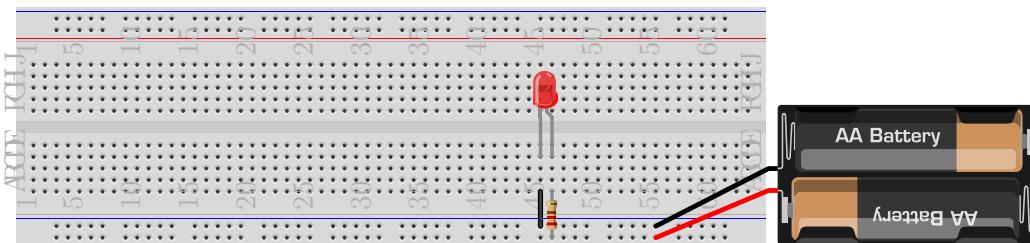
**FIGURE 1.1** – Photo d'une breadboard



**FIGURE 1.2** – Schéma des liaisons électriques dans la breadboard

## 1.2 La LED

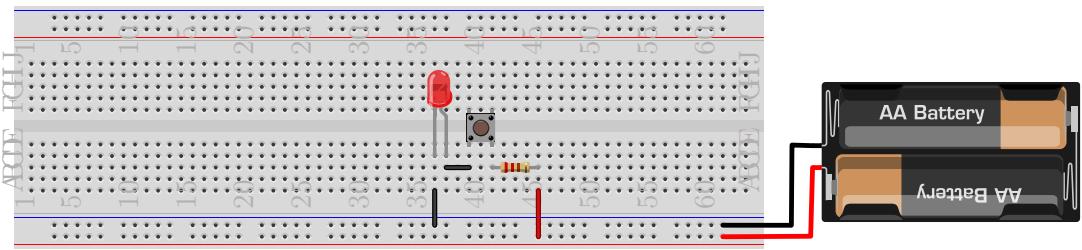
Cet exemple montre la manière de brancher une LED : il faut ajouter une résistance en série avec la LED sous peine de l'endommager. La plus grande patte de la led correspond au +.



**FIGURE 1.3** – Schéma du branchement d'une LED avec une résistance

### 1.3 L'interrupteur

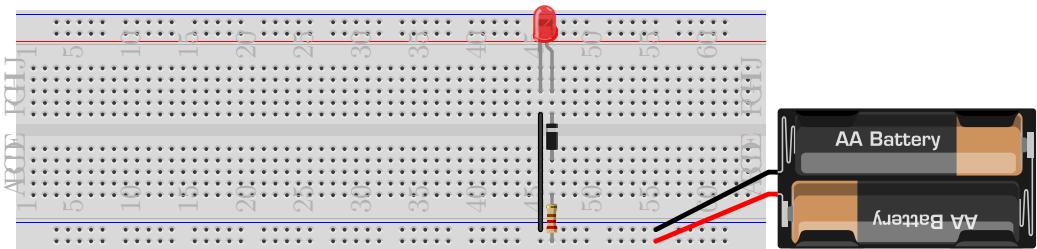
Avec ce montage, la LED s'allume quand l'interrupteur est pressé.



**FIGURE 1.4** – Schéma du branchement d'une LED avec un interrupteur

## 1.4 La diode et le sens du courant

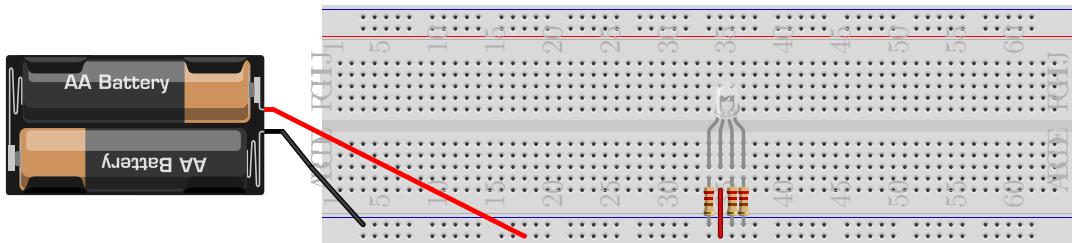
Dans cette exemple on va voir le principe de la diode : si on branche la diode dans l'autre sens, la LED ne s'allume plus. En effet, la diode laisse passer le courant dans un seul sens



**FIGURE 1.5** – Schéma du branchement d'une LED avec une résistance et une diode

## 1.5 La LED RGB (Red, Green, Blue)

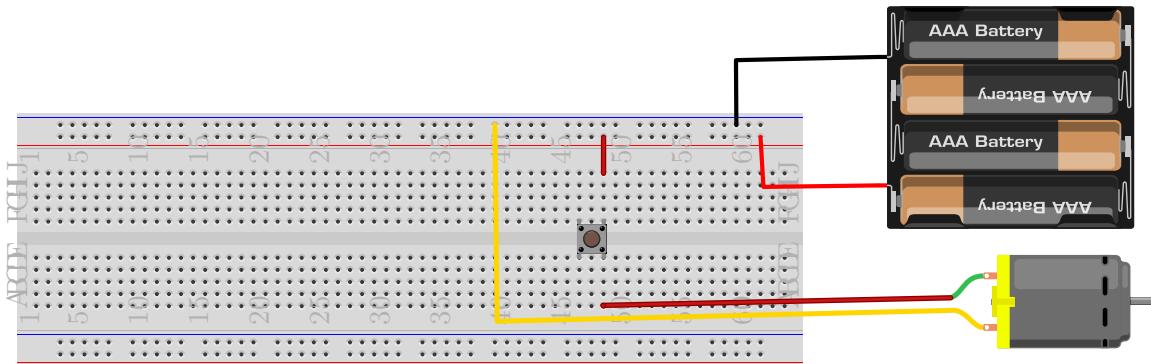
Ici, on utilise une LED RGB (Red, Green, Blue : Rouge, Vert, Bleu). Avec cette LED, on émettre toutes les couleurs visibles par un être humain : pour cela, on peut déconnecter des résistances pour ne pas éclairer d'une couleur ou changer la valeur des résistances pour éclairer plus ou moins.



**FIGURE 1.6** – Schéma du branchement d'une LED RGB

## 1.6 Le moteur

Avec ce montage, le moteur s'allume quand l'interrupteur est pressé.

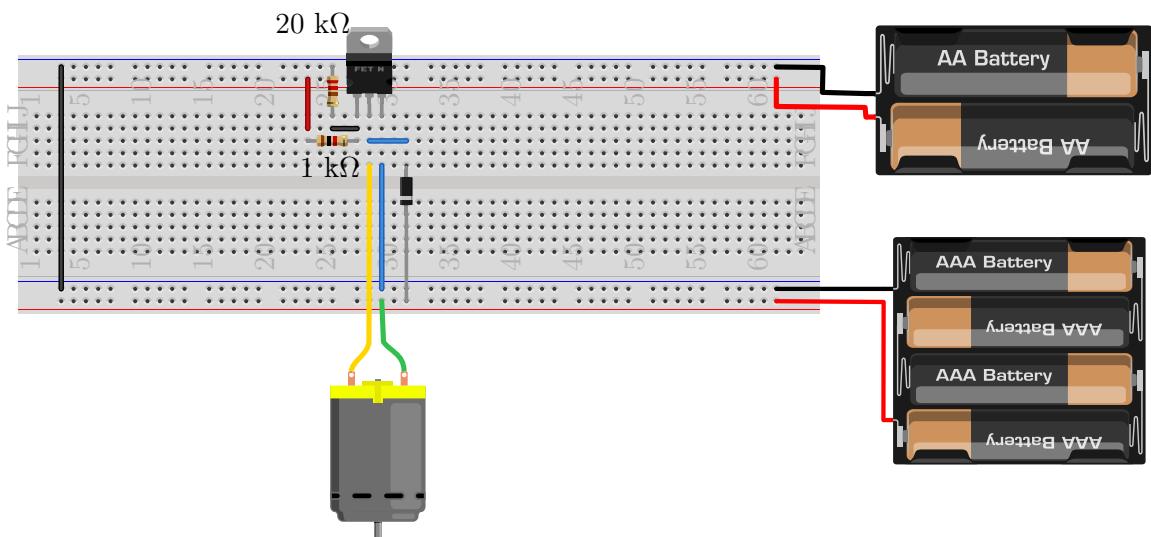


**FIGURE 1.7** – Schéma du branchement d'un moteur avec un interrupteur

## 1.7 Le transistor MOSFET

Pour ce dernier montage nous allons voir comment utiliser un transistor. Un transistor est permet de contrôler une grosse puissance avec une petite puissance. Par exemple, nous avons des piles (les 4) qui sont puissantes qui alimentent le moteur mais c'est le courant qui provient du petit bloc de piles qui commande le moteur. Concrètement, cela sert par exemple quand on veut contrôler un moteur avec une carte programmable (c'est ce principe qu'utilisent les téléphones pour actionner le vibrEUR).<sup>1</sup>

Ici, nous aurons un MOSFET N, c'est un transistor très efficace pour les grandes puissances et qui se met sur le - du moteur. La diode sert à protéger le MOSFET des pics de tension du moteur. La résistance de  $1k\Omega$  sert à protéger le micro contrôleur (ici il n'y en a pas mais il y en aura plus tard). La résistance de  $22k\Omega$  sert éteindre le moteur quand aucun signal n'est envoyé car les MOSFET sont très sensibles aux parasites.



**FIGURE 1.8** – Schéma du branchement d'un moteur avec un MOSFET

1. Les transistors sont aussi présents dans tous les processeurs : vous en avez quelques milliards dans votre téléphone ou dans votre ordinateur

# Chapitre 2

## Débuter avec Arduino

### 2.1 Présentation de la carte Arduino nano

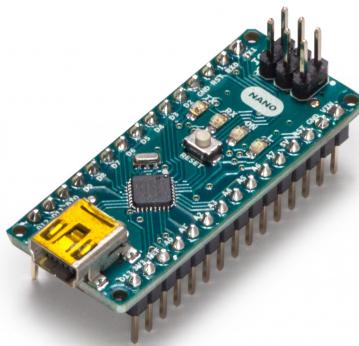


FIGURE 2.1 – Carte Arduino nano

En tapant `Arduino nano` dans un moteur de recherche sur internet, on trouve la page du site Arduino qui contient les informations de la carte en question : <https://store.arduino.cc/arduino-nano>. Voici un résumé des informations principales :

**14 pins numériques** : de 0 à 13 compris (attention, les pins 0 et 1 sont déjà utilisés par l'ordinateur pour la communication)

**8 pins analogiques** : de A0 à A7 compris

**Pins PWM** : 3, 5, 6, 9, 10, et 11

**Led intégrée** : sur le pin 13

**Bouton reset** : permet de relancer le programme en repartant au début

**Tension de fonctionnement** : 5V

**Courant max par pin** : 40 mA

**Courant max total** : 200 mA

**Un BUS I2C** : A4 (SDA) et A5 (SCL)

### 2.2 Résumé des différentes boucles, conditions et fonctions

#### 2.2.1 Créer un programme Arduino

Quand on crée un nouveau programme, on peut voir que deux fonctions sont déjà présentes :

```
void setup() {  
    // put your setup code here, to run once:
```

```

}

void loop() {
    // put your main code here, to run repeatedly:
}

```

Les instructions à l'intérieur du setup ne seront lancées qu'au début. Après le setup, les instructions à l'intérieur du loop seront exécutées en boucle.

### 2.2.2 Interagir avec les pin de l'Arduino

Pour pouvoir utiliser les pins de l'Arduino, il est nécessaire d'appeler la fonction `pinMode(pin, mode)` dans le setup. Il faut remplacer `pin` par le pin utilisé (ou la variable qui correspond au pin) et `mode` par :

**OUTPUT** : si on peut utiliser le pin comme une sortie numérique simple (par exemple pour allumer une LED)

**INPUT** : si on veut utiliser le pin comme une entrée numérique (pour lire une valeur qui sera 0 ou 1)

**INPUT\_PULLUP** : si on veut utiliser le pin comme une entrée numérique avec une résistance interne (il faut utiliser une résistance interne pour un bouton poussoir)

Une fois la déclaration du pin effectuée, on peut s'en servir avec les fonctions :

**digitalRead(pin)** : elle permet de lire l'état du pin : 0 (bas) ou 1 (haut), ce qui correspond respectivement à 0V ou 5V

**digitalWrite(pin, valeur)** : on peut mettre à l'état haut (**HIGH**) ou bas (**LOW**) le pin pour par exemple allumer une LED ou un moteur

**analogWrite(pin, valeur)** : on peut utiliser le PWM sur le pin, la valeur est comprise entre 0 et 255. Une explication du PWM est disponible après (partie 2.6).

**analogRead(pin)** : elle permet de lire la valeur analogique d'un pin (la valeur est comprise entre 0 et 1023 qui correspondent respectivement à 0V et 5V)

### 2.2.3 Les variables

Pour déclarer un entier il faut écrire `int age;`. Pour en changer la valeur on peut faire `age = 5`. On peut aussi combiner les deux pour déclarer une variable et lui affecter une valeur dès le début : `int age = 5;`.

On peut utiliser les principales opérations mathématiques avec les variables :  $+$   $-$   $\times$   $\div$ . Leurs symboles sur Arduino sont respectivement `+` `-` `*` `/`. Par exemple, si on peut ajouter 3 à la variable age on fera : `age = age + 3;`.

### 2.2.4 Les conditions

On peut tester l'égalité (`==`), l'inégalité (`!=`) ou la supériorité ou non (`>`, `<`, `>=`, `<=`). Pour utiliser une condition, on peut faire :<sup>1</sup>

```

if (age > 5) {
    // instructions si la condition est vérifiée
    // c'est à dire si age est strictement supérieur à 5
}
else //sinon
{
    //instructions si la condition n'est pas vérifiée
}

```

1. les `//` correspondent à un commentaire : tout ce qui est derrière les `//` ne sera pas lu par le programme. On peut aussi utiliser `/*` et `*/` pour faire un commentaire : ceci est du code `/*` ceci est un commentaire qui ne sera pas lu `*/` ceci est du code.

## 2.2.5 Les boucles

On utilise des boucles pour répéter une même action plusieurs fois.

**La boucle while()** Traduite en français par “tant que”, le code contenu dans la boucle est répété tant qu’une condition est respectée.

```
while(age < 50) {  
    /* contenu à répéter */  
    age = age + 4;  
}
```

**La boucle for** Elle permet de répéter un nombre de fois précis des instructions.

```
for (int i=0; i < 50; i = i + 1) {  
    // code lancé 50 fois de suite  
}
```

Trois “blocs” sont contenus dans les parenthèses du for :

- “int i = 0” : crée une variable i au début de la boucle dont la valeur est 0
- “i < 50” : c’est la condition dans laquelle on continue à lancer la boucle
- “i = i + 1” : on ajoute 1 à i à chaque fois qu’on a fait une fois les instructions

## 2.3 Allumer une LED et téléverser le programme sur la carte

Pour cet exemple, on va utiliser la LED intégrée à la carte sur la pin 13. Pour cela, un programme de test est déjà intégré au logiciel. Il se trouve dans *Fichier → Exemples → Basics → Blink*. Il suffit de choisir le type de carte (voir la figure 2.2) et de cliquer sur le bouton téléverser.

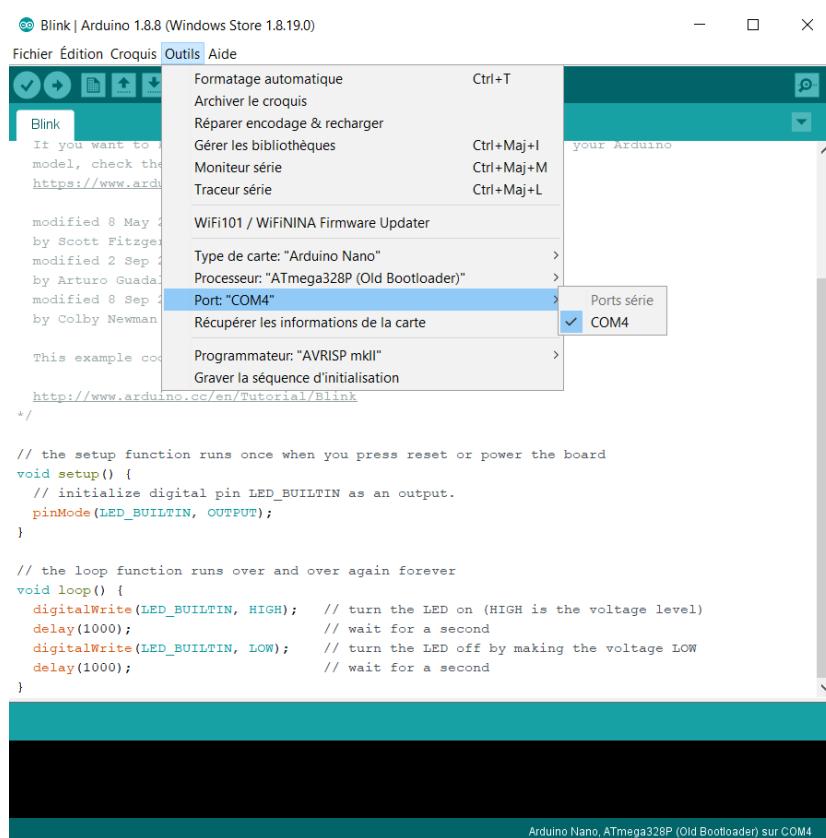


FIGURE 2.2 – Téléversement du programme sur une carte Arduino nano

## 2.4 Utiliser un bouton poussoir et une LED

Dans cet exemple, on va utiliser un bouton poussoir pour allumer une LED : quand le bouton sera appuyé, la LED sera allumée.

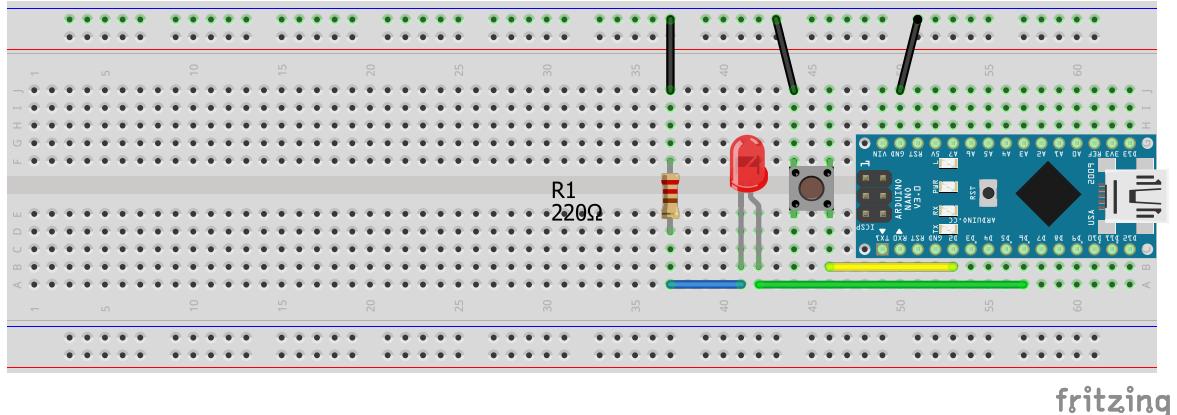


FIGURE 2.3 – Schéma du branchement d'un bouton et d'une LED sur une Arduino nano

```
int pin_bouton = 2;
int pin_led = 6;

void setup() {
    pinMode(pin_bouton, INPUT_PULLUP);
    //on declare le pin utilise par le bouton

    pinMode(pin_led, OUTPUT);
    //on declare le pin utilise par la led
}

void loop() {
    int etat_bouton = digitalRead(pin_bouton);
    //on lit l'état du bouton

    if(etat_bouton == LOW) //si le bouton est appuyé
    {
        digitalWrite(pin_led, HIGH);
        //on allume la led
    }
    else //sinon
    {
        digitalWrite(pin_led, LOW);
        //on éteint la led
    }
}
```

## 2.5 Utiliser la transmission série

La carte arduino nano offre la possibilité de retourner des informations à l'ordinateur avec une transmission série (`serial communication` en anglais). Quand on a un problème avec un programme qui ne fonctionne pas, cela permet de retourner les valeurs des variables à l'ordinateur pour les vérifier.

L'exemple en question est un code qui affiche toutes les 100 ms Hello World ! et l'état d'un bouton (0 ou 1) branché sur le pin 2.

Voici le code du programme :

```
int pin_bouton = 2;

void setup() {
    pinMode(pin_bouton, INPUT_PULLUP);

    Serial.begin(9600);
}

void loop() {
    Serial.print("Hello World !"); //On affiche Hello World !
    Serial.println(digitalRead(pin_bouton)); //On affiche la
    //valeur lu par la carte arduino sur le pin du bouton
    delay(100); //On attend 100ms avant de renvoyer l'information
}
```

## 2.6 Le PWM avec la fonction analogWrite

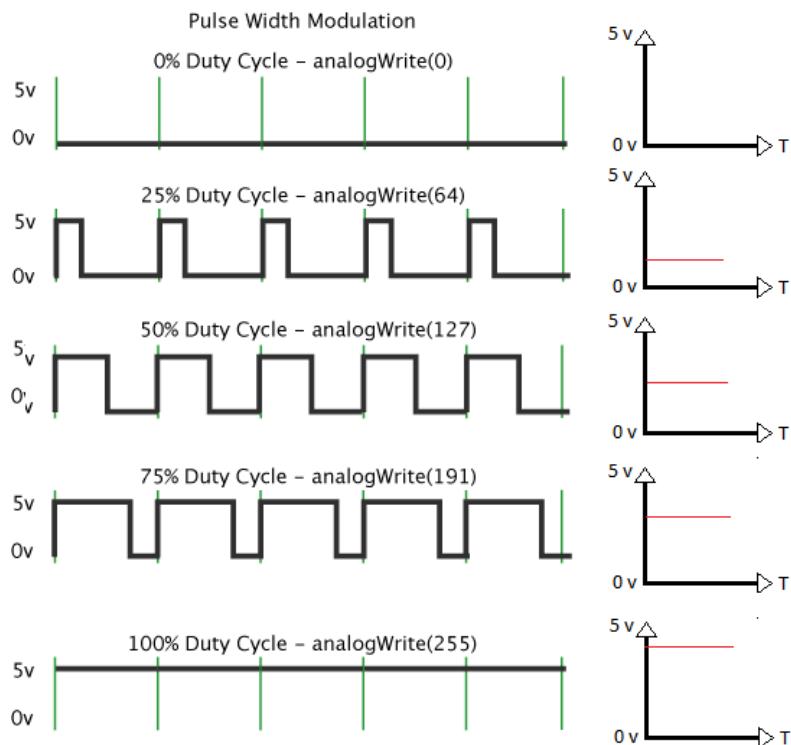


FIGURE 2.4 – Représentation de signaux PWM

PWM signifie Pulse Width Modulation ce qui, en français, donne : modulation de longueur d'impulsion. Le PWM sert à faire varier la puissance qu'on va appliquer à une LED ou un moteur. Plus le ratio (Duty cycle en anglais) augmente, plus la puissance sera élevée. On peut comparer cela à un pourcentage ou la valeur 255 correspondra à 100% de temps où la tension est à 5V. On peut en voir un représentation à la figure 2.4.

## 2.7 Utiliser un moteur

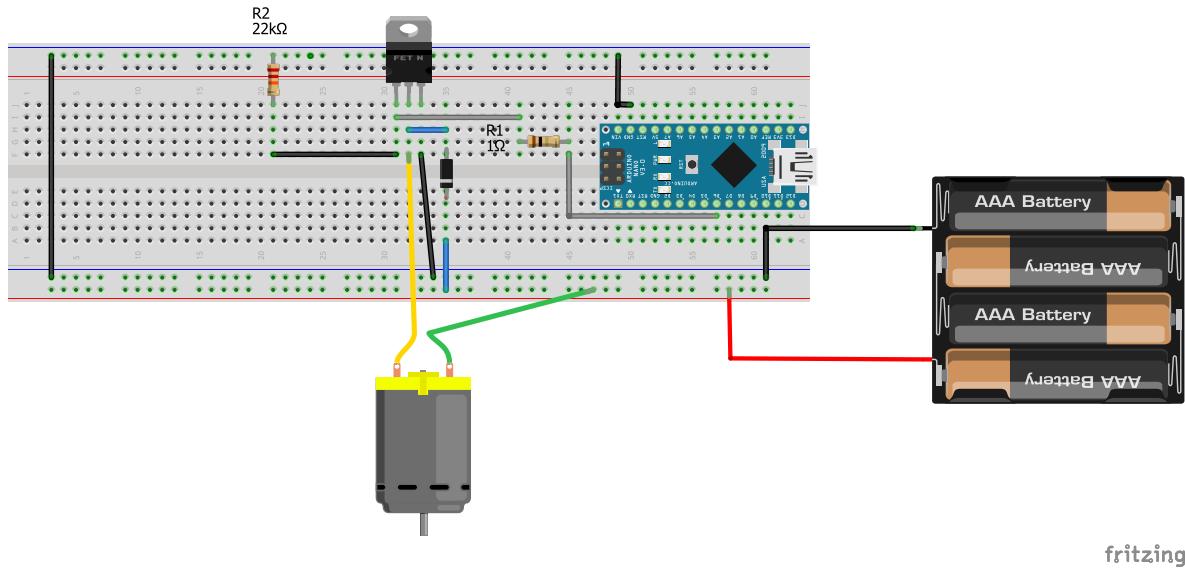


FIGURE 2.5 – Schéma du branchement d'un moteur sur une Arduino nano

```

int pin_moteur = 6;

void setup() {
    pinMode(pin_moteur, OUTPUT);
}

void loop() {
    digitalWrite(pin_moteur, HIGH);
    //on allume le moteur

    delay(2000); //On attend 2s

    digitalWrite(pin_moteur, LOW);
    //on éteint le moteur

    delay(5000); //On attend 5s
}

```

## 2.8 Utiliser un moteur en PWM

Il faut garder le même montage qui est sur la figure 2.5. Ce code augmente progressivement la vitesse du moteur puis l'éteint et attend 2s avant de recommencer :

```

int pin_moteur = 6;

void setup() {
    pinMode(pin_moteur, OUTPUT);
}

void loop() {
    //on augmente progressivement la vitesse du moteur
}

```

```

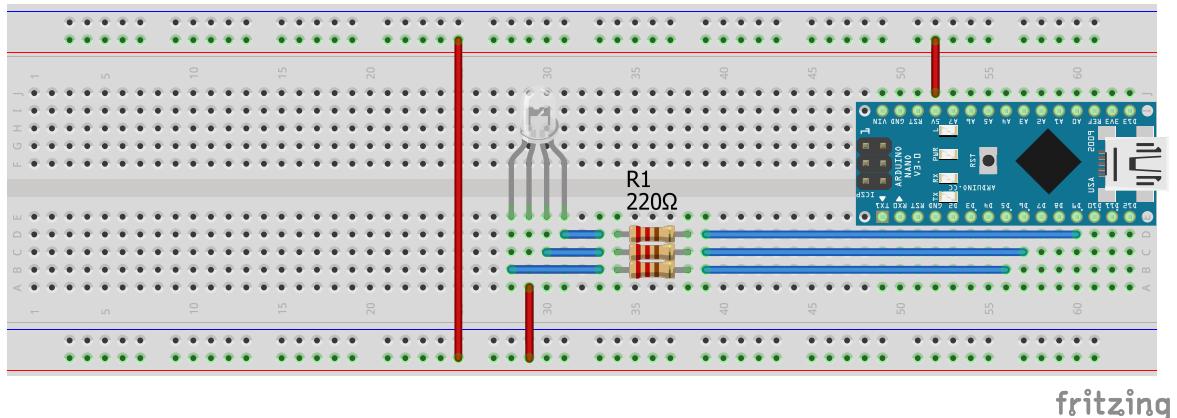
for(int i = 0 ; i < 255 ; i++)
{
    analogWrite(pin_moteur, i);
    delay(10);
}

analogWrite(pin_moteur, 0);
delay(2000); //On attend 2s avant de recommencer
}

```

## 2.9 La LED RGB

Les LED RGB servent à pouvoir afficher toutes les couleurs visibles par un être humain. Vous vous en êtes déjà servis dans une précédente manipulation mais vous allez découvrir ici une manière bien plus pratique de s'en servir ! Pensez à ajuster les variables des pins



**FIGURE 2.6 – Schéma du branchement d'une LED RGB sur une Arduino nano**

pour les faire correspondre à la bonne couleur. Pour cela, vous pouvez tester chacune des couleurs indépendamment avec ce code :

```

int pin_led = 5;

void setup() {
    pinMode(pin_led, OUTPUT);
}

void loop() {
    digitalWrite(pin_led, LOW);
    //on allume la led sur le pin pin_led
}

```

Ce code va allumer uniquement la couleur qui est branchée sur le pin `pin_led`.

Ensuite, vous pouvez vous amuser à faire apparaître différentes couleurs ! Voici un code exemple :

```

int pin_r = 5;
int pin_g = 6;
int pin_b = 9;

void setup() {

```

```

pinMode(pin_r, OUTPUT);
pinMode(pin_g, OUTPUT);
pinMode(pin_b, OUTPUT);
}

void loop() {
    analogWrite(pin_r, 128);
    analogWrite(pin_g, 0);
    analogWrite(pin_b, 255);

    delay(1000);

    analogWrite(pin_r, 255);
    analogWrite(pin_g, 255);
    analogWrite(pin_b, 0);

    delay(1000);
}

```

N'hésitez pas à ajouter des boucles, changer les delay() ou changer les valeurs du PWM (elles doivent être comprises entre 0 et 255).

*Bonus :* Un exemple de code pour faire changer les couleurs de manière fluide :

```

for(int i = 0 ; i < 255 ; i++)
{
    digitalWrite(pin_r, i);
    delay(10);
}

```

## 2.10 Le sonar HC-SR04

Le sonar HC-SR04 permet de mesurer une distance. Pour cela le capteur va envoyer des ultrasons et mesurer le temps qu'il mettent pour revenir. Avec ce temps, le capteur en déduit la distance de l'obstacle. Pour se servir de ce capteur, on peut utiliser une bibliothèque, cela simplifiera le code :

```

#include "NewPing.h"

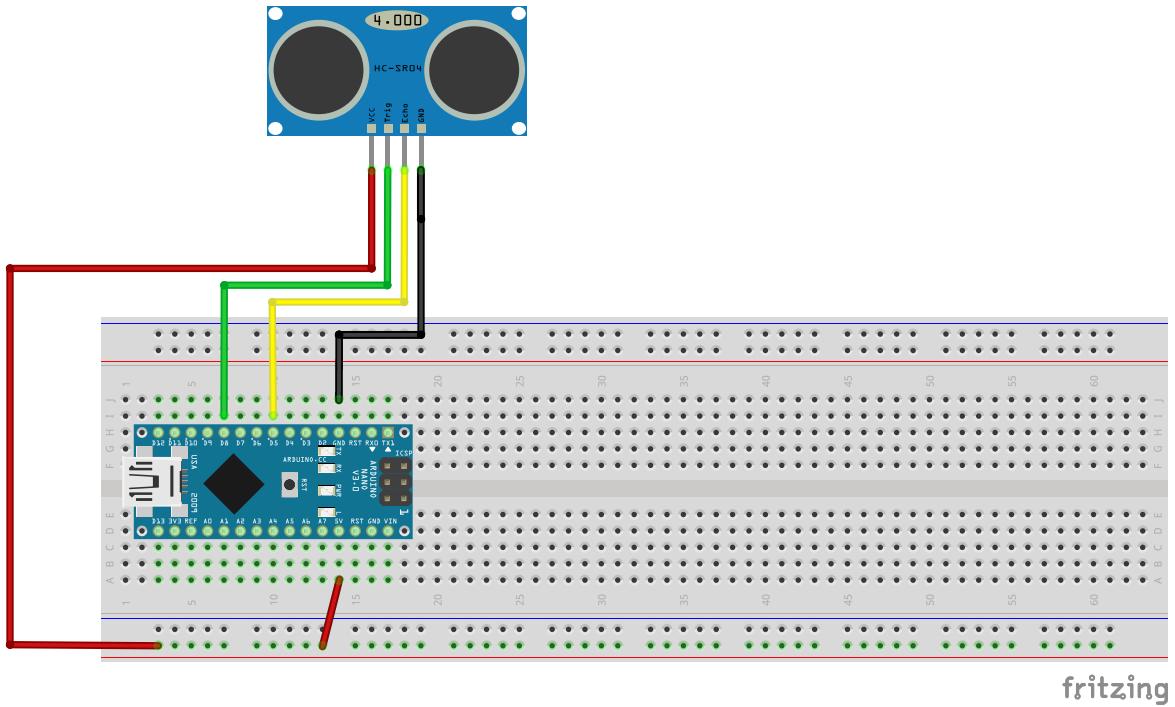
int trigger_pin 4
int echo_pin      2
int max_distance 200

NewPing sonar(trigger_pin, echo_pin, max_distance);

void setup() {
    Serial.begin(57600);
}

void loop() {
    delay(50);
    Serial.print("Ping: ");
    Serial.print(sonar.ping_cm());
    Serial.println("cm");
}

```



**FIGURE 2.7 – Schéma du branchement du sonar sur une Arduino nano**

## 2.11 Le pont en H L293D

Le L293D est un pont en H<sup>2</sup> capable de piloter deux moteurs en même temps. A titre d'exemple, le montage figure 2.9 présente les branchements à effectuer pour l'utiliser avec deux moteurs.

```
#include "Motor.h"

int pin_en = 11;
int pin_A = 5;
int pin_B = 6;
//declaration d'un moteur appele leftMotor :
Motor leftMotor(pin_en, pin_A, pin_B, 1);

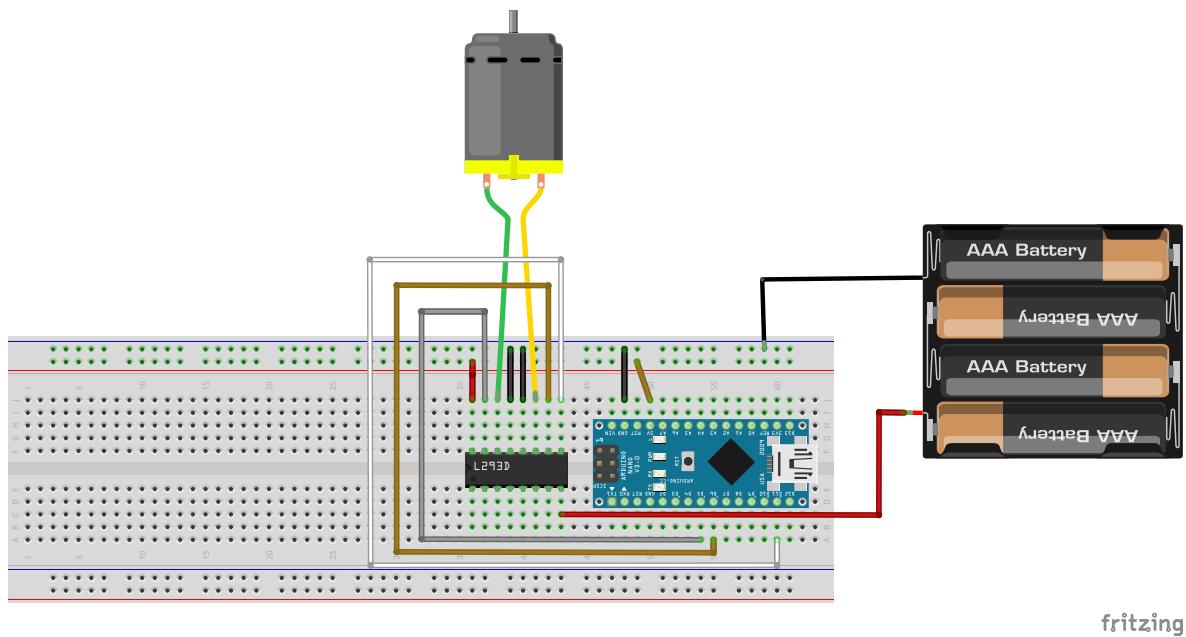
void setup() {

}

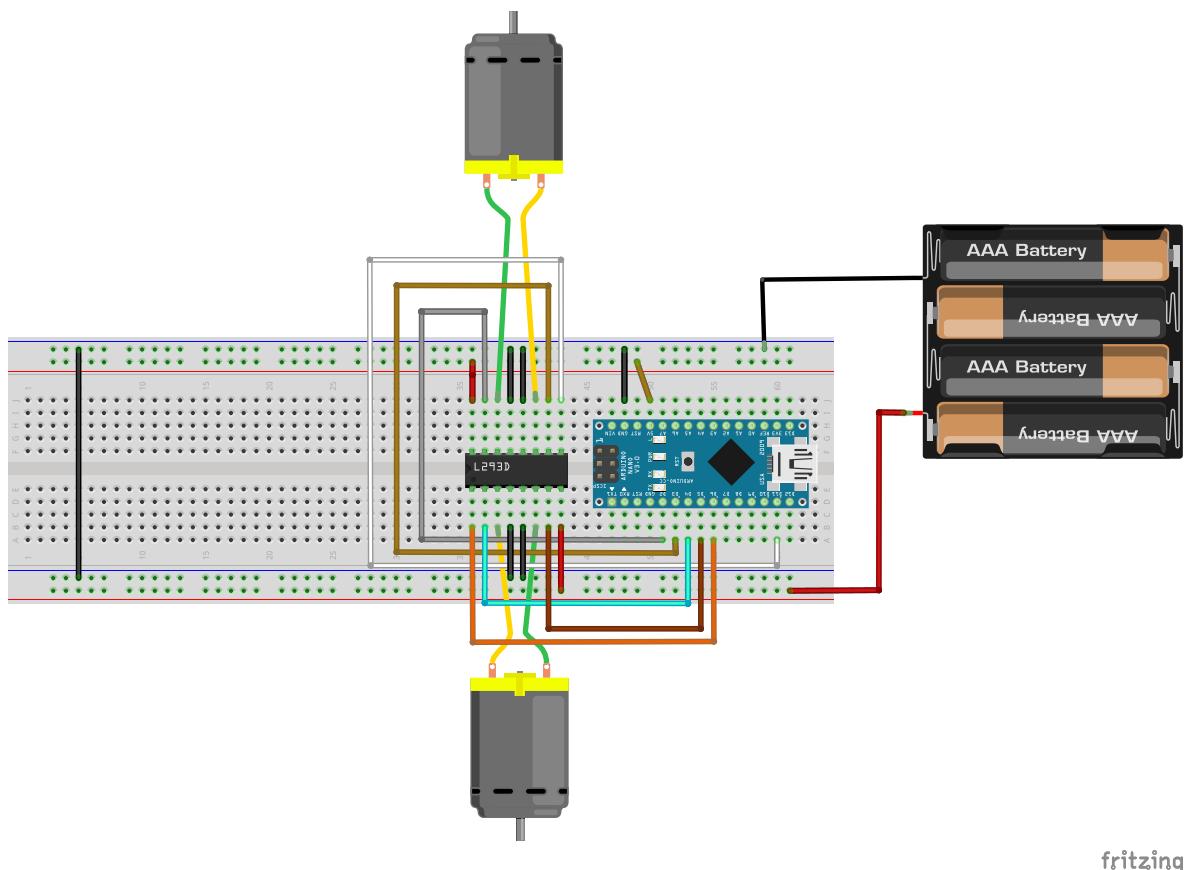
void loop() {
    leftMotor.setSpeed(170); //vitesse de -255 a 255
    delay(2000);
    leftMotor.setSpeed(-170);
    delay(2000);
}
```

Il faudra adapter le code précédent pour faire fonctionner le montage. Il faudra ajouter une ligne pour déclarer un nouveau moteur au début du programme : `Motor leftMotor(6, 4, 5, 1);` (vous pouvez mettre des variables pour les pins comme dans le premier exemple).

2. Un pont en H est un montage électronique qui sert notamment à piloter un moteur dans les deux sens (contrairement au transistor). Ici, le montage est présent sous la forme d'un circuit intégré. Le modèle de circuit intégré utilisé est un L293D



**FIGURE 2.8 – Schéma du branchement du pont en H pour un moteur sur une Arduino nano**



**FIGURE 2.9 – Schéma du branchement du pont en H pour deux moteur sur une Arduino nano**

## Chapitre 3

# Utiliser des bibliothèques

Le dossier `motor` contient la bibliothèque `Motor` pour Arduino permettant de contrôler des moteurs avec un pont en H.

Pour utiliser les bibliothèques, vous devez les avoir installé dans le logiciel que vous utilisez (Arduino IDE ou PlatformIO). Vous pouvez voir comment faire dans l'annexe A page 17. Si vous cherchez une bibliothèque particulière, n'hésitez à rechercher sur internet !

Si vous voulez aller plus loin que l'exemple ci-dessous, vous pouvez aller regarder le code source de la bibliothèque. Par exemple pour la bibliothèque `Motor`, le code se situe dans le dossier `motor`.

Une fois que vous êtes dans le dossier qui correspond à la bibliothèque que vous voulez regarder de plus près, deux fichiers apparaissent : un finissant par `.h` et un `.cpp`. Le fichier finissant par `.h` contient le descriptif de la bibliothèque : vous y trouverez toutes les fonctions disponibles ainsi que des commentaires afin d'en comprendre l'usage. Le fichier se terminant par `.cpp` contient le code de toutes les fonctions. Vous y trouverez ce que fait en détail chaque fonction. En général, la lecture du fichier se finissant par `.h` suffit pour réussir à utiliser la bibliothèque. N'hésitez pas à aller voir ces fichiers, vous y trouverez sûrement des informations utiles.

### 3.1 La bibliothèque Motor

Cette bibliothèque contient toutes les fonctions permettant de contrôler les moteurs du robot. Exemple d'utilisation pour contrôler un moteur :

```

#include <Motor.h> //ajout de la bibliotheque Motor au code principal

#define PIN_MOTEUR_D_1      7
#define PIN_MOTEUR_D_2      A2
#define PIN_MOTEUR_D_EN     6

Motor moteurDroit(PIN_MOTEUR_EN, PIN_MOTEUR_1, PIN_MOTEUR_2, 1);
//declaration d'un objet de type Motor nomme moteurDroit

void setup()
{
}

void loop()
{
    moteurDroit.setSpeed(255); //On met la vitesse du moteur a 255
    // (la vitesse est comprise entre -255 et 255)
    delay(5000); //on attend 5 secondes
    moteurDroit.setSpeed(0); //On met la vitesse du moteur a 0 = arret
    //du moteur
    delay(5000);
}

```

## Annexe A

# Les logiciels pour Arduino

### A.1 Les logiciels à installer

L'environnement de développement le plus simple à utiliser est le logiciel créé par Arduino : Arduino IDE. Tout de fois, il reste assez limité quand on se lance dans de plus gros projets. Certains logiciels on par exemple l'auto-complétion, ce qui rend plus agréable la saisie. D'autres permettent d'être plus souple au niveau de la gestion des bibliothèques.

Si vous souhaitez vous passer du logiciel Arduino IDE, deux solutions on retenue mon attention. Pour information, j'utilise comme système d'exploitation Linux (Ubuntu 18.04) ce qui peut influer sur mon expérience.

Une possibilité est d'utiliser l'éditeur de texte Atom avec le package PlatformIO.

Une autre solution est de se servir d'un éditeur de texte (n'importe lequel) et d'utiliser un Makefile pour compiler le programme et l'envoyer sur la carte Arduino. J'aurai tendance à vous déconseiller cette méthode si vous n'êtes pas sous Linux, cela rendra la configuration plus complexe. Personnellement, j'utilise celui la <https://github.com/sudar/Arduino-Makefile>. C'est un bonne chose d'ajouter une extension dans l'éditeur de texte pour l'auto-complétion.

J'ai utilisé ces logiciels pendant plusieurs mois et la solution que je préfère est l'éditeur Atom avec PlatformIO

Mes recommandations sont basées sur les expériences que j'ai eu avec ces solutions et sont donc totalement arbitraires, libre à vous de choisir !

*NB :* Sous Windows, si la carte Arduino nano n'est pas reconnue, il faut installer le pilote pour le CH340 (un composant de la carte). Vous pouvez le télécharger ici : <https://sparks.gogo.co.nz/ch340.html> (Download the Windows CH340 Driver).

### A.2 Ajouter une bibliothèque avec Arduino IDE

Avec le logiciel Arduino, pour ajouter une bibliothèque (library en anglais), seulement deux possibilités sont offertes : on peut ajouter la bibliothèque pour tout les projets Arduino sur l'ordinateur ou uniquement pour un seul projet.

#### A.2.1 Ajouter une bibliothèque pour tout les projets

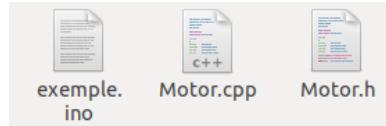
La méthode la plus simple pour cela est d'utiliser l'interface graphique du logiciel Arduino IDE : Aller dans Croquis puis Inclure une bibliothèque et Ajouter la bibliothèque .ZIP.... Il ne reste plus qu'à choisir le fichier .zip contenant la bibliothèque et valider.

L'autre méthode consiste à copier le dossier contenant la bibliothèque dans le dossier libraries d'Arduino (souvent situé dans Documents/Arduino)

Pour utiliser cette bibliothèque, il faudra écrire au début du code #include <Motor.h>. Les < > signifient que c'est une bibliothèque disponible pour tout les projets.

### A.2.2 Ajouter une bibliothèque pour un seul projet

Pour cela il faut placer les fichiers de la bibliothèque dans le même dossier que le code Arduino du projet. Voir l'exemple figure A.1.



**FIGURE A.1** – Exemple pour la bibliothèque Motor

### A.3 Ajouter une bibliothèque avec PlatformIO

Vous pouvez déposer le dossier contenant la bibliothèque n'importe où tant que l'emplacement est défini dans le fichier `platformio.ini`.

Un emplacement déjà défini dans le logiciel permet de juste mettre le dossier contenant la bibliothèque à cet endroit. Quand un nouveau projet est créé, pour l'utiliser cet emplacement, il faut mettre le dossier contenant la bibliothèque dans `nom_du_projet/lib`. Vous pouvez voir comment tout cela est organisé dans l'exemple sur GitHub.

## Annexe B

# Électronique et programmation

### B.1 Le code couleur des résistances

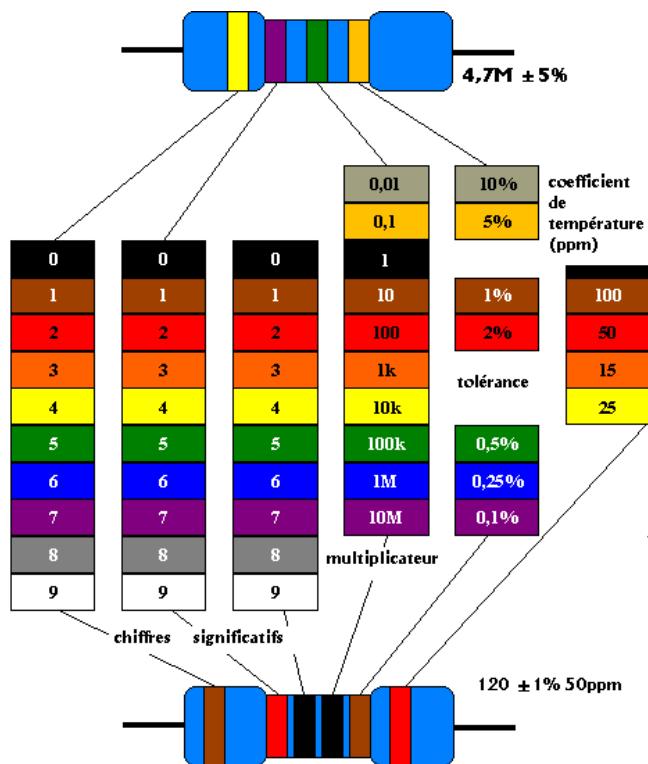


FIGURE B.1 – Signification du code couleur des résistances

## B.2 Les types de variables

Type	Taille	Valeur	A utiliser pour
byte	1 octet	0 à 255	un entier naturel inférieur ou égal à 255
char	1 octet	un caractère ASCII	un caractère sans accent
double	8 octets	$-2^{1023}$ à $2^{1023}$	un nombre décimal d'une grande précision <sup>1</sup>
float	4 octets	$-3.4 \cdot 10^{38}$ à $3.4 \cdot 10^{38}$	un nombre décimal
int	2 octets	$-32\,768$ à $32\,767$	un nombre entier relatif
long	4 octets	$-2.15 \cdot 10^9$ à $2.15 \cdot 10^9$	un grand nombre entier relatif

**FIGURE B.2** – Principaux types de variables

On peut aussi ajouter le paramètre `unsigned` ou `signed` devant le type de la variable. Par exemple, `unsigned long` sera un entier naturel compris entre 0 et 4 294 967 296 (Avec la même place en mémoire, on peut avoir un nombre plus grand mais qui ne sera pas négatif).

On utilise souvent une autre notation, qui permet d'être plus clair sur le type de la variable. Cela s'applique aux nombres entiers, par exemple écrire `int16_t` revient à écrire `int`. Cela peut paraître inutile mais dans certains codes, cela est très pratique car il suffit juste de connaître cette notation pour déclarer tout type d'entiers.

Type	Taille	Valeur (contient au moins)
<code>int8_t</code>	1 octet	$-127$ à $127$
<code>uint8_t</code>	1 octet	0 à 255
<code>int16_t</code>	2 octets	$-32\,767$ à $32\,767$
<code>uint16_t</code>	2 octets	0 à 65 535
<code>int32_t</code>	4 octets	$-2\,147\,483\,647$ à $2\,147\,483\,647$
<code>uint32_t</code>	4 octets	0 à 4 294 967 295

**FIGURE B.3** – Principaux types de variables (notation alternative)

---

1. Sur les carte basées sur des ATMEGA (Arduino uno, nano, mega...), l'implémentation sera la même que pour les variables float

# Table des figures

1.1	Photo d'une breadboard . . . . .	2
1.2	Schéma des liaisons électriques dans la breadboard . . . . .	2
1.3	Schéma du branchement d'une LED avec une résistance . . . . .	2
1.4	Schéma du branchement d'une LED avec un interrupteur . . . . .	3
1.5	Schéma du branchement d'une LED avec une résistance et une diode . . . . .	3
1.6	Schéma du branchement d'une LED RGB . . . . .	3
1.7	Schéma du branchement d'un moteur avec un interrupteur . . . . .	4
1.8	Schéma du branchement d'un moteur avec un MOSFET . . . . .	4
2.1	Carte Arduino nano . . . . .	5
2.2	Téléversement du programme sur une carte Arduino nano . . . . .	7
2.3	Schéma du branchement d'un bouton et d'une LED sur une Arduino nano .	8
2.4	Représentation de signaux PWM . . . . .	9
2.5	Schéma du branchement d'un moteur sur une Arduino nano . . . . .	10
2.6	Schéma du branchement d'une LED RGB sur une Arduino nano . . . . .	11
2.7	Schéma du branchement du sonar sur une Arduino nano . . . . .	13
2.8	Schéma du branchement du pont en H pour un moteur sur une Arduino nano .	14
2.9	Schéma du branchement du pont en H pour deux moteur sur une Arduino nano . . . . .	14
A.1	Exemple pour la bibliothèque Motor . . . . .	18
B.1	Signification du code couleur des résistances . . . . .	19
B.2	Principaux types de variables . . . . .	20
B.3	Principaux types de variables (notation alternative) . . . . .	20

Merci à Augustin PEDO, Clément-Joseph GALTEAU, Clément de LA BOURDON-NAYE et Anthony PACITTO pour avoir réalisé les schémas électroniques et les sections 2.2.3, 2.2.4 et 2.2.5.

Si vous avez des remarques à propos de ce document, vous pouvez me contacter à l'adresse mail suivante : roman.thomas1@utt.fr.