

*Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Elegoo Car Kit
Teil 2.8: Time Saving*

Von Charlotte und Andreas

Berechnung der Ziffern

Beispiel:

$213 / 100 = 2,13$. Ganzzahl = 2.

$213 \% 100$ (sprich: modulo) = 13.

$13 / 10 = 1,3$. Ganzzahl = 1.

$213 \% 10 = 3$.

Alt:

Die berechnete Ziffer wird sofort übertragen. Der Mikrocontroller **wartet** jedesmal, bis diese Übertragung beendet ist! Und das drei Mal, mal Anzahl der Werte.

Neu:

Die Daten werden nur berechnet und gespeichert. Übertragen geschieht im **Interrupt**.

```
.09_Time_Saving - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug Browser
Hex Arduino ATmega328P
Time.cpp* Interrupt_Service_Routines.h Drive_Functions.h USART_Functions.h Init.h
-> USART_Functions.h C:\Users\User\Desktop\Tutorials\Atmel\Software\Elegoo_09_Time_Saving\Elegoo_09_Time_Saving\USART
23 //Takes about 450us at 76800 Baud rate, who has so much time?
24 void Transmit_literal (char data) { //Transmit 8-Bit literal
25     char data_100 = data / 100; //How much hundred
26     char data_10 = data % 100 / 10; //How much ten
27     char data_1 = data % 10; //How much one
28     if (!data_100) { //There is no hundred
29         USART_Tra
30         if (!data
31         else USART
32     }
33     else {
34         USART_Transmit(data_100 + 48); //Hundred
35         USART_Transmit(data_10 + 48); //ten
36     }
37     USART_Transmit(data_1 + 48); //One
38     USART_Transmit('\t'); //Tabulator
39 }
40 //If we want to do it very, very much faster, we have to divide the transmission into single digits
41 //The transmission is not faster!!! But we don't wait while transmitting, we only initialize it!!!
42 //Initialisation every 256us.
43 void Computing_Transmission_Values (unsigned char transmitting_data) { //
44     Trans_data[hundred] = transmitting_data / 100; //It's the same
45     Trans_data[ten] = transmitting_data % 100 / 10; //dito
46     Trans_data[one] = (transmitting_data % 10) + 48;
47     if (!Trans_data[hundred]) { //Now we have to sav
48         Trans_data[hundred] = 32; //blank
49         if (!Trans_data[ten]) Trans_data[ten] = 32; //blank
50         else Trans_data[ten] = Trans_data[ten] + 48; //ASCII for zero is 48
51     }
52     else {
53         Trans_data[hundred] = Trans_data[hundred] + 48; //See above
54         Trans_data[ten] = Trans_data[ten] + 48; //dito
55     }
56     //But there is no transmission!!!! See "ISR(TIMER2_OVF_vect)"!
57 }
```

Zeitschema mit Hilfe von Timer 2:

Alle 128 Mikrosekunden(us) Timer 2 Interrupt

Zwei Zähler des Interrupts:

T2ck alle 128 us Inkrement $\rightarrow 2000 * 128 \sim 250\text{ms}$

Counter_US ebenfalls $\rightarrow 256 * 128 \sim 32\text{ms}$

USART-Transmission ca. 120us

T2ck & 0b00000001 \rightarrow jedes zweite Mal

$\rightarrow 2 * 128\text{us} = 256\text{us}$

Vier Uhren:

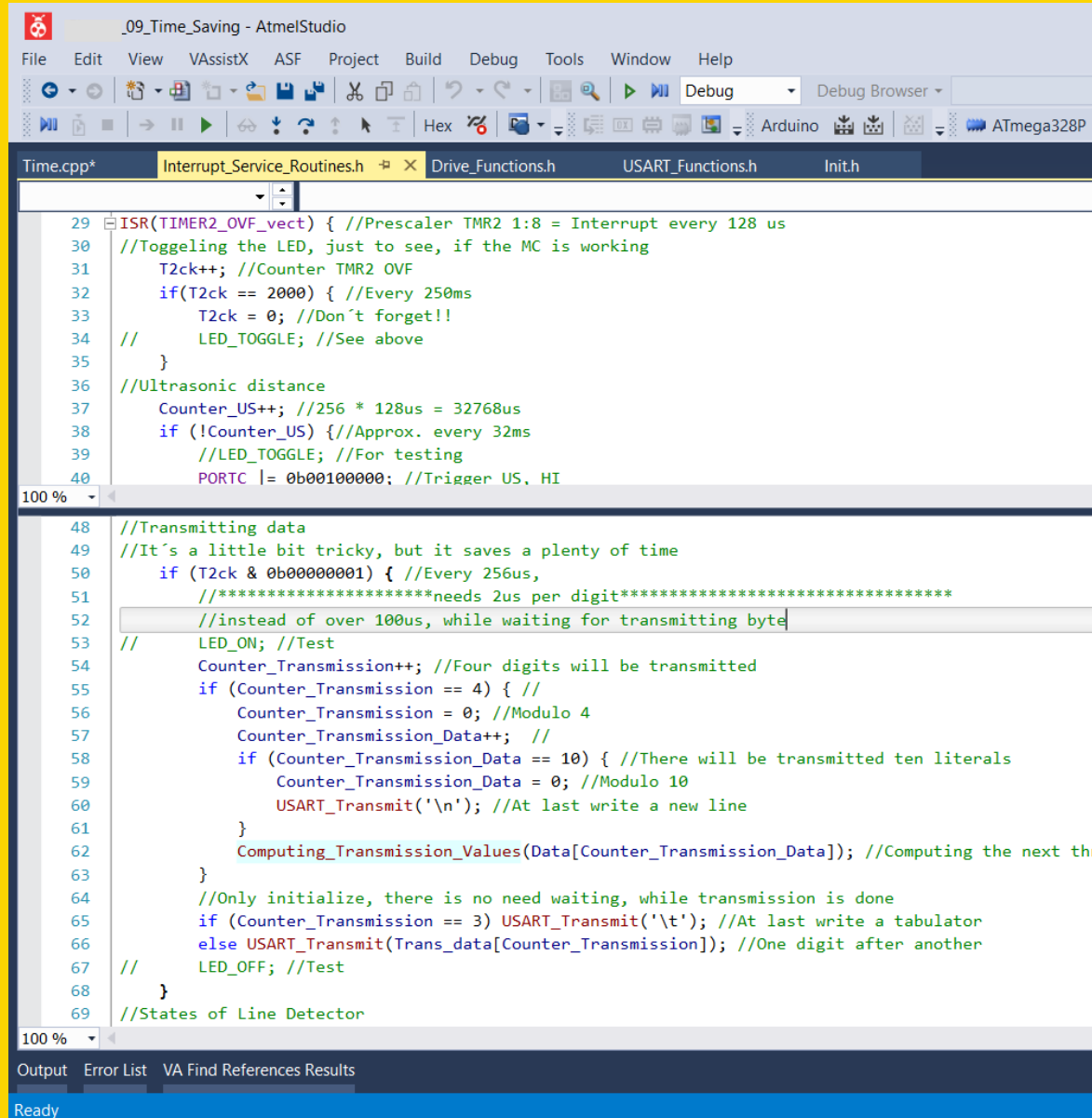
128 Mikrosekundentakt, kleinste Einheit

256 Mikrosekundentakt

32 Millisekundentakt

250 Millisekundentakt

```
36
37 //Definitions
38 #define FOSC           16000000 // Clock Speed
39 #define BAUD           76800//Baud rate, normally used by Arduino 9600
40 #define MYUBRR        FOSC/16/BAUD-1 //Calculate my baud rate
41 #define PWMA           OCR0A //Overflow compare register A
```



```
09_Time_Saving - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug Browser
Time.cpp* Interrupt_Service_Routines.h Drive_Functions.h USART_Functions.h Init.h
29 ISR(TIMER2_OVF_vect) { //Prescaler TMR2 1:8 = Interrupt every 128 us
30 //Toggeling the LED, just to see, if the MC is working
31 T2ck++; //Counter TMR2 OVF
32 if(T2ck == 2000) { //Every 250ms
33 T2ck = 0; //Don't forget!!
34 // LED_TOGGLE; //See above
35 }
36 //Ultrasonic distance
37 Counter_US++; //256 * 128us = 32768us
38 if (!Counter_US) { //Approx. every 32ms
39 //LED_TOGGLE; //For testing
40 PORTC |= 0b00100000; //Trigger US, HI
41
42
43
44
45
46
47
48 //Transmitting data
49 //It's a little bit tricky, but it saves a plenty of time
50 if (T2ck & 0b00000001) { //Every 256us,
51 //*****needs 2us per digit*****
52 //instead of over 100us, while waiting for transmitting byte
53 // LED_ON; //Test
54 Counter_Transmission++; //Four digits will be transmitted
55 if (Counter_Transmission == 4) { //
56 Counter_Transmission = 0; //Modulo 4
57 Counter_Transmission_Data++; //
58 if (Counter_Transmission_Data == 10) { //There will be transmitted ten literals
59 Counter_Transmission_Data = 0; //Modulo 10
60 USART_Transmit('\n'); //At last write a new line
61 }
62 Computing_Transmission_Values(Data[Counter_Transmission_Data]); //Computing the next th
63 }
64 //Only initialize, there is no need waiting, while transmission is done
65 if (Counter_Transmission == 3) USART_Transmit('\t'); //At last write a tabulator
66 else USART_Transmit(Trans_data[Counter_Transmission]); //One digit after another
67 // LED_OFF; //Test
68 }
69 //States of Line Detector
```

Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Elegoo Car Kit
Teil 2.9: Analog Digital Converter

Von Charlotte und Andreas