HeXart Care

Un cœur qui bat au rythme de vos rêves !



**Sommaire :**

1. **Module n°1 : Module Cardio**
2. **Réalisation du montage**
3. **Programmation Arduino**
   1. Afficher la fréquence cardiaque
   2. Placer le programme sur différents fichier (.c et .h)
   3. Placer les résultats dans un fichier .csv (fréquence cardiaque + le temps depuis le début du programme)
4. **Module n°2 : Cœur de LED**
5. **Réalisation du montage**
6. **Programmation Arduino des LEDS (au rythme du cœur)**
   1. Allumer les LEDS en même temps, puis les éteindre (clignotant)
   2. Allumer les LEDS 1 sur 2
   3. Allumer les LEDS 1 sur 3
   4. Allumer les LEDS 1 sur 4
   5. Allumer les LEDS 1 sur 5
   6. Allumer les LEDS 1 sur 6
   7. Allumer les LEDS 1 sur 7
   8. Allumer les LEDS 1 sur 8
   9. Allumer les LEDS 1 sur 9
   10. Allumer les LEDS 1 sur 10
   11. Allumer les LEDS en mode chenille
   12. Séparer le programme avec des fichiers cœur.c + cœur.h + param.h
7. **Programmation en C**
   1. Faire un programme en C permettant de gérer la création du param.h (avec des fichiers main.c, menu.c, menu.h, generationCode.c, generationCode.h)
   2. Tester le programme avec les LED
8. **Module n°3 : Processing et acquisition de données**

Utiliser processing pour envoyer les données de fréquence cardiaque sur un fichier .csv

1. **Module n°4 : Lecture et traitement de données**
2. **Réalisation en C**
3. Crée une structure
4. Réalisation du code en C : menu.c et .h ainsi que donnée .c et .h, pour finir actions .c et .h
5. Afficher fichier csv : Afficher les données dans l’ordre du fichier du csv :

• Afficher les données en ordre croissant/décroissant (selon le temps, selon le pouls)

• Rechercher et afficher les données pour un temps particulier

• Afficher la moyenne de pouls dans une plage de temps donnée

• Afficher le nombre de lignes de données actuellement en mémoire

• Rechercher et afficher les max/min de pouls (avec le temps associé)

• Autres (facultatif)…

* Quitter l’application

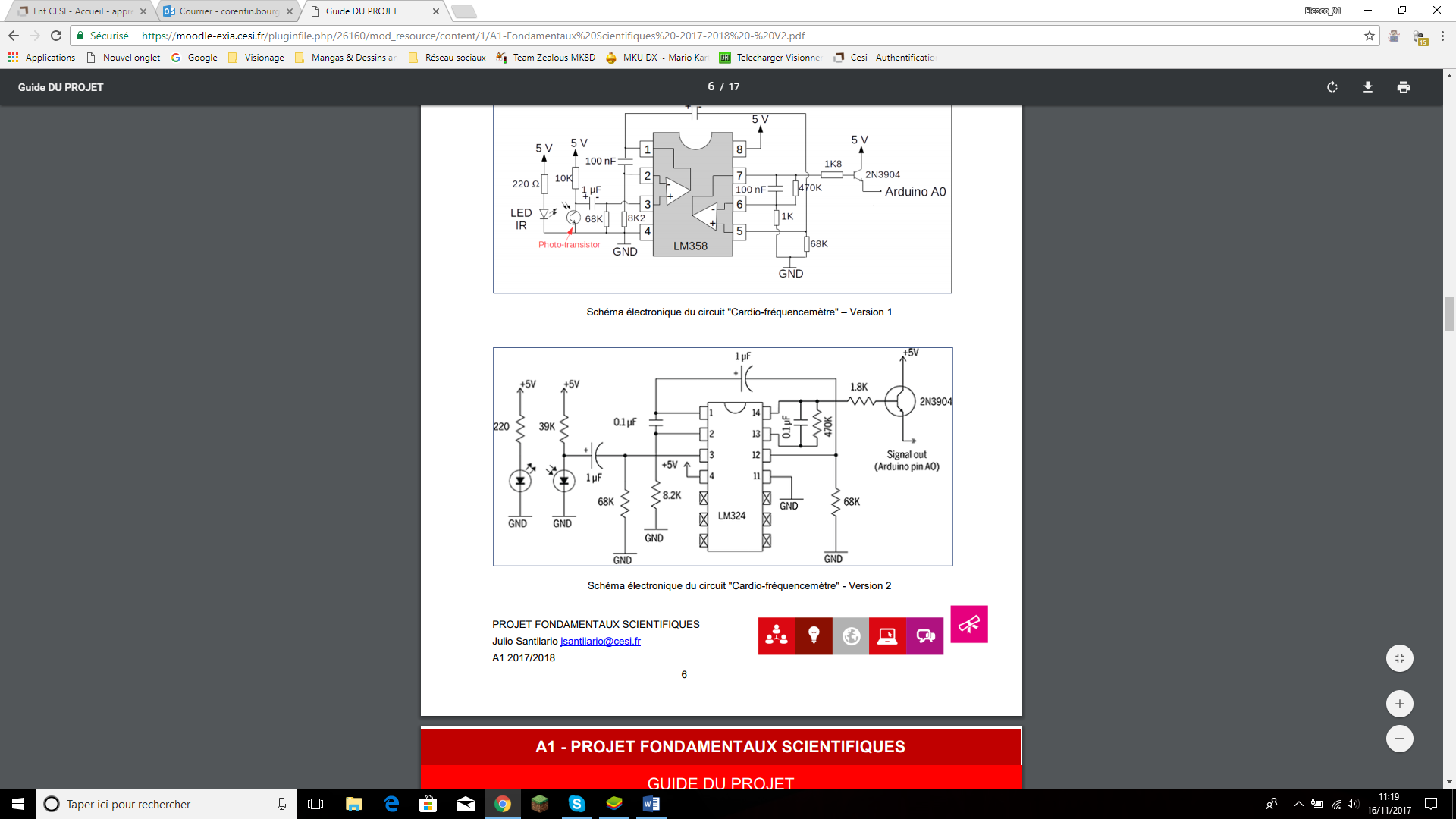
**Module n°1 : Module Cardio**

**Réalisation du montage :**

Condensateur

Nous devions réaliser le montage suivant :

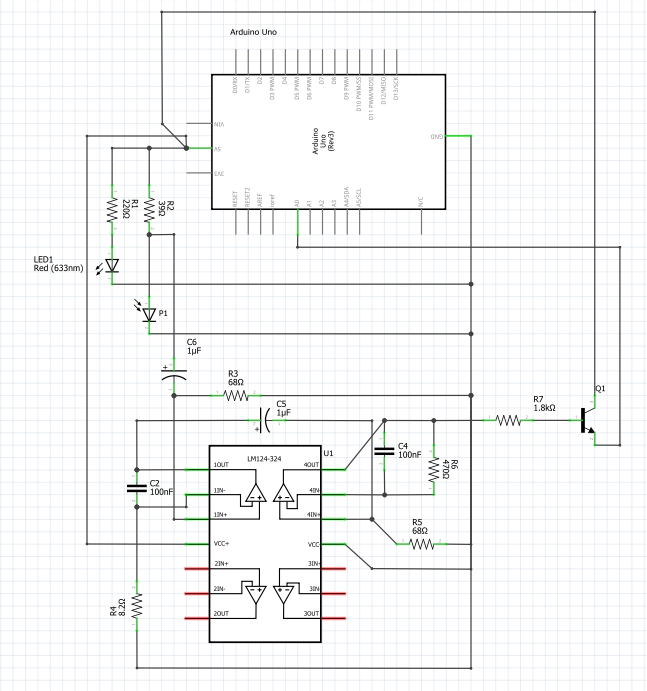
Transistor NPN

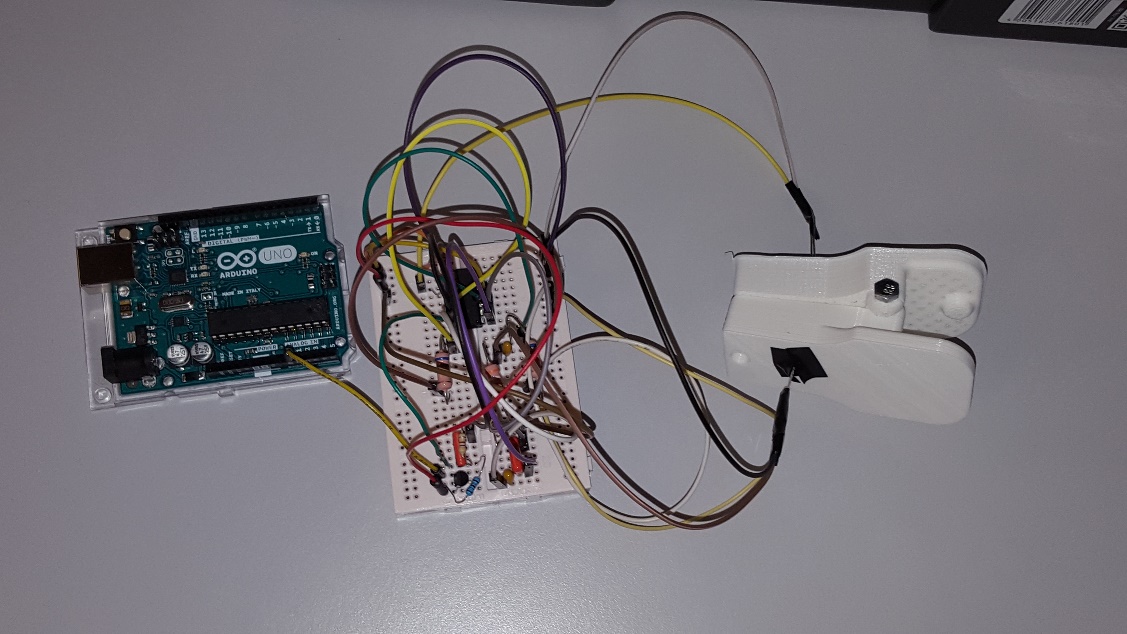


Phototransistor

LED (infrarouge)

Fritzing :





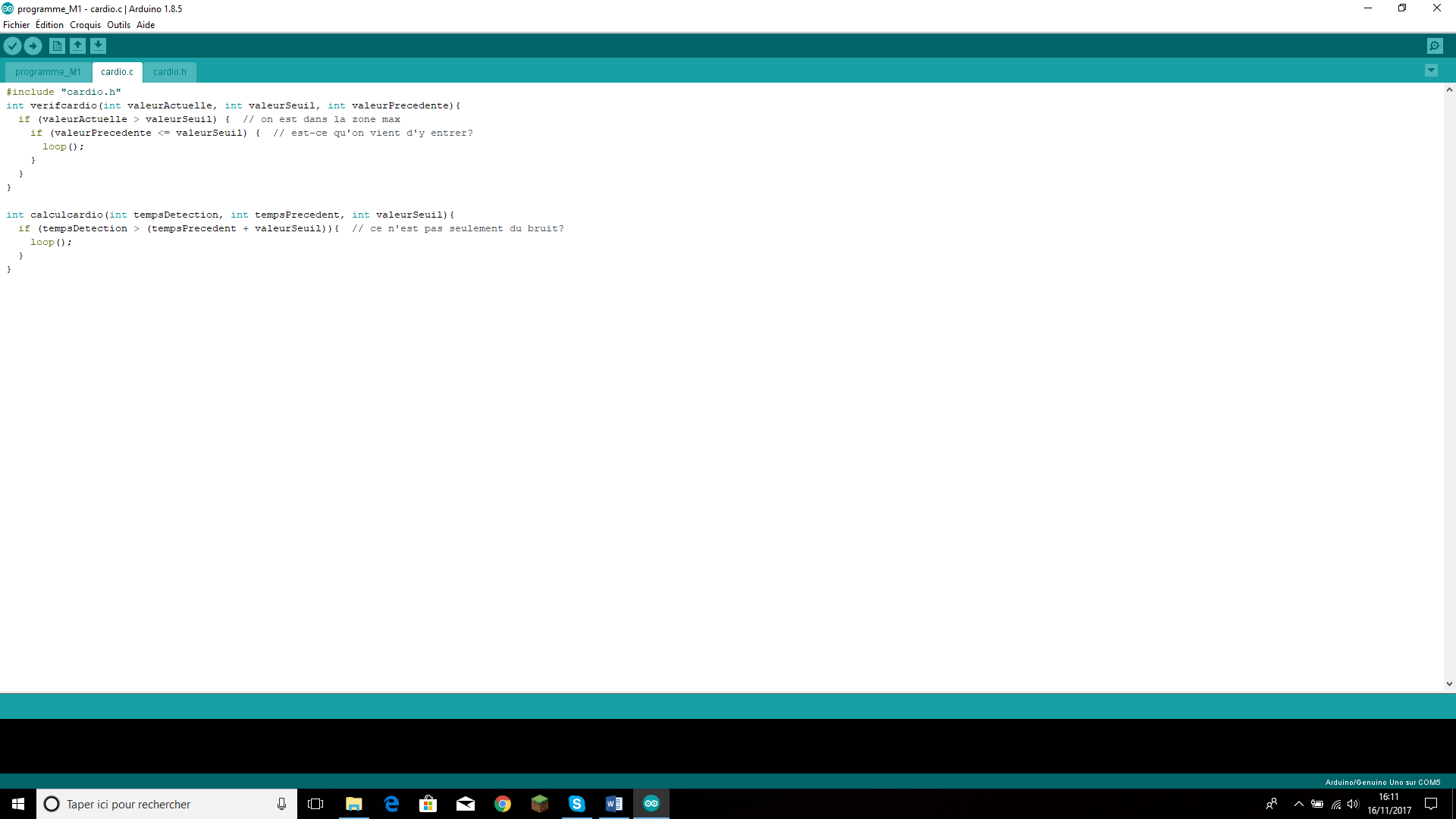
Ce montage avait pour but de renvoyer l’intensité lumineuse de la LED si nous mettons notre doigt entre la LED et le phototransistor.

Le phototransistor a pour but de détecter la lumière émise. Donc comme le volume de sang dans notre doigt ne sera pas le même entre les moments ou notre cœur bat, la lumière absorber par le sang n’est pas la même. Ainsi, nous pouvons déterminer une fréquence cardiaque.

**Programmation Arduino :**

Le programme Arduino est composé de de 3 fichiers. Le fichier cardio.h, cardio.c et le fichier main en . ino.

**Voici une partie importante du code :**

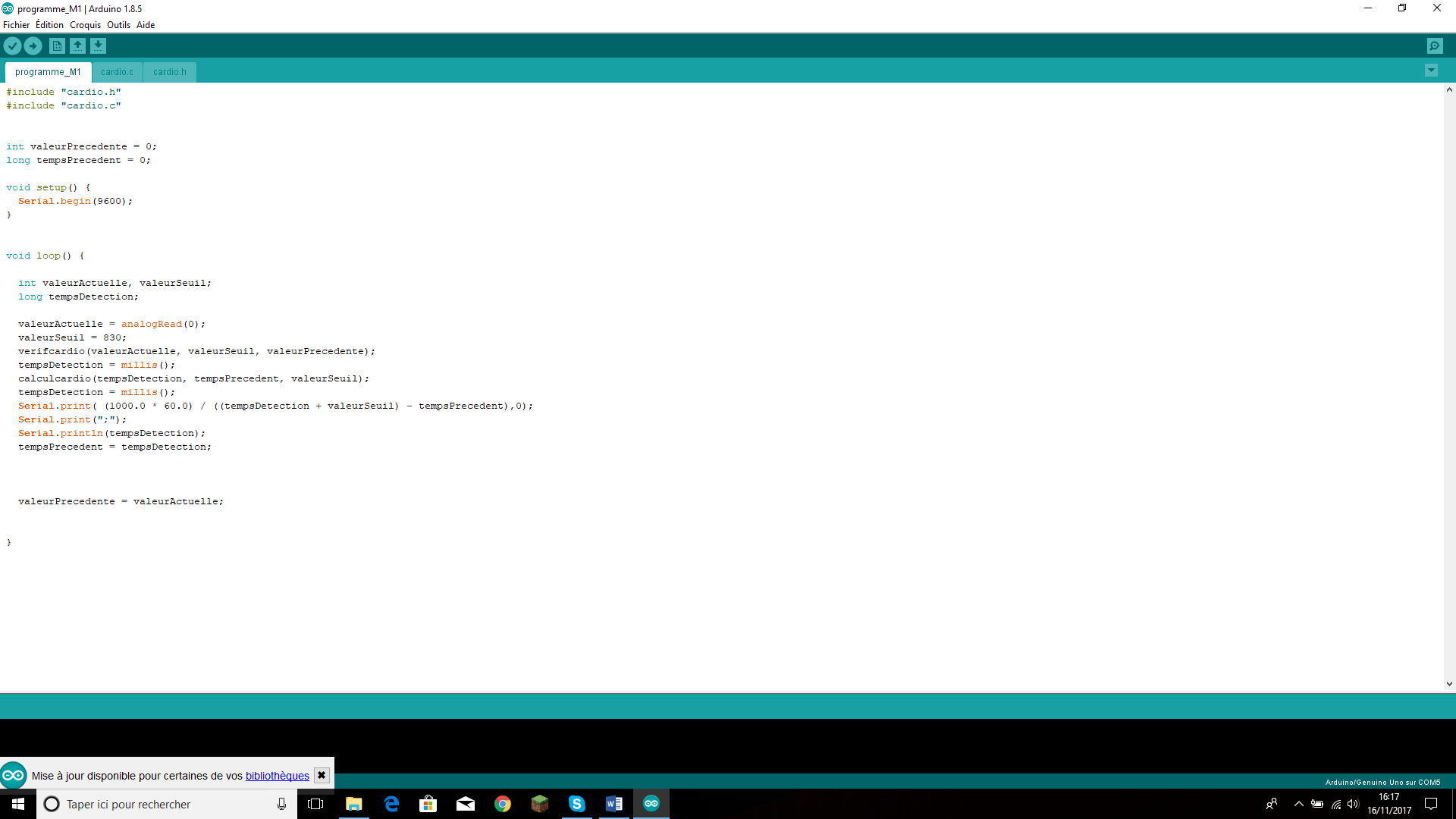


Ce programme permet de savoir le moment où nous avons une pulsation cardiaque. En effet, le phototransistor renvoie 2 valeurs différents : Les 2 intensités lumineuse lorsque le sang passe dans le doigt et le moment où il ne passe pas.

Donc un battement est la moyenne de ces 2 valeurs.

Le but du code est donc de voir si nous passons au-dessus de cette valeur et ensuite si nous passons en dessous. Si c’est le cas, nous avons un battement et nous pouvons donc calculer la fréquence cardiaque.

**Calcul de la fréquence cardiaque :**



Nous calculer nombre de battements par minutes grâce à un produit en croix :

1 battement 🡪 nombre de seconde trouver pour un battement

X battements 🡪 60 secondes

Donc comme le nombre de seconde d’un battement est (tempsDetection-tempsPrecedant).

De plus le 1000\*60 permet de passer en seconde car l’Arduino réfléchit en milliseconde (1000 ms = 1 s).

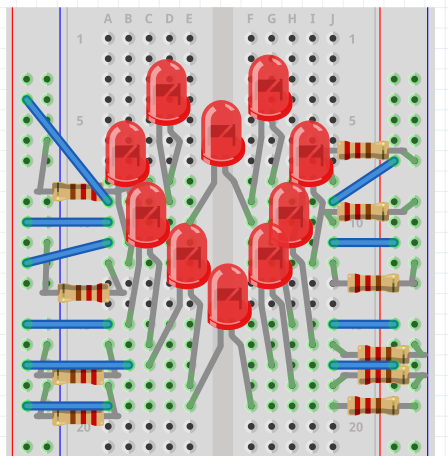
**Récupération de valeur avec Processing :**

Cette partie était demandée dans le module n°1 mais elle nous était expliquée dans le module n°3. Nous verrons donc cette partie plus tard.

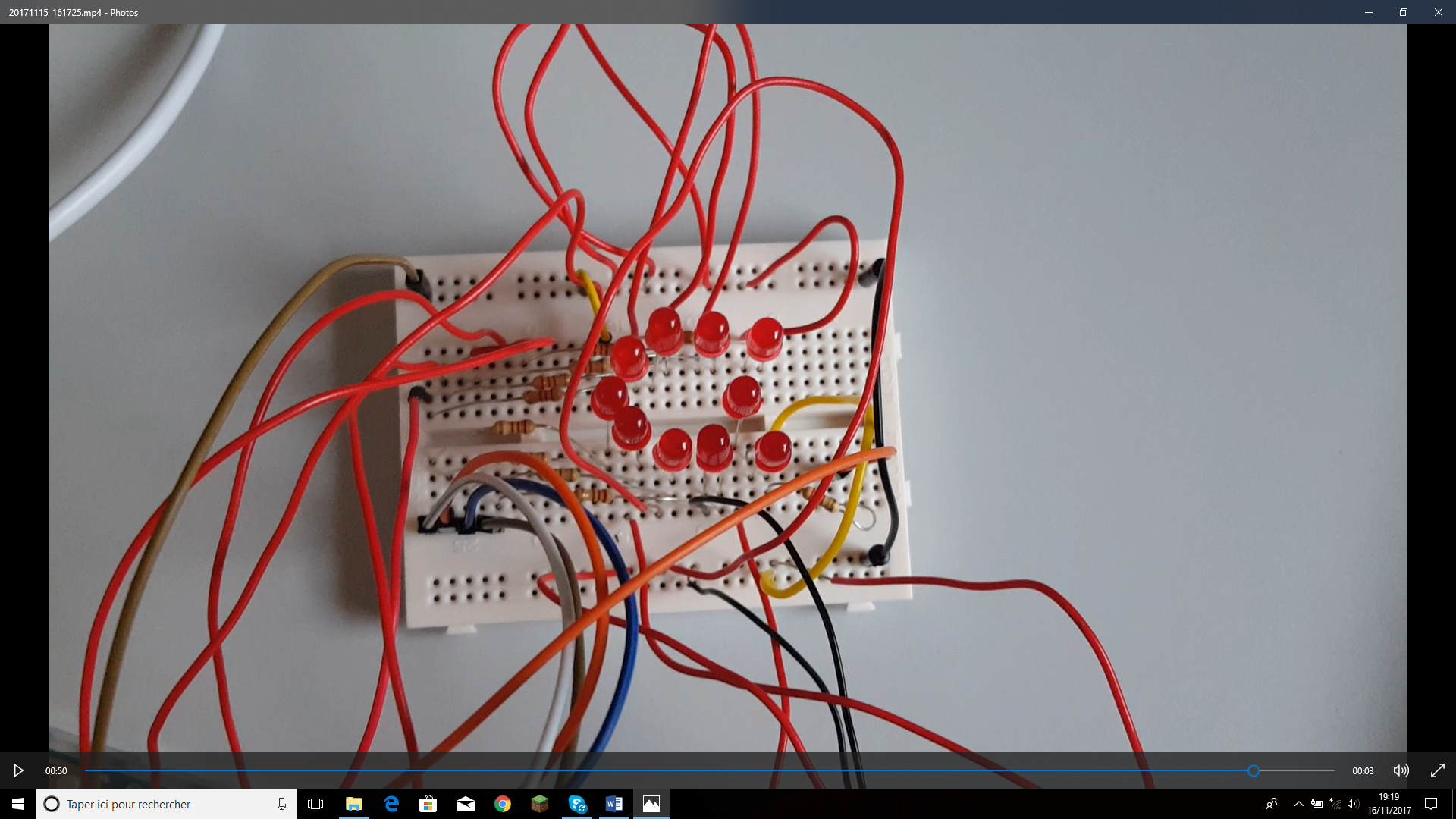
**Module n°2 : Cœur de LED**

**Réalisation du montage :**

Sur Fritzing :



Avec le matériel :



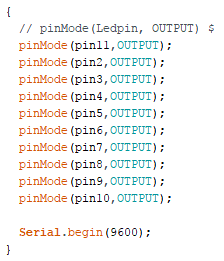
Ce montage et ce module ont pour but de faire clignoter les LED.

**Programmation Arduino :**

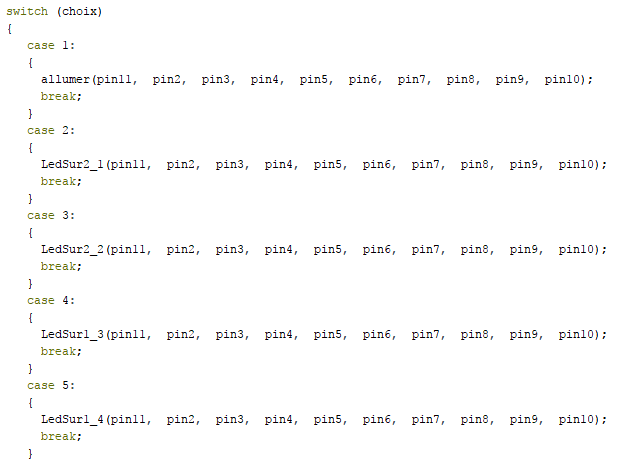
Dans cette partie, nous allons réunir par soucis pratique toutes les fonctions des différents types d’allumage de LED. En effet, les fonctions sont presque identiques : la seule chose qui change est le numéro des LEDS à allumer puis éteindre.

Dans le fichier main .c :

Pour commencer, on initialise tous les ports, puis on leur assigne l’OUTPUT pour qu’ils émettent du courant ce qui va permettre de pouvoir gérer le courant des LEDS.



Puis à l’aide d’un switch nous allons définir toutes les fonctions d’allumage que nous allons pouvoir réaliser avec le cœur (toute les LEDS sont allumer, ou les LEDS s’allume les unes après les autres).

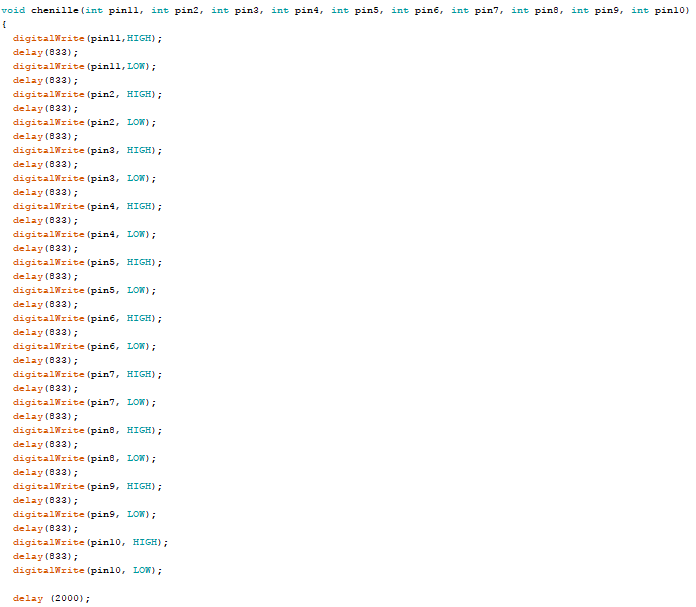


Dans la fonction choix que nous avons initialiser dans le fichier param .h, nous avons toutes possibilités d’allumage des LEDS.

Dans le fichier cœurs .h :

Nous allons vous montrer l’exemple pour une fonction mais toutes les autres fonctions se ressemble, la différence sont les sorties, si les LEDS sont allumées ou éteintes.

Prenons l’exemple de l’allumage des LEDS en chenille :



Nous disons d’allumer la première LED durant une période de temps, en l’occurrence nous avons pris avec mon battement de cœurs a 72, la fréquence associée est 833. Pour que cela clignote à la vitesse de mon battement de cœurs on a donc mis un délai de 0.833 secondes, dans le but que ça clignote 72 fois en 1 minute.

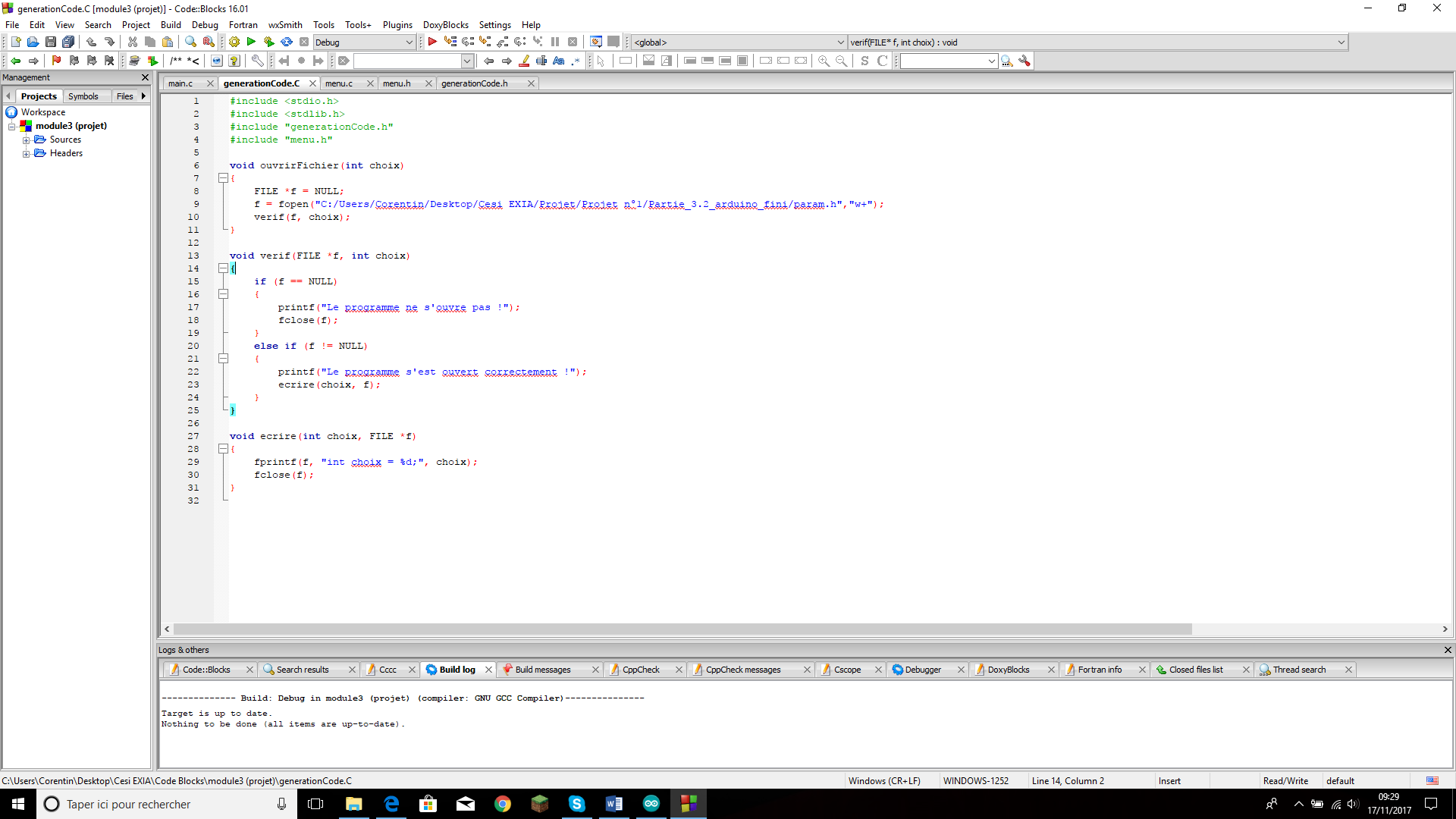
Puis nous éteignons les LEDS et nous faisons clignoter la suivante et ainsi de suite.

Pour les autres fonctions, le principe est toujours le même, nous avons pris le choix de décrire la fonction la plus technique à nos yeux.

**Programmation en C :**

Dans cette partie nous devions créer un programme en C qui modifie le fichier param.h qui est liée à l’Arduino pour que les LEDS s’allument comme nous le souhaitons.

Premièrement nous affichons dans la console les différents types d’allumage de LED que nous voulons. Puis nous entrons la valeur du choix. Le but maintenant est d’envoyer le choix sur le fichier param.h.



Dans la fonction ouvrir fichier, comme son nom l’indique, nous ouvrons le fichier. Ensuite nous vérifions s’il est ouvert, et si c’est le cas, nous écrivons dans param.h :  « int choix = x ; ».

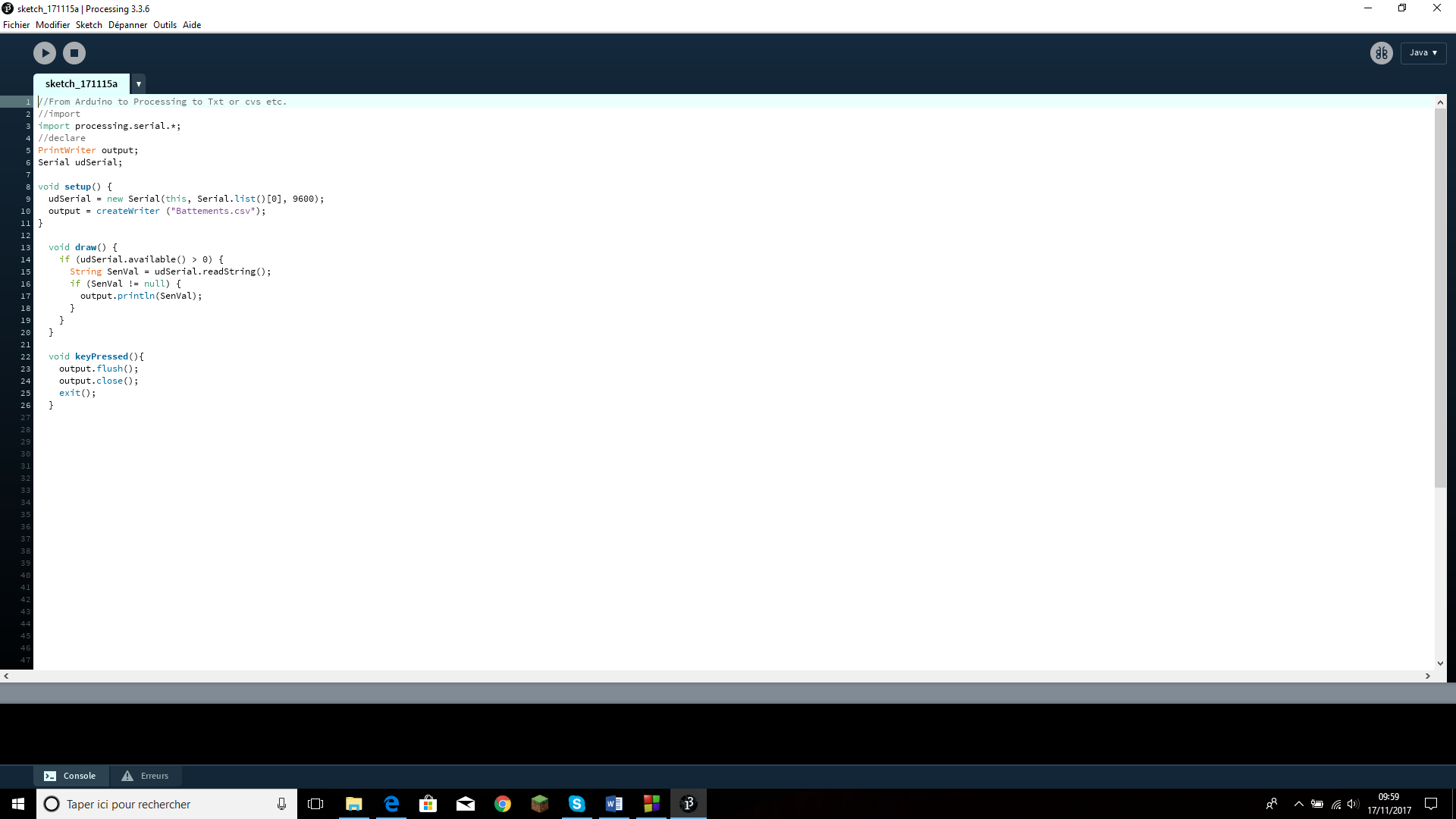
In ne reste qu’à démarrer Arduino et tester.

**Module n°3 : Processing et acquisition de données**

**Envoyer les données :**

Dans cette partie, nous devions envoyer les données de la fréquence cardiaque ainsi que le temps depuis le début du programme dans un fichier csv.

Pour cela, nous utilisons processing :



Envoyer les valeurs dans battement.csv

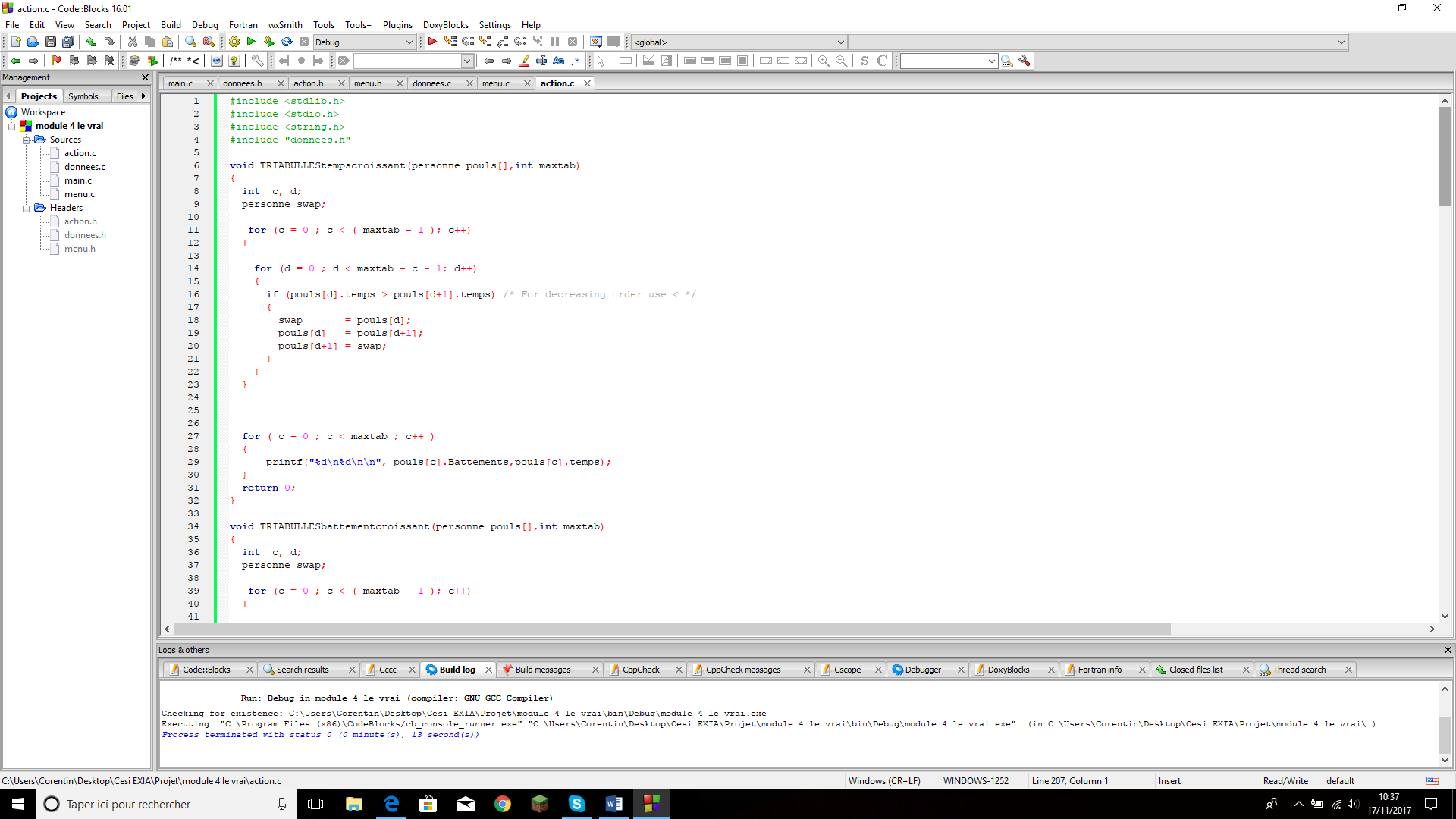
Le logiciel va donc créer un fichier csv dans le même dossier que le fichier processing. Dans la première colonne, nous avons la fréquence cardiaque et dans la deuxième colonne nous avons le temps depuis le début du programme.

**Module n°4 : Lecture et traitement de données**

**Classer par ordre croissant et décroissant (pouls + temps) :**

Nous avons ici, 4 fonctions différents…

Cependant, ces 4 fonctions fonctionnent de la même manière donc nous allons en expliquer une seule :



Dans cette situation, nous avons un tri à bulle. Nous avons une boucle qui va tourner sur possibilité et si la première valeur est supérieure à la deuxième, nous envoyons la première valeur dans un tableau de sauvegarde. Ensuite la première valeur prend la valeur de la deuxième. Puis la deuxième valeur prend la valeur de sauvegarde.

Ainsi, nos valeurs sont triées.

Pour trier en décroissant, c’est juste l’inverse.

**Algorithme de recherche :**

Il s’agit d’une recherche dichotomique. Nous prenons la valeur au milieu de la liste (trier à l’avance) et nous comparons avec les valeurs se trouvant à droite et à gauche de cette valeur. Nous choisissons la partie où se trouve notre valeur. Puis nous répétons l’opérations tant que nous n’avons pas obtenu la valeur.