Arduino und die Physical Computing Revolution

Playground AV 2011

Anton Bachmayr - Netzteil

Der Inhalt

Wir sind NETZTEIL

Was ist ein Microcontroller?

Arduino - Eine Übersicht

Die Arduino IDE

Die Arduino Programmiersprache

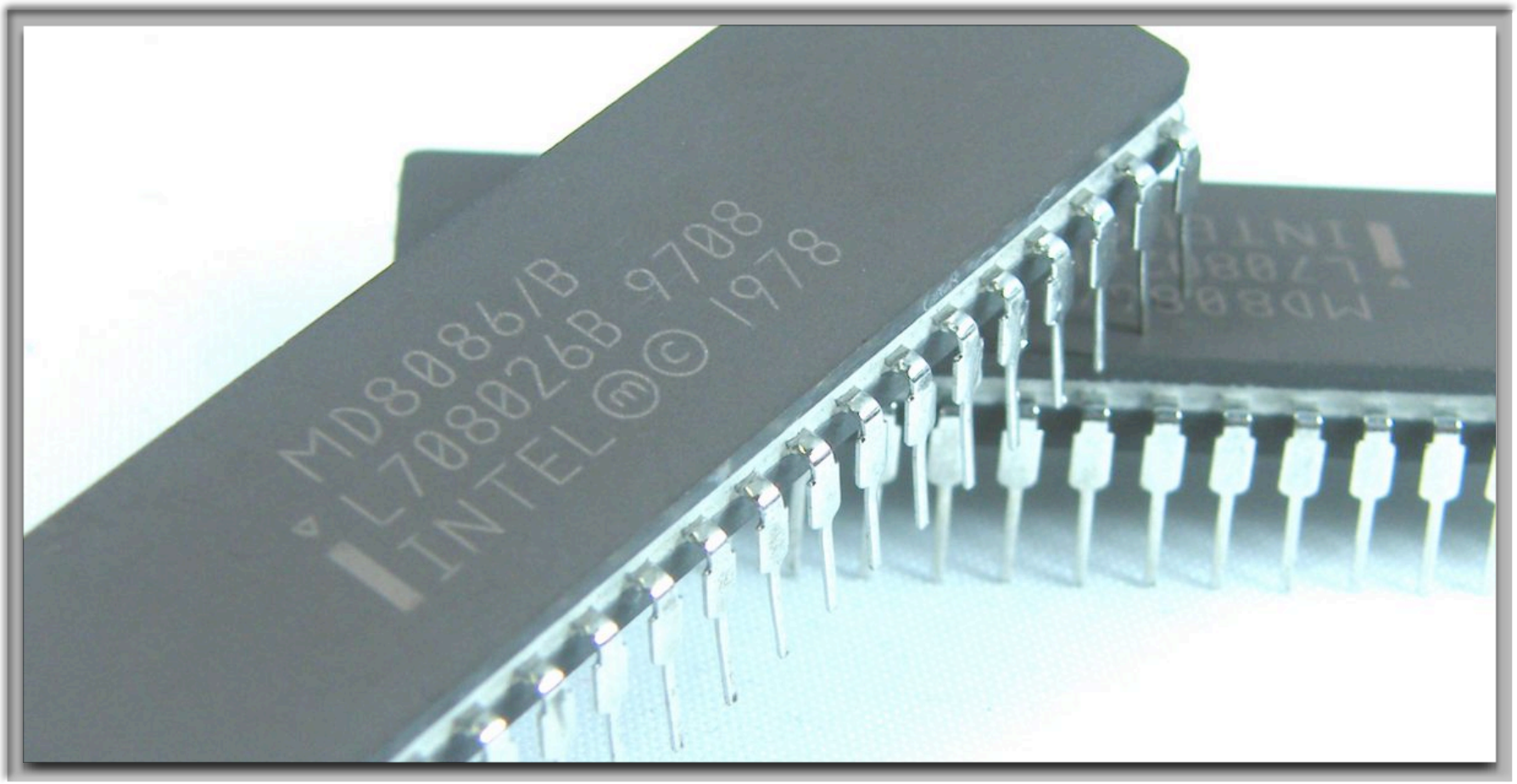
Das miniReACT Schild

Zusammensetzen der Elemente

Hands On

Hallo wir sind NETZTEIL

Martin Bachmayr & Anton Bachmayr



Was ist ein Microcontroller?

Er ist schließlich überall

Die Entstehung

1971 Intel bringt den ersten Mikroprozessor auf den Markt den 4004 - ein 4Bit CPU in einem 16 Pin Gehäuse

1974 Texas Instruments liefert den ersten Mikrocontroller aus - den TMS1000

1977 kontert Intel mit dem 8048 (auch MCS-48 genannt)

1993 bringt Microchip die ersten MCUs mit EEPROM auf den Markt - die PIC16x84 Serie

1993 Atmel startet mit den ersten MCUs mit Flash Speicher

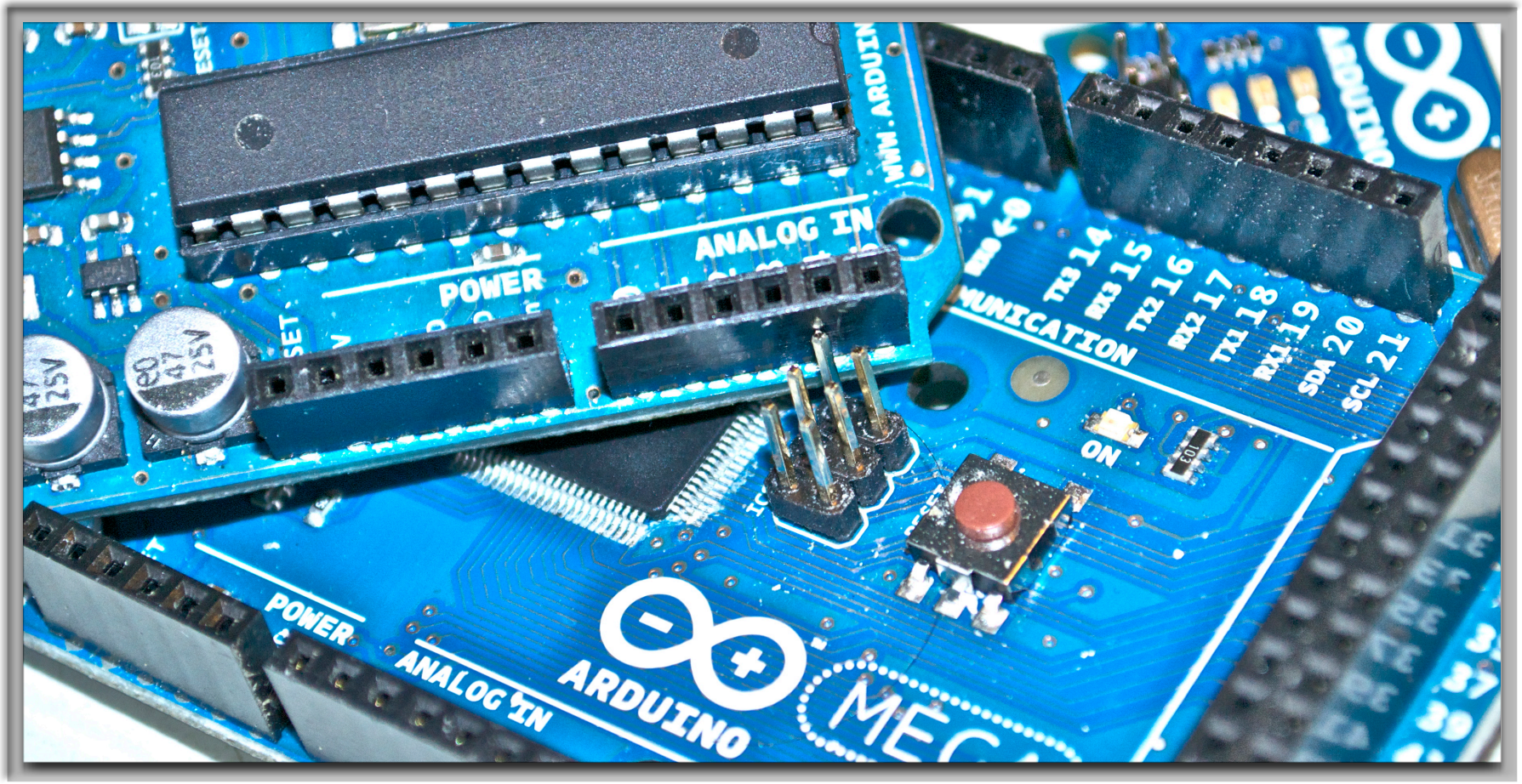
CPU, RAM und ROM vereint

MCUs beinhalten (fast) alles um von alleine zu funktionieren

mehrere unterschiedliche Architekturen

viele Hersteller

alles eine Frage des Geschmacks



Arduino - Eine Übersicht

Eine Erfolgsstory Made in Italy

120.000 Stück in den ersten 5 Jahren

Alles begann am Interaction Design Institute in Ivrea, Italien

Processing basiert auf dem „Design by Numbers“ Projekt

Wiring basierend auf Processing - vereinfachtes C++

Arduino verbindet Processing und Wiring

Arduino heisst übersetzt „starker freund“

Einfacher Zugang

Arduinos werden via Serieller Schnittstelle programmiert

Hohe Kompatibilität der Erweiterungen (Shield) durch definiertes Layout

Das Herzstück ist eigentlich die Arduino Entwicklungsumgebung

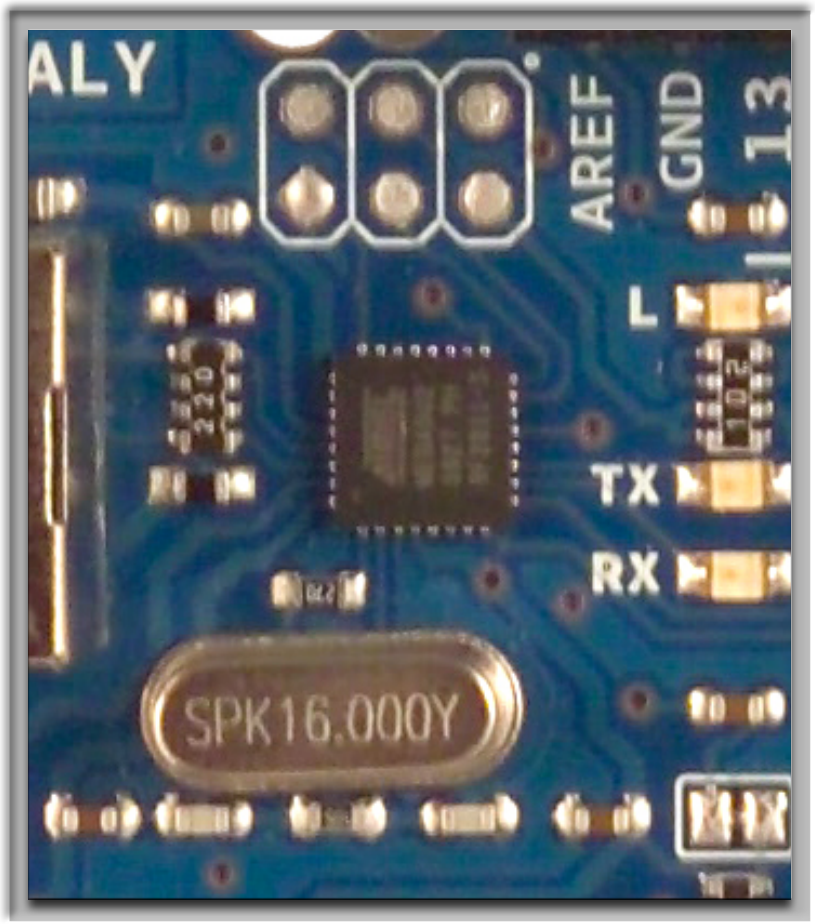
OpenSource Design - sharing is caring

Die Arduino Familie

Arduino	MCU	Flash	EEPROM	SRAM	I/Os	A out	A in	USB	Grösse
Diecimila	168	16K	0,5K	1K	14	6	6	FTDI	68,6x53,3mm
Duemilanove	168/328P	16/32K	0,5/1K	1/2K	14	6	6	FTDI	68,6x53,3mm
Uno	328P	32K	1K	2K	14	6	6	8U2	68,6x53,3mm
Mega	1280	128K	4K	8K	54	14	14	FTDI	101,6x53,3mm
Mega2560	2560	256K	4K	8K	54	14	14	8U2	101,6x53,3mm
Fio	328P	32K	1K	2K	14	6	6	keiner	40,6x27,9mm
Nano	168/328	16/32K	0,5/1K	1/2K	14	6	6	FTDI	43x18mm
Lilly Pad	168V/328V	16/32K	0,5/1K	1/2K	14	6	6	keiner	⌀ 50mm

Die Neue Generation

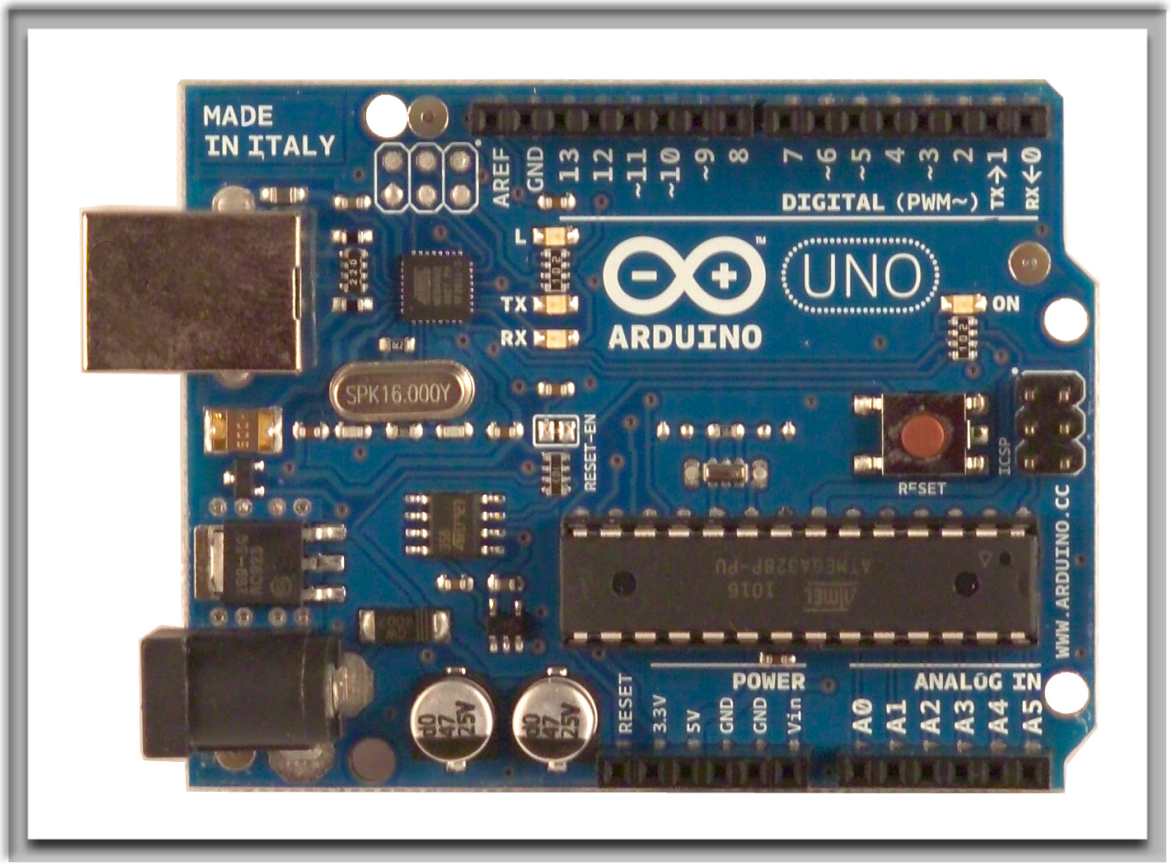
noch flexibler durch
USB Microcontroller



Arduino UNO

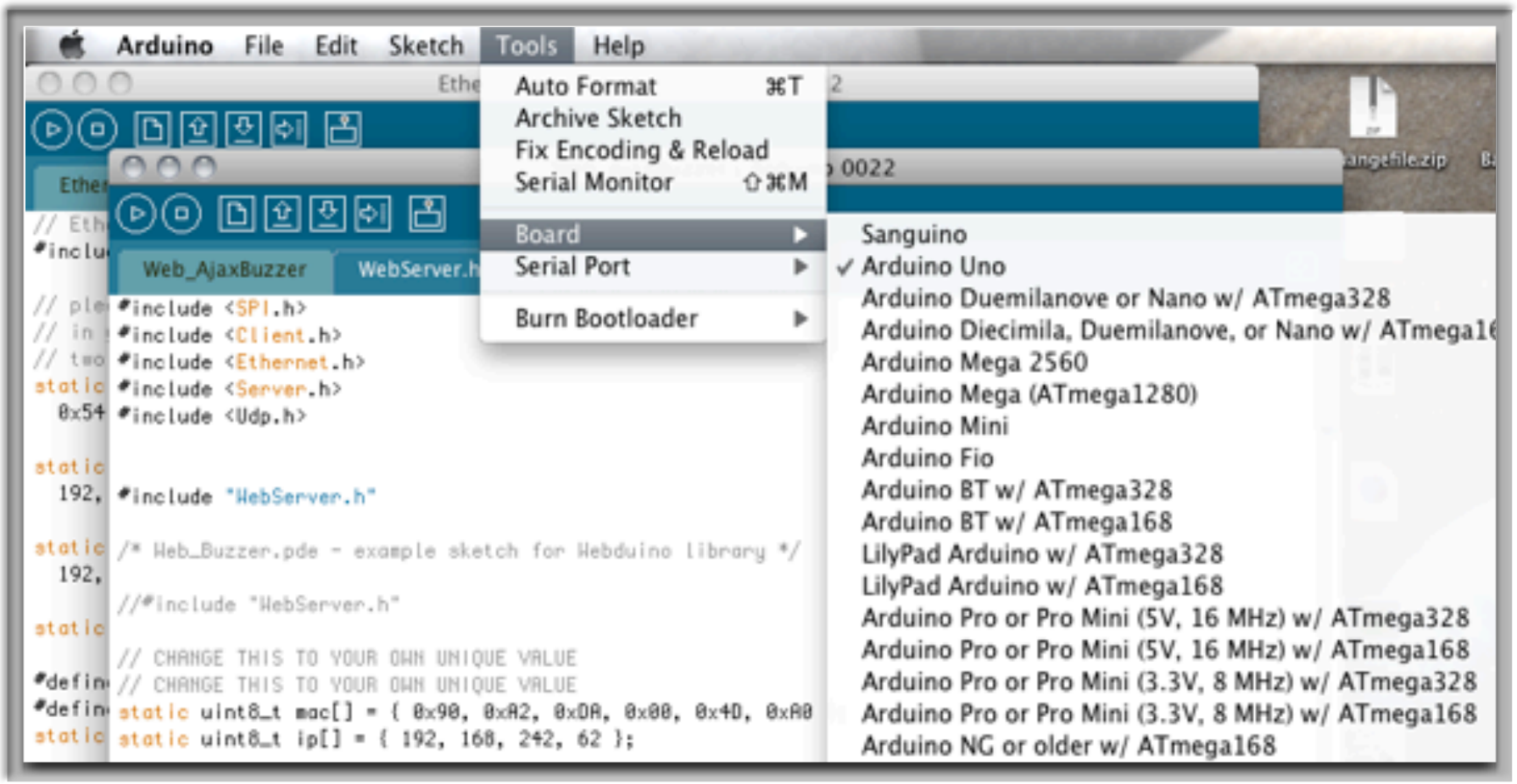
Digital Ports
1xSeriell 6x Analog Out (PWM)

USB Port



Spannungsversorgung

Analog Eingänge



Die Arduino IDE

mehr Basteln als Entwickeln

Press Play to Start

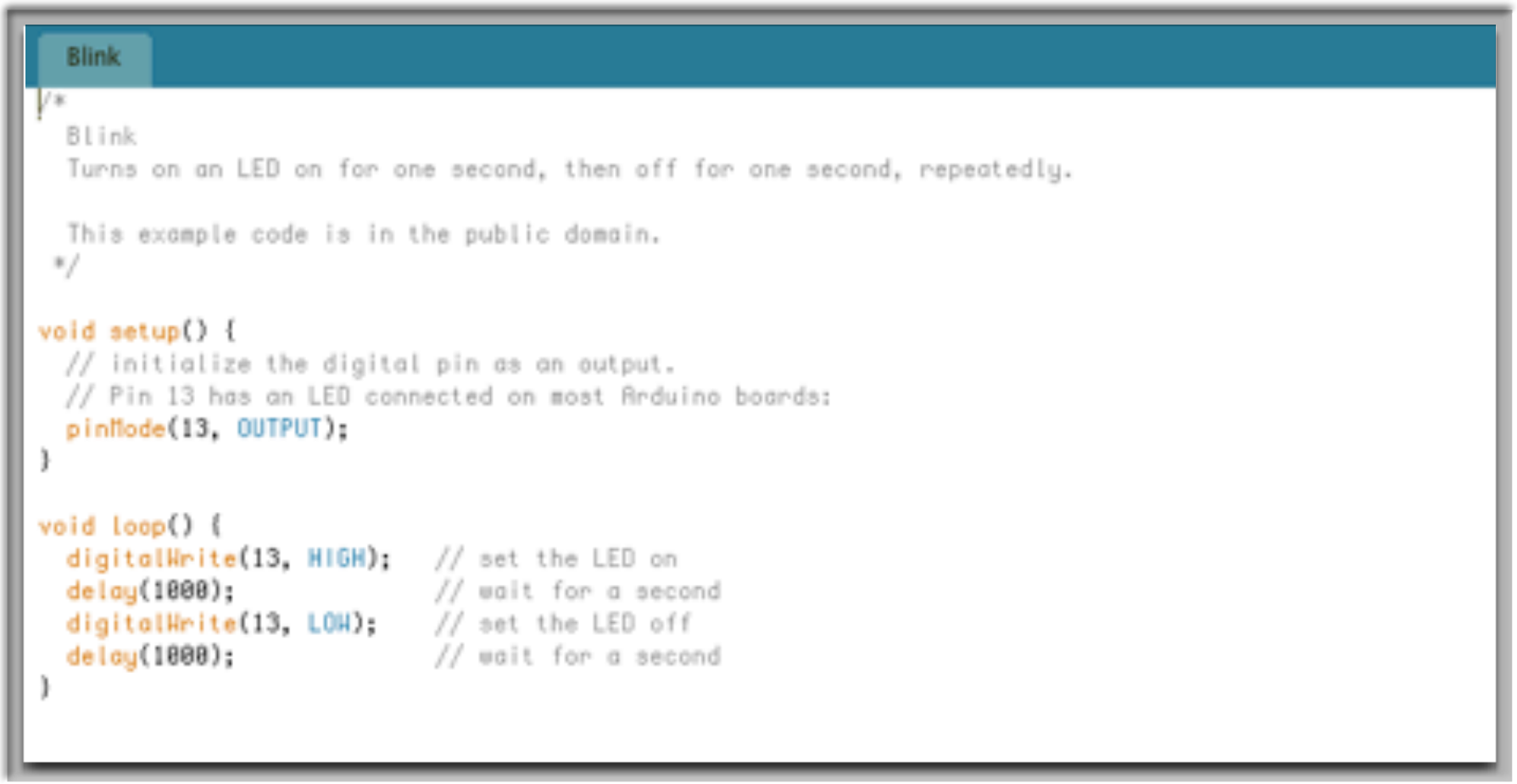
Plattformunabhängig

Sketch Konzept

Tab basiertes Interface

Syntax Hervorhebung

jede Zusatzbibliothek bringt Beispiele mit



Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Die Arduino Programmiersprache

C++ Crash Course - was du wissen solltest

Hauptprogramm

Ein Programm besteht immer aus den Funktionen setup und loop

Am Anfang der Datei holen wir uns mit #include Zusatzfunktionen

Das Ende eines Kommandos muss mit ; gekennzeichnet werden

```
#include Serial.h

void setup() {
   // initialisiere die Seriellen
   // Schnittstellen:
  Serial.begin(9600);
  Serial1.begin(9600);
}

void loop() {
   /* lese von Schnittstelle 1
      und sende es an an
      Schnittstelle 2: */
  if (Serial1.available()) {
    int inByte = Serial1.read();
    Serial.print(inByte, BYTE);
  }
}
```

Kommentare

Kommentare also Hilfestellungen und Gedankenstützen oder auch nicht benutzten Code können wir vor dem Prozessor verstecken

// sagt ignoriere diese Zeile

/* sagt ignoriere alles bis */

```
#include Serial.h

void setup() {
   // initialisiere die Seriellen
   // Schnittstellen:
  Serial.begin(9600);
  Serial1.begin(9600);
}

void loop() {
   /* lese von Schnittstelle 1
      und sende es an an
      Schnittstelle 2: */
  if (Serial1.available()) {
    int inByte = Serial1.read();
    Serial.print(inByte, BYTE);
  }
}
```

wenn dann sonst

wenn der Wert von Eingang grösser als 10 ist schalte Led 10 ein

```
if (Eingang > 10) {
   LedEin(ledPin10);
}
```

wenn der Wert von Eingang gleich 10 ist schalte Led 10 ein, ist der Wert jedoch ungleich 10 aber grösser als 5 ist schalte Led 5 ein, trifft dies auch nicht zu Schalten wir Led 1 ein

```
if (Eingang == 10) {
   LedEin(ledPin10);
}else if (Eingang > 5) {
   LedEin(ledPin5);
}else{
   LedEin(ledPin1);
}
```

während

lade die Variable var mit 0

while() überprüft nun, ähnlich wie if(), ob var kleiner als 200 ist, trifft dies zu wird die Led umgeschalten (von ein nach aus und umgekehrt) und zu var 1 addiert

ist var grösser als 199 wird die Schleife beendet

```
var = 0;
while(var < 200){
   // mach was
   toggleLed(ledPin2);
   var++;
   // macht das gleiche wie
   // var = var +1;
}

// Diese Anweisungen ergeben dass
// die Led 100x aufleuchtet
```

solange wie

lade die Variable var mit 0

for() überprüft nun ob var kleiner als 200 ist, trifft dies zu wird die Led umgeschalten und zu var 1 addiert

ist var grösser als 199 wird die Schleife beendet

```
for(var = 0; var < 200; var++){
   // mach was
   toggleLed(ledPin2);
}

// identisches Ergebnis wie zuvor:
// die Led leuchtet 100 mal auf
```

Unterprogramme

zuerst definiert man die Art des Rückgabewerts void, es wird nichts zurückgeliefert

dann folgt der Name der Funktion hier loop

optional kann man Eingabe Parameter die im inneren der Funktion benötigt werden definieren

```
datentyp name(eingabe parameter) {
   // Anweisungen die die Funktion
   // ausführen soll
}
```

```
void loop() {
   /* lese von Schnittstelle 1
      und sende es an an
      Schnittstelle 2: */
  if (Serial1.available()) {
    int inByte = Serial1.read();
    Serial.print(inByte, BYTE);
  }
}
```

Zugriff auf Variablen

Variablen Namen sind innerhalb von Funktionen geschützt

funktion1 weiß nicht dass die Funktion loop eine Variable mit dem selben Namen besitzt, umgekehrt ist es ebenso

var1 hingegen ist auf oberster Ebene definiert und somit für alle Funktionen sichtbar

```
int funktion1(in){
   // var mit dem Wert 5 laden
   int var = 5;
   // zu var in addieren
   var = var + in;
   return meine;
}
// Globale Variable definieren
int var1 = 15;

void loop() {
   // var mit dem Wert 10 laden
   int var =10;
   // an funktion1 var übergeben
   // und dessen Ergebnis in var2
   // speichern
   var2 = funktion1(var);
   //var2 hat nun den wert 15
   //var ist unverändert 10
   var = var2 + var1;
   // var1 ist eine globale
   // Variable somit hat var nun
   // nun den Wert 25;
}
```

Digitale I/O Pins

in der Funktion setup wird über den Befehl pinMode definiert welchen Pin wir als Ein- oder. Ausgang nutzen möchten

```
pinMode(pin nummer, richtung);

pinMode(13, INPUT); // Eingang
pinMode(12, OUTPUT); // Ausgang
```

digitalWrite schaltet einen Pin entweder Ein oder Aus

```
digitalWrite(pin nummer, status);

digitalWrite(12, HIGH); // ein
digitalWrite(12, LOW); // aus
```

digitalRead schaut ob an einem Pin Spannung anliegt oder nicht und liefert dies zurück

```
digitalRead(pin nummer);

var = digitalRead(13);
// bei geschlossenem Schalter hat
// var nun den Wert HIGH, bei
// offenem Schalter den Wert LOW
```

Analoge I/O Pins

analogRead liefert die Spannung am Pin zwischen 0V und 5V in 1024 Schritten zurück

analogWrite stellt am Pin die angegebene Spannung ein, welche von 0V bis 5V in 255 Schritten reicht.

analogWrite funktioniert nur mit den PWM Pins

```
analogRead(pin nummer);

var = analogRead(3);
// liefert in die Variable var
// einen Wert zwischen 0 und 1024
// zurück. 0 Bedeutet der Eingang
// liegt auf Masse, 1023 hingegen
// dass die Spannung gleich oder
// höher als die Referenzspannung
// (meist 5V) ist

analogWrite(pin nummer,wert);

analogWrite(9, 127);
// setzt den Pulsweitenmodulator
// auf halbes Tastverhältnis d.h.
// eine angeschlossene Led
// leuchtet mit halber Leuchtkraft
// 0 heisst Aus, 255 heisst Ein
```

Byte hereinschieben

shiftIn wird verwendet um Daten aus einem Schiebe-register zu lesen

neben den zu verwendenden Pins muss man auch die Reihenfolge Bits angeben

mit dieser Lösung lässt sich die Anzahl der Eingangspins auf einfache weise erhöhen

```
shiftIn(DatenPin, TaktPin, Reihe);

var = shiftIn(3, 2, LSBFIRST);
// liefert das durch das serielle
// einlesen erzeugte Byte zurück,
// der zuerst eingelesene Wert ist
// das unterste Bit
// Pin 3 ist der Eingang
// an Pin 2 wird ein Taksignal
// erzeugt welches den Sender dazu
// bringt ein Bit nach dem anderen
// zu senden

var = shiftIn(3, 2, MSBFIRST);
// wie oben, jedoch ist der zuerst
// eingelesene Wert das oberste
// Bit
```

Byte rausschieben

shiftOut wird verwendet um Daten in ein Schieberegister zu schreiben

neben den Pins und dem Wert muss man auch die Reihenfolge Bits angeben

mit dieser Lösung lässt sich die Anzahl der Ausgänge auf einfache weise erhöhen

```
shiftOut(DatenPin, TaktPin,
   Reihe, Wert);

shiftOut(3, 2, LSBFIRST,168);
// sendet 168 als die Binärfolge
// 10101000 durch Pin 3 zum
// Empfänger angefangen mit dem
// untersten Bit (0)
// an Pin 2 wird ein Taksignal
// erzeugt welches den Empfänger
// dazu bringt ein Bit nach dem
// anderen zu lesen

var = shiftIn(3, 2, MSBFIRST);
// wie oben, jedoch ist der zuerst
// ausgegebene Wert das oberste
// Bit (1)
```

Seriell senden

in der Startfunktion muss die Baudrate der Schnittstelle mit Serial.begin eingestellt werden

Serial.write sendet daten als bytes über den Seriellen Port

Serial.print sendet Daten als Text, Serial.println sendet zusätzlich einen Zeilenvorschub

```
void setup(){
   // Serielle Schnittstelle 0 mit
   // mit 9.600 Baud starten
  Serial.begin(9600);
   // zum testen senden wir
   //„Hallo“ und ein Enter
  Serial.println("Hallo");
}

int thisByte = 33;

void loop()
{ // sendet wert als byte: !
  Serial.write(thisByte);
   // sendet als dezimalzahl: 33;
  Serial.print(thisByte);
   // sendet wert als binärfolge:
   // 00100001 und ein Enter
  Serial.println(thisByte, BIN);
  thisByte++;
  if(thisByte == 126) {
     thisByte = 33;
  }
}
```

Seriell empfangen

die initialisierung findet wieder in der Startfunktion mit Serial.begin statt

Serial.availabe gibt an wie viele Bytes im Emfangspuffer liegen

Serial.read liest das älteste Byte aus dem Empfangspuffer

```
void setup(){
   // Serielle Schnittstelle 0 mit
   // mit 9.600 Baud starten
  Serial.begin(9600);
}

int inByte = 0;

void loop()
{  // liegen daten an?
   if (Serial.available() > 0) {
      // lies ein byte
     incomingByte = Serial.read();
   }
}
```

Datentypen

void: kein Rückgabewert

boolean: 1 bit true / false

char: 1byte mit Vorzeichen

unsigned char: 1byte ohne Vorzeichen

byte: 1 Byte ohne Vorzeichen

int: 2 Byte mit Vorzeichen

unsigned int: 2 Byte ohne Vorzeichen

word: 2 Byte mit Vorzeichen

long: 4 Byte mit Vorzeichen

unsigned long: 4 Byte ohne Vorzeichen

Datentypen

float: 4 Byte Kommazahl

double: 4 Byte Kommazahl identisch mit float keine höhere Genauigkeit

string: (char array) Ziechenkette aus hintereinander liegenden Variablen vom Typ char

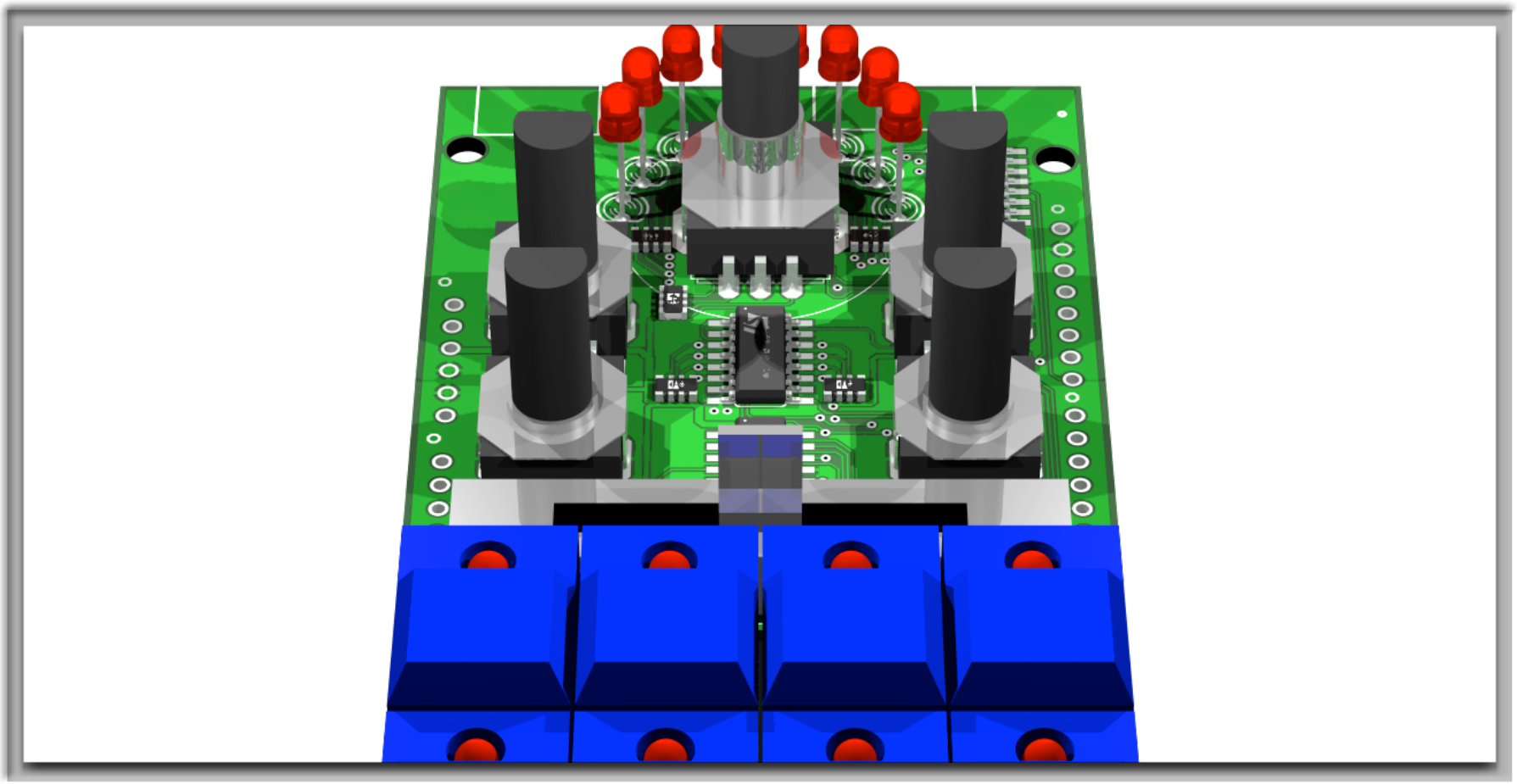
String: (object) Hilfsfunktionen der Arduino Umgebung zur Zeichenkettenbearbeitung

array: Sammelbehälter für Variablen des selben Typs

Nachschlagewerk zur Sprachreferenz

In der Entwicklungsumgebung Integriert

Online unter: http://arduino.cc/en/Reference/

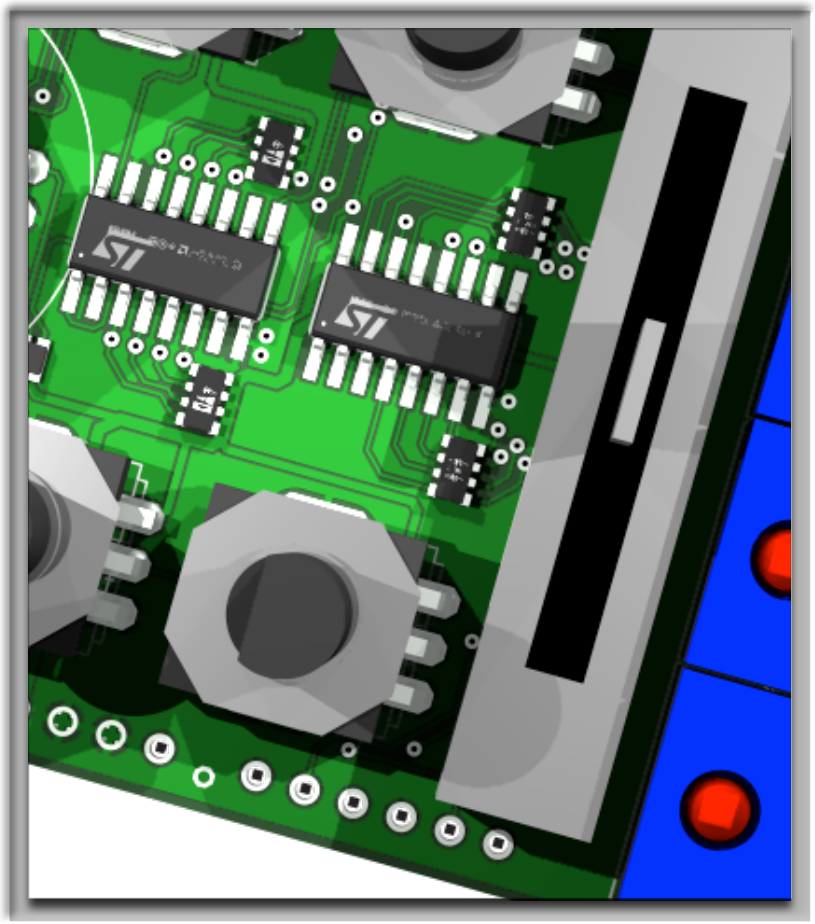


Das miniReACT Schild

Robust gebaut wie sein grosser Bruder

Mehr Tasten und LEDs als erlaubt

- 1 Endlosdrehregler
- 1 LED Ring mit 8 LEDs
- 1 Schieberegler
- 4 Drehpotis
- 8 ReACT Taster mit LED

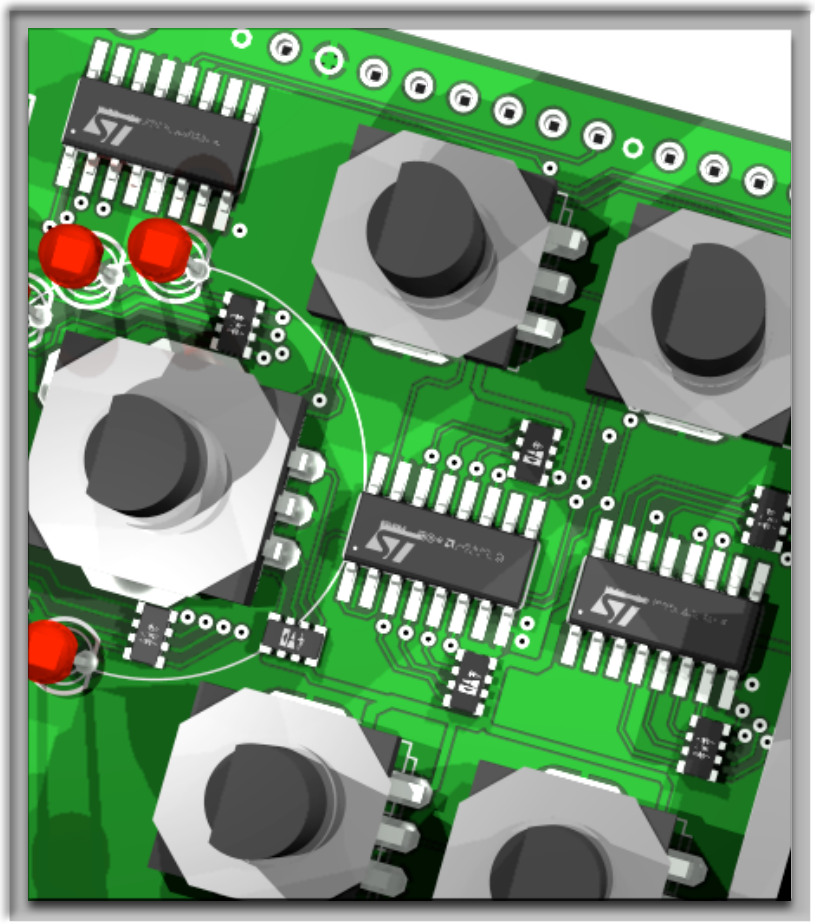


I/O Extender per Shiftregister

zwei 74HC595 8bit Shift Out
Register für die 16 LEDs
Daten Pin: #7
Takt Pin: #9

ein 74HC165 8bit Shift In
Register für die 8 Taster
Daten Pin: #6
Takt Pin: #9

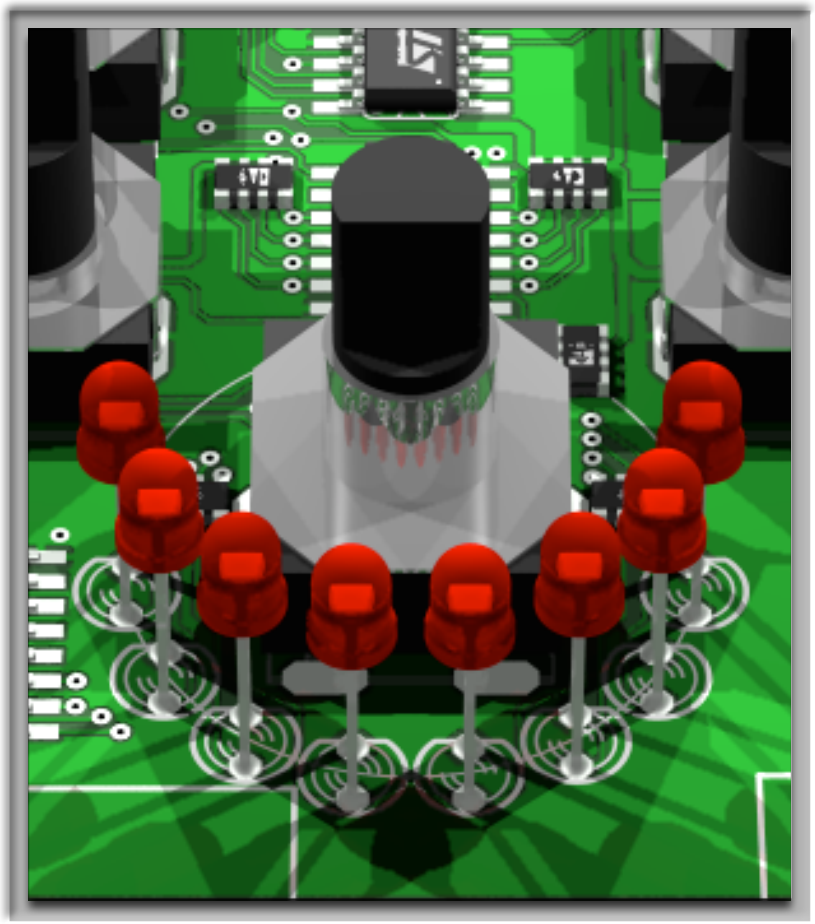
gemeinsamer Lade Pin: #8



Endlosdrehregler

Alps Rotary Encoder
Encoder A Pin: #2
Encoder A Pin: #3

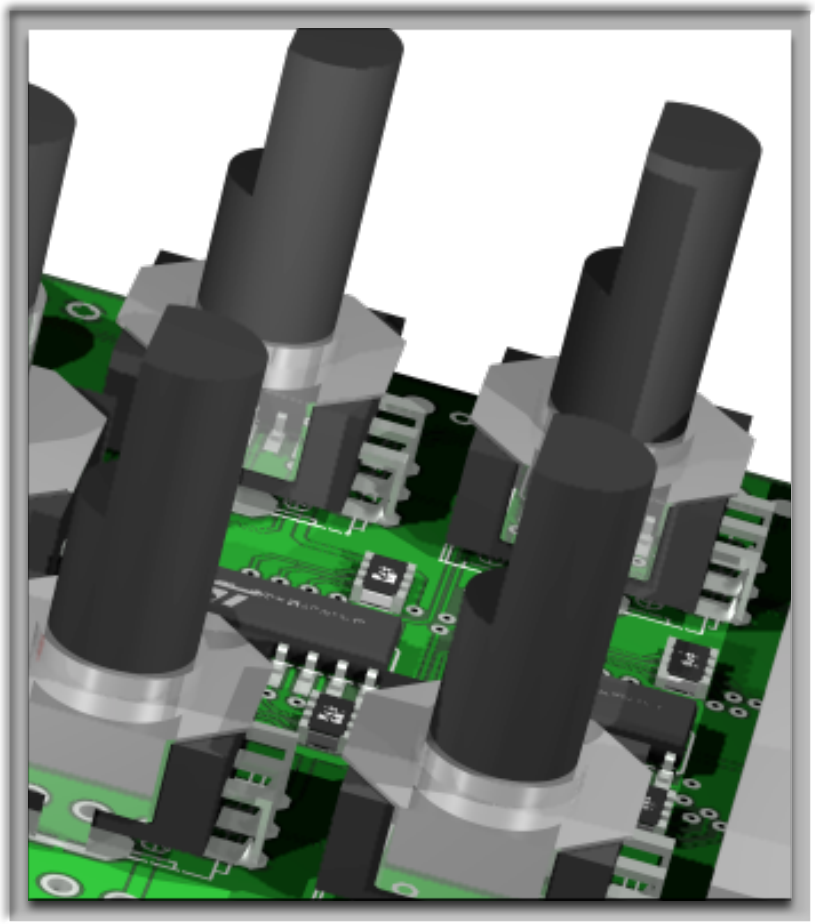
LED Ring via Output Shift Register
Bits 8-15



Drehregler

Alps Drehpotentiometer
Regler links oben: #A1
Regler links unten: #A2
Regler rechts oben: #A3
Regler rechts unten: #A4

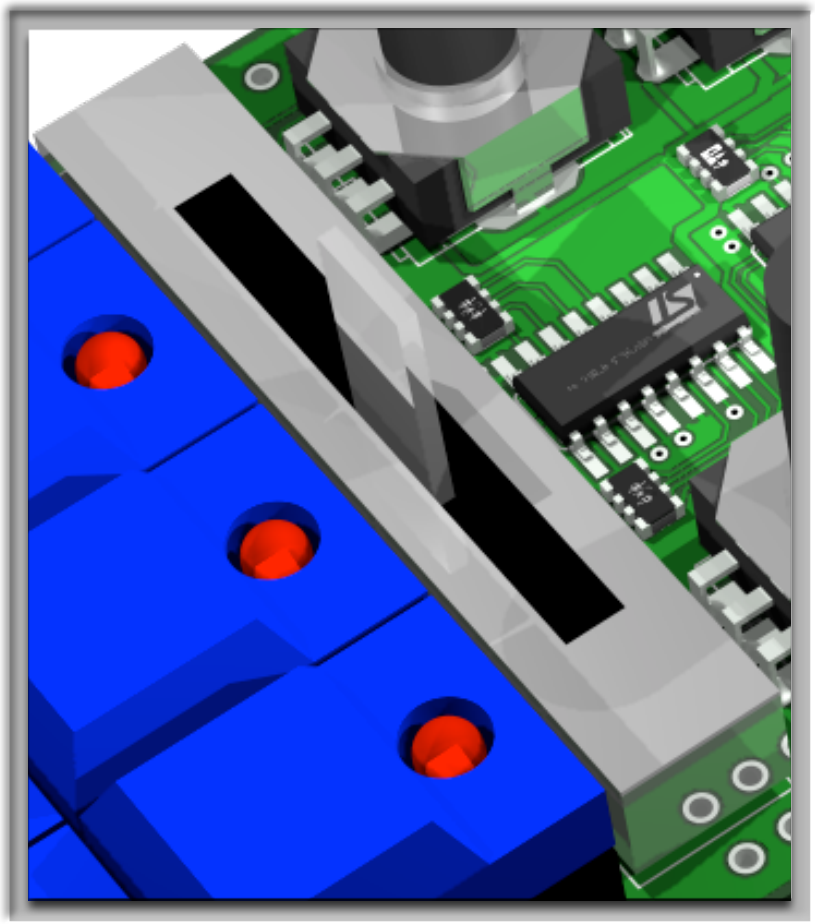
Wertebereich
Anschlag rechts: 255
Anschlag links: 0



„Cross“ Fader

Alps 35mm Fader
Angeschlossen an #A0

Wertebereich
Anschlag links: 255
Anschlag rechts: 0



Lichtpult Taster mit LED

8 Taster via Input Shift Register
Bits 0-7

8 LEDs via Output Shift Register
Bits 0-7