Computergestützte Experimente und Signalauswertung

PHY.W04UB/UNT.038UB

Coding & Debugging

by Jan Enenkel

Übersicht

Moveo Roboter Demo

Coding

Variablen & Befehle

Zahlenmanipulation

Debugging

Moveo Roboter Demo

Moveo Roboter - Übersicht

- Open Source Roboterarm – Thingiverse.com
- 3D Gedruckte Mechaniken
- 6x Stepper Motoren
- 1x Gripper – Servo Motor
- X-Box Controller – Bluetooth Low Energy
- Keine Positionskontrolle

Spannungsversorgung

Motoren: 12V / 5A

Stepper: 5V / 2A

Quelle: Xbox.com

Quelle: BNC3D Technologies

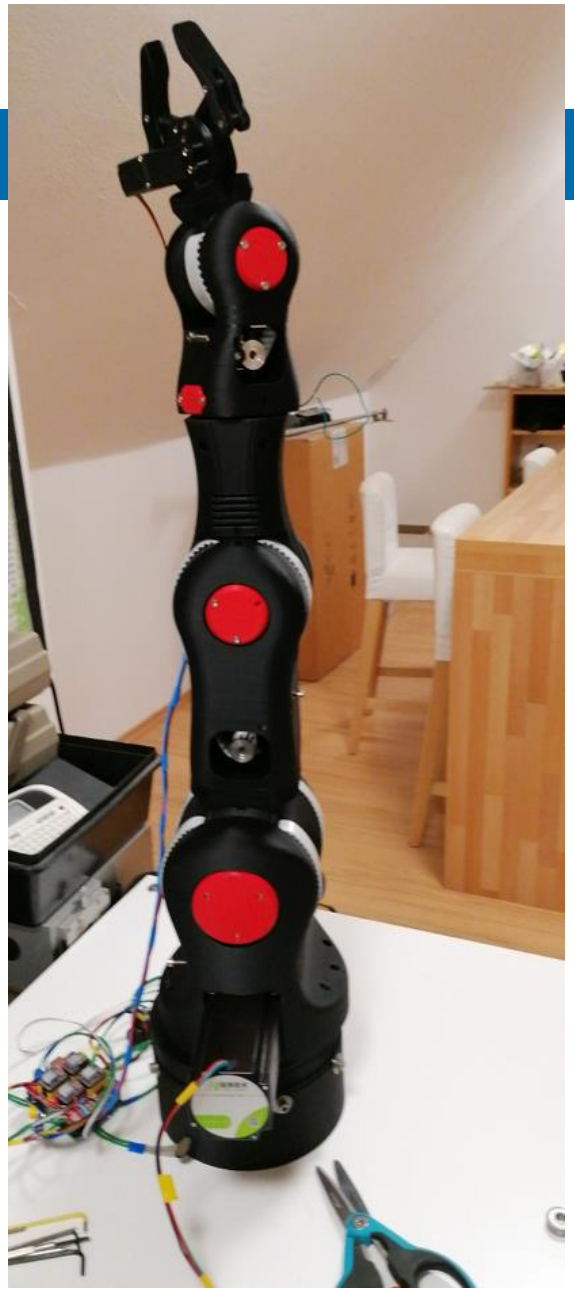
Projektaufwand

Aufwand

- Komponenten Erfassen & Bestellen - 6h
- 3D Drucken – 50h – 2x Prusia
- Zusammenbauen – 16h
  - Mechanik – 6h
  - Löten – 4h
  - HW-Debugging – 6h
- Coding ~ 20h
  - SW-Debugging ~10h
- Kosten ~ 500€
  - Motoren/Riemen/Endstufen/Xbox Controller
- Start: 20.12.2022 → Fertig ca. März

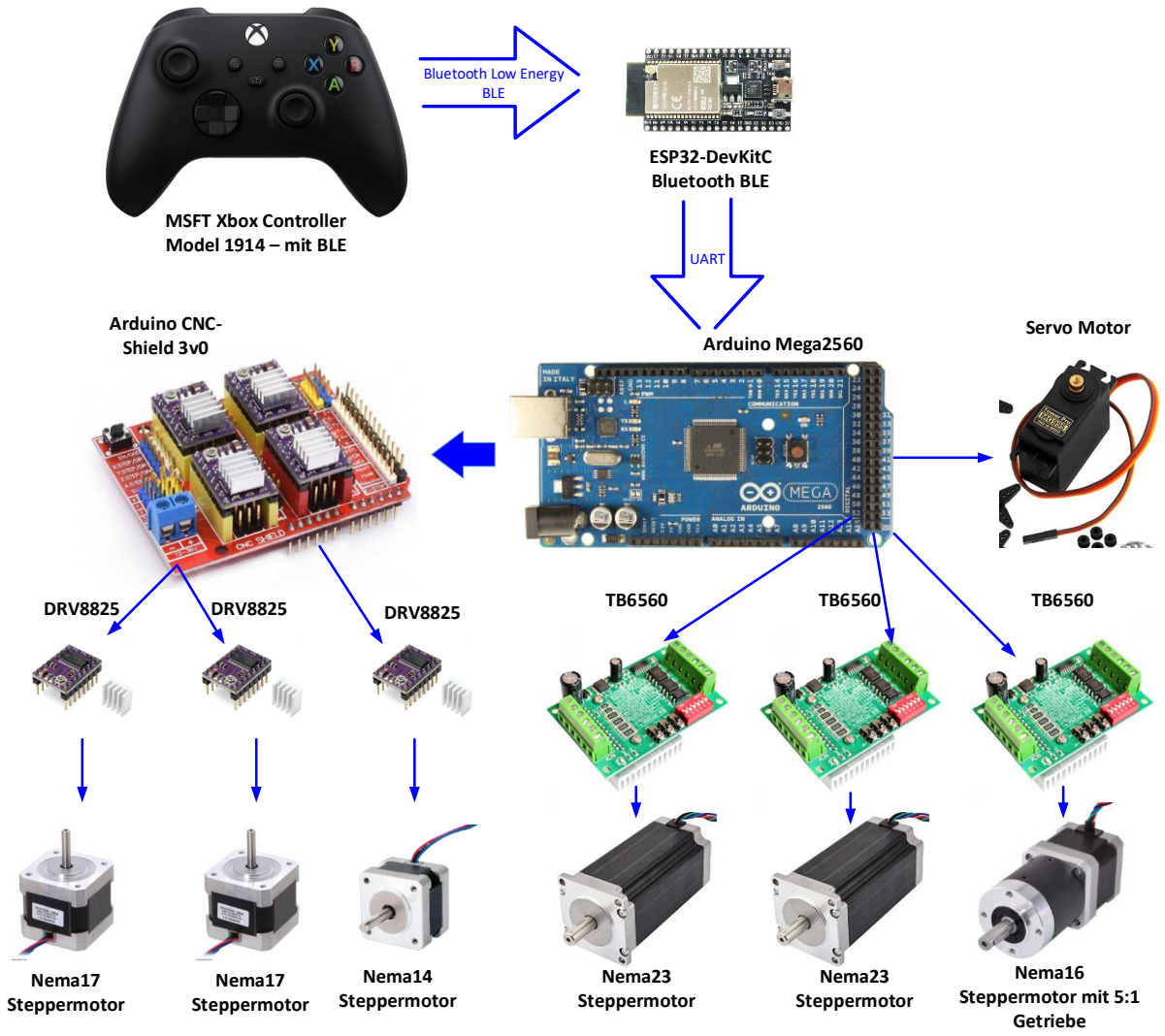
Bottlenecks

- Nicht verfügbare Teile/Amazon Lieferung
- Bluetooth BLE → ESP32
  - Nano IOT als auch Nano-BLE bräuchten andere Firmware
- Fehlerhafte 3D Designs → nachbessern
- Netzgeräte – 5A nötig!



Moveo Roboter – Hardware

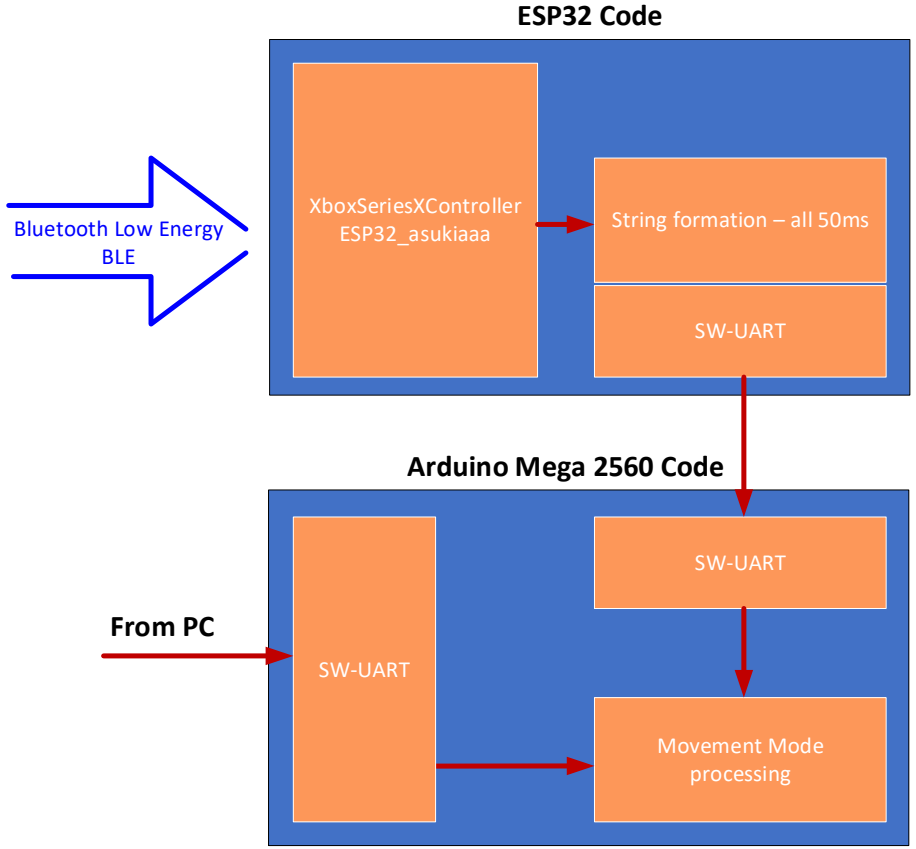
- Xbox Controller als Input
- ESP32 als Bluetooth BLE Empfänger
  - UART Strings an den Arduino Mega
- Arduino Mega 2560
  - Genug GPIOs um die Motortreiber anzusteuern
  - DRV8825 für die kleinen Motoren
  - TB6560 für mehr Motor-Strom
  - Servomotor für den Gripper



Sourcecode @ Moodle

Moveo Roboter – Software

- ESP32 als Bluetooth BLE Empfänger
  - Wandelt Bluetooth Signal auf SW-UART Signal um
- Arduino Mega 2560
  - Parsing der PC Strings
    - HW-UART
  - Parsing der Bluetooth Strings
    - SW-UART
  - Statemachine um die Motoren zu „Pulsen“
  - Einbau von Maximalgeschwindigkeiten
  - Mehrere Motoren nicht möglich(derzeit)



Moveo Roboter – als Projekt?

Wäre dies als Projekt ausreichend?

→ Nein, kein physikalisches System

Für ein Projekt - Möglichkeiten

- Beschreibung des physikalischen Systems
  - Zylinderkoordinaten in Kartesische umwandeln
  - Konstante Geschwindigkeit des Greifers?
  - Lineare Beschleunigung?
- Mögliche Analysen
  - Genauigkeit
  - Toleranzen (Vorzugsrichtung)
  - Versatz unter Belastung
- Nachregelung(en)
  - Messung der Winkel-Position mit Inkrementalgebern
  - Nachregelung von Positionen
  - Startbedingung

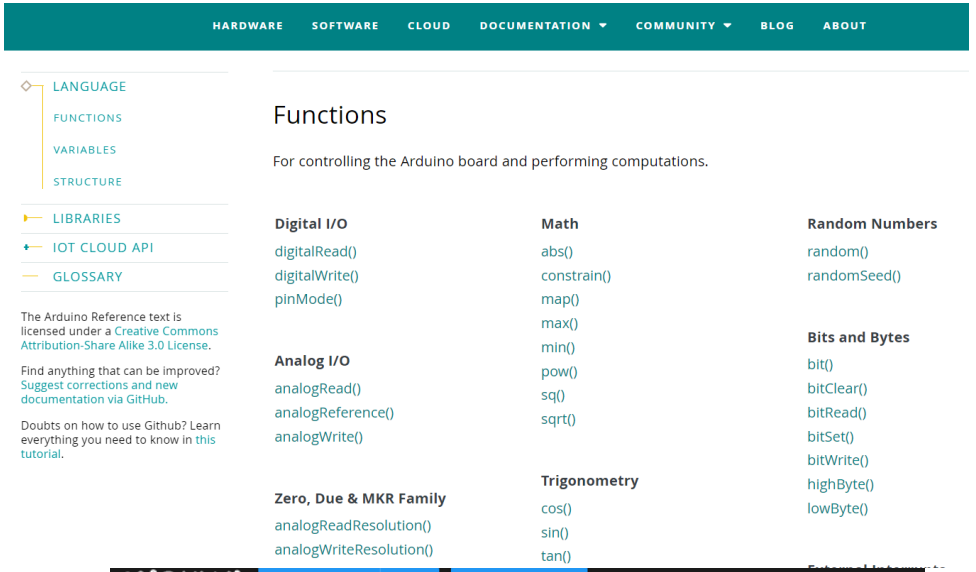
Quelle: BNC3D Technologies

Coding

Wie kann ich Coding üben?

www.Arduino.cc

- ‚Documentation → Reference‘
- Dokumentation aller Befehle
- Dokumentation vieler Bibliotheken



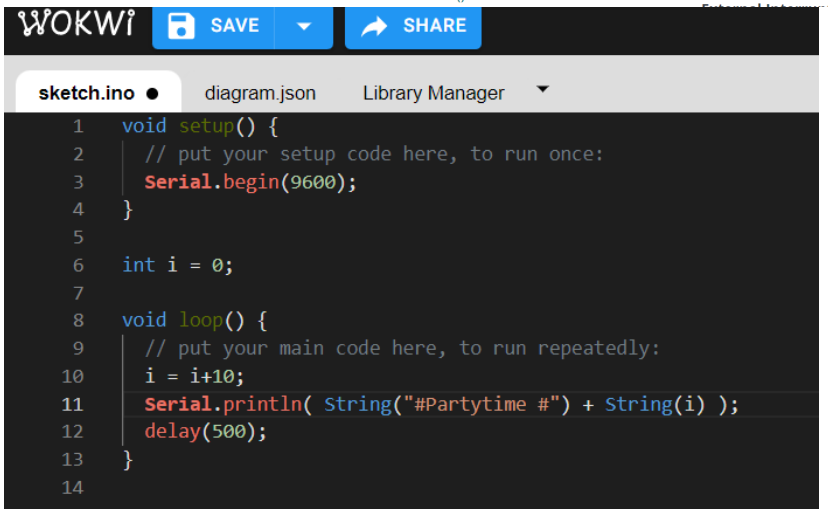
ChatGPT

- Sehr gut im erklären von Grundlagen und simplen Codes

Arduino Simulatoren

https://wokwi.com/projects/new/arduino-nano

- Vorteile
  - Kostenlos / Codesegmente testbar
  - Hardware teilweise Simulierbar



Variablen & Befehle

Variablen

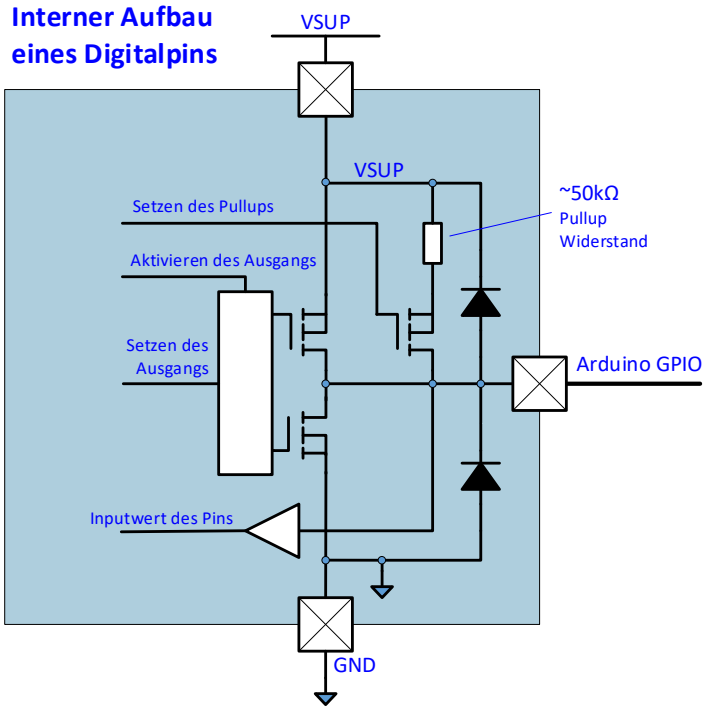
Befehl	Beschreibung
char wert;	8-Bit Variable mit Vorzeichen – Bereich -128 bis +127 Kann für ASCII Zeichen oder Zahlen verwendet werden
unsigned char wert;	8-Bit Variable ohne Vorzeichen – Bereich 0 bis 255
int wert;	16-Bit Variable mit Vorzeichen, von -32768 bis 32767
unsigned int	16-Bit Variable ohne Vorzeichen, von 0 bis 65535 (kann auch 32-Bit sein)
long	32-Bit Variable mit Vorzeichen (kann auch 64-Bit sein)
float	32-Bit Fließkommazahl
bool	1-Bit Wert, kann ‚true' oder ‚false' sein.

>> Variablen sind Prozessor/Compiler abhängig<<

Ein „int" bei Arduino kann bei einigen C-Compilern 16-Bit haben, bei anderen 32-Bit.

Arduino Befehle - GPIOs

Befehl	Beschreibung
pinMode(pin,value)	Konfiguriert den pin, als INPUT, OUTPUT oder INPUT_PULLUP
digitalWrite(pin,value)	Stellt einen digitalpin(GPIO) auf den Wert HIGH oder LOW
digitalRead(pin)	Ließt einen digitalPin(GPIO) aus, gibt 1 oder 0 zurück.
analogRead(pin)	Ließt den Analogwert des Pins aus, Befehl benötigt ca. 120µs, Ausgabewert hat 10-bit bei einem Arduino Nano
analogWrite(pin,value)	Gibt ein PWM Signal an pin aus (meistens 490Hz) mit Duty Cycle welcher zu Value korrespondiert, Value hat 8-bit



Arduino Befehle

Befehl	Beschreibung
delay(value)	Wartet in value millisekunden
Serial.begin(value)	Initialisiert die UART, value muss Baudrate betragen
Serial.println(string)	Sendet per UART einen string welcher mit „\n“ terminiert wird
millis()	Gibt einen long integer zurück wie viele Millisekunden vergangen sind seit der Arduino hochgefahren ist – „Zeitstempel“ – Überlauf nach 49 Tagen
micros()	Gibt einen long integer zurück wievielte Mikrosekunden vergangen sind seit der Arduino hochgefahren ist – „Zeitstempel“– Überlauf nach ~71 Minuten
map()	Rechnet variablen auf gewünschte Zahlenbereiche um. → y=kx+d

String Befehle – Ansi C Style

Befehl	Beschreibung
char string[50];	Definiert einen string (mit Namen string) mit 50 Zeichen.
strcpy(string,“Hallo“);	String-copy schreibt „Hallo“ in string rein, und schließt es mit „0“ ab.
strcat(string,“ Welt“);	String-Cat fügt „ Welt“ hinten an den string an.
sprintf(string,“val=%d“,value);	Printf welches in einen string schreibt, dabei wir der wert von value in den string geschrieben. Klappt nicht für floating points oder long integers (bei Arudino, in Regulärem C klappt es).
strlen(string);	Gibt die länge des strings zurück, bis im string eine ‚0‘ vorkommt

Arduino „String“-Macro – um long integers in einen String darstellen zu können:

```
sprintf(string, "Current position: %s",String(CurrentPosition, DEC).c_str());
```

Arduino workaround weil sprintf keine Floats ausgeben kann:

```
void debugString(float val)
{
  strcpy(string, "debug= ");
  dtostrf(val, 2, 2, &string[strlen(string)]);
  Serial.println(string);
}
```

String Befehle – Arduino Style

Befehl	Beschreibung
String myString = String(„hi“);	Definiert einen String mit dem Namen myString mit dem Inhalt „hi“

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   void loop() {
6     int messwert = analogRead(A0);    // Messwert von A0 pin einlesen
7     String myString = String("Messwert=") + String(messwert*5.0/1023.0) + String("[V]");
8     Serial.println(myString);        // Ausgabe per UART
9     delay(500);                      // 500ms pause
10  }
11
```

ASCII Tabelle

Scan-code	ASCII hex	dez	Zeichen
	00	0	NUL ^@
	01	1	SOH ^A
	02	2	STX ^B
	03	3	ETX ^C
	04	4	EOT ^D
	05	5	ENQ ^E
	06	6	ACK ^F
	07	7	BEL ^G
0E	08	8	BS ^H
0F	09	9	TAB ^I
	0A	10	LF ^J
	0B	11	VT ^K
	0C	12	FF ^L
1C	0D	13	CR ^M
	0E	14	SO ^N
	0F	15	SI ^O
	10	16	DLE ^P
	11	17	DC1 ^Q
	12	18	DC2 ^R
	13	19	DC3 ^S
	14	20	DC4 ^T
	15	21	NAK ^U
	16	22	SYN ^V
	17	23	ETB ^W
	18	24	CAN ^X
	19	25	EM ^Y
	1A	26	SUB ^Z
01	1B	27	Esc ^[
	1C	28	FS ^\
	1D	29	GS ^]
	1E	30	RS ^^
	1F	31	US ^_

Scan-code	ASCII hex	dez	Zch.
	20	32	SP
02	21	33	!
03	22	34	"
29	23	35	#
05	24	36	$
06	25	37	%
07	26	38	&
0D	27	39	'
09	28	40	(
0A	29	41	)
1B	2A	42	*
1B	2B	43	+
33	2C	44	,
35	2D	45	-
34	2E	46	.
08	2F	47	/
0B	30	48	0
02	31	49	1
03	32	50	2
04	33	51	3
05	34	52	4
06	35	53	5
07	36	54	6
08	37	55	7
09	38	56	8
0A	39	57	9
34	3A	58	:
33	3B	59	;
2B	3C	60	<
0B	3D	61	=
2B	3E	62	>
0C	3F	63	?

Scan-code	ASCII hex	dez	Zch.
	40	64	@
1E	41	65	A
30	42	66	B
2E	43	67	C
20	44	68	D
12	45	69	E
21	46	70	F
22	47	71	G
23	48	72	H
17	49	73	I
24	4A	74	J
25	4B	75	K
26	4C	76	L
32	4D	77	M
31	4E	78	N
18	4F	79	O
19	50	80	P
10	51	81	Q
13	52	82	R
1F	53	83	S
14	54	84	T
16	55	85	U
2F	56	86	V
11	57	87	W
2D	58	88	X
2C	59	89	Y
15	5A	90	Z
	5B	91	[
	5C	92	\
	5D	93	]
29	5E	94	^
35	5F	95	_

Scan-code	ASCII hex	dez	Zch.
0D	60	96	`
1E	61	97	a
30	62	98	b
2E	63	99	c
20	64	100	d
12	65	101	e
21	66	102	f
22	67	103	g
23	68	104	h
17	69	105	i
24	6A	106	j
25	6B	107	k
26	6C	108	l
32	6D	109	m
31	6E	110	n
18	6F	111	o
19	70	112	p
10	71	113	q
13	72	114	r
1F	73	115	s
14	74	116	t
16	75	117	u
2F	76	118	v
11	77	119	w
2D	78	120	x
2C	79	121	y
15	7A	122	z
	7B	123	{
	7C	124	|
	7D	125	}
	7E	126	~
53	7F	127	DEL

Moveo Coding
Beispiel @ Moodle

Was ist String-Parsing?

Daten werden in einem String Verpackt

- Text + Messwert + Einheit
- Sicherheiten einbauen.: „<„ + „>“
- Zb Übertragung über Interface
  - UART
  - Bluetooth
  - I2C
  - Usw..

Daten werden ‚Entpackt‘

- Text + Messwert + Einheit
- Umwandlung in eine Zahl

Nützlich bei

- Arduino zu PC
  - Verpacken mittels Arduino IDE
  - Entpacken mittels Matlab/Python
- Arduino zu Arduino

```
String s = "<" + String(xboxController.xboxNotif.btnA) + String(",")
               + String(xboxController.xboxNotif.btnB) + String(",")
               + String(xboxController.xboxNotif.btnY) + String(",")
               + String(xboxController.xboxNotif.btnX) + String(",")
               + String(trigLT) + String(",")
               + String(trigRT) + String(",")
               + String(Joystick_LeftHorizontal) + String(",")
               + String(Joystick_LeftVertical) + String(",")
               + String(Joystick_RightHorizontal) + String(",")
               + String(Joystick_RightVertical) + String(">");

Serial.println(s);
Serial2.println(s);
```

Bedingungen/Schleifen/Unterprogramme

Befehl	Beschreibung
for(Anfang; Ende; Inkrement)	for schleife
while(Bedingung)	Bleibt solange in der schleife wie Bedingung=1 ist.
if(Bedingung)	If Abfrage – nachfolgender code wird ausgeführt wenn Bedingung wahr ist.
else	Wird ausgeführt wenn Bedingung „!=„ 1 ist
break;	Beendet die jeweilige schleife
return;	Springt aus dem jeweiligen Unterprogramm, kann auch einen wert zurückgeben
void	Definition eines Unterprogramms

If-Abfrage

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   void loop() {
6     int messwert = analogRead(A0);            // Einlesen des Messwerts
7     float A0Spannung = messwert*5.0/1023.0;   // Umrechnen auf Spannung
8     if(messwert==0)
9       Serial.println(String("Messwert ist 0"));       // Ausgabe per UART
10    if(A0Spannung > 3.3)                      // Wenn Spannung größer 3.3V ist!
11      Serial.println(String("Spannung höher als 3.3V!!!"));       // Ausgabe per UART
12    delay(500);                       // 500ms pause
13  }
14
```

For-Schleife

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   void loop() {
6     long mittelwert_akku = 0;                 // Akkumulator auf 0 setzen
7     for( int i = 0 ; i <  10 ; i++ )          // 10x durchloopen
8       mittelwert_akku += analogRead( A0 );    // zusammenzählen
9     float mittelwert=(mittelwert_akku/10.0);  // Mittelwert bilden
10    Serial.println( String( mittelwert ) );               // Ausgabe
11    delay(500);
12  }
13
```

Code Ausführung / Unterprogramme

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   float gemitteltеMessung()
6   {
7     long mittelwert_akku = 0;                 // Akkumulator auf 0 setzen
8     for( int i = 0 ; i <  10 ; i++ )          // 10x durchloopen
9       mittelwert_akku += analogRead( A0 );    // zusammenzählen
10    return (mittelwert_akku/10.0);  // Mittelwert bilden
11  }
12
13  void loop() {
14    Serial.println( String( gemitteltеMessung() ) );              // Ausgabe
15    delay(500);
16  }
17
```

Zahlenmanipulation

Zahlensysteme Reminder

- Dem Hex wert steht immer ein ‚0x' voran
  - Zb 0xFF
- Hex und Binär oft Notwendig zum Konfigurieren von Sensoren
- Zum Zusammenfassen von Messdaten
- Notwendig bei diversen Schnittstellen
  - UART
  - I^2C

Beispiele:

0x10 + 0x0F = 0x1F

0x0C + 0x05 = 0x11

0x21 + 0x93 = 0xB4

Dezimal	Binär	Hexadezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Mathematische Operationen

Befehl	Beschreibung
a = b+c;	a ist die summe von b und c
a++;	Wert in „a“ wird um 1 erhöht.
a = b<<2;	In ‚a‘ wird der Wert von b um 2 nach links geshiftet gespeichert
a = b*4;	In ‚a‘ wird der Wert von b mit 4 multipliziert gespeichert
a = 0x0F & a;	Bei a werden die ersten 4 Bits ausmaskiert
a |= 0xF0;	Bei a werden die oberen 4 Bits gesetzt.
a = 1<<b;	In ‚a‘ wird 1 um b nach links geschiftet.
a = (0x0F & b)<<4;	Bei ‚b‘ werden die ersten 4 Bits maskiert und dann um 4 Bits nach links geschoben

Zahlenmanipulation
Codebeispiele

Beispiel Zahlenmanipulation

```
int Zahl = 0;
// Logische Operationen - einzelne Bits
Zahl = Zahl | 0x55;   // logisches ODER, Zahl = 0x55;
Zahl &= 0x0F;         // logisches UND, Zahl = 0x05;
Zahl += 1;            // Zahl um 1 erhöhen, Zahl = 0x06;
Zahl <<= 2;           // Zahl um 2 Bits rechts shiften,Zahl = 0x18
Zahl <<= 2;           // Zahl um 2 Bits rechts shiften, Zahl = 0x60


int messwert = 0x127;       // 295 in dezimal

// type-casting - umwandlung auf unsigned char
unsigned char resultat = (unsigned char)(messwert & 0xFF);
//(resultat=0x27)
```

Beispiel 2x8-Bit Werte zu 1x16-Bit Wert

```
// unsigned char mit 2x Bytes als Einlese Puffer
unsigned char buffer[2];
// unsigned int für den Messwert ( 16-Bit )
unsigned int messwert = 0;


// einlesen der Messwerte von I2C Bus
unsigned char read_bytes = I2C_BlockRead(0x39,0x94,2,buffer);


// Auswertung nur durchführen wenn 2x Bytes da sind
if(read_bytes == 0x02)
{
     // Berechnung des Messwerts, Klammern!
     messwert = (buffer[1]<<8)+buffer[0];


     if(messwert>0xF000)          // Auswertung
     {
          Serial.println(„Warnung - Zuviel Laserleistung!“);
     }
}
```

Mögliche Prüfungsfragen

Was für ein Wert wird ausgegeben?

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   void loop() {
6     int result = 0;
7     for( int i = 0; i < 10; i++ )
8       result += i;
9     Serial.println(String(result));
10    delay(500);
11  }
12
```

Was für ein Wert wird ausgegeben?

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   void loop() {
6     int zahl = 0;
7     zahl  = zahl | 0x55;
8     zahl &= 0x0F;
9     zahl += 1;
10    zahl <<=2;
11    Serial.println(String(zahl));
12    delay(5000);
13  }
14
```

Was bezweckt der folgende code?

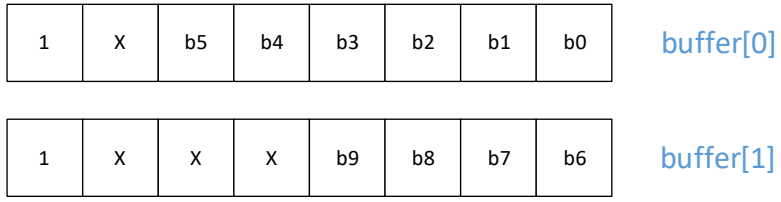
```
1   long zeitStempel = 0;   // Globale Variable!
2
3   void setup() {
4     Serial.begin(57600);    // Initialisieren der UART
5     long zeitStempel = millis();
6   }
7
8   void loop() {
9     while(millis()-zeitStempel> 1000)
10    {
11      zeitStempel = millis();
12      Serial.println(String("Yeah!"));
13    }
14  }
15
```

Was für ein Wert wird ausgegeben?

```
1   void setup() {
2     Serial.begin(57600);    // Initialisieren der UART
3   }
4
5   void loop() {
6     unsigned char zaehler = 0;
7
8     for(int i = 0; i < 30; i++ )
9       zaehler+=10;
10    Serial.println(String(zaehler));
11    delay(500);
12  }
13
```

Mögliche Prüfungsfragen

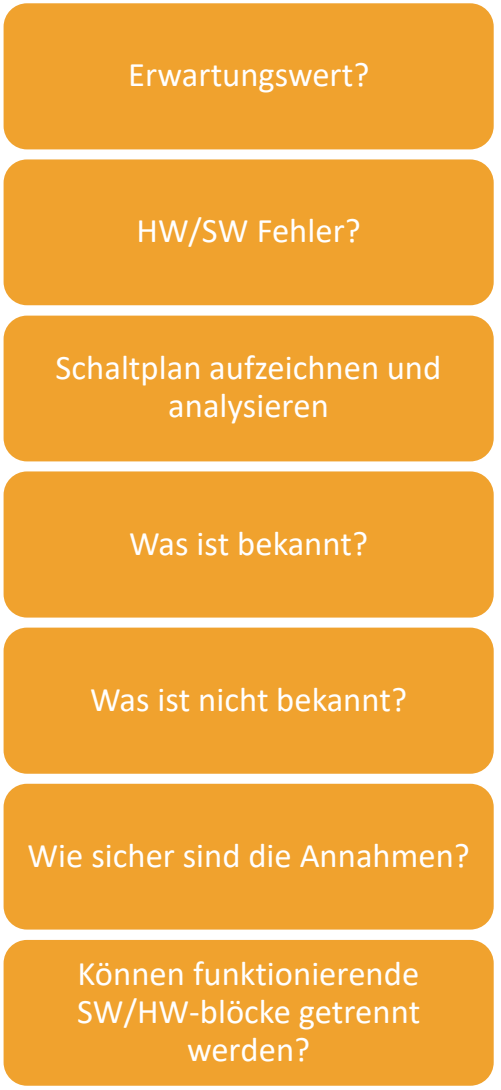
- Sie haben einen 10-Bit digitalen Luftfeuchtigkeitssensor gekauft welchen Sie nun per I²C auslesen. Die Arduino Bibliothek funktioniert nicht, ChatGPT kennt den Sensor nicht, und Sie entschließen sich den Sensor selber auszulesen. Dabei lesen Sie 2-Bytes per I²C aus. Das erste Byte (buffer[0]) beinhaltet laut Datenblatt 6x niederwertigere Bits des Messwerts. Das 2.te Byte (buffer[1]) beinhaltet die 4x höherwertigeren Bits. Beachten sie das die Bits welche nichts mit dem Messresultat zu tun haben sehr wohl 1 sein können. Schreiben Sie einen Pseudo/C/Arduino Code welcher einen Messwert als integer darstellt durch geeignete Berechnung. Beachten sie die Klammersetzung.



Debugging

System - Debugging

- Ein „BUG“ ist kein Fehler sondern ein Vorgang der nicht mit der Erwartung übereinstimmt!
- Korrekter Syntax?
- Eigene Mess/Signalkette aufzeichnen
- Fehlerquellen reduzieren
- Datenblätter **Lesen**
- Treiber Sourcecodes & Doku **Lesen**
- Kann ich HW & SW sauber voneinander trennen?
- Zuviel Speicher verbraucht?

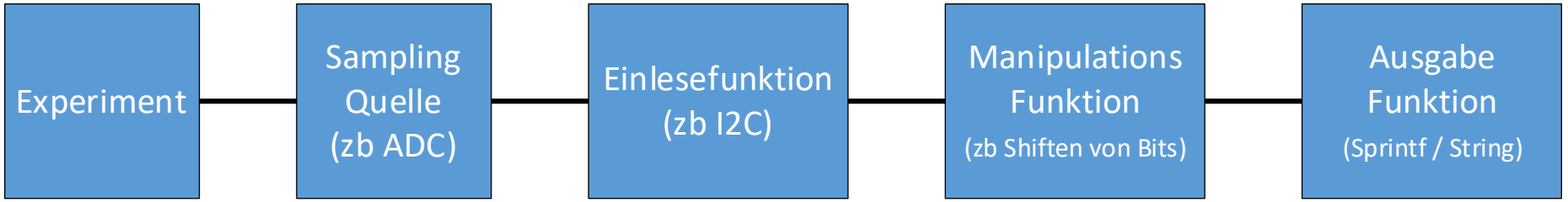


Code – Debugging

- Annahme die Hardware funktioniert!
- Abstraktion hilft nicht nur bei Leserlichkeit sondern Reduziert das Fehlerpotential!

Jeden Block trennen und Separat ausführen

- Funktioniert meine Ausgabe wie ich es erwarte? (zB.: für alle Zahlenwerte)
- Funktioniert mein I²C korrekt? (oder lese ich nur 0 Zurück)
- Funktioniert mein ADC korrekt?
- Habe ich die Manipulationsfunktionen verstanden?
- Habe ich ein Timing Problem?
  - Zb. zu schnelles nachfragen Resetiert meinen ADC
  - Zb. Delay einbauen



Vielen Dank für ihre Aufmerksamkeit!