

**Objectif de la séance :**

**Vacances :**

- Finaliser l'affichage des leds avec les boutons + test soudage
- Avancer sur le système de trappe (je n'ai pas le moteur donc je m'en chargerai plus tard, à la place je relie les boutons avec la solution).

**Séance (à la maison car forum pour présenter l'école le lundi 6 janvier)**

- Généraliser le programme dans l'ensemble pour n'avoir qu'à faire les branchements
- Demander au joueur de générer une solution (à deux joueurs)
- Créer une solution aléatoire (un joueur)
- Créer une fonction redémarrage

**Durant les vacances :**

J'ai pu commencer **la soudure des LEDS**, afin de tester si tout était opérationnel. Manon m'a donné une led pour que j'essaye chez moi.

Je dispose d'un appareil de soudure, voir photo ci-dessous, fonctionnant à gaz. J'ai eu un peu de mal au début mais j'ai réussi avec de la patience. Après avoir testé, les leds s'affichent correctement. Il faut faire attention car la première résistance que j'avais mise était trop élevé, je l'ai remplacé par une résistance de 330Ohm et tout marche correctement. Ensuite, j'ai rencontré mon plus gros problème jusqu'à présent je dirai; voici mon code :

Main :

```

main | Arduino 1.8.9 (Windows Store 1.8.21.0)
Fichier Edition Croquis Outils Aide

main functions.h

const int b_vert=2;
const int b_jaune=4;
int nbButton=2;
int buttonList[2]= {2,4};
int i=0;
int j=0;
const int LED_PIN=7;
const int NUM_LEDS=2;
int reponse[12][4];
int led[12][4]={{0,1,2,3},{4,5,6,7},{8,9,10,11},{12,13,14,15},{16,17,18,19},{20,21,22,23},{24,25,26,27},{28,29,30,31},{32,33,34,35},{36,37,38,39},{40,41,42,43},{44,45,46,47}};

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS,LED_PIN,NEO_GRB + NEO_KHZ800);

void setup() {
  Serial.begin(9600);
  i=0;
  j=0;
  pinMode(b_vert,INPUT);
  pinMode(b_jaune,INPUT);
  strip.begin();
  strip.show();
  strip.setPixelColor(led[i][j],255,0,0);
  strip.show();
  delay(1000);
  strip.clear();
  strip.show();
}

void loop() {
  buttonClicked(strip, nbButton, i, j, buttonList,reponse, led, b_jaune, b_vert);
}

Enregistrement terminé.
Le croquis utilise 4334 octets (14%) de l'espace de stockage de programmes. Le maximum est de 30720 octets.
Les variables globales utilisent 438 octets (21%) de mémoire dynamique, ce qui laisse 1610 octets pour les variables locales. Le maximum est de 2048 octets.

22:23 Arduino Nano sur COM5

```

### function.h :

```

main | functions.h | Arduino 1.8.9 (Windows Store 1.8.21.0)
Fichier Edition Croquis Outils Aide

main functions.h

#include <Adafruit_NeoPixel.h>

void allumerLeds(Adafruit_NeoPixel strip, int a, int i, int j, int led[12][4],int buttonList[2], int b_jaune, int b_vert){

  if (buttonList[a]==b_vert){ //si on appuie sur le bouton vert
    Serial.println("vert");
    strip.setPixelColor(led[i][j],0,255,0); //la led jème led de la ième ligne s'allume en vert
    strip.show();
  }
  else if (buttonList[a]==b_jaune){ //idem
    Serial.println("jaune");
    strip.setPixelColor(led[i][j],125,125,0);
    strip.show();
  }
}

void buttonClicked(Adafruit_NeoPixel strip, int nbButton, int i, int j, int buttonList[2], int reponse[12][4], int led[12][4],int b_jaune, int b_vert){
  Serial.println("début");

  for (int a=0;a<nbButton;a++){ //on parcourt la liste des boutons
    if (digitalRead(buttonList[a])==LOW){ //dès qu'une valeur est à high
      Serial.println(buttonList[a]); //on l'affiche pour vérifier que tout marche
      allumerLeds(strip, a, i, j, led, buttonList, b_jaune, b_vert); //on allume la led

      while (digitalRead(buttonList[a])==LOW){ //tant qu'on appuie sur le bouton rien ne se passe et le programme est en "pause"

        reponse[i][j]=a; //dès qu'on relâche le bouton, le programme reprend et on ajoute la réponse à la liste réponse.
        //Serial.println(reponse[i][j]);
      }
    }
  }
}

Enregistrement terminé.
Le croquis utilise 4334 octets (14%) de l'espace de stockage de programmes. Le maximum est de 30720 octets.
Les variables globales utilisent 438 octets (21%) de mémoire dynamique, ce qui laisse 1610 octets pour les variables locales. Le maximum est de 2048 octets.

11 Arduino Nano sur COM5

```

Tout me semblait correct, sauf lorsque je rentrais dans la fonction allumerdLed(), comme celle-ci est situé dans buttonClicked(), j'ai l'impression que cela empêchait l'allumage des LEDS, et je n'ai à ce jour pas résolu ce problème... C'est comme si la variable strip définie dans le main n'était pas reconnue ici, pourtant en ayant fait des tests.

Je suis sûr de bien rentrer dans les conditions puisque j'ai mis des serial.print dans celles-ci et tout s'affiche au moment voulu, en fonction du bouton sur lequel j'appuie etc. J'ai essayé

d'allumer une led tout simplement dans la fonction `buttonClicked()` et ça ne marchait pas non plus...

Donc j'ai décidé de refaire un programme classique en mettant tout dans la `loop` et en enlevant les fonctions, et voici le résultat :



```
testmainallumage
...
strip.setPixelColor(led[i][j],255,0,0);
strip.show();
delay(1000);
strip.clear();
strip.show();
}

void loop() {
  Serial.println("début");

  for (int a=0;a<nbButton;a++){ //on parcourt la liste des boutons
    if (digitalRead(buttonList[a])==LOW){ //dès qu'une valeur est à high
      Serial.println(buttonList[a]); //on l'affiche pour vérifier que tout marche
      if (buttonList[a]==b_vert){ //si on appuye sur le bouton vert
        Serial.println("vert");
        strip.setPixelColor(led[i][j],0,255,0); //la led jème led de la ième ligne s'allume en vert
        strip.show();
      }
      else if (buttonList[a]==b_jaune){ //idem
        Serial.println("jaune");
        strip.setPixelColor(led[i][j],125,125,0);
        strip.show();
      }
      while (digitalRead(buttonList[a])==LOW){ //tant qu'on appuye sur le bouton rien ne se passe et le programme est en "pause"
        reponse[i][j]=a; //dès qu'on relache le bouton, le programme reprend et on ajoute la réponse à la liste réponse.
        j+=1;
      }
    }
  }
}
```

Téléversement terminé

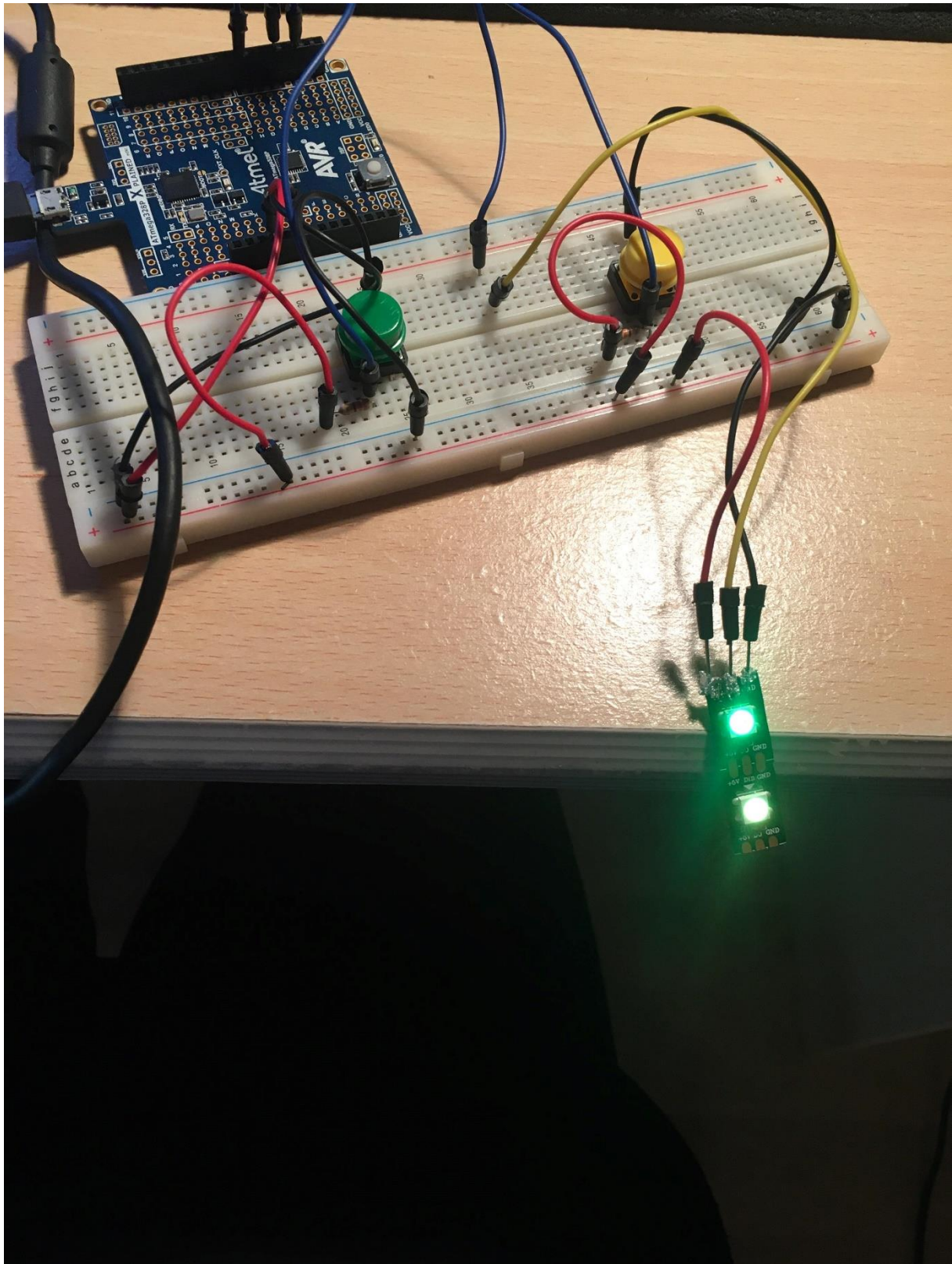
Le croquis utilise 4334 octets (14%) de l'espace de stockage de programmes. Le maximum est de 30720 octets.

Les variables globales utilisent 342 octets (16%) de mémoire dynamique, ce qui laisse 1706 octets pour les variables locales. Le maximum est de 2048 octets.

48 Arduino Nano sur COM5

**Tout marche à merveille !**

Le problème devait donc être celui que je supposais, j'aurai besoin de votre aide pour éclaircir ce problème, ayant cherché de nombreuses solutions par moi-même en vain. Néanmoins je peux toujours continuer ce que j'ai commencé en mettant tout dans le `main`. C'est dommage car je ne décortique pas tout en plusieurs fonctions...



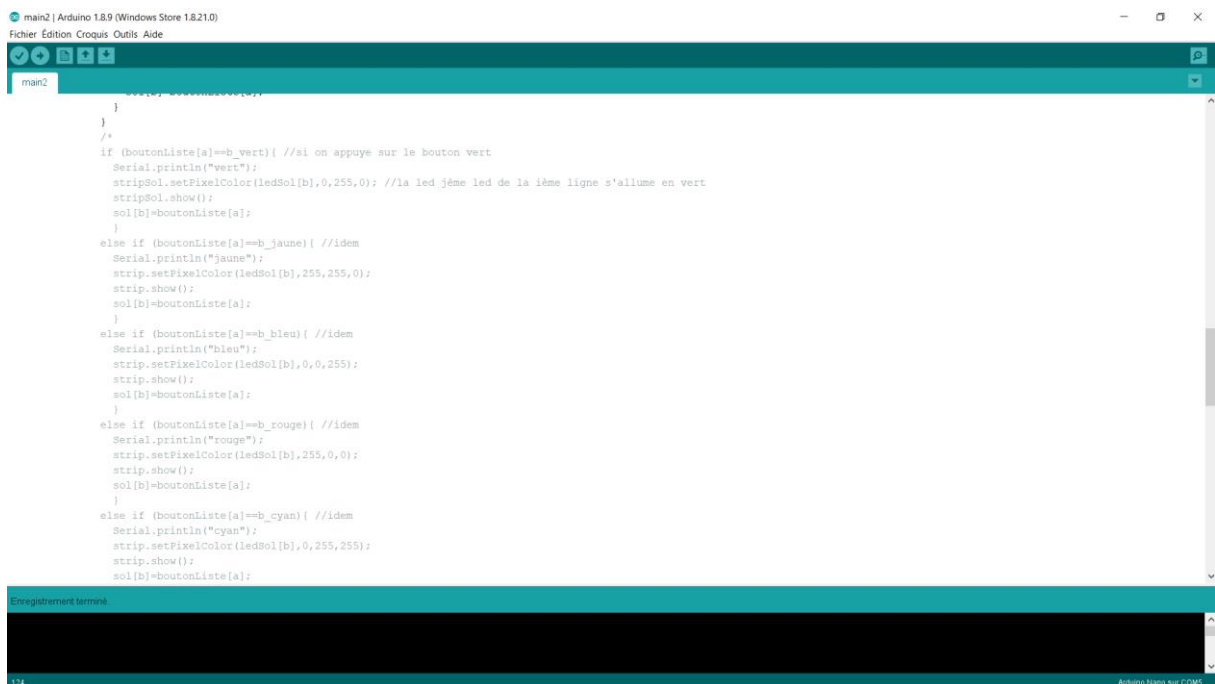
Au début du jeu les leds s'allument pour vérifier qu'elles marchent, puis elles s'éteignent et elles attendent une réponse de notre part ! **L'allumage est progressif avec les boutons.**

## Séance du 6 janvier (toujours à la maison) :

Même si je n'ai pas le bon nombre de LEDS et de fil, ayant déjà fait l'algorithme, je décide d'avancer pour pouvoir ajouter la solution, recommencer le jeu, et ajouter des conditions dans le programme général, pour pouvoir à la prochaine séance n'avoir qu'à faire les branchements.

Je continue sur le dernier programme présenté, pour pouvoir allumer les LEDS.

J'ai généralisé le programme pour notre jeu final, j'avais au début écrit des conditions comme ceci pour allumer les leds dans la trappe (même fonctionnement pour allumer les leds au cours du jeu) :




```
main2 | Arduino 1.8.9 (Windows Store 1.8.21.0)
Fichier Edition Croquis Outils Aide

main2
// ...
}
/*
if (boutonListe[a]==b_vert){ //si on appuie sur le bouton vert
  Serial.println("vert");
  stripSol.setPixelColor(ledSol[b],0,255,0); //la led jème led de la ième ligne s'allume en vert
  stripSol.show();
  sol[b]=boutonListe[a];
}
else if (boutonListe[a]==b_jaune){ //idem
  Serial.println("jaune");
  strip.setPixelColor(ledSol[b],255,255,0);
  strip.show();
  sol[b]=boutonListe[a];
}
else if (boutonListe[a]==b_bleu){ //idem
  Serial.println("bleu");
  strip.setPixelColor(ledSol[b],0,0,255);
  strip.show();
  sol[b]=boutonListe[a];
}
else if (boutonListe[a]==b_rouge){ //idem
  Serial.println("rouge");
  strip.setPixelColor(ledSol[b],255,0,0);
  strip.show();
  sol[b]=boutonListe[a];
}
else if (boutonListe[a]==b_cyan){ //idem
  Serial.println("cyan");
  strip.setPixelColor(ledSol[b],0,255,255);
  strip.show();
  sol[b]=boutonListe[a];
}
}
// ...
Enregistrement terminé
124 Arduino IDE sur COM5
```

C'est trop long et répétitif à mon gout, je dois changer ça et je pense avoir trouvé une manière de le faire

**J'ai réussi à simplifier clairement le code en faisant comme ceci :**

définition d'une liste permettant de relier les boutons et les couleurs :


```

main2 | Arduino 1.8.9 (Windows Store 1.8.21.0)
Fichier Edition Croquis Outils Aide

main2

#include <Adafruit_NeoPixel.h>

int nbJoueur=1;
const int restart=11;
const int b_rouge=2; //pin boutons
const int b_jaune=3;
const int b_vert=4;
const int b_cyan=5;
const int b_bleu=6;
const int b_violet=7;
int nbBouton=6; //nb de boutons
int boutonListe[6]= {2,3,4,5,6,7}; //liste des boutons et de leur pin
int b_color[6][3]={({255,0,0},{255,255,0},{0,255,0},{0,255,255},{0,0,255},{155,0,255})}; //permet de connaître la couleur en fonction du bouton (classé dans le même ordre que les boutons)
int l=0; //ligne
int j=0; //colonne
const int LED_PIN=8; //pin leds
const int LED_SOL_PIN=9; //pin leds solution
const int LED_ANALY_PIN=10; //pin leds analyseuses
const int NUM_LEDS=2; //nb de leds
const int NUM_LEDS_SOL=4; //nb leds solution
const int NUM_LEDS_ANALY=2; //nb leds analyseuses
int reponse[12][4]; //reponse du joueur
int led[12][4]={({0,1,2,3},{4,5,6,7},{8,9,10,11},{12,13,14,15},{16,17,18,19},{20,21,22,23},{24,25,26,27},{28,29,30,31},{32,33,34,35},{36,37,38,39},{40,41,42,43},{44,45,46,47})};
int ledSol[4]={0,1,2,3}; //leds solution (dans la trappe)
int sol[4]={9,9,9,9}; //etat initial, 9 ne correspond à aucune des valeurs possibles

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS,LED_PIN,NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel stripSol = Adafruit_NeoPixel(NUM_LEDS_SOL,LED_SOL_PIN,NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel stripAnaly = Adafruit_NeoPixel(NUM_LEDS_ANALY,LED_ANALY_PIN,NEO_GRB + NEO_KHZ800);

Enregistrement terminé

Le croquis utilise 5102 octets (16%) de l'espace de stockage de programmes. Le maximum est de 30720 octets.
Les variables globales utilisent 424 octets (20%) de mémoire dynamique, ce qui laisse 1624 octets pour les variables locales. Le maximum est de 2048 octets.

1192 Arduino Nano sur COM5

```

## On a dans la loop :



```

main2 | Arduino 1.8.9 (Windows Store 1.8.21.0)
Fichier Edition Croquis Outils Aide

main2

void loop() {

  if (digitalRead(restart)==LOW) {
    //restart(); // je ne sais pas encore si j'utilise une fonction ou pas, il faudra essayer et voir si cela marche
    while (digitalRead(restart)==LOW){ // tant qu'on appuie dessus
    }

    if (sol[0]==9){ //initialement, forcé à 9
      if (nbJoueur==1){ //si joueur en solo
        for (int b=0;b<nbBouton;b++){
          int r=random(0,5); //génération d'un chiffre aléatoire
          sol[b]=boutonListe[r]; //ajout du bouton, correspondant à l'emplacement du chiffre, à la liste solution
        }
      }
      else if (nbJoueur==2){
        int b=0; //variable permettant de compter les ledsSol
        while (sol[b]==9){
          for (int a=0;a<nbBouton;a++){ //on parcourt la liste des boutons
            if (digitalRead(boutonListe[a])==LOW){ //dès qu'une valeur est à high
              Serial.println(boutonListe[a]); //on l'affiche pour vérifier que tout marche
              for (int c=0;c<nbBouton;c++){ //boucle permettant de parcourir les couleurs des boutons
                if (boutonListe[a]==boutonListe[c]){ //on repère quel bouton est cliqué
                  Serial.println(boutonListe[c]); //affichage pour vérifier
                  stripSol.setPixelColor(ledSol[b],b_color[c][0],b_color[c][1],b_color[c][2]); //puis on allume la led numero b, de la couleur trouvée en c (numéro du bouton dans la liste)
                  stripSol.show();
                  sol[b]=boutonListe[a]; //on ajoute le bouton cliqué à la liste solution
                }
              }
              b++;
            }
          }
        }
      }
    }
  }
}

```

Enregistrement terminé

Le croquis utilise 5140 octets (16%) de l'espace de stockage de programmes. Le maximum est de 30720 octets.  
Les variables globales utilisent 424 octets (20%) de mémoire dynamique, ce qui laisse 1624 octets pour les variables locales. Le maximum est de 2048 octets.

1 Arduino Nano sur COM5

Il y a une liste sol, qui par défaut est remplie de 9 (aucun bouton ne donne 9), donc c'est comme si la liste était vide, cette condition me simplifiera par la suite le redémarrage du jeu.

**Voici mon récapitulatif des questions qu'il me reste à éclaircir :**

- Voir si la fonction restart est fonctionnelle, car je ne sais pas si en modifiant les variables dedans la fonction la variable seront modifiées dans tout le programme,
- Voir comment marche les boutons du choix du nb de joueurs et on/off pour allumer la carte arduino (matériel que vous m'avez donné, j'ai fait des essais mais je n'ai pas réussi)
- Voir comment peut fonctionner la trappe avec le moteur. (Il faut que je vous demande des conseils, car nous avons une idée de réalisation mais elle n'est pas définitive.)