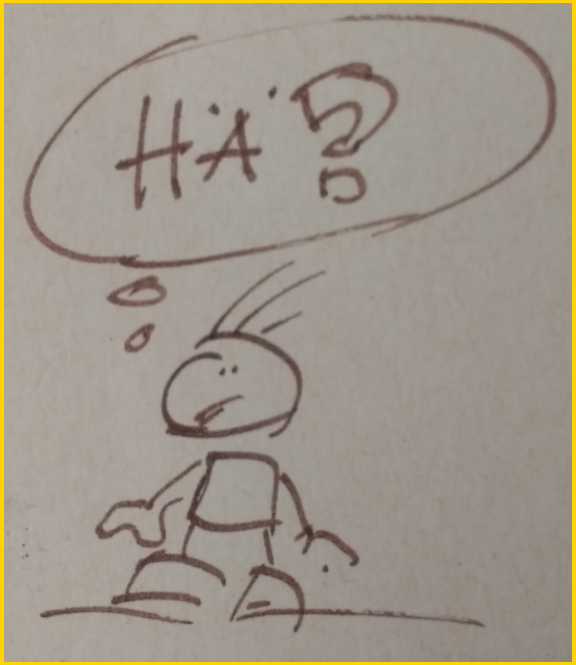
Montage und Programmierung eines Roboters für den Hessen SolarCup Disziplin: SolaRobot Teil 2.3: Data Visualizer

Von Charlotte und Andreas

Solution Explorer

Transmit_literal

C++



Header

USART

for

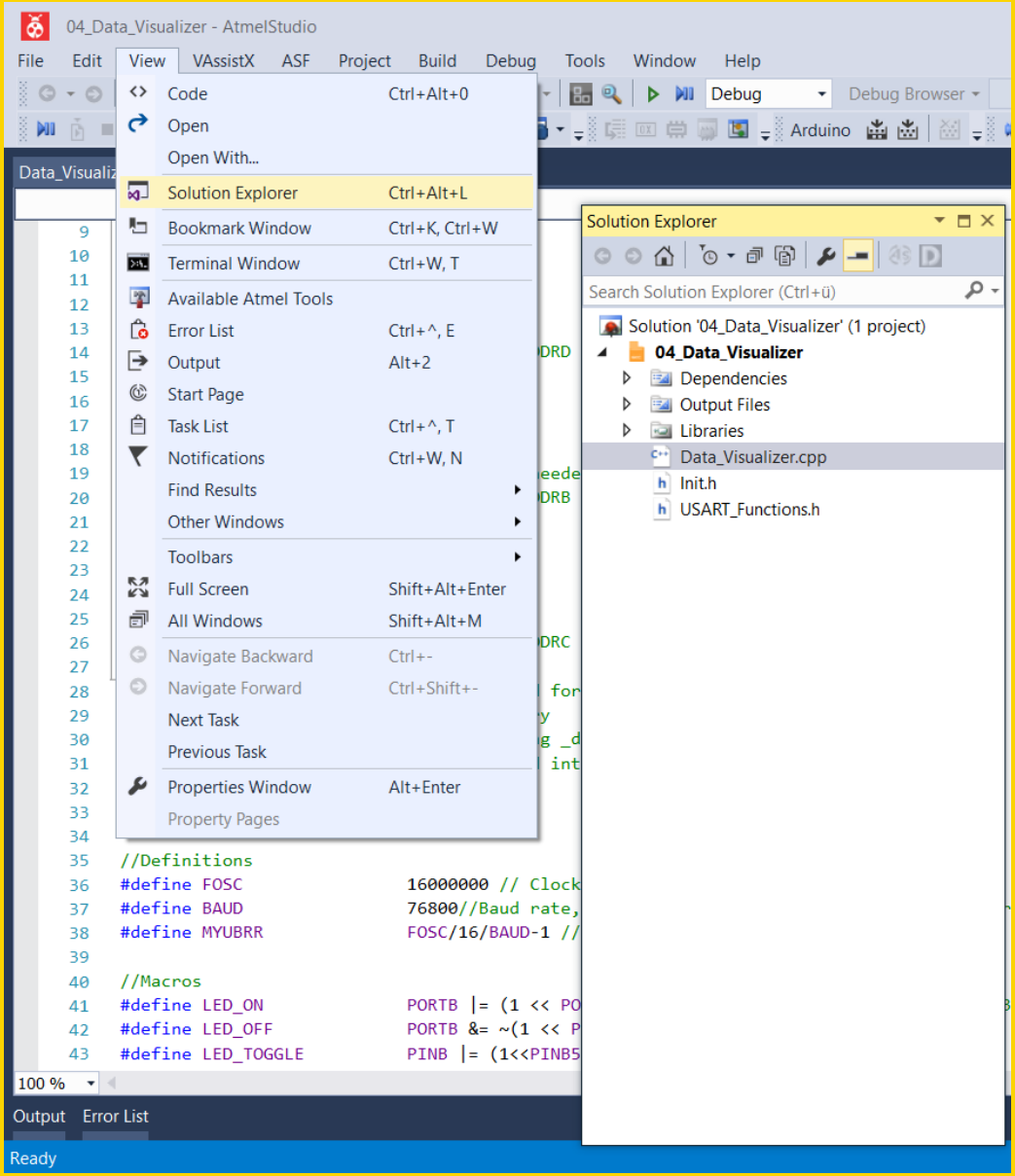
Integer

Init.h

Array

Data Visualizer

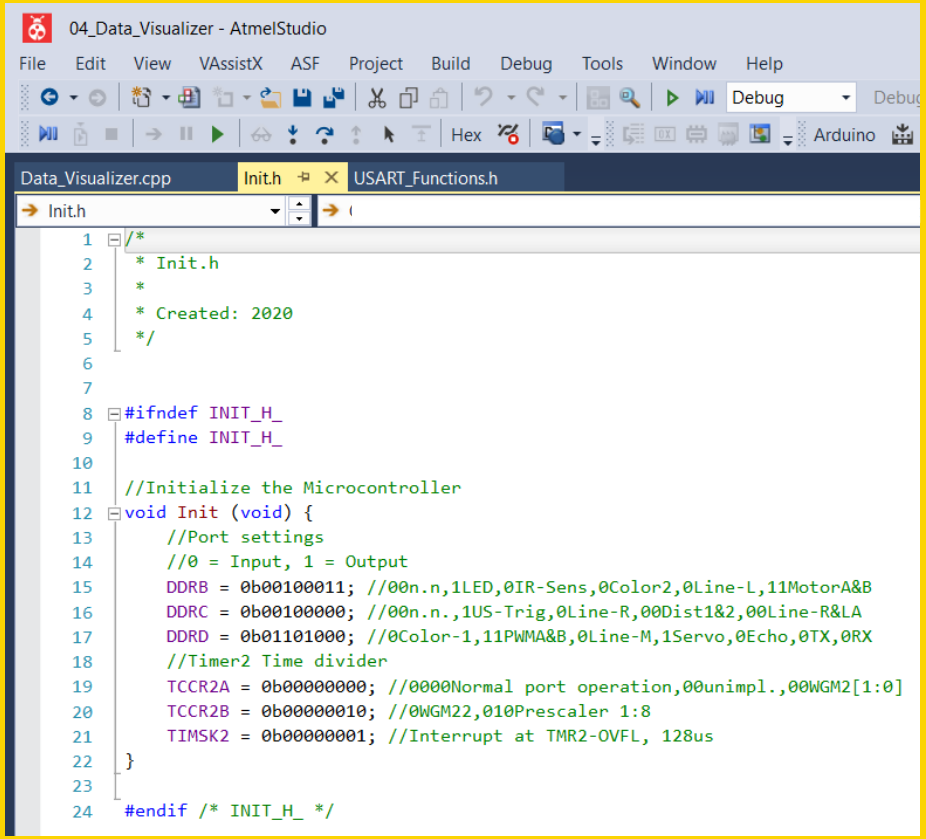
Data[7]



Header:

Init.h

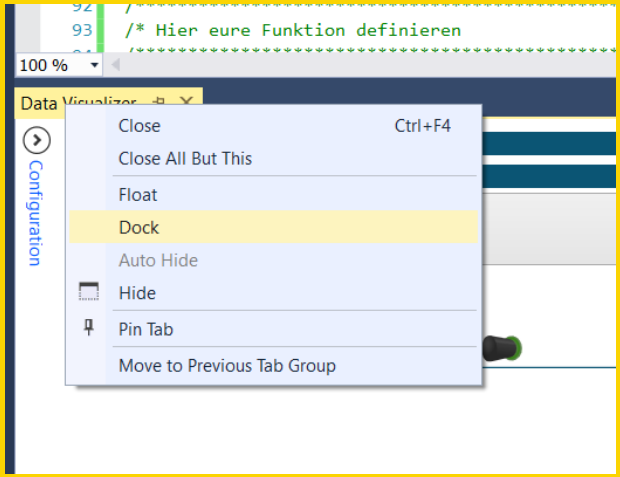
USART_Functions.h



Data Visualizer:

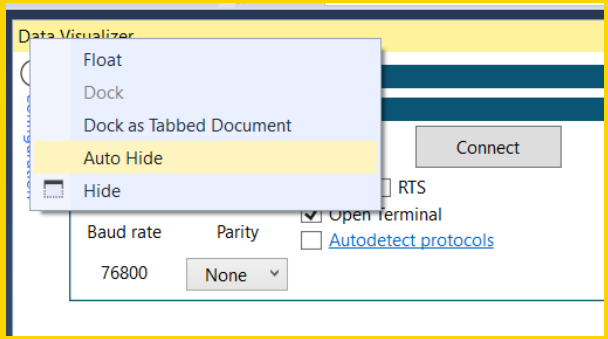
Baud rate muss geändert werden
von **9600** auf **76800**

Rechtsklick auf Data Visualizer

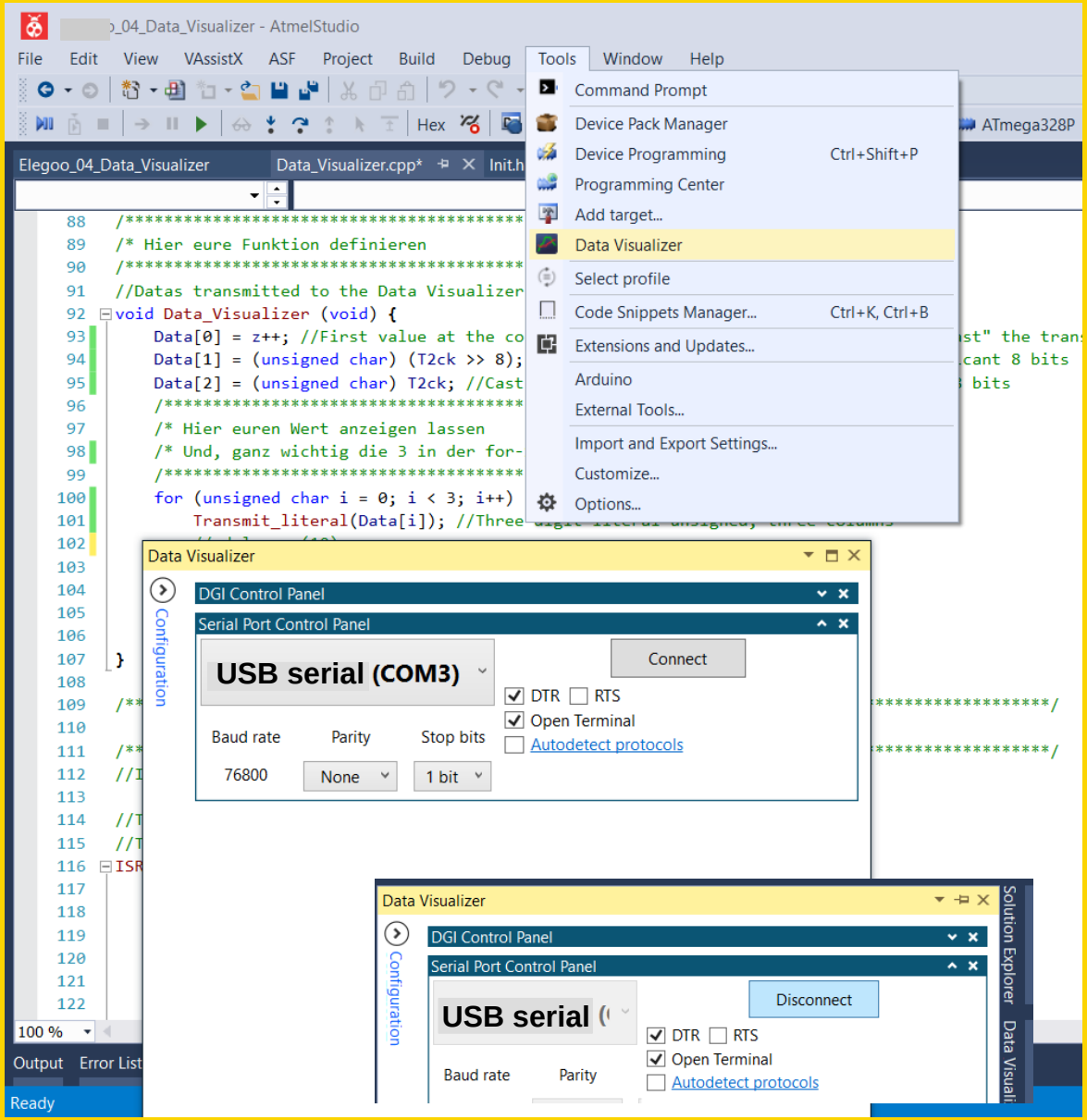


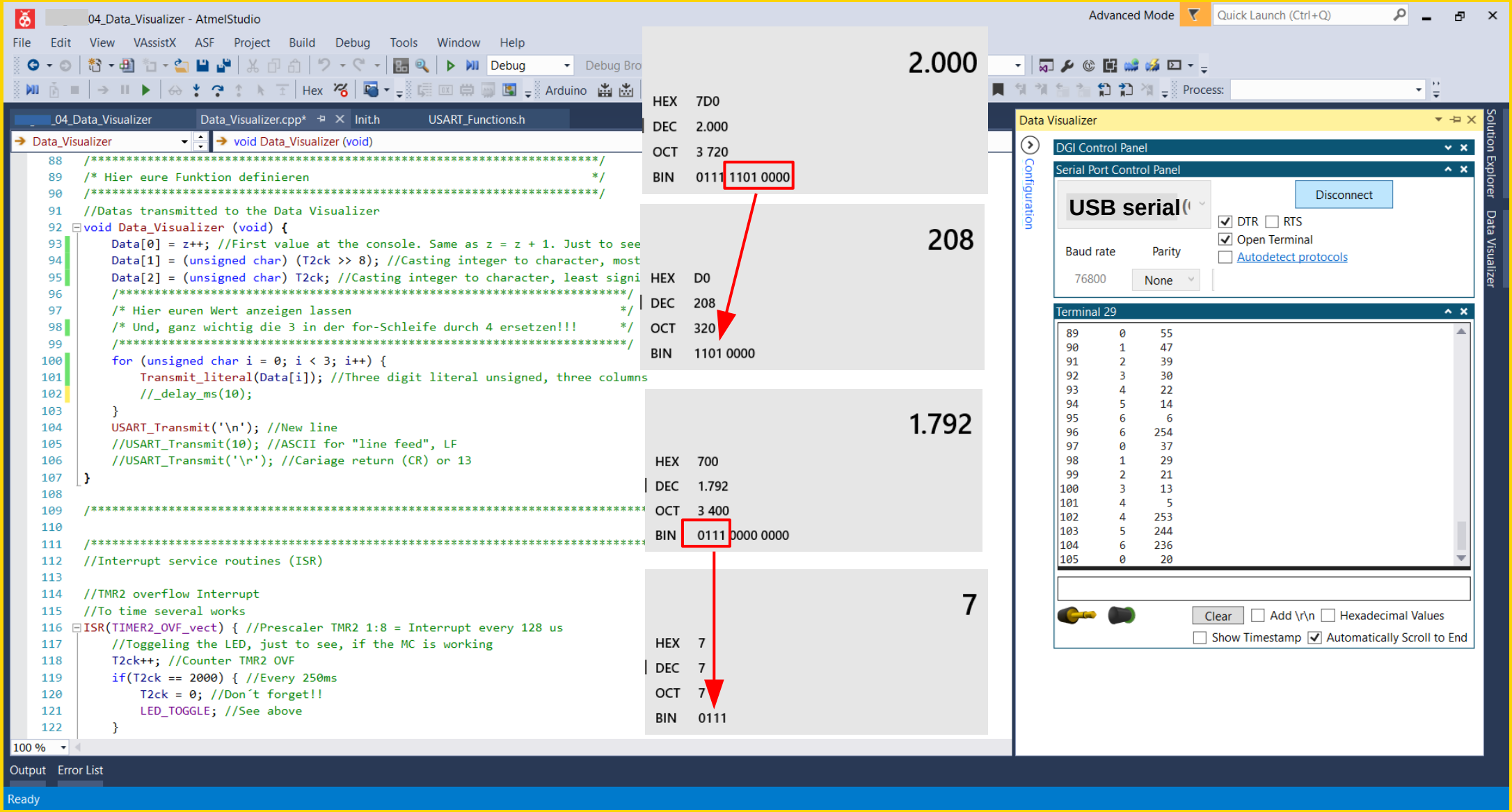
Klick Dock

Rechtsklick auf Data Visualizer



Klick Auto Hide




04_Data_Visualizer - AtmelStudio
Advanced Mode
Quick Launch (Ctrl+Q)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Hex
Arduino
Process:
04_Data_Visualizer
Data_Visualizer.cpp*
Init.h
USART_Functions.h
Data_Visualizer
void Data_Visualizer (void)
/* Hier eure Funktion definieren */
//Datas transmitted to the Data Visualizer
void Data_Visualizer (void) {
Data[0] = z++; //First value at the console. Same as z = z + 1. Just to see
Data[1] = (unsigned char) (T2ck >> 8); //Casting integer to character, most
Data[2] = (unsigned char) T2ck; //Casting integer to character, least signi
/* Hier euren Wert anzeigen lassen */
/* Und, ganz wichtig die 3 in der for-Schleife durch 4 ersetzen!!! */
for (unsigned char i = 0; i < 3; i++) {
Transmit_literal(Data[i]); //Three digit literal unsigned, three columns
//_delay_ms(10);
}
USART_Transmit('\n'); //New line
//USART_Transmit(10); //ASCII for "line feed", LF
//USART_Transmit('\r'); //Cariage return (CR) or 13
}
//Interrupt service routines (ISR)
//TMR2 overflow Interrupt
//To time several works
ISR(TIMER2_OVF_vect) { //Prescaler TMR2 1:8 = Interrupt every 128 us
//Toggeling the LED, just to see, if the MC is working
T2ck++; //Counter TMR2 OVF
if(T2ck == 2000) { //Every 250ms
T2ck = 0; //Don´t forget!!
LED_TOGGLE; //See above
}
100 %
Output Error List
Ready
2.000
HEX 7D0
DEC 2.000
OCT 3 720
BIN 0111 1101 0000
208
HEX D0
DEC 208
OCT 320
BIN 1101 0000
1.792
HEX 700
DEC 1.792
OCT 3 400
BIN 0111 0000 0000
7
HEX 7
DEC 7
OCT 7
BIN 0111
Data Visualizer
Configuration
DGI Control Panel
Serial Port Control Panel
USB serial
Disconnect
DTR RTS
Open Terminal
Autodetect protocols
Baud rate
Parity
76800
None
Terminal 29
89 0 55
90 1 47
91 2 39
92 3 30
93 4 22
94 5 14
95 6 6
96 6 254
97 0 37
98 1 29
99 2 21
100 3 13
101 4 5
102 4 253
103 5 244
104 6 236
105 0 20
Clear
Add \r\n
Hexadecimal Values
Show Timestamp
Automatically Scroll to End
Solution Explorer
Data Visualizer

Header einbinden

```
 *A2    Ana     PC2     Dist-1  In
 *A3    Ana     PC3     Dist-2  In
 *A4    Digit   PC4     Line-R  In
 *A5    Digit   PC5     US-Trig Out ->  DDRC = 0b00100000
 */
#define F_CPU 16000000UL //16MHz required for delay
#include <avr/io.h> //Input/Output library
#include <util/delay.h> //Needed for using _delay_...
#include <avr/interrupt.h> //External and internal Interrupts
#include "Init.h"
#include "USART_Functions.h"

//Definitions
#define FOSC                16000000 // Clock Speed
#define BAUD                76800//Baud rate, normally used by Arduino 9600, possible
#define MYUBRR              FOSC/16/BAUD-1 //Calculate my baud rate
```

Data Visualizer

```
//Datas transmitted to the Data Visualizer
void Data_Visualizer (void) {
    Data[0] = z++; //First value at the console. Same as z = z + 1. Just to see how "
    Data[1] = (unsigned char) (T2ck >> 8); //Casting integer to character, most signi
    Data[2] = (unsigned char) T2ck; //Casting integer to character, least significant
    /*************************************************************************/
    /* Hier euren Wert anzeigen lassen                                       */
    /* Und, ganz wichtig die 3 in der for-Schleife durch 4 ersetzen!!!       */
    /*************************************************************************/
    for (unsigned char i = 0; i < 3; i++) {
        Transmit_literal(Data[i]); //Three digit literal unsigned, three columns
        //_delay_ms(10);
    }
    USART_Transmit('\n'); //New line
    //USART_Transmit(10); //ASCII for "line feed", LF
    //USART_Transmit('\r'); //Cariage return (CR) or 13
}
```

Integer 0 - 65535

1 1 0 0 1 0 1 0 | 0 0 1 1 0 1 0 0 = 51764

HEX	FFFF
DEC	65.535
OCT	177 777
BIN	1111 1111 1111 1111

```
unsigned int T2ck; //Counter TMR2 overflow
/*************************************************************************/
/* Hier eure Variable (unsigned char) deklarieren                        */
/*************************************************************************/

//unsigned char Data[10]; //Number of datas displayed on Data Visualizer

//Funktions, declaration
void Init (void); //Initialize Microcontroller
//void USART_Transmit(unsigned char); //Transmit ASCII to Data Visualizer
void USART_Init(unsigned int); //Initialize USART Transmission/Reception
//void Transmit_literal (char); //Transmit literal to Data Visualizer Mon
//void Transmit_String (const char); //Transmit letters to Data Visualize
void Data_Visualizer (void); //Define datas to be vizualized
```

Array (Feld):

Data[10]
10 Elemente mit gleichem Namen
aber unterschiedlichen Indizes.
Data[0], Data[1] … Data[9]
Ein Array beginnt immer mit Element [0]

void Funktionsname (Variable)

Beim Aufruf der Funktion wird ein Wert mit übergeben an die Funktion.

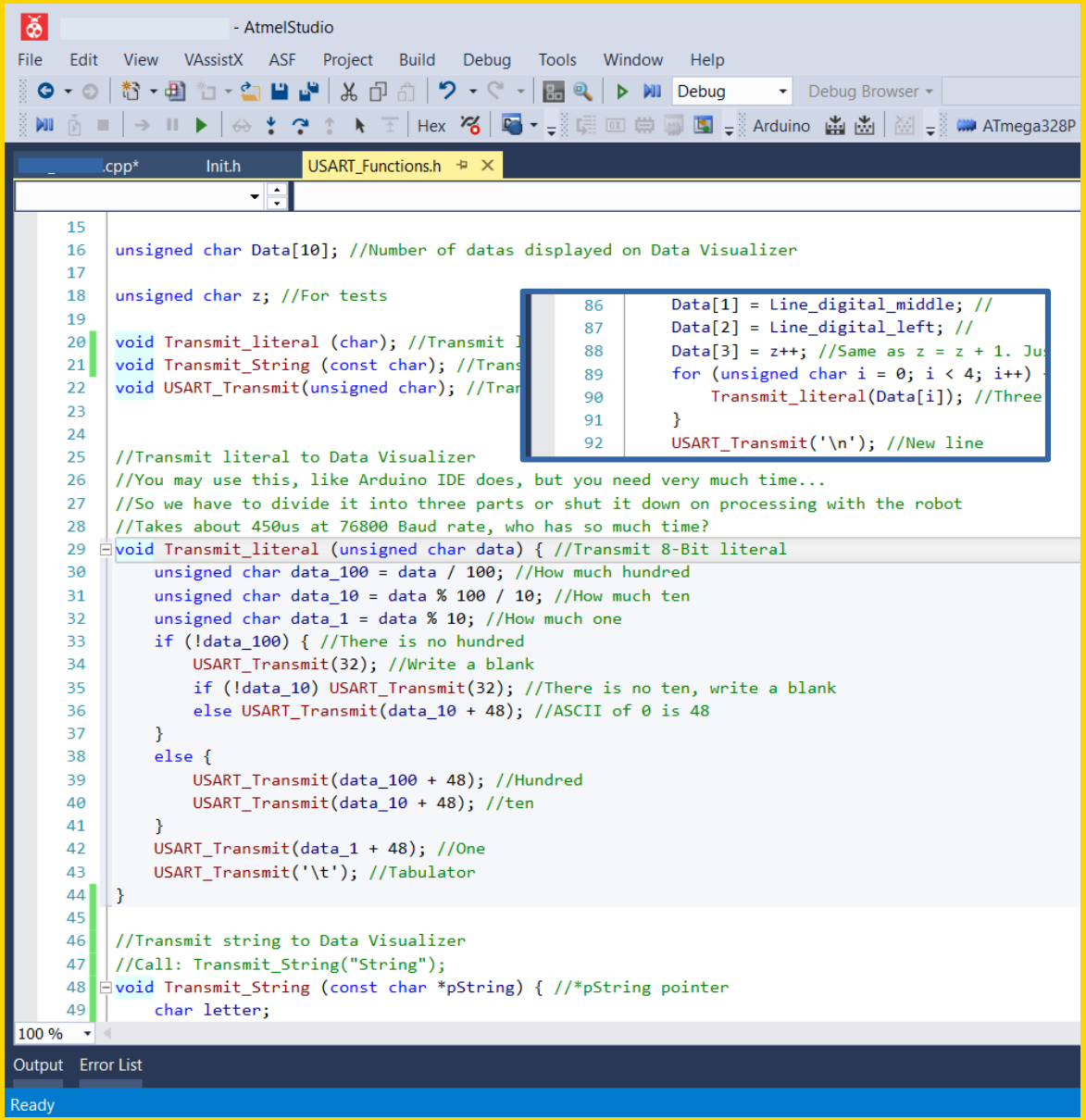
Beispiel:

```
…
      mal_drei_nehmen(8);
…

void mahl_drei_nehmen(Faktor) {
      Produkt = 3 * Faktor;
}
```

In Produkt ist jetzt der Wert 24 gespeichert.

Aufgabe: Implementiert (d.h.: baut in das bestehende Programm ein)
eine Funktion mit der ihr Werte verdoppelt, die an die Funktion übergeben wird. Und zeigt sie anschließend an.
Das alles soll im Hauptfenster (Data_Visualizer.cpp) geschehen.
(Hilfe: Zuerst eine Variable deklarieren, dann im Deklarationsteil die Funktion bekannt machen, [denkt daran, dass ihr einen unsigned char übergeben wollt] Unterhalb der main die Funktion definieren, { } nicht vergessen. In der main aufrufen. Im Data_Visualizer ausgeben.)
Schaut euch die Ergebnisse an. Was fällt euch auf?



Erst Aufgabe lösen, dann weiter schauen!

American Standard Code for Information Interchange (ASCII)

04_Data_Visualizer - AtmelStudio

File Edit View VAssistX ASF Project Build Debug Tools Window Help

Debug | Debug Browser | led | Hex | Arduino | ATmega328P | Simulator

Data_Visualizer.cpp* | Init.h | USART_Functions.h

```
13
14  //unsigned char z; //For tests
15
16  void Transmit_literal (char); //Transmit literal to Data Visualizer Monitor
17  void Transmit_String (const char); //Transmit letters to Data Visualizer Monitor
18  void USART_Transmit(unsigned char); //Transmit ASCII to Data Visualizer
19
20
21  //Transmit literal to Data Visualizer
22  //You may use this, like Arduino IDE does, but you need very much time...
23  //So we have to divide it into three parts or shut it down on processing with the robot
24  //Takes about 450us at 76800 Baud rate, who has so much time?
25  void Transmit_literal (unsigned char data) { //Transmit 8-Bit literal
26      unsigned char data_100 = data / 100; //How much hundred
27      unsigned char data_10 = data % 100 / 10; //How much ten
28      unsigned char data_1 = data % 10; //How much one
29      if (!data_100) { //There is no hundred
30          USART_Transmit(32); //Write a blank
31          if (!data_10) USART_Transmit(32); //There is no ten, write a blank
32          else USART_Transmit(data_10 + 48); //ASCII of 0 is 48
33      }
34      else {
35          USART_Transmit(data_100 + 48); //Hundred
36          USART_Transmit(data_10 + 48); //ten
37      }
38      USART_Transmit(data_1 + 48); //One
39      USART_Transmit('\t'); //Tabulator...
40      //USART_Transmit(9); //...or ASCII for horizontal tab, HT
41  }
42
43  //Transmit string to Data Visualizer
44  //Call: Transmit_String("String");
45  void Transmit_String (const char *pString) { //*pString pointer
46      char letter; //Local variable
47      letter = *pString; // The content of the registers, the pointer is
```

100 %

Output | Error List

Ready

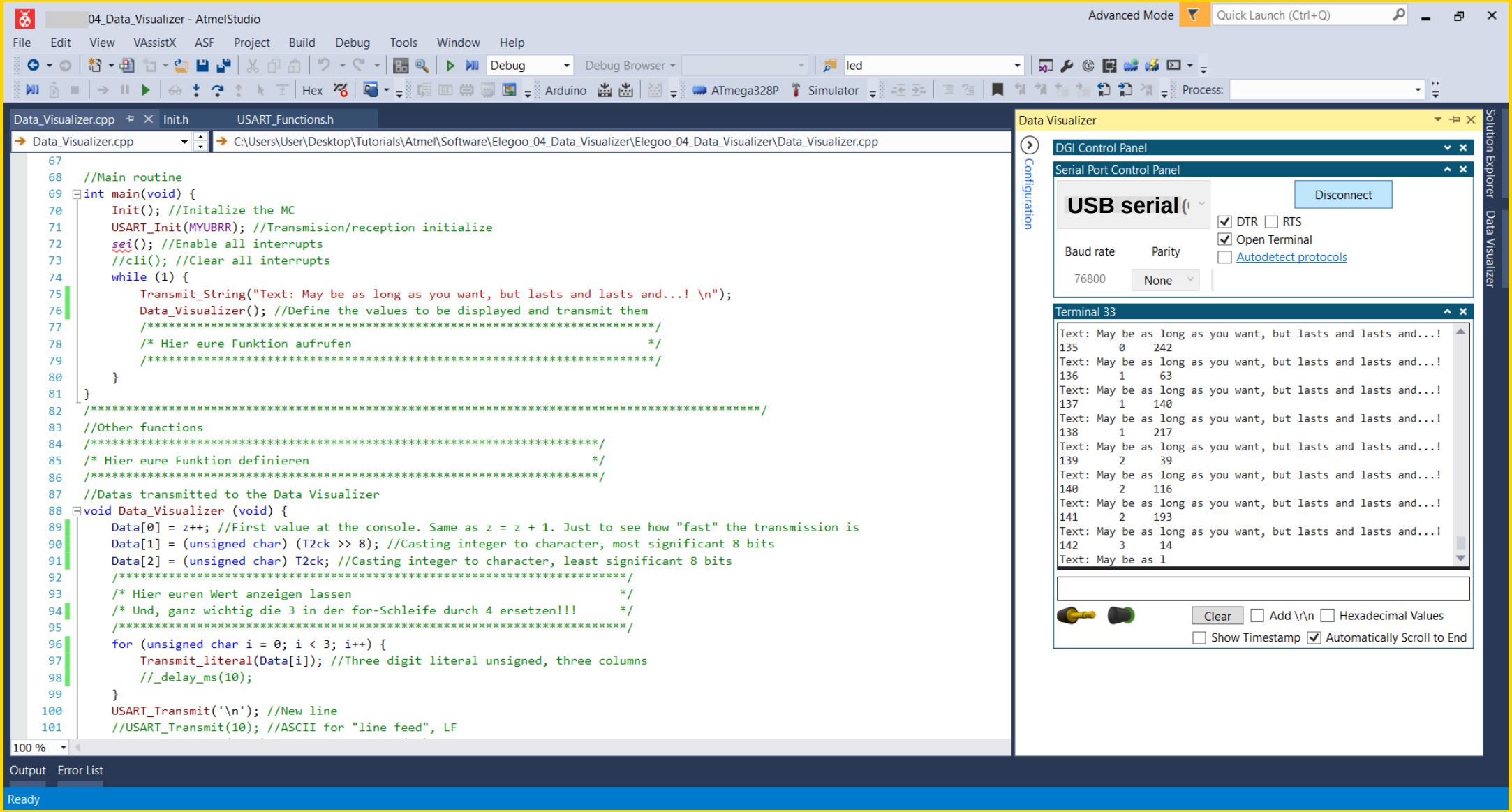
Solution Explorer | Data Visualizer

Die Großbuchstaben fangen bei 65 an. A → 65ASCII.

Die Kleinbuchstaben fangen bei 97 an. a → 97ASCII.

Die Ziffern fangen bei 48 an, 0 → 48ASCII.

Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII
0	00	000	NUL	32	20	040	SP	64	40	100	@	96	60	140	`
1	01	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	02	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	03	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	04	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	05	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	06	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	07	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	08	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	09	011	HT	41	29	051	)	73	49	111	I	105	69	151	i
10	0A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	0E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL


04_Data_Visualizer - AtmelStudio
Advanced Mode
Quick Launch (Ctrl+Q)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
led
Hex
Arduino
ATmega328P
Simulator
Process:
Data_Visualizer.cpp
Init.h
USART_Functions.h
C:\Users\User\Desktop\Tutorials\Atmel\Software\Elegoo_04_Data_Visualizer\Elegoo_04_Data_Visualizer\Data_Visualizer.cpp
67
68 //Main routine
69 int main(void) {
70 Init(); //Initalize the MC
71 USART_Init(MYUBRR); //Transmision/reception initialize
72 sei(); //Enable all interrupts
73 //cli(); //Clear all interrupts
74 while (1) {
75 Transmit_String("Text: May be as long as you want, but lasts and lasts and...! \n");
76 Data_Visualizer(); //Define the values to be displayed and transmit them
77 /***/
78 /* Hier eure Funktion aufrufen */
79 /***/
80 }
81 }
82 /***/
83 //Other functions
84 /***/
85 /* Hier eure Funktion definieren */
86 /***/
87 //Datas transmitted to the Data Visualizer
88 void Data_Visualizer (void) {
89 Data[0] = z++; //First value at the console. Same as z = z + 1. Just to see how "fast" the transmission is
90 Data[1] = (unsigned char) (T2ck >> 8); //Casting integer to character, most significant 8 bits
91 Data[2] = (unsigned char) T2ck; //Casting integer to character, least significant 8 bits
92 /***/
93 /* Hier euren Wert anzeigen lassen */
94 /* Und, ganz wichtig die 3 in der for-Schleife durch 4 ersetzen!!! */
95 /***/
96 for (unsigned char i = 0; i < 3; i++) {
97 Transmit_literal(Data[i]); //Three digit literal unsigned, three columns
98 //_delay_ms(10);
99 }
100 USART_Transmit('\n'); //New line
101 //USART_Transmit(10); //ASCII for "line feed", LF
100 %
Data Visualizer
Configuration
DGI Control Panel
Serial Port Control Panel
USB serial
Disconnect
DTR RTS
Open Terminal
Autodetect protocols
Baud rate
Parity
76800
None
Terminal 33
Text: May be as long as you want, but lasts and lasts and...!
135 0 242
Text: May be as long as you want, but lasts and lasts and...!
136 1 63
Text: May be as long as you want, but lasts and lasts and...!
137 1 140
Text: May be as long as you want, but lasts and lasts and...!
138 1 217
Text: May be as long as you want, but lasts and lasts and...!
139 2 39
Text: May be as long as you want, but lasts and lasts and...!
140 2 116
Text: May be as long as you want, but lasts and lasts and...!
141 2 193
Text: May be as long as you want, but lasts and lasts and...!
142 3 14
Text: May be as 1
Clear
Add \r\n
Hexadecimal Values
Show Timestamp
Automatically Scroll to End
Solution Explorer
Data Visualizer
Output
Error List
Ready

Musterlösung:

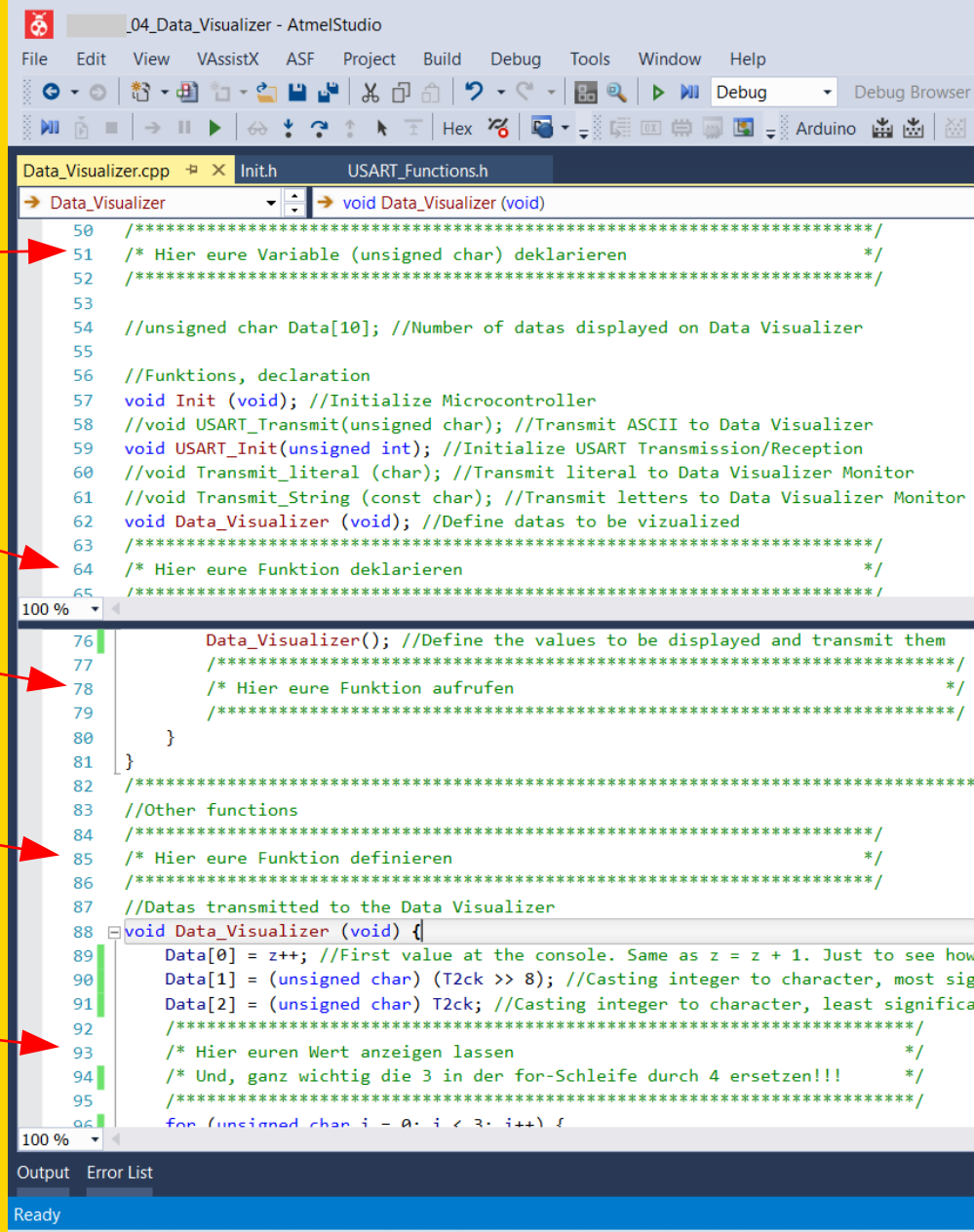
```
//Deklaration der Variablen in der der doppelte Wert gespeichert wird.
unsigned char Doppelt;

//Deklaration der Funktion
void Verdopplung (unsigned char);

//Aufruf der Funktion in der main
…
        Verdoppelung(z);
…

//Definition der Funktion
void Verdoppelung (unsigned char Faktor) {
        Doppelt = Faktor * 2;
}

//Anzeigen des Wertes
…
        Data[3] = Doppelt;
…
```



Montage und Programmierung eines Roboters für den Hessen SolarCup Disziplin: SolaRobot Teil 2.4: Line Sensor

Von Charlotte und Andreas