Teil 3

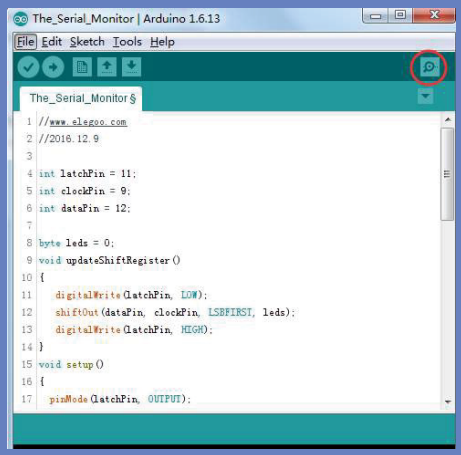
Lektion 3

Der Serielle Monitor

Übersicht

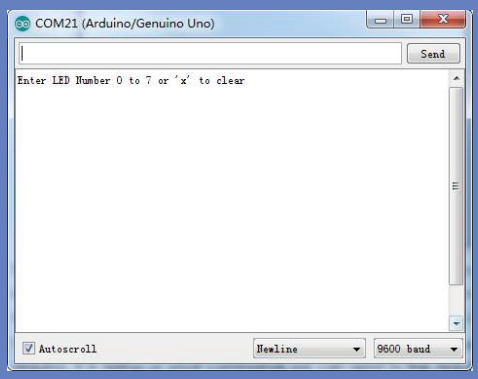
In dieser Lektion werden wir auf Lektion 22 teil2 aufbauen. Wir werden die Möglichkeit einbauen die LEDs vom Computer aus über den Seriellen Monitor zu steuern. Der Serielle Monitor ist die Verbindung zwischen Ihrem Computer und dem UNO Board. Durch ihn können Sie Textnachrichten senden und empfangen, was nützlich zum debuggen und zum steuern des UNO Boards sein kann. Zum Beispiel können Sie von Ihrem Computer einen Befehl senden, der die LEDs einschalten wird. In dieser Lektion werden die selben Teile und ein ähnliches Breadboard Layout wie in Lektion 22 teil2 benutzt. Lesen Sie zuerst Lektion 22 teil2, falls Sie es noch nicht getan haben.

Durchführung



- **Nachdem** Sie den Sketch auf Ihr Board hochgeladen haben, klicken Sie in der IDE auf das Symbol des Seriellen Monitors (unten in der Abbildung markiert).

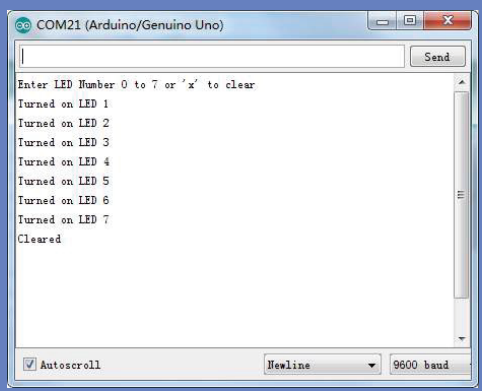
- **Das** folgende Fenster wird sich öffnen.
Die grundlegenden Informationen zum Seriellen Monitor haben Sie in Lektion 4 teil3 kennengelernt.



- **Dieses** Fenster ist der Serielle Monitor, der es Ihnen erlaubt sowohl Nachrichten von Ihrem Computer an das UNO Board zu senden, als auch Nachrichten vom UNO Board zu empfangen.

- **Die** erste Nachricht, die Sie bekommen, lautet: „Enter LED Number 0 to 7 or 'x' to clear“.Die Nachricht teilt uns mit, welche Befehle wir an das Board senden können: Ein„x“ schaltet alle LEDs aus. Eine beliebige Zahl von 0 – 7 schaltet die jeweilige LED ein, wobei 0 die LED ganz links ist und 7 die LED ganz rechts.

- **Versuchen** Sie die folgenden Befehle nacheinander über das Eingabefeld oben im Seriellen Monitor an Ihr Board zu senden. Drücken Sie nach jedem Befehl „Senden“: x 0 3 5



- **Der** erste Befehl (x) wird nichts auslösen, da die LEDs bereits alle ausgeschaltet sind. Beim Senden der Zahlen sollte jedoch die entsprechende LED aufleuchten und Sie sollten eine Bestätigungsnachricht vom Board im Seriellen Monitor erhalten. Die Anzeige im Seriellen Monitor sollte dann bei Ihnen ähnlich wie hier aussehen:

- **Geben** Sie erneut „x“ ein und klicken Sie auf „Senden“, um alle LEDs auszuschalten.

Code

- Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „**The_Serial_Monitor**“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 5 teil1 nocheinmal an.
- Wie Sie wahrscheinlich bereits geahnt haben, basiert unser Sketch auf dem Sketch aus Lektion 22 keil2. Also werden wir hier nur die neuen Teile aufgreifen.
- In der setup-Funktion gibt es drei neue Zeilen am Ende.

```
void setup()
{
pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
updateShiftRegister();
Serial.begin(9600);

while (! Serial); // Wait until Serial is ready - Leonardo
Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
}
```

- Zuerst haben wir den Befehl „Serial.begin (9600)“. Dieser leitet die Serielle Kommunikation mit einer Baudrate von 9600 ein. Die Baudrate ist die Geschwindigkeit der Datenübertragung. Sie können die Baudrate auf einen höheren Wert stellen, müssen dabei aber auch die Einstellung im Seriellen Monitor anpassen. Zunächst belassen Sie den wert aber bitte bei 9600.
- Die Zeile mit der „while“-Schleife wartet solange, bis eine Serielle Verbindung zum Computer hergestellt ist und fährt dann fort. Ohne diesen Code könnte es passieren, dass die erste Nachricht zu früh gesendet und daher nicht empfangen wird. Dies ist allerdings nur beim Arduino Leonardo notwendig, da beim UNO (ihrem Board) das Board automatisch zurückgesetzt wird, wenn Sie den Seriellen Monitor öffnen.
- Die letzte Zeile in der setup-Funktion sendet die „Begrüßungsnachricht“ aus, die wir später oben im Seriellen Monitor sehen werden.
- Die ganze Steuerung der LEDs passiert in der loop-Funktion:

```
void loop()
{
if (Serial.available())
{
char ch = Serial.read();
if (ch >= '0' && ch <= '7')
{
int led = ch - '0'; bitSet(leds, led);
updateShiftRegister();
Serial.print("Turned on LED ");
Serial.println(led);
}
if (ch == 'x')
{
leds = 0; updateShiftRegister();

Serial.println("Cleared");
}
}
}
```

- Alles, was innerhalb der loop-Funktion passiert, ist in einem if-Statement mit dem Parameter „Serial.available()“. Es wird also nur etwas passieren, wenn die Funktion Serial.available() „true“ (= wahr) zurückgibt.
- Serial.available() gibt true zurück, wenn Nachrichten über die Serielle Verbindung empfangen wurden. Eingehende Serielle Nachrichten werden in einem sogenannten „Buffer“ (= Puffer) gespeichert. Wenn dieser Buffer Nachrichten enthält, gibt Serial.available() true zurück.

- Wenn eine Nachricht empfangen wurde, kann man sie über diesen Befehl lesen:

```
char ch = Serial.read();
```

- Die Funktion „Serial.read()“ liest das nächste Zeichen aus dem Buffer und entfernt es aus diesem. Hier wird dieses Zeichen gleichzeitig der Variable „ch“ zugewiesen. Die ch-Variable ist eine Variable des Typs „char“, was für Charakter bzw Zeichen steht. Eine char-Variable kann also genau ein Zeichen abspeichern.
- Wenn Sie sich an die Instruktionen der ersten Nachricht im Seriellen Monitor halten, kann es sich bei dem empfangenden Character um den Buchstaben x oder eine Zahl zwischen 0 und 7 handeln.
- Das if-Statement in der nächsten Zeile überprüft ob es sich um eine Zahl zwischen 0 und 7 handelt, indem es überprüft ob es sich um eine Zahl handelt, die größer oder gleich 0 und gleichzeitig kleiner oder gleich 7 ist.
- Da es sich bei einem Zeichen auch um einen Buchstaben handeln kann, wird nicht direkt das Zeichen verglichen, sondern der sogenannte „ASCII“-Wert. (American Standard Code for Information Interchange). Jedes Zeichen hat einen einzigartigen vordefinierten ASCII-Wert in Form einer Zahl und diese Zahlen kann man vergleichen und so herausfinden, um welches Zeichen es sich handelt.
- Wenn das if-Statement wahr ist, kommen wir zur nächsten Zeile.

```
int led = ch – '0';
```

- Nun rechnen wir mit Charaktern! Wir subtrahieren die 0 vom Charakter, den die ch- Variable gerade enthält und speichern das Ergebnis in der Variable „led“. Wenn über die Serielle Verbindung eine 0 empfangen, wird (weil 0-0 = 0) eine 0 in der led- Variable gespeichert. Bei einer 7, würde 7-0 = 7 in der Variable gespeichert.
- Da wir nun die Nummer der LED wissen, die angeschaltet werden soll, können wir den entsprechenden Bit in der leds-Variable setzen und das Schieberegister updaten.

```
bitSet(leds, led);
updateShiftRegister();
```

- Die nächsten zwei Zeilen geben eine Bestätigungsnachricht an den Seriellen Monitor zurück.

```
Serial.print("Turned on LED ");
Serial.println(led);
```

- Die erste Zeile nutzt die Funktion „Serial.print()“ statt „Serial.println()“. Der unterschied zwischen den beiden Funktionen ist, dass die letztere zusätzlich zur angegebenen Nachricht noch einen Zeilenumbruch ausgibt, also die nächste Ausgabe in einer neuen Zeile anfangen wird. Erstere Funktion dagegen macht keinen Umbruch und man kann danach mit dem nächsten print-Befehl genau an der Stelle weiterschreiben, wo der letzte print-Befehl aufgehört hat. Wir benutzen in der ersten Zeile Serial.print(), da wir keinen Umbruch wollen, weil das nur der erste Teil der Nachricht ist. Mit der zweiten Zeile wird die LED-Nummer angehängt und die Nachricht ist fertig, weshalb wir Serial.println() nutzen, was am Ende automatisch in die nächste Zeile wechselt.
- Die Nummer der LED wird dabei in der led-Variable, eine Variable des Typs„int“ (Zahlenvariable) gespeichert. Eigentlich benötigt die Serial.println-Funktion einen „String“ (Text) als Parameter. Das ist aber nicht schlimm, da die Funktion int- Variablen automatisch in Text umwandelt.
- Nach dem if-Statement, dass sich um Zahleneingaben von 0 bis 7 kümmert, folgt das if-Statement, dass sich darum kümmert was passiert, wenn ein x empfangen wird (ch = x):

```
if (ch == 'x')
{
leds = 0;
updateShiftRegister();
Serial.println("Cleared");
}
```

- Bei einem „x“ werden alle Bits der leds-Variable auf 0 gesetzt, das Schieberegister geupdatet und eine Bestätigungsnachricht geschickt, dass alle LEDs ausgeschaltet wurden.