tutoriel bus CAN pour l'Arduino UNO R4

deux UNO R4 connectés au bus

Dogan Ibrahim (Royaume-Uni)

L'Arduino UNO R4, sorti en deux versions l'année dernière, prend en charge le bus CAN au niveau matériel et logiciel. Dans cet article, nous examinons quelques modules CAN pratiques, une configuration expérimentale et les bases pour afficher le trafic entre les nœuds CAN dans le moniteur série d'Arduino.

Note de l'éditeur. Cet article est un extrait du livre de 326 pages *Mastering the Arduino Uno R4* (Elektor 2023), formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions. Pour les contacter, voir l'encadré « **Questions ou commentaires ?** ».

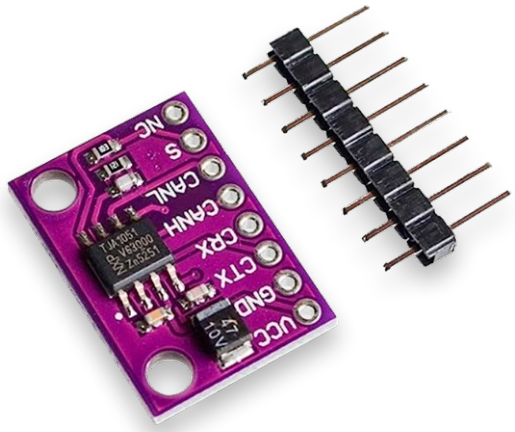

Figure 1. Le module émetteur-récepteur TJA1051.

L'Arduino UNO R4 Minima possède les deux broches suivantes que vous pouvez utiliser avec l'interface du bus CAN (*Controller Area Network*) : D5 (RX) et D4 (TX). Sur la version R4 WiFi, les broches du bus CAN sont D13 (RX) et D10 (TX). Bien qu'il existe des signaux d'interface de bus CAN de base, il est nécessaire de connecter des modules émetteurs-récepteurs à ces broches avant de les connecter au bus CAN.

Émetteurs-récepteurs du bus CAN

Un module émetteur-récepteur de bus CAN est une interface matérielle entre les broches du bus CAN de l'Arduino UNO R4 et le câble du bus CAN. Il existe plusieurs modules émetteurs-récepteurs disponibles sur le marché. Dans le livre mentionné ci-dessus, on utilise le TJA1051 (**figure 1**), un module émetteur-récepteur CAN à grande vitesse qui fournit une interface entre un contrôleur de protocole CAN et le bus CAN bifilaire.
Les caractéristiques de base du TJA1051 sont :

- Tension d'alimentation 4,5 à 5,5 V
- Conforme aux normes ISO 11898-2:2016 et SAE J2284-1 à SAE J2284-5
- Compatible avec les systèmes 12 V et 24 V
- Faible émission électromagnétique
- Courant d'alimentation : 1 mA en mode silencieux ; 5 mA en mode bus récessif ; 50 mA en mode bus dominant.

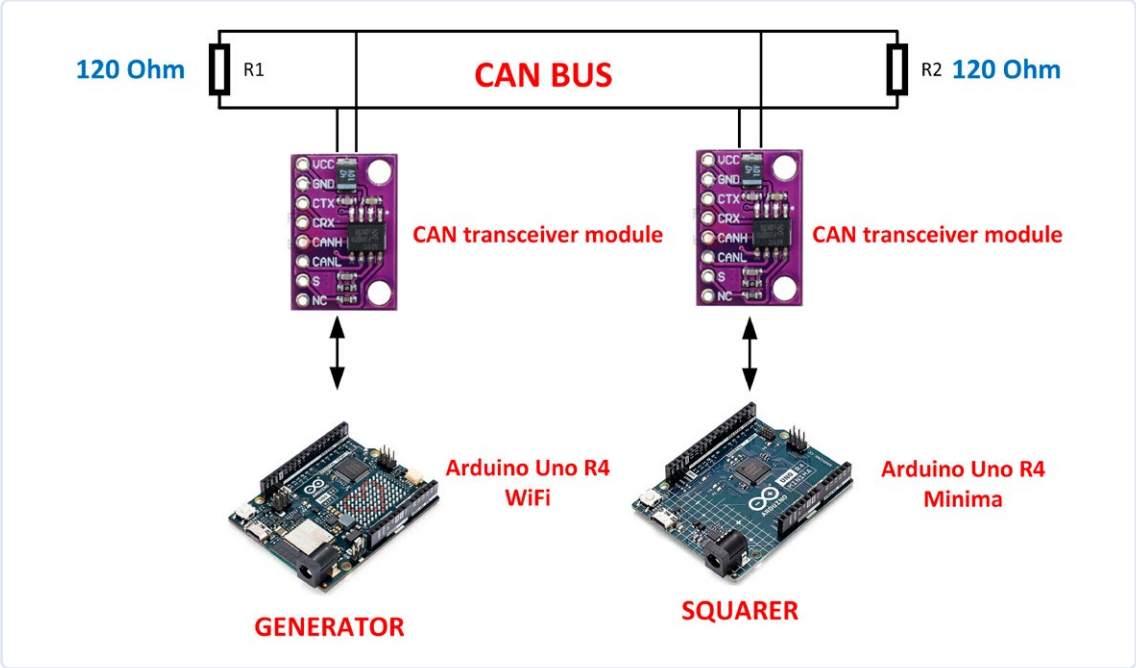

Figure 2. Schéma fonctionnel du projet de démonstration.

Le module émetteur-récepteur TJA1052 est doté des broches suivantes :

- CANH CAN_H : interface de bus
- CANL CAN_L : interface de bus
- VCC : +5 V
- GND : masse
- CTX : transmission des données d'entrée
- CRX : réception de données de sortie (lecture de données sur le bus)
- S : contrôle du mode silencieux (BAS en mode normal, HAUT en mode silencieux).

Dans la section suivante, nous présentons un projet de démonstration basé sur le bus CAN afin de montrer comment la fonction bus CAN de l'Arduino UNO R4 peut être utilisée. La bibliothèque *Arduino_CAN* intégrée est utilisée pour communiquer avec d'autres périphériques CAN.

Communication du bus CAN : UNO R4 WiFi à UNO R4 Minima

Pour réaliser ce projet, vous aurez besoin de deux cartes Arduino UNO R4. La carte Arduino UNO R4 WiFi est appelée le GENERATOR, tandis que la carte Arduino UNO R4 Minima est appelée le SQUARER. Les deux cartes sont connectées sur un bus CAN, ce qui signifie qu'il y a deux nœuds, appelés GENERATOR et SQUARER. Le nœud GENERATOR est programmé pour générer des nombres entiers aléatoires entre 1 et 20 et les envoyer au nœud SQUARER, où les nombres reçus sont élevés au carré et affichés sur le moniteur série de l'EDI. Cette opération est répétée après un délai de 3 secondes. Le but de ce projet est de montrer comment on peut connecter deux Arduino UNO R4 et les faire communiquer sur un bus CAN.

La **figure 2** montre le schéma fonctionnel du projet. Ici, le nœud GENERATOR est un Arduino UNO R4 WiFi, et le nœud SQUARER est un Arduino UNO R4 Minima. La figure montre également deux modules émetteurs-récepteurs de bus CAN et le câblage du bus CAN.

Remarque : l'offre groupée d'Elektor comprend le livre et une carte de développement Arduino R4 Minima. Vous pouvez vous procurer séparément du CI TJA1051, des modules émetteurs-récepteurs du bus CAN et des résistances de terminaison.

La **figure 3** montre le schéma de circuit du projet. Remarquez que le câble de bus est terminé par deux résistances de 120 Ω. Les bornes CANH et CANL des deux émetteurs-récepteurs sont connectées au câble de bus. Les broches CTX et CRX du module sont connectées aux broches D10

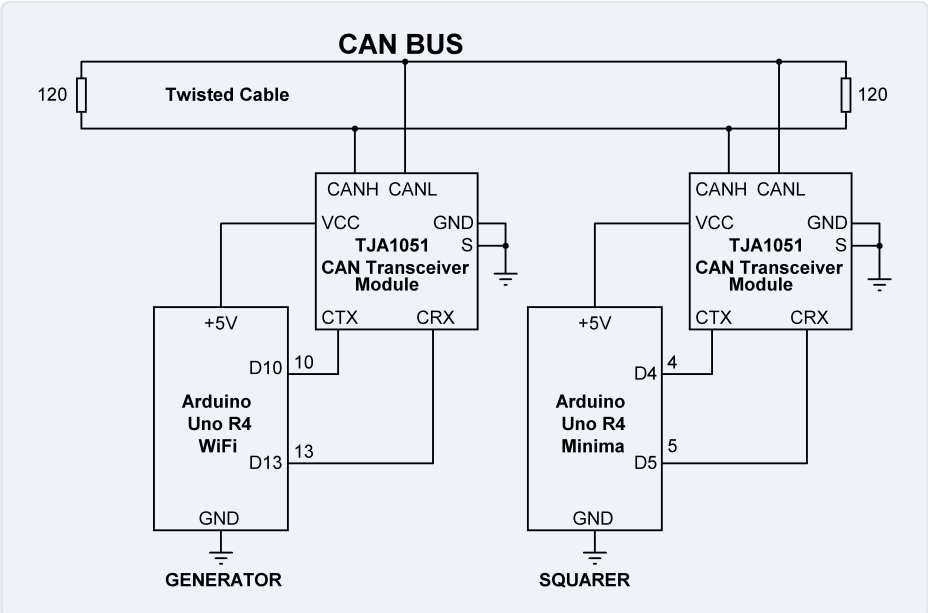

Figure 3. Schéma du circuit du projet de démonstration.

Listage 1. Programme pour le nœud GENERATOR.

```
//-------------------------------------------------------------------------
//            CAN BUS PROGRAM
//            ===============
//
// This program generates random numbers between 1 and 20 and sends them
// over the CAN bus to a node called SQUARER. The SQUARER node takes the
// square of these numbers and displays them on the Serial Monitor. CAN
// baud rate is set to 250 K. The process is repeated every 3 seconds
//
// Author: Dogan Ibrahim
// File  : GENERATOR
// Date  : July, 2023
//-------------------------------------------------------------------------
#include <Arduino_CAN.h>
static uint32_t const CAN_ID = 0x20;
uint8_t RandomNumber;
uint8_t msg_data[8];
void setup()
 {
   Serial.begin(9600);
   delay(5000);
  if (!CAN.begin(CanBitRate::BR_250k))
   {
     Serial.println("CAN begin failed...");
     while(1);
   }
   else
     Serial.println("CAN begin success...");
 }
void loop()
{
  RandomNumber = random(1, 21);                    // Random number
  Serial.print("Random number is: ");
  Serial.println(RandomNumber);
  msg_data[0] = RandomNumber;
  msg_data[1] = 0x00;
  msg_data[2] = 0x00;
  msg_data[3] = 0x00;
  msg_data[4] = 0x00;
  msg_data[5] = 0x00;
  msg_data[6] = 0x00;
  msg_data[7] = 0x00;
  CanMsg msg(CAN_ID, sizeof(msg_data), msg_data);
  int const rc = CAN.write(msg);
  if(rc < 0)
  {
    Serial.print("CAN write failed. Error code is: ");
    Serial.println(rc);
    while(1);
  }
  delay(3000);                                     // Wait 3 secs and repeat
}
```

Listage 2. Programme pour le nœud SQUARER.

```
//--------------------------------------------------------------------
//            CAN BUS PROGRAM
//            ===============
//
// This program receives random numbers between 1 and 20 over the CAN bus
// and takes the square of these numbers and then displays the numbers on
// the Serial monitor
//
// Author: Dogan Ibrahim
// File  : SQUARER
// Date  : July, 2023
//--------------------------------------------------------------------
#include <Arduino_CAN.h>
static uint32_t const CAN_ID = 0x20;
void setup()
{
  Serial.begin(9600);
  delay(5000);
  if (!CAN.begin(CanBitRate::BR_250k))
  {
    Serial.println("CAN begin failed...");
    while(1);
  }
  else
    Serial.println("CAN begin success...");
}

void loop()
{
  if(CAN.available())

 }
```

et D13 de l'Arduino UNO R4 WiFi. De même, les broches CTX et CRX de l'autre émetteur-récepteur sont connectées aux broches D4 et D5 de l'Arduino UNO R4 Minima, respectivement. Dans ce projet, on a utilisé un câble à paires torsadées d'environ un mètre pour faire office de câble de bus CAN.

La bibliothèque *Arduino_CAN* prend en charge les constantes de vitesse de transmission suivantes : *BR_125k, BR_250k, BR_500k, BR_1000k*. Le débit en bauds requis doit être spécifié lors de l'initialisation. Par exemple, pour le débit de 250 kbit/s, on doit appeler la fonction suivante : `CAN.begin(CanBitRate::BR_250k)`.

Envoi de données : un objet message `CanMsg` est créé avec l'identifiant CAN_ID, la taille et les données, et est envoyé via le bus avec la fonction `CAN.write()`.

Réception de données : `CAN.available()` est utilisé pour vérifier la présence de données, puis si des données sont disponibles, elles sont lues avec la fonction `CAN.read()`.

Le **listage 1** montre le code du programme GENERATOR. Au début du programme, nous incluons les fichiers d'en-tête nécessaires. Ensuite, le tableau `msg_data` est initialisé avec huit éléments. Ce tableau stockera les données à envoyer sur le bus CAN. L'identifiant *CAN_ID* est fixé à 0x20. Dans la fonction `setup()`, la vitesse de transmission du bus CAN est fixée à 250 kbit/s et la communication sur le bus CAN est effectivement lancée.

Dans la boucle main, un nombre aléatoire compris entre 1 et 20 est d'abord généré, puis envoyé sur le bus CAN. Notez que seul le premier octet du tableau de données de 8 octets `msg_data` est initialisé. Ce processus est répété après un délai de trois secondes, lorsqu'un nouveau nombre aléatoire est généré et envoyé au nœud SQUARER.

Le **listage 2** montre le code du programme SQUARER. Le début et la fonction `setup()` de ce programme sont similaires aux code présenté dans le listage 1. Dans la boucle *main*, le programme vérifie s'il y a des messages sur le bus CAN et lit ensuite ces messages. Dans ce programme, un message consiste en un nombre entier compris entre 1 et 20. Le nombre reçu est stocké dans la variable `Num`, et son carré est affiché dans le moniteur série.

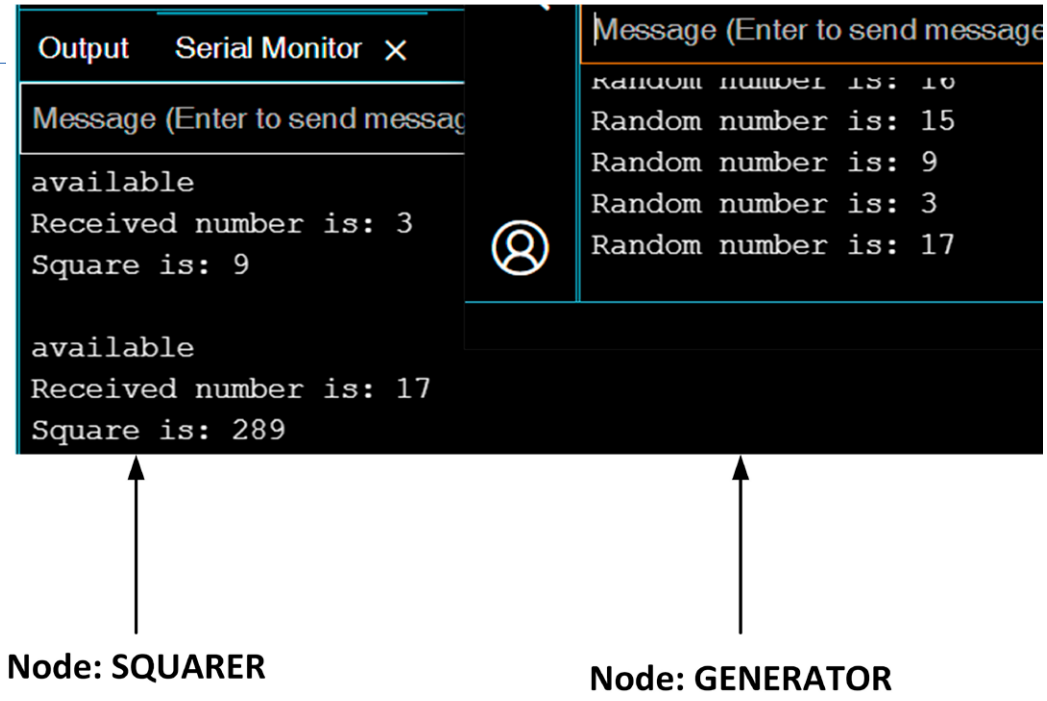

Figure 4.
Exemple de données de sortie affiché dans le moniteur série.

Les deux programmes présentés ici sont inclus dans l'archive logicielle publiée à l'appui du livre " Mastering the Arduino R4 ". L'archive est disponible en téléchargement gratuit dans la section " Livres " du site de l'e-choppe Elektor [1]. Une fois sur la page web, faites défiler jusqu'à Téléchargements et cherchez le fichier *Software_Mastering the Arduino UNO R4 (161.84 MB Zip file)*, téléchargez-le, puis et sauvegardez-le sur votre système. Décompressez le fichier d'archive et localisez les fichiers du programme d'exemple correspondant au chapitre 18 du livre.

Tester le projet

Montez le projet comme indiqué dans le schéma du circuit et alimentez les deux Arduino UNO R4s. Lancez le moniteur série sur le nœud SQUARER, et vous devriez voir les nombres reçus et leurs carrés affichés toutes les trois secondes. L'exemple de sortie est montré dans la **figure 4**. Vous voyez une image similaire ? Félicitations ! Vous êtes sur le bon bus , et en bonne compagnie !

230622-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Dogan Ibrahim est titulaire d'une licence (avec mention) en ingénierie électronique, d'un MSc en automatisation et ingénierie de contrôle et d'un doctorat en traitement des signaux numériques et microprocesseurs. Dogan a travaillé dans de nombreuses organisations et est membre de l'Institution of Engineering and Technology (IET) au Royaume-Uni et ingénieur électricien agréé. Dogan est l'auteur de plus de 100 livres techniques et de plus de 200 articles techniques sur l'électronique, les microprocesseurs, les microcontrôleurs et les domaines connexes. Dogan est un professionnel certifié d'Arduino et a de nombreuses années d'expérience avec presque tous les types de microprocesseurs et de microcontrôleurs.

Produits

- **Mastering the Arduino Uno R4 – offre groupée Livre + carte Arduino Uno R4 Minima :** www.elektor.fr/20649
- **The CAN Bus Companion (+ Module de bus CAN gratuit)** Livre + module MCP2515 : www.elektor.fr/20405 Livre numérique : www.elektor.fr/20406

LIEN

[1] Téléchargements de logiciels du livre : Mastering the Arduino Uno R4 : https://elektor.fr/mastering-the-arduino-uno-r4