Comment augmenter le nombre d'UART dans une application Arduino ?

Jean-Luc Levant F4GSC

La plateforme de développement Arduino est maintenant largement employée pour des applications mêlant à la fois l'électronique et l'informatique.

De nombreux modules électroniques (Shield) existent et ils peuvent se connecter facilement aux multiples cartes Arduino par l'intermédiaire d'un UART (Universal Asynchronous Receiver and Transmitter).

A quoi sert un UART ?

La figure 1 montre un exemple classique de l'utilisation d'un UART. Il assure la communication entre la carte Arduino UNO et le PC.
La ligne TXD de la carte envoie des données vers le PC. La ligne RXD reçoit des données du PC.
Un circuit translateur de niveau (+5 V ← → +/-12 V) est nécessaire pour assurer l'application de la norme RS232.

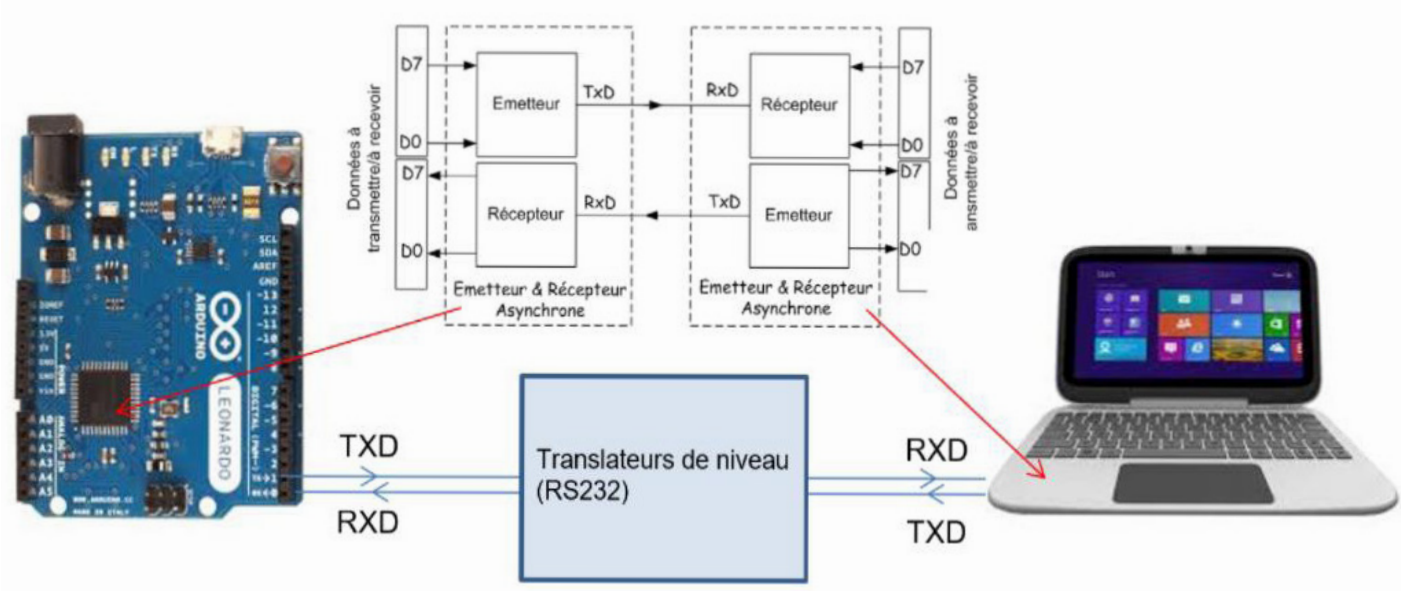


Figure 1 : Interface série asynchrone pour connecter la carte Arduino au PC.

Les principales caractéristiques d'une liaison série asynchrone sont mentionnées ci-après :

- **Asynchrone :** l'horloge de cadencement n'est pas incluse dans la transmission. Elle est reconstituée à la réception dans le récepteur
- **Format de la donnée :** 5,6,7 et 8-bits + 1 x bit start + 1 x bit stop
- **Bit de parité :** Premier niveau de correction d'erreur
- **Vitesse de transmission :** 300 ... 115 200 bauds (vitesses standards)
- **Niveau du signal TX et RX :**
  - RS232 → +/-12v
  - EIA-485/422 → +/-6v, signaux différentiels
  - La carte Arduino UNO, NANO... fournit des signaux 0/+5v.

La figure ci-dessous montre la constitution d'une trame série asynchrone qui commence par un « bit start », puis par la donnée dont le format est programmable entre 5 et 8-bits, puis par un bit de parité (option) et enfin par un « bit stop ».

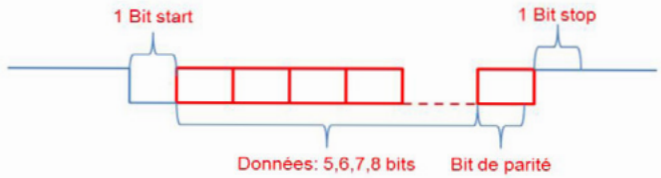


Figure 2 : Description de la trame gérée par l'UART.

Un autre exemple de l'utilisation de l'UART est la connexion des modules électroniques à une carte Arduino.

Cette interface nécessite seulement un fil pour la réception (RX) et un autre pour l'émission (TX).

Nombre d'UART dans une carte Arduino comportant un Mega328

Une carte Arduino à base de Mega328 (Uno, micro, nano…) possède un seul UART matériel (intégré dans le microcontrôleur).

Il est utilisé pour les phases de programmation et de mise au point. Il est possible de l'utiliser lorsque l'application est en fonctionnement normal.

Un deuxième UART peut-être émulé par logiciel. Il faut alors utiliser une librairie facilement téléchargeable depuis le Web (SotfwareSerial). Un seul UART logiciel est utilisable sur ces cartes.

Si plus de deux UART logiciels sont nécessaires, il faut alors changer de carte Arduino et prendre par exemple la carte Arduino DUE qui possède quatre UART matériels ou utiliser la technique décrite ci-après.

Comment augmenter le nombre d'UART sur une carte Arduino UNO, MICRO… ?

Dans le cas d'une carte à base de Mega328, le nombre d'UART peut être augmenté en utilisant deux circuits logiques : le multiplexeur et le démultiplexeur. Ce besoin est apparu lorsque j'ai voulu piloter quatre modules radio VHF/UHF SA818. La figure ci-dessous montre cet exemple de réalisation.

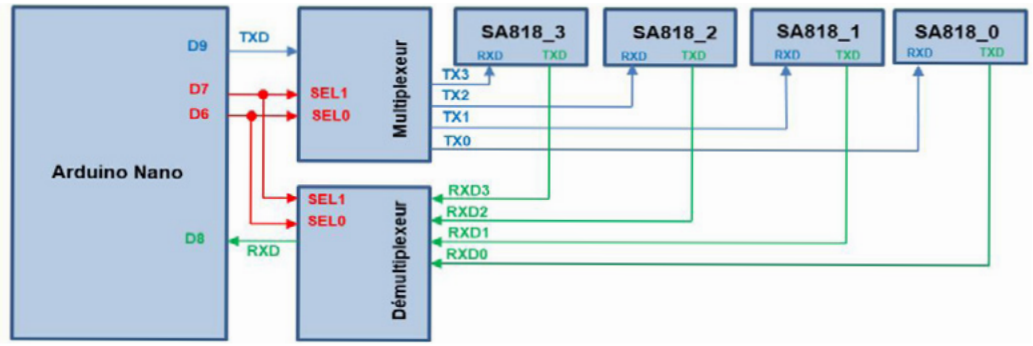


Figure 3 : Exemple d'application de pilotage de 4 modules radio SA818.

La carte Arduino Nano pilote quatre modules radio VHF/UHF de la série SA818. La liaison série matérielle étant occupée par les fonctions de programmation et de mise au point, un UART logiciel (SotfwareSerial) est utilisé pour la communication avec les quatre modules.

Le multiplexeur permet d'envoyer les caractères émis par la carte Arduino (TXD) vers l'entrée RXD du circuit radio sélectionné par les deux signaux SEL1 et SEL0. La table ci-dessous établit la correspondance entre les signaux de sélection SEL1 et SEL0 et le module radio choisi.

SEL1	SEL0	SA818-n
0	0	SA818-0
0	1	SA818-1
1	0	SA818-2
1	1	SA818-3

Table 1 : Table de sélection des modules radio SA818.

Le démultiplexeur permet d'envoyer vers l'entrée série de l'Arduino (RXD) les caractères reçus par l'un des modules radio sélectionné par les deux signaux SEL1 et SEL0.

Programme

Voici ce que doit comporter le programme pour sélectionner et piloter les modules radio SA818. Un exemple d'application est donné dans le cas où il est nécessaire d'obtenir le niveau du signal reçu (RSSI).

Le programme dans ce cas transmet les chaînes de commande pour :

- Programmer la fréquence de réception.
- Demander l'envoi du niveau du signal reçu (RSSI).
- Obtenir le niveau du signal reçu.

L'entête du programme précise quels entrées/sorties du microcontrôleur sont connectés aux circuits de multiplexage et de démultiplexage.

```
#define RXD 8    //— La broche 8 de l'Arduino est connectée sur la sortie RX du démultiplexeur

#define TXD 9    //— La broche 9 de l'Arduino est connectée sur l'entrée TX du multiplexeur

#define SEL0 7   //— La broche 7 permet de sélectionner le module radio

#define SEL1 6   //— La broche 6 permet de sélectionner le module radio
```

Un UART logiciel (SerialDraU) est installé et configuré avec la fonction « SoftwareSerial ».

Le fichier de définition devra être préalablement importé et inclus dans le programme de votre application (#include <SoftwareSerial.h>).

```
#include <SoftwareSerial.h>
SoftwareSerial SerialDraU(RXD,TXD);
```

Le récepteur de l'UART est connecté au signal RXD et l'émetteur est lui associé au signal TXD. Par défaut la vitesse de transmission est de 9 600 bauds, 1 bit de stop et de start. La fonction « selectModule » sélectionne le module grâce à la variable « SEL ». Elle peut prendre une valeur entre 0 et 3 (numéro du module radio).

```
//=============================================
//-- Sélectionne le module radio
//=============================================
void selectModule( byte SEL)
{
switch (SEL)
//=============================================
//-- sélectionne le module 0
//=============================================
case 0: //-- sélectionne le module 0
{
digitalWrite(SEL1,0);digitalWrite(SEL0,0);
break;
}
//=============================================
//-- sélectionne le module 1
//=============================================
case 1:
{
digitalWrite(SEL1,0);digitalWrite(SEL0,1);
break;
}
//=============================================
//-- sélectionne le module 2
//=============================================
case 2:
{
digitalWrite(SEL1,1);digitalWrite(SEL0,0);
break;
}
//=============================================
//-- sélectionne le module 3
//=============================================
case 3:
{
digitalWrite(SEL1,1);digitalWrite(SEL0,1);
break;
}
default: break;
}
```

L'exemple d'utilisation ci-dessous permet de récupérer le niveau du signal reçu par le module radio sélectionné par la variable « SEL ».

```
void lecture_RSSI (byte SEL)
{
switch (SEL)
{
case 0:
{
//— Sélectionne le canal 0
selectModule(0);
//—La commande est envoyée vers le module radio pour demander le renvoi du niveau du signal reçu.
SerialDraU.println(«RSSI?»);
//— L'UART enregistre les caractères transmises par le module dans la mémoire "buf_1" jusqu'à ce que
//    le caractère de fin de ligne est reçu.
SerialDraU.readBytesUntil('\n',buf_1,9); // return result
//—- La chaîne de caractère est convertie en un nombre entier.
rssi_0 = atoi(&buf_1[5]);
break;
}
case 1:
{
//— Sélectionne le module 1

selectModule(1);
SerialDraU.println(«RSSI?»);
SerialDraU.readBytesUntil('\n',buf_1,9); // return result
rssi_1 = atoi(&buf_1[5]);
break;
}
case 2:
{
//— Sélectionne le module 2

selectModule(2);
SerialDraU.println(«RSSI?»);
SerialDraU.readBytesUntil('\n',buf_1,9); // return result
rssi_1 = atoi(&buf_1[5]);
break;
}
case 3:
{
//— Sélectionne le module 3

selectModule(3);
SerialDraU.println(«RSSI?»);
SerialDraU.readBytesUntil('\n',buf_1,9); // return result
rssi_1 = atoi(&buf_1[5]);
break;
}
default: break;
}
}
//— Sélectionne la voie radio 0

selectModule(0);
//— La chaîne de commande programmation du volume est envoyée vers le module radio 0
SerialDraU.print(«AT+DMOSETVOLUME= 6»);
```

Pour transmettre un caractère ou une chaîne de caractères, la fonction « selectModule (byte SEL) » sélectionne la voie radio choisie par la variable « SEL ».

Un exemple de réglage du volume de la voie radio 0 programmé à la valeur 6 est donné ci-dessous.

Choix des circuits intégrés assurant les fonctions de multiplexage et de démultiplexage

Il existe plusieurs références de multiplexeur et de démultiplexeur qui dépendent de la technologie choisie (TTL, CMOS, HCMOS...). Quelques-unes de ces références MOS sont indiquées ci-dessous.

MC14051	CD4051	74HC4051	HCF4051	HE4051

La version 4051 est choisie car elle intègre les deux fonctions dans un même circuit. En effet, comme le montre la figure ci-dessous, ce circuit MOS se comporte comme un circuit analogique qui contrôle la résistance d'isolement entre les broches IN/OUT et OUT/IN et donc le passage bidirectionnel de l'information.

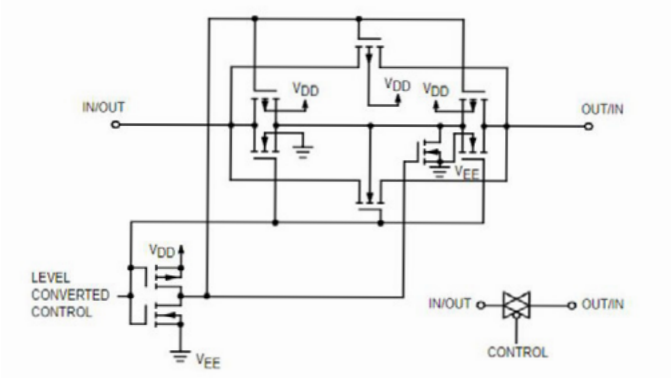


Figure 4 : La résistance de l'isolement entre les broches « IN/OUT » et « OUT/IN » est contrôlée par l'entrée « CONTROL ».

Il existe plusieurs versions en fonction du nombre de voies à multiplexer et à démultiplexer. La figure ci-dessous montre trois versions de ce circuit MOS.

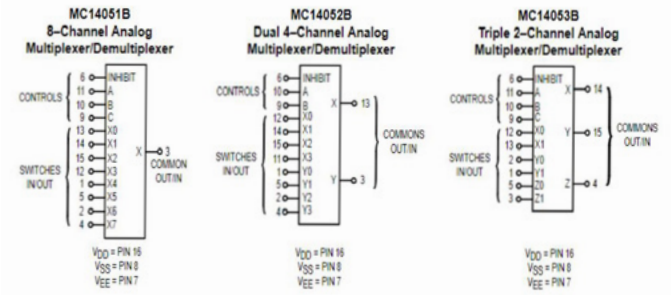
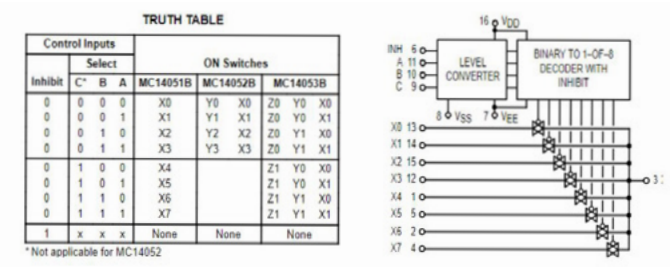


Figure 5. : À gauche, 8 voies vers 1. Au centre, 2 x 4 voies vers 2 x 1. À droite, 3 x 2 voies vers 3 x 1.

La table de codage est montrée ci-dessous. L'entrée « Inhibit » doit être mise à zéro pour que ce circuit fonctionne.

L'entrée « C » est aussi mise à zéro puisque seulement quatre voies sont utilisées dans cet exemple. Les entrées « A » et « B » permettent de sélectionner le module radio.



TRUTH TABLE

Control Inputs				ON Switches		
	Select					
Inhibit	C*	B	A	MC14051B	MC14052B	MC14053B
0	0	0	0	X0	Y0 X0	Z0 Y0 X0
0	0	0	1	X1	Y1 X1	Z0 Y0 X1
0	0	1	0	X2	Y2 X2	Z0 Y1 X0
0	0	1	1	X3	Y3 X3	Z0 Y1 X1
0	1	0	0	X4		Z1 Y0 X0
0	1	0	1	X5		Z1 Y0 X1
0	1	1	0	X6		Z1 Y1 X0
0	1	1	1	X7		Z1 Y1 X1
1	x	x	x	None	None	None

* Not applicable for MC14052

Figure 6 : Table de codage pour la sélection des modules radio.

Un exemple de réalisation est montré sur la figure ci-dessous. Deux circuits MC14051 (c'est ce que j'avais dans mon tiroir) assurent les fonctions de multiplexage et de démultiplexage.

Une carte Arduino Nano est utilisée pour le pilotage de ces deux circuits.

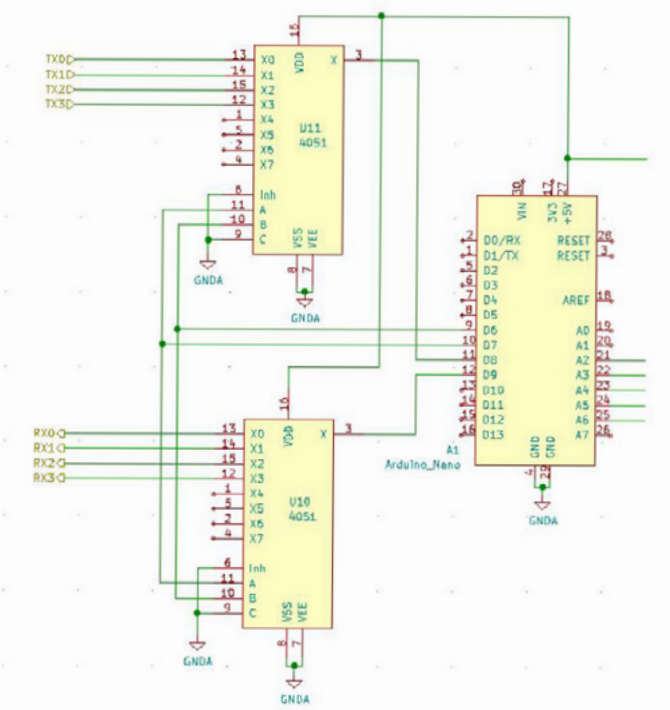


Figure 7 : Exemple de câblage des fonctions de multiplexage et démultiplexage avec un Arduino Nano.

Conclusion

Les fonctions de multiplexage et de démultiplexage ne sont pas limitées aux UART mais peuvent être utilisées à chaque fois que le nombre limité de périphériques doit être augmenté.

En utilisant les circuits MOS, ces périphériques peuvent être aussi bien analogiques que numériques.