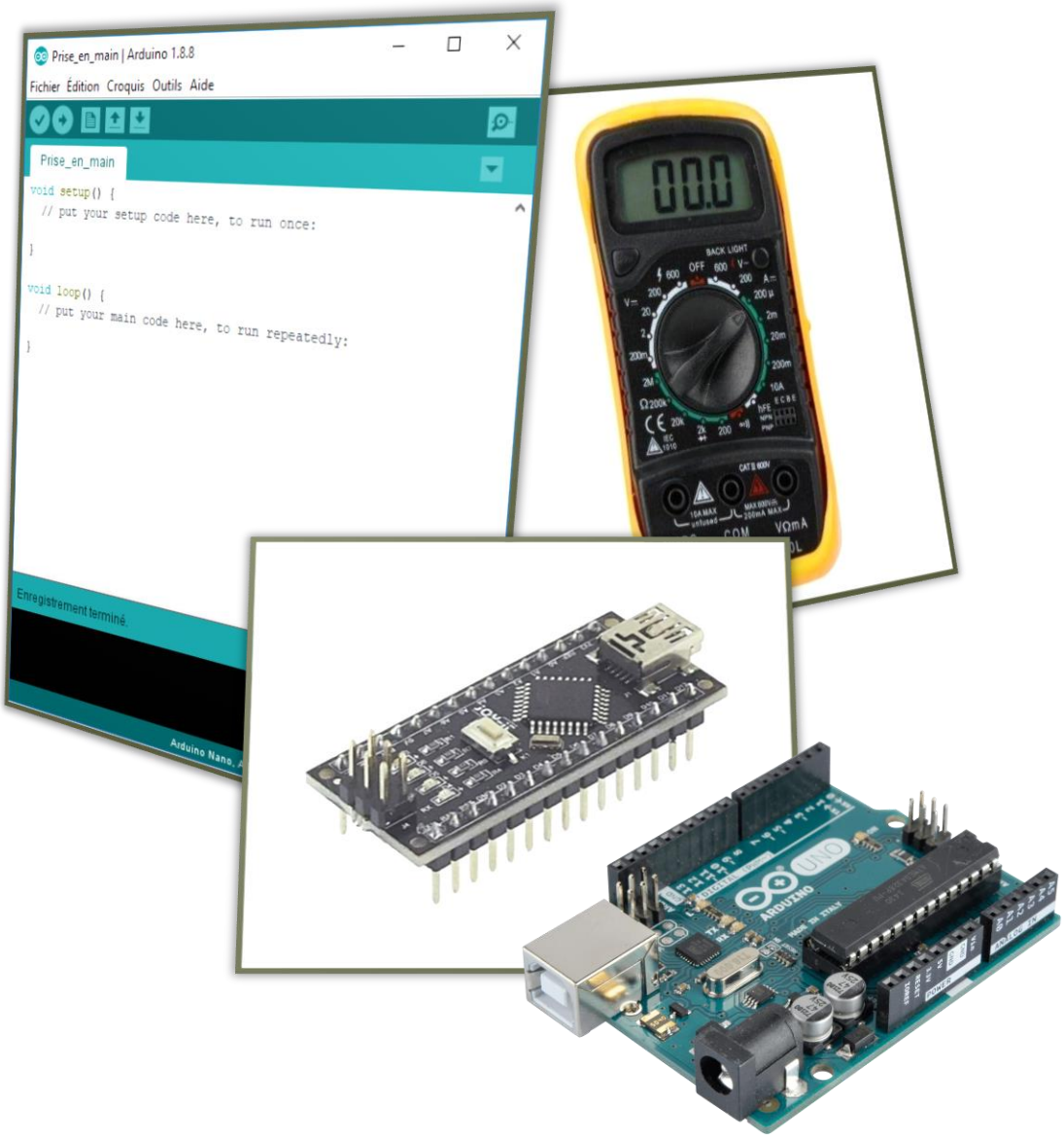
ARDUINO

PRISE EN MAIN



LYCEE CLOS MAIRE

TABLE DES MATIERES

INTRODUCTION

OBJECTIFS

Le but de ce document est de vous faire découvrir la carte Arduino Nano et son interface de programmation.

CONSIGNES

Lire ce tutoriel dans l'ordre et en entier.

A chaque encadré comme celui-ci se trouve une manipulation à effectuer ou une question à répondre.

Vous répondrez sur un document Word soigneusement présenté et intitulé

« **NOM_Arduino_Introduction** ».

Commencez par créer votre document Word de réponse et enregistrez-le dans votre espace personnel.

MATERIELS NECESSAIRES

Lorsque vous ferez de la programmation Arduino, vous serez amené à utiliser soit une carte Nano, soit une carte UNO.

Carte Nano :

- Carte Nano
- Câble USB-Mini-B
- Plaque de montage
- Multimètre + câbles

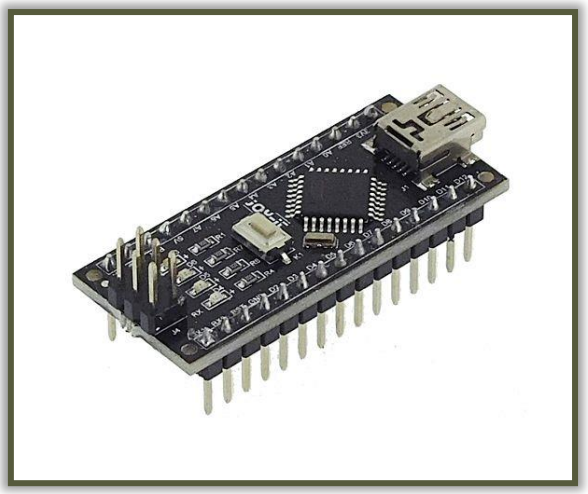
Carte UNO :

- Carte UNO
- Câble USB-B
- Multimètre + câbles

LA CARTE ARDUINO NANO

La carte Arduino est un élément fiable et capable d'exécuter un programme complet. Elle représente la solution adaptée pour contrôler l'électronique. Il en existe plusieurs types. Celles que vous utiliserez cette année sont les types Uno et Nano. Elles ont toutes les deux des similitudes et des différences.

L'Arduino Nano est une carte de dimensions 45mm x 18mm. Très légère, elle ne pèse que 5g, ce qui lui permet de s'adapter aux petits systèmes électroniques embarqués et aux petits robots.

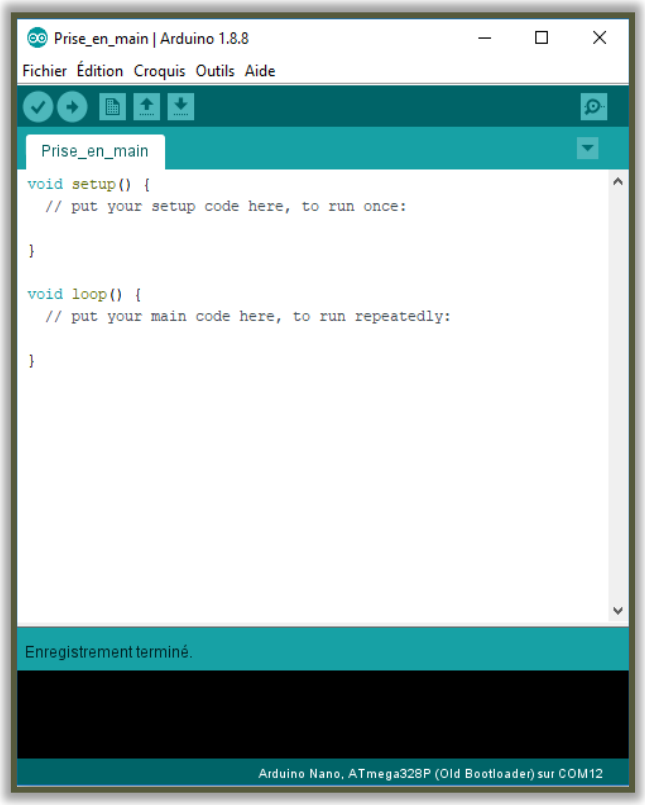


LA CARTE ARDUINO UNO

Elle a les mêmes fonctions que la carte Nano mais est plus grosse en taille et a de meilleures performances.

L'INTERFACE LOGICIEL ARDUINO

L'interface logiciel de Arduino permet de programmer la carte Arduino en langage C/C++.



ANALYSE DE LA CARTE ARDUINO NANO ET UNO

LES COMPOSANTS ET CARACTERISTIQUES PRINCIPALES

- Microprocesseur : ATMega328

Le microprocesseur est le cerveau de la carte. Votre programme créé sur l'ordinateur sera transmis à celui-ci pour qu'il traite les données de la façon dont vous lui avez demandé.

- Mémoire flash : 32 ko

La mémoire flash est la mémoire de la carte Arduino. Elle permet de stocker le programme que vous lui transmettez. Cette mémoire est de 32ko soit 32 000 octets.

L'ordinateur acquiert, traite et communique les données en binaire.
Le binaire est un langage constitué seulement de 0 et de 1, qu'on appelle des bits.

Un octet est constitué d'une suite de 8 bits.

Question 1 - Convertir 32ko en bits.

- Mémoire SRAM : 2 ko

La mémoire SRAM est la mémoire dynamique. Elle servira à stocker les valeurs des variables qui pourront changer durant le fonctionnement du programme.

- 14 broches d'entrées et sorties, dont 6 broches PWM (D0 à D13 – D0 et D1 étant notée RX0 et TX1)

Ces broches d'entrées et sorties permettent de communiquer avec l'extérieur soit en acquérant une information (entrée) soit en communiquant une information (sortie).

Ces broches ne peuvent prendre que 2 états binaires : soit 0, soit 1 (soit LOW : état BAS, soit HIGH : état haut). Autrement dit la tension sur ces broches sera soit de 0 soit de 5 Volts.

RX et TX sont des bornes spéciales qui permettent de communiquer en liaison série. Nous verrons leur utilisation plus tard.

Les broches PWM (signifiant Pulse Width Impulsion ou MLI en français : Modulation de Largeur d'Impulsion) permet de d'alterner la sortie (toujours avec 0 et 1) et de faire varier le temps pour lequel la valeur de sortie de la broche est à 1 ou à 0. Cela permet par exemple de contrôler la vitesse des moteurs à courant continu (plus le temps pour lequel la broche est à 1 est long, plus le moteur tournera vite). Nous étudierons cela en détail plus tard.

- 6 broches d'entrées analogiques 10 bits

Les entrées analogiques peuvent prendre plusieurs valeurs puisqu'elles ne sont pas codées sur un seul bit. Elles sont codées sur 10 bits. Ce qui veut dire qu'elles peuvent aller de 0000000000 à 1111111111 en binaire.

Avec 1 bit on a 2 possibilités : 0 ou 1.
Avec 2 bits on a 4 possibilités : 00, 01, 10, 11.
Avec 3 bits on a 8 possibilités : 000, 001, 010, 011, 100, 101, 110, 111.

Avec **n** bits nous avons $\mathbf{2^n}$ possibilités.

Question 2 – Calculer le nombre de possibilités avec 10 bits.

Le nombre que vous avez trouvé question 2 est donc le nombre de valeur que pourront prendre les entrées analogiques.

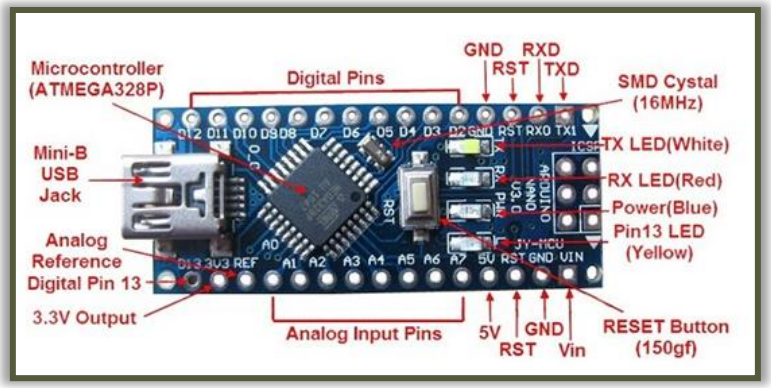
- Fréquence d'horloge : 16 MHz

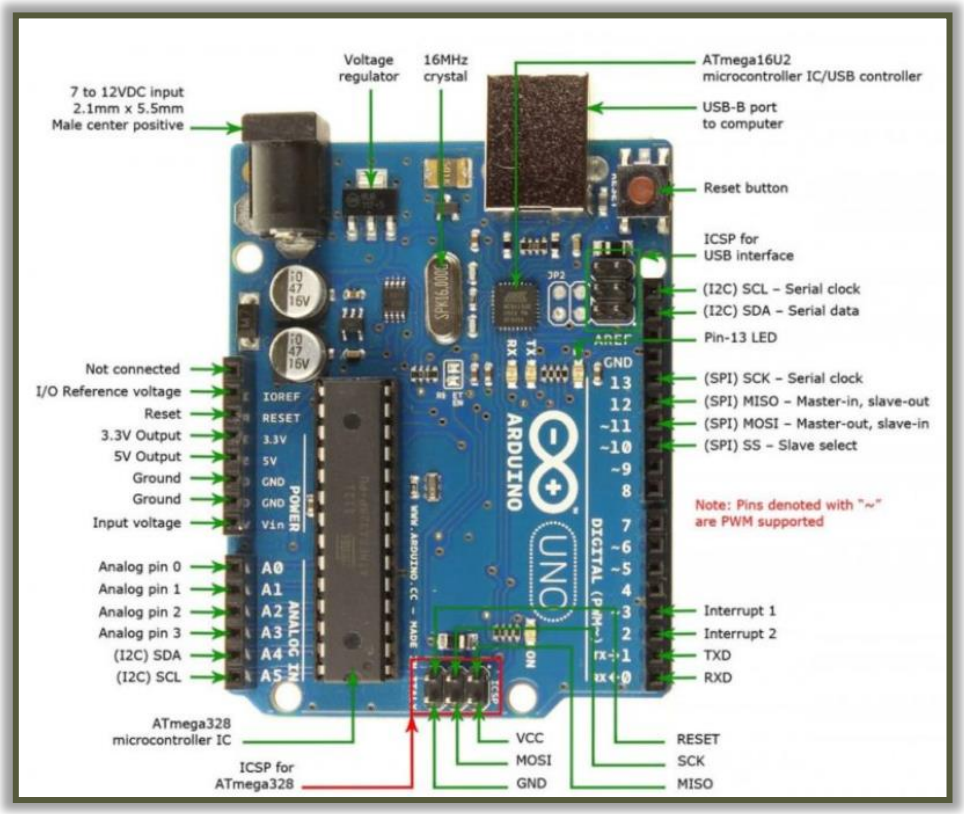
La fréquence d'horloge définit le nombre d'opération qu'il est possible d'effectuer par seconde.

16 MHz signifie 16 Méga Hertz.
Le Hertz est l'unité de la fréquence.
Méga est un préfixe d'unité signifiant qu'il faut multiplier le nombre par 10^6 soit 1 000 000.
La fréquence est définie par l'inverse de la période : $f = \frac{1}{T}$

Question 3 – Calculer la période T.

Le nombre que vous avez trouvé question 3 est le temps pour lequel le microprocesseur effectue une opération.

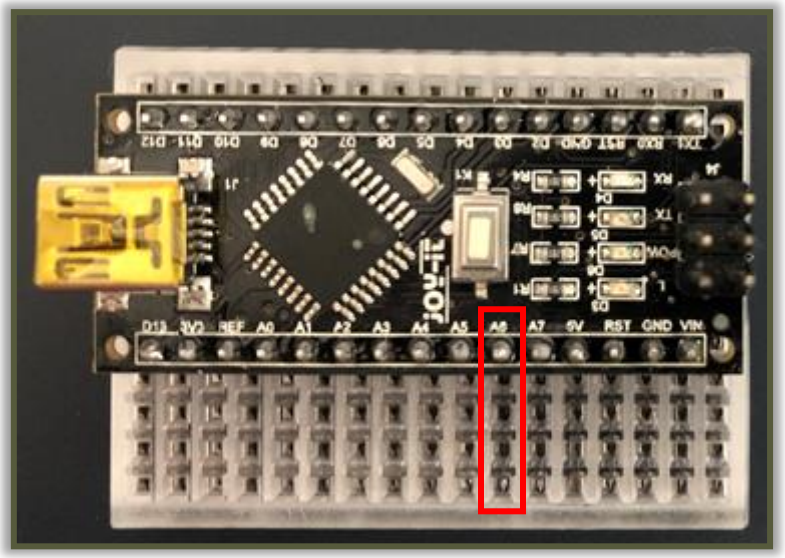




INSTALLATION PLAQUE DE MONTAGE ET MESURES

Pour simplifier les câblages lors des différents tests nous utiliserons des plaques de montages. Celles-ci permettent d'effectuer des connexions sans soudure.

Poser la carte Nano sur la plaque de montage comme sur la photo ci-dessous de façon à laisser une rangée de trou aux bornes digitales et 4 trous aux bornes analogiques.



Grâce à ce montage la borne A6 est maintenant accessible via les 4 connections entourées en rouge.

Connecter la carte à l'ordinateur grâce au câble USB.

La LED POW devrait s'allumer signifiant que la carte est bien alimentée en énergie.

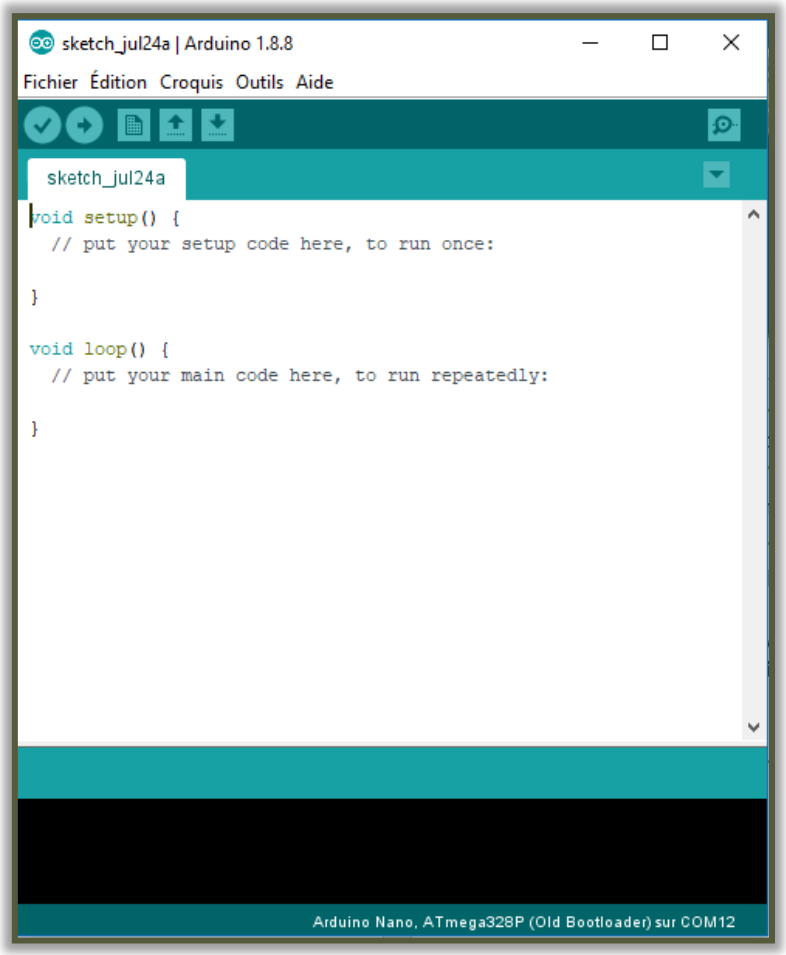
Vérifier à l'aide du multimètre que les connexions à la plaque sont bien effectuées :
Mettre le calibre du multimètre sur 20 Volts en continue.
Mettre la borne positive (V) sur une des bornes du 5V.
Mettre la borne négative (COM) sur une des bornes du GND (Ground).

Question 4 – Relever la tension obtenue.

CREATION D'UN PROGRAMME

Sur le bureau de l'ordinateur, ouvrir le dossier « STIDD » et ouvrir le logiciel « Arduino ».

A l'écran s'affiche alors la fenêtre suivante :



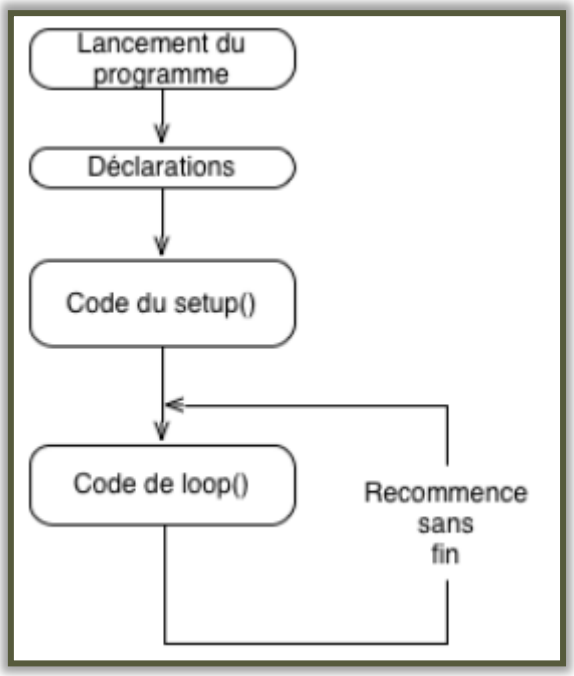
Ceci est l'interface de programmation pour Arduino où vous allez pouvoir rentrer du code pour programmer la carte.

Quelques lignes de bases sont déjà écrites :

- « void setup() » signifie que la fonction « setup » est créée. Cette fonction est particulière à Arduino et sera exécutée qu'une seule fois au début du programme. **Cette fonction exécutera une seule fois le code qui sera situé entre les deux accolades qui la suivent.**
- « // put your setup code here, to run once » est un commentaire puisque la phrase est précédée de 2 slashs. Un commentaire signifie que cette ligne ne sera pas exécutée en tant que code. Elle permet à l'utilisateur de noter des informations, des explications sur son code. C'est très utile lorsque vous écrivez un programme et que quelqu'un d'autre est susceptible de s'en servir. Cela lui permettra de comprendre plus rapidement votre code.
- « void setup() » signifie que la fonction « loop » est créée. Cette fonction est particulière à Arduino et sera exécutée en boucle après que la fonction setup ai été exécutée. **Cette fonction exécutera en boucle le code qui sera situé entre les deux accolades qui la suivent.**

Avant le setup une partie importante sera aussi exécutée : la partie déclaration. Cette partie permettra de déclarer plusieurs éléments dont le programme sera dépendant comme des bibliothèques (permettant l'utilisation de fonction déjà codées) ou des variables (permettant de stocker des valeurs).

Un programme s'exécutera donc comme sur cet algorigramme (un algorigramme est une manière graphique de représenter un programme) :



ENREGISTREMENT D'UN PROGRAMME

Cliquer sur « Fichier » puis « Enregistrer sous… ».
Enregistrer le fichier dans votre espace personnel à l'endroit où vous avez enregistrer votre document Word en le nommant

« **NOM_Arduino_Introduction** ».

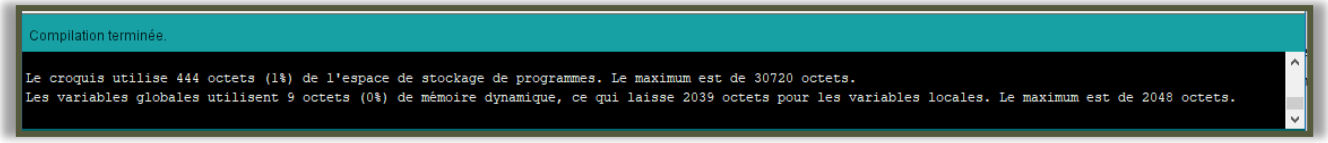
VERIFICATION D'UN PROGRAMME

La vérification d'un programme permet de s'assurer qu'il n'y a pas d'erreur de syntaxe ou de référence dans votre programme.

Pour vérifier un programme il suffit de cliquer sur le bouton « Vérifier » en haut à droite de la fenêtre.

Cliquer sur « Vérifier ».

Dans la console en bas de la fenêtre d'affiche alors des informations concernant la vérification de programme. Il n'y a pas eu de lignes en rouge, le programme est donc correct.



Il est important de vérifier son programme régulièrement lorsque vous faites de la programmation. Il est aussi important de prendre le temps de lire et déchiffrer les erreurs lors de la vérification pour comprendre où est-ce qu'il y une erreur dans votre programme.

TELEVERSEMENT D'UN PROGRAMME

Le téléversement d'un programme permet de transférer le programme que vous avez créé sur le logiciel vers la carte Arduino.

Avant de faire cela il faut vérifier plusieurs points :

1. Vérifier que votre carte Arduino est bien branchée à l'ordinateur via le câble USB.
2. Dans « Outils », vérifier que le « Port » est bien dirigé vers un « COM ».
3. Dans « Outils », vérifier que le « Type de carte » est le bon sélectionné :
    - Si vous travaillez avec l'Arduino UNO alors il faut sélectionner « Arduino Genuino/Uno ».
    - Si vous travaillez avec l'Arduino Nano alors il faut sélectionner « Arduino Nano » et sélectionner en plus dans « Processeur », « ATmega328P (Old Bootloader) ».

Il suffit ensuite de cliquer sur le bouton « Téléverser » situé à gauche du bouton « Vérifier » pour téléverser votre programme dans la carte Arduino.

Après avoir effectué toutes les vérifications décrites ci-dessus, cliquer sur « Téléverser ».

Si aucuns messages rouges ne s'affichent dans la console en bas de la fenêtre alors votre programme est maintenant dans la carte Arduino. Il est normal que rien n'est changé puisque ce programme est vide.

PROGRAMMATION DU MONITEUR SERIE

Le moniteur série est un outil d'Arduino permettant la communication entre la carte et l'ordinateur lorsqu'un programme est en cours de fonctionnement.

La carte Arduino pourra alors envoyer plusieurs informations à l'ordinateur en les affichant sur le moniteur série. Cela est très utile pour débugger un programme par exemple en demandant d'afficher une valeur de variable à un moment précis.

INITIALISATION

La première étape consiste à initialiser le moniteur série dans le « setup » du programme afin de définir le débit d'envoie et de réception des données.

Dans le « setup » du programme entre les accolades, ajouter la ligne : « **Serial.begin(9600);** ».

Après chaque ligne d'instruction il faut un point-virgule !

AFFICHAGE

Nous allons ensuite demander à la carte d'afficher un texte simple dans le moniteur série.

Dans la « loop » du programme entre les accolades, ajouter la ligne : « **Serial.print("Hello STIDD");** »

Vous devez avoir un programme comme celui-ci :

```
Arduino_Introduction §
void setup() {

  Serial.begin(9600);

}

void loop() {

  Serial.print("Hello STIDD");

}
```

Téléverser votre programme dans la carte.

Au moment où vous téléversez le programme sur la carte vous pouvez observer que la LED RX de la carte s'allume signifiant que la carte est en train de recevoir des données.

Vous pouvez ensuite observer que la LED TX s'allume signifiant que la carte envoie des données.

Pour ouvrir le moniteur série il suffit d'appuyer sur le bouton « Moniteur série » situé en haut à gauche de la fenêtre.

Ouvrir le moniteur série et observer ce qu'il se passe.

Question 5 – Expliquer le résultat obtenu.

Changer de place la ligne « **Serial.print("Hello STIDD");** » pour la mettre dans le setup du programme.
Téléverser le programme et observer le moniteur série.

Question 6 – Expliquer le nouveau résultat.

Ajouter les deux lignes de commandes suivantes dans la loop du programme :

```
Serial.println("Hello STIDD");
delay(1000) ;
```

Téléverser le programme et observer le moniteur série.

Question 7 – Expliquer le nouveau résultat.
Question 8 – Expliquer ce que fait la fonction delay.
Question 9 – Expliquer la différence entre la fonction « Serial.print » et « Serial.println » (Rappelez-vous le résultat de la question 5).

Observer la LED TX de l'Arduino.

Question 10 – A quelle fréquence (en Hz) clignote-t-elle ?

PROGRAMMATION D'UNE LED

Sur les cartes Arduino, il y a toujours une LED qui est programmable. Le but ici est de programmer cette LED pour l'allumer et l'éteindre. Sur ces cartes Arduino il est possible de piloter une LED intégrer à la carte grâce à la borne digital 13.

Supprimer l'intérieur de la loop.

Avant le setup, déclarer une variable LED prenant la valeur 13 correspondant au numéro de la borne que nous voulons piloter : Ecrire la ligne « **const int LED = 13;** » avant le setup.

Après chaque ligne d'instruction il faut un point-virgule !

const : permet de définir la variable LED comme étant une constante : elle ne pourra donc pas changer au cours du programme.

int : permet de définir la variable LED comme étant un nombre entier.

Nous devons maintenant définir que nous voulons utiliser la borne 13 comme une sortie.

Dans le setup, écrire la ligne « **pinMode(LED, OUTPUT);** ».

La fonction pinMode permet de définir si on utilise une borne comme une entrée ou une sortie. Ici, nous informons que nous utilisons la borne LED (donc la borne 13), comme étant une sortie.

Nous souhaitons maintenant allumer la LED.

Dans la loop, écrire la ligne « **digitalWrite(LED, HIGH);** ».

La fonction digitalWrite signifie que nous écrivons une valeur sur une borne. Ici, nous définissons la valeur de la borne LED (donc la borne 13), à la valeur haute (soit 1 en binaire, soit 5 Volts en tension).

Vous devez obtenir un programme comme celui-là :

Arduino_Introduction §

```
const int LED = 13;

void setup() {

  Serial.begin(9600);
  Serial.print("Hello STIDD");
  pinMode(LED, OUTPUT);

}

void loop() {

  digitalWrite(LED, HIGH);

}
```

Téléverser le programme et observer l'allumage de la LED orange sur la carte Arduino.

La ligne de commande « digitalWrite(LED, LOW); » permet donc d'éteindre la LED.

En utilisant le programme déjà effectué, la ligne de commande ci-dessus et la fonction delay, créer un programme qui permet de faire clignoter la LED à une fréquence de 2 Hz.

Question 11 – Copier-coller votre programme.