Arduino UNO Trousse Educationelle

ABRA

www.abra-electronics.com

ARD-EDU-KIT

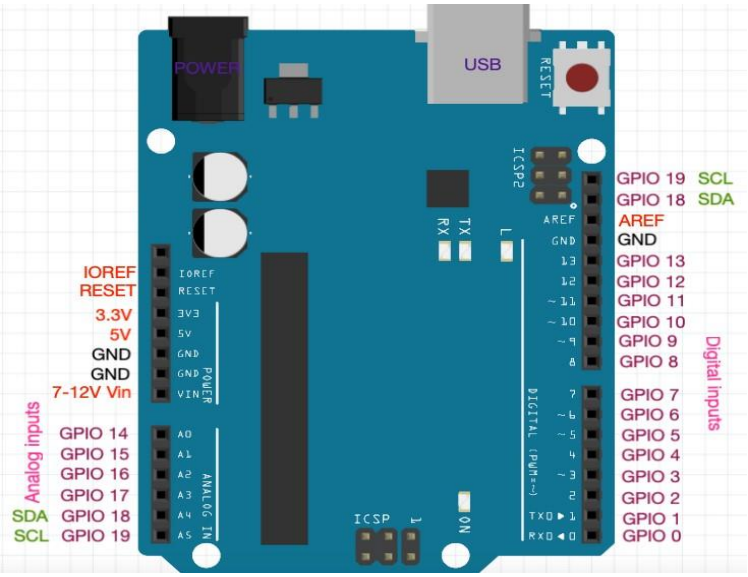
TEL: 1-800-361-5237

S.no	Description de la composante	Référence de pièce	Quantité
1	Carte PCB Arduino UNO	ABRAUNO	1
2	5V Relai	SRD-5VDC-SL-C	1
3	Télécommande avec receveur infrarouge	WL-110	1
4	Servo Moteur SG90	SG90	1
5	16x2 LCD Module avec I2C	LCD-MOD-13	1
6	Module Capteur d'Humidité	SENS-DHT11-BB	1
7	PS2 Module de manette	AM-JOYSTICK	1
8	Module d'alimentation	PSM-297	1
9	Moteur Pas à pas avec CI pilote	MOT-28BYJ48	1
10	Avertisseur sonore Passif	BUZ-125	1
11	Avertisseur sonore Actif	BUZ-120	1
12	Plaque Expérimentale-830 Points d'attache	ABRA-12-LC	1
13	Afficheur 7-Segment 1 Chiffre Rouge CC	S-5101AS	1
14	Module capteur d'inclinaison	SENS-39	1
15	Module capteur Ultrasonique	HC-SR04	1
16	4x3 Clavier	419-ADA	1
17	PIR Capteur	SENS-PIR	1
18	Capteur de détection d'eau	SENS-78	1
19	RTC Module Horloge temps réel	ARD-DS3231	1
20	Module Capteur de son	SENS-42	1
21	Bouton poussoir momentané	PBS-315BK	5
22	Photorésistance	PHOTO-300	2
23	NPN Transistor	S8050	5
24	NPN Transistor	PN2222A	5
25	1N4007 Diode	1N4007	1
26	Résistances (10,100,220,330,1K,2K,5K,10K,100K,1M) Ω		100
27	10K Thermistor	334-103	1
28	5mm DEL (Rouge, Vert, Jaune, Bleu, Blanc)		25
29	RGB DEL (Rouge, Vert, Bleu) CC	LED-5RGB-4-CC	1
30	CI, Pilote de moteur	L293D	1
31	CI, Registre à décalage 8 bits	74HC595	1
32	Condensateur électrolytique (10uf 50V)	10R50	2
33	Condensateur électrolytique r (100uf 50V)	100R50	2
34	Condensateur céramique disque (220) 22pf	CD220	5
35	Condensateur céramique disque (104) 0.1uf	CD104	5
36	Ventilateur de Moteur DC	MOT-PROP-L	1
37	Moteur DC	MOT-500	1
38	Potentiomètre 10K	P10K-MIN-PC	2
39	Encodeur rotatif	AM-127	1
40	Fils de pontage femelle à mâle	JW-MF-20-6	1
41	Fils de pontage mâle à mâle	JW-MM-20-6	1
42	Adaptateur mural 9VAC	PSC12R-090	1
43	Boitier plastique	CB-13	1
44	Carte de Résistance		1
45	Manuel		1

Contenu

1 Introduction à Arduino

L' Arduino Uno est un microcontrolleur basé sur le ATmega328P.

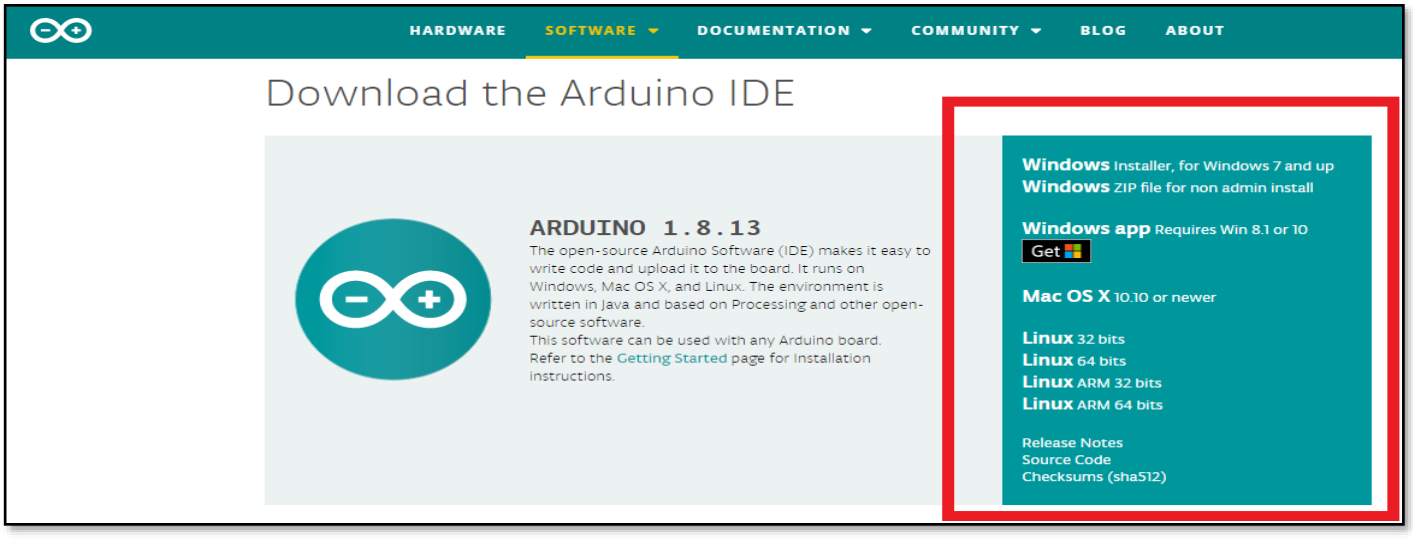


A0-A5: Les données analogiques telles que la photorésistance du capteur sont lues par ces broches et le convertisseur analogique-numérique (ADC)

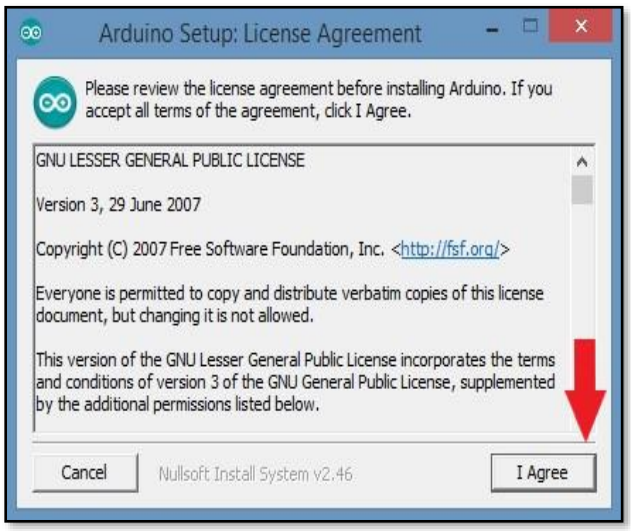
1.1 Installation du Arduino

Étape 1: Visitez https://www.arduino.cc/. In software section select Downloads.

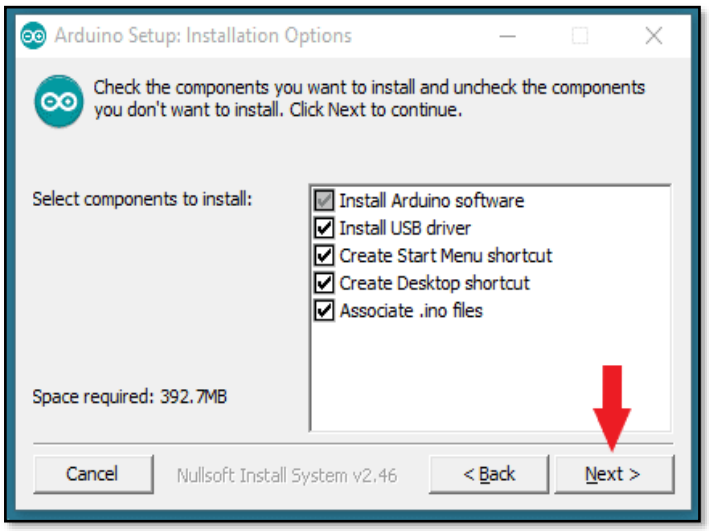
Étape 2: Selon le système d'exploitation de votre ordinateur, sélectionnez l'IDE Arduino en conséquence. Si Windows, cliquez sur le programme d'installation au lieu du fichier ZIP- étant plus facile pour installer.



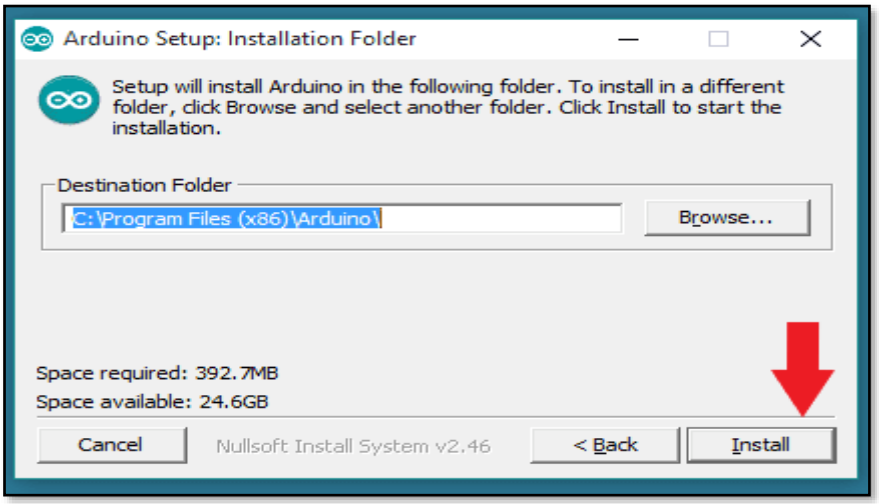
Étape 3: Cliquez acceptance **I agree**



Étape 4: Cliquez suivant **Next**

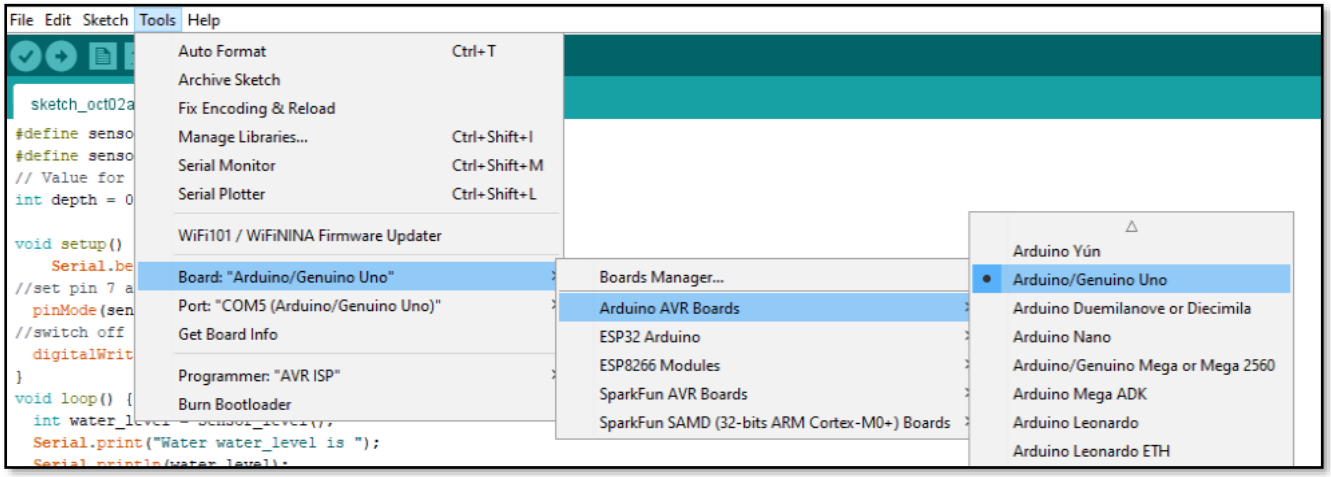


Étape 5: Sélectionnez le chemin où l'IDE Arduino doit être installé. Par défaut, il s'agit du lecteur C mais vous avez toujours la possibilité de naviguer et de sélectionner votre propre chemin.

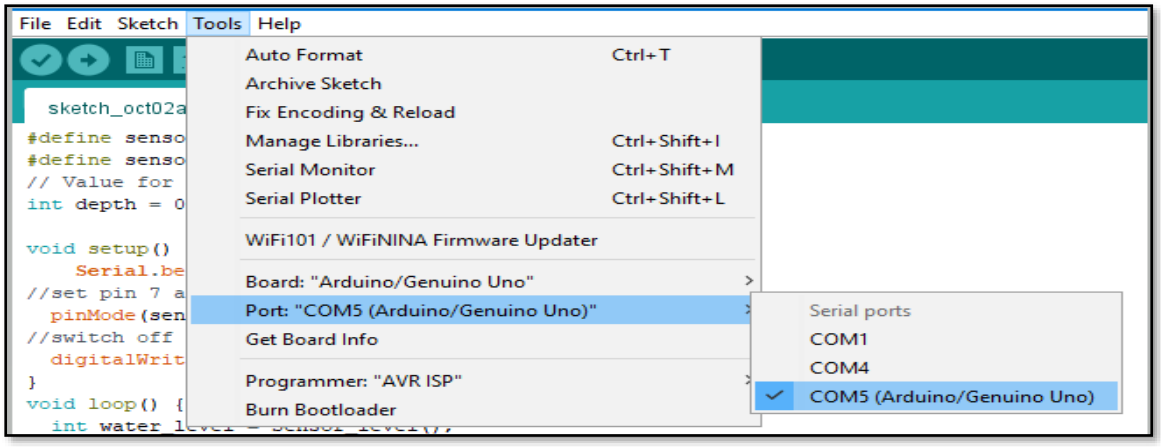


1.2 Démarrer avec IDE

Ouvrez l'IDE et connectez votre carte Arduino à votre ordinateur. Avant d'écrire votre code, assurez-vous que vous avez sélectionné la carte PCB. **Tools->Board->Arduino AVR boards-> Arduino/Genuino Uno**

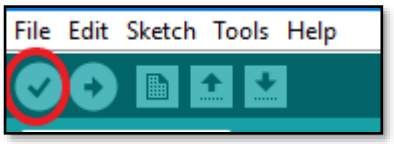


Sélectionnez maintenant le port pour la carte Arduino. Dans cet exemple, c'est **Com 5 (Arduino/Genuino Uno).**



Maintenant, écrivez votre code et compilez et téléchargez le code sur le tableau.

Pour compiler votre code, cliquez sur le bouton à cocher figurant dans la boite



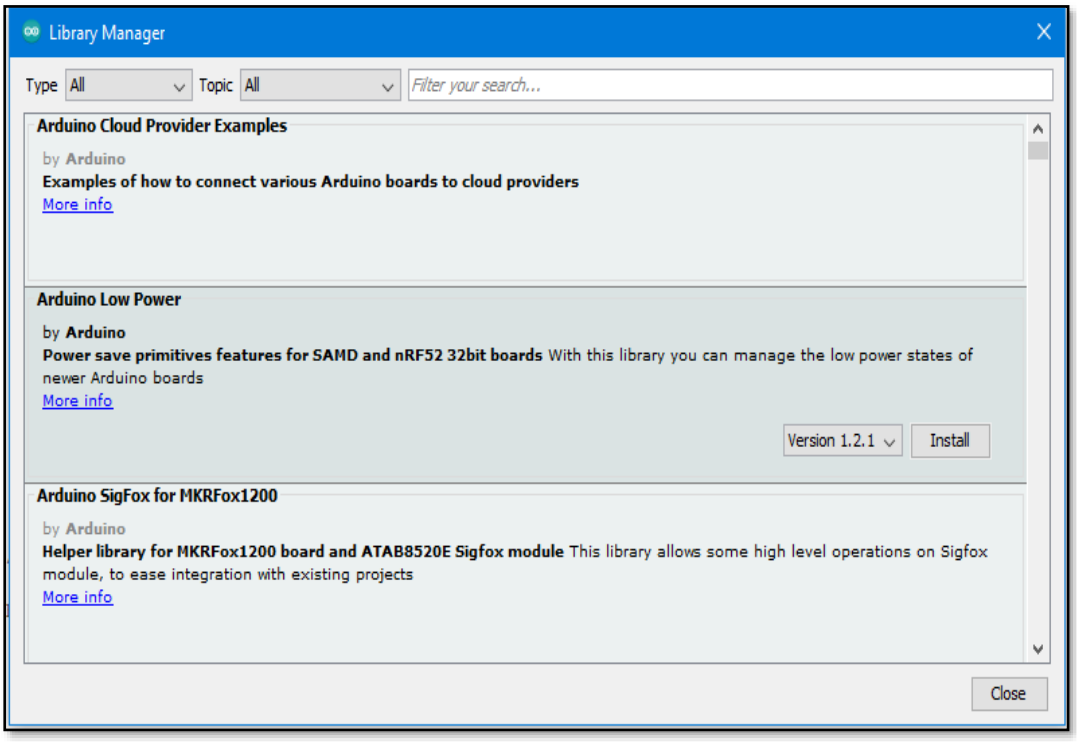
Pour télécharger le code, sélectionnez **Sketch-> upload**

1.3 Ajouter les bibliothèques.

L'environnement Arduino peut être étendu à l'aide de bibliothèques, comme la plupart des plateformes de programmation. Les bibliothèques fournissent des fonctionnalités supplémentaires pour l'utilisation de croquis, par exemple pour travailler avec du matériel ou manipuler des données. Pour utiliser une bibliothèque dans un croquis, sélectionnez-la parmi Sketch > Import Library.

Comment installer une bibliothèque

- *Utiliser le gestionnaire de la bibliothèque:* Pour installer une nouvelle bibliothèque dans votre EDI Arduino, vous pouvez utiliser le gestionnaire de bibliothèque. Ouvrez l'IDE et cliquez sur le menu "Sketch" et ensuite ***Include Library > Manage Libraries.*** Cherchez ensuite la bibliothèque que vous recherchez et cliquez sur "Installer".



- *Importation d'une bibliothèque .zip: Les bibliothèques sont souvent distribuées sous la forme d'un fichier ou d'un dossier ZIP. Le nom du dossier est le nom de la bibliothèque. Le dossier contient un fichier .cpp, un fichier .h et souvent un fichier keywords.txt, un dossier examples et d'autres fichiers requis par la*

bibliothèque. À partir de la version 1.0.5, vous pouvez installer des bibliothèques tierces dans l' IDE. Ne décompressez pas la bibliothèque téléchargée, laissez-la telle quelle. Dans l'Arduino IDE, naviguer à **Sketch > Include *Library > Add .ZIP Library.*** Vous serez invité à sélectionner la bibliothèque que vous souhaitez ajouter. Naviguez jusqu'à l'emplacement du fichier .zip et ouvrez-le.

1.4 Bases de Code

Certaines fonctions de base qui facilitent le codage dans Arduino sont énumérées ci-dessous.

E/S numérique (I/O) pour les broches numériques

- digitalRead() : Lit la valeur à partir d'une broche numérique spécifiée, soit HAUTE soit BASSE.
  Syntax: digitalRead(pin)
- digitalWrite() : Inscrivez une valeur HAUTE ou BASSE sur une broche numérique. Si la broche a été configurée comme une SORTIE avec pinMode(), son voltage sera réglé à la valeur correspondante : 5V (ou 3,3V sur les cartes 3,3V) pour le HAUT, 0V (masse) pour le BAS. Si la broche est configurée comme une ENTREE, digitalWrite() activera (HIGH) ou désactivera (LOW) le pullup(traction) interne sur la broche d'entrée.
  Syntax: digitalWrite (pin,value)

- pinMode() : Configure la broche spécifiée pour qu'elle se comporte soit comme entrée soit comme une sortie.
  Syntax: pinMode(pin, mode).
  broche: le numéro de broche Arduino pour définir le mode de. Mode est soit INPUT, soit OUTPUT (sortie).

```
int ledPin = 12;  // DEL connection sur broche digital pin 13
int inPin = 7;    // bouton poussoir connecte sur broche digital pin 7
int val = 0;      // variable pour entreposer la valeur de lecture(read value)

void setup() {
  pinMode(ledPin, OUTPUT);  // établit la broche digitale pin 12 comme sortie
  pinMode(inPin, INPUT);    // établit la broche digitale pin 7 comme entrée
pinMode(13, OUTPUT);    // établit la broche digitale pin 13 comme sortie
}
void loop() {
  val = digitalRead(inPin);   // lecture de la broche d'entrée
  digitalWrite(ledPin, val);  // établit le DEL à la valeur du bouton poussoir
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // établit la broche digitale pin 13 off (éteindre)
  delay(1000);            // attend pour une seconde
}
```

E/S analogique (I/O) pour les broches analogiques

- analogRead() : Lit la valeur à partir de la broche analogique spécifiée
  Syntax: analogRead(pin)
  Example: digitalread(10); // lit la valeur soit BASSE soit HAUTE pour la broche # 10
- analogWrite() : Ecrit une valeur analogique (onde PWM) sur une broche. Peut être utilisé pour allumer un DEL à différentes luminosités ou pour faire tourner un moteur à différentes vitesses.
- Syntax: analogWrite(pin, value)

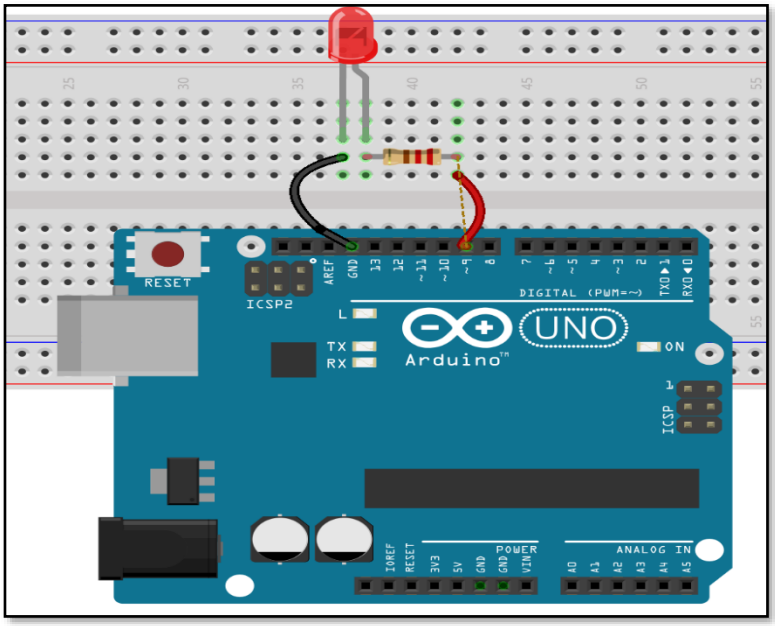
```
int ledPin = 9;       // DEL connecté sur la broche digitale #9
int analogPin = 3;    // potentiomètre connecté sur broche analogue #3
int val = 0;          // variable pour entreposer la valeur de lecture read value

void setup() {
  pinMode(ledPin, OUTPUT);  // établit la broche comme sortie
}

void loop() {
  val = analogRead(analogPin);  // lire la broche d'entrée
  analogWrite(ledPin, val / 4); // valeurs analogRead vont de 0 à 1023, valeurs
analogWrite de 0 à 255
}
```

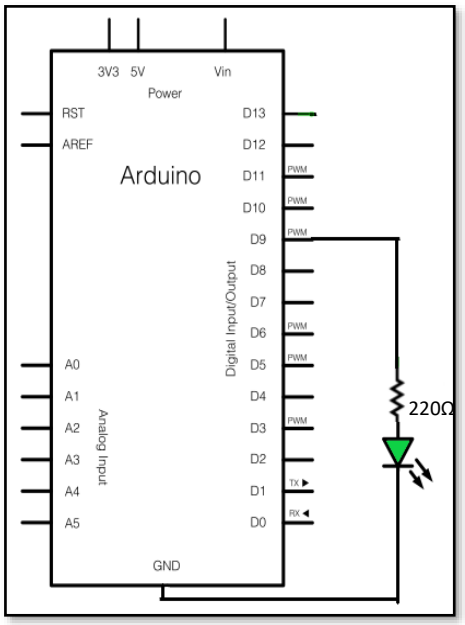
2 Projets

2.1 Clignotement de DEL

En bref: Grâce à ce projet, nous apprendrons comment allumer et éteindre un DEL. Si la broche a été configurée comme une SORTIE avec pinMode(), sa tension sera réglée à la valeur correspondante pour digitalWrite: 5V pourr HIGH, 0V (mise à la terre) pour LOW.

Composants: DEL, RESISTANCE 220Ω

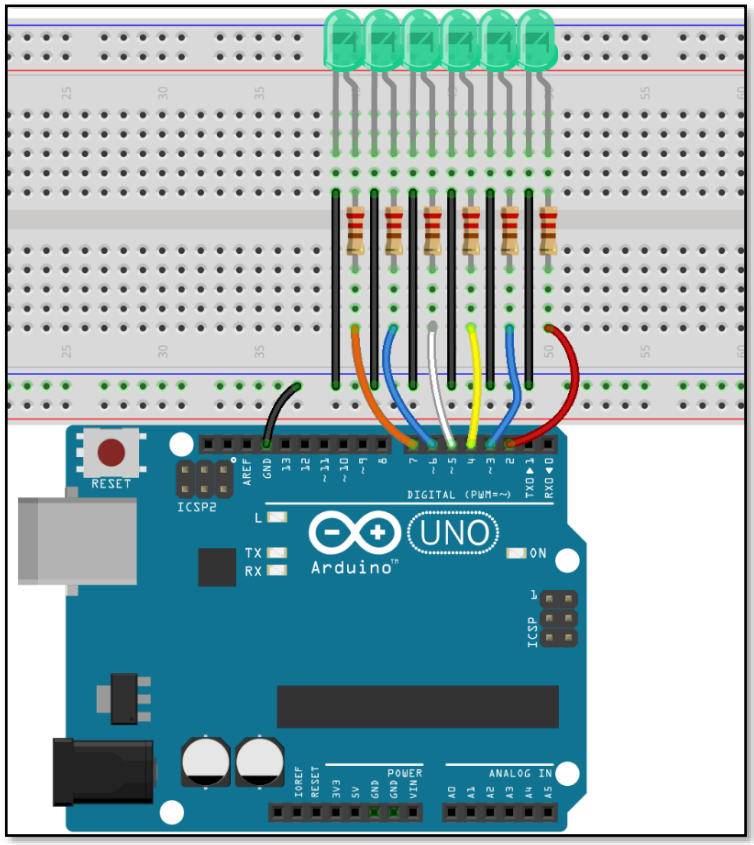


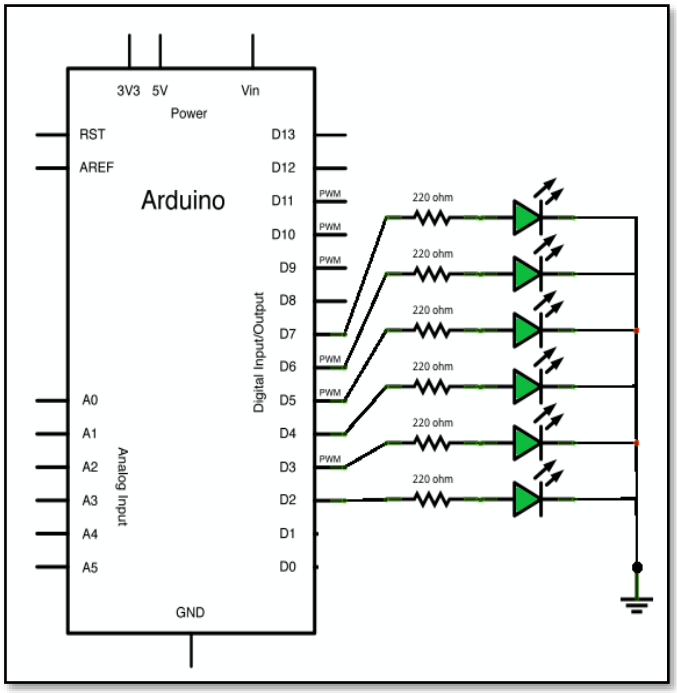


```
const int ledPin = 9;            // le # de broche pour le DEL sur la carte PCB
void setup ()
{
pinMode (ledPin, OUTPUT);  //initializer la broche digitale # 9 comme sortie
}
void loop ()       // la routine de la boucle est répétée à l'infini
{
digitalWrite(ledPin,HIGH);                  /allumez le DEL rouge /
delay(500);                 //attendez pour une demie seconde
digitalWrite(ledPin,LOW);                   //débranchez l'alimentation du DEL
delay(500);                  // attendez pour une demie seconde
}
```

2.2 Effet de suivi des DEL

En bref: Grâce à ce projet, nous apprendrons à allumer les DEL un par un à l'aide d'Arduino. Nous utilisons ici une boucle pour activer les DELs connectées à la broche 2, 3,4,5,6,7 l'une après l'autre avec un retard de 500 c'est-à-dire une demi-seconde.
Composants: DEL, 220Ω résistances

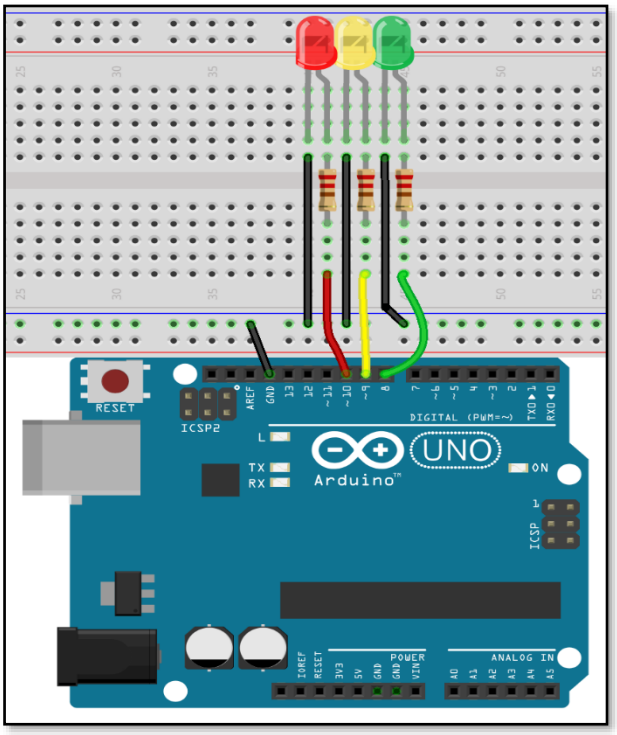


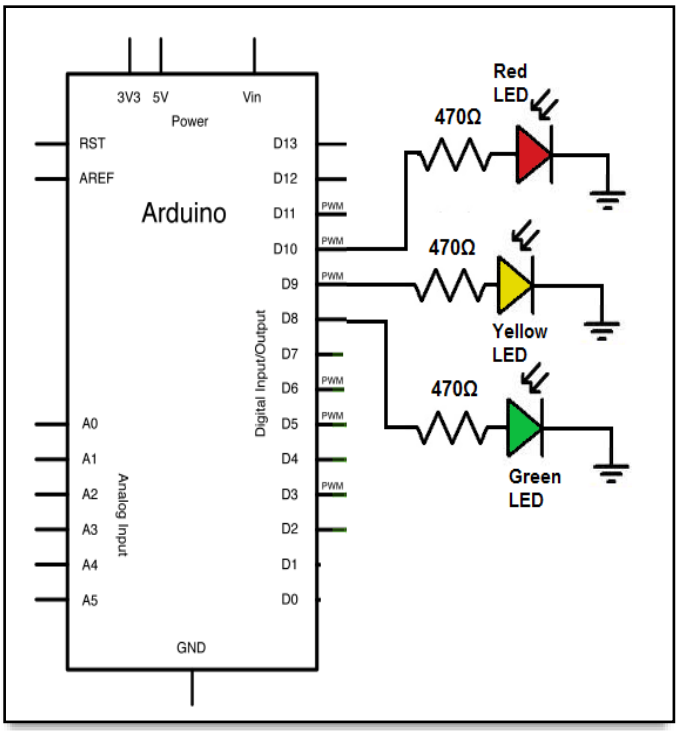


```
int timer=500;
void setup() {
  // initialisez broche 2 à la broche 7 comme sorties
for (int number = 2; number < 8; number++) {
pinMode(number, OUTPUT);
  }
}
void loop() {
 // boucle de la broche 2 à la broche 7
for (int number = 2; number < 8; number++) {
digitalWrite(number, HIGH);   // allumer la broche
delay(timer);
digitalWrite(number, LOW); // débrancher la broche
  }   // boucle de la broche 7 à la broche 2
for (int number = 7; number >= 2; number--) {
digitalWrite(number, HIGH);    // allumer la broche
delay(timer);
digitalWrite(number, LOW);     // débrancher la broche
  }
}
```

2.3 Lumières de signalisation

En bref: Le contrôleur de lumières de signalisation peut être conçu en donnant un délai déterminé entre le changement de couleur des lumières de signalisation. Dans le code, chaque lumière est allumée pendant une seconde tandis que les deux autres couleurs sont éteints et le cycle se répète avec un retard d'une seconde.
Composants requis: DEL (rouge, vert, jaune), Résistances 220Ω





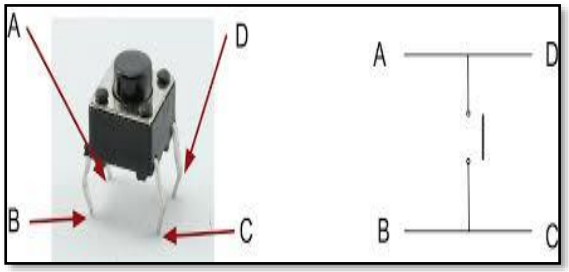
2.4 Contrôle des DELs par boutons

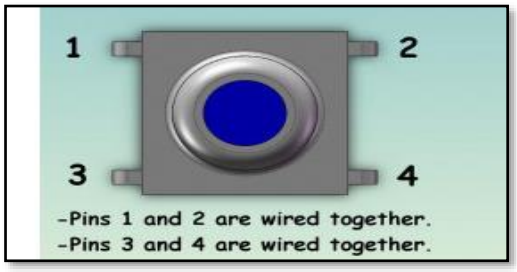
En bref: Nous utilisons un interrupteur à bouton-poussoir pour allumer et éteindre la LED. Lorsque le bouton est enfoncé, le circuit est fermé, ce qui fait que la LED s'allume. Selon le code donné ci-dessous, nous lisons la valeur sur la broche 12, si le bouton est enfoncé, la valeur sur la broche 12 sera HAUTE, par la suite la LED s'allume en mettant la broche de la LED sur HAUTE.

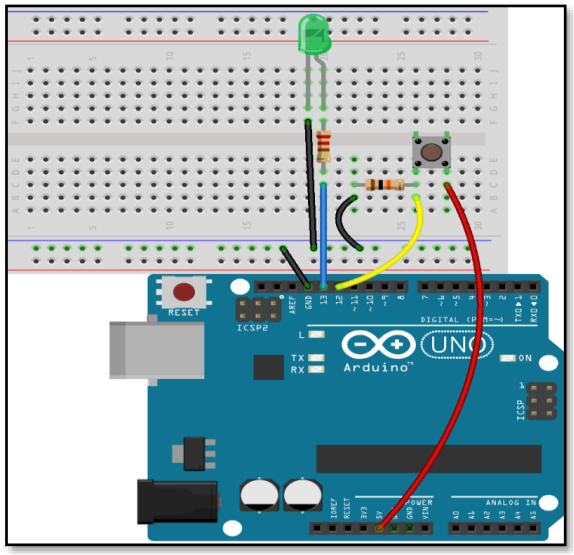
Composants requis: Bouton-poussoir, DEL, Résistance 220Ω, 10KΩ, carte expérimentale.

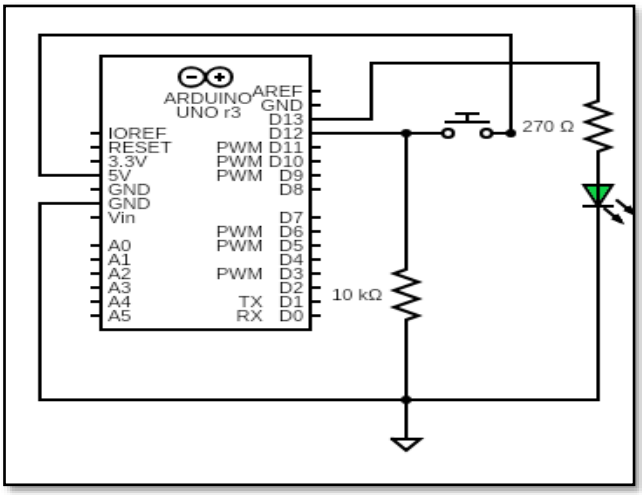
Le bouton-poussoir fonctionne selon le principe suivant

- Bouton non enfoncé = circuit déconnecté
- Bouton enfoncé = circuit connecté





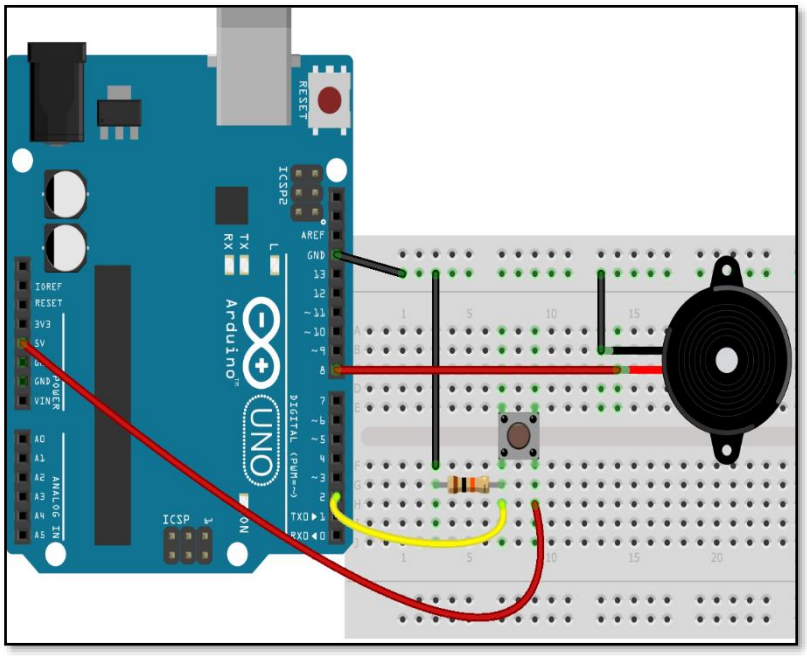


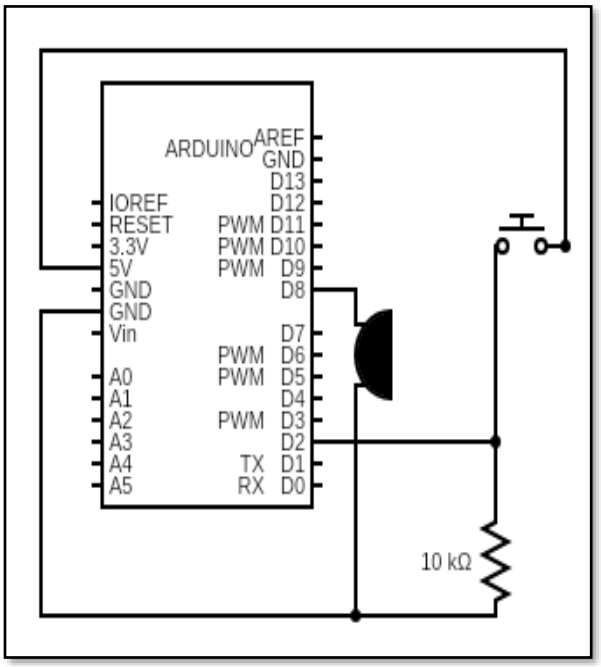



```
//DEL est allumé ou fermé par le biais d'un bouton poussoir
// Written by ABRA ELECTRONICS
const int buttonPin = 12; //le bouton poussoir connecte à la broche # 12
const int ledPin = 13; //le DEL connecte à la broche # 13
int buttonState = 0; // variable pour lecture du statut du bouton possoir
void setup()
{
pinMode(buttonPin, INPUT); //initializez le buttonPin comme entrée
pinMode(ledPin, OUTPUT); //initialisez la broche DEL comme sortie
}
void loop ()
{   // lire le statut de la valeur du bouton poussoir
buttonState = digitalRead (buttonPin);
 if (buttonState == HIGH)       // vérifiez si le bouton poussoir est HAUT (enclenché)
          {
digitalWrite (ledPin, HIGH); //allumez le DEL
          }
else
          {
digitalWrite (ledPin, LOW); //débranchement du DEL (off)
          }
}
```

2.5 Sonnette par le biais d'un avertisseur sonore

En bref: Un avertisseur sonore actif génère une tonalité à l'aide d'un oscillateur interne, de sorte que tout ce dont nous avons besoin est une tension continue que nous fournissons maintenant par l'intermédiaire d'une des broches numériques sur Arduino. Composants: BUZ-120, Resistor 220Ω, Bouton poussoir.



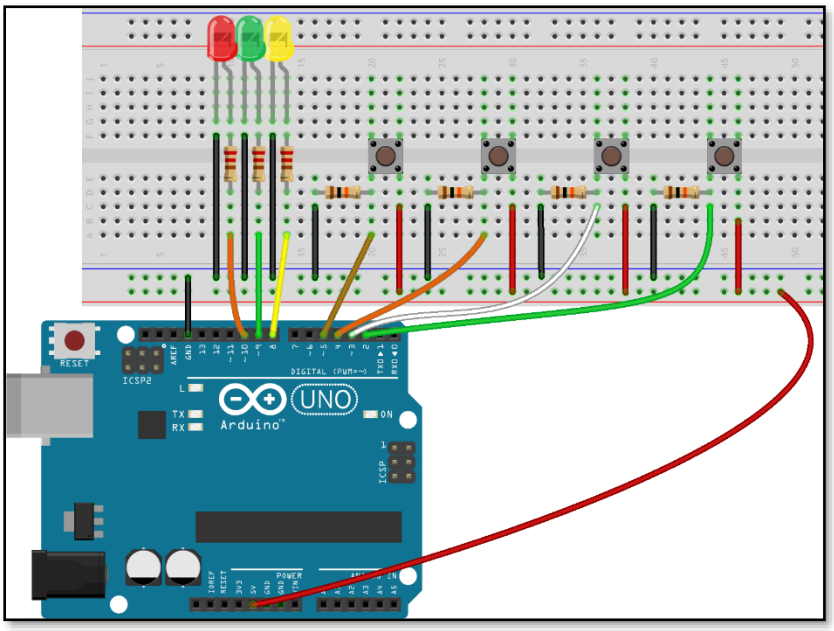


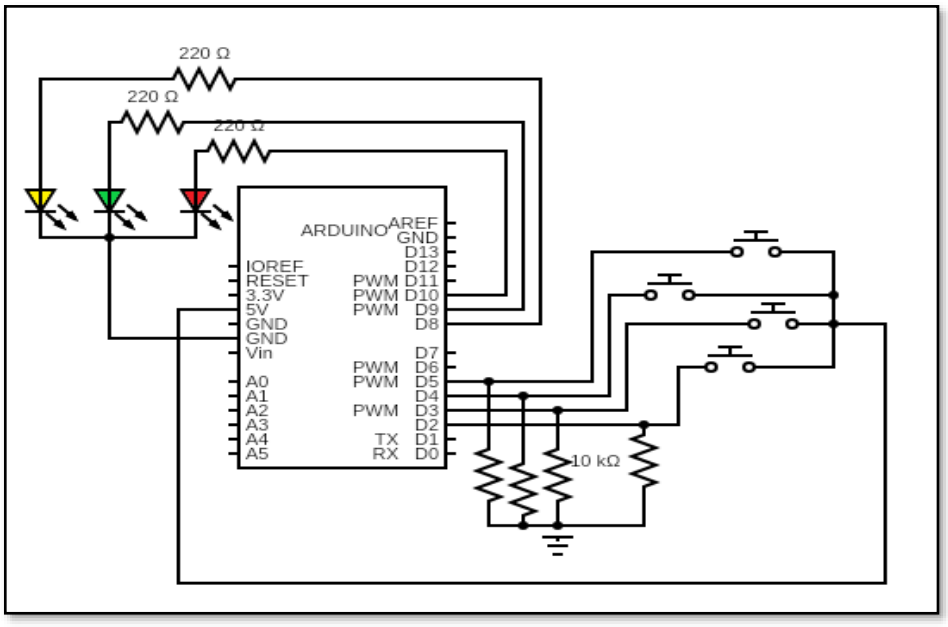

```
//Sonnette
//Allume/ferme avertisseur sonore par le biais du bouton poussoir.
// Écrit par ABRA ELECTRONIQUE
const int buttonPin = 2; //le bouton poussoir se connecte à la broche #2
const int buzzerPin = 8;  //la DEL se connecte à la broche #8
int buttonState = 0;  // variable pour lecture du bouton poussoir     statut
void setup() {
pinMode(buttonPin, INPUT); //initialisez le bouton poussoir comme entrée
pinMode(buzzerPin, OUTPUT); //initialisez la broche de l'avertisseur sonore comme sortie
      }
void loop() {
       //lire le statut  de la valeur du bouton poussoir
 buttonState = digitalRead(buttonPin);
        if (buttonState == HIGH ) { //et vérifiez si le bouton poussoir est enclenché, si oui, le statut est HAUT
for (int i = 0; i < 25; i++) {
digitalWrite(buzzerPin, HIGH); //Activez l'avertisseur sonore
delay(500);       //attendre une demie seonde
digitalWrite(buzzerPin, LOW); //Déactivez l'avertisseur sonore
delay(500);       // attendre une demie seonde
         }
       }
     }
```

2.6 Expérimentation responsive

En bref: Grâce à cette expérience, nous apprenons à contrôler les différentes couleurs des DEL. Nous utilisons également le bouton de réinitialisation pour réinitialiser (mettre toutes les LED à l'état éteint).

Composants: Bouton poussoir (4), DEL rouge, DEL jaune, DEL verte, Résistances 220Ω (3), 10KΩ (4)

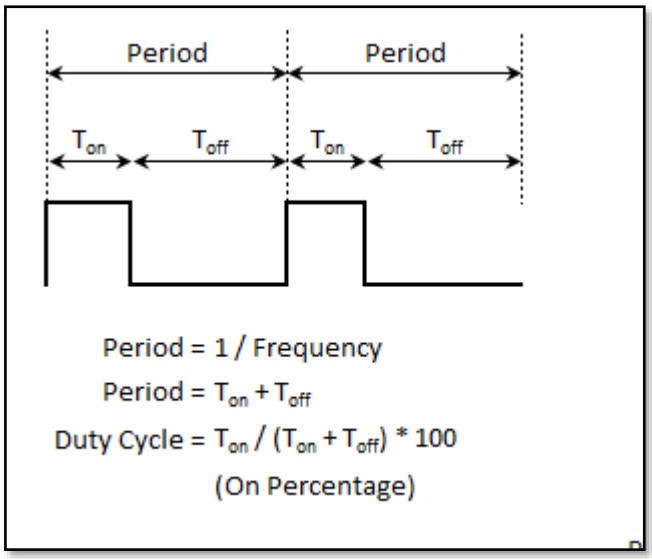


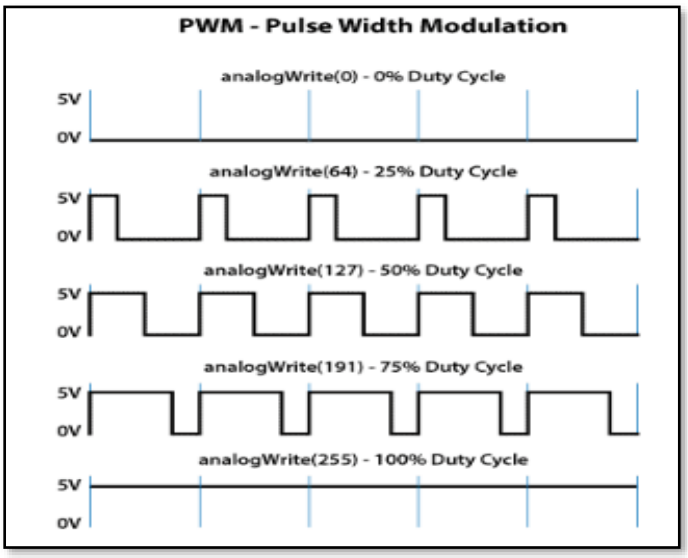


```
int redlight=10;    //assignez la DEL rouge à la broche #10
int yellowlight =8;  // assignez la DEL jaune à la broche 8
int greenlight =9;   //a assignez la DEL verte à la broche 9
int redbutton=5;    //assignez bouton rouge à la broche 5
int yellowbutton=4;  // assignez bouton jaune à la broche 4
int greenbutton=3;   // assignez bouton verte à la broche 3
int reset=2;   // assignez bouton reset à la broche 2
int red, yellow, green;
void setup() {
pinMode(redlight,OUTPUT); //assignez broche 10 (redlight) comme sortie
pinMode(yellowlight ,OUTPUT); //assignez broche 8 (yellowlight) comme sortie
pinMode(greenlight ,OUTPUT); // assignez broche 9 (greenlight) comme sortie
pinMode(redbutton,INPUT);
pinMode(yellowbutton,INPUT);
pinMode(greenbutton,INPUT);
}
void loop()  {
red=digitalRead(redbutton); //lire valeur bouton à la broche 5
yellow=digitalRead(yellowbutton); // lire valeur bouton à la broche 4
green=digitalRead(greenbutton); // lire valeur bouton à la broche 3
if(red==HIGH)  OnRed(); //vérifiez enclenchement de bouton rouge, si bouton est enclenché  alors
observez fonction OnRed
if(yellow==HIGH) Onyellow(); // vérifiez enclenchement de bouton rouge, si bouton est enclenché
alors observez fonction OnYellow
if(green==HIGH)  Ongreen();//vérifiez enclenchement de bouton rouge, si bouton est enclenché
alors observez fonction OnGreen
void OnRed() { // fonction pour bouton rouge enclenché
while(digitalRead(reset)==0) {
digitalWrite(redlight, HIGH); //allumage lumière rouge
digitalWrite(greenlight , LOW);
digitalWrite(yellowlight , LOW);
 }
clear_lights();
}
void Onyellow() { // fonction pour bouton jaune enclenché
while(digitalRead(reset)==0){
digitalWrite(redlight,LOW);
digitalWrite(greenlight ,LOW);
digitalWrite(yellowlight ,HIGH); // allumage lumière jaune
  }
clear_lights();
}
void Ongreen() { // fonction pour bouton vert enclenché
  while(digitalRead(reset)==0)
  {
  digitalWrite(redlight,LOW);
  digitalWrite(greenlight ,HIGH); // allumage lumière verte
  digitalWrite(yellowlight ,LOW);
  }
  clear_lights();
}
void clear_lights() // fonction pour extinction de tous les lumières connectées à la broche 8 à 10
{
  for (int i = 8; i < 11; i++) {
digitalWrite(i, LOW);
}
```

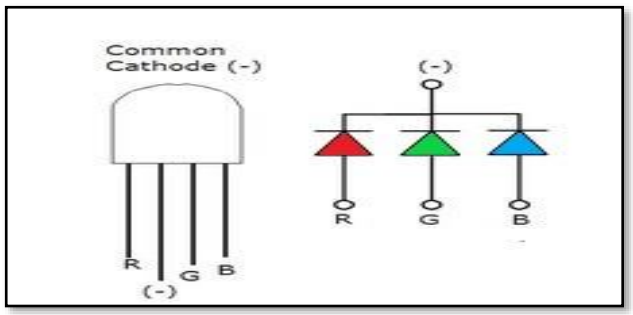
2.7 DEL RVB

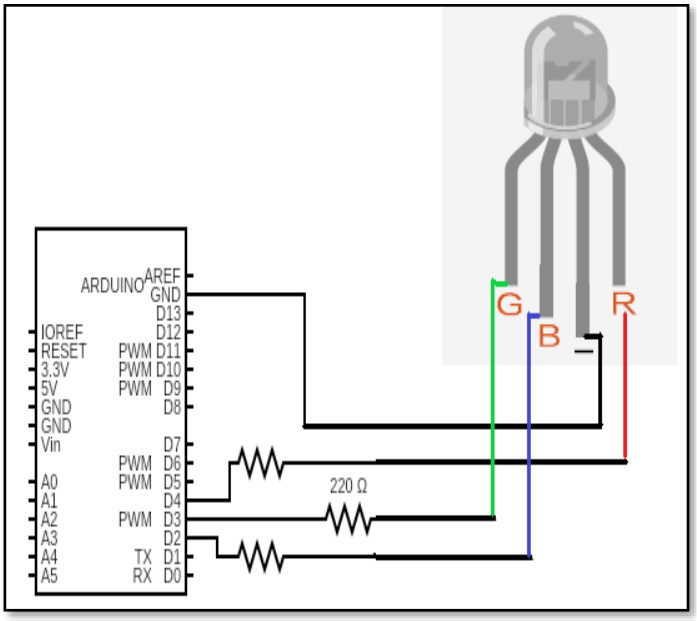
En bref: Dans cet exemple, nous apprendrons à faire varier la couleur des DELs RVB à l'aide d'Arduino. La variation des valeurs PWM provoque le changement de couleur. La modulation de largeur d'impulsion (ou PWM) est une technique de contrôle de la puissance. Nous l'utilisons également ici pour contrôler la luminosité de chacune des DEL. La PWM permet de contrôler la quantité de puissance fournie à un appareil. Les concepts de cycle de service et de fréquence sont utilisés dans la PWM pour contrôler la luminosité des DEL RGB. Le rapport cyclique indique la durée pendant laquelle l'impulsion est ÉLEVÉE sur sa période. En termes simples, ce rapport cyclique est une valeur en pourcentage de l'état ON par rapport à l'état OFF. La figure ci-dessous montre la formule de calcul du rapport cyclique. Il est mesuré en pourcentage et indique le voltage entre les niveaux OFF et ON (généralement 0V et 5V). Composants: Bouton poussoir (4), DEL rouge, DEL jaune, DEL verte, Résistances 220Ω (3), 10KΩ (4), Carte Expérimentale.

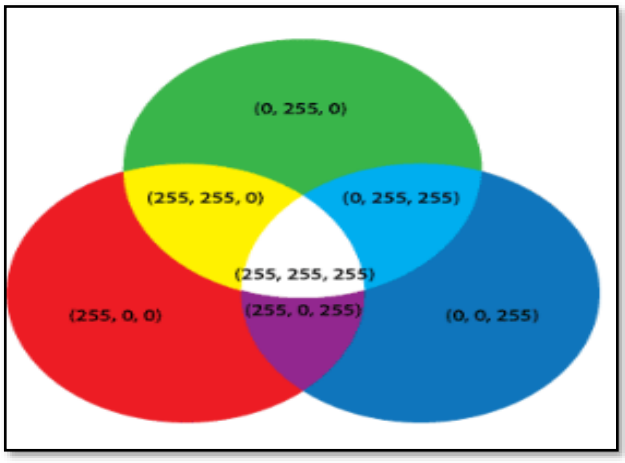


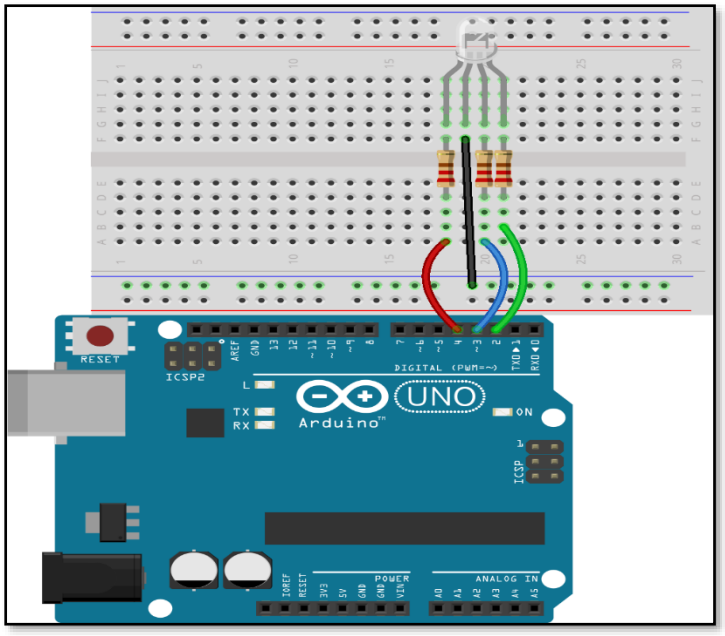


33







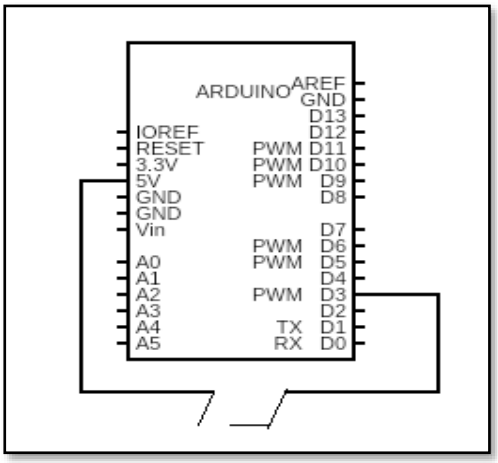


Pour étendre la durée de couleur vous pouvez augmenter le délai dans le code.

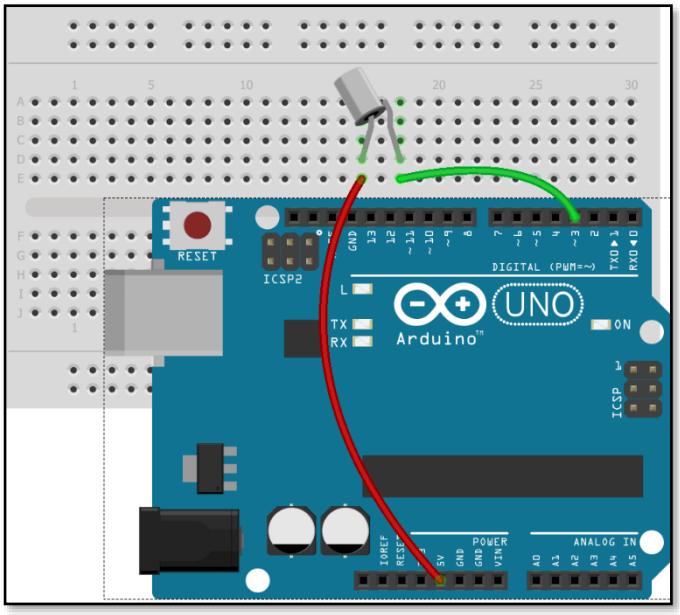
```
//DEL RVB
// Écrit par ABRA ELECTRONIQUE
int blue_pin = 2;
int green_pin = 3;
int red_pin= 4;
void setup() {
  pinMode(red_pin, OUTPUT);
  pinMode(green_pin, OUTPUT);
  pinMode(blue_pin, OUTPUT);
}
void loop() {
  RGB_CODE(255, 0, 0); // Couleur rouge
  delay(500); // délai par demie-seconde
  RGB_CODE(0, 255, 0); // Couleur verte
  delay(500);
  RGB_CODE(0, 0, 255); // Couleur bleue
  delay(500);
RGB_CODE(255, 255, 0); // Couleur jaune
  delay(500);
  RGB_CODE(255, 255, 255); // Couleur blanche
  delay(500);
RGB_CODE(255, 255, 125); // Couleur framboise
  delay(500);
  RGB_CODE(0, 255, 255); // Couleur cyan
  delay(500);
  RGB_CODE(255, 0, 255); // Couleur Magenta
  delay(500);
}
void RGB_CODE(int red_code, int green_code,
int blue_code)
{ // fonction pour lecture les valeurs du RVB
  analogWrite(red_pin, red_code);
  analogWrite(green_pin, green_code);
  analogWrite(blue_pin, blue_code);
}
```

2.8 Commutateur d'inclinaison (SENS-39)

En bref: Le commutateur d'inclinaison est un interrupteur qui ouvre et ferme un circuit électrique lorsqu'il est incliné à certains angles. Après avoir connecté le circuit, si la plaque expérimentale est inclinée selon un certain angle, la DEL s'allume. Ces interrupteurs sont petits, peu coûteux, de faible puissance et faciles à utiliser. Composants: SENS-39



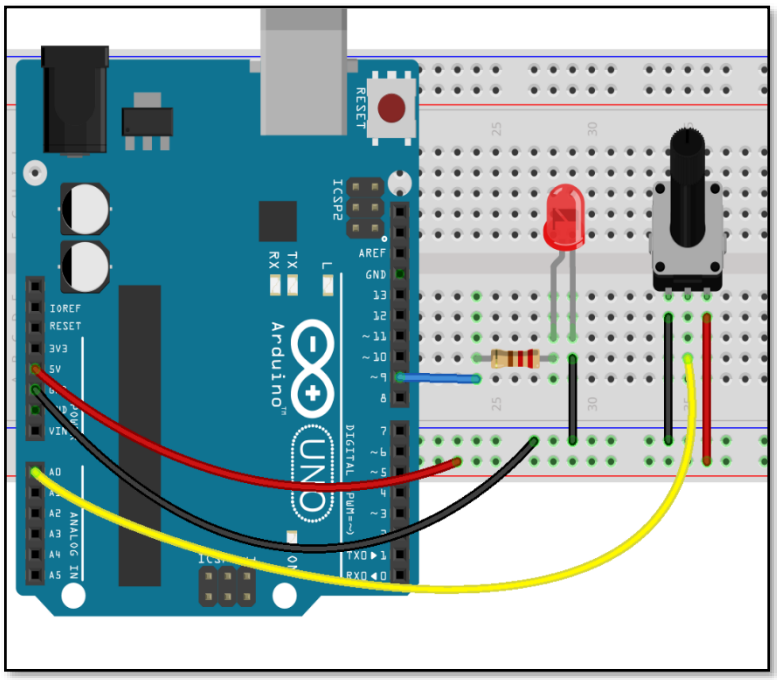
```
// Commutateur d'inclinaisom
// Écrit par Abra Électronique Inc.
int led_pin = 13; // DEL interne connecte la broche 13
int tilt_switch = 3; // commutateur d'inclinaison
connect à la broche 3.
int value;

void setup()
{
        pinMode(led_pin, OUTPUT);
        pinMode(tilt_switch, INPUT);
}
void loop()
{
value = digitalRead(tilt_switch); //Lire valeur de
l'inclinaison
        if(value == HIGH)
        {
                digitalWrite(led_pin, HIGH);
        }
        else
        {
                digitalWrite(led_pin, LOW);
```

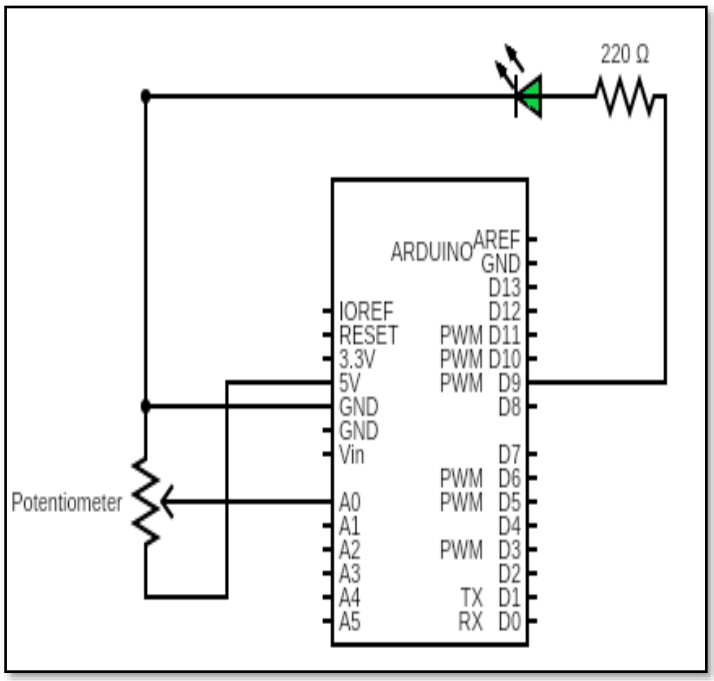


2.9 Contrôle d'une DEL par potentiomètre

En bref: La luminosité de la DEL est contrôlée à l'aide d'un potentiomètre en modifiant la valeur de la résistance.

Composants: Plaque expérimental, 220Ω résistance, DEL.

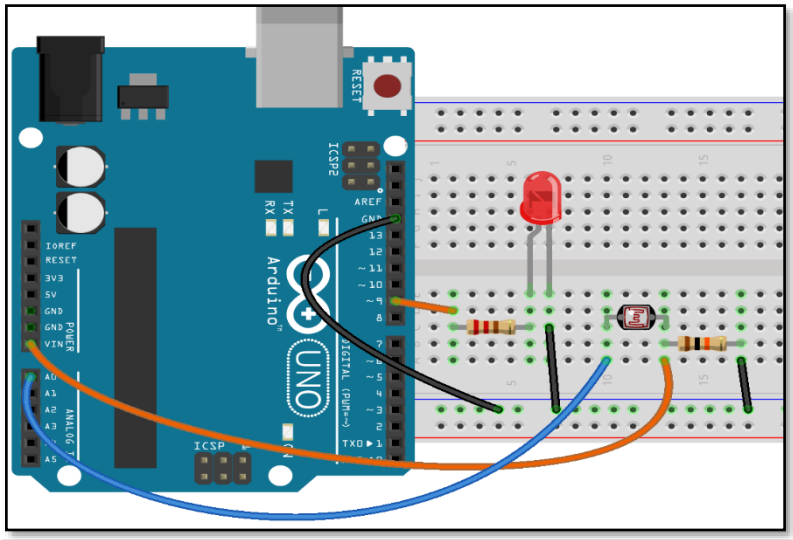


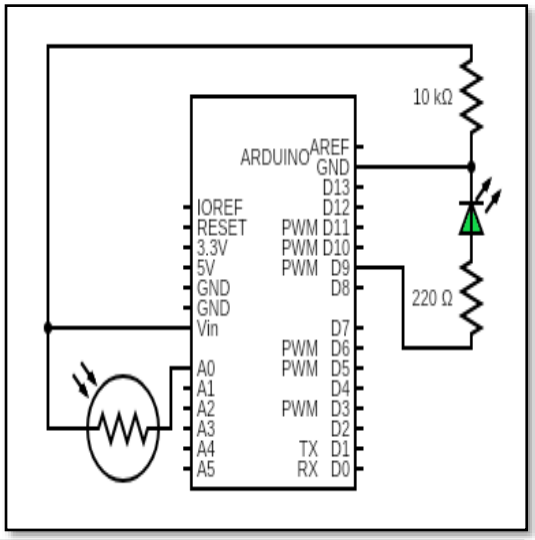


```
//        Contrôle d'une DEL par potentiomètre
int Analoginput=A0;  //Broche analogue A0 reçoit données contrôle par potentiomètre.
int out1=9;   // Broche 9 est la sortie auquelle est connectée lea DEL.
void setup()
{
pinMode(out1, OUTPUT); //Broche 9 est une sortie
}
void loop()
{
int recievedinput=analogRead(Analoginput); //Lecture de la tension par le potentiomètre
int ledbrightness=recievedinput/4;      //Division par 4 pour l'amener dans une gamme de 0 – 255
analogWrite(out1, ledbrightness);
}
```

2.10 Photorésistance (Photo-300)

En bref: Une photorésistance ou cellule photoélectrique est une résistance variable commandée par lumière. Elle fonctionne selon le principe de la photoconductivité. La résistance d'une photorésistance diminue lorsque l'intensité de la lumière incidente augmente. Lorsque la résistance diminue, le courant circule à travers elle. Servo (SG-90)



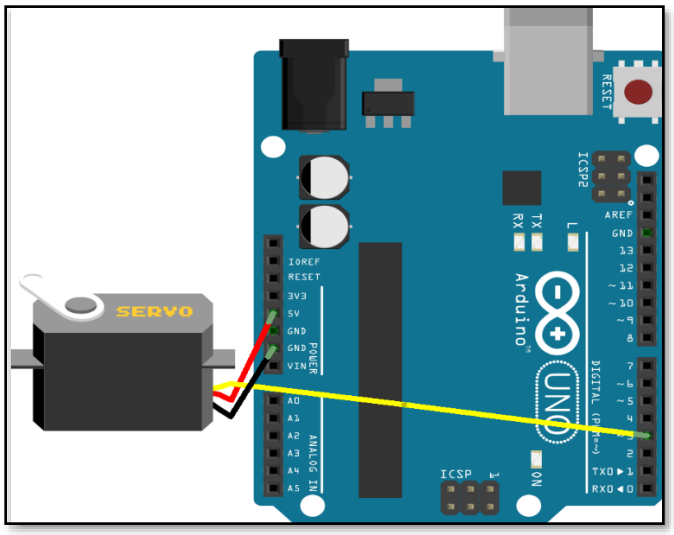


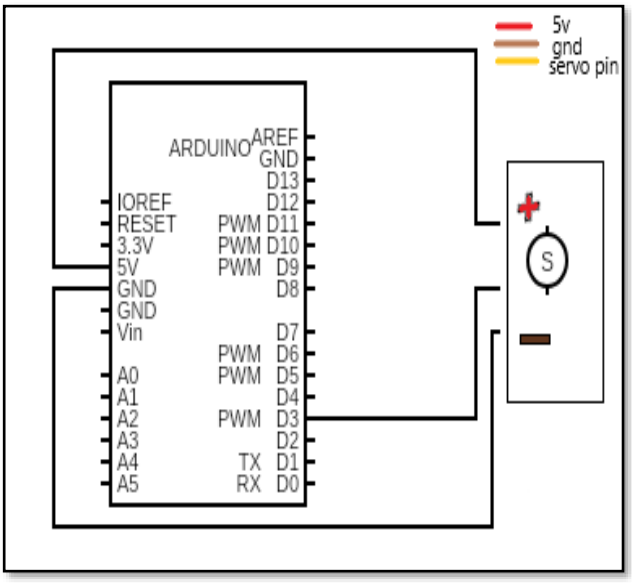

```
//Usage d'une photorésistance pour allumer une DEL dans le noir.
const int photocell = A0; // Photorésistance à la broche A0 analogue de la carte Arduino
const int ledPin=9;      // broche DEL à la broche 9  Arduino
int digitalvalue; // pour entreposer valeur digitale de la photorésistance (0-1023)
void setup(){
 pinMode(ledPin, OUTPUT);  // Fixez  broche DEL-  # 9 comme sortie
 pinMode(photocell, INPUT);
}
void loop(){
  digitalvalue = analogRead(photocell);

  if (digitalvalue > 30){
    digitalWrite(ledPin, LOW);  // Éteignez la DEL s'il y a suffisance de lumière dans la salle
  }
  else{
    digitalWrite(ledPin, HIGH); // Allumez la DEL si la salle est sombre
  }
}
```

2.11 Servo Moteur

En bref: Un servomoteur est un dispositif électrique qui peut faire tourner un objet avec une grande précision. Si vous voulez faire tourner un objet à un angle spécifique, nous utilisons un servomoteur. Le servomoteur est contrôlé par PWM (Pulse width Modulation). Dans ce projet, nous pouvons connecter de petits servomoteurs directement à un Arduino pour contrôler la position de la tige avec une grande précision. L'Arduino envoie un signal PWM au servomoteur qui tourne alors d'un angle dépendant de la largeur de l'impulsion.

Composants: Servo moteur

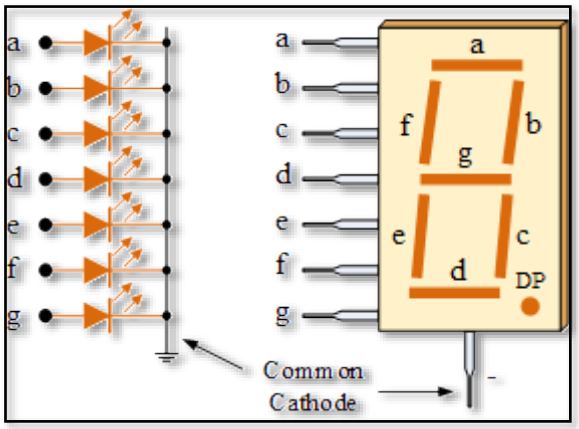


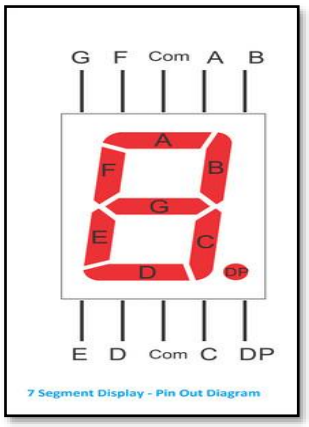



```
#include <Servo.h>  // Inclus la bibliothèque du servo moteur
int ser_pin = 4; //  Broche Servo connectée à la broche # 4
Servo ser_obj;  //  instancier objet servo
int i=0;
void setup() {
  ser_obj.attach(ser_pin);
}
void loop(){
   while(ser_pin=HIGH){
  for (i = 0; i <= 360; i++) { //sens anti horaire
   ser_obj.write(i);
   delay(10);
  }
  for ( i = 360; i>= 0; i--) { //sens horaire
   ser_obj.write(i);
   delay(10);
  }
   }
}
```

2.12 Affichage 7 Segment 1-chiffre (S-5101AS)

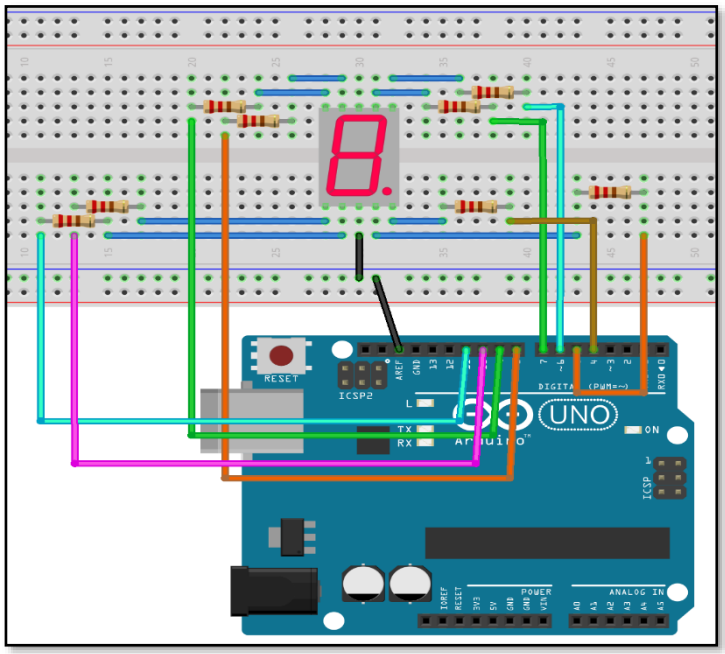
En bref: Interface d'un affichage de sept segments à Arduino. Dans l'affichage à cathode commune, toutes les connexions de cathode des segments de LED sont reliées à la terre. Les segments sont éclairés en appliquant un signal "HIGH", ou un signal logique "1" via une résistance de limitation de courant pour polariser les bornes de l'anode dans le sens direct. Par exemple, si vous voulez éclairer le segment "a", la broche 7 de l'Arduino qui est connectée au segment "a" est mise en position HAUT

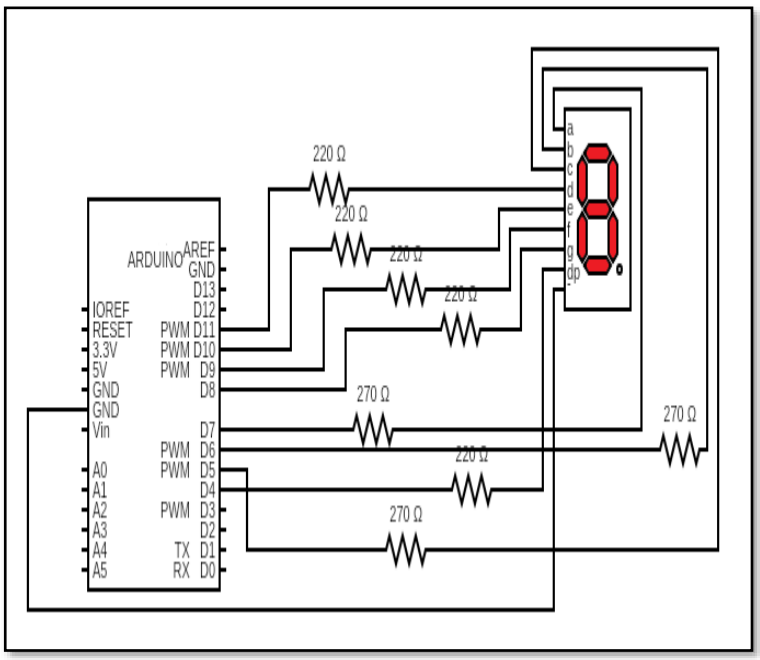




7 segment	Uno pin
a	7
b	6
c	5
d	11
e	10
f	8
g	9
-	gnd

Table for Connections





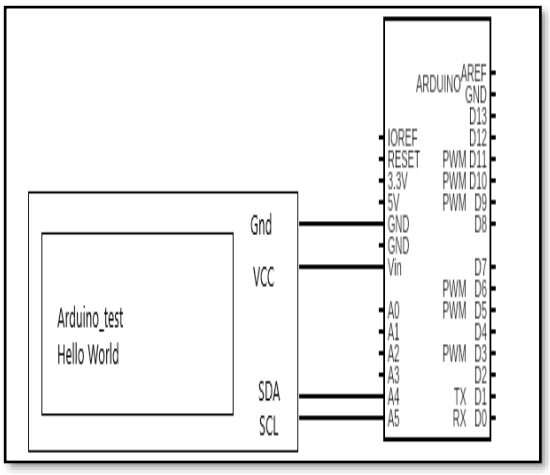
```
// Déclarez réseau pour 7 bits et activez le segment requis, le segment est dans l'ordre suivant (c,b,a,f,g,e,d)
int alpha[7][7] = { { 1,0,1,1,1,1,1 },// G
                    { 0,0,1,1,1,1,0 },   // F
                    { 0,0,1,1,1,1,1 },   // E
                    { 1,1,1,1,0,1,1 },   // D
                    { 0,0,1,1,0,1,1 },   // C
                    { 1,1,1,1,1,1,1 },   // B
                    { 1,1,1,1,1,1,0 }  };  // A
void print_data(int);
void setup() {
   int  i;
   for(i=5;i<=11;i++)
      pinMode(i,OUTPUT);
}
void loop() {
  for (int counter = 7; counter >0; counter--) {
   delay(1000);
   print_data(counter-1);
  }
  delay(1000);
}

// cette fonction écrit les valeurs aux broches de l'afficheur sept segment
void print_data(int number) {
  int out_pin=5;
  for (int j=0; j < 7; j++) {
   digitalWrite(out_pin, alpha[number][j]);
   out_pin++;
  }
}
```

2.13 LCD 16x2 avec I2C (LCD-MOD-13)

En bref: Le LCD I2C utilise l'interface I2C, il a donc 4 broches NAMELY GND, VCC, SDA (signal de données I2C), SCL (signal d'horloge I2C).

Composants : LCD I2C, fils de raccordement.
Remarque : si la LED n'affiche pas de caractères, essayez de régler le contraste de l'écran LCD.

```
#include <Wire.h> // Bibliothèque pour communication I2C
#include <LiquidCrystal_I2C.h> // installez cette bibliothèque
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C adresse 0x27, 16 colonnes et  2 rangées

void setup()
{
  lcd.init();// initialisez le lcd
  lcd.backlight();
  lcd.setCursor(1,0);       // fixez  cursor à  (0, 0)
  lcd.print("Arduino_test");  // imprimez message à (0, 0)
  lcd.setCursor(0,1);       //Fixez au rangée de bas
  lcd.print("Hello World"); // imprimez message à (0,1)
}
void loop()
{}
```



2.14 Thermistor (334-103)

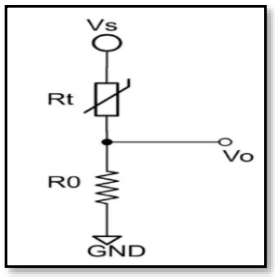
En bref: Um thermistor est un type of résistance qui est dépendant sur la température. Les thermistors sont de deux types opposés:

- Avec un thermistor **NTC**, la résistance ***décroit*** alors que la température monte. Un thermistor NTC sert communément comme capteur de température.
- Avec un thermistor **PTC**, la résistance ***augmente*** alors que la température monte.

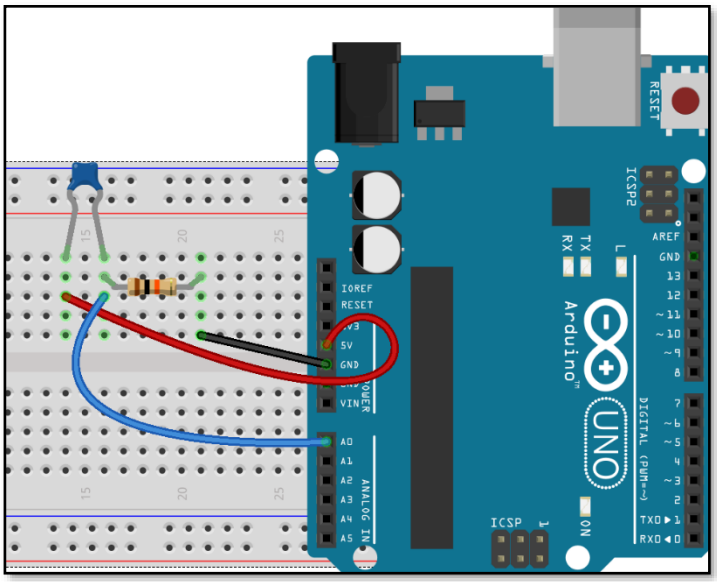
Pour cet expériment, nous créérons un diviseur de tension entre le thermistor et une résistance de 10kΩ et ferons le calcul.

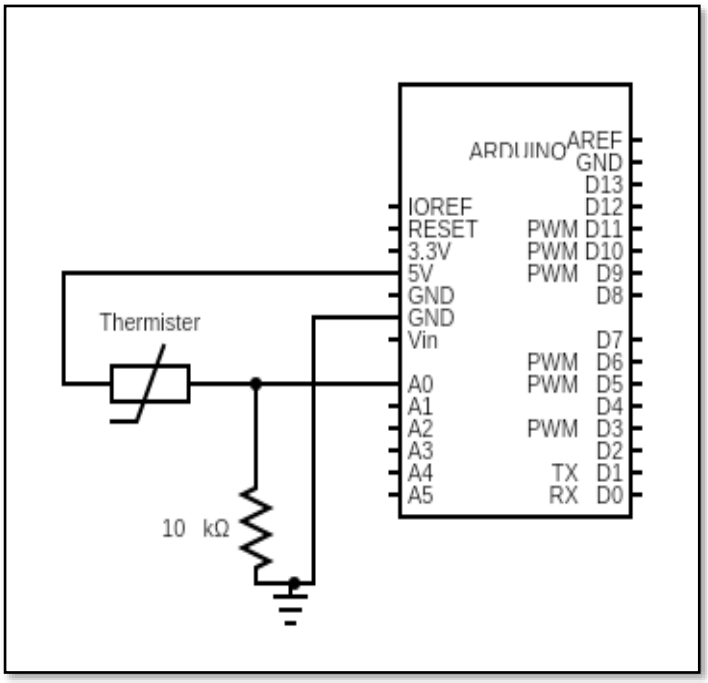
- **Vo = Vs * (R0 / (Rt + R0**
- **Rt = R0 * ((Vs / Vo) - 1)**
- **1/T = A + B*ln(R) + C*(ln(R))^3**

Composants: 10K thermistor, 10kΩ résistance.



```
#include <math.h>
int thermistor = A0;
int Vo;
float R1 = 10000; / / valeur de résistance (R1)
float logR2, R2, T, Tc, Tf;
float A = 1.009249522e-03, B = 2.378405444e-04, C =
2.019202697e-07; / / Valeur de constants

void setup() {

Serial.begin(9600);// fixez vitesse de transmission
pour communication en série

}
// calculs
void loop() {

  Vo = analogRead(thermistor);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (A + B*logR2 + C*logR2*logR2*logR2));
  Tc = T - 273.15; //conversion de Kelvin à Celcius
  Tf = (Tc * 9.0)/ 5.0 + 32.0; //celcius à Fahrenheit

  Serial.print("Temperature measured: ");
  Serial.print(Tf);
  Serial.print(" Fahrenheit; ");
  Serial.print(Tc);
  Serial.println(" Celcius");
  delay(500);
}
```



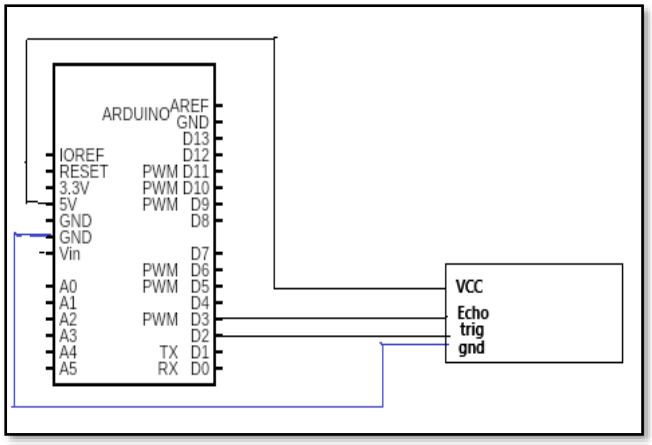


2.15 Capteur Ultrasonique (HC-SR04)

En bref: Le capteur ultrasonique HC-SR04 est un capteur qui mesure la distance. Il émet un signal ultrasonique à 40000 Hz (40kHz) qui se déplacer dans l'air et qui rebondira au module s'il rencontre un objet ou obstacle sur sa route. Utilisant le temps de voyage et la vitesse du son la distance est calculée.

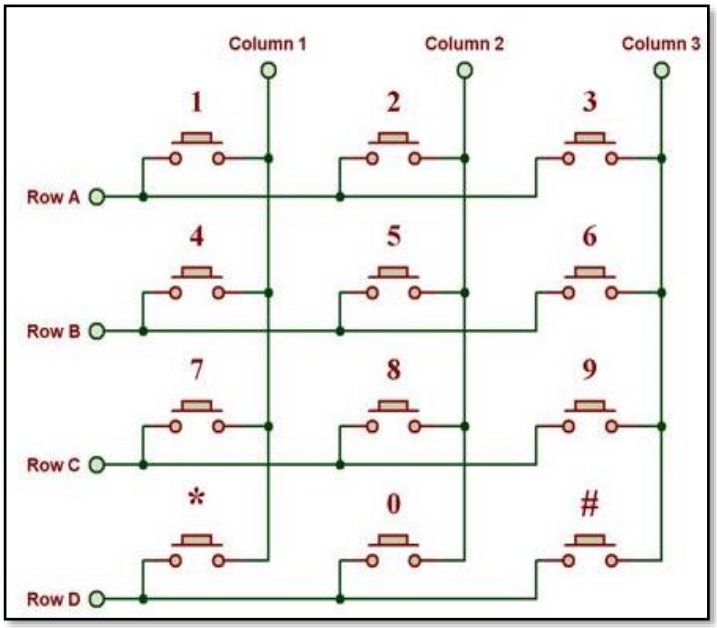
Si x est la distance entre point A et point B alors 2x sera la distance de voyage de A à B et de B à A. Nous savons que la vitesse de l'ultrason est d'environ 340 mètres par seconde.

- Distance = Vitesse*Temps.
- Distance=2x.
- 2x= 340*Temps Maintenant le temps est rendu par pulsation ln() function. x=340*Temps/2




```
const int echo=3; //  D3 sur broche Arduino à la broche Echo du HC-SR04

const int trig=2 ;// D2 sur Arduino à la broche Trig de HC-SR04
long travel_time; //   temps de voyage de l'onde de son
int distance; // variable pour la mesure de distance
void setup() {
  pinMode(trig, OUTPUT); // Fixez trig (broche 2) comme SORTIE
  pinMode(echo, INPUT); // Fixez broche 3 écho) comme ENTRÉE
  Serial.begin(9600); // fixez vitesse de transmission  pour communication
}
void loop() {
  // ôtez la condition trig
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);  // activez trig pour 20 microsecondes
  delayMicroseconds(20);
  digitalWrite(trig, LOW);
  // Lit une pulsation (soit HAUTE ou BASSE) sur une broche. A titre d 'exemple, si la valeur est HAUTE
  // pulseIn() attend que la broche se rende de BAS à HAUTE,  commence la temporisation, par la suite attend sur
la broche d'aller au BAS et termine la temporisation.
  travel_time = pulseIn(echo, HIGH);
  // Calcul de la distance
  distance = travel_time * 0.034 / 2; // Temps de voyage unidirectional = (de et là temps/2)
  // Affiche la distance sur le moniteur en série
  Serial.print("Distance measured is: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

2.16 4x3 Clavier (419-ADA)



En bref: Le clavier 4x3 comporte 4 lignes et 3 colonnes. Si aucune touche n'a été enfoncée, toutes les colonnes resteront en HAUT. L'appui sur une touche permet de court-circuiter l'une des lignes de la ligne vers l'une des lignes de la colonne, permettant ainsi au courant de circuler entre elles. Par exemple, lorsque la touche "2" est enfoncée, la colonne 2 et la ligne 1 sont court-circuitées, de sorte que la colonne 2 et la ligne 1 sont BAS.

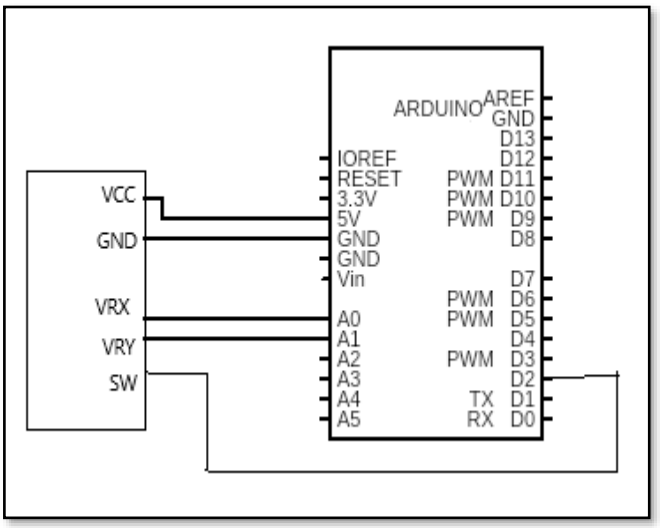
Composants : Clavier 4x3, fils de pontage

Remarque : lors de la compilation du code, s'il y a une erreur concernant Keypad.h, par exemple "library does not exist", assurez-vous d'ajouter "keypad library" en faisant le croquis->inclure la bibliothèque->gérer la bibliothèque. Dans le type sélectionnez Arduino search for keypad et installez keypad for matrix type library.

2.17 Manette PS2 (AM-JOYSTICK)

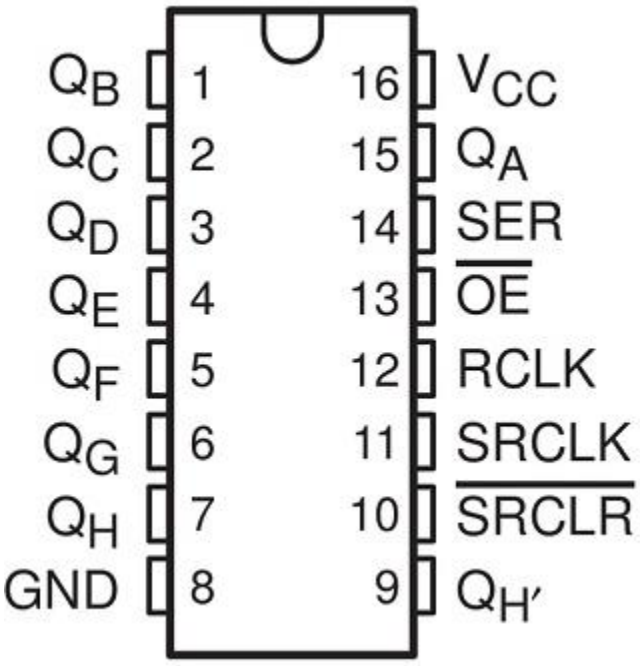
En bref: Les contrôleurs PS2 ont deux manettes analogiques. Les manettes analogiques sont essentiellement des potentiomètres, ils renvoient donc des valeurs analogiques. Ici, deux potentiomètres sont positionnés à angle droit l'un par rapport à l'autre sous la manette. Le courant circule en continu à travers les manettes et la quantité de courant est déterminée par la quantité de résistance. La résistance est augmentée ou diminuée en fonction de la position de la manette. L'objectif de la manette est de communiquer le mouvement en 2D (axe X et axe Y) à Arduino. Par conséquent, deux potentiomètres 10K indépendants (un par axe) sont utilisés dans la manette.

Composants: Manette, fils.



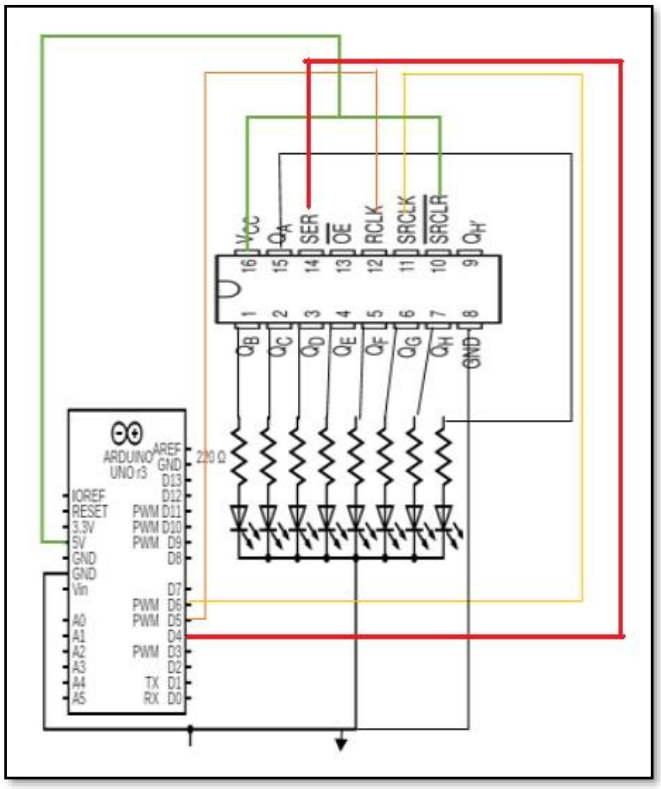
```
// JOYSTICK
const int SW = 2; // broche digitale connectée au commutateur
const int VRx = A0; // broche analogue connectée à sortie X
const int VRy =A1; // broche analogue connectée à la sortie Y
int mapX=0;
int mapY=0;
int x=0;
int y=0;
void setup() {
  pinMode(SW, INPUT);
  digitalWrite(SW, LOW);
  Serial.begin(9000);
}
void loop() {
  Serial.print("Joystick  ");
  Serial.print(digitalRead(SW));
  Serial.print("\n");
  Serial.print("X-axis: ");
  x=analogRead(VRx);
   mapX = map(x, 0, 1023, -512, 512);
  Serial.print(x); // imprimez valeur de x
  Serial.print("\n");
  Serial.print("Y-axis: ");
   y=analogRead(VRy);
  mapY = map(y, 0, 1023, -512, 512);
  Serial.print(y); // imprimez valeur de y
  Serial.print("\n\n");
  delay(500);
}
```

2.18 Interface d'un registre à décalage (74HC595)



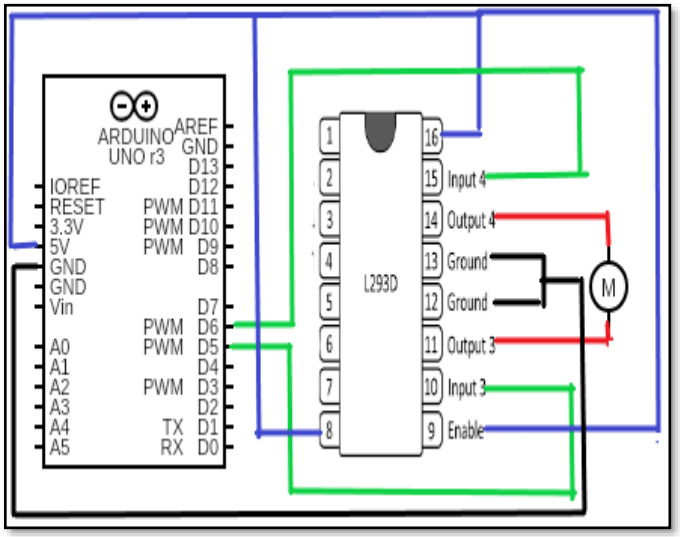
- GND devrait se brancher sur le mise à la terre du Arduino.
- VCC est l'alimentation pour le 74HC595 registre à décalage que nous connectons à la broche 5V sur l' Arduino.
- La broche SER (Entrée de série) sert à fournir les données dans le registre à décalage un bit à la fois.
- SRCLK (Horloge registre à décalage) est l'horloge pour le registre à décalage. Les bits sont décalés sur la pointe montante de l'horloge.
- RCLK (Horloge Registre / Verrouillage lorsque la logique est HAUTE, le contenu du registre à décalage est copié dans l' Entreposage/Verrouillage du registre; qui ultimement se manifeste comme la sortie.
- La broche SRCLR (Registre de décalage Clear) nous permet de réinitialiser le registre de décalage, rendant tous ses bits 0, tous ensemble. Ceci est une broche de logique négative, nous devons initialise la broche SRCLR comme BAS pour la réinitialisation. Lorsque la réinitialisation n'est pas requise, cette broche devrait être HAUTE.
- QA–QH (Output Enable- Sortie permise) sont les broches de sortie connectées aux sorties tel les DELS.

```
int latchPin = 5; //à RCLK
int clockPin = 6; //à SRCLK
int dataPin = 4; //à SER
void setup()
{
   Serial.begin(9600);
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
//comptez de 1 à 265 et affichez les valeurs binaires
sur les DELs
  for ( int i = 0; i < 256;i++)
  {
  digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, MSBFIRST, i);
   Serial.print(i);
   digitalWrite(latchPin, HIGH);
    delay(500);
  }
}
```



2.19 Interface du pilote de moteur L293D (L293D)

En bref: e LD293D pilote de Moteur peut tourner deux moteurs à la fois dans un sens ou dans l'autre. Dans cette expérience, nous utilisons le LD293D pour faire fonctionner un seul moteur à courant continu. Il fonctionne sur le principe du pont en H qui permet de faire circuler la tension dans les deux sens.

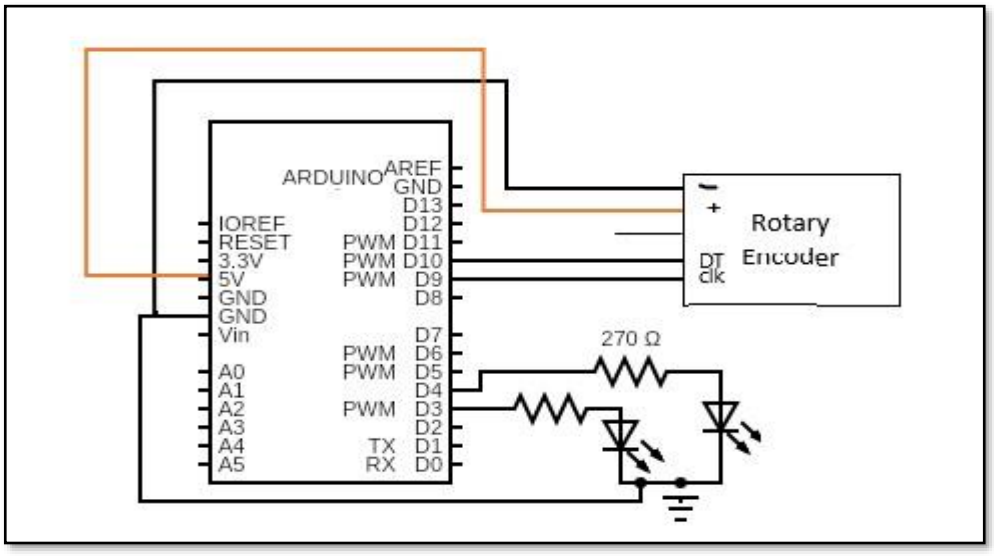



```
// Interface du pilote de moteur L293D
const int out_pin1 = 5;  // à broche 14 de L293
sens horaire
const int out_pin2  = 6;  //Broche 11 de L293
sens antihoraire
void setup(){
   //Fixez broches de sorties
   pinMode(out_pin1, OUTPUT);
   pinMode(out_pin2, OUTPUT);
}
void loop(){
 while(1){ // rotation sens horaire
  digitalWrite(out_pin1, HIGH);
   digitalWrite(out_pin2, LOW);
   delay(5000);
  // stop motor
   digitalWrite(out_pin1, LOW);
   digitalWrite(out_pin2, LOW);
    delay(500);
  // rotation sens antihoraire
   digitalWrite(out_pin1, LOW);
   digitalWrite(out_pin2, HIGH);
   delay(5000);
}
```

2.20 Encodeur Rotatif (AM-127)

En bref: Un encodeur rotatif est un capteur de position utilisé pour déterminer la position angulaire d'une tige en rotation.

Nous avons utilisé 2 DELs: l'une pour indiquer le sens des aiguilles d'une montre et l'autre pour le sens inverse.



```
// Encodeur rotatif
const int DT=10; // broche DT de l'encodeur rotatif
const int CLK=9;// broche CLK de l'encodeur rotatif
const int out1=3;//DEL rouge
const int out2=4;//DEL verte
 int Position = 0;
 int current_position;
 int previous_position;
 void setup() {
  pinMode (CLK,INPUT);
  pinMode (DT,INPUT);
  pinMode (out1,OUTPUT);
  pinMode (out2,OUTPUT);
  Serial.begin (9600);// fixez taux de baud
    previous_position = digitalRead(CLK); // lire valeur courante de CLK
 }
 void loop() {
  current_position = digitalRead(CLK);
    if (current_position != previous_position)
    {
       if (digitalRead(DT) != current_position)
       {
     Position ++;
     digitalWrite(out1, HIGH); //sur DEL si en sens horaire
      digitalWrite(out1, LOW);
    }
    else {
     Position --;
     digitalWrite(out2, HIGH); // sur DEL si en sens antihoraire
      digitalWrite(out2, LOW);
    }
    Serial.print("Position: ");
    Serial.println(Position);
  }
  previous_position = current_position; // update previous state of CLK with the current state
 }
```

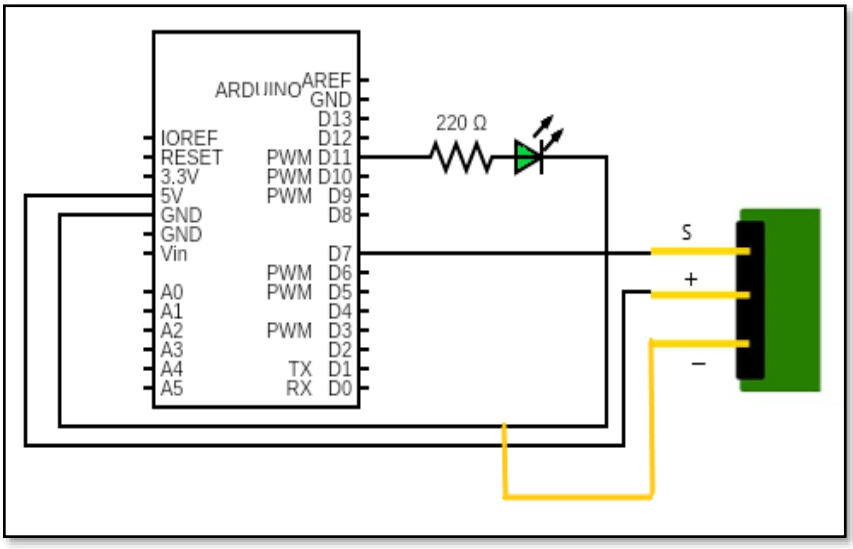
2.21 Télécommande à distance (WL-110)

En bref: Dans ce projet, nous apprenons comment allumer et éteindre une LED à l'aide d'une télécommande. Une télécommande IR appelée émetteur utilise la lumière pour transmettre les signaux de la télécommande à l'appareil qu'elle contrôle. Elle émet des impulsions de lumière infrarouge qui correspondent à des codes binaires spécifiques. Ces codes représentent des opérations telles que la mise en marche, la baisse du volume ou l'augmentation du canal. L'appareil contrôlé (également appelé récepteur) décode les impulsions de lumière infrarouge en code binaire que son microprocesseur interne comprend. Une fois le signal décodé, le microprocesseur exécute les commandes. Les télécommandes infrarouges nécessitent une ligne de vision, ce qui signifie que vous ne devez pas bloquer le chemin entre l'émetteur et le récepteur.

Les différents types de télécommandes ont des valeurs différentes pour les touches. Le tableau ci-dessous donne la valeur de code des touches de la télécommande que nous allons utiliser.

Composants : DEL, Résistance 220Ω, WL-100 (Télécommande)

Code valeur	Clés
FFA25D	1
FF629D	2
FFE21D	3
FF22DD	4
FF02FD	5
FFC23D	6
FFE01F	7
FFA857	8
FF906F	9
FF6897	*
FF9867	0
FFB04F	#
FF18E7	Haut
FF4AB5	Bas
FF10EF	Gauche
FF5AA5	Droit

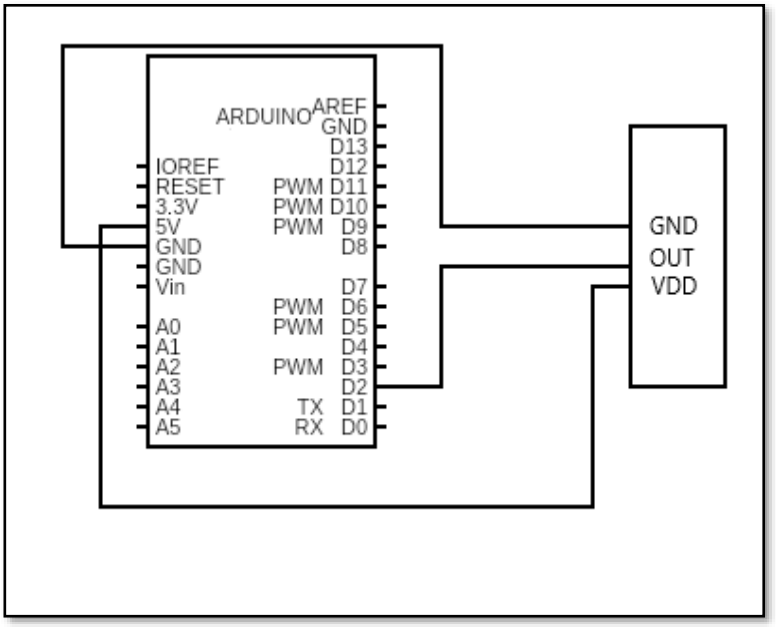



```
#include <IRremote.h>

const int RECV_PIN = 7;
IRrecv irrecv(RECV_PIN);  // créer un objet
decode_results results;
const int greenPin = 11;

void setup(){
 irrecv.enableIRIn();
 irrecv.blink13(true);
 pinMode(greenPin, OUTPUT);
}
void loop(){
   if (irrecv.decode(&results)){
      switch(results.value){
        case 0xFFA25D: //bouton clavier "1" allumez DEL
        digitalWrite(greenPin, HIGH);
        }
      switch(results.value){
        case 0xFF629D: // bouton clavier "2" éteignez DEL
        digitalWrite(greenPin, LOW);
        }
      irrecv.resume();
   }
}
```

2.22 Capteur PIR (SENS-PIR)

Bref: Un capteur infrarouge passif (capteur PIR) est un capteur électronique qui mesure la lumière infrarouge (IR) rayonnant des objets dans son champ de vision. Ils sont le plus souvent utilisés dans les détecteurs de mouvement basés sur le PIR. Les capteurs PIR sont couramment utilisés dans les alarmes de sécurité et les applications d'éclairage automatique. Une fois que le circuit est construit et téléchargé dans Arduino, essayez de déplacer des objets devant le capteur IRP, ce qui fait briller la LED, la sortie peut également être vue sur un moniteur en série.

```
int ledPin = 13;  // DEL montée
int PIR_pin = 2;
int PIR_state = LOW;
int state = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(PIR_pin, INPUT);
  Serial.begin(9600);
}
void loop(){
  state = digitalRead(PIR_pin);
  if (state == HIGH) {       //mouvement détecté
allumez DEL
    digitalWrite(ledPin, HIGH);
     Serial.println("Motion detected");
    }
  else {
    digitalWrite(ledPin, LOW); // éteignez DEL
     Serial.println("No Motion");
    }
}
```

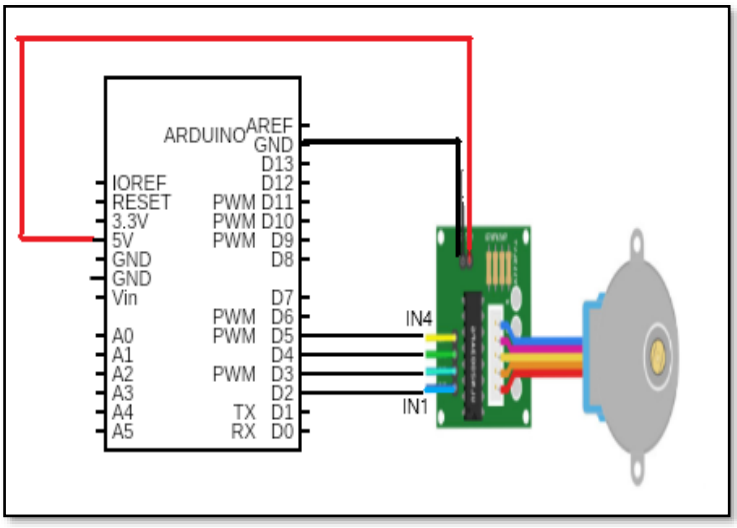


2.23 Stepper motor with driver (MOT-28BYJ-48)

Bref : Le moteur pas à pas 28BYJ-48 est un moteur pas à pas, qui convertit des impulsions électriques en une rotation mécanique discrète. Le moteur pas à pas 28BYJ-48 consomme beaucoup de courant et nous devrons donc utiliser un circuit de commande IC ULN2003 pour contrôler le moteur avec un microcontrôleur comme l'Arduino.

```
#include <Stepper.h> // inclure bibliothèque pas à
pas
const int stepsPerRevolution = 1048;  // nombre de
pas par rotation:
// Broche 2 à IN1 sur le pilote ULN2003
// Broche 3 à IN2 sur le pilote ULN2003
// Broche 4 à IN3 sur le pilote ULN2003
// Broche 5 à IN4 sur le pilote ULN2003
// Créer objet pas à pas nommé 'myStepper', notez
l'ordre des broches:
Stepper myStepper = Stepper(stepsPerRevolution, 2,
4, 3, 5);
void setup() {
  myStepper.setSpeed(10); // Fixez la vitesse à 10
rpm:
  Serial.begin(9600);
}
void loop() {
  Serial.println("clockwise");   // rotation sens horaire
  myStepper.step(stepsPerRevolution);
  delay(500);
  Serial.println("Anticlockwise"); // rotation sens
antihoraire  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

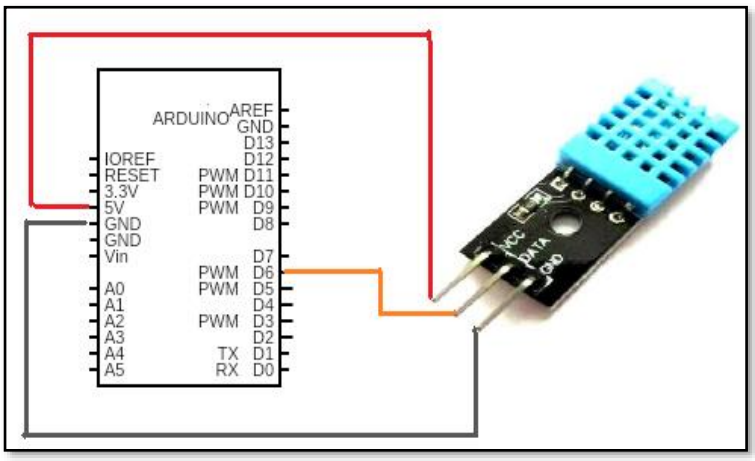
Components: MOT-28BYJ-48, Jumper wires



2.24 Capteur Humidité (SENS-DHT11-BB)

Bref: Dans ce projet, nous interfacons un capteur d'humidité pour mesurer la température et l'humidité.

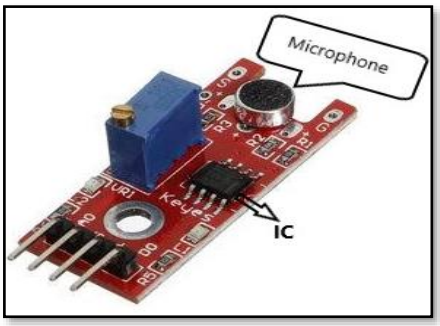
Le DHT11 se compose d'un composant de détection de l'humidité, d'un capteur de température NTC (ou thermistance) et d'un circuit intégré. Il possède un composant de détection d'humidité qui possède deux électrodes avec un substrat de maintien de l'humidité entre elles. Lorsque l'humidité change, la conductivité du substrat change ou la résistance entre ces électrodes change. Ce changement de résistance est mesuré et traité par le circuit intégré, ce qui fait que les données sont lues par un microcontrôleur.



Composants : Capteur d'humidité (SENS-DHT11-BB), fils de pontage/connexion

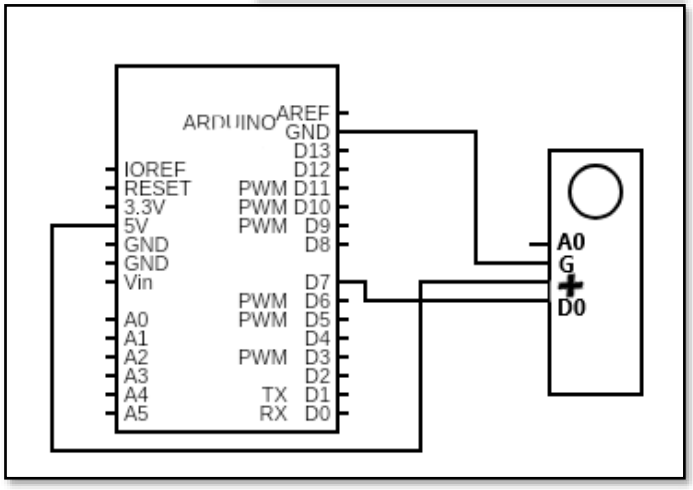
2.25 Détecteur de Son

Bref: Ici, nous utilisons SENS-42 pour détecter le son des applaudissements. Lorsqu'un son est détecté, un message s'affiche sur le moniteur en série. Le microphone du module de détection du son détecte le son. Ce son est transmis au circuit intégré LM393. Ce circuit intégré est un comparateur de tension qui compare la tension de seuil avec la sortie du microphone.



Composants : SENS-42, fils de pontage/raccordement

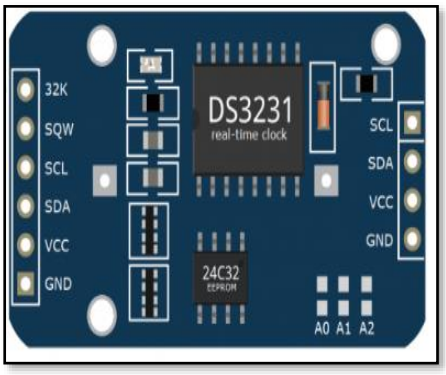
```
#define sensor_Pin 7

void setup() {
  pinMode(sensor_Pin, INPUT);  // Fixez
broche capteur comme ENTREE
  Serial.begin(9600);
}

void loop() {
  // Lire capteur de son
  int sensorData =
digitalRead(sensor_Pin);
  // Isi la broche se rend à BAS, son est
détecté
  if (sensorData == LOW) {

     Serial.println("Clap detected!");
   }
```



2.26 Module RTC (ARD-DS-3231)

Bref: Nous utilisons un module RTC pour afficher l'heure, la date et le jour actuels.

Composants: I2C LCD, ARD-DS3231



32K - sortie de la puce DS3231 un oscillateur très précis à 32KHz

SQW - émet un signal en ondes carrées à partir de la puce DS3231. La fréquence de l'onde carrée peut être modifiée entre 1 Hz, 4 kHz, 8 kHz ou 32 kHz par programmation. Cette broche peut également être utilisée comme sortie d'interruption.

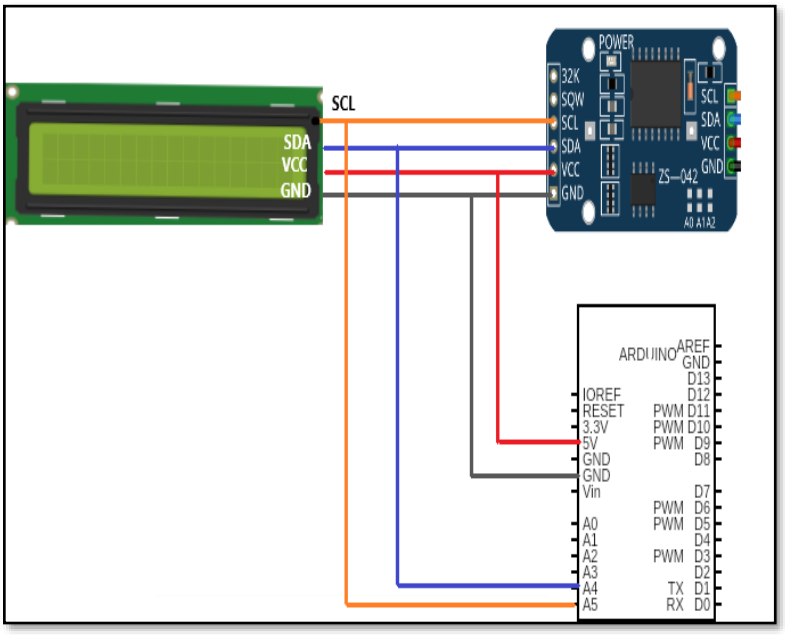
SCL - broche d'entrée pour l'horloge série I2C

SDA - broche d'entrée/sortie pour les données série I2C

VCC - broche d'entrée de la source d'alimentation pour le 32K - outputs from the DS3231 chip a very accurate 32KHz module; can be any voltage from +3.3V to +5.5V DC

GND - Connexion de la broche de terre

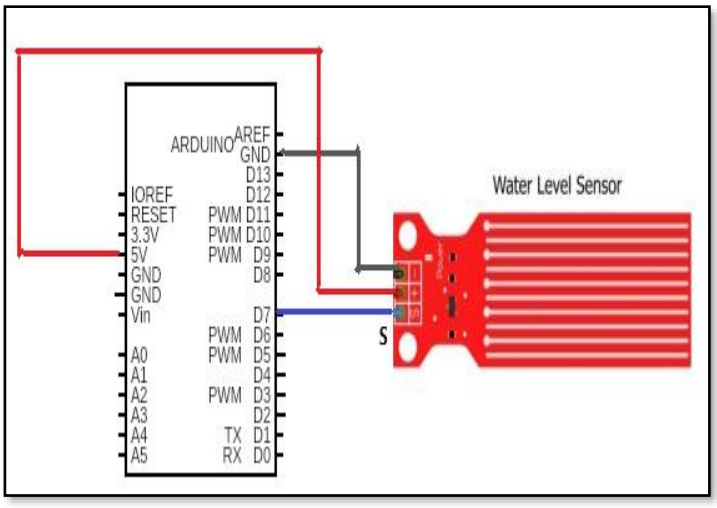
```
#include "RTClib.h"
#include <LiquidCrystal_I2C.h> // installez cette bibliothèque
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C adresse 0x27, 16 colonnes et 2 rangées
DateTime now;
RTC_DS3231 rtc;
//enum pour daysOfTheWeek (jours de la semaine)
char daysOfTheWeek[7][12] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
void setup ()
{
 //Serial.begin(9600);
 lcd.init();// initialisez le lcd
 lcd.backlight();// tpour allumer le commutateur sur le backlight du LCD
}
void loop () {
   DateTime now = rtc.now();
 lcd.setCursor(0,0);
 lcd.print(now.day());
 lcd.print('/');
 lcd.print(now.month());
 lcd.print('/');
 lcd.print(now.year());
 lcd.setCursor(11,0); // fixez première rangée et 11ème colonne
lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);
 lcd.setCursor(1,1);
 lcd.print("Time:");
 lcd.print(now.twelveHour());
 lcd.print(':');
 lcd.print(now.minute());
 lcd.print(':');
 lcd.print(now.second());
```



2.27 Capteur de Niveau d'eau (SENS-78)

Bref: Le travail du capteur de niveau d'eau est basé sur la résistance variable de la série de conducteurs parallèles exposés (tout comme un potentiomètre) dont la résistance change en fonction du niveau d'eau. La résistance est inversement proportionnelle à la hauteur de l'eau.

Composants : SENS-78, fils de pontage/ raccordement.




```
#define sensor_switch 7

#define sensor_input A0
// Valeur pour Entreposage du niveau d'eau
int depth = 0;

void setup() {
   Serial.begin(9600);
//fixez la broche 7 comme l'interrupteur
 pinMode(sensor_switch, OUTPUT);
//interrupteur off capteur
 digitalWrite(sensor_switch, LOW);
}
void loop() {
 int water_level = Sensor_level();
 Serial.print("Water water_level is ");
 Serial.println(water_level);
 delay(100);
}
int Sensor_level() {
 digitalWrite(sensor_switch, HIGH);  // Allumez le capteur ON
 delay(10);
 depth = analogRead(sensor_input);    // Lire la valeur analogue du capteur
 digitalWrite(sensor_switch, LOW);   // Eteindre le capteur OFF
 return depth;          // transmettre profondeur du niveau d'eau
}
```