Roue de la fortune

Groupe 27

Corbel Théo
Brault Killian
Cartaya Agustin
Boupacha Sid Ali

Fonctionnement et composition de la roue

Principe du jeux

On appuie sur le bouton de mise en route. La roue est constituée de 8 portions. Seule deux sont gagnantes. Si après l'arrêt de la roue, la portion désignée est gagnante, l'éclairage des leds aura lieu avec un buzzer et un grand GAGNÉ !!!. Dans le cas contraire, il y aura l'affiche de PERDU sur écran et PC.

Obtention du résultat du jeu

Pour obtenir les résultats du jeu, une batterie 3.3v et une diode led bleue ont été placées à l'extrémité de la tête rotative (source de luminosité) voir figure 1. Une photo-résistance a été placée dans chaque partie gagnante du cercle (récepteurs de luminosité) voir figure 1, si après la rotation de la tête rotative, la diode led pointe vers l'espace inclus par l'une des photorésistances le jeu sera gagné, sinon ce sera perdu.

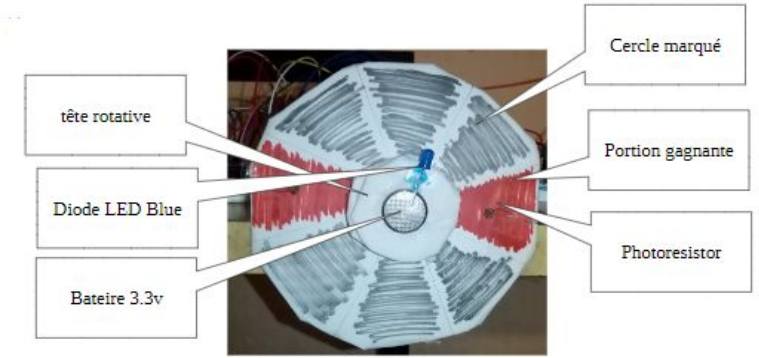


Figure 1

Composants

- 1 Moteur DC
- 1 cercle marqué et tête rotative connecté à l'axe du moteur (à faire à la main)
- 1 diode LED et une batterie de 3.3v positionnée au bout de la flèche
- 1 écran LCD
- 3 diodes LED
- 1 buzzer
- 1 potentiomètre
- 1 bouton (pour mettre en farce les tour du moteur)
- 2 photorésistances
- 1 source d'alimentation (pour l'alimentation du moteur)
- 3 Résistances de 220 ohms
- 2 Résistances 1k
- 1 Résistance 2k

Montage

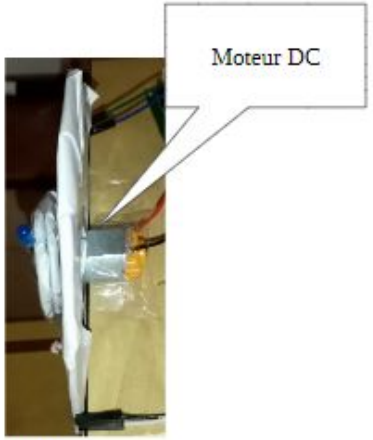


Figure 2

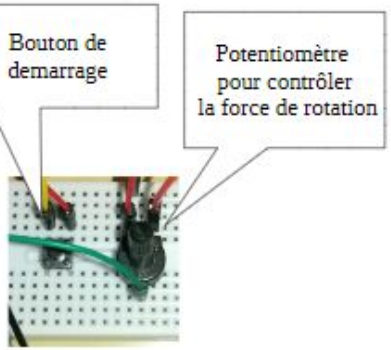


Figure 3

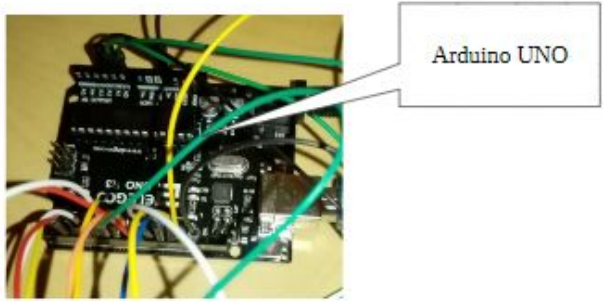


Figure 4

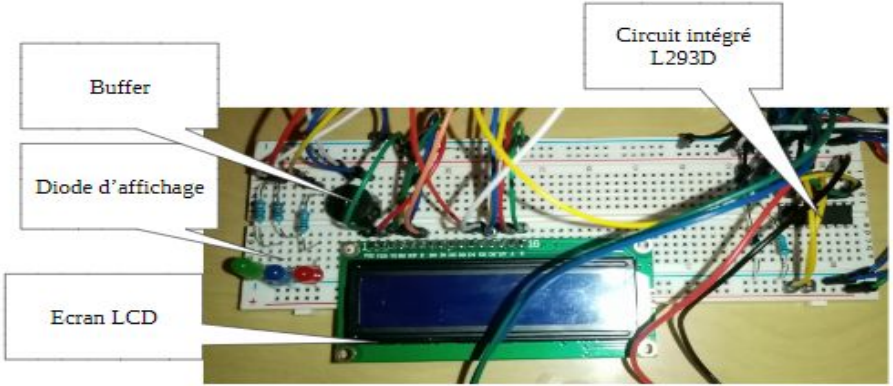
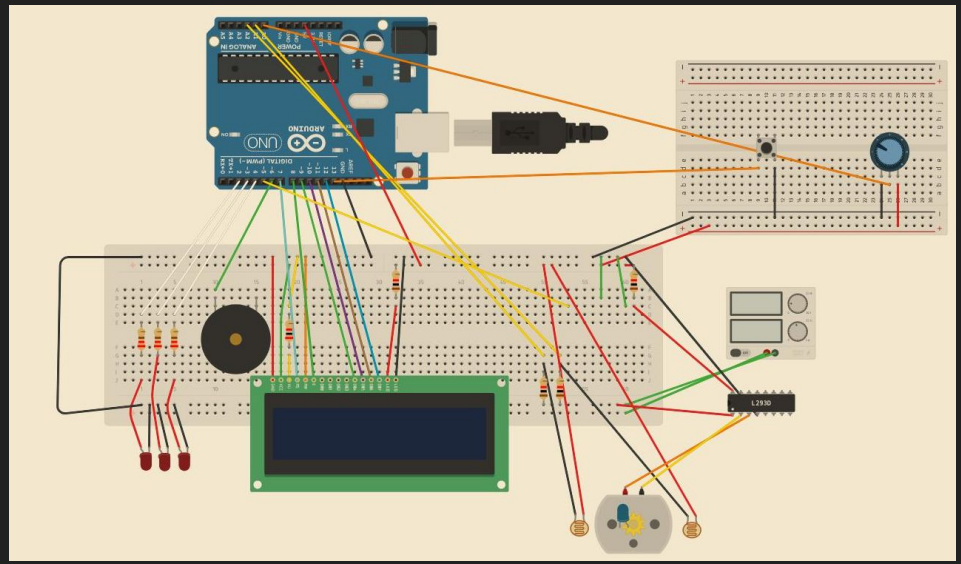


Figure 5



Librairies et Variables globales

Librairies utilisées

- LiquidCrystal.h (utilisé pour manipuler l'écran LCD)
- pitches.h (contient toutes les constantes avec les valeurs des notes utilisées pour faire sonner le buzzer)

Variables globales

Contrôle du buzzer

- int melody[]: tableau contenant les valeurs numériques des notes utilisées
- int toneDuration : durée en millisecondes de chaque note

Contrôle de l'écran LCD

- LiquidCrystal lcd : Objet de type LiquidCrystal qui facilite la manipulation de l'écran LCD

Contrôle des photorésistances

- int vph1 : contient la valeur de la lumière capturée par la photorésistance gauche
- int vph2 : contient la valeur de la lumière capturée par la photorésistance droite
- int ambientLight : contient la valeur de la lumière ambiante
- int maxLight: contient la valeur de la lumière maximale capturée par l'une des photorésistances
- int rang : rang des photorésistances

Contrôle du moteur

- int f :contient la force de rotation de la valeur du moteur (valeur entre 180 et 255)
- int f_ant : support de f pour indiquer un éventuel changement
- int tmpR : temps de rotation du moteur en secondes

Données de jeu

- int jeuxGagne :nombre de parties gagnées
- int jeuxJoues : nombre de parties jouées

Contrôle du temps

- unsigned long tm : temps en millisecondes pour décompter pendant la rotation du moteur
- unsigned long tm1 : temps en millisecondes pour actualiser les données de l'écran LCD
- boolean initialize : vrai si le jeu est en cours d'exécution (après avoir appuyé sur le bouton de démarrage

Explication du code

void setup() :

- initialiser tout les PINS numériques de l'arduino.
- démarre l'écran LCD et le port série.
- met à la terre les LED d'information, le moteur et le buzzer.
- initialiser la variable tm1.

void loop() :

- cette méthode est dans l'attentede pression du bouton de démarrage pour démarrer le jeu.
- Actualise l'écran chaque fois que la force de rotation du moteur est modifiée.

void start() :

cette méthode est responsable du démarrage du jeu et du calcul de la lumière ambiante moyenne

- réinitialise les valeurs obtenues par les photorésistances.
- efface l'écran LCD et affiche un message de début de partie.
- allume les LEDS d'information indiquant que le jeu est en cours.
- calcule la lumière ambiante moyenne en faisant 3 échantillons différents, ce processus prend 3 secondes (le temps est affiché sur l'écran LCD).
- après 3 secondes, il affiche un message sur l'écran LCD et fait sonner une fois le buzzer.
- fait tourner le moteur.
- vérifier si l'état de la partie. :
- indique sur le port série, la valeur de la lumière environnementale et la valeur de la lumière maximale.
- éteint les LEDS.

void rotate()

Méthode en charge de faire tourner le moteur avec la force initiale choisie et de l'arrêter progressivement dans un délai de 10 secondes, lors de la rotation du moteur on calcule le maximum de luminosité puisque la diode située sur la tête de rotation passe plusieurs fois devant les photorésistances permettant alors obtenir cette valeur. Un compte à rebours s'affiche en parallèle pour indiquer au joueur le temps restant.

- initialise et calcule les variables locales int prc, int prcAnt, qui contiennent le temps de rotation du moteur, et qui permet de l'arrêter progressivement. ce temps (en secondes) est également affichée sur l'écran LCD.
- initialise et calcule int forceAct qui contiendra la force de rotation du moteur au cours du temps de rotation, au début elle est égale à la force choisie par le joueur (f) et diminue linéairement jusqu'à atteindre 0.
- le luminosité maximal est calculé à partir des valeurs maximales obtenues par les photorésistances.
- arrête le moteur et attend 1,5 secondes.

void verify()

Vérifier le résultat du jeu. Les résultats sont calculés à partir de la moyenne des valeurs obtenues par les photorésistances après l'arrêt du moteur.

- Calculer la moyenne de chaque photorésistance en faisant 10 échantillons
- affiche 3 résultats possibles:
  - ERREUR: si la lumière ambiante est très proche de la lumière maximale, cela indique que la LED de la tête rotative ne s'allume pas correctement ou qu'il y a beaucoup de lumière dans l'environnement.
  - GAGNE: si la valeur de l'une des photorésistances est suffisamment proche de la luminosité maximale.
  - PERDU: si aucune valeur des photorésistances ne s'approche de la valeur de la luminosité maximale.
- exécuter l'un des 3 protocoles en fonction du résultat obtenu en appelant une des trois méthodes MauvaisReglageLumiereMax (), gagner (), perdre ()
- Après avoir implémenté l'un des 3 protocoles, il affiche à nouveau les détails du jeu sur l'écran LCD

void calcPH()
lire les bornes analogiques A1 et A2 contenant les valeurs obtenues par les photorésistances

int calcAL()
renvoie la luminosité ambiante en prenant le minimum des valeurs obtenues par les deux photorésistances

int calcML()
renvoie la luminosité maximale, en prenant le maximum des valeurs obtenues par les deux photorésistances

void rstLights()
réinitialise les valeurs de luminosité ambiante et de luminosité maximale

void printData(boolean ob)
affiche sur l'écran LCD la valeur de la force de rotation, le nombre de parties jouées et le nombre de parties gagnées

void controlLights(int phr1, int phr2, int al,int ml)
montre par le port série, les valeurs des photorésistances, la valeur de la luminosité ambiante et la luminosité maximale

void leds(int m)
méthode responsable de l'allumage et de l'extinction des LED d'information, avec un paramètre qui en fonction de sa valeur les LED s'éteindront, s'allumeront ou clignoteront

- 0: toutes les LED éteintes
- 1: toutes les LED allumées
- autre: les LED ont un comportement aléatoire

void gagner()
protocole qui est exécuté quand une partie est gagnée

- une partie gagnée est ajoutée au compteur
- un message s'affiche «You Win !!! »À l'écran LCD et sur le port série
- le buzzer est activé en jouant une succession de notes 2 fois
- allume et éteint les LEDs d'information au hasard

void perdre()
protocole qui s'exécute lorsqu'un jeu est perdu

- toutes les LED s'éteignent
- un message s'affiche «Vous avez perdu !!! »À l'écran et sur le port série
- le buzzer est activé en jouant une succession de notes

void MauvaisReglageLumiereMax()
protocole qui est exécuté lorsque la luminosité maximale ne peut pas être calculé correctement

- toutes les LED s'éteignent
- un message "Game failed increase pw led" s'affiche à l'écran et sur le port série
- le tampon sonne à plusieurs reprises indiquant qu'il y a eu un problème

void lcdClearUpDown(boolean down, int pos)
méthode utilisée pour effacer les informations de l'écran LCD

Code du projet

#define

```
//LEDS
#define LED1 2
#define LED2 3
#define LED3 4


//Motor
#define MOTOR 5

//Bufer
#include "pitches.h"
#define BUFER 6
int melody[] = {31, 33, 35, 37, 39, 41, 44, 46};
int toneDuration = 300;  //duration de chaque tone

//Screen
#include <LiquidCrystal.h>
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

//Button
#define BT 13

//Potentiometer
#define FORCE A0


//Photo resistors
#define PH1 A1 //gauche
#define PH2 A2 //droit
```

Initialisation des valeurs

```
//valeurs des photoresistors
int vph1 = 0; //gauche
int vph2 = 0; //droit

int ambientLight = 0;
int maxLight = 0;

//rang
int rang = 20;

//donnes du jeu
int f = 200;
int f_ant = 0;
int jeuxGagne = 0;
int jeuxJoues = 0;

//contreul du temps
unsigned long tm = 0;
unsigned long tml = 0;

//temps de rotation en secondes
int tmpR = 10;

//vrai si le jeu est en cours d'exécution
boolean initialize = false;
```

```
void setup() {

  //Init Components
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);

  pinMode(MOTOR,OUTPUT);

  pinMode(BUFER,OUTPUT);

  pinMode(BT,INPUT);
  digitalWrite(BT, HIGH);

  lcd.begin(16, 2);
  Serial.begin(9600);


  //reset components
  analogWrite(MOTOR, 0);
  digitalWrite(BUFER, LOW);
  leds(0);
  tml = 0;

}

void loop() {

  //à attente du demarrage du jeu
  if(!initialize &&  digitalRead(BT) ){
    initialize = true;
    start();
    initialize = false;
  }


  //mise à jour de l'écran
  if(millis()  - tml > 200){
    tml = millis();
    f = map(analogRead(FORCE), 0, 1023, 180, 255);
    printData(false);

  }
}
```

Démarrage du jeu ->

```
void start(){
 Serial.println("---------------- Start game-----------------");

 //réinitialiser les valuers des photorésistances
 rstLights();

 //nombre d'échantillons pour obtenir la lumière d'ambiante
 int ech = 3;

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("    Start in   ");

 leds(1);
 //obtenir la moyenne de la lumière ambiante
 for(int i = ech; i > 0; i--){
   lcd.setCursor(7, 1);
   lcd.print(i);
   ambientLight += calcAL();
   delay(1000);
 }
 ambientLight /= ech;

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("    Good look");

 tone(BUFER, melody[5], toneDuration);

 //Control
 Serial.print("Ambient light: ");
 Serial.println(ambientLight);

 rotate();
 verify();

 jeuxJoues++;
 printData(true);

 //Control
 Serial.print("MAX Light: ");
 Serial.println(maxLight);
 Serial.println("-----------------------------\n");

 leds(0);

}
```

```
/**
* Rotation du moteur.
*
*/
void rotate(){

  //pourcentage de rotation
  int prc = 0;
  int prcAnt = 0;

  //réduire la force linéairement
  int forceAct = 0;

  tm = millis();

  //réduction de la puissance du moteur et affichage du temps restant
  while(millis() - tm < (tmpR * 1000) ){

     prc = 10 - ((int)(millis() - tm)/1000);
     if( prc != prcAnt){
       lcdClearUpDown( 1, 7);
       lcd.print(prc);
       prcAnt = prc;
     }

     //réduction de la puissance
     forceAct = (int)(prc * f / 10);
     analogWrite(MOTOR,forceAct );

     //obtenir la valeur maximum de lumière
     maxLight = max(maxLight, calcML());

  }

  //arret du moteur
  analogWrite(MOTOR,0 );
  delay(1500);



}
```

```
/*
* Verifier le resultat du jeu
*/
void verify(){
  int vph1_temp = 0;
  int vph2_temp = 0;

  //nombre d'échantillons pour obtenir la les resultats
  int ech = 10;

  //obtenir la moyenne des photorésistances
  for(int i = 0; i<ech; i++){
    calcPH();
    vph1_temp += vph1;
    vph2_temp += vph2;
  }
  vph1_temp /= ech;
  vph2_temp /= ech;


  //Control
  controlLights(vph1_temp, vph2_temp, ambientLight, maxLight);



  if( (ambientLight + rang) >= maxLight )
    MauvaisReglageLumiereMax();
  else if ( (vph1_temp + rang) >= maxLight || (vph2_temp + rang) >= maxLight)
    gagner();
  else
    perdre();

  printData( true );
}

/**
* calculer les valeurs des photorésistances
**/
void calcPH(){
  vph1 = analogRead(PH1);
  vph2 = analogRead(PH2);
}
```

```
/**
* calculer la lumière ambiante
*/
int calcAL(){
  calcPH();
  int al= min(vph1, vph2);
  controlLights( vph1, vph2, al, maxLight );
  return al;
}

/**
* calculer la lumière maximale
*/
int calcML(){
  calcPH();
  return max(vph1, vph2);
}


/**
* réinitialiser les valeurs des photorésistances
*/
void rstLights(){
  ambientLight = 0;
  maxLight = 0;
}

/**
* imprimer les données du jeu sur lcd
*/
void printData( boolean ob){

  if( f != f_ant || ob){
    lcd.clear();
    lcd.setCursor(0, 0);

    //Force de rotation
    lcd.print("F: ");
    lcd.print(f);


    //jeux jouées
    lcd.print(" PJ: ");
    lcd.print(jeuxJoues);

    lcd.setCursor(0, 1);
    //jeux gagnés
    lcd.print("PG: ");
    lcd.print(jeuxGagne);

    f_ant = f;
  }



}

/**
* contrôle de la lumière obtenue par les photorésistances
*/
void controlLights( int phr1, int phr2, int al,int ml  ){

  //lumière obtenue par le photorésist gauche
  Serial.print("PHR_1: ");
  Serial.print(phr1);

  //lumière obtenue par le photorésist droite
  Serial.print("\tPHR_2: ");
  Serial.print(phr2);

  //lumière ambiante
  Serial.print("\tAmbient_Light: ");
  Serial.print(al);

  //lumière maximale
  Serial.print("\t Max_Light: ");
  Serial.println(ml);
}


/**
* contrôle des leds d'information
*/
void leds(int m){
  //jugar con los leds
  if(m == 0){
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
  }
  else if(m == 1){
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
  }
  else{
  digitalWrite(LED1, random(0, 2));
  digitalWrite(LED2, random(0, 2));
  digitalWrite(LED3, random(0, 2));
  }
}


/**
* protocole exécuté quand une partie est gagnée
*/
void gagner(){

  jeuxGagne++;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("     You Win!!!    ");

  Serial.println("     You Win!!!    ");

  for(int i = 0; i<2; i++){

    for (int thisNote = 0; thisNote < 8; thisNote++) {

      tone(BUFER, melody[thisNote], toneDuration);
      leds(thisNote % 2);
      delay(toneDuration);
    }
    leds(0);
  }
}
```

```
/**
* protocole exécuté lorsque la lumière maximale
* n'a pas pu être calculée correctement
*/

void MauvaisReglageLumiereMax(){

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Game failed");
  lcd.setCursor(0, 1);
  lcd.print("increase pw led");

  Serial.println("Game failed");
  Serial.println("increase pw led");

  for(int i =0; i<20; i++){
     if( i%2 == 0 )
       tone(BUFER, melody[0], toneDuration);
     delay(toneDuration);
  }

}

/**
* effacer une partie de l'écran LCD
*/
void lcdClearUpDown( boolean down, int pos){
  lcd.setCursor(0, down);
  lcd.print("                ");
  lcd.setCursor(pos, down);
}
```

```
/**
* protocole exécuté quand une partie est perdue
*/
void perdre(){

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("     You lost...    ");

  Serial.println("     You lost...    ");

    for (int thisNote = 7; thisNote >= 0; thisNote--) {
      tone(BUFER, melody[thisNote], toneDuration);
      delay(toneDuration);
    }

}
```

Lien Tinkercad du montage :

https://www.tinkercad.com/things/6xBiHrt9zUo-spi3projet/editel?sharecode=tc5DMxP0FGMJ67nYOQzsar1TbdODjCWg45tAKnqOIg4

Ports de connexion Arduino

Digital Ports

Port	Capteur	Type
2	Diode LED1	OUTPUT
3	Diode LED2	OUTPUT
4	Diode LED3	OUTPUT
5	Motor DC	OUTPUT
6	buzzer	OUTPUT
7,8,9,10,11,12	Écran LCD	OUTPUT
13	Switch	INPUT

Analog ports

Port	Capteur
A0	Potentiomètre
A1	Photorésistance
A2	Photorésistance