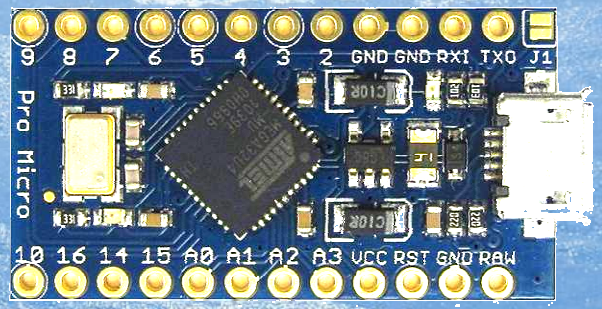
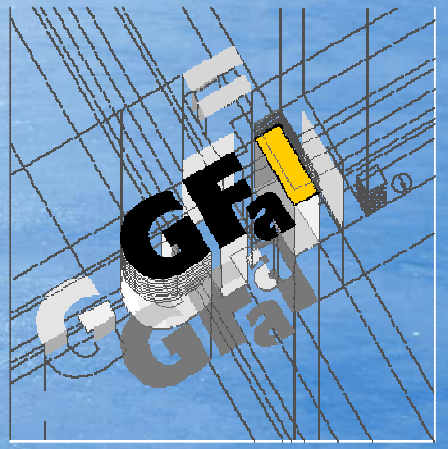
Arduino in Beispielen - Überblick und Grenzen





Dr. G. Heinz, GFaI e.V.
Volmerstr. 3
12489 Berlin
Tel. +49 (30) 814563-490
Fax. -302
www.gfai.de
heinz@gfai.de

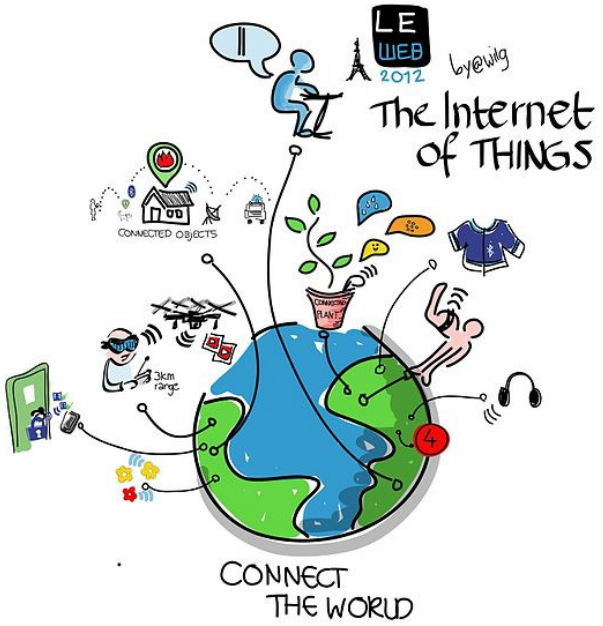
Wer oder was ist Arduino?

- Embedded Soft- und Hardware
- Bar in Ivrea bei Turin, nach Arduin von Ivrea (König von Italien 1002-1014)
- Board No. "Uno" 2005 von Massimo Banzi und David Cuartielles, C++ by David Mellis
- Compilerkonzept aufbauend auf "Processing" (Java) und GNU-C
- Erste 200 Unos wurden an Schulen verkauft
- Prozessor ATmega8 (8bit, 8kB, 16MHz, 5V), später ATmega328 (32kB, 5V)
- Erste Honorierung: 2006 "Prix Ars Electronica"
- Marken-Rechtsstreit 2015: Genuino

Kennzeichen:

- USB-Bootloader (0,5 … 2kB)
- komplexe Software-Libraries (C++ Class Libraries)
- nach Funktion sortierte Pin-Nummern
- vereinfachende, generalisierende Konzeptidee

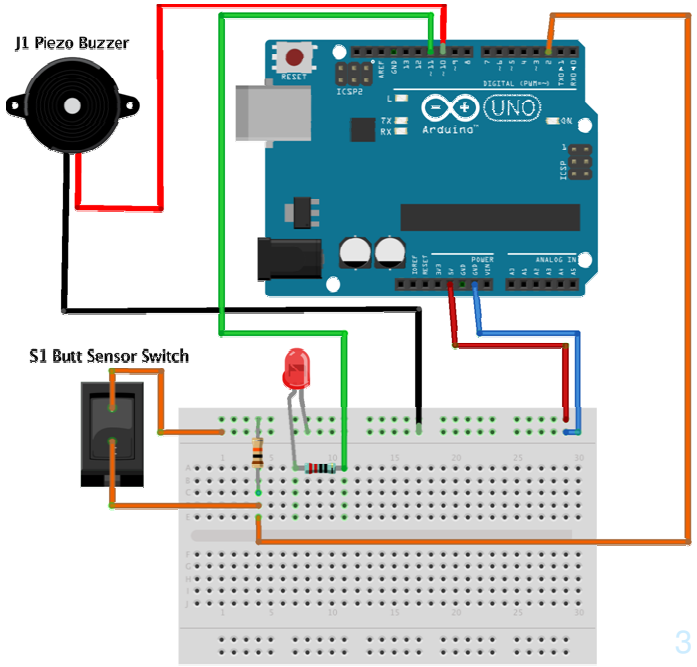
Learning Aspect of Arduino Project



Adressiert sind

- Hobby
- Studentenausbildung
- Versuchsbau
- Gerätebau
- Prototypen

... wo es schnell gehen muß



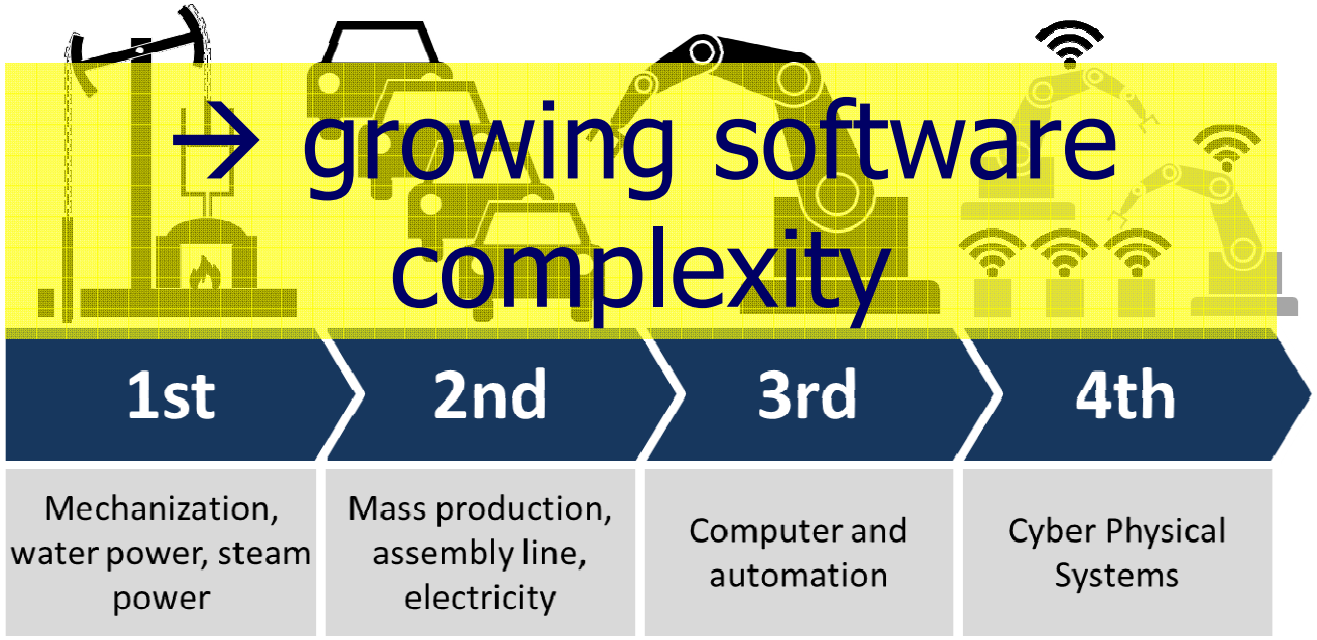
Zur Dokumentation: fritzing.org

Fourth Industrial Revolution

- IoT - Internet of Things
- IoS - Internet of Services
  - sensor-intelligence
  - incomplete information proc.
  - data communication & security
  - big data
  - monitored actors

"Industrie 4.0 is the fourth industrial revolution"
Klaus Martin Schwab

Biography: *30.3.1938, Ravensburg, German engineer and economist, founder and executive chairman of the World Economic Forum and the Schwab Foundation for Social Entrepreneurship



Geschwindigkeit contra Abstraktion

Bits setzen auf Registerebene:

- höchste Codeeffiziens
- geringster Speicherbedarf
- schnellste Ausführung
- hoher Programmieraufwand
- geringe, erreichbare Komplexität

```
// Beispiel AVR-Studio4 Soft-UART für ATtiny85
void startTimer(uint8_t time2wait) { // UART bit clock ATtiny85, see p.64 ff
  // timer config (prescal 8; stop at 125 for 1 bitclock or 187 for 1.5 bitclocks
  TCCR0B |= (1<<CS01);          // prescaler division by 8, p.72
  TCCR0A  = (1<<WGM01);         // CTC-mode output compare match, p.69
  TCNT0 = 0;                    // reset timer value
  OCR0A         = time2wait - 1;  // set output compare match register, p.74
  TIMSK0        = (1<<OCIE0A); // enable output compare A interrupt, p.74
 sei();
}
```

code-size ~ 10 byte

C++ Bibliotheken nutzen:

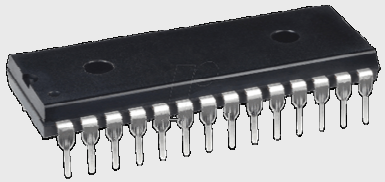
- hoher Speicherbedarf
- reduzierte Flexibilität
- langsamste Ausführung
- geringer Programmieraufwand
- hohe, erreichbare Komplexität

```
// Beispiel Arduino USB-HID-Mouse
#include <SoftwareSerial.h>    // Soft-UART mit Pin-change Interrupt
SoftwareSerial SoftUart(9, 8);   // Arduino-proMicro Pins (RX, TX)

void setup()                  // Hardware-UART hängt an USB-Adapter
{                             // mehr als 57600 schafft die Soft-UART nicht
  Serial.begin(57600);        // USB-HID Kommunikation als Output UART
  SoftUart.begin(57600);  // zusätzliche Soft-UART als Input (!)
  Mouse.begin();          // initialize USB-HID mouse control (!)
}
```

code-size ~ 10 kilobyte

(Arduino: höchste Effiziens bei langsamster Ausführung)

Wie sieht ein moderner Einchip-µC aus?



CPU

Speicher

Flash

EPROM

SRAM

erfordert EPROM-Fertigungsprozeß!

I/O

dig I/O ↔ schalten

ADC ← analog In

DAC → analog Out

Timer ← Zeit messen

Timer → PWM

Serial Comm.

UART ↔

SPI ↔

I²C ↔

... ↔

USB ↔

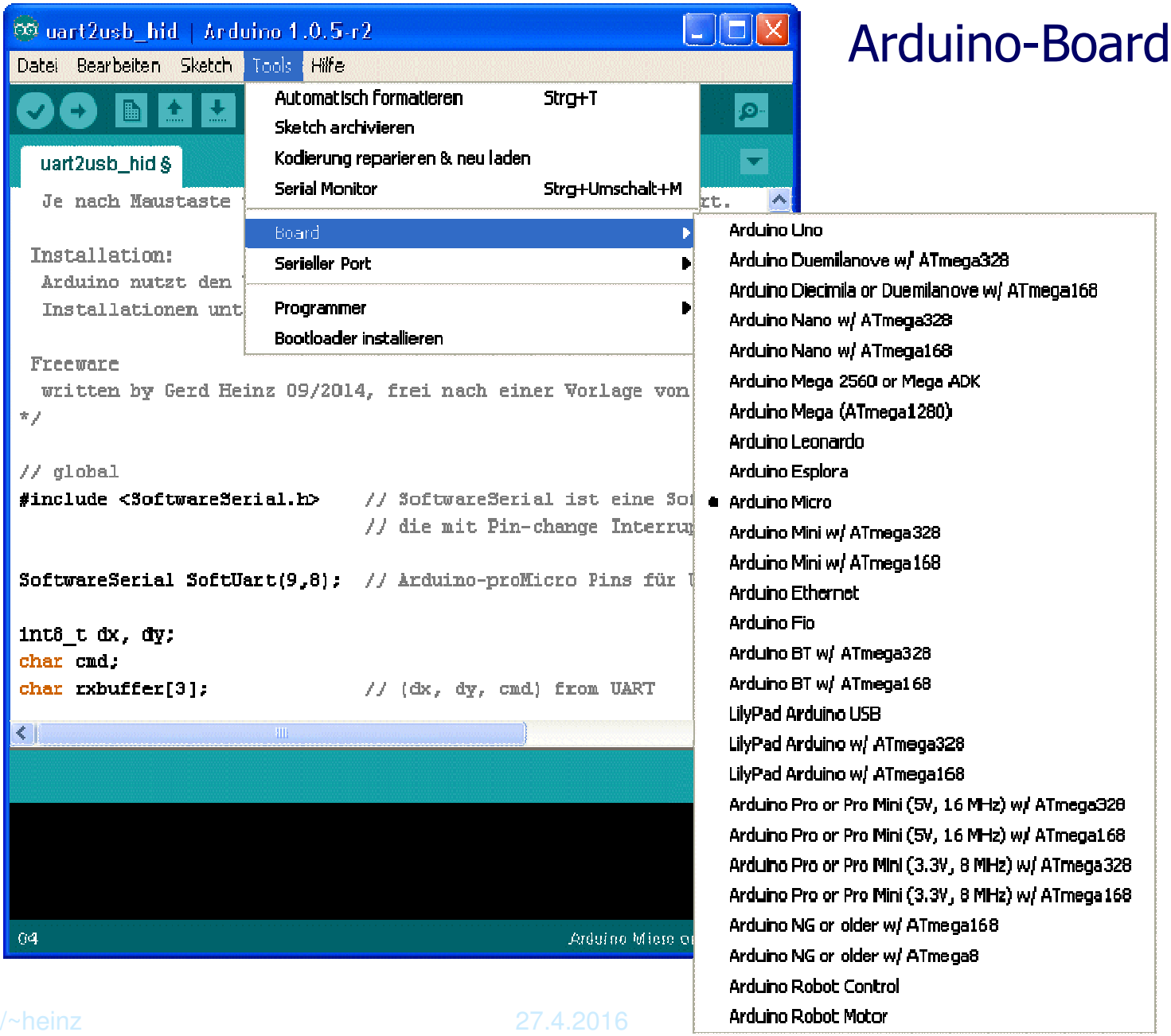
VCC

GND

Pins

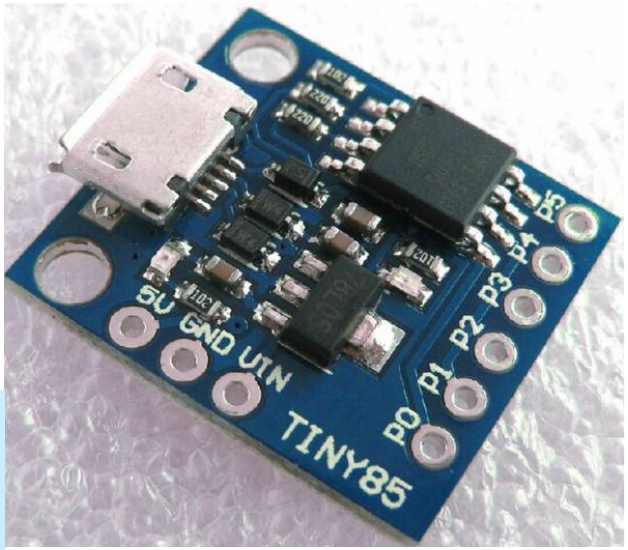
Nicht jedes Pin hat Zugriff auf alle peripheren Baugruppen → Datenblatt

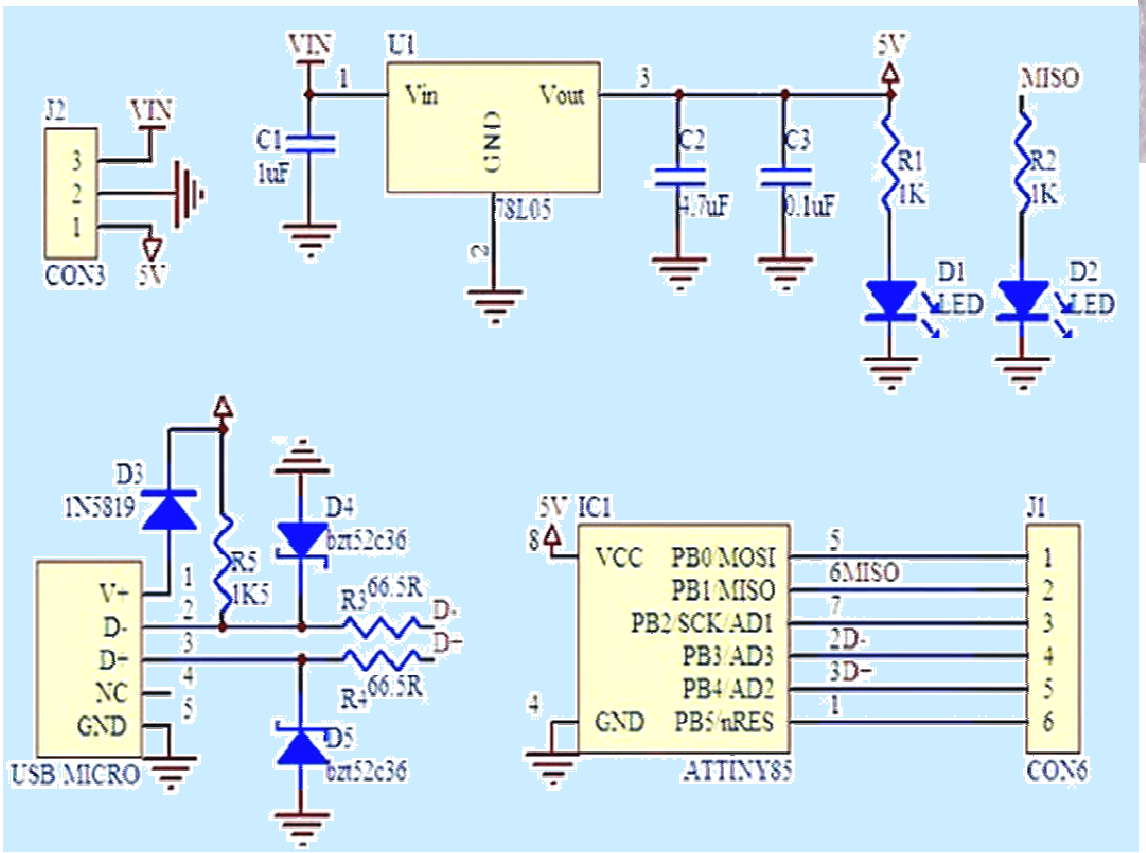
Arduino-Boards



Kleinster: Digispark TINY85

- 5 Pins, 5 Volt, 8 Bit, RAM 0.5kB, Flash 8kB, 8MHz
- nur USB-Code-Download, kein VCP
- Win32 Libusb, Zadig Micronucleus mit 1.5kB
- wartet vor dem Start 5 Sekunden auf USB





"Echte" Arduinos

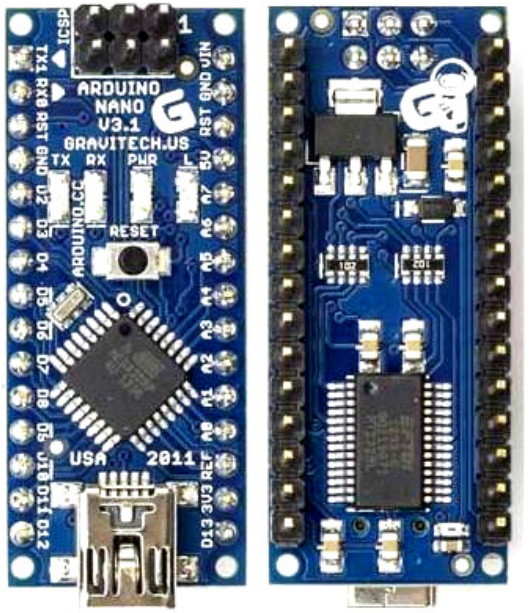
Kennzeichen:

- 5Volt, 8-Bit Prozessor + Quarz + LED + USB-Bootloader

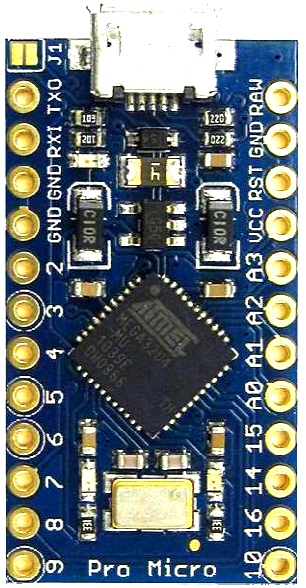
Kompaktklasse:

- Nano: ATmega328 + FT232RL
- proMicro: ATmega32U4 (32MHz, 6 USB-Endpoints)
- Mini: Nano ohne USB-Chip

Nano



proMicro



Populärste Nachahmer:

- mbed (ARM) 32bit Cortex-M-Controller Boards aller Hersteller werden bedient
- STM32F401 Nucleo
- Intel Galileo, Intel Edison …

32 Bit Arduino für IoT

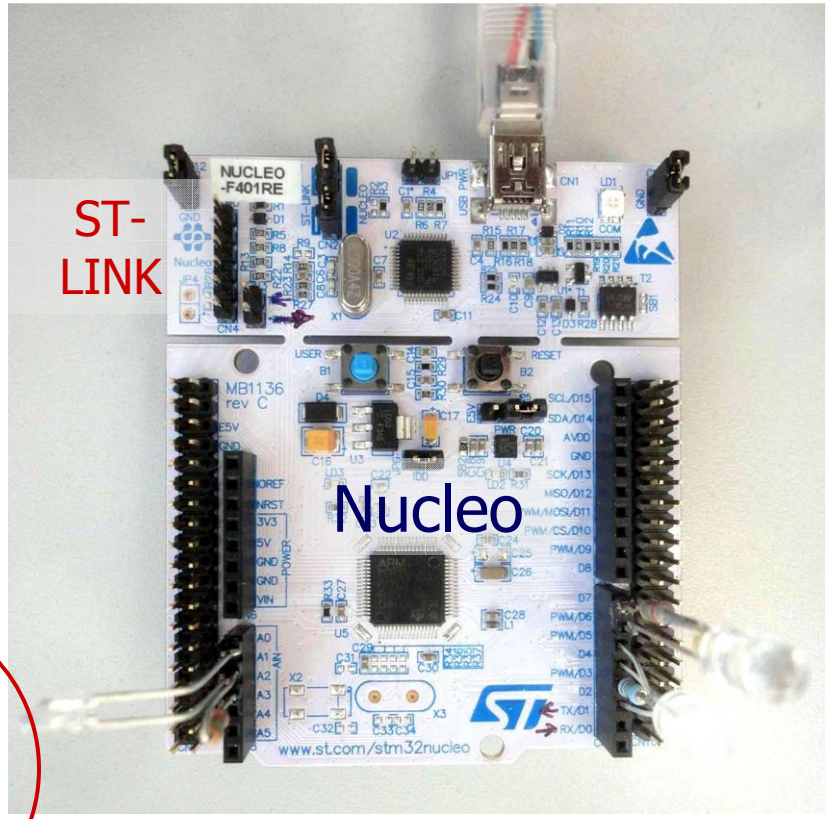
- MKR1000: µC + USB + WLAN + LiPo-Akku

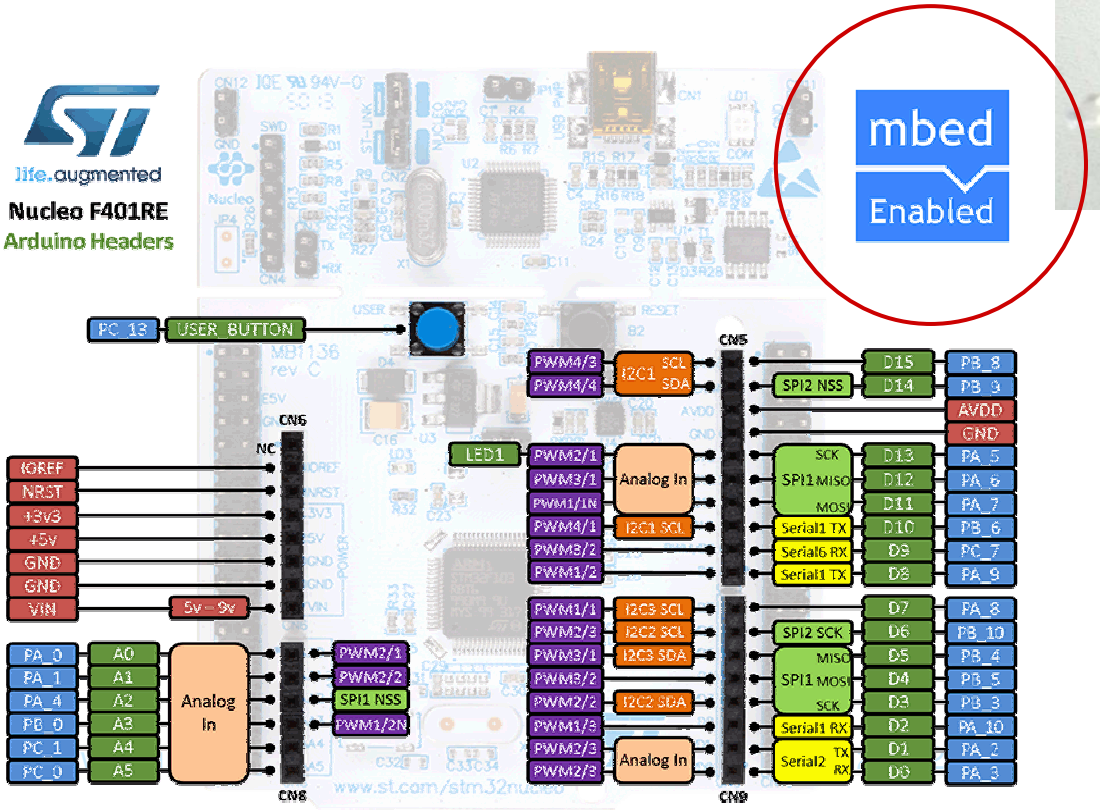
- Prozessor ATMEL SAMW25:
  - 3.3V, 32 Bit, 48 MHz
  - Cortex-M0 ARM MCU
  - Wi-Fi WINC1500 2.4GHz IEEE 802.11 b/g/n
  - CryptoAuthentication ECC508

MKR1000

Nucleo STM32F4…

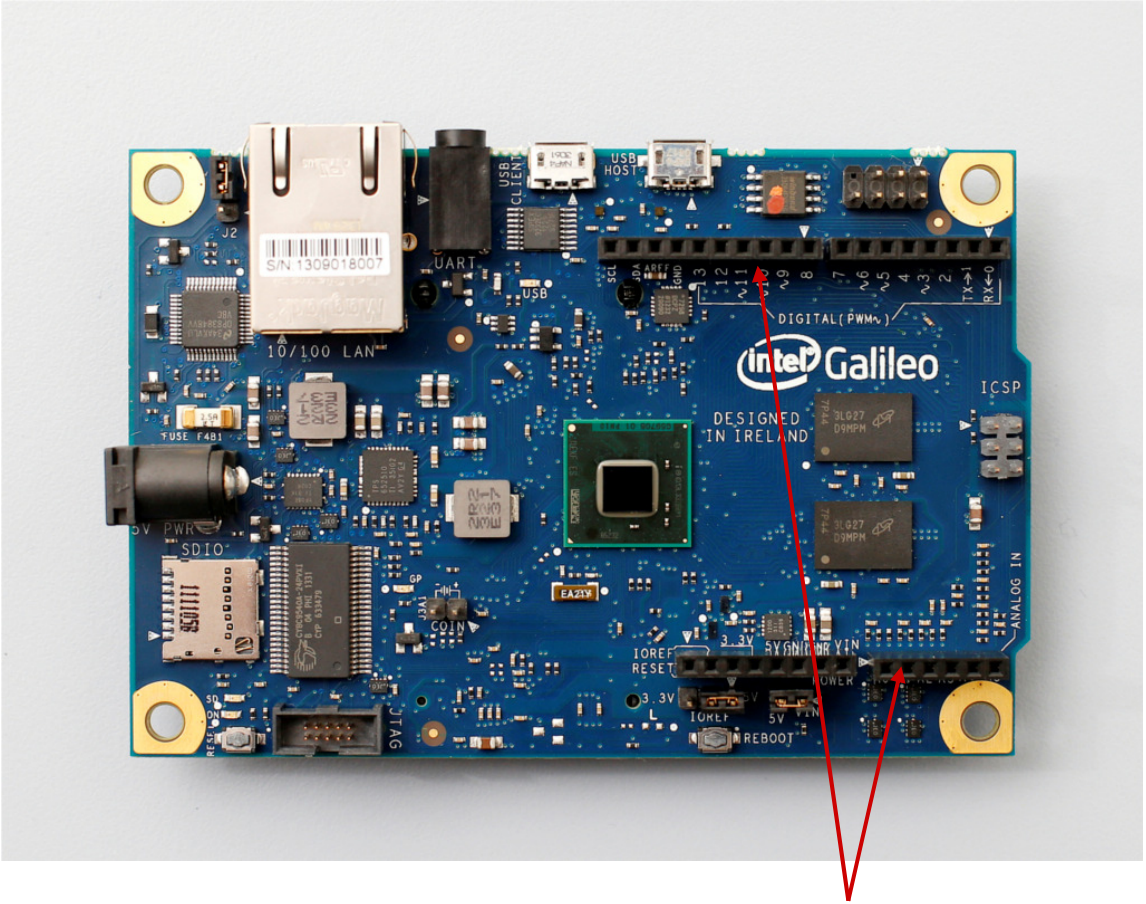
- 32 Bit, 3.3V, 84MHz, 128…512kB, FPU
- ST-LINK (brrr!) Serial Wire Debug
- USB: VCP + CDC + debug
- Arduino-Stecker (CN6, CN5)
- mbed gehört ARM





Intel Galileo

- 32 Bit Pentium
- 100 MHz µC
  "Quark SoC X1000"
- PCI-express
- ETH 100Mb
- SD-slot
- USB Host
- USB Client
- 8MB NOR-Flash
- 512kB SRAM
- "eco" 5V/800mA = 4 Watt (!)
- "Blinky" USB-Boot: ~ 60kB (!)



Arduino-Uno-Stecker (!)

https://www.arduino.cc/en/ArduinoCertified/IntelGalileo

Intel Edison

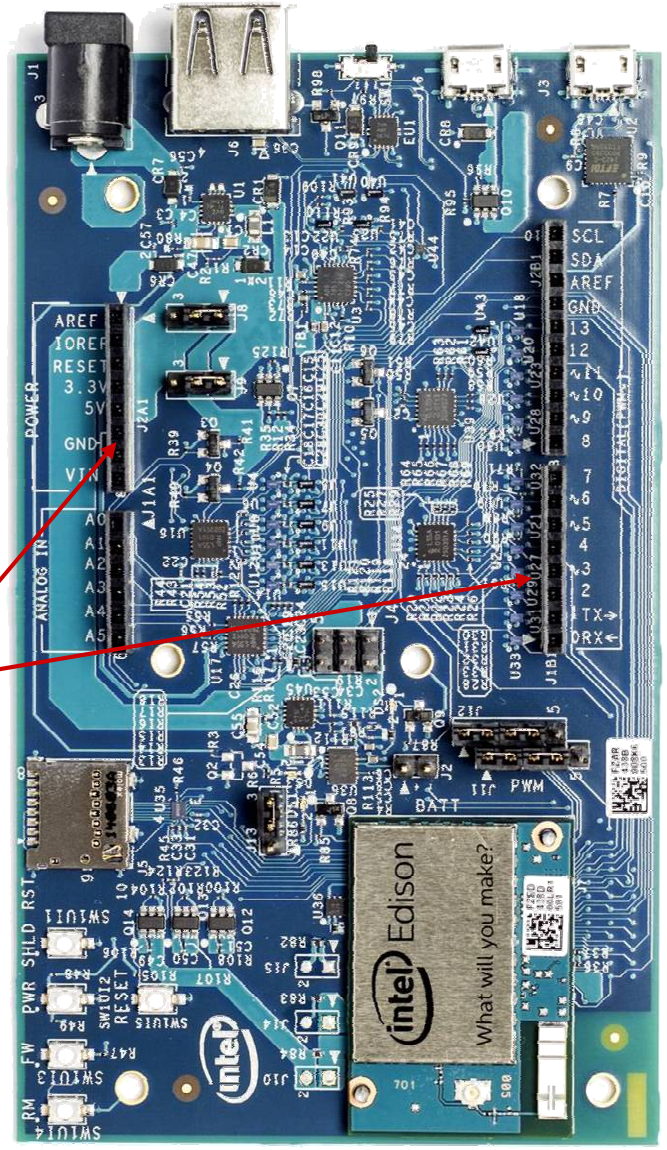
- Dual Core Intel Atom 500MHz
  + Intel Quark microcontroller 100MHz
- USB
- ETH
- WiFi
- Bluetooth
- 4 PWM outputs (!)
- SD-slot

Support for

- Yocto Linux
- Python
- Node.js
- Wolfram

→ is this realy Arduino???

https://www.arduino.cc/en/ArduinoCertified/IntelEdison



Software Arduino

- C++ Class Libraries
- C-Syntax, GNU-Compiler, Java IDE
- main.cpp unsichtbar, nur
  - void setup() und
  - void loop() verfügbar
- cpp-Files heißen *.ino

Interessante Abstraktionen, z.B.:

- analogRead meint den AnalogDigitalConverter (ADC)
- analogWrite ruft Timer für Pulsweitenmodulation (PWM)
- Serial initialisiert Hard-UART
- SoftwareSerial baut Soft-UART
- Mouse und Keyboard erzeugen USB-HID-Devices

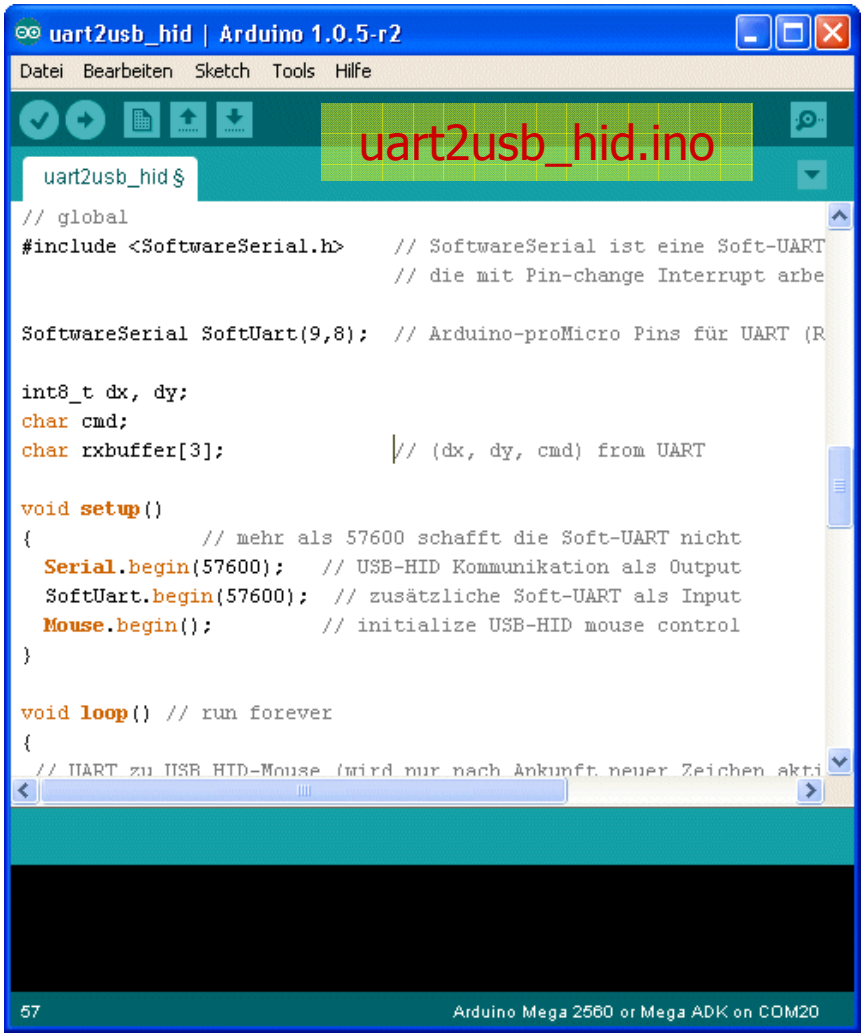
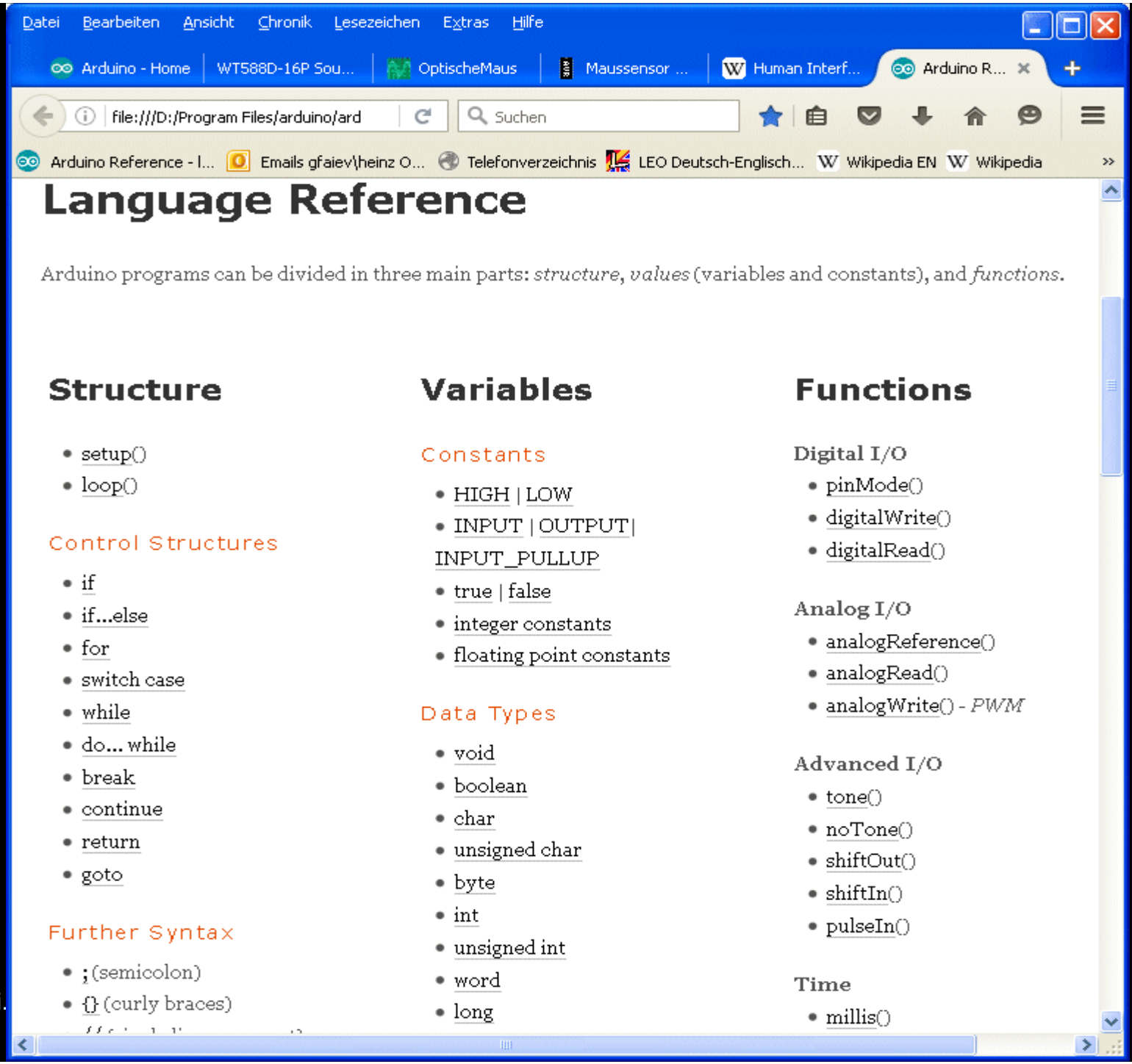
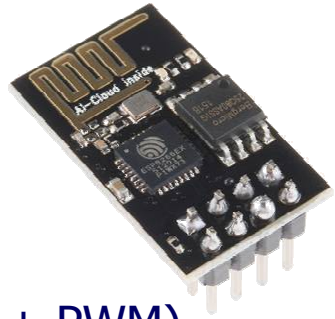


Bild: uart2usb_hid.ino
Drei Zeilen Setup genügen, um eine USB-HID-Maus zu bauen: genial.


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Arduino - Home
WT588D-16P Sou...
OptischeMaus
Maussensor ...
Human Interf...
Arduino R...
file:///D:/Program Files/arduino/ard
Suchen
Arduino Reference - I...
Emails gfaiev\heinz O...
Telefonverzeichnis
LEO Deutsch-Englisch...
Wikipedia EN
Wikipedia
Language Reference
Arduino programs can be divided in three main parts: structure, values (variables and constants), and functions.
Structure
setup()
loop()
Control Structures
if
if...else
for
switch case
while
do... while
break
continue
return
goto
Further Syntax
; (semicolon)
{} (curly braces)
Variables
Constants
HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
true | false
integer constants
floating point constants
Data Types
void
boolean
char
unsigned char
byte
int
unsigned int
word
long
Functions
Digital I/O
pinMode()
digitalWrite()
digitalRead()
Analog I/O
analogReference()
analogRead()
analogWrite() - PWM
Advanced I/O
tone()
noTone()
shiftOut()
shiftIn()
pulseIn()
Time
millis()

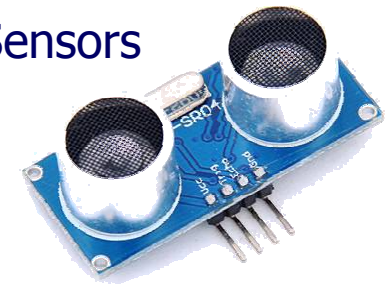
Add-On Module

www.sparkfun.com



Sensoren

- Humidity-Temperature Sensors
- Gas-Sensoren
- Ultraschall-Sensoren
- GPS-Receiver
- 3-Achs Gyrometer
- Optical Flow Sensors (Mouse)
- Mikrofone
- Capacitive Touch Sensors
- Triangulation

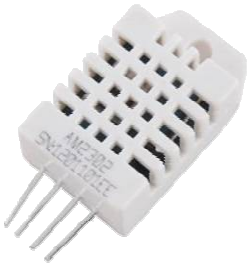


Kommunikation

- CAN-, RS485-Transceiver
- WiFi (ESP8266)
- Bluetooth-Transceiver
- Ethernet-Shields

Aktoren

- Power-MOSFETs (I/O + PWM)
- Servos (analog + digital)
- Brushed Driver
- Brushless Driver
- Stepper Driver
- MP3-, WAV-Audio Player
- Audio Amplifier + Speaker

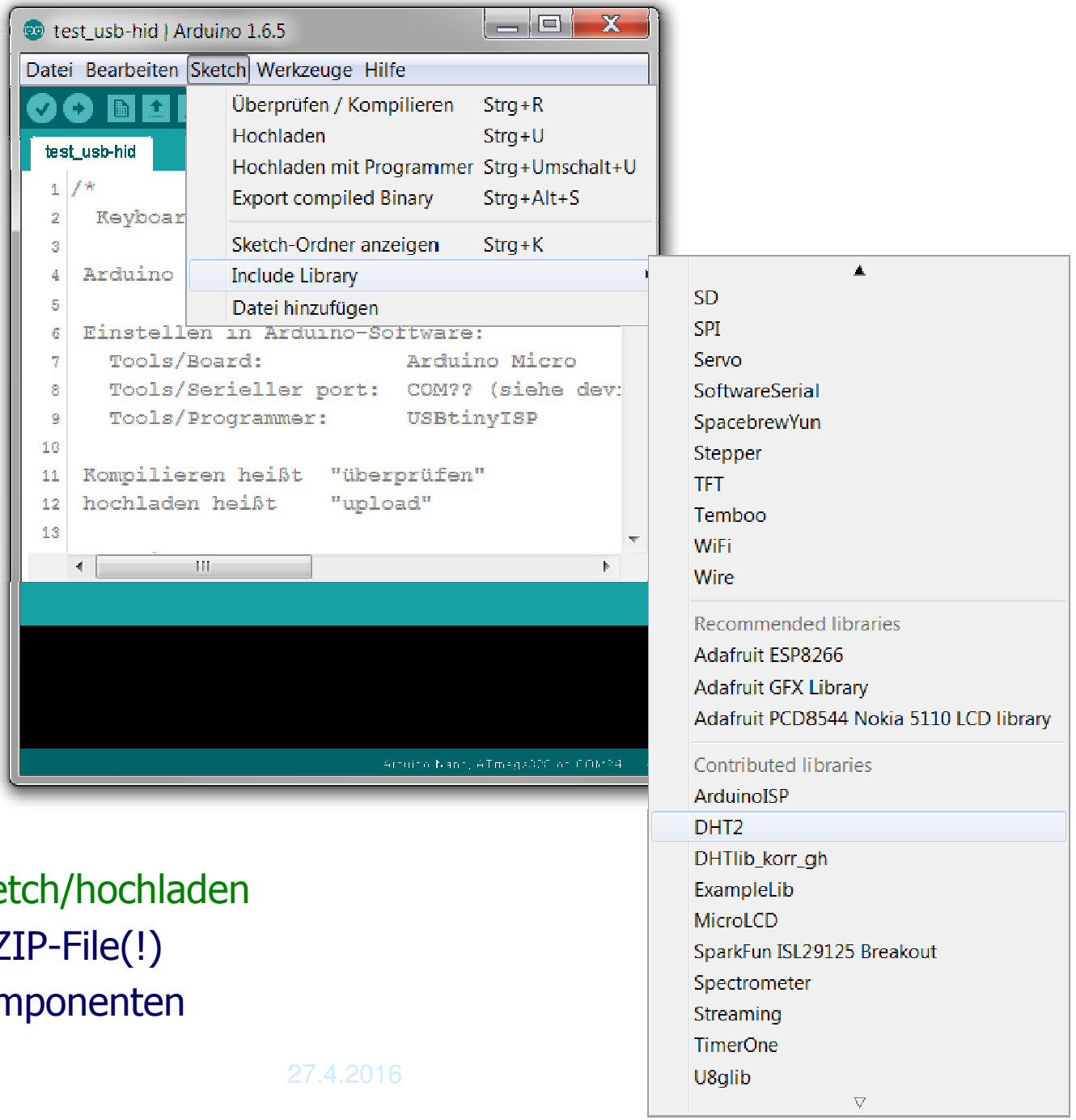


Displays

- RGB-LEDs
- Alpha LED-Displays
- Numerical LCD-Displays
- Graphical LCD-Displays

... dazu hunderte Libraries

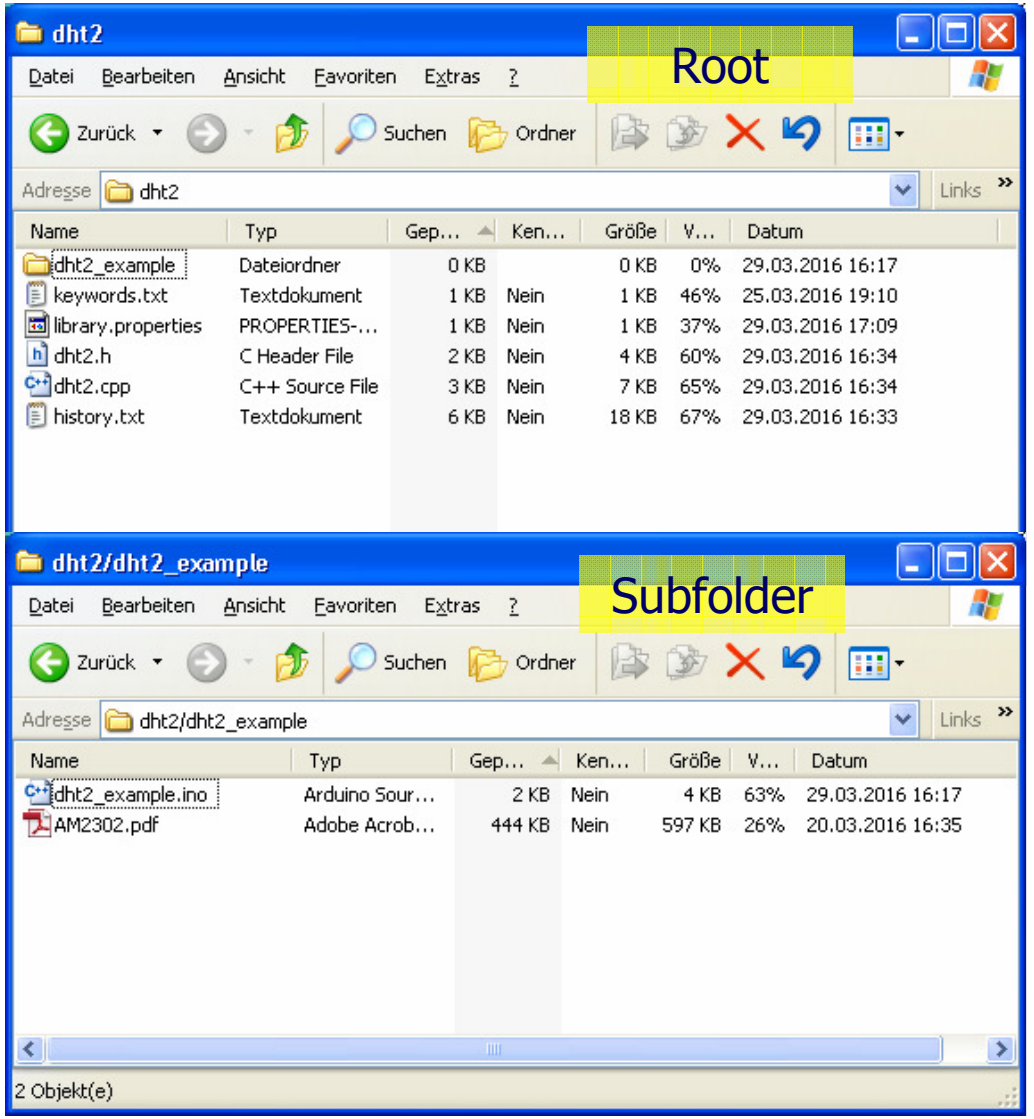
Libraries einbinden



- hochladen: Sketch/hochladen
- Library ist ein ZIP-File(!)
- enthält alle Komponenten

Inhalt einer Library (Beispiel dht2.zip)

- *.cpp und *.h
  sind die eigentliche Library
- library.properties
  Help-documentation
- history.txt
  What is new etc.
- keywords.txt
  highlightening words, like #include, #define ...
- Subfolder with example as *.ino



… Aufbau der Bibliothek (Beispiel dht2.h)

```
class dht2
{               // now multiple sensors possible - bug removed, gh
public:
    // read() return values:
    // DHTLIB_OK
    // DHTLIB_ERROR_CHECKSUM
    // DHTLIB_ERROR_TIMEOUT
    // DHTLIB_ERROR_CONNECT
    // DHTLIB_ERROR_ACK_L
    // DHTLIB_ERROR_ACK_H
    int read(uint8_t pin);  // for DHT22 / AM2302 and compatible
    double humidity;        // relative humidity in %
    double temperature;  // temperature in °Celsius
    double abshumidity;  // absolute humidity in g/m³  - new gh
    double taupunkt;        // condensation point in °Celsius – new gh

private:
    uint8_t databyte[5];   // buffer to receive serial data
    int readSensor(uint8_t pin, uint8_t wakeupDelay, uint8_t leadingZeroBits);
    double satdampfdr(double T);                        // new gh
    double dampfdruck(double relH, double T);      // new gh
    double abshum(double relH, double T);           // new gh
    double dewp(double relH, double T);             // new gh
};
```

Arduinos versteckte Datei: main.cpp

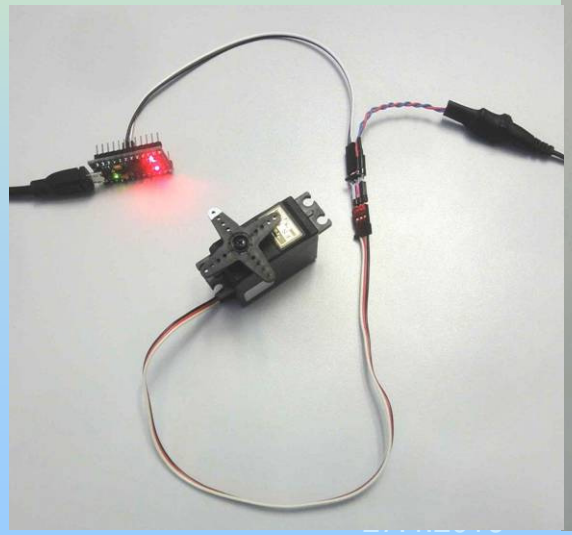
```
#include <Arduino.h>

int main(void) {
    init();
#if defined(USBCON)
    USBDevice.attach();
#endif
    setup();
    for (;;) {
        loop();
        if (serialEventRun) serialEventRun();
    }
    return 0;
}
```

main.cpp

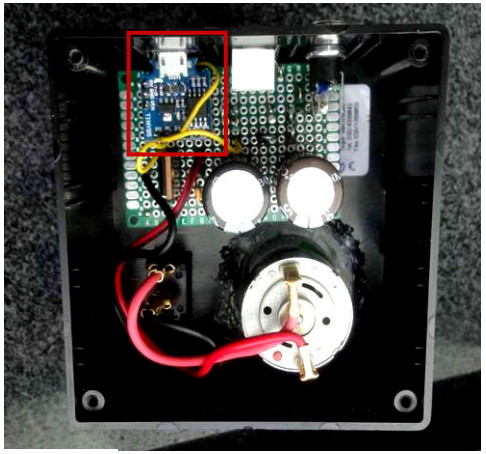
Speicherort: …\Program Files\arduino\arduino105\hardware\arduino\cores\arduino\main.cpp

Beispiele

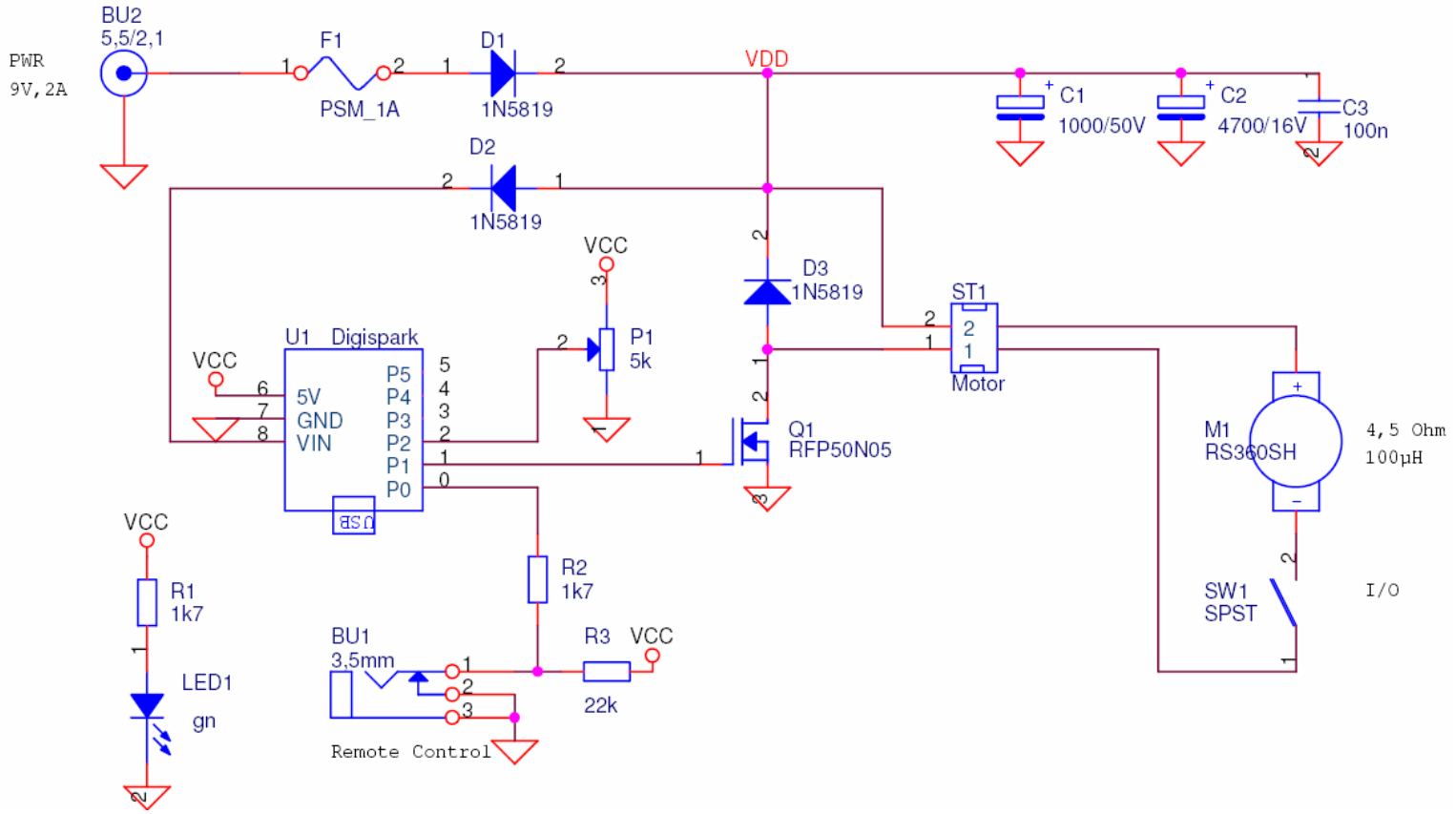
- 10chl DHT-Recorder
- Step Motor Driver
- Servocontrol
- Ultrasonic Distance Monitor
- USB-HID-Mouse Control
- Signalgenerator
- DC-DC-Converter Zahnradpumpe
- Graphic LCD-Control
- .....



Digispark TINY85 für Steuerung einer Zahnradpumpe



- Motor 3V/2A soll an einem Netzteil 9V/1A betrieben werden
- Lösung: PWM-Regler mit Digispark-TINY85 (6.50€ Segor)



zu Digispark Tiny85: Steuerung der Zahnradpumpe

```
void init_pins(void) {
  pinMode(remotepin, INPUT);            // Fernschaltung
  pinMode(motorpin, OUTPUT);            // Motor-PWM Ausgang
  pinMode(potipin, INPUT);              // Potentiometer
}

void setup() {
  init_pins();
  analogWrite(motorpin, 0);             // Motor Stop
}

void loop() {
    remoteval = digitalRead(remotepin); // Fernschaltung abfragen
    potival = analogRead(1);            // potipin ablesen, returns 0...1023
    // Skalierung: out = map(input, fromL, fromH, toL, toH)
    uint8_t pwm = (uint8_t) map(potival, 1023, 0, pwmmin, pwmmax);
    if (remoteval != 0) pwm = 0;        // Remote-Abschaltung, falls offen
    analogWrite(motorpin, pwm);         // 0...255, PWM mit 490 Hz (!Grenze!)
    delay(100)        // ms
}
```

Digital Humidity Recorder

COM5

Senden

```
>h
Help
 a  actual values of all channels
 c  calibrate all active channels
 d  show calibration differences
 h  help, show this menue
 i  show infos about the tool
 p  periodic scan (3 seconds)
 r  reset channel calibration
 s  show channel status messages
 v  version

>i
 10-Chl Humidity and Temperature Recorder
 Sensors: Adsong AM2302 (DHT22-compatible)
 USB-VCP: FTDI, UART: 115200 Baud, 8n1n
 Board: Arduino Nano3.0, 5V
 Microcontroller: ATmega328 + FT232RL
 Program version: dht_recorder_v20 2016/04/20
 Compiler: Arduino 1.05-r2
 DHT-Library: DHT2.zip v0.1.20
 Hotline: heinz@gfai.de
 Pins RJ10/RJ45:  3:VDD  4:SDA  5:nc  6:GND

>a
[Chl; Temp(°C); relHum(%); absHum(g/m³); dewP(°C);]
 0;     31.1;   26.0;   8.37;   9.3;
 1;     27.3;   28.3;   7.41;   7.4;
 2;     27.2;   29.6;   7.70;   7.9;
```

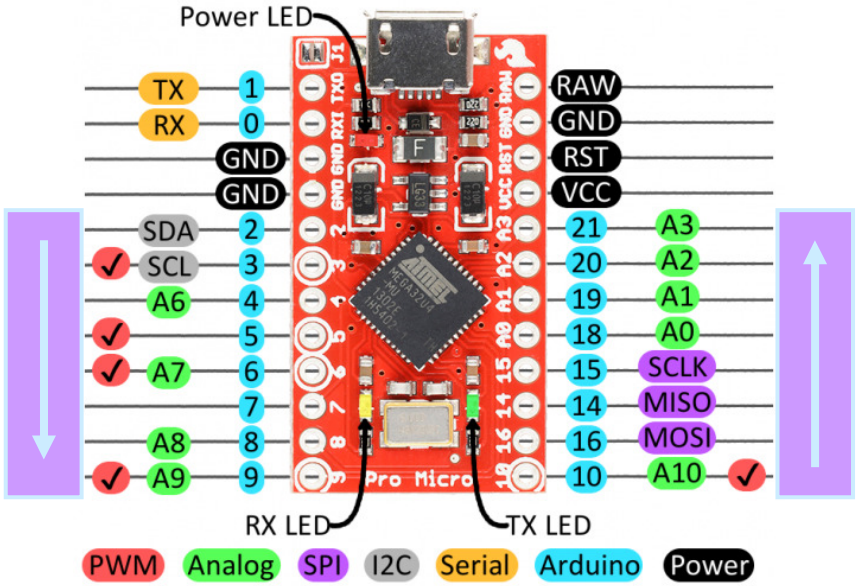
Automatisch scrollen | Kein Zeilenende | 115200 baud

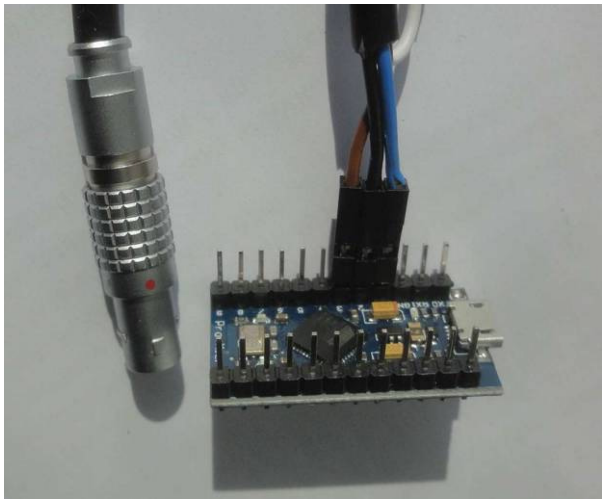
Signalgenerator

- Signalgenerator mit Arduino Micro 5V 16MHz
- generiert 16 binär geteilte Werte
- Auf den Pins erscheinen die Frequenzen

Pin	Bit	Freq	Zeit
2	0	2048 Hz	(!Grenze!)
3	1	1024 Hz	
4	2	512 Hz	
5	3	256 Hz	
6	4	128 Hz	
7	5	64 Hz	
8	6	32 Hz	
9	7	16 Hz	
10	8	8 Hz	
16	9	4 Hz	
14	10	2 Hz	
15	11	1 Hz	== 1 sec
18	12		2 sec
19	13		4 sec
20	14		8 sec
21	15		16 sec

- RXLED liegt beim Micro auf Pin 17
- Betriebsspannung über USB
- Frequenz 2048Hz ist stark verrauscht, ungenau (!Grenze!)





USB-HID Maus

Funktion:

- Ein proMicro bekommt 3 Byte von einer Software-UART*
- konvertiert diese in Steuerbefehle für eine USB-HID-Maus
- proMicro immitiert eine Maus, die über die Soft-UART gesteuert wird

Die drei Byte der UART rxbuffer[3] = (dx, dy, cmd) sind

- dx: Verschiebung in x-Richtung
- dx: Verschiebung in x-Richtung
- cmd: Mausklick:
  - 'l' left click
  - 'r' right click
  - 'd' double click left

b.w.

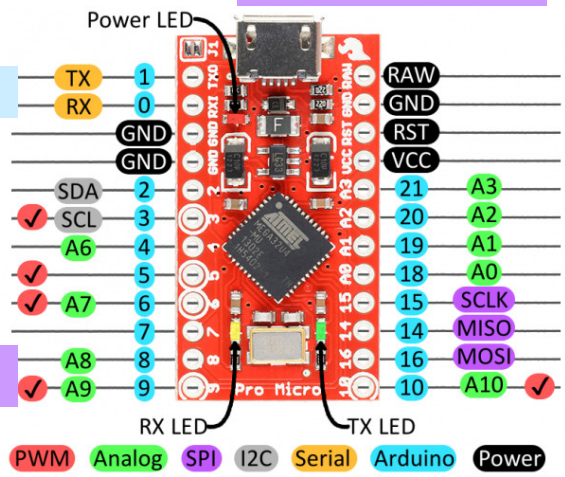
*UART: Universal Asynchronous Receiver and Transmitter (Basis der RS232)

USB-HID Maus

Output: USB-HID

Hard-UART/USB: n.c.

Input: Soft-UART




```
#include <SoftwareSerial.h>
// SoftwareSerial ist eine zusätzliche UART (in Software realisiert)
// ATmega32U4 hat nur eine Hard-UART am USB-VCP

SoftwareSerial SoftUart(9,8);     // Anlegen meiner SoftUart
// Arduino-proMicro Pins für Soft-UART (Rx, Tx)

// globale
int8_t dx, dy;
char cmd;
char rxbuffer[3];    // (dx, dy, cmd) from UART

void setup()
{                              // mehr als 57600 geht nicht (!Grenze!)
  Serial.begin(57600);     // USB-VCP communication
  SoftUart.begin(57600); // Soft-UART
  Mouse.begin();           // initialize USB-HID mouse control
}
```

```
void loop() {
if (SoftUart.available() == 3) {   // empfangene Zeichen
    SoftUart.readBytes(rxbuffer, 3);   // buffer, lenght, Lesefolge: 0...2

  // Protokoll
  dx  = rxbuffer[0];    // delta x
  dy  = rxbuffer[1];    // delta y
  cmd = rxbuffer[2];    // Maustastendruck (r, l, d)

  // Mouse shift
  Mouse.move(dx,dy);
  TXLED1; RXLED1;  // LEDs aus

  // Mouse keys
  switch(cmd) {
    case 'r':      // right klick - open menue
      Mouse.click(MOUSE_RIGHT);
      TXLED0; RXLED1;      break;

    case 'l':      // left klick - highlight something
      Mouse.click(MOUSE_LEFT);
      TXLED1; RXLED0;      break;

    case 'd':     // double klick
      Mouse.click(MOUSE_LEFT);
      Mouse.click(MOUSE_LEFT);
      TXLED0; RXLED0;      break;

    default:      // LEDs aus
      TXLED1; RXLED1;      break;
  }
  SoftUart.print(rxbuffer);    // zur UART zurück
 }
}
```