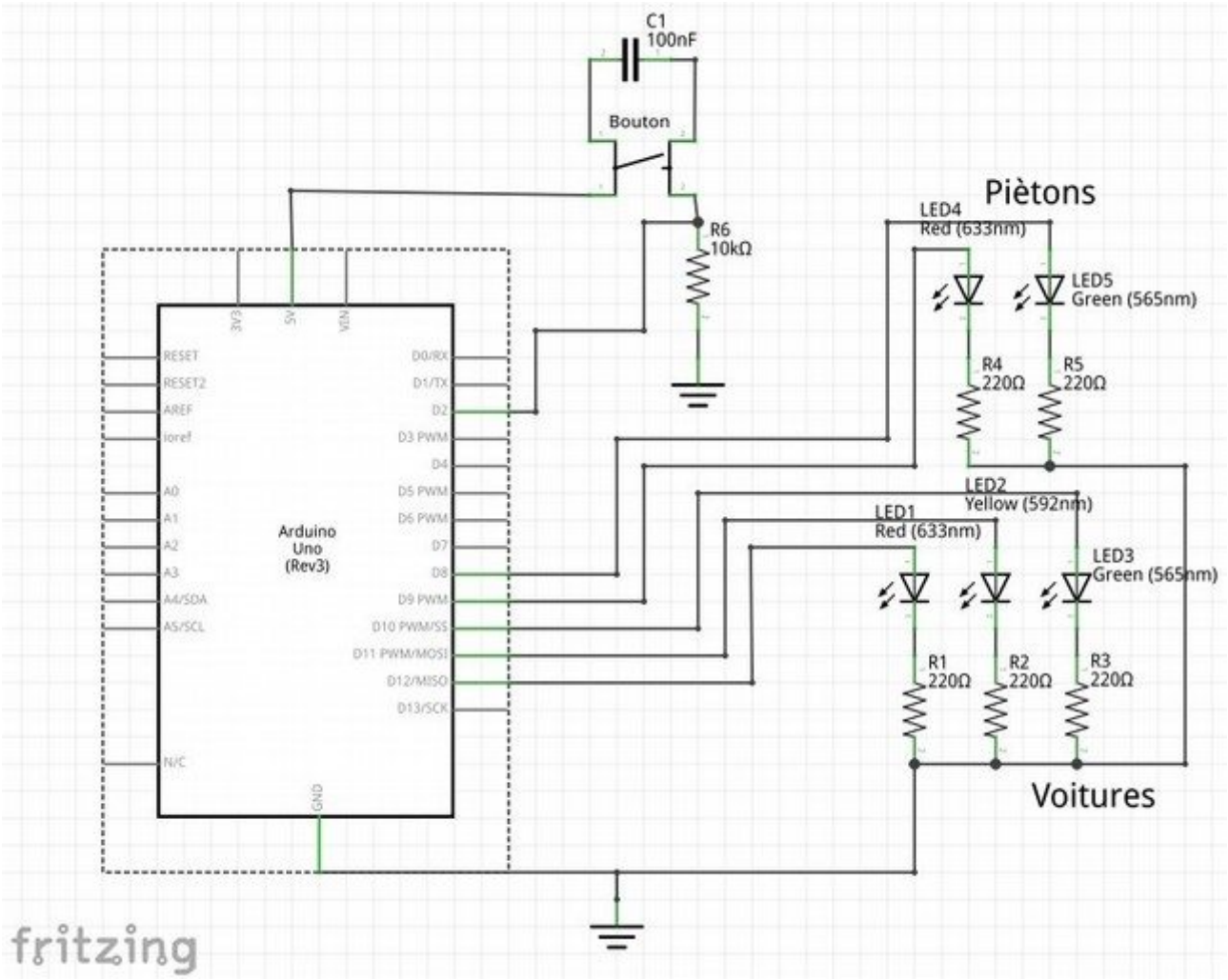
Compte rendu du Lundi 18/04/2016à la MJC
Groupe- »Fablab découverte du numérique

Nous étions 7 à nous retrouver ce soir. Une nouvelle personne qui nous a rejoint

Paul Benoit
il a 15 ans , il est intéressé par l'Arduino et la robotique

1 - Nous avons réexpliqué le schéma du Bouton poussoir piéton et des LEDs feux voiture



Pour éviter les entrées flottante, la résistance et branchée en Pull-Down , on envoie un moins(-) permanent sur l'entrée , quand on appuie sur le bouton on envoie un plus (+)

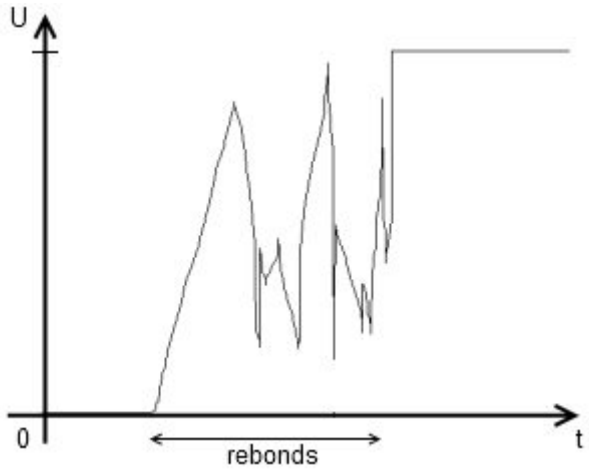
Le condensateur en // sur le bouton poussoir permet d 'éviter les rebonds du bouton.

Un peu de théorie sur le rebond :

Filtrer les rebonds

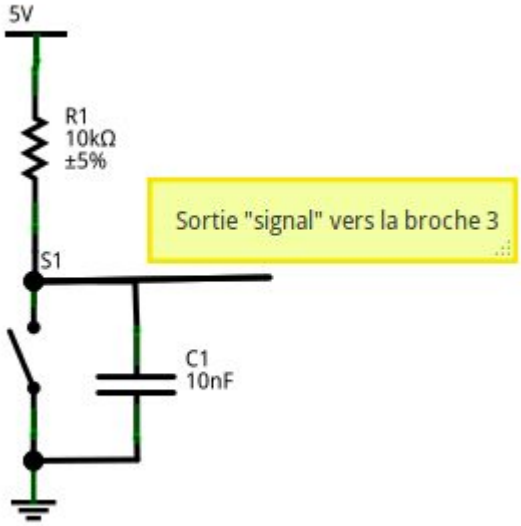
Les boutons ne sont pas des systèmes mécaniques parfaits. Du coup, lorsqu'un appui est fait dessus, le signal ne passe pas immédiatement et proprement de 5V à 0V. En l'espace de quelques millisecondes, le signal va « sauter » entre 5V et 0V plusieurs fois avant de se stabiliser. Il se passe le même phénomène lorsque l'utilisateur relâche le bouton. Ce genre d'effet n'est pas désirable,

car il peut engendrer des parasites au sein de votre programme (si vous voulez détecter un appui, les rebonds vont vous en générer une dizaine en quelques millisecondes, ce qui peut-être très gênant dans le cas d'un compteur par exemple). Voilà un exemple de chronogramme relevé lors du relâchement d'un bouton poussoir :



Pour atténuer ce phénomène, nous allons utiliser un condensateur en parallèle avec le bouton. Ce composant servira ici « d'amortisseur » qui absorbera les rebonds (comme sur une voiture avec les cahots de la route). Le condensateur, initialement chargé, va se décharger lors de l'appui sur le bouton. S'il y a des rebonds, ils seront encaissés par le condensateur durant cette décharge. Il se passera le phénomène inverse (charge du condensateur) lors du relâchement du bouton. Ce principe est illustré à la figure suivante :

On utilise une résistance en pull-up (on insère un +5v permanent sur l'entrée), quand on appuie sur le bouton on envoie un moins (-)



2- En prenant exemple sur le programme des feux voiture nous avons détaillé la structure d'un programme Arduino

En 1er au début de programme (en haut)

Nous déclarons les variables du programme exemple :

```
const int Feu_Vert_Voiture = 2;  // « const »  la variable ne change pas en cours de programme
const int Feu_Orange_Voiture = 3;
 const int Feu_Rouge_Voiture = 4;
```

Les **variables** déclarées ici sont **globales**, c'est à dire accessible dans toutes les fonctions du programmes

ou

```
#define  Feu_Vert_Voiture  2     // sans signe = et sans ;
```

on peut y voir aussi la déclaration de librairies avec #include

```
#include <LiquidCrystal.h> // une librairie pour contrôler un afficheur LCD
```

on verra plus tard à quoi peuvent servir les librairies (il y en a beaucoup …)

En deuxième on à la fonction Setup

```
void setup()
{
Serial.begin(9600) ;
 pinMode(2, OUTPUT);

}
```

La fonction setup() est appelée au démarrage du programme. Cette fonction est utilisée pour initialiser les variables, le sens des broches, les librairies utilisées. La fonction setup n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino.
La fonction **setup()** , même vide, est obligatoire dans tout programme Arduino

En troisieme la Fonction Loop

```
void loop()
{
 if (!digitalRead(7))

  {
  digitalWrite(6, HIGH); // on allume le feux Rouge pieton
  digitalWrite(2, HIGH); // on allume le feux Vert voiture
 …// etc …….
  else
 {
  digitalWrite(3,HIGH);// on allume le feux orange voiture
  delay(1000);// on attend 1s
…//etc …….
  }
}
```

Après avoir créé une fonction **setup()**, qui initialise et fixe les valeurs de démarrage du programme, la fonction **loop ()** (boucle en anglais) fait exactement ce que son nom suggère et s'exécute en boucle sans fin, permettant à votre programme de s'exécuter et de répondre. Utiliser cette fonction pour contrôler activement la carte Arduino.
La fonction loop() est obligatoire, même vide, dans tout programme.

En quatrième on peut déclarer d'autres fonctions ou sous programmes qui servent à alléger la compréhension du programme, le rendre plus clair. En Basic on utilisait « Gosub ».

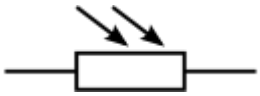
Voir référence Arduino en Français

En résumé :
1- Au début du programme , on initialise les variables globales
2- Dans la fonction Setup, on initialise les broches, la vitesse de l'interface série, les librairies
3- Dans la fonction loop, on réalise le programme principal

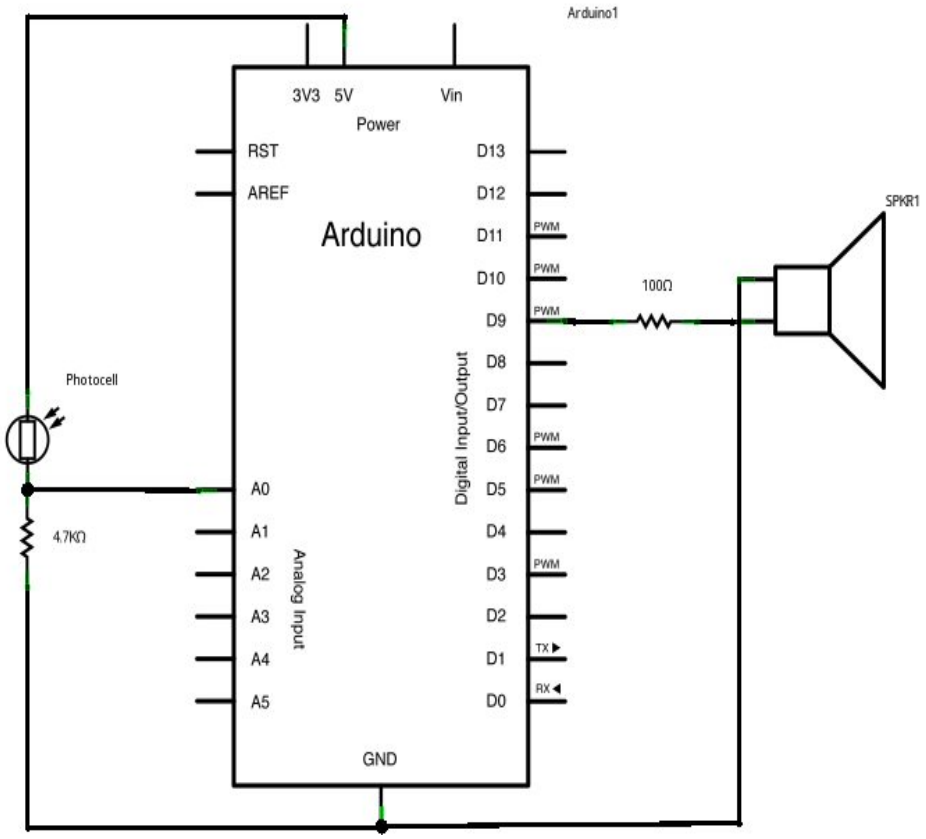
3- on a ensuite étudié le schéma proposé avec la Photorésistance (LDR - Light Dependent Resistor) .ou *cellule photoconductrice*.

Une **photorésistance** est un composant électronique dont la résistivité varie en fonction de la quantité de lumière

Symbole



Voici le schéma de raccordement et voir le Compte rendu du 11042016



4- on étudié la fonction map

Syntaxe

map (valeur lue , limite_basse_source, limite_haute_source, limite_basse_destination, limite_haute_destination)

Exemple :
valeur_sortie = map(Val_A0, 0, 1023, 50, 30000) ;

Paramètres

- **VAL_A0** : le nombre à ré-étalonner (
- **0**: la valeur de la limite inférieure de la fourchette de départ
- **1023**: la valeur de la limite supérieure de la fourchette de départ
- **50**: la valeur de la limite inférieure de la fourchette de destination
- **30000**: la valeur de la limite supérieure de la fourchette de destination

0 à 1023 sont les valeurs possibles pour une entrée analogique Arduino .
Codage en 10 bits (2 puissance10 = 1024 valeurs)

50 à 30000 Hz sont les valeurs possibles pour faire fonctionner un Buzzer

Description

Ré-étalonne un nombre d'une fourchette de valeur vers une autre fourchette.

Cette fonction est très utile pour effectuer des changements d'échelle automatiques.

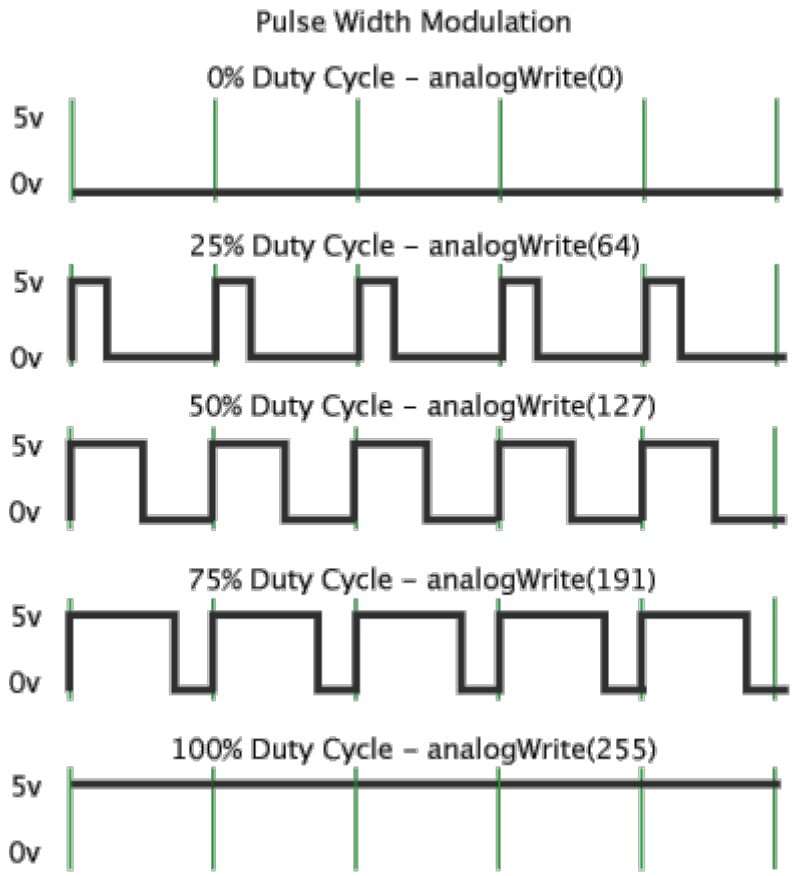
Ainsi, une valeur basse source sera étalonnée en une valeur basse de destination, une valeur haute source sera étalonnée en une valeur haute de destination, une valeur entre les deux valeurs source sera étalonnée en une valeur entre les deux valeurs destinations, en respectant la proportionnalité.

5- et enfin on a essayer de comprendre ce que faisaient les sorties digital PWM

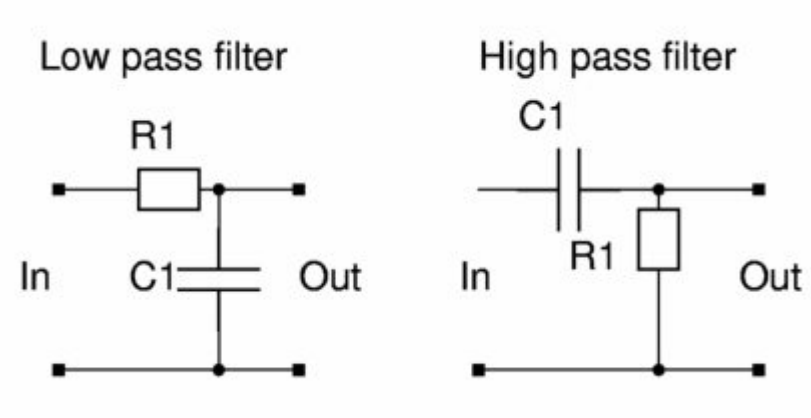
Sur votre carte Arduino, vous devriez disposer de 6 broches qui soient compatibles avec la génération d'une PWM. Elles sont repérées par le symbole tilde ~ . Voici les broches générant une PWM : 3, 5, 6, 9, 10 et 11.

dans le schéma ci-dessous on peut voir différents cycle sur une tension de 5Volts , si l'on envoie un 1 en sortie plus ou moins longtemps on aura une tension moyenne (théorique) se rapprochant de 5V

La commande analogWrite(x) permet de gérer. la sortie PWM , x prenant la valeur de 0 à 255
Avec la valeur x= 0 , on à une tension de 0 volt en sortie , avec x = 255 on a une tension de 5volts
X prenant les valeurs de 1 à 254 on à les valeurs intermédiaire entre 0V et 5v.
Si l'on voulait vraiment avoir une tension analogique qui varie de 0v à 5v il faudrait ajouter un circuit Résistance et Condensateur, circuit RC pour « lisser la tension » (on le verra une autre fois si cela vous intéresse)



Le circuit RC fait appel à des notions d'électronique que l'on verra plus tard



Pour ceux que cela interresse toujours …..:) On continue samedi sur l'exercice proposé :

L'exercice consiste à modifier l'exemple *AnalogInOutSerial* et l'adapter pour que la luminosité mesurée par la photorésistance (ou potentiomètre) soit proportionnelle à la fréquence jouée un buzzer branché sur la broche 6 : plus la luminosité reçue par la photorésistance est importante, plus la fréquence jouée par le buzzer sera élevée (et inversement, si on passe notre main devant le capteur de luminosité alors le son émis sera grave→fréquence basse).

Voir le schéma sur la photorésistance et le Buzzer plus haut

Quelques indices

- Vous aurez besoin de la fonction `tone` qui prend deux paramètres :
  - **la broche** où le buzzer est branché
  - **la fréquence** que l'on veut jouer avec le buzzer (exprimée en Hertz)

```
tone(8, 440);  // la note La (440Hz) jouée sur un buzzer branché sur la broche 8
```

- Lorsque l'on reçoit beaucoup de lumière sur notre photo-résistance, la fréquence générée devra être proche de 30 000Hz et lorsque la lumière est faible, la fréquence devra être proche de 50Hz.
- *Le buzzer doit avoir une patte connectée à la broche 6 et l'autre au GND*

Pour samedi prochain nous avions réservé la MJC de 14h à 18h .

Pourriez vous me dire si vous compter venir samedi 23 avril de 14h à 18h

Par courriel, tel (02 47 55 00 82) ou en répondant sur le site

Lien framadate : https://framadate.org/JQGzQ2izrchXnoYq

Je vous souhaite une bonne semaine

Gerard Laumonier