Variable et *if* en Arduino

Compétences travaillées :

- Connaitre les types booléen et entier en C/C++.
- Utiliser des variables et un test *if* en C/C++.
- Développer et programmer pour l'Arduino.
- Notions de montages électroniques et de programmation embarquée.

Très utiles pour les TIPE !

1. Rappels sur les variables et les condition *if*

1.1. Les variables

Une **variable** possède trois caractéristiques :

- Un **nom** qui l'identifie. Le nom peut être n'importe qu'elle chaîne de caractères commençant par une lettre, sans espace et sans opérateur (+, -, etc).
- Une **valeur** qui peut varier lors du déroulement du programme.
- Un **type** qui désigne que la nature des valeurs stockées. Par exemple, des chiffres, des lettres, des booléens (vrai ou faux).

En C et C++, une variable se déclare comme ceci *type nom;* ou *type nom = valeur ;*

Q.1. **Remplir le tableau suivant en donnant les caractéristiques des variables à partir de leur déclaration.**

Code C/C++	Nom	Valeur	Type
int t = 3 ;			
bool clig = false ;			
int t ;			

Une variable existe à l'intérieur des accolades les plus proches. On appelle **bloc de code** une section comprise entre deux accolades. Par exemple, dans le code suivant, la variable *t* existe dans tout le programme mais la variable *t2* n'existe que dans la section *loop*. On appelle **variable globale** une variable qui est déclarer hors de toute accolade, comme *t* ici.

```
int t = 3 ;
void loop() {
        int t2 = 11 ;
}
```

1.2. Les conditions si ou *if*

Une **condition *if*** permet d'ajuster le comportement du code en fonction de la valeur d'une ou plusieurs variables. Il est possible de construire des conditions plus compliquées mais dans cette activité, on se contentera de tester la valeur de variables booléennes, c'est-à-dire vraie ou fausse.

Une condition *if* peut contenir une **clause *else*** (« sinon ») qui correspond au code à exécuter si la condition est fausse.

En C et C++, le code d'une condition « si/sinon » s'écrit

```
if (condition) {
        // le code si la variable a pour valeur true (vrai)
} else {
        // le code si la variable a pour valeur false (faux)
}
```

2. Lecture et écriture de pins et condition *if*

2.1. Création du montage sur TinkerCAD

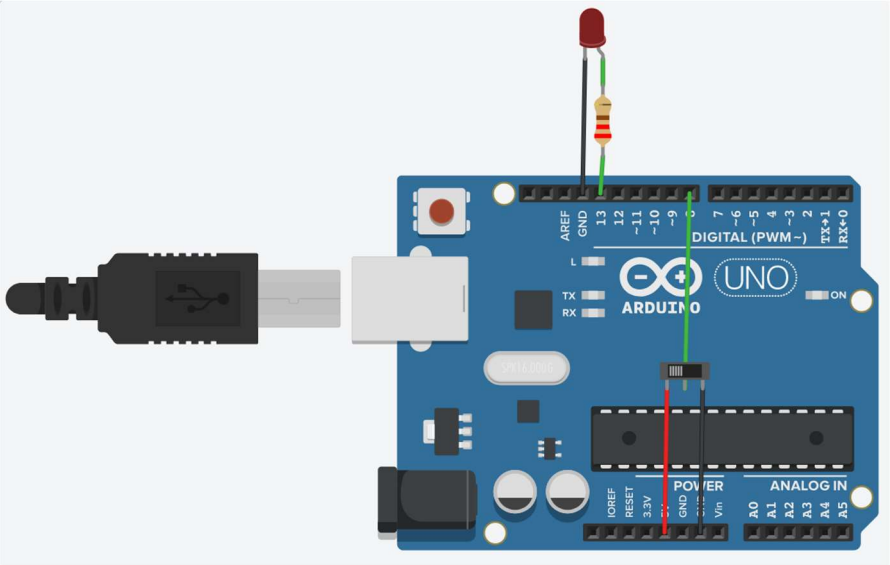
Q.2. **Se connecter sur le site www.tinkercad.com/ et rejoindre la classe via le lien :**

https://www.tinkercad.com/joinclass/TT7SWWR9P > Join with Nickname > Votre prénom sans majuscule avec accent (s'il y a deux fois le même prénom dans la classe, le second, par ordre alphabétique est 'prénom2'.)

Q.3. **Dans le menu de la classe, rejoindre l'activité « Découverte Arduino ».**

Q.4. **Dans cette activité, créer une nouvelle conception de type circuit.**

Q.5. **Dans votre conception, réaliser le montage suivant à partir d'une « Arduino », d'une « LED », d'une « résistance » et d'un « interrupteur à glissière ». S'assurer de mettre l'interrupteur dans la même position. Le fil vert est connecté à la patte centrale de l'interrupteur.**



2.2. Clignotement d'une LED

Q.6. **Dans le menu en haut à droite, cliquer sur l'onglet « Code ».**

Q.7. **Sélectionner ensuite dans le menu déroulant le mode « Texte » à la place du mode « Blocs ». Accepter la réinitialisation du code par défaut.**

Q.8. **Sachant que *LED_BUILTIN* correspond aussi au pin 13 de la carte, en déduire ce qui se passe lors de l'exécution du code. La simulation peut être exécutée en haut à droite via '*Start Simulation*'.**

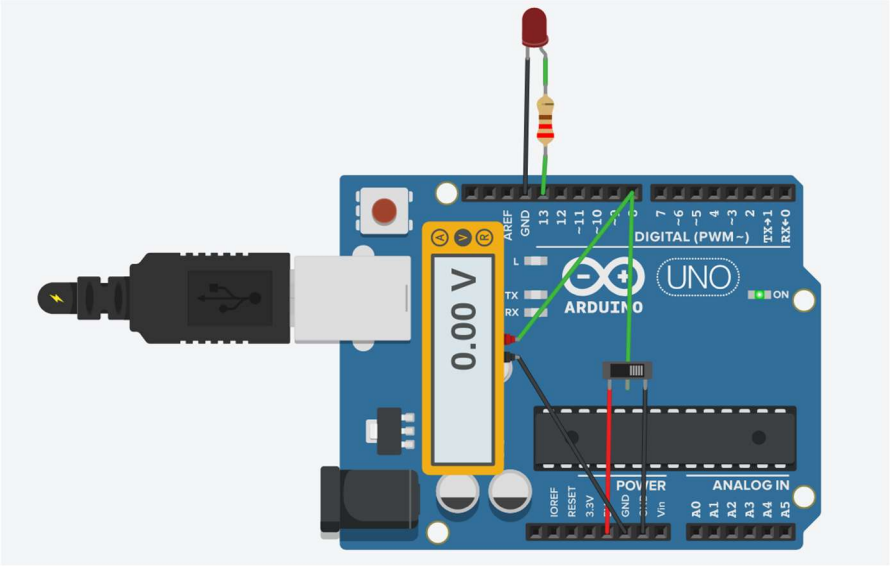
Q.9. **Modifier le code pour que la fréquence de clignotement de la LED soit de 5 Hz.**

Q.10. **Remplir les cases vides des deux premières lignes dans le tableau à la fin du document.**

2.3. Lecture de pin et condition

Q.11. **Ajouter un « Voltmètre » et le câbler comme ceci. Quelle est la tension mesurée ?**

Q.12. **Lancer une simulation et faire varier l'état de l'interrupteur. Observer la valeur de la tension sur le voltmètre.**



Les questions suivantes visent à créer un programme qui ne fait clignoter la LED que lorsque l'interrupteur est glissé vers la gauche, c'est à dire quand la tension du voltmètre est à 5V.

Q.13. **Recopier le code de la partie 1.2. au début de votre *loop* (attention au copier/coller).**

Q.14. **Remplacer la condition du *if* entre les parenthèses par l'appel de fonction permettant de lire le pin 8 et de renvoyer *true* ou *false* selon que la tension est à 5V ou 0V. Vous pouvez vous aider du tableau à la fin du document pour choisir la fonction adaptée.**

Q.15. **Déplacer tout le code permettant de faire clignoter la LED dans la zone correspondant au cas où la condition est vraie.**

Q.16. **Dans le cas où la condition est fausse, ajouter un *delay* de 100 ms.**

Q.17. **Tester le fonctionnement du code et déboguer si besoin.**

3. Utilisation des variables

3.1. Variable entière

Le code contient 3 fois le chiffre 100 car toutes les attentes sont de 100 ms. Il est préférable de remplacer les redondances de code par des utilisations de variables ou de fonctions. Ici, on va vouloir utiliser une variable entière pour définir la durée des attentes.

Q.18. **Observer le code et la position des accolades pour trouver le plus petit bloc de code contenant l'ensemble des *delay(100)*.**

Q.19. **Définir une variable de type entière nommée *attente* et contenant la valeur 100. La définition de la variable se fera au tout début du bloc identifié à la question précédente.**

Q.20. **Remplacer tous les « 100 » par le nom de la nouvelle variable. Observer que le programme produit le même comportement.**

Q.21. **Changer la valeur de la variable pour que la LED clignote deux fois plus vite. Conclure sur l'un des intérêts d'utiliser une variable.**

3.2. Variable booléenne

Q.22. **Déclarer une variable booléenne nommée « clignote » sans valeur initiale. Cette variable devra être une variable globale.**

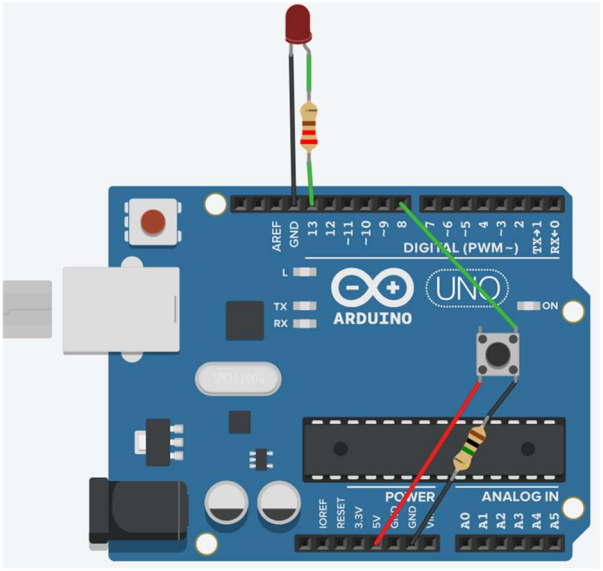
Q.23. **Au début de la *loop*, stocker dans la variable *clignote* la valeur lue sur le pin 8. S'aider du tableau de la dernière partie et des codes précédents pour choisir la bonne fonction.**

Q.24. **Remplacer la condition du *if* par la variable *clignote* et observer que le code a toujours le même fonctionnement. Assurez-vous que vous comprenez bien pourquoi le code marche ou demandez de l'aide au professeur.**

4. Utilisation d'un bouton poussoir

4.1. Montage

Vous pouvez remplacer votre montage précédent, ajouter une autre carte Arduino à côté ou bien commencer une nouvelle conception TinkerCAD.



Q.25. **Réaliser le montage à droite à l'aide d'un « bouton poussoir ». La valeur de la résistance sera fixée à 1 MΩ.**

Q.26. **Ajouter un voltmètre pour lire la tension au pin 8 et observer son comportement lorsque le bouton est appuyé ou lâché.**

Q.27. **Tester le code précédent avec des appuis longs.**

4.2. Déclenchement du clignotement

On souhaite maintenant que la LED soit éteinte au début du programme. Un appui sur le bouton déclenche le clignotement de la LED qui ne s'arrêtera plus jamais par la suite.

Q.28. **Faire en sorte que le programme ne stocke la valeur du pin 8 dans *clignotement* que si la variable contient la valeur *false*.**

Q.29. **Tester et commenter le programme.**

4.3. Gestion de la vitesse de clignotement

Q.30. **En autonomie, faire en sorte que la LED ait le comportement suivant :**

1. La LED clignote à 0,2 Hz au lancement du programme.
2. A chaque pression sur le bouton, la LED clignote deux fois plus vite. Quand le bouton est relâché, la fréquence reste accélérée.

5. Tableau des fonctions classiques d'Arduino

Nom de la fonction	Description	Argument 1	Argument 2	Retour
delay			Aucun	Rien
digitalWrite	**Digital** : 0 ou 1 **Write** : écrire Écrire une valeur valant 0 ou 1 sur un pin			Rien
digitalRead	**Digital** : 0 ou 1 **Read** : lire Lire une valeur valant 0 ou 1 depuis un pin	Pin à lire.	Aucun	*True* si le pin est à plus de 2,5V. *False* si le pin est à moins de 2,5V.