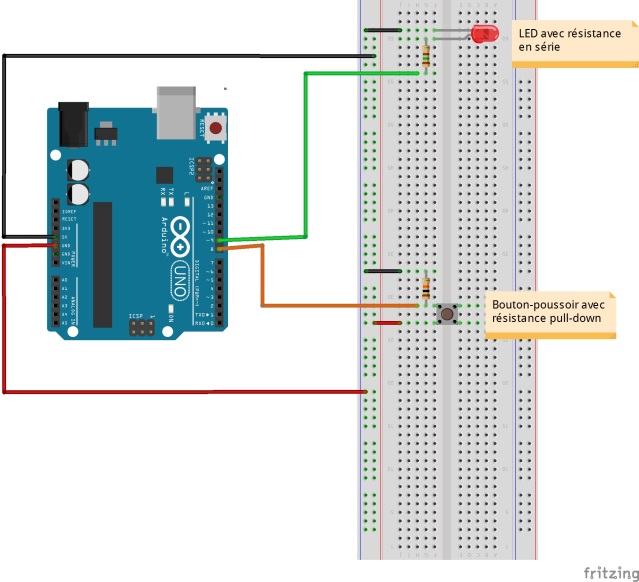
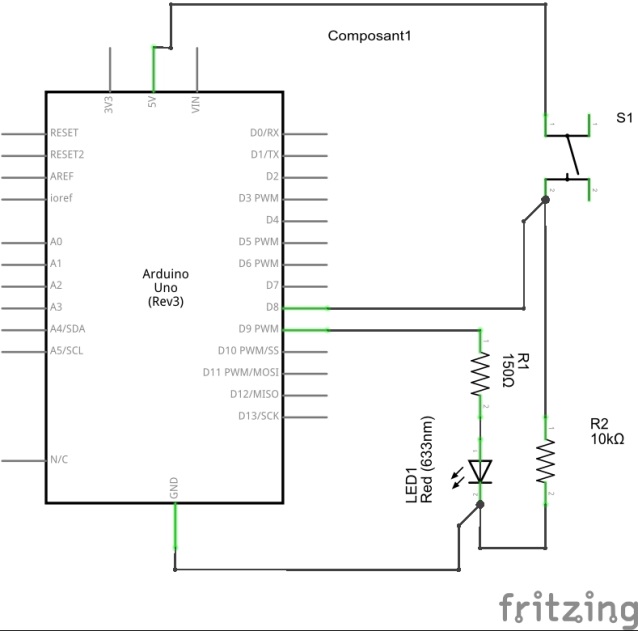
# Intention pédagogique

|  |  |  |  |
| --- | --- | --- | --- |
| **Public** | Tous  MLDS  HPI | Avoir réalisé la Fiche 1.  Avoir déjà réalisé un circuit avec l’arduino. | **Niveau nécessaire**  (fiche déjà réalisée) |
| **Pour quoi ?** | | | |
| **Durée** | **2 heures** | | |
| **Domaine scientifique abordée** | **Electronique, Electricité** | | |
| **Objectifs de la séance** | **Commander une DEL avec un interrupteur, découvrir les résistances de fonction pull-Down et pull-up** | | |
| **Comment ?** | | | |
| **Compétences visées** | * **Découverte la commande pour acquérir une entrée digital.** * **Piloter la DEL en fonction de l’état de l’interrupteur.** | | |
| **Production finale** | **La réalisation individuelle d’un circuit avec un interrupteur et une DEL.** | | |
| **Materiel a mettre à disposition** | * **un ordinateur pour chaque apprenant avec le logiciel arduino (1) installée**   **(1) Pour télécharger le logiciel arduino se rendre sur le site** [**www.arduino.cc**](www.arduino.cc) | | |
| **Matériel apporté par l’animateur** | * **Une carte arduino pour chaque apprenant** * **Un cable de liaison entre l’ordinateur et la carte arduino** * **une DEL (starter kit)** * **un intérrupteur** | | |
| **connaissances acquises lors de l’expérience** | **en arduino :**   * **La programmation d’une broche numérique en Entrée (iNput) ;** * **Structure de controle (IF, Else)** * **enregistrer l’état d’un bouton dans une variable** * **Fonction digitalRead() ;** | | |
| **En electronique :**   * **Résistance PULL-DOWN et PULL-UP** * **Le pont diviseur de tension** | | |

# Schéma électrique





La résistance pour la DEL est de 150 Ω et celle du bouton-poussoir est une résistance de 10KΩ.

[Lien vers la vidéo du montage](Fiche%20n°2/Fiche%20n°2.html)

# Code de l’expérience

const int Led = 8; // LED en broche 8

const int Bouton = 9; // le bouton en borche 9

int BoutonEtat; // variable pour enregistrer l'état du bouton

void setup() {

// initialize the digital pin as an output.

pinMode(Led, OUTPUT); // Broche de la led en tant que sortie

pinMode(Bouton, INPUT); // Broche du bouton poussoir en tant d'entrée

}

void loop() {

BoutonEtat = digitalRead(Bouton);

if (BoutonEtat == HIGH) {

digitalWrite(Led, HIGH); // Allume la LED

}

else {

digitalWrite(Led, LOW); // éteint la LED

}

}

# Revue de CODE

Fonctionnement du programme :

Non

Démarrage de la boucle

Bouton enfoncée ?

Oui

DEL Allumée

Niveau Haut

DEL Eteinte

Niveau Bas

L’organigramme se lit de la manière suivante, lorsque l’exécution du programme arrive dans la partie de la boucle sans fin, nous regardons toujours l’état de l’entrée du bouton-poussoir et enregistrer dans une variable (ici BoutonEtat), La ligne qui permet de faire ce traitement est la suivant :

## Fonction DIGITALREAD() :

Non de la fonction permettant la lecture de l’entrée. Cette fonction prend en paramètre le numéro de l’entrée à lire

**BoutonEtat = digitalRead(Bouton);**

Variable d’enregistrement des valeurs de retour de la fonction digitalRead() ;

Paramètre : Numéro du pin à définir celui-ci peut aussi être inclut dans une variable

Comme vous pouvez le voir la fonction digitalRead() à un élément qui diffère de la fonction vue précédemment digitalWrite() et cet élément est le fait qu’elle revoie une information une fois le traitement terminé. Cette information une information de type booléen (soit HiGH ou LOW). Le traitement de cette information est effectué à la suite par une structure de contrôle de if – else (SI – Alors – Sinon).

## La structure de contrôle (IF – ELSE):

**Forme de la structure de contrôle :**

Nom de l’instruction

Instruction à faire si la condition est Vrai (la partie Si- Alors)

Instruction à faire si la condition est fausse (la partie Sinon)

Condition à vérifier

**if (BoutonEtat == HIGH) {**

**digitalWrite(Led, HIGH); // Allume la LED**

**}**

**else {**

**digitalWrite(Led, LOW); // éteint la LED**

**}**

**△**Une erreur très fréquente, est de confondre l’opérateur d’égalité et l’opérateur d’affectation. L’opérateur d’égalité est « == » et l’opérateur d’affectation est = ont des missions complétement différentes, mais sont souvent utilisés l’un à la place de l’autre.

Nom de la variable = déclaration

Initialisation de la variable

Présentation des différents types de variable

|  |  |
| --- | --- |
| INT | Pour une variable de type nombre entier |
| Char | Pour une variable de type chaine des caractères |
| Boolean | Pour une variable de type binaire |
| Float | Pour une variable de type nombre à virgule |

**Int led = 13** ;

Type de variable

Instruction pour définir le comportement du pin.

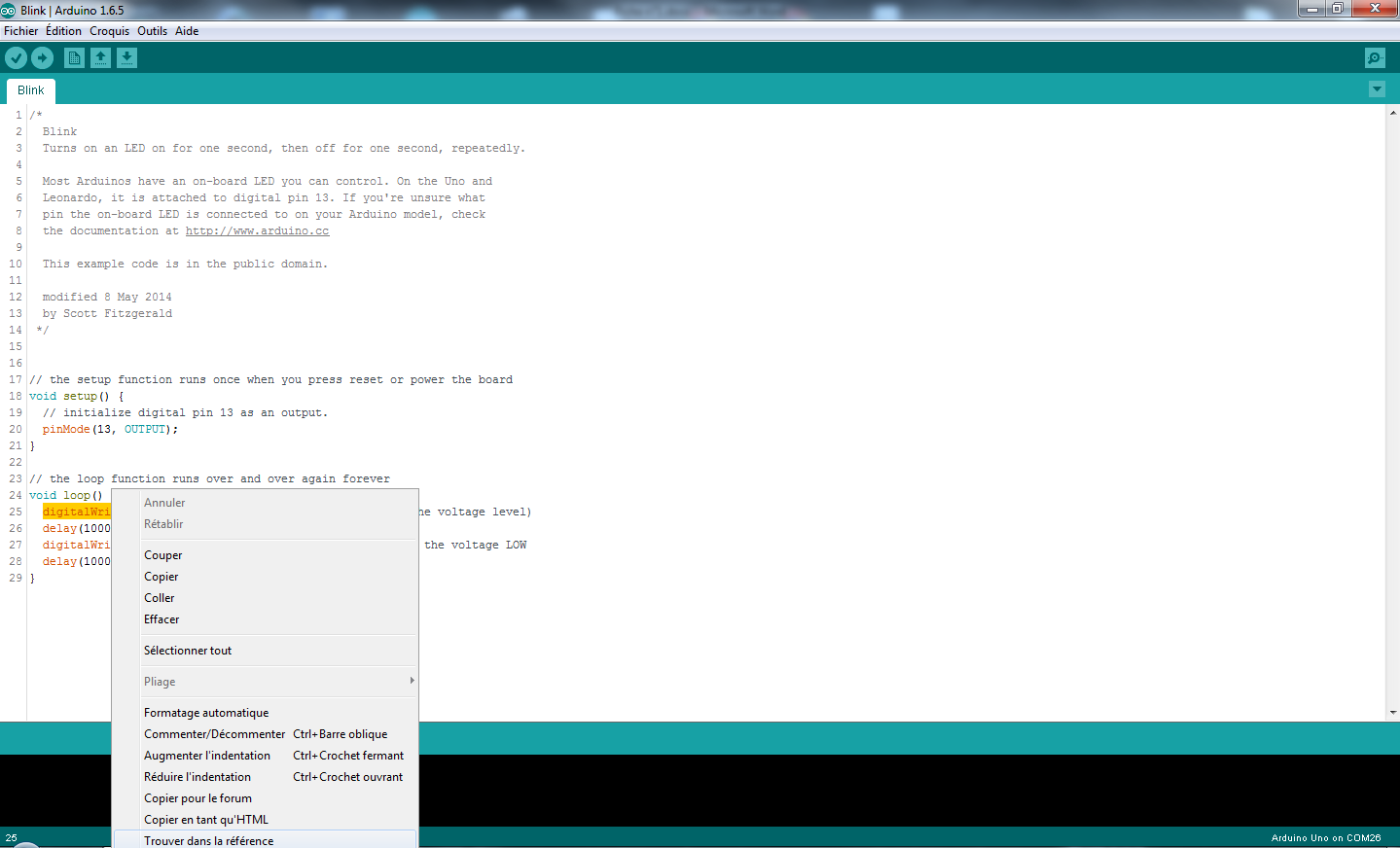
(fonction arduino)

**pinMode(led, OUTPUT);**

Argument 2 : Définition du mode du pin en entrée (INPUT) ou en sortie (OUTPUT).

Argument 1 : Numéro du pin à définir celui-ci peut aussi inclut dans une variable

Une fonction en informatique utilise le plus souvent des arguments. Ils se trouvent entre les parenthèses juste après le nom de la fonction. Pour connaitre la position des arguments et leur utilité, vous pouvez vous reporter à la référence d’arduino, en sélectionnant le nom de la fonction, en faisant un clic droit et en cliquant sur « Trouver dans la référence ».



Instruction pour mettre en pause l’évolution du programme (fonction arduino)

Instruction pour définir l’état du pin soit Haut ou Bas.

(fonction arduino)

**digitalWrite(led,HIGH) ; delay(1000) ;**

Argument 1 : permet de définir le temps d’attente en milliseconde.

Nota : cette fonction bloque intégralement l’arduino pendant le temps indiqué, peut créer des difficultés lorsque l’on attend le changement d’un état d’une entrée si celle-ci intervient lors du blocage.

Argument 2 : Définition de l’état de pin soit à l’état Haut à 5V (HIGH) soit à l’état bas à 0V (LOW)

Argument 1 : Numéro du pin à définir celui-ci peut aussi inclut dans une variable

# Chronologie de la séance

Toutes les étapes énoncées par la suite doivent être réalisées par les apprenants.

1. Présentation par l’intervenant sur un arduino, le résultat attendu pour la fin de la séance.
2. Définir le fonctionnement du programme que l’on souhaite faire dans la boucle « loop »,

1 – Allumer la DEL de la broche 13.

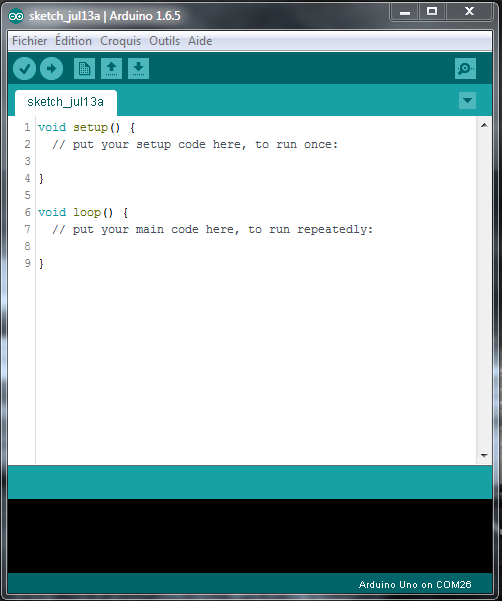
2 – Attendre une seconde.

3 – éteindre la DEL de la broche 13.

4 – Attendre une seconde.

5 – Revenir au point 1, puis recommencer

1. Ecrire sur un paperboard le programme de la partie « loop »
2. Définir le nombre de variables utilisées dans la loop
3. Définir le type des variables
4. Une fois tout cela fait, écrire sur paperboard le programme de la partie « setup »
5. Pour finir, donner à chaque apprenant un arduino, une DEL, et le câble de liaison. Chaque apprenant peut aller recopier le programme fait collégialement dans le logiciel arduino.
6. Une fois le programme écrit, nous montrons le bouton permettant de vérifier si le code a correctement été écrit et le bouton permettant d’envoyer le programme vers la carte arduino.



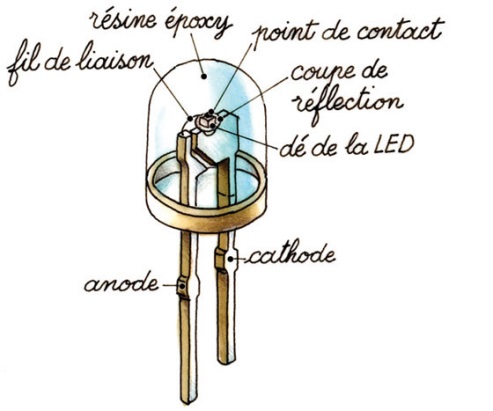
Bouton pour transférer le code vers la carte arduino

Bouton pour vérifier le code

1. Après toutes ces étapes, la DEL devrait clignoter.

Pour aller plus loin, il est possible de faire changer le temps pendant lequel la DEL est éteinte ou allumée. Et si le temps le permet, nous pourrons commencer à aborder la notion de modulation de largeur d’impulsion, et de rapport cycle, le temps d’une période en traçant le chronogramme.

# Astuce

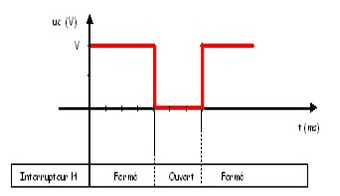


Patte la plus grande à brancher sur le +

Patte la plus petite à brancher sur le -

# Exercice complémentaire

Dans cette exercice complémentaire, je vous propose de modifier le programme de telle sorte que le temps durant lequel la DEL est allumée ou éteinte soit déterminé par deux variables, afin de pouvoir changer facilement le rapport cyclique. Ce dernier est par le rapport du temps de conduction sur le temps de la période totale.



T

t

Rapport cyclique =.

Consigne : Programmer le programme de telle sorte que la DEL reste allumée pendant 500 ms et éteinte pendant 1s. Calculer le rapport cyclique.

Réponse : le rapport cyclique est de 33%.