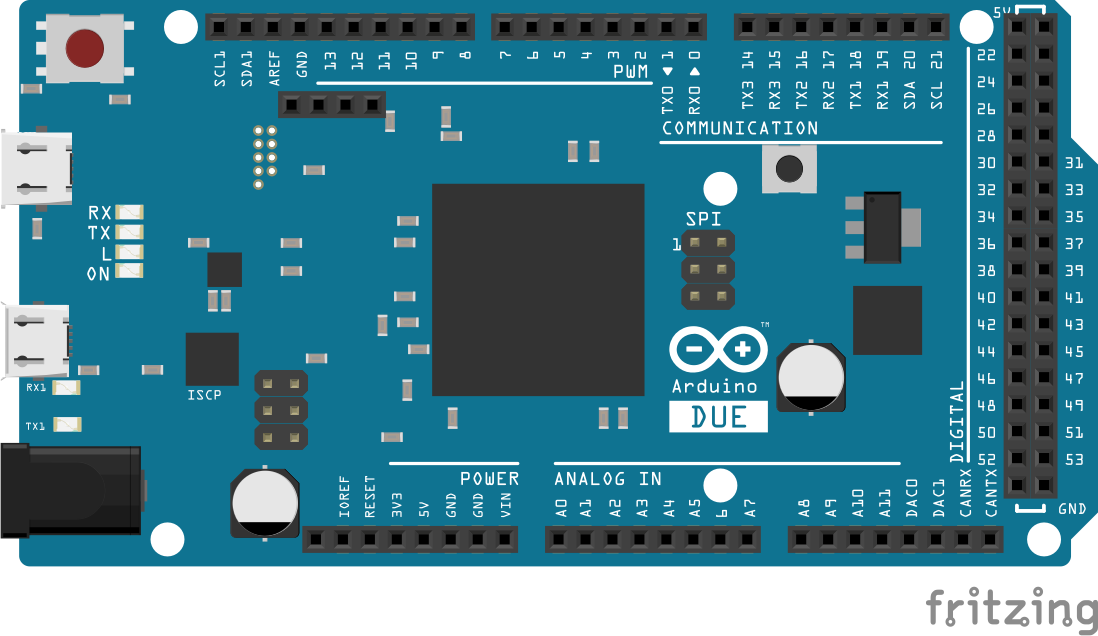


2017-2018

Rapport de TIPE

NAHMIAZ Thomas REBECQ Victor



Est-il possible de faire un jeu sous Arduino ?

PTSI 1

Table des matières

[Introduction : 2](#_Toc513470048)

[Gérer plusieurs modules en même temps : 3](#_Toc513470049)

[Faire un jeu réutilisable : 5](#_Toc513470050)

[Solutions : 5](#_Toc513470051)

[Mise en œuvre : 5](#_Toc513470052)

# Introduction :

L’objet de ce TIPE sera de réaliser un jeu physique sur arduino Nous avons choisi le thème d’une bombe à désamorcer par équipe de deux joueurs.

Nous nous sommes basés sur le cahier des charges suivant pour réaliser ce projet :

|  |
| --- |
| Exigences : |
| Faire un jeu : Priorité aux interactions entre les joueurs |
| Utiliser le minimum de pins |
| Gérer plusieurs modules en même temps |
| Utiliser des sons |
| Avoir un programme propre |
| Pouvoir s’échanger des données pendant le projet |
| Avoir un manuel propre et intuitif |
| Faire un jeu réutilisable |
| Pouvoir ajouter d’autres modules facilement |

Critique du projet (à mettre à la fin) : arduino n’est pas un bon support pour des taches complexes il vaut mieux utiliser des automates.

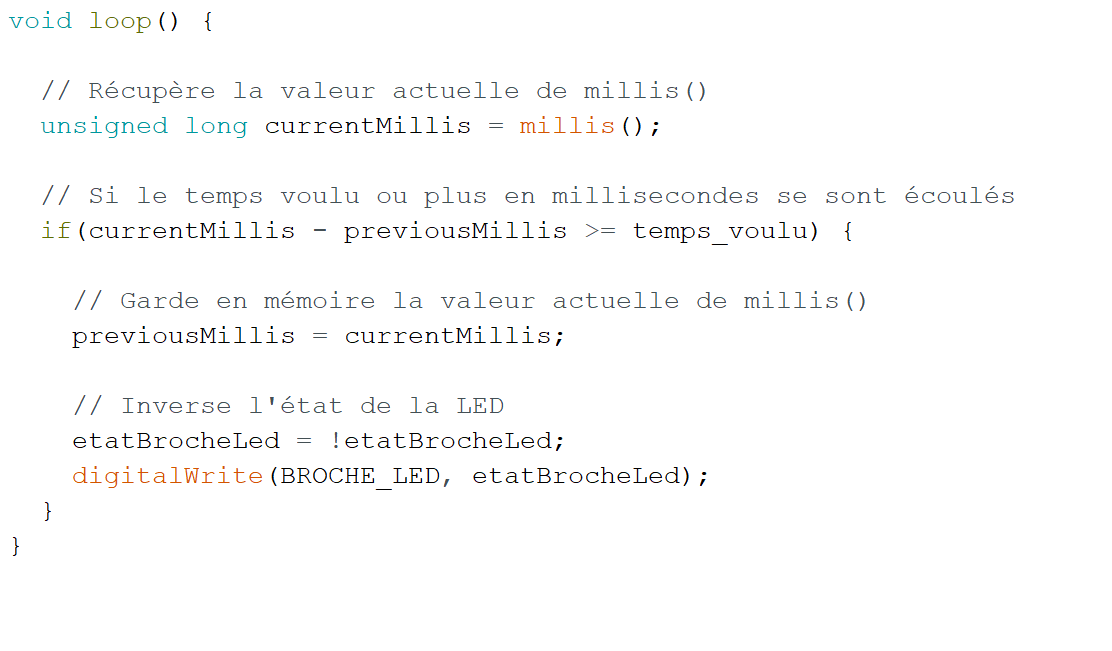
Il faut parler de fritzing aussi

Nous avons en effet écrit les programmes de chaque module puis les avons réunis.

La bombe en elle-même : Au total, 3 erreurs peuvent être faites avant que la partie ne soit perdue (avant que la bombe « n’explose »). Chaque module devra être en mesure d’incrémenter une variable globale décomptant les erreurs lorsqu’une est faite.

# Gérer plusieurs modules en même temps :

La fonction Delay est une fonction bloquante qui ne permet pas d’effectuer plusieurs tâches en parallèle. Nous avons tout d’abord utilisé la fonction attachInerrupt qui à chaque appui sur un bouton effectue une tâche en priorité sur les autres. Cette fonction n’est cependant pas durable sur des programmes longs. Nous finalement opté pour l’utilisation de la fonction millis() qui gère le temps sans être une fonction bloquante selon le modèle :



Ici l’inversion de l’état de la LED est un exemple d’action.

# Echange des données pendant le projet :

Nous avons utilisé le logiciel GitHub Desktop afin de réunir les programmes et autres fichiers dans un dossier en ligne. Cela permet le partage des programmes et d’autres fichiers et de les ouvrir sur n’importe quel ordinateur ayant les droits d’accès au projet.

Module Morse :

Objectif : déchiffrer un mot codé en morse.

Cahier des charges : Le mot est choisi parmi neuf mots au hasard a chaque démarrage de la carte. Le mot doit être transcrit par une LED qui clignotera depuis l’allumage de la carte jusqu’à la résolution du module. Une LED verte s’allumera alors. La saisie de la réponse se fera par l’appui répété sur un bouton correspondant au chiffre associé à chaque mot. La réponse du module (désamorcé ou erroné) se fera 5 secondes après le dernier appui du démineur sur le bouton

On a commencé par utiliser la fonction random or celle-ci n’apporte qu’une valeur par téléversement et pas par démarrage de la carte. Nous avons donc choisi la fonction Randomseed qui a chaque démarrage ou reset de la carte choisi une valeur aléatoire parmi une plage de chiffres en fonction des influx physiques d’une pin de la carte.

Nous avons utilisé les commande switch case pour choisir le mot en fonction de la valeur apportée par la commande Randomseed : plus simple que d’utiliser if else

Afin de pouvoir choisir n’importe quel mot rapidement nous convertirons les mots en morse directement dans le programme afin que le codage du mot soit aussi plus sûr (méthode de traduction automatisée)

Pour cela nous utilisons la commande .Replace qui change un caractère en un autre caractère ou suite de caractère.

N’ayant pas de distinction entre les lettres majuscules et minuscules en morse nous mettons toutes les lettres en majuscules afin de réduire le nombre de caractère à traduire en morse grâce à la fonction .ToUpperCase.

Pour transcrire le message morse en signal lumineux nous utilisons la fonction .substring pour sélectionner chaque caractère du message.

Le processus de clignotement de la LED s’est d’abord fait grâce à la fonction Delay qui bloque la carte durant une valeur indiquée par l’utilisateur. Or la fonction Delay est une fonction bloquante, la carte Arduino ne pouvais donc pas effectuer d’autres taches en même temps comme par exemple l’affichage du temps restant au démineur/désamorceur de la bombe à chaque instant. Ce problème a été l’enjeu majeur de ce module. Nous avons utilisé la fonction millis, plus difficile à utiliser pour traduire un temps d’attente de la carte mais celle-ci n’est pas une fonction bloquante, elle utilise l’avantage de la carte Arduino et de tous les autres appareils électroniques : la vitesse de calcul. Le décompte de chaque milliseconde par la carte permet de définir un moment à partir duquel effectuer une tache.

Dans le message traduit en morse il y’a quatre caractères : « . » , « - », « ; », « | » qui correspondent respectivement au court, au long, à l’espacement entre deux lettres et à l’espacement entre chaque mot. Dans ce projet, un seul mot est traduit, « | » correspond donc l’espacement entre chaque mot mais également à l’espacement entre chaque cycle puisque que la LED traduit le mot en boucle.

Nous avons rencontré un autre problème,

# Faire un jeu réutilisable :

Solutions :

Nous avons décidé d’utiliser des valeurs aléatoires et de programmer plusieurs chemins possibles ne fonctions de ces valeurs pour que le jeu présente différents cas et qu’ainsi les parties de soient pas répétitives. Ces différents chemins se feront directement à l’intérieur du module comme pour le morse ou au travers du numéro de série pour le piano. Le chemins changent à chaque redémarrage de la carte pour proposer des chemins différents à chaque nouvelle partie.

## Mise en œuvre :

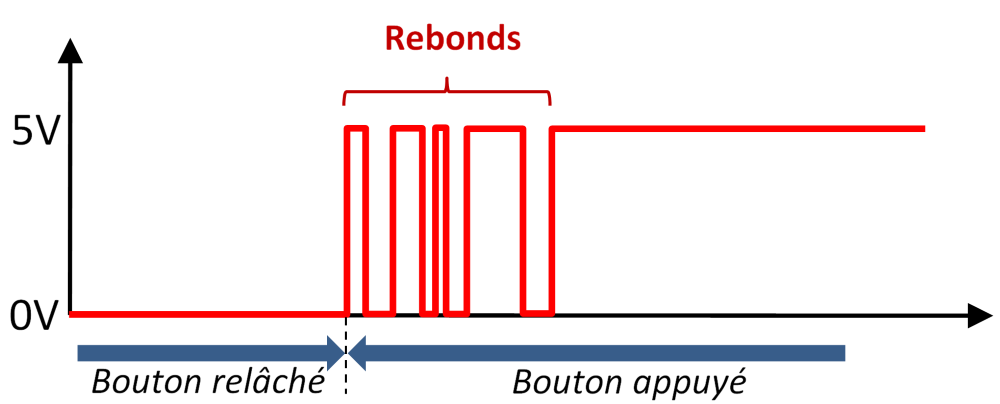
Nous avons utilisé la fonction random(min, max) pour générer des variables aléatoires. Il est à noter que la valeur minimale est inclue et que la valeur maximale est exclue, c’est pourquoi dans le module morse pour neuf mots avons pris la valeur alea = random(1,10).

La fonction random ne génère qu’une valeur aléatoire par télé versement, et non pas à chaque redémarrage de la carte. Nous avons donc rajouté la fonction randomSeed(analogRead(0)) qui lit une grandeur physique sur la pin 0 qui n’est branchée à rien. Cette fonction permet de générer une valeur aléatoire différente à chaque mise sous tension de la carte en plus d’obtenir des valeurs plus aléatoires.

# Autres problèmes rencontrés :

## Rebond du bouton :

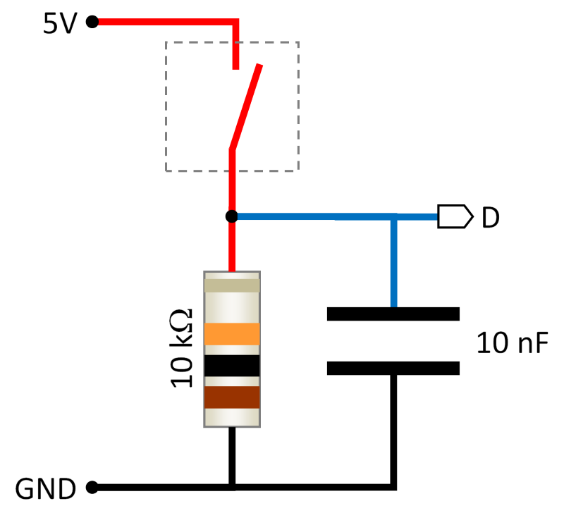
L’appui sur un bouton poussoir peut ne pas transmettre qu’un seul contact mais plusieurs, dépendant de la qualité du bouton.



<http://arduino.blaisepascal.fr/index.php/2017/11/30/le-bouton-poussoir/> (schéma pour l’utilisation d’un condo)

Il y’a deux façons de résoudre ce problème :

Utiliser un condensateur en parallèle pour adoucir le passage transitoire



Ou attendre logiciellement un court instant (250 millisecondes) avant le décompte d’un autre appui sur le bouton pour être sûr de ne pas prendre en compte les rebonds. C’est la deuxième solution qui a été choisie.