

DÉTECTION DE 3 NIVEAUX DE LUMINOSITÉ ET PILOTAGE DE SORTIES D'UN MICRO-CONTRÔLEUR ARDUINO

10/01/2022

Auteur(s) :

Delphine Chareyron

ENS de Lyon /
DGESCO

Antoine Bérut

Institut Lumière Matière, Université de Lyon

Publié par :

Delphine Chareyron

Résumé

Dans ce document, nous proposons un accompagnement pour débiter avec le micro-contrôleur Arduino et apprendre à piloter deux sorties numériques.

Table des matières

- [1. Fonction du montage et matériel](#)
- [2. Dimensionnement](#)
- [3. Mise en place d'une première sortie pilotée en fonction d'une entrée analogique](#)
 - [3.1 Schéma de câblage](#)
 - [3.2 Programme de pilotage de la sortie en fonction de la valeur de la tension d'entrée \$U_0\$](#)
 - [3.3 Ajout de l'affichage des tensions mesurées](#)
- [4. Pilotage de deux sorties en fonction de la réponse d'une entrée analogique](#)
 - [4.1 Schéma de câblage](#)
 - [4.2 Programme de pilotage de deux sorties en fonction de la valeur d'une entrée analogique](#)
- [5. Documentations et références](#)

Nous présentons dans cet article le pilotage d'une ou plusieurs sorties en fonction de la réponse d'un composant analogique en entrée.

Pour une première prise en main avec Arduino, on pourra consulter l'article « [Débuter avec Arduino](#) ». On trouvera des informations sur le matériel, l'alimentation, l'utilisation de la plaquette ou *breadboard*, l'installation du logiciel et le téléversement des programmes dans la carte.

1. FONCTION DU MONTAGE ET MATÉRIEL

Nous proposons ici de créer un montage dont la sortie effectue différentes fonctions selon la luminosité ambiante.

Nous repérons 3 modes de luminosité : éclairage sous une lampe de bureau, éclairage plafond de la salle (néons) et quasi-obscurité. Nous choisissons les différentes opérations de sorties, selon la valeur de l'entrée, dans le tableau suivant :

Tableau 1.

Quasi-obscurité	Éclairage plafond de la salle	Éclairage sous une lampe de bureau
Déclenchement d'un signal lumineux (LED)	Pas de lumière ni de son	Déclenchement d'un signal audio (buzzer)

Le matériel nécessaire pour les expériences proposées ici est regroupé sur la figure 1.

<https://culturesciencesphysique.ens-lyon.fr/ressource/TP-Arduino-contrôle-sorties.xml> - Version du 11/01/22

Nous utilisons un micro-contrôleur Arduino Uno ; il est tout-à-fait possible d'utiliser d'autres versions de cartes plus petites ou plus élaborées : Arduino Nano, Léonardo, etc... Nous aurons besoin d'une *breadboard* (ou plaque) sur laquelle réaliser les montages et des fils ou *jumper wires* pour connecter les éléments entre eux et quelques composants : photorésistance, résistances (240 Ω et 3,2 k Ω), une diode électroluminescente et un buzzer. Un petit « chapeau » en papier violet servira à masquer la photorésistance pour représenter la quasi-obscurité.

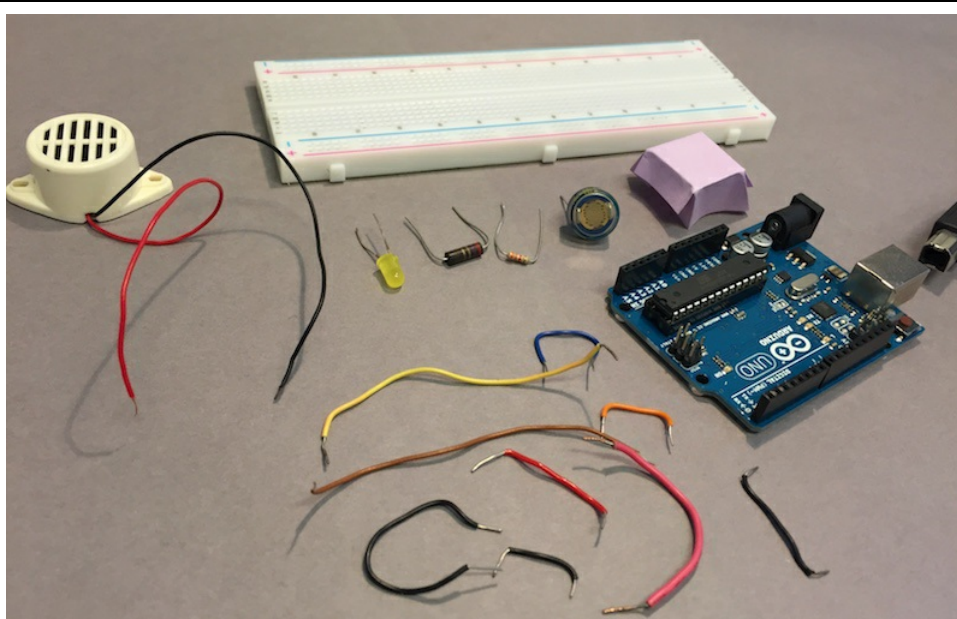


Figure 1. Matériel utilisé pour le montage

Le micro-contrôleur Arduino sera ici alimenté en 5 V via son câble USB relié à un ordinateur. Nous nous servirons aussi d'un multimètre pour vérifier les niveaux de tension et tester les connections.

2. DIMENSIONNEMENT

Nous allons utiliser une photorésistance pour "détecter" les trois niveaux d'éclairement. À l'aide d'un ohmmètre, nous mesurons la résistance de la photorésistance (R_P) sous chacun des éclairagements. Nous trouvons environ : 34 k Ω dans la quasi-obscurité ; 1,15 k Ω avec l'éclairage plafond de la salle et 170 Ω sous l'éclairage de la lampe de bureau.

Nous réalisons le montage de contrôle à l'aide d'une résistance R_0 de 3,2 k Ω , figure 2.

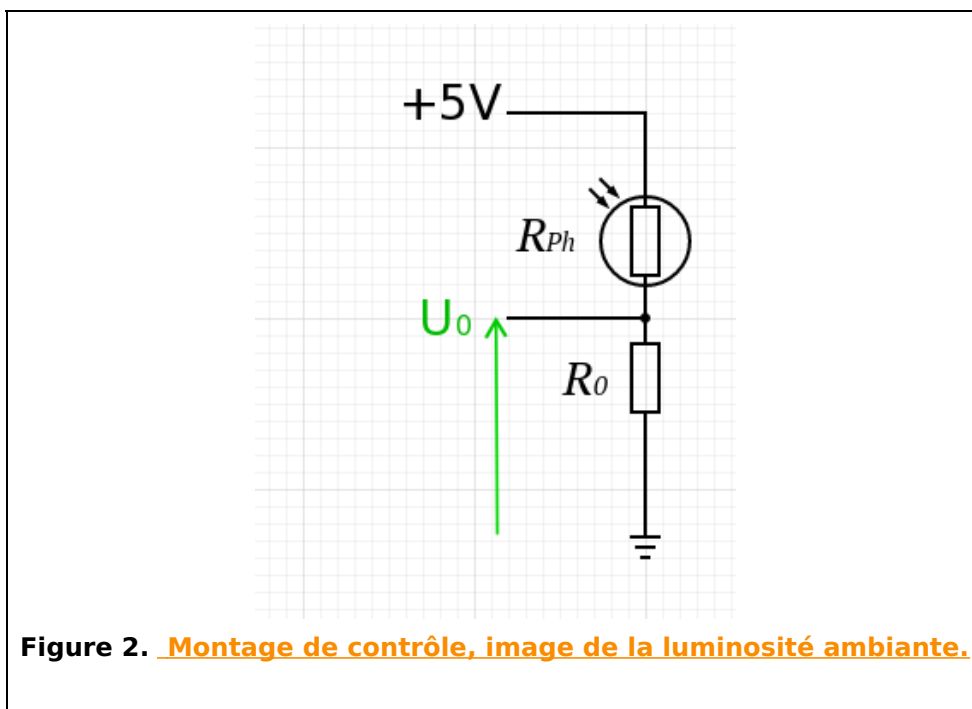


Figure 2. Montage de contrôle, image de la luminosité ambiante.

En fonction de l'éclairement la valeur R_{Ph} de la photorésistance change. La tension d'alimentation (+5 V) étant fixe, le courant appelé, et donc la tension U_0 aux bornes de R_0 prendront alors 3 valeurs différentes pour les 3 éclairagements.

On peut maintenant estimer la tension U_0 pour chaque éclairage à l'aide des mesures des résistances réalisées précédemment, puis, à l'aide du multimètre, vérifier les estimations, tableau 2.

Tableau 2. Dimensionnement du circuit de contrôle de l'éclairage

Éclairage et valeur de R_{Ph} correspondante	Quasi-obscurité $R_{Ph} \approx 34 \text{ k}\Omega$	Éclairage plafond $R_{Ph} \approx 1,15 \text{ k}\Omega$	Éclairage sous la lampe de bureau $R_{Ph} \approx 170 \Omega$
Intensité appelée estimée en prenant les valeurs de R_{Ph} et R_0 mesurées précédemment $I = \frac{5}{R_{Ph} + R_0}$	0,134 mA	1,15 mA	1,48 mA
Tension U_0 estimée $U_0 = R_0 \times I \text{ avec } R_0 = 3,2 \text{ k}\Omega$	0,430 V	3,68 V	4,74 V
Tension U_0 mesurée au voltmètre	0,470 V	3,72 V	4,76 V

Comparaison des valeurs estimées avec les valeurs mesurées de la tension de contrôle U_0 .

3. MISE EN PLACE D'UNE PREMIÈRE SORTIE PILOTÉE EN FONCTION D'UNE ENTRÉE ANALOGIQUE

Dans un premier temps, nous allons commencer par piloter une seule sortie correspondant à la détection de la quasi-obscurité.

Nous allons programmer le micro-contrôleur pour que la LED s'allume si la tension U_0 est inférieure à un certain seuil que nous choisissons ici égal à 3 V.

3.1 SCHÉMA DE CÂBLAGE

Le câblage du circuit de contrôle de l'éclairage et du pilotage de la LED est présenté sur la figure 3.

Les deux lignes (rouge (+) et bleu (masse)) de la plaquette ou *breadboard* sont connectées respectivement à la sortie alimentation +5 V et à la masse du micro-contrôleur.

La tension à mesurer U_0 , aux bornes de R_0 , est envoyée sur l'entrée analogique A_0 de la carte.

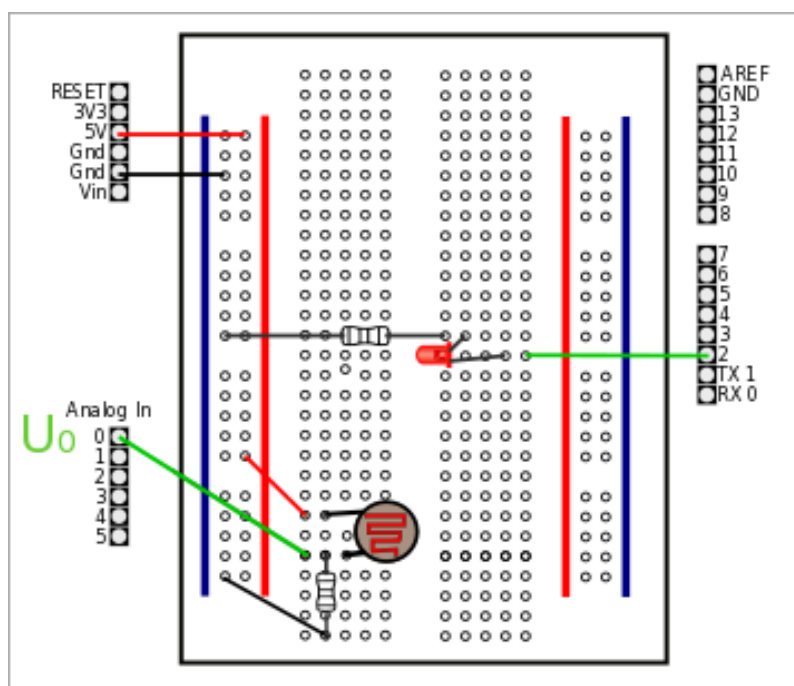


Figure 3. Schéma de câblage de pilotage de l'éclairage de la LED en fonction de la luminosité ambiante

Le contrôle est réalisé par la mesure de la tension U_0 .

Nous allons utiliser la sortie numérique 2 pour piloter l'alimentation du circuit comportant la LED (celle-ci est protégée par une résistance de 240 Ω pour limiter le courant la traversant).

3.2 PROGRAMME DE PILOTAGE DE LA SORTIE EN FONCTION DE LA VALEUR DE LA TENSION D'ENTRÉE U_0

Le programme est présenté dans la figure 4.

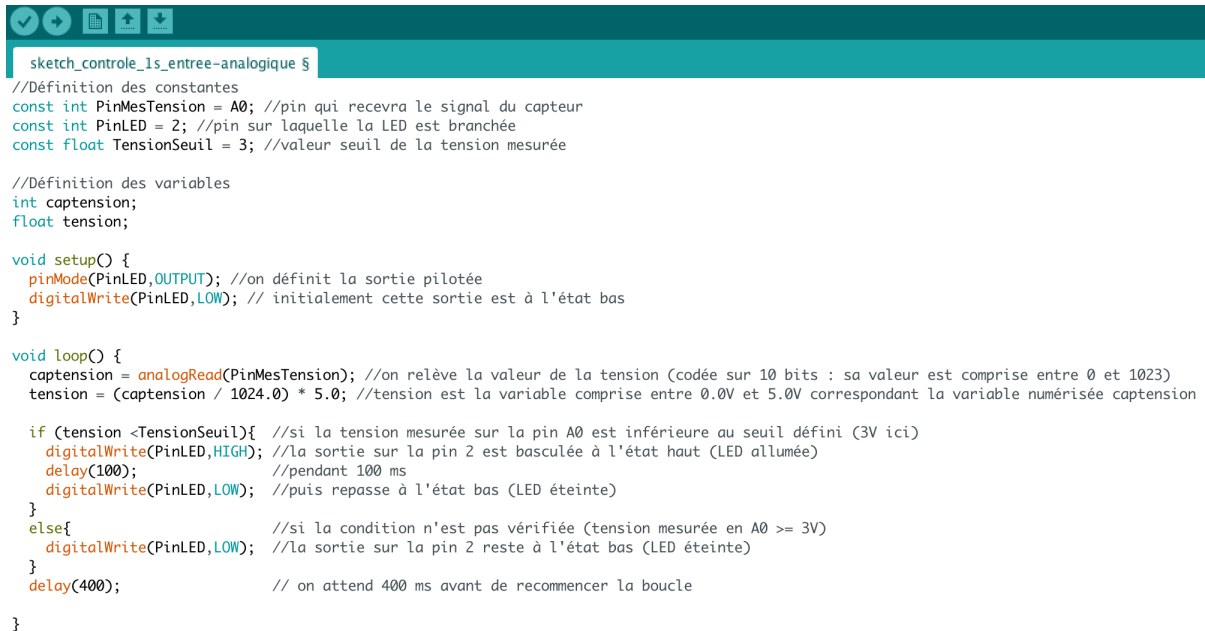
Dans un premier temps, nous définissons les constantes et variables :

- Nous appelons `PinMesTension` la variable qui stockera le nom de la *pin* A_0 du contrôleur, sur laquelle on appliquera la tension U_0 (figures 2 et 3).
- Nous appelons `PinLED` la variable qui stockera le nom du connecteur (la « *pin* ») du micro-contrôleur sur laquelle sera branchée la LED (figure 3).
- Nous appelons `TensionSeuil` la variable qui stockera la valeur seuil de tension en dessous de laquelle nous allumerons la LED (3V ici).

Ces variables sont toutes définies comme des constantes [*const*] car ce sont des variables à lecture seule : leur valeur ne sera jamais modifiée lors de l'exécution du programme. C'est en général une bonne pratique de définir en début de programme les différentes variables utilisées : cela permettra de les modifier facilement par la suite si nécessaire (par exemple, si l'on décide finalement de brancher la LED sur une autre *pin*).

Les noms de *pin* de l'Arduino sont toujours déclarés comme des valeurs entières [*int*] (*int* pour *integer*), la tension seuil est déclarée comme une variable décimale à virgule flottante [*float*].

- Nous appelons `captension` la variable dans laquelle nous allons stocker le résultat de la lecture sur le connecteur `PinMesTension`. C'est une variable numérique entière [*int*]. La carte Arduino contient un **convertisseur analogique-numérique 10 bits**, cela signifie que les tensions d'entrée comprises entre 0 et 5V vont être numérisées en valeurs entières comprises entre 0 et 1023 (2^{10} valeurs).
- Nous créons la variable `tension` représentative de la valeur analogique de `captension`, c'est-à-dire comprise entre 0 et 5 V (comme on la lirait avec un voltmètre). C'est une variable décimale à virgule flottante [*float*].



```

sketch_controle_1s_entree-analogique §
//Définition des constantes
const int PinMesTension = A0; //pin qui recevra le signal du capteur
const int PinLED = 2; //pin sur laquelle la LED est branchée
const float TensionSeuil = 3; //valeur seuil de la tension mesurée

//Définition des variables
int captension;
float tension;

void setup() {
  pinMode(PinLED,OUTPUT); //on définit la sortie pilotée
  digitalWrite(PinLED,LOW); // initialement cette sortie est à l'état bas
}

void loop() {
  captension = analogRead(PinMesTension); //on relève la valeur de la tension (codée sur 10 bits : sa valeur est comprise entre 0 et 1023)
  tension = (captension / 1024.0) * 5.0; //tension est la variable comprise entre 0.0V et 5.0V correspondant la variable numérisée captension

  if (tension < TensionSeuil){ //si la tension mesurée sur la pin A0 est inférieure au seuil défini (3V ici)
    digitalWrite(PinLED,HIGH); //la sortie sur la pin 2 est basculée à l'état haut (LED allumée)
    delay(100); //pendant 100 ms
    digitalWrite(PinLED,LOW); //puis repasse à l'état bas (LED éteinte)
  }
  else{
    digitalWrite(PinLED,LOW); //si la condition n'est pas vérifiée (tension mesurée en A0 >= 3V)
    //la sortie sur la pin 2 reste à l'état bas (LED éteinte)
  }
  delay(400); // on attend 400 ms avant de recommencer la boucle
}

```

Figure 4. Pilotage d'une sortie de la carte Arduino en fonction de la valeur de l'entrée mesurée sur la *pin* A₀

Dans la boucle d'initialisation [*setup()*] (exécutée une fois à chaque démarrage de l'Arduino), on déclare la *pin* choisie pour alimenter la LED (la *pin* 2 ici) comme une sortie numérique [*pinMode(PinLED,OUTPUT)*]. La valeur de cette sortie est initialement fixée à l'état bas (0 V, donc LED éteinte).

Dans la boucle de fonctionnement [*loop()*] (exécutée en boucle jusqu'à l'arrêt de l'Arduino), on stocke dans la variable *captension* le résultat de la lecture de la tension relevée sur la *pin* A₀.

On convertit alors la valeur numérique *captension* en valeur analogique stockée dans la variable *tension*. En utilisant la notation décimale 5.0 et 1024.0, nous forçons le calcul avec des nombres à virgule.

On réalise maintenant le test sur la valeur de la tension U_0 . Si celle-ci est inférieure à 3 V :

- La sortie numérique numéro 2 passe à l'état haut, alimentant alors le circuit comportant la LED. Elle reste alimentée pendant 100 ms.
- puis elle passe à l'état bas, la LED s'éteint.

Si la tension U_0 est supérieure à 3 V, la sortie numérique numéro 2 reste à l'état bas.

Puis la boucle recommence 400 ms plus tard [*delay(400)*].^[1]

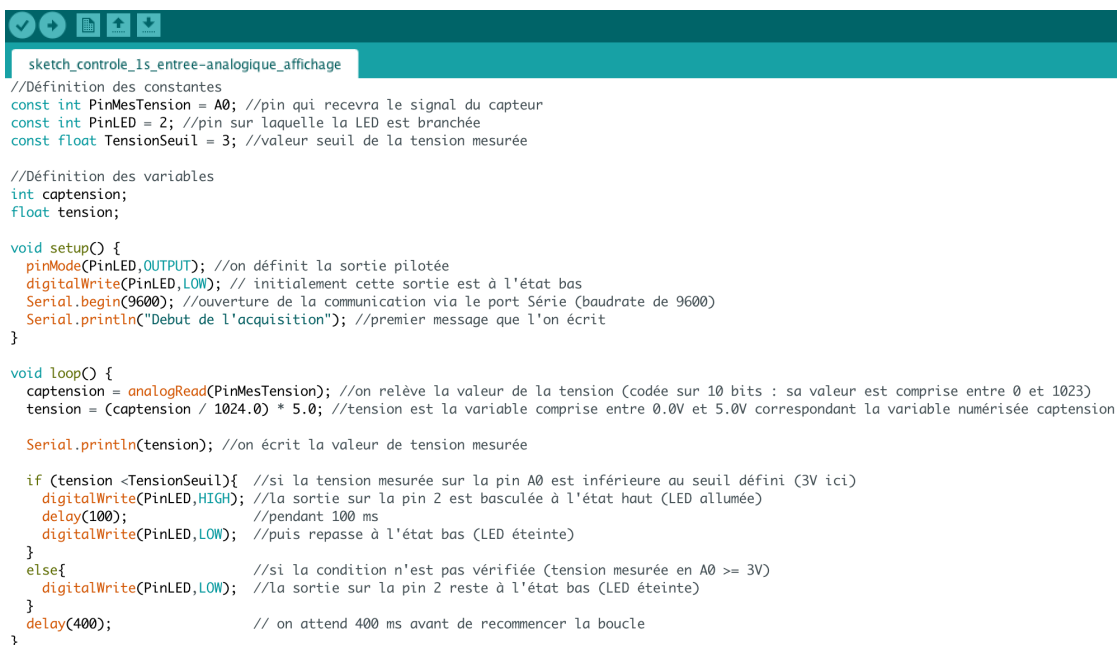
Nous compilons le script et l'envoyons dans la carte. On peut observer que dans la quasi-obscurité, la LED se met à clignoter, alors qu'elle s'éteint lorsqu'il y a plus de lumière. On retrouve ce même principe pour la détection d'allumage ou d'extinction des lampadaires publics sur la voirie.

3.3 AJOUT DE L'AFFICHAGE DES TENSIONS MESURÉES

On souhaite pouvoir accéder à la valeur mesurée de la tension U_0 . Nous allons alors modifier le script du programme afin de lire les valeurs dans la fenêtre du moniteur série.

Le code modifié est présenté sur la figure 5. Dans la boucle d'initialisation [*setup()*], on initialise la connexion par le port série en renseignant le débit de communication [*Serial.begin(9600)*]. On affiche ensuite l'indication du début de l'acquisition à l'écran.

Dans la boucle de fonctionnement [*loop()*], on affiche, à chaque répétition de la boucle, la valeur de la tension U_0 mesurée. La fonction [*Serial.println()*] permet d'afficher une valeur (ou du texte) directement suivie par un saut de ligne (pour faciliter la lecture).



```

sketch_controle_1s_entree-analogique_affichage
//Définition des constantes
const int PinMesTension = A0; //pin qui recevra le signal du capteur
const int PinLED = 2; //pin sur laquelle la LED est branchée
const float TensionSeuil = 3; //valeur seuil de la tension mesurée

//Définition des variables
int captension;
float tension;

void setup() {
  pinMode(PinLED,OUTPUT); //on définit la sortie pilotée
  digitalWrite(PinLED,LOW); // initialement cette sortie est à l'état bas
  Serial.begin(9600); //ouverture de la communication via le port Série (baudrate de 9600)
  Serial.println("Debut de l'acquisition"); //premier message que l'on écrit
}

void loop() {
  captension = analogRead(PinMesTension); //on relève la valeur de la tension (codée sur 10 bits : sa valeur est comprise entre 0 et 1023)
  tension = (captension / 1024.0) * 5.0; //tension est la variable comprise entre 0.0V et 5.0V correspondant la variable numérisée captension

  Serial.println(tension); //on écrit la valeur de tension mesurée

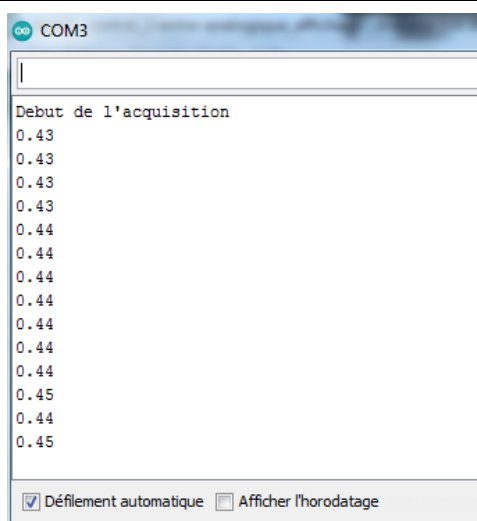
  if (tension < TensionSeuil){ //si la tension mesurée sur la pin A0 est inférieure au seuil défini (3V ici)
    digitalWrite(PinLED,HIGH); //la sortie sur la pin 2 est basculée à l'état haut (LED allumée)
    delay(100); //pendant 100 ms
    digitalWrite(PinLED,LOW); //puis repasse à l'état bas (LED éteinte)
  }
  else{ //si la condition n'est pas vérifiée (tension mesurée en A0 >= 3V)
    digitalWrite(PinLED,LOW); //la sortie sur la pin 2 reste à l'état bas (LED éteinte)
  }
  delay(400); // on attend 400 ms avant de recommencer la boucle
}

```

Figure 5. Pilotage d'une sortie de la carte Arduino en fonction de la valeur de l'entrée mesurée sur la pin A₀ et affichage dans la fenêtre du moniteur série

Lorsque la photorésistance est sous le chapeau, dans la quasi-obscurité, la valeur de la tension est mesurée, figure 6.

On retrouve bien une valeur proche de la tension attendue ou mesurée au voltmètre. Sur la figure 7, sous l'éclairage néon, on note un peu plus de fluctuations de luminosité propres à ce type d'éclairage. Indépendamment du type d'éclairage, les variations de tension observées sont aussi dues au fait que l'alimentation en 5V par le port USB de l'ordinateur n'est pas stabilisée. On devrait avoir de meilleurs résultats avec une tension de référence plus stable, pour une discussion plus en détail sur le sujet, on pourra voir par exemple les montages proposés au lien suivant : <http://www.skillbank.co.uk/arduino/measure.htm>.



```

COM3
Debut de l'acquisition
0.43
0.43
0.43
0.43
0.44
0.44
0.44
0.44
0.44
0.44
0.45
0.44
0.45

```

Figure 6. Affichage de la valeur de la tension U₀ dans la fenêtre du moniteur série dans une quasi-obscurité

Maintenant, sous un éclairage plafond de la salle (néons), la tension U₀ qui s'affiche est présentée figure 7.

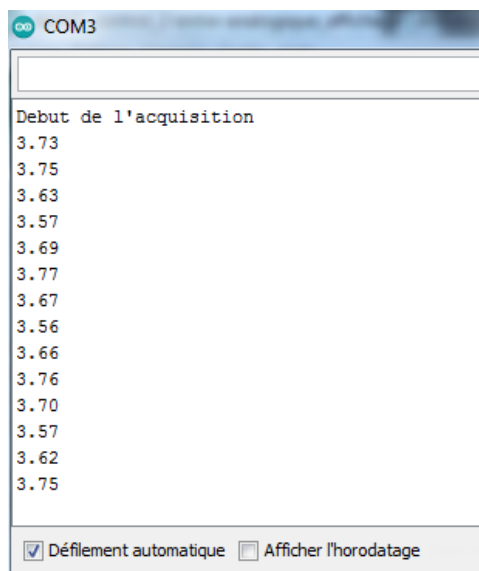


Figure 7. Affichage de la valeur de la tension U_0 dans la fenêtre du moniteur série sous un éclairage néon

4. PILOTAGE DE DEUX SORTIES EN FONCTION DE LA RÉPONSE D'UNE ENTRÉE ANALOGIQUE

4.1 SCHÉMA DE CÂBLAGE

On souhaite maintenant piloter une seconde sortie numérique, pour émettre un signal sonore au-dessus d'un seuil d'intensité lumineuse.

On rajoute au circuit précédent un buzzer alimenté sur la sortie numérique numéro 3, figure 8.

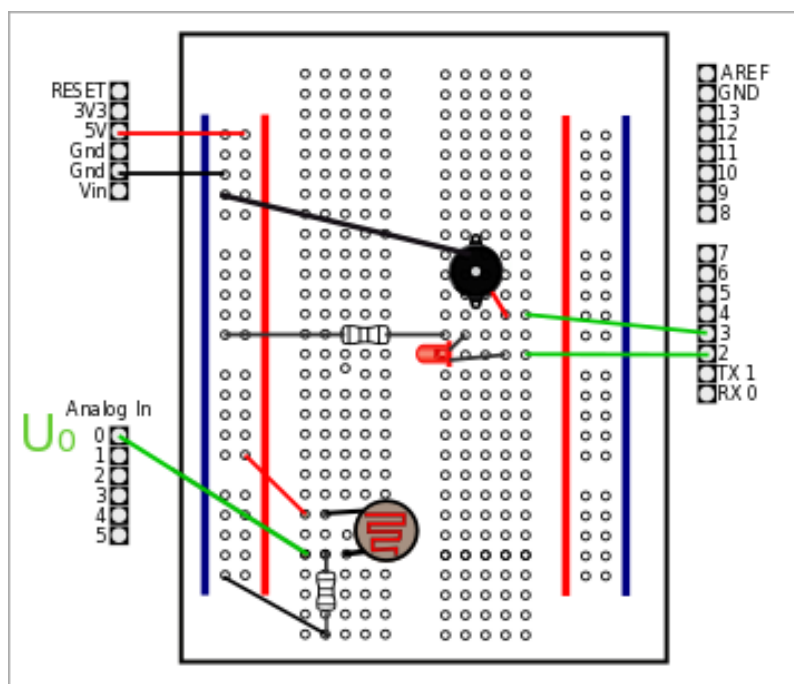
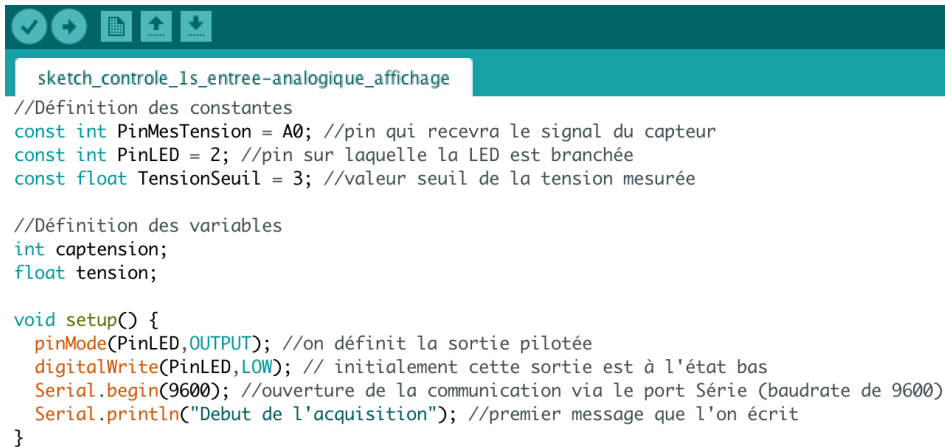


Figure 8. Schéma de câblage pour le pilotage des pins 2 et 3 en fonction de la tension d'entrée U_0

4.2 PROGRAMME DE PILOTAGE DE DEUX SORTIES EN FONCTION DE LA VALEUR D'UNE ENTRÉE ANALOGIQUE

Nous modifions le code précédent en définissant les deux sorties utilisées (pins 2 et 3) et en les initialisant à l'état bas, figure 9.



```

sketch_controle_1s_entree-analogique_affichage
//Définition des constantes
const int PinMesTension = A0; //pin qui recevra le signal du capteur
const int PinLED = 2; //pin sur laquelle la LED est branchée
const float TensionSeuil = 3; //valeur seuil de la tension mesurée

//Définition des variables
int captension;
float tension;

void setup() {
  pinMode(PinLED,OUTPUT); //on définit la sortie pilotée
  digitalWrite(PinLED,LOW); // initialement cette sortie est à l'état bas
  Serial.begin(9600); //ouverture de la communication via le port Série (baudrate de 9600)
  Serial.println("Debut de l'acquisition"); //premier message que l'on écrit
}

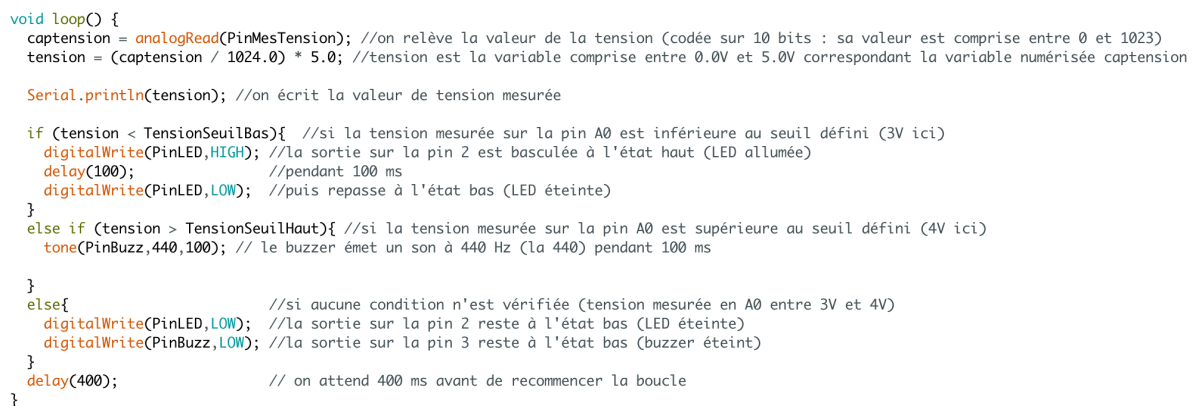
```

Figure 9. Boucle d'initialisation et de fonctionnement du programme de pilotage de deux sorties de la carte Arduino en fonction de la valeur de l'entrée mesurée sur la *pin* A₀

Dans la boucle du code (figure 10), nous rajoutons un test sur la valeur de la tension U_0 mesurée. Si celle-ci est supérieure ou égale à 4 V, la sortie 3 passe au niveau haut. Afin de mettre en vibration la membrane du buzzer, il est nécessaire que celui-ci soit alimenté par un signal alternatif. Nous utilisons la fonction *tone* qui permet de générer un signal créneau de fréquence et durée déterminée. Ici nous choisissons une fréquence de 440 Hz et une durée de 100 ms [*tone*(PinBuzz, 440, 100)].

Cette fonction peut être utilisée sur toutes les *pins* numériques compatibles PWM (sur un Arduino Uno, il s'agit des *pins* 3, 5, 6, 9, 10 et 11, marquées d'un petit symbole « ~ »).

De la même manière que précédemment, la boucle est répétée toutes les 400 ms.



```

void loop() {
  captension = analogRead(PinMesTension); //on relève la valeur de la tension (codée sur 10 bits : sa valeur est comprise entre 0 et 1023)
  tension = (captension / 1024.0) * 5.0; //tension est la variable comprise entre 0.0V et 5.0V correspondant la variable numérisée captension

  Serial.println(tension); //on écrit la valeur de tension mesurée

  if (tension < TensionSeuilBas){ //si la tension mesurée sur la pin A0 est inférieure au seuil défini (3V ici)
    digitalWrite(PinLED,HIGH); //la sortie sur la pin 2 est basculée à l'état haut (LED allumée)
    delay(100); //pendant 100 ms
    digitalWrite(PinLED,LOW); //puis repasse à l'état bas (LED éteinte)
  }
  else if (tension > TensionSeuilHaut){ //si la tension mesurée sur la pin A0 est supérieure au seuil défini (4V ici)
    tone(PinBuzz,440,100); // le buzzer émet un son à 440 Hz (la 440) pendant 100 ms
  }
  else{ //si aucune condition n'est vérifiée (tension mesurée en A0 entre 3V et 4V)
    digitalWrite(PinLED,LOW); //la sortie sur la pin 2 reste à l'état bas (LED éteinte)
    digitalWrite(PinBuzz,LOW); //la sortie sur la pin 3 reste à l'état bas (buzzer éteint)
  }
  delay(400); // on attend 400 ms avant de recommencer la boucle
}

```

Figure 10. Affichage de la valeur de la tension U_0 dans la fenêtre du moniteur série sous l'éclairage de la lampe de bureau

Lorsque la lampe de bureau est allumée, on retrouve une tension U_0 proche de 4,80 V, figure 11.

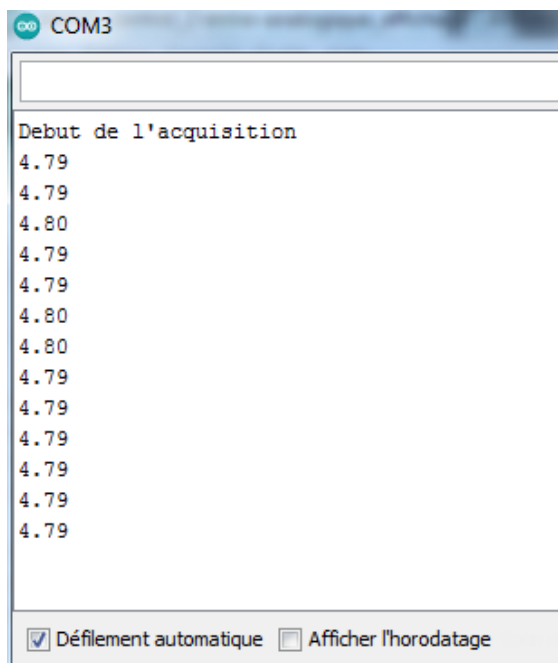
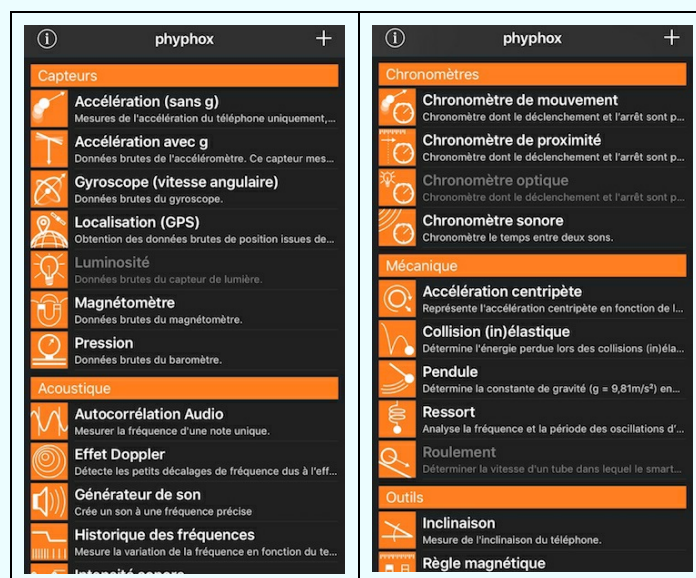
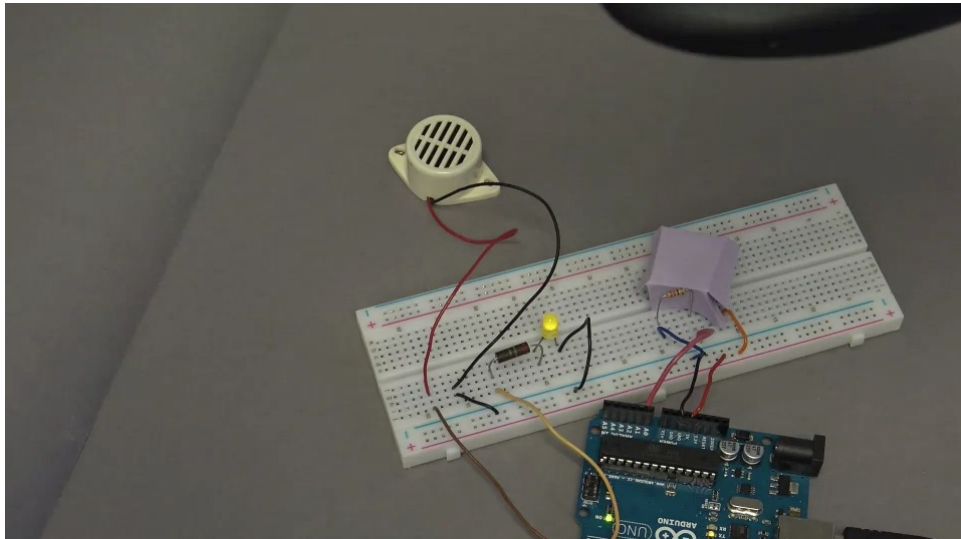


Figure 11. Affichage de la valeur de la tension U_0 dans la fenêtre du moniteur série sous l'éclairage de la lampe de bureau

Il est possible de vérifier le bon fonctionnement du programme avec différents capteurs. Nous utilisons dans la vidéo, figure 12, un smartphone sur lequel nous avons téléchargé la suite [Phyphox](#).

[Phyphox/1, 2/](#) permet d'utiliser les capteurs déjà présents dans les smartphones pour recueillir des données, les tracer et les exporter simplement.





Format mp4Format webm

Source - © 2021 CultureSciences-Physique

Figure 12. La LED clignote lorsque l'intensité lumineuse est moindre, et lorsque la luminosité est très importante un signal sonore est déclenché.

À l'aide du chronomètre sonore de la boîte à outils Phyphox, nous vérifions que pour la durée totale de la boucle (500 ms = 100 ms pour le test son ou lumière + 400 ms entre chaque boucle), nous retrouvons une fréquence de 2 Hz.

PROGRAMMES ARDUINO :

Télécharger le programme pour le pilotage d'une sortie de la carte Arduino en fonction de la valeur de l'entrée mesurée sur la *pin* A_0 : « [sketch_controle_1s_entree_analogique.ino](#) ».

Télécharger le programme pour le pilotage d'une sortie de la carte Arduino en fonction de la valeur de l'entrée mesurée sur la *pin* A_0 et affichage dans la fenêtre du moniteur série :
« [sketch_controle_1s_entree-analogique_affichage.ino](#) ».

Télécharger le programme pour le pilotage de deux sorties de la carte Arduino en fonction de la valeur de l'entrée mesurée sur la *pin* A_0 et affichage dans la fenêtre du moniteur série :
« [sketch_controle_2s_entree-analogique_affichage.ino](#) ».

5. DOCUMENTATIONS ET RÉFÉRENCES

- [1] [Phyphox.org](#)
- [2] [Traduction de l'application Phyphox](#), La Physique Autrement
- [3] [Langage Arduino](#), Arduino.cc

Pour citer cet article :

Détection de 3 niveaux de luminosité et pilotage de sorties d'un micro-contrôleur Arduino, Delphine Chareyron et Antoine Bérut, janvier 2022. CultureSciences Physique - ISSN 2554-876X,
<https://culturesciencesphysique.ens-lyon.fr/ressource/TP-Arduino-controle-sorties.xml>

^[1] Si l'on prévoit d'avoir à changer la durée, ce paramètre pourrait également être déclaré dans les constantes au début du programme [`const int Delai = 400 ;`] et appelé sous la forme [`delay(Delai)`].