



Arduino

à l'école 1

Apprendre aux élèves à programmer sur Arduino, c'est non seulement leur donner la clé d'accès aux ordinateurs, mais aussi aux objets qui les entourent





Merci à Damien, Lionel, Kristijan et Nikola, élèves de l'option robotique 2012-2013 des Écoles d'Ecublens, qui ont eu la difficile tâche de tester ce cours.

Merci à Arnold Pasche de m'avoir mis le pied à l'étrier et pour ses conseils en matière de soudure et d'électricité.

Et enfin un merci tout particulier au Professeur Jean-Daniel Nicoud pour sa précieuse aide et son soutien et pour avoir créé le Diduino en pensant aux besoins des écoles.

Vous appréciez ce cours? N'hésitez pas à me le faire savoir et à me faire parvenir vos commentaires, corrections ou suggestions!

info [at] edurobot.ch



Consignes de sécurité

L'électricité peut être mortelle! Pour éviter tout risque, en particulier avec des élèves, ne travailler qu'avec de la **très basse tension (TBT)**. La tension de fonctionnement de l'Arduino se situe autour de 5 Volts. Avec les élèves, je me suis volontairement limité à travailler en courant continu avec une différence de potentiel maximum de 24 Volts.



- ✿ Ne jamais interfacer directement l'Arduino avec le secteur (230 Volts alternatif).
- ✿ Pour l'alimentation des projets, utiliser des transformateurs répondants aux normes de sécurité en vigueur.
- ✿ Avec des élèves, ne pas utiliser de transformateurs ou appareils sur le secteur *fait maison*.
- ✿ Ne pas démonter d'appareils électroniques, sans supervision. Certains composants, comme les condensateurs, peuvent délivrer des décharges électriques mortelles, même lorsqu'ils ne sont pas connectés au secteur.



Préface	8
Introduction	9
Références	9
Objectifs du plan d'études romand	10
À propos des schémas électroniques	12
Arduino: présentation	13
Le matériel	15
Installer Arduino sur Mac: aide-mémoire	17
Pilote	17
<i>Installation du pilote</i>	17
Installation du logiciel Arduino	18
Configuration du logiciel Arduino	18
<i>Sélection du port série</i>	18
<i>Sélection du type de carte Arduino</i>	18
Aide	18
La gestion de classe	19
Gérer le matériel	19
Réaliser des projets	19
LEÇON 1: Découverte de la plateforme Arduino	20
Arduino ou Diduino?	21
Constitution de la carte	22
Le microcontrôleur	23
Alimentation	23
La connectique	24
Connecteurs Diduino	26
La platine d'expérimentation	26



Leçon 2: La soudure	27
Leçon 3: Les bases de l'électronique	28
L'électricité	28
Le circuit électrique	30
Les diodes	31
Les résistances	33
Leçon 4: faire clignoter la LED	37
Introduction	37
Le menu	38
La LED	39
<i>Code 1</i>	39
<i>Introduction au code</i>	40
Le déroulement du programme	40
Le code minimal	41
La fonction	41
Les instructions	42
Les points virgules ;	42
Les accolades { }	42
Les commentaires	43
Les accents	43
<i>Analyse du code 1</i>	44
<i>À toi de jouer!</i>	45
Monter une LED	46
<i>Code 2</i>	47
Variante: faire clignoter deux LEDs	48
<i>Exercice 1:</i>	48
<i>Exercice 2:</i>	48



Leçon 5: les variables	49
Une variable, qu'est ce que c'est ?	49
<i>Le nom d'une variable</i>	49
Définir une variable	50
L'incrémentation	51
Faire clignoter une LED 10 fois	52
<i>Code 3</i>	52
<i>Analyse du code</i>	53
Faire clignoter deux LED 20 fois	54
Définir les pattes du microcontrôleur	55
<i>Résumons!</i>	56
Évaluation: K2000	57
Introduction	57
Les objectifs	58
Étape 1: Schémas	59
Étape 2: Programme	60
Les feux de circulation	61
Étape 1: Schémas	62
Étape 2: Programme	63
Corrigé	64
Variantes	64
Leçon 6: Input	65
Le bouton poussoir	65
<i>Le bouton poussoir normalement ouvert (NO)</i>	65
<i>Le bouton poussoir normalement fermé (NF)</i>	65
<i>Les interrupteurs</i>	65
Exercice 1: allumons cette LED	66



Exercice 2: mais par où passe le courant?	67
Protéger l'Arduino	68
Résistance Pull-Down / Pull-Up	69
<i>Circuit avec une résistance pull-down</i>	69
<i>Circuit avec une résistance pull-up</i>	71
<i>Résistance pull-down ou pull-up?</i>	71
Exercice 3: montage avec résistance pull-down (rappel au moins)	72
Une petite variation de code	75
Petits exercices: bouton poussoir et LED qui clignote	76
Evaluation: vers l'infini et au-delà!	77
Les objectifs	77
Etape 1: les schémas	79
Étape 2: Programme	80
Leçon 7: le barregraphe	81
Introduction	81
Schémas	81
Le code	83
Variation: Le barregraphe à 10 LEDs	87
Variation: l'afficheur numérique	88
<i>Identification de la position des LEDs</i>	89
Leçon 8: apprendre à compter	94
Objectif	94
Schéma électronique du montage	94
Plan de montage	95
Le code	96
Conclusion	101



Préface

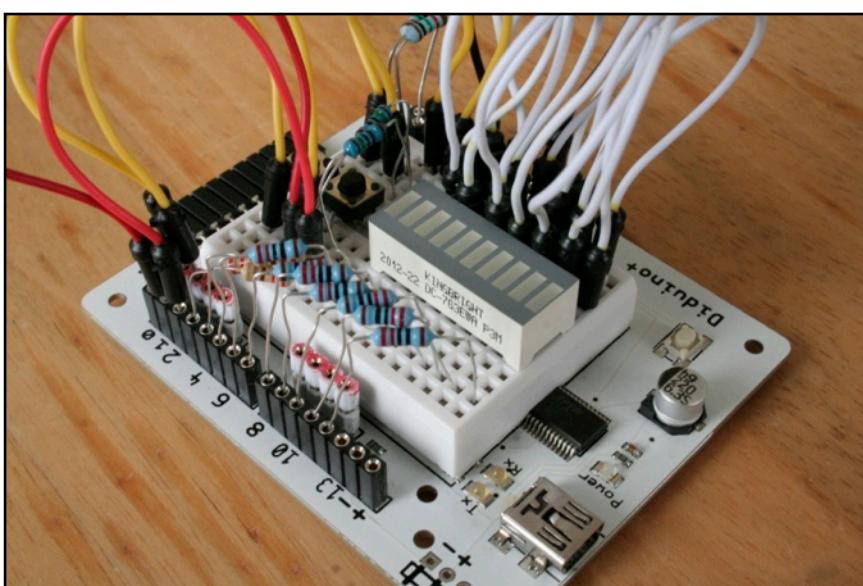
Lorsque Massimo Banzi et ses collègues de l'*Interaction Design Institute* d'Ivrea, en Italie, ont développé l'Arduino, l'objectif était de permettre aux étudiants de pouvoir disposer d'une plateforme valant le prix d'une pizza pour réaliser des projets interactifs.

Ainsi, l'Arduino a été conçu pour être bon marché, doté d'une grande quantité d'entrées et de sorties, compatible Mac, Windows et Linux, programmable avec un langage très simple et open source. Il n'y a là que des avantages pour le monde scolaire, en particulier parce qu'il se situe au croisement entre l'informatique, l'électronique et les travaux manuels¹.

L'approche pédagogique de l'Arduino est particulière. Il ne s'agit pas d'aborder la matière d'une manière linéaire, mais en bricolant et en «bidouillant»: on câble, on branche et on regarde ce que cela donne. C'est une approche par la pratique et l'expérimentation, qui convient très bien à des élèves, même (et surtout) peu scolaires. Il y a bien sûr un risque de «griller» un Arduino; mais il ne s'agit que de 30 francs de matériel, et pas d'un ordinateur à 1'200 francs!

Ce qui est sans doute la meilleure preuve que l'Arduino est parfaitement adapté aux élèves, c'est qu'en quelques leçons, ils sont déjà prêts à réaliser des projets concrets.

Ce cours a été pensé pour des élèves (et des enseignants) qui n'ont aucune notion en programmation et en électronique. Par rapport au gigantesque potentiel de l'Arduino, il ne va pas très loin, mais il s'efforce d'être progressif et surtout axé sur la pratique.



¹ Références: <http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino> et <http://www.edurobot.ch/?p=1554>



Introduction

Références

Ce document est une compilation de textes et d'exercices, parfois repris *in extenso* depuis les sources suivantes:

Sources principales:

- <http://arduino.cc/fr/>
- <http://sciences.siteduzero.com/tutoriel-3-515602-arduino-pour-bien-commencer-en-electronique-et-en-programmation.html>
- <http://mediawiki.e-apprendre.net/index.php/Diduino-Robot>

Sources annexes:

- http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ARDUINO
- <http://www.didel.com/diduino/Cours01.pdf>
- <http://www.didel.com/diduino/Cours02.pdf>
- <http://chamayou.franck.free.fr/spip/spip.php?article177>
- <http://www.hexapod-arduino.com>
- <http://www.lagrottedubarbu.com>
- <http://blog.makezine.com/arduino/>
- <http://www.ladyada.net/learn/arduino/>
- <http://www.craslab.org/arduino/livrehhtml/LivretArduinoCRAS.html>
- <http://arduino103.blogspot.ch>
- <http://www.semageek.com>

En cas d'utilisation de ce document, partiellement ou dans sa totalité, vous êtes priés d'y inclure ces sources.

Ce cours ne permet qu'une introduction à l'électronique. Un cours bien plus complet et très bien fait est disponible ici:

<http://sciences.siteduzero.com/tutoriel-3-483697-l-electronique-de-zero.html>



Objectifs du plan d'études romand

Voici une partie des objectifs du PER abordés avec ce cours:

Objectifs de la discipline MSN

MSN 33 — Résoudre des problèmes numériques et algébriques...

- en reconnaissant les caractéristiques mathématiques d'une situation et en la traduisant en écritures numérique ou littérale
- en utilisant des propriétés des opérations (+, -, ×, ÷, puissance, racine carrée et cubique)
- en choisissant l'outil de calcul le mieux approprié à la situation proposée
- en mobilisant l'algèbre comme outil de calcul (équations), de preuve ou de généralisation
- en construisant, en exerçant et en utilisant des procédures de calcul (calcul réfléchi, algorithmes, calculatrice, répertoire mémorisé) avec des nombres réels
- en estimant un résultat et en exerçant un regard critique sur le résultat obtenu

MSN 34 — Mobiliser la mesure pour comparer des grandeurs...

- en connaissant le système international d'unités de mesures
- en explorant des aspects culturels et historiques liés au système d'unités
- en mobilisant l'instrument et l'unité de mesure adaptés
- en exprimant une mesure dans différentes unités
- en estimant l'importance relative des grandeurs dans un phénomène naturel ou social
- en estimant la mesure des grandeurs
- en calculant des grandeurs (résistances, courant,...)

MSN 35 — Modéliser des phénomènes naturels, techniques, sociaux ou des situations mathématiques...

- en mobilisant des représentations graphiques (codes, schémas, tableaux, graphiques,...)
- en associant aux grandeurs observables des paramètres
- en triant, organisant et interprétant des données
- en communiquant ses résultats et en présentant des modélisations

MSN 36 — Analyser des phénomènes naturels et des technologies à l'aide de démarches caractéristiques des sciences expérimentales...

- en formulant des hypothèses
- en confrontant les hypothèses émises à des résultats expérimentaux
- en définissant des stratégies d'exploration et d'expérimentation en lien avec les hypothèses émises
- en proposant des explications et en les confrontant à celles de ses pairs et aux informations de médias variés

**Objectifs de la discipline AC&M**

A 33 AC&M — Exercer diverses techniques plastiques et artisanales...

- en maîtrisant des habiletés de motricité globale et fine (souplesse, précision, coordination, rapidité du geste,...)
- en choisissant et utilisant divers outils et matériaux en fonction d'un projet
- en maîtrisant des gestes artisiaux spécifiques, en exerçant la précision
- en utilisant des techniques numériques

Objectifs de (FG) — MITIC

FG 31 — Exercer des lectures multiples dans la consommation et la production de médias et d'informations...

- en étudiant les manifestations de la «société de l'information et de la communication» et certaines de ses conséquences
- en analysant des images fixes et animées au moyen de la grammaire de l'image
- en vérifiant les informations reçues des médias et en produisant selon les mêmes modes

Objectifs de (FG) — Vivre ensemble et exercice de la démocratie

FG 34 — Planifier, réaliser, évaluer un projet et développer une attitude participative et responsable...

- en prenant une part active et des responsabilités dans un projet
- en évaluant ses actes et ses attitudes, en les ajustant si nécessaire
- en élaborant les étapes du projet, en recourant aux ressources pertinentes et en les évaluant
- en négociant une décision commune tout en tenant compte des intérêts et des besoins particuliers
- en débattant et en recherchant des réponses face à des problèmes concrets et des questions éthiques

Objectifs de la discipline MEP

MEP 34 – Développer des sujets, des projets favorisant une ouverture sur les sciences du passé, actuelles et du futur...

- Familiarisation avec des façons de faire et de raisonner propres à la logique robotique, notamment en:
 - identifiant les objectifs d'une activité ainsi que les notions théoriques et pratiques qui en résultent
 - planifiant, construisant et programmant le robot en fonction des observations et des mesures effectuées
 - testant le robot afin d'identifier d'éventuelles erreurs de conception et en mettant en place des stratégies de remédiation
- Initiation à l'utilisation d'outils et de procédés simples afin de faire fonctionner un robot de manière autonome



À propos des schémas électroniques

L'électronique a ceci d'intéressant qu'on passe d'un schéma électronique à un montage, ou au contraire, du montage au schéma électronique. Dans les deux cas, il s'agit d'une activité qui permet de travailler les objectifs suivants:

Objectifs de la discipline MSN

MSN 35 — Modéliser des phénomènes naturels, techniques, sociaux ou des situations mathématiques...

- en mobilisant des représentations graphiques (codes, schémas, tableaux, graphiques,...)
- en associant aux grandeurs observables des paramètres
- en triant, organisant et interprétant des données
- en communiquant ses résultats et en présentant des modélisations

MSN 36 — Analyser des phénomènes naturels et des technologies à l'aide de démarches caractéristiques des sciences expérimentales...

- en formulant des hypothèses
- en confrontant les hypothèses émises à des résultats expérimentaux
- en définissant des stratégies d'exploration et d'expérimentation en lien avec les hypothèses émises
- en proposant des explications et en les confrontant à celles de ses pairs et aux informations de médias variés

Pour réaliser les schémas électroniques, nous utilisons les outils suivants, disponibles gratuitement sur Mac et PC:

Solve Elec: <http://www.physicsbox.com/indexsolveelec2fr.html>

Fido Cadj: <http://davbucci.chez-alice.fr/index.php?argument=elettronica/fidocadj/fidocadj.inc>

Fritzing: <http://fritzing.org>

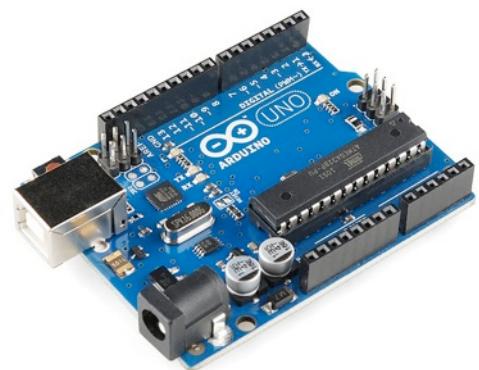


Arduino: présentation

L'Arduino est une plateforme open source d'électronique programmée qui est basée sur une carte à microcontrôleur et un logiciel. Plus simplement, on peut dire que l'Arduino est un module électronique, doté d'un microcontrôleur programmable. Il peut être utilisé pour développer des objets interactifs, munis d'interrupteurs ou de capteurs, et peut contrôler une grande variété de lumières, moteurs ou toutes autres sorties matérielles.

Le principe de fonctionnement est simple:

1. *On réalise le programme sur un ordinateur.*
2. *On connecte l'ordinateur à l'Arduino via une prise USB.*
3. *On envoie le programme sur l'Arduino.*
4. *L'Arduino exécute enfin le programme de manière autonome.*



La programmation se fait à l'aide d'un langage proche du C/C++, dont les bases sont faciles d'accès. Le logiciel nécessaire fonctionne à la fois sur Mac OSX, Windows et GNU/Linux et demande très peu de ressources. Il fonctionnera ainsi très bien sur un ancien ordinateur ou, par exemple, un Raspberry Pi². Comme le prix d'un module Arduino est faible (moins de 30 CHF), nous disposons donc d'un excellent outil pédagogique, peu onéreux qui nous permettra d'apprendre les bases de la programmation, de l'électricité, de l'électronique et de l'automation.

Arduino, Diduino, Seeduino, Freeduino...

Arduino est basé sur le concept de l'*open source hardware*, soit du matériel libre de droits. Ainsi, toute la plateforme matérielle et logicielle est libre de droits. N'importe qui peut donc développer et commercialiser un clone d'Arduino. Néanmoins, le nom et le logo Arduino sont des marques déposées, afin d'identifier le matériel officiel. Il y a donc cinq **catégories** de matériel: le matériel **officiel**, les **clones**, les **dérivés**, les **compatibles** et les **contrefaçons**³.

Le **matériel officiel**⁴ suit à la lettre le concept développé par l'équipe Arduino, contribue financièrement au développement du logiciel et de nouveaux matériaux et est fabriqué par des entreprises autorisées. Il est aussi entièrement documenté.

² Plus d'informations sur le Raspberry Pi: <http://swissraspi.ch>

³ Plus d'informations ici: <http://blog.arduino.cc/2013/07/10/send-in-the-clones/>

⁴ Matériel Arduino officiel: <http://arduino.cc/en/Main/Products>



Les **clones Arduino** sont des répliques plus ou moins fidèles de l'Arduino, qui ne portent pas le nom Arduino (mais souvent un nom en *XXX-duino*), ni le logo et respectent de ce fait le droit de la marque. En achetant un clone, nous aurons un produit de qualité, mais qui ne supportera pas financièrement le projet Arduino.

Les **dérivés Arduino** sont basés sur la technologie Arduino, mais proposent souvent des différences matérielles, ou de nouvelles fonctions. C'est le cas du Diduino.

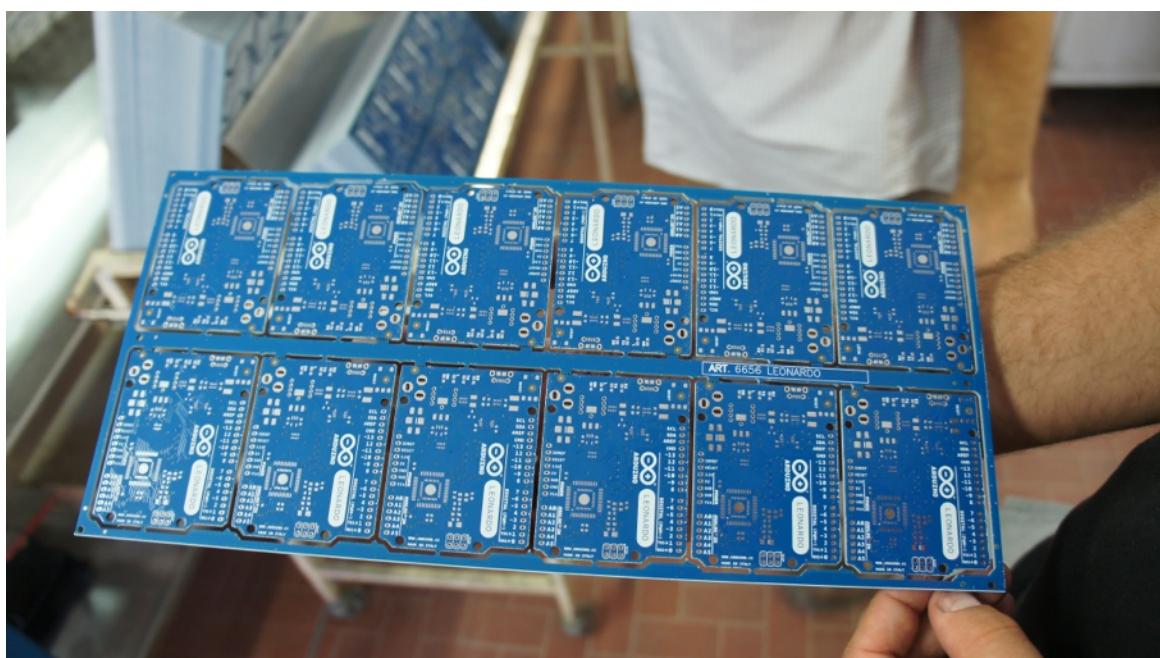
Les **compatibles Arduino** sont souvent du matériel très éloigné du concept de base, souvent avec des processeurs différents. Ces produits sont en général dédiés à des utilisations spécifiques.

Les **contrefaçons** sont des copies illégales d'Arduino, qui ressemblent à l'original, en portent la marque et le logo, ainsi que les signes distinctifs et qui, en général, fonctionnent comme l'original. Le problème n'est pas la qualité, mais l'abus du consommateur, qui pense acheter un module officiel, et ce faisant supporter la communauté open source Arduino.

Il faut en particulier se méfier des modules Arduino vendus sur des sites d'enchères ou sur Amazon; ils sont la plupart du temps des contrefaçons⁵.

La liste officielle des distributeurs de matériel Arduino est disponible ici:

<http://arduino.cc/en/Main/Buy>



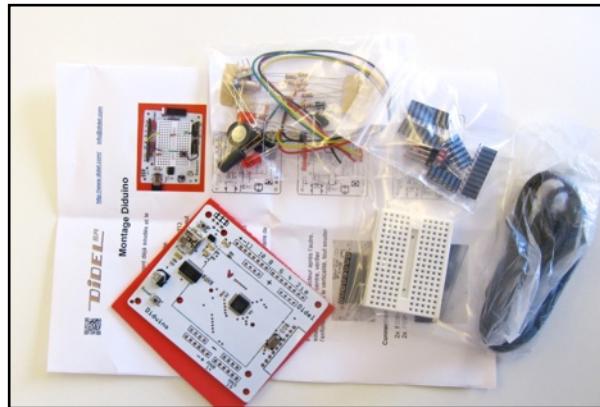
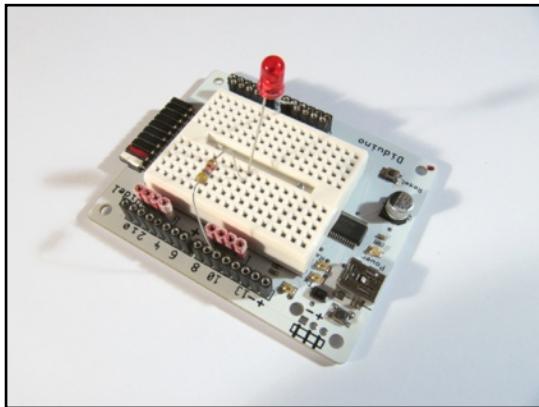
Modules Arduino en cours de fabrication. Source: <http://www.bunniestudios.com/blog/?p=2407>

⁵ Comment reconnaître une contrefaçon: <http://arduino.cc/en/Products/Counterfeit>



Le matériel

Ce cours est basé sur un module dérivé Arduino, appelé Diduino.



Nous avons choisi le Diduino, car il possède une platine d'expérimentation directement intégrée et se trouve livré avec de nombreux accessoires, tels que des résistances, des LEDs, des capteurs... bref, tout ce qu'il est nécessaire pour débuter sur Arduino. Le Diduino possède aussi de nombreuses alimentations +5V, ainsi que de nombreuses masses. Cela facilite le montage de circuits complexes. Enfin, le Diduino nécessite que l'on soude les connecteurs; cela ajoute ainsi une compétence à nos élèves.

Il est tout à fait possible de suivre ce cours avec un module Arduino original ou une copie d'Arduino. Dans ce cas, il vous faudra une platine d'expérimentation. L'Arduino *Starter Kit* convient très bien pour ce cours.

Attention: ce cours part du principe que le voltage de travail est de 5V! Il n'est pas prévu pour un Arduino Due. Le modèle recommandé est l'Arduino Uno ou Duemilanove.



Dans l'idéal, il faut prévoir un Arduino/Diduino par élève.



Pour ce cours, le matériel minimum nécessaire par élève est le suivant:

- Un Diduino (ou un Arduino)
- Une platine d'expérimentation
- 4 LEDs rouges
- 2 LEDs jaunes ou orange
- 2 LEDs vertes
- 1 LED bleue
- 3 résistances 100Ω
- 10 résistances 220Ω ou 470Ω
- 3 résistances $1k\Omega$ ou $10k\Omega$
- 3 boutons poussoirs
- 1 barregraphe 10 LEDs⁶ (en option)
- 1 afficheur numérique 7 ou 8 segments à LEDs⁷
- 1 fer à souder et de l'étain (si Diduino)

Note: sauf avis contraire, dans les montages, une résistance de 220Ω peut être remplacée par une de 470Ω , et une de $1k\Omega$ par une autre de $10k\Omega$.

Le Diduino et tout le matériel nécessaire peuvent être commandés à cette adresse:

- <http://www.didel.com>

Il est aussi possible de commander du matériel Arduino et électronique à ces adresses:

Suisse:

- <http://www.play-zone.ch>
- <http://shop.boxtec.ch> (point de vente Diduino, y compris pour l'Europe)
- <http://www.conrad.ch>

France & Belgique:

- <http://mchobby.be>
- <http://www.conrad.fr>
- <http://www.zartronic.fr>
- <http://boutique.semageek.com/fr/>

Dans un cadre scolaire, afin de limiter les coûts, l'enseignant peut donner ou mettre à disposition un kit Diduino par élève (40 CHF). Il met ensuite à disposition les différents composants nécessaires: LEDs, résistances, câbles, affichage 7 segments, platines d'expérimentations.

⁶ Par exemple, Kingbright DC-10EWA ou DC7G3EWA, disponibles ici: <http://00.lc/gy> ou ici: <http://00.lc/gz>.

⁷ Par exemple, le Kingbright SA56-11 SRWA, disponible ici: Disponible ici: <http://00.lc/hp>



Installer Arduino sur Mac⁸: aide-mémoire

Pilote

Afin de pouvoir connecter un Diduino sur un Mac, il faut au préalable installer un pilote série-USB.

La dernière version du pilote se trouve ici: <http://www.ftdichip.com/Drivers/VCP.htm>

Currently Supported VCP Drivers:

Operating System	Release Date	Processor Architecture								Comments
		x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4		
Windows*	2011-04-12	2.08.14	2.08.14	-	-	-	-	-	2.08.14 WHQL Certified Available as setup executable Release Notes	
	2011-08-26	2.08.17(Beta)	2.08.17(Beta)	-	-	-	-	-	2.08.17 Beta Version Release Notes	
Linux	2009-05-14	1.5.0	1.5.0	-	-	-	-	-	Included in 2.6.31 kernel and later ReadMe	
Mac OS X	2011-02-28	2.2.16	2.2.16	2.2.16	-	-	-	-	Customers wishing to have a VID/PID combination added should contact FTDI Support	
Windows CE 4.2-5.2**	2012-01-06	1.1.0.10	-	-	1.1.0.10	1.1.0.10	1.1.0.10	1.1.0.10		
Windows CE 6.0	2012-01-06	1.1.0.10	-	-	1.1.0.10	1.1.0.10	1.1.0.10	1.1.0.10		

- ✿ Mac doté d'un processeur PowerPC (G4, G5):
 - ✿ choisir **PPC**
- ✿ Mac doté d'un processeur Intel:
 - ✿ 10.3 à 10.5: choisir **x86 (32-bit)**
 - ✿ 10.6 et 10.7: choisir **x86 (64-bit)**

Deux pilotes sont fournis: l'un pour Mac OS 10.3 et l'autre pour Mac OS 10.4 à 10.6. Ce dernier fonctionne aussi sous Mac OS 10.7 Lion.



FTDIUSBSerialDriver_10_3



FTDIUSBSerialDriver_10_4_10_5_10_6

Installation du pilote

Décompresser l'archive et ouvrir le pilote choisi. Suivre ensuite le processus d'installation standard.

⁸ Pour installer le logiciel sur PC: <http://arduino.cc/en/Main/Software>



Installation du logiciel Arduino

Télécharger le logiciel Arduino depuis cette page: <http://arduino.cc/en/Main/Software>

Une fois l'archive décompressée, glisser le logiciel dans les *applications*.

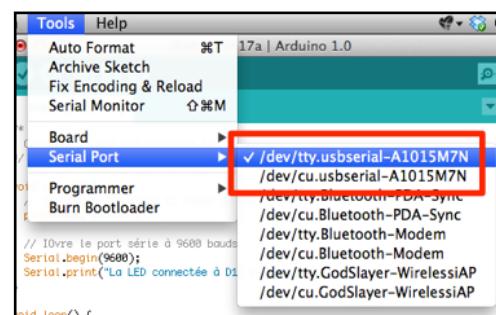


Configuration du logiciel Arduino

Sélection du port série

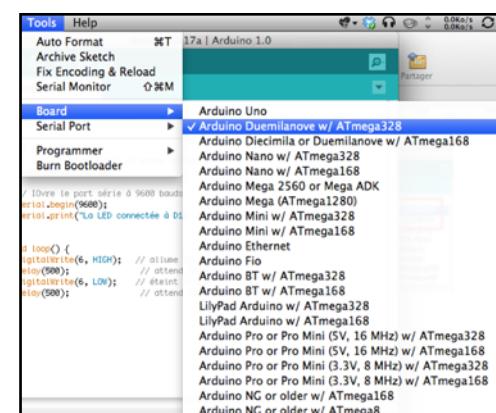
Aller dans le menu *Tools/Serial Port* et sélectionner à choix */dev/tty.usbserial-A1015M7N* ou */dev/cu.usbserial-A1015M7N*.

Selon l'usage, utiliser celui qui fonctionne le mieux.



Sélection du type de carte Arduino

Aller dans le menu *Tools/Board* et sélectionner *Arduino Duemilanove w/ ATmega328*



Aide

Erreur *programmer is not responding* lors de l'envoi du programme (*upload*) sur le robot: appuyer sur le bouton *reset* du robot avant chaque envoi du programme. Débrancher et rebrancher le câble USB produit le même effet.



Sources

- <http://www.didel.com/diduino/Start.pdf>
- <http://arduino.cc/en/Guide/MacOSX>
- http://fr.flossmanuals.net/_booki/arduino/arduino.pdf
- <http://myarduino.blogspot.com>



La gestion de classe

Ce cours est prévu pour l'enseignement de l'informatique et de l'électronique sur Arduino pour des élèves de 14 à 16 ans à l'école. En Suisse Romande, cela correspond aux classes 10 et 11 Harmos.

Devant la richesse du sujet, il est tout à fait possible de travailler avec ce cours sur une année complète, avec une dotation hebdomadaire de deux périodes de 45 minutes.

Gérer le matériel

Lorsque cela est possible, et devant le faible prix du kit Diduino, l'idéal est d'en offrir (ou d'en faire acheter) un par élève. Cela apporte quelques avantages, comme l'initiation à la soudure, le respect du matériel, la possibilité pour l'élève de travailler à la maison...

L'enseignant devrait avoir une réserve de petit matériel électronique (câbles, LEDs, résistances,...) à disposition des élèves, afin de compléter le kit. L'avantage est que ce matériel est lui aussi très avantageux.

Pour ce qui est des outils, outre le matériel de soudure, l'enseignant devrait disposer de quelques tournevis et pinces dédiés à l'électronique, ainsi que d'un multimètre. L'idéal étant d'en choisir un avec la fonction *autorange*, qui permet d'éviter de se soucier si on mesure des millivolts ou des centaines de volts. Enfin, quelques lampes de bureau équipées de loupes sont un vrai atout.⁹

Réaliser des projets

L'Arduino permet non seulement de facilement s'initier à l'électronique et à la programmation, mais aussi de réaliser des projets concrets. Il suffit souvent de quelques LEDs ou de quelques relais pour réaliser des choses étonnantes.

Par exemple, avec les élèves, nous avons piraté le sapin de Noël du Directeur de l'école¹⁰! Cela permet aux élèves de mobiliser et de mettre en valeur leurs compétences dans des projets concrets. Un enseignant aurait tort de se passer de cet excellent facteur de motivation, en particulier s'il fait un projet visible et qu'il met en avant les compétences des élèves.

⁹ Par exemple: <http://www.edurobot.ch/?p=1417>

¹⁰ Projet à voir ici: <http://www.edurobot.ch/?p=1251>



LEÇON 1: Découverte de la plateforme Arduino

Qu'est-ce qu'un ordinateur? La réponse la plus évidente est sans doute de dire qu'un ordinateur, c'est ceci:



Source: apple.com

Mais est-ce aussi simple? Car un ordinateur, c'est aussi, cela:



Source: arduino.cc

Écris **ta** définition du mot *ordinateur*:

.....
.....

Cite des **objets** qui contiennent des ordinateurs:

.....
.....



Arduino ou Diduino?

Arduino est un projet *open source*. Cela signifie que tout le monde a accès librement aux plans, au logiciel (et à son *code source*), ainsi qu'au langage de programmation. Tout le monde peut donc modifier l'Arduino, créer des accessoires ou même faire des copies de l'Arduino et les vendre.

Les cartes originelles sont vendues par Arduino. Mais de nombreux autres fabricants en commercialisent, plus ou moins proches des originaux. C'est le cas de l'entreprise Didel¹¹, qui commercialise le Diduino.

Nous avons choisi de travailler avec le Diduino pour plusieurs raisons:

- Didel est une entreprise locale. Cela compte aussi.
- Le Diduino possède une platine d'expérimentation (*breadboard*) intégrée, ce qui facilite son utilisation en classe
- Le Diduino possède des connecteurs pour les Kidules¹², ce qui permet d'ajouter de nombreuses fonctions et de réaliser de nombreuses expériences.
- Le Diduino doit en partie être monté et soudé. C'est un excellent travail pour les élèves.

Ce cours reste néanmoins parfaitement utilisable avec un Arduino ou un compatible Arduino

Cherche et note les définitions suivantes:

Open-source

.....
.....
.....

Code source

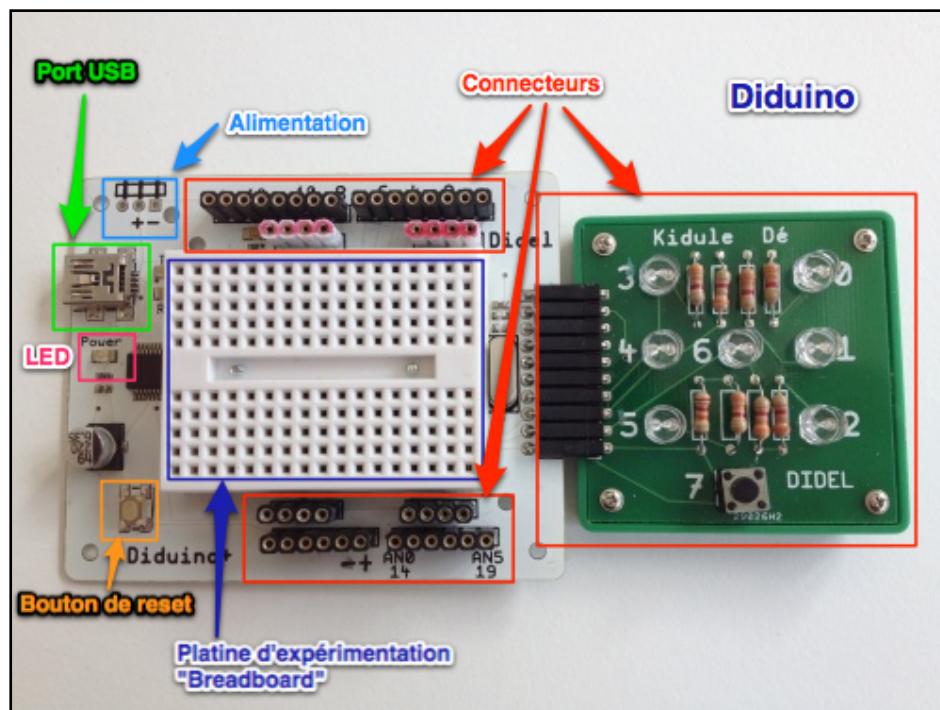
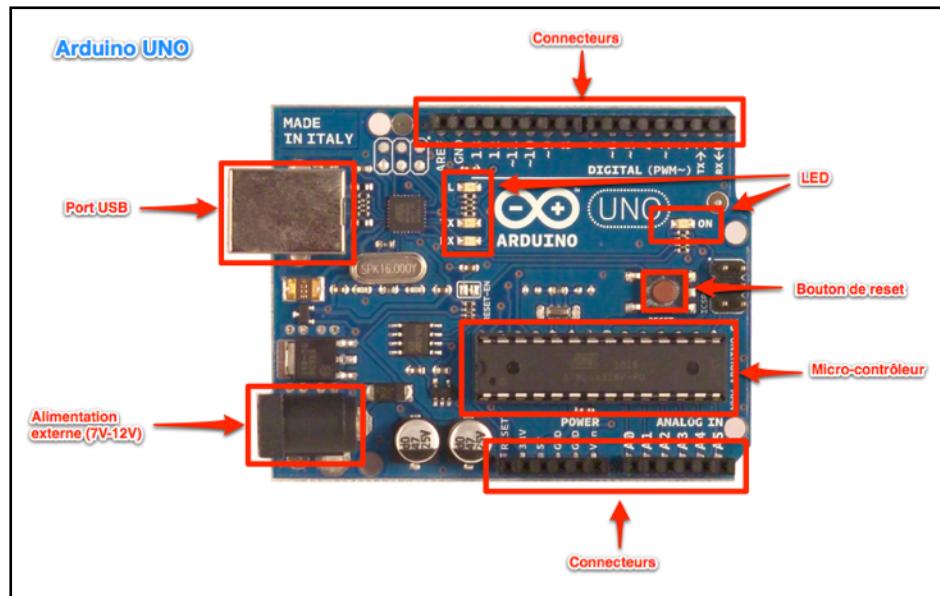
.....
.....

¹¹ <http://www.didel.com>

¹² <http://www.didel.com/KidulesPub.pdf>



Constitution de la carte





Le microcontrôleur

Voilà le cerveau de notre carte. C'est lui qui va recevoir le programme que nous allons créer et qui va le stocker dans sa mémoire puis l'exécuter. Grâce à ce programme, il va savoir faire des choses, qui peuvent être : faire clignoter une LED, afficher des caractères sur un écran, envoyer des données à un ordinateur...

Sur le Diduino, le processeur est caché sous la platine d'expérimentation.

Avant de fixer la platine d'expérimentation sur ton Diduino, fais un schéma aussi précis que possible de la carte et entoure en rouge le microcontrôleur.

Alimentation

Pour fonctionner, une carte Arduino a besoin d'une alimentation. Le microcontrôleur fonctionnant sous 5V, la carte peut être alimentée en 5V par le port USB ou bien par une alimentation externe qui est comprise entre 7V et 12V. Un régulateur se charge ensuite de réduire la tension à 5V pour le bon fonctionnement de la carte. Pas de danger de tout griller donc! Veuillez seulement à respecter l'intervalle de 7V à 15V (même si le régulateur peut supporter plus, pas la peine de le retrancher dans ses limites).

Attention: ne surtout pas mettre de plus de 6V sur la carte Diduino. Il n'y a pas de régulateur et donc pas de connecteur pour une alimentation extérieure. On peut mettre un bloc de 3 ou 4 accus (ou piles). **La tension acceptée va de 3 à 6V.**

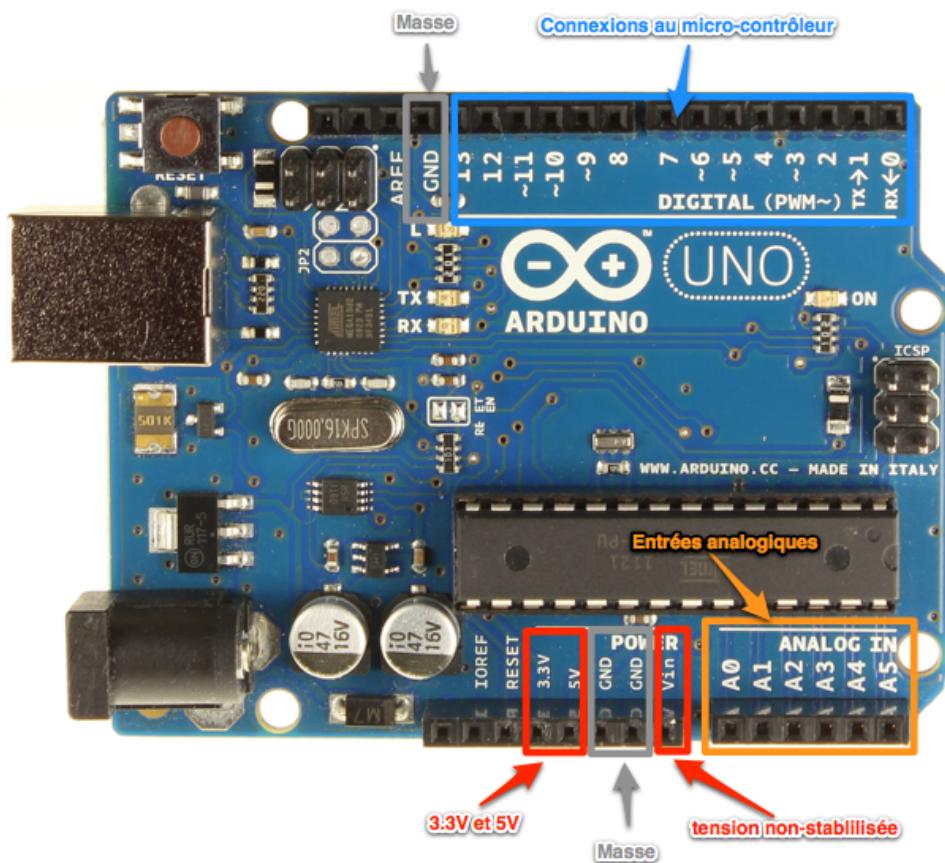
Attention (oui, encore!): les nouvelles cartes Arduino Due fonctionnent avec un voltage de 3.3 V! Dans ce cours, nous partons du principe que le voltage des montages est en 5 Volts.

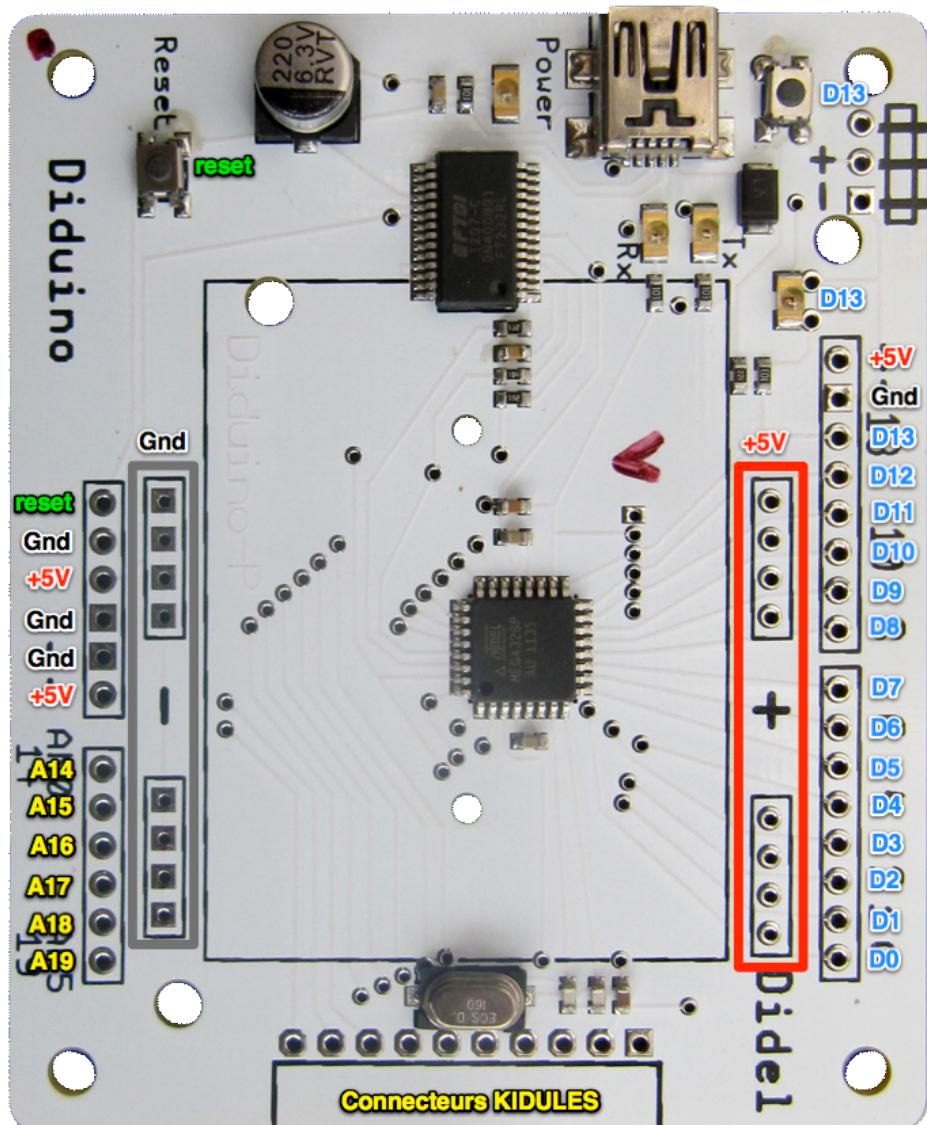


La connectique

La carte Arduino ne possédant pas de composants (résistances, diodes, moteurs...) qui peuvent être utilisés pour un programme, il est nécessaire de les rajouter. Mais pour ce faire, il faut les connecter à la carte. C'est là qu'interviennent les connecteurs de la carte.

Sur les Arduino et sur beaucoup de cartes compatibles Arduino, les connecteurs se trouvent au même endroit. Cela permet de fixer des cartes d'extension, appelée *shields*.





Sur Diduino, les sorties analogiques sont numérotées A14 à A19.

Sur Arduino, les sorties analogiques sont numérotées A0 à A5.



Connecteurs Diduino

- +5V:** tension +5V
- Gnd:** masse (tension 0V)
- D0-D13:** Connexions aux pattes du micro-contrôleur
- A14-A19:** Entrées analogiques pour capteurs
- Kidules:** Connecteurs pour Kidules

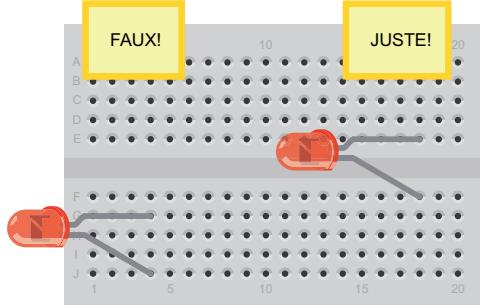
La platine d'expérimentation



La platine d'expérimentation est intégrée sur le Diduino.

Elle permet de connecter les composants entre eux et de créer des circuits. Il y a 17 rangées. Dans chaque rangée, il y a deux groupes de 5 colonnes, séparés par une tranchée au milieu.

Tous les connecteurs dans une rangée de 5 sont connectés entre eux. Donc si on branche deux éléments dans un groupe de 5 connecteurs, ils seront reliés entre eux.



Sur la plupart des autres cartes Arduino, il n'y a pas de breadboard. Il faut donc l'acquérir à part. Il en existe de plusieurs types et de plusieurs tailles.



Leçon 2: La soudure

Certains modules Arduino, comme le Diduino, nécessitent un petit travail de soudure.

Pour réaliser cette étape: nous avons besoin:

- D'un fer à souder, à panne¹³ fine
- De fil à souder, fin
- De tresse à dessouder, en cas d'erreur
- Éventuellement, de graisse à souder
- D'un support pour le module
- D'une pince coupante fine
- Apprendre à souder:
 - <http://www.didel.com/Soudure.pdf>
 - http://mightyohm.com/files/soldercomic/translations/Souder%20c'est%20facile_FR.pdf

Attention!



Le fer à souder va devenir très chaud! Attention aux brûlures!

Ne pas respirer les vapeurs de soudure! Toxique!

La soudure doit être réalisée aussi soigneusement que possible. Il faut prendre son temps et être soigneux.

¹³ La panne est la pointe du fer à souder



Leçon 3: Les bases de l'électronique

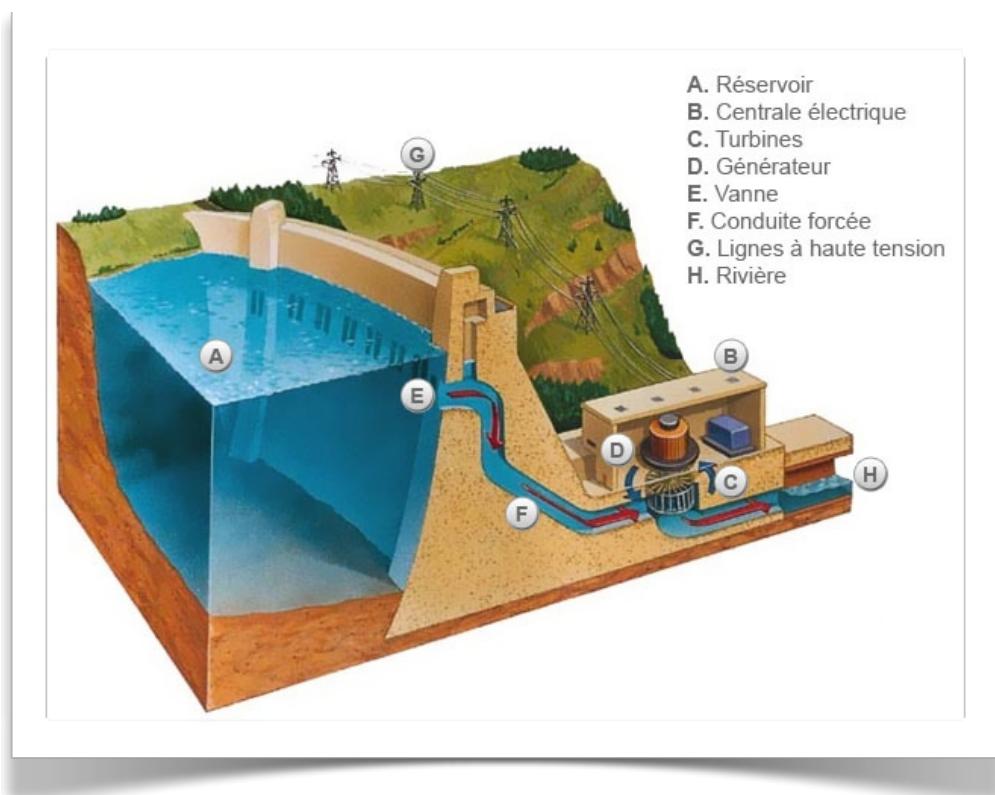
L'électricité

L'électricité est un déplacement d'électrons dans un milieu conducteur.

Pour que ces électrons se déplacent tous dans un même sens, il faut qu'il y ait une différence du nombre d'électrons entre les deux extrémités du circuit électrique.

Pour maintenir cette différence du nombre d'électrons, on utilise un générateur (pile, accumulateur, alternateur...)

La différence de quantité d'électrons entre deux parties d'un circuit s'appelle la **différence de potentiel** et elle se mesure en **Volts (V)**.



On peut comparer le fonctionnement d'un circuit électrique à celui d'un barrage hydroélectrique:

- Les électrons seraient l'eau
- Le générateur serait le réservoir d'eau
- Les conducteurs sont naturellement les conduites forcées
- Le consommateur (une ampoule ou une diode, par exemple) est la turbine, qui exploite l'énergie du déplacement des électrons



Sur un barrage, plus la différence entre l'altitude du niveau du réservoir et celui de la turbine est importante, plus la pression de l'eau sera importante. Pour un barrage on appelle cette différence d'altitude *hauteur de chute*. Cela équivaut sur un circuit électrique à la *différence de potentiel*, qui se mesure en *Volts* (V) et se note U .

Le *débit de l'eau* (=la quantité d'eau qui s'écoule par unité de temps) correspond à *l'intensité*, aussi appelée *courant*, qui est donc le débit d'électrons. Elle se mesure en *Ampères* (A) et se note I .

La puissance électrique se note P et se mesure en *Watts* (W). Elle exprime la quantité de courant (I), transformée en chaleur ou en mouvement. Sur un barrage, elle correspond à l'énergie produite par la turbine.

La puissance P est le produit de la tension U et du courant I

$$P=U \cdot I$$

Complète les expressions suivantes:

$$U =$$

$$I =$$

Pour une découverte ludique de l'électricité, je vous conseille de regarder le *C'est pas sorcier* consacré à ce thème:

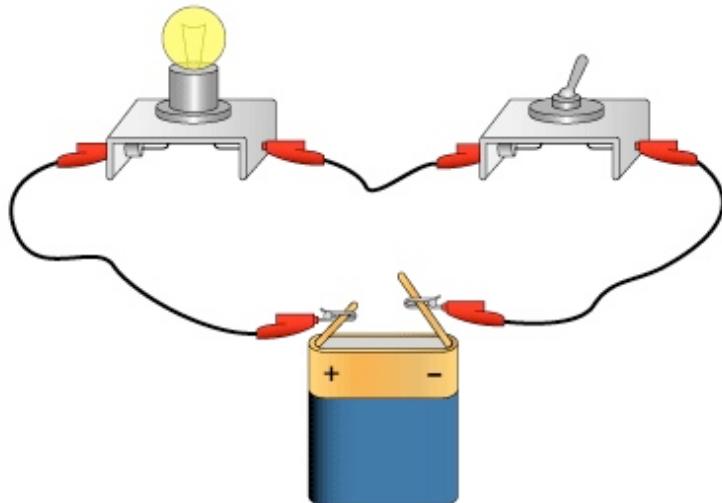
Partie 1: <http://00.lc/cl>

Partie 2: <http://00.lc/cm>



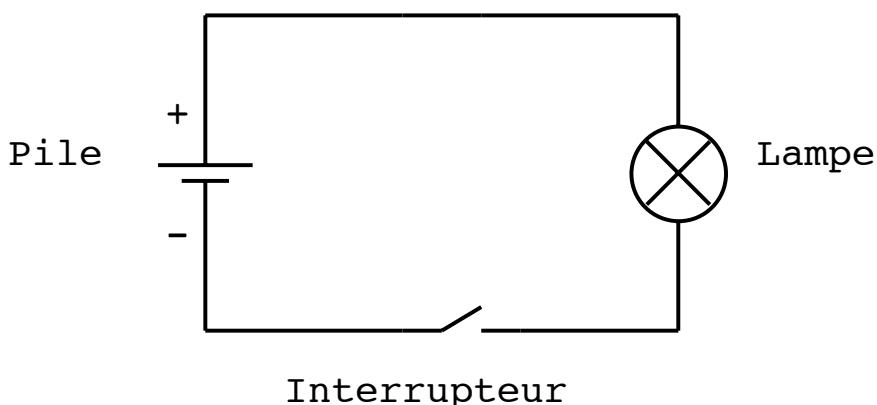
Le circuit électrique

Ainsi, une pile est constituée d'un milieu contenant de nombreux électrons en trop, et d'un second milieu en manque d'électrons. Quand on relie les deux pôles de la pile (le + et le -) avec un fil électrique (le conducteur), les électrons vont alors se déplacer du milieu riche vers le milieu pauvre. Si on intercale une lampe électrique, le passage des électrons va générer de la lumière.



Circuit électrique

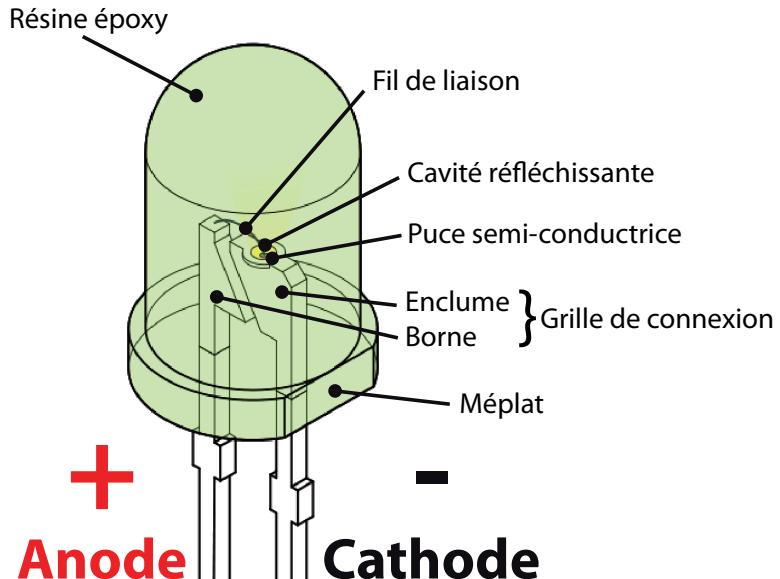
Voici le schéma électrique du circuit ci-dessus:





Les diodes

Il est possible de remplacer l'ampoule par une diode électroluminescente, aussi appelée LED¹⁴. Elle a la particularité de ne laisser passer le courant électrique que dans un sens.



Le courant électrique ne peut traverser la diode que dans le sens de l'anode vers la cathode.

On reconnaît l'anode, car il s'agit de la patte la plus longue. Lorsque les deux pattes sont de même longueur, on peut distinguer l'anode de la cathode, par un méplat du côté de cette dernière.

Le symbole de la LED est le suivant:



Attention: le courant produit par l'Arduino est trop important pour y brancher directement une LED dessus. L'utilisation d'une résistance est obligatoire, pour ne pas endommager la LED.

¹⁴ http://fr.wikipedia.org/wiki/Diode_électroluminescente



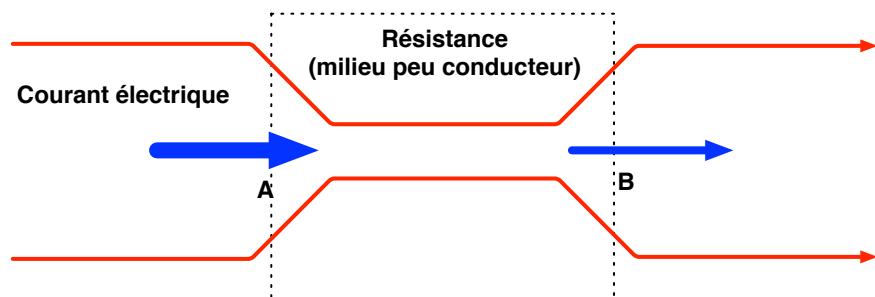
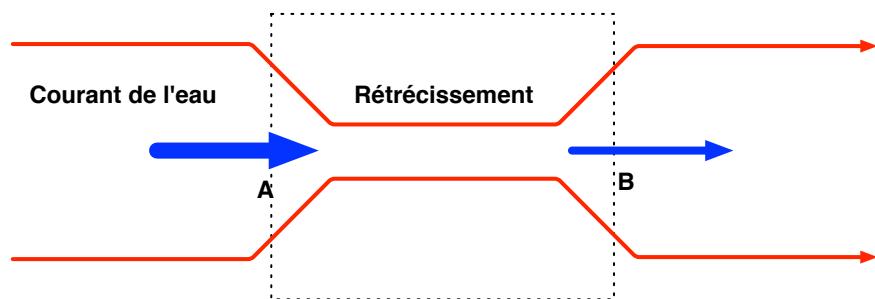


Les résistances

Une **résistance** est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (mesurée en ohms: Ω) à la circulation du courant électrique.



On peut alors comparer, le débit d'eau au courant électrique **I** (qui est d'ailleurs le débit d'électrons), la différence de pression à la différence de potentiel électrique (qui est la tension **U**) et, enfin, le rétrécissement à la résistance **R**.



Ainsi, pour une tension fixe, plus la résistance est faible, plus le courant la traversant est fort. Cette proportion est vérifiée par la loi d'Ohm:

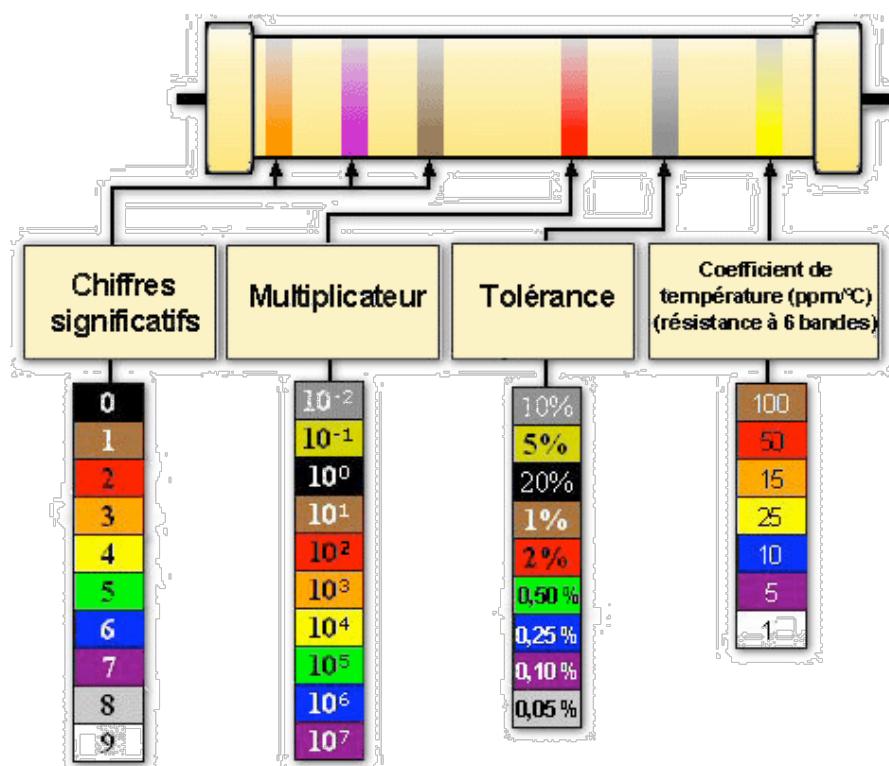
$$U=R \cdot I \text{ et donc } R=\frac{U}{I} \text{ et } I=\frac{U}{R}$$



Une résistance est un milieu peu conducteur; les électrons peinent à s'y déplacer. Leur énergie se dissipe alors en général sous forme de chaleur. C'est ce principe utilisé pour les bouilloires électriques ou les ampoules à filaments.

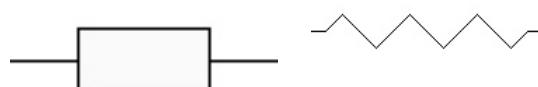


La valeur de la résistance se mesure en Ohms (Ω). La valeur d'une résistance est déterminée par ses bandes de couleurs.



Outre le tableau ci-dessus, on peut s'aider du petit mode d'emploi qui se trouve ici:
<http://www.apprendre-en-ligne.net/crypto/passecret/resistances.pdf>

La résistance est schématisée de ces deux manières:



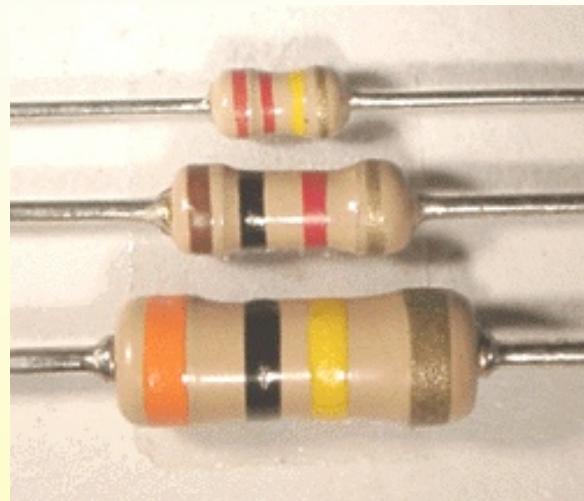
Un calculateur de valeurs de résistances est disponible à cette adresse:



<http://edurobot.ch/resistance/>



Quelle est la valeur de ces résistances?



Et celle-ci?

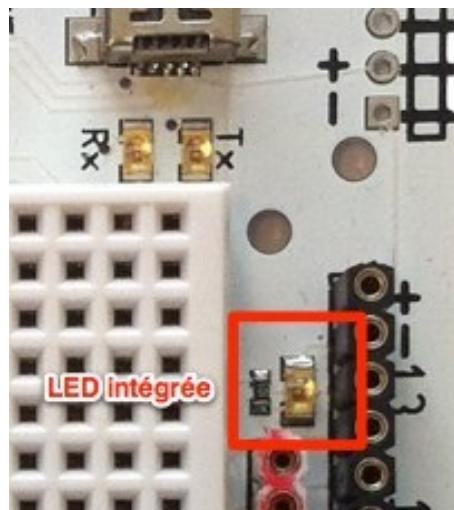




Leçon 4: faire clignoter la LED

Introduction

Le Diduino possède une LED montée sur le port 13.

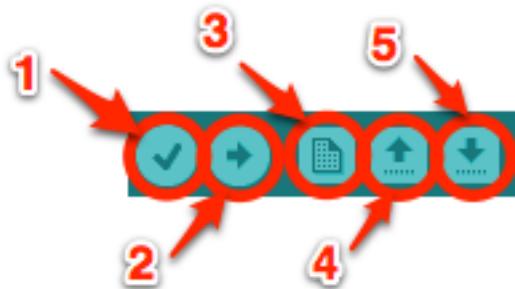


Comme premier programme, nous allons la faire clignoter. La programmation se fait dans le logiciel Arduino.



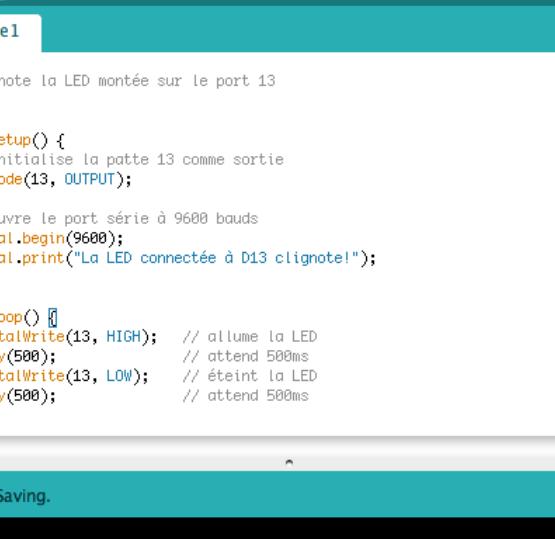


Le menu



- 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

Commençons tout de suite par un petit code. Nous l'analyserons ensuite:



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** "code1 | Arduino 1.0"
- Toolbar:** Standard icons for file operations (New, Open, Save, Upload, Download).
- Code Editor:** The main area contains the following Arduino sketch:

```
/*
  Clignote la LED montée sur le port 13
*/

void setup() {
  // Initialise la patte 13 comme sortie
  pinMode(13, OUTPUT);

  // Ouvre le port série à 9600 bauds
  Serial.begin(9600);
  Serial.print("La LED connectée à D13 clignote!");
}

void loop() {
  digitalWrite(13, HIGH);    // allume la LED
  delay(500);               // attend 500ms
  digitalWrite(13, LOW);     // éteint la LED
  delay(500);               // attend 500ms
}
```
- Status Bar:** "Done Saving."
- Bottom Status:** "Arduino Duemilanove w/ ATmega328 on /dev/cu.usbserial-A501D4YZ"



La LED

Code 1

```
/*
  Code 1 - Edurobot.ch, destiné au Diduino/Arduino
  Objectif: faire clignoter la LED montée sur le port 13
*/



//***** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et une seule fois, au démarrage
du programme

void setup()          // début de la fonction setup()
{
    pinMode(13, OUTPUT);      // Initialise la patte 13 comme sortie
    Serial.begin(9600);      // Ouvre le port série à 9600 bauds
}                      // fin de la fonction setup()

//***** FONCTION LOOP = Boucle sans fin = cœur du programme *****
// La fonction loop() s'exécute sans fin en boucle aussi longtemps que
l'Arduino est sous tension

void loop()           // début de la fonction loop()
{
    digitalWrite(13, HIGH);   // Met la patte 13 au niveau haut = allume la LED
    delay(500);             // Pause de 500ms
    digitalWrite(13, LOW);    // Met la patte 13 au niveau bas = éteint la LED
    delay(500);             // Pause 500ms
}                      // fin de la fonction setup()
```

Télécharger le code: <http://www.edurobot.ch/code/code1.txt>

Une fois le code écrit (ou collé) dans la fenêtre de programmation, il faut l'envoyer sur le Diduino. Pour cela, il faut cliquer sur le bouton *upload (téléverser)*, naturellement après avoir connecté le Diduino à l'ordinateur!



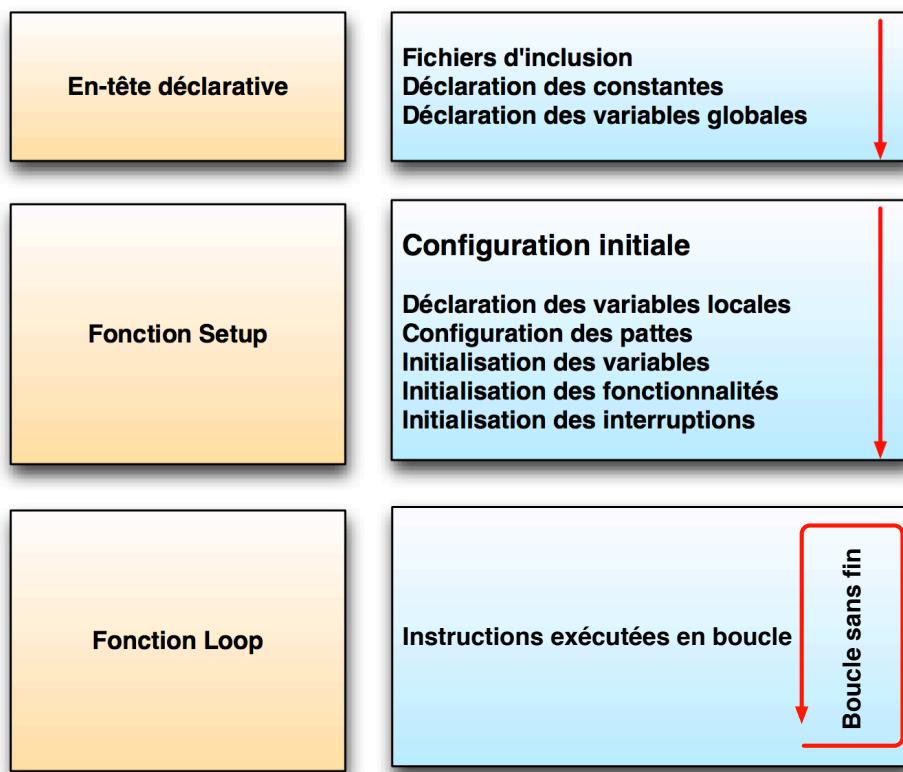
Introduction au code

Le déroulement du programme

Le programme se déroule de la façon suivante :

1. Prise en compte des instructions de la partie déclarative
2. Exécution de la partie configuration (*fonction setup()*),
3. Exécution de la boucle sans fin (*fonction loop()*): le code compris dans la boucle sans fin est exécuté indéfiniment.

Déroulement du programme



Nous verrons petit à petit les divers éléments présents dans le schéma. L'important pour le moment est de savoir qu'un programme est divisé en trois parties: en-tête déclarative, fonction setup et fonction loop.

La suite va nous permettre d'entrer dans le code minimal: les fonctions setup et loop.



Le code minimal¹⁵

Avec Arduino, nous devons utiliser un code minimal lorsque l'on crée un programme. Ce code permet de diviser le programme que nous allons créer en deux grosses parties.

```
void setup()      //fonction d'initialisation de la carte
{
    //contenu de l'initialisation
}

void loop()       //fonction principale, elle se répète (s'exécute) à l'infini
{
    //contenu du programme
}
```

Nous avons donc devant nous le code minimal qu'il faut insérer dans notre programme. Mais que peut-il bien signifier pour quelqu'un qui n'a jamais programmé ?

La fonction

Dans ce code se trouvent deux fonctions. Les fonctions sont en fait des portions de code.

```
void setup()      //fonction d'initialisation de la carte
{
    //contenu de l'initialisation
    //on écrit le code à l'intérieur
}
```

Cette fonction **setup()** est appelée **une seule fois** lorsque le programme commence. C'est pourquoi c'est dans cette fonction que l'on va écrire le code qui n'a besoin d'être exécuté qu'une seule fois. On appelle cette fonction : "*fonction d'initialisation*". On y retrouvera la mise en place des différentes sorties et quelques autres réglages. C'est un peu le check-up de démarrage. Imaginons un pilote d'avion dans sa cabine qui fait l'inventaire :

- patte 2 en sortie, état haut ?
- OK
- timer 3 à 15 millisecondes ?
- OK

¹⁵ Source de cette partie: <http://sciences.siteduzero.com/tutoriel-3-515617-le-langage-arduino-1-2.html>



Une fois que l'on a initialisé le programme, il faut ensuite créer son "cœur", autrement dit le programme en lui même.

```
void loop()      //fonction principale, elle se répète (s'exécute) à l'infini
{
    //contenu du programme
}
```

C'est donc dans cette fonction **loop()** que l'on va écrire le contenu du programme. Il faut savoir que cette fonction est appelée en permanence, c'est-à-dire qu'elle est exécutée une fois, puis lorsque son exécution est terminée, on la ré-exécute et encore et encore. On parle de *boucle infinie*.

Les instructions

Maintenant que nous avons vu la structure des fonctions, regardons ce qu'elles peuvent contenir.

Les instructions sont des lignes de code qui disent au programme : "fait ceci, fait cela..." C'est tout bête, mais très puissant, car c'est ce qui va orchestrer notre programme.

Les points virgules ;

Les points virgules terminent les instructions. Si par exemple je dis dans mon programme : "*appelle la fonction couperDuSaucisson*" je dois mettre un point virgule après l'appel de cette fonction.

Les points virgules (;) sont synonymes d'erreurs, car il arrive très souvent de les oublier à la fin des instructions. Par conséquent le code ne fonctionne pas. Il faut donc être très attentif à ne pas oublier de point virgule!

Les accolades { }

Les accolades sont les "*conteneurs*" du code du programme. Elles sont propres aux fonctions, aux conditions et aux boucles. Les instructions du programme sont écrites à l'intérieur de ces accolades.

Pour ouvrir une accolade sur Mac, taper alt-8 et alt-9 pour la fermer.



Les commentaires

Les commentaires sont des lignes de codes qui seront ignorées par le programme. Elles ne servent en rien lors de l'exécution du programme.

Mais alors c'est inutile ?

Non, car cela va nous permettre à nous et aux programmeurs qui liront notre code (s'il y en a) de savoir ce que signifie la ligne de code que nous avons écrite. C'est très important de mettre des commentaires et cela permet aussi de reprendre un programme laissé dans l'oubli plus facilement !

Ligne unique de commentaire :

```
//cette ligne est un commentaire sur UNE SEULE ligne
```

Ligne ou paragraphe sur plusieurs lignes :

```
/*cette ligne est un commentaire, sur PLUSIEURS lignes  
qui sera ignoré par le programme, mais pas par celui qui lit le code ;) */
```

Les accents

Il est formellement interdit de mettre des accents en programmation! Sauf dans les commentaires.



Analyse du code 1¹⁶

Revenons maintenant à notre code.

```

/*
  Code 1 - Edurobot.ch, destiné au Diduino
  Objectif: faire clignoter la LED montée sur le port 13
*/
void setup() {
  // Initialise la patte 13 comme sortie
  pinMode(13, OUTPUT);
  // Ouvre le port série à 9600 bauds
  Serial.begin(9600);
  Serial.print("La LED connectée à D13 clignote!");
}
void loop() {
  digitalWrite(13, HIGH); // allume la LED
  delay(500);
  digitalWrite(13, LOW); // éteint la LED
  delay(500);
}

Enregistrement terminé.
Taille binaire du croquis : 2 418 octets (d'un max de 30 720 octets)

```

- La ligne `pinMode(13, OUTPUT);` initialise la patte 13 du microcontrôleur comme sortie, c'est-à-dire que des données seront envoyées depuis le microcontrôleur vers cette patte.
- La ligne `Serial.begin(9600);` initialise le port série qui permet au robot d'envoyer et de recevoir des informations à l'ordinateur.
- Avec l'instruction `digitalWrite(13, HIGH);`, le microcontrôleur connecte la patte D13 au **+5V** ce qui a pour effet d'allumer la LED.
- L'instruction `delay(500);` indique au microcontrôleur de ne rien faire pendant 500 millisecondes, soit $\frac{1}{2}$ seconde.
- Avec l'instruction `digitalWrite(13, LOW);`, le microcontrôleur connecte la patte D13 à la masse (Gnd) ce qui a pour effet d'éteindre la LED.
- L'instruction `delay(500);` indique au microcontrôleur à nouveau de ne rien faire pendant 500ms soit $\frac{1}{2}$ seconde.
- Le résultat est donc que la LED s'allume pendant $\frac{1}{2}$ seconde, puis s'éteint pendant une $\frac{1}{2}$ seconde puis s'allume pendant $\frac{1}{2}$ seconde... elle clignote donc.

¹⁶ Source: <http://00.lc/ce>

**À toi de jouer!**

Fais varier les valeurs de l'instruction *delay*. Indique ci-dessous la valeur choisie et le résultat obtenu:

Essai 1:

```
digitalWrite(13, HIGH);  
delay(.....);  
digitalWrite(13, LOW);  
delay(.....);
```

Résultat: _____

Essai 2:

```
digitalWrite(13, HIGH);  
delay(.....);  
digitalWrite(13, LOW);  
delay(.....);
```

Résultat: _____

Essai 3:

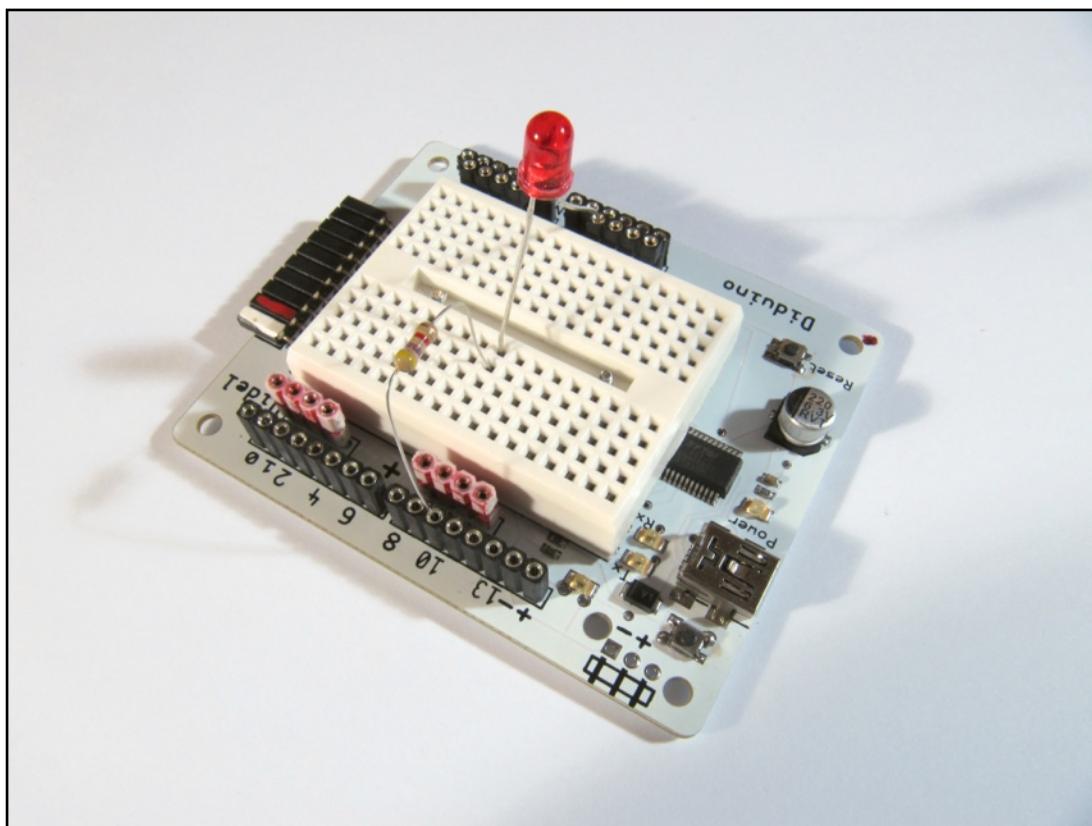
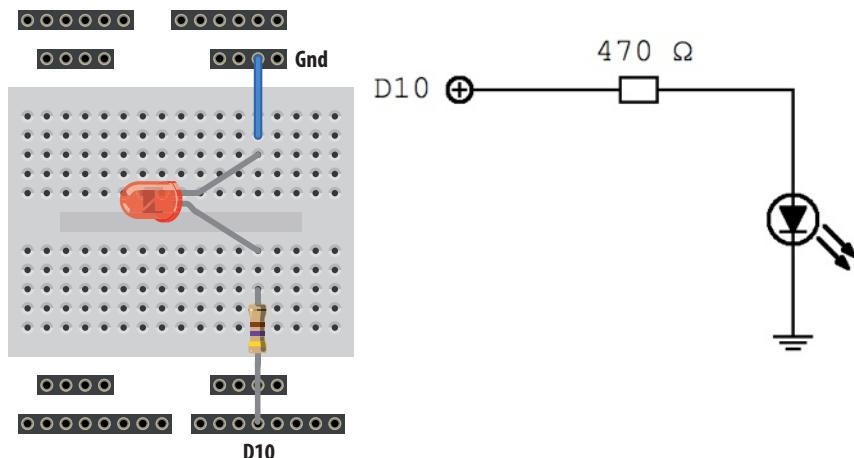
```
digitalWrite(13, HIGH);  
delay(.....);  
digitalWrite(13, LOW);  
delay(.....);
```

Résultat: _____



Monter une LED

Nous allons maintenant réaliser le schéma suivant:



Fais attention à ne pas te tromper de résistance et à bien monter la LED dans le bon sens.



Code 2

Modifie le code ci-dessous, afin de faire clignoter la LED sur la patte 13.

```
/*
  Code 2 - Edurobot.ch, destiné au Diduino
  Objectif: faire clignoter la LED montée sur la patte 13
*/

void setup() {
    // Initialise la patte 13 comme sortie
    pinMode(13, OUTPUT);

    // Ouvre le port série à 9600 bauds
    Serial.begin(9600);
}

void loop() {
    digitalWrite(13, HIGH);      // allume la LED
    delay(500);                // attend 500ms
    digitalWrite(13, LOW);       // éteint la LED
    delay(500);                // attend 500ms
}
```

Tu trouveras le corrigé ici: <http://edurobot.ch/code/code2.txt>

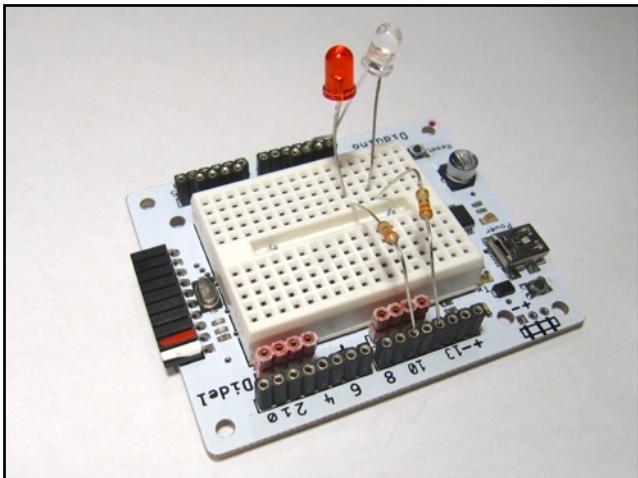
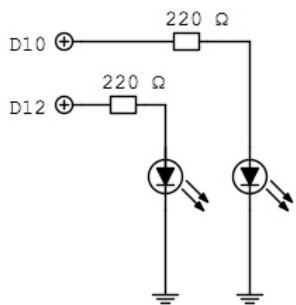
Remplace maintenant la résistance par une autre de 220 Ω. Que constates-tu, et comment l'expliques-tu?

Réalise le schéma électrique du montage que tu viens de réaliser.



Variante: faire clignoter deux LEDs

Réalise le montage suivant:



Exercice 1:

Modifie le code 2 pour faire clignoter les deux LEDs en même temps.

Corrigé: <http://www.edurobot.ch/code/code2-1.txt>

Exercice 2:

Modifie ton code pour faire clignoter les LEDs en alternance.

Corrigé: <http://www.edurobot.ch/code/code2-2.txt>



Leçon 5: les variables

Nous l'avons vu, dans un microcontrôleur, il y a plusieurs types de mémoire. Nous nous occuperons seulement de la mémoire "vive" (RAM) et de la mémoire "morte" (EEPROM).

Je vais vous poser un problème. Imaginons que vous avez connecté un bouton poussoir sur votre carte Arduino. Comment allez-vous stocker l'état du bouton (appuyé ou pas) ?

Une variable, qu'est ce que c'est ?

Une variable est un nombre. Ce nombre est stocké dans un espace de la mémoire vive (RAM) du microcontrôleur. La manière qui permet de les stocker est semblable à celle utilisée pour ranger des chaussures : dans un casier numéroté.

Ce nombre a la particularité de *changer de valeur*. Étrange n'est-ce pas ? Et bien pas tant que ça, car une variable est en fait le conteneur du nombre en question. Et ce conteneur va être stocké dans une case de la mémoire. Si on matérialise cette explication par un schéma, cela donnerait :

nombre → variable → mémoire

le symbole " \rightarrow " signifiant : "est contenu dans..."

Imagine que tu stockes ta variable dans un container. Chaque container est lui, déposé dans une case bien précise, afin de le retrouver.



Le nom d'une variable

Le nom de variable n'accepte que l'alphabet alphanumérique ([a-z], [A-Z], [0-9]) et _ (underscore)



Définir une variable

Si on donne un nombre à notre programme, il ne sait pas si c'est une variable ou pas. Il faut le lui indiquer. Pour cela, on donne un type aux variables. Oui, car il existe plusieurs types de variables ! Par exemple la variable "x" vaut 4 :

```
x = 4;
```

Et bien ce code ne fonctionnerait pas car il ne suffit pas ! En effet, il existe une multitude de nombres : les nombres entiers, les nombres décimaux, ... C'est pour cela qu'il faut assigner une variable à un type.

Voilà les types de variables les plus répandus

Type	Quel nombre il stocke ?	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 octets
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

Par exemple, si notre variable "x" ne prend que des valeurs décimales, on utilisera les types *int*, *long*, ou *char*. Si maintenant la variable "x" ne dépasse pas la valeur 64 ou 87, alors on utilisera le type *char*.

```
char x = 0;
```

Si en revanche x = 260, alors on utilisera le type supérieur (qui accepte une plus grande quantité de nombre) à *char*, autrement dit *int* ou *long*.

Mais t'es pas malin, pour éviter les dépassemens de valeur ont met tout dans des *double* ou *long* !

Oui, mais NON. Un microcontrôleur, ce n'est pas un ordinateur 2GHz multicore, 4Go de RAM ! Ici on parle d'un système qui fonctionne avec un CPU à 16MHz (soit 0,016 GHz) et 2 Ko de SRAM pour la mémoire vive. Donc deux raisons font qu'il faut choisir ses variables de manière judicieuse :

- La RAM n'est pas extensible, quand il y en a plus, y en a plus !



Le processeur est de type 8 bits (sur Arduino UNO), donc il est optimisé pour faire des traitements sur des variables de taille 8 bits, un traitement sur une variable 32 bits prendra donc (beaucoup) plus de temps !

Si à présent notre variable "x" ne prend jamais une valeur négative (-20, -78, ...), alors on utilisera un type *non-signé*. C'est à dire, dans notre cas, un *char* dont la valeur n'est plus de -128 à +127, mais de 0 à 255.

Voici le tableau des types non signés, on repère ces types par le mot *unsigned* (de l'anglais : non-signé) qui les précède :

Type	Quel nombre il stocke ?	Valeurs maximales du nombre stocké	Nombre sur X bits	Nombre d'octets
<code>unsigned char</code>	entier non négatif	0 à 255	8 bits	1 octets
<code>unsigned int</code>	entier non négatif	0 à 65 535	16 bits	2 octets
<code>unsigned long</code>	entier non négatif	0 à 4 294 967 295	32 bits	4 octets

Une des particularités du langage Arduino est qu'il accepte un nombre plus important de types de variables, listées dans ce tableau:

Type	Quel nombre il stocke ?	Valeurs maximales du nombre stocké	Nombre sur X bits	Nombre d'octets
<code>byte</code>	entier non négatif	0 à 255	8 bits	1 octets
<code>word</code>	entier non négatif	0 à 65535	16 bits	2 octets
<code>boolean</code>	entier non négatif	0 à 1	1 bits	1 octets

L'incrémentation

Il est possible d'appliquer à des variables diverses opérations mathématiques. Commençons tout de suite par un petit exemple: *l'incrémentation*. Il s'agit simplement d'additionner 1 à la variable. A chaque fois qu'on répète le code, on ajoute 1 à la variable.

Cela se fait grâce à ce code (*var* étant une variable à choix):

```
var++;
```

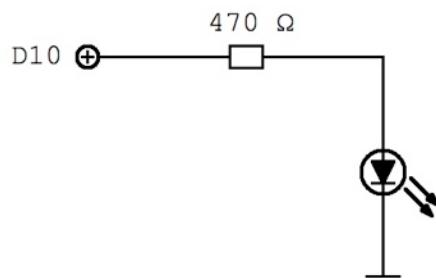
`var++;` revient à écrire : "var = var + 1;"



Faire clignoter une LED 10 fois

Code 3

L'objectif est de faire clignoter une LED 10 fois. Le montage est toujours le même.



```
/*
Code 3 - Edurobot.ch, destiné au Diduino/Arduino
Objectif: faire clignoter 10 fois la LED montée
sur le port 10
*/
```

```
//***** EN-TETE DECLARATIVE *****
// On déclare les variables, les constantes, ...

byte compteur; //On définit la variable "compteur"

//***** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et une seule fois, au démarrage
du programme

void setup()
{
    pinMode(10, OUTPUT); // Initialise la patte 10 comme sortie
    Serial.begin(9600); // Ouvre le port série à 9600 bauds

    // Exécute le programme entre accolades en partant de zéro et en incrémentant à
    // chaque fois la valeur de +1: 0+1/2+1/3+1... jusqu'à ce que la variable
    // "compteur" soit égale à 10.

    for(compteur=0 ; compteur<10 ; compteur++)
    {
        // début du programme exécuté 10 fois
        digitalWrite(10, HIGH); // allume la LED
        delay(500); // attend 500ms
        digitalWrite(10, LOW); // éteint la LED
        delay(500); // attend 500ms
    } // fin du programme exécuté 10 fois
}

void loop() { // vide, car programme déjà exécuté dans setup
}
```

Télécharger le code: <http://edurobot.ch/code/code3.txt>



Analyse du code¹⁷

La ligne ***byte compteur;*** permet de créer une variable *compteur*. *Byte* indique le type de la variable, c'est-à-dire le type de données que l'on pourra stocker dans cette variable. Comme nous l'avons déjà vu, le type byte permet de stocker des valeurs comprises entre 0 et 255.

La ligne ***for(compteur=0 ; compteur<10 ; compteur++) { instructions }*** sert à faire varier la variable *compteur* de 0 à 9 (en l'augmentant à chaque fois de 1: c'est ce que fait l'instruction ***compteur++***) et à chaque fois exécute toutes les instructions jusqu'à l'accolade ***}***. Cela va donc avoir l'effet de répéter le code suivant 10 fois:

```
digitalWrite(12, HIGH);    // allume la LED
delay(500);               // attend 500ms
digitalWrite(12, LOW);     // éteint la LED
delay(500);               // attend 500ms
```

Que se passe-t-il si tu remplaces la ligne ***for(compteur=0 ; compteur<10 ; compteur++)*** en ***for(compteur=0 ; compteur<20 ; compteur++)*** ?

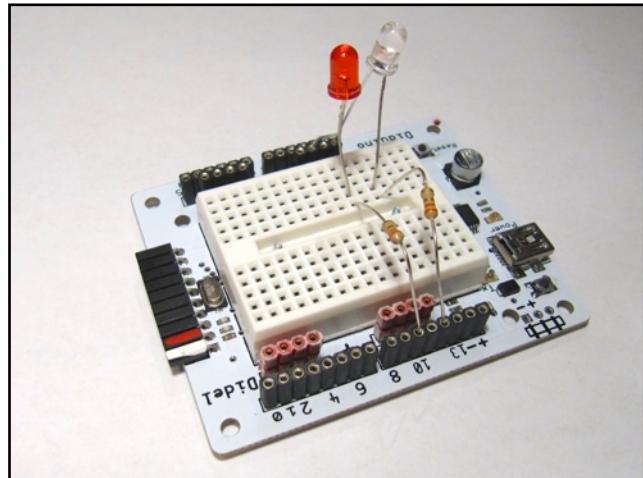
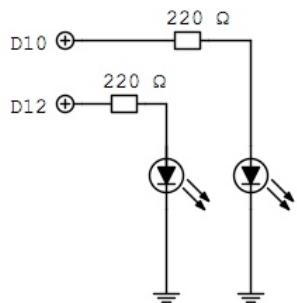
Que se passe-t-il si tu remplaces la ligne ***for(compteur=0 ; compteur<10 ; compteur++)*** en ***for(compteur=15 ; compteur<20 ; compteur++)*** ?

¹⁷ Source: http://mediawiki.e-apprendre.net/index.php/D%C3%A9couverte_des_microcontr%C3%B4leurs_avec_le_Diduino



Faire clignoter deux LED 20 fois

Réalise le montage suivant:



Modifie ton code pour que les deux LED clignotent en même temps vingt fois.

Tu trouveras le corrigé ici: <http://edurobot.ch/code/code4-1.txt>

Modifie maintenant le programme pour que les LED clignotent en alternance vingt fois.

Tu trouveras le corrigé ici: <http://edurobot.ch/code/code4-2.txt>

Comment faire pour augmenter la fréquence du clignotement?

Tu trouveras une vidéo d'exemple à cette adresse: <http://00.lc/cq>



Définir les pattes du microcontrôleur

Jusqu'à maintenant, nous avons identifié les pattes du microcontrôleur à l'aide de leurs numéros, comme dans l'exemple suivant: `pinMode(10, OUTPUT);`

Il est aussi possible de créer une *variable* définissant la broche utilisée, ensuite, définir si la patte utilisée doit être une entrée du microcontrôleur ou une sortie.

Premièrement, donc, définissons la patte utilisée du microcontrôleur :

```
const int led_rouge = 2;      //définition de la patte 2 de la carte en tant que variable
```

Le terme *const* signifie que l'on définit la variable comme étant constante. Par conséquent, on change la nature de la variable qui devient alors constante.

Le terme *int* correspond à un type de variable. En définissant une variable de ce type, elle peut stocker un nombre allant de -2147483648 à +2147483647 ! Cela nous suffit amplement ! Ainsi, la patte 2 devient *led_rouge*.

Nous sommes donc en présence d'une variable, nommée *led_rouge*, qui est en fait une constante, qui peut prendre une valeur allant de -2147483648 à +2147483647. Dans notre cas, cette constante est assignée à 2. Le chiffre 2.

Lorsque le code sera compilé, le microcontrôleur saura ainsi que sur sa broche numéro 2, il y a un élément connecté.

Profitons maintenant pour voir ce que signifie le terme *Output* utilisé jusqu'à présent.

Il s'agit de préciser si cette patte est une entrée ou une sortie. Oui, car le microcontrôleur a la capacité d'utiliser certaines de ses pattes en entrée ou en sortie. C'est fabuleux ! En effet, il suffit simplement d'interchanger UNE ligne de code pour dire qu'il faut utiliser une patte en entrée (récupération de donnée) ou en sortie (envoi de donnée).

Cette ligne de code justement, parlons-en ! Elle doit se trouver dans la fonction *setup()*.

La fonction se trouve être *pinMode()*. Pour utiliser cette fonction, il faut lui envoyer deux paramètres :

- Le nom de la variable que l'on a défini à la broche
- Le type de broche que cela va être (entrée ou sortie)

```
void setup()
{
    pinMode(led_rouge, OUTPUT); //initialisation de la patte 2 comme étant une sortie
}
```

Ce code va donc définir la *led_rouge* (qui est la broche numéro 2 du microcontrôleur) en sortie, car *OUTPUT* signifie en français : sortie.



Résumons!

Concrètement, qu'est-ce que cela signifie?

Et bien, dans notre code 1, nous avions défini la patte du microcontrôleur de cette façon:

```
void setup() {
    // Initialise la patte 13 comme sortie
    pinMode(13, OUTPUT);

    // Ouvre le port série à 9600 bauds
    Serial.begin(9600);

}

void loop() {
    digitalWrite(13, HIGH);      // allume la LED
    delay(500);                // attend 500ms
    digitalWrite(13, LOW);       // éteint la LED
    delay(500);                // attend 500ms
}
```

Ainsi, nous pouvons aussi définir la patte de cette manière:

```
const int led_rouge = 13;      //La patte 13 devient led_rouge

void setup() {
    // Initialise led_rouge comme patte de sortie
    pinMode(led_rouge, OUTPUT);

    // Ouvre le port série à 9600 bauds
    Serial.begin(9600);

}

void loop() {
    digitalWrite(led_rouge, HIGH);    // allume la LED
    delay(500);                    // attend 500ms
    digitalWrite(led_rouge, LOW);     // éteint la LED
    delay(500);                    // attend 500ms
}
```

À quoi est-ce que cela sert? Quand nous aurons de nombreuses pattes en fonction, cela nous permettra de les identifier plus facilement. Ainsi, si nous avons plusieurs LED, nous pouvons les appeler *L1*, *L2*, *L3*... et si nous utilisons des LED de plusieurs couleurs, nous pourrons les appeler *rouge*, *vert*, *bleu*...

Enfin, si on veut changer la patte utilisée, il suffit de corriger la variable, sans devoir corriger tout le code.



Évaluation: K2000

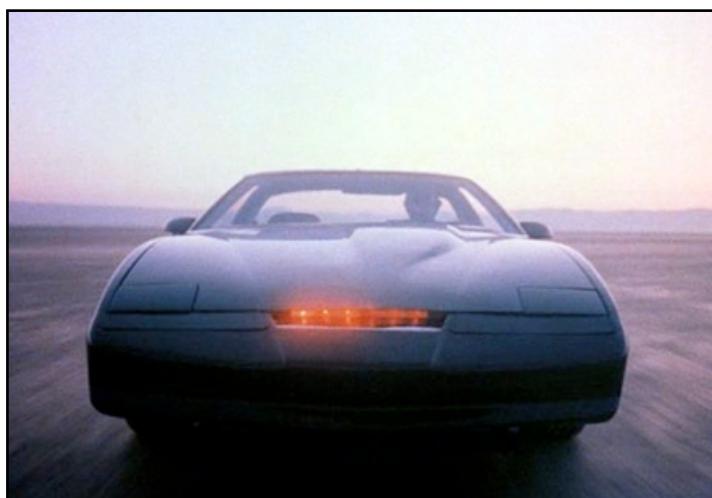
Introduction

«Les exploits d'un chevalier solitaire, dans un monde dangereux; le chevalier et sa monture...»

Il y longtemps de cela, du temps de ma jeunesse (soit peu de temps après la disparition du dernier mammouth), un héros hantait la petite lucarne: Michael Knight et sa voiture parlante, Kitt.

Je t'invite à revoir le générique de cette série d'anthologies (en terme de mauvais goût capillaire) ici: http://www.youtube.com/watch?v=PI_1sNt1cl8¹⁸

Comme tu peux le voir, la voiture possède sur son capot des LED qui simulent un mouvement de droite à gauche et de gauche à droite (avec un magnifique son du style *fschouiiii-fschouiiii*).



Ta mission, si tu l'acceptes (mais tu as intérêt... c'est quand même une évaluation!), est de reproduire cet accessoire indispensable à toute automobile qui se respecte: le bandeau de LED... mais sans le son (le mauvais goût à des limites).

Pour cela, tu auras besoin de 6 LED rouges (à priori... mais tout est possible) et de 6 résistances (des $220\ \Omega$ à $470\ \Omega$ feront parfaitement l'affaire).

Pour t'aider, n'hésite pas à regarder cette petite vidéo d'exemple: <http://00.lc/k2000>

Corrigé¹⁹: <http://www.edurobot.ch/code/k2000.pdf>

¹⁸ Lien court: <http://00.lc/cp>

¹⁹ Fichier protégé par un mot de passe. Pour l'obtenir: info [at] edurobot.ch



Les objectifs

Objectifs de la discipline MSN

MSN 35 — Modéliser des phénomènes naturels, techniques, sociaux ou des situations mathématiques...

- en mobilisant des représentations graphiques (codes, schémas, tableaux, graphiques,...)
- en associant aux grandeurs observables des paramètres
- en triant, organisant et interprétant des données
- en communiquant ses résultats et en présentant des modélisations

MSN 36 — Analyser des phénomènes naturels et des technologies à l'aide de démarches caractéristiques des sciences expérimentales...

- en formulant des hypothèses
- en confrontant les hypothèses émises à des résultats expérimentaux
- en définissant des stratégies d'exploration et d'expérimentation en lien avec les hypothèses émises
- en proposant des explications et en les confrontant à celles de ses pairs et aux informations de médias variés

Si applicable:

Objectifs de la discipline MEP

MEP 34 – Développer des sujets, des projets favorisant une ouverture sur les sciences du passé, actuelles et du futur...

Familiarisation avec des façons de faire et de raisonner propres à la logique robotique, notamment en:

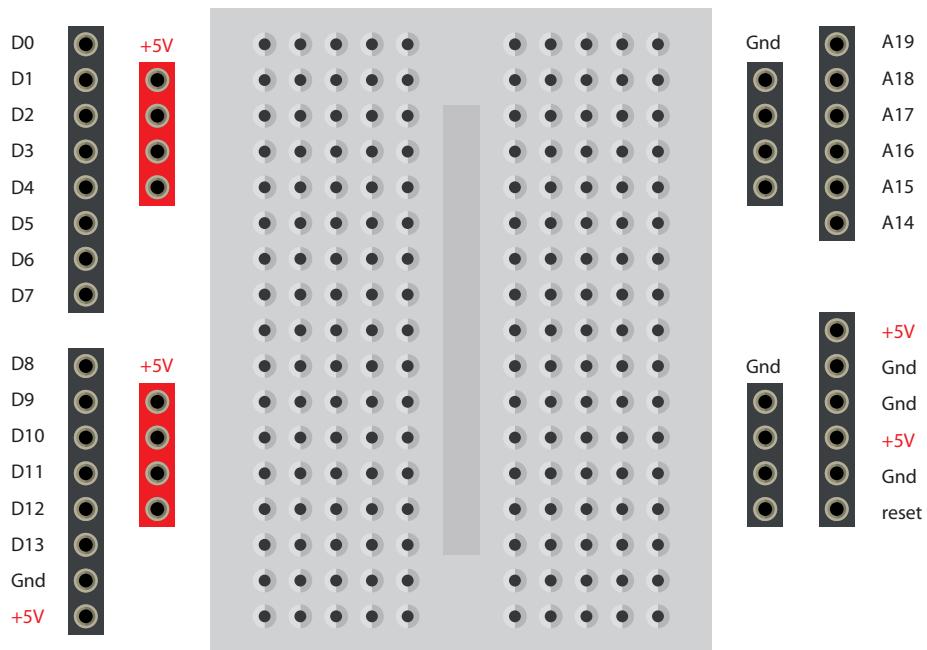
- identifiant les objectifs d'une activité ainsi que les notions théoriques et pratiques qui en résultent
 - planifiant, construisant et programmant le robot en fonction des observations et des mesures effectuées
 - testant le robot afin d'identifier d'éventuelles erreurs de conception et en mettant en place des stratégies de remédiation
- Initiation à l'utilisation d'outils et de procédés simples afin de faire fonctionner un robot de manière autonome

Capacités transversales exercées: *stratégies d'apprentissage, démarche réflexive*

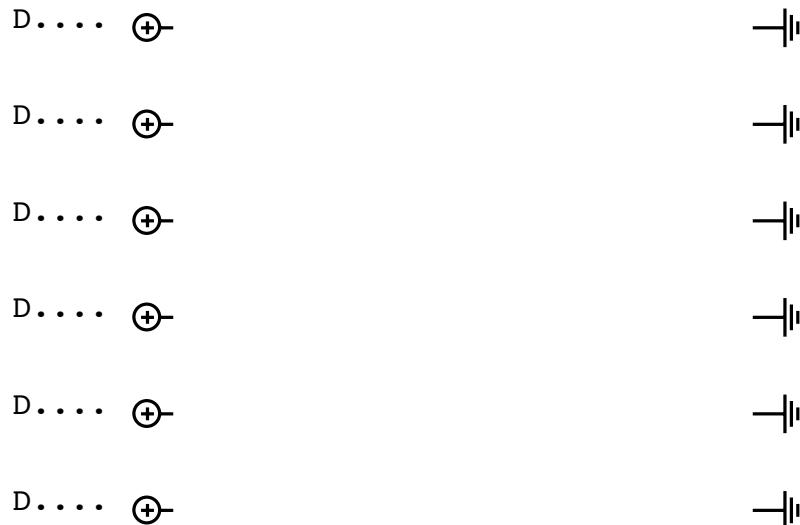


Étape 1: Schémas

Réalise le plan du montage que tu vas utiliser:



Réalise maintenant le schéma électrique du montage:





Étape 2: Programme

Rédige ton programme ici:

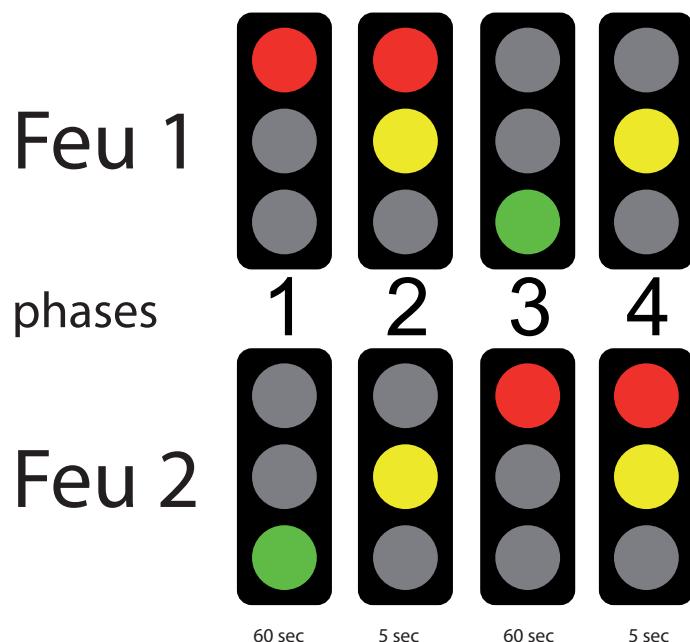
A large rectangular grid area designed for handwriting, intended for users to write their program code.



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 tu dois recréer:



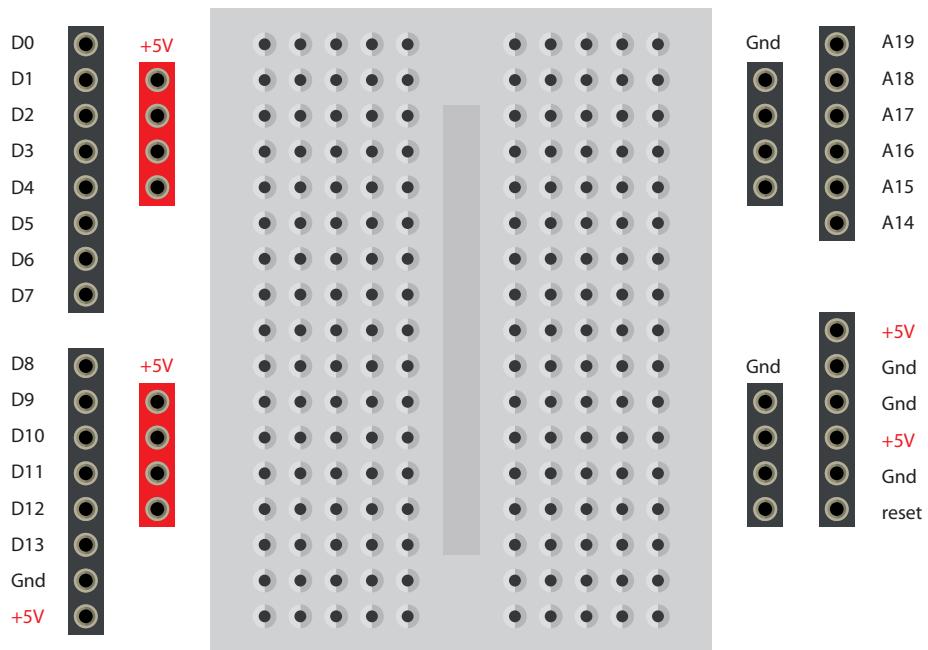
Tu peux visionner une vidéo de présentation ici: <http://00.lc/ct>

Établis la liste des composants que tu auras besoin pour réaliser le projet:

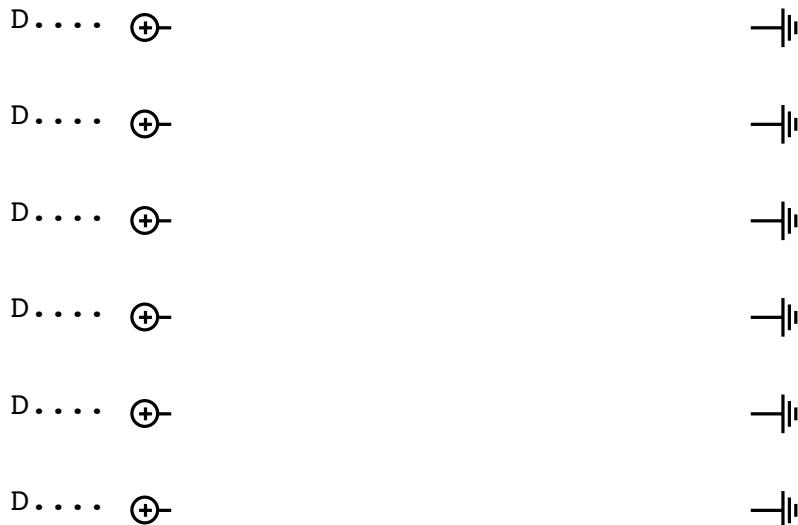


Étape 1: Schémas

Réalise le plan du montage que tu vas utiliser:



Réalise maintenant le schéma électrique du montage:





Étape 2: Programme

Rédige ton programme ici:

A large rectangular grid area designed for handwriting, intended for users to write their program code.



Corrigé

Tu trouveras le corrigé à l'adresse suivante: <http://edurobot.ch/code/code5.txt>

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.

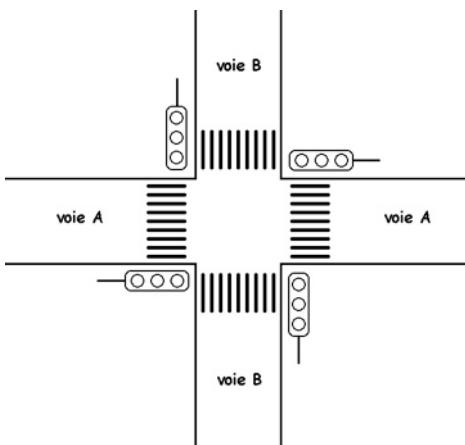
Variante 2

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

Réaliser le schéma électrique et programmer les feux.

Variante 3

Réaliser les feux pour un carrefour:



Source: http://minne.romain.free.fr/MPl/tp10_carrefour/Gestion_d_un_carrefour.html



Leçon 6: Input

Le bouton poussoir

Le bouton poussoir normalement ouvert (NO)

C'est le bouton le plus fréquemment rencontré:

- Relâché : le courant ne passe pas, le circuit est déconnecté ; on dit que le circuit est *ouvert*.
- Appuyé : le courant passe, on dit que le circuit est *fermé*.

Habituellement le bouton poussoir a deux broches, mais en général ils en ont 4 reliées deux à deux.



Le bouton poussoir normalement fermé (NF)

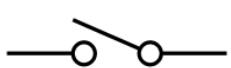
Ce type de bouton est l'opposé du type précédent, c'est-à-dire que lorsque le bouton est relâché, il laisse passer le courant. Et inversement :

- Relâché : le courant passe, le circuit est connecté ; on dit que le circuit est *fermé*.
- Appuyé : le courant ne passe pas, on dit que le circuit est *ouvert*.

Les interrupteurs

À la différence d'un bouton poussoir, l'interrupteur agit comme une bascule. Un appui ferme le circuit et il faut un second appui pour l'ouvrir de nouveau. Il possède donc des états stables (ouvert ou fermé). Nous en rencontrons tous les jours lorsque nous allumons la lumière.

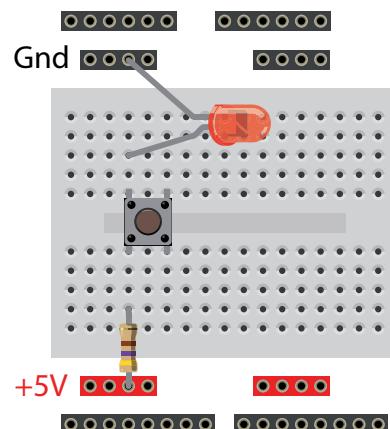
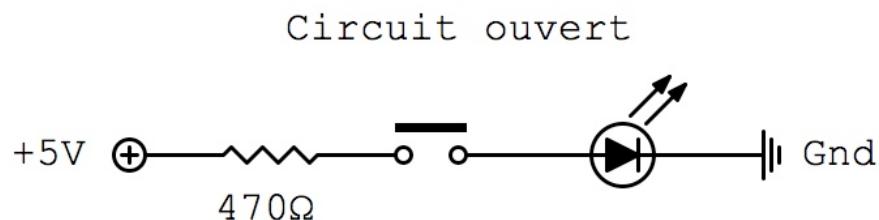


 Bouton poussoir NO	 Bouton poussoir NF	 Interrupteur
---	---	---

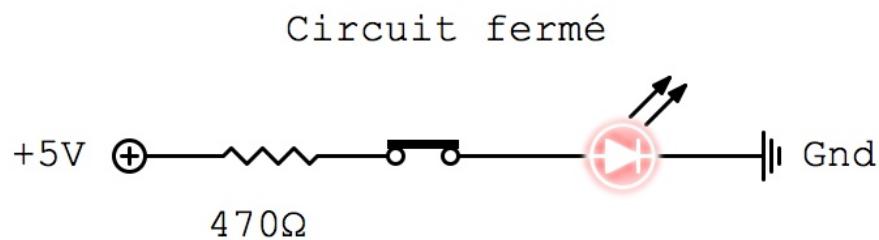


Exercice 1: allumons cette LED

Réalise le circuit suivant:



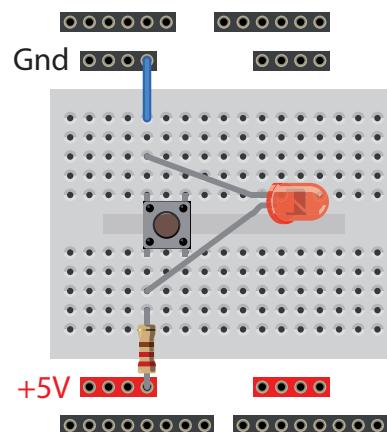
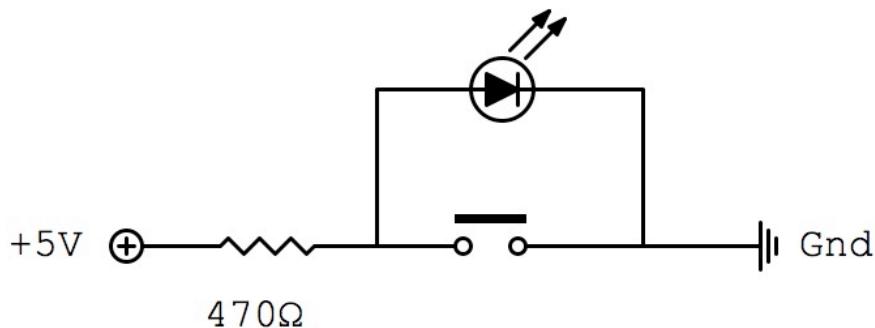
Lorsqu'on appuie sur le bouton poussoir, on ferme le circuit. Le courant passe alors et la LED s'allume.





Exercice 2: mais par où passe le courant?

Réalise le circuit suivant:



Il y a plus de résistance au passage du courant lorsqu'il traverse une LED qu'un bouton poussoir. Or, le courant passera là où il rencontrera le moins de résistance.

C'est pourquoi, quand le bouton poussoir est relâché, le courant passe par la LED. Mais lorsque tu appuis sur le bouton, la LED s'éteint. En effet, le courant préfère passer au travers du bouton poussoir.

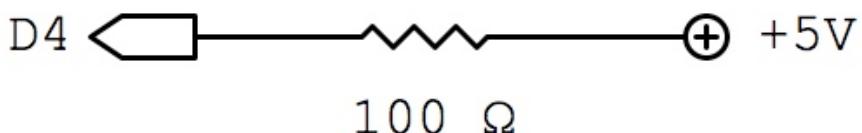


Protéger l'Arduino

Jusqu'à maintenant, nous nous sommes toujours contentés de faire circuler du courant du microcontrôleur à la masse.

Au lieu d'utiliser les pattes du microcontrôleur seulement sous forme de **sortie** (*OUTPUT*), nous allons aussi les utiliser sous forme d'**entrée** (*INPUT*); c'est-à-dire qu'au lieu d'être raccordée à la masse, la patte sera raccordée au +5V. Tout se passera bien, si au niveau du programme, la patte est configurée comme *input*. Mais si elle devait être configurée en *output* par erreur, il est presque certain que le microcontrôleur finira immédiatement au paradis des puces électroniques grillées.

Ainsi, pour éviter de condamner notre microcontrôleur à la chaise électrique, nous allons devoir le protéger. Cela peut se faire en connectant sur la patte du microcontrôleur utilisé comme *input* une résistance d'une centaine d'Ohms. Comme le Diduino est fourni avec des résistances de 220 Ω , c'est celle que nous utiliserons.



ATTENTION!

Lorsque vous branchez le Diduino à l'ordinateur, le dernier programme reçu est exécuté. Avant de commencer un nouveau montage, il est plus que conseillé d'envoyer un programme d'initialisation, du modèle de celui-ci:

```

//Programme vide d'initialisation
void setup() {
}

void loop() {
}
    
```

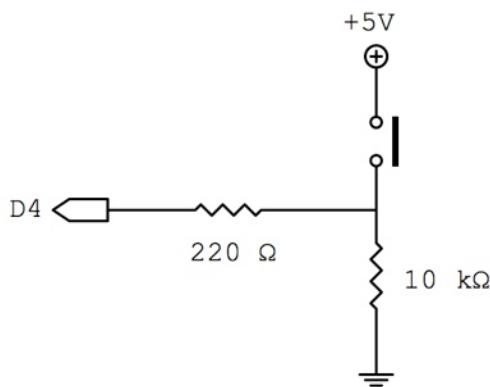
Maintenant que ce point est réglé, voyons comment utiliser le bouton poussoir avec le microcontrôleur.



Résistance Pull-Down / Pull-Up

Circuit avec une résistance pull-down

Observons ce schéma:



Lorsque l'on presse le bouton poussoir, on envoie +5 Volts sur D4, qui sera interprété comme un **1**.

Lorsqu'on relâche le bouton, il n'y a plus de courant qui arrive sur D4, ce qui sera interprété comme un **0**. On a donc un signal binaire: allumé/éteint (on/off).

Ce montage contient une résistance de $10\text{ k}\Omega$ (soit $10'000\ \Omega$), qu'on appelle pull-down (littéralement *tirer-en-bas*). Cette résistance permet de *tirer* le potentiel vers le *bas* (*pull-down*). En français, on appelle aussi ceci un *rappel au moins*.

En effet, contrairement au cas théorique, fermer ou ouvrir un circuit (via un interrupteur ou un bouton poussoir) ne génère pas forcément un signal clair:

- Le circuit non connecté à la source peut agir comme une antenne dans un environnement pollué d'ondes électromagnétiques (proximes des moteurs par exemple). Cela va générer dans le circuit un courant qui peut être interprété comme un signal.
- Il peut rester dans un circuit récemment ouvert un niveau d'énergie. Cela provoque des cas d'indétermination. Le signal qui a été coupé peut être considéré comme encore actif par un microcontrôleur.

Le but d'une résistance de pull-down est donc d'évacuer les courants vagabonds et de donner un signal clair.

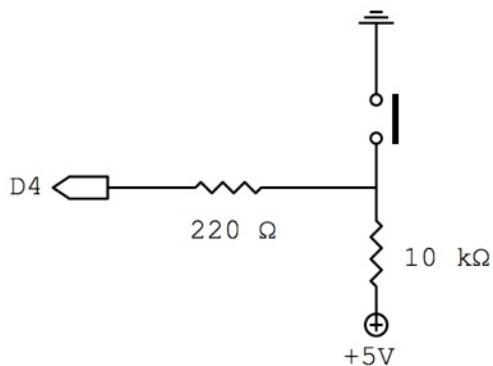
Si on presse le bouton, 5 Volts sont alors appliqués sur l'entrée. Le courant va prendre le chemin le plus simple, soit par la résistance de $220\ \Omega$ et finit par arriver sur D4. Si on relâche le bouton, la résistance pull-down ramène l'entrée à la masse (puisque elle est connectée à la masse).





Circuit avec une résistance pull-up

Observons maintenant ce schéma:



Si on le compare au schéma d'un circuit avec une résistance pull-down, on constate une inversion entre la terre et le +5V.

En effet, lorsque le circuit est ouvert, une tension de +5V est appliquée à l'entrée D4. Lorsqu'on appuie sur le bouton poussoir, le courant électrique va passer par le chemin offrant le moins de résistance, soit directement par la masse (Gnd), sans passer par l'entrée D4.

Le fonctionnement est donc inversé par rapport à la résistance pull-down.

Résistance pull-down ou pull-up?

A notre niveau, la seule chose qui va changer entre une résistance pull-down et une résistance pull-up est la lecture de l'information:

Avec une résistance pull-down, par défaut, l'entrée sur la patte est égale à 0.

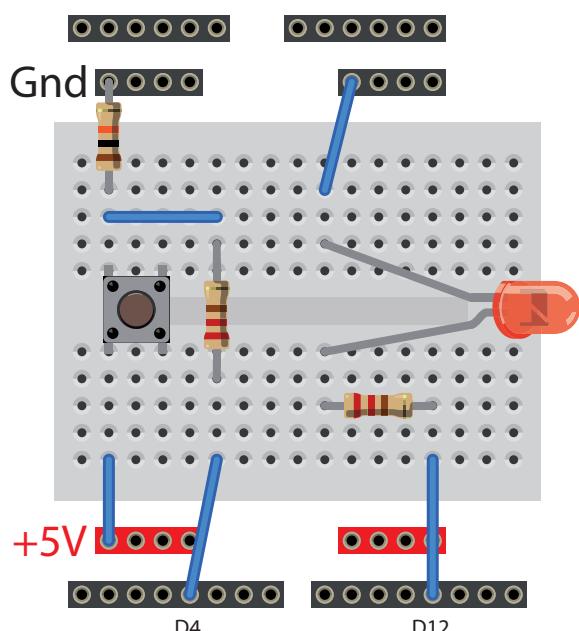
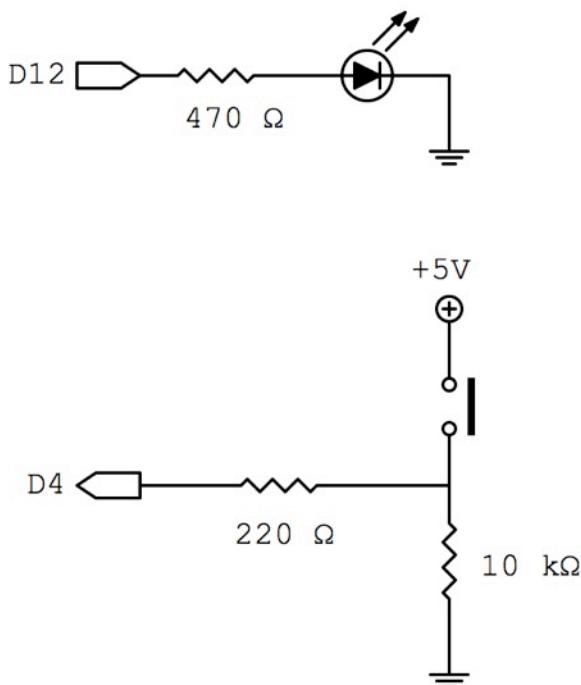
Avec une résistance pull-up, par défaut, l'entrée sur la patte est égale à 1.

Dans le code `if (digitalRead(bouton) == 1)`, sera respectivement la valeur d'entrée lorsque le circuit est fermé (pull-down) ou ouvert (pull-up).



Exercice 3: montage avec résistance pull-down (rappel au moins)²⁰

Réalisons maintenant le circuit suivant:



Indique sur le schéma ci-dessus où se trouve la résistance pull down et la résistance de protection du microcontrôleur

Lorsqu'on appuie sur le bouton poussoir, La LED doit s'allumer. Et naturellement, lorsqu'on relâche le bouton poussoir, la LED s'éteint.

Nous allons utiliser une nouvelle instruction: *if ... else (si ... sinon)*. C'est donc une *condition*.

Voici ce qui va se passer:

Si le bouton poussoir est pressé (*digitalRead(4) == 1*), allumer la LED (*digitalWrite(12, HIGH)*), **sinon** éteindre la LED (*digitalWrite(12, LOW)*).

L'instruction **==** vérifie si deux expressions sont égales. Si elles le sont, alors le résultat sera vrai (*true*) sinon le résultat sera faux (*false*).

²⁰ Plus d'informations ici: <http://00.lc/dn>



Observons maintenant le code:

```
/*
Code 7 - Edurobot.ch, destiné au Diduino
Allume LED en fonction de l'état du bouton poussoir
*/
// On déclare les variables

const int bouton = 4;           // la patte 4 devient bouton
const int led = 12;             // la patte 12 devient led

void setup()
{
    pinMode(bouton, INPUT);     // Initialise la patte 4 comme entrée

    pinMode(led, OUTPUT);       // Initialise la patte 12 comme sortie

    Serial.begin(9600);         // Ouvre le port série à 9600 bauds
}

void loop()
{
// Si bouton poussoir appuyé
    if (digitalRead(bouton) == 1) //teste si le bouton a une valeur de 1
    {
        digitalWrite(led, HIGH); // allume la LED
    }

// Sinon
    else                      //teste si le bouton a une valeur de 0
    {
        digitalWrite(led, LOW); // éteint la LED
    }
}
```

Télécharger le code: <http://www.edurobot.ch/code/code7.txt>

**Analysons le code:**

```
const int bouton = 4;           // la patte 4 devient bouton
const int led = 12;             // la patte 12 devient led
```

On renomme les pattes 4 et 12 respectivement *bouton* et *led*. Cela facilitera la lecture du code.

```
void setup() {
    // Initialise la patte 4 comme entrée
    pinMode(bouton, INPUT);

    // Initialise la patte 12 comme sortie
    pinMode(led, OUTPUT);

    // Ouvre le port série à 9600 bauds
    Serial.begin(9600);
}
```

Voilà, maintenant notre microcontrôleur sait qu'il y a quelque chose de connecté sur sa patte 4 et qu'elle est configurée en entrée (*input*). De même, la patte 12 est configurée en sortie (*output*).

Maintenant que le bouton est paramétré, nous allons chercher à savoir quel est son état (appuyé ou relâché).

Si le microcontrôleur détecte une différence de potentiel de +5V à sa patte 4, alors il stockera une valeur de *1*.

Dans le cas contraire (différence de potentiel de 0V), il stockera une valeur de *0*.

Pour lire l'état de la patte 4, nous allons utiliser l'expression *digitalRead*. Ainsi:

```
if (digitalRead(bouton) == 1)
    digitalWrite(led, HIGH); // allume la LED
```

doit se comprendre comme:

*si la patte 4 a une valeur égale à 1
alors la patte 12 est en position haute = LED allumée*

et les lignes:

```
else {
    digitalWrite(led, LOW); // éteint la LED
```

doivent se comprendre comme:

*sinon (c'est à dire: la patte 4 a une valeur égale à 0)
la patte 12 est en position basse = LED éteinte*

Voici une petite vidéo qui présente l'activité: <http://00.lc/cx>



Une petite variation de code

Le code précédent est simple, mais manque un peu de finesse; un peu comme un barbare avec une hache à deux mains dans une réception de Monsieur l'Ambassadeur. Voici donc une autre solution:

```
/*
Code 8 - Edurobot.ch, destiné au Diduino
Allume LED en fonction de l'état du bouton poussoir
*/
const int bouton = 4;           // la patte 4 devient bouton
const int led = 12;             // la patte 12 devient led
int etatbouton;                // variable qui enregistre l'état du bouton

void setup()
{
    pinMode(bouton, INPUT);      // Initialise le bouton comme entrée
    pinMode(led, OUTPUT);        // Initialise la led comme sortie
    etatbouton = LOW;            // Initialise l'état du bouton comme relâché

    Serial.begin(9600);          // Ouvre le port série à 9600 bauds
}

void loop()
{
    etatbouton = digitalRead(bouton); //On mémorise l'état du bouton

    if(etatbouton == LOW) //teste si le bouton a un niveau logique BAS
    {
        digitalWrite(led,LOW); //la LED reste éteinte
    }
    else //teste si le bouton a un niveau logique différent de BAS (donc HAUT)
    {
        digitalWrite(led,HIGH); //le bouton est appuyé, la LED est allumée
    }
}
```

Pour commencer, nous allons créer une variable que nous choisirons d'appeler *etatbouton*. Elle servira à stocker l'état du bouton (logique, non?):

```
int etatbouton;
```

On inscrit l'état du bouton dans la variable de cette manière, avec la fonction *digitalRead*:

```
etatbouton = digitalRead(bouton);
```



Il ne nous reste plus qu'à appeler l'état du bouton, stocké dans la variable *etatbouton*, et à lui faire passer un petit test avec *if... else* (si... sinon):

Si (if) l'état du bouton est *LOW* (c'est-à-dire = 0), **alors la LED est *LOW*** (éteinte). **Sinon (else)**, **la LED est *HIGH*** (en effet, si l'état du bouton n'est pas *LOW*, il ne peut être que *HIGH*, soit allumée...)

Et cela donne donc ceci:

```
if(etatbouton == LOW) //teste si le bouton a un niveau logique BAS
{
    digitalWrite(led,LOW); //la LED reste éteinte
}
else //teste si le bouton a un niveau logique différent de BAS (donc HAUT)
{
    digitalWrite(led,HIGH); //le bouton est appuyé, la LED est allumée
}
```

Rappelons que l'instruction **`==`** vérifie si deux expressions sont égales.

Petits exercices: bouton poussoir et LED qui clignote

Imaginons maintenant le cas de figure suivant: avec le même circuit: lorsque nous appuyons sur le bouton, la LED commence à clignoter.

Modifie le programme pour arriver à ce résultat.

Télécharger la réponse: <http://www.edurobot.ch/code/code10.txt>

Et maintenant, modifie le programme pour que lorsque nous branchons le Diduino à l'ordinateur la LED s'allume, et reste allumée. Par contre, quand nous appuyons sur le bouton, la LED clignote.

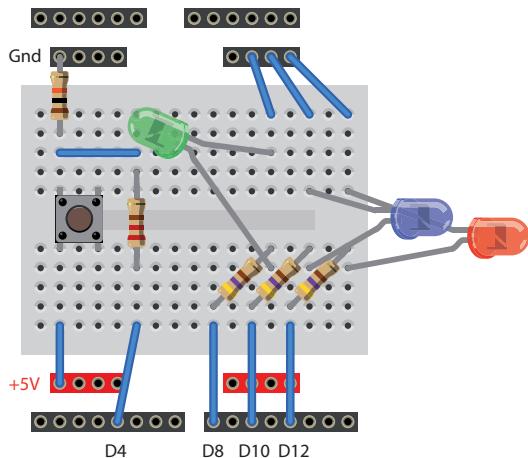
Télécharger la réponse: <http://www.edurobot.ch/code/code11.txt>



Evaluation: vers l'infini et au-delà!

Bienvenue à bord, Astronaute! L'heure est grave! La galaxie est menacée par les infâmes VoRteX. Pour la sauver, tu dois concevoir un pistolet protoplasmique à dodécaneutrino.

Voici le montage que tu devras réaliser (tu peux écarter LEDs pour avoir plus de place):



Son fonctionnement est on ne peut plus simple: quand le pistolet protoplasmique à dodécaneutrino est sous tension, une LED verte est allumée, pour montrer que tout fonctionne parfaitement. Quand on appuie sur le bouton, La LED verte s'éteint et les LEDs rouges et bleues se mettent à clignoter alternativement, générant un plasma positronique.

Les objectifs

Objectifs de la discipline AC&M

A 33 AC&M — Exercer diverses techniques plastiques et artisanales...

- en maîtrisant des habiletés de motricité globale et fine (souplesse, précision, coordination, rapidité du geste,...)
- en choisissant et utilisant divers outils et matériaux en fonction d'un projet
- en maîtrisant des gestes artisiaux spécifiques, en exerçant la précision
- en utilisant des techniques numériques

**Objectifs de la discipline MSN**

MSN 35 — Modéliser des phénomènes naturels, techniques, sociaux ou des situations mathématiques...

- en mobilisant des représentations graphiques (codes, schémas, tableaux, graphiques,...)
- en associant aux grandeurs observables des paramètres
- en triant, organisant et interpréter des données
- en communiquant ses résultats et en présentant des modélisations

MSN 36 — Analyser des phénomènes naturels et des technologies à l'aide de démarches caractéristiques des sciences expérimentales...

- en formulant des hypothèses
- en confrontant les hypothèses émises à des résultats expérimentaux
- en définissant des stratégies d'exploration et d'expérimentation en lien avec les hypothèses émises
- en proposant des explications et en les confrontant à celles de ses pairs et aux informations de médias variés

Si applicable:

Objectifs de la discipline MEP

MEP 34 – Développer des sujets, des projets favorisant une ouverture sur les sciences du passé, actuelles et du futur...

Familiarisation avec des façons de faire et de raisonner propres à la logique robotique, notamment en:

- identifiant les objectifs d'une activité ainsi que les notions théoriques et pratiques qui en résultent
 - planifiant, construisant et programmant le robot en fonction des observations et des mesures effectuées
 - testant le robot afin d'identifier d'éventuelles erreurs de conception et en mettant en place des stratégies de remédiation
- Initiation à l'utilisation d'outils et de procédés simples afin de faire fonctionner un robot de manière autonome

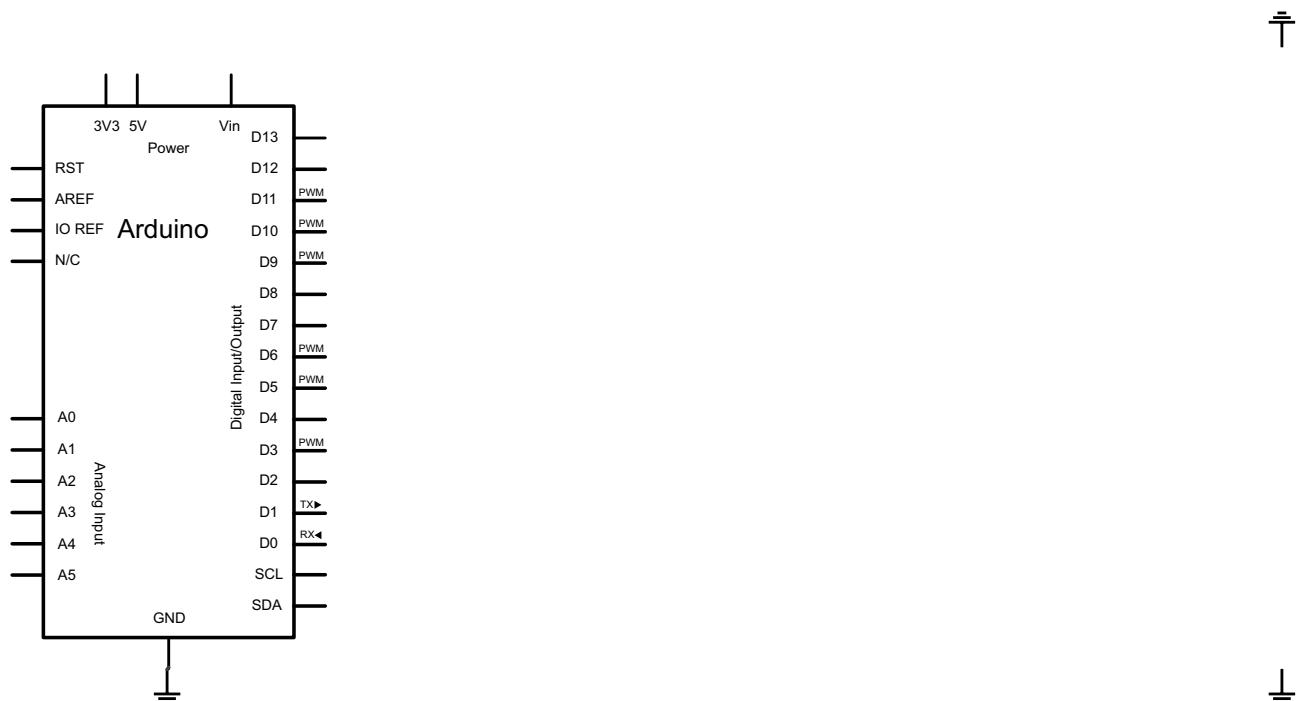
L'élève est capable de:

- Réaliser un montage à l'aide d'un bouton poussoir, de LEDs et d'une résistance pull down.
- D'utiliser l'Arduino de manière sûre
- De programmer correctement l'Arduino, pour lui faire exécuter la tâche demandée



Etape 1: les schémas

Commence par réaliser le schéma électronique:



Sur ton schéma, entoure en rouge la résistance pull down.

A quoi sert-elle?

Observe la vidéo suivante: <http://00.lc/d2>

Liste la ou les pattes qui sont programmées en INPUT et celles qui le sont en OUTPUT:

INPUT	OUTPUT



Étape 2: Programme

Rédige ton programme ici:

A large rectangular grid area with a light blue background and a grid pattern, designed for writing code or drawing programs.

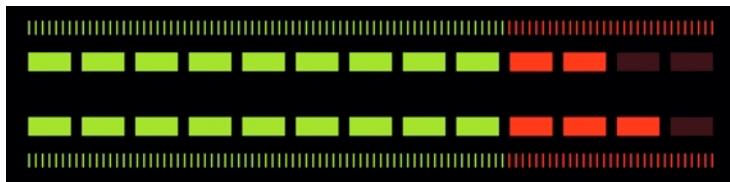
Tu peux naturellement aussi l'imprimer et le joindre à ton évaluation.



Leçon 7: le barregraphe

Introduction

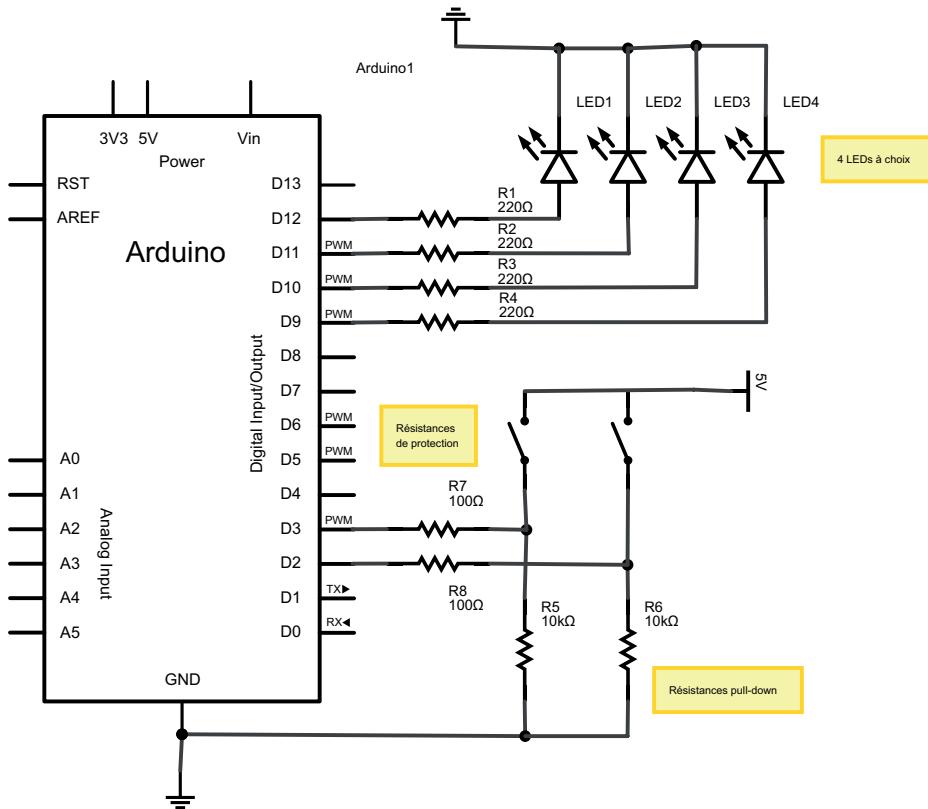
Un barregraphe est un afficheur qui indique une quantité, provenant d'une information quelconque (niveau d'eau, puissance sonore, etc.), sous une forme lumineuse. Le plus souvent, on utilise des LEDs alignées en guise d'affichage.



L'objectif de cet exercice est de réaliser un barregraphe de 4 LEDs, avec deux boutons poussoirs. L'un d'eux servira à incrémenter la valeur sur le barregraphe (à savoir augmenter le nombre de LEDs allumées), alors que l'autre servira à le décrémenter.

Schémas

Voici le schéma du circuit à construire:

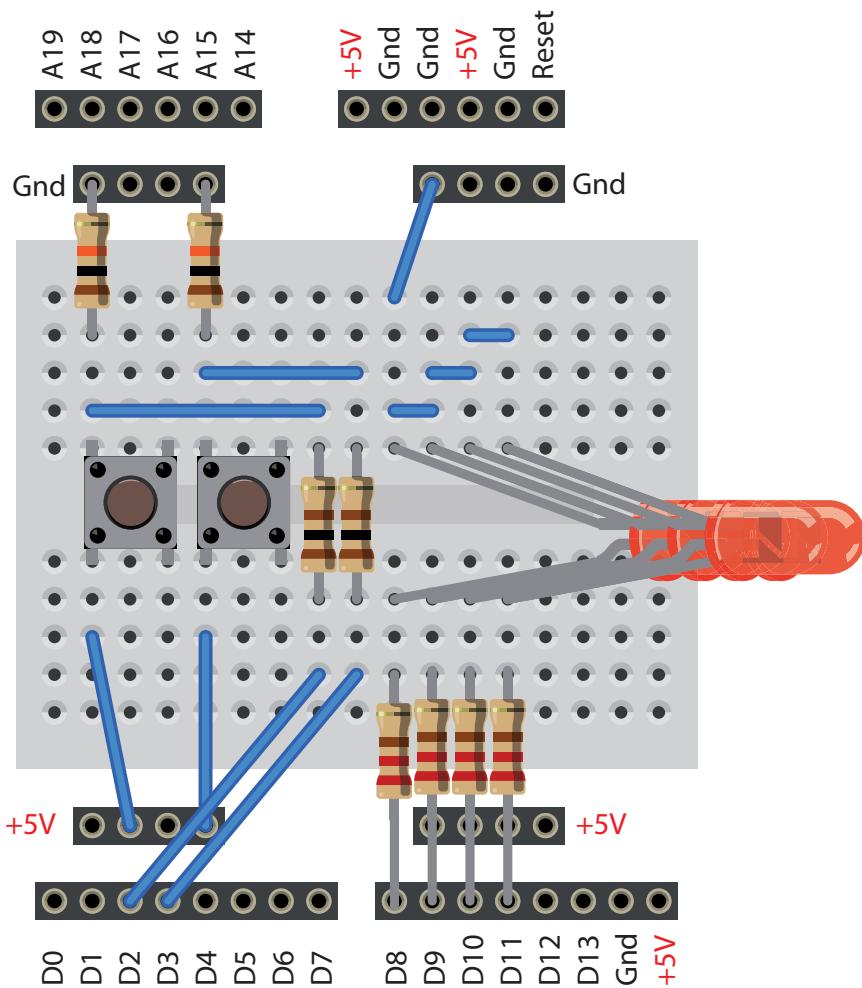




Comme on peut l'observer, les résistances R5 et R6 sont montées en pull-down (rappel au moins). Ce circuit pourrait tout à fait être monté avec des résistances pull-up.

Normalement, à ce stade, nous devrions être capable d'exécuter le montage à l'aide du schéma électrique!

Dans le cas contraire, voici le montage à réaliser:



Il ne faut pas hésiter à écarter un peu plus les LEDs si nécessaire!

Une vidéo présentant le barregraphe est visible ici: <http://www.edurobot.ch/?p=1316>

Le code complet est ici: <http://www.edurobot.ch/code/code13.txt>



Le code

Analysons le code, maintenant:

En se référant au schéma de montage, on déclare les constantes pour chaque patte: les boutons poussoirs sont connectés sur les pattes 2 et 3, alors que les LEDs sont connectées aux pattes 8 à 11.

```
/*
Déclaration des constantes pour les noms des pattes
*/
const int btn_minus = 2;      //Bouton 1 pour décrémenter le nombre de LEDs allumés
const int btn_plus = 3;       //Bouton 2 pour incrémenter le nombre de LEDs allumés
const int led_0 = 8;          //Led 0
const int led_1 = 9;          //Led 1
const int led_2 = 10;         //Led 2
const int led_3 = 11;         //Led 3
```

On crée ensuite les variables nécessaires et les initialise leur état.

```
/*
Déclaration des variables utilisées pour le comptage et le décomptage
*/
int nombre_led = 0;           //le nombre qui sera incrémenté et décrémenté
int etat_bouton;              //lecture de l'état des boutons (un seul à la fois mais une variable suffit; en effet, il n'est pas prévu d'appuyer sur les deux boutons simultanément)
int memoire_plus = LOW;        //état relâché par défaut pour le bouton 2
int memoire_minus = LOW;       //état relâché par défaut pour le bouton 1
```

On initilaise ensuite les pattes, selon qu'il s'agit des entrées (les boutons) ou des sorties (les LEDs).

```
/*
Initialisation des pattes en entrée/sortie: entrées pour les boutons, sorties pour les LEDs
*/
void setup()
{
    pinMode(btn_plus, INPUT);
    pinMode(btn_minus, INPUT);
    pinMode(led_0, OUTPUT);
```



```

pinMode(led_1, OUTPUT);
pinMode(led_2, OUTPUT);
pinMode(led_3, OUTPUT);
}

```

Allons-y pour la partie du programme dans la boucle sans fin (loop)

On utilise **digitalRead** pour lire l'état du bouton (**btn_plus**) et inscrire cet état dans la variable **etat_bouton**.

```

void loop()
{
    //lecture de l'état du bouton d'incrémentation (on lit l'état du btn_plus et
    on l'inscrit dans la variable etat_bouton)

    etat_bouton = digitalRead(btn_plus);
}

```

Maintenant, ça devient sérieux. Regardons ce qui suit:

```
//Si le bouton a un état différent que celui enregistré ET que cet état est
"appuyé"
```

```
if((etat_bouton != memoire_plus) && (etat_bouton == HIGH))
```

Traduisons cela en français: *si l'état du bouton 2 est différent de celui enregistré, et que quelqu'un appuye sur le bouton, alors...*

Le **!=** signifie **est différent de** (teste la différence entre deux variables) et l'opérateur logique **&&** signifie **ET** (Pour être précis, nous avons: *si ... et... avec le if ... &&...* Exemple: *si il fait beau et chaud, alors on va à la plage*).

Rappelons-nous maintenant de l'opération **++**, qui incrémente une variable (variable = variable + 1, c'est-à-dire; on ajoute à chaque fois 1 à la variable). Dans notre cas, il s'agit de la valeur de la variable **nombre_led** qu'on incrémentera.

```
{
    nombre_led++; //on incrémente la variable qui indique combien de LEDs
    devrons s'allumer
}
```

Et zou! On enregistre le nouvel état du bouton pour la suite!

```
memoire_plus = etat_bouton; //on enregistre l'état du bouton pour le tour
suivant
```

Maintenant, on recommence le tout, mais pour le bouton 1.

```
//et maintenant pareil pour le bouton qui décrémente
```



```

etat_bouton = digitalRead(btn_minus); //lecture de son état

//Si le bouton a un état différent que celui enregistré ET que cet état est
"appuyé"
if((etat_bouton != memoire_minus) && (etat_bouton == HIGH))
{
    nombre_led--; //on décrémente la valeur de nombre_led
}
memoire_minus = etat_bouton; //on enregistre l'état du bouton pour le tour
suivant

```

Nous allons maintenant limiter le nombre de LEDs à connecter. En effet, dans notre cas, nous avons 4 LEDs. Alors si on appuie 10 fois sur le bouton 2, cela n'allumera que les 4 LEDs.

```

//on applique des limites au nombre pour ne pas dépasser 4 ou 0 (puisque'on a
4 LEDs)
if(nombre_led > 4)
{
    nombre_led = 4;
}

```

Traduisons: *si on appuie plus de 4 fois sur le bouton, le résultat sera égal à 4*

Même chose maintenant pour le bouton 1.

```

if(nombre_led < 0)
{
    nombre_led = 0;
}

```

Maintenant, nous devons gérer l'allumage des LEDs. Pour simplifier le code, on va créer une fonction qui servira à gérer l'affichage. Nous allons appeler cette fonction **affiche**, avec un paramètre **int valeur_recue**. Ce paramètre représente le nombre à afficher.

```

//On crée une fonction affiche() pour l'affichage du résultat
//on lui envoie alors en paramètre la valeur du nombre de LED à éclairer

affiche(nombre_led);
}

void affiche(int valeur_recue)

```

On commence d'abord par éteindre toutes les LEDs.

```

{
    //on éteint toutes les leds
    digitalWrite(led_0, LOW);
    digitalWrite(led_1, LOW);
    digitalWrite(led_2, LOW);
    digitalWrite(led_3, LOW);
}

```



Si la fonction reçoit le nombre 1, on allume la LED 1. Si elle reçoit le nombre 2, elle allume la LED 1 et 2. Si elle reçoit 3, elle allume la LED 1, 2 et 3. Enfin, si elle reçoit 4, alors elle allume toutes les LEDs:

```
//Puis on les allume une à une

if(valeur_recue >= 1) // "si la valeur reçue est plus grande ou
également à 1..."
{
    digitalWrite(led_0, HIGH); // "on allume la LED 0
}
if(valeur_recue >= 2) // "si la valeur reçue est plus grande ou
également à 2..."
{
    digitalWrite(led_1, HIGH); // "on allume la LED 1 (sous-entendu que
la LED 0 est allumée, puisque la valeur est plus grande que 1)
}
if(valeur_recue >= 3) // "si la valeur reçue est plus grande ou
également à 3..."
{
    digitalWrite(led_2, HIGH); // "on allume la LED 2
}
if(valeur_recue >= 4) // "si la valeur reçue est plus grande ou
également à 4..."
{
    digitalWrite(led_3, HIGH); // "on allume la LED 3
}
```

Le symbole **>=** signifie: ...est supérieur ou égal à.... Il s'agit de tester la supériorité ou l'égalité d'une variable par rapport à une valeur. Ainsi, dans:

```
if(valeur_recue >= 4)
{
    digitalWrite(led_3, HIGH); // "on allume la LED 3
```

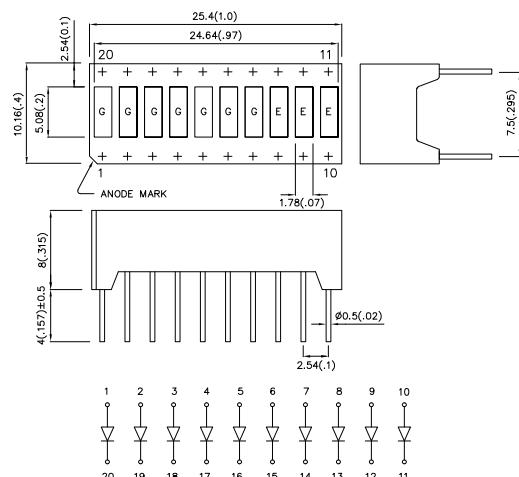
Il faut lire: *si la variable valeur_recue est supérieure ou égale à 4, alors on allume la LED 3.*



Variation: Le barregraphe à 10 LEDs

Il existe des barregraphes intégrants 10 LEDs²¹. On peut les utiliser en remplacement des 4 LEDs du précédent montage.

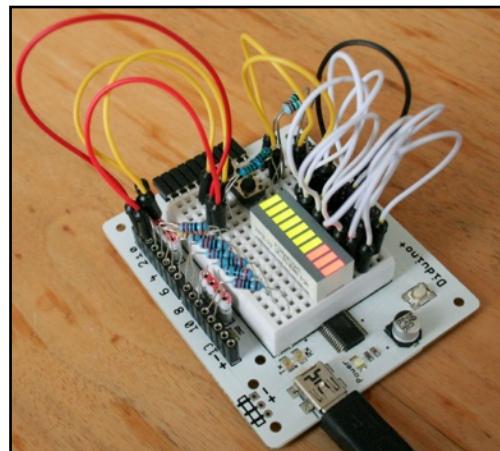
Références utilisées dans cet exercice: Kingbright DC-10EWA ou DC7G3EWA.



D'un côté se trouvent les cathodes, de l'autre les anodes. Dans notre cas, les anodes sont du côté où se trouvent les inscriptions.

Le montage d'un tel barregraphe sur la platine d'expérimentation est possible, mais cela reste délicat de câbler 10 LEDs. L'idéal est d'utiliser une platine d'expérimentation indépendante.

Le code, lui, doit être adapté en conséquence pour 10 LEDs. Un exemple se trouve ici:



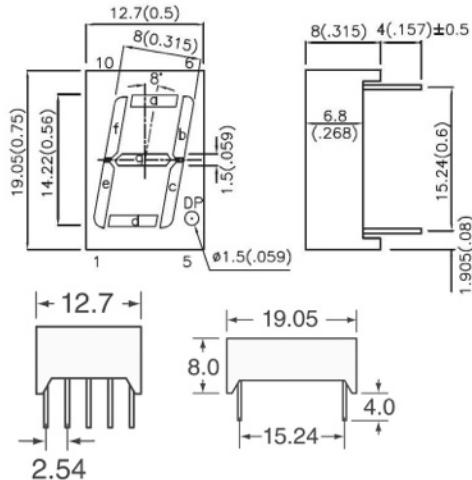
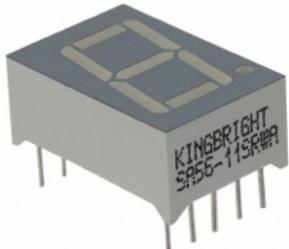
Code barregraphe 10 LEDs: <http://www.edurobot.ch/code/code14.txt>

²¹ On les trouve par exemple ici: <http://00.lc/gy> ou ici: <http://00.lc/gz>.

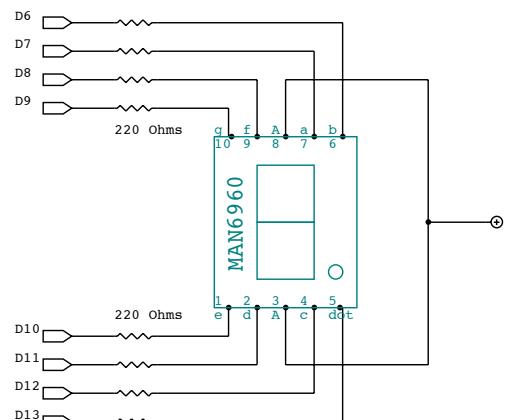
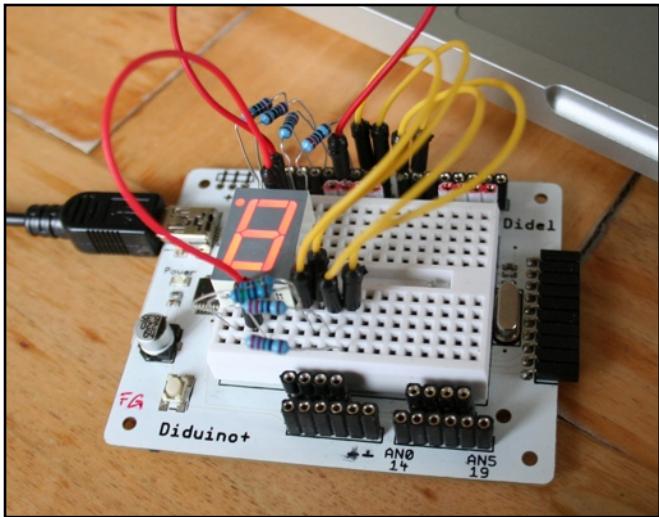


Variation: l'afficheur numérique

Pour ce montage, nous allons utiliser un afficheur numérique à LED.



La référence utilisée ici est un afficheur à anode commune Kingbright SA56-11 SRWA²².



Dans le cas d'une anode commune, cette dernière est branchée sur le +5V. Les cathodes sont branchées sur les pattes.

Au niveau du code, cela implique que les LEDs sont allumée en position LOW et éteintes en position HIGH.

Avec un composant à cathode commune, c'est le contraire.

Dans notre exemple, nous allons commencer par allumer toutes les diodes, puis l'une après l'autre, pour les identifier. On peut ensuite écrire des chiffres.

²² Disponible ici: <http://00.lc/hp>



Identification de la position des LEDs

Le code suivant permet d'identifier la position des LEDs en les allumant l'une après l'autre:

```
/*
Code 15 - Edurobot.ch, destiné au Diduino
L'afficheur 7 segments
L'objectif est d'afficher des chiffres sur un afficheur 7 segments (7 digits).
Ce code a pour objectif d'allumer toutes les LEDS puis l'une après l'autre
pour identifier leur position.
*/
/*
Déclaration des constantes pour les noms des pattes
*/
const int led_1 = 6;          //Led 1
const int led_2 = 7;          //Led 2
const int led_3 = 8;          //Led 3
const int led_4 = 9;
const int led_5 = 10;
const int led_6 = 11;
const int led_7 = 12;
const int led_8 = 13;

void setup()
{
    pinMode(led_1, OUTPUT);
    pinMode(led_2, OUTPUT);
    pinMode(led_3, OUTPUT);
    pinMode(led_4, OUTPUT);
    pinMode(led_5, OUTPUT);
    pinMode(led_6, OUTPUT);
    pinMode(led_7, OUTPUT);
    pinMode(led_8, OUTPUT);
}

/*
Et c'est parti pour le programme!
*/
void loop()
{
    //on allume toutes les leds

    digitalWrite(led_1, LOW);
    digitalWrite(led_2, LOW);
    digitalWrite(led_3, LOW);
    digitalWrite(led_4, LOW);
    digitalWrite(led_5, LOW);
```



```
digitalWrite(led_6, LOW);
digitalWrite(led_7, LOW);
digitalWrite(led_8, LOW);
delay(1500);

//on éteint toutes les leds

digitalWrite(led_1, HIGH);
digitalWrite(led_2, HIGH);
digitalWrite(led_3, HIGH);
digitalWrite(led_4, HIGH);
digitalWrite(led_5, HIGH);
digitalWrite(led_6, HIGH);
digitalWrite(led_7, HIGH);
digitalWrite(led_8, HIGH);
delay(500);

//on allume une diode après l'autre pour identifier sa position

digitalWrite(led_1, LOW);
delay(1500);

digitalWrite(led_1, HIGH);
digitalWrite(led_2, LOW);
delay(1500);

digitalWrite(led_2, HIGH);
digitalWrite(led_3, LOW);
delay(1500);

digitalWrite(led_3, HIGH);
digitalWrite(led_4, LOW);
delay(1500);

digitalWrite(led_4, HIGH);
digitalWrite(led_5, LOW);
delay(1500);

digitalWrite(led_5, HIGH);
digitalWrite(led_6, LOW);
delay(1500);

digitalWrite(led_6, HIGH);
digitalWrite(led_7, LOW);
delay(1500);

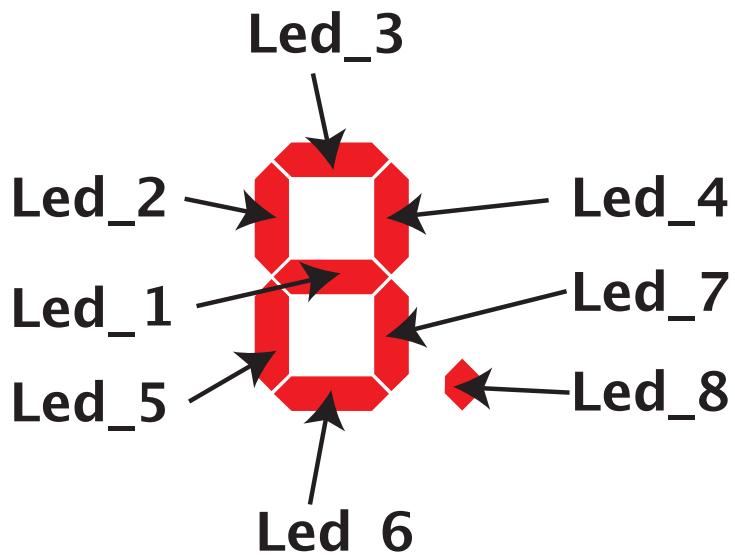
digitalWrite(led_7, HIGH);
digitalWrite(led_8, LOW);
delay(1500);

}

Le code est disponible ici: http://www.edurobot.ch/code/code15.txt
```



L'objectif est d'identifier chaque LED sur le schéma ci-dessous:



Voici donc, dans notre cas, les LEDs qui doivent être allumées pour écrire les chiffres suivants:

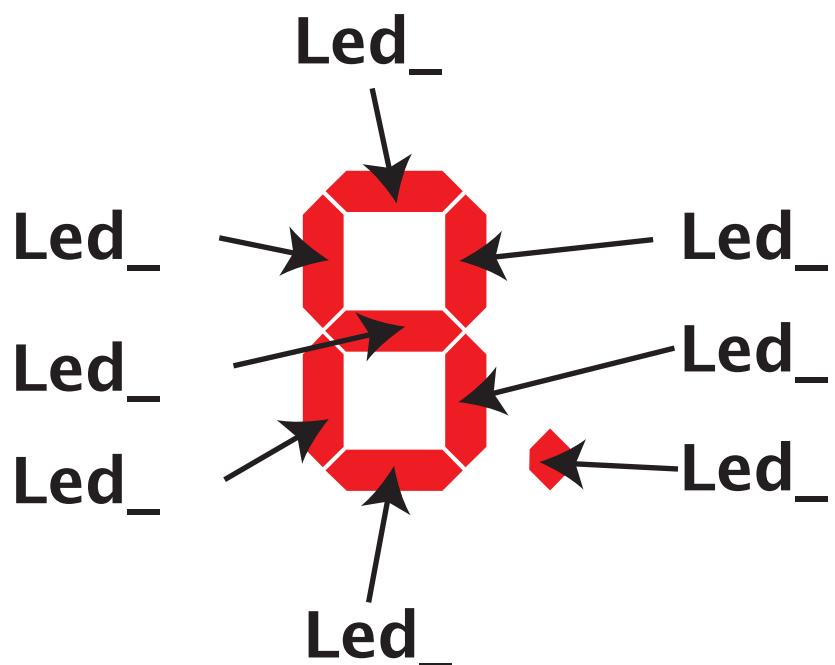
#	led_1	led_2	led_3	led_4	led_5	led_6	led_7
1				X			X
2	X		X	X	X	X	
3	X		X	X		X	X
4	X	X		X			X
5	X	X	X			X	X
6	X	X	X		X	X	X
7			X	X			X
8	X	X	X	X	X	X	X
9	X	X	X	X		X	
0		X	X	X	X	X	X

Nous n'utilisons pas la led_8, qui représente un point.

Le code pour compter de 0 à 9 est ici: <http://www.edurobot.ch/code/code16.txt>



A ton tour:



#	led_1	led_2	led_3	led_4	led_5	led_6	led_7
1							
2							
3							
4							
5							
6							
7							
8							
9							
0							

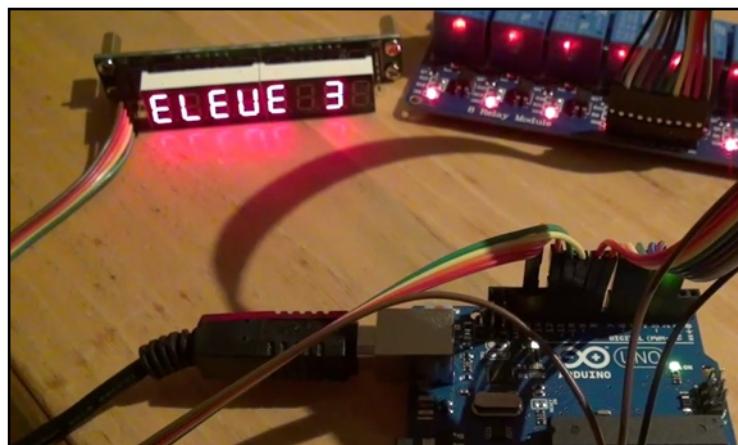


Il est aussi possible d'écrire les lettres de l'alphabet à l'aide d'un affichage 7 segments²³:

A 0065	B 0066	C 0067	D 0068	E 0069	F 0070	G 0071	H 0072	I 0073	J 0074	K 0075	L 0076	M 0077	N 0078
O 0079	P 0080	Q 0081	R 0082	S 0083	T 0084	U 0085	V 0086	W 0087	X 0088	Y 0089	Z 0090		
												4	2

Exemple:

Evidemment, cela reste artisanal; mais ça peut donner des résultats acceptables, comme on peut le voir ici: <http://00.lc/kj>.



²³ Source de la police: <http://www.dafont.com/7led.font>

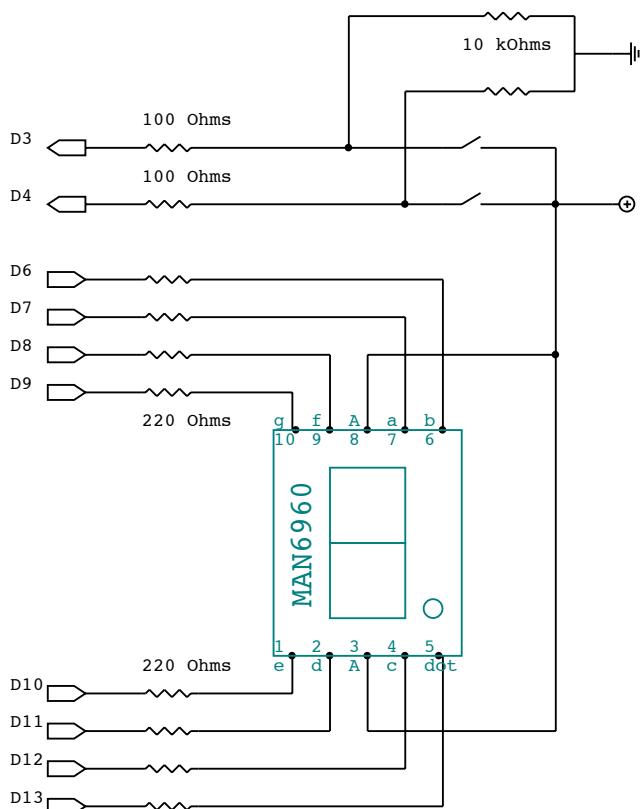


Leçon 8: apprendre à compter

Objectif

Cette dernière leçon va synthétiser tout ce que nous avons vu jusqu'à maintenant. L'objectif est de réaliser un montage, avec deux boutons-poussoirs et un affichage 7 segments. L'un des boutons va servir à incrémenter les chiffres sur l'affichage, et l'autre à les décrémenter. Ainsi, en appuyant 6 fois sur le bouton-poussoir, les chiffres de 1 à 6 vont successivement s'afficher.

Schéma électronique du montage

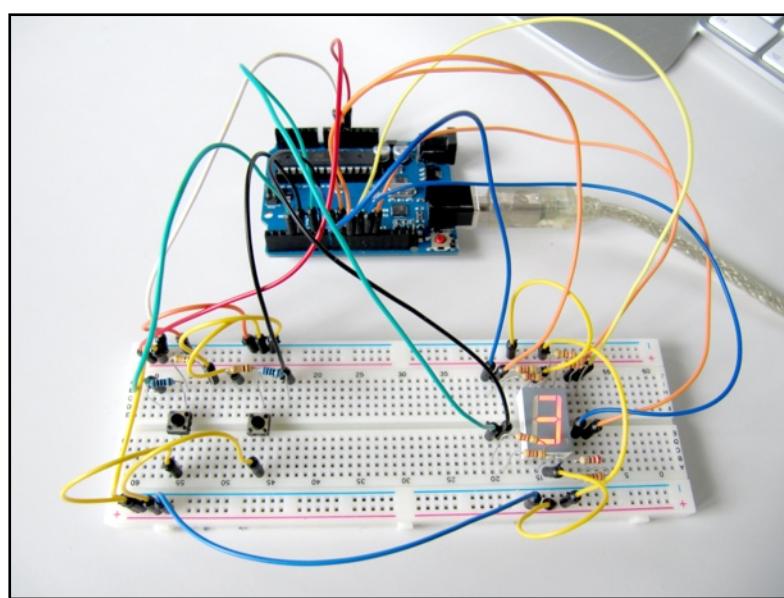
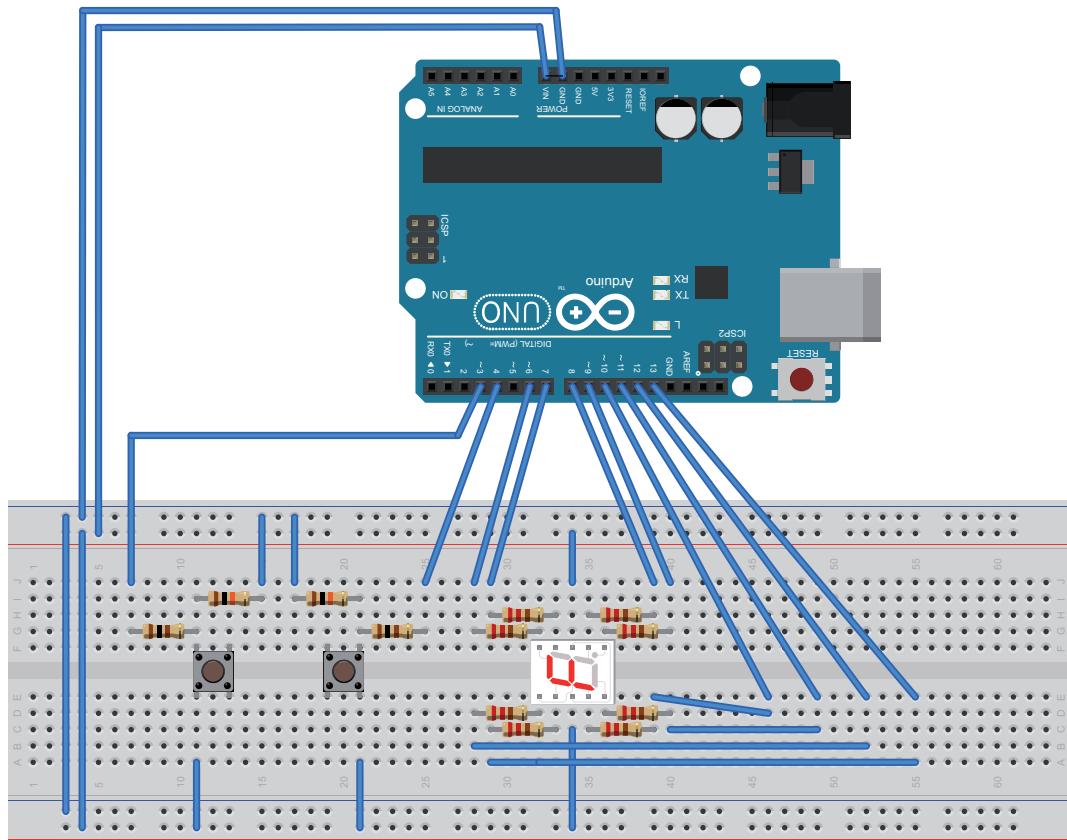


Sur ce schéma, l'affichage 7 segments est à anode commune. Comme d'habitude, chaque bouton-poussoir est doté d'une résistance *pull-down* de $1\text{k}\Omega$ ou $10\text{k}\Omega$ et d'une résistance de protection de l'*input* de 100Ω .

Il est possible de réaliser ce montage sur la petite platine d'expérimentation du Diduino; mais l'idéal est d'utiliser une platine plus grande. C'est ce que nous avons fait dans notre exemple, par souci de clareté.



Plan de montage





Le code

Le code peut être téléchargé à l'adresse suivante: <http://edurobot.ch/code/code16.txt>

Il est très similaire aux codes précédents. La principale différence réside dans le `if(valeur_recue == 1)`, qui dans ce cas signifie "*si la valeur reçue est égale à 1, on allume les segments pour afficher le chiffre 1*". Contrairement au `if(valeur_recue >= 1)`, qui permet de cumuler l'allumage des LEDs (1 LED pour 1, deux LEDs pour 2,...), l'objectif est ici de n'allumer que les LEDs nécessaires à l'affichage du bon chiffre. Il ne faudrait en effet pas qu'un **1** devienne ensuite un **2** au lieu d'un **1**, parce qu'on cumule les LEDs allumées.

```
/*
Code 16 - Edurobot.ch, destiné au Diduino
Apprendre à compter
L'objectif est d'afficher des chiffres sur un afficheur 7 segments. Un bouton
permet d'incrémenter le chiffre, l'autre de le décrémenter.
On compte ainsi de 0 à 9 ou de 1 à 0.
*/
```

```
/*
Déclaration des constantes pour les noms des pattes.
Notre affichage est un 8 segments, avec donc un point. Même si nous ne
l'utilisons pas, nous l'avons quand même câblé.
*/
```

```
const int btn_minus = 3;      //Bouton 1 pour décrémenter les chiffres
const int btn_plus = 4;       //Bouton 2 pour incrémenter les chiffres
const int led_1 = 6;          //Led 1
const int led_2 = 7;          //Led 2
const int led_3 = 8;          //Led 3
const int led_4 = 9;          //Led 4
const int led_5 = 10;
const int led_6 = 11;
const int led_7 = 12;
const int led_8 = 13;
```

```
/*
Déclaration des variables utilisées pour le comptage et le décomptage
*/
```

```
int nombre_led = 0;           //le nombre qui sera incrémenté et décrémenté
int etat_bouton;              //lecture de l'état des boutons (un seul à la fois
mais une variable suffit; en effet, il n'est pas prévu d'appuyer sur les deux
boutons simultanément)
int memoire_plus = LOW;        //état relâché par défaut pour le bouton 2
int memoire_minus = LOW;       //état relâché par défaut pour le bouton 1
```



```

/*
Initialisation des pattes en entrée/sortie: entrées pour les boutons, sorties
pour les LEDs
*/
void setup()
{
    pinMode(btn_plus, INPUT);
    pinMode(btn_minus, INPUT);
    pinMode(led_1, OUTPUT);
    pinMode(led_2, OUTPUT);
    pinMode(led_3, OUTPUT);
    pinMode(led_4, OUTPUT);
    pinMode(led_5, OUTPUT);
    pinMode(led_6, OUTPUT);
    pinMode(led_7, OUTPUT);
    pinMode(led_8, OUTPUT);
}

/*
Et c'est parti pour le programme!
*/
void loop()
{
    //lecture de l'état du bouton d'incrémentation (on lit l'état du btn_plus et
    //on l'inscrit dans la variable etat_bouton)
    etat_bouton = digitalRead(btn_plus);

    //Si le bouton a un état différent que celui enregistré ET que cet état est
    //"appuyé"
    if((etat_bouton != memoire_plus) && (etat_bouton == HIGH))
    {
        nombre_led++; //on incrémente la variable qui indique combien de LEDs
        devrons s'allumer
    }

    memoire_plus = etat_bouton; //on enregistre l'état du bouton pour le tour
    suivant

    //et maintenant pareil pour le bouton qui décrémente
    etat_bouton = digitalRead(btn_minus); //lecture de son état

    //Si le bouton a un état différent que celui enregistré ET que cet état est
    //"appuyé"
    if((etat_bouton != memoire_minus) && (etat_bouton == HIGH))
    {
        nombre_led--; //on décrémente la valeur de nombre_led
    }

    memoire_minus = etat_bouton; //on enregistre l'état du bouton pour le tour
    suivant

```



```

//on applique des limites au nombre pour ne pas dépasser 4 ou 0 (puisque'on a
4 LEDs)
if(nombre_led > 10)
{
    nombre_led = 10;
}
if(nombre_led < 0)
{
    nombre_led = 0;
}

//On crée une fonction affiche() pour l'affichage du résultat
//on lui envoie alors en paramètre la valeur du nombre de LED à éclairer

affiche(nombre_led);
}

void affiche(int valeur_recue)
{
    //on éteint toutes les leds
    digitalWrite(led_1, HIGH);
    digitalWrite(led_2, HIGH);
    digitalWrite(led_3, HIGH);
    digitalWrite(led_4, HIGH);
    digitalWrite(led_5, HIGH);
    digitalWrite(led_6, HIGH);
    digitalWrite(led_7, HIGH);
    digitalWrite(led_8, HIGH);

    //Pour chaque chiffre, le plus simple est de lister toutes les LEDs
    //et de leur attribuer la valeur voulue

        if(valeur_recue == 1)          // "si la valeur reçue est égale à 1, on
allume les segments pour afficher le chiffre 1"
    {

        digitalWrite(led_1, LOW);
        digitalWrite(led_2, HIGH);
        digitalWrite(led_3, HIGH);
        digitalWrite(led_4, LOW);
        digitalWrite(led_5, HIGH);
        digitalWrite(led_6, HIGH);
        digitalWrite(led_7, HIGH);
        digitalWrite(led_8, HIGH);

    }

        if(valeur_recue == 2)          // "si la valeur reçue est égale à 2, on
allume les segments pour afficher le chiffre 2"
    {

```



```
digitalWrite(led_1, LOW);
digitalWrite(led_2, LOW);
digitalWrite(led_3, LOW);
digitalWrite(led_4, HIGH);
digitalWrite(led_5, LOW);
digitalWrite(led_6, LOW);
digitalWrite(led_7, HIGH);
digitalWrite(led_8, HIGH);

}

if(valeur_recue == 3) // "si la valeur reçue est égale...
enfin... tu connais la suite...
{

digitalWrite(led_1, HIGH);
digitalWrite(led_2, LOW);
digitalWrite(led_3, LOW);
digitalWrite(led_4, HIGH);
digitalWrite(led_5, LOW);
digitalWrite(led_6, LOW);
digitalWrite(led_7, LOW);
digitalWrite(led_8, HIGH);

}

if(valeur_recue == 4)
{

digitalWrite(led_1, HIGH);
digitalWrite(led_2, HIGH);
digitalWrite(led_3, LOW);
digitalWrite(led_4, LOW);
digitalWrite(led_5, HIGH);
digitalWrite(led_6, LOW);
digitalWrite(led_7, LOW);
digitalWrite(led_8, HIGH);

}

if(valeur_recue == 5)
{

digitalWrite(led_1, HIGH);
digitalWrite(led_2, LOW);
digitalWrite(led_3, LOW);
digitalWrite(led_4, LOW);
digitalWrite(led_5, LOW);
digitalWrite(led_6, HIGH);
digitalWrite(led_7, LOW);
digitalWrite(led_8, HIGH);

}

if(valeur_recue == 6)
{
```



```
digitalWrite(led_1, LOW);
digitalWrite(led_2, LOW);
digitalWrite(led_3, LOW);
digitalWrite(led_4, LOW);
digitalWrite(led_5, LOW);
digitalWrite(led_6, HIGH);
digitalWrite(led_7, LOW);
digitalWrite(led_8, HIGH);

}

if(valeur_recue == 7)
{

digitalWrite(led_1, HIGH);
digitalWrite(led_2, HIGH);
digitalWrite(led_3, HIGH);
digitalWrite(led_4, HIGH);
digitalWrite(led_5, LOW);
digitalWrite(led_6, LOW);
digitalWrite(led_7, LOW);
digitalWrite(led_8, HIGH);

}

if(valeur_recue == 8)
{

digitalWrite(led_1, LOW);
digitalWrite(led_2, LOW);
digitalWrite(led_3, LOW);
digitalWrite(led_4, LOW);
digitalWrite(led_5, LOW);
digitalWrite(led_6, LOW);
digitalWrite(led_7, LOW);
digitalWrite(led_8, LOW);

}

if(valeur_recue == 9)
{

digitalWrite(led_1, HIGH);
digitalWrite(led_2, LOW);
digitalWrite(led_3, LOW);
digitalWrite(led_4, LOW);
digitalWrite(led_5, LOW);
digitalWrite(led_6, LOW);
digitalWrite(led_7, LOW);
digitalWrite(led_8, LOW);

}
```



```

if(valeur_recue == 10)
{
    digitalWrite(led_1, LOW);
    digitalWrite(led_2, LOW);
    digitalWrite(led_3, HIGH);
    digitalWrite(led_4, LOW);
    digitalWrite(led_5, LOW);
    digitalWrite(led_6, LOW);
    digitalWrite(led_7, LOW);
    digitalWrite(led_8, LOW);

}
}

```

Le code se termine avec le chiffre 0, à défaut du 10.

Il est naturellement tout à fait possible de câbler un deuxième afficheur 7 segments.

Conclusion

Fin!

Ah ! Non ! C'est un peu court, jeune homme !
On pouvait dire... oh ! Dieu ! ... bien des choses en somme...

Lorsque j'ai entrepris la rédaction de ce cours, jamais je n'avais imaginé qu'il me conduirait aussi loin; d'autant plus qu'un tome 2 est en gestation. Mais de fil en aiguille, ou devrais-je dire de résistance en LED, aidé de mes élèves (c'est à eux que je dois l'ultime exercice de cet ouvrage), et entraîné par le rythme effréné de leurs apprentissages, il a bien fallu suivre !

Ici se conclut une année de cours avec mes élèves, à la découverte de cet incroyable outil pédagogique qu'est l'Arduino et, en particulier, le Diduino. Le très grand facteur de motivation qu'il a induit chez mes élèves, et la curiosité qu'ils ont ensuite développée est un vrai plaisir pour un enseignant. À cela, il ne faut naturellement pas oublier les compétences acquises par les élèves: électronique, électricité, informatique, programmation, gestion de projet... C'est avec plaisir que j'ai vu ces élèves, à l'issue de leur scolarité, s'engager dans des professions techniques.

Rendez-vous au tome 2 !