

*Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Arduino Nano
Teil 2.2: Serieller Monitor*

Benutzung des seriellen Monitors:

Wir wollen Daten vom Mikrocontroller zum Computer schicken und sie dort anzeigen.

Dazu müssen wir die Übertragungsfrequenz, die Baud Rate, festlegen, mit der die Daten übermittelt werden.

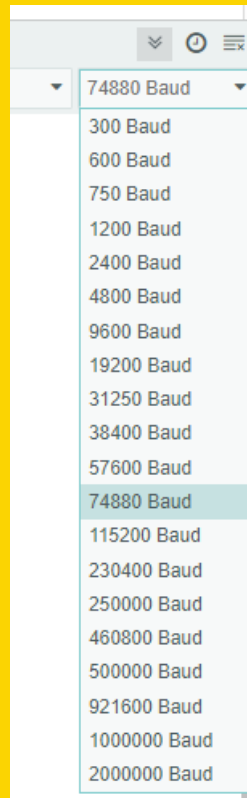
```
void setup() {  
  pinMode(LED, OUTPUT);  
  Serial.begin(74880); //Baud Rate
```

Für die Festlegung der Baud Rate benutzen wir die Funktion „Serial.begin()“. Als Parameter setzen wir 74880 ein.

(Eigentlich müssten es 76800 sein, ein Vielfaches von 9600. Aber das bietet der serielle Monitor der Arduino IDE nicht an...)

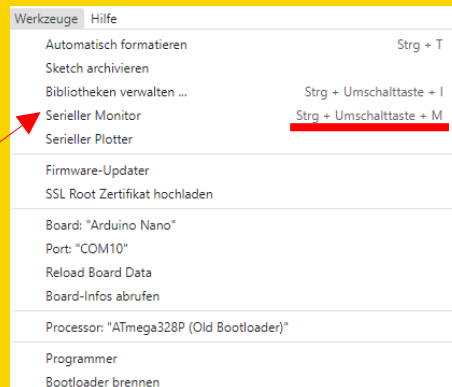
Der serielle Monitor lässt sich auf drei verschiedene Arten einschalten:

- Klicken auf den Button, oben rechts bei der Arduino IDE.
- Im Menü Werkzeuge auswählen auf Serieller Monitor klicken
- Strg + Umschalttaste + M auf dem Keyboard drücken



Daten, die auf dem Monitor ausgegeben werden.

Ausgabe Serieller Monitor x							
Nachricht (drücke Enter zum Senden für 'Arduino Nano' auf 'COM10')							
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247
248	249	250	251	252	253	254	255
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	



Das Programm:

Wir sollten immer eine Testschleife einbauen, in der wir **Teile** des Programms ausprobieren können.

Dazu benutzen wir den Befehl „while()“.

Solange die Aussage in den Klammern **wahr** ist, wird der Programmteil zwischen den geschweiften Klammern ausgeführt.

Eine Aussage ist dann **wahr**, wenn sie

NICHT NULL

ist.

Der Parameter der Testschleife ist jedoch Null, daher ist sie im Moment ausgeschaltet.

Unser Hauptprogramm beginnt mit Zeile 54.

Die bewirkt die Ausgabe einer Leerzeile im Monitor.

Dann folgt die Endlosschleife:

`while(1) { }` Der Parameter in der while-Schleife ist 1.

Eins ist **nicht Null**, also ist die Aussage **wahr**.

Also wird die Schleife immer ausgeführt.

```
47 void loop() {
48   /**Testschleife*****
49   while(0) {
50     Serial.println(123);
51   }
52   /**Ende Testschleife*****
53
54   Serial.print("\n"); //Neue Zeile
55   while (1) {
56     Serial.print(Counter++); //Die Variable auf dem Monitor ausgeben, anschließend um 1 erhöhen
57     Serial.print("\t"); //Spring zum nächsten Tabulator
58     Line_Counter = Line_Counter + 1; //Line_Counter um 1 erhöhen oder Line_Counter++
59     if (Line_Counter == 8) { //Acht Zahlen in eine Zeile
60       Serial.print("\n"); //In die nächste Zeile zum Anfang springen
61       Line_Counter = 0; //Line_Counter zurück setzen
62     }
63
64     /*Alternative Programmierung:*****
65     // for (byte i = 0; i < 8; i++) {
66     //   Serial.print(Counter++); //Die Variable auf dem Monitor ausgeben, anschließend um 1 erhöhen
67     //   Serial.print("\t"); //Spring zum nächsten Tabulator
68     // }
69     // Serial.print("\n"); //In die nächste Zeile zum Anfang springen
70     /*Alternative Programmierung Ende*****
71
72     //At Pin 5 PORTB the LED of the Arduino board is mounted (Pin 13)
73     digitalWrite(LED, HIGH); //Push PORTB pin 5 high (LED, pin 13 "Nano")
74     _delay_ms(50);
75     digitalWrite(LED, LOW); //Push PORTB pin 5 low (LED, pin 13 "Nano")
76     _delay_ms(50);
77     /*Alternative Programmierung:*****
```

Im Programm verwenden wir zwei Variablen:
„Counter“ und „Line Counter“.

Damit der Mikrocontroller damit arbeiten kann, müssen wir sie ihm bekannt machen, deklarieren. Das geschieht in den Zeilen 35 und 36.

In Zeile 56 geben wir auf dem Monitor den aktuellen Wert von „Counter“ aus und erhöhen den Wert anschließend um 1.

Variable++ ist die Abkürzung von Variable = Variable + 1.

(= ist dabei natürlich kein Gleichheitszeichen im mathematischen Sinn, sondern es weist der Variablen einen Wert zu.)

In Zeile 57 geben wir einen Tabulator aus.

In Zeile 58 erhöhen wir die Variable Line_Counter um 1.

(geht natürlich auch mit Line_Counter++)

if() Anweisung:

Wenn die Aussage in den runden Klammern wahr ist, werden die Anweisungen zwischen den geschweiften Klammern ausgeführt. Die Aussage ist dann wahr, wenn Line_Counter den Wert 8 angenommen hat. **Unbedingt darauf achten, dass in einem Vergleich == benutzt wird, denn = ist ja eine Zuweisung!**

Dann wird in Zeile 60 der Steuerbefehl „\n“ an den Monitor übergeben, was eine neue Zeile auf dem Monitor generiert. Anschließend wird Line_Counter wieder auf Null gesetzt.

```
34 #define LED 13 //LED of the Arduino board
35 byte Counter;
36 byte Line_Counter;
37
```

```
47 void loop() {
48   /**Testschleife*****
49   while(0) {
50     Serial.println(123);
51   }
52   /**Ende Testschleife*****
53
54   Serial.print("\n"); //Neue Zeile
55   while (1) {
56     Serial.print(Counter++); //Die Variable auf dem Monitor ausgeben, anschließend um 1 erhöhen
57     Serial.print("\t"); //Spring zum nächsten Tabulator
58     Line_Counter = Line_Counter + 1; //Line_Counter um 1 erhöhen oder Line_Counter++
59     if (Line_Counter == 8) { //Acht Zahlen in eine Zeile
60       Serial.print("\n"); //In die nächste Zeile zum Anfang springen
61       Line_Counter = 0; //Line_Counter zurück setzen
62     }
63
64     /*Alternative Programmierung:*****
65     // for (byte i = 0; i < 8; i++) {
66     //   Serial.print(Counter++); //Die Variable auf dem Monitor ausgeben, anschließend um 1 erhöhen
67     //   Serial.print("\t"); //Spring zum nächsten Tabulator
68     // }
69     // Serial.print("\n"); //In die nächste Zeile zum Anfang springen
70     /*Alternative Programmierung Ende*****
71
72     //At Pin 5 PORTB the LED of the Arduino board is mounted (Pin 13)
73     digitalWrite(LED, HIGH); //Push PORTB pin 5 high (LED, pin 13 "Nano")
74     _delay_ms(50);
75     digitalWrite(LED, LOW); //Push PORTB pin 5 low (LED, pin 13 "Nano")
76     _delay_ms(50);
77     /*Alternative Programmierung:*****
```

Was sehen wir auf dem Monitor?

Es werden 8 Spalten mit Zahlen von 0 bis 255 dargestellt.

Dann fängt die Zählung wieder von vorne an.

Die Aufteilung in 8 Spalten kommt durch die Variable

`Line_Counter` (die natürlich eigentlich Column-Counter heißen müsste).

Immer wenn sie den Wert 8 hat, wird sie ja auf Null zurück gesetzt und gleichzeitig ein Zeilensprung ausgelöst.

Du kannst ja mal den Wert verändern und schauen, wie deine Liste dann aussieht.

Aber woher kommt es, dass die Zahlen bei 255 aufhören zu wachsen und wieder bei Null anfangen?

Davon steht ja nichts im Programm...

Wir haben Counter als „byte“ deklariert.

Ein Byte besteht bei unserem Mikrocontroller aus 8Bits.

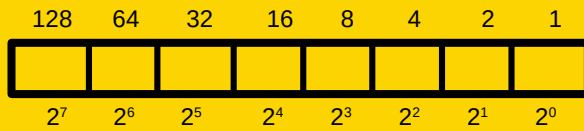
Es ist ein 8Bit Prozessor. Mehr als 8Bit kann er nicht auf ein Mal verarbeiten.

```
34 #define LED 13 //LED of the Arduino board
35 byte Counter;
36 byte Line_Counter;
37
```

Ausgabe Serieller Monitor x							
Nachricht (drücke Enter zum Senden für 'Arduino Nano' auf 'COM10')							
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247
248	249	250	251	252	253	254	255
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	

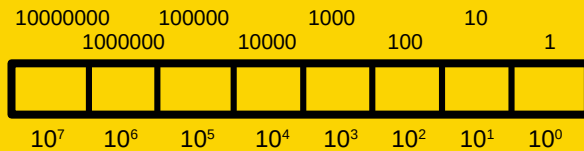
8Bits eines Registers ist eine Zahl die aus 8 Nullen und Einsen besteht.
Dabei haben die Bits entsprechend ihrer Stellung einen Wert:

Stellenwerte der Bits



Das ist Vergleichbar mit dem gewohnten Dezimalsystem, in dem wir statt nur die Ziffern 0 und 1, die Ziffern 0 bis 9 zur Verfügung haben:

Stellenwerte von Dezimalzahlen



So hat die Binär-Zahl

0b10101100 den Dezimalwert (0b kennzeichnet eine Ziffernfolge als binär):

$$1*128 + 0*64 + 1*32 + 0*16 + 1*8 + 1*4 + 0*2 + 0*1 = 172$$

Die Dezimal-Zahl

0d5024 besteht aus **5** Tausendern, **0** Hundertern, **2** Zehnern und **4** Einern und hat damit den Dezimalwert **5024** (0d kennzeichnet eine Ziffernfolge als dezimal)

Sind alle Stellen des Bytes mit einer 1 voll, dann hat es den Wert 255.

Wenn wir jetzt noch eine 1 addieren dann ergibt das die Binär-Zahl **0b100000000**. Die Zahl ist aber mit 8 Bit nicht darstellbar, die vordere 1 fällt weg. Das Ergebnis ist eine Null!

Ausgabe Serieller Monitor x

Nachricht (drücke Enter zum Senden für 'Arduino Nano' auf 'COM10')

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247
248	249	250	251	252	253	254	255
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	

Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Arduino Nano
Teil 2.3: Interrupt