

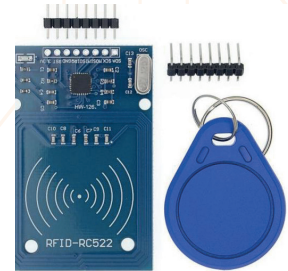
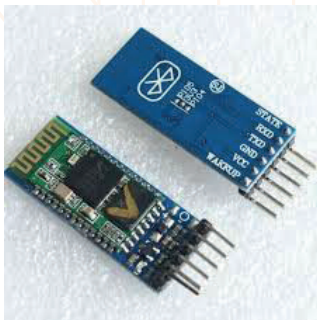
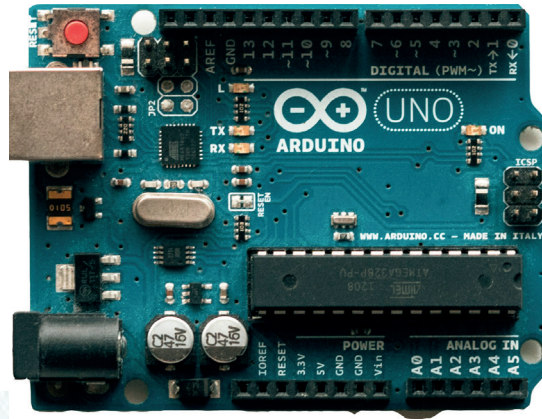
YoupiLab

DEMYSTIFYING ELECTRONICS

ARDUINO POUR LES NULS



1^{ère}
ÉDITION



Sommaire

Introduction.....	3
Où nous contacter ?	4
Matériel nécessaire.....	5
Qu'est-ce qu'il vous faut pour démarrer ?	5
Présentation de l'Arduino	7
Le logiciel Arduino IDE	11
Petit rappel sur l'électronique	13
Connecter votre Arduino à votre IDE	13
Projet 1 : Feux de circulation.....	15
Projet 2: Luminosité des LED sur un écran LCD 16x2.....	17
Projet 3 : Guide complet pour les ultrasons Capteur HC-SR04 avec Arduino	20
Projet 4: Capteur de stationnement.....	23
Projet 5: Arduino avec capteur de mouvement PIR.....	25
Projet 6: Contrôlez les LEDs avec la télécommande IR.....	27
Projet 7: Teensy/Arduino - Jeu de mémoire	32
Projet 8: : Guide pour le capteur de gaz/fumée MQ-2 avec Arduino	34
Projet 9 : Allumage d'une LED avec une photorésistance	37
Projet 10: Contrôle de l'intensité la luminosité d'une LED.....	39
Projet 11: Bargraphe manuelle	41
Projet 12: Accès sécurisé à l'aide de la RFID MFRC522 avec Arduino.....	43
Projet 13: Enregistreur de données de température avec Arduino et module de carte SD48	
Projet 14 : RGB+MODULE BLUETOOTH.....	51
Projet 15 : Contrôle de moteur à courant continu via Bluetooth.....	57
RESSOURCES.....	61

Introduction

Cet ebook compile certains de nos projets Arduino les plus populaires. Pour découvrir plus de projets Arduino, consultez notre référentiel de projets à l'adresse suivante : [<https://education.youpilab.com/courses/free>]

Nous vous encourageons à regarder certaines de nos démonstrations vidéo sur ce lien <https://youtube.com/@youpilab?si=qyrkCfEXUNIAdeU9>. Certains projets sont plus faciles à comprendre en voyant le circuit en action.

L'objectif de cet ebook est de vous inspirer à créer quelque chose d'incroyable avec l'électronique et la programmation. Après avoir réalisé quelque chose de remarquable, nous espérons que vous le partagerez avec la communauté. C'est tout l'esprit de notre communauté formidable.

Où nous contacter ?

Vous pouvez nous contacter pour toute question ou suggestion via les moyens suivants :

- Email: sales@youpilab.com

- Téléphone: [+22961041622](tel:+22961041622)

- Adresse postale : Godomey Togoudo, République du Benin

- Site web: <https://youpilab.com/>

➤ Réseaux sociaux

- Facebook: <https://www.facebook.com/youpilab>

- YouTube: <https://youtube.com/@youpilab?si=qyrkCfEXUNIAdeU9>

- LinkedIn: <https://www.linkedin.com/in/youpilab-youpilab-213984185/>

-GitHub:

Nous sommes impatients de vous entendre et de vous aider dans vos projets Arduino !

Matériel nécessaire

Pour réaliser des projets Arduino, il vous faut des composants électroniques en plus de la carte Arduino. Pour chaque projet, nous fournirons une liste complète des pièces nécessaires ainsi que des liens vers [notre boutique en ligne](#). Vous y trouverez les composants requis, accompagnés de descriptions détaillées, au meilleur prix.



En achetant vos pièces via nos liens d'affiliation, vous soutenez notre travail. Si vous cherchez un composant ou un outil, nous vous recommandons de consulter nos suggestions.

Qu'est-ce qu'il vous faut pour démarrer ?

Selon nous, la manière la plus efficace de débiter avec Arduino est d'acquérir un kit de démarrage Arduino. Ce kit contient tous les composants nécessaires pour apprendre les bases et commencer à réaliser des projets.



[kit de démarrage complet pour Arduino](#)

Nous vous recommandons également de vous équiper d'outils supplémentaires tels qu'un multimètre et un fer à souder disponible également sur notre plateforme.



[multimètre](#)



[fer à souder](#)

Nous disposons de plusieurs articles destinés à vous guider dans le choix optimal d'un multimètre et d'un fer à souder pour débutants :

- ❖ [Meilleurs multimètres sur YoupiLab](#)
- ❖ [Fer à souder sur YoupiLab](#)

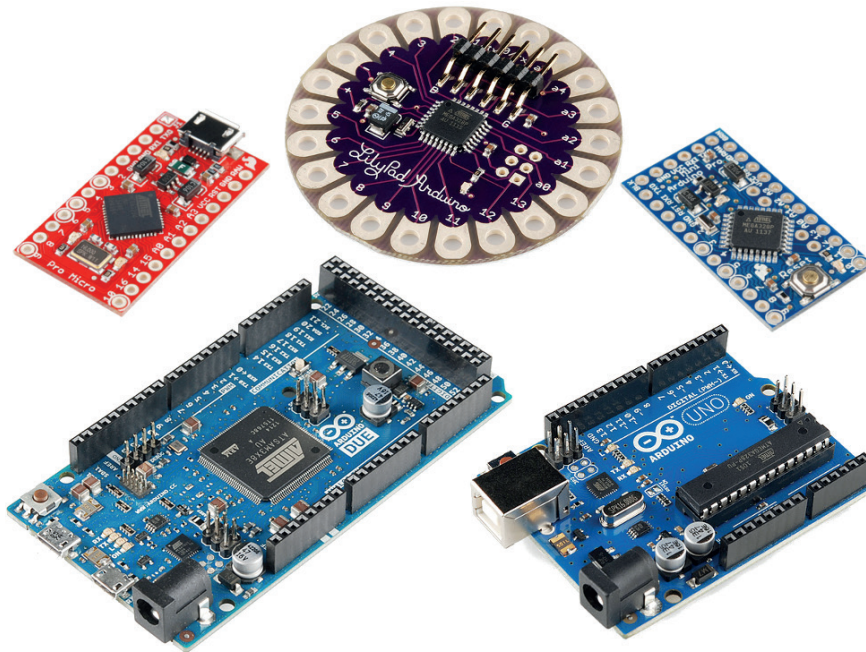
Présentation de l'Arduino

L'Arduino est un microcontrôleur programmable qui vous permet de capturer des données de votre environnement et de contrôler des dispositifs externes. Cette flexibilité est rendue possible grâce à la connexion de divers appareils et composants à l'Arduino, vous permettant de réaliser une large gamme d'applications selon vos besoins.

Vous pouvez accomplir des projets remarquables avec l'Arduino ; les possibilités sont infinies et en laissant libre cours à votre imagination, tout devient réalisable !

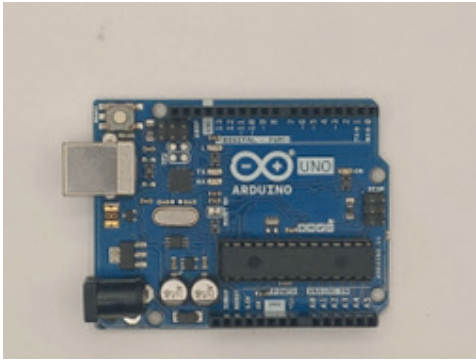
➤ Qu'est-ce qu'un Arduino ?

L'Arduino est l'une des cartes illustrées dans la figure ci-dessous.

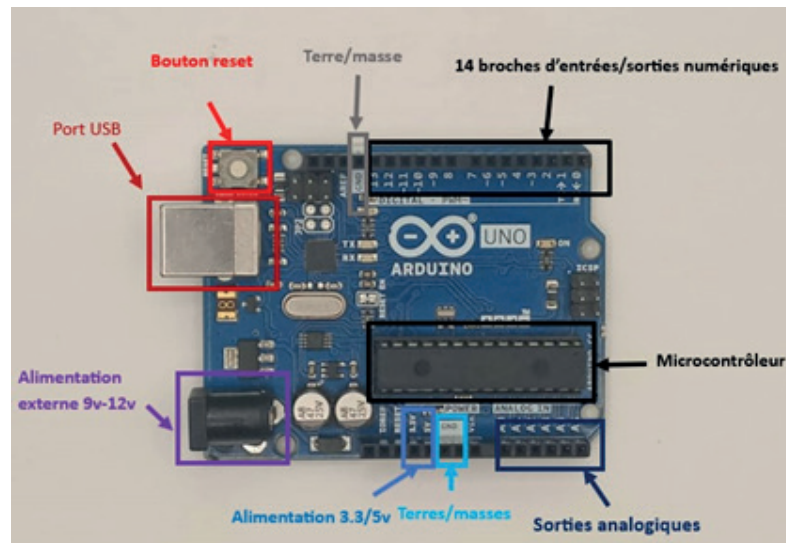


Arduino fait partie de la famille des platines de développement. Une platine de développement est en général un circuit imprimé équipé d'un microprocesseur ou d'un microcontrôleur. Comme Arduino est open source, il existe un grand nombre de clones et de platines compatibles, tout comme il existe de nombreux modèles d'Arduino officiels, avec des fonctions particulières.

Pour ce cours, nous nous baserons sur le plus connu : l'Arduino Uno. Mais un autre modèle ou un clone fera aussi bien l'affaire.

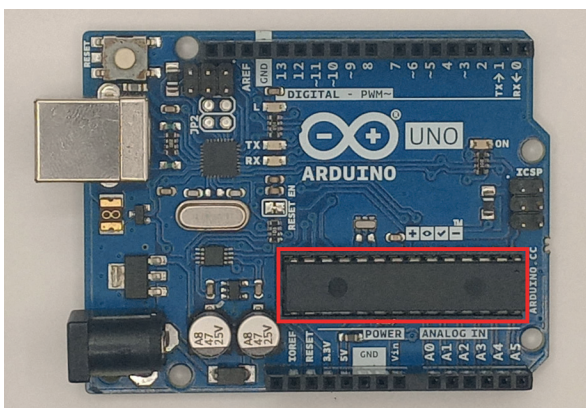


➤ Schéma d'une platine Arduino Uno

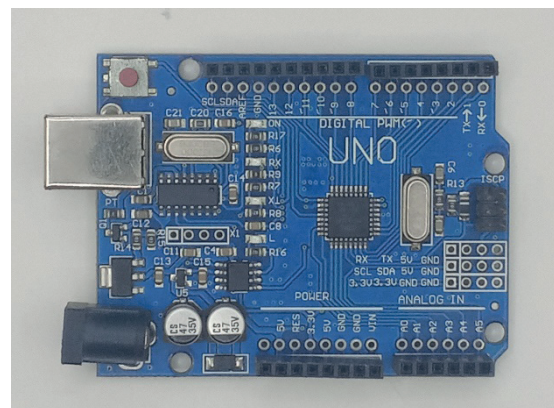


➤ Le microcontrôleur

C'est le cerveau de notre carte. Il va recevoir le programme que nous allons créer et va le stocker dans sa mémoire avant de l'exécuter. Grâce à ce programme, il va savoir faire des choses, qui peuvent être : faire clignoter une LED, afficher des caractères sur un écran, envoyer des données à un ordinateur, mettre en route ou arrêter un moteur... Il existe deux modèles d'Arduino UNO : l'un avec un microcontrôleur de grande taille R3, et un autre avec un microcontrôleur dit SMD (SMD: Surface Mounted Device, soit composants montés en surface, en opposition aux composants qui traversent la carte électronique et qui sont soudés du côté opposé). D'un point de vue utilisation, il n'y a pas de différence entre les deux types de microcontrôleurs. Les couleurs de l'Arduino peuvent varier du bleu au bleu-vert, en fonction des modèles et années de production.



Arduino UNO R3



Arduino UNO SMD

➤ Broches et connecteurs de l'Arduino UNO

Broches Numériques :

- **Description** : L'Arduino UNO dispose de 14 broches numériques étiquetées de 0 à 13. Elles peuvent être configurées comme entrées ou sorties.
 - **Entrées** : Lorsqu'elles sont définies comme entrées, ces broches peuvent lire la tension et distinguer deux états : HAUT ou BAS.
 - **Sorties** : Lorsqu'elles sont définies comme sorties, ces broches peuvent appliquer une tension de 5V (ÉLEVÉ) ou 0V (BAS).

Broches PWM :

- **Description** : Certaines broches numériques (11, 10, 9, 6, 5 et 3) sont marquées d'un ~ et supportent la modulation de largeur d'impulsion (PWM). Le PWM permet aux broches numériques de produire des tensions variables simulées. Vous en apprendrez plus sur le PWM plus tard.

Broches TX et RX :

- **Description** : Les broches numériques 0 et 1 sont utilisées pour la communication série.
 - **TX (Transmettre)** : Utilisée pour envoyer des données.
 - **RX (Recevoir)** : Utilisée pour recevoir des données.
 - **Usage** : L'Arduino utilise ces broches pour communiquer avec d'autres appareils électroniques ainsi que lors du téléchargement de nouveau code depuis l'ordinateur. Évitez d'utiliser ces broches pour d'autres tâches que la communication série, sauf si nécessaire.

LED Attachée à la Broche Numérique 13 :

- **Description** : Une LED est connectée à la broche numérique 13, utile pour un débogage facile des croquis Arduino.

LED TX et RX :

- **Description** : Ces LED clignent lorsque des informations sont échangées entre l'ordinateur et l'Arduino.

Broches Analogiques :

- **Description** : Les broches analogiques, étiquetées de A0 à A5, sont souvent utilisées pour lire des capteurs analogiques. Elles peuvent lire des valeurs de tension entre 0 et 5V. Ces broches peuvent également être utilisées comme entrées/sorties numériques, tout comme les broches numériques.

Broches d'Alimentation :

- **Description** : L'Arduino fournit 3,3V ou 5V via ces broches, nécessaires pour la plupart des composants. Les broches étiquetées «GND» sont les broches de terre.

Bouton de Réinitialisation :

- **Description** : Lorsque ce bouton est pressé, le programme en cours d'exécution sur l'Arduino redémarre. Une broche de réinitialisation est également disponible à côté des broches d'alimentation, permettant de réinitialiser l'Arduino en appliquant une petite tension.

LED d'Alimentation :

- **Description** : Cette LED s'allume dès que l'Arduino est sous tension.

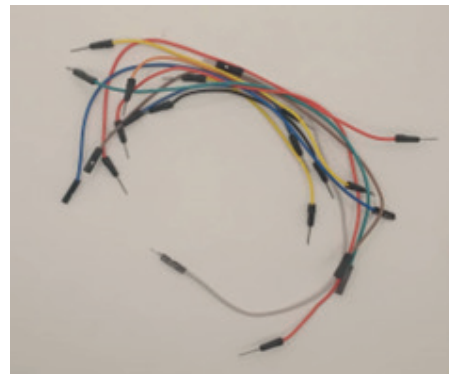
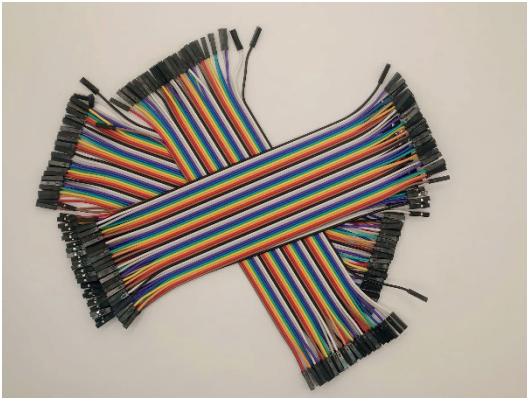
Prise USB :

- **Description** : Utilisez un câble USB A mâle vers USB B mâle pour télécharger des programmes depuis votre ordinateur vers votre carte Arduino. Ce câble alimente également votre Arduino.



- Pour fonctionner, une carte Arduino a besoin d'une alimentation. Le microcontrôleur fonctionnant sous 5V, la carte peut être alimentée en 5V par le port USB ou bien par une alimentation externe qui est comprise entre 7V et 12V. Un régulateur se charge ensuite de réduire la tension à 5V pour le bon fonctionnement de la carte.

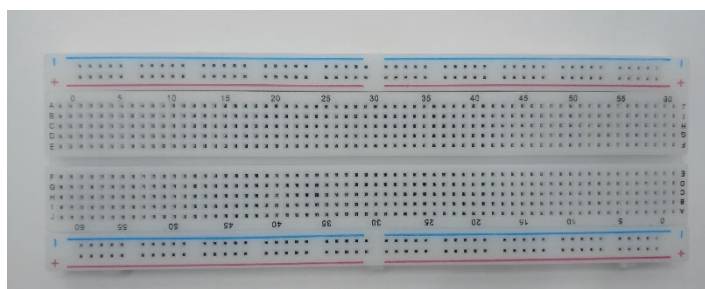
Les connexions entre les composants sont réalisées par des jumpers, sortes de petits câbles.



Jumpers

➤ La platine d'expérimentation

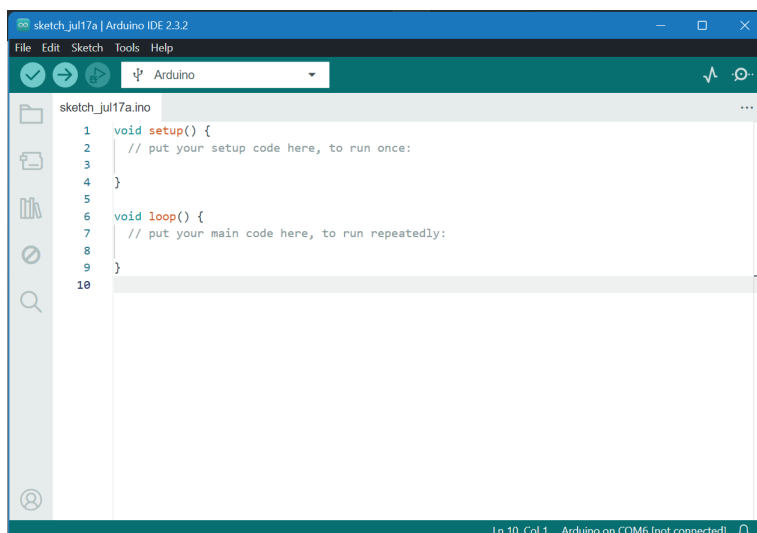
Une platine d'expérimentation (appelée breadboard) permet de réaliser des prototypes de montages électroniques sans soudure et donc de pouvoir réutiliser les composants.



Le logiciel Arduino IDE

Le logiciel Arduino IDE fonctionne sur Mac, Windows et Linux. C'est grâce à ce logiciel que nous allons créer, tester et envoyer les programmes sur l'Arduino.

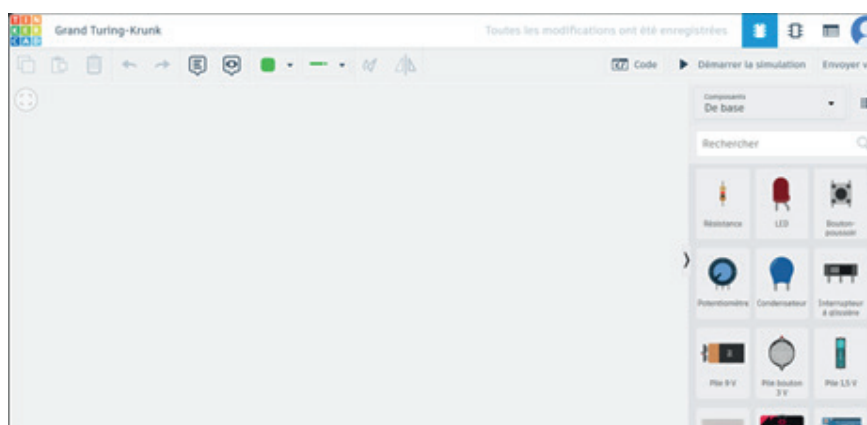
L'IDE est téléchargeable à l'adresse suivante : <https://www.arduino.cc/en/software>



À propos des schémas électroniques

Pour réaliser les schémas électroniques, nous utiliserons l'outil **Fritzing**, disponible gratuitement sur Mac et PC.

Un simulateur Arduino est aussi disponible à cette adresse : [tinkercad](https://www.tinkercad.com). Il permet la réalisation de schémas d'implantation des composants sur les platines d'expérimentation, mais aussi la réalisation de schémas et de circuits électroniques.



Contrôler une sortie et lire une entrée

Les broches d'un Arduino peuvent être classées en trois types principaux en fonction de leurs fonctionnalités :

Broches Numériques :

- **Fonction** : Ces broches peuvent être utilisées pour lire des signaux numériques (0 ou 1) ou pour envoyer des signaux numériques (HIGH ou LOW).
- **Caractéristiques** : Chaque broche numérique peut être configurée en tant que sortie (pour

envoyer des signaux) ou en tant qu'entrée (pour lire des signaux). Elles sont souvent utilisées pour le contrôle d'appareils simples comme des LED, des relais, ou pour communiquer avec d'autres dispositifs numériques.

Broches Analogiques :

- **Fonction :** Ces broches peuvent lire des signaux analogiques issus de capteurs, qui varient sur une échelle continue plutôt que d'être simplement ON ou OFF.
- **Caractéristiques :** Les broches analogiques peuvent convertir des tensions analogiques en valeurs numériques, permettant ainsi à l'Arduino de mesurer des grandeurs comme la lumière, la température, ou d'autres phénomènes physiques mesurables par des capteurs analogiques.

Broches PWM (Modulation de Largeur d'Impulsion) :

- **Fonction :** Ces broches permettent de générer des signaux PWM, où la largeur des impulsions est modulée pour contrôler la puissance fournie à des dispositifs tels que des moteurs, des servomoteurs ou des LED.
- **Caractéristiques :** Bien que les broches PWM émettent des signaux numériques, elles peuvent simuler des signaux analogiques en ajustant le rapport cyclique (rapport entre la durée d'un état actif et la période totale) des impulsions. Cela permet un contrôle précis de la vitesse, de la luminosité ou de toute autre propriété ajustable.

*Pour contrôler une sortie numérique, vous utilisez la fonction `digitalWrite()` et entre parenthèses vous écrivez la broche que vous souhaitez contrôler, puis HIGH ou LOW.

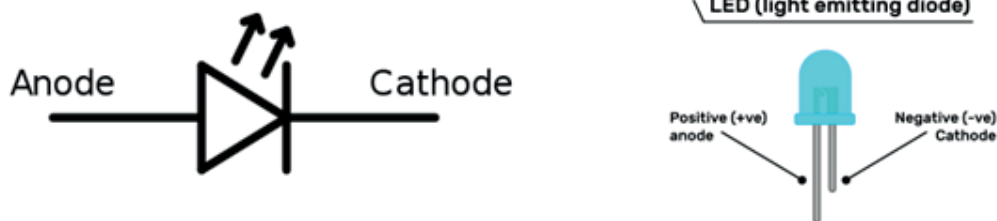
*Pour contrôler une broche PWM, vous utilisez la fonction `analogWrite()` et entre parenthèses vous écrivez la broche que vous souhaitez contrôler et un nombre compris entre 0 et 255.

*Pour lire une entrée analogique vous utilisez la fonction `analogRead()` et pour une entrée numérique vous utilisez `digitalRead()`.

Petit rappel sur l'électronique

➤ Les diodes

Il est possible de remplacer l'ampoule par une diode électroluminescente, aussi appelée LED. Elle a la particularité de ne laisser passer le courant électrique que dans un sens.



Le courant électrique ne peut traverser la diode que dans le sens de l'anode vers la cathode. On reconnaît l'anode, car il s'agit de la broche la plus longue. Lorsque les deux broches sont de même longueur, on peut distinguer l'anode de la cathode, par un méplat du côté de cette dernière.

Attention : Le courant produit par l'Arduino est trop important pour y brancher directement une LED dessus. L'utilisation d'une résistance est obligatoire, pour ne pas griller la LED.

➤ Les résistances

Une résistance est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (mesurée en ohms : Ω) à la circulation du courant électrique.

Pour en apprendre plus, consultez le blog disponible sur la plateforme YoupiLab à travers cette adresse : [Resistance_Démystifiées](#)



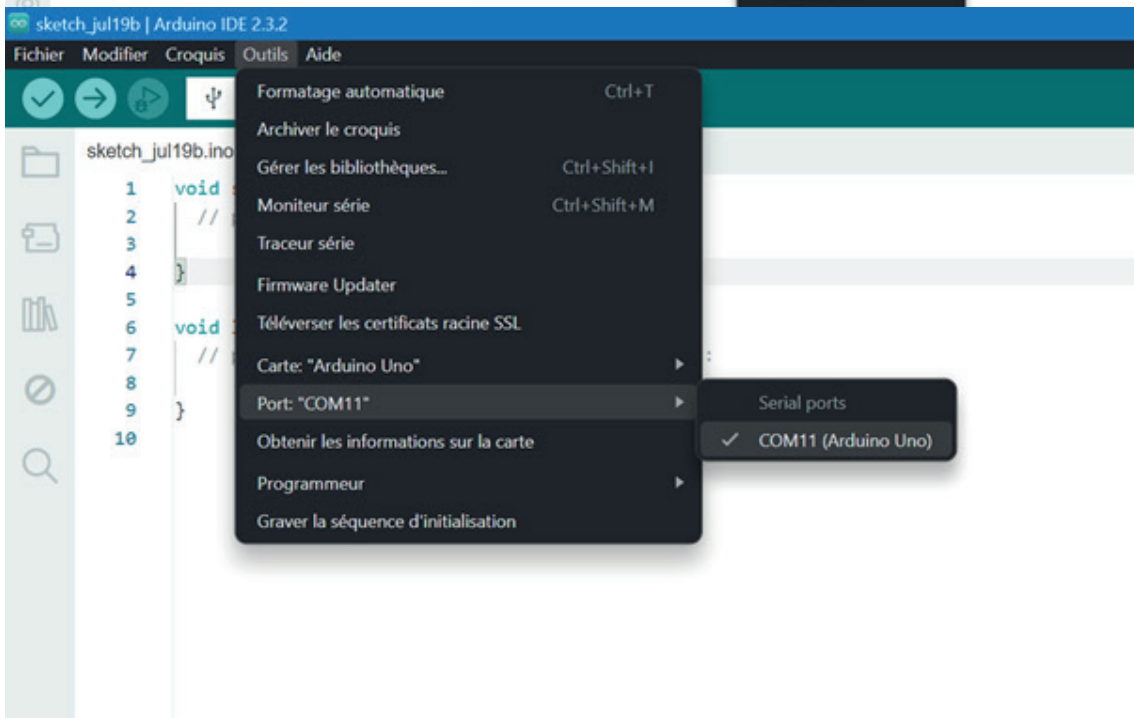
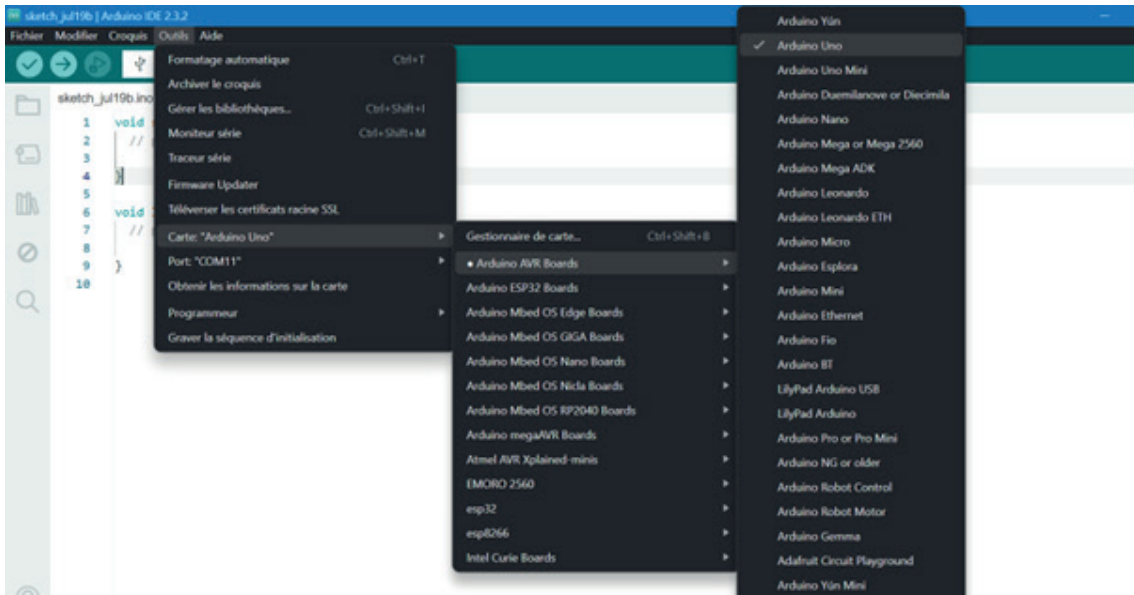
Connecter votre Arduino à votre IDE

Connectez votre Arduino UNO à votre ordinateur via USB.

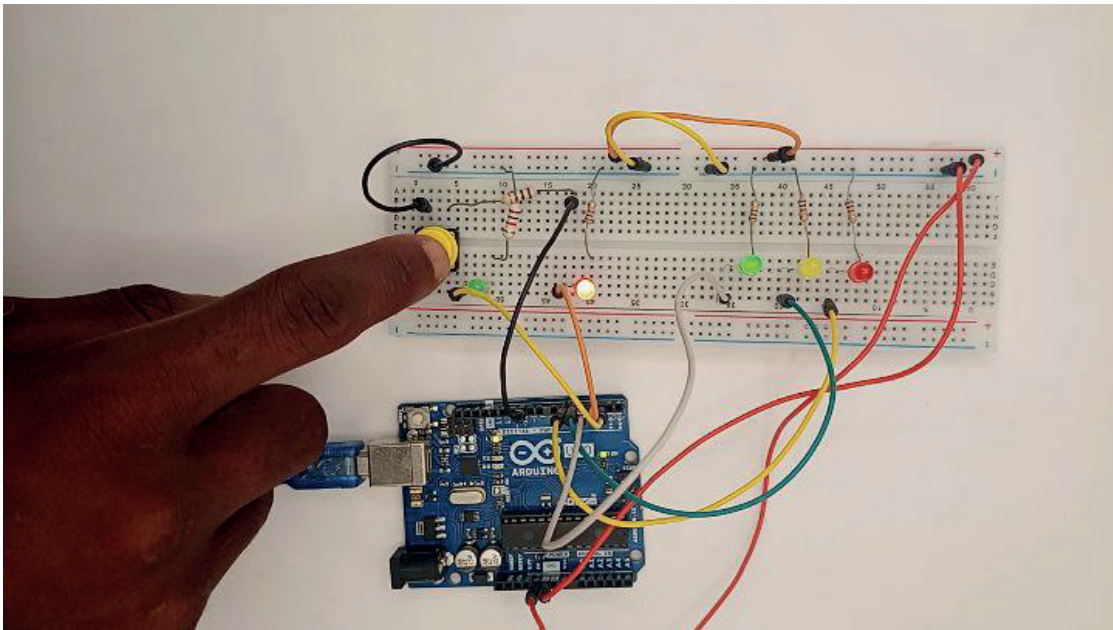
Après avoir connecté votre Arduino avec un câble USB, vous devez vous assurer que le Arduino IDE a sélectionné le port et la bonne carte.

Dans notre cas, nous utilisons Arduino Uno, nous devrions donc aller dans Outils Carte ;

Arduino/ Uno puis Outils Port /COM*



Projet 1 : Feux de circulation



Description

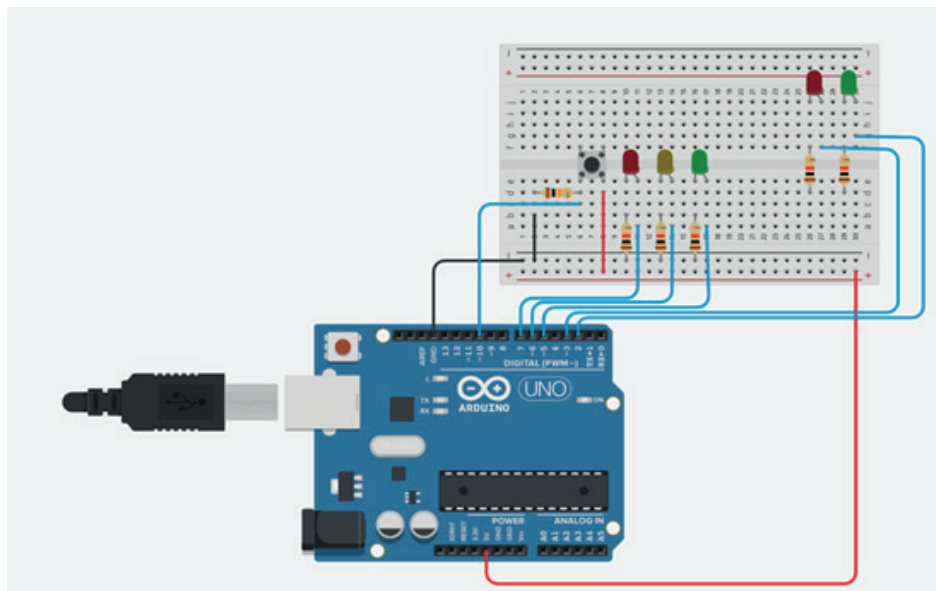
Dans ce projet, vous allez construire un système de feux de circulation.

Vous utiliserez trois LED de différentes couleurs (vert, jaune et rouge) pour simuler les feux de circulation pour voitures. De plus, deux LEDs de couleurs différentes (vert et rouge) représenteront le trafic piéton. Un bouton poussoir sera utilisé pour imiter ceux des feux de circulation pour piétons.

Liste des composants

- ❖ 1 Arduino UNO
- ❖ 5 LEDs
- ❖ 6 Résistances (5x1k Ω et 1x10k Ω)
- ❖ 1 Bouton Poussoirs
- ❖ Jumpers

Circuit



Code:

Le code en entier ici [🔗](#) :

Vous n'avez besoin d'aucune bibliothèque pour ce code.

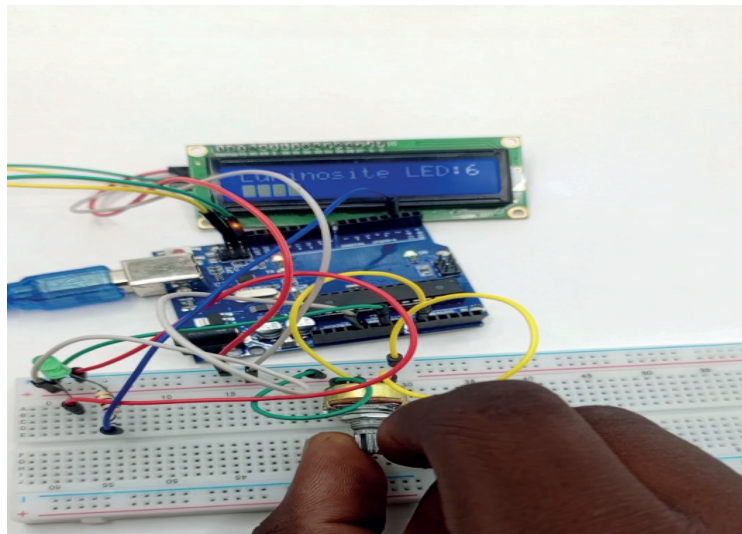
Voici quelques conseils pour mieux comprendre le fonctionnement des feux de signalisation :

- Le feu de la voiture reste toujours vert, et le feu des piétons est toujours rouge, sauf dans les cas suivants :
- Lorsque quelqu'un appuie sur le bouton, voici ce qui se passe :
 - * Le feu de la voiture passe au jaune, puis au rouge.
 - * Le feu des piétons passe au vert.



```
File Edit Sketch Tools Help
sketch_aug7a.ino
1 int redCar = 6; // Broche pour le feu rouge des voitures
2 int yellowCar = 5; // Broche pour le feu jaune des voitures
3 int greenCar = 4; // Broche pour le feu vert des voitures
4 int greenPed = 2; // Broche pour le feu vert des piétons
5 int redPed = 3; // Broche pour le feu rouge des piétons
6 int button = 10; // Broche pour le bouton poussoir
7 int crossTime = 2000; // Temps de traversée (en millisecondes)
8 unsigned long changeTime; // Temps du dernier changement de feux
9
```


Projet 2: Luminosité des LED sur un écran LCD 16x2



Description

Il s'agit d'un projet pour débutant dans lequel vous utiliserez un écran LCD 16 × 2 pour afficher la luminosité des LED.

Dans peu de temps, dans ce projet, nous contrôlerons la luminosité d'une LED à l'aide d'un potentiomètre. La luminosité de la LED sera affichée sur l'écran LCD à l'aide d'une barre de progression

Présentation de l'Écran LCD 16x2 avec Interface I2C



LCD I2C

Caractéristiques Générales :

- **Type d'affichage** : Écran LCD 16x2 (16 caractères par ligne, 2 lignes)
- **Technologie** : Cristal liquide (LCD)
- **Interface de communication** : I2C (Inter-Integrated Circuit)
- **Tension de fonctionnement** : 5V DC
- **Rétroéclairage** : LED (souvent bleu avec caractères blancs)

- **Contrôle de contraste** : Potentiomètre ajustable intégré
- **Consommation électrique** : Faible consommation, typiquement moins de 20mA sans rétroéclairage
- **Dimensions** : Environ 80mm x 36mm x 12mm

Avantages de l'Interface I2C :

- **Simplicité de câblage** : Seules 4 connexions nécessaires (VCC, GND, SDA, SCL)
- **Adresse unique** : Permet la connexion de multiples dispositifs sur le même bus I2C avec des adresses distinctes
- **Compatibilité** : Large support par les microcontrôleurs tels que Arduino, Raspberry Pi, ESP8266, et autres
- **Efficacité** : Réduit l'encombrement des broches sur le microcontrôleur comparé à une connexion parallèle

Utilisations Typiques :

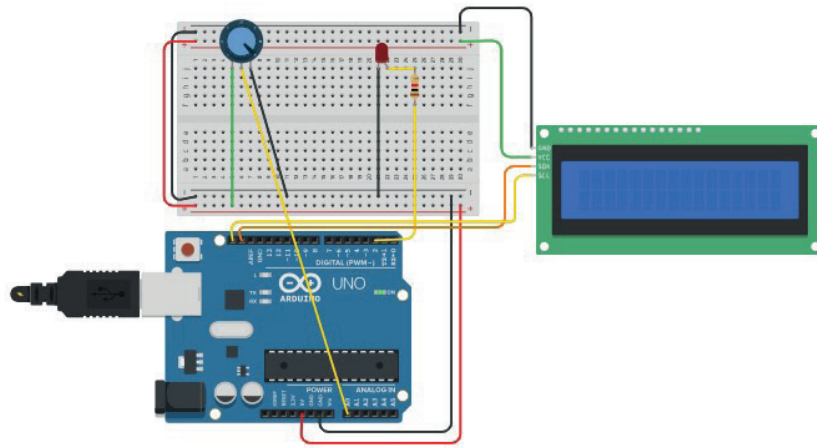
- **Projets de bricolage et de prototypage** : Affichage de données de capteurs, de messages, de menus dans des projets DIY
- **Applications éducatives** : Idéal pour l'apprentissage de la programmation et des interfaces de communication
- **Systèmes embarqués** : Utilisé dans des appareils intégrés pour afficher des informations utilisateur
- **Projets domotiques** : Affichage des informations dans des systèmes de maison intelligente

Cet écran LCD avec interface I2C est un composant polyvalent et pratique pour de nombreux projets électroniques, offrant une solution d'affichage claire et facile à implémenter.

Liste des composants

- ❖ 1 Arduino UNO
- ❖ 1 Led
- ❖ 1 Résistances 1kΩ
- ❖ 1 Ecran LCD 16X2 (**LCD I2C**)
- ❖ 1x Potentiomètre 100 k
- ❖ Jumpers

Circuit



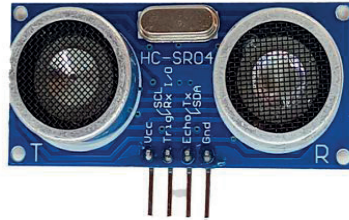
Code :

Le code en entier ici [👉](#) :

Le code est bien commenté afin que vous puissiez facilement comprendre son fonctionnement et le modifier pour l'inclure dans vos propres projets.

```
sketch_jul22a | Arduino IDE 2.3.2
Fichier Modifier Croquis Outils Aide
Arduino Uno
sketch_jul22a.ino
1 #include <LiquidCrystal_I2C.h> // Inclut la bibliothèque LiquidCrystal_I2C pour contrôler un écran LCD via I2C
2
3 // Initialise l'objet LCD avec l'adresse I2C 0x27 et un écran de 16 colonnes par 2 lignes
4 LiquidCrystal_I2C lcd(0x27,16,2);
5
6 // Définit les constantes pour la broche du potentiomètre et la broche de la LED
7 #define potVal A0 // Broche analogique où est connecté le potentiomètre
8 #define led 2 // Broche numérique où est connectée la LED
9
10 // Définir le caractère spécial pour une barre de progression
11 byte Bar[] = {
12   B11111, // Ligne pleine
13   B11111, // Ligne pleine
14   B11111, // Ligne pleine
15   B11111, // Ligne pleine
16   B11111, // Ligne pleine
17   B11111, // Ligne pleine
18   B11111, // Ligne pleine
19   B11111 // Ligne pleine
20 };
21
22 void setup() {
23   lcd.init(); // Initialise l'écran LCD
24   lcd.clear(); // Efface l'écran LCD
25   lcd.backlight(); // Active le rétroéclairage de l'écran LCD
--
```

Projet 3 : Guide complet pour les ultrasons Capteur HC-SR04 avec Arduino



Capteur à ultrasons HC-SR04

Ce projet concerne le capteur à ultrasons HC - SR04. Nous expliquerons son fonctionnement, montrerons quelques fonctionnalités et partagerons un exemple de projet Arduino.

Description

Le capteur à ultrasons HC-SR04 utilise un sonar pour déterminer la distance à un objet comme la police les chauves-souris ou les dauphins. Il offre une excellente détection de plage sans contact avec une précision élevée et des conférences d'écuries dans un boîtier facile à utiliser. De 2 cm à 400 cm ou 1 "à 13 pieds. Son fonctionnement n'est pas traité par la lumière du soleil ou un matériau noir comme les télémètres Sharp (bien que des matériaux acoustiquement doux comme le tissu qui soit difficile à détecter). Il vient complet avec module émetteur et récepteur à ultrason s.

Détails techniques :

- Alimentation : + 5V DC
- Courant de repos : <2mA
- Courant de travail : 15 mA
- Angle effectif : <15°
- Distance de portée : 2 cm - 400 cm / 1» - 13 pieds
- Résolution : 0,3 cm
- Angle de mesure : 30 degrés
- Largeur d'impulsion d'entrée de déclenchement : 10uS
- Dimensions : 45 mm x 20 mm x 15 mm
- Poids: 8g

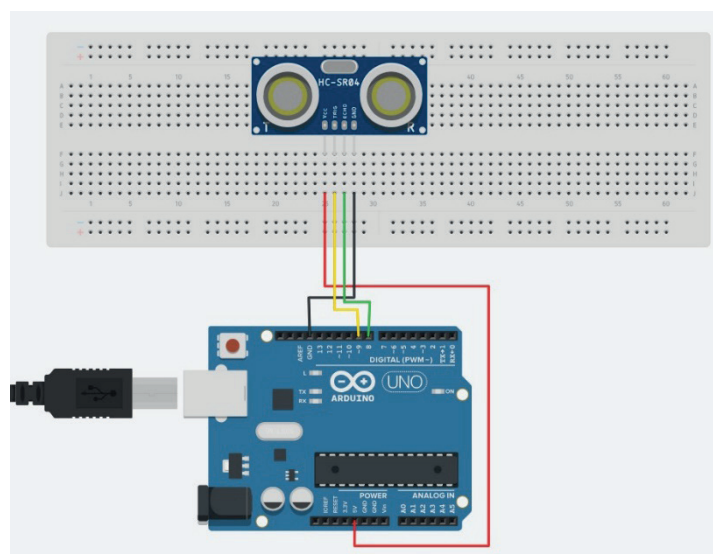
Comment faire le branchement HC-SR04 à l'Arduino



- VCC : 5V VCC
- GND : GND
- Trig : Entrée (9 de l'Arduino)
- Echo : Sortie (8 de l'Arduino)

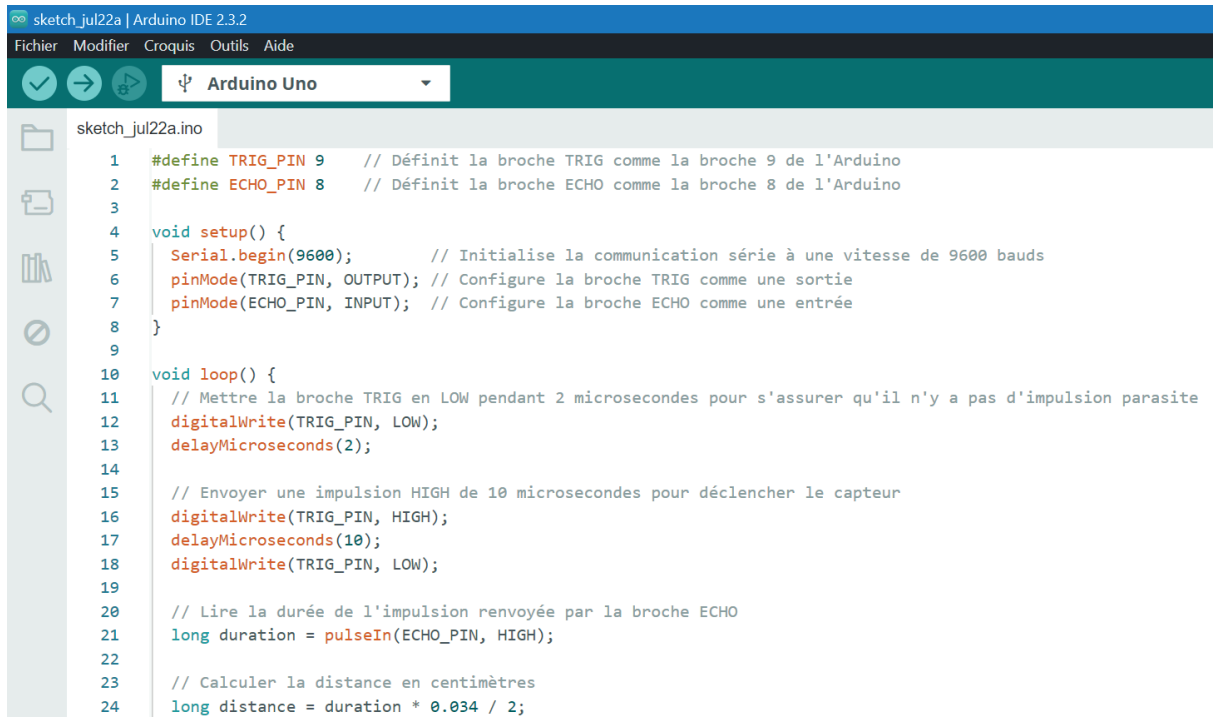
Le schéma de câblage est illustré dans la figure suivante. Notez que le télémètre à ultrasons HC SR04 a une plage de mesure de 2 cm à 400 cm et fonctionne à des températures de 0° à 60° C. Pour commencer, nous utilisons un simple croquis, en utilisant la bibliothèque Ultrasonic. Après avoir connecté le télémètre HC SR04 à l'Arduino, chargez le code suivant.

Circuit



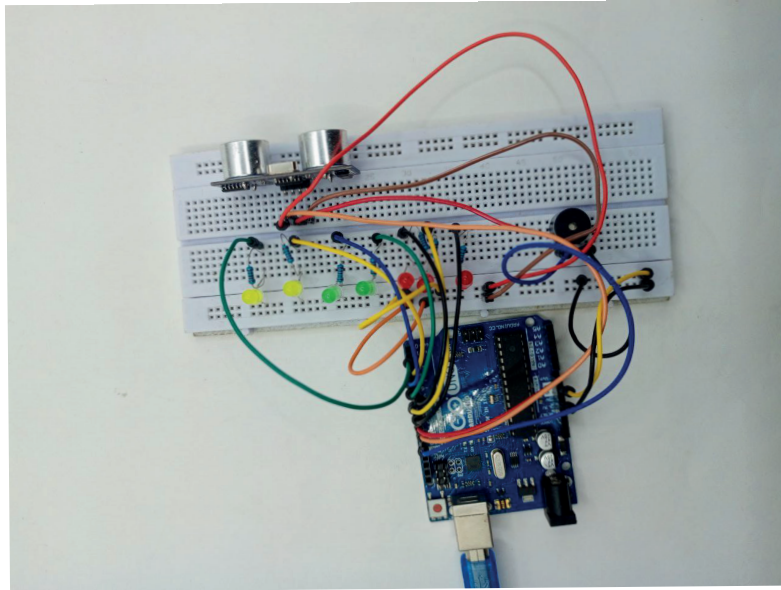
Code :

Le code en entier ici [🔗](#) :



```
sketch_jul22a.ino
1  #define TRIG_PIN 9 // Définit la broche TRIG comme la broche 9 de l'Arduino
2  #define ECHO_PIN 8 // Définit la broche ECHO comme la broche 8 de l'Arduino
3
4  void setup() {
5      Serial.begin(9600); // Initialise la communication série à une vitesse de 9600 bauds
6      pinMode(TRIG_PIN, OUTPUT); // Configure la broche TRIG comme une sortie
7      pinMode(ECHO_PIN, INPUT); // Configure la broche ECHO comme une entrée
8  }
9
10 void loop() {
11     // Mettre la broche TRIG en LOW pendant 2 microsecondes pour s'assurer qu'il n'y a pas d'impulsion parasite
12     digitalWrite(TRIG_PIN, LOW);
13     delayMicroseconds(2);
14
15     // Envoyer une impulsion HIGH de 10 microsecondes pour déclencher le capteur
16     digitalWrite(TRIG_PIN, HIGH);
17     delayMicroseconds(10);
18     digitalWrite(TRIG_PIN, LOW);
19
20     // Lire la durée de l'impulsion renvoyée par la broche ECHO
21     long duration = pulseIn(ECHO_PIN, HIGH);
22
23     // Calculer la distance en centimètres
24     long distance = duration * 0.034 / 2;
```

Projet 4: Capteur de stationnement



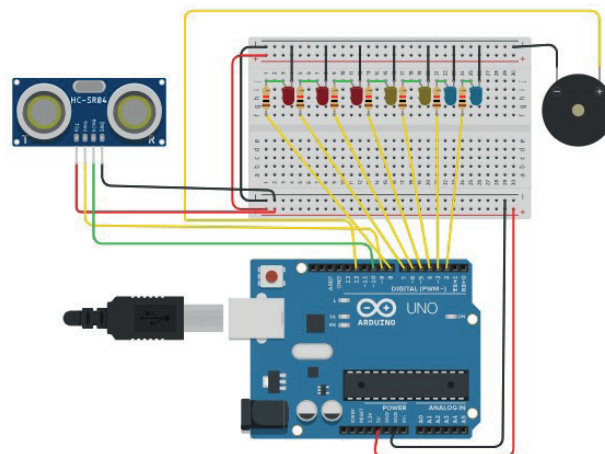
Description

Dans ce projet, nous avons un capteur à ultrasons qui mesure la distance et un graphique à barres LED qui s'allume en fonction de notre distance par rapport au capteur. À mesure que nous nous rapprochons du capteur, le buzzer émet un bip différent. Ce circuit peut fonctionner comme un capteur de stationnement.

Liste des composants

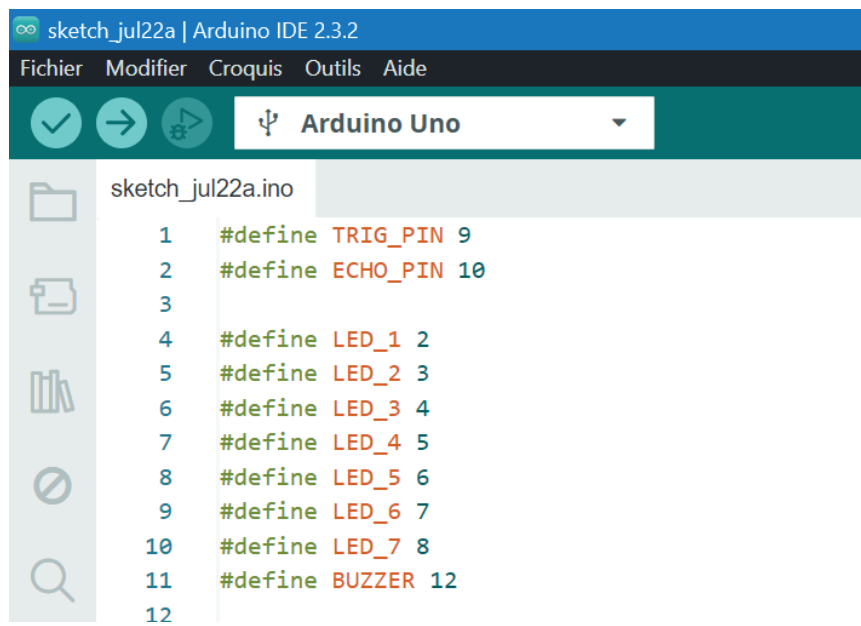
- 1 Arduino
- 1 Buzzer
- 1 Capteur à ultrason HC-SR04
- 7 LEDS (3 Rouges;2 Jaunes;2 Bleus)
- 7 Résistances de 1k Ω

Circuit



Code

Le code en entier ici [🔗](#) :

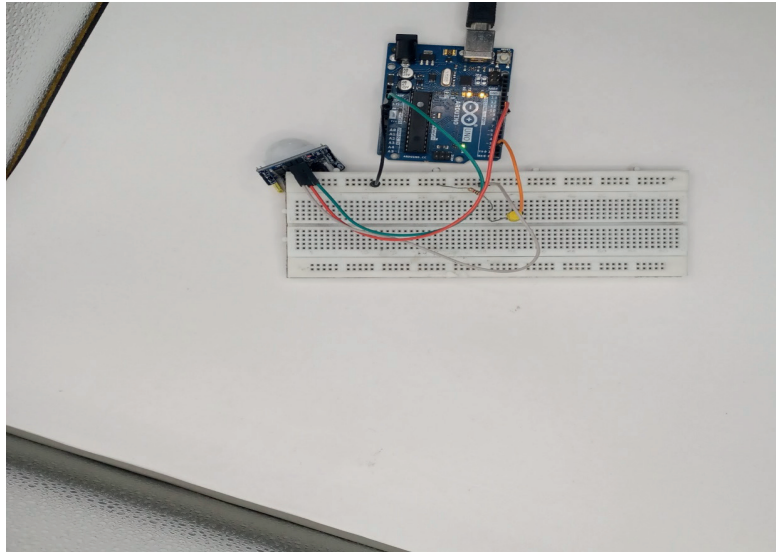


```
sketch_jul22a | Arduino IDE 2.3.2
Fichier  Modifier  Croquis  Outils  Aide

[Check] [Next] [Upload]  Arduino Uno

sketch_jul22a.ino
1  #define TRIG_PIN 9
2  #define ECHO_PIN 10
3
4  #define LED_1 2
5  #define LED_2 3
6  #define LED_3 4
7  #define LED_4 5
8  #define LED_5 6
9  #define LED_6 7
10 #define LED_7 8
11 #define BUZZER 12
12
```


Projet 5: Arduino avec capteur de mouvement PIR

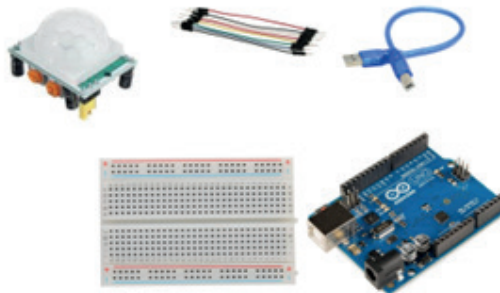


Description

Dans ce projet, nous allons créer un circuit simple avec un mouvement Arduino et PIR.

Pour résumer dès que le capteur détecte un mouvement, il le signale en allumant une LED qui restera allumer pendant 1 seconde.

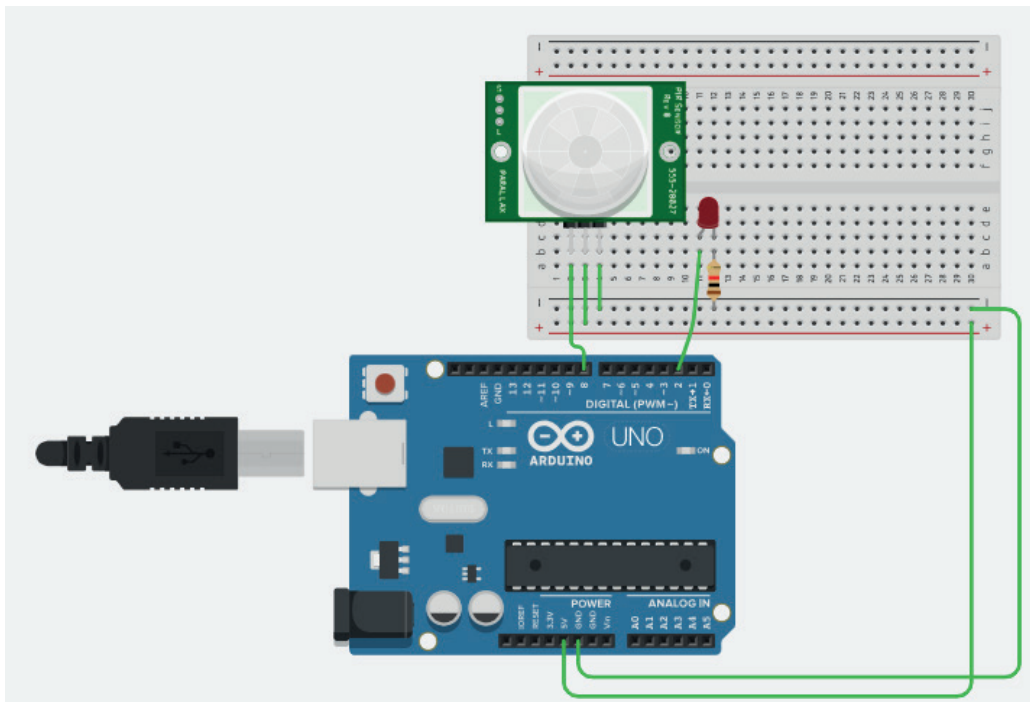
Pièces requises



Listes des composants

- 1 Arduino
- 1 Capteur de mouvement PIR
- 1 LED
- 1 Résistance de 1kΩ

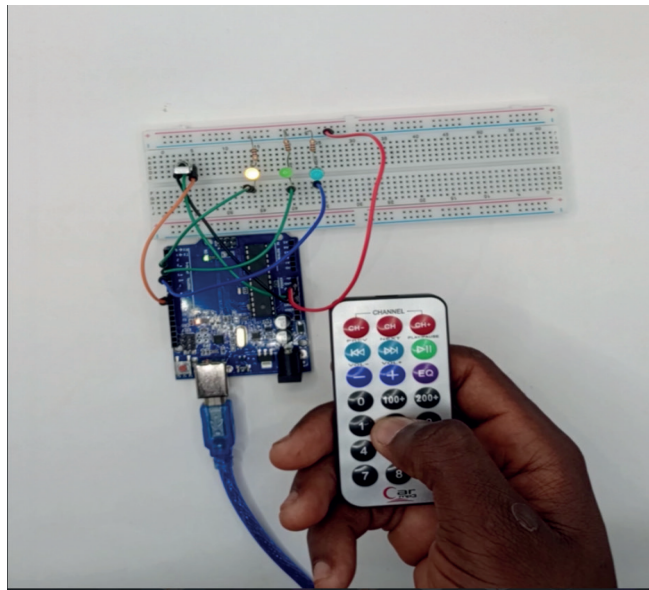
Circuit



Code :

```
sketch_jul22b | Arduino IDE 2.3.2
Fichier Modifier Croquis Outils Aide
Arduino Uno
sketch_jul22b.ino
1 #define Capteur_PIR 8
2 #define LED 13
3
4 void setup() {
5   // put your setup code here, to run once:
6   pinMode(Capteur_PIR,INPUT);
7   pinMode(LED,OUTPUT);
8   Serial.begin(9600);
9 }
10
11 void loop() {
12   // put your main code here, to run repeatedly:
13   Serial.println(digitalRead(Capteur_PIR));
14   if(digitalRead(Capteur_PIR)==HIGH){
15     digitalWrite(LED,HIGH);
16     delay(1000);
17   }
18   else{
19     digitalWrite(LED,LOW);
20     delay(1000);
21   }
22 }
23 }
24 }
```

Projet 6: Contrôlez les LEDs avec la télécommande IR



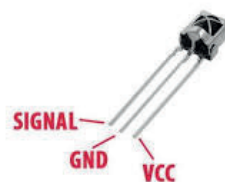
Description

Dans ce projet, vous utiliserez un récepteur infrarouge (IR) pour contrôler 3 LED avec une télécommande. Vous les allumerez et les éteindrez avec les boutons de votre télécommande. Il s'agit d'un bon projet pour débutant pour vous familiariser avec le récepteur infrarouge.

Ce projet est divisé en deux parties :

- Vous décoderez les signaux IR transmis par votre télécommande
- Vous utiliserez ces informations pour effectuer une tâche avec votre Arduino (contrôler 3 LED)

Récepteur infrarouge (IR)



Comme vous pouvez le voir, il comporte trois broches :

- **OUT (SORTIE)**
- **GND (terre)**
- **VCC (source de tension)**

Lorsque vous appuyez sur votre télécommande, elle envoie des signaux modulés infrarouges. Ces signaux contiennent des informations collectées par votre récepteur.

Chaque bouton envoie des informations spécifiques. Nous pouvons donc attribuer ces informations à un bouton spécifique.

Listes des composants

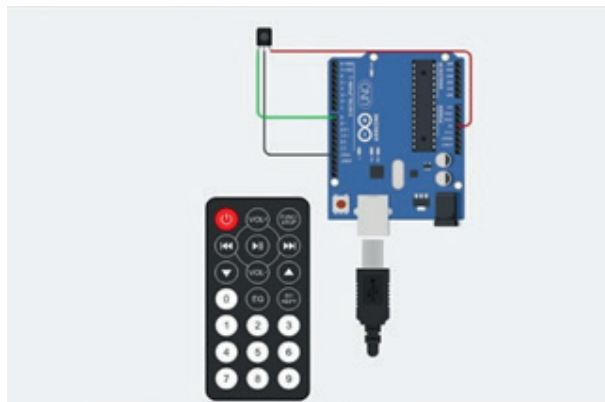
Pour ce projet, vous aurez besoin d'une télécommande. Si vous n'avez pas de télécommande chez vous, vous devez en acheter une. N'importe quelle télécommande fera l'affaire, même si elle est très ancienne et que vous n'utilisez pas. En fait, si vous en avez un ancien, c'est parfait. C'est un nouveau but pour quelque chose d'ancien.

- 1 Arduino
- 1 [Télécommande IR](#)
- 1 [Récepteur IR \(TSOP4838\)](#)
- 3 LEDS
- 3 Résistances de 1K Ω

1. Décoder les signaux IR

Dans cette partie du projet, vous devez décoder les signaux IR associés à chaque bouton.

Circuit



Connectez le récepteur IR conformément aux schémas ci-dessus.

Code

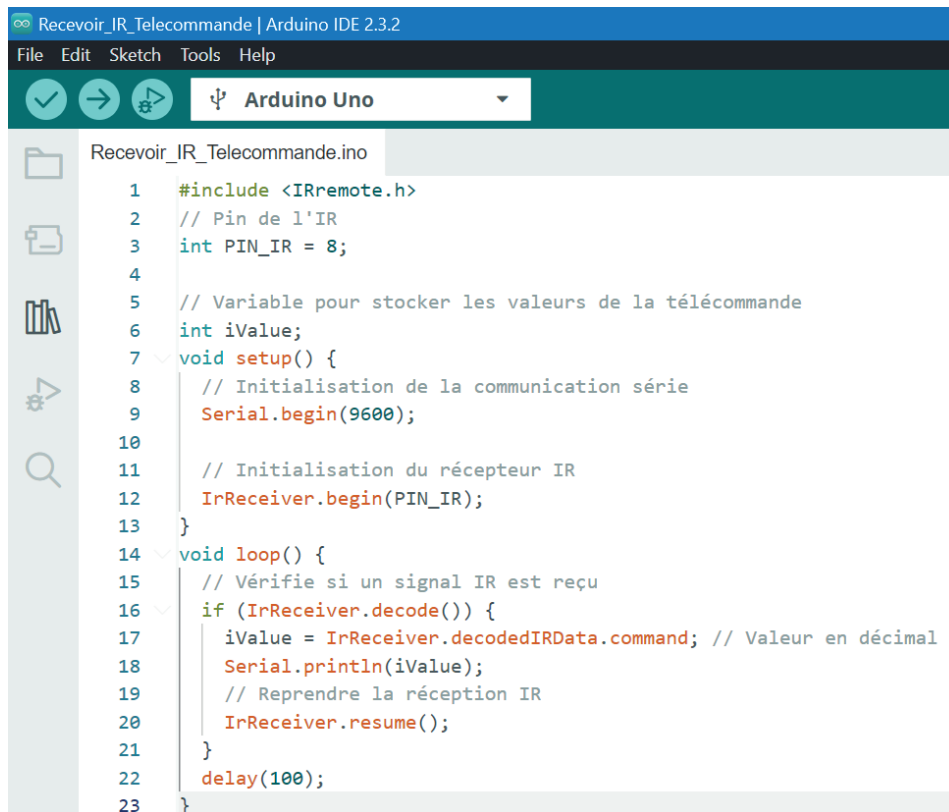
Le code en entier ici [📄](#) :

Pour contrôler le récepteur IR, vous devez installer la bibliothèque IRremote dans l'IDE Arduino.

- Installation de la bibliothèque IRremote
 1. Cliquez [ici](#) pour télécharger la bibliothèque IRremote. Vous devriez avoir un dossier .zip dans vos téléchargements
 2. Déplacez le dossier IRremote vers le dossier des bibliothèques d'installation de votre IDE Arduino
 3. . Enfin, rouvrez votre IDE Arduino

Copiez le code suivant sur votre IDE Arduino et téléchargez-le sur votre carte Arduino.

Assurez-vous que vous avez sélectionné la bonne carte et le bon port COM.



```
Recevoir_IR_Telecommande | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
Recevoir_IR_Telecommande.ino
1 #include <IRremote.h>
2 // Pin de l'IR
3 int PIN_IR = 8;
4
5 // Variable pour stocker les valeurs de la télécommande
6 int iValue;
7 void setup() {
8 // Initialisation de la communication série
9 Serial.begin(9600);
10
11 // Initialisation du récepteur IR
12 IrReceiver.begin(PIN_IR);
13 }
14 void loop() {
15 // Vérifie si un signal IR est reçu
16 if (IrReceiver.decode()) {
17 iValue = IrReceiver.decodedIRData.command; // Valeur en décimal
18 Serial.println(iValue);
19 // Reprendre la réception IR
20 IrReceiver.resume();
21 }
22 delay(100);
23 }
```

Ouvrez le moniteur série à un débit en bauds de 9 600.



Dans ce projet, vous souhaitez contrôler 3 LED. Choisissez 6 boutons pour les tâches suivantes :

1. LED1 – ALLUMÉE
2. LED1 – ÉTEINTE
3. LED2 – ALLUMÉE
4. LED2 – ÉTEINTE
5. LED3 – ALLUMÉE
6. LED3 – ÉTEINTE

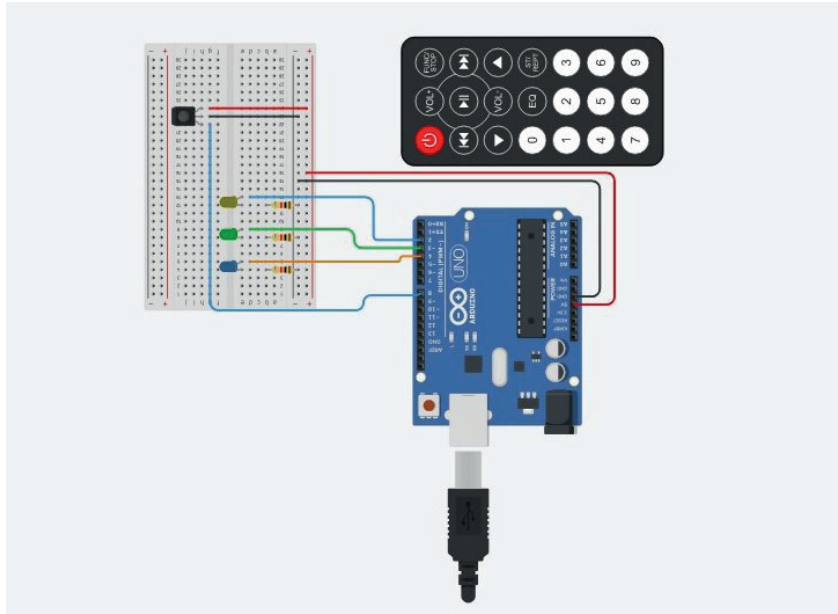
Appuyez sur le bouton numéro 1. Vous devriez voir un code sur le moniteur série. Appuyez plusieurs fois sur le même bouton pour vous assurer que vous avez le bon code pour ce bouton. Si vous voyez quelque chose comme FFFFFFFF, ignorez-le, c'est de la poubelle.

Faites de même pour les autres boutons. Notez le code associé à chaque bouton, car vous en aurez besoin informations plus tard.

2. Construire le circuit final

Dans cette partie, vous construirez le circuit avec trois LED qui seront contrôlées à l'aide de votre télécommande. Assemblez toutes les pièces en suivant les schémas ci-dessous.

Circuit



Code

Le code en entier ici [📄](#) :

Copiez le croquis suivant sur votre IDE Arduino. Écrivez vos propres valeurs décimales dans le croquis fourni dans les lignes du cas et téléchargez-le sur votre carte Arduino. Assurez-vous que vous avez sélectionné la bonne carte et le bon port COM.

```
Commande_LED_Commande | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
Commande_LED_Commande.ino
1 #include <IRremote.h>
2 // Pin de l'IR
3 int PIN_IR = 8;
4 // pin des leds
5 int pin_ledJ=2;
6 int pin_ledV=3;
7 int pin_ledB=4;
8 // Variable pour stocker les valeurs de la télécommande
9 int iValue;
10 void setup() {
11 // Initialisation de la communication série
12 Serial.begin(9600);
13 pinMode(pin_ledJ, OUTPUT);
14 pinMode(pin_ledV, OUTPUT);
15 pinMode(pin_ledB, OUTPUT);
16 // Initialisation du récepteur IR
17 IrReceiver.begin(PIN_IR);
18 }
```

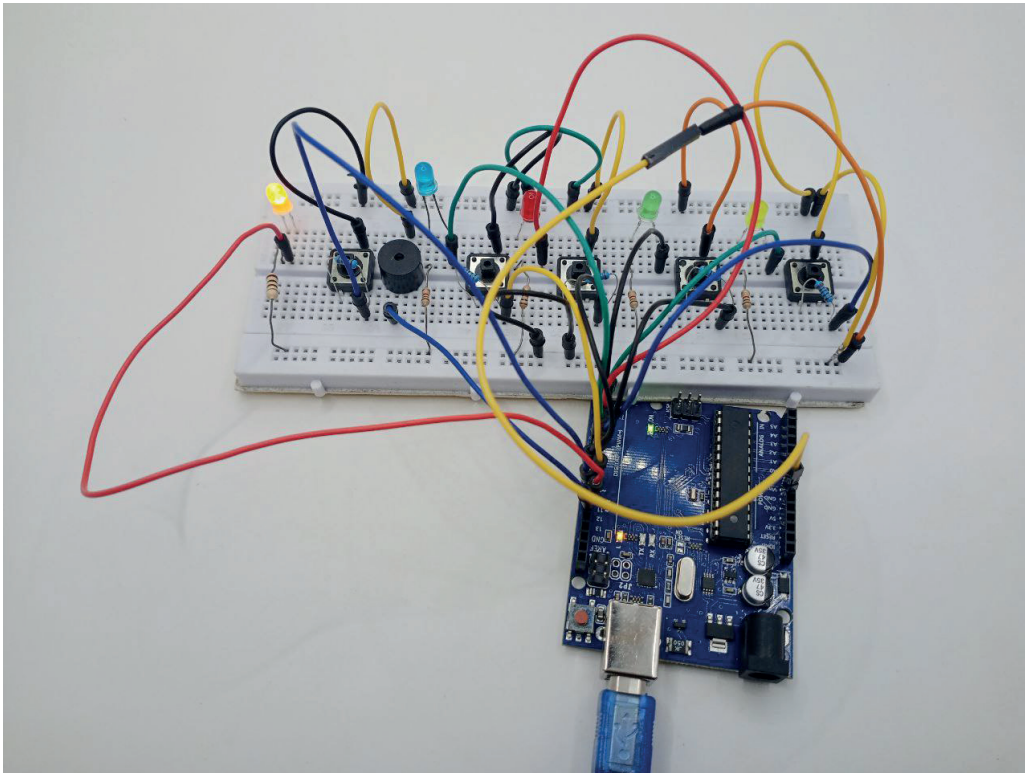
Manifestation

Vous pouvez désormais contrôler chaque LED individuellement à l'aide des boutons de votre télécommande. C'est un excellent projet pour en savoir plus sur le récepteur IR. Il existe des possibilités infinies quant à ce que

vous pouvez en faire.

Par exemple, vous pouvez remplacer ces LED par un relais pour contrôler vos appareils électroménagers. Cela peut être particulièrement utile car certaines télécommandes comportent de nombreux boutons que vous n'utilisez jamais. Alors pourquoi ne pas les utiliser pour faire quelque chose ?

Projet 7: Teensy/Arduino - Jeu de mémoire



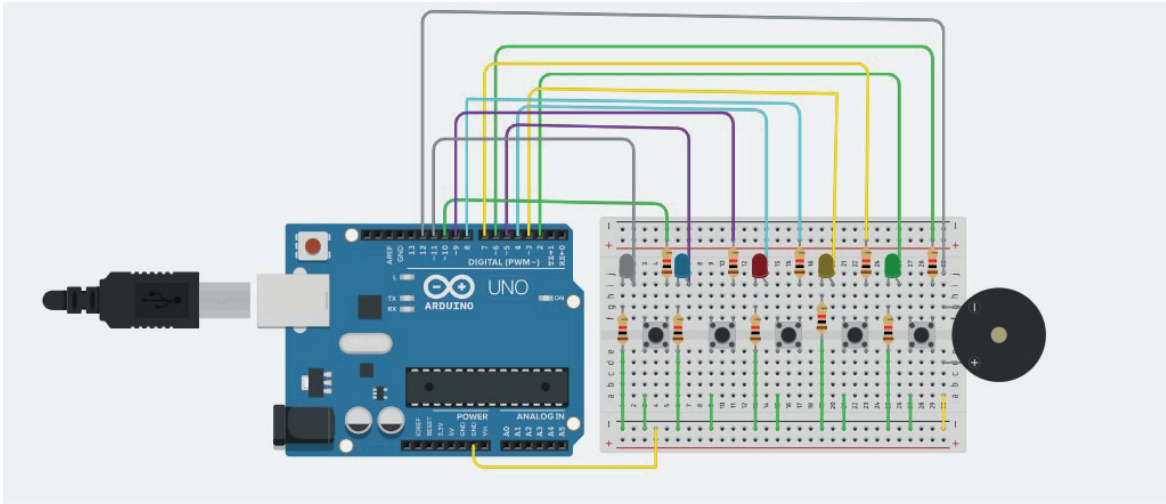
Description

Dans ce projet, vous créez un jeu simple pour tester votre mémoire.

Liste des composants

- ❖ 1 Arduino UNO
- ❖ 4 LEDs
- ❖ 8 Résistances (8x1k Ω)
- ❖ 4 Boutons poussoirs
- ❖ 1 Buzzer
- ❖ Jumpers

Circuit 8



Code :

Le code en entier ici [👉](#) :

```
Jeu_Memoire | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
Jeu_Memoire.ino
1  const int MAX_LEVEL = 100;
2  int sequence[MAX_LEVEL];
3  int your_sequence[MAX_LEVEL];
4  int level = 1;
5
6  int velocity = 1000;
7
8  int const led_1 = 2;
9  int const led_2 = 3;
10 int const led_3 = 4;
11 int const led_4 = 5;
12
13 int const bouton_4 = 9;
14 int const bouton_3 = 8;
15 int const bouton_2 = 7;
16 int const bouton_1 = 6;
17
18 int const TEM = 11;
19 int const BUZZER = 12; // Broche du buzzer
20
21 void setup() {
```

Projet 8 : : Guide pour le capteur de gaz/fumée MQ-2 avec Arduino



Capteur de gaz MQ-2

Définition

Le capteur de gaz et de fumée MQ-2 est un capteur populaire utilisé pour détecter une variété de gaz combustibles tels que le GPL, l'isobutane (C_4H_{10}), le propane (C_3H_8), le méthane (CH_4), l'hydrogène (H_2), l'alcool et la fumée. Ce module est compatible avec Arduino via une sortie analogique ou digitale.

Principe de fonctionnement

Le capteur de gaz et de fumée MQ-2 utilise le principe de la **chimisorption** pour détecter la présence de gaz combustibles et de fumée. Il contient un matériau sensible appelé **dioxyde d'étain (SnO_2)**, qui est un semi-conducteur. Ce matériau est déposé sur un substrat chauffant en céramique et recouvert par une grille de protection pour permettre le passage des gaz.

Le capteur a un élément chauffant interne (une petite résistance) qui maintient le dioxyde d'étain à une température optimale. Cela permet au matériau de réagir avec les gaz présents dans l'environnement.

En présence de gaz combustibles (comme le méthane, le propane, le butane, etc.) ou de fumée, ces gaz réagissent avec l'oxygène adsorbé à la surface du dioxyde d'étain. Cette réaction chimique provoque une **réduction de la concentration d'oxygène** à la surface du matériau sensible, ce qui modifie la densité des porteurs de charge (électrons et trous) dans le semi-conducteur.

La réaction chimique avec les gaz cibles entraîne une modification de la **résistance électrique** du dioxyde d'étain. En l'absence de gaz combustibles, la résistance du capteur est relativement élevée. Lorsqu'il y a des gaz présents, la résistance diminue de manière significative.

Cette variation de résistance est convertie en une tension électrique que le microcontrôleur (comme un Arduino) peut lire. En utilisant un circuit de pont de Wheatstone ou en mesurant directement la tension analogique, le changement de résistance peut être quantifié et corrélé à la concentration de gaz présents.

Caractéristiques techniques

- ❖ Taille : 32mm x 22mm x 27mm
- ❖ Puce : Le LM393, ZYMQ-2 Capteur de Gaz
- ❖ Tension De Fonctionnement: DC 5 V
- ❖ Sortie analogique (AO): tension de sortie analogique 0 ~ 5 V
- ❖ Sortie numérique (DO): sortie 0V ou 5V
- ❖ Durée de préchauffage: brochage de 20 s

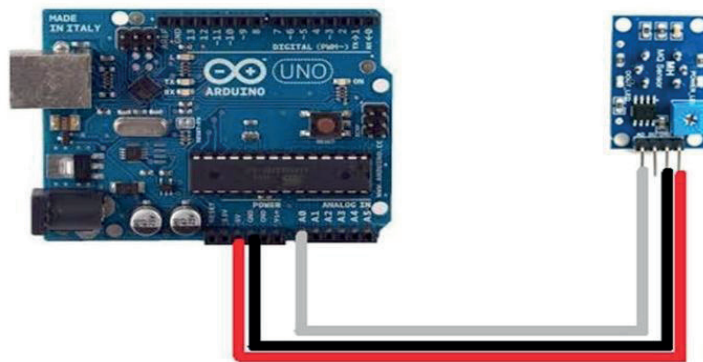
- VCC (+ 5 V)
- GND (0 V)
- DO (sortie numérique)
- AO (sortie analogique)

- ❖ Plage de mesure : 10 à 10000 ppm
- ❖ Sensibilité : 2 à 20 K Ω
- ❖ Température de service : -20 à 50 °C
- ❖ Compatibilité : Arduino

Connexion à un Arduino

Pour connecter le MQ-2 à un Arduino, voici un exemple de schéma de câblage et un code pour lire les valeurs analogiques du capteur.

Circuit




- ❖ VCC à 5V de l'Arduino
- ❖ GND à GND de l'Arduino
- ❖ AO à une broche analogique de l'Arduino (par exemple A0)
- ❖ DO (optionnel) à une broche numérique de l'Arduino si vous souhaitez utiliser la sortie numérique

Liste des composants

- ❖ Arduino UNO
- ❖ Capteur de gaz MQ2
- ❖ Jumpers

Code :

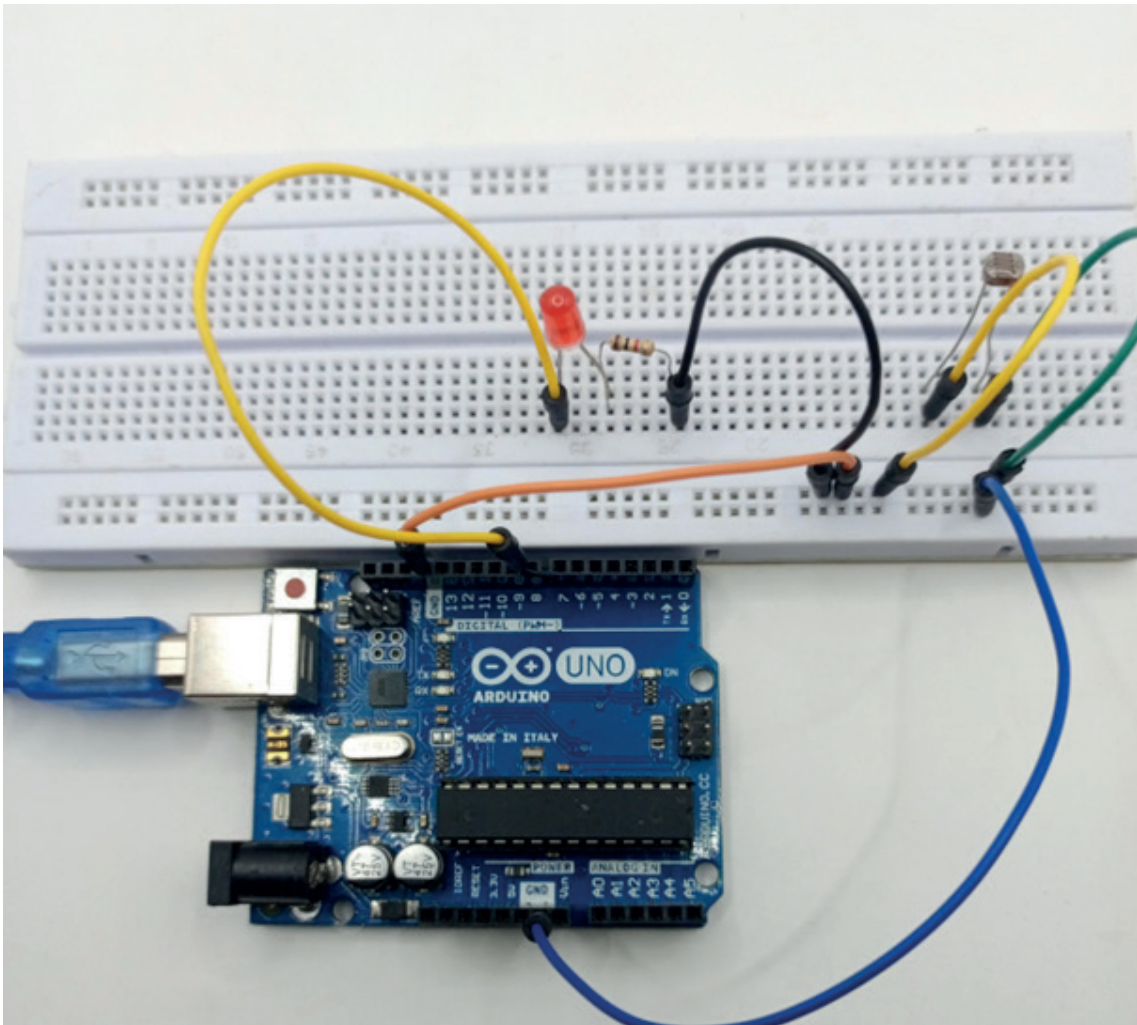
Le code en entier ici  :

```
sketch_jul25a.ino
1  const int gasSensorPin = A0; // Pin analogique où est connecté le capteur de gaz
2
3  void setup() {
4      Serial.begin(9600); // Initialisation de la communication série
5      pinMode(gasSensorPin, INPUT); // Définir la broche du capteur comme entrée
6  }
7
8  void loop() {
9      int sensorValue = analogRead(gasSensorPin); // Lire la valeur du capteur
10     Serial.print("MQ2 Gas Sensor Value: ");
11     Serial.println(sensorValue); // Afficher la valeur du capteur sur le moniteur série
12
13     delay(1000); // Attendre 1 seconde avant de lire à nouveau
14 }
15
```

Applications

- Détection de Gaz Domestique : Utilisé dans les systèmes de détection de fuites de gaz dans les maisons et les bureaux.
- Systèmes de Sécurité : Intégré dans les systèmes d'alarme pour détecter la fumée ou les gaz dangereux.
- Projets DIY : Utilisé dans divers projets de bricolage pour la détection de gaz et de fumée.

Projet 9 : Allumage d'une LED avec une photorésistance



Description

Le projet consiste à allumer une LED lorsque la luminosité ambiante tend vers l'obscurité

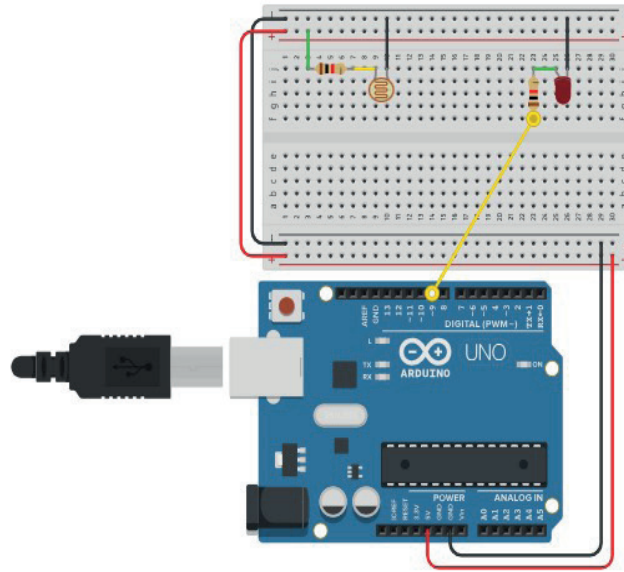
Liste des composants

- ❖ 1 Arduino UNO
- ❖ 1 LED
- ❖ 2 Résistances (2x1k Ω)
- ❖ Jumpers

Câblage

- ❖ Connection de la cathode de la LED à la masse (GND) via des résistances et l'anode à la broche D9
- ❖ Connection d'une patte de la photorésistance au VCC et de l'autre au GND via la résistance

Circuit :

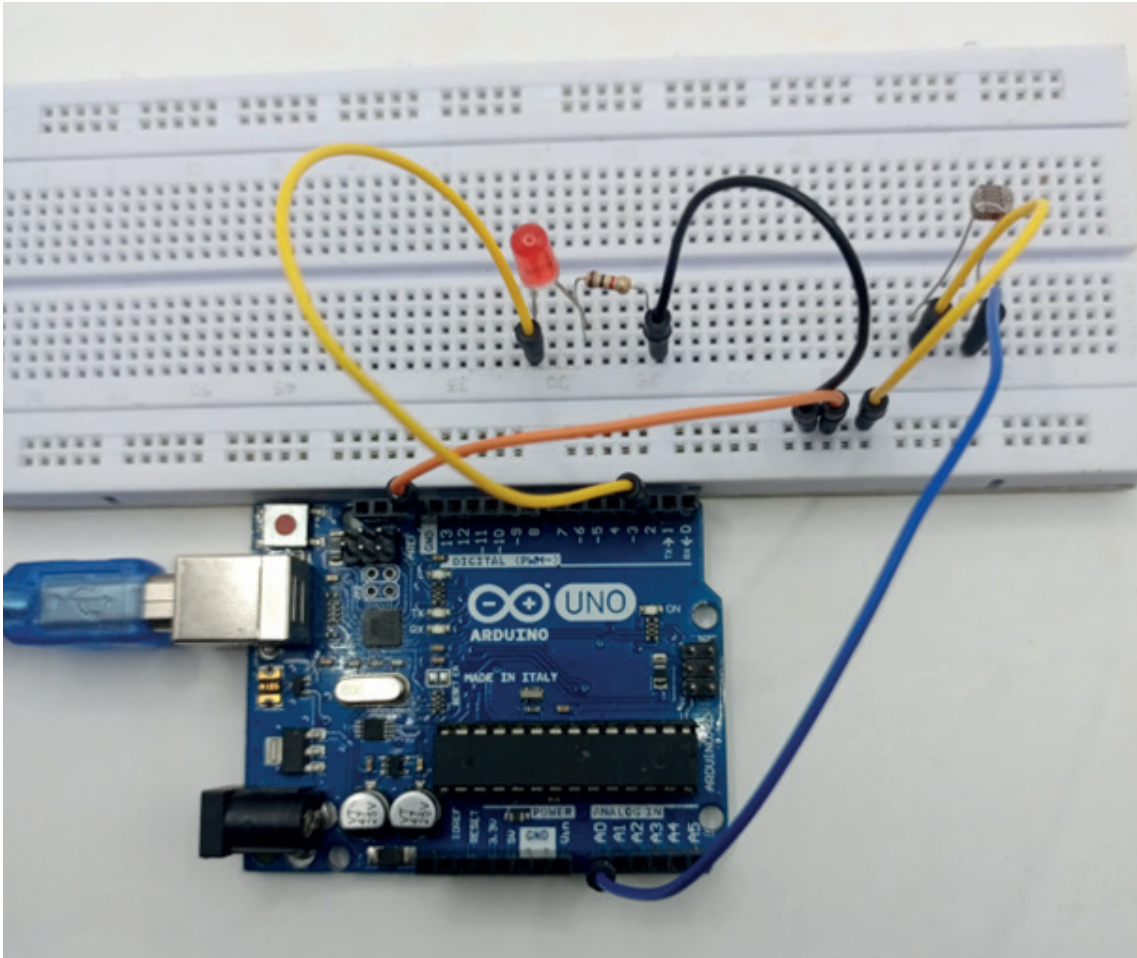


Code :

Le code en entier ici [🔗](#) :

```
sketch_jul25a.ino
1  const int lightPin = A0; // Pin analogique où le photoresistor est connecté
2  const int ledPin = 2;   // Pin où la LED est connectée
3  const int seuil = 900; // Seuil de luminosité
4
5  void setup() {
6    pinMode(ledPin, OUTPUT); // Définir la LED comme sortie
7    pinMode(lightPin, INPUT);
8    Serial.begin(9600);      // Initialiser la communication série pour le déb
9  }
10
11 void loop() {
12   int lightLevel = analogRead(lightPin); // Lire la valeur du photoresistor
13   Serial.println(lightLevel);           // Imprimer la valeur lue (facultat
14
15   if (lightLevel > seuil) {
16     digitalWrite(ledPin, HIGH); // Éteindre la LED si la luminosité dépasse
17   } else {
18     digitalWrite(ledPin, LOW); // Allumer la LED sinon
19   }
20
21   delay(100); // Petite pause pour éviter de surcharger la boucle
22 }
```

Projet 10: Contrôle de l'intensité la luminosité d'une LED



Description

Le projet consiste à faire varier l'intensité d'une LED en fonction de la luminosité ambiante à l'aide d'une photorésistance. La LED sera branchée sur une broche PWM et va varier en fonction de la valeur renvoyer par la photorésistance. Plus il y a de luminosité plus l'intensité de la lampe est faible et moins il y a de luminosité plus la lampe brille.

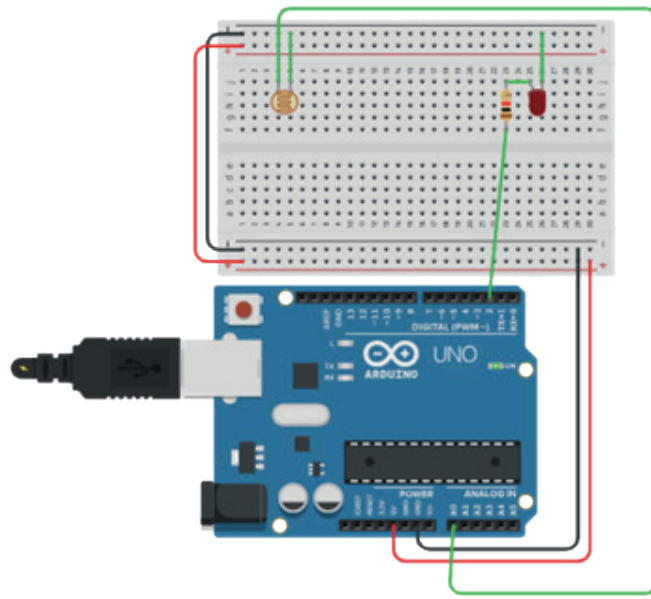
Liste des composants

- ❖ 1 Arduino UNO
- ❖ 1 LED
- ❖ 2 Résistances (2x1k Ω)
- ❖ Jumpers

Câblage

- Connection de la cathode de la LED à la masse (GND) via des résistances et l'anode à la broche Dé
- Connection d'une patte de la photorésistance a la borne A0 et de l'autre au GND via la résistance

Circuit



Code :

Le code en entier ici [👉](#) :

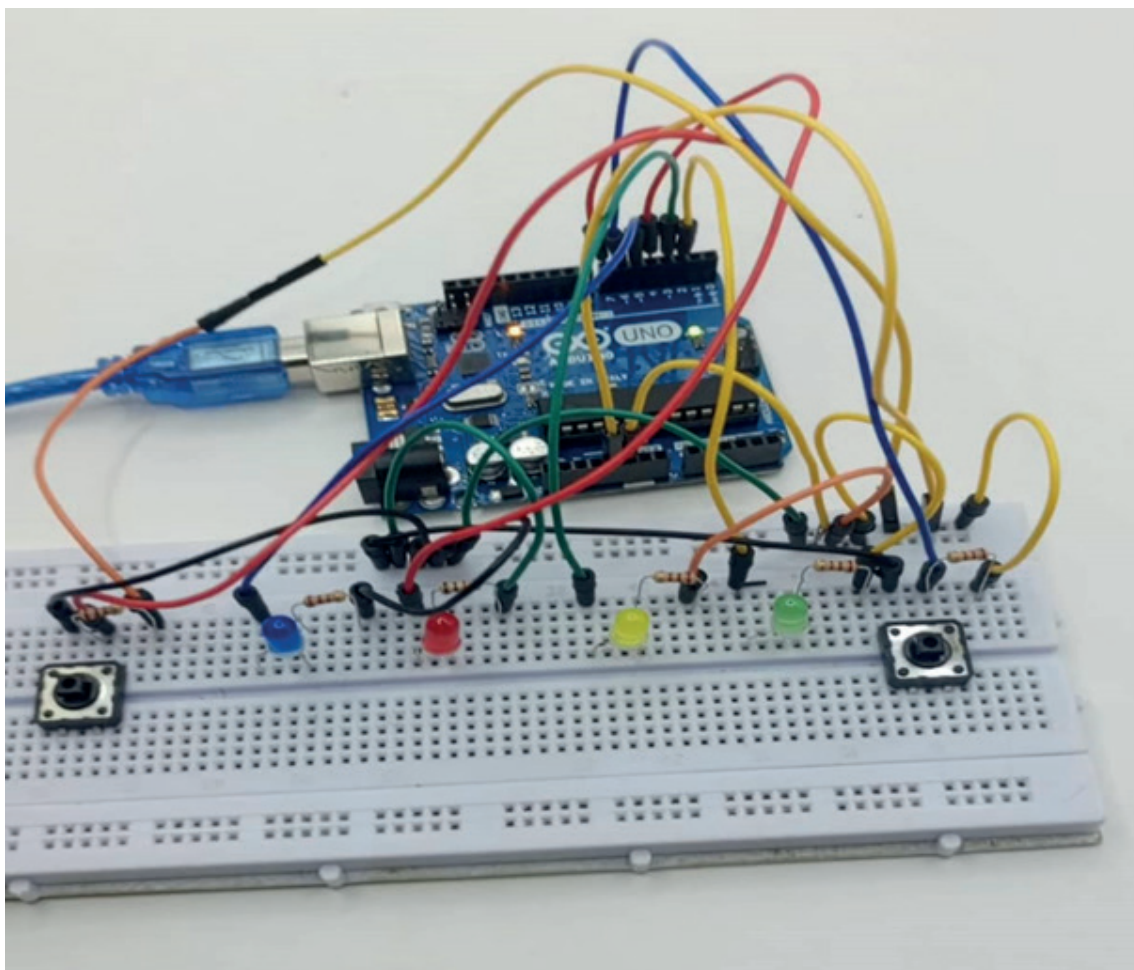
```
const int lightPin = A0; // Pin analogique où le photoresistor est connecté
const int ledPin = 9; // Pin où la LED est connectée (doit être une pin PWM)

void setup() {
  pinMode(ledPin, OUTPUT); // Définir la LED comme sortie
  Serial.begin(9600); // Initialiser la communication série pour le débogage
}

void loop() {
  int lightLevel = analogRead(lightPin); // Lire la valeur du photoresistor
  Serial.println(lightLevel); // Imprimer la valeur lue (facultatif, pour le débogage)

  int ledBrightness = map(lightLevel, 500, 1023, 0, 255); // Convertir la valeur lue (0-1023) en une valeur de PWM (0-255)
  analogWrite(ledPin, ledBrightness); // Ajuster la luminosité de la LED
  if (lightLevel <= 500) {
    analogWrite(ledPin, 0);
  }
  delay(100); // Petite pause pour éviter de surcharger la boucle
}
```


Projet 11: Bargraphe manuelle



Description

Un bargraphe est un dispositif d'affichage visuel composé de plusieurs segments lumineux, souvent sous forme de LEDs. Chaque segment représente une unité de mesure et s'illumine pour indiquer la valeur d'une variable mesurée. Plus la valeur est élevée, plus il y a de segments allumés. Les bargraphes sont utilisés pour des applications telles que l'indication du niveau de batterie, du volume sonore, ou encore de la température. L'objectif de cet exercice est de réaliser un bargraphe à 4 LEDs avec deux boutons poussoir l'un d'eux permettra d'incrémenter la valeur sur le bargraphe (à savoir augmenter le nombre de LEDs allumées) et l'autre permettra de la décrémenter.

Liste des composants

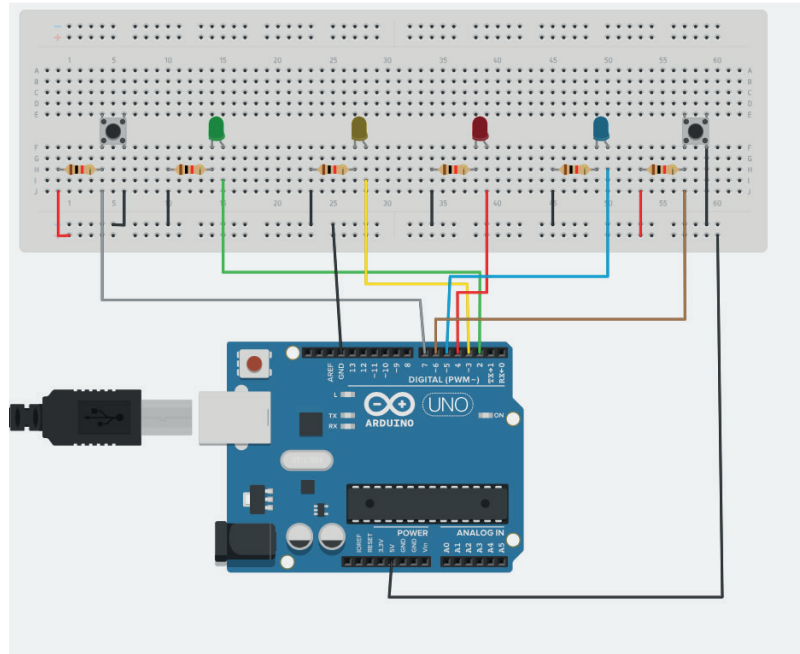
- ❖ 1 Arduino UNO
- ❖ 4 LEDs
- ❖ 6 Résistances (6x1k Ω)
- ❖ 2 Bouton poussoir
- ❖ Jumpers

Câblage

- ❖ Connection des cathodes des 4 LEDs à la masse (GND) via des résistances
- ❖ Connection des anodes des LEDs aux broches numériques de l'Arduino (D2 à D5).

- ❖ Connexion d'un côté de chaque bouton poussoir à la masse (GND).
- ❖ Connexion de l'autre côté de chaque bouton poussoir aux broches numériques de l'Arduino (par exemple, D6 et D7) et ajoutez une résistance pull-up interne en configurant les broches de manière appropriée dans le code.

Circuit



Code :

Le code en entier ici [👉](#) :

barographe-manuelle.ino

```

1 // Déclaration des broches pour les LEDs et les boutons poussoirs
2 const int ledPins[] = {2, 3, 4, 5}; // Broches numériques pour les 4 LEDs
3 const int buttonIncrement = 6; // Broche numérique pour le bouton d'incréméntation
4 const int buttonDecrement = 7; // Broche numérique pour le bouton de décrémentation
5
6 int ledState = 0; // Variable pour suivre le nombre de LEDs allumées
7
8 void setup() {
9 // Configurer les broches des LEDs comme sorties
10 for (int i = 0; i < 4; i++) {
11 | pinMode(ledPins[i], OUTPUT);
12 | }
13
14 // Configurer les broches des boutons poussoirs comme entrées avec résistances pull-up internes
15 pinMode(buttonIncrement, INPUT_PULLUP);
16 pinMode(buttonDecrement, INPUT_PULLUP);
17 }
18
19 void loop() {
20 // Vérifier si le bouton d'incréméntation est appuyé
21 if (digitalRead(buttonIncrement) == LOW) {
22 | incrementLEDs(); // Incrémentation le nombre de LEDs allumées
23 | delay(300); // Attendre 300 ms pour éviter les rebonds
24 | }
25 }

```

Projet 12: Accès sécurisé à l'aide de la RFID MFRC522 avec Arduino



Module RFID

Ce projet montre un exemple simple d'utilisation du lecteur RFID MFRC522.

Description

RFID signifie identification par radiofréquence. La RFID utilise des champs électromagnétiques pour transférer des données sur de courtes distances. La RFID est utile pour identifier des personnes, effectuer des transactions, etc... Vous pouvez utiliser un système RFID pour ouvrir une porte. Par exemple, seule la personne disposant des bonnes informations sur sa carte est autorisée à entrer. Un système RFID utilise :

- des tags attachés à l'objet à identifier, dans cet exemple nous avons un porte-clés et une carte électromagnétique. Chaque balise possède sa propre identification (UID).

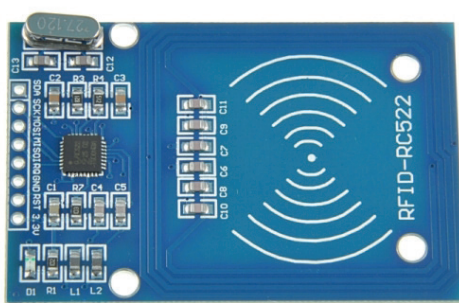


Etiquette RFID 125 kHz



Carte RFID

- émetteur-récepteur radio bidirectionnel, le lecteur, qui envoie un signal au tag et lis sa réponse.



Caractéristiques

Courant de fonctionnement de 13-26 mA / CC 3,3 V

Courant d'inactivité : 10-13 mA / CC 3,3 V

Courant de veille : <80uA

Courant de crête : <30 mA

Fréquence de fonctionnement : 13,56 MHz

Propriétés physiques du produit : Taille : 40 mm x 60 mm

Environnement Température de fonctionnement : -20-80 degrés Celsius

Environnement Température de stockage : -40-85 degrés Celsius

Humidité relative : 5% -95%

- Poids:8g
- Dimension:59mm x 39mm x 5mm

Téléchargement de la bibliothèque

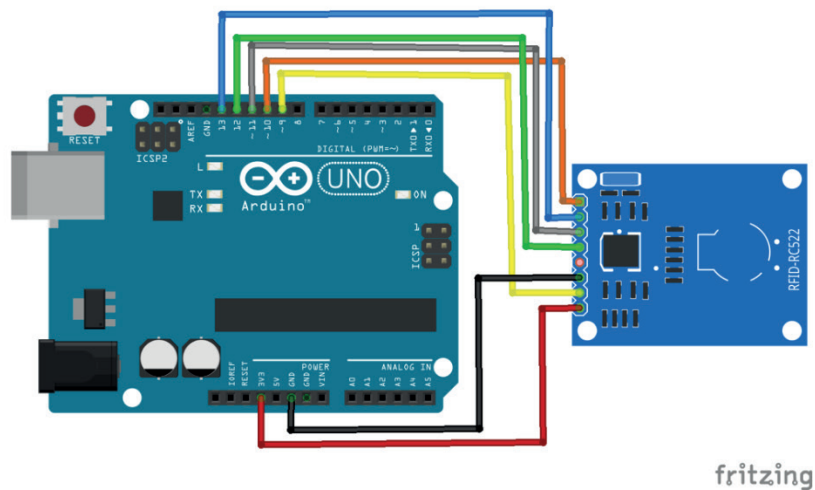
Voici la bibliothèque dont vous avez besoin pour ce projet :

1. [Cliquez ici pour télécharger la bibliothèque RFID](#)
2. Vous devriez avoir un dossier .zip dans votre téléchargement
3. Décompressez le dossier .zip et vous devriez obtenir le dossier RFID-master
4. Renommez votre dossier de RFID-master en RFID
5. Déplacez le dossier RFID vers le dossier des bibliothèques d'installation de votre Arduino IDE.
6. Enfin, rouvrez votre IDE Arduino

Brochages

MFRC522	Arduino Uno	Arduino Nano	Arduino Méga
GND	GND	GND	GND
VCC	3,3V	3,3V	3,3V
MOSI	11	11	51
MISO	12	12	50
SCK	13	13	52
SDA	10	10	10
RST	9	9	9
IRQ	-	-	-

Circuit



Code :

Le code en entier ici [🔗](#) :

Lire les données d'une étiquette RFID

Une fois le circuit prêt, allez dans Fichier Exemples MFRC522 DumpInfo et téléchargez le code. Ce code est disponible dans votre IDE Arduino après avoir installé la bibliothèque RFID.

```

1  #include "SPI.h"
2  #include "MFRC522.h"
3
4  #define RST_PIN  9 // RES pin
5  #define SS_PIN  10 // SDA (SS) pin
6
7  MFRC522 mfrc522(SS_PIN, RST_PIN);
8
9  void setup() {
10 |   Serial.begin(9600);
11 |   SPI.begin();
12 |   mfrc522.PCD_Init();
13 |   delay(4);
14 |   mfrc522.PCD_DumpVersionToSerial();
15 |   Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks.
16 | })
17
18 void loop() {
19 |   // réinitialiser le cycle s'il n'y a pas de carte sur le lecteur
20 |   if ( ! mfrc522.PICC_IsNewCardPresent() ) {
21 |       return;
22 |   }

```

Ensuite, ouvrez le moniteur série. Vous devriez voir quelque chose comme la figure ci-dessous :

The screenshot shows the Serial Monitor window with the following output:

```

13:14:30.056 -> []Firmware Version: 0xB2 = (unknown)
13:14:31.742 -> Scan PICC to see UID, SAK, type, and data blocks...

```

Rapprochez la carte RFID ou le porte-clés du lecteur. Rapprochez le lecteur et l'étiquette jusqu'à ce que toutes les informations soient affichées.

The screenshot shows the Serial Monitor window with the following output:


```

13:14:30.056 -> []Firmware Version: 0xB2 = (unknown)
13:14:31.742 -> Scan PICC to see UID, SAK, type, and data blocks...
13:16:01.173 -> Card UID: 23 B4 8F AC
13:16:01.208 -> Card SAK: 08
13:16:01.241 -> PICC type: MIFARE 1KB
13:16:01.241 -> Sector Block  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
13:16:01.339 ->  15      63 MIFARE Read() failed: Collision detected.

```

Il s'agit des informations que vous pouvez lire sur la carte, y compris l'UID de la carte indiquée par la flèche. Les informations sont stockées dans la mémoire qui est divisée en segments et blocs comme vous pouvez le voir sur l'image précédente. Vous disposez de 1024 octets de stockage de données répartis en 16 secteurs. Notez votre carte UID car vous en aurez besoin plus tard.

Code :

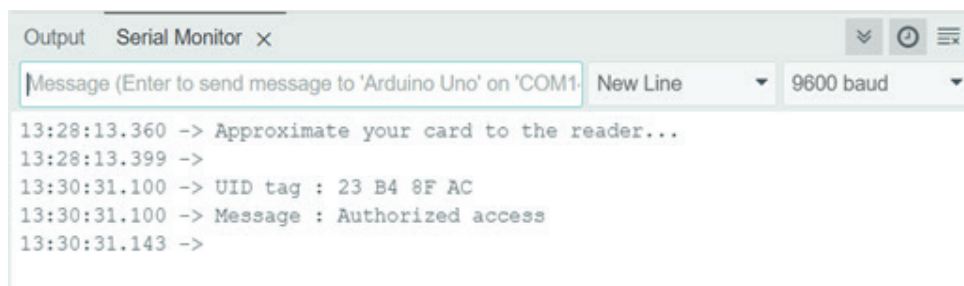
Le code en entier ici  :

```
MFRC522_RFID_Reader_with_Arduino.ino
1  #include <SPI.h>
2  #include <MFRC522.h>
3
4  #define SS_PIN 10
5  #define RST_PIN 9
6  MFRC522 mfc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
7
8  void setup()
9  {
10     Serial.begin(9600); // Initiate a serial communication
11     SPI.begin(); // Initiate SPI bus
12     mfc522.PCD_Init(); // Initiate MFRC522
13     Serial.println("Approximate your card to the reader...");
14     Serial.println();
15
16 }
```

Manifestation

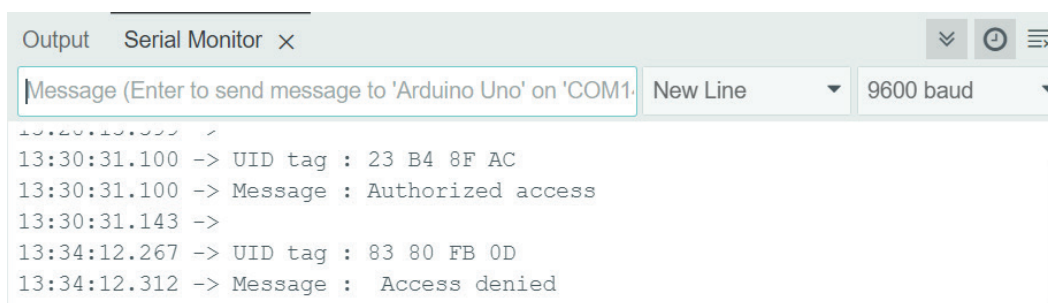
Ouvrez le moniteur série.

Rapprochez la carte que vous avez choisi de donner accès et vous verrez :



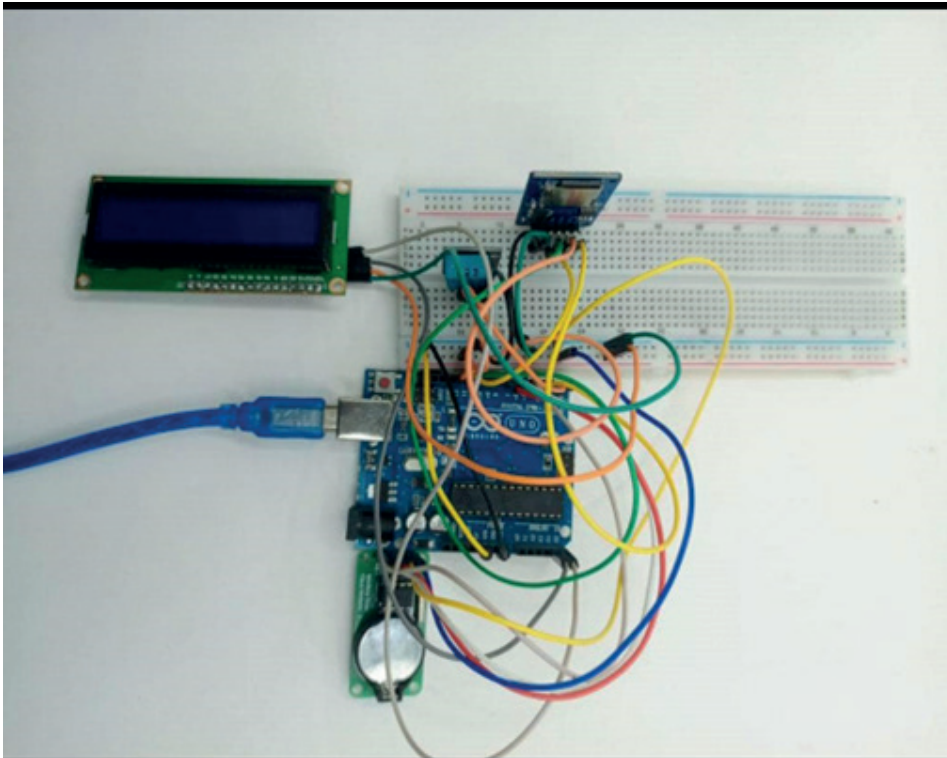
```
Output Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM1: New Line 9600 baud
13:28:13.360 -> Approximate your card to the reader...
13:28:13.399 ->
13:30:31.100 -> UID tag : 23 B4 8F AC
13:30:31.100 -> Message : Authorized access
13:30:31.143 ->
```

Si vous rapprochez une balise avec un autre UID, le message de refus s'affichera :



```
Output Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM1: New Line 9600 baud
13:28:13.360 -> Approximate your card to the reader...
13:28:13.399 ->
13:30:31.100 -> UID tag : 23 B4 8F AC
13:30:31.100 -> Message : Authorized access
13:30:31.143 ->
13:34:12.267 -> UID tag : 83 80 FB 0D
13:34:12.312 -> Message : Access denied
```

Projet 13: Enregistreur de données de température avec Arduino et module de carte SD



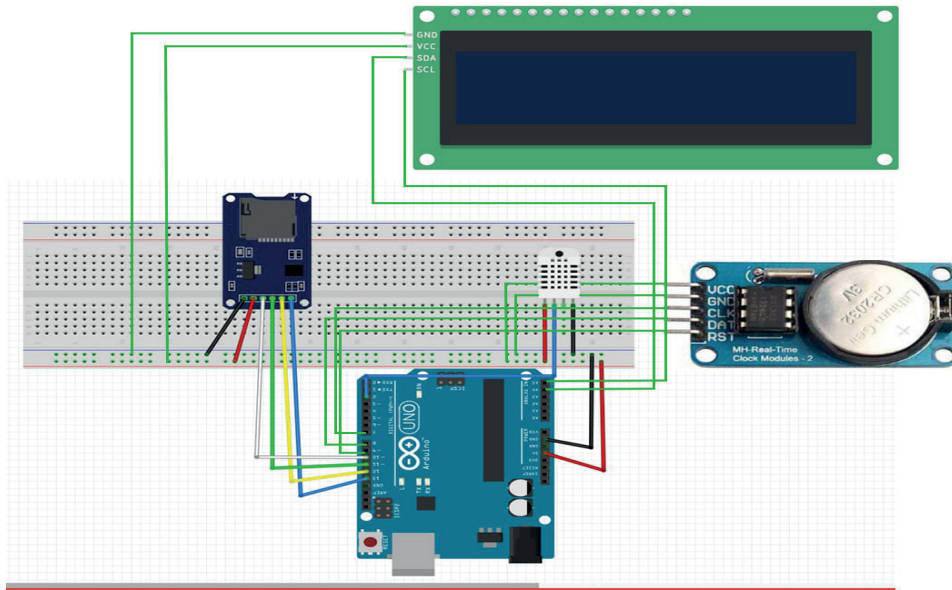
Description :

Le projet consiste à concevoir une station météo en affichant la température et l'humidité sur un écran LCD avec un le capteur de température dht11. Puis enregistrer les données (Température, Humidité et l'heure) dans un fichier texte sur une carte SD à l'aide du module le module SD Card.

Pièces requises

- ❖ Arduino UNO
- ❖ Capteur de Température DHT11
- ❖ Lecteur de micro Carte SD
- ❖ Module horloge RTC1302 ou RTC1307
- ❖ Ecran LCD 16x2 + I2C
- ❖ Jumpers
- ❖ 1 Breadboard

Circuit



Brochages :

Arduino	Lecteur Carte SD	RTC 1302	LCD16x2_I2C	DHT11
VCC	VCC	VCC	VCC	VCC
GND	GND	GND	GND	GND
A4	-	-	SDA	-
A5	-	-	SCL	-
2	-	-	-	SIGNAL
7	-	CLK	-	-
8	-	DAT	-	-
9	-	RST	-	-
10	CS	-	-	-
11	MOSI	-	-	-
12	MISO	-	-	-
13	SCK	-	-	-

Code:

Le code en entier ici [👉](#) :

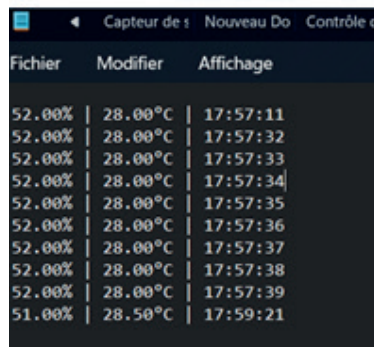
Installation de la bibliothèque de capteur DHT

- Ouvrez l'IDE Arduino.
- Allez dans Sketch > Include Library > Manage Libraries....
- Dans la fenêtre qui s'ouvre, tapez "DHT sensor library" dans la barre de recherche.
- Recherchez "DHT sensor library by Adafruit" et cliquez sur "Install".

```
File Edit Sketch Tools Help
Arduino Uno
sketch_aug7a.ino
1 #include <LiquidCrystal_I2C.h> // Inclut la bibliothèque pour contrôler un écran LCD via I2C.
2 #include "DHT.h" // Inclut la bibliothèque pour le capteur DHT (température et
3 #include <DS1302.h> // Inclut la bibliothèque pour interagir avec le module RTC DS
4 #include <SPI.h> // Inclut la bibliothèque pour la communication SPI.
5 #include <SD.h> // Inclut la bibliothèque pour interagir avec une carte SD.
6
7 // Définit le pin auquel le capteur DHT est connecté.
8 #define DHTPIN 2
9
10 // Crée un objet pour interagir avec un écran LCD I2C de 16 colonnes et 2 lignes, avec l'adres
11 LiquidCrystal_I2C lcd(0x27, 16, 2);
12
13 // Définit le type de capteur DHT utilisé (DHT11 dans ce cas).
14 #define DHTTYPE DHT11
15 DHT dht(DHTPIN, DHTTYPE); // Crée un objet pour le capteur DHT.
16
17 float h; // Variable pour stocker la valeur de l'humidité.
18 float t; // Variable pour stocker la valeur de la température.
19
20 // Initialise un objet pour interagir avec le module RTC DS1302, connecté aux pins 7, 8, et 9.
21 DS1302 rtc(7, 8, 9);
22 Time T; // Crée un objet pour stocker l'heure et la date lues à partir du module RTC.
23
```

Ln 172, Col 1 Arduino Uno on COM14 [not connected]

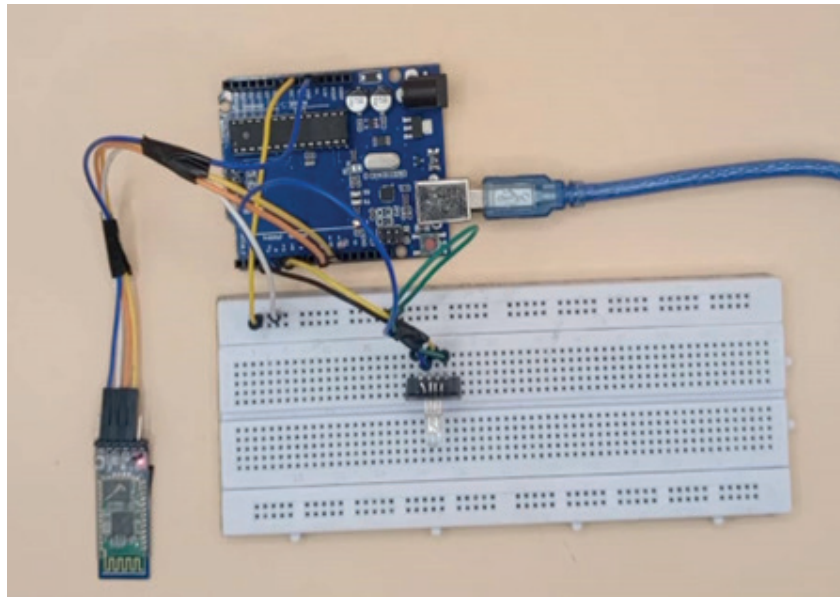
Résultat :



Capteur de s Nouveau Do Contrôle d

Fichier	Modifier	Affichage
52.00%	28.00°C	17:57:11
52.00%	28.00°C	17:57:32
52.00%	28.00°C	17:57:33
52.00%	28.00°C	17:57:34
52.00%	28.00°C	17:57:35
52.00%	28.00°C	17:57:36
52.00%	28.00°C	17:57:37
52.00%	28.00°C	17:57:38
52.00%	28.00°C	17:57:39
51.00%	28.50°C	17:59:21

Projet 14 : RGB+MODULE BLUETOOTH



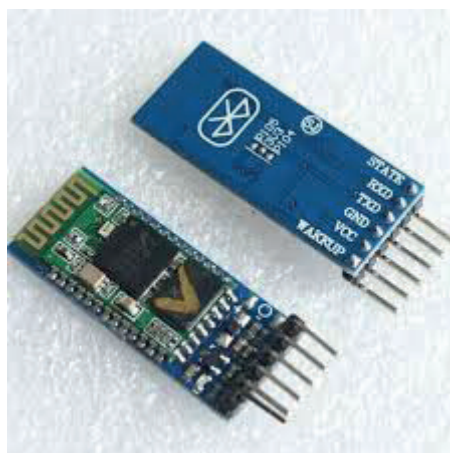
Description du projet

L'objectif de ce projet est de contrôler une LED RGB en utilisant des commandes envoyées via Bluetooth. En fonction de la commande reçue, une des couleurs de la LED RGB (rouge, vert, bleu) sera allumée. Les commandes sont envoyées depuis un smartphone ou un autre dispositif Bluetooth.

Ce projet est idéal pour en savoir plus sur :

- ❖ L'interface Arduino avec votre smartphone
- ❖ Le module Bluetooth
- ❖ La LED RGB

Le **module Bluetooth HC-05** est un composant largement utilisé pour ajouter des fonctionnalités Bluetooth à divers projets électroniques. Il est compatible avec des microcontrôleurs comme l'Arduino, permettant une communication sans fil facile et efficace.



Module Bluetooth HC-05

Caractéristiques Techniques :

- **Version Bluetooth** : V2.0+EDR (Enhanced Data Rate)
- **Tension d'alimentation** : 3.6V à 6V DC (couramment utilisé avec 5V)
- **Tension de fonctionnement** : 3.3V (niveau logique des broches)
- **Portée** : Jusqu'à 10 mètres (en champ libre)
- **Modes de fonctionnement** : Master et Slave (configurable)
- **Protocole de communication** : UART (Universal Asynchronous Receiver-Transmitter)
- **Baud Rate** : Configurable, avec une valeur par défaut de 9600 bps
- **Consommation de courant** : Environ 30 mA en fonctionnement normal
- **Dimensions** : Environ 28 mm x 15 mm x 2.35 mm

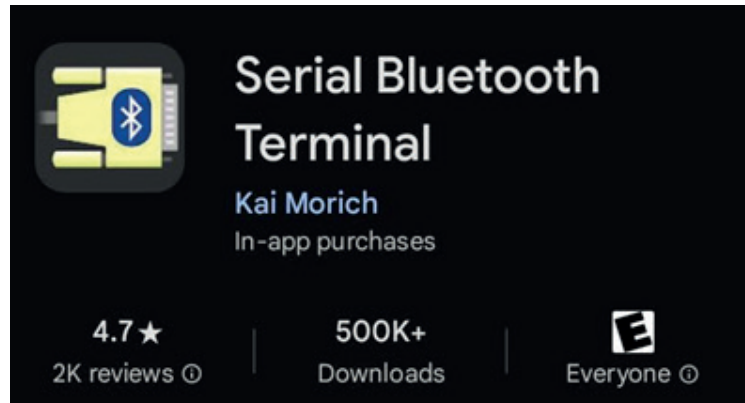
Brochage

1. **EN** : Broche pour activer le mode AT Command (haute = mode AT, basse = fonctionnement normal)
2. **VCC** : Alimentation du module (3.6V - 6V)
3. **GND** : Masse
4. **TXD** : Transmission des données (doit être connecté à RX (définir avec l'aide de la bibliothèque <SoftwareSerial.h>) de l'Arduino)
5. **RXD** : Réception des données (doit être connecté à TX (définir avec l'aide de la bibliothèque <SoftwareSerial.h>) de l'Arduino)
6. **STATE** : Indique l'état de la connexion (connecté ou non)

Code test du module HC-05 :

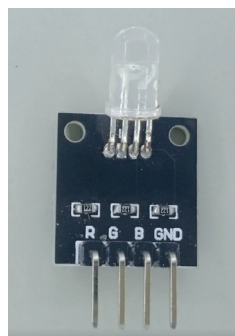
```
1  #include <SoftwareSerial.h>
2
3  SoftwareSerial BTSerial(10, 11); // RX | TX
4
5  void setup() {
6    Serial.begin(9600);
7    BTSerial.begin(9600);
8    Serial.println("Bluetooth module ready");
9  }
10
11 void loop() {
12   if (BTSerial.available()) {
13     char data = BTSerial.read();
14     Serial.write(data);
15   }
16
17   if (Serial.available()) {
18     char data = Serial.read();
19     BTSerial.write(data);
20   }
21 }
22
```

Pour pouvoir vous connecter a votre module bluetooth vous aurez à utiliser l'application serial bluetooth disponible sur playstore



Description de la LED RGB

Une LED RGB (Red, Green, Blue) est un type de diode électroluminescente capable de produire une large gamme de couleurs en combinant différentes intensités des trois couleurs de base : rouge, vert et bleu. Les LEDs RGB sont couramment utilisées dans les projets électroniques et les applications d'éclairage pour créer des effets de lumière colorée.



LED RGB

Fonctionnement

La LED RGB contient trois LEDs distinctes dans un seul boîtier : une rouge, une verte et une bleue. En contrôlant individuellement l'intensité de chaque couleur, il est possible de produire presque toutes les couleurs visibles.

Brochages :

Une LED RGB a généralement quatre broches :

1. **Broche commune** (cathode commune)
2. **Broche rouge (R)**
3. **Broche verte (G)**
4. **Broche bleue (B)**

Schéma de connexion pour une LED RGB

1. **Cathode commune** : Connecter à **GND**.
2. **Broche rouge (R)** : Connecter à une résistance, puis à une broche numérique de l'Arduino (par exemple, D9).

3. **Broche verte (G)** : Connecter à une résistance, puis à une broche numérique de l'Arduino (par exemple, D10).
4. **Broche bleue (B)** : Connecter à une résistance, puis à une broche numérique de l'Arduino (par exemple, D11).

Code test

L'intégralité du code est disponible ici :

```
int redPin = 9; // Broche connectée à la LED rouge
int greenPin = 10; // Broche connectée à la LED verte
int bluePin = 11; // Broche connectée à la LED bleue

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  // Afficher la couleur rouge
  analogWrite(redPin, 255); // Intensité maximale pour le rouge
  analogWrite(greenPin, 0); // Éteindre le vert
  analogWrite(bluePin, 0); // Éteindre le bleu
  delay(1000); // Attendre une seconde

  // Afficher la couleur verte
  analogWrite(redPin, 0); // Éteindre le rouge
  analogWrite(greenPin, 255); // Intensité maximale pour le vert
  analogWrite(bluePin, 0); // Éteindre le bleu
  delay(1000); // Attendre une seconde

  // Afficher la couleur bleue
  analogWrite(redPin, 0); // Éteindre le rouge
  analogWrite(greenPin, 0); // Éteindre le vert
  analogWrite(bluePin, 255); // Intensité maximale pour le bleu
}
```

Revenons à notre projet. Pour sa réalisation nous aurons besoin de :

Matériaux nécessaires

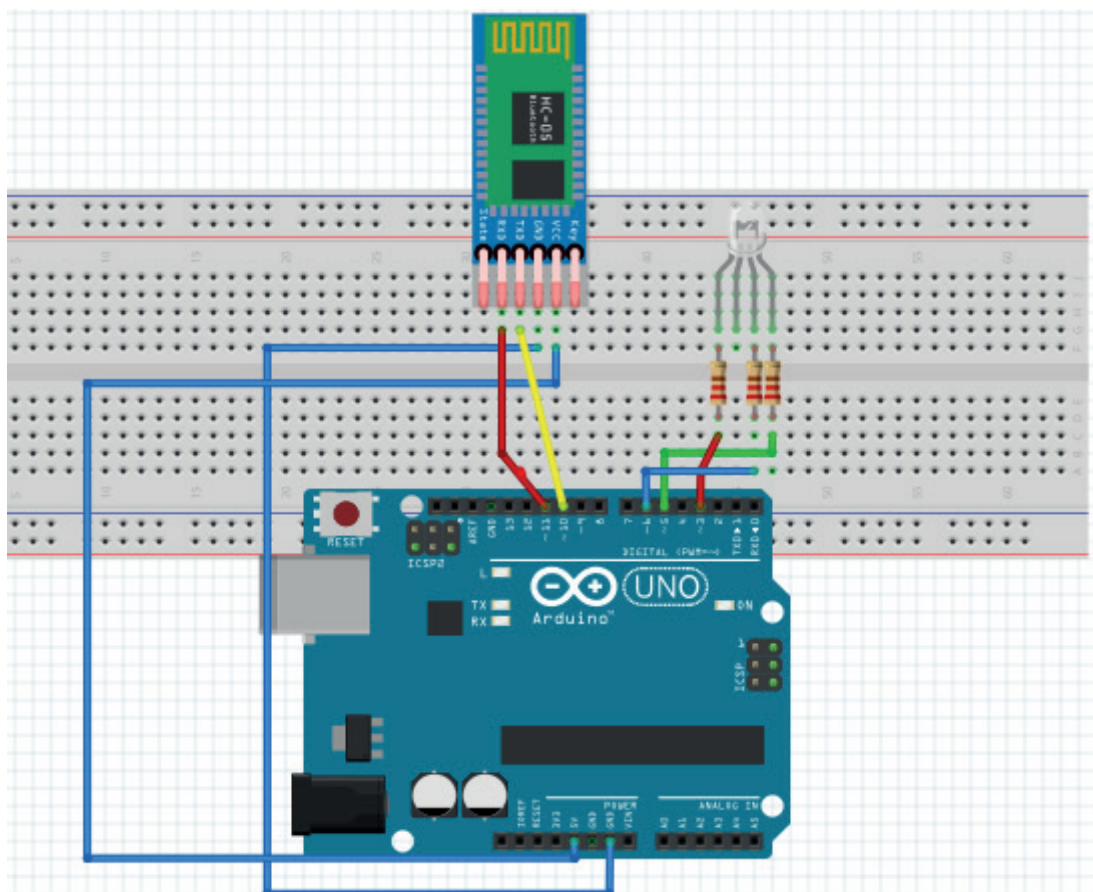
- ❖ Arduino Uno
- ❖ Module Bluetooth HC-05
- ❖ LED RGB
- ❖ Résistances 3x220 Ohms
- ❖ Jumpers
- ❖ Breadboard

Brochages des composants :

Arduino	LED RGB	HC-05
GND	GND	GND
5V	-	VCC
3	LED Rouge	-
5	LED Verte	-
6	LED Bleue	-
10	-	RX
11	-	TX

N'oubliez pas de mettre une résistance de 220 ohms ou autres entre les LEDs et l'arduino pour assurer la sécurité de vos LEDs.

Schéma de connexion

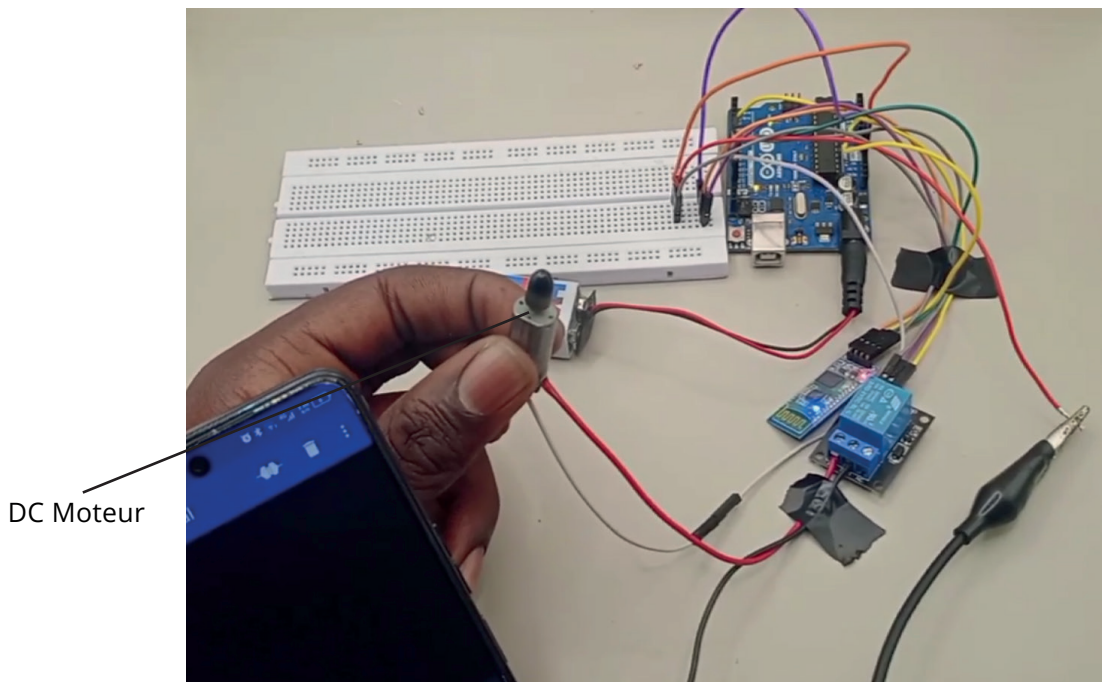


Code

Le code en entier ici [🔗](#) :

```
1 #include <SoftwareSerial.h> // Librairie pour créer une nouvelle connexion s
2 #define RED_PIN 3
3 #define GREEN_PIN 5
4 #define BLUE_PIN 6
5 SoftwareSerial BTSerial(10, 11); // RX | TX = > BT-TX=10 BT-RX=11
6 void setup()
7 {
8     Serial.begin(9600);
9     Serial.println("Enter a command:");
10    BTSerial.begin(9600); // HC-05 9600 baud
11
12    // Configurer les broches comme sorties
13    pinMode(RED_PIN, OUTPUT);
14    pinMode(GREEN_PIN, OUTPUT);
15    pinMode(BLUE_PIN, OUTPUT);
16
17    // Éteindre toutes les LEDs au démarrage
18    digitalWrite(RED_PIN, LOW);
19    digitalWrite(GREEN_PIN, LOW);
20    digitalWrite(BLUE_PIN, LOW);
21 }
22 void loop()
23 {
24     String message;
```


Projet 15 : Contrôle de moteur à courant continu via Bluetooth



Description

Dans ce projet, nous contrôlerons un moteur à courant continu avec un smartphone via Bluetooth. Ce projet est idéal pour en savoir plus sur :

- ❖ Les moteurs à courant continu
- ❖ Le module relais

Pièces requises

- ❖ Arduino UNO
- ❖ Moteur DC
- ❖ Module bluetooth HC-05
- ❖ Module relais
- ❖ Breadboard
- ❖ Jumpers

Le moteur DC :

Les moteurs à courant continu (moteurs DC) sont largement utilisés dans diverses applications en raison de leur simplicité, de leur efficacité et de leur capacité à être facilement contrôlés. Voici ce que vous devez savoir sur les moteurs DC :

Principe de Fonctionnement

- ❖ **Principe de Base** : Un moteur DC convertit l'énergie électrique en énergie mécanique par le biais de l'interaction entre les champs magnétiques générés par les bobines de l'induit (rotor) et les aimants (stator) ou les bobines du stator.
- ❖ **Commutation** : Les moteurs DC utilisent un commutateur mécanique appelé collecteur (ou commutateur) pour inverser la direction du courant dans les bobines du rotor, assurant ainsi une rotation continue.

Contrôle des Moteurs DC

- ❖ **Contrôle de la Vitesse** : La vitesse des moteurs DC peut être contrôlée en ajustant la tension d'alimentation ou en utilisant une modulation de largeur d'impulsion (PWM).
- ❖ **Contrôle du Sens de Rotation** : Le sens de rotation peut être inversé en inversant la polarité de l'alimentation.
- ❖ **Contrôle du Couple** : Le couple peut être ajusté en contrôlant le courant fourni au moteur.

Applications des Moteurs DC

- ❖ **Robots** : Utilisés pour entraîner les roues et les articulations des robots.
- ❖ **Véhicules RC (Radio-Controlled)** : Utilisés pour propulser les voitures, les bateaux et les drones.
- ❖ **Systèmes de Ventilation** : Utilisés dans les ventilateurs et les systèmes de refroidissement.
- ❖ **Automobile** : Utilisés pour des applications telles que les lève-vitres, les essuie-glaces, et les systèmes de direction assistée.
- ❖ **Électroménagers** : Utilisés dans des appareils comme les mixeurs, les perceuses, et les aspirateurs.

Le module relais :



Module relais 1 canal

Ce module relais à 1 canal permet de contrôler/commuter facilement avec un signal bas une charge élevée. La connexion est accessible grâce à des bornes à vis et peut être connectée automatiquement à une carte avec des fils. Ce module est conçu pour commuter 1 charge. Le relais est activé lorsque la tension d'entrée est à un niveau bas (0V) et désactivé à un niveau haut (5V).

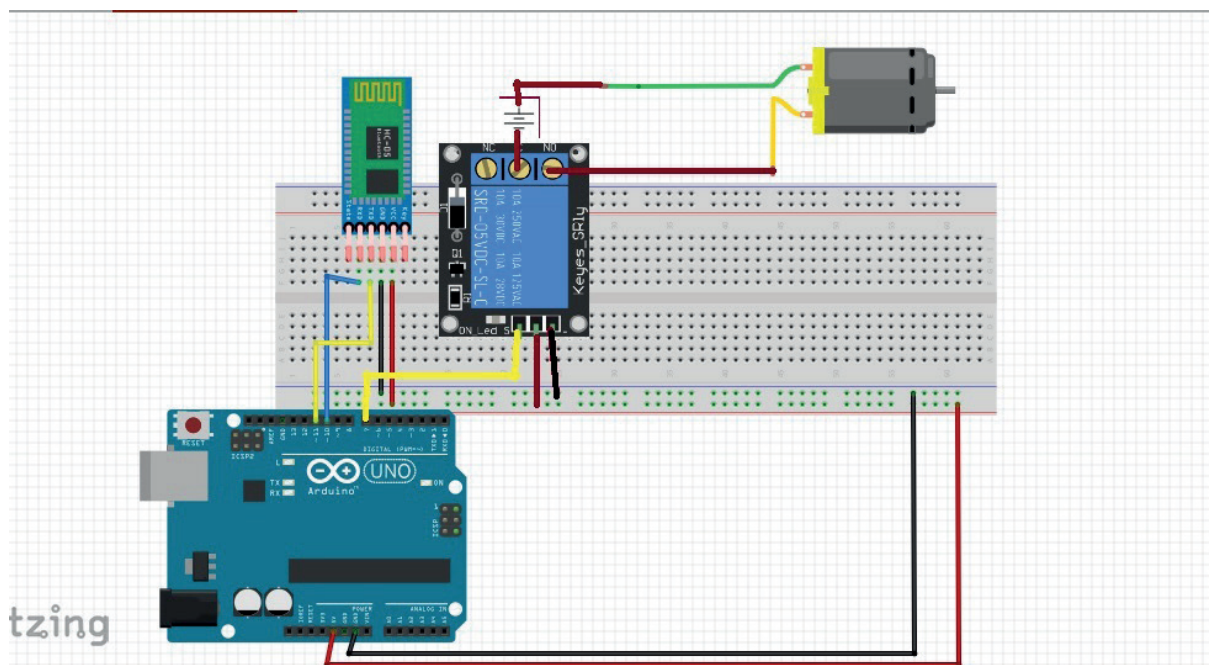
Caractéristiques

- Module relais à 1 canal, LOW
- Le relais à besoin de 15 - 20 mA pour commuter
- Type de relais : commutateur
- contacts accessibles par des bornes à vis
- LED d'indication de l'état du relais

Brochages des composants :

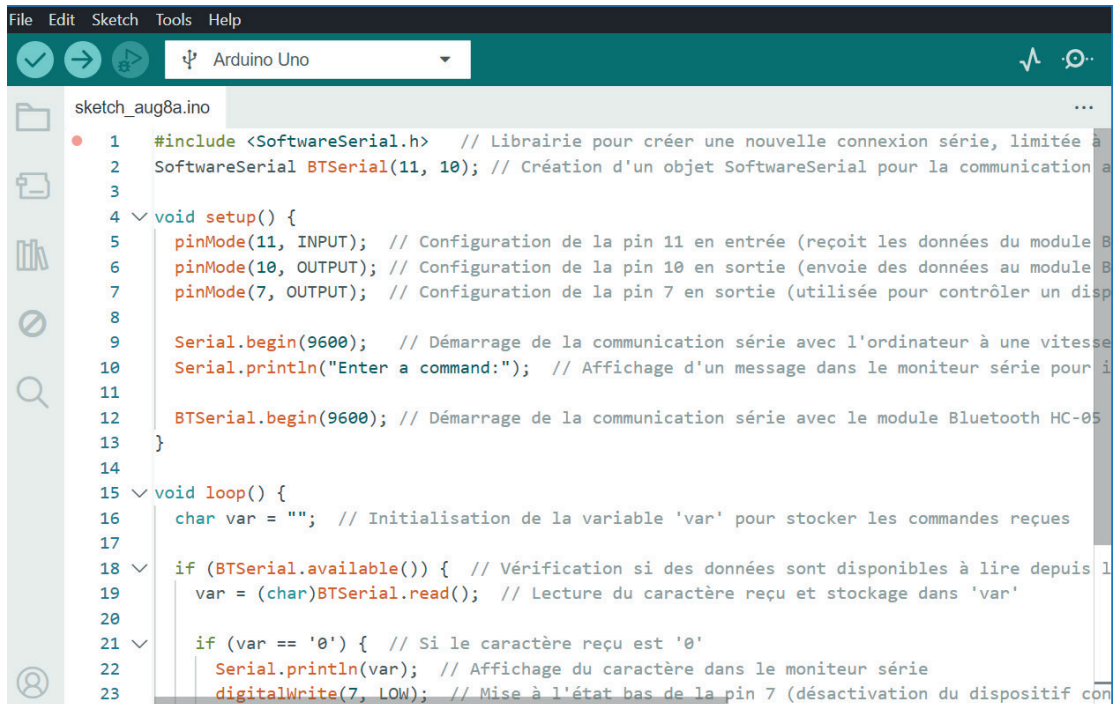
Arduino UNO	HC-05	Relais	Moteur DC
GND	GND	GND	-
5V	VCC	VCC	-
10	TXD	-	-
11	RXD	-	-
7	-	IN	-
-	-	COM	Broche +
-	-	NO	Broche -
-	-	NC	-

Schéma de connexion



Code

Le code en entier ici [🔗](#) :



```
File Edit Sketch Tools Help
sketch_aug8a.ino
1 #include <SoftwareSerial.h> // Librairie pour créer une nouvelle connexion série, limitée à
2 SoftwareSerial BTSerial(11, 10); // Création d'un objet SoftwareSerial pour la communication a
3
4 void setup() {
5   pinMode(11, INPUT); // Configuration de la pin 11 en entrée (reçoit les données du module B
6   pinMode(10, OUTPUT); // Configuration de la pin 10 en sortie (envoie des données au module B
7   pinMode(7, OUTPUT); // Configuration de la pin 7 en sortie (utilisée pour contrôler un disp
8
9   Serial.begin(9600); // Démarrage de la communication série avec l'ordinateur à une vitesse
10  Serial.println("Enter a command:"); // Affichage d'un message dans le moniteur série pour i
11
12  BTSerial.begin(9600); // Démarrage de la communication série avec le module Bluetooth HC-05
13 }
14
15 void loop() {
16   char var = ""; // Initialisation de la variable 'var' pour stocker les commandes reçues
17
18   if (BTSerial.available()) { // Vérification si des données sont disponibles à lire depuis l
19     var = (char)BTSerial.read(); // Lecture du caractère reçu et stockage dans 'var'
20
21     if (var == '0') { // Si le caractère reçu est '0'
22       Serial.println(var); // Affichage du caractère dans le moniteur série
23       digitalWrite(7, LOW); // Mise à l'état bas de la pin 7 (désactivation du dispositif con
```

La bibliothèque <SoftwareSerial.h> est déjà préinstallée dans l'IDE Arduino. Il vous faut juste télécharger le code exemple et ajouter les conditions d'allumage.

Pour la communication Android avec le module Bluetooth, j'ai utilisé le BlueTerm application. C'est totalement gratuit, il vous suffit donc d'aller sur « Play Store » et de le télécharger. Ensuite, vous connectez votre smartphone au module Bluetooth. Pensez à retirer les câbles TX et RX.

Je n'ai défini que 2 commandes pour contrôler le moteur à courant continu :

- '0' - Éteint le moteur à courant continu
- '1' - Le moteur à courant continu se met en marche

RESSOURCES

Brochages Arduino

