

Low Level AVR

Arduino? Nein danke!

Felix Queißner

shackspace

2020

Wer bin ich?

- ▶ Felix „xq“ Queißner
- ▶ Baujahr 1993
- ▶ Mit 12 angefangen, zu programmieren
- ▶ Mit 13 angefangen, Elektronik zu basteln
- ▶ Mache gerne Dinge mit Code und alten Computern

Worum geht's?

- ▶ Grundlagenwissen AVR-Programmierung vermitteln
- ▶ „Wie komme ich klar, wenn es mit der Arduino-IDE nicht mehr weiter geht?“
- ▶ „Awareness schaffen“ für Code Bloat
- ▶ **Nicht:** AVR/Arduino löten
- ▶ **Nicht:** C programmieren lernen

Outline

1. Teaser
2. The Arduino way
3. Zum Ziel in vier Schritten
 - 3.1 Dokumentation lesen
 - 3.2 Code schreiben
 - 3.3 Compilen & Linken
 - 3.4 Flashen
4. Fazit
5. Wie geht's weiter?

Wie kommen wir von hier ...

```
1 int button = getPressedButton();
2 if (button == 1) {
3     digitalWrite(RELAIS_1_PIN, HIGH);
4     delay(350);
5     digitalWrite(RELAIS_2_PIN, LOW);
6     digitalWrite(RELAIS_3_PIN, LOW);
7     digitalWrite(RELAIS_4_PIN, LOW);
8 }
9 if (button == 2) {
10    digitalWrite(RELAIS_1_PIN, LOW);
11    digitalWrite(RELAIS_2_PIN, HIGH);
12    delay(350);
13    digitalWrite(RELAIS_3_PIN, LOW);
14    digitalWrite(RELAIS_4_PIN, LOW);
15 }
16 if (button == 3) {
17    digitalWrite(RELAIS_1_PIN, LOW);
18    digitalWrite(RELAIS_2_PIN, LOW);
19    digitalWrite(RELAIS_3_PIN, LOW);
```

... nach da?

```
1 int button = getPressedButton();  
2 PORTD = 0x00;  
3 if (button != 0) {  
4     delay(350);  
5     PORTD = (1 << (button + 2));  
6 }
```

The Arduino way

- ▶ Schnell von „Idee“ zu „Es blinkt schon mal was“ kommen
- ▶ Single-Click-Solution
- ▶ Mit was man genau arbeitet ist eigentlich egal, es erfüllt alles den gleichen Zweck

Blink

- ▶ Lässt eine LED mit $\frac{1}{2}$ Hertz blinken
- ▶ Benutzt `setup()` und `loop()`

```
1 void setup() {  
2     pinMode(LED_BUILTIN, OUTPUT);  
3 }  
4  
5 void loop() {  
6     digitalWrite(LED_BUILTIN, HIGH);  
7     delay(1000);  
8     digitalWrite(LED_BUILTIN, LOW);  
9     delay(1000);  
10 }
```


Serial

- ▶ Gibt "Hello, World!" auf der Konsole aus
- ▶ Spiegelt anschließend die Texteingabe zurück
- ▶ Benutzt `setup()` und `loop()`
- ▶ Benutzt `Serial.*`

```
1 void setup() {  
2     Serial.begin(19200);  
3     Serial.print("Hello, World!\r\n");  
4 }  
5  
6 void loop() {  
7     Serial.write(Serial.read());  
8 }
```

Pro/Contra Arduino

Pro:

- ▶ Man kommt schnell zum Ziel
- ▶ Zugrunde liegende Hardware ist schnell ausgetauscht
- ▶ Es gibt viele Beispiele für die Arduino-Welt
- ▶ „Batterien inklusive“

Contra:

- ▶ Es passiert sehr viel *Magie*
- ▶ Arduino-Programme sind „fett und träge“
- ▶ Wir haben keine wirkliche Kontrolle über das Programm

In vier Schritten zum Ziel

1. Dokumentation lesen
2. Code schreiben
3. Compilen & Linken
4. Flashen

Warum tun wir uns das an?

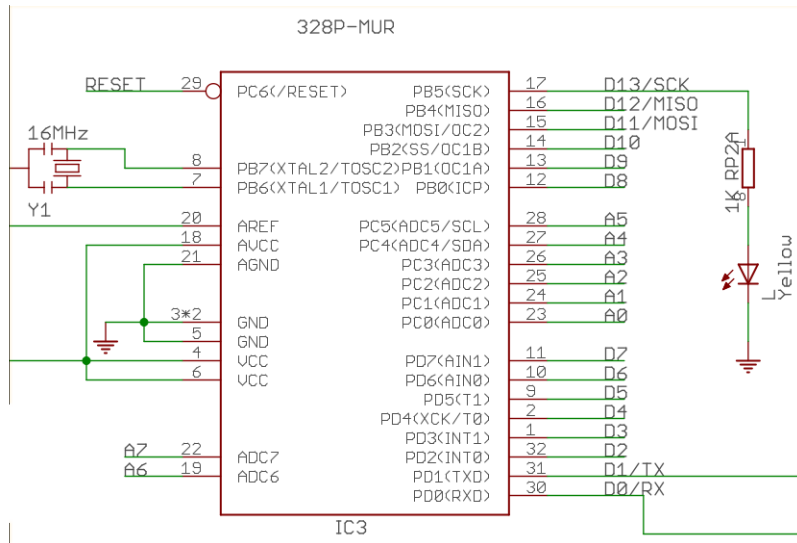
- ▶ Programme werden kleiner
- ▶ Programme werden schneller
- ▶ Wir können die Hardware voll ausnutzen

Schritt 1: Dokumentation lesen

Welche Dokumente brauchen wir?

- ▶ Schaltplan
- ▶ Datenblatt

Schaltplan



Schaltplan

- ▶ *Ground Truth* für Pinbelegungen
- ▶ Zeigt uns, wie die Hardware funktioniert
- ▶ Offenbart manchmal undokumentierte Möglichkeiten

11.3.3 Alternate Functions of Port D

The Port D pins with alternate functions are shown in [Table 11-9](#).

Table 11-9. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input) PCINT23 (Pin Change Interrupt 23)
PD6	AIN0 (Analog Comparator Positive Input) OC0A (Timer/Counter0 Output Compare Match A Output) PCINT22 (Pin Change Interrupt 22)
PD5	T1 (Timer/Counter 1 External Counter Input) OC0B (Timer/Counter0 Output Compare Match B Output) PCINT21 (Pin Change Interrupt 21)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input) PCINT20 (Pin Change Interrupt 20)

Datenblatt

- ▶ *Ground Truth* für Hardware-Funktionalität
- ▶ Zeigt uns, wie der Microcontroller funktioniert
- ▶ Hilfreich: Doppelfunktionen für Pin-Belegungen

Schritt 2: Code schreiben

Allgemein:

- ▶ C ist die primäre Embedded-Programmiersprache
- ▶ C++, Basic, Pascal, Zig, Rust, ... sind auf AVR ebenfalls verfügbar
- ▶ Dynamische Speicherverwaltung wird selten benötigt

Hands on:

1. Code
2. Datenblatt verwenden
3. Komplexer Hardware-Init
4. Interrupt

Hands on: Code (Blink 1)

```
1 #include <stdbool.h>
2 #include <avr/io.h>
3 #include <util/delay.h>
4
5 int main()
6 {
7     DDRB = (1 << PIN5);
8     while (true)
9     {
10         PORTB ^= (1 << PIN5);
11         for (int i = 0; i < 5; i++)
12             _delay_ms(200);
13     }
14 }
```

Hands on: Datenblatt (Blink 2)

```
1  int main() {
2      TCCR1A = 0;
3      TCCR1B = (1 << CS12) | (1 << CS10) | (1 << WGM12);
4      TCCR1C = 0;
5
6      OCR1AH = (TIMER1_LIMIT >> 8) & 0xFF;
7      OCR1AL = (TIMER1_LIMIT & 0xFF);
8
9      TIMSK1 = (1 << OCIE1A);
10
11     sei();
12     while (true);
13 }
14 ISR(TIMER1_COMPA_vect) {
15     PORTB ^= (1 << PIN5);
16 }
```

Hands on: Hardware-Init (UART 1)

```
1 void uart_tx(char c) {
2     while((UCSROA & (1<<UDRE0)) == 0);
3     UDR0 = c;
4 }
5 char uart_rx() {
6     while((UCSROA & (1<<RXC0)) == 0);
7     return UDR0;
8 }
9
10 int main() {
11     UCSROA = 0;
12     UCSROB = (1<<RXEN0) | (1<<TXEN0);
13     UCSROC = (1<<UCSZ01) | (1<<UCSZ00);
14     UBRR0 = (UBRR_BAUD & 0xFF);
15
16     while(true) {
17         char c = uart_rx();
18         uart_tx(c);
19     }
20 }
```

Hands on: Interrupts (UART 2)

```
1 int main() {  
2     ...  
3  
4     UCSROB |= (1<<RXCIE0);  
5  
6     sei();  
7  
8     while(true);  
9 }  
10  
11 ISR(USART_RX_vect) {  
12     UDR0 = UDR0;  
13 }
```

Schritt 3: Compilen & Linken

- Was ist ein - Compiler? - Linker? - Bibliothek? - Makefile? -
Dateiformate - Tools

Was ist ein Compiler?

Was ist ein Linker?

Was ist eine Bibliothek?

Was ist ein Makefile?

Dateiformate

- Dateiformate - elf object - elf executable - ihex

Tools

- objdump - size - addr2line

objdump

size

addr2line

Schritt 4: Flashen

- Möglichkeiten - High Voltage Serial Programming - In-System Programming - Bootloader - Tools - avrdude - PonyProg - AVRStudio - ...

In-System Programming

Bootloader

avrdude

Was haben wir gewonnen?

- ▶ Wir haben die Kontrolle über den Code
- ▶ Wir können die Hardware voll ausnutzen
- ▶ Unsere Programme sind kleiner
- ▶ Unsere Programme benötigen weniger CPU-Zeit

Vergleich: Blink

Vergleich: Serial

Wie geht's weiter?

Programmierung:

- ▶ C++
- ▶ Bibliotheken

Links:

- ▶ mikrocontroller.net
- ▶ Roboternetz.de
- ▶ Quellcode aus den Slides
<https://github.com/MasterQ32/avr-tutorial>

Fragen?