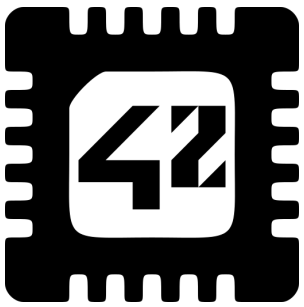
ATELIER DÉCOUVERTE

ARDUINO & PROGRAMMATION ÉLECTRONIQUE

Durée estimée : 8 à 12 heures
Matériel requis : ordinateur personnel avec port usb A



Bienvenue à toi qui découvre les cartes programmables et l'électronique !

Le Labelec de 42 te propose de suivre cette série d'exercices pratiques. Ils sont pensés pour être réalisables avec le kit qui t'est fourni. Après ça, tu seras en mesure de comprendre les fondamentaux de l'utilisation d'une carte programmable, et quelques bases d'électronique hardware. Bref tu seras prêt pour te lancer dans tes propres projets.

Trois règles :

1. *Prends ton temps.*
2. *Si besoin, pose des questions à tes camarades ou aux membres du Labelec.*
3. *Crée un fichier pour chaque étape de l'atelier.*

Ah oui, petite précision, les circuits à réaliser seront représentés sous la forme de schéma électrique, avec des symboles normalisés. Pas de panique, tout te sera expliqué.

*Si tu es pressé, contente toi de l'**Exercice** de chaque étape. Tu trouveras parfois une proposition d'**Exercice +** si tu souhaites approfondir.*

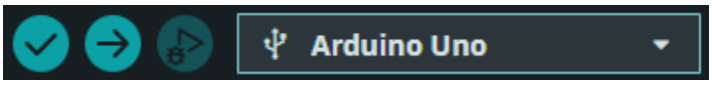
ETAPE 0 : Another “Hello World”

Bon, commençons. Tout d’abord tu peux télécharger et installer l’IDE Arduino sur ton ordinateur :

https://www.arduino.cc/en/software

Pour les utilisateurs LINUX, n’utilisez pas la version “AppImage” : l’installation du package “Fuse” qui est proposé fera crash votre ordinateur au boot. 💣

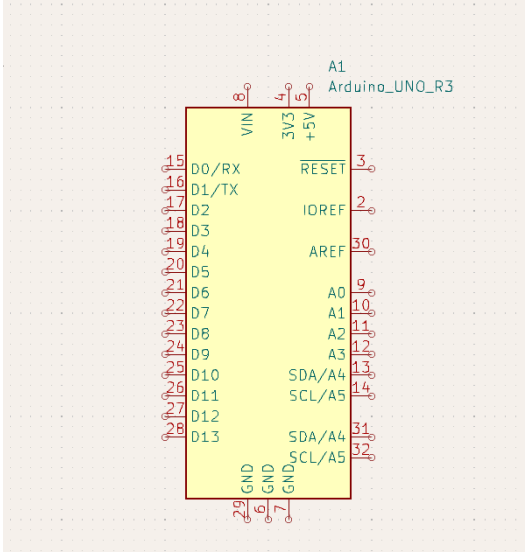
Une fois que c’est fait, lance le logiciel puis branche la carte à l'aide du câble USB fourni. Ce câble a deux utilités, il permet d’envoyer du code vers la carte, mais aussi, il l’alimente ! Pour cet atelier, on n'aura donc pas besoin d’une alimentation extérieure. En haut doit apparaître en surgras **Arduino Uno**, comme ca :



Ça signifie que ta carte est détectée ! Sinon, clique sur le menu déroulant et paramètre le port COM (USB) et le type de carte que nous utilisons. Tu verras, il en existe beaucoup, mais nous c’est bien l’Arduino Uno que nous employons, la base de la base !

Maintenant tu vas envoyer un premier programme vers la carte. Ouvre le fichier “File” "Examples">"01.Basics">"Blink". Mais que va t-on donc faire clignoter? Une LED bien sûr ! Il existe sur l’Arduino Uno une LED intégrée, et c’est bien pratique pour tester la carte sans faire le moindre circuit. Tu peux cliquer sur la flèche bleu à côté de la case **Arduino Uno** et envoyer le code vers la carte. Après quelques secondes, la magie opère : la LED sur ta carte clignote à un intervalle d’une seconde ON, puis une seconde OFF.

Pour le moment voici notre schéma électrique :



Tu remarqueras que nous n'avons pas de source d'alimentation dans ce circuit, car nous n'alimentons pas la carte avec les pins disponibles (les petits trous), mais directement avec le câble USB. Pour la suite de l'atelier, nous ferons toujours ainsi !

Regardons maintenant le contenu du programme : en haut on retrouve le *setup()*, il correspond à l'initialisation de la board, il n'est lu qu'une fois. En dessous, on retrouve la *loop()*, elle est lue en boucle, tant que ta carte n'est pas réinitialisée. Mais bon, tout ça est déjà très bien expliqué en commentaire sur le programme "Blink", je t'invite à le décortiquer.

Exercice : Ouvre un nouveau fichier puis rédige et téléverse un programme qui renvoie les signes "4" et "2" en langage morse.

Exercice + : Essaie de rédiger le même programme, mais sans le moindre chiffre ou nombre dans ta *loop()*. Cherche de la doc sur les *#define*, et tu devrais t'en sortir assez vite.

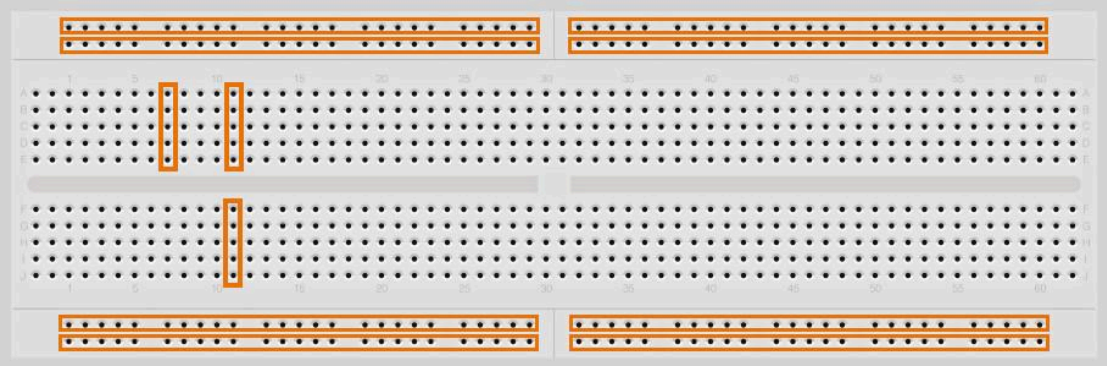
ETAPE 1 : Que la lumière soit ...

Maintenant, réalisons notre premier vrai circuit de prototypage. Pour cela, tu peux prendre la breadboard, cette sorte de planche à trous, et quelques jumpers, les câbles avec les embouts métalliques. On les appelle parfois aussi "cables duPont". Si tu te poses la question : ils n'ont absolument rien à voir avec Tintin.

La breadboard permet de prototyper un circuit sans faire de soudure. Les trous sont connectés entre eux par des connecteurs, sous le plastique.

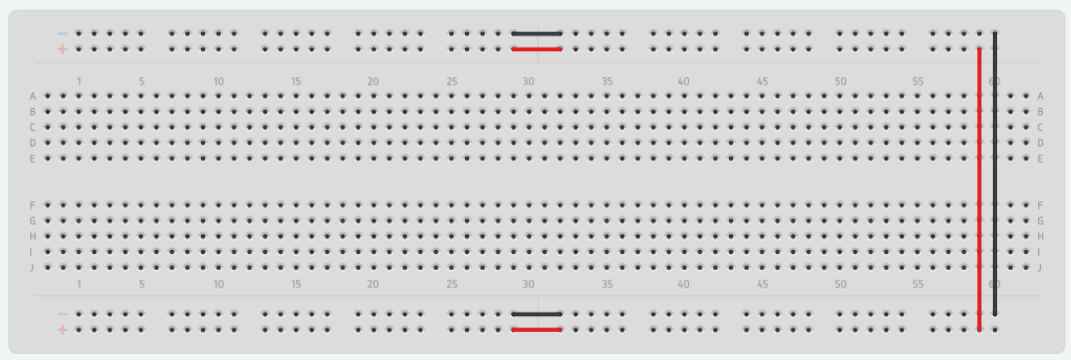
Au milieu de la planche, en colonne, ils sont connectés par groupe de 5 trous. Tu as donc 128 points de jonctions en colonne, pouvant réunir chacun 5 entrées.

En haut et en bas, en ligne, tu as 8 groupes de 25 trous. On les utilise généralement pour l'alimentation +/-.



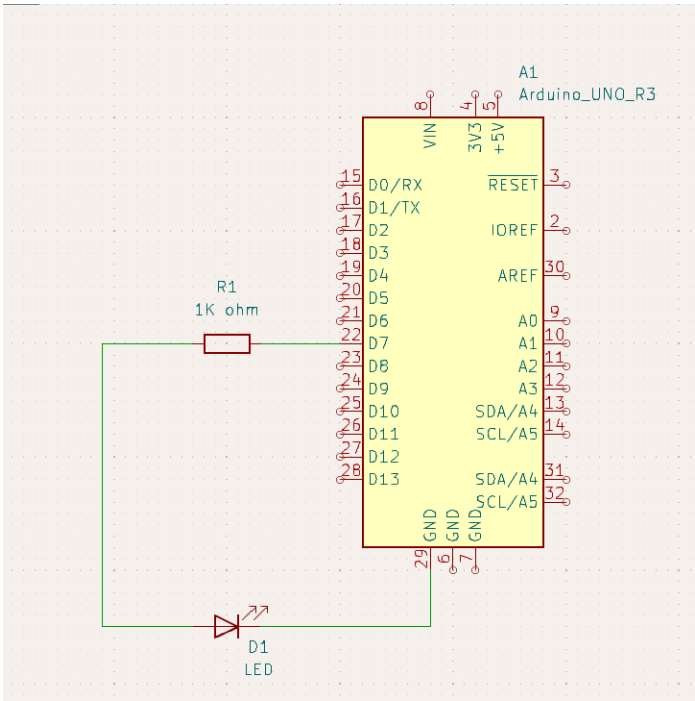
2 conseils :

Comme on oublie souvent que ces longues lignes de 50 trous sont en réalité divisées au milieu, on aura tendance à faire de ces demi-lignes de 25 trous des longues lignes de 50 trous en utilisant des jumpers. Il est aussi impératif de toujours connecter les ports d'alimentation ensemble : tous les 5V sont interconnectés, de même pour les grounds (0V). Pour partir sur une base saine, tu peux connecter ta board ainsi :

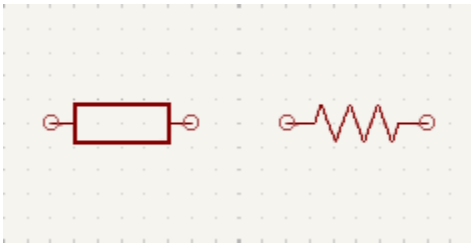


Les jumpers ont de ravissantes couleurs que tu peux utiliser à profit pour t'y retrouver. Ça n'a aucun impact sur le fonctionnement du circuit, mais ça peut être très utile lorsqu'il commence à se complexifier. Il existe des conventions : on utilise par exemple le rouge pour le +, le noir pour le - (ou 0V), mais tu peux develloper ton propre vocabulaire coloré pour t'y retrouver.

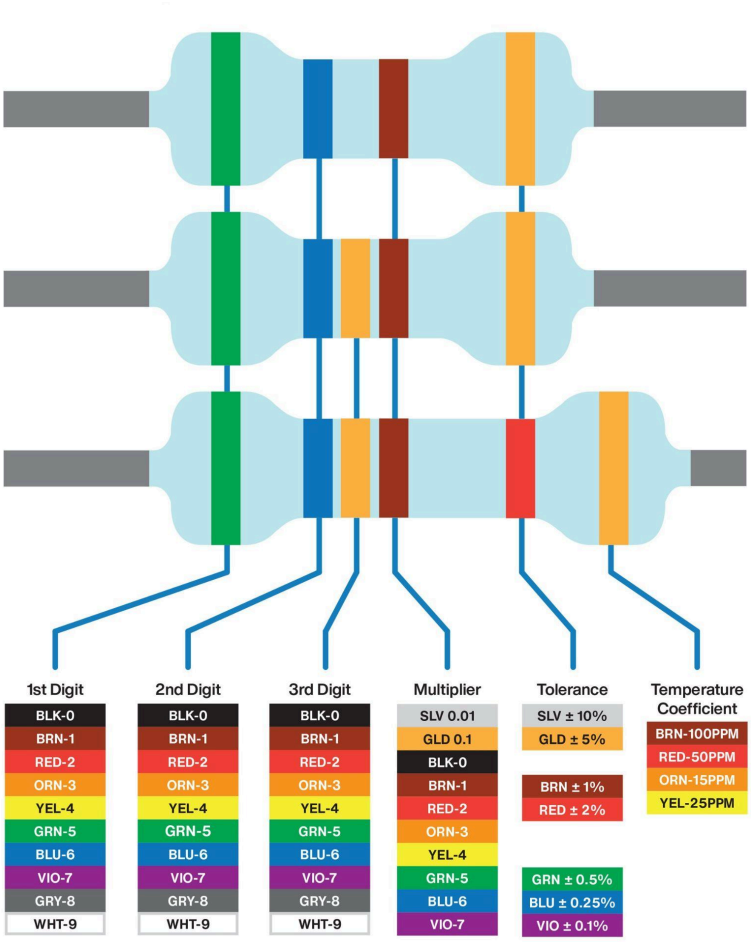
Maintenant que tu comprends tout du fonctionnement de ce magnifique outil, tu peux réaliser le circuit électronique suivant, **n'oublie pas que lorsque tu réalises les connections, il faut débrancher ta carte de l'alimentation :**



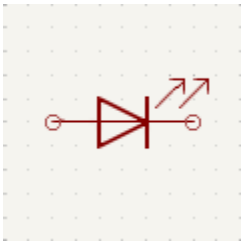
"OULAH, mais qu'est ce que c'est que ce rectangle et ce bouton PLAY?!" me diras-tu, et bien se sont une résistance et une LED, schématisées l'une et l'autre ainsi :



Ca, se sont les résistances, il existe deux manières de les symboliser. C'est un composant non polarisé : Il n'y a pas de sens spécifique dans lequel le brancher. Le monde est bien fait : c'est un dessin symétrique. Pour choisir la bonne résistance de 1K soit 1000 (l'unité est en "ohm"), refere toi a ce genre de tableau. Il y en a pleins sur le web 👍



Si le doute persiste, tu peux vérifier avec un multimètre. Un membre du Labelec peut te montrer comment ça fonctionne :)



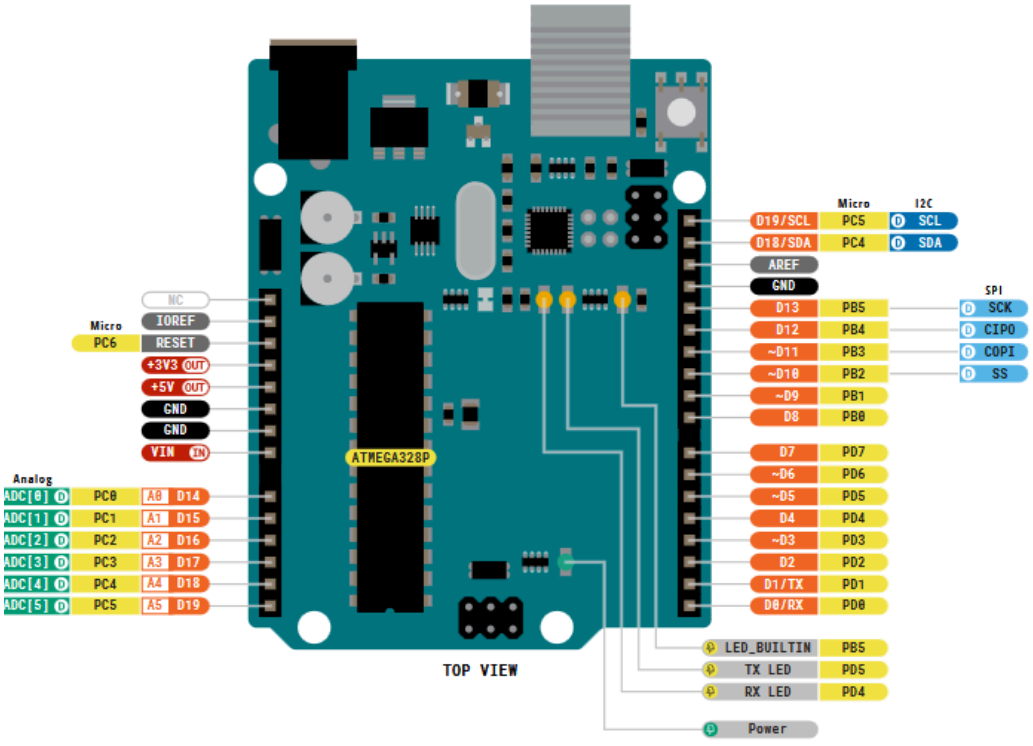
Ca, c'est une LED, c'est comme le dessin d'une diode (cette sorte de bouton Play/Pause), mais avec deux flèches en plus. Comme tu t'en doutes, c'est un composant polarisé, il faut le brancher dans un sens spécifique. Pour s'y retrouver dans le monde réel, les LED sont vendues avec une patte longue et une patte courte. **La patte courte va vers le -**.

“Mais est ce vraiment nécessaire d'utiliser une résistance pour allumer une LED ?” te questionneras-tu, et bien oui on pourrait imaginer se contenter de l’alimentation et de la LED ! Hélas ta LED n'aura même pas le temps de briller qu’elle sera déjà bonne pour la poubelle, si tu essaies de te passer de la résistance.

C’est un peu long à expliquer, mais en gros la résistance protège ta LED : Elle réduit le courant fourni par la batterie (ici la carte Arduino). Pour te donner une image, pense à une sorte de barrage qui régule le flux d’eau fourni, pour qu’un village en aval puisse être alimenté en eau (la LED s’allume). Trop fort (ohm trop élevé), le barrage retient trop d’eau et le village est sous alimente, la LED ne s’allume plus. Trop faible ou sans barrage, le village termine comme l’Atlantide et la LED grille.

“Mais où sont le + et le - dans ce circuit, je ne lis que D7 et GND ?” te demanderas-tu, et encore une fois, tu te poseras une question pertinente.

Pour faire très simple, tu retrouveras trois grands pôles de connecteurs sur ta carte Arduino Uno.



1. <u>Les ports “Analog” ou “ADC”</u>

Ils sont en bas à gauche, au nombre de 6, allant de ADC[0] à ADC[5]. Ils servent à recevoir un voltage analogique et le convertissent en une valeur numérique allant de 0 à 1023. Si tu veux en savoir plus : https://opentp.fr/card/entrees-analogiques/

2. Les ports d'alimentation

Ils sont en haut à gauche, et servent à l'alimentation du circuit et de la carte. Les plus utiles sont :

- **VIN** sert à alimenter la carte Arduino sans passer par le câble USB
- **+ 5V** sert à alimenter le reste du circuit en 5 volts.
- **+3.3V** sert à alimenter le reste du circuit en tu te doutes bien combien de volts.
- **GND** est l'acronyme de "Ground", il est situé à plusieurs endroits sur la carte mais ils sont tous interconnectés. Il est l'équivalent du 0 volt, autrement dit du -.

3. Les ports digitaux

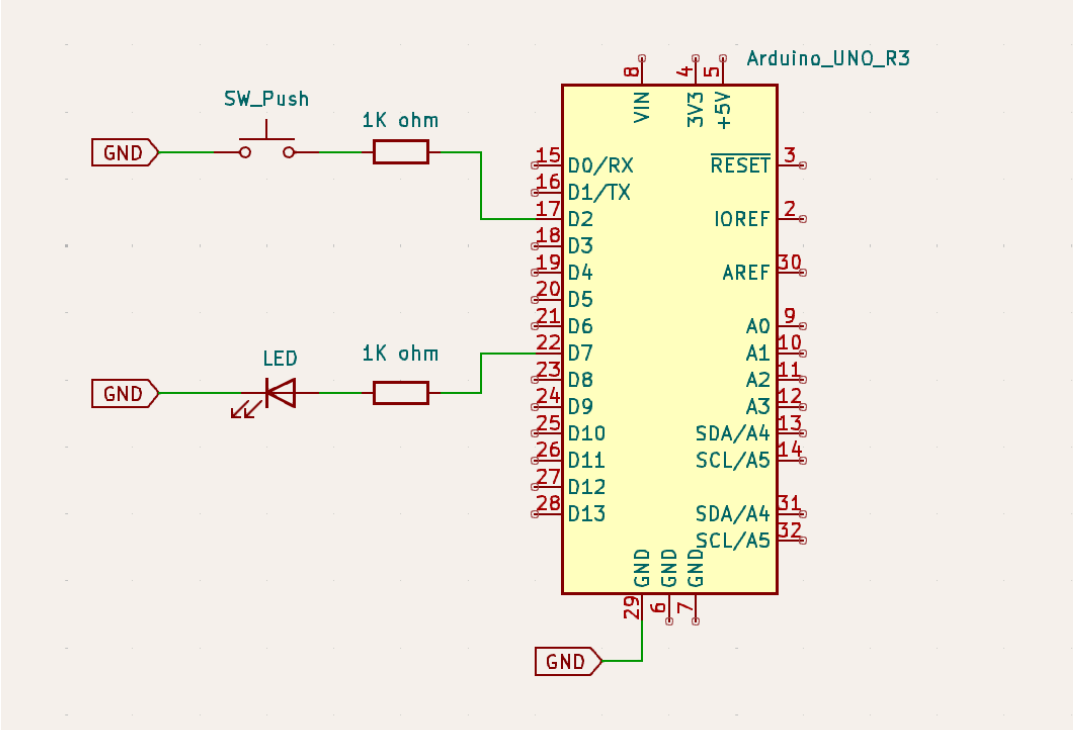
Ils sont principalement sur tout le côté droit de la carte. Ils peuvent fonctionner dans les deux sens, recevoir (en entrée) ou envoyer (en sortie). **En entrée**, ils sont capables de lire un état haut ou bas. **En sortie** ils peuvent générer une tension pilotable (en volt), qui ne peut avoir que deux états, "LOW" (GND ou 0 V), ou "HIGH" (environ 5 V).

Exercice : Maintenant, tu en sais largement assez pour faire ton circuit à l'aide de la breadboard! Lorsque c'est fait, appel un membre du Labelec pour qu'il vérifie avant d'alimenter ta carte.

Ensuite, reprend la structure du programme de l'étape 0 pour faire le même code morse, mais cette fois-ci, en utilisant la LED présente sur la breadboard.

ETAPE 2 : ... mais pas trop.

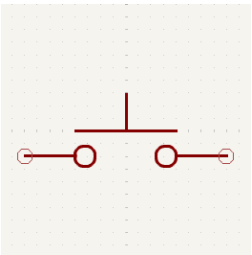
Cette puissante lumière me brûle la rétine ! Il serait temps de modérer tout ça. Pour ça, rien de mieux qu'un simple bouton. Voici le circuit que je te propose comme circuit :



Décortiquons tout ça. Déjà, sache que tu peux laisser la LED et la résistance 1k ohm de l'étape 1, tel quel sur la breadboard. Oui, même si le dessin a changé, le petit drapeau, aussi appelé “Label,” symbolise le fait que la sortie de la LED est toujours branché au GND de la pin 29 :

Les labels permettent de simplifier les schémas électriques. Dès lors que deux labels ont le même nom sur un même schéma, ils sont connectés ensemble. C’est très pratique pour éviter les schémas électriques trop compliqués, ou tout s'emmêlent.

Ensuite, on découvre ce symbole :



Comme tu t'en doutes, c'est un interrupteur bouton, aussi appelé switch push. Tu sais comment ça marche : pressé le courant passe, mais relâché il ne passe pas. Encore une fois, c'est un composant non-polarisé, mais **attention** les pattes d'interruptions sont croisées: la patte supérieure gauche fonctionne avec la patte inférieure droite. Tu peux aussi tester ca au multimètre ⚡

Tu ne remarques rien de bizarre dans ce circuit? Dans le rapport qu'entretiennent le bouton et la LED ? Effectivement, le bouton n'est pas "en série" avec la LED, leur alimentation viennent de deux ports digitaux différents. Ce n'est donc **pas directement la fermeture du bouton qui allumera la LED, il faudra le programmer.** Hey? alors pourquoi ne pas faire l'inverse ? 😀

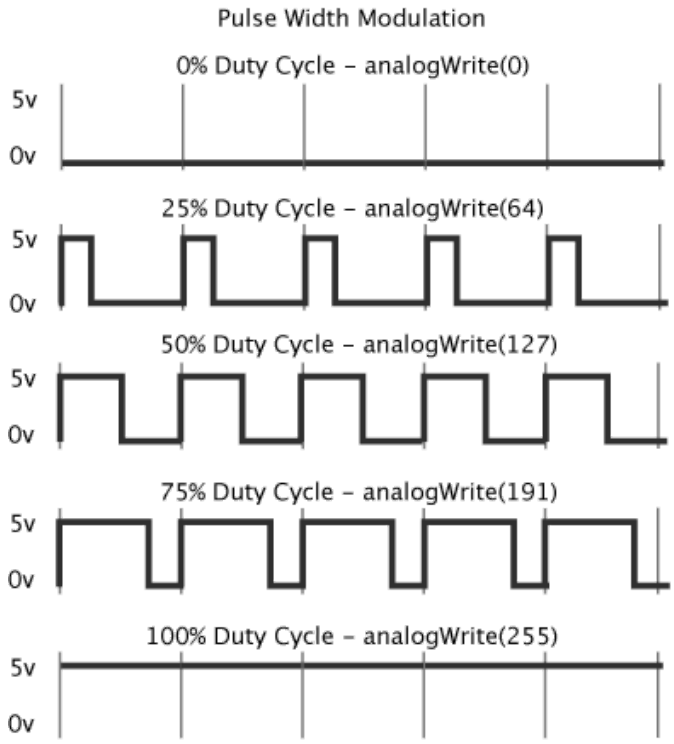
Exercice : Réalise le circuit électrique ci-dessus, puis sollicite un membre du Labelec pour qu'il vérifie ta board **avant de la brancher en USB**. Ensuite, à l'aide de conditions comme *if* ou *while*, allume la LED lorsque le bouton est relâché et éteint la lorsque le bouton est pressé.

Exercice + : En plus de l'exercice ci-dessus, tu dois pouvoir afficher en parallèle et en continu le code morse "4" et "2"sur la LED BUILT IN, comme dans l'étape zero. Pour cela regarde du côté des *interrupts()* ...

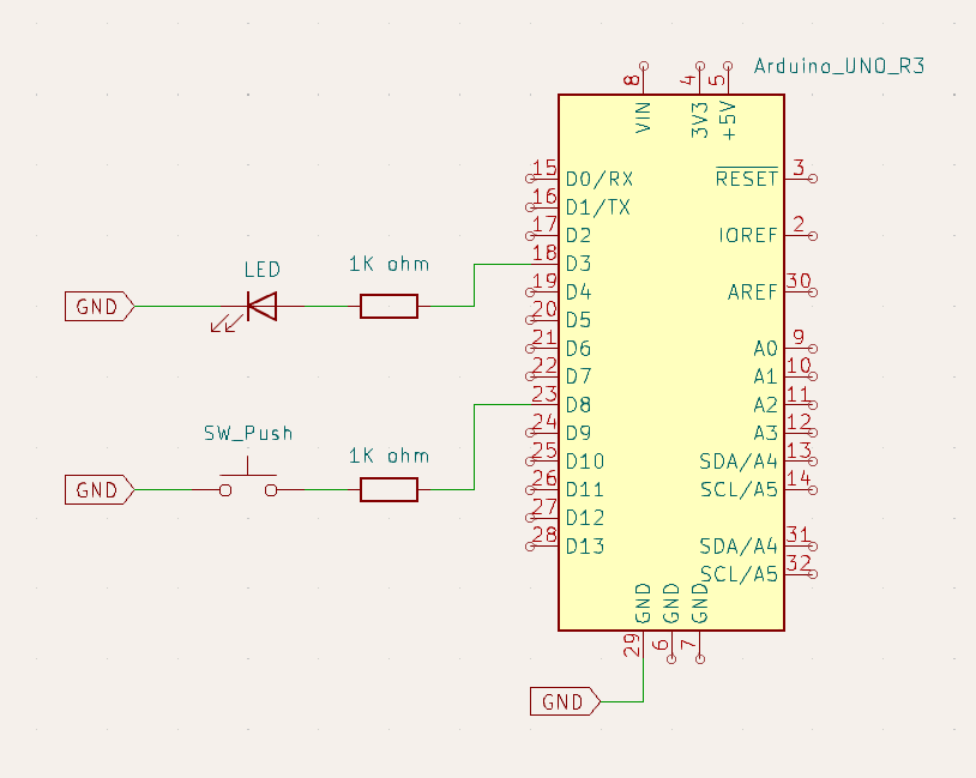
Etape 3 : Here Comes the Sun ~

Si tu regardes ta carte Arduino Uno en détail, tu verras à proximité de certains ports digitaux le symbole "**~**" ou "***tilde***". Il signifie "**PWM**" ou "**Pulse With Modulation**". Triste nouvelle : la carte arduino ne possède pas vraiment de sortie analogique avec une tension ajustable entre 0 volt et 5 volts. Pour compenser cette terrible lacune, les ports digitaux dont le chiffre est précédé d'un ~ sont capables, en sortie, d'alterner entre etat HIGH et etat LOW a très haute vitesse. Cette solution, un peu bancale je te l'accorde, est finalement suffisante dans bien des cas pour simuler un voltage intermédiaire.

On utilisera la fonction *analogWrite()* pour niveler la part de durée d'etat HIGH et la part de durée d'etat LOW, le tout entre 0 et 255.



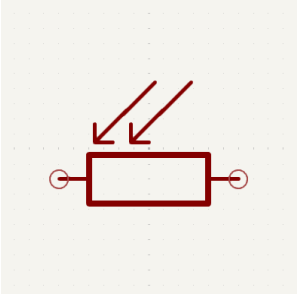
Ah oui, n’oublions pas de décaler la LED sur un port digital equipe du PWM :



Exercice : Tu aimes les couchers de soleil ? Tout le monde aime les couchers de soleil ! Programme le lever et le coucher du soleil (allumage et extinction progressive de la LED) conditionnés à la pression du bouton. Le bouton pressé, le soleil se couche progressivement, relâché, il se relève progressivement aussi. Attention, pour cet exercice, n'utilise pas la fonction *interrupts()*.

ETAPE 4 : Day'n'nite

Maintenant, penchons-nous sur les ports analogiques. Nous allons utiliser une photorésistance, résistance lumineuse ou "Light Dependent Resistor" (LDR) en anglais. Son symbole électrique ressemble à ca :

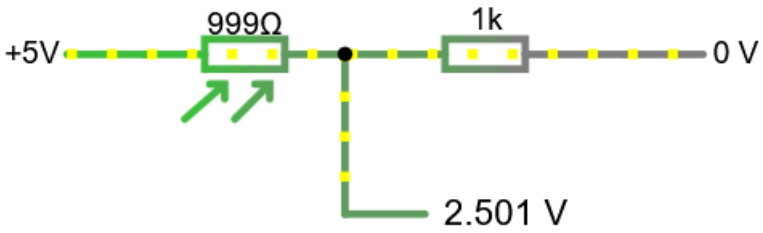


Mais en réalité, elle ressemble plutot a ca, tu peux la prendre dans ton kit :

Ce petit capteur optique peut faire varier sa résistivité. Plus elle est éclairée, plus sa valeur en ohm baisse, et inversement.

"C'est bien intéressant tout ça, mais c'est pas très pratique à convertir en valeur numérique !" penses-tu. Rappelle-toi, dans l'étape 1, les ADC sont capables de lire une valeur en voltage et la convertir en numérique de 0 à 1023, pas une valeur en ohm.

Alors comment faire? **Une solution existe**, et elle repose sur de la bonne vieille électronique analogique, en faisant ce que l'on appelle un "**diviseur de tension**". C'est un circuit très simple ou deux résistances sont positionnées l'une après l'autre (en série) entre la tension positive (ici 5 volts) et le ground (0 volt). De cette manière on obtient une tension intermédiaire à la borne qu'elles ont en commun.



Avec deux résistances de 1K Ohm (ou presque) entre 0 volt et 5 volts, on obtient à leur point de jonction 2.5 volts, simple non?

Bon évidemment, on utilisera pas toujours des résistances de mêmes valeurs, dans notre cas même, la valeurs en ohm d'une de nos deux résistances évolue selon la lumière. Pour calculer le voltage obtenue entre les deux bornes, on utilisera cette savante fonction :

$$U_2 = U \times R_2 / (R_1 + R_2)$$

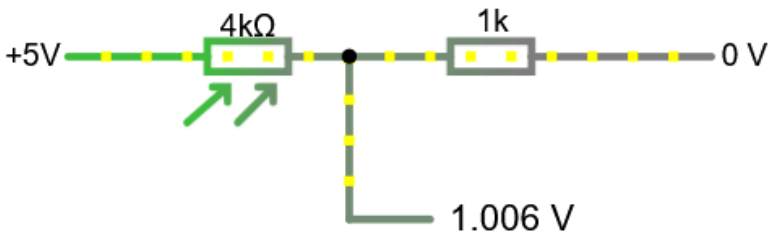
Autrement dit :

Volt.Intermédiaire = Volt.positif x Résistance côté GND / (Résistance côté Volt.positif + Resistance côté GND)

Testons avec une résistance de 4k côté alimentation et 1k côte Ground :

Volt.Intermédiaire = 5V x 1000ohm / (4000ohm + 1000ohm)
Volt.Intermédiaire = 5 x 1000 / (4000 + 1000)
Volt.Intermédiaire = 5 x 1000 / 5000
Volt.Intermédiaire = 5000 / 5000
Volt.Intermédiaire = 1

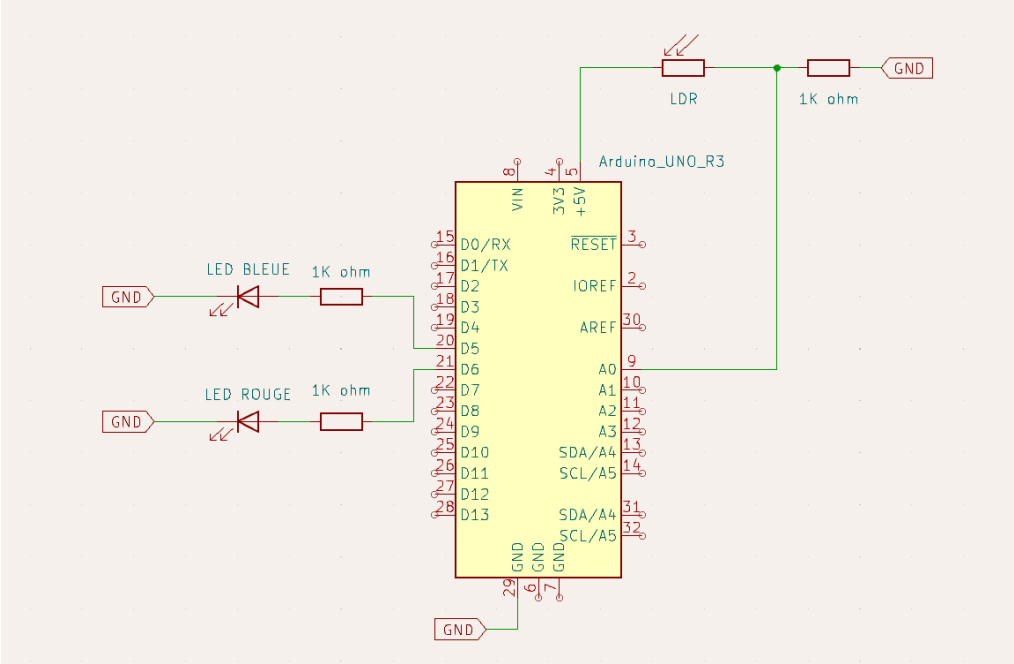
Maintenant, vérifions dans Falstad, (c'est un super site pour tester des circuits) :



Tadaaaaaam, on a grosso modo 1 Volt. N'est ce pas magnifique?

Bon on se disperse, tu n'as pas vraiment besoin de tout ça dans l'immédiat. C'est surtout pour que tu saisisse l'idée essentielle : **en électronique, le numérique et l'analogique se complètent bien**. 🤜🤛

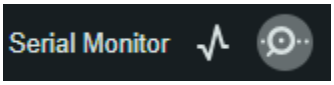
Tu peux déjà réaliser le circuit suivant :



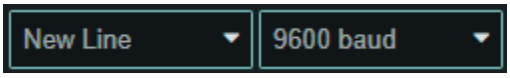
Demande à un membre du Labelec de vérifier ton circuit avant de passer à la suite de l'atelier.

Avant de passer à l'exercice de cette étape 4, repassons sur du numérique. Nous n'en avons pas encore fait usage, mais il existe une fenêtre de debug utilisable dans l'IDE Arduino. Elle permet de communiquer entre ta carte et le logiciel. Nous allons en avoir grand besoin.

Pour afficher cette fenêtre, clique dans l'angle supérieur droit sur l'icône de Loupe :



Normalement, elle ouvre une fenêtre dans la partie inférieure de ta fenêtre, une fenêtre tristement vide. Pour la remplir et tester la communication entre ta carte et le logiciel, ouvre le fichier "File" "Examples">"01.Basics">"AnalogReadSerial". Tu vas voir dans le *setup();* la fonction *Serial.begin();* elle est paramétrée sur une vitesse de communication à 9600 baud rate (c'est l'unité de mesure de cette vitesse). Il te faut alors paramétrer ton logiciel de la même manière, à la même vitesse de transmission, dans la partie inférieure droite.



Et là, tu obtiens un retour numérique de la valeur captée analogiquement. Enfin, pour être plus exact, tu reçois une conversion numérique, entre 0 et 1023 du voltage obtenu suite à notre diviseur de tension.

Ce qui te permet d'obtenir ce retour régulier dans la fenêtre de debug, c'est la fonction *Serial.println();*, présent dans la *loop();* ou le *ln* sert à faire un retour ligne.

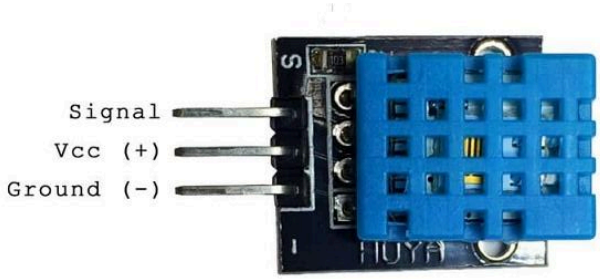
Normalement si tu caches complètement la tête de la photorésistance entre tes doigts la valeur numérique sera aux alentours de 20, en pleine lumière, on sera plus aux alentours de 300. C'est évidemment loin de couvrir toute l'échelle des possibles de 0 à 1023, mais c'est largement assez pour convertir ces valeurs et allumer une LED.

Exercice : Utilise les valeurs de la photorésistance pour contrôler l'allumage des 2 LEDs de ton circuit. En pleine lumière, la LED rouge doit être allumée à pleine intensité et la LED bleue complètement éteinte.Quand la lumière est bloquée, la LED bleue doit être allumée et la LED rouge doit s'éteindre. Pour les niveaux intermédiaires de lumière, la luminosité des LEDs doit être ajustée proportionnellement. Utiliser la fonction *map()* peut t'aider.

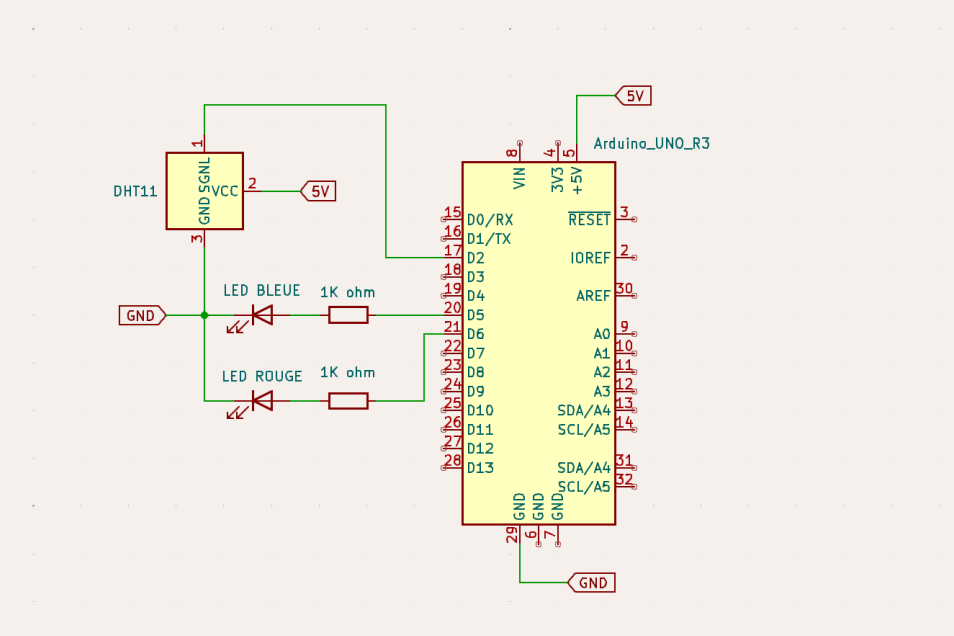
Exercice+ : mince, l'allumage de la led avec le PWM n'est pas linéaire ... Si tu as le courage, tu peux essayer d'adoucir tout ça pour que l'allumage des 2 LEDs se chevauchent un peu moins au niveau des valeurs intermédiaires (attention c'est galère) !

ETAPE 5 : les bibliothèques de la flemme

Désormais un nouveau capteur rentre en jeu, le **DHT11** ! Derrière cette douce appellation se cache un capteur d'humidité et de température assez archaïque et imprécis, mais largement assez efficace au vu de nos besoins. Voici à quoi il ressemble :



Je t'invite à le connecter comme sur le schéma électrique ci-dessous :

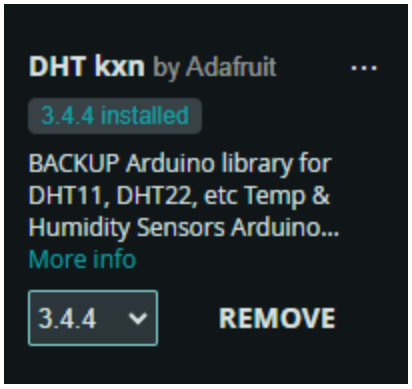


Demande à un membre du Labelec de vérifier ton circuit avant de passer à la suite de l'atelier.

Cette fois-ci pas besoin de mapper les résultats comme avec la résistance lumineuse. On va simplement faire appel à des librairies. Des gens particulièrement motivés les ont rédigés pour nous faciliter les choses, alors profitons-en !

Sur le menu gauche de l'IDE Arduino, tu peux ouvrir le "library manager", avec l'icône représentant des livres, sur le volet gauche de l'interface graphique :

Ensuite, installe "DHT kxn" de Adafruit. Cette librairie va te permettre de travailler avec différents capteurs de la série DHT, dont notre cher DHT11.



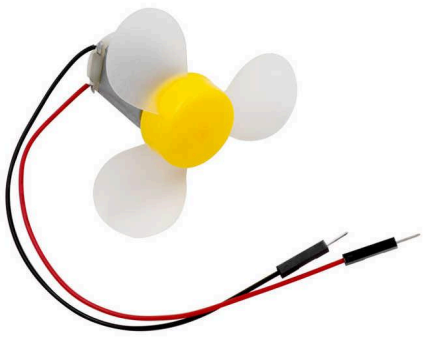
Après, tu peux ouvrir le fichier suivant “File”>"Examples">"DHT kxn">"DHTTester". L’installation de librairies va souvent de paires avec quelques programmes de démonstrations. Ils te permettront de comprendre comment intégrer les fonctionnalités de la librairie en question à tes besoins. Je t'invite à lire scrupuleusement son contenu et à tester ton capteur (tu as une petite manipulation dans le programme a effectuer pour le faire fonctionner)!

Bon, tu peux désormais lire dans le serial la température ambiante et le pourcentage d'humidité capte. En soufflant dessus, de prêt, tu peux faire évoluer les valeurs.

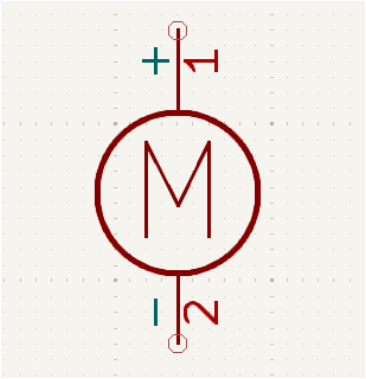
Exercice : Si le capteur détecte une valeur en deçà de 80 % d'humidité, la LED est bleue. Lorsque l’on passe au-dessus des 80%, la LED passe devient rouge. Ton programme sera forcément lent, en raison de la faible qualité du capteur, sur ce point la, pas de miracle possible!

ETAPE 6 : Prévision canicule

Tu es désormais prêt pour aborder la dernière étape : la création d’un ventilateur automatique ! Pour concevoir cet incroyable outil qui flirte avec la géo-ingénierie, tu vas avoir besoin du moteur et des pales de ventilateurs. Fixe les pales sur l’embout de ton moteur pour qu’une fois assemblé, il ressemble a ca :

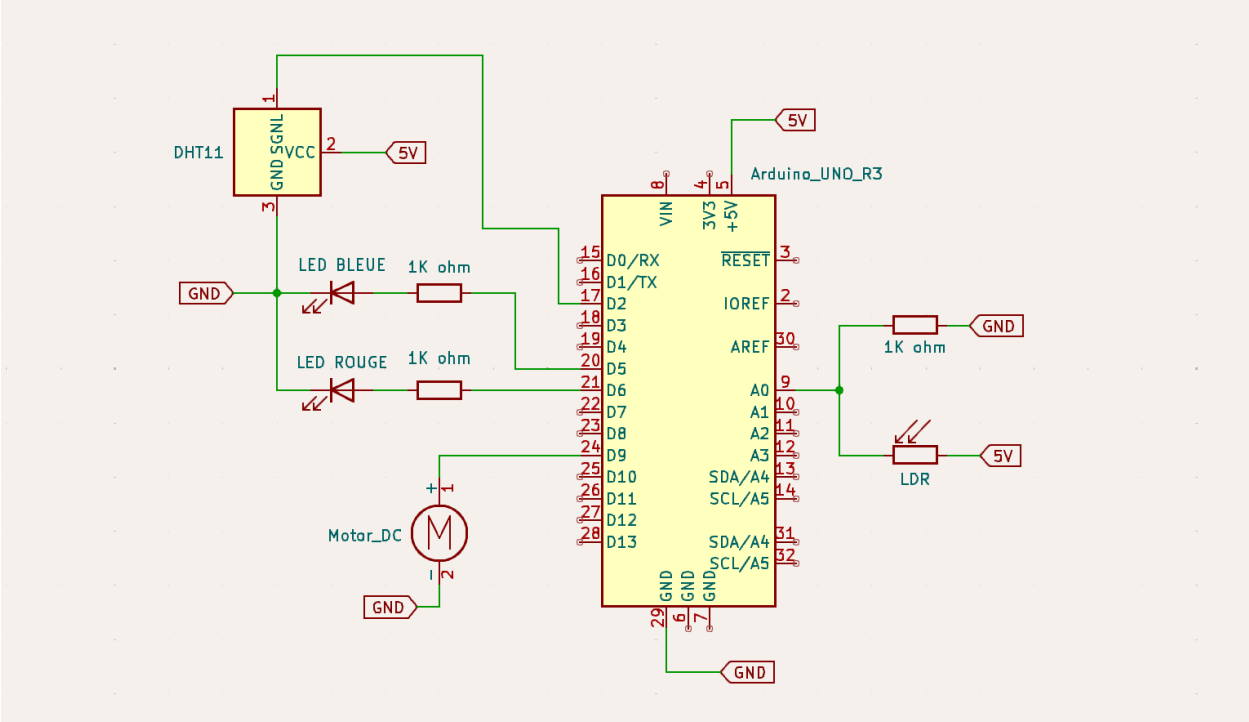


Pour information, voici le symbole électrique du moteur DC (ou “a courant continu”) :



Il est parfois aussi représenté avec une barre sous le “M”. Ah oui, petite précision, il pourrait sembler polarisé, avec le “+” et le “-”, mais il ne l’est pas vraiment. Lorsque tu l’alimentes dans un sens, il tourne dans le sens horaire, si tu l’alimentes dans l’autre sens, il tourne dans le sens anti-horaire. Dans notre situation, la polarité n’a d’importance que pour la direction du vent !

Ensuite, tu peux concevoir ce circuit :



Tu constateras que finalement, peu de choses changent par rapport au circuit précédent. On retrouve seulement la photo-résistance montée en diviseur de tension sur une pin analogique, comme à l'étape 4, ainsi que le moteur DC sur une pin digitale.

Demande à un membre du Labelec de vérifier ton circuit avant de passer à la suite de l'atelier. Ensuite tu peux passer à l'exercice final.

Exercice final : Construit un système automatique de ventilation pour lutter contre les fortes chaleurs ! A température ambiante, seule la LED bleue est allumée. Si la température excède la température ambiante de la salle de +0.5 ° celsius, la LED bleue s'éteint, le moteur se déclenche et la LED rouge s'allume.

Exercice final + : Le système de ventilation doit être **automatique et manuel**. L'exercice est le même sauf que le ventilateur peut aussi être allumé et éteint manuellement, grâce à la photorésistance qui sert alors d'interrupteur. La couleur des LEDS dépend toujours de la température et non de l'état du ventilateur. De plus, pour affiner l'affichage des informations, au-delà du seuil de +0,5°C, la vitesse de clignotement de la LED rouge est proportionnelle à l'excès de chaleur par palier de 0.1°C. Autrement dit, plus la chaleur est élevée, plus la LED rouge clignote vite.

Petit conseil, il va falloir se passer de pas mal de *delay()* et regarder du côté de *millis()*.

Félicitations ! Tu es arrivé au bout des exercices de l'atelier ! Maintenant que tu maitrises mieux les outils, n'hésite pas à solliciter le Labelec pour tes projets. D'autres ateliers seront bientôt proposes, donc garde un oeil sur le Discord de l'association.

Merci pour ta participation 🙂