Produit : Candélabre Intelligent

1. Problématique et performances attendues

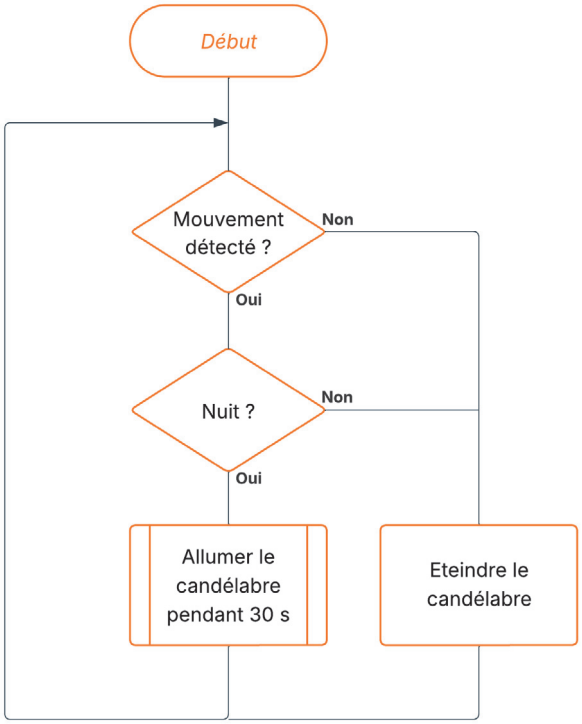
En éclairage public, les lampes à décharge électrique sont « à haute pression ». Le moyen le plus efficace pour obtenir une lumière blanche. Une lampe à décharge électrique pour l'éclairage public utilise essentiellement deux éléments sous leur forme gazeuse : le mercure et le sodium. La lampe à vapeur de sodium émet une lumière jaune très vive. Elle est caractérisée par une efficacité lumineuse très élevée et un indice de rendu des couleurs (IRC) faible.

Aujourd'hui un candélabre urbain doit fournir un éclairage efficace, homogène et sécurisé tout en consommant peu d'énergie, souvent grâce à la technologie LED. Il doit être durable, et limiter son impact environnemental (pollution lumineuse et faible consommation énergétique). Automatiser un candélabre urbain permet d'adapter l'éclairage en temps réel selon la présence, l'heure ou la luminosité ambiante, ce qui optimise la consommation d'énergie, augmente la sécurité des usagers, et limite la pollution lumineuse.

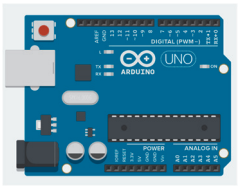
La ville de Versailles veut équiper chaque candélabre de capteurs pour collecter des données (mouvement, luminosité, météo, ...). Éventuellement installer un contrôleur afin d'ajuster l'intensité lumineuse en fonction des informations reçues et utiliser une plateforme logicielle pour le pilotage de zones. L'étude ici présente porte sur l'intégration de capteurs afin de piloter un candélabre d'éclairage urbain.

Dans le cadre de la mise en place de candélabres intelligents, les ingénieurs informaticiens et électroniciens se sont fixés un cahier des charges correspondant à l'algorigramme suivant et au choix du matériel ci-dessous pour réaliser une maquette de fonctionnement :

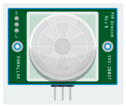
Algorigramme de fonctionnement



Carte Arduino UNO :



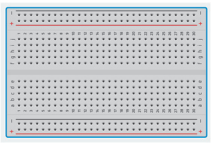
Capteur de mouvement infrarouge :



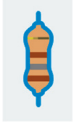
Photorésistance :

Une DEL blanche :

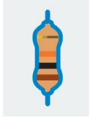
Une platine d'expérimentation :



Une résistance de 180 Ω :

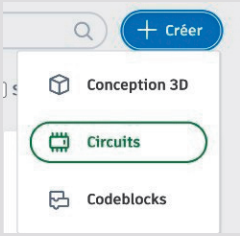


Une résistance de 10 kΩ :

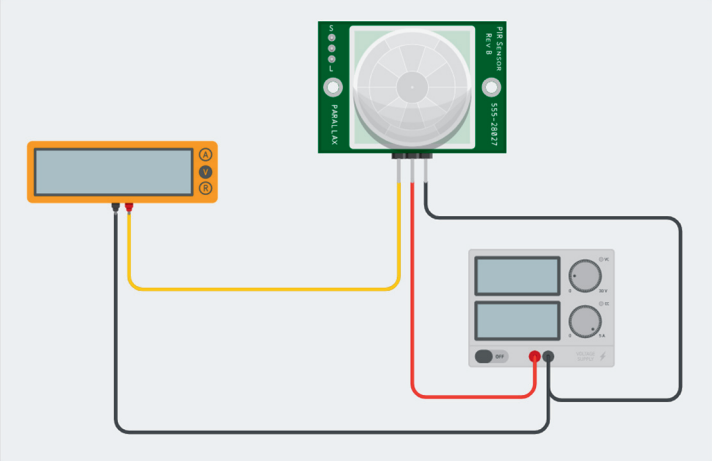


2. Simulation

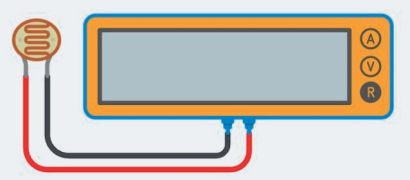
- Ouvrir le site **www.tinkercad.com** et utiliser le nom d'utilisateur fourni. Créer un nouveau circuit :



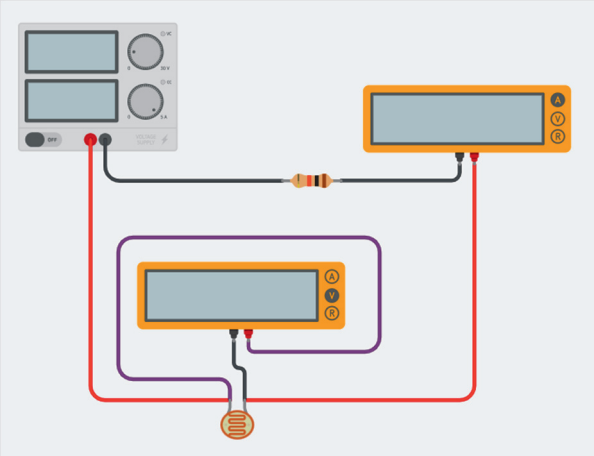
- Pour tester un capteur PIR réaliser le montage suivant (Multimètre, capteur PIR et bloc d'alimentation). Simuler un mouvement et observer l'affichage sur l'écran du multimètre.



- Pour tester une photorésistance, on branche un multimètre que l'on passe en mode R directement aux bornes de celle-ci :

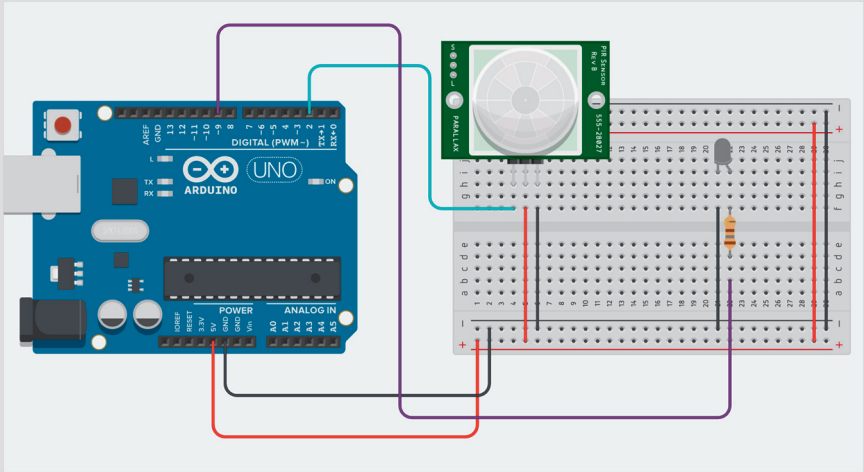


- Pour mesurer la tension et le courant aux bornes de la photorésistance, réaliser le montage suivant (Mode V pour voltmètre et mode A pour ampèremètre), en prenant une résistance de 1 kΩ :



3. Conception

- Branchement d'un capteur de mouvement, d'une DEL, à une carte Arduino à l'aide d'une platine d'essai (la résistance fait 180 Ω):



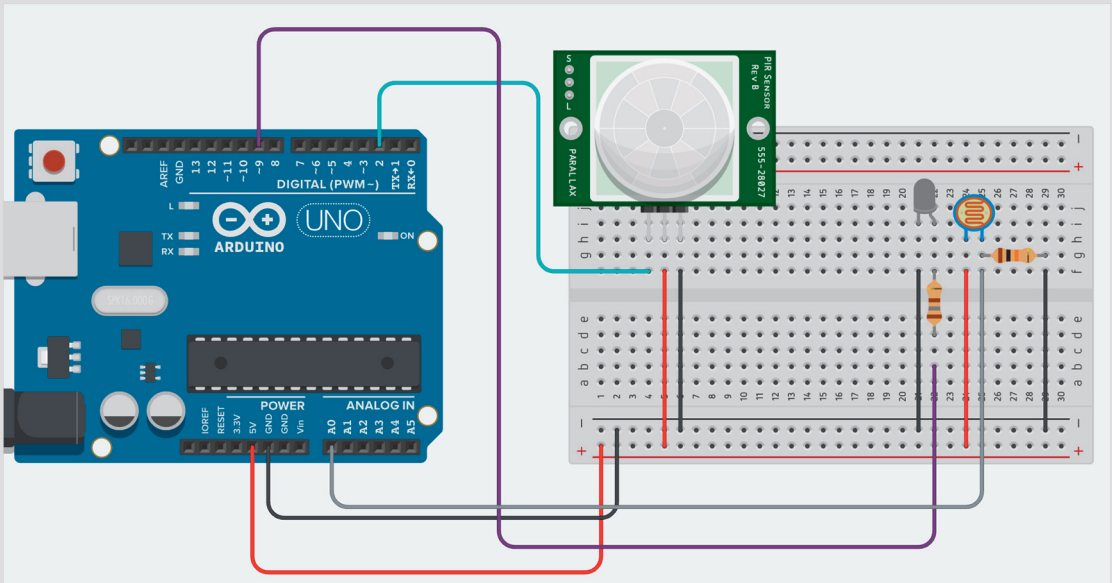
- Programmation de ce montage en Arduino :

```
// Allumage d'une DEL sur détection de mouvement

void setup()
{
    pinMode(2,INPUT); // broche 2 configurée en entrée
    pinMode(9,OUTPUT); // broche 9 configurée en sortie
}

void loop()
{
  if (digitalRead(2) == HIGH) // test de détection de mouvement
  {
    digitalWrite(9, HIGH); // allumer la DEL
    delay(100); // attente (valeur à modifier pour obtenir 30 s)
  }
  else
  {
    digitalWrite(9, LOW); // éteindre la DEL
  }
}
```

- Ajout de la photorésistance (La résistance branchée sur la photorésistance aura une valeur de 10 kΩ) :



- Le programme permettant de lire la valeur de sortie d'une photo résistance utilise les fonctions **analogRead()** (lecture sur une prise analogique), et **Serial.println** pour écrire cette valeur dans le moniteur série. Cela nous permettra de connaître la valeur lue sur la photorésistance en fonction de la lumière ambiante :

```
// Mesure de la résistance de la photorésistance pour calibrage

int valPhoto = 0; // variable de la valeur lue de la photorésistance

void setup()
{
    Serial.begin(9600); // démarrage de la liaison série
    pinMode(A0, INPUT); // broche A0 configurée en entrée
}

void loop()
{
  valPhoto = analogRead(A0); // rangement de la valeur lue dans valPhoto
  Serial.println(valPhoto); // envoi de ValPhoto vers le moniteur série
  delay(100); // attente de 100 ms pour rendre l'affichage lisible
}
```

Relever la valeur minimale (quand il fait nuit) renvoyée par la photorésistance dans le moniteur série.

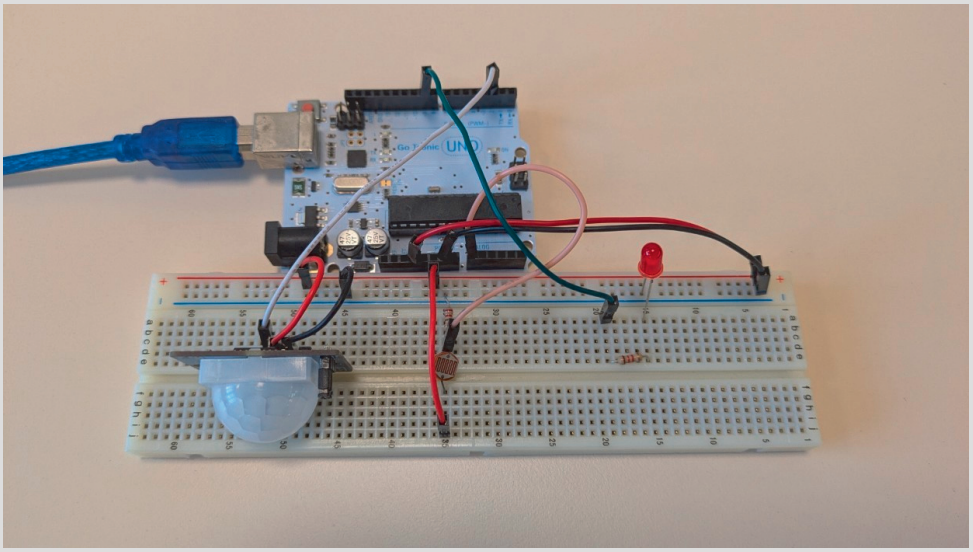
Il reste maintenant à réaliser le programme final. Nous allons devoir utiliser le premier programme cité en exemple et ajouter la lecture de la valeur de la photorésistance.

Comme lu dans le logigramme fourni il va falloir utiliser une fonction **ET** dans le test **IF.** Pour réaliser une telle fonction en Arduino (C++), il faut utiliser le double caractère suivant : **&&**

if (condition1 && condition2) à lire **si (mouvement et lumière inférieure à n)**

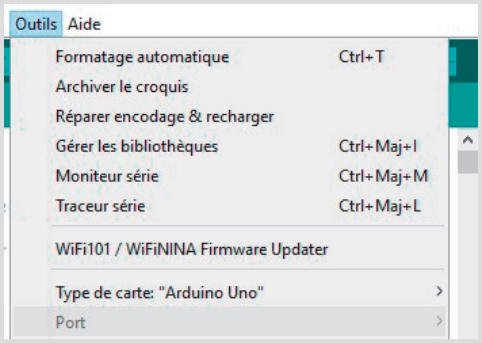
4. Expérimentation

- Effectuer le montage avec le matériel fourni (montage identique à celui réalisé sous TinkerCad) :



- Ouvrir l'application Arduino

- Brancher la carte Arduino sur un port USB disponible de votre ordinateur.
- Choisir le bon port com pour connecter la carte Arduino :



- Télécharger le programme précédemment rédigé sur Tinkercad sur la carte Arduino

- Vérifier le bon fonctionnement du programme et du montage