MAX

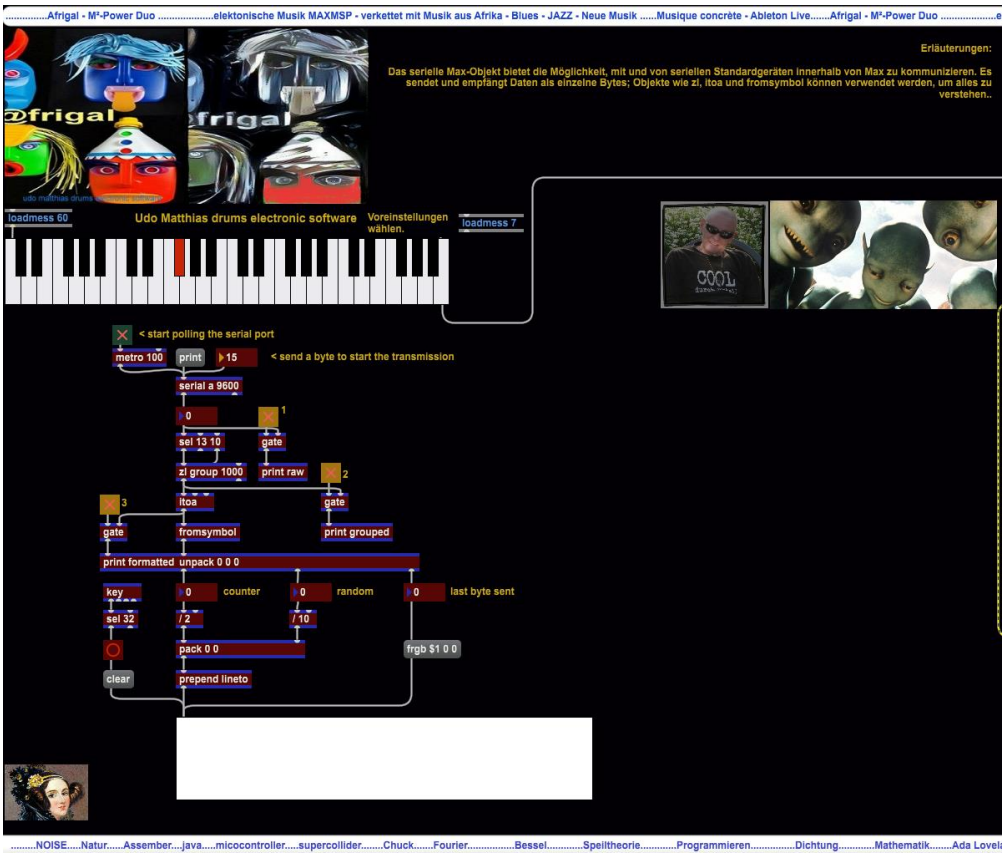
Udo matthias 07626-2 999 847
mobil: 017621-605276
http://www.udomatthias.com
https://www.facebook.com/udo.matthias.3
info@udomatthias.com

If it feels good, it must be in time!!

Kommunikations2: Serielle Kommunikation

Einführung

Dieses Tutorial behandelt das Thema der Verwendung der seriellen Kommunikation in Max. Eine beliebige Anzahl externer Geräte verwendet ein serielles Protokoll (z. B. RS-232, Bluetooth) für die Kommunikation mit einem Computer, und serielle Streams können sogar für die Kommunikation mit geringer Bandbreite zwischen Computern verwendet werden (denken Sie an DFÜ-Internet!). Geräte wie professionelle Videomischer, DVD-Player und Kinobeleuchtungssysteme verwenden serielle Schnittstellen zum Empfangen von Befehlen, und Mikrocontrollersysteme, die von der physischen Computergemeinschaft verwendet werden, sind auf serielle Kommunikation angewiesen, um Sensordaten zurück an einen Computer zu übertragen.



In diesem Kapitel verwenden wir das Beispiel der seriellen Kommunikation mit einer sehr beliebten Elektronik-Prototyping-Plattform namens Arduino. Es besteht aus einem kleinen Mikrocontroller, der auf einer Prototyping-Karte montiert ist, **die in einer C-ähnlichen Sprache programmiert ist.** Es kann über eine serielle Leitung über USB mit einem Host-Computer kommunizieren. Selbst wenn wir keinen Arduino haben, sollten die in diesem Kapitel gezeigten Prinzipien uns genügend Einblick geben, um mit jedem seriellen Gerät zu kommunizieren, das wir benötigen.

Udo matthias 07626-2 999 847
mobil: 017621-605276
http://www.udomatthias.com
https://www.facebook.com/udo.matthias.3
info@udomatthias.com
If it feels good, it must be in time!!

Der Arduino-Code

Bevor wir uns den Max-Patcher ansehen, finden wir unten den Code, den wir in diesem Kapitel zum Programmieren unseres Arduino verwendet haben. In diesem Beispiel liest der Arduino keine Sensordaten und macht nichts Besonderes: **Er wartet nur auf Eingaben vom Computer und beginnt dann, eine serielle Rückmeldung von Zahlen zurück zu streamen. Der Code für das Arduino ist hier:**

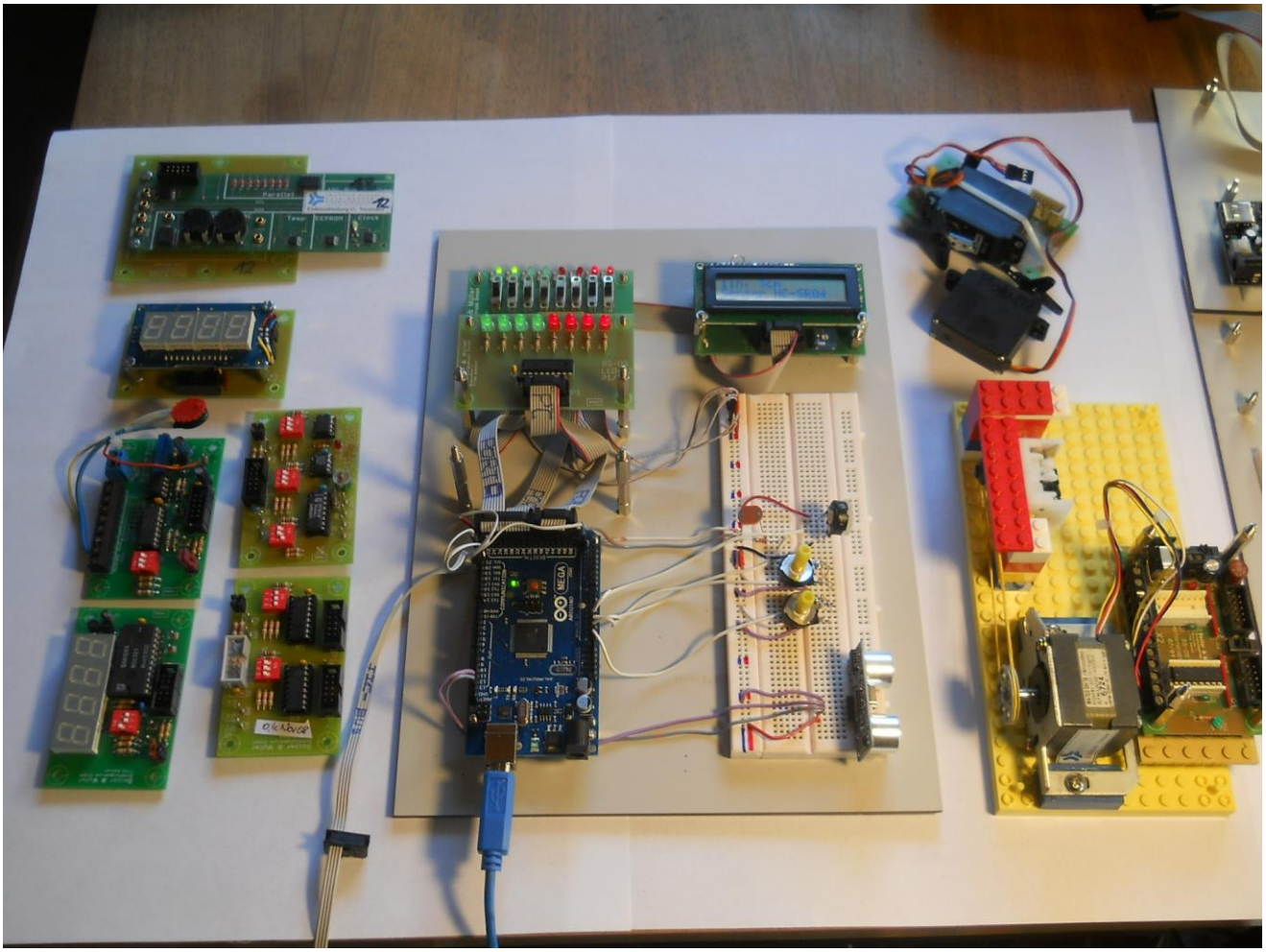
```
// Arduino Serial Tester
// rld, cycling'74, 3.2008
long randomvalue = 0; // random value
long countervalue = 0; // counter value
int serialvalue; // value for serial input
int started = 0; // flag for whether we've received serial yet
void setup()
{
  Serial.begin(9600); // open the arduino serial port
}
void loop()
{
  if(Serial.available()) // check to see if there's serial data in the buffer
  {
    serialvalue = Serial.read(); // read a byte of serial data
    started = 1; // set the started flag to on
  }
  if(started) // loop once serial data has been received
  {
    randomvalue = random(1000); // pick a new random number
    Serial.print(countervalue); // print the counter
    Serial.print(" "); // print a space
    Serial.print(randomvalue); // print the random value
    Serial.print(" "); // print a space
    Serial.print(serialvalue); // echo the received serial value
    Serial.println(); // print a line-feed
    countervalue = (countervalue+1)%1000; // increment the counter
    delay(100); // pause
  }
}
```

Auch wenn die Besonderheiten der Programmiersprache Arduino nicht bekannt sind, können wir uns die Kommentare für jede Zeile ansehen und schauen, was passiert. Der Mikrocontroller initialisiert sich selbst und öffnet eine serielle Schnittstelle mit

Udo matthias 07626-2 999 847
mobil: 017621-605276
http://www.udomatthias.com
https://www.facebook.com/udo.matthias.3
info@udomatthias.com

If it feels good, it must be in time!!

9600 Baud - dies ist die Geschwindigkeit, mit der er mit unserem Max-Patcher kommuniziert. Es tritt dann in eine Schleife ein, in der überprüft wird, ob Daten gesendet wurden, und sobald dies der Fall ist, werden die Nummern zurück zum Computer gestreamt. Es wird eine Zahl generiert, die immer wieder von 0 bis 1000 zählt, eine Zufallszahl (im gleichen Bereich) und ein Echo des letzten Wertes, den wir in den Chip gesendet haben. Diese **drei Werte werden (wie auf einem Terminal) über die serielle Leitung gedruckt.**



Betrachtet man das serielle Objekt

Schauen wir uns den Patch an.
Er besteht aus einem seriellen Objekt oben mit einer unterstützenden Logik am Ausgang, die schließlich einen LCD-Graphen der vom Arduino kommenden Daten steuert.

➔**Das serielle Max-Objekt wird für die Kommunikation mit seriellen Geräten verwendet,** die keine speziellen Treiber haben, die sie in eine andere Kategorie einordnen (z. B. MIDI oder HID).

➔**Diese generischen Geräte kommunizieren mit Max über eine serielle Schnittstelle,** die als erstes Argument für das Objekt angegeben wird. In Max sind diese Ports mit Buchstaben (a, b usw.) gekennzeichnet.

➔**Klicke oben im Patcher auf das Feld zum Drucken von Nachrichten:**

Udo matthias 07626-2 999 847
mobil: 017621-605276
http://www.udomatthias.com
https://www.facebook.com/udo.matthias.3
info@udomatthias.com
If it feels good, it must be in time!!

Die verfügbaren seriellen Anschlüsse Ihres Computers werden in der Max-Konsole zusammen mit ihren Buchstabenzuordnungen angezeigt. Der Arduino sollte dort zusammen mit anderen seriellen Geräten angezeigt werden. Bei ➔Bedarf können wir das Buchstabenargument in das serielle Objekt ändern, indem wir den Patcher entsperren und das Objekt erneut eingeben.

➔**Zusätzlich zur seriellen Schnittstelle benötigt das serielle Objekt eine Reihe zusätzlicher Argumente**, um das Kommunikationsprotokoll einzurichten.

- Das zweite Argument des Objekts legt die Baudrate fest und muss mit der des von uns verwendeten Arduino übereinstimmen. Da wir dem Arduino befohlen haben, Daten mit 9600 Baud zu übertragen, haben wir das gleiche Argument für unser serielles Objekt angegeben.
- Zusätzliche Argumente für das Objekt ermöglichen es uns, die Anzahl der Datenbits, Stoppbits und die Parität (gerade oder ungerade) des Geräts anzugeben, mit dem wir sprechen. Da der Arduino ein ziemlich typisches Gerät ist, funktionieren die Standardeinstellungen. ➔w.s. Assembler programmieren!!

Daten übertragen und abrufen

Das serielle Objekt kann serielle Daten sowohl senden als auch empfangen. Es überträgt Daten, indem es ein Byte (eine Ganzzahl im Bereich von 0 bis 255) in seinen Eingang nimmt und es über die ausgewählte serielle Schnittstelle sendet. Diese Bytes können Befehlswerte darstellen oder in ASCII übersetzt werden (die gleichen Zahlen werden vom Schlüsselobjekt ausgegeben).
Um Daten zu empfangen, muss das serielle Objekt abgefragt werden (genau wie Mousestate). Wenn ein Bang vom Objekt empfangen wird, leert es seinen internen Puffer von wartenden seriellen Nachrichten und sendet sie nacheinander (als Bytes) aus seinem Ausgang.

Angenommen, unser Arduino ist mit unserem Programm (oben) geladen und mit dem Computer verbunden, sollten wir in der Lage sein, mit ihm zu sprechen, vorausgesetzt, das serielle Objekt ist auf den richtigen Port eingestellt.
➔Schalte das metro -Objekt mit dem Umschaltfeld ein. Wie wir dem Nummernfeld am Objekt entnehmen können, **geschieht nichts.**
Dies liegt daran, dass wir in unserem Arduino-Code darauf warten, dass der Computer zuerst mit dem Mikrocontroller kommuniziert.

Gib die Nummer 10 in das Nummernfeld ein, das an das serielle Objekt angehängt ist, und drücke die Eingabetaste. Unter der Annahme, dass das serielle Objekt das Byte korrekt übertragen hat, sollten Zahlen außerhalb des Objekts erscheinen. Aktiviere den Schalter mit der Bezeichnung 1, um die 'rohen' seriellen Daten in der Max Console zu drucken.

Udo matthias 07626-2 999 847
mobil: 017621-605276
http://www.udomatthias.com
https://www.facebook.com/udo.matthias.3
info@udomatthias.com
If it feels good, it must be in time!!

Formatieren der Daten

➔Wir werden feststellen, dass die Nummern, die in der Max Console von der 'rohen' seriellen Ausgabe angezeigt werden, wenig sinnvoll erscheinen.
Dies liegt daran, dass der Arduino seine **Werte als ASCII** überträgt, als ob sie auf einem Computer eingegeben worden wären. Somit wird die Zahl 15 als ASCII-Wert 49 ('1') gefolgt von ASCII-Wert 53 ('5') dargestellt.

Leerzeichen zwischen unseren drei Werten werden durch das ASCII für ein Leerzeichen (32) dargestellt, und jeder Satz von drei Zahlen wird durch einen Wagenrücklauf und einen Zeilenvorschub (ASCII 13 gefolgt von 10) abgeschlossen.
➔Um diese Werte in Max zu verwenden, müssen wir sie zuerst gruppieren und dann aus ASCII in eine für Menschen lesbare Zeichenfolge konvertieren.

Deaktiviere den Schalter mit der Bezeichnung 1 und aktiviere den Schalter mit der Bezeichnung 2. Dadurch wird die Ausgabe des zl-Gruppenobjekts im Patcher ausgedruckt.
Das zl-Objekt nimmt Sequenzen dieser ASCII-Werte und gruppiert sie in einer Liste. Das Auswahlobjekt über dem zl erzwingt, dass die Liste ausgegeben (und gelöscht) wird, wenn eine 13 (Wagenrücklauf) empfangen wird. Außerdem werden Zeilenvorschubzeichen (ASCII 10) aus dem Stream entfernt.
➔Da der rechte Ausgang von select alles ausgibt, was nicht vom Objekt ausgewählt wurde, fließen die restlichen Werte in das zl. Infolgedessen zeigt die Max Console eine Liste von ASCII-Werten an, die drei durch Leerzeichen getrennte Zahlen darstellen sollen (ASCII 32).

Deaktiviere den 2-Schalter und schalte den 3-Schalter ein. Dies zeigt die Ausgabe des itoa-Objekts, das die gruppierte Liste verwendet und die Ganzzahlen in ihre relevanten ASCII-Zeichen konvertiert. Infolgedessen sollte die Max-Konsole etwas anzeigen, **das angesichts unseres Arduino-Codes endlich Sinn macht:**
➔Wir sollten einen Wert zwischen 0 und 1000 sehen, gefolgt von einem zufälligen Wert gefolgt von 10 (dem Byte, das wir zuvor an den Mikrocontroller gesendet haben).

Das itoa-Objekt erstellt ein Max-Symbol, das von uns zum Debuggen gelesen werden kann, aber nicht ganz das ist, was wir für unseren Patcher benötigen.
Während wir erkennen, dass die Nachricht eine Liste von drei Ganzzahlen ist, werden Objekte wie das Entpacken dies nicht tun.
Das fromsymbol-Objekt analysiert ein Symbol und trennt verschiedene Datentypen basierend auf dem Leerraum.
Infolgedessen wird unser Symbol in eine Liste von drei Ganzzahlen umgewandelt, die das Entpackungsobjekt aufteilen kann.
➔Beachte, dass die Zahlenfelder aus dem Auspacken

Udo matthias 07626-2 999 847
mobil: 017621-605276
http://www.udomatthias.com
https://www.facebook.com/udo.matthias.3
info@udomatthias.com

If it feels good, it must be in time!!

unsere Werte korrekt anzeigen. Ändere das Nummernfeld oben erneut, und wir werden sehen, dass der dritte Wert aus dem seriellen Objekt entsprechend aktualisiert wird.

Schauen wir uns den Zeichnungsteil des Patchers an und sehen wie er funktioniert. Wie viele unserer lcd-basierten Beispiele visualisiert es einfach die Ausgabe des seriellen Objekts. In diesem Fall wird ein Diagramm erstellt, das auf dem Zähler (x), dem Zufallswert (y) und dem Echo-Byte (Farbe) basiert. **➔Diese Werte können leicht angewendet werden, um zu steuern, was wir möchten.**

Zusammenfassung

Das serielle Max-Objekt bietet die Möglichkeit, mit und von seriellen Standardgeräten innerhalb von Max zu kommunizieren. Es sendet und empfängt Daten als einzelne Bytes; Objekte wie zl, itoa und fromsymbol können verwendet werden, um alles zu verstehen.

Name	*Description*
Using Max with Hardware	Using Max with Hardware
serial	Send and receive from a serial port
itoa	Convert integers to UTF-8 (Unicode) character
fromsymbol	Convert a symbol into numbers/messages

Max Comm Tutorial 1: Human-Interface Devices

Max Tutorials: Table of Contents

Max Comm Tutorial 3: UDP Networking