

TP2 – Jouons un peu avec Arduino

1. Lancement de l'environnement

- Lancer un terminal (une console).
- Tapez : `/home/shared/arduino/arduino-1.6.0/arduino`
et l'environnement se lancera.

Si vous ne souhaitez pas retaper cette commande à chaque fois, vous pouvez faire un lien symbolique depuis votre *home*, (faire `cd ~` dans le terminal) vers cette commande avec :

```
ln -s /home/shared/arduino/arduino-1.6.0/arduino arduino
```

et par la suite, depuis votre *home*, il suffira de taper `./arduino`

Et voilà !

LA PROGRAMMATION ARDUINO EN BREF

Arduino est programmé en langage C. Ceci est un petit cours à destination des personnes ayant déjà une petite expérience de la programmation et qui ont besoin d'un petit briefing sur la syntaxe du C et sur l'environnement de développement (IDE) Arduino.

STRUCTURE

Chaque programme Arduino (que l'on nomme aussi sketch) requiert 2 fonctions (aussi appelées procédures).

```
void setup(){ }
```

Tout le code entre les deux accolades sera exécuté une fois au démarrage du programme sur votre Arduino.

```
void loop(){ }
```

Cette fonction s'exécute après la fin du setup. Après une première exécution, il est ré-exécuté puis encore et encore jusqu'à ce que l'alimentation soit coupée.

SYNTAXE

L'un des éléments qui peut paraître frustrant en C est son formatage nécessaire (cela le rend aussi très puissant). Si vous vous souvenez de ce qui suit vous devriez vous en sortir.

```
// (commente une seule ligne)
```

Il est souvent très utile d'écrire des notes pour soi-même au fur et à mesure de l'élaboration de votre code. Pour ce faire inscrivez deux slashes et tout ce que vous inscrirez après sera ignoré par votre programme.

```
/* */ (commente plusieurs lignes)
```

Si vous avez beaucoup à écrire, vous pouvez commenter plusieurs lignes. Tout ce qui se trouve entre ces deux symboles sera ignoré par le programme.

```
{ } (accolades)
```

On les utilise pour définir où commence et où se termine un bloc de code (utilisé dans les fonctions aussi bien que dans les boucles).

```
;
```

Chaque ligne doit se terminer par un point-virgule (l'oubli d'un point-virgule est souvent la raison d'un refus du programme de compiler).

VARIABLES

Un programme n'est rien de plus que des instructions qui manipulent des nombres de façon intelligente. Les variables sont utilisées pour stocker les valeurs.

int (entier)

Le type le plus utilisé, il contient un nombre sur deux octets (16 bits). Il n'a pas de virgule et stocke des valeurs comprises entre -32 768 et 32 767.

long (long)

Variables utilisées lorsqu'un entier n'est pas assez grand. Ils prennent 4 octets (32 bits) de RAM et prennent des valeurs comprises entre -2 147 483 648 et 2 147 483 647.

boolean (booléen)

Variables qui prennent simplement les valeurs Vrai ou Faux (True et False). Très utiles, car elles ne prennent que peu de place (8 bits).

float (flottant)

Utilisés pour les variables flottantes. Ils prennent 4 octets (32 bits) de RAM et sont compris entre -3.4028235E+38 et 3.4028235E+38.

char (caractère)

Stocke un caractère en utilisant la table ASCII (exemple 'A' = 65). Utilise un octet (8 bits) de RAM. Arduino stocke les chaînes en tant que tableau de char.

..Pour une description complète du langage ..
..veuillez aller sur..
<http://ardx.org/PROG>

03 PROG

Bases
de Programmation

OPERATEURS MATHÉMATIQUES

Opérateurs utilisés pour effectuer des opérations mathématiques (ils fonctionnent simplement comme en mathématiques)

= (égalité) rend quelque chose égal à un autre (ex : $x = 10 * 2$ (x est maintenant égal à 20))
% (modulo) donne le reste de la division d'un nombre par un autre (ex $12 \% 10$ (donne 2))
+ (addition)
- (soustraction)
* (multiplication)
/ (division)

COMPARATEURS

Ces opérateurs sont utilisés pour les comparaisons logiques

== (égal à) (ex : $12 == 10$ est Faux et $12 == 12$ est Vrai)
!= (différent de) (ex : $12 != 10$ est Vrai et $12 != 12$ est Faux)
< (inférieur à) (ex : $12 < 10$ est Faux et $12 < 12$ est Faux et $12 < 14$ est Vrai)
> (supérieur à) (ex : $12 > 10$ est Vrai et $12 > 12$ est Faux et $12 > 14$ est Faux)

STRUCTURE DE CONTRÔLE

Les programmes sont chargés de gérer l'enchaînement des instructions. Voici les éléments de contrôle basiques (vous en découvrirez beaucoup plus sur Internet).

```
if(condition){  
else if( condition ){  
else {  
}
```

Le code entre accolades sera exécuté si la condition if (si) est vraie sinon la condition else if (sinon si) sera testée et si le résultat est encore faux le code else (sinon) sera exécuté.

```
for(int i = 0; i <  
#répétitions; i++){  
}
```

Utilisé quand vous voulez répéter un morceau de code un certain nombre de fois (il est possible de compter i++ et de décompter i-- ou encore d'utiliser n'importe quelle variable).

NUMÉRIQUE

```
pinMode(patte, mode);
```

Utilisé pour changer le mode d'une patte, patte est le numéro de la patte sur laquelle vous voulez agir (0-19, les analogiques 0-5 correspondent aux 14-19). Le mode peut être INPUT (entrée) et OUTPUT (sortie).

```
digitalWrite(patte, value);
```

Lorsqu'une patte est mise en OUTPUT, il est possible de changer son état. L'état peut être HIGH (mis à +5V) ou LOW (mis à la masse).

```
int digitalRead(patte);
```

Lorsqu'une patte est mise en INPUT vous pouvez utiliser cette fonction qui retourne son état, HIGH lorsqu'elle est mise à +5V et LOW lorsqu'elle est mise à la masse.

ANALOG

L'Arduino est une carte numérique mais elle peut aussi agir dans le domaine analogique. Voici comment utiliser des composants qui ne sont pas numériques.

```
int analogWrite(patte,  
value);
```

Certaines pattes de l'Arduino supportent les PWM (3, 5, 6, 9, 10, 11). Cela permet de changer l'état de la patte très rapidement et ainsi agir comme une sortie analogique. La valeur peut aller de 0 (0% de rapport cyclique ~0V) et 255 (100% de rapport cyclique ~5V).

```
int analogRead(patte);
```

Quand une patte analogique est mise en INPUT vous pouvez obtenir sa tension. Une valeur comprise entre 0 (pour 0V) et 1023 (pour 5V) est retournée.

L'ELECTRONIQUE EN BREF

Aucune expérience en électronique n'est requise pour s'amuser avec ce kit. Vous trouverez ici quelques informations à propos de chaque composant, afin de les identifier et peut-être de comprendre un peu mieux leur fonctionnement. Si vous vous posez des questions sur le fonctionnement d'un composant ou pourquoi il ne fonctionne pas, Internet est une mine d'informations et de conseils. Vous pouvez aussi nous contacter via : help@oomlout.com

LES COMPOSANTS EN DÉTAIL

LED/DEL

(Diode Electro-Luminescente)



Ce qu'il Fait:

Emet de la lumière lorsqu'elle est traversée par un faible courant (seulement dans un seul sens)

Identification:

Ressemble à une petite ampoule

Nombre de Pattes:

2 (une plus longue connectée au positif)

Les Choses à Savoir:

- Ne fonctionne que dans un seul sens.
- Nécessite une résistance pour limiter le courant.

Plus de Détails:

<http://ardx.org/LED>

Diode



Ce qu'il Fait:

L'équivalent électronique à une valve à sens unique. Elle autorise le courant à passer dans un sens mais pas dans l'autre.

Identification:

Généralement un cylindre avec des pattes à chaque extrémité (et une bague découpée indiquant la polarité)

Nombre de Pattes:

2

Les Choses à Savoir:

- Fonctionne dans un seul sens (le courant passe lorsque la patte avec la bague est connectée à la masse).

Plus de Détails:

<http://ardx.org/DIODE>

Résistance



Ce qu'il Fait:

Réduit le courant qui peut passer dans le circuit.

Identification:

Un cylindre avec des pattes à chaque extrémité. La valeur est affichée au moyen d'un code couleur (voir la page suivante pour plus de détails)

Nombre de Pattes:

2

Les Choses à Savoir:

- Très facile de prendre la mauvaise valeur (regarder à deux fois la valeur avant de l'utiliser).

Plus de Détails:

<http://ardx.org/RESI>

Transistor



Ce qu'il Fait:

Utilise un petit courant pour commuter ou amplifier un courant plus important.

Identification:

Se trouve dans un grand nombre de types de boîtiers mais vous pouvez y lire la référence (P2N2222AG dans ce kit et trouver la datasheet sur Internet).

Nombre de Pattes:

3 (Base, Collecteur, Emetteur)

Les Choses à Savoir:

- Le brancher dans le bon sens (une résistance pour limiter le courant est souvent requise sur la base).

Plus de Détails:

<http://ardx.org/TRAN>

Servo Hobby



Ce qu'il Fait:

Prend une pulsation et la convertit en une position angulaire sur son axe de rotation.

Identification:

Une boîte en plastique avec 3 fils qui sortent d'un côté et un axe avec des bras en plastique sur le dessus.

Nombre de Pattes:

3

Les Choses à Savoir:

- Le connecteur n'est pas détrompé donc faire attention à le brancher dans le bon sens.

Plus de Détails:

<http://ardx.org/SERV>

Moteur CC



Ce qu'il Fait:

Tourne lorsqu'un courant le traverse.

Identification:

C'est simple, ça ressemble à un moteur. Généralement c'est un cylindre avec un axe d'un côté.

Nombre de Pattes:

2

Les Choses à Savoir:

- S'utilise avec un transistor ou un relay dimensionné pour le type de moteur utilisé.

Plus de Détails:

<http://ardx.org/MOTO>

LES COMPOSANTS EN DÉTAIL (SUITE)

Buzzer Piezo



Ce qu'il Fait:

Une impulsion de courant lui fera émettre un clic.
Une chaîne d'impulsions lui fera émettre une tonalité.

Identification:

Dans ce kit il ressemble à un petit fut de plastique mais parfois il s'agit juste d'un disque doré.

Nombre de Pattes:

2

Les Choses à Savoir:

- Difficile de mal l'utiliser

Plus de Détails:

<http://ardx.org/PIEZ>

IC (Circuit Intégré)



Ce qu'il Fait:

Contient toute sorte d'électronique compliquée dans un boîtier facile à utiliser.

Identification:

La référence est écrite sur le boîtier. (Il est parfois nécessaire d'avoir beaucoup de lumière ou une loupe pour la lire).

Nombre de Pattes:

2 - XXX (dans ce kit il y en a un avec 3(TMP36) et un avec 16 (74HC595)).

Les Choses à Savoir:

- Le placer dans le bon sens (chercher le repère qui montre la patte 1).

Plus de Détails:

<http://ardx.org/ICIC>

Bouton poussoir



Ce qu'il Fait:

Ferme le circuit lorsqu'il est pressé

Identification:

Un petit rectangle noir avec des pattes en dessous et un bouton sur le dessus.

Nombre de Pattes:

4

Les Choses à Savoir:

- Ils sont presque carré donc ils peuvent être tournés à 90 degrés.

Plus de Détails:

<http://ardx.org/BUTT>

Potentiomètre



Ce qu'il Fait:

Résistance variable dépendante de la position angulaire de son axe.

Identification:

ils peuvent se présenter sous différentes formes, cherchez le bouton rotatif pour les identifier.

Nombre de Pattes:

3

Les Choses à Savoir:

- En acheter un à échelle logarithmique par accident.

Plus de Détails:

<http://ardx.org/POTE>

Photo Resistance



Ce qu'il Fait:

Résistance variable dépendante de la lumière ambiante.

Identification:

Généralement un disque avec une ligne courbe sur le dessus.

Nombre de Pattes:

2

Les Choses à Savoir:

- Souvenez-vous qu'il faut l'inclure dans un pont diviseur pour l'utiliser.

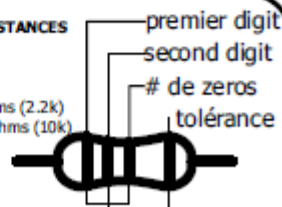
Plus de Détails:

<http://ardx.org/PHOT>

CODE COULEUR DES RÉSISTANCES

Exemples :

vert-bleu-marron - 560 ohms
rouge-rouge-rouge - 2 200 ohms (2.2k)
marron-noir-orange - 10 000 ohms (10k)



0 - Noir	5 - Vert
1 - Marron	6 - Bleu
2 - Rouge	7 - Violet
3 - Orange	8 - Gris
4 - Jaune	9 - Blanc

20% - rien
10% - argent
5% - or

RACCOURCISSEMENT DES PATTES

Certains composants de ce kit sont fournis avec de longues pattes. Pour les rendre plus compatibles avec une breadboard, quelques adaptations sont nécessaires.

LEDs :

Coupez les pattes de façon à ce que la plus longue fasse ~10mm et la plus courte ~7mm,

Résistances :

Piez les pattes de façon à ce qu'elles soient à 90 degrés de la partie cylindrique. Ensuite coupez les pattes pour avoir ~6mm.

Autres Composants :

D'autres composants peuvent nécessiter un raccourcissement. Faites-le de façon réfléchie.

CIRC-01

.:Débuter:. .:(Faire clignoter des LEDs):.



CE QUE NOUS ALLONS FAIRE:

Les LEDs (diode électroluminescente) sont utilisées dans toutes sortes de choses cools, c'est pourquoi nous les avons incluses dans ce kit. Nous allons commencer par quelque chose de très simple, en allumant et éteignant l'une d'elle, à plusieurs reprises, produire un agréable effet de clignotement. Pour commencer, réunissez le matériel listé ci-dessous, placez la feuille de connexions sur votre breadboard et insérez tous les composants. Une fois le circuit assemblé, vous devrez transférer le programme. Pour ce faire, branchez l'Arduino à votre port USB. Sélectionnez ensuite le port de communication approprié dans **Tools > Serial Port > (le port com de votre Arduino)**. Ensuite transférez le programme en faisant **File > Upload to I/O Board (ctrl+U)**. Finalement, contemplez votre œuvre et les possibilités qu'offre le contrôle de la lumière.

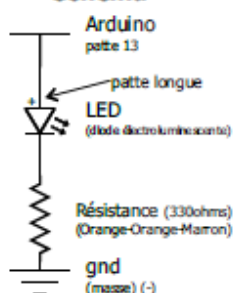
Si vous rencontrez des problèmes pour transférer le programme, un guide complet de résolution de problème peut-être trouvé ici : <http://ardx.org/TRBL>

LE CIRCUIT:

Composants :

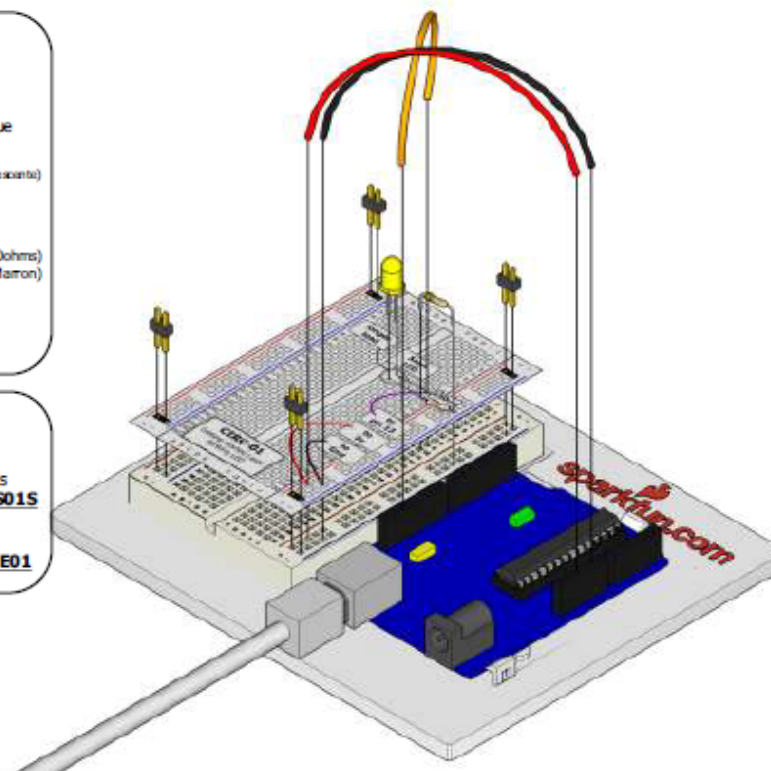
- | | | | |
|---|---|---|---|
|  CIRC-01
Feuille de Connexions
x1 |  Connecteur 2 pattes
x4 |  LED Jaune 5mm
x1 |  Fil |
|  Résistance 330 Ohms
Orange-Orange-Marron
x1 | | | |

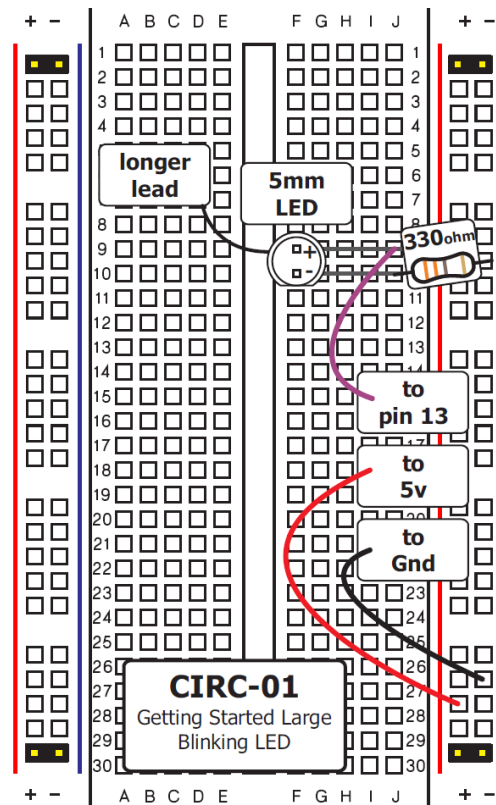
Schéma



Internet

.:Télécharger:.
Feuille de Connexions
<http://ardx.org/B8LS01S>
.:Voir:.
Vidéo de Montage
<http://ardx.org/VIDE01>





Pour ne pas taper le code ci-dessous, on peut ouvrir un exemple qui le reprend dans « Fichiers », « Exemples », « 1. Basic », « Blink » (par contre les commentaires seront en anglais).

CODE (Il n'est pas nécessaire de taper quoi que ce soit)

File > Examples > 1.Basic > Blink

(exemple du site arduino.cc, allez voir pour trouver d'autres idées)

```

/* Blink
 * Allume une LED pendant une seconde puis l'éteint pendant une seconde,
 * de façon répétée.
 * Créé le 1er juin 2005 par David Cuartielles
 * http://arduino.cc/en/Tutorial/Blink
 * d'après H. Barragan
 */

int ledPin = 13;    //LED connectée sur la patte numérique 13

//La fonction setup() s'exécute une fois, lorsque le programme se lance
void setup() {      //initialise la patte numérique en tant que sortie :
  pinMode(ledPin, OUTPUT);
}

//La fonction loop() s'exécute encore et encore,
// aussi longtemps que l'Arduino est alimentée
void loop()
{
  digitalWrite(ledPin,HIGH);    //allume la LED
  delay(1000);                  //attend une seconde
  digitalWrite(ledPin,LOW);     //éteint la LED
  delay(1000);                  //attend une seconde
}

```

CIRC-01

CELA FONCTIONNE PAS ? (3 choses à essayer)

La LED ne s'allume pas ?

Une LED ne fonctionne que dans un seul sens. Essayez de la tourner de 180 degrés. (Ne vous inquiétez pas brancher une LED à l'envers ne cause pas de dommage).

Le programme ne se charge pas

Cela arrive parfois. Le plus souvent à cause de la sélection du port COM. Vous pouvez le changer avec :
tools>serial port>

Cela fonctionne toujours pas ?

Un circuit qui ne fonctionne pas ce n'est jamais sympa. Envoyez nous un mail, nous vous répondrons dès que possible.

help@oomlout.com

1) Avant de faire la partie « Améliorer le montage », changez la résistance (par exemple en mettant 1 kOhms).

Que constatez-vous ?

2) Revenez au montage initial proposé ci-dessus et commentez le deuxième « delay ». Que constatez-vous et pourquoi ?

3) Remettez le « delay » et faites varier les durées.

4) Vous pouvez maintenant faire la partie « Améliorer le montage ».

AMÉLIORER LE MONTAGE

Changer les branchements :

La LED est connectée à la patte 13 mais nous pouvons utiliser toutes les pattes de l'Arduino. Pour la changer prenez le fil branché à la patte 13 et déplacez le sur la patte de votre choix (de 0 déjà 13)(vous pouvez aussi utiliser les entrées analogiques 0-5, l'analogique 0 est la patte 14...)

Ensuite, dans le code, il faut changer la ligne :
`int ledPin = 13; -> int ledPin = nouvellepatte;`

Ensuite envoyez le code : (ctrl-u)

Changer la vitesse de clignotement :

mécontent avec une seconde allumée et une seconde éteinte ?

dans le code, changez les lignes :
`digitalWrite(ledPin, HIGH);
delay(temps on); //(secondes * 1000)
digitalWrite(ledPin, LOW);
delay(temps off); //(secondes * 1000)`

Contrôler la luminosité :

Comme avec les E/S numériques, l'Arduino peut contrôler certaines pattes en tant que pattes analogiques.(vous trouverez plus de détails à ce propos dans les prochains circuits). Pour jouer avec

Changer la LED pour la patte 9 : (changez aussi le fil)
`ledPin = 13; -> int ledPin = 9;`

Remplacez le code entre {} de la fonction loop() par celui-ci :
`analogWrite(ledPin, nouveau nombre);`

(nouveau nombre) = n'importe quel nombre entre 0 et 255.
0 = éteinte, 255 = allumée, entre les deux = différentes luminosités

Variation de luminosité :

Nous allons utiliser un autre exemple. Pour l'ouvrir allez dans :

File > Examples > 3.Analog > Fading

Ensuite envoyez le code sur la carte et regardez la LED croître et décroître en luminosité

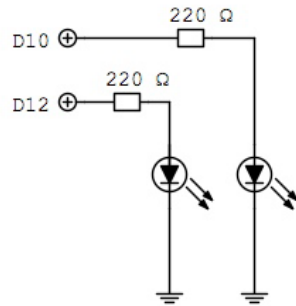
PLUS, PLUS, PLUS :

Plus de détails, où acheter des composants, où poser plus de questions :

<http://ardx.org/CIRC01>

09

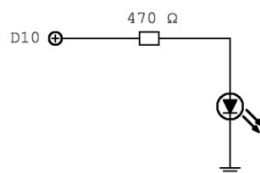
5) Maintenant que vous maîtrisez un peu mieux Arduino, réalisez le montage suivant (avec des résistances 330 Ohms pour nous) :



6) Écrire le code pour faire clignoter les deux LEDs en même temps.

7) Modifiez votre code pour faire clignoter les LEDs en alternance

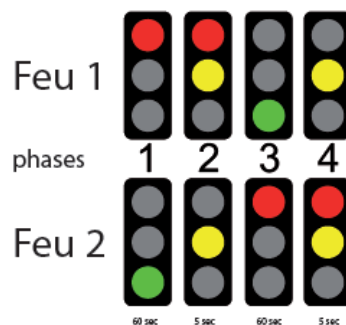
8) L'objectif est maintenant de faire clignoter une LED 10 fois et seulement 10 fois. Écrivez le code. Le montage est toujours le même (avec une résistance 330 ohms pour nous). Il y a deux solutions.



9) Les feux de circulation

L'objectif est de créer deux feux de circulation et de les faire fonctionner de manière synchrone.

Voici les phases de deux feux de circulation que vous devez recréer (comme nous n'avons que deux couleurs de LEDs, celle du milieu sera identique à une des deux autres) :



Établissez la liste des composants que vous avez besoin pour réaliser le projet, réalisez votre schéma électrique du montage



Programmez l'arduino !

On pourrait imaginer des variantes :

Variante 1

Il y a souvent un décalage entre le passage d'un feu au rouge et le passage au vert de l'autre feu. C'est en particulier le cas pour les feux de chantier. Cela permet aux voitures encore engagées dans la zone de chantier de la quitter, avant de laisser la place aux voitures venant en face.

Décrire les phases des feux et les programmer pour tenir compte du délai.

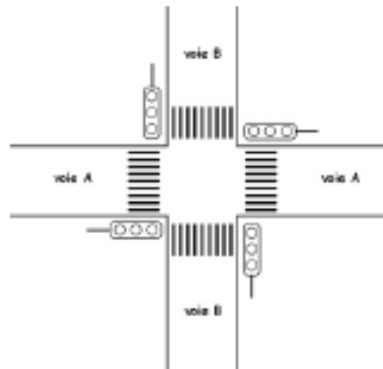
Variante 2

Intégrer un troisième feu, qui passe du rouge au vert en alternance avec les deux autres feux.

Réaliser le schéma électronique et programmer les feux.

Variante 3

Réaliser les feux pour un carrefour:



CIRC-07

.:Presser des boutons.: .:Bouton poussoir.:



CE QUE NOUS ALLONS FAIRE:

A l'heure actuelle nous nous sommes concentrés uniquement sur les sorties, il est temps de mettre votre Arduino à l'écoute. Nous allons commencer avec un simple bouton poussoir. Connecter le bouton poussoir est simple. Il y a un composant qui paraît inutile, la résistance de pull-up. Elle est ajoutée car l'Arduino ne ressent pas les choses comme nous (bouton pressé ou non). A la place elle regarde la tension sur la patte et détermine si c'est HIGH ou LOW (haut ou bas). Le bouton est mis de façon à ce que la patte de l'Arduino soit LOW quand le bouton est pressé. Cela veut dire que quand le bouton est au repos la patte est « en l'air » (ce qui peut poser des problèmes occasionnels). Pour permettre à l'Arduino de lire l'état en tant que HIGH quand le bouton est au repos, nous ajoutons une résistance de pull-up. (note : le premier programme d'exemple utilise un seul des deux boutons)

LE CIRCUIT:

Composants :



CIRC-07
Feuille de Connexions
x1



Connecteur 2 pattes
x4



Bouton poussoir
x2



Fil



Résistance 10k Ohms
Marron-Noir-Orange
x2

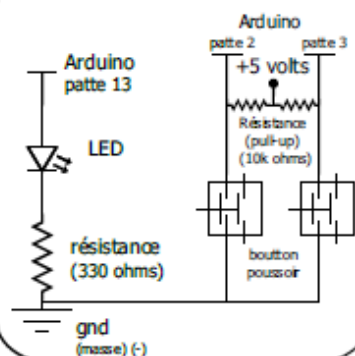


Résistance 330 Ohms
Orange-Orange-Marron
x1



Rouge LED
x1

Schéma



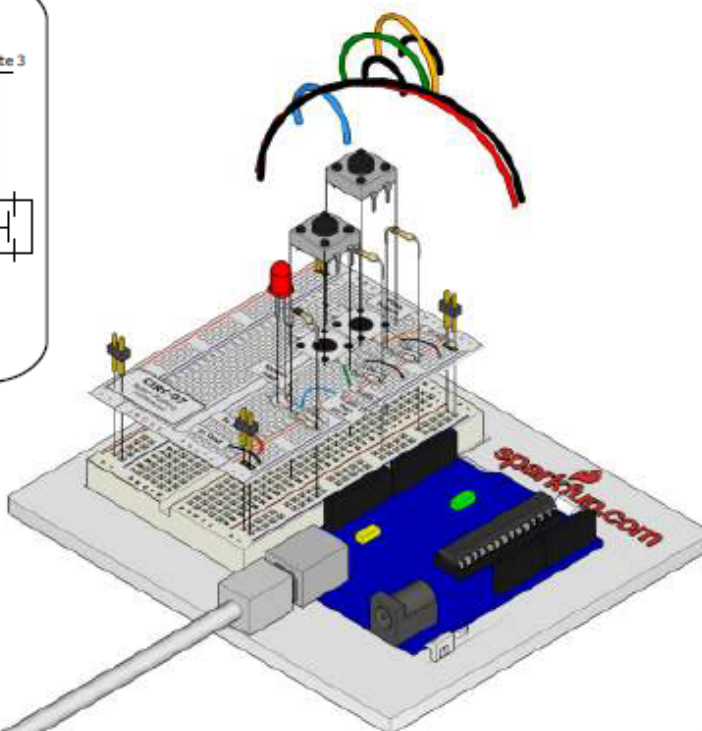
Internet

.:Télécharger.:
Feuille de Connexions

<http://ardx.org/BBS07S>

.:Voir.:
Vidéo de Montage

<http://ardx.org/VIDE07>



Reprenons le TP et insérons **un** bouton (même si le schéma ci-dessous en présente deux).
 Pour ne pas taper le code ci-dessous, on peut ouvrir un exemple qui le reprend dans « Fichiers », « Exemples », « 2. Digital », « Button » (par contre les commentaires seront en anglais).

CODE (Il n'est pas nécessaire de taper quoi que ce soit)

File > Examples > 2.Digital > Button

(exemple du site arduino.cc, allez y faire un tour pour trouver d'autres idées)

```
/* Bouton
 * par DojoDave <http://www.0j0.org>
 *
 * Allume et éteint une Diode Electroluminescente (DEL/LED) connectée à la patte
 * numérique 13, lorsque le bouton poussoir connecté à la patte 7 est enfoncé.
 * http://www.arduino.cc/en/Tutorial/Button
 */
int ledPin = 13; //Sélectionne la patte pour la LED< br>int inputPin = 2;
int val = 0; //détermine la patte d'entrée (pour un bouton poussoir)
//variable pour lire l'état de la patte< br>

void setup() {
  pinMode(ledPin, OUTPUT); //déclare la LED en sortie
                          //pinMode(inputPin, INPUT);
                          //déclare le bouton en entrée
}

void loop(){
  val = digitalRead(inputPin); //lit la valeur d'entrée
  if (val == HIGH) { //regarde si l'entrée est haute
    digitalWrite(ledPin, LOW); // éteint la LED
  } else {
    digitalWrite(ledPin, HIGH); // allume la LED
  }
}
```

CIRC-07

CELA FONCTIONNE PAS ? (3 choses à essayer)

La lumière ne s'allume pas

Le bouton poussoir est carré et donc facile à placer dans le mauvais sens.
 Tournez le de 90 degrés et regardez si ça marche.

Le fondu de lumière ne fonctionne pas

Une erreur que nous faisons constamment, quand vous changez du simple clignotement au fondu: n'oubliez pas de changer le fil de la LED de la patte 13 à la 9.

Pas impressionné ?

Pas d'inquiétudes, ces circuits sont super simplifiés pour rendre le jeu avec les composants facile. Mais une fois assemblés, le ciel est la seule limite.

1) Avant de faire la partie « Améliorer le montage », faites-en sorte que la LED soit toujours allumée et qu'elle s'éteigne lors de l'appuie sur le bouton.

AMÉLIORER LE MONTAGE

Bouton On bouton OFF :

L'exemple de base peut être un peu décevant (on n'a pas vraiment besoin d'une Arduino pour faire ça). C'est parti pour compliquer le montage. Un bouton allumera la LED et un autre l'éteindra. Changez le code avec :

```
int ledPin = 13; //sélectionne la patte pour la LED
int inputPin1 = 3; // bouton 1
int inputPin2 = 2; // bouton 2

void setup() {
  pinMode(ledPin, OUTPUT); //place la LED en sortie
  pinMode(inputPin1, INPUT); //place le bouton 1 en entrée
  pinMode(inputPin2, INPUT); //place le bouton 2 en entrée
}

void loop() {
  if (digitalRead(inputPin1) == LOW) {
    digitalWrite(ledPin, LOW); //éteint la LED
  } else if (digitalRead(inputPin2) == LOW) {
    digitalWrite(ledPin, HIGH); //allume la LED
  }
}
```

Envoyez le programme à la carte et commencez à allumer et éteindre la LED.

Fondu :

Contrôlons un signal analogique avec les boutons. Pour cela vous allez devoir changer la connexion de la LED de la patte 13 à la patte 9 ainsi que le code suivant.

```
int ledPin = 13; ----> int ledPin = 9;
// Ensuite changez le code de loop().

int value = 0;
void loop() {
  if (digitalRead(inputPin1) == LOW) { value--; }
  else if (digitalRead(inputPin2) == LOW) { value++; }
  value = constrain(value, 0, 255);
  analogWrite(ledPin, value);
  delay(10);
}
```

Changer le temps de fondu :

Si vous voulez accélérer ou ralentir le fondu, il n'y a qu'une ligne de code à changer :

```
delay(10); ----> delay(nouveau delais );
```

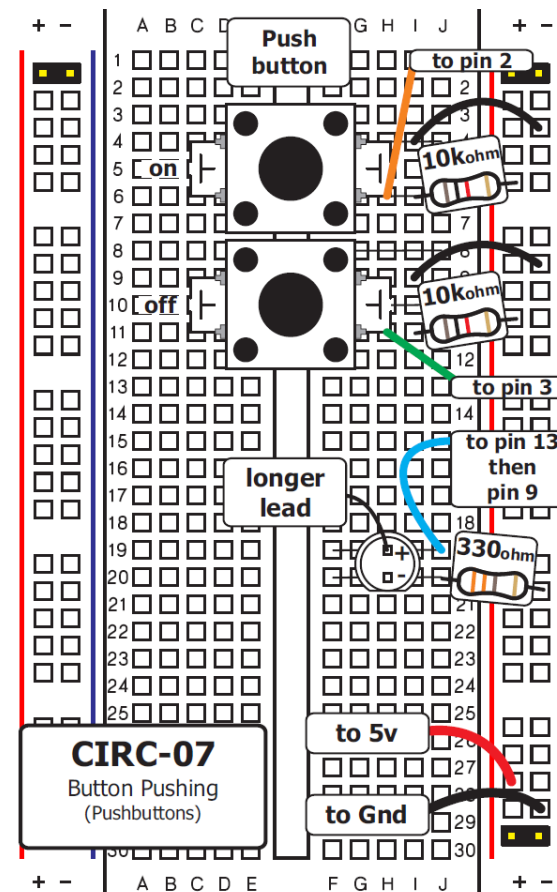
Pour accélérer réduisez le nombre et pour ralentir augmentez le.

PLUS, PLUS, PLUS :

Plus de détails, où acheter des composants, où poser plus de questions :

<http://ardx.org/CIRC07>

21



Avec un potentiomètre ! Cette fois on passe dans le monde de l'analogique !

CIRC-08

..Rotation.. ..Potentiomètres..



CE QUE NOUS ALLONS FAIRE:

En plus des pattes numériques, l'Arduino possède 6 pattes qui peuvent être utilisées pour des entrées analogiques. Ces entrées prennent un voltage (entre 0 et 5V) et le convertissent en un nombre compris entre 0 (0V) et 1024 (5V) (résolution de 10 bits). Un composant très utile qui utilise ces entrées est le potentiomètre (auss appelé résistance variable). Quand il est connecté avec 5V entres ces pattes extérieures, la patte centrale ressort des valeurs comprises entre 0 et 5V en fonction de l'angle selon lequel son axe est tourné (ex : 2,5V au milieu). Nous pouvons ensuite utiliser cette valeur comme variable de notre programme.

LE CIRCUIT:

Composants :

CIRC-08
Feuille de Connexions
x1

Connecteur 2 pattes
x4



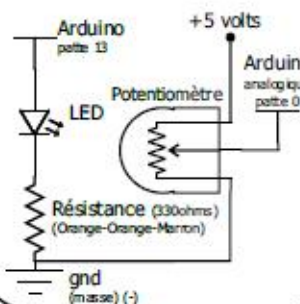
Potentiomètre
10k ohms
x1

Fil

LED Jaune
x1

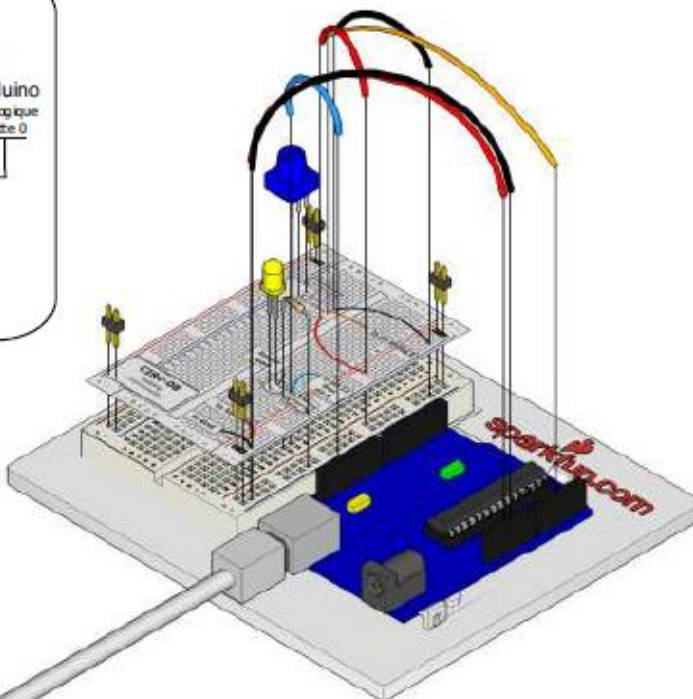
Résistance 330 Ohms
Orange-Orange-Marron
x1

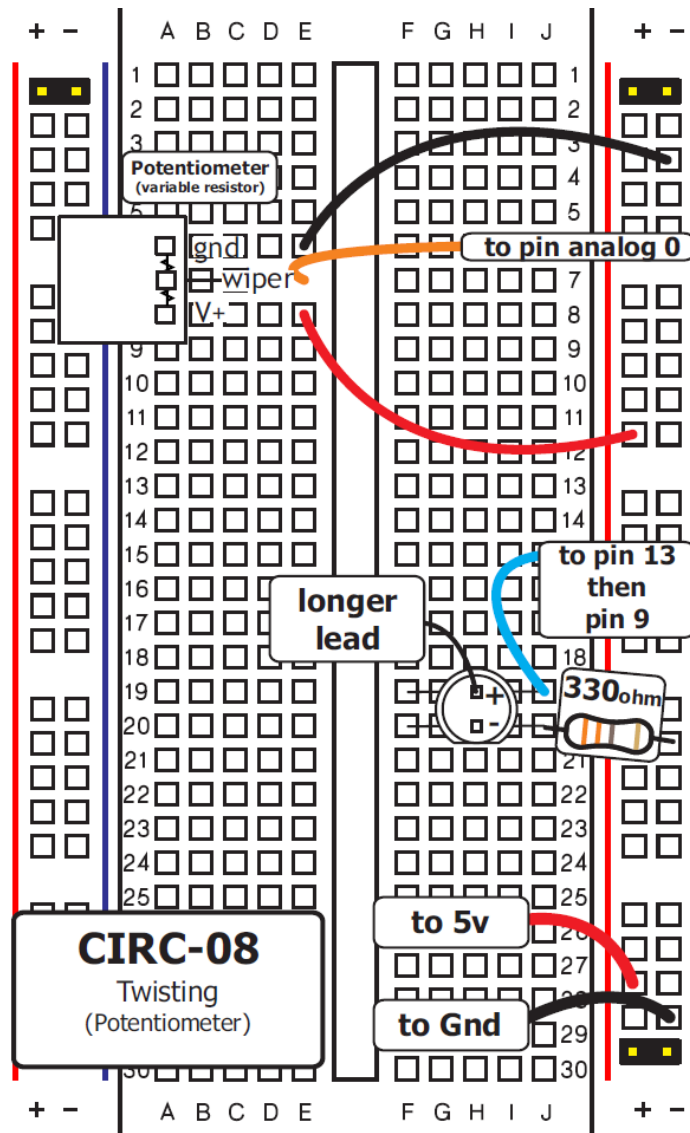
Schéma



Internet

..Télécharger..
Feuille de Connexions
<http://ardx.org/BBL508S>
..Voir..
Vidéo de Montage
<http://ardx.org/VIDE08>





Pour ne pas taper le code ci-dessous, on peut ouvrir un exemple qui le reprend dans « Fichiers », « Exemples », « 3. Analog », « AnalogInput » (par contre les commentaires seront en anglais).

CODE (Il n'est pas nécessaire de taper quoi que ce soit)

File > Examples > 3.Analog > AnalogInput

(exemple du site arduino.cc, allez y faire un tour pour trouver d'autres idées)

```
/* Entrée analogique
 * Explication de l'entrée analogique en lisant un capteur analogique
 * sur
 * Le pin analogique 0 et en allumant une LED connectée au pin digital 13.
 * Le temps pendant lequel la LED sera allumée ou éteinte dépend de la valeur obtenue par
 * analogRead().
 * Créé David Cuartielles
 * Modifié le 16 Juin 2009
 * Par Tom Igoe
 * http://arduino.cc/en/Tutorial/AnalogInput
 */

int sensorPin = 0; // sélectionne le pin d'entrée pour le potentiomètre
int ledPin = 13; // sélectionne le pin pour la LED
int sensorValue = 0; // variable pour stocker la valeur venant du capteur

void setup() {
  pinMode(ledPin, OUTPUT); //déclare ledPin comme une sortie:
}

void loop() {
  sensorValue = analogRead(sensorPin); // lit la valeur du capteur:
  digitalWrite(ledPin, HIGH); // met ledPin à l'état haut
  delay(sensorValue); // arrête le programme pendant <sensorValue> millisecondes:
  digitalWrite(ledPin, LOW); // met ledPin à l'état bas:
  delay(sensorValue); // arrête le programme pendant <sensorValue> millisecondes:
}
```

CIRC-08

CELA FONCTIONNE PAS ? (3 choses à essayer)

Fonctionne par intermittence

On dirait un mauvais branchement du potentiomètre. Généralement vous pouvez le régler en l'enfonçant un peu plus.

Ne fonctionne pas

Vérifiez que vous n'avez pas connecté par erreur la sortie du potentiomètre sur le pin digital 2 au lieu du pin analogique 2. (La rangée de pins en dessous des pins d'alimentation)

Toujours à l'envers

Vous pouvez essayer de faire fonctionner le circuit à l'envers. Parfois ça fonctionne.

AMÉLIORER LE MONTAGE

Commutation selon un seuil:

Parfois vous voulez changer la valeur d'une sortie quand une valeur excède un certain seuil. Pour faire cela avec un potentiomètre, changez le code de la fonction loop() par:

```
void loop() {
  int threshold = 512;
  if(analogRead(sensorPin) > threshold){
    digitalWrite(ledPin, HIGH);}
  else{ digitalWrite(ledPin, LOW);}
}
```

La LED s'allume quand la valeur est au dessus de 512 (environ la moitié), vous pouvez ajuster la sensibilité en changeant la valeur du seuil.

Luminosité:

Contrôlons la luminosité de la LED directement avec le potentiomètre. Tout d'abord, il faut changer le pin auquel la LED est connecté. Déplacez le fil du pin 13 au pin 9 et changez cette ligne de code.

```
int ledPin = 13; ----> int ledPin = 9;
```

Puis changez le code dans loop.

```
void loop() {
  int value = analogRead(potPin) / 4;
  analogWrite(ledPin, value);
}
```

Chargez le code et voyez comme la luminosité de la LED varie en fonction du potentiomètre. (Note: la raison pour laquelle on divise la valeur par 4 est que la fonction analogRead() retourne une valeur entre 0 et 1023 (10 bits), et analogWrite() prend une valeur entre 0 et 255 (8 bits))

Contrôler un servo:

C'est vraiment un bel exemple et il permet d'associer plusieurs circuits. Câblez le servo comme pour le circuit CIRC-04 et ouvrez le programme d'exemple Knob (File > Examples > Servo > Knob), puis changez une ligne de code.

```
int potpin = 0; ----> int potpin = 2;
```

Chargez sur votre Arduino et observez le servo tourner en fonction du potentiomètre.

PLUS, PLUS, PLUS :

Plus de détails, où acheter des composants, où poser plus de questions :

<http://ardx.org/CIRC08>

23