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.

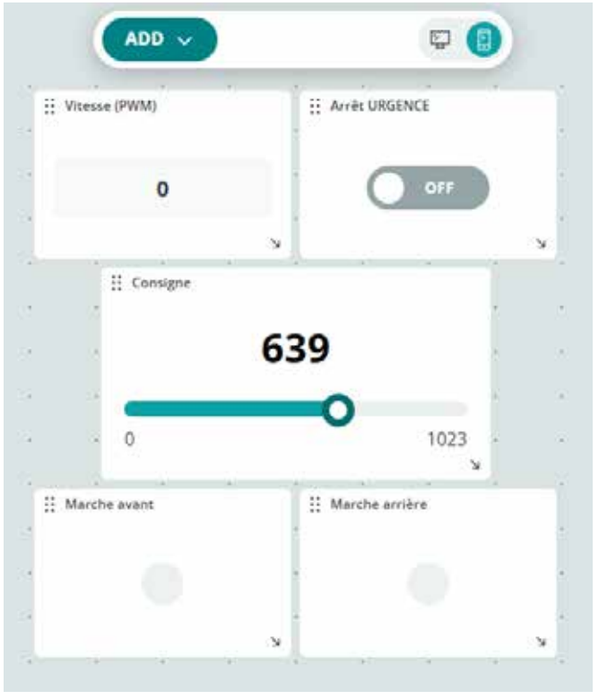

Figure 9-5

Tableau de bord d'un TGV

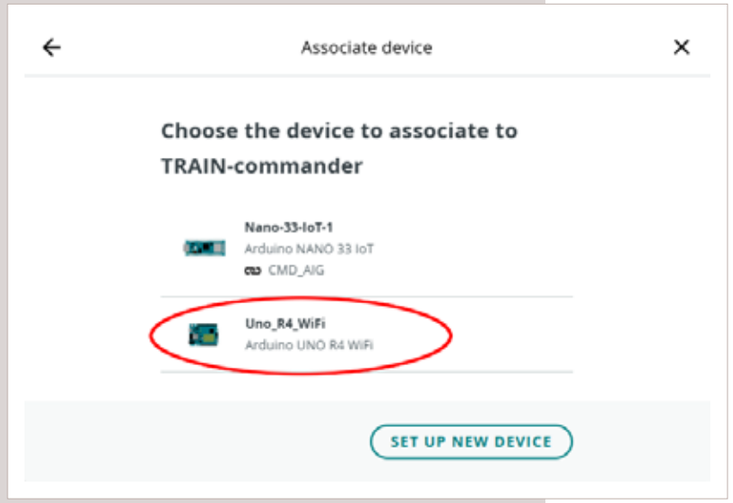

Figure 9-6

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 ?