34-Communication en série

introduction

Ce tutoriel traite de l'utilisation de la communication en série dans Max. Un grand nombre de périphériques externes utilisent un protocole en série (RS-232, Bluetooth, par exemple) pour communiquer avec un ordinateur, et les flux en série peuvent même être utilisés pour la communication à faible bande passante entre ordinateurs (pensez à Internet par accès commuté!). Des périphériques tels que les mixeurs vidéo de niveau professionnel, les lecteurs de DVD et les systèmes d'éclairage de theâtre utilisent des interfaces en série pour recevoir des commandes, et les systèmes à microcontrôleur utilisés par la communauté informatique s'appuient sur une communication en série pour transmettre les données du capteur à un ordinateur.

Pour ce tutoriel, nous allons utiliser l'exemple de la communication en série avec une plate-forme de prototype électronique très populaire appelée *Arduino*. Il consiste en un petit microcontrôleur monté sur une carte de prototypage programmée par un langage de type C. Il peut communiquer avec un ordinateur hôte via une ligne en série via USB. Même si vous ne possédez pas d'Arduino, les principes présentés dans ce didacticiel devraient vous donner suffisamment d'informations pour communiquer avec n'importe quel périphérique en série pour vos besoins.

Pour ouvrir le patch du didacticiel, cliquez sur le bouton vert **Ouvrir didacticiel** dans le coin supérieur droit de la fenêtre de documentation.

Le code Arduino

Avant d'examiner le patcheur Max, voici le code que nous avons utilisé pour programmer notre Arduino dans ce didacticiel. Dans cet exemple, l'Arduino ne lit pas les données des capteurs et n'exécute aucune tâche particulière: il attend simplement une entrée de l'ordinateur, puis commence à transmettre en flux de nombres en série. Le code pour l'Arduino est celui-ci:

```
// Testeur de série Arduino
// rld, cyclisme'74, 3.2008

long randomvalue = 0; // valeur aléatoire
long contervalue  = 0; // valeur de compteur
int serialvalue; // valeur pour l'entrée sérielle
int started = 0; // Indique si nous avons encore reçu une publication en série

void setup()
{
  Serial.begin (9600); // ouvre le port série arduino
}

void loop ()
{
  if (Serial.available ()) // vérifie s'il existe des données série dans la mémoire tampon
  {
    serialvalue = Serial.read (); // lit un octet de données série
    started = 1; // active l'indicateur démarré
  }

  if (started) // boucle après la réception des données série
```

```
  {
   randomvalue = random (1000); // choisir un nouveau nombre aléatoire
   Serial.print (countervalue); // imprimer le compteur
   Serial.print (""); // imprimer un espace
   Serial.print (randomvalue); // affiche la valeur aléatoire
   Serial.print (""); // imprimer un espace
   Serial.print (serialvalue); // renvoie la valeur de série reçue
   Serial.println (); // imprimer un saut de ligne
   countervalue = ( countervalue + 1)% 1000; // incrémente le compteur
   delay (100); // pause
  }
}
```

Même si les spécificités du langage de programmation Arduino ne vous sont pas familières, vous pouvez regarder les commentaires de chaque ligne et voir ce qui se passe. Le microcontrôleur s'initialise et ouvre un port série à 9600 bauds - c'est à cette vitesse qu'il communiquera avec notre patcheur Max. Il entre ensuite dans une boucle dans laquelle il vérifie si nous lui avons envoyé des données et, lorsque c'est le cas, il commence à renvoyer des chiffres à l'ordinateur. Il génère un nombre qui compte de **0** à **1000** encore et encore, un nombre aléatoire (dans le même intervalle) et un écho de la dernière valeur que nous avons envoyée dans la puce. Ces trois valeurs sont imprimées (comme sur un terminal) sur la ligne en série.

En regardant l'objet *serial*

Jetez un coup d'œil au tutoriel. Il s'agit d'un objet *serial* au sommet avec une logique de soutien à la sortie, qui finit par piloter un graphe *lcd* des données provenant de l'Arduino. L'objet *serial* Max est utilisé pour la communication avec des périphériques en série ne disposant pas de pilotes spéciaux qui les placent dans une autre catégorie (par exemple, MIDI ou HID). Ces périphériques génériques communiquent avec Max via un **port** *serial*, spécifié comme le premier argument de l'objet. Dans Max, ces ports sont désignés par des lettres (**a**, **b**, etc.). Cliquez sur la boîte de *message* d'impression en haut du patcheur: les ports *serial* disponibles sur votre ordinateur apparaîtront dans la Console Max, avec leurs correspondances en lettres. L'Arduino devrait y apparaître, ainsi que tout autre périphérique en série que vous pourriez avoir. Si nécessaire, vous pouvez changer l'argument de lettre de l'objet *serial* en déverrouillant le patcheur et en ressaisissant l'objet.

En plus du port en série, l'objet *serial* utilise un certain nombre d'arguments supplémentaires pour configurer le protocole de communication. Le deuxième argument de l'objet définit le débit en bauds et doit correspondre à celui de l'Arduino que nous utilisons. Puisque nous avons demandé à l'Arduino de transmettre des données à **9600** bauds, nous avons appliqué le même argument à notre objet *serial*. Les arguments supplémentaires de l'objet nous permettent de spécifier le ***nombre de bits de données***, ***les bits d'arrêt*** et ***la parité*** (paire ou impaire) du périphérique auquel nous nous adressons. Comme l'Arduino est un périphérique assez typique, les paramètres par défaut fonctionneront.

Transmission et interrogation des données

L'objet *serial* peut à la fois émettre et recevoir des données en série. Il transmet les données en prenant un ***octet*** (un entier compris dans la plage de **0** et **255**) dans son entrée et en l'envoyant sur le port en série sélectionné. Ces octets peuvent représenter des valeurs de commande ou peuvent être traduits en ASCII (les mêmes nombres sont générés à partir de l'objet *key*). Afin de recevoir des données, l'objet *serial* doit être ***interrogé*** (exactement comme le *mousestate*). Lorsqu'un objet reçoit un **bang**, il vide son tampon interne de tous les messages sériels en attente et les envoie de manière

séquentielle (sous forme d'octets) à sa sortie.

En supposant que notre Arduino soit chargé avec notre programme (ci-dessus) et connecté à l'ordinateur, nous devrions être en mesure de communiquer avec lui à condition que l'objet *serial* soit réglé sur le bon port. Activez l'objet *metro* avec la boîte *toggle*. Comme vous pouvez le voir à partir de la boîte de *nombres* attachée à l'objet, rien ne se passera. En effet, dans notre code Arduino, nous attendons que l'ordinateur communique d'abord avec le microcontrôleur.

Tapez le nombre **10** dans la boîte de *nombres* attachée à l'objet *serial* et appuyez sur la touche retour. En supposant que l'objet *serial* a correctement transmis l'octet, vous devriez voir des nombres commencer à apparaître hors de l'objet. Activez le *toggle* étiqueté **1** pour imprimer les données sérielles "brutes" dans la Max Console.

Formateage des données

Vous remarquerez que les nombres qui apparaissent dans la Max Console à partir de la sortie en série "brute" ne semblent pas avoir beaucoup de sens. Cela est dû au fait que l'Arduino transmet ses valeurs au format ASCII, comme si elles étaient saisies sur un ordinateur. Ainsi, le nombre **15** est représenté par la valeur ASCII **49** ("1") suivie de la valeur ASCII **53** ("5"). Les espaces entre nos trois valeurs sont représentés par le code ASCII pour un espace (**32**), et chaque ensemble de trois nombres est terminé par un retour chariot et un saut de ligne (ASCII **13** suivi de **10**). Pour utiliser ces valeurs dans Max, nous devons d'abord les regrouper, puis les convertir d'ASCII en une chaîne lisible.

Décochez le *toggle* **1** et cochez le *toggle* **2**. Ceci imprime la sortie de l'objet *z l* **group** dans le patcheur. L'objet *zl* prend des séquences de ces valeurs ASCII et les regroupe dans une liste; l'objet *select* au-dessus de *zl* force la sortie de liste (et son effacement) chaque fois qu'un **13** (retour chariot) est reçu. De plus, il élimine les caractères de saut de ligne (ASCII **10**) du flux. Comme la sortie droite de *select* affiche tout ce qui n'est pas sélectionné par l'objet, le reste des valeurs se déverse dans le *zl*. En conséquence, la console Max affiche une liste de valeurs ASCII qui doivent représenter trois nombres séparés par des espaces (ASCII **32**).

Décocher le *toggle* **2** et cochez le *toggle* **3**. Ceci montre la sortie de l'objet *itoa*, qui prend la liste groupée et convertit les entiers en leurs caractères ASCII correspondants. En conséquence, la console Max devrait afficher quelque chose qui a finalement du sens compte tenu de notre code Arduino: nous devrions voir une valeur comptant de **0** à **1 000** suivie d'une valeur aléatoire suivie de **10** (l'octet que nous avons envoyé au microcontrôleur plus tôt).

L'objet *itoa* crée un ***symbole*** Max, qui est lisible par nous à des fins de débogage, mais pas tout à fait ce dont nous avons besoin pour notre patcheur. Bien que nous reconnaissions que le message est une liste de trois entiers, les objets tels que *unpack* ne le feront pas. L'objet *fromsymbol* ***analyse*** un symbole, en séparant différents types de données en fonction d'un espace blanc; en conséquence, notre symbole est converti en une liste de trois entiers que l'objet *unpack* peut scinder. Notez que les boîtes de *nombres* qui sortent de l'objet *unpack* affichent correctement nos valeurs. Modifiez à nouveau la boîte de *nombres* en haut et vous verrez que la troisième valeur de l'objet *serial* sera mise à jour en conséquence.

Jetez un coup d'œil à la partie de dessin du patcheur et voyez comment elle fonctionne. Comme beaucoup de nos exemples basés sur l'écran *lcd*, il visualise simplement la sortie de l'objet *serial*, réalisant dans ce cas un graphique basé sur le compteur (x), la valeur aléatoire (y) et l'octet renvoyé (couleur). Ces valeurs pourraient facilement être appliquées pour contrôler ce que vous voulez.

Résumé

L'objet *serial* Max offre la possibilité de communiquer vers et à partir de périphériques en série standard dans Max. Il transmet et reçoit des données sous forme d'octets simples ; des objets tels que *zl*, *itoa* et *fromsymbol* peuvent être utilisés pour donner un sens à tout cela.