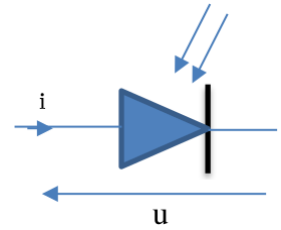


Photodiode

Objectifs : Étudier les caractéristiques d'une photodiode en fonction de l'intensité lumineuse, et s'en servir de capteur.

Compétences expérimentales exigibles : Mettre en œuvre un capteur optique.



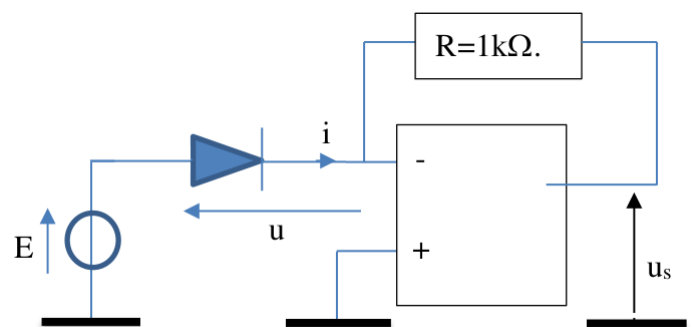
Une diode (schéma ci-dessus) est un dipôle non-linéaire qui laisse passer le courant en *régime passant* ($u > 0, i > 0$), et l'empêche de passer en *régime bloqué* ($u < 0, i \approx 0$). En réalité, il existe un *courant inverse* en régime bloqué, de valeur très faible. Dans une photodiode, on exploite ce courant inverse dont l'intensité dépend du flux lumineux reçu (car les photons absorbés génèrent des paires électron-trou). L'amplification de ce courant permet alors de s'en servir comme capteur pour mesurer l'intensité lumineuse, ou pour commander une action en fonction de l'intensité lumineuse.

I. Utilisation en capteur d'intensité lumineuse

• MANIP 1 : Mesure du courant inverse via un convertisseur courant-tension

Réaliser le montage ci-contre, dans lequel la tension constante E sera produite par un GBF. Attention, on s'assurera que la diode est montée en inverse ^a

a. Pour repérer le sens passant on connecte préalablement la photodiode au multimètre placé en Ohmmètre, symbole diode, puis appuyer sur F1 et dans le menu s'affichant en bas, choisir diode. Celui-ci mesure une tension de l'ordre de 0,6V dans le sens passant et donne l'indication Overflow dans le sens bloqué.



- Commencer par E quasi-nul, puis le faire décroître (donc augmenter en valeur absolue) de telle sorte que $E < 0$ pour une polarisation en inverse, ce qui donne $u_s = Ri > 0$.
Si $u < 0$ c'est que la diode est montée dans le sens direct. En ce cas u ne dépasse pas 0,6 V et le courant augmente rapidement. Il faut alors inverser les bornes de la diode.
- Faire varier E de 0 à -5 V (que l'on gardera pour la suite). Mesurer u_s . En déduire le courant inverse.

• MANIP 2 : Effet de l'éclairement sur le courant inverse

- Éclairer la photodiode par une lanterne LED munie d'un potentiomètre pour faire varier l'intensité lumineuse, dont on retirera le verre dépoli.
- Focaliser le faisceau sur la photodiode de façon à obtenir un maximum d'intensité.
- On dispose d'un *luxmètre* qui donne l'intensité lumineuse en Lux ^a. Montrer que le courant inverse est proportionnel à l'intensité lumineuse et mesurer le facteur de proportionnalité.

a. 1 Lux (lx, puissance surfacique ou éclairement) = 1 lumen/m² (lm.m⁻²), 1 lumen (puissance) = 1 candela × 1 steradian (1cd.sr), 1 candela = unité fondamentale d'éclat lumineux (ou intensité lumineuse). Ces unités appartiennent au domaine de la *photométrie visuelle*, c'est-à-dire la photométrie basée sur la perception de l'œil humain. Par définition, la candela correspond à une perception par un œil moyen d'une lumière monochromatique de référence ($\nu = 5,40 \times 10^{14}\text{ Hz}$, $\lambda = 555\text{ nm}$ et puissance par stéradian $\frac{1}{683}\text{ W.sr}^{-1}$ fixées). Pour une fréquence différente, la même sensation d'éclat correspond à une puissance par stéradian différente). Le lux et le lumen ne peuvent donc pas être converties de façon simple et systématique en une puissance surfacique et une puissance respectivement. Toutefois on peut s'en satisfaire pour une approche énergétique à condition de travailler avec une source lumineuse fixée.

• MANIP 3 : Effet de l'éclairement sur la caractéristique

À faire uniquement s'il reste du temps après les manipulations ci-dessous...

- Relever et tracer la caractéristique courant tension $i = f(u)$. Il faut donc explorer les deux parties $u < 0$ et $u > 0$. Dans ce cas il faut surveiller l'intensité qui ne doit pas dépasser 100 mA.
- Changer d'intensité lumineuse et tracer la nouvelle caractéristique.

II. Réalisation d'un capteur d'intensité lumineuse

On veut réaliser un dispositif qui déclenche une guirlande de Noël (ici une DEL) si la luminosité est trop faible. Nous utilisons une carte Arduino pour réaliser la tâche d'allumer la DEL. Les éléments nécessaires à l'utilisation de la carte sont rappelés en annexe. Notons qu'il faut impérativement que u_s soit positif et compris entre 0 et 5 V, donc surveiller le signe de la tension E appliquée et le sens de branchement de la diode. Pour éviter de dépasser 5 V on intercalera une résistance de 1 k Ω et une diode Zener entre la sortie u_s et l'entrée de la carte.

• MANIP 4 : Commande de l'allumage d'une DEL

- En suivant les instructions de l'annexe, alimenter la carte ARDUINO, la paramétrer et implémenter le programme (*sketch*).
- Connecter la carte Arduino, sur la bonne entrée afin de lire la tension aux bornes de la diode Zener (ne pas oublier la masse).
- Lancer le programme. Pour voir à l'écran les affichages des valeurs lues (`print`) il faut cliquer sur l'icône située en haut à droite. Le résultat est donné en bit entre 0 et 1024.
- Ajouter une instruction pour que s'affiche l'intensité en Lux. Attention il faudra convertir V en flottant : `float(V)`.
- Brancher la DEL sur la bonne sortie. Attention celle-ci doit être mise en série avec une résistance de 100 Ω pour ne pas qu'elle grille.
- Vérifier que la DEL s'allume lorsque la lumière est occultée.

III. ANNEXE : Réalisation d'un automate avec ARDUINO

III.1. Carte Arduino

Arduino est la marque d'une plateforme de prototypage open-source qui permet aux utilisateurs de créer des objets électroniques interactifs à partir de cartes électroniques matériellement libres sur lesquelles se trouve un *microcontrôleur*¹. Le microcontrôleur peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses. Le langage utilisé est le C++ avec des bibliothèques associées. Un logiciel d'environnement de programmation open-source est fourni pour cela (Arduino IDE). La carte Arduino peut être utilisée pour construire des objets interactifs indépendants (prototypage rapide), ou bien peut être connecté à un ordinateur pour communiquer avec ses logiciels. Les différentes versions des Arduino fonctionnent sous le même principe général, décrit succinctement ci-dessous.

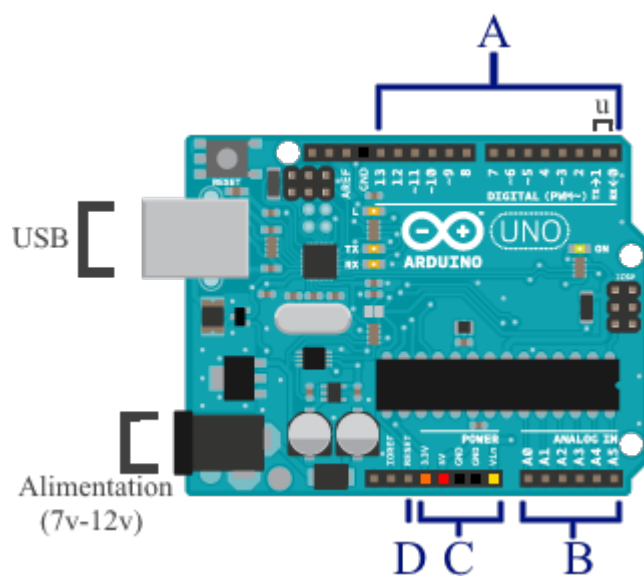


FIGURE 1 – Carte Arduino UNO (Par Sad-Cloud – Travail personnel, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=37290368>).

- A : 14 broches dites *numériques* c'est-à-dire à fonctionnement binaire (1 ou 0, HIGH ou LOW en anglais); elles offrent en sortie du 5V et acceptent en entrée du 5V sur le même principe. Les broches indiquant le symbole $\sim$ permettent aussi une sortie « pseudo analogique » de type PWM^a. Fonctions associées : `digitalWrite()` et `digitalRead()`, et pour les ports PWM `analogWrite()`.
- B : 6 broches dites *analogiques* (A0–A5), reçoivent des tensions entre 0V et 5V, converties en entiers naturels par le CAN sur 10 bits, pour des fréquences de 10kHz maximum. Fonction `analogRead()`.
- C : les différentes broches d'**alimentation en sortie** (où la carte sert de générateur de tension pour un élément externe) : 5V, 3.3V, la masse (GND), et V_{in} reliée à l'alimentation de la carte (7 V-12 V).

^a. Pulse Width Modulation, ou Modulation de Largeur d'Impulsions, technique couramment utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits à fonctionnement binaire.

Lorsqu'elle est reliée à l'ordinateur par le port USB, la carte est alimentée par ce biais en 5V.



Toute erreur de polarité (signe de la tension appliquée) ou dépassement de la valeur nominale (5V) peut conduire à la **destruction de la carte**! On prendra donc soin de s'assurer que les signaux appliqués sont bien dans l'intervalle attendu.

III.2. Environnement de programmation Arduino

L'interface de programmation du microcontrôleur s'ouvre (Fig. 2) lorsque l'on clique sur un fichier de type `programme.ino` (ci-après appelé *sketch*), après avoir branché la carte à l'ordinateur via le port USB². On commence par vérifier le choix du type de carte Arduino (ici UNO en ce qui nous concerne) dans le menu

Outils -> Type de carte

1. Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties. Il constitue une alternative plus accessible par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels, avec de nombreuses applications notamment dans la domotique, la robotique et plus généralement les systèmes embarqués.

2. Au branchement, la diode ON située à côté du nom UNO doit s'allumer.

ainsi que le port proposé pour la connection :

Outils -> Port série

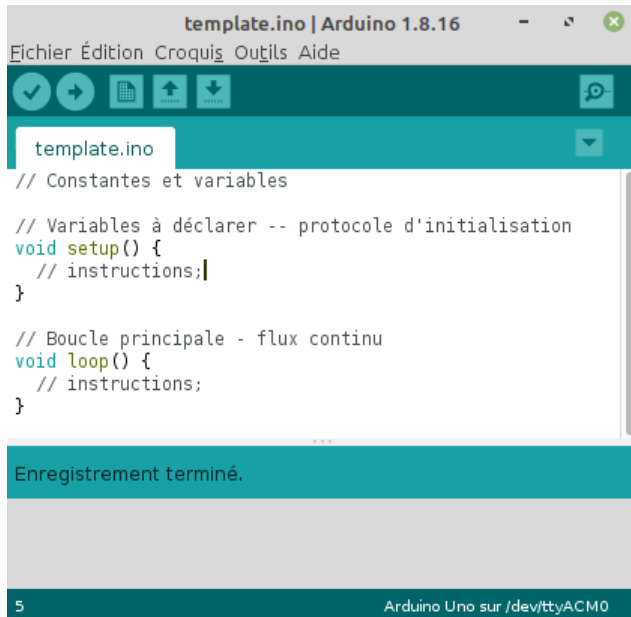


FIGURE 2 – Interface Arduino IDE et structure type d'un programme.



: Bouton pour compiler et téléverser le sketch dans le microcontrôleur.

Pour tester le bon fonctionnement de la carte on peut ouvrir puis téléverser le sketch **Blink**, que l'on trouvera ici :

Fichiers -> Exemples -> 01.Basics -> Blink et qui a pour effet de faire clignoter la LED L.

La structure générale d'un sketch Arduino est illustrée ci-contre. Après la définition éventuelle de quelques constantes,

- une première procédure d'initialisation `setup()` est exécutée une seule fois ;
- une seconde procédure `loop()` est exécutée en boucle indéfiniment, c'est-à-dire jusqu'à l'arrêt de la carte, ou le téléversement d'un nouveau sketch.

III.3. Installation du code C++ dans le microcontrôleur (sketch)

On commencera par téléverser, à l'aide de l'interface Arduino IDE, le programme du microcontrôleur ci-dessous (à ne pas modifier), qui demande à la carte de lire le signal d'intensité lumineuse, puis d'allumer ou non la DEL en fonction d'un seuil qu'il est possible d'ajuster.

```

1  int V=0; // déclare la variable V comme un entier et l'initialise à 0.
2
3  void setup() { // void désigne une procédure. setup() met en place les entrée-sorties et variables associées.
4    pinMode(A0,INPUT); // définit le pin A0 comme une entrée.
5    pinMode(7,OUTPUT); // définit le pin 7 comme une sortie.
6    Serial.begin(9600); // définit la vitesse de transmission de données entre la carte et l'ordinateur.
7  }
8
9  void loop() { // Emplacement du code principal : séquence d'instructions à exécuter de façon répétitive.
10   V=analogRead(0); // acquiert la tension de l'entrée analogique A0, la convertit sur 10 bits par un CAN
    (valeur entre 0 et 1023), et l'affecte à la variable V.
11   Serial.println(V); // affiche la valeur de la variable V sur l'écran du PC, puis effectue un retour à la ligne.
12   if (V<100) {digitalWrite(7,1);} // affecte la valeur +5V à la sortie 7.
13   else {digitalWrite(7,0);} // affecte la valeur 0V à la sortie 7.
14   Serial.print("\t"); // facultatif, ajoute une tabulation dans l'affichage.
15   delay(1); // ajoute une temporisation (attente) de 500 ms
16 }
```

On note les différences avec python : les instructions sont séparées par des ; il n'y a pas d'indentation obligatoire. Les blocs d'instructions sont bornés par { }.