LES GUIDES PRATIQUES DU TRAIN MINIATURE

AUTOMATISEZ VOTRE RÉSEAU

Concevoir un projet avec Arduino
et l'Intelligence Artificielle

09

LES GUIDES PRATIQUES DU TRAIN MINIATURE

AUTOMATISEZ VOTRE RÉSEAU

Concevoir un projet avec Arduino et l'Intelligence Artificielle

Éditions LR Presse

crédits

Texte et Illustrations :
Christian Bézanger sauf
page 56 : Yann Baude
page 69, figure 56 et
page 70, figure 58 : MTB

- longueur max.
- SURL?

Introduction

Le guide pratique « **Animez votre réseau : Initiation au système Arduino** » a permis à de nombreux modélistes de découvrir les microcontrôleurs et ce qu'on peut en faire pour animer un réseau de trains miniatures. Le lecteur pouvait voir fonctionner de nombreux petits montages sans même avoir à les construire, grâce à des liens vers des simulateurs d'Arduino comme **Tinkercad** ou **Wokwi**, une première dans le domaine du modélisme ferroviaire. Le succès a été immédiat et mon but était atteint : donner envie aux modélistes de se mettre à l'électronique programmable. Aussi, quand Christian Fournereau m'a proposé de faire un tome 2 du guide, plutôt qu'une simple mise à jour, j'ai été immédiatement séduit par l'idée. Cela représentait plus de travail, mais le jeu en valait la chandelle car les choses évoluent très vite dans le domaine de l'électronique et de l'informatique.

Si vous avez envie d'aller plus loin dans l'utilisation des microcontrôleurs, le guide que vous avez en main, que j'appellerai « tome 2 », est fait pour vous. Cependant, il n'est pas possible au sein d'un guide qui doit rester accessible à tous, de décrire de façon complète et détaillée toutes ces nouvelles techniques : nouveau logiciel plus convivial, nouvelles cartes plus performantes, nouveaux langages, programmation en ligne, Cloud, objets connectés, Intelligence Artificielle, bref tout ce que vous devez savoir pour être capable d'aller encore plus loin. Mon but est de vous ouvrir des portes mais ce sera à vous d'en franchir le seuil pour découvrir ce qui se cache derrière et qui peut vous aider à progresser. C'est pourquoi je me suis forcément limité à l'essentiel dans mes explications, mais je reste persuadé que vous réussirez parfaitement à acquérir les connaissances qui vous seront nécessaires pour construire vos projets.

Mon but est de vous donner une méthode plutôt que décrire l'ensemble des connaissances, ce qui aurait sans doute ennuyé certains lecteurs. Et plutôt que remplir ce guide d'informations que vous pourriez trouver sur internet, j'ai préféré me recentrer sur le train miniature en proposant des exemples chaque fois que c'était possible. Il vous est ainsi possible de franchir les portes qui vous intéressent et de laisser pour plus tard d'autres portes qui vous sont moins utiles pour l'instant. Bien évidemment, je vous ai donné des pistes à approfondir, des sites internet à découvrir, des livres ou des articles à lire.

Les quatre premiers chapitres donnent des généralités sur les nouveaux outils et les nouvelles cartes à votre disposition, y compris celles développées par des concurrents d'Arduino. Dans cette section, j'aborde l'utilisation de l'Intelligence Artificielle qui peut parfois aider les débutants et leur servir de professeur pour progresser. Et comme il arrive toujours un moment où on se retrouve bloqué dans la conception d'un projet, j'ai donné quelques conseils pour trouver de l'aide, car on n'est jamais mieux servi que par soi-même.

Les chapitres 5 à 10 sont consacrés au train miniature ; ils décrivent ce qu'on doit installer sur un réseau (capteurs et actionneurs), comment architecturer l'ensemble de l'électronique, et surtout comment concevoir et réaliser un projet complet, et parfois complexe, de modélisme ferroviaire. Après avoir décrit quelques montages simples qui compléteront ceux du tome 1, j'ai décrit trois projets complets dont un objet connecté. Ces projets avaient été plus ou moins esquissés dans le tome 1, mais non développés par manque de place.

Enfin, le chapitre 11 vous donnera quelques pistes pour aller encore plus loin, en vous faisant découvrir ce qu'il y a sous le capot d'un microcontrôleur et en vous proposant d'autres langages pour programmer vos cartes. Je me suis aussi fait plaisir en décrivant ce que pourrait être le train miniature du futur, ma vision des choses, une solution qui vous paraîtra peut-être étrange mais qui résoudrait quelques problèmes que nous rencontrons tous aujourd'hui dans l'exploitation de notre réseau. Mais qui peut vraiment connaître l'avenir ? Il va nous réserver, une fois de plus, de bien belles surprises.

Sur cette pensée optimiste, je vous souhaite beaucoup de plaisir à lire ce tome 2.

Sommaire

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…

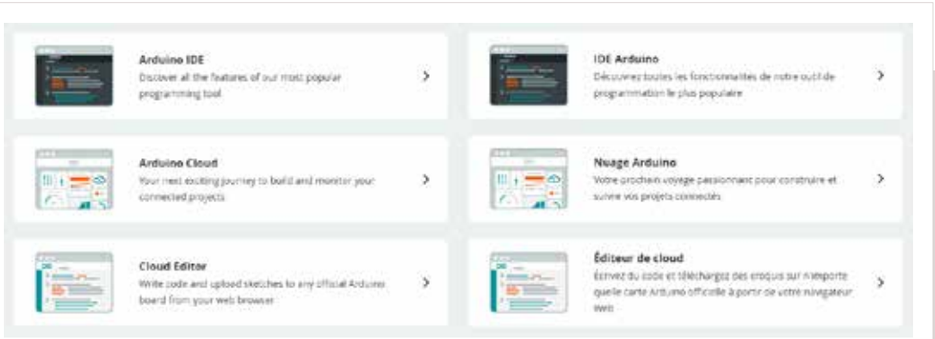


Figure 1-1
Comment programmer les cartes, version anglaise ou française traduite par le navigateur.

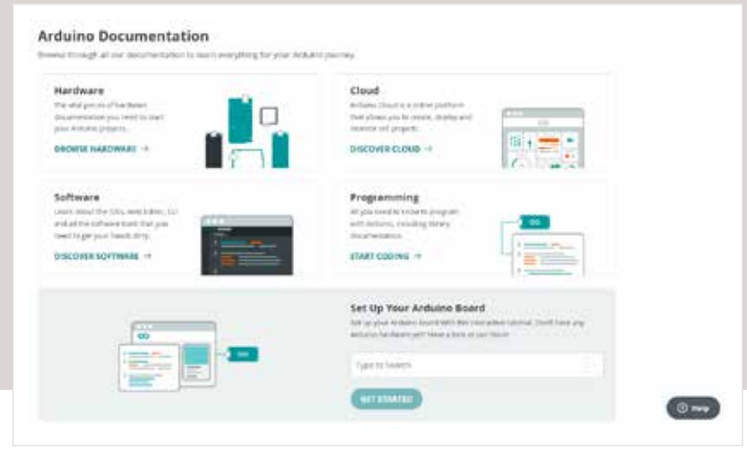


Figure 1-2
Accès à la documentation (source Arduino)

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**).

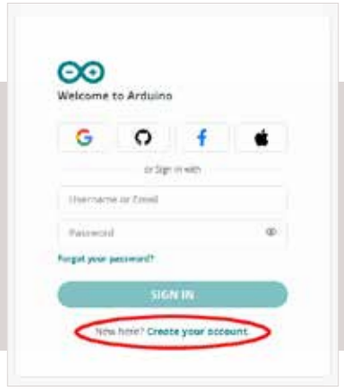


Figure 1-3
(source Arduino)

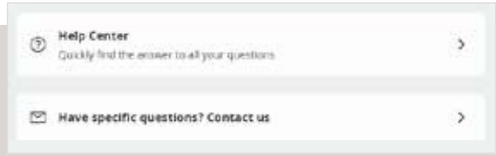


Figure 1-4
Obtenir de l'aide (source Arduino)

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**).

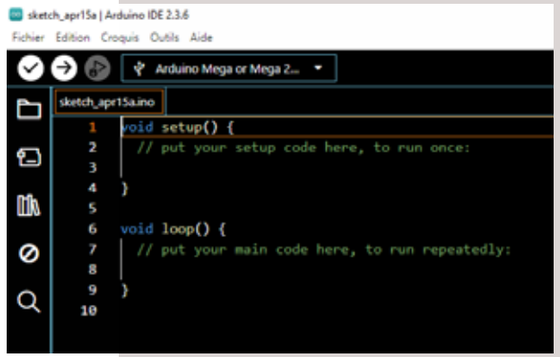


Figure 1-5

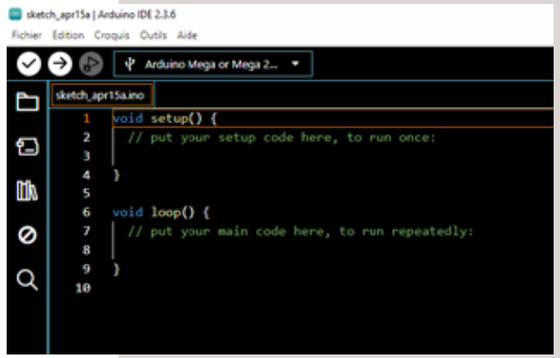


Figure 1-6

Figure 1-7
Interface du traceur série

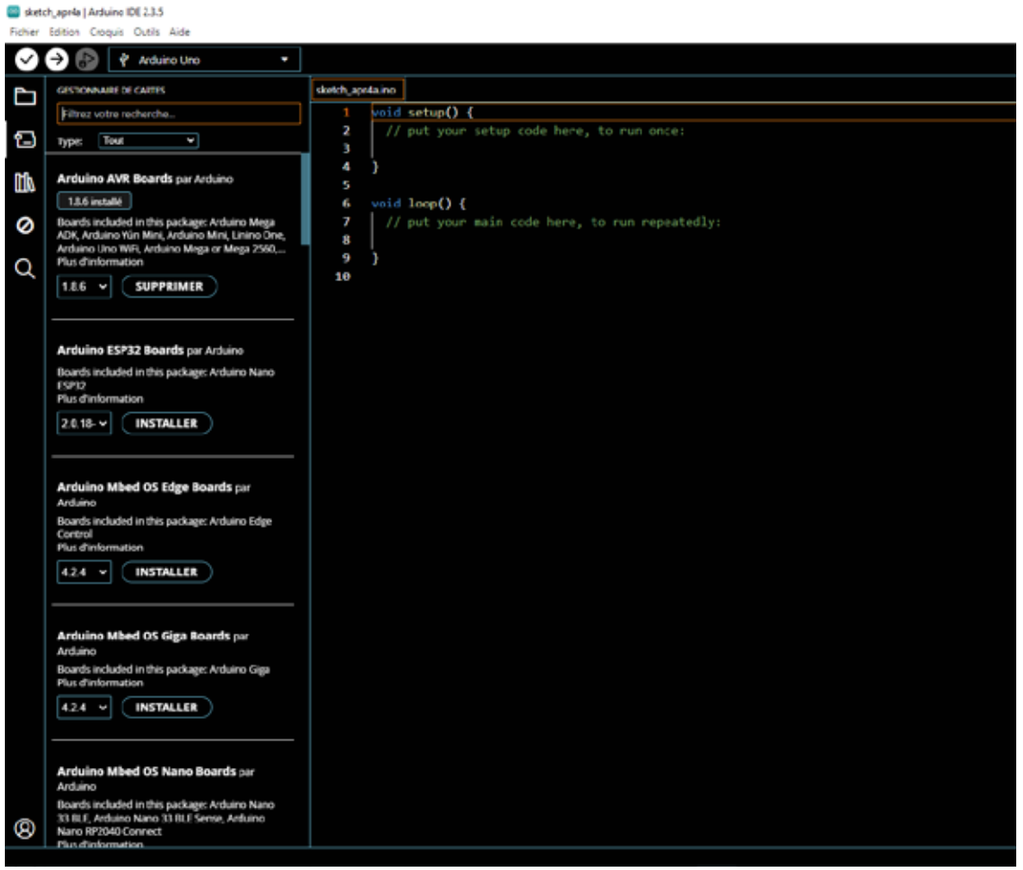


Figure 1-8

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**.

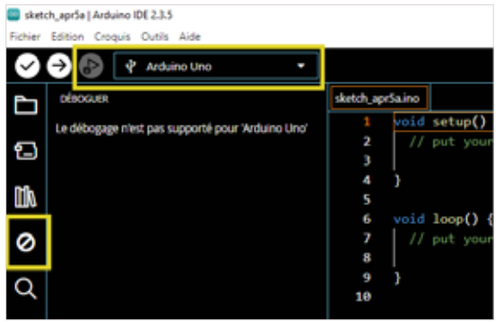


Figure 1-10a
Carte ne permettant pas le débogage

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

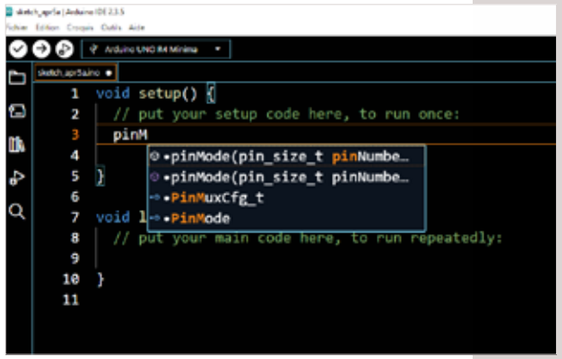


Figure 1-11

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.

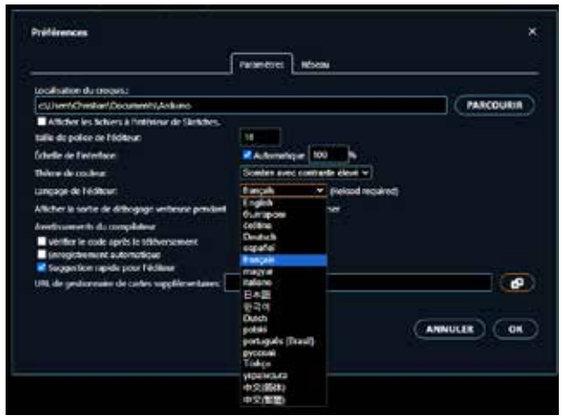


Figure 1-14

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**).

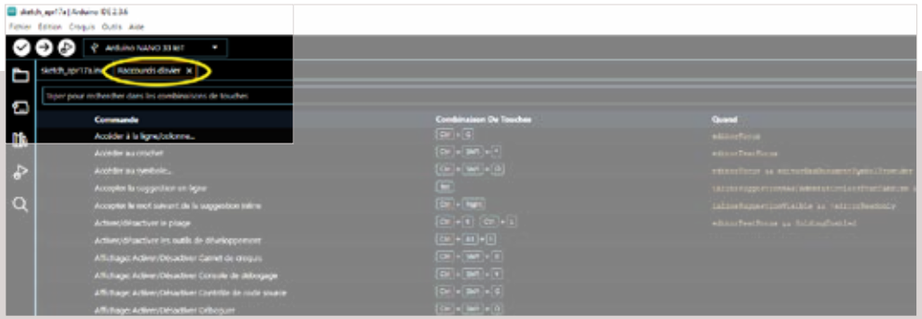


Figure 1-15


```
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)

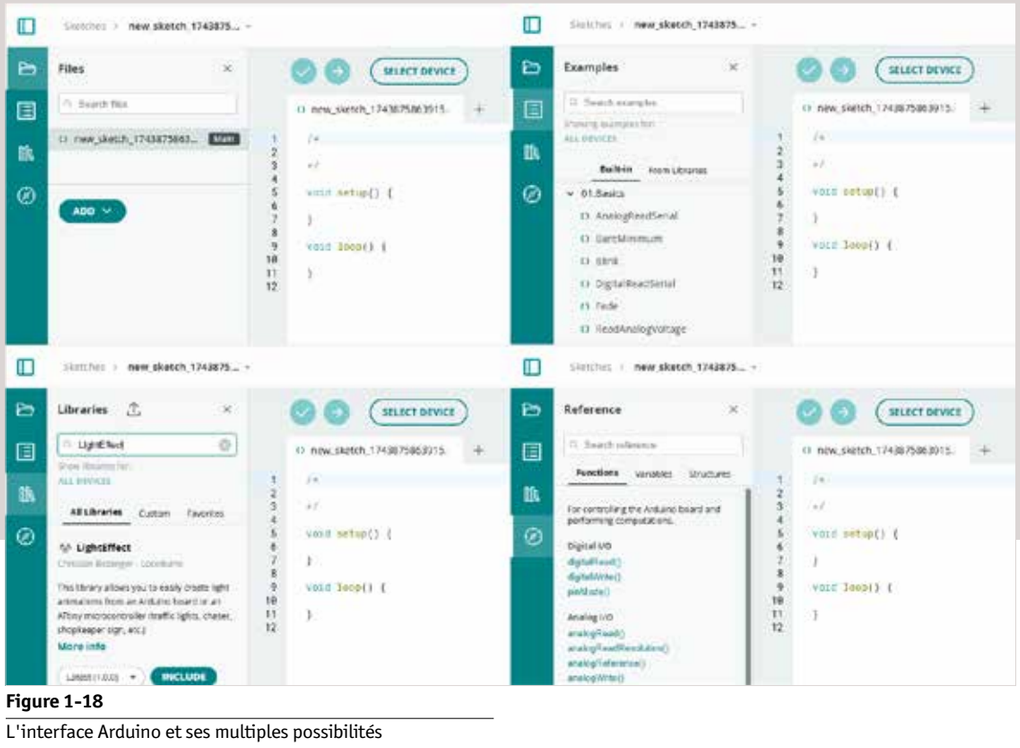


Figure 1-18

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.

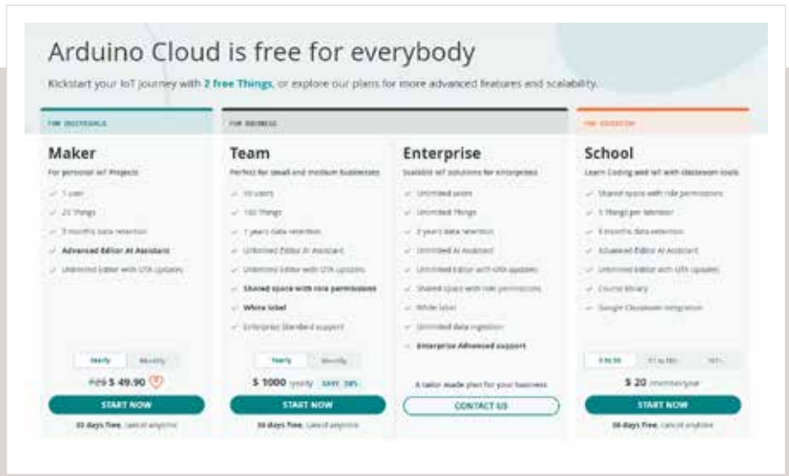


Figure 1-19

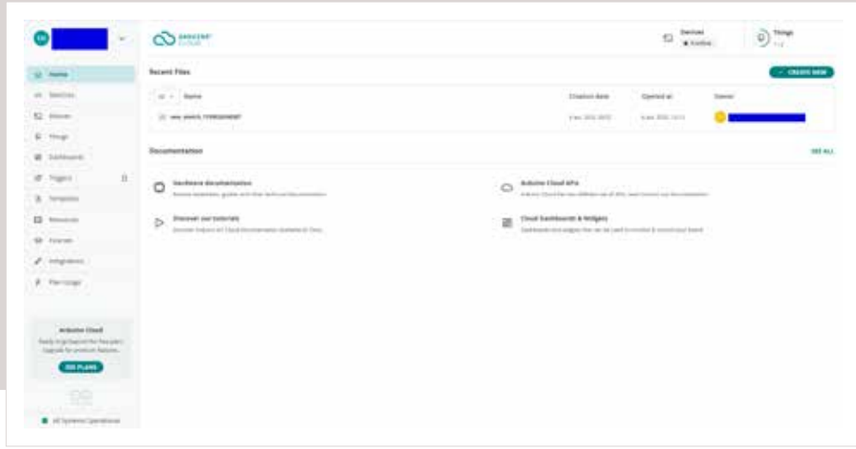


Figure 1-20

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.

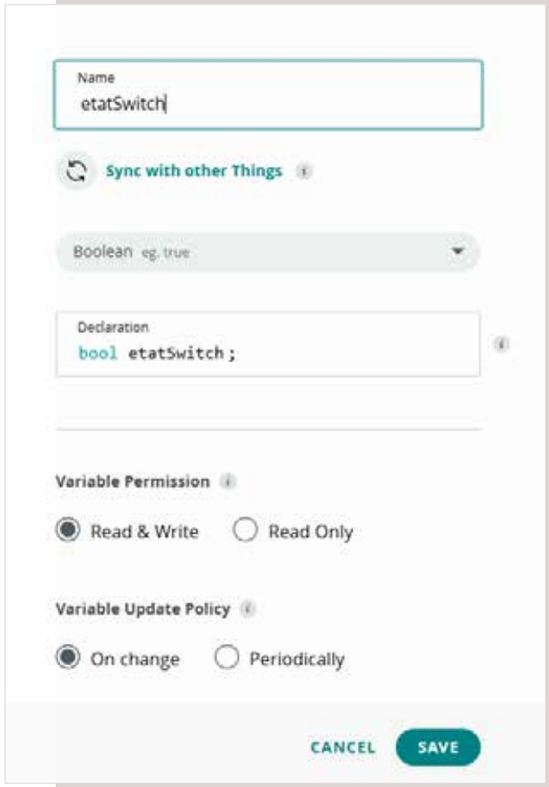


Figure 1-21

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).

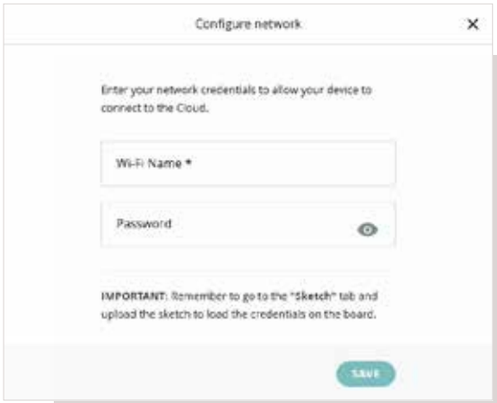


Figure 1-23

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.

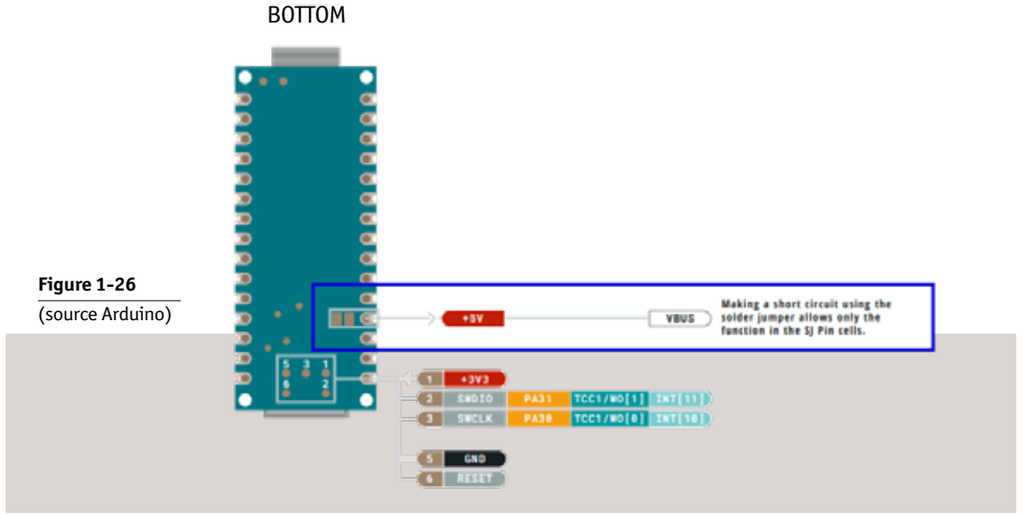


Figure 1-26
(source Arduino)

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

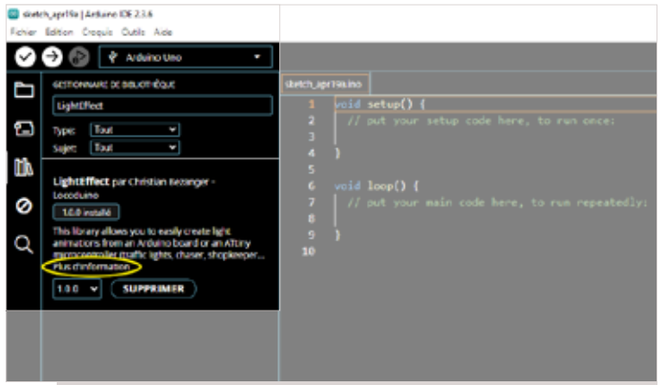


Figure 1-27

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.

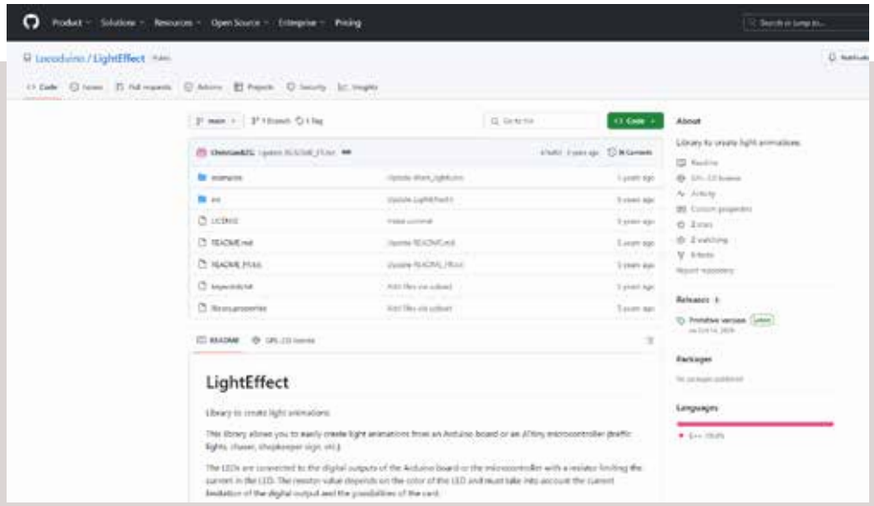


Figure 1-28

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

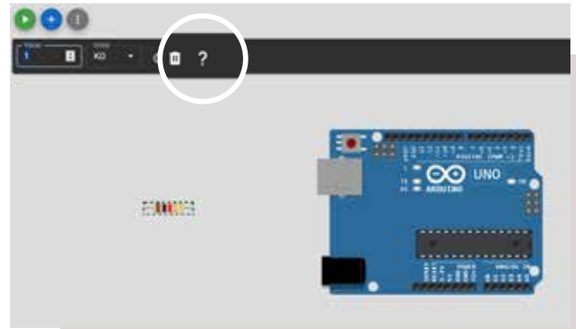


Figure 1-29

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.

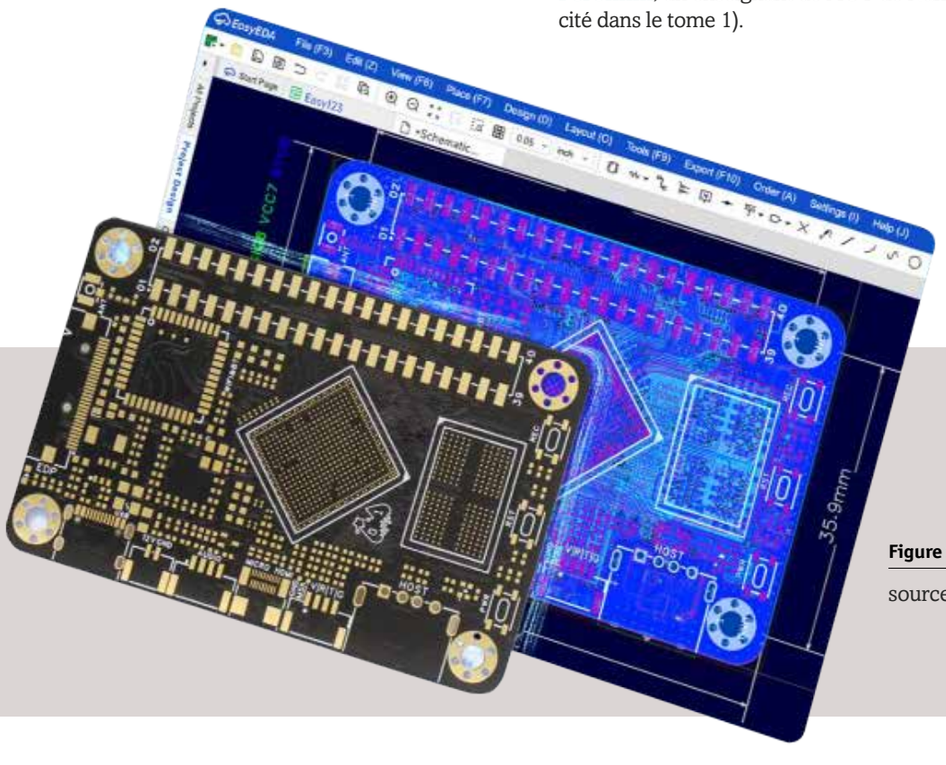


Figure 1-30

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.

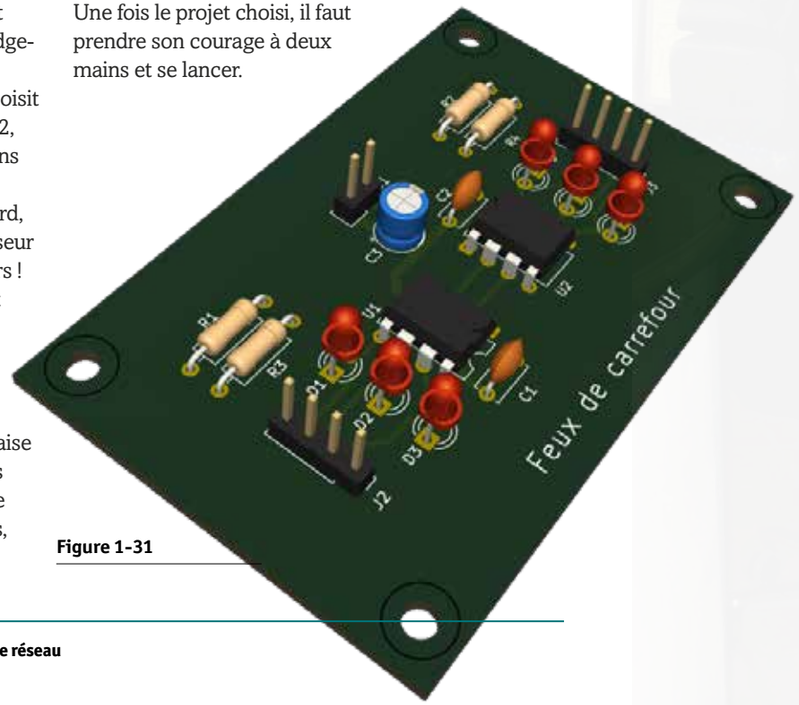


Figure 1-31

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 !

02 Savoir comment trouver de l'aide

Tracteur en panne pour repousser le vol d'Air France : savoir trouver de l'aide ! (MiniaturWunderland à Hambourg)

Si le chapitre précédent a fait le tour des connaissances à acquérir pour concevoir un projet d'automatisation, il peut arriver de se sentir bloqué dans les différentes techniques à appliquer. Un peu d'aide est nécessaire et plusieurs possibilités existent pour la trouver. En fait, on peut se considérer comme un expert si on sait se tirer seul de ce genre de mauvais pas et ce chapitre donne quelques pistes à suivre.

2.1 / TROUVER DE L'AIDE SUR LE SITE D'ARDUINO

Le site Arduino vous permet de trouver de l'aide si vous butez sur un point, d'une part avec une documentation extrêmement fournie, d'autre part par sa communauté d'adeptes et son forum. Comme quasiment toutes les entreprises, Arduino a également sa chaîne YouTube et vous pouvez trouver des tutos pour vous former.

Commençons par voir la **documentation** et imaginons que nous voulions utiliser une carte Giga R1 WiFi. On va dans la partie hardware, Mega Family, et on clique sur le nom de la carte (**figure 2-1**).

On arrive sur une page entièrement consacrée à la carte Giga R1 WiFi commençant par quelques photos montrant le produit. La **figure 2-2** montre la carte en perspective : format Mega avec des connecteurs en plus pour le Display Shield.

Figure 2-1
(source Arduino)

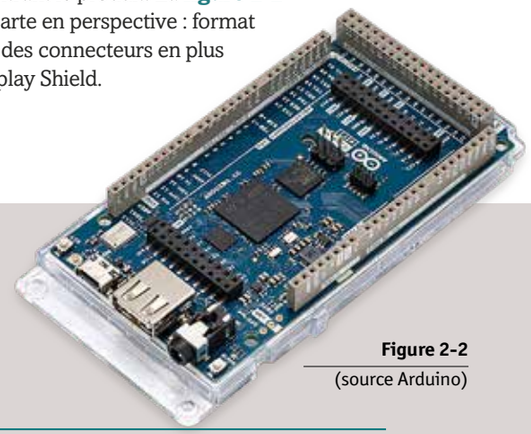
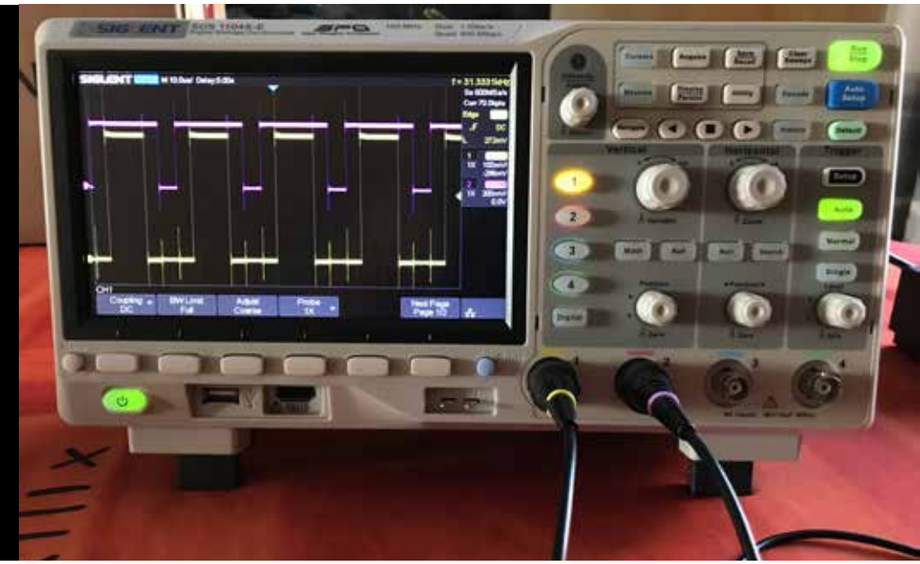


Figure 2-2
(source Arduino)



L'achat d'un oscilloscope n'est pas nécessaire mais procure une aide non négligeable

Juste au-dessous se trouve une description de la carte : si vous avez du mal avec l'anglais, vous pouvez utiliser le traducteur de votre navigateur s'il est disponible pour la page (malgré quelques erreurs, c'est mieux que rien). Dans cet aperçu, vous avez des liens en bleu que vous pouvez ouvrir dans un autre onglet, comme par exemple la datasheet du microcontrôleur de la carte.

Juste après se trouvent les **spécifications techniques** sous forme d'un tableau où on peut voir les différentes ressources de la carte. **La figure 2-3** donne le début du tableau. Bon, le bus CAN a été traduit par PEUT, mais c'est un traducteur et pas un électronicien !

Continuons à descendre la page : nous trouvons la documentation (**figure 2-4**) permettant de télécharger deux fichiers PDF, d'une part la **notice de 22 pages de la carte**, d'autre part le **schéma de câblage de la carte** (n'oublions pas que tout est en OpenSource, donc à la disposition de tous). Je vous invite, surtout si vous achetez la carte, à **bien télécharger ces deux PDF** qui vous seront d'une grande utilité.

Figure 2-3
(source Arduino)

Figure 2-4
(source Arduino)

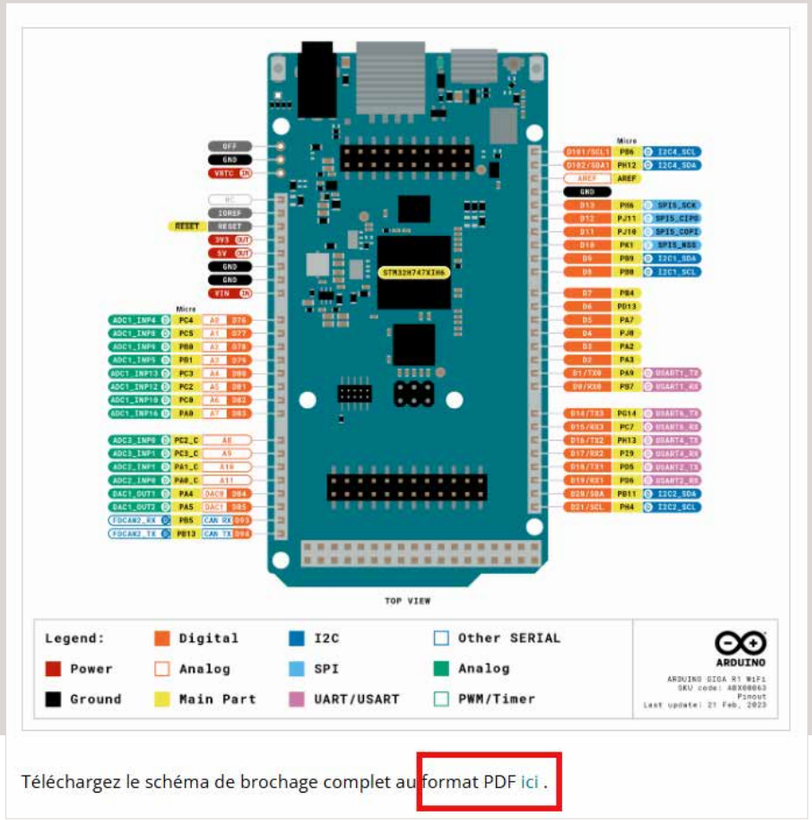


Figure 2-5
(source Arduino)

Mais s'il y a bien un schéma à conserver sous la main, c'est le **schéma de brochage de la carte** (**figure 2-5**). Et le mieux est encore de télécharger le PDF car au lieu d'une seule figure, on a un livret de 9 pages avec toutes les fonctions possibles pour les broches.

Vous avez aussi accès à une FAQ (**Frequently Asked Questions** ou questions fréquemment posées), ainsi qu'un lien vers encore plus de documentation comme le montre la **figure 2-6**.

Figure 2-6
(source Arduino)

Ce lien ouvre une nouvelle page en anglais uniquement, véritable caverne d'Ali Baba consacrée à la carte Giga R1 WiFi et proposant des **manuels**, des **schémas**, des **tutos**, etc. Lire tout cela prendrait certainement des heures : quoi que vous cherchiez, il n'y a qu'à cliquer ! Vous pouvez faire cela pour chacune des cartes proposées et lorsque vous serez habitué à la structure du site, vous trouverez n'importe quel document en un instant. **Notice, schéma et brochage d'une carte sont vraiment des documents indispensables pour développer un projet**.

La communauté et son forum permettent aussi d'obtenir des réponses : on accède aux différentes rubriques à partir de la page d'accueil, menu community, avec le résultat montré par la **figure 2-7**. De là, on peut choisir d'accéder au forum ou bien à la chaîne YouTube, ou encore au Centre de projets ou à la page GitHub (voir chapitre 1).

Figure 2-7
(source Arduino)

Mais avant de poser une question sur le forum, il convient de chercher si la réponse n'a pas déjà été donnée. Il convient aussi de regarder si la réponse n'est pas dans la documentation car ce serait stupide (et peu courtois) de poser une question si la réponse est dans la documentation et cela risquerait de montrer qu'on n'a pas envie de chercher. Les membres du forum sont des bénévoles et répondre prend du temps, alors ne leur faisons pas perdre du temps pour des banalités. Le forum d'Arduino doit rester **le dernier recours après avoir épuisé toutes les autres possibilités** dont je parle un peu plus loin.

La chaîne YouTube est aussi une autre façon d'apprendre et de trouver des réponses. Les vidéos sont en anglais, mais il est possible (du moins pour certaines) d'obtenir une traduction simultanée dans la langue de votre choix, sous forme de sous-titres. En accédant à la page des vidéos, on se rend compte qu'elles sont **classées par sujets**. Il est bien évidemment possible de s'abonner à la chaîne si on veut rester au courant des dernières nouveautés.

2.2 / FAIRE DES RECHERCHES SUR INTERNET

Internet permet de trouver de la documentation, par exemple des datasheets (notices) pour les composants électroniques, mais aussi des réponses sur différents sujets (explication sur le bus CAN par exemple, ou bien sur le fonctionnement du transistor). Parfois, on arrive même à trouver des thèses de doctorat, c'est dire si **l'information obtenue peut être de haut niveau**. Le site Wikipédia peut ainsi être consulté pour dégrossir un sujet. Si le document est en anglais, un traducteur comme Google traduction peut être d'une grande utilité (voir chapitre 3). Mais la mode actuelle est plutôt de **rechercher de l'information sous forme de vidéos** ; on en trouve dans toutes les langues. Ces vidéos sont **chapitrées** et parfois **sous-titrées** et il n'y a donc aucune excuse de ne pas les utiliser. Enfin, il est aussi possible de demander aux Intelligences Artificielles de vous expliquer certains sujets (voir chapitre 3) ; vous avez ainsi une véritable conversation entre l'IA et vous, ce qui vous permet d'expliquer ce que vous ne comprenez pas, alors qu'un tuto sera réalisé pour le plus grand nombre et ne répondra peut-être pas à vos attentes. Il faut tout de même **se méfier de ce qu'on peut trouver sur internet** : il y a sans doute autant de vérités que de bêtises ! À vous de trier et pour cela, de regarder plusieurs sources de documentation.

2.3 / UTILISER LES SITES SPÉCIALISÉS ET LEUR FORUM

Il existe des sites spécialisés en tout et notre hobby ne fait pas exception. Le site de **LR-Modélisme** vous a déjà permis de trouver cet ouvrage, et vous en propose d'autres sur de nombreux sujets, y compris Arduino. **Loco-Revue** publie régulièrement des articles ou des fiches pratiques sur Arduino et a même un forum spécialisé sur le sujet (**figure 2-8**).

Le site **Dronebot Workshop** (en anglais) aborde tous les sujets concernant la mise en œuvre d'Arduino sous forme de tutos en vidéo ou d'articles écrits. C'est à mon avis la meilleure façon d'apprendre les techniques pour Arduino : l'orateur est facile à comprendre, les montages sont expliqués de façon très progressive et les programmes sont très bien commentés. Avec ces vidéos, on comprend vraiment ce qu'il faut faire et je m'en suis souvent inspiré pour mettre en œuvre de nouveaux composants (puce GPS par exemple).

Enfin, le meilleur site à utiliser est sans aucun doute **LOCODUINO** qui a les mérites d'être en **français** et d'**être spécialisé en modélisme ferroviaire** (analogique et numérique). Le site éditorial propose des articles de formation ou bien décrivant des montages. Tout ce qui se trouve sur un réseau de trains miniatures a été traité et certains articles ont même été écrits en collaboration avec Loco-Revue, ce qui permettait aux lecteurs de la revue d'approfondir les explications et de récupérer les programmes. Il y en a pour tous les niveaux, du débutant (animation lumineuse) à l'expert (programmation en assembleur), mais la plupart est pour des gens confirmés qui ont envie de progresser. LOCODUINO a également un forum qui ne traite que des cartes à microcontrôleur et leurs applications en modélisme ferroviaire.

Certains enseignants mettent à la disposition du public les tutos qu'ils réalisent pour leurs étudiants. Ainsi les **vidéos YouTube d'EricPeronnin** de l'université de Nantes, traitent de nombreux sujets d'électronique, de cartes Arduino, d'utilisation de logiciels spécialisés (KiCad par exemple pour concevoir des circuits imprimés), etc. Une grande source d'inspiration pour ceux qui veulent en savoir toujours plus.

Figure 2-8

RÉSUMÉ DU CHAPÎTRE 2

Il arrive toujours un moment où on peut se sentir bien seul face à un problème qu'on ne sait pas résoudre. Mais nous vivons une époque formidable où l'accès à la connaissance devient facile grâce à internet. Plusieurs solutions s'offrent à nous pour trouver de l'aide : tutos, communautés, documentation, etc. Ce chapitre a évoqué quelques pistes possibles et le chapitre suivant nous permettra d'avoir un professeur particulier et gratuit !

2.4 / TÉLÉCHARGER UN PETIT COURS D'ÉLECTRONIQUE GRATUIT

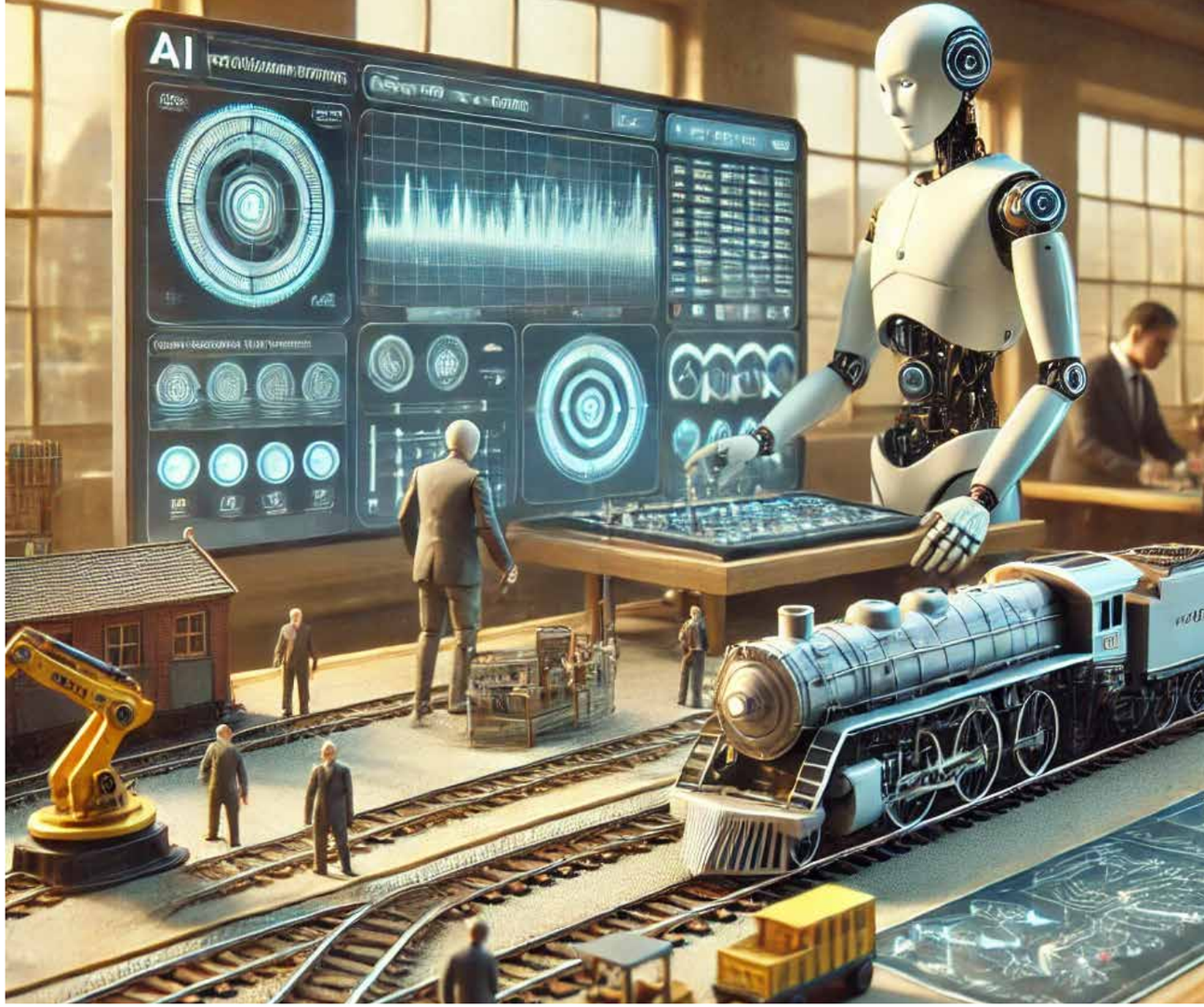
Il y a quelques années, j'ai produit sur le forum de Loco-Revue un petit cours d'électronique que J'ai amélioré et qui est aujourd'hui en **téléchargement gratuit sous forme PDF** sur le site de LOCODUINO (**https://locoduino.org/spip.php?article186**). Ce cours, d'un niveau très facile, n'a aucune prétention mais il pourra peut-être vous aider à comprendre l'électronique câblée et l'électronique programmable.

Voici le plan de ce cours :

Chapitre 1 : Devenir électronicien amateur
Chapitre 2 : Le courant électrique
Chapitre 3 : Les résistances et les condensateurs
Chapitre 4 : Les diodes et les diodes électroluminescentes
Chapitre 5 : Les transistors
Chapitre 6 : Les relais
Chapitre 7 : Les régulateurs de tension
Chapitre 8 : Généralités sur les circuits intégrés
Chapitre 9 : La logique et les portes logiques
Chapitre 10 : Le trigger de Schmitt
Chapitre 11 : Les compteurs
Chapitre 12 : Le temporisateur NE555
Chapitre 13 : L'amplificateur opérationnel
Chapitre 14 : Vers une électronique moderne et simplifiée
Chapitre 15 : Electronique programmable et réseaux de trains miniatures
Chapitre 16 : Présentation du module Arduino Uno et de l'environnement de développement intégré
Chapitre 17 : Analyse des tâches à exécuter et écriture du programme
Chapitre 18 : Comment gérer les signaux d'entrée et les signaux de sortie
Chapitre 19 : Le bouton-poussoir dans tous ses états
Chapitre 20 : Les fonctions électroniques en électronique programmable
Chapitre 21 : Interface d'entrée et interface de sortie
Chapitre 22 : Les capteurs
Chapitre 23 : La commande du réseau
Chapitre 24 : Les boucles et les tests conditionnels
Chapitre 25 : Les interruptions externes
Chapitre 26 : Pourquoi le programme ne fonctionne-t-il pas ?
Chapitre 27 : Sous le capot de l'Uno, un moteur : l'ATmega328P
Chapitre 28 : La face cachée des fonctions pinMode, digitalWrite et digitalRead
Chapitre 29 : Plusieurs modules Arduino sur un même réseau
Chapitre 30 : Le µC ATtiny45
Chapitre 31 : Quelques conseils pour aller plus loin

Beaucoup de choses ont évolué depuis la rédaction de ce cours, mais je ne doute pas qu'il vous sera certainement utile.

03 l'Intelligence Artificielle



Quand l'IA jouera au train ! (Image réalisée par une Intelligence Artificielle)

On entend beaucoup parler de l'Intelligence Artificielle depuis quelques temps grâce aux media qui relaient les progrès réalisés dans ce domaine. Et on a tous une idée plus ou moins précise sur la question. Nous allons voir dans ce chapitre que l'IA peut être une aide pour programmer et un excellent professeur pour progresser.

3.1 / QU'EST-CE QUE L'IA ? CE QU'ELLE PEUT FAIRE ET NE PEUT PAS FAIRE

Qu'est-ce que l'IA ? Voilà un vaste sujet qui peut déboucher sur un vaste débat ! Il y a ceux qui sont farouchement pour et ceux qui sont farouchement contre, mais il n'est pas certain que ces deux catégories sachent exactement ce qu'est une IA, comment elle fonctionne, ce qu'on peut en attendre. Je me situe entre ces deux extrêmes, peut-être pas exactement au centre, penchant sans doute plutôt pour. En effet, je ne considère pas l'IA comme la panacée qu'on nous annonce mais plutôt comme **un nouvel outil** qu'il vaut mieux savoir utiliser. Car, n'en doutons pas, les IA vont se développer et finiront par s'imposer ; la preuve, elles sont déjà parmi nous. Je ne vais donc pas débattre des bienfaits et des méfaits des IA pour l'humanité et mon rôle se cantonnera à décrire comment il faut faire pour que l'IA soit une aide.

J'ai acquis en quelques mois de travail acharné, une certaine expertise sur l'utilisation des IA et j'ai écrit un premier article sur l'apport des IA dans le modélisme ferroviaire dans Loco-Revue qui, une fois de plus, a été la première revue de modélisme ferroviaire en France à aborder ce sujet de haute technologie (LR 928 de novembre 2024). Dans cet article, **j'ai démontré qu'une IA peut produire du code pour nos cartes Arduino et que le programme peut fonctionner … ou pas !** Et j'ai aussi démontré que si l'IA ne sait pas faire, elle peut apprendre de notre part et finir par y arriver. Tant que cela fonctionne, l'IA peut être une aide pour un débutant qui n'a plus besoin d'apprendre à coder. Oui, mais quand cela ne fonctionne pas, ce même débutant est bien incapable d'en comprendre la raison. Doit-il renoncer pour autant ?

Fort de ces premières expériences, j'ai persévéré, ce qui m'a permis de mieux comprendre comment utiliser les IA. L'expérience était tentante de concevoir un petit réseau miniature (un réseau de débutant), géré par une carte Arduino dont les programmes avaient été écrits par l'IA ChatGPT à plus de 95%. Ce réseau, appelé **EX_MACHINA**, a été décrit dans Loco-Revue de juillet 2025, la carte Arduino gérant la circulation de deux trains simultanément ou l'un après l'autre selon les conditions demandées, ainsi que la signalisation lumineuse. Si une IA a été capable de comprendre les problèmes à résoudre, de proposer des solutions et d'écrire les programmes en conséquence, alors c'est la preuve que **les IA peuvent nous aider dans notre hobby**.

Pour autant, la réussite de l'IA à écrire les programmes d'EX_MACHINA, n'est pas due au fait qu'elle est supérieurement intelligente, mais au fait qu'elle a été **sollicitée en lui fixant un cadre strict et des consignes rigoureuses**. Elle n'a pas conçu le circuit, ni le décor (d'ailleurs je n'en avais pas mis), ni le but à atteindre. Elle n'a fait que produire du code pour Arduino et trouver des algorithmes qui fonctionnent, et ce n'est déjà pas si mal.

3.2 / LES DIFFÉRENTES IA ET LEURS SPÉCIALISATIONS

Il n'existe pas qu'une seule IA mais bien plusieurs, développées par différentes entreprises, dans différents pays car **les IA représentent un enjeu majeur**. Certaines IA sont spécialisées dans un domaine particulier : moteur de recherche, traduction de texte, élaboration d'images ou de vidéos, création musicale, reconnaissance vocale ou faciale, reconnaissance et manipulation d'objets, conduite assistée, etc. Par exemple, FluxPro est une IA spécialisée dans la génération d'images alors que Novelai est spécialisée en littérature et Sora en production vidéo. **On ne choisit donc pas une IA parce que la télévision ou les réseaux sociaux en parlent, mais parce que cette IA est spécialisée dans ce qu'on veut faire**. Aux États-Unis, les IA sont déjà utilisées pour produire des fonds de décor sur les réseaux miniatures.

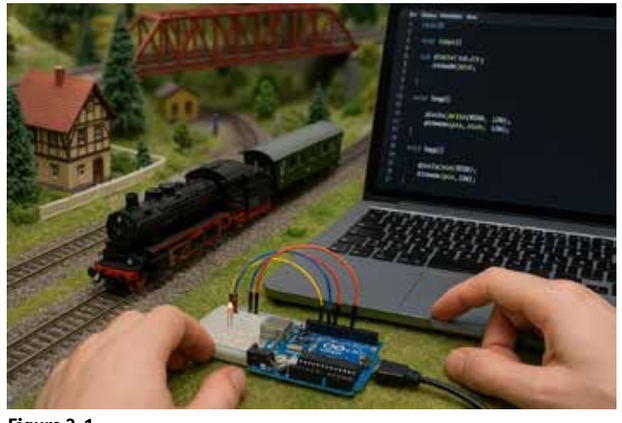


Figure 3-1
image générée par ChatGPT

La **figure 3-1** montre une image générée par une IA pour illustrer le train miniature associé à la programmation des cartes Arduino par l'intelligence artificielle.

3.3 / GOOGLE TRADUCTION : UNE IA QUI NOUS AIDE VRAIMENT

Google traduction est une IA qui fait de la **traduction de textes d'une langue à une autre**. Comme je vous l'ai déjà fait remarquer, il lui arrive de faire quelques erreurs, mais souvent les traductions sont faites sans que l'IA ne connaisse le contexte. Malgré ces petites erreurs, Google traduction va considérablement aider ceux qui ne maîtrisent pas la langue de Shakespeare utilisée dans toutes les notices de composants. Comme ces notices sont fournies sous forme électronique (souvent des fichiers PDF), il est très facile de faire un copier-coller d'une phrase dans Google traduction et avoir ainsi sa traduction.

La **figure 3-2** donne un exemple de traduction du français vers l'anglais, du paragraphe précédent.

Ce n'est pas si mal, même si j'aurais traduit « notices de composants » par « datasheets » au lieu de « component instructions », mais encore une fois, le contexte n'était pas évident.

Pour les documents plus élaborés, il suffit d'entrer le document complet avec un glisser-déposer (**figure 3-3**). J'ai essayé avec la datasheet de la carte Giga R1 WiFi (document de 4 Mo contenant 22 pages) et le résultat pouvait être affiché ou bien téléchargé (**figure 3-4**). Même si je n'ai pas tout contrôlé, le résultat semblait probant.

Avec cette intelligence artificielle, vous avez un outil qui peut vous aider vraiment à comprendre toutes sortes de documents, en anglais ou autres langues.

3.4 / LES IA GÉNÉRATIVES

Figure 3.1
ChatGPT, DeepSeek, Perplexity

Les IA génératives sont les plus connues du public : **ChatGPT, DeepSeek, le Chat de Mistral** (IA française) pour ne citer que quelques exemples. Issues de **LLM (Large Language Model ou Modèle Linguistique Étendu)**, elles sont capables de comprendre un texte et d'y répondre, dans plusieurs langues. On peut donc converser avec elles comme on le ferait avec un interlocuteur. Elles sont programmées pour simuler des réactions humaines, et grâce au « Deep Learning », leur domaine de connaissances est étendu. Elles ne connaissent pas tout pour autant et il n'y a pas (encore) d'IA spécialisée en modélisme ferroviaire !

Comme ces IA sont spécialisées dans le langage, **elles connaissent les langages de programmation comme le C/C++, HTML, Python, CSS, etc**. Elles sont donc capables de générer du code qui peut fonctionner ou pas comme je l'ai déjà dit. Les chances de réussite d'un programme peuvent être améliorées **si on donne des instructions précises à l'IA** : c'est ce qu'on appelle **rédiger le prompt, ou dans le langage des IA « prompter »**.

Comme il y a plusieurs IA pour générer du code, on peut les mettre en concurrence et voir laquelle donne le code le plus compact et le plus efficace. Il suffit d'ouvrir un compte sur les IA qu'on veut utiliser : ceux-ci sont payants mais une formule de base, limitée à quelques interventions par jour, est gratuite et suffisante pour découvrir ce nouvel outil. La **figure 3-5** montre quelques logos d'Intelligence Artificielle.

Figure 3-2

Figure 3-3

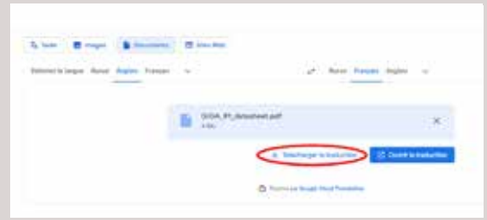


Figure 3-4

3.5 / COMMENT RÉDIGER LE PROMPT ?

Pour obtenir un résultat cohérent, **la rédaction du prompt est très importante**. Nous allons le découvrir ensemble sur l'exemple donné par la **figure 3-6**, qui représente un montage de motorisation d'une aiguille par servomoteur. Un poussoir permet de lancer la manœuvre de l'aiguille et deux LED permettent de connaître sa position : LED verte si aiguille droite et LED rouge si déviée. Le servomoteur se déplace entre deux positions extrêmes (70° et 110°) correspondant à aiguille droite ou déviée. Nous allons voir comment demander à une IA d'écrire le programme.

Nous devons expliquer le but à atteindre : **obtenir un mouvement lent du servomoteur durant deux secondes, pour passer de la position 70 à 110 et réciproquement chaque fois qu'on appuie sur le poussoir.** Il faut dire également à l'IA comment sont connectés les différents composants : servomoteur, poussoir, LED verte et rouge. Voici le prompt :

Écrire un programme pour carte Arduino Uno pour obtenir un mouvement lent d'un servomoteur durant deux secondes, pour passer de la position 70° à 110° et réciproquement chaque fois qu'on appuie sur un poussoir. Le servomoteur est connecté sur la broche 10, le poussoir sur la broche 2. Une LED verte connectée en 5 doit être allumée uniquement si la position du servomoteur est 70°. Une LED rouge connectée en 7 doit être allumée uniquement si la position du servomoteur est 110°.

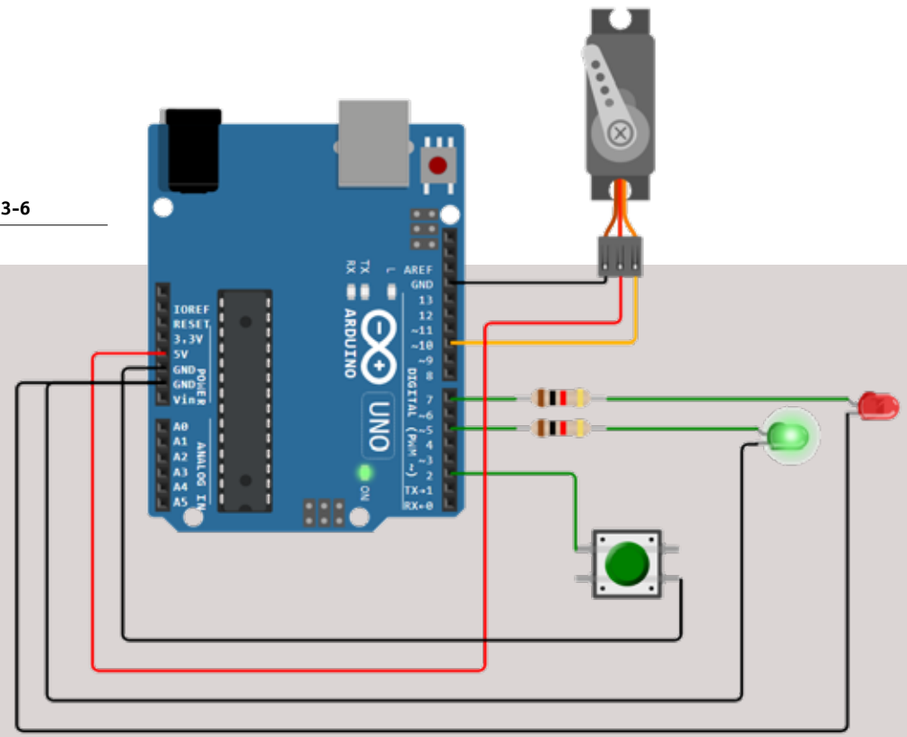


Figure 3-6

L'IA résume la demande, puis donne son programme. Elle donne aussi quelques conseils de branchement des composants. Une fois le programme importé sur le montage, je constate que le fonctionnement est identique au montage original. Voici le lien vers le programme proposé : **https://wokwi.com/projects/427841977627908097**. J'en profite pour lui demander une modification à laquelle je n'avais pas pensé moi-même :

Peux-tu corriger ce programme pour que les LED soient toutes les deux éteintes durant le mouvement du servomoteur et allumées ensuite pour indiquer la position du servomoteur ?

Le programme est à nouveau opérationnel (**https://wokwi.com/projects/427842483370860545**) du premier coup. Je modifie une fois de plus la demande pour reproduire cette fois le fonctionnement d'une barrière de PN (avec deux LED rouges) :

Peux-tu modifier ce programme pour que le servomoteur aille de la position 90° à 180° lors du premier appui, et faire clignoter les deux LED pendant et après le mouvement. Un deuxième appui doit éteindre les LED immédiatement, puis faire le mouvement du servomoteur de la position 180° à 90°. Et ainsi de suite.

Cette fois, le programme ne fonctionne pas tout à fait comme il devrait. Je le lui indique en demandant la correction (**https://wokwi.com/projects/427843196418382849**) :

Peux-tu corriger ce programme car lors du premier appui sur le poussoir, les LED ne clignotent pas dès le début du mouvement du servomoteur mais seulement quand il arrive en position finale. Le mouvement du servomoteur doit se faire en 5 secondes.

L'IA commence par analyser son erreur (figure 3-7) et cette fois, son programme fonctionne comme prévu (**https://wokwi.com/projects/427843639794124801**).

Figure 3-7

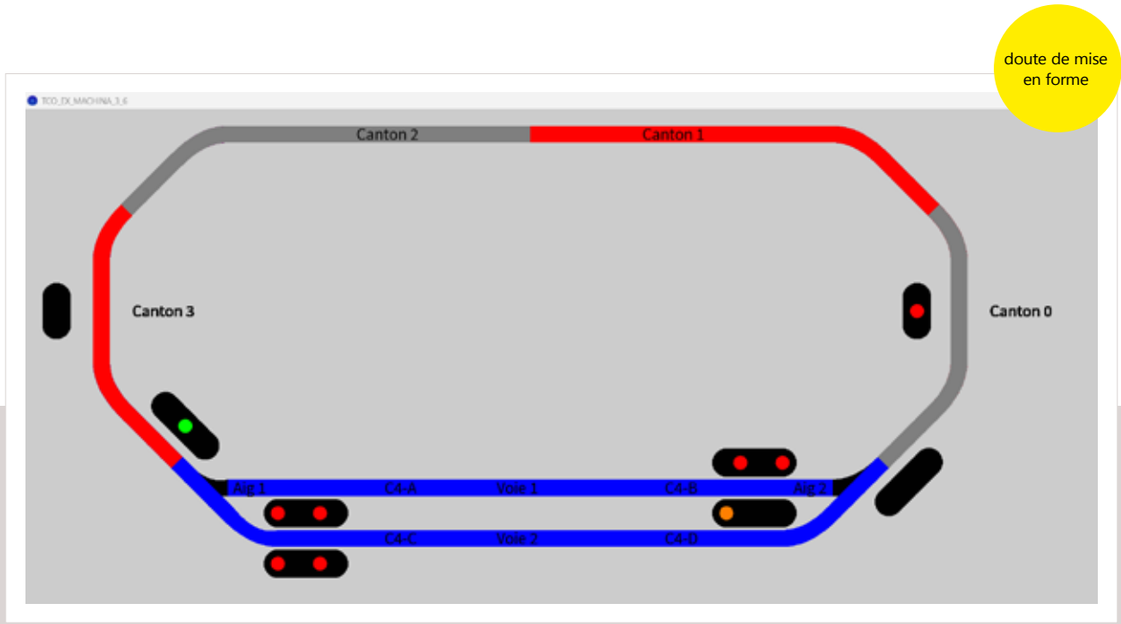
Comme vous le voyez, dans chacun de mes prompts j'ai essayé d'être **le plus précis possible** sur ma demande et c'est sans doute pourquoi cela a fonctionné assez bien. Grâce à l'IA nous sommes passés d'une commande d'aiguille à une commande de barrière de PN. Cette démonstration nous a permis de voir que **les IA peuvent aussi se tromper, mais elles peuvent corriger le tir si on leur explique ce qui ne fonctionne pas**. Ce que nous venons de faire est à la portée d'un modéliste qui ne connaît pas la programmation.

L'erreur de l'IA pourrait devenir un problème sur un cas plus compliqué ; **il faut toujours décomposer un cas compliqué en une suite de cas plus simples** et on finit par y arriver. D'ailleurs, pour travailler sur un projet complexe, il est indispensable d'agir par étapes et de demander à l'IA de **créer des fonctions** qui peuvent ensuite être jointes au programme principal. Et pour que tout s'articule bien, il peut être nécessaire d'imposer à l'IA le nom et le type de certaines variables dans nos prompts. C'est comme cela que j'ai mis au point **le réseau EX_MACHINA (décrit dans LR936 de juillet 2025)**. Tout cela n'est pas si compliqué, et **travailler avec une IA permet aussi d'apprendre très vite** car il suffit d'analyser son programme pour découvrir certaines règles d'écriture du langage C (ou d'un autre langage).

3.6 / COMMENT CONTRÔLER LE LOGICIEL DÉLIVRÉ ?

Il suffit de faire un copier-coller du programme de l'IA dans l'IDE d'Arduino pour vérifier qu'il compile sans problème pour la carte que vous utilisez. Honnêtement, je ne crois pas que l'IA puisse faire une erreur qui empêcherait la compilation dans la majorité des cas, mais cela m'est tout de même arrivé lorsque j'ai voulu la faire travailler sur une bibliothèque très peu connue, qui nécessitait d'instancier un objet d'une certaine façon.

Le fait qu'un programme compile ne signifie pas qu'il fonctionne comme on voudrait. C'est là que les simulateurs d'Arduino sont intéressants car si le montage peut être réalisé avec les bons composants, il suffit d'importer le programme de l'IA dans le simulateur et de passer en mode simulation. S'il manque des composants au simulateur pour reproduire le montage, alors il faut tester le programme sur le prototype et programmer la carte chaque fois qu'un nouveau programme est fourni par l'IA.



TCO d'EX_MACHINA, réseau conçu par une Intelligence Artificielle

3.7 / QUAND LES IA HALLUCINENT

Les IA ont tendance à vouloir nous donner une réponse systématiquement, même lorsqu'elles disposent d'assez peu de données sur un sujet ou bien si on leur demande un travail dans un cadre pour lequel elles ne sont pas conçues. Ces réponses sont alors farfelues mais pourraient passer pour crédibles car les IA vont les présenter d'une façon appropriée : **on dit alors que les IA hallucinent** et c'est pourquoi il est nécessaire de conserver un esprit critique sur ce qu'elles délivrent. Dans le domaine de la production de code, il est facile de vérifier le programme délivré, comme l'explique le paragraphe précédent. Dans d'autres domaines, on peut se rendre compte que les IA sont incapables de produire des résultats exploitables.

J'ai essayé par exemple d'utiliser ChatGPT pour réaliser un circuit imprimé (PCB) : l'IA devait produire les fichiers Gerber en utilisant KiCad 9.0 (voir chapitre 1). L'IA avait très bien compris la demande et devait nous fournir l'ensemble des fichiers après un certain délai. Ne voyant rien arriver au bout d'un quart d'heure, j'ai relancé l'IA qui m'a soutenu qu'elle travaillait à élaborer les fichiers et qu'il n'était pas nécessaire de la relancer. J'ai pris mon mal en patience, mais après une heure d'attente, je l'ai à nouveau sollicitée pour finalement obtenir une suite de fichiers vides et donc inexploitables. Lorsque j'ai fait la remarque à l'IA, elle a reconnu qu'elle ne pouvait pas générer ces fichiers à partir du logiciel KiCad, ce qu'elle aurait pu me dire dès le début.

C'est la même chose dans le domaine de la génération d'images ou de vidéos qui peuvent être fournies avec des artéfacts (personnages irréels avec six doigts, animaux dont la tête ne correspond pas au corps, etc.). Une chose est sûre : les IA n'en sont qu'au début et vont forcément s'améliorer puisque d'énormes programmes de recherche sont lancés dans tous les pays avec de forts financements (17 milliards d'euros en France, décision prise à « Choose France » à Versailles en mai 2025). Il est à parier qu'un jour, les IA seront capables de dessiner nos PCB et commander leurs fabrications directement auprès d'une entreprise ; ce sera toujours à nous de payer la facture, sauf si elles nous demandent notre numéro de carte bancaire !

RÉSUMÉ DU CHAPÎTRE 3

L'Intelligence Artificielle n'est sans doute pas la solution à tout mais c'est une aide indéniable. Elle peut, comme je l'ai montré, écrire des programmes à notre place. Mais surtout, elle peut nous aider à progresser à condition d'avoir envie de comprendre ce qu'elle délivre. Il suffit de regarder comment elle s'y prend et il n'y a certainement pas de professeur plus patient pour nous enseigner la programmation. Cet outil s'imposera de plus en plus dans l'avenir et c'est maintenant qu'il faut apprendre à le maîtriser !

04 Les nouvelles cartes à microcontrôleur

De nouvelles cartes à microcontrôleur sortent régulièrement, affichant des performances sans cesse croissantes, tant en matière de rapidité de traitement qu'en matière de stockage d'informations. Mieux, ces cartes sont pourvues de ressources qui présentent un intérêt certain pour le modélisme ferroviaire, comme par exemple un contrôleur CAN ou la possibilité de se connecter à un réseau WiFi.

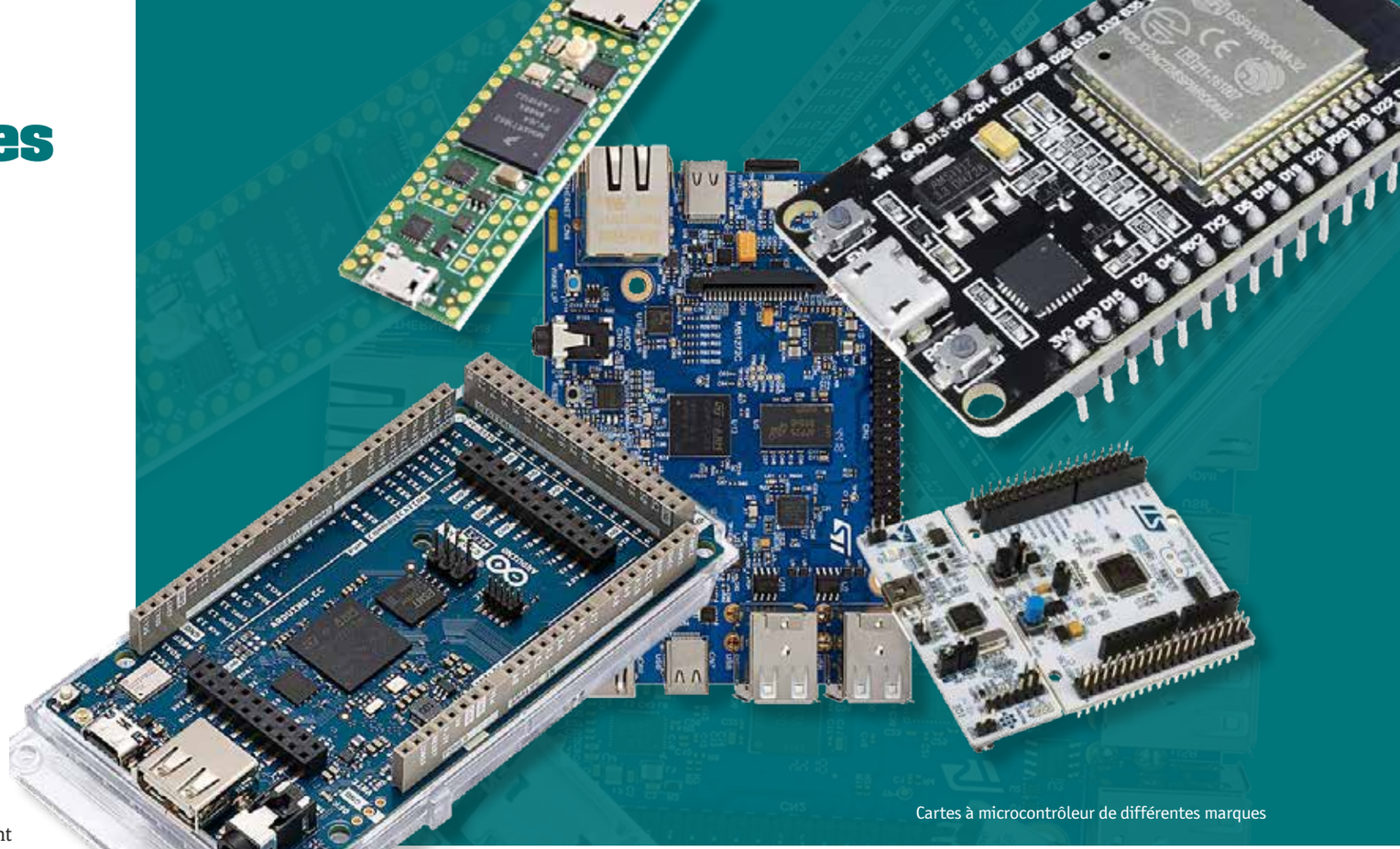
4.1 / LES CARTES D'ORIGINE ARDUINO : QUATRE GRANDES FAMILLES

Arduino classe ses cartes à microcontrôleur en quatre grandes familles :

La **famille classique** qui comprend 8 cartes, souvent au format Uno, et trois cartes shield (moteur, relais, ethernet). C'est dans cette catégorie qu'on retrouve la carte Uno R3 ou encore les nouvelles cartes Uno R4 dont je parle un peu plus loin.

La **famille Nano** qui comprend 7 cartes dont la carte Nano qui est équivalente à une Uno R3 mais en format plus petit, la Nano-33-IoT, la Nano RP2040 (microcontrôleur du Raspberry Pi), la Nano ESP32 (microcontrôleur des cartes ESP de chezEspressif) et une carte « Carrier » qui reçoit la carte au format Nano en son centre, un peu comme un shield à l'envers, et qui permet entre autre de contrôler quatre moteurs DC et quatre servomoteurs.

Figure 4-1
source Arduino



Cartes à microcontrôleur de différentes marques

La **famille MKR** (pour Maker) qui comprend 9 cartes à base de microcontrôleur Arm© Cortex© M0 32 bit SAMD21 et 9 cartes shield. Cette catégorie, très spécialisée, **s'adresse à ceux qui développent des projets de A à Z**, électronique, informatique, faisant appel à la conception de circuits imprimés professionnels et à l'impression 3D. Ces projets sont généralement embarqués et autonomes sur batterie et consomment assez peu.

La **famille Mega** (**figure 4-1**) comprend 3 cartes au format Mega dont la classique Mega 2560, mais aussi la carte Due avec un microcontrôleur plus puissant ou encore la Giga R1 WiFi qui peut être considérée comme la Rolls des cartes Arduino. Cette famille comprend aussi le shield Giga Display conçu pour la Giga R1. Les prix de ces deux derniers produits de haute gamme sont en rapport avec leurs performances, mais **l'association des deux peut avoir son utilité en modélisme ferroviaire** pour réaliser une centrale DCC graphique ou un TCO ou les deux à la fois.

Il convient donc de bien étudier la gamme proposée en fonction du projet à réaliser, **notamment en termes de ressources**, mais plutôt que se précipiter sur le plus cher, on doit se poser la question de l'utilité de ces ressources. Dans notre domaine, un microcontrôleur 8 bits sera déjà bien suffisant et un 32 bits pas forcément nécessaire, sauf que les prix de ces derniers ont beaucoup baissé, alors pourquoi s'en priver ?

4.2 / UNO R4 MINIMA ET R4 WIFI

En 2023, Arduino a sorti deux nouvelles cartes qui ont la forme de la carte Uno R3, les mêmes dimensions, les mêmes connecteurs avec un microcontrôleur **RA4M1, un Arm© Cortex© M4 core de Renesas, de 32 bits** donc plus puissant, plus rapide et avec plus de mémoire. Ces cartes ont été appelées Uno R4 et se déclinent en deux versions : **Uno R4 Minima** (version minimaliste si on peut dire) à 22 euros et **Uno R4 WiFi** (connectable en WiFi et Bluetooth) à 30 euros (prix avril 2025). Ces deux cartes travaillent également en 5 V ce qui leur permet **de remplacer une carte Uno R3 dans un projet déjà abouti** afin de lui donner plus de possibilités, et ce sans avoir à modifier les autres composants électroniques. La R4 Wifi présente les mêmes caractéristiques que la R4 Minima mais peut se connecter en WiFi et Bluetooth et dispose en plus d'une matrice de 12 x 8 LED rouges. Comme les connecteurs sont les mêmes, **les cartes R4 acceptent les cartes shields qui étaient conçues pour la R3**. Un programme écrit pour la R3 doit théoriquement tourner sur les R4 **à la condition qu'il ait respecté l'API Arduino** (ensemble des fonctions livrées avec Arduino). Par contre, si le programme manipule les registres internes du microcontrôleur Atmega328P, alors il ne tournera pas sur une R4 dont le microcontrôleur est différent. Il en est de même si le programme utilise une bibliothèque non compatible. Ces deux cartes nécessitent un câble USB de type C pour être reliées à l'ordinateur (prise A normale du côté ordinateur et USB-C du côté carte, câble souvent vendu avec l'appellation USB > USB-C sur l'emballage). Le tableau ci-dessous donnent quelques différences entre la carte Uno R3 et les cartes Uno R4. Bien évidemment, les cartes Uno R4 permettent des possibilités supplémentaires que n'avaient pas les R3 comme un module **CAN** (pour communiquer), une horloge temps réel (**RTC**), une interface humain-machine (**Human Interface Device ou HID**), et surtout une vraie sortie analogique grâce à la présence d'un **DAC (Digital to Analog Converter)**. Enfin, les cartes disposent aussi d'un ampli opérationnel.

	Uno R3	Uno R4 Minima	Uno R4 WiFi
Microcontrôleur	ATmega328P (8 bits)	RA4M1 (32 bits)	RA4M1 (32 bits)
Fréquence	16 MHz	48 MHz	48 MHz
Mémoire Flash	32 k	256 k	256 k
Mémoire SRAM	8 k	32 k	32 k
Mémoire EEPROM	1 k	8 k (data memory)	8 k (data memory)
E/S numériques	14	14	14
PWM	6	6	6
Entrées analogiques	6	6	6
Sortie analogique	0	1	1
WiFi, Bluetooth, IoT	non	non	oui

Attention : alors qu'une sortie de la carte Uno R3 pouvait fournir ou absorber un courant de 20 mA, les cartes Uno R4 sont limitées à 8 mA maximum. Afin de ne pas endommager ces cartes, il est nécessaire de tenir compte de cette limitation et de choisir des résistances de limitation de courant en conséquence (ne jamais descendre en dessous de 470 ohms).

4.3 / CARTE UNO R4 MINIMA

La **figure 4-2** montre les deux côtés de la carte : effectivement, cela ressemble à la carte Uno R3 mis à part l'interface USB-C et le microcontrôleur.

La carte dispose aussi d'un connecteur **SWD (Serial Wire Debug)** que n'a pas la R4 WiFi et qui permet à des utilisateurs avancés et équipés de disposer d'options de débogage.

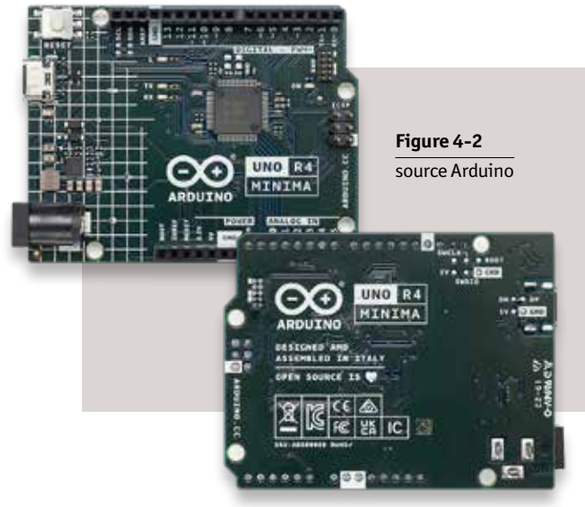


Figure 4-2
source Arduino

4.4 / CARTE UNO R4 WIFI

La **figure 4-3** montre les deux côtés de la carte : on distingue la matrice de 96 LED et la puce (un microcontrôleur ESP32 de chez Espressif) qui permet la connexion en WiFi ou Bluetooth.

Attention car si la carte fonctionne en 5 V, **l'ESP32 fonctionne en 3,3 V**. Cette carte est évidemment compatible avec l'Arduino IoT Cloud, ce qui permet de créer facilement des objets connectés (voir chapitres 1 et 9). On remarque aussi sur la carte un connecteur **Qwiic** qui permet des connexions en I2C avec des composants de l'écosystème Qwiic développé par Sparkfun et Adafruit sous les noms Qwiic ou STEMMA. On peut remarquer aussi la présence de **trois broches supplémentaires OFF, GND et VRTC**. La broche OFF permet de mettre la carte hors tension alors que la broche VRTC permet de garder l'alimentation de l'horloge temps réel (RTC) pour que cette dernière puisse continuer à fonctionner (il suffit d'appliquer une tension comprise entre 1,6 et 3,6 V sur cette broche).

Le site d'Arduino est très bien documenté concernant les nouvelles fonctionnalités de ces cartes par rapport à la carte Uno R3 (utilisation de l'horloge RTC, du DAC, du HID, du CAN, de l'ampli opérationnel, de la matrice de LED, du WiFi et du Bluetooth) mais si vous ne lisez pas l'anglais, vous pouvez vous reporter à cet article que j'ai publié chez LOCODUINO : **https://locoduino.org/spip.php?article340**.

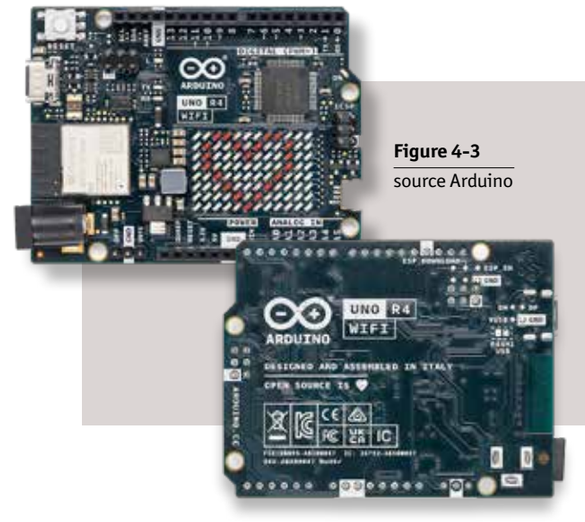


Figure 4-3
source Arduino

L'article fait également le point sur ce que ces cartes ont sous le capot mais aussi sur de petites différences de conception entre la R4 Minima et la R4 WiFi **qui peuvent donner lieu à problèmes si on essaie de remplacer l'une par l'autre dans un projet déjà terminé**.

4.5 / IMPORTATION DANS L'IDE DES FICHIERS NÉCESSAIRES À LA PROGRAMMATION DE CES CARTES

Dans l'IDE, choisissez l'option « Sélectionnez une autre carte et un autre port… » et tapez Uno. La **figure 4-4** montre les cartes Uno connues et on peut voir que les Uno R4 ne sont pas encore installées car elles n'apparaissent pas en brillant.

Figure 4-4

Sélectionnez l'icône « Gestionnaire de carte » (1) et entrez dans la case de recherche UNO R4 (2) ; vous obtenez un package qui est fait pour les deux cartes Uno R4 Minima et Uno R4 WiFi. Il suffit alors de cliquer sur « INSTALLER » (3) comme le montre la **figure 4-5**.

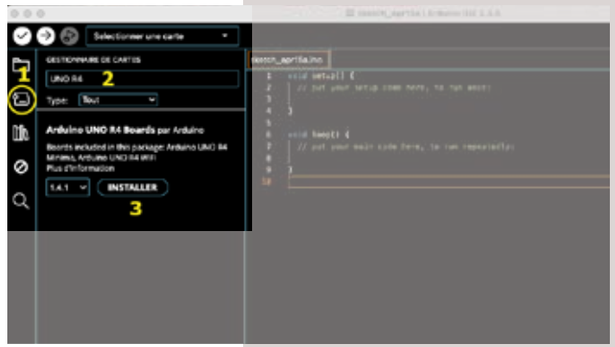


Figure 4-5

Vous aurez à accepter que le logiciel modifie votre ordinateur, comme c'est le cas lorsqu'on veut installer un nouveau logiciel, et pour cela il faut que vous soyez administrateur. Après quelques secondes, vous obtenez le message « **Platform arduino:renesas_uno@1.4.1installed** » (ou une version supérieure à la 1.4.1).
Si vous retournez dans « Sélectionnez une autre carte et un autre port… », vous pouvez constater que les Uno R4 sont bien installées comme le montre la **figure 4-6**.

Figure 4-6

4.6 / LES CARTES ESP

L'écosystème Arduino n'est pas limité aux cartes de marque Arduino. D'autres constructeurs proposent également des cartes à microcontrôleur qui peuvent être programmées par l'IDE d'Arduino. La firme chinoiseEspressif a sorti des cartes bon marché mais très performantes, comprenant un **microcontrôleur ESP32** qui a d'ailleurs été repris par Arduino dans sa carte Nano ESP32.

Fondée en 2008, la société chinoise Espressif fabrique des MCU (Microcontroller Unit) : en 2013, elle a sorti l'ESP8089 doté du WiFi mais qui n'existait qu'en tant que composant et pas en tant que module. L'arrivée en 2014 de l'**ESP8266** (MCU ou module) a comblé cette lacune et le petit module est très vite devenu populaire. En 2016, l'ESP32 a été une amélioration de l'ESP8266 avec une documentation conséquente et la possibilité de kits de développement (**ESP-WROOM ESP DevKit V1**).

L'ESP32 est un MCU 32 bits, généralement doté du WiFi et du Bluetooth (mais pas toujours : voir plus loin), avec un ADC 12 bits (10 bits seulement pour les cartes Arduino de l'époque) et un DAC de 8 bits, un GPIO, le Hall Sensor et des bus de communication comme UART, SPI, I2C, I2S.

Plusieurs séries existent pour l'ESP32 :

- **ESP32** 32 bits single ou dual core, bus caméra et interface SD
- **ESP32-S** en 2020 (S2 single core, WiFi, 12 bits ADC bus caméra – S3 dual core, WiFi et Bluetooth 5, 12 bits ADC, bus caméra, mémoire externe possible)
- **ESP32-C** en 2020 (C2 single core qui remplace l'ESP8266 – C3 qui contient un RTC, C6 qui permet le protocole de communication Thread/Zigbee)
- **ESP32-H** (H2 spécialisé application IoT, avec possibilité de Flash externe)
- **ESP32-C61** utilisé comme MCU ou comme co-processor d'un autre MCU, avec WiFi 6 et Flash externe
- **ESP32-C5** utilisé comme MCU ou comme co-processor d'un autre MCU, avec WiFi 6 double bande et Flash externe
- **ESP32-P** (P4 MCU dual core haute performance, **pas de WiFi ni Bluetooth**)

Les ESP32 existent soit en **SoC (System on a Chip, ou encore composant électronique simple)**, soit en **module (board)** comme WROOM, WROVER, SOLO, PICO, MINI, soit en **DevKit** (voir plus haut). Pour notre hobby, seules les deux dernières possibilités sont exploitables. Devant une offre aussi pléthorique, il n'est pas facile de bien choisir l'ESP32 qu'il nous faut. Pour comparer les différentes versions d'ESP, le site **https://products.espressif.com/#/product-selector** vous sera d'une grande utilité.

Deux cartes populaires de chez Espressif : ESP32 WROMER à gauche et ESP8266 à droite (kit de découverte Freenove)

Plusieurs marques proposent des cartes à base d'ESP32 ; elles sont programmables avec l'IDE 2 d'Arduino ou d'autres plateformes (PlatformIO ou ESP IDF (pour IoT Development Framework) d'Espressif. Les langages sont C/C++, Python, MicroPython, CircuitPython.

Attention : certaines cartes ne peuvent pas être enfichées sur une breadboard et il faut bricoler deux breadboards et les réunir côte à côte sur un même support.

Arduino a sorti une carte au format Nano munie d'un ESP32 : la Nano ESP32. La marque Seeduino a sorti la carte XIAO-ESP32S3 et la marque DFRobot (rappelez-vous du Mini-DFPlayer décrit dans le tome 1) a sorti une carte minuscule DFRobot Beetle-ESP32C3. Ces cartes ont un ou deux USB, un nombre variable de broches E/S (I/O) accessibles et elles restent bon marché (moins de 25 dollars). Assurez-vous cependant d'avoir la documentation de la carte pour vous y référer car elle est indispensable. Le simulateur Wokwi propose quelques cartes ESP32, ce qui peut vous permettre d'en découvrir certaines avant de faire votre choix définitif (S2, S3, C3, C6, H2 et P4 ainsi que les cartes XIAO ESP32-C3, -C6 et -S3). Pour apprendre à les utiliser, il y a de nombreux tutos sur internet, notamment sur YouTube, et beaucoup sont en français.

Du fait de leur connectivité, de leurs hautes performances et de leur prix plutôt bas, les cartes ESP32 sont devenues rapidement aussi populaires que la carte Uno. Elles travaillent en 3.3 V, ce qui n'est qu'un détail, et on peut trouver sur le marché asiatique des kits de début à base de carte ESP32 comme on en trouve d'autres à base de carte Uno. C'est aussi une excellente façon de découvrir ces cartes ESP32 et de progresser avec elles.

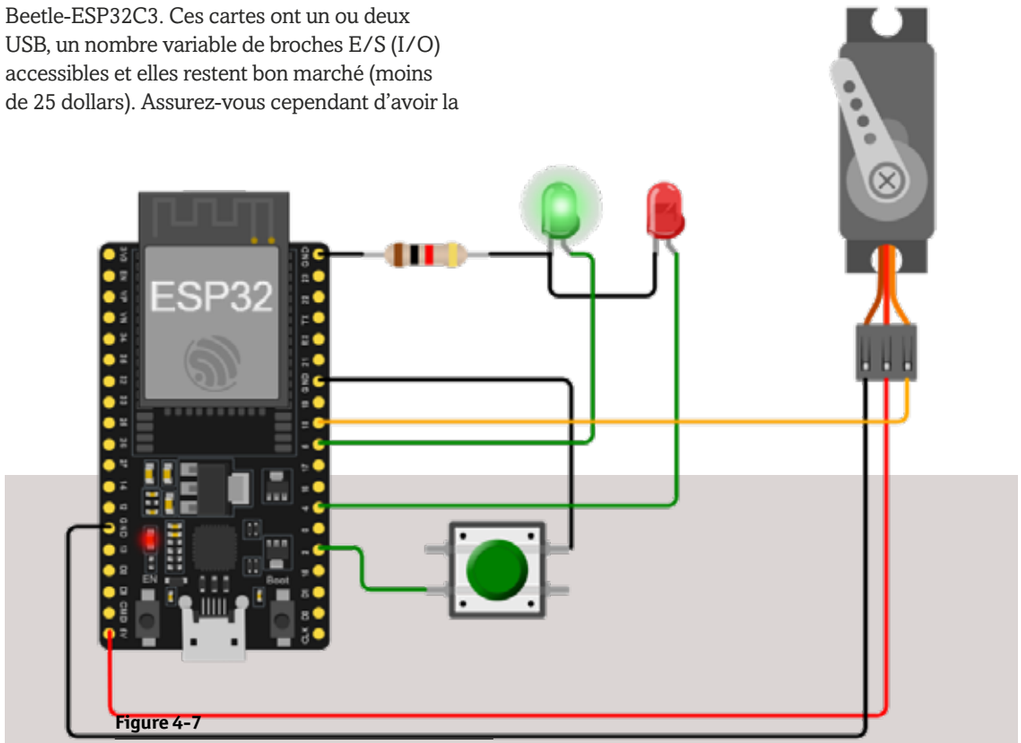


Figure 4-7

Attention : les broches de ces cartes ne sont pas tolérantes au 5 V (comme beaucoup de cartes qui sont en 3,3 V).

En modélisme ferroviaire, de nombreux montages à base d'ESP32 ont été publiés par le site LOCODUINO (pont automatisé Fleischmann, manette DCC, TCO Web interactif, LaBox, etc.) : il n'y a qu'à piocher pour avoir une bonne base pour développer ses propres projets. Et il n'est pas difficile de modifier un programme pour carte Arduino en programme pour carte ESP. Au chapitre 3, j'avais demandé à l'IA de réaliser une commande d'aiguille. J'ai repris son programme pour l'adapter à une carte ESP, comme le montre la **figure 4-7**. Un changement important est d'utiliser la bonne bibliothèque (ESP32Servo à la place de Servo) et il a fallu également modifier les broches utilisées.

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/428316358472666113** .

4.7 / LES AUTRES CARTES

Il existe de nombreux autres constructeurs de cartes à microcontrôleur. Les cartes sont architecturées autour d'un microcontrôleur 32 bits et elles ne coûtent pas forcément plus cher que les cartes à 8 bits. PJRC propose ses cartes Teensy qui ont besoin d'un add-on pour être programmées avec l'IDE d'Arduino. STM (STMicroelectronics) propose sa gamme de cartes Nucleo. Enfin, Raspberry connu pour son nano-ordinateur puissant et compact fonctionnant sous une distribution de Linux (le Raspberry Pi), a sorti une carte Raspberry Pi Pico architecturée autour d'un RP2040, qui se rapproche du format Arduino Nano, mais qui ne peut pas délivrer autant de courant qu'un ATmega328P ; il faudra donc veiller à ce point et utiliser des amplificateurs comme l'ULN2803A.

Il est bien évident que je ne peux pas les décrire en détail, mais je vous incite à les découvrir car leurs possibilités sont étonnantes.Wokwi propose les STM32 Nucleo-64 C031C6 et STM32 Nucleo-64 L031K6, ainsi que Raspberry Pi Pico et Raspberry Pi Pico W. En plus, changer de marque et de modèle de microcontrôleur ne peut que vous faire progresser dans la connaissance des ressources que ces cartes proposent (notamment en matière de WiFi ou Bluetooth).

RÉSUMÉ DU CHAPÎTRE 4

Les cartes à microcontrôleur évoluent sans cesse et nous proposent de nouvelles fonctionnalités dont certaines peuvent avoir leur utilité sur un réseau de trains miniatures. Même si notre hobby n'a pas besoin d'une énorme puissance de calcul, la possibilité de se connecter au réseau internet permet dans certains cas de simplifier le câblage entre composants. Ce chapitre a montré que certaines cartes sont à privilégier comme la Uno R4 d'Arduino ou bien l'ESP32 d'Espressif.

05 le réseau miniature

Le réseau miniature est constitué de différents éléments qu'il faut savoir commander dans un projet d'automatisation : aiguilles, cantons, signalisation lumineuse, passage à niveau, gare cachée, etc. Ce chapitre présente plusieurs solutions pour faire interagir les éléments entre eux afin de constituer un ensemble cohérent.

5.1 / ARCHITECTURE DES SOLUTIONS ÉLECTRONIQUES DANS UN RÉSEAU MINIATURE

Plusieurs solutions existent pour commander un réseau de trains miniatures avec une carte à microcontrôleur. Nous allons les passer en revue en insistant sur les avantages et les inconvénients.

Réseau miniature publié dans Loco-Revue (couverture de LR930)

RÉSEAU À CARTE UNIQUE DE GESTION

La première solution qui s'impose à l'esprit est d'avoir **une seule et unique carte pour gérer l'ensemble du réseau**. La **figure 5-1** montre ce genre d'architecture : on place des capteurs sur le réseau (détection d'occupation par exemple) et les signaux transmis sont mis en forme par une petite carte électronique (en vert clair) pour être compatibles avec Arduino. De la même façon, la carte Arduino envoie des signaux qui sont éventuellement amplifiés par une carte électronique (en jaune) pour les **actionneurs** (moteurs d'aiguilles, servomoteurs, etc.).

Si la carte s'occupe de tout, il lui faudra de nombreuses entrées-sorties pour être reliée aux différents capteurs et aux différents actionneurs (voir un peu plus loin). Il n'est pas forcément nécessaire d'avoir une carte qui travaille rapidement car nos modèles de trains ne se déplacent pas si vite que cela, en comparaison avec la vitesse de travail d'un microcontrôleur, même s'il n'a que 8 bits. **Une première limitation est donc le nombre d'entrées-sorties de la carte.**

La **figure 5-2** montre l'ensemble de tous les composants réunis autour d'une carte Arduino Mega 2560 pour le réseau **EX_MACHINA décrit dans Loco-Revue 936 de juillet 2025** (les poussoirs sont remplacés par des I.L.S sur le réseau). Comme on peut le voir, cette carte est déjà bien saturée et ce réseau est un petit réseau à deux aiguilles seulement. Si j'avais voulu mettre l'ensemble de la signalisation lumineuse, la carte n'aurait pas suffi et un choix sur les signaux a dû être fait. L'intégration des différentes fonctions a été possible car la carte Mega se contente de détecter les trains et de gérer le réseau comme un B.A.L, gérant à la fois la signalisation lumineuse et l'arrêt des trains devant un sémaphore. **Le programme a été écrit par une Intelligence Artificielle, ce qui a bien réduit le temps de développement.**

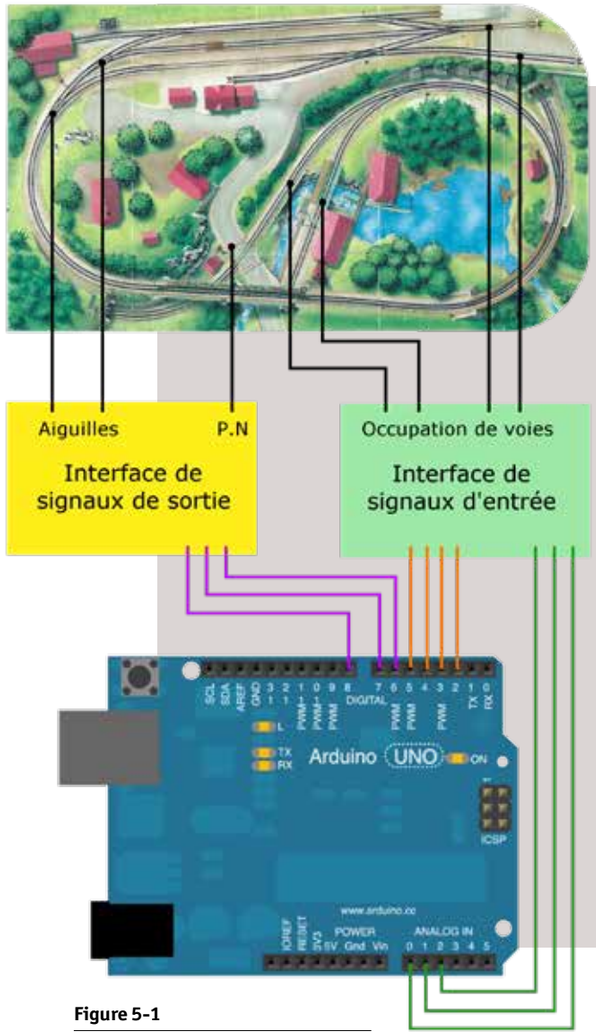


Figure 5-1

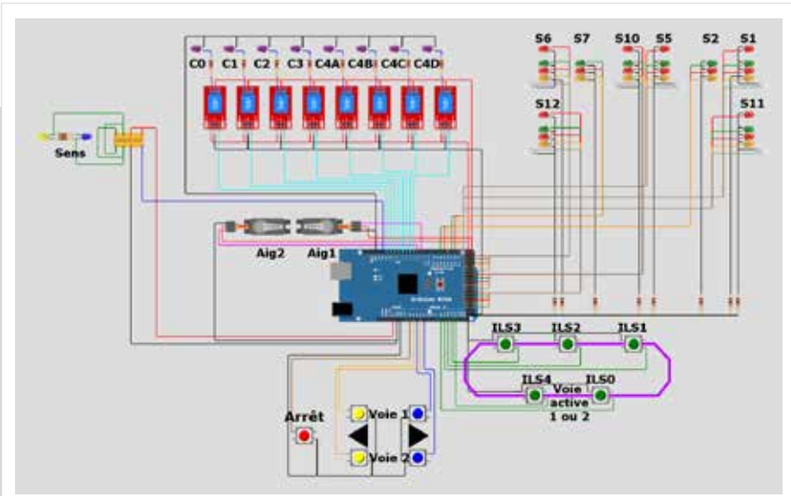


Figure 5-2

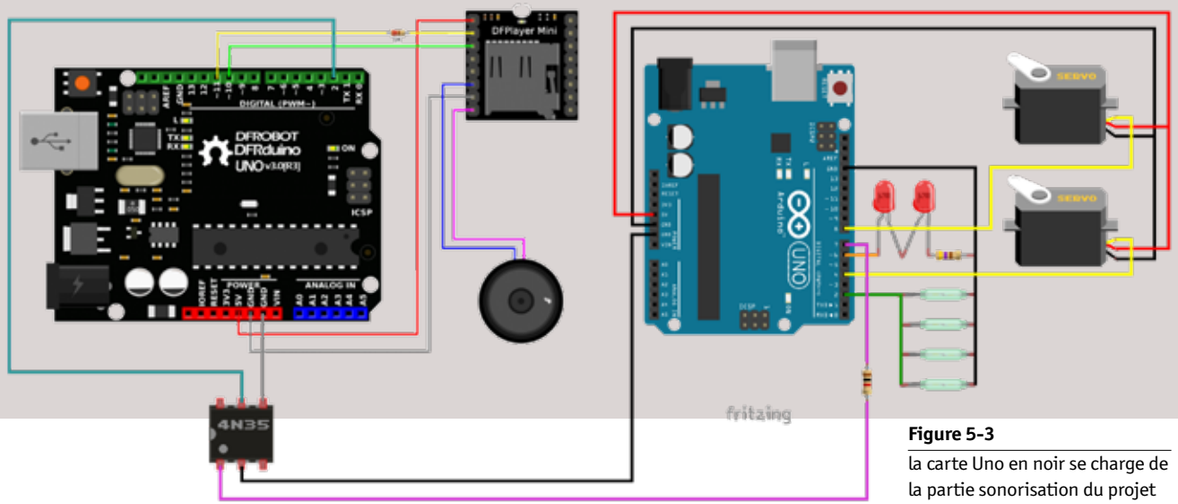


Figure 5-3
la carte Uno en noir se charge de la partie sonorisation du projet

Une deuxième limitation est la **difficulté à gérer du multitâches**. Ajouter une fonctionnalité dans un système déjà opérationnel, sans ajouter des erreurs qui peuvent se révéler critiques pour les autres fonctions, n'est pas toujours très simple. Par exemple, lorsque j'ai développé un PN (passage à niveau) pour le réseau Train In Box (**PN décrit dans Loco-Revue 894 de janvier 2022 et 895 de février 2022**),j'avais confié la partie sonorisation à une deuxième carte Uno (voir **https://locoduino.org/spip.php?article264**) car développer cela sur la carte qui gérait la détection des trains et le mouvement des barrières, créait des perturbations. J'aurais pu finir par trouver une solution mais certainement au prix de modifications drastiques de la partie non sonorisée du PN qui était parfaitement opérationnelle. Vu le prix d'une carte Uno, j'ai préféré en ajouter une deuxième qui ne fait que jouer le fichier sonore. La liaison entre les deux cartes est un simple fil : le niveau d'une sortie de la carte 1 commande, à travers un optocoupleur qui assure l'isolation galvanique, la carte 2 qui exécute le programme de lecture, comme le montre la **figure 5-3**.

Ceci nous amène à un autre concept : diviser pour mieux régner !

RÉSEAU À CARTES MULTIPLES SPÉCIALISÉES ET AUTONOMES

La **figure 5-4** montre un **réseau géré par plusieurs cartes Arduino** : chaque carte est spécialisée, la A et la B gèrent les aiguilles, la C ne gère que le passage à niveau et la D gère l'éclairage des bâtiments. Cette spécialisation fait que **les programmes n'interfèrent pas les uns avec les autres**, ce qui est plus simple à mettre au point. Le coût est bien évidemment plus élevé mais le temps gagné peut le justifier et **la maintenance du système est facilitée**. Il est parfois nécessaire que ce que fait une carte soit connu des autres cartes. Il faut alors les faire communiquer entre elles au moyen d'un bus de communication.

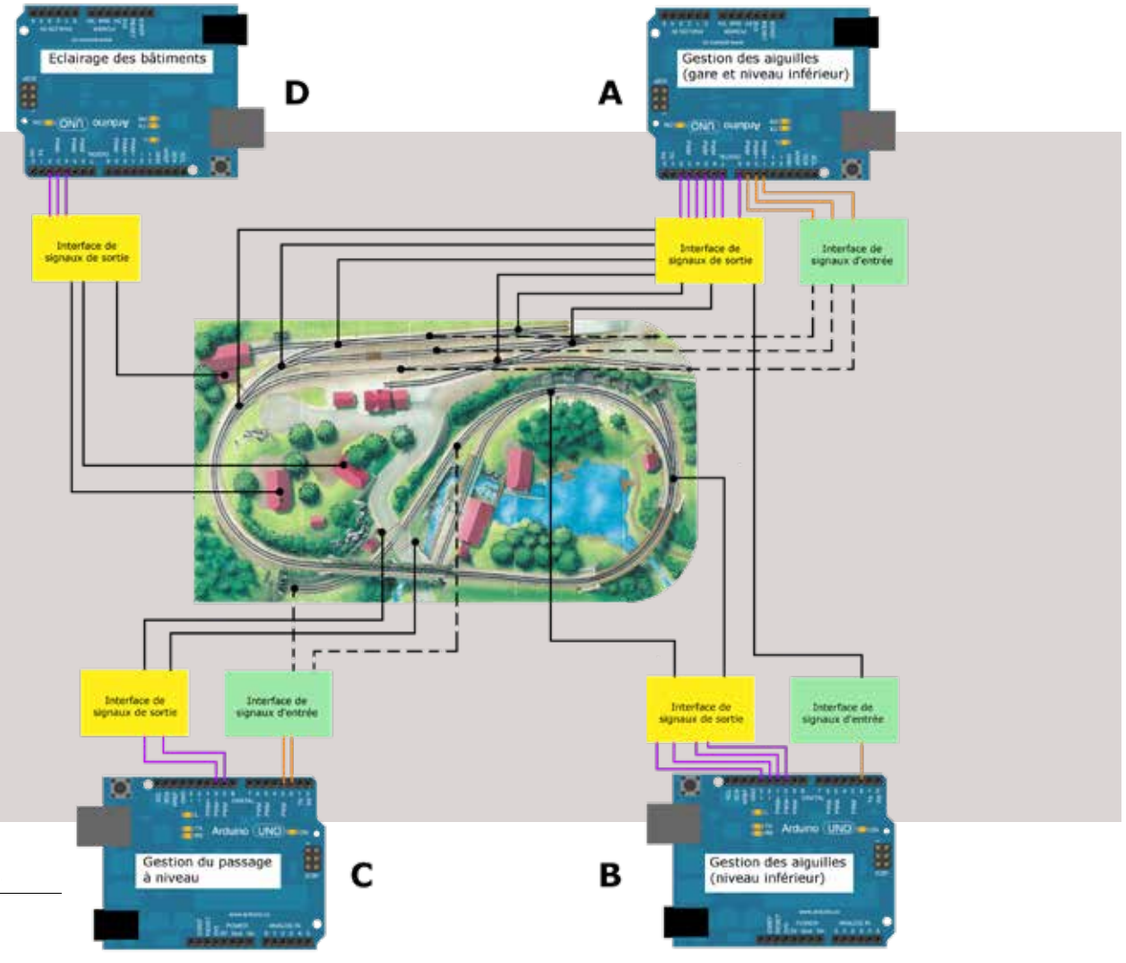


Figure 5-4

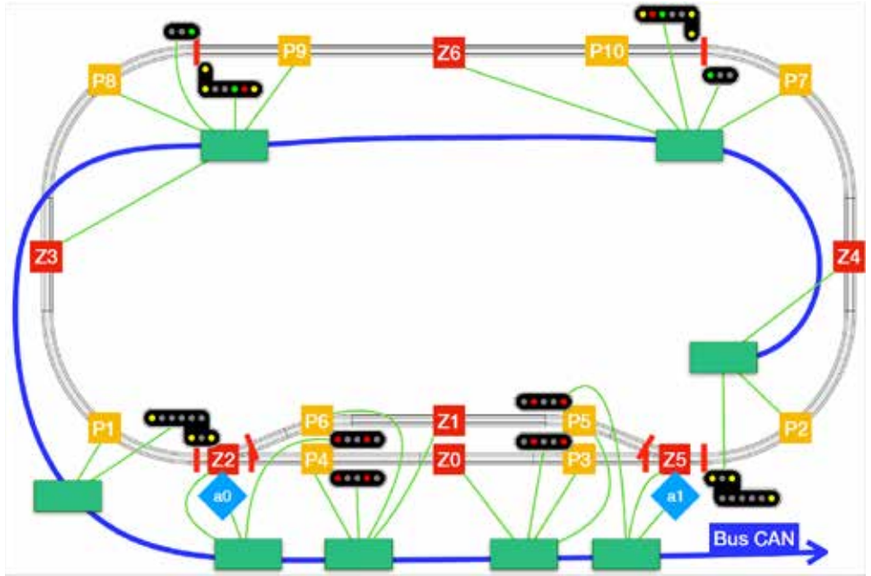


Figure 5-5
source LOCODUINO

RÉSEAU À CARTES MULTIPLES COMMUNIQUANT ENTRE ELLES

On peut imaginer que la gestion du réseau soit confiée à **plusieurs cartes à microcontrôleur communiquant entre elles**. Par exemple, une première carte peut détecter le train et communiquer à d'autres cartes sur quel canton il se trouve, quel est son itinéraire, s'il doit ou non s'arrêter en gare. Une deuxième carte, recevant les informations, peut alors se charger de la signalisation lumineuse, une troisième carte peut gérer les aiguilles pour former le bon itinéraire et une quatrième carte peut s'occuper du passage en gare et de l'éventuel arrêt.

Une telle architecture peut laisser penser qu'on est en train d'inventer une véritable usine à gaz, mais il n'en est rien. Au contraire, chaque carte est plus facile à mettre au point pour la fonction qui la concerne et **seule la communication reste le point principal à régler**. On peut aussi imaginer que les cartes ont toutes des fonctions identiques et sont identiques, ce qui fait descendre le coût de production, et qu'elles communiquent avec une carte gestionnaire : **ce principe a été introduit dans le modélisme ferroviaire en premier par LOCODUINO** qui a conçu des cartes satellites reliées à une carte gestionnaire du réseau. Chaque satellite pouvait assurer trois détections, permettre l'allumage de 9 LED et piloter un servomoteur. L'ensemble des satellites était relié **à un bus CAN (**voir plus loin) et recevait toutes les informations d'état du réseau et en tenait éventuellement compte pour agir sur le réseau. La **figure 5-5** montre ce concept. Les satellites sont les rectangles verts : ils sont tous reliés au bus CAN et sont connectés à des détecteurs de consommation de courant. Les signaux sont représentés, les détecteurs d'occupation de zone sont repérés par **Z**, les détecteurs de position par **P** et les servomoteurs d'aiguilles par **a**. Vous pouvez consulter l'ensemble de ce projet à cette adresse : **https://locoduino.org/spip.php?article242** .

DIFFÉRENTS BUS POSSIBLES POUR COMMUNIQUER

Plusieurs bus de communication font partie de l'écosystème Arduino et je vais décrire ceux qui sont les plus fréquemment utilisés pour communiquer avec un périphérique ou pour faire communiquer des cartes entre elles. Ce sont tous **des bus série**, ce qui veut dire que les bits qui constituent les octets des données à transmettre sont émis les uns après les autres, soit en commençant par le bit le moins significatif (bit 0), soit par le bit le plus significatif (bit 7).

SPI : l'acronyme signifie **Serial Peripheral Interface**. Ce bus est couramment utilisé pour interfacer des composants comme un **lecteur de tag RFID (Radio Frequency Identification)** ou bien des écrans LCD ou OLED. Il fait appel à la notion de contrôleur et de périphérique (anciennement appelée maître-esclave).

Le bus utilise quatre fils pour échanger les données entre contrôleur et périphérique :

- **Un fil SCK** avec les signaux d'horloge, permettant de synchroniser la transmission des bits
- **Un fil CIPO** (Controller In Peripheral Out) permettant l'échange dans le sens périphérique vers contrôleur (anciennement MISO master in slave out)
- **Un fil COPI** (Controller Out Peripheral In) permettant l'échange dans le sens contrôleur vers périphérique (anciennement MOSI master out slave in)
- **Un fil CS** (Chip Select) qui permet au contrôleur de sélectionner le périphérique avec lequel il veut communiquer (anciennement SS slave select)

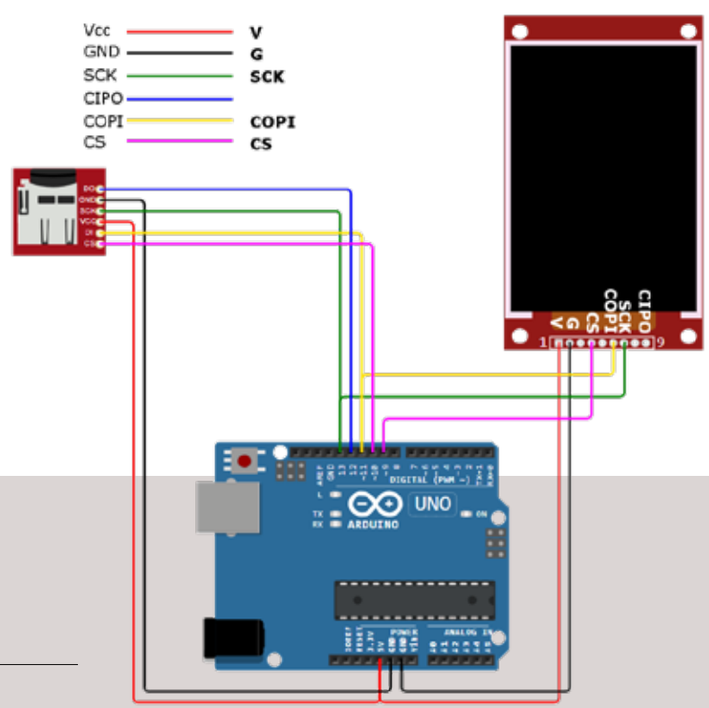


Figure 5-6

La **figure 5-6** montre une carte Arduino communiquant avec deux composants (un écran et un lecteur de carte SD) grâce au bus SPI pour lequel une bibliothèque existe. L'écran reçoit les données à afficher mais ne retourne rien : il n'y a donc pas de liaison CIPO.

Selon le mode de synchronisation entre horloge et donnée, le bus travaille selon quatre modes qui sont expliqués ici : **https://docs.arduino.cc/learn/communication/spi/** . Il faut aussi préciser la vitesse de transmission et si on commence par le bit le plus significatif (MSBFIRST) ou par le bit le moins significatif (LSBFIRST).

I2C : l'acronyme signifie **Inter-Integrated Circuits**. Le bus ne comporte que deux fils (plus le fil de masse) :

- **Le fil SCL** qui véhicule les signaux d'horloge
- **Le fil SDA** qui véhicule les données

La **figure 5-7** montre comment deux cartes peuvent être reliées en I2C pour échanger des données. Chaque périphérique a sa propre adresse (définie par construction) et peut ainsi recevoir la transmission qui le concerne. La bibliothèque Wire permet de définir soi-même ces adresses dans le cas de deux cartes.

L'utilisation du bus I2C est décrite ici : **https://docs.arduino.cc/learn/communication/wire/** Des exemples sont donnés dont on pourra s'inspirer pour un projet de modélisme ferroviaire.

CAN : l'acronymesignifie Controller Area Network. Simple, robuste et peu coûteux, le bus CAN est la solution idéale pour relier plusieurs composants sur un même réseau, et **LOCODUINO a été le premier à l'utiliser dans le cadre du modélisme ferroviaire**. C'est un bus différentiel, peu sensible aux parasites, constitué d'une simple paire de câbles torsadés CANHigh et CANLow. On trouve, pour un prix dérisoire, des petites cartes électroniques permettant d'interfacer des anciennes cartes Arduino (Uno, Nano, Mega) avec un bus CAN ; les nouvelles cartes sont souvent pourvues d'un contrôleur CAN mais elles nécessitent tout de même un transceiver (composant spécialisé). Il existe aujourd'hui plusieurs bibliothèques pour communiquer et mettre en œuvre un bus CAN.

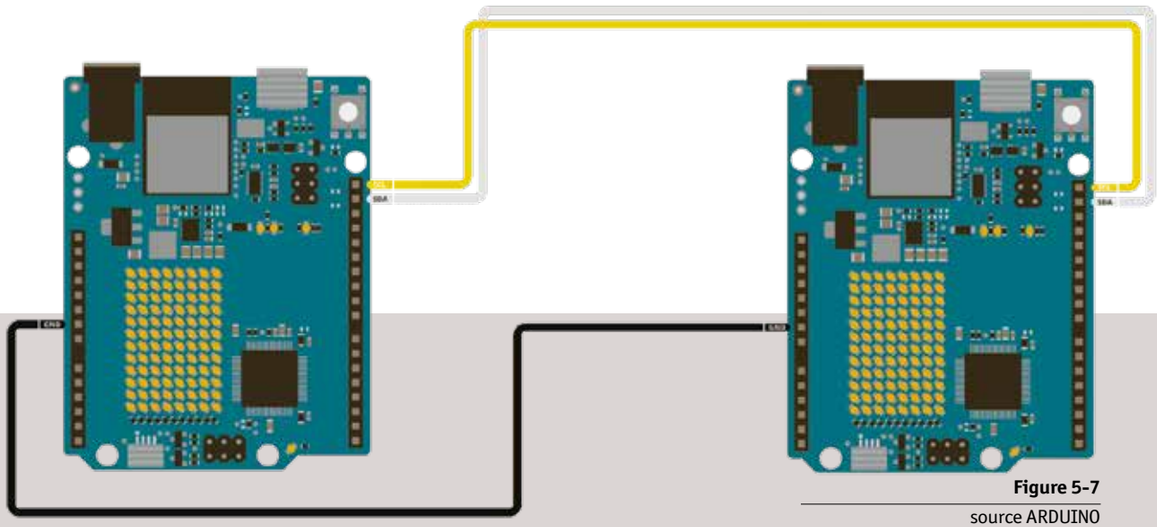


Figure 5-7
source ARDUINO

Cet article **https://locoduino.org/spip.php?article268** est parfait pour démarrer dans ce domaine, mais si vous voulez maîtriser ce bus, le livre **« CAN et CAN FD » de Pierre Molinaro**, chez Elektor, est vraiment la bible du CAN.

La **figure 5-8** montre comment relier deux cartes Uno en bus CAN : la carte de droite pourrait gérer un passage à niveau tandis que la carte de gauche s'occuperait de surveiller l'arrivée des trains en remplaçant les poussoirs par des I.L.S sur le réseau.

Ou bien encore, la carte de gauche pourrait être un TCO électronique envoyant ses ordres vers la carte de droite qui gèrerait les servomoteurs d'aiguilles et la signalisation lumineuse.

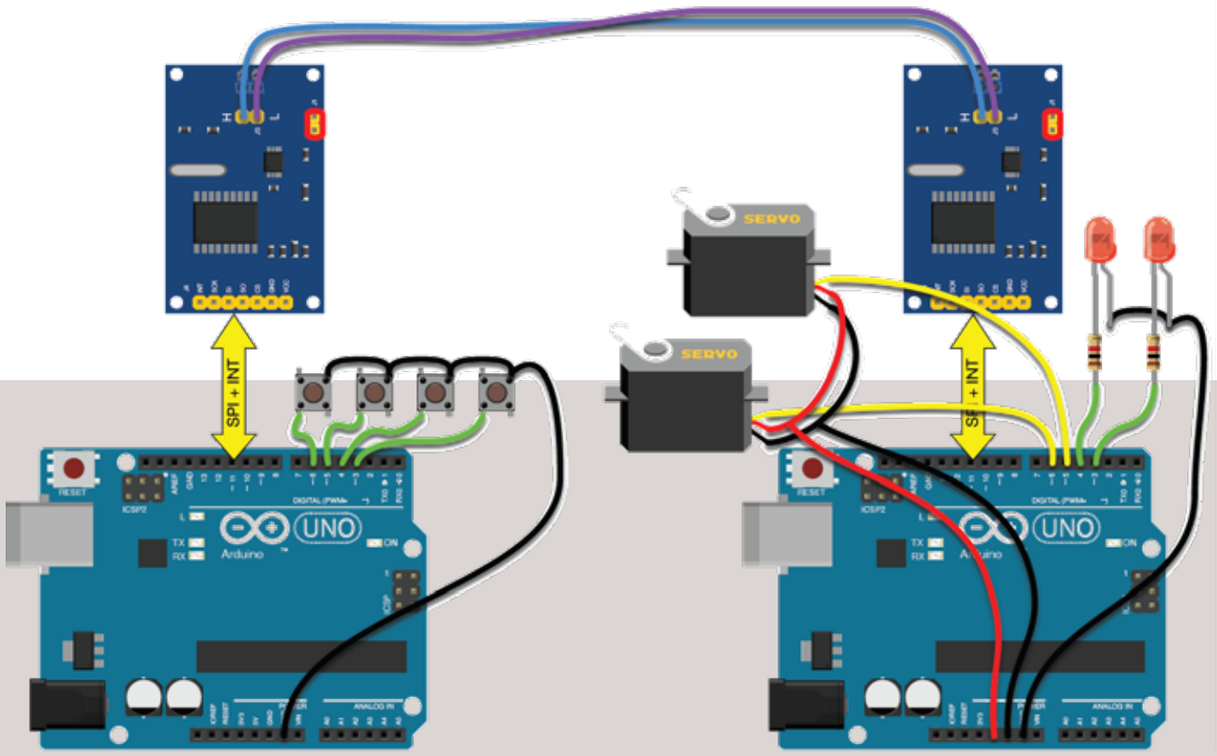


Figure 5-8
source LOCODUINO

5.2 / LES ACTIONNEURS DU MODÉLISME FERROVIAIRE

Dans ce paragraphe, je ne traiterai que des capteurs installés sur un réseau miniature. Les capteurs servent essentiellement à localiser les trains, mais une simple photorésistance peut aussi allumer l'éclairage du réseau lorsque la nuit commence à tomber. **Pour repérer les trains, nous avons deux solutions** :

- Le détecteur d'occupation
- Le détecteur de passage

Le **détecteur d'occupation** permet de savoir qu'un train occupe une portion de voie, sans savoir exactement où il se trouve sur cette voie. Il fournit un signal pendant une **durée relativement longue**, signal qu'il faut rendre compatible avec la tension possible sur l'entrée qui l'analyse (3,3 V ou 5 V). Comme le signal dure un certain temps (tant que le train se déplace sur la portion de voie), la carte ne peut pas le louper et le programme qui gère ce capteur est plus facile à écrire. Généralement, ce type de détecteur utilise la consommation de courant par la locomotive ou les wagons (**figure 5-9**) : **son comportement est donc meilleur en numérique car la voie reste alimentée en permanence**. En analogique, ce détecteur ne peut plus rien détecter lorsque la portion de voie n'est pas alimentée avec une locomotive à l'arrêt. Il faut faire appel à un autre système, généralement basé sur l'**injection sur la voie d'une tension ne servant qu'à la détection** et pas à la traction. Les détecteurs qu'on trouve dans le commerce sont assez onéreux, de l'ordre de 10 à 15 euros par zone où la détection doit se faire.

Le **détecteur de passage** permet de savoir qu'un train est à un endroit précis du réseau. Il peut être basé sur la détection d'un **champ magnétique** (I.L.S interrupteur à lames souples ou bien capteur à effet Hall), ou bien sur la détection ou l'interruption d'un **faisceau lumineux infra-rouge** (IR), ou encore sur un **lecteur RFID** (identification par radio fréquence). Si c'est un champ magnétique qui déclenche le détecteur, alors les locomotives doivent être équipées d'un aimant mais dans ce cas, seule la locomotive est détectée et par les wagons. Pour éviter cela, on peut aussi mettre un aimant sur chaque wagon : l'I.L.S ou le capteur Hall réagit alors à chaque véhicule, et locomotive et wagons peuvent être ainsi **comptés à l'entrée d'une zone et décomptés à la sortie**. Un détecteur d'entrée de zone peut aussi être considéré comme détecteur de sortie de la zone précédente car les zones se suivent (cas des cantons par exemple). En général, on se contente d'un aimant sur le wagon de queue, ce qui permet de contrôler qu'il n'y a pas de rupture d'attelage (voir chapitre 7).

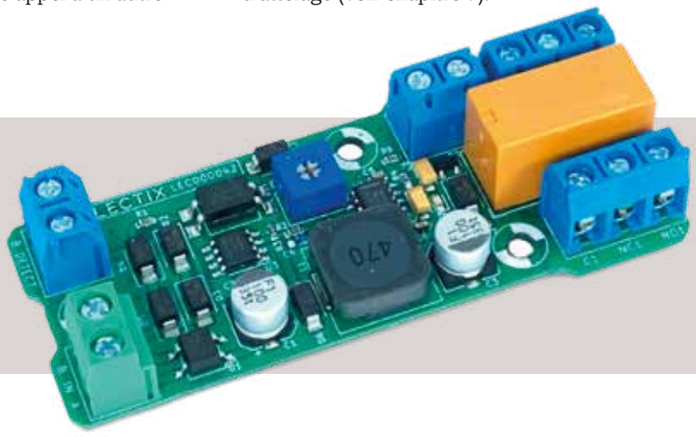


Figure 5-9

Ce genre de détecteur de passage est **facile à ajouter sur un réseau déjà terminé**, ce qui ne serait pas le cas des détecteurs d'occupation qui nécessitent un tronçonnage d'un rail et sa réalimentation en courant. L'I.L.S est le plus simple mais il faut avoir en tête que c'est avant tout un interrupteur mécanique et qu'il **produit des rebonds** que le logiciel devra savoir traiter. Le capteur à effet Hall n'a pas ce problème, mais il doit être alimenté en courant (généralement 5 V) ce qui fait un fil de plus. Le signal délivré par l'I.L.S ou le capteur Hall est **fugace** (le temps que l'aimant le survole) et le programme doit donc **surveiller le capteur en permanence** pour ne pas louper d'événements. Il est donc plus difficile à écrire et il faut souvent utiliser les interruptions externes du microcontrôleur.

Le **capteur RFID** est un capteur qui non seulement détecte le passage d'un train, mais **en plus peut l'identifier grâce à son tag** ; la technique a été décrite dans le tome 1 et je n'y reviendrai pas. L'identification par le programme permet de décider du trajet du train, de la voie qu'il doit emprunter, s'il doit s'arrêter en gare on non, s'il doit klaxonner, etc. La **figure 5-10** montre deux lecteurs de badge RFID qui peuvent se glisser sous une voie et le tag qui peut se coller sous la locomotive.

En conclusion, les capteurs sur un réseau vont **envoyer une information à la carte Arduino** et cette information doit être traitée avant d'être utilisée. Par exemple, s'il ne faut pas louper de survols d'un I.L.S, il ne faut pas non plus en compter plusieurs à cause des rebonds.

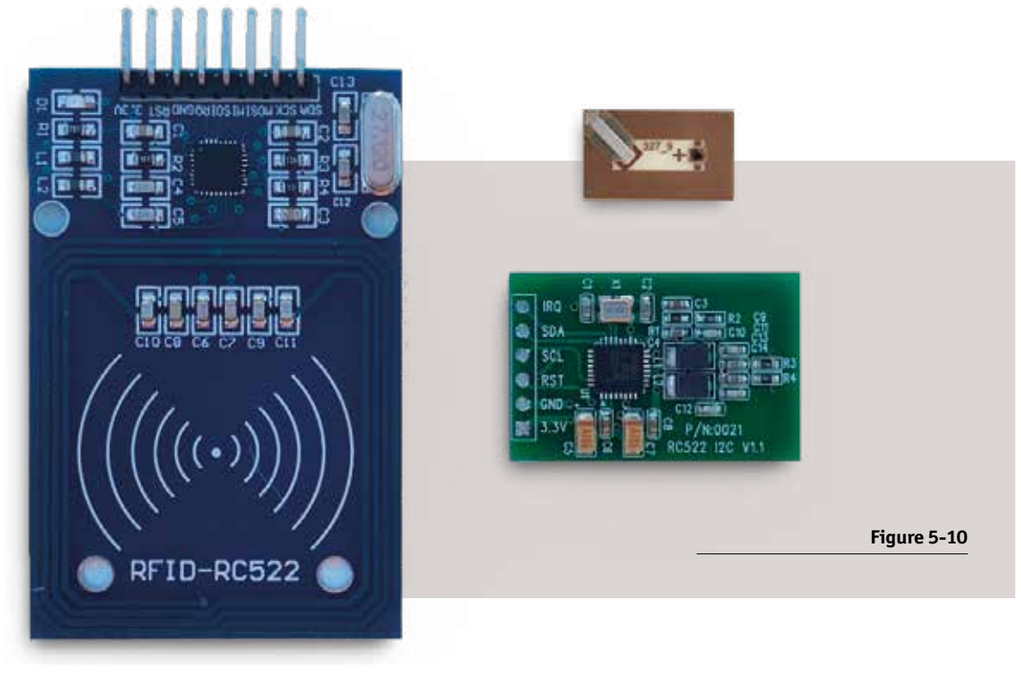


Figure 5-10

LES ACTIONNEURS DU MODÉLISME FERROVIAIRE

Dans ce paragraphe, je ne traiterai que des actionneurs installés sur un réseau miniature. En fonction de son programme, la carte Arduino **envoie des signaux vers des actionneurs pour produire un certain effet**, signaux qu'il faut parfois amplifier. Une simple **LED** est déjà un actionneur, par exemple celle qui équipe les feux de PN et c'est bien la carte qui la fait clignoter quand un train approche. Un **relais** alimentant une portion de voie est aussi un actionneur. Les **moteurs d'aiguilles** sont également des actionneurs, qu'ils utilisent un **servomoteur** ou bien un **moteur lent**, mais autant un seul servomoteur peut utiliser le 5 V fourni par la carte, autant le moteur lent nécessite sa propre alimentation.

Les moteurs électriques, qu'ils soient à **courant continu** (DC) ou bien **pas à pas**, sont également des actionneurs très prisés sur un réseau. Les premiers nécessitent une interface qui permettra de déterminer sens et vitesse de rotation. Certains circuits intégrés sont prévus pour cela mais on trouve aussi des cartes qui font le travail. Ces mêmes cartes peuvent aussi commander un moteur pas à pas bipolaire, alors que les moteurs pas à pas unipolaires nécessitent des cartes à base de circuit intégré matrice de transistor comme l'ULN2003 ou l'ULN2803. Tous ces moteurs permettent de **mouvoir les éléments de décor** du réseau comme un rotor d'hélicoptère, une roue de moulin, une grande roue de fête foraine, etc.

Enfin, les **écrans** peuvent servir à reproduire les espaces publicitaires d'un quai de gare de notre époque moderne ou bien les affichages municipaux d'un village, ce qui donne de très jolies animations. La mise en œuvre de ces actionneurs a été décrite dans le tome 1.

En conclusion, les actionneurs sur un réseau vont **recevoir des informations de la part de la carte Arduino**, les signaux étant le plus souvent amplifiés ou bien confiés à un bus de communication (voir plus haut).

LES CIRCUITS INTÉGRÉS INDISPENSABLES EN MODÉLISME FERROVIAIRE

Il existe deux sortes de **circuits intégrés (CI)**, les « traversant » (DIP pour **Dual Inline Package**) qui ont des broches qui traversent le circuit imprimé et se soudent côté verso, et les « CMS » qui sont des Composants Montés en Surface (SMD en anglais, pour **Surface MountedDevice**) dont les broches sont soudées côté recto du circuit imprimé. Les CMS sont beaucoup plus petits et donc plus difficiles à souder, même si on y arrive avec un peu de soin et d'entraînement. Lorsqu'on fait réaliser un circuit imprimé sur mesure, on peut demander à ce que certains CMS soient soudés dessus lors de la phase de fabrication. Aussi, les CI que vous devez avoir en stock sont plutôt des CI DIP qui peuvent s'enficher sur une breadboard (voir tome 1) afin de fabriquer un prototype du montage pour l'expérimenter. La **figure 5-11** montre la différence entre DIP et SMD : pour les DIP, il est toujours possible d'utiliser des supports de CI.

Je vais donc parler des **CI vraiment indispensables**, ceux qu'on trouve fréquemment sur les montages à carte Arduino. Je ne parlerai pas des composants discrets comme les diodes ou les transistors. Bien évidemment, la liste que je vous propose n'est pas exhaustive.

Les **optocoupleurs** sont fréquemment utilisés pour assurer l'isolation galvanique entre carte et périphérique dont les tensions d'alimentation sont différentes (3,3 V et 5 V par exemple ou bien 5 V et 12 V d'un réseau).

Les **matrices de transistors** comme l'ULN2003 ou l'ULN2803 sont utilisées pour amplifier les signaux de sorties et aussi servir d'inverseur de signal, ce qui peut permettre de brancher des signaux lumineux à anodes communes sur un montage réalisé pour des signaux à cathodes communes. Ces CI sont aussi utilisés comme interface pour moteur pas à pas unipolaire.

Les **CI pont en H** comme le L293D ou le L298 sont utilisés pour régler sens et vitesse de rotation de deux moteurs à courant continu ou bien encore comme interface pour un moteur pas à pas bipolaire.

Le **CI 74HC495** est un registre à décalage qui permet de commander 8 LED avec seulement trois sorties d'Arduino ; on l'utilise beaucoup pour la signalisation lumineuse du réseau quand le nombre de sorties de la carte se révèle insuffisant.

On peut ajouter à cette gamme différentes **portes logiques (soit NOR soit NAND)** ainsi qu'un **amplificateur opérationnel** comme le 741. Si vous avez besoin de bricoler une tension régulée, la gamme des **CI 78XX** (XX étant la tension positive désirée) et **79XX** (XX étant la tension négative désirée) est très pratique. Enfin, le **CI NE555** est considéré par les électroniciens comme le couteau suisse pour réaliser des montages, mais en électronique programmable, vous ne l'utiliserez pas souvent car le microcontrôleur peut reconstituer tout ce qu'il fait.

Ces différents CI étant fabriqués en grande série, ils sont bon marché et en avoir quelques-uns d'avance ne coûte pas une fortune et peut rendre service. Votre stock sera complété par des LED de différentes couleurs et une boîte de résistances de différentes valeurs, des diodes, des transistors (ceux à effet de champ sont plus souvent utilisés avec les cartes Arduino), des I.L.S, des poussoirs, des interrupteurs et quelques potentiomètres. Tout ce qui a un intérêt pour les montages Arduino se trouve **dans les kits d'initiation** qui sont souvent moins chers que les composants achetés séparément ; c'est pourquoi je ne fournis aucune valeur ou référence pour ces composants discrets.

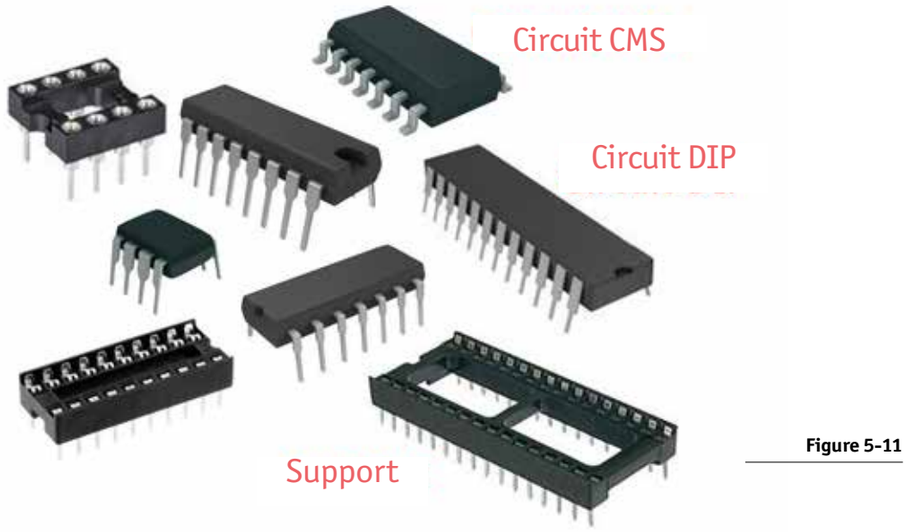


Figure 5-11

Utilisation d'écrans à Miniatur Wunderland à Hambourg, devant une foule enthousiaste

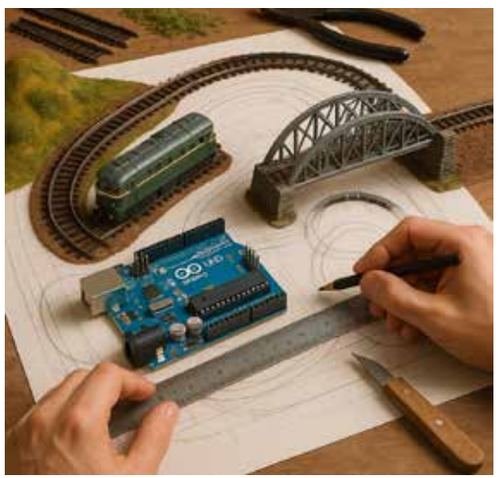
RÉSUMÉ DU CHAPÎTRE 5

Si vous avez acheté ce guide pour automatiser votre réseau, alors ce chapitre est important pour faire les bons choix lors de la conception d'un projet. Ce chapitre a rappelé qu'il y a plusieurs solutions d'architecture avec chacune des avantages mais aussi des inconvénients. Il faut donc bien réfléchir à toutes les possibilités car revenir en arrière plus tard est extrêmement difficile. Ce chapitre a aussi montré qu'on peut trouver dans le commerce de nombreuses cartes électroniques qu'il n'y a qu'à assembler entre elles pour réaliser les interfaces électroniques dont on a besoin.

06 Concevoir son projet

Dans ce chapitre, on va analyser comment concevoir un projet de modélisme ferroviaire et on prendra comme exemple le projet de PN (*passage à niveau*) qui a été imaginé pour le réseau Train' In Box (*TIB*), projet qui a été publié dans Loco-Revue (*janvier et février 2022*) en collaboration avec le site LOCODUINO (figure 6-1).

La première question qu'il faut se poser est **que veut-on vraiment faire ?** Il faut définir un **cahier des charges**, puis penser aux **différentes étapes** qu'il faut réussir avant de passer aux étapes suivantes. Car un projet quel qu'il soit doit se construire par étapes, en **commençant toujours par quelque chose de simple qu'on va compliquer progressivement**. Vouloir tout faire d'un coup est voué à l'échec.



Concevoir son projet (image réalisée par une IA)

6.1 / LE BUT À ATTEINDRE : CONCEPTION D'UN CAHIER DES CHARGES

Le but, on le connait : reproduire exactement un PN automatisé de type SAL 2 (deux barrières). La première chose à faire est de chercher sur YouTube des vidéos de ce type de PN. Cela permet de définir les différentes phases de fonctionnement :

1. Détection d'un train en approche
2. Sonnerie et clignotement des feux routiers
3. Attendre un certain délai (de l'ordre de 8 secondes)
4. Abaissement et fermeture des barrières
5. Arrêt de la sonnerie alors que les feux continuent à clignoter
6. Train qui passe, une fois passé :
7. Arrêt du clignotement et ouverture des barrières simultanément

Figure 6-1
Réseau Train'in Box de l'auteur

Le **point 1** peut se faire soit avec des détecteurs d'occupation par consommation de courant, soit avec des capteurs ponctuels comme des I.L.S. Mais le PN doit pouvoir être installé sur un réseau déjà fini ; la solution I.L.S est préférable car simple à ajouter.

Le **point 2** est facile si on emploie un mini-player de fichier MP3 ; de plus, il existe une bibliothèque publiée par LOCODUINO pour le clignotement des feux en simulant l'inertie thermique du filament.

Le **point 3** n'est pas évident sur un petit réseau comme TIB, il faut donc pouvoir régler ce délai à 0 seconde sur TIB, tout en gardant la possibilité de le régler à 8 secondes sur un réseau plus grand. Les paramètres doivent être réglables pour que le PN soit adapté au réseau possédé par le modéliste.

Le **point 4** fait appel à des servomoteurs synchronisés dont le mouvement dépend de la géométrie de la barrière.

Les **points 5, 6 et 7** ne présentent pas de difficultés particulières. Il faut ajouter à cela d'autres contraintes : le PN doit pouvoir être construit sans machines-outils spécialisées (tour, fraiseuse, imprimante 3D) et avoir l'électronique la plus simple possible, son prix de revient doit être inférieur à ce qu'on peut trouver d'équivalent dans le commerce. De plus, le PN doit pouvoir être ajouté à un réseau déjà construit sans avoir à réaliser d'importants travaux comme tronçonner des rails et réalimenter des portions de voies.
Tout cela forme donc un cahier des charges à respecter.

6.2 / LES ÉTAPES SUCCESSIVES

Comme je l'ai déjà dit, un projet important ne peut **se construire que par étapes successives**. Pour ce PN, les différentes étapes à réussir sont les suivantes :

1. Détecter les trains à l'approche du PN
2. Faire clignoter les feux de PN
3. Faire manœuvrer les barrières
4. Ajouter le son

Pour l'**étape 1**, on peut définir une zone autour du PN pouvant être constituée de plusieurs voies avec des I.L.S aux frontières. Comme il ne faut pas louper de déclenchement d'I.L.S, on opte pour une **interruption externe**.

Pour l'**étape 2**, on peut utiliser **la bibliothèque LightDimmer de LOCODUINO** publiée quelques années auparavant. Elle a pour avantage de reproduire l'inertie thermique des lampes à filament.

L'**étape 3** est la plus difficile : la barrière utilisée (Auhagen réf41625) nécessite **un déplacement de la tige de commande en plastique de 2,5 mm** ! Le problème devient donc purement mécanique. Pour respecter cette cote, une came excentrique tournant de 90° est la meilleure solution, mais la tige étant flexible et il est nécessaire de la renforcer (**figure 6-2**).

L'**étape 4** a déjà été réalisée par LOCODUINO pour un projet d'annonces en gare ; cependant, à l'époque de la conception de ce PN, je n'ai pas réussi à intégrer cette fonction dans ce que j'avais déjà produit et je suis parti sur l'idée d'une **deuxième carte Arduino gérant uniquement la partie son**.

Figure 6-2

la came (à gauche) et le support du servomoteur (à droite)

Aucune étape ne doit être entreprise sans que les étapes précédentes aient donné satisfaction. Finalement, l'intégration du PN sur le réseau TIB peut se faire sans trop de problèmes, le circuit étant divisé en deux parts pratiquement égales, dont une est la zone du PN (les trains doivent être détectés assez loin du PN). Pour TIB, il n'y a pas de délai d'attente entre sonnerie et fermeture des barrières (le réseau étant trop petit) mais ce délai peut être réglé si on dispose d'un réseau plus grand.

Figure 6-3

Avec toutes ces contraintes, des essais d'endurance ont été réalisés et l**e train a parcouru plus de 100 fois le tour du réseau TIB** sans qu'il n'y ait le moindre dysfonctionnement (**figure 6-3**).

Des améliorations sont possibles, notamment réaliser un PN pour deux voies indépendantes l'une de l'autre. Sur le réseau Train' In Box, un seul train circule à la fois et les deux voies peuvent être traitées comme une seule. Ce qu'on sait faire pour une voie peut être recopié pour une deuxième voie. Il suffit ensuite d'un peu de logique pour déterminer l'état d'occupation de la zone PN. La zone PN est libre si la voie 1 ET la voie 2 sont libres. La zone PN est occupée dès que la voie 1 OU la voie 2 est occupée. Une fois l'état de la zone PN connu, le traitement est le même que pour le réseau TIB.

Le lien ci-dessous vous permet de voir le fonctionnement du PN avec deux voies : **https://wokwi.com/projects/428656938257258497**

Remarquez que **les voies 1 et 2 peuvent être utilisées dans les deux sens et que le refoulement est pris en compte**. Si on appuie deux fois sur un poussoir en moins de 2 secondes, seul le premier appui est pris en compte car on estime que c'est le même survol d'I.L.S. Le délai entre clignotement et fermeture des barrières est de 4 secondes, mais on peut le modifier en ligne 20 du programme (selon la longueur du réseau). En montage réel, on mettrait deux LED rouges en série avec une résistance de 470 ohms. Il ne reste plus qu'à ajouter la partie sonorisation, ce qui est très facile vu qu'on a fait le choix de la réaliser avec une carte à part.

6.3 / UTILISATION DES SIMULATEURS D'ARDUINO

Les simulateurs d'Arduino comme **Tinkercad** ou **Wokwi** permettent de gagner du temps pour la mise au point de programme : on économise le temps de téléchargement de la carte et on peut commencer la mise au point avant même d'avoir construit le prototype. J'ai utilisé Wokwi pour **tester les programmes délivrés par l'IA ChatGPT dans mon projet de réseau EX_MACHINA**. Je voyais en temps réel si le programme fonctionnait ou bien devait être corrigé. La plupart des fonctions ont marché du premier coup (voir le chapitre 3), **preuve que l'IA peut rendre bien des services**. Mais même si on développe sans elle, l'utilisation des simulateurs, malgré leurs limites, est fortement recommandée. Le projet de pont tournant du chapitre 8 a été mis au point en partie (mais pas totalement) sur le simulateur **Wokwi** pour la partie commande du pont et contrôle de la position sur un écran I2C. Même les relais ont été simulés par de simples LED pour tester leur mise en route. On comprend alors **qu'il ne faut pas s'arrêter au fait que tous les composants ne sont pas disponibles** : on peut les remplacer par d'autres ou les simuler par des LED. Les simulateurs **permettent aussi de partager son travail avec d'autres personnes** et de travailler en équipe (voir chapitre 1). Après la mise au point sur simulateur, on peut entreprendre la construction du prototype, faire les essais in-situ et faire également des **tests d'endurance**, comme cela a été fait pour le PN.

6.4 / UTILISATION DE L'INTELLIGENCE ARTIFICIELLE

L'intelligence artificielle permet aussi de gagner du temps dans le développement d'un projet, mais à condition qu'elle soit bien utilisée. L'utilisation de l'IA a été évoquée dans le chapitre 3. Il faut la considérer comme un outil et à ce titre, elle peut écrire du code à notre place à condition de bien lui expliquer toutes les contraintes à respecter et le but exacte à atteindre. Mais l'IA ne peut pas tout faire ou tout décider et votre rôle est de rester chef d'orchestre, seul et unique décideur du projet que vous voulez réaliser. Certains projets peuvent avoir recours à plusieurs langages (C, C++, Processing, HTML) : imaginez le temps que vous perdriez à les apprendre alors que l'IA les connaît. Il faut tout de même se méfier car l'IA peut aussi ne pas connaître certaines bibliothèques qui sont peu connues (par exemple, celle que j'aiécrite et qui n'est pas suffisamment utilisée dans le monde pour faire partie des connaissances de l'IA).

6.5 / TRAVAILLER EN ÉQUIPE

Travailler en équipe permet d'être plus créatif et, une fois de plus, de gagner du temps. Par exemple, vous pouvez travailler sur une centrale DCC parce que vous en avez déjà fabriqué une, pendant qu'un ami travaillera sur l'intégration d'un bus CAN sur cette centrale. Un autre, plus féru d'électronique, peut se charger de la conception du circuit imprimé et de choisir les composants. L'utilisation de la plateforme GitHub (voir chapitre 1) permet justement le partage des résultats et devient l'outil idéal si vous ne résidez pas dans la même région. Comme je l'ai dit plus haut, la rédaction d'un cahier des charges, et d'un plan de progression si possible associé à un calendrier, permet de planifier la réalisation du projet, notamment si le résultat doit être montré en exposition dont la date est déjà fixée. Il convient de **définir les rôles de chaque membre de l'équipe en fonction des compétences et de la disponibilité**. Il est prudent de compter large pour le calendrier car certains problèmes peuvent retarder les choses. Enfin, une équipe de travail dans un loisir n'a rien à voir avec une équipe dans une entreprise ; les gens sont là pour se faire plaisir et personne ne peut les forcer à agir. Il est donc indispensable de bien choisir ses partenaires et de bien connaître les motivations de chacun.

6.6 / RÉALISATION DES CARTES ÉLECTRONIQUES ET DES INTERFACES

Les cartes à microcontrôleur ont **besoin d'un peu d'électronique pour bien fonctionner**, par exemple pour assurer l'isolation galvanique de tensions très différentes, pour adapter les signaux aux tensions des périphériques, pour amplifier des signaux de sortie, pour filtrer des signaux inutiles, etc. Il y a deux solutions pour produire cette électronique : chercher ce qui existe déjà ou bien concevoir sa propre carte.

SOLUTION TOUTE FAITE EXISTANT SUR LE MARCHÉ

Il suffit de **chercher sur les sites de vente par correspondance** pour trouver ce qu'on veut et à des prix souvent modiques, et on aurait tort de se priver. Inutile donc de vouloir développer sa carte relais puisqu'on en trouve à des prix défiant toute concurrence, inutile également de développer sa carte moteur car il en existe quasiment dans toutes les marques. On trouve aussi des cartes permettant d'adapter les signaux aux tensions requises (3,3 ou 5 V) et des cartes qui amplifient les signaux de sorties des cartes Arduino. Profitons de la pléthore des offres, la difficulté étant de trouver vraiment ce qu'on cherche (dans certains cas, ce n'est pas si facile et il faudra vous renseigner auprès d'autres amateurs qui ont peut-être eu le même problème avant vous). La **figure 6-4** montre deux cartes du commerce, celle de gauche est une carte à optocoupleurs de 4 canaux, celle de droite est une carte bus CAN.

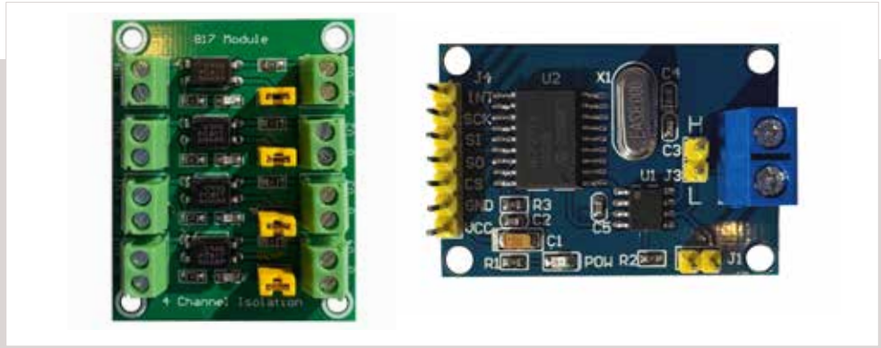


Figure 6-4

CONCEPTION AB-NIHILO

Figure 6-5

Si la carte désirée ne se trouve pas, il reste la solution de la **développer vous-même et en faire profiter ensuite la communauté**. Pour avoir une carte de qualité professionnelle, il faut concevoir un circuit imprimé et le faire réaliser par une entreprise spécialisée (voir le chapitre 1). Souvent la carte Arduino vient s'enficher sur la carte électronique (solution pour la série Nano) mais on peut aussi développer la carte électronique au format des cartes shield et elle vient alors s'enficher sur la carte Arduino. Dans certains cas, il peut être préférable de développer une carte à part qui sera connectée à la carte Arduino par des **câbles en nappes**, comme c'était le cas pour les ordinateurs avec leurs disques durs avant l'arrivée de la norme SATA (transmission série). La **figure 6-5** montre une carte double face développée pour des animations lumineuses à base d'ATtiny45.

RÉSUMÉ DU CHAPÎTRE 6

Ce chapitre est le complément du chapitre précédent. Un projet d'automatisation est bien souvent une tâche difficile qu'il faut savoir diviser en un ensemble de tâches élémentaires plus simples. En effet, il faut toujours partir de quelque chose de simple et complexifier au fur et à mesure. Une tâche doit être parfaitement opérationnelle pour passer à la tâche suivante, comme je l'ai expliqué au sujet du développement d'un PN pour le réseau TIB. L'utilisation de l'Intelligence Artificielle permet de gagner du temps et l'utilisation de simulateurs pour vérifier ce qu'elle délivre, permet souvent de corriger certains problèmes avant même de fabriquer le premier prototype.

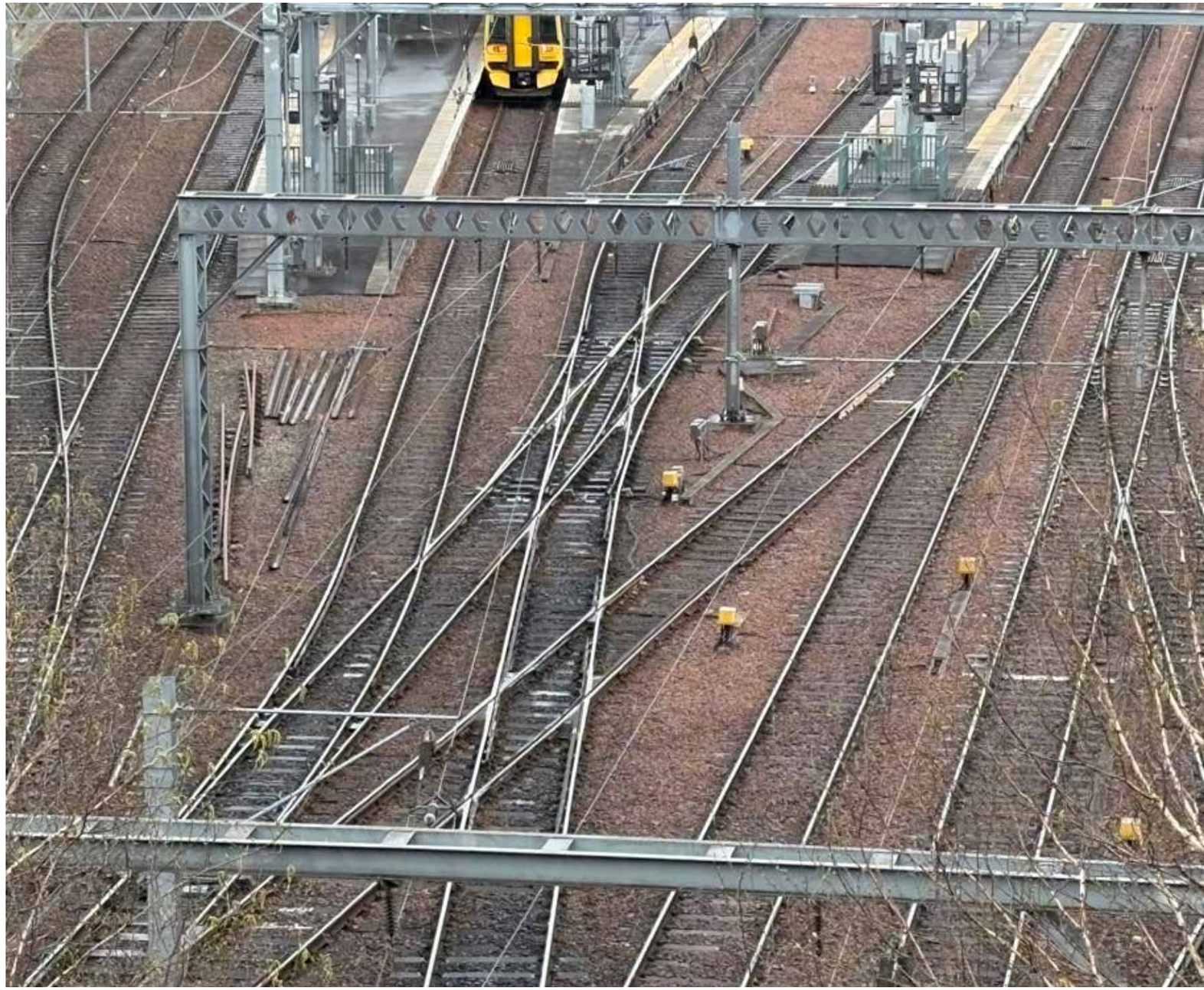
Concevoir son projet
(image réalisée par une IA)

Concevoir son projet (image réalisée par une IA)

07 Montages utiles sur un réseau miniature

Dans ce chapitre, je vais vous faire découvrir quelques montages qui peuvent être reproduits et améliorés pour figurer sur votre réseau. Ils ont été développés et testés sur le simulateur Wokwi et les liens fournis vous permettront de récupérer le programme et de les voir fonctionner, comme cela avait été fait dans le tome 1.

Levier d'aiguillage à la SNCF



Le gril de la gare d'Edimbourg

7.1 / CLIGNOTEMENT MULTIPLE

Le décor d'un réseau miniature comprend beaucoup de LED clignotantes de différentes couleurs : jaune pour un véhicule d'intervention (travaux, dépannage), bleu pour la police, rouge pour les pompiers, etc. Les rythmes de clignotement ne sont pas toujours les mêmes. S'il est simple de faire clignoter une LED, c'est toujours plus difficile d'en faire clignoter plusieurs à des rythmes différents. Grâce à la puissance du langage C, cela devient en fait assez facile. Le montage de la **figure 7-1** montre un exemple pour faire clignoter 5 LED avec des fréquences allant de 0,5 Hz à 4 Hz ; encore une fois, la carte Uno peut être remplacée par une Nano.

Ce montage a été décrit dans la **fiche pratique III.87 dans Loco-Revue 930 de janvier 2025**. L'instruction qui concerne une LED est :

```
digitalWrite(LED0, (millis() % 2000) <100);
```

Le premier nombre (2000) est la période voulue pour le clignotement (ici 2 sec) et le deuxième nombre (100) est le temps pendant lequel la LED doit être allumée (100 millisecondes, soit 0,1 sec). Cette LED bleue va émettre un flash de 0,1 seconde toutes les 2 secondes. Si le deuxième nombre est la moitié du premier, alors la LED est autant allumée qu'éteinte (clignotement symétrique), ce qui est le cas pour les 4 autres LED (LED1 à LED4). Avec une carte Uno ou Nano, on peut ainsi faire clignoter comme on le veut jusqu'à 20 LED.

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/406095025547560961**

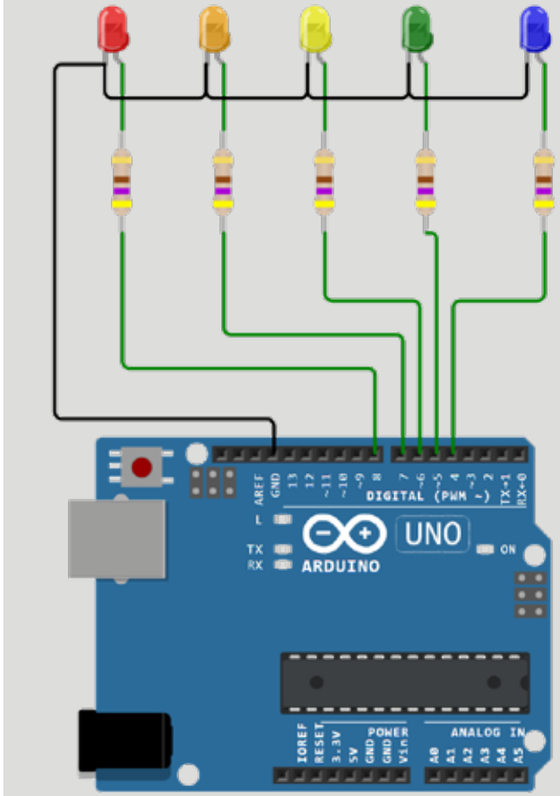


Figure 7-1

RÉSUMÉ

Ce montage nous a montré comment bannir la fonction delay qui reste bloquante et peut gravement nuire à la surveillance de capteurs.

7.2 / RÉGLAGE DE SERVOMOTEUR D'AIGUILLES

On peut **utiliser un servomoteur pour réaliser un mouvement lent d'aiguilles,** mais il est nécessaire de repérer la position « aiguille droite » et la position « aiguille déviée ». En général, ces deux positions sont à un certain angle, autour de la position médiane 90° du servomoteur. Celui-ci doit donc être positionné à 90° avant d'installer le palonnier commandant la tige. La **référence 6170 de chez Décapod** permet de trouver cette position médiane : il suffit d'alimenter le module via un câble mini-USB et de brancher le servomoteur pour qu'il se mette à la position 90°. Décapod vend un ensemble **support et servomoteur** pour un prix modique, sous la **référence 6171**. L'angle de déplacement du palonnier qui commande l'aiguille dépend de la géométrie de la voie, de l'échelle et de l'écartement pratiqués, et de l'épaisseur du plateau. Supposons que cet angle soit 20° de part et d'autre de la position médiane, cela signifie que le servomoteur doit aller de 70° à 110°. Selon le soin apporté au montage, il se peut que ces valeurs ne soient pas tout à fait justes et qu'en réalité, le servomoteur aille de 68° à 105°. Le montage proposé par la **figure 7-2** permet de trouver les valeurs exactes qu'on pourra ensuite entrer dans le programme de la carte qui gère le réseau (on peut utiliser une carte Uno à la place de la Nano).

L'utilisation est simple : la position du servomoteur est affiché sur l'écran OLED, le poussoir vert permet de diminuer la position de 1° à chaque appui et le poussoir bleu de l'augmenter. On peut aussi rester appuyé ce qui entraîne le servomoteur dans un sens ou dans l'autre. Lorsque les lames sont presque en position, il faut y aller degré par degré jusqu'à ce qu'elles soient bien collées. Ne pas insister au-delà car cela pourrait endommager la transmission, voire le servomoteur. Lorsque la position est atteinte, il suffit de noter la valeur. Le montage peut être alimenté avec une simple pile 9 V (sur Vin et GND) et c'est l'occasion de concevoir un circuit imprimé pour recevoir le Nano, les deux poussoirs, l'écran OLED et une sortie à trois broches pour le servomoteur.

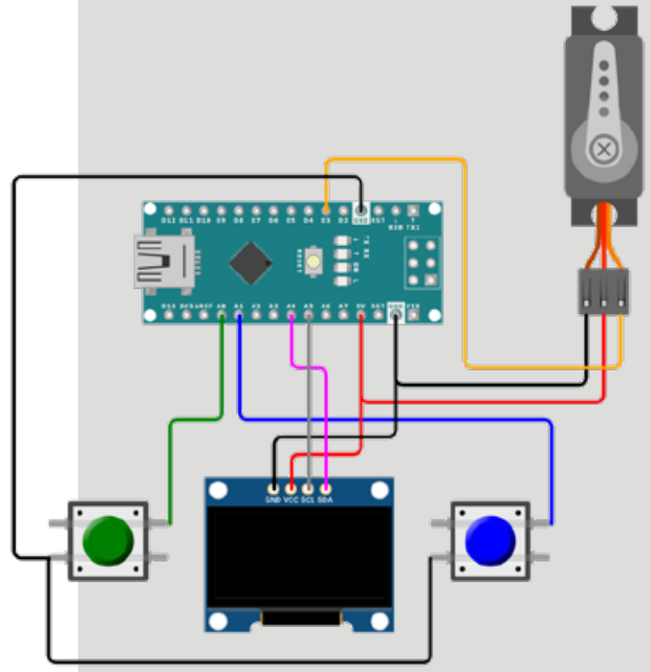


Figure 7-2

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/428556837997113345** .

RÉSUMÉ

Ce montage constitue un outil que tous les modélistes devraient posséder et une bonne occasion pour apprendre à dessiner un PCB recueillant l'ensemble des composants.

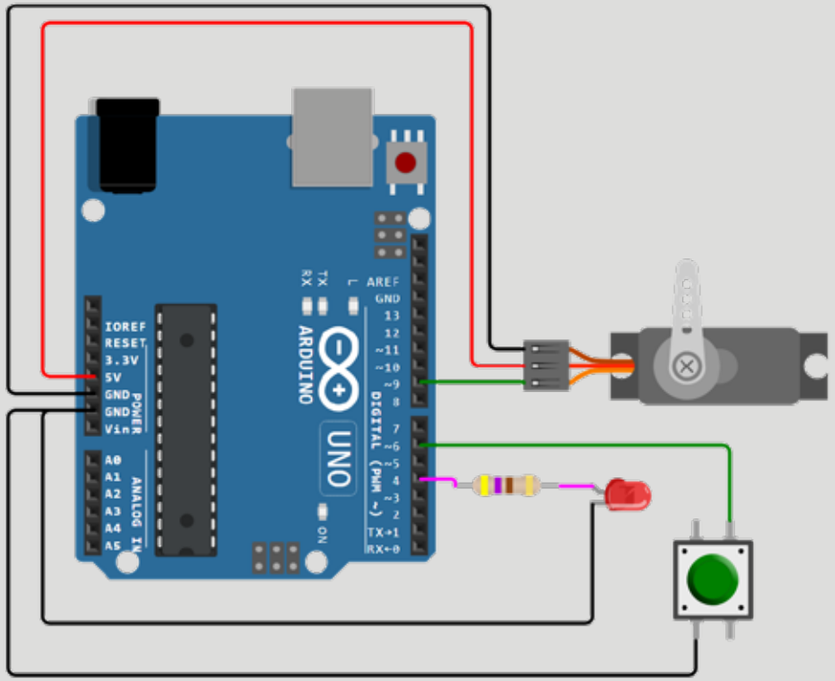


Figure 7-3

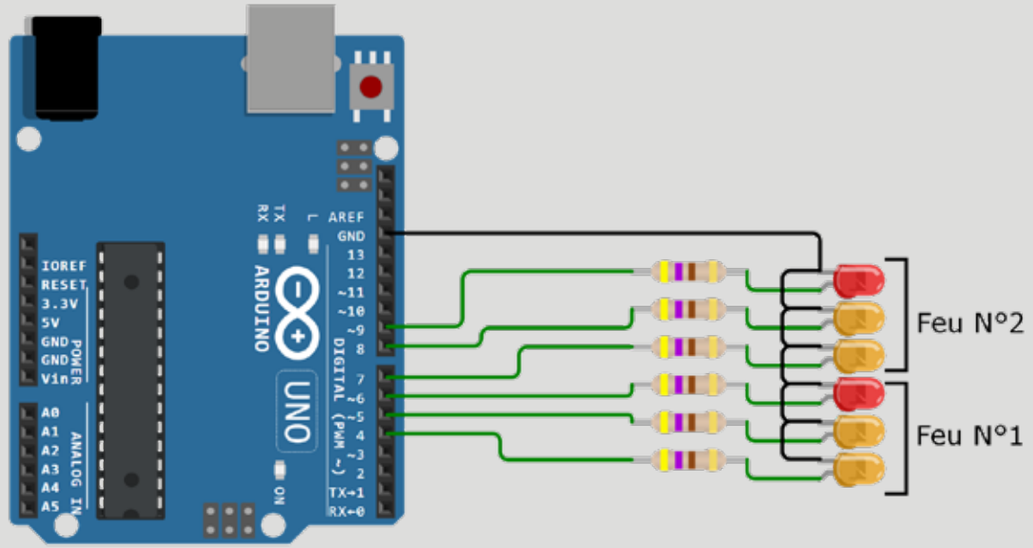


Figure 7-4

7.3 / PASSAGE À NIVEAU DÉCLENCHÉ PAR UN BOUTON POUSSOIR

Le montage de la **figure 7-**3 reproduit le **fonctionnement d'une barrière de PN** : feux qui clignotent, puis fermeture de barrière, clignotement des feux jusqu'à la réouverture de la barrière. Le poussoir vert permet aussi bien de fermer que d'ouvrir la barrière.

Pour installer ce PN sur un réseau, il suffit de mettre **deux I.L.S en parallèle à la place du poussoir** : le premier I.L.S déclenché fermera la barrière et le deuxième l'ouvrira. Les I.L.S doivent être placés suffisamment loin du PN pour permettre la manœuvre de la barrière avant que le train n'arrive. Vous pouvez aussi réduire le temps de clignotement avant fermeture (réglé à 5 secondes) en modifiant la ligne 37. Le train peut circuler dans les deux sens. **Le programme de ce petit PN a été écrit par l'IA ChatGPT.**

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/404910709193537537 .**

RÉSUMÉ

Il y a toujours la place sur un réseau pour y mettre un passage à niveau. Avec un deuxième servomoteur, on peut disposer de deux barrières. Le poussoir peut être remplacé par un détecteur qui permet d'automatiser les barrières en fonction de l'arrivée d'un train.

7.4 / FEU DE CIRCULATION ALTERNÉE

Si on trouve dans le commerce des reproductions de feux tricolores de carrefour, dont la séquence d'allumage n'est pas toujours bien respectée, on ne trouve pas à notre connaissance de **feux de circulation alternée** suite à la fermeture d'une des deux voies (travaux par exemple). Voilà une idée pour nos industriels, d'autant que j'ai mis le programme dans le domaine public et qu'ils peuvent l'utiliser. La **figure 7-4** montre le montage : la carte Uno peut être remplacée par une carte Nano, voire même un ATtiny84 (voir le tome 1).

Les six LED constituent deux feux dont les ampoules vertes ont été remplacées par des oranges pour rappeler qu'il faut rester prudent même si on est autorisé à passer. La séquence avec les deux feux au rouge est assez longue pour laisser le temps de dégager la voie avant d'autoriser l'autre côté à passer. **Le programme a été écrit par ChatGPT à qui j'avais demandé d'utiliser ma bibliothèque LightEffect décrite dans le tome 1, et il a fini par y arriver.**

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/405092208869763073 .**

RÉSUMÉ

Ce programme démontre qu'on a toujours intérêt à utiliser des bibliothèques dans un projet afin de ne pas avoir à réinventer la roue. Ici, LightEffect permet d'obtenir différentes animations lumineuses pour le décor du réseau.

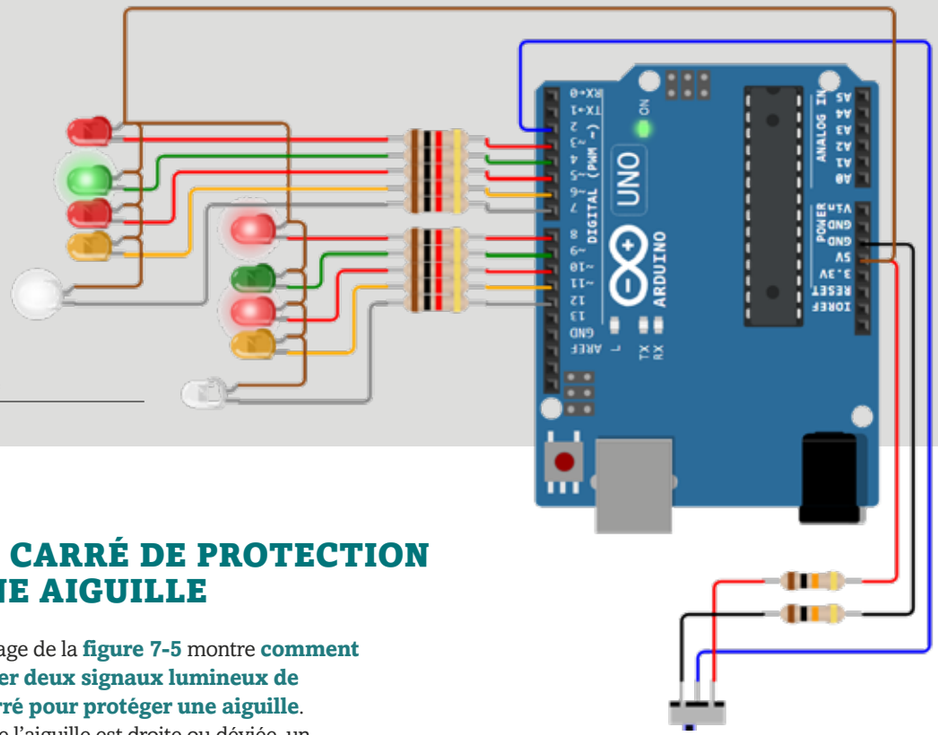


Figure 7-5

7.5 / CARRÉ DE PROTECTION D'UNE AIGUILLE

Le montage de la **figure 7-5** montre **comment alimenter deux signaux lumineux de type carré pour protéger une aiguille**. Selon que l'aiguille est droite ou déviée, un des signaux est au vert (voie libre) et l'autre en carré (aiguille non positionnée pour la voie).

Sur un réseau, l'interrupteur peut être remplacé par un des interrupteurs de fin de course du moteur de l'aiguille, l'autre interrupteur alimentant la pointe de cœur. Les **signaux sont à anodes communes** mais peuvent être remplacés par des signaux à cathodes communes : dans ce cas, le fil commun (marron) doit être relié au GND et il faut modifier les lignes 10 et 11. Remarquez que les signaux sont tous les deux branchés sur des sorties qui se suivent et dans l'ordre des feux de haut en bas (rouge, vert, rouge, orange, œilleton). On peut alors se contenter de ne **déclarer que le branchement de la première LED** et le programme calcule les autres positions. Notez que **la fonction setSignala été écrite par ChatGPT** où l'état du signal est représenté par une chaîne de caractère (voie libre, carré) entre guillemets.

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/428500889162145793 .**

RÉSUMÉ

Les aiguilles sont nombreuses sur un réseau miniature et leur protection par une signalisation lumineuse constitue une animation très facile à mettre en place. Vous pouvez vous inspirer de la fonction écrite par l'IA pour animer n'importe quel type de signal, surtout si vous la combinez avec le montage 7-1 qui vous permettra de faire aussi clignoter les lampes du signal.

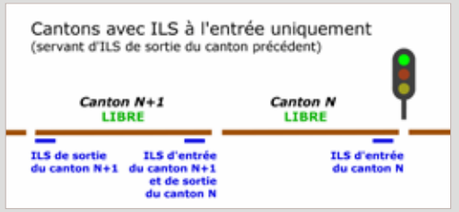


Figure 7-6

les cantons sont représentés séparés mais les rails ne sont pas tronçonnés

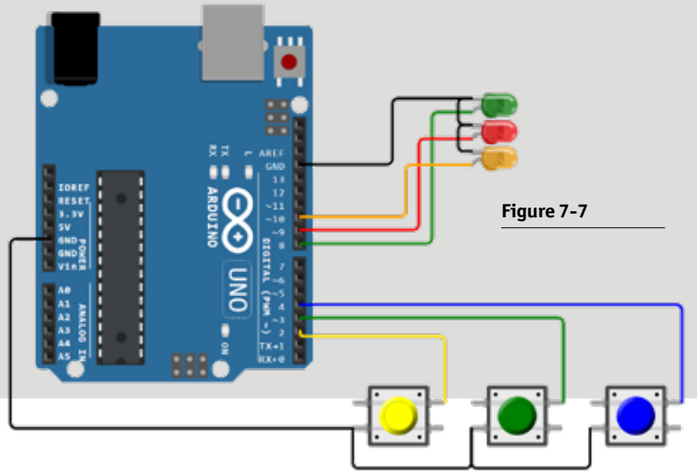


Figure 7-7

7.6 / SIGNAL DE BLOCK AUTOMATIQUE LUMINEUX AVEC DEUX CANTONS FICTIFS

Un **canton fictif** est une portion de voie comprise entre deux I.L.S ; il n'y a donc pas de rail à tronçonner et à réalimenter en courant, ce qui fait que **ce système peut facilement être ajouté sur un réseau** sans avoir trop de travail à faire, juste implanter les I.L.S. Avec trois I.L.S, on peut donc définir deux cantons fictifs qui se suivent et placer un signal lumineux à trois feux à l'entrée du premier canton (**figure 7-6**). Lorsque le train entre sur le premier canton, le signal passe au rouge. Lorsque le train a libéré le premier canton et se trouve sur le deuxième, le signal est à l'orange. Lorsque le train a libéré les deux cantons, le signal redevient vert. Le montage s'occupe **uniquement du signal lumineux** et pas de l'arrêt des trains, mais vous pouvez rajouter cette fonctionnalité. Les trains doivent être munis d'un aimant sous la locomotive et sous le dernier wagon, **selon le principe décrit dans la fiche pratique III.88 du Loco-Revue 931 de février 2025**.

La **figure 7-7** montre le montage à réaliser ; pour la simulation, les I.L.S ont été remplacés par des poussoirs qu'il faut appuyer en imaginant le déplacement du train. **Chaque canton a un compteur** qui est incrémenté chaque fois qu'un I.L.S d'entrée est déclenché et décrémenté chaque fois qu'un I.L.S de sortie est déclenché. Pour qu'un canton soit libre, il faut que son compteur soit à zéro. Le système fonctionne aussi si tous les véhicules (locomotive et wagons) sont munis d'un aimant : cela revient à compter les véhicules entrants et les véhicules sortants : si les nombres ne sont pas identiques, c'est qu'une rupture d'attelage a eu lieu et des wagons ont été abandonnés sur le canton qui n'est alors pas libre. En fait, il est plus simple de mettre des aimants uniquement à chaque extrémité du train.

Vous pouvez récupérer le programme ici : **https://wokwi.com/projects/428565479294163969 .**

RÉSUMÉ

Le cantonnement d'un réseau est à envisager dès qu'on veut automatiser la circulation des trains et ce montage vous montre une solution possible. Ici, seule la signalisation lumineuse est commandée mais il n'est pas compliqué d'y rajouter des zones d'arrêt devant un sémaphore.

7.7 / GARE CACHÉE AUTOMATIQUE

Souvent située à l'arrière du réseau, la **gare cachée** permet de stocker des trains afin de faire varier les circulations. Celle que je vous propose est constituée de quatre voies dont 3 peuvent accueillir un train et les six aiguilles sont motorisées par des servomoteurs. L'ensemble est fait **pour un seul sens de circulation** mais peut être reproduit dans sa totalité pour le sens opposé ; dans ce cas, on peut utiliser une deuxième carte Uno et mettre tous les poussoirs verts sur le même tableau de commande (voir plus loin), ou bien utiliser une carte Mega qui permettra aussi d'avoir plus de voies. La **figure 7-8** montre la géométrie de la gare : quatre voies numérotées de 0 à 3 et six aiguilles A1 à A3 qui constituent les aiguilles d'entrée de gare et A4 à A6 qui constituent les aiguilles de sorties. Sur chaque voie, une zone d'arrêt ZA est commandée par un relais 1RT. La zone d'arrêt ZA0 (voir **figure 7-9**) permet de protéger l'entrée en gare : le train suivant s'arrête dessus pendant que le train précédent entre ou sort. Les zones d'arrêt ZA1 à ZA3 sont là pour que le train s'arrête ; elles ne sont alimentées que pour faire partir le train qui est stocké sur la voie.

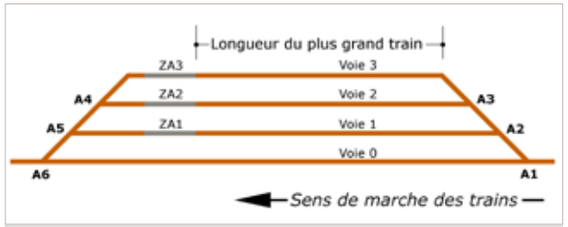


Figure 7-8

La **figure 7-9** montre l'implantation de la gare sur le réseau : on constate une balise d'entrée et une balise de sortie (deux I.L.S) dont on précisera les rôles un peu plus loin. La **balise d'entrée** est située près de la gare alors que la **balise de sortie** est située plus loin pour être certain que le train a bien dégagé l'emprise avant d'autoriser

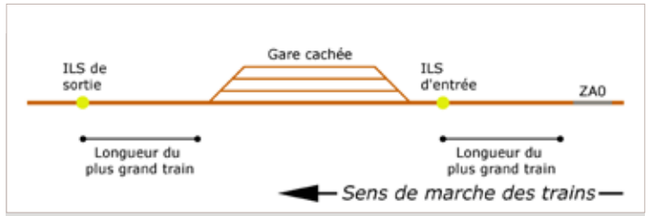


Figure 7-9

d'autres trains à utiliser la gare. Par contre, la ZA0 est située assez loin de la gare **pour bien espacer deux trains qui se suivent**.

Le principe de l'automatisme de cette gare est assez simple : on recherche si une voie est possible pour l'arrivée et le stockage d'un train. Cette voie d'arrivée est **la voie libre située la plus à droite dans le sens de circulation des trains**. S'il existe bien une voie d'arrivée, les aiguilles sont positionnées vers elle ; ainsi le train peut entrer vers la voie et s'arrête puisque les ZA sont coupées en permanence. Si les trois voies sont occupées, alors il n'y a pas de voie d'arrivée possible et **le train passe son chemin** sur la voie V0 sans s'arrêter. L'état d'occupation des voies est signalé par deux LED verte ou rouge sur un tableau de commande. Pour libérer un train, il suffit d'appuyer sur le poussoir vert lié à la voie.

Il y a donc un certain timing à respecter : un train se présente à la balise d'entrée, on coupe la ZA0 pour protéger la gare et on va attendre un certain délai qui doit être réglé pour votre réseau (longueur des voies, vitesse des trains) de manière à laisser le temps au train pour rejoindre sa voie et s'y arrêter. À la fin du délai qu'on choisira bien supérieur au temps réel, on autorise à nouveau l'utilisation de la gare en alimentant la zone ZA0.

Pour faire sortir un train, il faut protéger la gare, positionner les aiguilles de sortie, puis alimenter la ZA de la voie concernée. On pourrait utiliser un délai pour considérer que le train est sorti et la

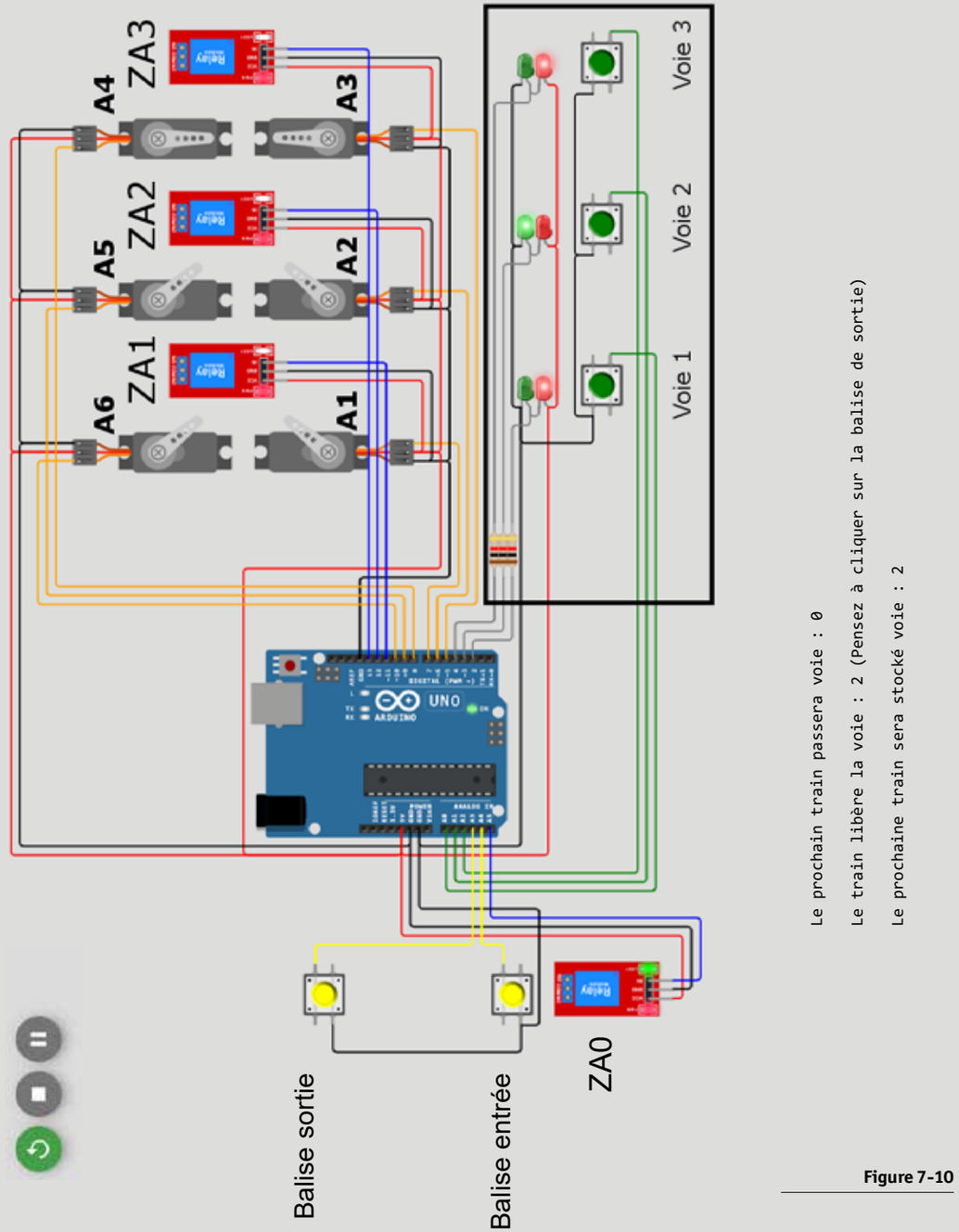


Figure 7-10

gare utilisable à nouveau. Mais imaginons qu'un problème se produise et que le train ne démarre pas : **au bout du délai, la voie serait considérée libre alors qu'elle ne l'est pas**. On ajoute donc une balise de sortie qui donne un signal lorsque le train la survole, preuve qu'il s'est suffisamment éloigné de la gare. C'est la raison pour laquelle cette balise est assez loin de la gare (**figure 7-9**).
Pour des raisons de sécurité, l'initialisation se fait **en considérant que toutes les voies sont occupées** et c'est au joueur de libérer les voies qui sont libres, mais dans ce cas, il n'y a pas de train pour déclencher la balise de sortie : ceci peut être fait manuellement **en branchant un poussoir en parallèle sur l'I.L.S**, qui, appuyé, reviendra au même que l'I.L.S déclenché par un aimant.

Pour réaliser cet automatisme, il vous faut 3 LED vertes et 3 LED rouges, 3 résistances pour les LED, quatre relais de type 1RT, 6 servomoteurs, 4 boutons poussoirs et deux I.L.S. Et bien entendu de quoi fabriquer la gare, six aiguilles et des coupons droits. La **figure 7-10** montre que tout tient sur une carte Uno et **donne le schéma complet** de montage. Le cadre noir est le tableau de commande de cette gare cachée, les poussoirs jaunes sont des I.L.S sur le réseau (le poussoir en parallèle de la balise de sortie n'est pas représenté), les relais sont appelés comme les zones d'arrêt qu'ils commandent et les servomoteurs reprennent la numérotation des aiguilles de la **figure 7-8**. Les servomoteurs n'étant manœuvrés qu'un par un, l'alimentation par la carte Uno devrait suffire : au besoin prévoyez une alimentation à part. Si la LED verte située sur une des cartes relais est allumée, c'est que la zone d'arrêt est alimentée.

Le commentaire de tête du programme donne les consignes d'utilisation. Pour la simulation, il faut cliquer sur les balises (poussoirs jaunes) mais sur le réseau, l'I.L.S sera déclenché lorsque le train le survolera. Pour libérer un train, il faut appuyer le poussoir correspondant à la voie et attendre que le train déclenche la balise de sortie. Il est donc important de bien regarder les affichages du moniteur pour comprendre les actions à faire.

Vous aurez à modifier certaines variables comme la durée d'entrée en gare (fixée à 5 secondes pour la démonstration) et les positions des servomoteurs en fonction de la voie qu'ils desservent.

Vous pouvez récupérer le programme à cette adresse : **https://wokwi.com/projects/429480671346128897 .**

Après ces quelques montages très simples, il est temps d'aborder des projets plus complexes. Je vous en propose trois dans les chapitres qui suivent.

RÉSUMÉ

La gare cachée permet de faire varier les convois qui roulent sur un réseau ; c'est donc un élément essentiel en automatisation. Ce montage montre qu'un programme peut gérer une telle gare et limiter les interventions au choix du convoi à faire sortir.

RÉSUMÉ DU CHAPÎTRE 7

Les différents montages que je vous ai proposés dans ce chapitre peuvent constituer une base pour des projets d'automatisation plus sophistiqués. En effet, les grandes composantes d'un réseau sont les aiguilles, le cantonnement, le passage à niveau et la gare cachée. Chaque composante peut être gérée par sa propre carte Arduino qu'on peut faire communiquer avec les autres cartes. Il devient alors facile de constituer un ensemble complet de gestion de réseau.

Début de cantons sur la ligne TGV Perpignan-Gérone

08 Un pont tournant pour le réseau

Un pont tournant alimentant une jolie rotonde où on peut parquer ses plus beaux modèles de locomotives est très certainement la pièce maîtresse d'un réseau de trains miniatures. En cas de manque de place, une simple plaque tournante permettant de virer une locomotive dans une petite gare terminus est également une animation intéressante. Nous allons donc voir comment concevoir un pont tournant ou une plaque tournante, en utilisant un moteur pas à pas géré par Arduino. Ce projet, conçu pour un réseau analogique, fait appel à un petit moteur pas à pas peu puissant et est plutôt destiné à un pont à l'échelle N ou une plaque en N ou H0 pour virer une petite locomotive légère. Cependant, il peut servir de base d'inspiration pour un pont motorisé avec un moteur plus puissant dans un réseau numérique.

Prototype du pont de l'auteur pour vérifier le programme

Pont tournant sur un réseau miniature (photo Yann Baude LR908)

8.1 / CAHIER DES CHARGES

J'ai déjà évoqué la commande d'un moteur pas à pas par Arduino dans le guide « Animez votre réseau » (paragraphe 4.6 page 86) et j'ai indiqué quelques considérations théoriques à prendre en compte pour une plaque tournante. Cette fois, **je vais décrire le projet de A à Z en appliquant la méthode décrite au chapitre 6.** Commençons donc par décrire ce que doit réaliser le projet et définir ainsi ce qu'on pourrait appeler cahier des charges.

Le pont (terme utilisé par la suite bien que ce soit la même chose pour une petite plaque tournante) doit permettre plusieurs choses :

1. Le pont doit être relié à une voie permettant l'entrée et la sortie des locomotives.
2. Des voies de garage doivent permettre de parquer les locomotives.
3. L'alimentation de ces voies doit permettre à la locomotive d'avancer ou de reculer et également d'être à l'arrêt.
4. La polarité des voies doit être coordonnée afin de ne pas créer de court-circuit lors des manœuvres d'entrée ou de sortie.
5. La voie située sur le pont doit permettre toutes ces manœuvres (entrée, sortie, arrêt).
6. Les fils d'alimentation de la voie du pont ne doivent pas s'entortiller suite à de nombreuses manœuvres du pont.
7. La rotation du pont doit permettre un alignement parfait avec la voie sélectionnée.
8. La rotation du pont doit être commandée par une interface simple à utiliser.
9. Un écran doit permettre de contrôler toutes les manœuvres du pont.

Comme on le voit, on doit à la fois régler des **problèmes de mécanique** (positionnement du pont) et des **problèmes d'électricité** avec l'alimentation des voies sans créer de court-circuit. Le **point 1** est évident, la voie d'entrée est reliée au reste du réseau alors que le pont et ses voies de garage constituent un sous-ensemble à part. Les **points 2 et 7** sont faciles à solutionner en utilisant un moteur pas à pas : pour aller d'une voie à une autre, le logiciel sait combien de pas il faut parcourir dans un sens ou dans l'autre. On peut utiliser un **clavier à 16 touches pour commander le pont dans sa globalité** (positionnement et alimentation des voies), ce qui résoud le **point 8**, ainsi qu'un écran I2C pour résoudre le **point 9**. Les **points 3 et 5** nécessitent des relais inverseurs afin de changer la polarité des rails (sens d'avancement) ainsi que des **relais simples** pour établir ou couper les alimentations des voies. Le logiciel s'occupera de coordonner la polarité des voies pour éviter les courts-circuits, ce qui résoud le **point 4**. Enfin, **pour éviter aux fils d'alimentation du pont de s'entortiller** (**point 6**), il faut interdire à celui-ci de faire un tour complet et définir une position par laquelle il ne doit jamais repasser. Notre cahier des charges a été défini et quelques solutions envisagées. Il reste à les mettre en pratique.

Le projet est ensuite construit en plusieurs étapes : lecture des touches du clavier afin de définir la voie, mouvement du pont dans un sens ou dans l'autre de la quantité de pas permettant de s'aligner avec la voie choisie, affichage sur l'écran des différentes commandes ou positions du pont, commande des cartes à relais. On ne peut passer qu'à l'étape suivante **que si l'étape précédente donne entièrement satisfaction**.

8.2 / CHOIX DES COMPOSANTS

Pour ce projet, on utilisera le moteur pas à pas 28BYJ-48 (2048 pas entier par tour, 4096 demi-pas), **afin de motoriser le pont tournant Peco échelle N** vendu en kit à monter. Le choix de l'échelle N est imposé par le fait que le moteur choisi n'est pas très puissant et qu'il faut donc limiter le poids sur le pont à manœuvrer. L'interface de ce moteur est commandée avec quatre sorties d'Arduino.

On utilise également **un clavier à 16 touches** pour sélectionner les voies (interface homme-machine), ce qui nécessite 8 sorties supplémentaires (4 lignes et 4 colonnes). Ce clavier doit permettre de choisir la voie devant laquelle le pont se positionne, mais aussi de faire manuellement des mouvements de large amplitude ou bien d'un seul pas.

Enfin, il faut aussi commander l'alimentation des voies avec des **cartes relai**s et faire en sorte de ne pas créer de court-circuit ; plus on a de voies reliées au pont, plus il faut de relais et de sorties pour les commander. La polarité du courant alimentant les voies est confiée à des **relais DPDT** (Double Pole Double Throw) **ou relais 2RT** (deux contacts repos-travail).

L'écran de type I2C ne nécessite que les sorties SDA et SCL pour être commandé. Il est très vite apparu qu'une carte Arduino Uno n'est pas suffisante (sauf si le pont n'est relié qu'à deux ou trois voies) ; le choix de la carte Arduino s'est donc porté sur la **carte Mega 2560 Rev3**, qui permet de compléter ultérieurement le projet en ajoutant des voies. **Le projet est décrit ici avec une voie d'entrée et quatre voies de sorties vers la remise.**

8.3 / GÉOMÉTRIE DU PROJET

Comme on l'a dit, l'écart angulaire entre deux voies doit permettre un nombre entier de pas de déplacement. En choisissant un angle de 22,5°, il faut 128 pas entiers ou bien 256 demi-pas pour ce moteur pour aller d'une voie à la voie adjacente. On peut travailler en pas entiers et utiliser la bibliothèque Stepper, mais pour une plus grande précision, il vaut mieux travailler en demi-pas car ce n'est pas plus compliqué comme on le verra un peu plus loin. Avec un angle de 22,5°, nous obtenons 16 positions possibles pour les voies (numérotées C0 à C15, C pour cible). La **figure 8-1** montre la géométrie du projet et la façon dont ces voies sont positionnées sur ces positions cibles. Cette figure définit un index (bleu ciel) situé en face de C0 qui représente l'arrière d'une machine qui entre sur le pont (on y reviendra plus loin).

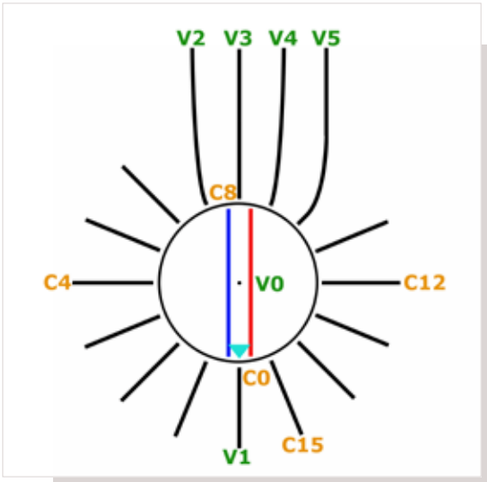


Figure 8-1

La voie d'entrée du pont est la voie V1, sur la position C0. Les voies de sortie vers la remise sont les voies V2 à V5, positionnées sur C7 à C10. Ce choix du nombre de voie et de leur position est arbitraire ; vous pouvez en faire un autre, c'est le programme qui fera la correspondance voie-position cible. **La voie sur le pont est la voie V0.**

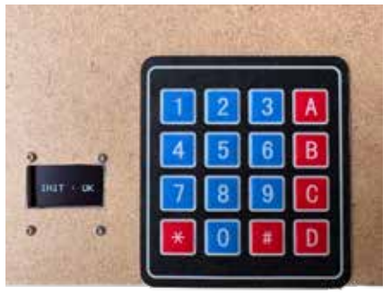
8.4 / UTILISATION D'UN MOTEUR PAS À PAS UNIPOLAIRE EN DEMI-PAS

Pour faire tourner un moteur unipolaire en pas entiers, il suffit d'**alimenter successivement ses bobines dans un certain ordre** (qui détermine le sens de rotation) et à une certaine fréquence (qui détermine la vitesse de rotation). Donc, on alimente successivement bobine 1, bobine, 2, bobine 3, bobine 4. **Pour travailler en demi pas**, il faut alimenter la bobine 1 (le rotor se cale juste en face d'elle), puis on alimente simultanément bobine 1 ET bobine 2 (le rotor se cale entre la bobine 1 et la bobine 2 et a donc parcouru un demi-pas), puis on alimente QUE la bobine 2 (le rotor parcourt un demi-pas et se cale en face d'elle), etc. avec les autres bobines. Or, pour alimenter les bobines, il suffit d'alimenter les entrées de l'interface de commande à base d'ULN2003 ou ce qui revient au même, les sorties de la carte Arduino reliées à ces entrées de l'interface. Si ces sorties sont choisies sur un même Port, on peut écrire directement dans le Port au lieu d'utiliser des digitalWrite. Ici, **nous utilisons le Port F** (sorties 54 à 61) de la carte Mega pour commander le moteur pas à pas qui n'a besoin d'être relié qu'aux quatre premières sorties du Port (54 à 57 ou encore A0 à A3). Les valeurs à afficher sur le Port sont successivement 1, 3, 2, 6, 4, 12, 8, 9 (si vous convertissez en binaire, vous retrouverez la séquence décrite plus haut pour alimenter les bobines).

8.5 / INTERFACE HOMME-MACHINE

Un **clavier à 16 touches**, comme on en trouve dans les kits de début, suffit pour commander le pont tournant ; il est relié aux broches 4 à 11 de la carte (de 4 à 7 pour les colonnes et de 8 à 11 pour les rangées). Sa gestion est confiée à la **bibliothèque Keypad** qu'il faut installer dans l'IDE. La fonction getKey() permet de récupérer le caractère de la touche enfoncée, une variable de type char appelée keyPressed : en fonction de ce caractère, on réalise telle ou telle fonction. C'est le caractère récupéré qui permet de faire la correspondance entre voie sélectionnée et cible à aller chercher (ce qui dépend de votre disposition de voies).

Les touches **1 à 9** permettent de sélectionner la voie en face de laquelle le pont se positionne. La touche '**A**' (comme ahead ou avance) permet la marche avant, la touche '**B**' (comme back ou recule) permet la marche arrière et la touche '**C**' (comme calm ou stop) permet d'arrêter la machine.



Interface Homme Machine (IHM) du pont : un clavier pour commander, un écran pour contrôler

D'autres touches permettent un déplacement précis du pont en rotation. La touche '**D**' du clavier sert à effectuer un déplacement de 2**56 demi-pas sens des aiguilles d'une montre (CW pour clockwise)**, ceci afin de faire un positionnement grossier du pont. La touche '**0**' permet un déplacement de **16 demi-pas dans le sens CW**. La touche '*****' permet un déplacement **d'un demi-pas dans le sens CW** alors que la touche '**#**' permet **la même chose dans le sens contraire des aiguilles d'une montre (CCW pour counterclockwise)**. On utilise ces déplacements possibles autant de fois que nécessaire pour obtenir l'alignement parfait avec la voie d'entrée V1.

Pont et rotonde (photo Yann Baude)

8.6 / INITIALISATION DU PONT

Pour que le pont sache combien de demi-pas il doit parcourir pour aller chercher une voie, il faut que **son initialisation soit faite juste en face de la voie d'entrée qui constitue alors sa référence (C0)**. Une fois que cette initialisation est faite, tout devient très simple. Si CD est la cible de départ et CA la cible d'arrivée, le déplacement à faire est (CA – CD) * 256 : si ce déplacement est positif, le sens de rotation doit être celui des aiguilles d'une montre (CW), si le déplacement est négatif, c'est le sens inverse (CCW pour counterclockwise). Ce déplacement peut être long, **mais il ne traverse jamais la position C0**, ce qui permet de ne pas entortiller les câbles d'alimentation de la voie du pont. Le sens de déplacement est géré par une variable booléenne qui vaut 'true' pour CCW et 'false' pour CW. De plus, **le déplacement du pont ne peut se faire que si la voie V0 n'est pas alimentée**, ce qui assure que la machine sur le pont est bien arrêtée.

Il faut maintenant aller chercher la voie d'entrée pour procéder à l'initialisation du pont. Comme le moteur est démultiplié, on ne peut pas le manœuvrer à la main ; il faut donc aller chercher l'alignement parfait avec la voie d'entrée en utilisant le moteur. Grâce aux touches '**D**', '**0**', '*****' et '**#**', on peut **manœuvrer le pont jusqu'à être parfaitement en face de la voie d'entrée V1**.

Une fois le pont amené précisément en face de la voie d'entrée, **il suffit de reseter la carte** (donc le programme) pour que le pont prenne la voie V1 (donc la cible C0) pour référence de ses déplacements. Cette référence peut être perdue si le pont n'est pas en face de la voie V1 et que l'alimentation est coupée, qu'on effectue un reset, qu'on ouvre le moniteur série (ce qui équivaut à un reset) ; **dans ce cas, la procédure d'initialisation est à refaire**. Mais cette référence peut aussi être conservée si on prend soin de mettre le pont face à la voie 1 avant de couper l'alimentation du montage.

Lorsque l'initialisation est faite, il suffit de taper sur le clavier la voie voulue (de 1 à 5 pour ce projet) pour que le pont aille la chercher.

8.7 / GESTION DE L'ALIMENTATION DES VOIES

Il y a **deux objectifs** à atteindre :

1. Une machine doit pouvoir entrer et sortir du pont mais aussi s'arrêter dessus pour permettre le mouvement de rotation du pont.
2. La voie du pont V0 ne doit pas être en court-circuit avec les autres voies (V1 à V5).

Dans le premier cas, il faut un relais 2RT pour inverser le courant sur la voie V0 et un relais 1RT (R0) pour couper l'alimentation de la voie V0, soit deux sorties de commande des relais. Pour le deuxième cas, seule la voie située en face de l'index (voir **figure 8-1**) **doit être alimentée temporairement et de la même façon que la voie V0 pour permettre l'entrée ou la sortie d'une machine**. Ceci nécessite donc d'autres relais de type 1RT (Rx) et d'autres sorties pour les commander. La **figure 8-2** montre l'alimentation des voies de sortie en fonction de l'alimentation du feeder de la voie V0.

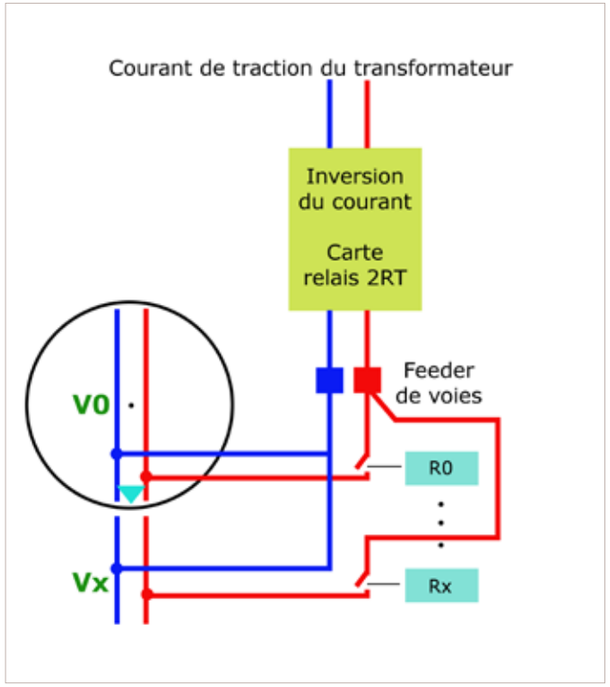


Figure 8-2

Le programme gère le déplacement d'une machine sur le pont avec une variable **sensMarche** qui vaut 0 pour arrêt, 1 pour marche avant et 2 pour marche arrière. Ses trois cas sont commandés par les touches '**A**', '**B**' et '**C**'.

L'index bleu ciel (**figure 8-1**) représentant l'arrière d'une locomotive, il faut aller chercher **la cible C8 (voie V3 pour ce projet) si on veut sortir du pont en marche avant**. Et pour sortir, il faut que C0 (voie V1) soit aussi alimentée avec la bonne polarité, et pour l'instant elle n'est pas alimentée puisque seule la voie en face de l'index l'est. La solution la plus simple consisterait à prévoir une alimentation à part pour lepont et ses voies de garage (V0 et V2 à V5) et à alimenter la voie V1 par l'alimentation du réseau en veillant à ce que ces deux alimentations soient polarisées comme il convient pour les entrées ou sorties du pont. **Pour une automatisation totale**, on peut aussi prévoir un **deuxième relais 2RT** pour inverser le courant sur V1 chaque fois que le pont est en face de la voie V3 (C8) et qu'on veut sortir en marche avant. **Dans ce cas, le pont tournant alimente une portion du réseau : la voie qui arrive jusqu'à lui.**

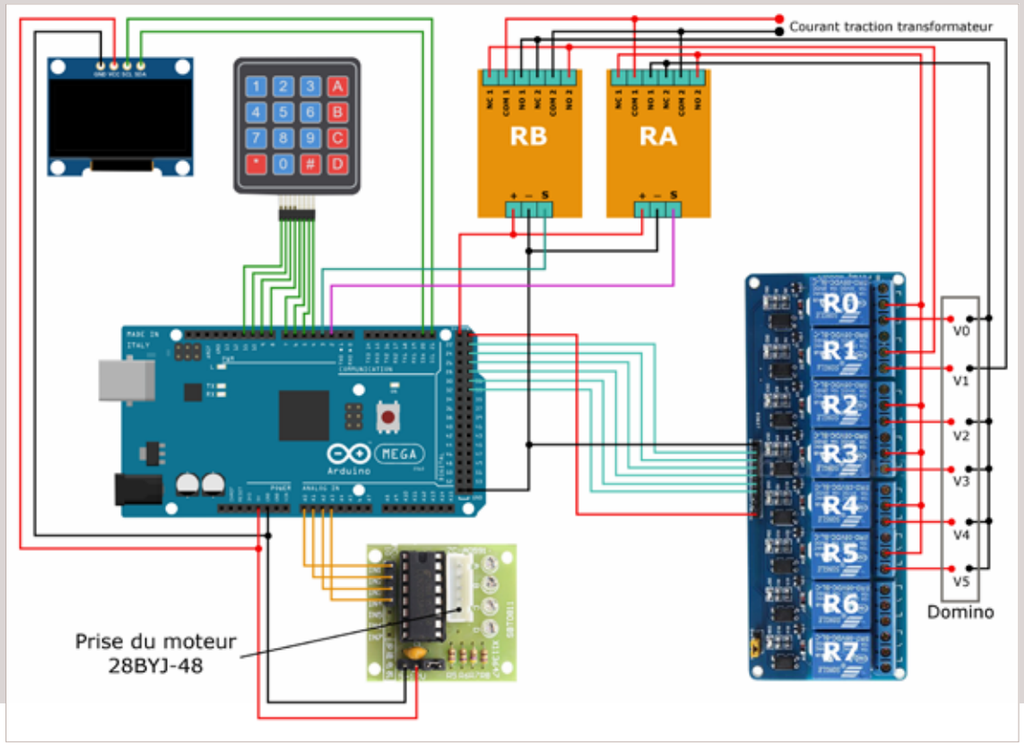


Figure 8-3

8.8 / PLAN DU MONTAGE

La **figure 8-3** montre comment relier tous les composants à la carte Mega 2560, **pour cette deuxième solution**. Outre la carte, nous trouvons l'interface de commande du moteur, le clavier à 16 touches, une carte 8 relais 1RT (R0 à R7), deux cartes à un relais 2RT (RA et RB), et un écran OLED relié en I2C. Le clavier est raccordé aux broches 4 à 11 et l'interface de commande du moteur est reliée aux broches A0 à A3. La carte relais 1RT utilise 6 relais (R0 à R5) pour alimenter ou non les voies V0 à V5 ; ces relais sont commandés par un état LOW sur la sortie concernée. Le relais **RA permet d'inverser le courant sur le feeder voie V0** et le relais **RB sur la voie V1** ; ces relais sont commandés par un état HIGH sur la broche 2 ou 3.

Les voies de garage du pont (V2 à V5) sont reliées au feeder de la voie 0 (un simple domino) et un rail passe par les relais R2 à R5, conformément à la **figure 8-2**. La sortie du relais RB alimente la voie V1 et un rail passe par le relais R1. Il est **conseillé de prévoir une alimentation de 5 V DC pour alimenter les cartes relais** (fil marron (+) et bleu (GND)). C'est le programme qui commande le relais RB en fonction du sens de marche voulu lorsque la voie 3 est sélectionnée.

Si vous préférez alimenter la voie 1 indépendamment du pont, n'utilisez pas le relais R1 et ne connectez pas la carte relais RB.

8.9 / INCORPORATION AU PONT TOURNANT PECO

Soudez deux fils souples aux extrémités des rails du tablier du pont, puis passez ces fils sous le tablier en pratiquant deux petites encoches dans le but de rejoindre l'axe de rotation (**Figure 8-4**). Fixez les fils sous le tablier et passez-les le long de l'axe de rotation.Ils ne doivent pas frotter avec le trou dans la fosse du pont ; pour cela, utilisez un petit bout de tube en cuivre (de plomberie, diamètre 12 mm) d'une part pour que les fils soient à l'intérieur, d'autre part pour centrer le mouvement dans la fosse (**figure 8-4**). Laissez tout de même un peu de mou entre ces fils et l'arrivée du courant (feeder de voies) pour permettre le mouvement. **Pour le reste du montage, suivez la notice sans monter les pièces de cuivre faisant contact.** L'axe du moteur **doit être confondu avec l'axe du pont** ; pour cela, fabriquez une pièce en bois, résine ou métal, réunissant l'axe du tablier et l'axe du moteur. Sans tour ni imprimante 3D, vous pouvez récupérer la partie filetée d'un boulon M4 x 0,70 pour constituer un axe collé à l'époxy dans le trou du tablier ; cet axe est vissé dans un **manchon (tube fileté intérieurement au pas de 0,70)**, ce qui permet de régler précisément la hauteur du tablier tournant si nécessaire. Une gorge de 3 mm de large est fraisée à la lime aiguille pour recevoir l'arbre moteur comme le montre la **figure 8-4**.

Toutes les pièces se trouvent facilement sur internet. Il est nécessaire d'agrandir le trou de la fosse du pont en raison de la présence du tube de cuivre. Il faut réduire les sources possibles de frottement et pour cela, **on peut envisager un petit roulement à billes** monté sur le tube en cuivre.

Les voies qui arrivent au pont doivent être posées **en respectant bien l'angle de 22,5° entre deux voies**, être au même niveau que les voies du tablier et leurs rails doivent être légèrement chanfreinés (voir paragraphe 7 de la notice de montage). Tout cela demande un travail soigné mais qui est tout à fait à portée d'un modéliste. Le programme peut aussi prévoir **une correction de position** pour chaque voie (en nombre de demi-pas en plus ou en moins) pour tenir compte du fait qu'une voie n'est pas tout à fait en face de la position cible.

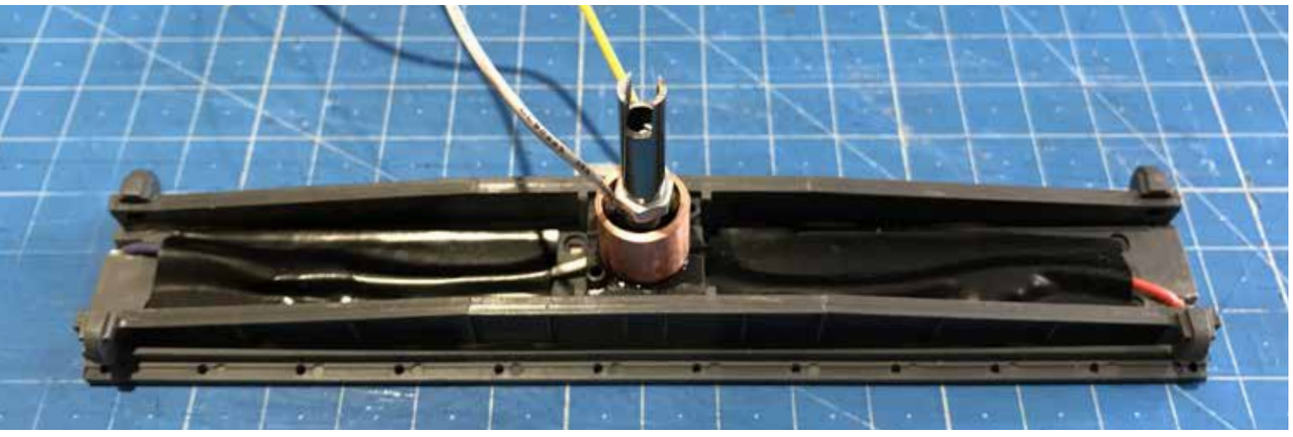


Figure 8-4

8.10 / UTILISATION DU PONT

Positionnez l'index du pont en face de la voie 1 pour permettre l'entrée d'une locomotive avec les touches '**D**', '**0**', '*****' et '**#**'. Lorsque l'alignement est précis, resetez la carte : **l'initialisation du pont est faite**. Faites avancer la locomotive à vitesse réduite avec la touche '**A**' **pour bien la centrer sur le pont**, puis arrêtez la avec la touche '**C**'. Sélectionnez la **voie à atteindre** avec la touche adéquate (2 à 5 dans ce projet) : le pont tourne jusqu'à aller en face de cette voie. Faites ensuite **reculer la locomotive** à vitesse réduite avec la touche 'B' pour la faire pénétrer dans la voie de sortie. Arrêtez la avec la touche '**C**'. Sélectionnez la voie 1 pour que le pont effectue une rotation vers celle-ci et **soit prêt à recevoir une autre machine**. Vous pouvez aussi sélectionner une autre voie pour sortir une autre machine de la remise. **Pour sortir une locomotive du pont, il faut sélectionner la voie V3** (cible C8), puis agir sur la touche '**A**' pour faire avancer la machine.

Le courant dans les voies est automatiquement coupé quand le pont tourne. Le programme affiche des messages sur l'écran OLED et dans le moniteur série, et **il faut bien attendre l'affichage de ces messages avant d'appuyer sur une touche du clavier**.

Il n'est pas possible de faire un tel montage sur les simulateurs d'Arduino. Cependant, vous pouvez récupérer le programme sur Wokwi : **https://wokwi.com/projects/383924965574672385.**

Seuls carte Arduino Mega, clavier et écran sont visibles côté composants. Une barre de LED a été ajoutée pour **simuler tous les relais** : les deux LED les plus à droite représentent les relais RA et RB. Il est donc possible de simuler le fonctionnement, mais sans voir la rotation du moteur. Lorsque celle-ci est terminée, un message est affiché sur l'écran « **Voie X OK** ».

8.11 / AMÉLIORATION POSSIBLE

Les ponts du commerce sont généralement construits avec un espacement angulaire fixe entre les différentes voies possibles ; le principe de ce pont est donc parfait pour le commander. Cependant, si vous construisez vous-même le pont, il n'est pas toujours possible de bien espacer les voies au demi-degré près. Grâce au montage précédent, vous pouvez aligner le pont rigoureusement en face de chacune des voies et noter sa position en nombre de demi-pas. Il est alors possible de rentrer dans le programme ces positions pour que le nombre de demi-pas à parcourir soit calculé en fonction de la position cible et de la position actuelle. Vous serez ainsi certain de toujours être en face des voies. Ceci ne dispense pas de l'initialisation qui doit être faite de façon rigoureuse en début d'utilisation, afin d'avoir la bonne référence de départ (position C0).

RÉSUMÉ DU CHAPÎTRE 8

Automatiser un pont tournant est un projet complexe qui demande de la réflexion : choix du moteur, choix de l'interface homme-machine, choix des alimentations nécessaires. Ce chapitre a permis de voir comment résoudre les problèmes pour éviter les courts-circuits et faire en sorte que les voies s'alignent parfaitement pour éviter les déraillements. La méthode de conception est à appliquer quel que soit le projet sur votre réseau.

09 Projet de commande d'un train analogique via un smartphone

Ceux qui pratiquent le DCC peuvent commander leurs trains depuis une tablette ou un smartphone. Ce chapitre va démontrer qu'il est aussi possible de le faire avec un réseau analogique. Nous allons donc rentrer dans le monde des objets connectés grâce à la nouvelle carte Uno R4 et nous nous apercevrons que développer un objet connecté sophistiqué est extrêmement facile grâce aux techniques de l'IoT d'Arduino, dont nous avons déjà parlé au début de ce guide. Voici donc une manette pour commander un train, que vous pourrez étendre à deux trains avec le même matériel, une carte Uno et une carte shield.

Ce projet avait aussi été évoqué dans ses grandes lignes dans le tome 1. Je vais en faire une **description complète réalisée avec la carte Uno R4 WiFi et la carte Arduino Motor Shield**. Cette carte Uno R4 WiFi peut être connectée en WiFi. Le WiFi dont le taux est de 150 Mbps, utilise la bibliothèque WiFiS3 incluse dans le noyau Uno R4. De nombreux exemples d'utilisation sont donnés sur le site d'Arduino. La carte Uno R4 WiFi étant compatible avec l'IoT d'Arduino, elle est parfaite pour créer une **commande de trains analogiques à partir d'un smartphone**. Ce sera l'occasion de mettre en pratique le fonctionnement de l'IoT d'Arduino. Ce projet a aussi été décrit dans ses grandes lignes dans le **Loco-Revue de mai 2022 avec une carte Nano 33 IoT**, mais l'arrivée de la carte R4 WiFi rend encore plus facile la réalisation de l'application puisqu'on peut utiliser une carte Arduino Motor Shield qui s'enfiche sur la carte Uno R4 et qui est capable de commander deux moteurs DC. De plus, les deux cartes travaillent en 5 V alors que ce n'était pas le cas avec la carte Nano 33 IoT. En fait, nous n'utiliserons qu'un seul canal pour commander une seule locomotive via les rails (le canal A). Pour la faire avancer, il faut gérer deux variables :

Commande de vos trains sur un smartphone

la **direction du mouvement** et la **vitesse** qui peut être obtenue par un signal de type PWM.

Le matériel nécessaire est donc réduit à une carte **Uno R4 WiFi, le shield moteur et une alimentation en courant continu de 12 V pour l'alimentation du moteur donc des rails** (par exemple la sortie courant de traction de votre transformateur). Et évidemment, un smartphone (ou une tablette) sur lequel vous aurez le panneau de commande, soit un prix de revient de moins de 60 euros (prix en avril 2025) si on considère que vous avez déjà un smartphone. Ce projet est néanmoins réservé à ceux qui ont déjà l'expérience de la programmation sur Arduino car il n'est pas impossible d'avoir à résoudre de petits problèmes lors de votre premier accès à l'IoT (Internet Of Things) ; le site d'Arduino décrit très bien comment faire. Le fonctionnement du Cloud et de l'IoT a été décrit dans le chapitre 1.

9.1 / LES VARIABLES DU PROJET

La finalité de ce projet est d'obtenir une interface graphique comme le montre la **figure 9-1**, afin de commander une locomotive analogique.

Cette interface dispose d'un potentiomètre linéaire (au centre) qui permet de donner une consigne de vitesse et sens de mouvement. Deux voyants (en bas) donnent une indication du sens de marche. Au sommet, un indicateur reprend la valeur de PWM réglant la vitesse de la locomotive et un bouton (en haut à droite) permet de commander un arrêt d'urgence de la locomotive. **Cette interface graphique sera aisément créée grâce aux widgets proposés par l'IoT d'Arduino.** Pour que l'IoT nous propose un programme, il faut lui indiquer quelles variables sont à prendre en considération.

Figure 9-1

Les variables à ajouter au projet sont **une variable de type int (entier sur 16 bits) pour la valeur du potentiomètre appelée « potValue »**, et **deux variables de type boolean pour les deux LED virtuelles** (si true, la LED est allumée, si false, elle est éteinte). On les appelle « **greenLEDstate** » et « **redLEDstate** ». Ces variables définiront également le sens de marche de la loco. Une variable **« vitesse » de type int** sera affichée sur le tableau de bord (elle correspond à la valeur du rapport cyclique de PWM). Enfin, **une variable booléenne « stopState »** prendra en compte une demande d'arrêt d'urgence. Chaque variable créée implique une modification automatique du sketch.

Créer une variable reste extrêmement simple, comme nous l'avons vu au chapitre 1 : il suffit de renseigner un menu graphique comme le montre la **figure 9-2**.

Figure 9-2

9.2 / MODIFICATION DU PROGRAMME

Les variables sont définies mais encore faut-il que la carte Arduino en fasse quelque chose. Le sketch a été généré et modifié automatiquement chaque fois qu'une variable a été créée. La carte Arduino peut donc récupérer les valeurs de variable pour agir sur la carte shield. Par exemple, à partir de la consigne de vitesse, elle peut définir la direction du mouvement et la vitesse sous forme d'un rapport cyclique de PWM, puis injecter ce rapport cyclique sur la broche qui convient à la carte shield. Pour connaître cette broche, le mieux est d'aller voir la **documentation de la carte shield** sur le site d'Arduino : pour le canal A, cette PWM est sur la broche **D3**. Quant à la direction du mouvement, c'est l'état de la broche **D12** qui le détermine (HIGH ou LOW). C'est donc pour modifier l'état de ces broches qu'il faut rajouter quelques lignes de programme dans le sketch.

Le potentiomètre linéaire **définit à la fois direction et vitesse du mouvement**. Le point milieu du potentiomètre correspond à l'arrêt du train (avec une certaine plage). Si on bouge le curseur vers la gauche, on définit le sens **marche avant**, vers la droite, **marche arrière**. Or la position du curseur correspond à un nombre compris entre 0 et 1023 (potValue) : un petit traitement est nécessaire pour obtenir le rapport cyclique de la PWM réglant la vitesse et la direction du mouvement. La variable vitesse est égale à potValue – 512 (elle est donc comprise entre -512 et 511). Si vitesse est négative, le sens est en avant, si elle est positive, le sens est en arrière. On prend alors la valeur absolue et on la divise par deux pour rester dans une plage compatible pour une PWM (entre 0 et 255). Enfin, on limite la vitesse à 240 ; ainsi la valeur de PWM est toujours comprise entre 0 et 240 (vitesse maximale).

La **figure 9-3** montre comment modifier la fonction onPotValueChange.

La **figure 9-4** montre ce qu'il faut entrer dans la loop pour commander la carte shield.

```
/*
  Since PotValue is READ_WRITE variable,
  onPotValueChange() is executed every time a new
  value is received from IoT Cloud
*/
void onPotValueChange() {
  // Add your code here to act upon PotValue change
  vitesse = potValue - 512;
  if(vitesse) >= -50 && vitesse <= 50) {
    greenLEDstate = false;
    redLEDstate = false;
  }
  if(vitesse) < -50) {
    greenLEDstate = true;
    redLEDstate = false;
  }
  if(vitesse) > 50) {
    greenLEDstate = false;
    redLEDstate = true;
  }
  vitesse = abs(vitesse);
  if(vitesse <= 50) {vitesse = 0;}
  vitesse = vitesse /2;
  if(vitesse > 240) {vitesse = 240;}

  Serial.println(vitesse);
}
```

Figure 9-3

```
void loop() {
  ArduinoCloud.update();
  // Your code here
  analogWrite(3, vitesse);  // Set work duty
  if(greenLEDstate) == true) {
    digitalWrite(12, HIGH);  // Set direction
  }
  else {
    digitalWrite(12, LOW);  // Opposite direction
  }
  if(stopState == true) {
    digitalWrite(9, HIGH);  // Activate brake
  } else {
    digitalWrite(9, LOW);  // Release brake
  }
}
```

Figure 9-4

Ne pas oublier d'initialiser les broches 3, 9 et 12 en sortie dans le setup, et de mettre les variables greenLEDstate et redLEDstate à true dans la fonction onStopStateChange et vitesse à zéro (quand l'arrêt d'urgence est déclenché, on arrête la vitesse, on sert le frein (broche 9) et on allume les deux LED pour montrer qu'il y a un problème).

9.3 / LE TABLEAU DE BORD

Il reste maintenant à **créer un tableau de bord** (dashboard) pour contrôler notre objet connecté. Il suffit de choisir parmi les widgets proposés et les relier aux variables. On commence par choisir un afficheur pour la donnée vitesse, puis un switch (interrupteur) qui, mis sur ON, arrête la loco en urgence. On choisit un slider (potentiomètre linéaire) qu'on dispose horizontalement pour plus de visibilité et on règle ses valeurs extrêmes (de 0 à 1023). Enfin, deux LED virtuelles pour les deux voyants, une verte et une rouge, rappellent le sens de marche. Chaque widget **doit avoir son nom** et **être lié à une variable**. Le widget slider est appelé « Consigne » et est lié avec la variable « potValue », alors que les widgets LED sont appelés « Marche avant » et « Marche arrière » et sont liés avec les variables « greenLEDstate » et « redLEDstate ». La variable « greenLEDstate » permet de **régler la sortie D12** définissant le sens de marche. L'afficheur est appelé « Vitesse (PWM) » et est lié à la variable « vitesse » qu'il affiche. Enfin, le switch est appelé « Arrêt-URGENCE » et est lié à la variable « stopState » ; si le switch est sur ON, alors la vitesse est mise à 0 et les deux LED sont allumées (et la sortie D9 (Brake) est commandée). Les différents widgets peuvent être organisés comme bon vous semble et vous pouvez aussi voir comment votre dashboard se présentera sur votre smartphone. La **figure 9-5** montre l'élaboration du dashboard.

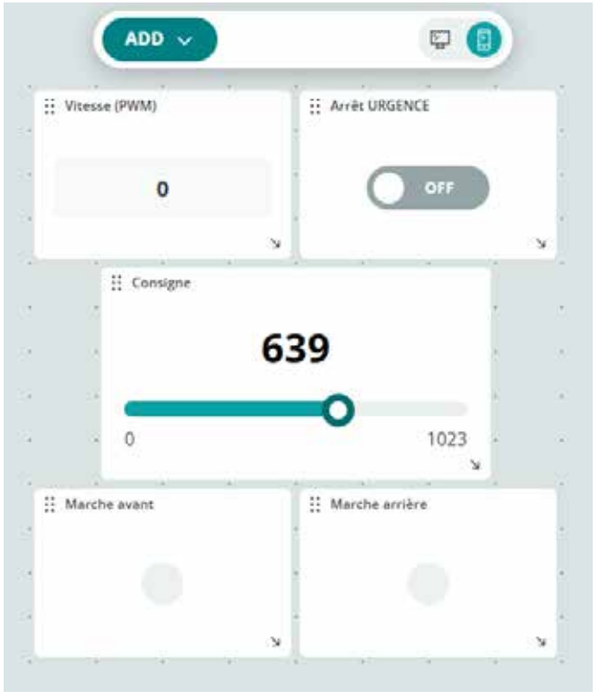


Figure 9-5

Tableau de bord d'un TGV

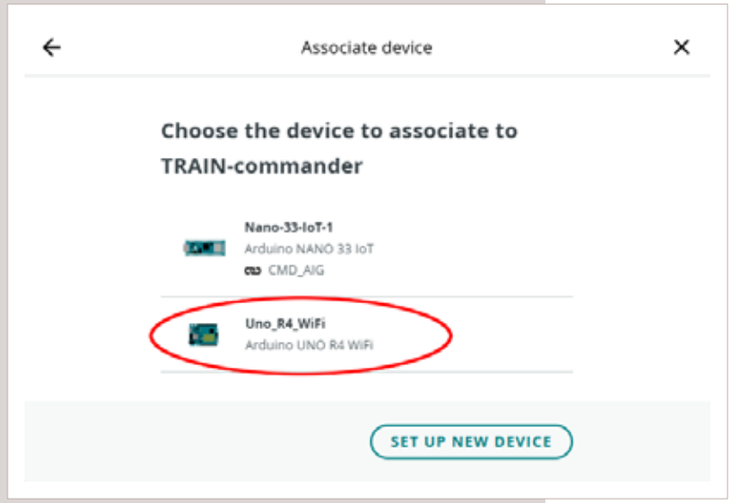


Figure 9-6

9.4 / FINALISATION DU PROJET

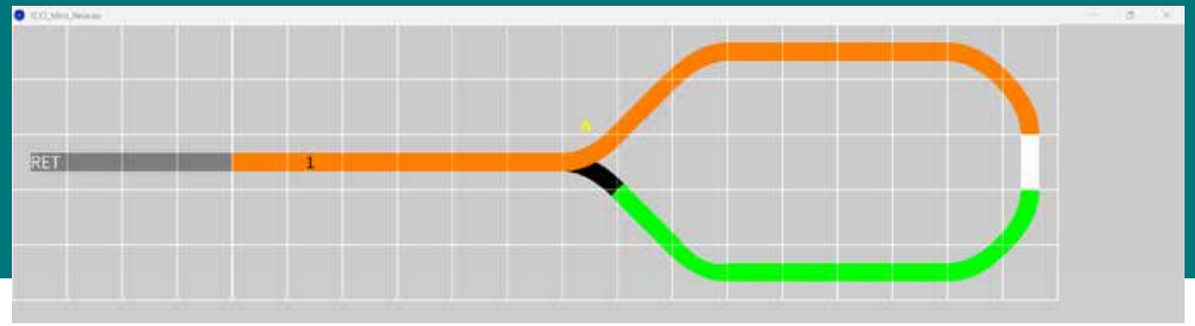
Pour que le projet fonctionne, il faut lui dire quelle carte est utilisée (parmi celles qui sont éligibles à l'IoT) et quel réseau le projet doit utiliser (par exemple, votre box internet). Ceci peut se faire via des menus graphiques comme le montre la **figure 9-6**.

Pour mettre au point ce projet, il faut,dans un premier temps, tester l'interactivité entre le dashboard et la carte Arduino. Lorsque tout semble fonctionner, il ne reste plus qu'à ajouter dans le programme le traitement des données par Arduino pour agir sur la carte shield. Malgré une apparente complexité, **cette commande de train a été mise au point en moins d'une journée** tellement l'IoT d'Arduino facilite la tâche.

RÉSUMÉ DU CHAPÎTRE 9

Contrairement aux apparences, créer un objet connecté est très facile avec l'IoT d'Arduino, peut-être même plus facile qu'écrire un programme de A à Z. L'avantage réside dans la simplification du câblage entre différents composants car l'information, au lieu de circuler dans des câbles, est transportée par ondes (OTA Over The Air). On peut donc imaginer une carte Arduino capable de repérer la position des trains, communiquant par WiFi avec une autre carte gérant un passage à niveau. L'ensemble des deux devient fonctionnel et cohérent. D'autres possibilités existent et avec les objets connectés, on entre dans la modernité alors pourquoi s'en priver ?

10 Projet de petit réseau commandé par la souris sur TCO virtuel



Le TCO virtuel de ce projet de mini-réseau

Le projet développé dans ce chapitre apporte un peu de modernité à un réseau analogique. Il s'agit de commander un mini-réseau à partir d'un TCO sur écran d'ordinateur en cliquant avec la souris.

10.1 / PRÉSENTATION DU MINI-RÉSEAU

Le mini-réseau (voir **figure 10-1**) est constitué d'une voie terminus comprenant un garage. Cette voie arrive sur une aiguille qui permet de partir, vers le nord ou vers le sud, sur une boucle de retournement permettant un retour vers le terminus. Ce mini-réseau va vous paraître bien fade, mais il permet de découvrir **comment gérer certains problèmes liés aux réseaux analogiques : inverser le sens du courant pour partir ou revenir au terminus, gérer le mouvement de l'aiguille avec un servomoteur, éviter le court-circuit sur la boucle de retournement, etc.** De plus, il peut servir à un projet de réseau de tramway avec une gare terminus desservant une petite ville, ce qui lui donnerait plus d'attrait.

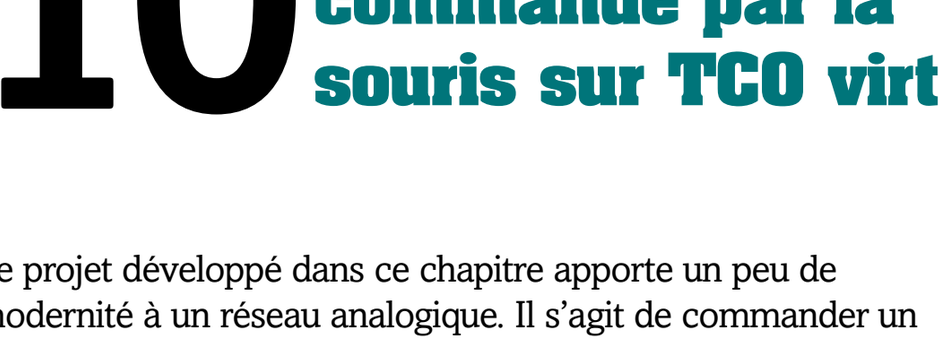


Figure 10-1

10.2 / LIAISON ARDUINO-RÉSEAU-ORDINATEUR

La **figure 10-2** montre la philosophie du projet : le TCO est dessiné sur l'écran d'un ordinateur, lui-même relié à une carte Arduino (un modèle UNO suffit amplement) via le câble USB. Cette carte Arduino commande le servomoteur de l'aiguille ainsi qu'une carte relais qui permet d'inverser le courant sur les voies pour obtenir le sens de marche voulue. La voie est alimentée par un classique transformateur à rhéostat (Alim voie).

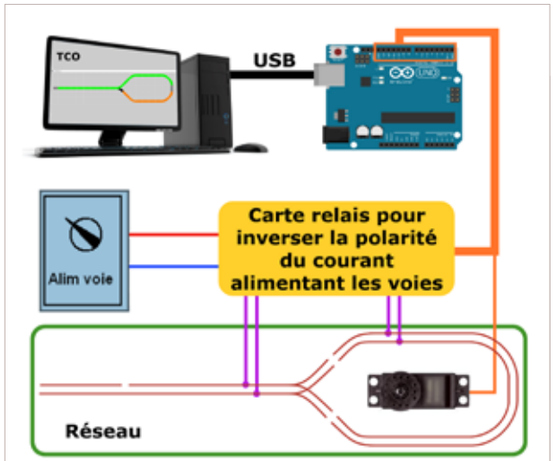


Figure 10-2
philosophie du projet

10.3 / TCO ET PROCESSING

Le TCO est dessiné avec Processing, un logiciel qui permet à des artistes de créer des images et qui est parfait pour travailler avec des cartes Arduino (en fait, les cartes Arduino ont été conçues pour utiliser Processing). La **figure 10-3** présente le dessin du TCO.

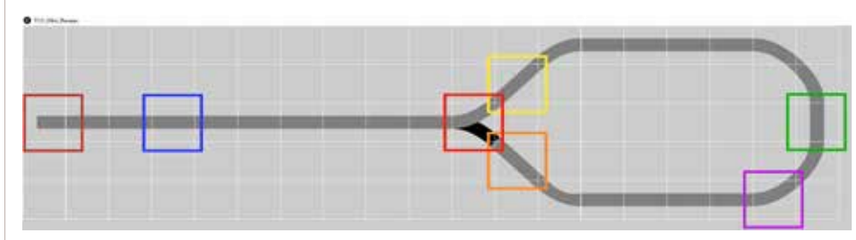


Figure 10-3

On voit que celui-ci est un **maillage de différents pavés** : pavé vide (gris), pavé butoir (marron), pavé droit horizontal (bleu) ou vertical (vert), pavé aiguille (rouge), pavé barre (jaune) et anti-barre (orange) et différents pavés arc (violet par exemple). En reprenant ces pavés, il vous sera possible de construire d'autres TCO pour des réseaux plus évolués.

La **figure 10-4** montre que le garage est isolé de la voie (coupure sur un rail) et shunté avec une diode. Le rail positif est en rouge et le négatif en bleu. Dans le sens 'arrivée', la diode est bloquée et le garage n'est pas alimenté donc **la motrice s'arrête**. Dans le sens 'départ', la diode devient passante, le garage est alimenté et **la motrice repart**.

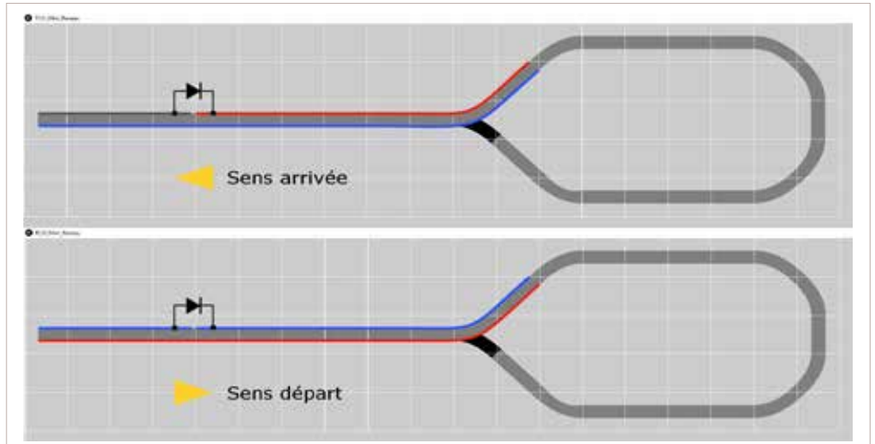


Figure 10-4

Le sens de circulation et la position de l'aiguille doivent être affichés, mais aussi transmis à la carte Arduino pour que celle-ci agisse soit sur la carte relais inverseur, soit sur le servomoteur.

Le logiciel Processing doit donc communiquer avec la carte Arduino. La **figure 10-5** montre comment le sens de circulation est affiché sur le TCO, grâce à deux couleurs (vert et orange).

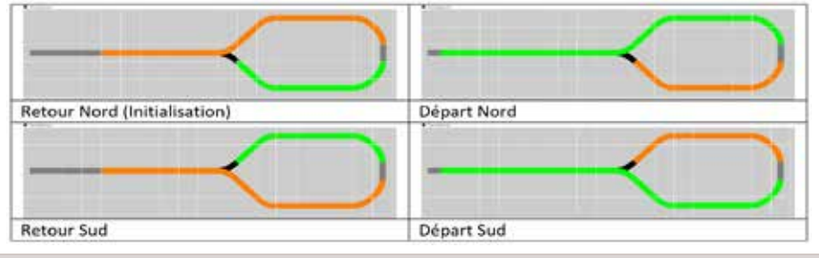


Figure 10-5

On peut remarquer sur la **figure 10-5** que le garage n'est pas alimenté (représenté en gris) dans le sens retour alors qu'il est alimenté dans le sens départ. La boucle de retournement doit être isolée de la voie principale pour éviter le fatal court-circuit, comme le montre la **figure 10-6**.

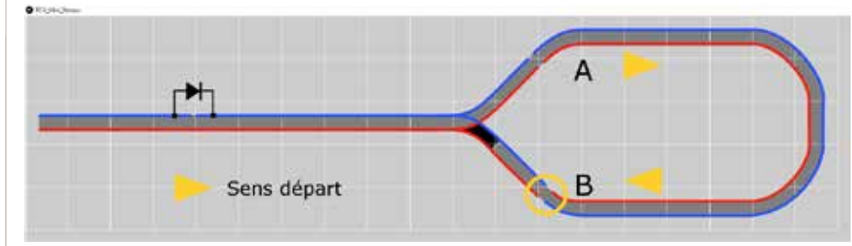
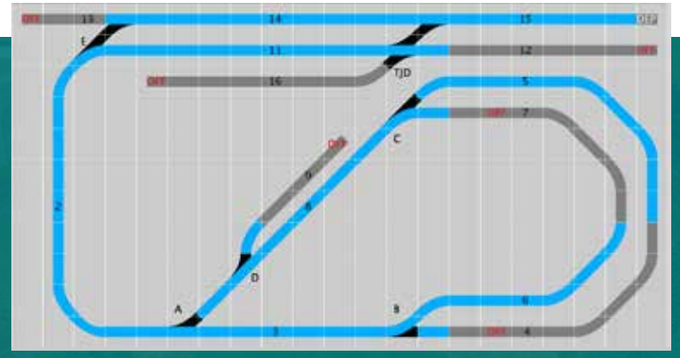


Figure 10-6

Le TCO du réseau de l'auteur selon les principes décrits dans ce chapitre



Le réseau de l'auteur en cours de construction, commandé par TCO sur écran et Arduino

10.4 / GESTION DE L'ALIMENTATION DES VOIES

Les alimentations des voies doivent donc être gérées par la carte relais qui est commandée par Arduino. **C'est un I.L.S qui envoie l'information à la carte Arduino qu'il est temps d'inverser le courant** : lorsqu'il est survolé, le train est entièrement sur la boucle et on peut inverser le courant sur la voie principale pour effectuer le retour. La **figure 10-7** montre le montage de tous les composants. Nous voyons en haut le réseau avec la diode et l'ILS, le servomoteur de l'aiguille avec son bouton poussoir qui permet de choisir entre trajet nord et trajet sud (en plus du fait de cliquer sur le TCO), la carte constituée de deux relais DPDT et d'un ULN2803A pour amplifier les signaux d'Arduino et enfin, trois LED de contrôle, le tout étant commandé par la carte Arduino. Les fils en vert envoient le courant de traction sur la voie principale et les fils en violet sur la boucle de retournement.

La commande de ce mini-réseau se fait en cliquant soit sur l'aiguille, ce qui la place dans la bonne position pour un départ vers le nord ou vers le sud, soit sur le mot RET (sur le butoir) qui signifie 'retour' ; RET est alors remplacé par DEP qui signifie 'départ' et la polarité des voies est changée.

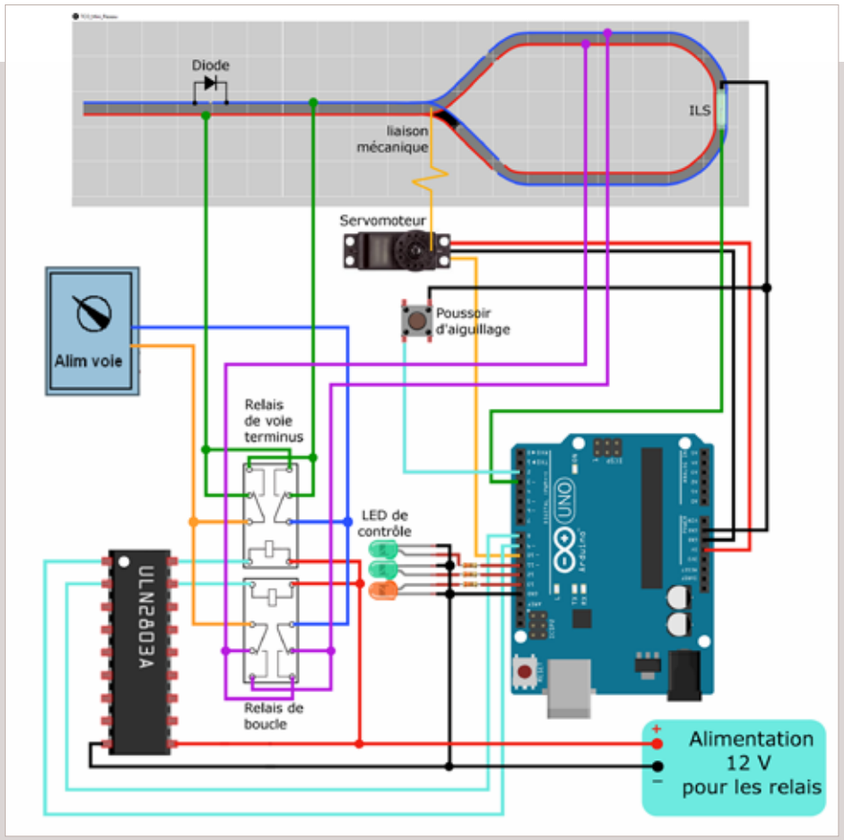


Figure 10-7

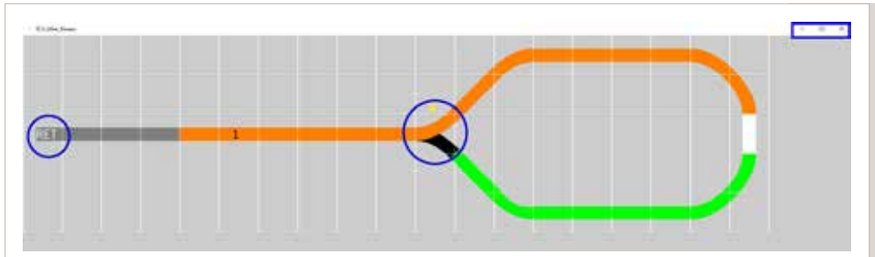


Figure 10-8
en bleu, les zones où on clique

10.5 / PROGRAMMES POUR ARDUINO ET POUR PROCESSING

Des explications supplémentaires sont ici : **https://locoduino.org/spip.php?article226** . Je ne vais pas décrire en détail les programmes qui sont téléchargeables sur LOCODUINO sous forme d'un fichier ZIP. Bien évidemment, il faut installer Processing sur votre ordinateur ; la version 4.3.3 est conseillée (on peut la trouver sur GitHub) car la dernière version 4.4.1 peut présenter un problème d'affichage.

Une fois les dossiers extraits, vous obtenez un dossier Arduino et un dossier Processing ; le dossier Arduino contient un répertoire « **TCO_Mini_Reseau** » contenant le programme '.ino'. C'est lui qu'il faut téléverser dans votre carte Uno, après avoir modifié les lignes 155 et 156 qui dépendent de la géométrie de l'aiguille. Le dossier Processing contient également un répertoire « TCO_Mini_Reseau » contenant le programme principal Processing « TCO_Mini_Reseau.pde » et tous les autres onglets dessinant les différents pavés. Dans le programme principal, vous devez modifier la ligne 14 **pour indiquer le port sur lequel est connectée votre carte Uno**. Appuyez sur la flèche (lecteur) pour exécuter le programme : le TCO s'affiche dans une fenêtre (**figure 10-8**) et vous pouvez jouer avec (les zones en bleu sont celles où on peut cliquer). La ligne 60 permet de redimensionner la fenêtre d'affichage.

Maintenant que vous avez vu le fonctionnement de ce TCO, il est **temps de construire le réseau**. Posez les voies sans oublier la diode du garage, les éclisses isolantes de la boucle de retournement et l'I.L.S. Motorisez l'aiguille avec un kit 6171 de chez Decapod (en vente chez LR-Modélisme). Fabriquez une petite interface avec 2 relais DPDT (pas de type verrouillable, mais de type monostable) et un CI ULN2803A : prévoyez une alimentation de même voltage que les relais. Rajoutez la carte Arduino (programmée), le poussoir et les LED de contrôle. Tout doit être câblé comme sur la **figure 10-7**.

Voilà, votre mini-réseau peut être commandé **avec la souris ou avec le doigt si vous avez un écran tactile**, car cela fonctionne aussi.

RÉSUMÉ DU CHAPÎTRE 10

L'avantage d'un TCO virtuel sur écran d'ordinateur (ou de tablette), c'est qu'il est meilleur marché qu'un TCO électromécanique et qu'il est évolutif car ajouter des voies ou des fonctions ne demandent qu'un peu de programmation. Avec ce mini-réseau, nous avons appris à traiter les éléments constitutifs d'un réseau analogique : commande de la voie et de sa polarité, commande du moteur d'une aiguille, représentation des informations. Cela simplifie considérablement le câblage puisqu'il suffit d'un simple câble USB entre ordinateur et Arduino pour afficher l'état du réseau. J'ai utilisé ce principe pour mon réseau (voir illustration) et pour EX_MACHINA, comme vous l'avez vu au chapitre 3, où le TCO affiche même la signalisation lumineuse.

11 Et pour aller encore plus loin ?

Le but de ce guide est de vous emmener le plus loin possible dans la maitrise des cartes à microcontrôleur et les chapitres précédents ont bien rempli ce rôle. Mais le chemin ne s'arrête jamais et il y a toujours un domaine à explorer. C'est ce que je vais vous démontrer maintenant tout en vous proposant un voyage fictif vers un futur qui ne l'est peut-être pas tant que cela.

11.1 / S'INTÉRESSER À CE QU'IL Y A SOUS LE CAPOT

Il n'est pas question, dans le cadre de ce guide, de rentrer dans les détails mais il n'est pas inutile de donner quelques points de repère de ce qui constitue un microcontrôleur (ou micro, noté µC). Vous comprendrez pourquoi un programme écrit pour une carte ne fonctionne pas avec une autre. Peut-être même aurez-vous envie d'approfondir vous-même certains des points que je vais aborder.

Pour comprendre la structure d'un microcontrôleur, on peut comparer ce dernier à une maison. Il y a de petites maisons (µC à 8 bits) et de grandes maisons (µC à 32 bits), mais ce qu'on trouve à l'intérieur est toujours la même chose : une cuisine où on va préparer les repas en suivant une recette, un salon ou une salle à manger, des chambres, une ou plusieurs salles de bain, des placards pour ranger ses affaires, d'autres pièces spécialisées (buanderie, atelier, garage). La maison a aussi des portes et des fenêtres qui communiquent avec l'extérieur, et une horloge bien placée pour que chacun puisse voir le temps qui passe. Le micro, c'est un peu pareil. Il a aussi une **horloge** qui délivre des signaux carrés et chaque créneau donne le rythme de travail. Il a des placards qui lui servent à ranger des informations, ce qu'on appelle **mémoire**, une cuisine qui lui sert à effectuer sa tâche en suivant une recette appelée programme avec des ingrédients bien rangés dans des placards, appelés **instructions ou données**. Le micro a également des **unités spécialisées** qui servent à certaines tâches bien spécifiques, tout comme une buanderie peut être équipée d'une machine à laver le linge (qui rend le linge propre) et d'un sèche-linge (qui rend le linge sec). Enfin, pour communiquer avec l'extérieur, le micro a aussi des portes appelées **PORT**, chacun étant constitué d'un ensemble de lignes d'entrée-sortie ; lorsque la porte d'une maison est ouverte, c'est pour laisser entrer ou sortir quelqu'un, et de la même façon, **un PORT est soit une entrée soit une sortie pour une information**. De même qu'une maison peut avoir une porte principale et des portes fenêtres qui donnent sur le jardin, le micro a **plusieurs PORT qui communiquent avec l'extérieur** (capteurs ou actionneurs). On va maintenant voir en détail chacun des sous-ensembles qui constituent un microcontrôleur, tout en gardant à l'esprit notre analogie avec une maison.

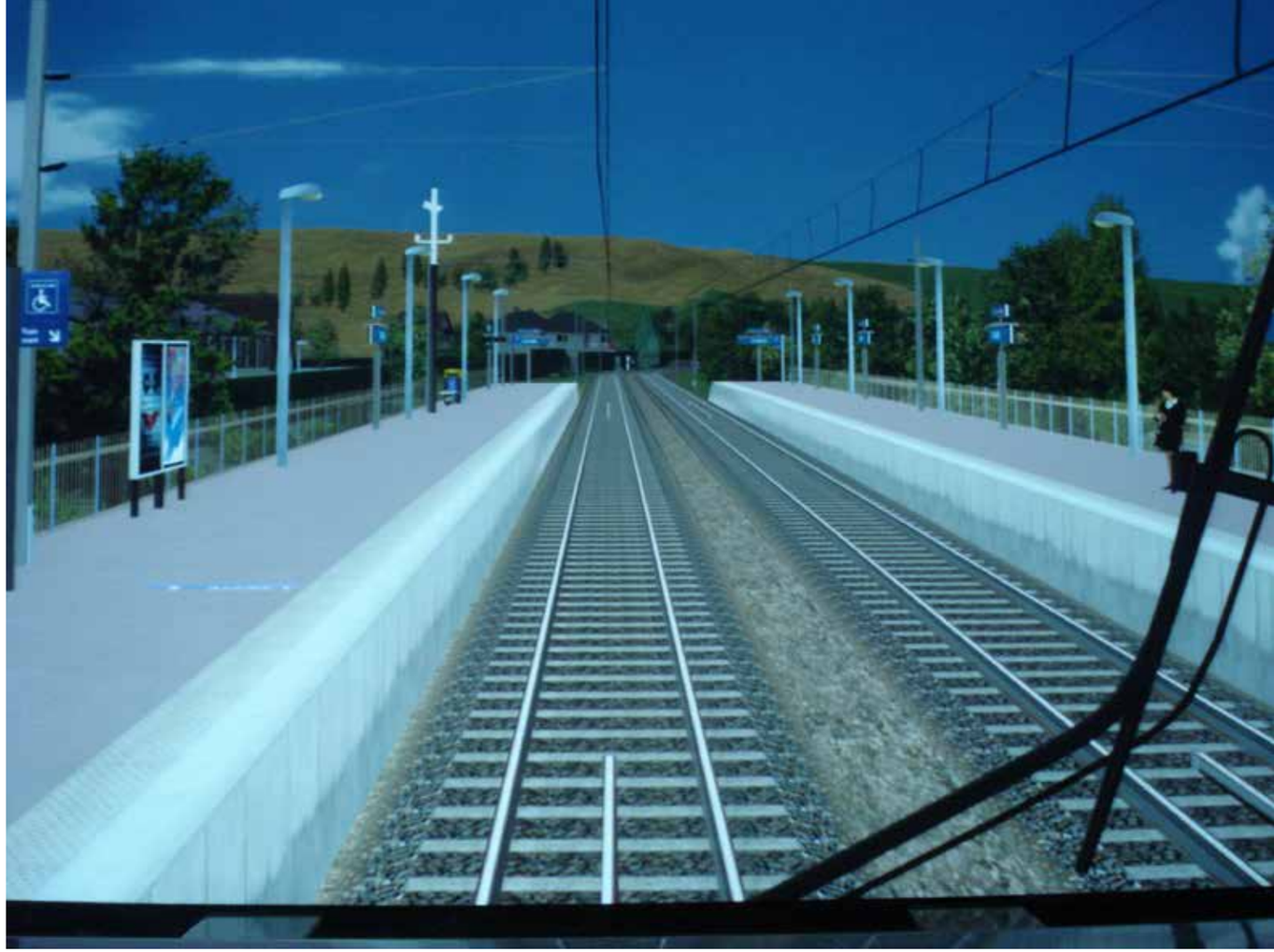


Image de conduite d'un simulateur réel à la SNCF

Un microcontrôleur est constitué de plusieurs unités, chacune spécialisée dans une tâche, et travaillant ensemble (**figure 11-1** reprenant la structure de l'ATmega328P). L'**ALU** (Arithmetic and Logic Unit) est l'unité qui est capable d'effectuer des **calculs arithmétiques et logiques** sur les octets de données (addition, soustraction, incrémentation, ET, OU, NOT, etc.). C'est le programme qui indique à l'ALU quelles opérations effectuer et sur quelles données. L'ALU est en quelque sorte la cuisine, car c'est là qu'on travaille pour préparer un plat (une tâche) avec des ingrédients (**variables**) en suivant une recette (**programme**).

Le programme étant constitué d'instructions, il faut stocker ces dernières ainsi que les données. Pour ranger ses affaires dans une maison, on dispose de placards qui peuvent avoir plusieurs formes (penderie, commode, etc.). De même, les affaires de Madame ne sont généralement pas au même endroit que les affaires de Monsieur, mais cette règle est un peu sexiste et on pourrait tout à fait mettre tous les vêtements dans la même penderie. Pour un micro, les affaires sont les informations qui sont de deux natures : **instructions et données**. Pour les stocker, c'est le rôle de la mémoire qui peut être **permanente ou bien volatile**. Le micro a une architecture **Von Neumann** si les données et les instructions sont stockées dans un **unique espace mémoire** (lui-même constitué de mémoire permanente ou volatile) : il n'y a donc qu'un seul bus d'adresse permettant de localiser le bon octet au bon endroit. Le micro a une architecture **Harvard** si la **mémoire pour les instructions est séparée de la mémoire pour les données** : c'est le cas pour l'ATmega328P des cartes Uno et il y a deux bus d'adresses puisque les espaces de stockage sont différents. Les données sont stockées en **mémoire volatile (SRAM)** alors que les instructions sont stockées en **mémoire permanente (Flash memory)** pour ne pas avoir à recharger le programme à chaque utilisation.

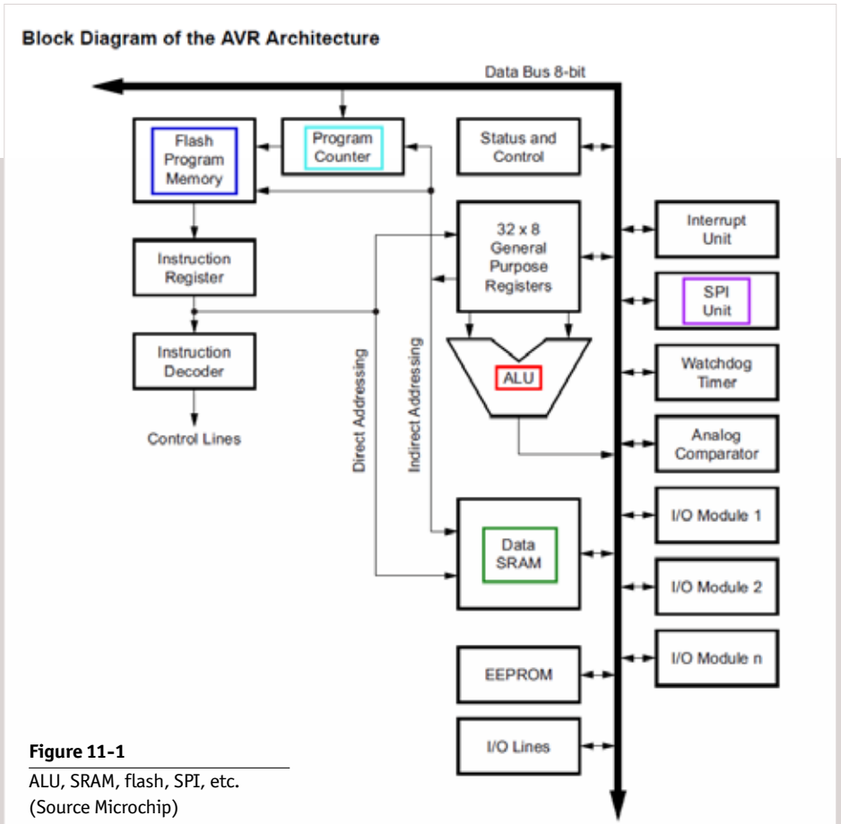


Figure 11-1
ALU, SRAM, flash, SPI, etc.
(Source Microchip)

Le **program counter** est un compteur qui permet de repérer l'instruction en cours, un peu comme si nos recettes de cuisines avaient des lignes numérotées. Lorsqu'une instruction a été exécutée, le program counter est incrémenté d'une unité et on va chercher l'instruction suivante pour l'exécuter elle aussi. En cas de saut vers un sous-programme, **il faut le sauvegarder** de manière à reprendre le programme principal (là où on s'était arrêté) lorsque le sous-programme est entièrement exécuté.

Une maison dispose d'une sonnette, et lorsque le facteur sonne, vous interrompez ce que vous êtes en train de faire pour aller réceptionner le courrier, et lorsque c'est fait, vous reprenez ce que vous étiez en train de faire. L**e micro aussi peut être interrompu dans sa tâche pour aller exécuter une tâche encore plus urgente** ; c'est ce qu'on appelle une **interruption** et dans ce cas, il faut aussi sauvegarder le program counter ainsi que d'autres registres importants. Parfois, on a envie d'être tranquille et on inhibe la sonnette de la maison ; on peut faire la même chose et **inhiber les interruptions** si la tâche qu'effectue le micro ne doit pas être interrompue (respect d'un timing précis par exemple). Tout cela est à régler dans les différents registres du micro et il faut donc apprendre à les connaître.

Comme un micro doit communiquer avec l'extérieur, **ces broches sont organisées en PORT** avec différents registres pour les contrôler : ce sont les portes fenêtres de notre maison avec les serrures pour les ouvrir ou les fermer. Tant qu'on utilise les fonctions d'Arduino (digitalWrite par exemple), le logiciel IDE sait que telle broche correspond à tel registre et le programmeur n'a à s'occuper de rien. Mais on peut aussi travailler directement avec les registres des PORT, ce que j'avais fait dans le tome 1 pour la croix de pharmacie. Et comme les PORT ne sont pas organisés de la même façon d'un micro à un autre, **un programme qui fonctionne sur Uno ne fonctionnera plus sur Mega** ! Pour la croix de pharmacie, j'utilisais le PORTD de la carte Uno qui correspond aux broches 0 à 7. Sur la carte Mega, les broches 0 à 7 font partie des PORT E, G et H, comme le montre la **figure 11-2** tirée de la documentation « pinout » des cartes.

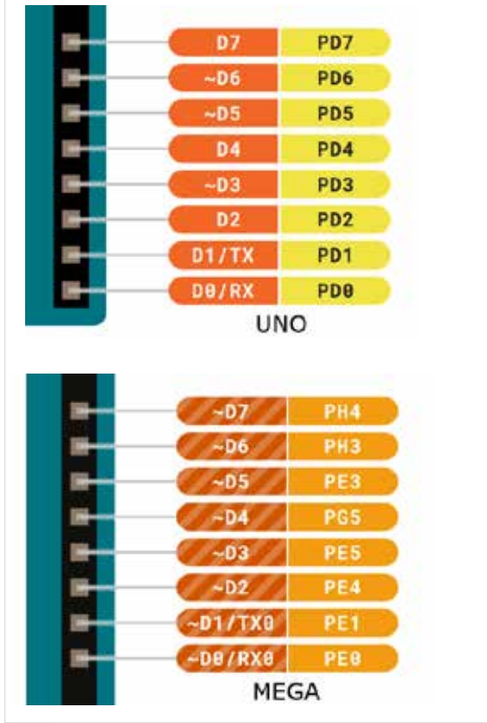


Figure 11-2

Un micro possède aussi des **timers** pour compter le temps (tout comme il y a plusieurs pendules dans une maison) et on peut les programmer en écrivant directement dans les registres qui leur sont associés. Les timers ont la propriété d'effectuer leur **comptage du temps indépendamment du programme, en tâche de fond**. Mais d'un micro à l'autre, les timers ne sont pas les mêmes, ce qui peut encore expliquer qu'un programme pour une carte ne fonctionne pas sur une autre. Pour éviter ces désagréments, le mieux est soit de respecter les fonctions d'Arduino, soit de passer

par des bibliothèques. Une bibliothèque n'est pas pour autant universelle : quand on l'utilise, on doit se renseigner sur les cartes qu'elle prend en charge. Par exemple, Servo ne fonctionne pas sur une carte ESP et il faut charger la bibliothèque ESP32Servo, comme on l'a vu un peu plus haut. Certaines fonctions (ou bibliothèques) d'Arduino utilisent les timers, ce qui peut inhiber d'autres tâches : **il faut bien regarder la documentation des fonctions ou bibliothèques**. Par exemple, Servo empêche de produire de la PWM sur les broches 9 et 10, qu'un servomoteur y soit branché ou non.

Un timer a un rôle particulier : le **WDT pour Watch Dog Timer ou encore chien de garde**. Dans un programme, on peut l'utiliser ou pas, et si on l'utilise, son rôle est d'éviter qu'un programme soit bloqué parce qu'il tourne en rond (boucle sans fin, blocage dû à un parasite). Le principe est simple, le WDT décompte son compteur et s'il arrive à zéro, il effectue un reset de la carte. Pour que le programme fonctionne normalement, il faut relancer la minuterie régulièrement pour l'empêcher d'arriver à zéro. Dans une boucle sans fin, on ne relance plus la minuterie donc le WDT finit par arriver à zéro : un reset a lieu et le programme peut redémarrer et s'exécuter normalement.

D'autres unités spécialisées existent au sein du micro (l'équivalent de la buanderie d'une maison), **par exemple pour communiquer** (SPI, I2C, UART, WiFi, Bluetooth, etc.) ou **pour produire de la PWM** ou encore pour **transformer un signal analogique en numérique ou réciproquement**. Pour chaque unité, on trouve des registres de contrôle dont il faut connaître l'usage. Le mieux est de se plonger dans la datasheet, mais certaines font plusieurs centaines de pages. Tout cela ne se justifie que pour des besoins très particuliers : en modélisme ferroviaire, l'utilisation des fonctions d'Arduino suffit amplement, et si ma croix de pharmacie a utilisé le PORTD, c'est juste pour rendre le programme plus lisible (je gagne aussi en temps d'exécution mais cela ne se voit pas à l'œil nu).

Une dernière chose à avoir en tête, c'est que la mémoire d'un microcontrôleur n'est en rien comparable à la mémoire d'un micro-ordinateur. On ne parle pas en Giga-octets et il faut donc **apprendre à économiser cette mémoire** ; cela peut se faire en choisissant le bon type de variable ou en stockant les chaînes de caractères en mémoire programme (PROGMEM). De plus, la mémoire qui sert à stocker les variables sert aussi au microcontrôleur qui sauvegarde certaines données lors des appels de subroutines (interruptions par exemple). C'est ce qu'on appelle **la pile** et si nos précieuses données viennent l'écraser, c'est le dysfonctionnement assuré du programme et ce n'est pas toujours facile à comprendre.

Ce petit tour d'horizon nous a permis de comprendre que **la puissance d'un microcontrôleur est due à plusieurs facteurs** : sa taille en bits, sa fréquence d'horloge, la capacité de ses mémoires sont des éléments essentiels. Mais il faut aussi prendre en compte d'autres éléments comme les unités spécialisées qui sont à choisir en fonction de l'application qu'on veut obtenir (numérisation de signal, communication CAN ou I2C, nombre de broches pour communiquer avec l'extérieur, présence du WiFi, etc.). Les premières questions à se poser concernent donc le projet de modélisme ferroviaire qu'on a en tête et à partir de là, on se pose la question du µC qui peut le mieux convenir : c'est généralement celui qui dispose des interfaces utiles au projet (CAN, WiFi par exemple) car la vitesse n'est pas forcément un élément bloquant pour nos trains miniatures. Mais il faut aussi tenir compte des logiciels délivrés avec le µC, notamment les bibliothèques qui vous permettront de faire fonctionner des actionneurs particuliers à votre projet.

11.2 / APPRENDRE D'AUTRES LANGAGES DE PROGRAMMATION

Il existe d'autres langages de programmation que le langage Arduino qui est lui-même du langage C ou C++. On a évoqué **Processing** dans le chapitre 9 et comme l'environnement de Processing ressemble à l'environnement d'Arduino, vous ne devriez pas être trop perdu à consacrer un peu de temps à apprendre ce langage. Cela permet de faire des **interfaces graphiques à un programme**, comme par exemple un TCO pour un réseau, ou bien une interface de commande. Cette dernière peut aussi être faite avec le langage HTML (Hyper Text Markup Language) et vous trouverez sur le site de LOCODUINO quelques exemples à ce sujet (le HTML complétant le programme en C/C++).

En 2024, le langage **Python** était le plus utilisé dans le monde. Les cartes Arduino n'ont pas échappé à ce phénomène et peuvent maintenant être programmé en **MicroPython**, à condition d'être compatibles avec ce langage. Le Python est un langage interprété et non compilé, donc un peu plus lent : pour le train, cela n'a pas une grosse incidence. On peut trouver de nombreuses applications en MicroPython et s'en servir. Enfin, **le Python est le langage idéal pour commencer la programmation car il est plus simple dans sa syntaxe** ; il n'y a pas besoin de définir le type de variable par exemple, et il n'y a pas besoin de terminer une ligne par un point-virgule, le retour chariot suffisant. On trouve des boucles for ou while et des tests if-else (comme en C/C++) et il est aussi possible de définir des fonctions. Les fortes similitudes avec le C/C++ font que la connaissance d'un des deux langages permet d'acquérir facilement le deuxième sans être trop perdu. Je vous encourage donc à essayer, ne serait-ce que pour varier les plaisirs.

Enfin, si on recherche la rapidité d'exécution et la compacité du code, **l'assembleur est le langage idéal car c'est celui qui suit au plus près l'intimité du microcontrôleur**. Hélas, la rédaction des programmes et leur mise au point est un **véritable casse-tête**. Si vous vous lancez dans l'aventure, attendez-vous à de nombreuses heures à chercher les bugs. Comme je l'ai dit dans une série d'articles publiée sur LOCODUINO (**https://locoduino.org/spip.php?article280**), l'assembleur n'apporte pas grand-chose dans notre hobby. Dans certains cas cependant, l'assembleur peut avoir une utilité comme, par exemple, respecter des timings très précis ou gagner en vitesse d'exécution. Quant à la compacité du code, faire clignoter une LED avec le programme Blink demande 924 octets de programme, alors qu'en assembleur, **il n'en faut que … 30 !** Malgré ces avantages, l'assembleur a beaucoup de défauts : vous devez penser à tout car personne ne le fait à votre place. Il faut parfaitement connaître le microcontrôleur et son architecture. Il faut un logiciel pour générer le code (par exemple MicrochipStudio 7) et il est préférable d'avoir un programmeur (on en trouve chez Microchip, anciennement Atmel). Il faut aussi apprendre à utiliser tout cela. Bref, c'est beaucoup de temps consacré pour une cause qui ne se justifie pas dans notre domaine. À vous de voir …

11.3 / IMAGINER LE TRAIN MINIATURE DE DEMAIN

Le train miniature a beaucoup évolué ces vingt dernières années : on est passé de l'analogique au numérique. Pourquoi ? Et bien déjà pour résoudre certains problèmes propres à l'analogique. Par exemple, pour garer un engin moteur sur une voie, il faut créer une section de garage non alimentée. En conséquence, cela complique le câblage du réseau. En plus, on ne peut pas faire circuler sur une même voie deux locomotives indépendamment l'une de l'autre : impossible donc d'envoyer une locomotive en chercher une autre pour s'y atteler et créer une unité multiple. Enfin, ce qui est garé en analogique ne reçoit plus de courant donc pas d'éclairage des feux ou des voitures, sauf à ajouter un système électronique qui ajoute du courant de haute fréquence sur la voie. **Tout cela a été résolu par l'arrivée du numérique** : les locomotives sont commandées indépendamment les unes des autres, la voie est constamment alimentée ce qui permet de garder l'éclairage, et au final il n'y a que deux fils à relier aux rails pour que cela fonctionne. Oui, c'est plus cher mais cela permet plus de choses. Mais si on veut que des trains se suivent en toute sécurité, alors il faut recréer des cantons, donc un câblage plus compliqué. Le numérique n'a pas résolu le court-circuit des boucles de retournement mais un module peut s'en occuper. Enfin, la captation de courant se fait mieux car la voie, étant alimentée en alternatif, s'encrasse moins qu'en analogique, mais il reste tout de même des problèmes de captation, notamment sur les aiguilles pour des engins courts, ce qui oblige à polariser la pointe de cœur : il faut s'y connaître et encore une fois, cela complique le câblage. Pour repérer la position des trains sur le réseau, il faut des capteurs ou bien un logiciel coûteux qu'il faut calibrer.

Le numérique a donc résolu beaucoup de choses mais **des problèmes, il en reste encore !** Le train du futur sera imaginé justement pour résoudre les problèmes restants. La meilleure solution pour résoudre les problèmes de câblage du réseau, le court-circuit des boucles de retournement et les problèmes de captation dus à l'encrassement ou aux appareils de voie, c'est d'avoir **une batterie à bord du train** (concept « dead rail » aux USA). Plus besoin de soigner la pose de la voie, elle ne sert qu'à guider les roues. Plus de problème de câblage, la voie n'est pas alimentée. Plus de court-circuit, on peut dessiner le réseau qu'on veut. Bien sûr, on rajoute le problème de la recharge des batteries, mais celles-ci ont fait de gros progrès et en feront encore : les batteries de demain auront **de plus en plus de capacité et le temps de recharge sera abaissé**. C'est déjà le cas des **batteries à graphène**. Cette recharge pourra d'ailleurs se faire par induction, lorsque le train fait une halte par exemple ou lorsque le réseau ne sert pas.

Mais comment les décodeurs recevront-ils les ordres ? Et bien avec des techniques qui ressemblent à la **radio** (cela existe déjà) ou au **WiFi** (objet connecté) ou encore au **Bluetooth**. Une transmission **OTA (Over The Air)** permettra d'ailleurs un bien plus gros débit que celui du numérique d'aujourd'hui. Avec l'ordre reçu sans fil et l'énergie déjà à bord, les décodeurs fonctionneront de la même manière et permettront sans doute encore plus de choses qu'aujourd'hui.

Mais il faudra toujours des capteurs pour localiser les trains ? Pas forcément, si on utilise le **GPS RTK (Global Positioning System Real Time Kinematic)** qui permet une précision de l'ordre du cm. Avec cette précision, on sait exactement où se trouve le train sur le réseau. Aujourd'hui, ces techniques coûtent encore cher mais elles se démocratisent chaque jour un peu plus et on trouve déjà sur YouTube des tutos pour le GPS RTK. La communication des décodeurs sera très certainement **dans les deux sens**, ce qui fait que le train pourra envoyer à la centrale sa position et celle-ci pourra l'afficher sur un écran de TCO et

Un projet n'est pas toujours simple : ici le Viaduc de Garabit

aubiner la signalisation lumineuse. On peut donc aussi imaginer que **la centrale DCC du futur fera plus de choses que certains logiciels de contrôle de réseau actuels, pour un prix inférieur**.

Enfin, il est à parier, vu la miniaturisation qui existe déjà à l'heure actuelle, que toutes les locomotives seront **équipées d'une micro-caméra à l'intérieur des postes de conduite**, et que l'image de notre propre réseau sera renvoyée sur notre smartphone ou tablette. La conduite se fera alors comme si on avait pu rentrer dans notre modèle réduit, ce qui sera plus intéressant que regarder les trains passer comme on le fait actuellement. Les aéromodèles et les drones permettent déjà cette fonctionnalité, ce qui a donné un nouvel engouement pour le pilotage des modèles réduits.

Tout cela est-il concevable ? Il y a une vingtaine d'années, les aéromodélistes volaient en thermique avec des moteurs capricieux, polluants et bruyants. Et si quelqu'un évoquait le vol en électrique, il se faisait railler car les batteries étaient à l'époque trop lourdes. Aujourd'hui, quasiment tous les aéromodélistes volent en électrique car batteries et moteurs ont fait des progrès et pourtant la consommation de ces derniers est bien supérieure à la consommation d'un train à l'échelle H0. Concevable est donc le terme qui convient, mais qui connaît l'avenir ? Pour ma part, j'y crois fortement et comme l'avenir se construit dès aujourd'hui, alors il me reste du pain sur la planche. Un peu d'aide de votre part serait la bienvenue !

RÉSUMÉ DU CHAPÎTRE 11

Plus vous programmerez des microcontrôleurs et plus vous aurez envie de connaître leurs secrets. La meilleure façon de progresser, c'est de ne pas s'en tenir à ce qu'on connait mais de découvrir de nouveaux langages et de nouvelles techniques. Quant au train miniature du futur, on peut se contenter d'attendre sa venue ou bien précéder le mouvement et participer à son élaboration. Certains s'y sont déjà mis.

Conclusion

Mon but est atteint et je vous ai ouvert des portes : aurez-vous la curiosité d'aller voir ce qui se cache derrière ? Apprendrez-vous à concevoir vos propres circuits imprimés et à les faire construire ? Aurez-vous le courage d'apprendre le langage Processing pour dessiner votre TCO ? Prendrez-vous l'habitude de regarder la datasheet du microcontrôleur qui équipe votre dernière carte Arduino ? Utiliserez-vous l'Intelligence Artificielle pour vous aider dans vos projets ?

Il y a tant de possibilités ouvertes à ceux qui veulent progresser. Je fais de l'Arduino depuis maintenant 12 ans et je n'ai pas fini d'apprendre. J'en sais juste un peu plus qu'à l'époque où j'ai commencé. Mais mon enthousiasme est toujours intact ; il reste tant à découvrir. J'espère vous avoir transmis une partie de ce qui m'anime, et que vous aurez envie vous aussi de découvrir tout cela. Le train miniature est un formidable terrain de jeu pour ceux qui veulent créer des automatismes, et les électroniciens l'ont compris depuis longtemps et ont toujours essayé de faire mieux et plus réaliste. Avec les techniques d'aujourd'hui, c'est à la portée de tout le monde, c'est à votre portée !

Merci de m'avoir lu et bon modélisme ferroviaire.

LR PRESSE

2, rue de Suède
BP 30104
56401 AURAY CEDEX
www.lrpresse.fr
accueil@lrpresse.fr

Conception graphique
Antoine SIMON (LR Presse)

Mise en page
Pierre-Frank SEGUINEAU

Impression
Spektar (Bulgarie)

ISBN
978-2-37536-061-3

Dépôt légal
1e trimestre 2023

MAJ infos

Retrouvez-nous sur
trains.lrpresse.com

nouvelles couv?

texte 4eme

LES GUIDES PRATIQUES DU TRAIN MINIATURE

AUTOMATISEZ VOTRE RÉSEAU

Concevoir un projet avec Arduino et l'Intelligence Artificielle

Commander la signalisation lumineuse de votre réseau en fonction de la position des trains ou des aiguilles, utiliser des servomoteurs pour commander vos barrières de PN, des écrans LCD pour contrôler un automatisme, des moteurs pas à pas pour commander une plaque tournante, et bien d'autres choses encore… Telle est l'ambition de ce guide pratique : vous faire démarrer dans ce domaine en proposant de nombreux montages pour les réseaux de trains miniatures, montages relativement simples puisque ce guide s'adresse essentiellement à des débutants. Après quelques considérations théoriques sur l'électronique programmable et ce qui constitue l'univers des cartes à base de microcontrôleur (Arduino ou autre), vous trouverez dans ce guide des explications sur le principe de fonctionnement ainsi qu'un plan de câblage des composants. Peu importe que votre réseau soit analogique ou numérique, la plupart des montages fonctionneront sur l'un comme sur l'autre. Seuls quelques montages sont spécifiques à l'analogique et ils sont repérés dans ce cas. La dernière partie du guide est faite pour vous permettre d'aller plus loin, en proposant des

09

LR PRESSE

Les Guides Pratiques du train Miniature - **Automatisez votre réseau**

code barre / ISBN

09

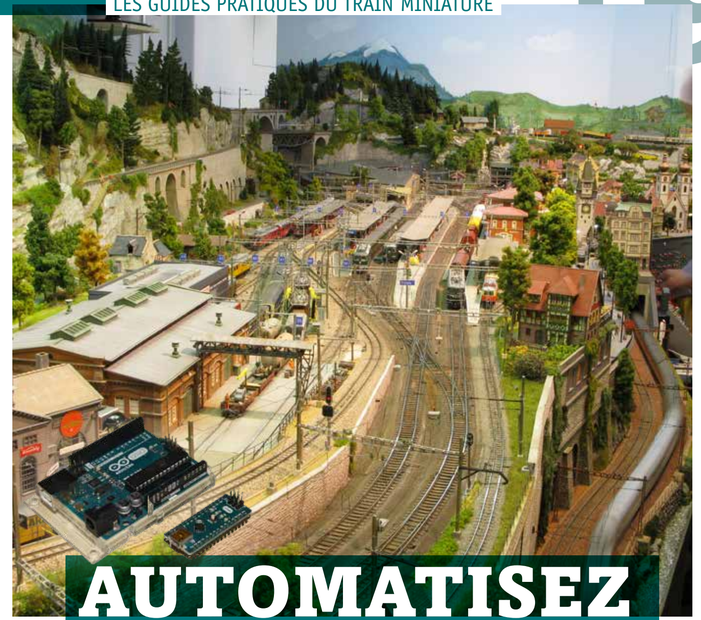
LES GUIDES PRATIQUES DU TRAIN MINIATURE

AUTOMATISEZ VOTRE RÉSEAU

LR PRESSE

LES GUIDES PRATIQUES DU TRAIN MINIATURE

09



AUTOMATISEZ VOTRE RÉSEAU

Concevoir un projet avec Arduino et l'Intelligence Artificielle

09

Éditions LR Presse

texte rabat

Chez le même éditeur :

Chemins de fer réels

Les 141 TA et le dépôt d'Ussel, par P. Ratinaud, 2017
Conducteur de train à la SNCF, mon métier, ma vie, par X. Acker, 2017
Les Carnets de Robert Nobécourt, 2018-2020, R. Nobécourt
Toutes les lignes et gares de France en cartes, par C. Lamming, 2019
Les 230 PB, par G. Fleurus, 2019
Les tramways de Valenciennes, par Cl. Wagner, 2019
Le réseau Breton, tome 1, par D. Parra et P-H. Emangard, 2020
Le réseau Breton, tome 2, par D. Parra et P-H. Emangard, 2021
Trains d'exception, par J-M. Vuagny et P-Y. Testevuide, 2022

Modélisme ferroviaire

Trains miniatures, découverte d'une passion, par C. Lamming, 2016
Le réseau miniature, recueil de guides pratiques pour réaliser un réseau, ouvrage collectif, 2010-2018
Collection B.A.-BA, De Vecchi
Un talent naturel, par X. Acker, 2019
Divoc Baie, un réseau avec les moyens du bord, par F. Fontana, 2020

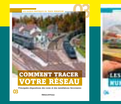
Jouets de collection

Jouef, les petits trains de notre enfance, par C. Lamming, 2008
Un siècle de trains miniatures en France 1915-2015, par C. Lamming, 2015
Fournereau trois générations de passion pour le modélisme ferroviaire, par Th. Fournereau, 2017
VB, la grande marque des petits trains, par J. Roberts, 2018
50 ans de trains miniatures HO en France, par J. Roberts, 2020

Dans la même collection « Les guides pratiques du train miniature »

Bien débuter avec l'échelle N

Signalisation du réseau miniature



Comment tracer votre réseau

Les commandes numériques

L'électricité du réseau

Voies réelles & voies modèles

50 plans et projets

Animez votre réseau (tome1)

couv #7