01 Avoir les bons outils

Pour devenir un bon artisan, il faut avoir de bons outils et savoir bien les utiliser. Pour le travail manuel, cela demande de s'approprier la précision du geste afin de bien guider l'outil. Pour le travail intellectuel ou créatif, cela demande de bien connaître les possibilités des différents outils logiciels (traitement de texte, logiciel de dessin ou de retouche photos, montage vidéo, etc.) afin de choisir l'outil le plus approprié et l'utiliser à bon escient. Je vais donc faire **un tour assez descriptif des outils qui sont à votre disposition** pour générer du code informatique ou pour concevoir un montage ou encore pour travailler en équipe.

Logiciel d'apprentissage de la conduite de trains à la SNCF

Salle de contrôle de Miniatur Wunderland à Hambourg

1.1 / NOUVEAU SITE D'ARDUINO

Le site internet d'**Arduino** a fait peau neuve au début de 2025. Il faut dire qu'il change régulièrement de look car c'est là un moyen efficace de faire savoir qu'on est vivant et en bonne santé, deux choses indispensables dans un domaine en constante évolution comme peut l'être la microélectronique programmable. Alors si cela se trouve, ce que j'écris aujourd'hui n'aura peut-être déjà plus cours lorsque ce livre sera publié. Ce n'est pas si grave car si la forme change, le fond reste identique et tout ce qui existe aujourd'hui se retrouvera dans les versions de demain. Rappelons l'adresse du site qui est toujours la même : **https://www.arduino.cc/** .

Arduino étant un site commercial, tout est fait pour nous inviter à acheter et la page s'ouvre sur une vidéo grand format présentant les dernières nouveautés. Il serait illusoire de vouloir décrire en quelques lignes un site aussi évolué et complet, aussi vais-je essayer de ne donner que quelques points de repère. Le bandeau supérieur permet de choisir plusieurs catégories : **professionnels** (professionnals), **éducation** (education), **créateurs** (makers). Ce guide va faire de vous des créateurs et cette rubrique risque de vous intéresser ou vous donner des idées. Sur la droite, on trouve de quoi faire son marché : **produits** (products) à considérer au sens large (matériel et logiciel), **communauté** (community), **documentation** (documentation) et **boutique** (shop), la seule icône à être bien repérable en étant entourée de la couleur fétiche d'Arduino, le « teal » du langage HTML (une nuance de vert).

Une petite flèche vers le bas indique que le menu est développable. Ainsi, la catégorie « produits » permet d'accéder à deux autres catégories, le matériel (**hardware**) et le logiciel (**software**). Dans cette dernière catégorie, on retrouve ce qui permet de télécharger le logiciel IDE comme le montre la **figure 1-1**.

Dans la catégorie « **matériel** », on accède à plusieurs cartes comme la fameuse UNO, traduite par le navigateur par ONU ; en effet, UNO est le sigle anglais pour United Nation Organization, qui se traduit en français par Organisation des Nations Unies ou ONU, mais il ne s'agit pas d'une nouvelle carte ! De même, le **Cloud** est traduit par le Nuage alors que le cloud a su s'imposer dans la langue française. Le traducteur de page internet peut donc rendre service à certains d'entre vous mais il faut tout de même s'en méfier (ou en sourire) et mieux vaut s'habituer à quelques mots d'anglais. Les prochaines captures d'écran seront faites à partir du site en anglais…

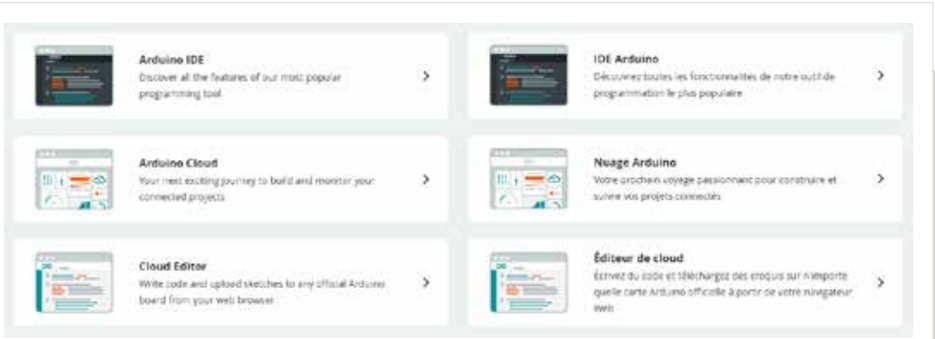

Figure 1-1
Comment programmer les cartes, version anglaise ou française traduite par le navigateur.

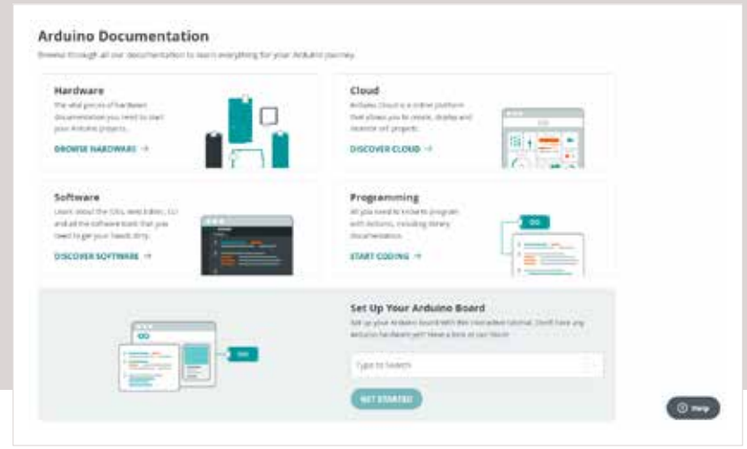

Figure 1-2
Accès à la documentation (source Arduino)

La catégorie « **matériel** » propose d'autres cartes, des kits de plusieurs sortes, ainsi que la possibilité d'enregistrer un produit Arduino.

La catégorie « **community** » vous donne accès au forum d'Arduino, au dépositaire GitHub dont je reparlerai, ainsi qu'à la chaîne vidéo YouTube d'Arduino. Je vous laisse explorer tout cela si vous en avez envie.

La catégorie « **documentation** » vous permet d'accéder à toutes sortes de documentation sur les cartes, la programmation, les librairies, le Cloud, des tutoriels et surtout la **référence du langage Arduino**, c'est-à-dire des fonctions proposées par Arduino pour programmer simplement les cartes à microcontrôleur (**figure1-2**).

La **loupe** permet de faire une recherche dans le site Arduino et la touche Cloud permet d'accéder au cloud à condition d'avoir un compte (**figure 1-3**).

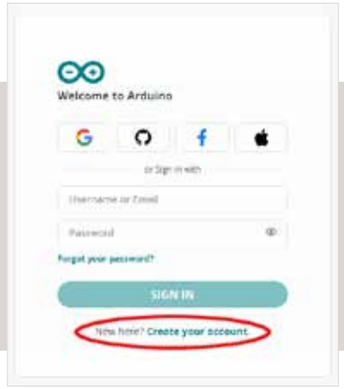

Figure 1-3
(source Arduino)

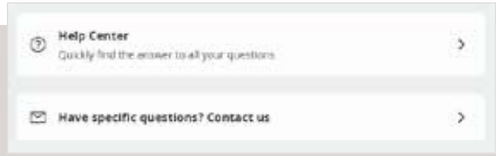

Figure 1-4
Obtenir de l'aide (source Arduino)

Vous pouvez aussi accéder à un **centre d'aide** (Help center) où vous pouvez voir les questions fréquemment posées ; si cela ne suffit pas, vous pouvez écrire directement à l'équipe Arduino, mais ne prenez pas l'habitude de le faire sans avoir déjà cherché par vous-même la réponse (**figure 1-4**).

Vous savez l'essentiel sur le site Arduino : comment télécharger la dernière version du logiciel IDE, où trouver de la documentation, où trouver les différentes cartes, comment trouver de l'aide. Comme je vous l'ai dit, ce site est très riche et vous pourrez y passer des heures à le consulter. Consultez-le souvent afin de vous familiariser avec sa structure et avoir une meilleure connaissance de ce qu'on y trouve.

1.2 / PROGRAMMER FACILEMENT LES MICROCONTRÔLEURS AVEC L'IDE 2

Le logiciel qui permet de programmer les cartes Arduino est distribué aujourd'hui en version 2 (2.3.6 à l'heure où ces lignes sont écrites, mais certaines figures de ce tome 2 ont été faites avec la 2.3.5) ; **cette version remplace définitivement la version 1** (1.8.19) qui n'est plus mise à jour. Les fonctionnalités sont quasiment les mêmes mais l'interface est plus moderne tout en restant aussi intuitive.

La **figure 1-5** montre à quoi ressemble l'interface après avoir choisi le français comme langue (voir plus loin). On reconnaît une première ligne de 5 menus (**Fichier, Modifier, Croquis, Outils, Aide**), une deuxième ligne d'icônes, sur la gauche « vérifier », « télécharger », « débogage », puis une fenêtre pour sélectionner la carte et sur la droite « traceur » et « moniteur ».

Le **moniteur série** s'affiche maintenant en bas de l'écran comme le montre la **figure 1-6** : pour le refermer, une petite croix à droite du titre « Moniteur série » ou recliquer sur l'icône qui l'a fait apparaître. Pour connaître l'utilisation des autres icônes du moniteur, il suffit de les pointer avec la souris (défilement automatique, horodatage, effacement de la sortie). Il y a deux zones : une fenêtre pour entrer le message à envoyer à la carte Arduino, et le reste pour afficher les données issues de la carte.

Le **traceur série** s'ouvre quant à lui dans une nouvelle fenêtre (**figure 1-7**).

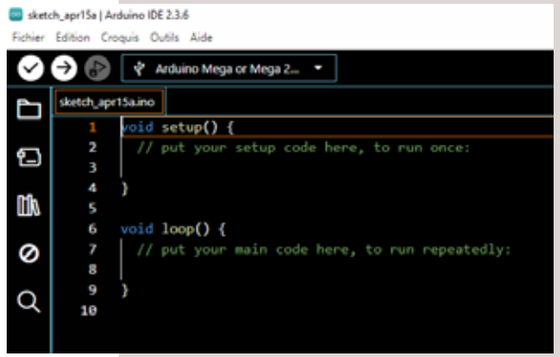

Figure 1-5

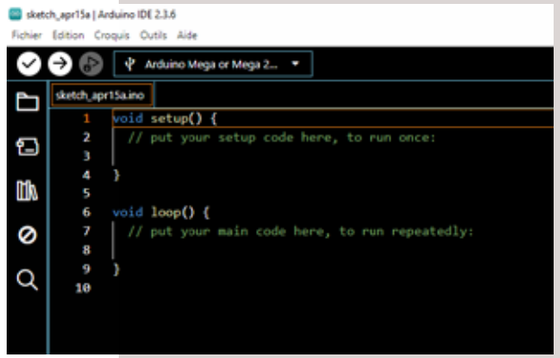

Figure 1-6

Figure 1-7
Interface du traceur série

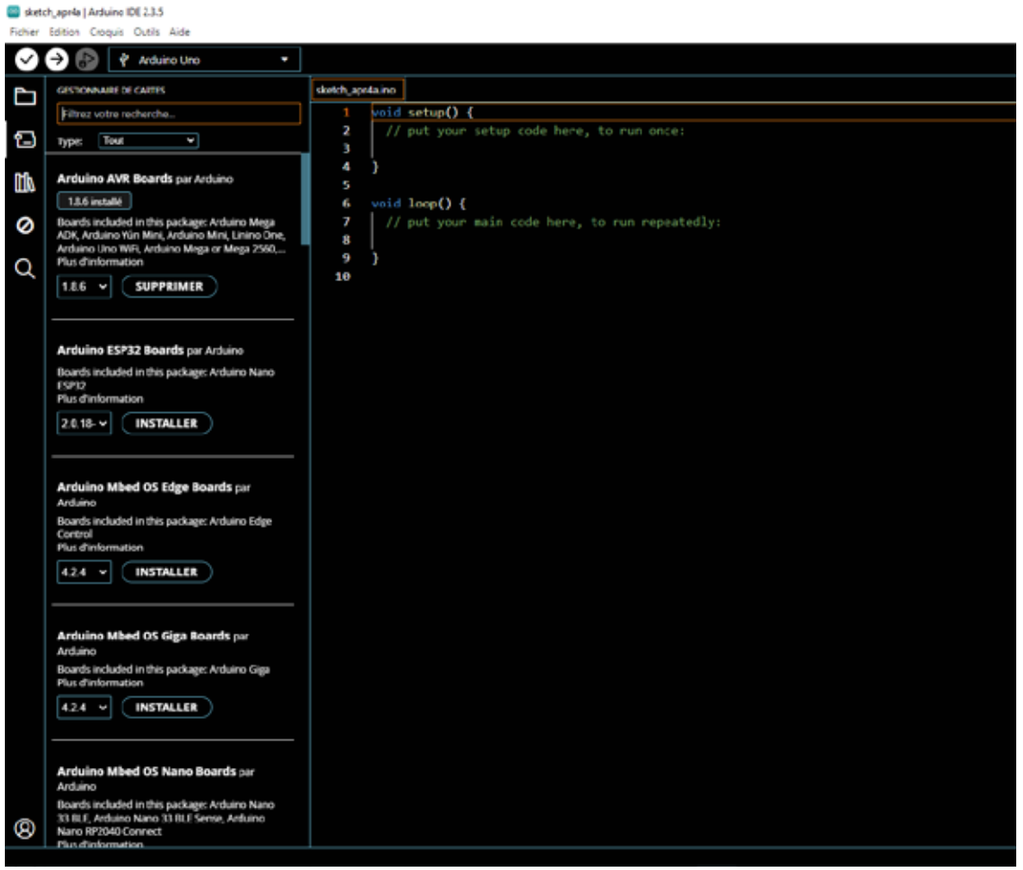

Figure 1-8

Sur la partie gauche, 6 icônes positionnées verticalement, sont fréquemment utilisées.

La première (de haut en bas) permet de **sélectionner votre carnet de croquis**, qui n'est autre que le répertoire Arduino dans votre dossier Documents, là où sont enregistrés vos sketches (programmes). Vous pouvez créer dans ce répertoire Arduino des sous-répertoires, par exemple un par projet, et enregistrer vos programmes dedans ; cette méthode est plus efficace pour retrouver facilement tout ce qui concerne un projet et je vous conseille donc de l'adopter. Cette icône permet aussi d'accéder à votre carnet de croquis du Cloud.

La deuxième icône permet **d'installer une carte** avec la possibilité de rentrer son nom dans la case de recherche ainsi que son type (**figure 1-8**) : nous sommes là dans le **Gestionnaire de cartes**. Installer une carte permet à l'IDE de travailler avec un nouveau type de carte, qui n'est d'ailleurs pas forcément de la marque Arduino, et **cela est différent du fait de sélectionner une carte**, celle avec laquelle vous travaillez pour votre projet, qui se fait dans la case située à droite des icônes de vérification et téléversement.

La troisième icône nous amène au **Gestionnaire de bibliothèque**, avec également la possibilité de rentrer son nom, son type et le sujet concerné ; tapez dans la case de recherche le nom LightEffect et vous tomberez sur ma bibliothèque spécialement écrite pour des animations lumineuses pour le modélisme ferroviaire (**figure 1-9**).

Figure 1-9

La quatrième icône sert au **débogage** à condition que la carte le permette ; cette icône change d'ailleurs d'aspect selon que le débogage est possible ou non comme le montre la **figure 1-10**.

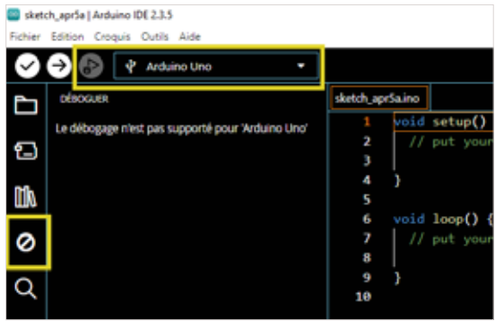

Figure 1-10a
Carte ne permettant pas le débogage

Figure 1-10b
Carte permettant le débogage

Enfin, **la cinquième icône** (loupe) permet de **faire une recherche**, comme pour la plupart des logiciels. Cette icône permet aussi de remplacer un mot par un autre, par exemple si vous voulez changer le nom d'une variable. L'icône tout en bas à gauche permet d'accéder au Cloud (éditeur ou IoT : nous en parlerons plus loin) et son aspect change selon que vous êtes ou non connecté.

Juste à droite de ces icônes se trouve l'espace dans lequel vous entrez votre programme avec les fonctions setup et loop que vous n'avez plus qu'à compléter. Pour faciliter vos saisies, il y a une **présentation automatique des fonctions** et des arguments comme le montre la **figure 1-11** ; c'est parfois un peu déroutant au début mais

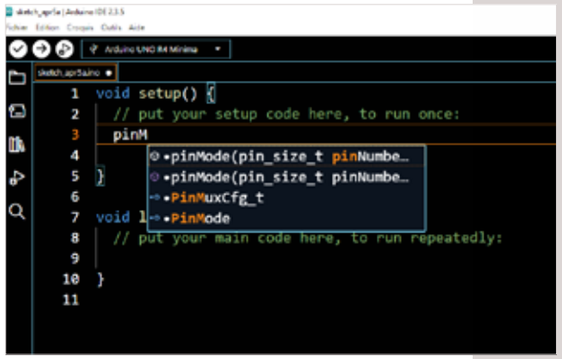

Figure 1-11

on s'y habitue très vite et vous pouvez toujours inhiber cette fonction dans le menu Préférences (voir plus bas).

Figure 1-12
écran de téléchargement et d'installation du logiciel

Contrairement à la version 1, il n'y a pas d'icône pour sauvegarder un programme, il faut passer par le menu Fichier puis « **Sauvegarder sous** » ou bien « **Save** » (qui n'a pas été traduit, tout comme « **Paste** » qui signifie coller dans le menu Edition).

Pour installer l'IDE 2, tout commence à cette page : **https://www.arduino.cc/en/software**. La version 2 existe pour tous les systèmes d'exploitation (**figure 1-12**). Il suffit donc de choisir et de se laisser guider pour les différentes étapes. Il faut au moins Windows 10 pour cette version 2 : comme pour la version 1, je vous conseille de **télécharger le logiciel à partir d'un fichier ZIP**. Une fois que vous avez tout extrait, vous obtenez un répertoire que vous pouvez placer sur votre bureau ou dans vos documents. Dans ce répertoire, faites un clic droit sur Arduino IDE.exe puis **Envoyer vers > Bureau** (créer un raccourci). Vous obtenez sur votre bureau l'icône qui vous permet de lancer l'IDE.

Une fois l'IDE ouvert, aller dans le menu **Fichier > Préférences** : une fenêtre s'ouvre (**figure 1-13**) et vous permet de régler la taille de la police pour écrire vos programmes ainsi que le thème de couleur (fond sombre ou clair selon vos préférences). Cochez également les cases « **Afficher la sortie de débogage verbeuse pendant** » car cela vous permettra **d'avoir des messages** durant la compilation (vérification) et le téléversement (programmation). Vous pouvez aussi cocher les cases « **Vérifier le code après le téléversement** » et « **Suggestion rapide pour l'éditeur** » qui vous proposera alors la syntaxe de chaque fonction d'Arduino comme expliqué un peu plus haut.

Figure 1-13

Profitez-en pour régler la langue du logiciel (**figure 1-14**). Cliquez ensuite sur OK dans le bas de la fenêtre pour que vos modifications soient prises en compte. Pour que la langue soit prise en compte, il faut refermer puis rouvrir l'IDE.

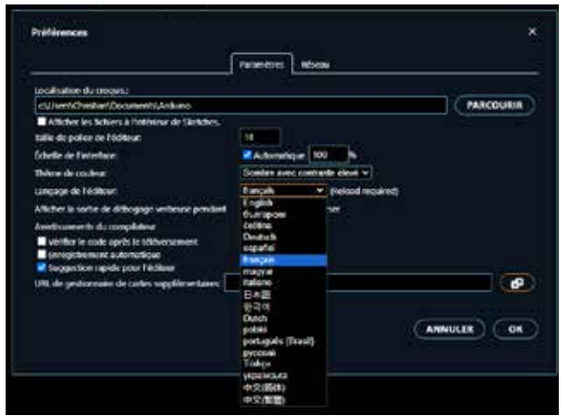

Figure 1-14

L'IDE 2 est très simple à utiliser et en même temps très riche de fonctionnalités. On ne peut pas les décrire toutes mais avec un peu de pratique, vous allez acquérir les bons réflexes. Avant de clore ce paragraphe, nous allons voir quelques exemples sur l'utilisation des menus de l'IDE 2.

Menu **Fichier > Avancé > Référence des raccourcis clavier** vous permet d'accéder à la liste de tous les raccourcis clavier et ils sont nombreux. Pour fermer cette liste, c'est la croix à côté de raccourcis clavier (**figure 1-15**).

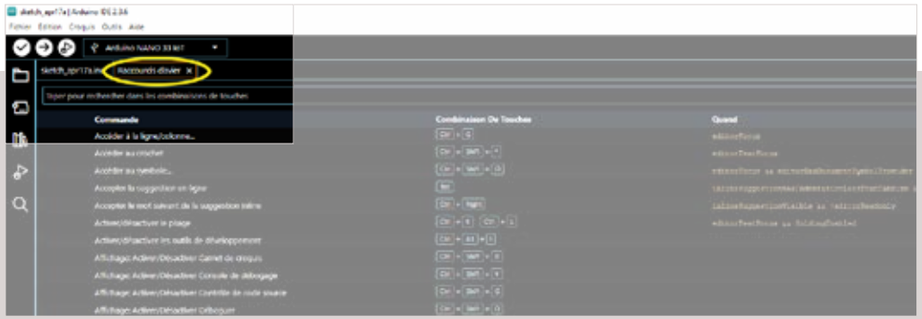

Figure 1-15


```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8    for(int i = 2; i <= 10; i++) {
9    digitalWrite(i, HIGH);
10   if(i == 4) {
11   digitalWrite(i, LOW);
12   }
13   }
14 }
15
```

```
1  void setup() {
2    // put your setup code here, to run once:
3  }
4
5  void loop() {
6    // put your main code here, to run repeatedly:
7    for (int i = 2; i <= 10; i++) {
8      digitalWrite(i, HIGH);
9      if (i == 4) {
10       digitalWrite(i, LOW);
11     }
12   }
13 }
14
```

Figure 1-16

Menu **Edition > Commenter/Décommenter** vous permet de commenter (ou décommenter) une partie du programme que vous avez sélectionnée. Ceci peut être pratique pour tester un programme et retirer ce qui vous paraît inutile, tout en gardant la possibilité d'y revenir éventuellement. Lorsque le programme sera au point, il sera temps de faire le ménage.

Menu **Edition > Formatage automatique ou bien Outils > Formatage automatique** (ou encore CTRL + T) permet de formater le texte du programme pour avoir une meilleure lisibilité comme le montre la **figure 1-16** où la partie gauche a été intentionnellement écrite sans formatage (indentation).

Menu **Croquis > Exporter les binaires compilés** permet de sauvegarder le fichier hexadécimal créé par la compilation du programme. Ce fichier est le code binaire qui est envoyé au microcontrôleur lors du téléversement, même s'il est présenté sous forme hexadécimale, et **ce code binaire contient tout ce qui est nécessaire au fonctionnement du projet**, y compris les parties de bibliothèques nécessaires. Parfois, si les bibliothèques évoluent, le programme peut ne plus être fonctionnel ; en reprogrammant la carte à partir de ce fichier de binaires, on retrouve son fonctionnement. La technique pour programmer une carte à partir des binaires compilés a été décrite dans cet article : **https://locoduino.org/spip.php?article279 .**

Enfin, le menu **Outils > Firmware Updater** permet la mise à jour du firmware d'une carte ou de vérifier si de telles mises à jour existent. Le **firmware est le programme qui permet à la carte de fonctionner** et notamment de recevoir le programme lors de la phase de téléversement. Si une mise à jour est nécessaire, il suffit de suivre les indications à l'écran. J'ai réalisé une mise à jour d'une carte Nano33IoT pour qu'elle soit compatible avec l'IoT d'Arduino et cela s'est fait sans problème.

1.3 / TRAVAILLER EN LIGNE AVEC LE CLOUD ARDUINO

Arduino propose un **éditeur de programme en ligne**, c'est-à-dire via une page internet. Le design ressemble beaucoup à l'IDE 2 comme le montre la **figure 1-17** où un nouveau sketch est ouvert. Cette fois, non seulement les fonctions setup et loop sont proposées, mais aussi un espace de commentaires **permettant de décrire ce que réalise le programme**.

Pour travailler avec le Cloud d'Arduino sur Windows, macOS ou Linux, vous devez installer le **Cloud Agent** (anciennement Create Agent), un plugin qui permet la **communication série entre votre carte et le Cloud d'Arduino**. Le système vous demandera de le faire si ce n'est pas déjà fait et il suffit de suivre les indications à l'écran. Certaines cartes, un peu anciennes, peuvent avoir besoin d'une mise à jour de leur firmware : encore une fois, il suffit de suivre les indications, la mise à jour devant parfois être faite avec une connexion en USB et le Cloud Arduino qui s'occupera de tout.

On reconnait les boutons vérifier et téléverser ainsi qu'un bouton permettant de choisir la carte utilisée. Les icônes le long du bord gauche permettent d'accéder aux **programmes** (Files), aux **exemples** (Examples) pour chaque catégorie comme le proposait déjà l'IDE 1, aux **bibliothèques** (Libraries) ainsi qu'aux **références** du langage Arduino (Reference). La **figure 1-18** montre les différentes possibilités. L'icône en bas à gauche est l'équivalent du menu **Préférences** (Settings).

On ne va pas s'étendre sur cet éditeur en ligne qui s'utilise comme l'IDE : il a deux avantages, d'une part il n'y a rien à installer sur votre ordinateur, d'autre part vos projets et programmes sont accessibles partout dans le monde et à n'importe quel moment, à partir de n'importe quel point d'accès à internet. Revers de la médaille, la confidentialité n'est peut-être pas complètement assurée, même si l'option « Private » est sélectionnée dans le menu Share Sketch.

Figure 1-17

L'éditeur de programme en ligne (source Arduino)

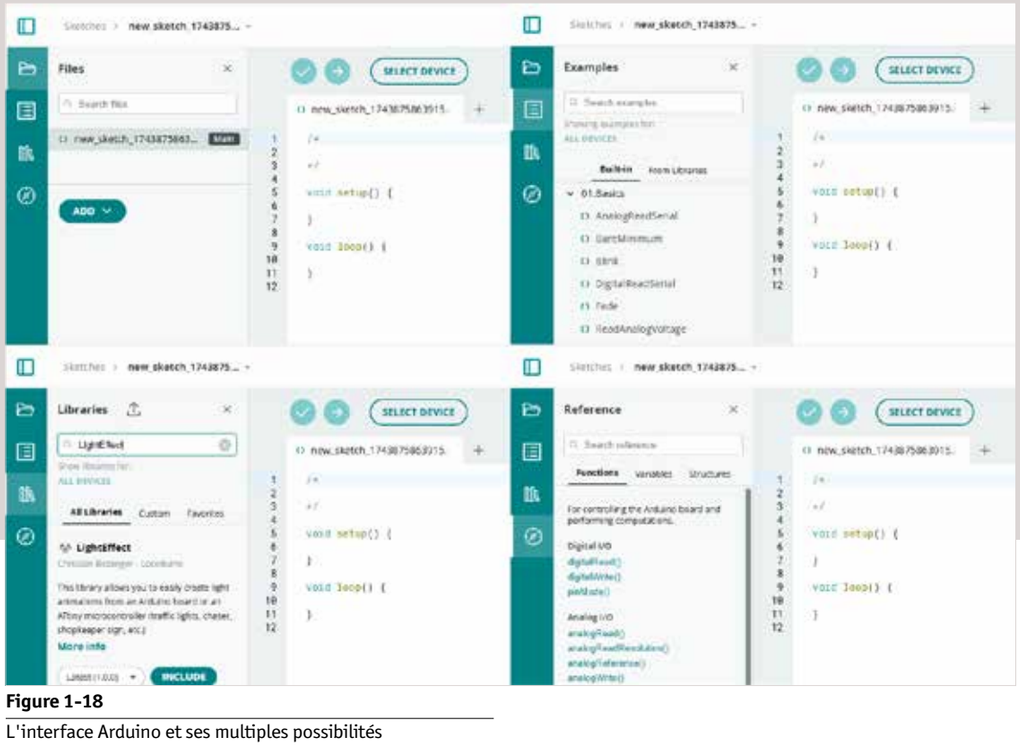

Figure 1-18

L'interface Arduino et ses multiples possibilités

1.4 / L'IOT OU L'INTERNET DES OBJETS

IoT signifie Internet Of Things, ce qui peut se traduire par internet des objets. On entre dans le monde des **objets connectés** qui sont de plus en plus nombreux dans notre quotidien : alarme de votre maison, motorisation de stores, programmation de vos radiateurs, etc. Même le train miniature est concerné puisque de plus en plus de centrale DCC peuvent être commandées à partir d'une tablette ou d'un smartphone. Je vous propose même, dans le chapitre 9, de fabriquer un objet connecté : une commande pour locomotive analogique. Mais avant d'en arriver là, il convient de décrire comment fonctionne l'IoT d'Arduino.

Élaborer un projet en IoT demande de **réfléchir aux variables** que votre projet contrôlera et ce sont ces variables qui contrôleront capteurs et actionneurs, qu'on peut appeler périphériques. Pour modifier ces variables ou pour afficher leurs valeurs, il faut définir un **tableau de bord (dashboard)** qui va communiquer dans les deux sens avec la carte Arduino via le Cloud : modifiez la valeur d'une variable et la carte Arduino le saura et modifiera son comportement. Bien évidemment, la carte Arduino doit être connectable en WiFi et **être compatible avec l'IoT d'Arduino**.

C'est une nouvelle façon de concevoir un projet car à part définir les variables de votre projet, vous n'avez quasiment rien d'autre à faire. En effet, la plateforme Arduino **génère automatiquement un sketch (programme)** chaque fois que vous créez une variable (ou elle modifie le sketch en cours). Tout est déjà prévu pour vous connecter au Cloud et la seule chose à faire est de rajouter quelques lignes de programme dans ce sketch, pour que la carte Arduino réagisse aux changements de variables.

La création du tableau de bord n'est pas très compliquée non plus puisque vous disposez de **widgets** que vous pouvez arranger comme bon vous semble sur l'écran du smartphone ou de la tablette. Chaque widget est rattaché à une variable et peut la contrôler : par exemple, si le widget représente un potentiomètre, il peut contrôler une variable dont les valeurs sont définies entre deux extrêmes (0 à 1023 par exemple) exactement comme si un véritable potentiomètre était relié à la carte Arduino afin de convertir une valeur de tension en valeur de ladite variable.

Pour travailler avec l'IoT d'Arduino, il faut se créer un compte, généralement payant.

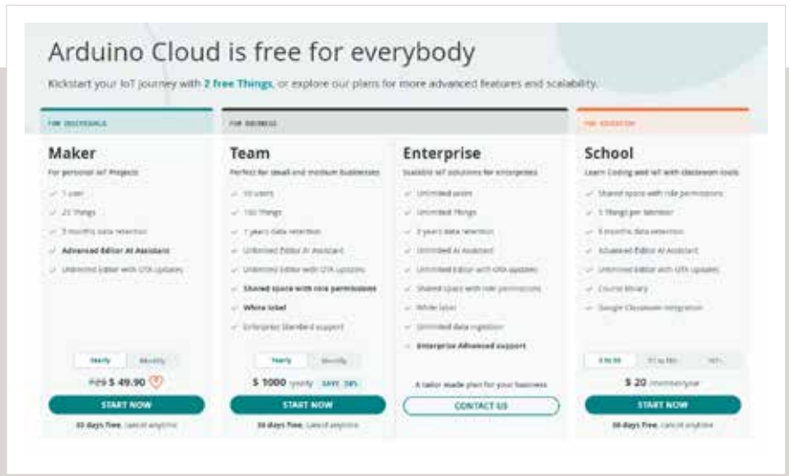

Figure 1-19

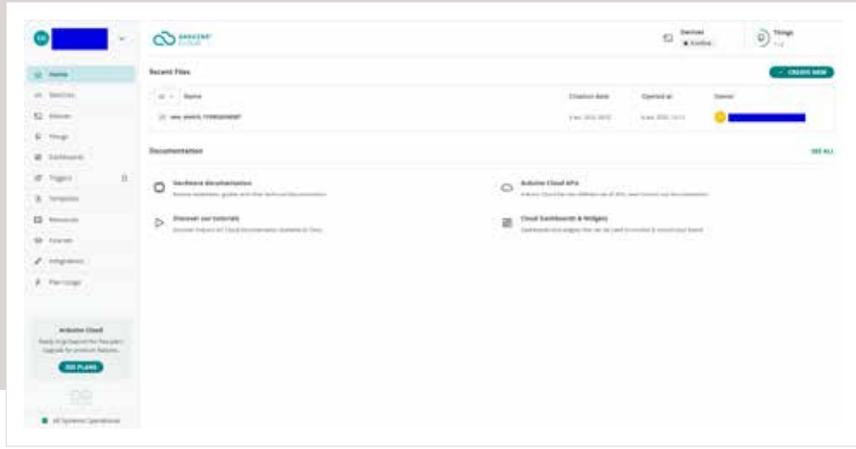

Figure 1-20

Cependant, un compte gratuit est proposé mais il est limité à deux projets en même temps. C'est une excellente formule pour découvrir l'IoT mais vous aurez vite envie de passer à la vitesse supérieure en choisissant une des options payantes proposées. Toutes les cartes ne sont pas compatibles avec l'IoT mais les cartes modernes proposées par Arduino le sont dès qu'elles sont équipées du WiFi. **L'IoT d'Arduino est aussi compatible avec les cartes ESP du constructeur Espressif**.

On accède au Cloud d'Arduino à partir de leur site ; il faut commencer par se créer un compte et choisir une formule d'abonnement. La **figure 1-19** montre différentes formules payantes possibles en avril 2025.

Vous avez la possibilité de payer au mois ou à l'année. En payant sur l'année, vous faites une économie de l'ordre de 15%. Le plan « **Maker** » offre le meilleur rapport qualité-prix et permet d'avoir 25 objets en même temps pour moins de 7 dollars par mois. On peut à tout instant migrer d'un plan vers un autre en fonction de ses besoins.

La **figure 1-20** montre la page Cloud-home : sur la gauche, on accède aux différents sketches, aux différentes cartes (devices), aux différents objets déjà créés (things), aux différents tableaux de bords associés aux objets (dashboards), etc. En haut à droite, le bouton « **Create new** » permet de créer un nouveau sketch ou objet ou tableau de bord ou importer un programme local (depuis l'ordinateur que vous utilisez) ou encore ajouter la dernière carte que vous avez achetée.

Voyons la marche à suivre pour créer notre premier objet connecté. Pour rester dans le domaine du modélisme ferroviaire, je vous propose de créer une motorisation d'aiguille avec un servomoteur. Utiliser un smartphone pour bouger une aiguille sur un réseau peut paraître un peu excessif, mais ce n'est qu'un exemple dont vous pourrez vous inspirer pour d'autres projets (éventuellement commander toutes vos aiguilles). La première chose à faire depuis la page home est **« Create new » > Thing**. On arrive sur une nouvelle page où notre nouvel objet est sans nom (untitled) mais on peut le renommer CMD_AIG pour « commande d'aiguille ». La page nous invite à créer des variables (bouton ADD). Sur la droite de la page, on peut associer cet objet à une carte (device) et à un réseau (network) internet (votre box). On peut aussi associer l'objet à une enceinte connectée comme Alexa d'Amazon ou bien Google Home ou encore faire communiquer cet objet avec des services externes.

Une variable dont on a besoin est l'état de l'aiguille qui n'a que deux possibilités : droite ou déviée. C'est à partir de cette variable qu'une fonction commandera le servomoteur. Cependant, pour commander le mouvement, on va utiliser, pour switcher la position de l'aiguille, un bouton poussoir sur notre tableau de bord qui n'a que deux possibilités : repos ou enfoncé. On va donc choisir une **variable booléenne**, false si repos et true si enfoncé. On donne un nom à la variable (etatSwitch), on choisit son type et on coche « **Read & Write** » et « **On change** » et on clique sur « **ADD VARIABLE** » (**figure 1-21**).

Un programme a été créé qu'on peut regarder avec le bouton </> Sketch en haut à droite. Le programme contient setup, loop ainsi qu'une fonction appelée '**onEtatSwitchChange()**' qu'il faut compléter par ce qui est surligné.

```
void onEtatSwitchChange()  {
  // Add your code here to act  upon
     EtatSwitch change
   if(etatSwitch == true) {
    etatAiguille = !etatAiguille;
   }
}
```

La variable ainsi créée (**etatSwitch**) est déclarée automatiquement dans le programme ; par contre, **la variable etatAiguille doit être déclarée** comme variable booléenne. On peut aussi penser qu'il serait intéressant d'avoir deux témoins de contrôle sur notre tableau de bord pour indiquer l'état de l'aiguille (l'équivalent de deux LED verte et rouge indiquant droite ou déviée). **On va donc ajouter deux autres variables** greenLED et redLED, booléenne également. Le programme a été modifié et deux autres fonctions sont prêtes à être complétées pour indiquer ce que doit faire le programme chaque fois que ces variables changent.

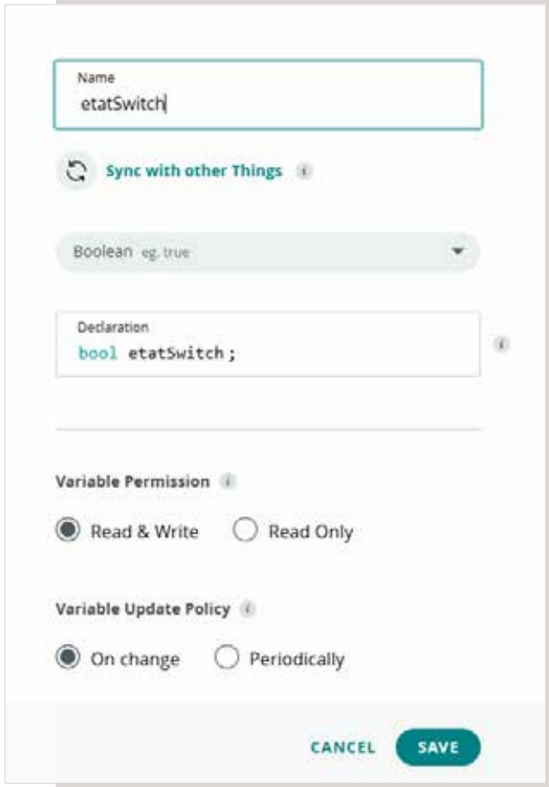

Figure 1-21

Figure 1-22

On sort de la page </> Sketch et on revient sur le setup de l'objet qui indique bien les trois variables qu'on vient de créer. On va dans « **Associated Device** » et on sélectionne une carte (ou on en ajoute une) : par exemple Nano-33-IoT-1 parmi celles proposées (**figure 1-22**).

Attention : lorsque vous ajoutez une nouvelle carte compatible IoT (Set up new device), il est attribué un **numéro d'identification (Device ID) et une clé secrète (Secret Key)**. Conservez précieusement ces deux informations qui pourront vous être demandées ultérieurement : vous avez la possibilité de les sauvegarder sous forme d'un fichier PDF à télécharger.

La carte est ajoutée mais apparait **off-line** (non branchée). Deux boutons permettent de détacher cette carte de l'objet ou de changer de carte pour cet objet. La **figure 1-23** montre comment rentrer les **paramètres de son réseau WiFi** : il suffit d'entrer le nom du réseau et le mot de passe et sauvegarder (SAVE).

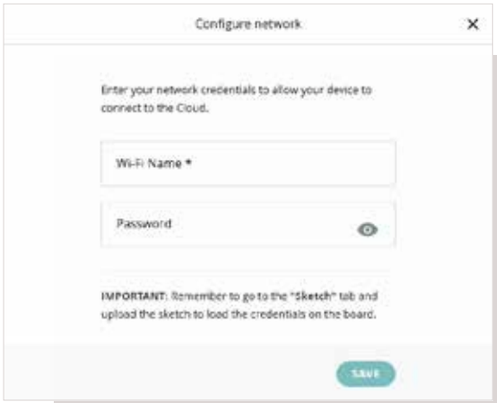

Figure 1-23

Pour créer le tableau de bord (**dashboard**), on va dans la catégorie Dashboard depuis la page Home : une nouvelle page s'ouvre et on clique sur +DASHBOARD pour en créer une nouvelle. On va sur Edit et on ajoute les widgets nécessaires : un poussoir pour commander l'aiguille et deux LED pour visualiser l'état. **Chaque widget doit être relié à une variable**, ce qui se fait de façon conviviale. Validez avec la touche DONE. On ajoute une LED, on choisit la couleur verte, on lui donne le nom « Droit », on l'associe avec la variable greenLED et on valide. Idem avec une LED rouge, nommé « Dévié » et associée à la variable redLED. Le tableau de bord est terminé ; on peut maintenant **réorganiser les widgets et les redimensionner** si on le souhaite. La **figure 1-24** montre à gauche le tableau de bord brut où on a choisi le format smartphone (au lieu de tablette) et à droite le tableau de bord réorganisé selon son goût.

Figure 1-24

On peut encore l'améliorer en remplaçant Push Button par Aiguillage, et en retirant le nom de l'objet sur les widgets LED (l'option « Show Thing name on widget » à décocher) : pour cela, il suffit de cliquer sur le widget et de choisir **l'engrenage (setup)** pour revenir à ses propriétés. Lorsque tout est bien configuré, on clique sur DONE. La **figure 1-25** montre comment la dashboard apparaîtra sur l'écran du smartphone.

Il reste maintenant à compléter le programme (voir plus loin) et à le téléverser dans la carte Nano-33-IoT. Le servomoteur doit se déplacer entre une position qui correspond à l'aiguille droite (par exemple 70°) et une position qui correspond à l'aiguille déviée (par exemple 110°). Dans cet exemple, le servomoteur se déplace de 20° de part et d'autre de sa position centrale qui vaut 90°, mais les valeurs exactes dépendent de vos aiguilles et du montage de la timonerie de déplacement. À l'initialisation, l'aiguille se met en position non déviée, la LED verte doit être allumée sur l'écran du smartphone et la LED rouge éteinte. En appuyant sur le poussoir virtuel, on fait basculer l'état de l'aiguillage : la carte Nano positionne le servomoteur et modifie l'état des LED pour n'en allumer qu'une seule.

Figure 1-25

Les quelques lignes ajoutées sont surlignées dans le texte de programme ci-dessous :

```
#include "thingProperties.h"
#include <Servo.h>
Servo AIG;

bool etatAiguille = true;
bool old_etatAig = etatAiguille;
int positionDroite = 70;
int positionDeviee = 110;
```

À la fin du setup, ajoutez :

```
AIG.attach(10);
AIG.write(positionDroite);
greenLED = true;
redLED = false;
```

Dans la loop, ajoutez :

```
void loop() {
  ArduinoCloud.update();
  // Your code here
  moveServo();
}
```

La fonction qui bouge le servomoteur est la suivante :

```
void moveServo() {
 if(etatAiguille != old_etatAig) {
  if(etatAiguille == true) {
   for(int i = 70; i <= 110; i++) {
    AIG.write(i);
    delay(50);
   }
   greenLED = false;
   redLED = true;
  }
  if(etatAiguille == false) {
   for(int j = 110; j >= 70; j--) {
    AIG.write(j);
    delay(50);
   }
   redLED = false;
   greenLED = true;
  }
  old_etatAig = etatAiguille;
 }
}
```

Les fonctions onGreenLEDChange et onRedLEDChange ne sont pas à compléter.
La fonction onEtatSwitchChange est à compléter comme indiqué plus haut.

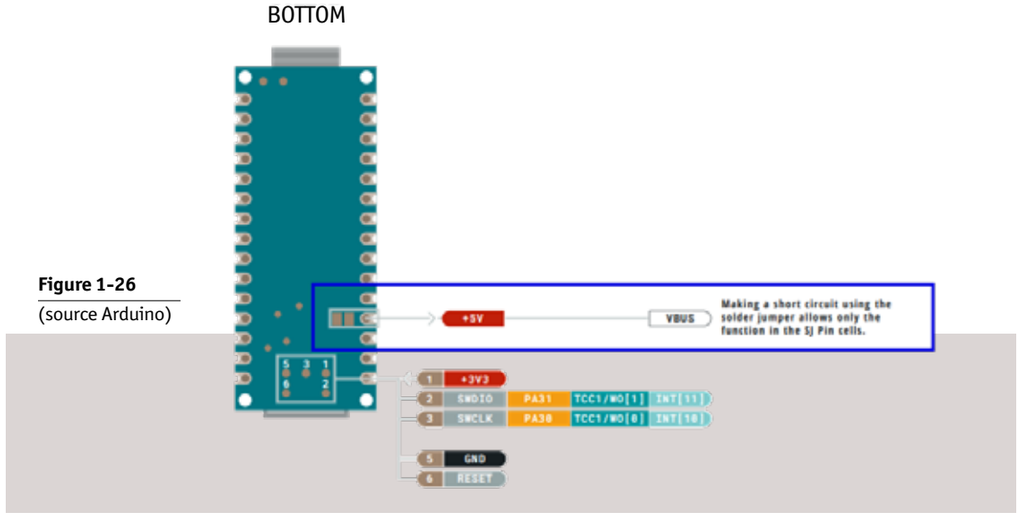

Figure 1-26
(source Arduino)

Il reste maintenant à relier le servomoteur à la carte Nano-33-IoT : la sortie 10 est directement reliée au fil de signal du servomoteur. Une broche de la carte peut délivrer du 5 V à la **condition d'avoir fait un pont de soudure** pour relier deux plots au verso de la carte et d'**alimenter la carte par l'USB** (ceci est expliqué dans la documentation). Le mieux est sans doute de prévoir une alimentation en 5 V à part pour le (ou les) servomoteur(s) : dans ce cas, la masse de l'alimentation doit être reliée à la masse (GND) de la carte.

N'oubliez pas que quand vous ajoutez des variables, **celles-ci sont déclarées dans le programme automatiquement généré** mais cela n'apparaît pas comme dans un programme classique : ne les déclarez pas une seconde fois, sinon cela génèrera une erreur à la compilation. Il faut ensuite téléverser le programme dans la carte et ceci ne peut se faire que par le Cloud (donc sans utiliser l'IDE). Le tableau de bord peut être récupéré sur votre smartphone ou tablette en **installant l'application Arduino Remote**.

L'IoT d'Arduino est plus simple : le programme est généré automatiquement et il n'y a plus qu'à le compléter et vous n'avez pas à calculer la valeur des résistances de limitation des LED.

L'IoT d'Arduino est plus économique : ici, on a économisé un poussoir, deux résistances et deux LED. L'IoT d'Arduino est plus souple : vous accédez à votre montage où que vous soyez et vous pouvez réorganiser vos widgets avec votre ordinateur ou directement depuis le dashboard.

Pour résumer, il faut :

1. **Créer un objet**
2. **Ajouter les variables**
3. **Affecter une carte compatible IoT à l'objet**
4. **Renseigner le réseau WiFi (attention au nom et mot de passe qui doivent être entrés sans faute)**
5. **Compléter le programme automatiquement généré et le vérifier puis le téléverser**
6. **Créer un tableau de bord et le récupérer via l'application Arduino Remote**

Créer votre premier objet vous paraîtra sans doute un peu déroutant car c'est une nouvelle façon de travailler où on raisonne en terme de variables : la plateforme s'occupe du reste, mais c'est tout de même **à vous de modifier le programme** pour ajouter ce qu'il doit faire en fonction des valeurs de ces variables. Vous finirez par adorer cette façon de travailler, même si on n'a pas besoin d'une multitude d'objets connectés sur un réseau.

1.5 / RENDRE SES FICHIERS ACCESSIBLES ET TRAVAILLER EN ÉQUIPE AVEC GITHUB

Le **site internet github.com** est un site dépositaire (repository en anglais) qui permet à chacun de mettre son travail à la disposition de tous. Si vous voulez écrire une bibliothèque et qu'elle soit accessible dans l'écosystème d'Arduino, il faudra la déposer sur **GitHub** (la méthode pour écrire une bibliothèque est décrite dans la doc Arduino). Le site GitHub permet aussi de gérer les différentes versions d'un projet, et surtout il facilite le travail en équipe. **Tous les développeurs dignes de ce nom ont un compte sur GitHub et ont appris à travailler selon les spécifications de cette plateforme collaborative**, et je ne peux que vous conseiller d'ouvrir votre compte. Même sans cela, la consultation de GitHub permet d'accéder à différentes informations sur les bibliothèques que vous utilisez dans vos projets avec Arduino, mais comme ce site est international, c'est l'anglais qui est le plus souvent utilisé. La **figure 1-27** montre comment accéder à des informations

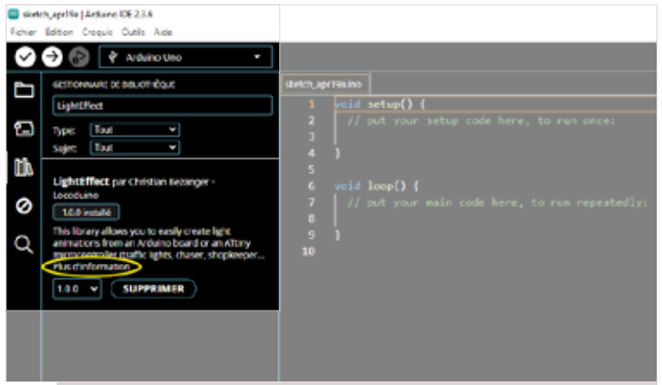

Figure 1-27

concernant la bibliothèque LightEffect à partir de l'IDE 2. En cliquant, on ouvre la page GitHub concernant la bibliothèque (**figure 1-28**) et on accède aux informations expliquant comment l'utiliser. En bas de page (non visible sur la figure), on accède aux exemples de la bibliothèque. Tout cela est évidemment en anglais pour servir au plus grand nombre. Le pavé vert (**figure 1-28**) sert à télécharger la bibliothèque. Beaucoup de liens du site d'Arduino renvoient vers des pages de GitHub.

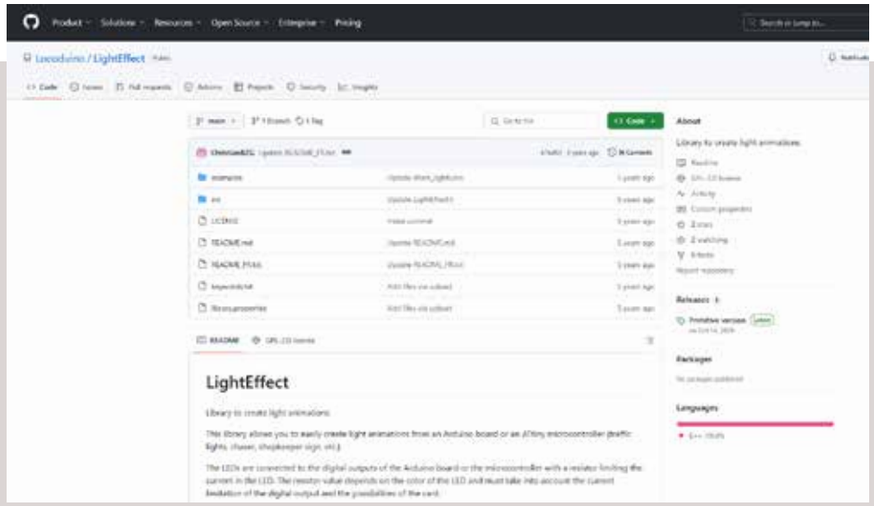

Figure 1-28

1.6 / UTILISER AU MIEUX LES SIMULATEURS D'ARDUINO

Les **simulateurs d'Arduino**, Tinkercad ou Wokwi, sont des outils précieux pour mettre au point un projet ; ils ont été décrits dans le tome 1. Il faut tout de même avoir à l'esprit **qu'ils ont également leurs limitations** : le composant électronique peut ne pas exister ou bien l'association de composants entre eux peut ne pas fonctionner comme dans la réalité. Selon le simulateur choisi, certains montages du tome 1 n'étaient pas possibles ou ne fonctionnaient pas.

Il est tout de même intéressant de faire appel à eux chaque fois que c'est possible et certains projets ont été mis au point sur simulateur avant de passer aux composants réels. Une modification du programme peut se tester en temps réel sans avoir besoin de passer par la phase de téléversement sur la carte. On gagne encore plus de temps lorsqu'il s'agit de modifier le montage lui-même. **Référez-vous au tome 1 pour découvrir les simulateurs Tinkercad et Wokwi**, mais comme ceux-ci évoluent, il se peut que vous découvriez quelques fonctionnalités supplémentaires par rapport à ce que j'avais décrit. Par exemple, le simulateur Wokwi a évolué et offre de nouvelles possibilités. Vingt langues sont proposées et l'allemand a été ajouté pour consulter la documentation du simulateur, en plus de l'anglais, du portugais et du chinois. Enfin, un menu graphique s'ouvre lorsqu'on clique sur une résistance, ce qui permet de modifier aisément sa valeur et son orientation, de la supprimer ou d'accéder à la documentation en cliquant sur le point d'interrogation (voir **figure 1-29**).

Vous gagnerez beaucoup de temps en utilisant les simulateurs (le choix étant fait en fonction des composants proposés) mais **vous n'échapperez pas à la phase de test en réel avec de vrais composants, une fois votre projet abouti**. Les simulateurs sont également une aide précieuse

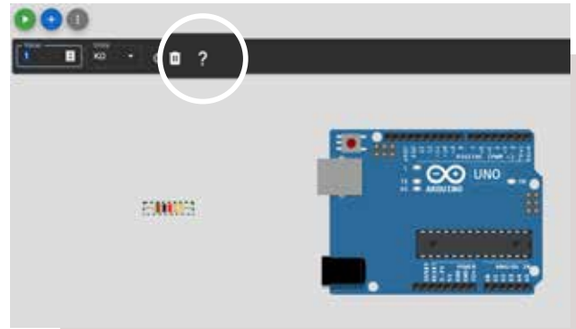

Figure 1-29

si vous voulez partager un montage ou un projet avec un ami, soit pour qu'il en bénéficie, soit pour travailler en équipe. Avec Wokwi, vous pouvez **générer un lien qu'il suffit de recopier** (menu SHARE) **ou bien sauvegarder votre projet sous forme de fichier ZIP** (flèche bleue du menu SAVE). Le ZIP contient le programme (sketch.ino) et le fichier des composants (diagram.json) ; avec un copier-coller du contenu de ces deux fichiers dans les onglets adéquats, vous reconstituez votre projet et il ne vous reste plus qu'à importer les bibliothèques (si le projet en utilise).

Victime de son succès, Wokwi oblige maintenant à passer par une file d'attente pour compiler un programme ; au bout d'un certain temps, la demande peut être rejetée pour surcharge. Il suffit de recommencer et au bout de quelques fois, on finit par y arriver. La jauge de charge affichée ne veut pas dire grand-chose car un programme peut être rejeté alors qu'elle est dans le vert ou bien compilé alors qu'elle est dans le rouge. Si cela vous agace, vous pouvez compiler plus rapidement en **adhérant à un plan payant**, compris entre 7 et 25 dollars par mois, avec possibilité d'une ristourne si vous payez à l'année.

1.7 / CONCEVOIR UN CIRCUIT IMPRIMÉ DE QUALITÉ PROFESSIONNELLE

Vous avez la possibilité de faire **fabriquer un circuit imprimé de qualité professionnelle** pour un prix modique, surtout si vous vous contentez d'un circuit double face, ce qui est en général suffisant pour les projets de modélisme ferroviaire (**figure 1-30**).

Ce circuit imprimé est aussi appelé **PCB** (pour **Printed Circuit Board**) dans le jargon des électroniciens. Le routage est l'opération qui consiste à dessiner les pistes cuivrées qui relient les différents composants électroniques. Au risque de vous décevoir, il n'y a pas de routage automatique et aucun logiciel n'est capable de créer seul le dessin des pistes à partir d'un schéma électronique.

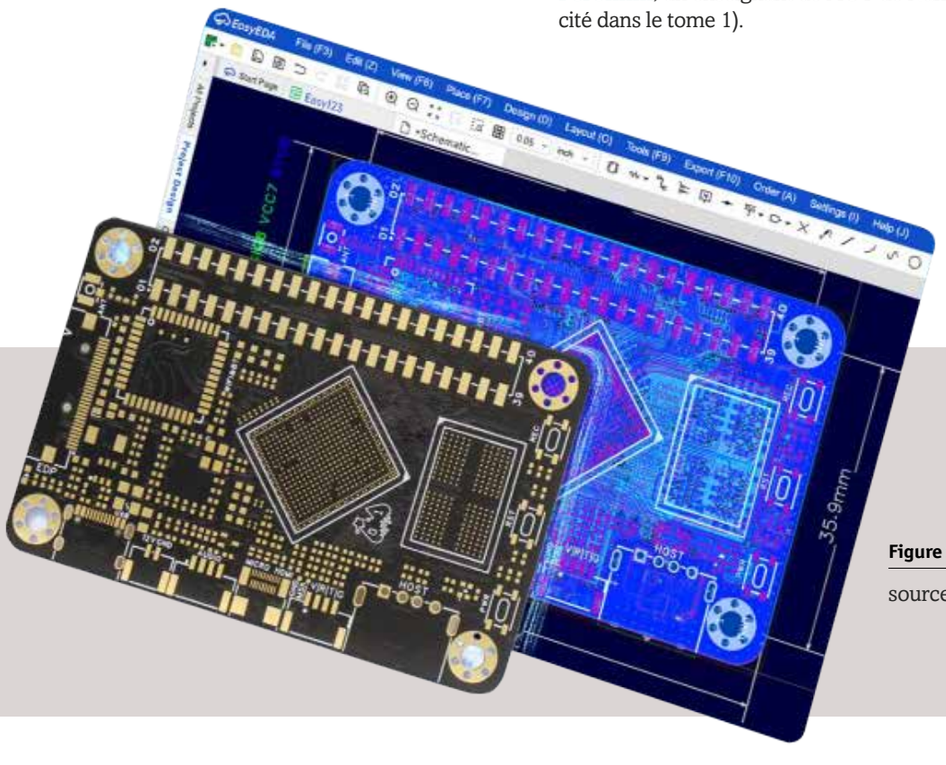

Figure 1-30

source EasyEDA

Cependant, plusieurs logiciels vous aident à passer du schéma électronique au dessin des pistes du circuit imprimé, mais il ne s'agit que d'une aide et l'essentiel du travail vient de vous :

- **KiCad** (gratuit et open-source)
- **EAGLE** (gratuit pour usage personnel)
- **Altium Designer** (professionnel)
- **EasyEDA** (en ligne, facile à prendre en main)

Pour ma part, j'ai choisi KiCad qui a été créé par J.P Charras de l'IUT de Grenoble et repris en 2013 par une unité du CERN, d'une part parce qu'il est gratuit, et d'autre part parce que de nombreux tutos de prise en main se trouvent sur YouTube. **Regardez plus spécialement les tutos d'Eric Peronnin**, un enseignant de l'IUT de Nantes (déjà cité dans le tome 1).

À l'heure où ces lignes sont écrites, KiCad en est à la version 9.0.1 et constitue un véritable environnement de développement intégré. Ce logiciel est vraiment complet et pour l'utiliser au mieux, il faut un certain temps à s'entraîner. Pour créer un PCB, la marche à suivre est la suivante. Après avoir ouvert un nouveau projet, la **première étape** est de saisir le schéma électronique grâce à une bibliothèque de symboles et à relier les différents composants. Si pour un composant le symbole n'existe pas, il y a la possibilité de le récupérer sur internet et de l'ajouter à la bibliothèque, ou bien encore le créer à partir de la notice (datasheet) du fabricant. Le logiciel numérote les composants et il vaut mieux le laisser faire pour ne pas avoir deux fois un composant avec le même numéro, mais **c'est à vous de préciser la valeur** (par exemple 330 pour une résistance de 330 ohms). Pour ne pas surcharger le schéma, on utilise des symboles d'équipotentiel, par exemple pour la tension d'alimentation Vcc ou la masse GND. À la fin de ce travail, il faut **vérifier les connexions** (fonction ERC pour Electrical Rule Check) pour contrôler qu'il n'y a pas de court-circuit dans la conception du schéma.

Les composants électroniques se trouvent avec plusieurs types de boitiers : il y a des composants traversant (**DIP Dual Inline Package**) qui ont des broches qui traversent le circuit imprimé et se soudent au verso, et les composants **CMS (composant monté en surface, SMD en anglais pour Surface Mounted Device)** dont les broches sont soudées au recto. Pour un PCB, le recto est la couche de devant (F pour front) et le verso est la couche arrière (B pour back). La **deuxième étape** est d'associer à chaque composant du schéma une **empreinte physique** qui sera présente sur le PCB et permettra de souder le composant. Bien évidemment, l'empreinte doit correspondre au composant que vous achetez ; il y a une bibliothèque d'empreintes dans KiCad ainsi qu'un éditeur d'empreintes si rien ne se trouve en bibliothèque.

La **troisième étape** est le **routage**, c'est-à-dire le dessin des pistes du circuit imprimé. On commence par importer le schéma et toutes les empreintes se retrouvent sur le PCB, reliées les unes aux autres par des traits fins qu'on appelle le « chevelu » (ou **nets**) et qui correspondent aux pistes à tracer. Il faut placer les composants sur le PCB en respectant la logique du schéma électronique qui impose parfois de **regrouper certains composants**, et en essayant d'avoir les futures pistes **les plus courtes possibles**. C'est cette étape qui demande un peu d'expérience et aussi quelques essais pour placer le composant en le tournant, le déplaçant légèrement, l'alignant avec d'autres pour l'esthétique, etc. Le chevelu est remplacé par les pistes pour les couches de cuivre utilisables (face avant et face arrière si PCB à deux couches). On peut régler la largeur des pistes, la taille des **vias** (trous métallisés qui permettent de relier électriquement la face avant avec la face arrière) ou l'arrondi des pistes. On peut aussi créer des **zones de même potentiel** (plan de masse par exemple). Parfois, il est nécessaire de bouger un composant pour permettre à une piste de trouver son chemin. Tout ce travail demande une certaine dextérité qui ne peut s'acquérir que par la pratique, en commençant par des circuits simples. Pour terminer le routage, il faut contrôler, grâce à KiCad, les erreurs de conception (**DRC Design Rule Check**), ce qui évite de faire construire des PCB qui finiraient à la poubelle.

Il est temps alors de passer à la phase de fabrication et pour cela, il faut **générer les fichiers Gerber** (format standard utilisé par les fabricants pour produire les PCB). Ces fichiers concernent la **sérigraphie** (texte en blanc pour repérer les composants, inclure un logo, etc.) qui est généralement faite par imprimante à jet d'encre, les **couches de cuivre** (face avant et arrière), les **couches de vernis** qui vont protéger les couches de cuivre (mais le vernis ne doit pas se retrouver sur les pastilles où on soude), les **indications de perçage** (composants traversant ou vias ou trous

de fixation du PCB), et enfin le **contour du PCB** (rectangulaire ou polygonal ou courbe). Si on utilise des CMS, on peut demander aussi la fabrication d'un masque de brasure, feuille métallique très fine et trouée, permettant de déposer avec une raclette de la pâte à braser sur les plots de soudage des composants. Les fichiers Gerber générés sont mis dans un **dossier compressé qu'il suffit alors d'envoyer à un fabricant de PCB**.

Toute cette marche à suivre représente, pour un débutant, beaucoup de travail mais avec l'habitude, créer un PCB devient aussi facile que rédiger une lettre avec un traitement de texte. Je n'insisterai pas sur les subtilités d'utilisation de KiCad que vous pourrez apprendre grâce aux tutos d'internet : elles sont nombreuses car KiCad est un logiciel professionnel. D'autres logiciels peuvent mieux vous convenir mais la méthode pour créer un PCB reste la même.

Les fichiers Gerber sont à envoyer à un fabricant via internet (**JLCPCB** (Chine, très bon marché), **PCBway**, **Eurocircuits** (Europe), **OSH Park** (USA)). Tous les fichiers utiles à la fabrication doivent être présents dans le fichier ZIP : F-Cu et B-Cu pour le cuivre, F-Silk pour la sérigraphie, edge-cut pour la découpe des bords, masque si CMS, fichier de perçage au format Excellon, etc. On choisit l'épaisseur de la carte, le nombre de couches (1, 2, 4, etc.), la finition et la couleur du PCB ; les options sont pré-choisies mais on peut les modifier. Par exemple, JLCPCB propose comme choix standard, 5 circuits de 100 x 100 mm, double face et épaisseur 1,6 mm, sept couleurs au choix, pour deux dollars ! En comparaison, les frais de port vous paraîtront élevés et il faut être vigilant sur ce point car des frais de dossier ou de douane peuvent s'ajouter ; préférez alors l'option DDP (Prepay customs duties and taxes) où JLCPCB les paie et les facture au moment de la commande, sans mauvaise surprise lors de la réception du colis. Si vous êtes patient sur le délai de livraison, vous pouvez faire baisser les frais de port (Europacket : 8 à 10 jours, Global standard direct line : 11 à 14 jours).

D'autres options s'ajoutent à cette offre, comme **l'implantation de composants électroniques SMD (Surface Mounted Device) déjà soudés** sur le circuit imprimé, à conditions qu'ils soient au catalogue du fabricant. Ces options supplémentaires sont évidemment payantes : par exemple, si vous voulez une épaisseur de 2 mm (au lieu de 1,6), le prix passe à 36 dollars et une journée supplémentaire de fabrication est nécessaire ! Néanmoins, l'option finition RoHS (Restriction of Hazardous Substances) coûte moins d'un euro et peut contribuer à préserver la planète.

Le site de JLCPCB propose également l'accès au logiciel en ligne **EasyEDA** qui permet aussi de visualiser le circuit en 3D, tout comme KiCad qui propose cette option également (**figure 1-31**).

Afin de prendre en main un logiciel de conception de PCB, je vous invite à commencer par un circuit relativement simple comme les **montages proposés au chapitre 7**, par exemple celui pour régler vos servomoteurs d'aiguilles (un Arduino Nano, deux poussoirs, un écran OLED et une sortie à trois broches pour le servomoteur). Une fois le projet choisi, il faut prendre son courage à deux mains et se lancer.

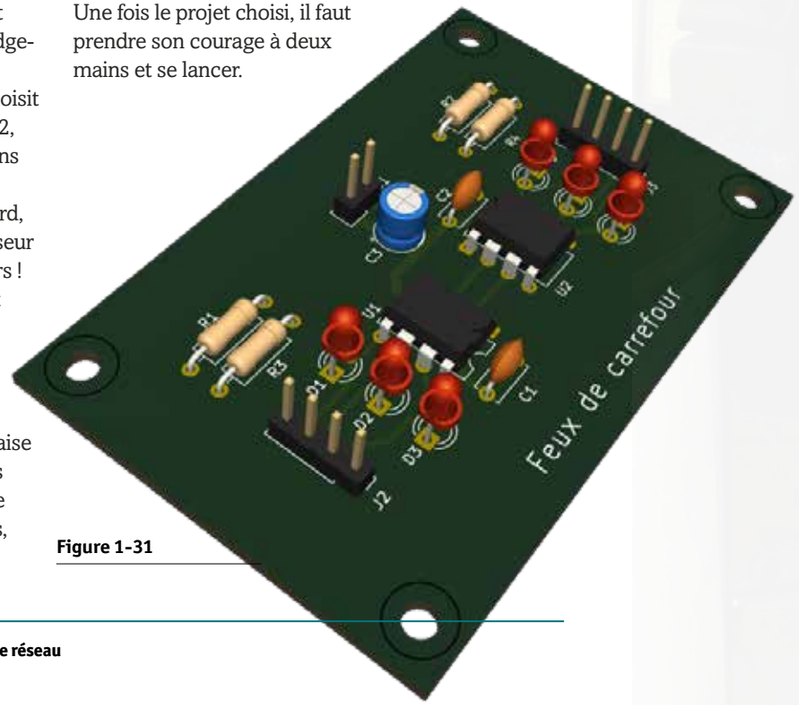

Figure 1-31

RÉSUMÉ DU CHAPÎTRE 1

Dans ce chapitre, nous avons découvert de nouveaux outils pour travailler plus efficacement sur nos projets d'automatisation : nouveau site d'Arduino, nouvel environnement de développement (IDE 2), Cloud et Internet des objets, site GitHub pour travailler en équipe, simulateurs d'Arduino, conception de circuit imprimé. Plus vous manipulerez ces nouveaux outils et plus vous serez à l'aise et gagnerez en efficacité. N'hésitez donc surtout pas à les utiliser, même si l'apprentissage peut vous paraître un peu difficile au début ; cela vient vite !