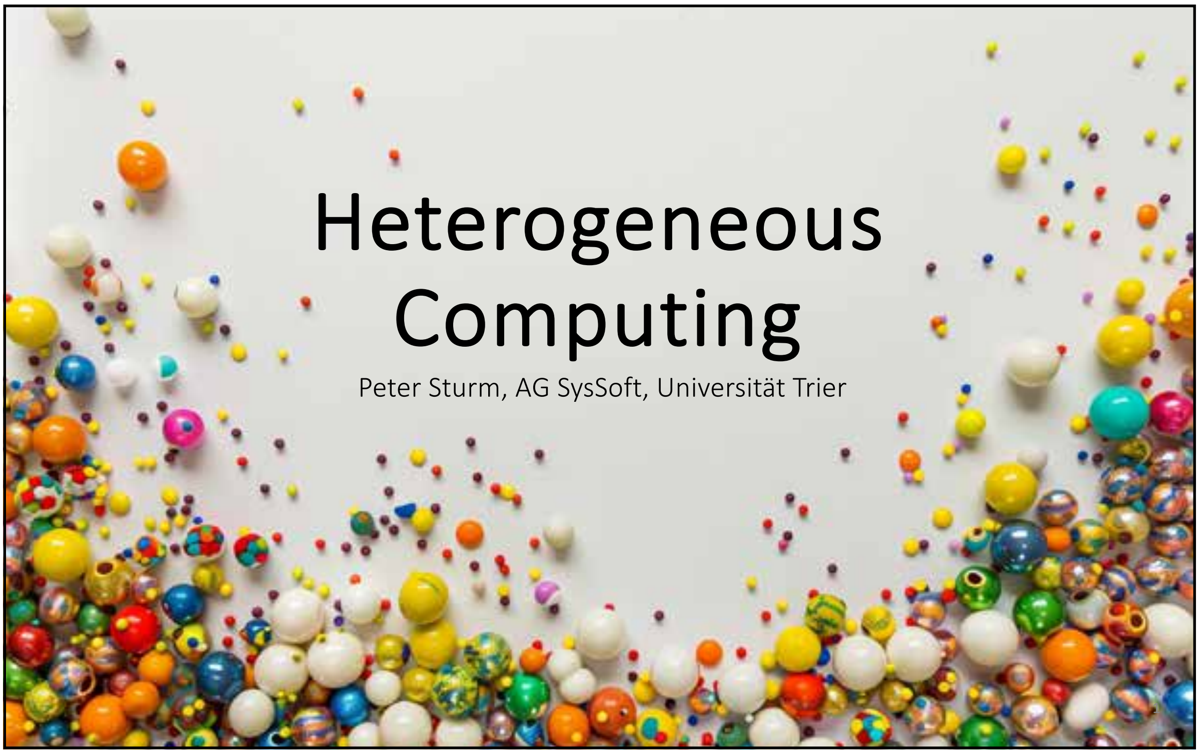
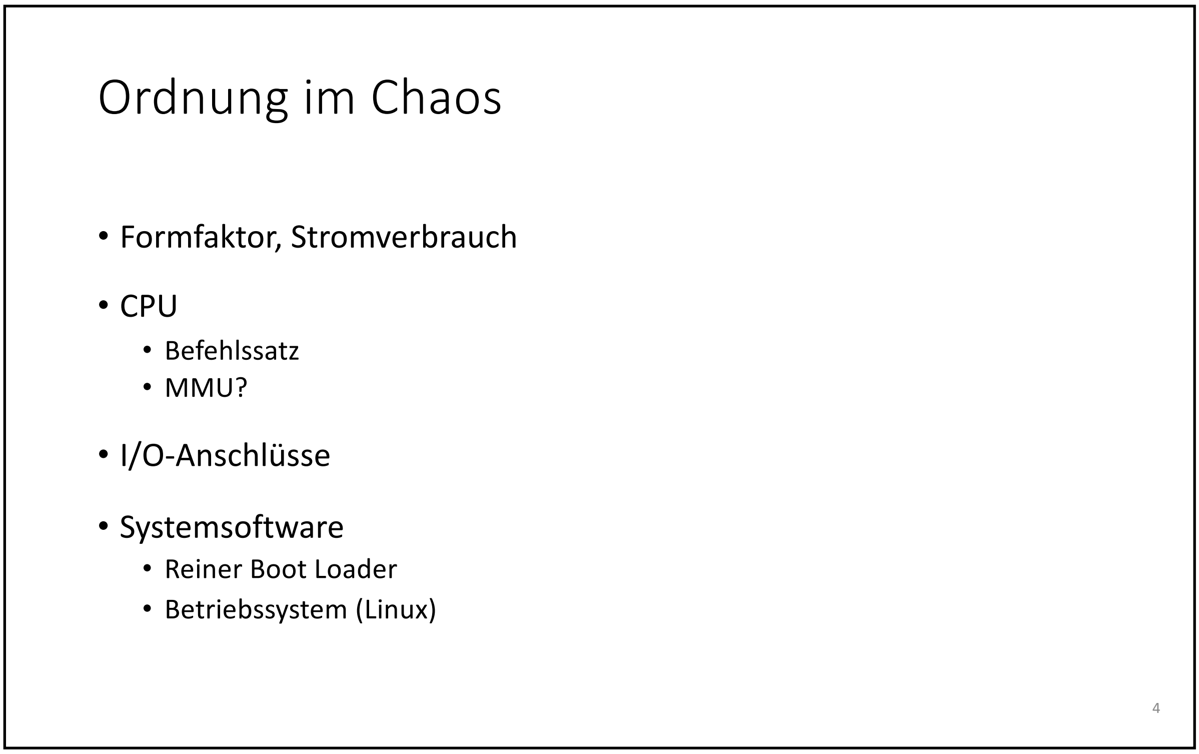


Heterogeneous Computing

Peter Sturm, AG SysSoft, Universität Trier

2. Arduino & Co



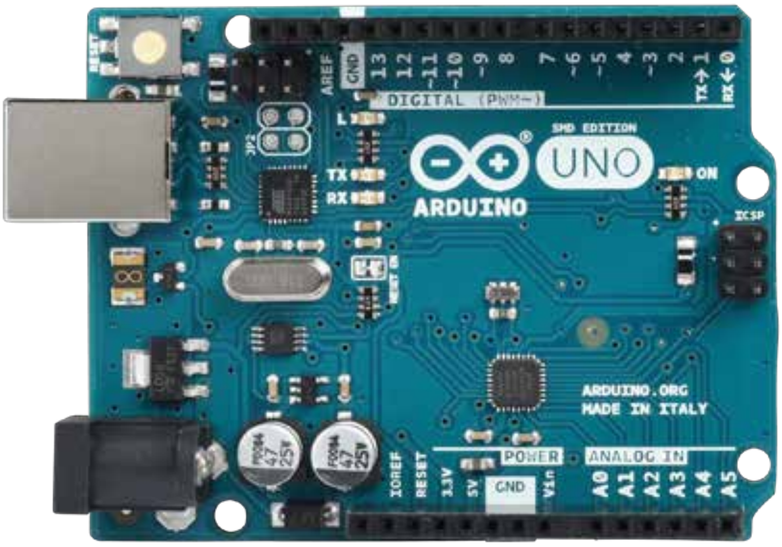
Ordnung im Chaos

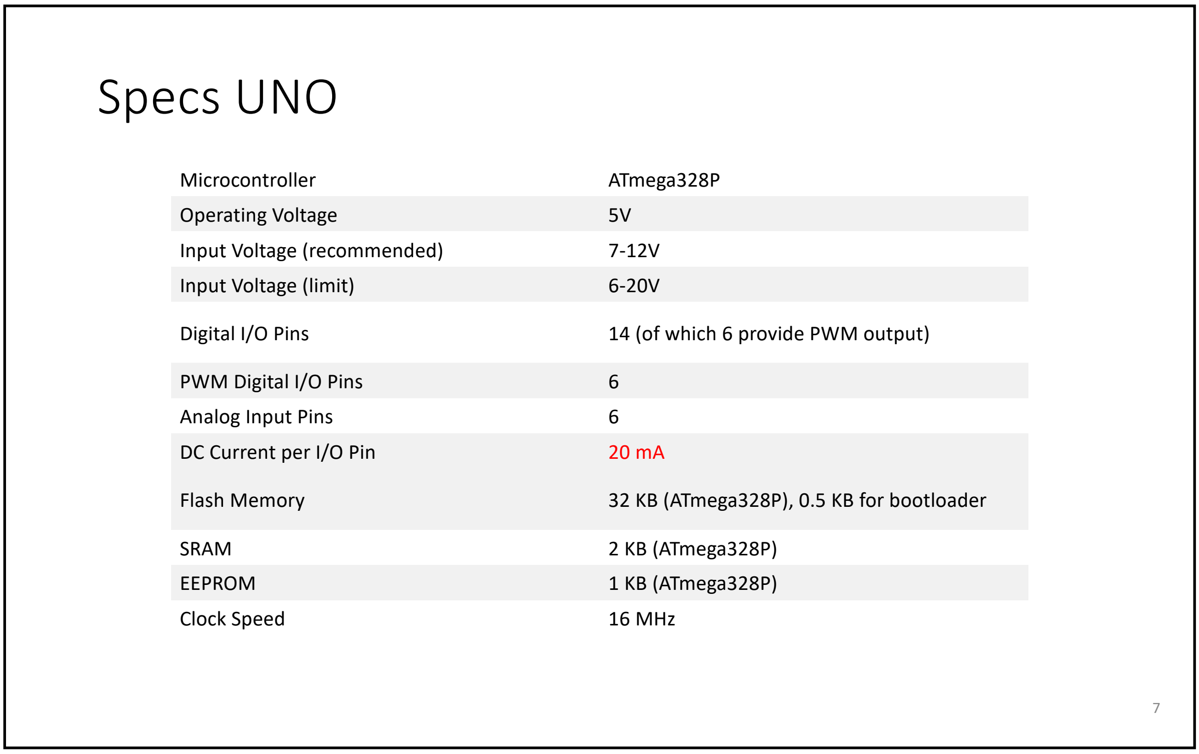
- Formfaktor, Stromverbrauch
- CPU
  - Befehlssatz
  - MMU?
- I/O-Anschlüsse
- Systemsoftware
  - Reiner Boot Loader
  - Betriebssystem (Linux)

4

2.1 Nur Boot Loader

Arduino Uno

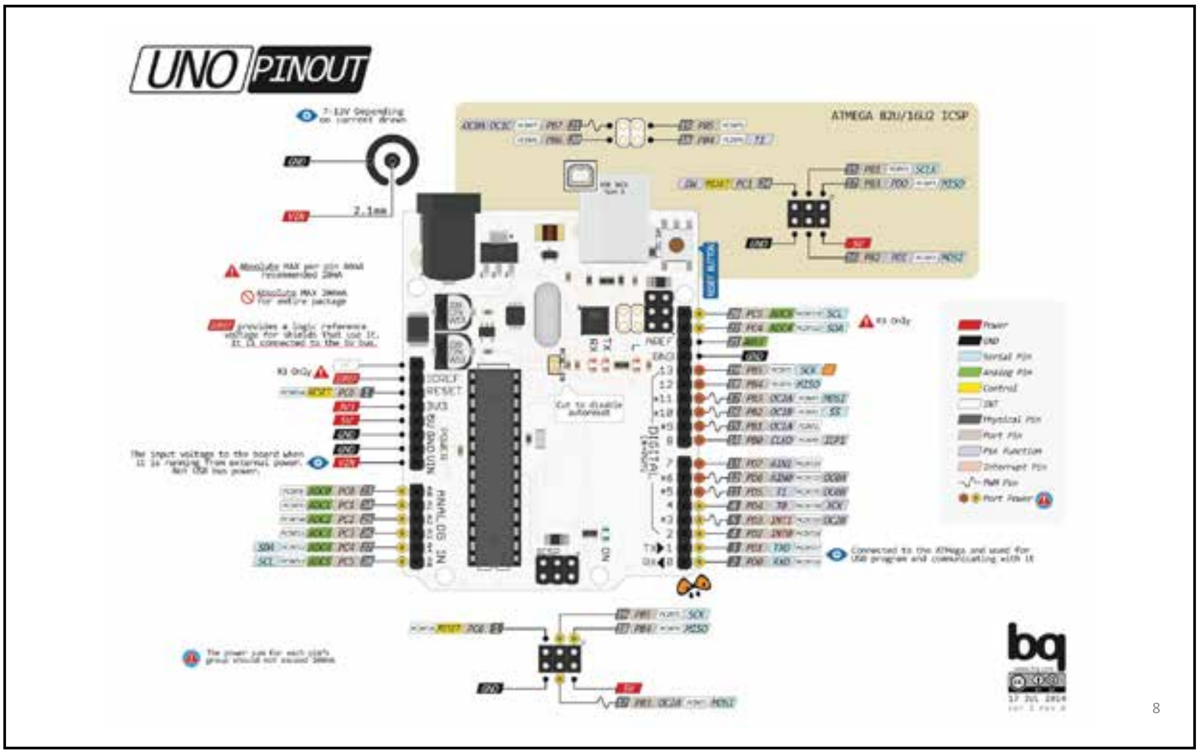




Specs UNO

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
Flash Memory	32 KB (ATmega328P), 0.5 KB for bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz

7

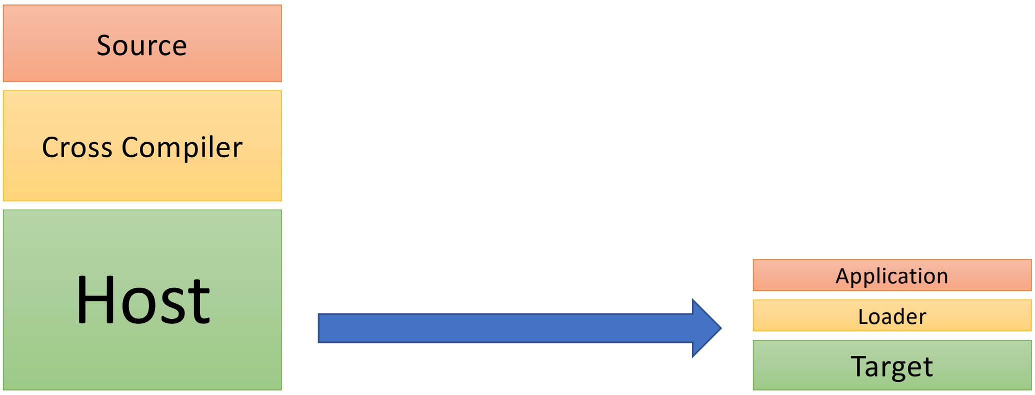


8

Ein- und Ausgabe

- Digital
  - 0 und 1 = 0V und Vcc
- Analoger Eingang
  - ADC (Analog Digital Converter)
  - Spannung zwischen 0V und einer Referenzspannung
  - Ausgabe uint
  - Präzision (10 Bit und mehr)
  - 10 Bit und Vref=5V: Auflösung 5/1024 = 4.88mV
- Analoger Ausgang
  - DAC (Digital Analog Converter)
  - PWM (Pulse Width Modulation)

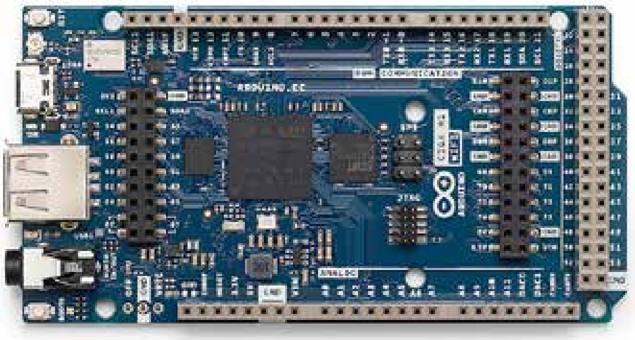
Cross Compile



Arduino Uno: ”Hello World”

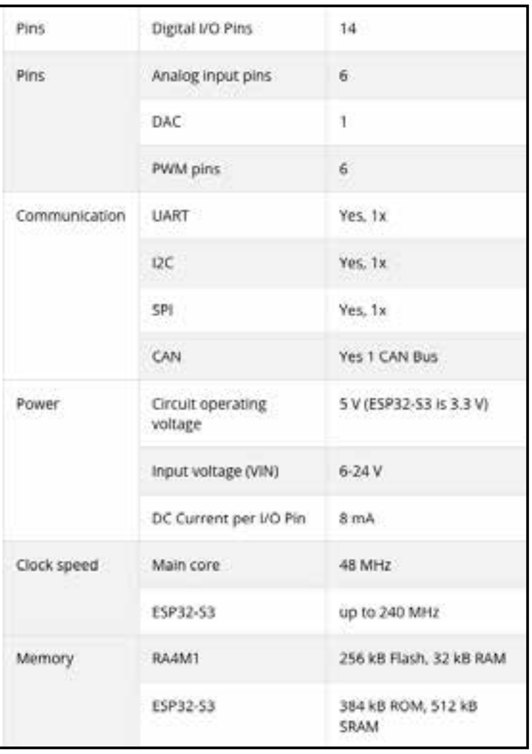
Blinkende LED

Arduino



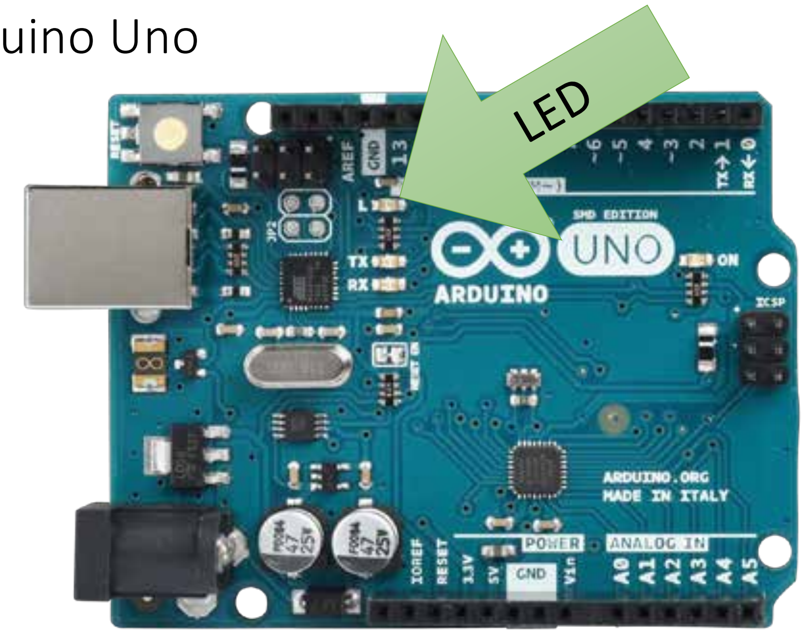
Arduino Nano Matter

Arduino Uno R4 Wifi



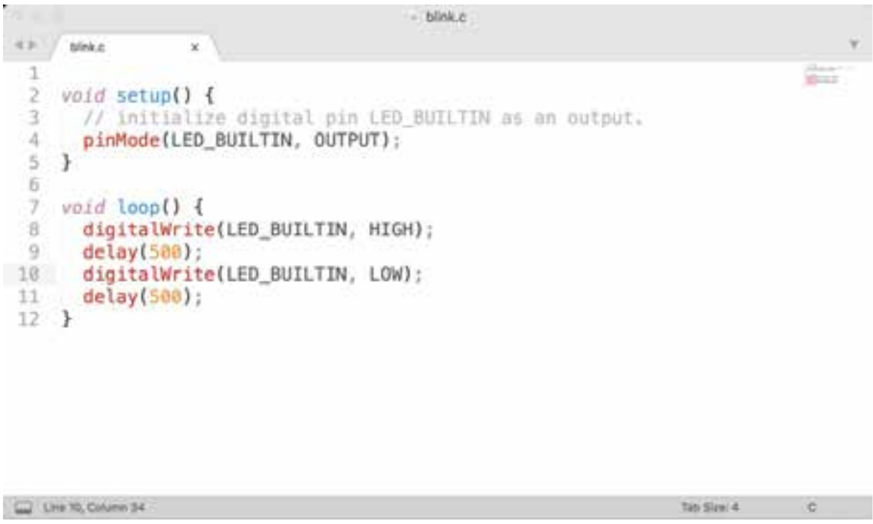
Pins	Digital I/O Pins	14
Pins	Analog input pins	6
	DAC	1
	PWM pins	6
Communication	UART	Yes, 1x
	I2C	Yes, 1x
	SPI	Yes, 1x
	CAN	Yes 1 CAN Bus
Power	Circuit operating voltage	5 V (ESP32-S3 is 3.3 V)
	Input voltage (VIN)	6-24 V
	DC Current per I/O Pin	8 mA
Clock speed	Main core	48 MHz
	ESP32-S3	up to 240 MHz
Memory	RA4M1	256 kB Flash, 32 kB RAM
	ESP32-S3	384 kB ROM, 512 kB SRAM

Arduino Uno



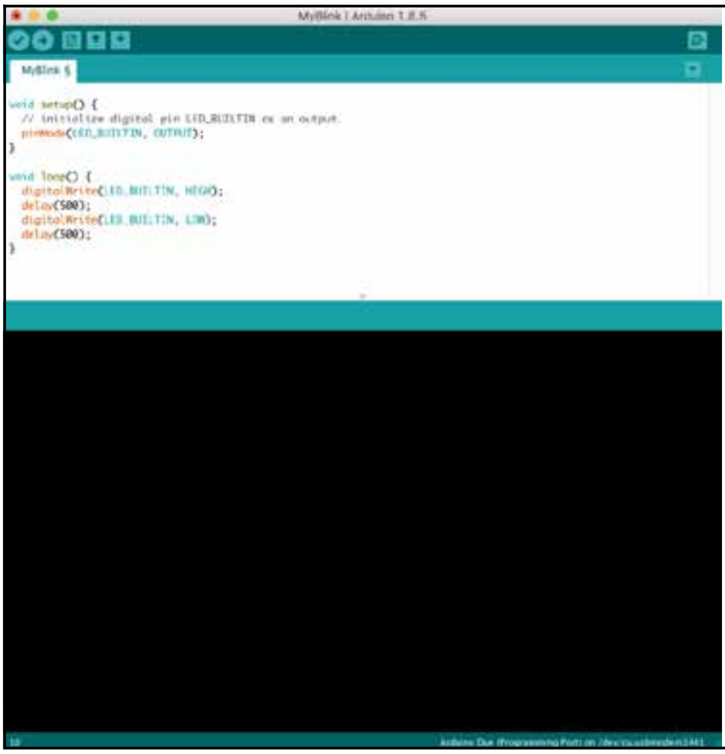
14

Sketch



```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
}
```

15

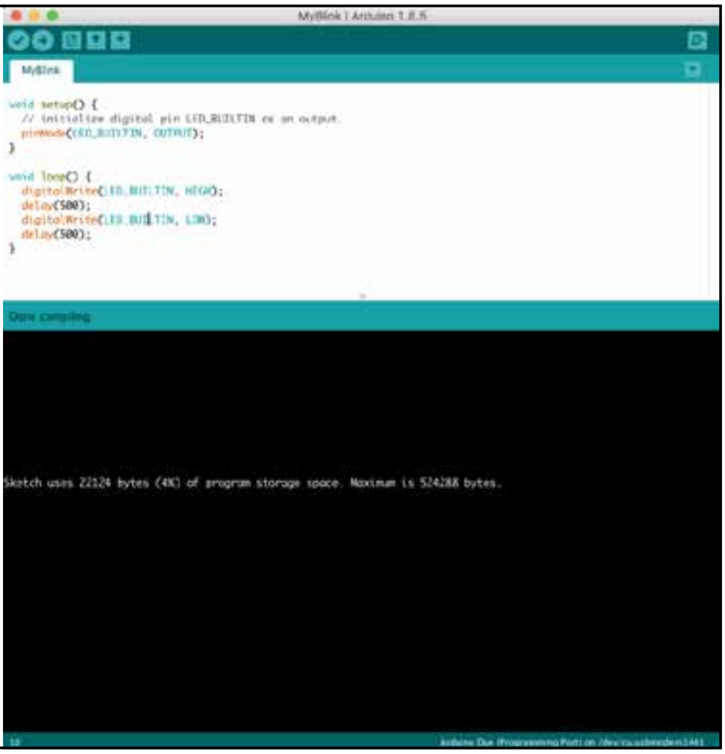


```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
}
```

Arduino IDE

16

Compile

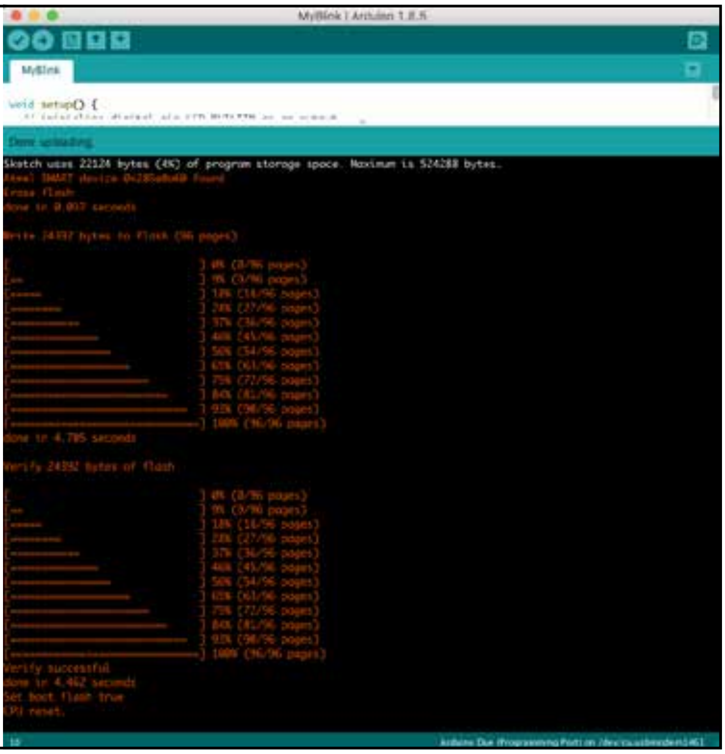


```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
}
```

Done compiling

```
Sketch uses 22124 bytes (4%) of program storage space. Maximum is 524288 bytes.
```

Upload

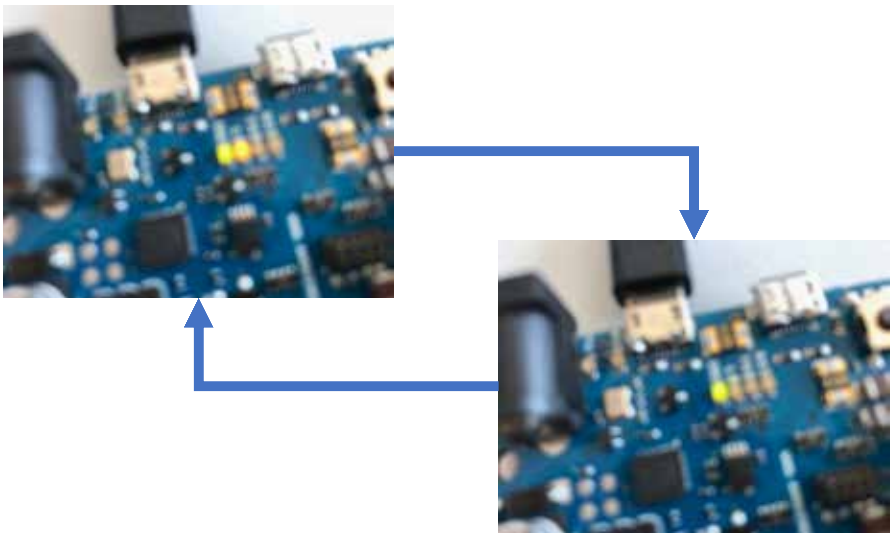


```
void setup() {
```

Done uploading

```
Sketch uses 22124 bytes (4%) of program storage space. Maximum is 524288 bytes.
Atmel SMART device 0x285e0a60 found
Erase flash
done in 0.037 seconds

Write 24332 bytes to flash (96 pages)

[                              ] 0% (0/96 pages)
[==                            ] 9% (9/96 pages)
[=====                         ] 18% (18/96 pages)
[========                      ] 28% (27/96 pages)
[===========                   ] 37% (36/96 pages)
[==============                ] 46% (45/96 pages)
[================              ] 56% (54/96 pages)
[===================           ] 65% (63/96 pages)
[======================        ] 75% (72/96 pages)
[=========================     ] 84% (81/96 pages)
[============================  ] 93% (90/96 pages)
[==============================] 100% (96/96 pages)
done in 4.705 seconds

Verify 24332 bytes of flash

[                              ] 0% (0/96 pages)
[==                            ] 9% (9/96 pages)
[=====                         ] 18% (18/96 pages)
[========                      ] 28% (27/96 pages)
[===========                   ] 37% (36/96 pages)
[==============                ] 46% (45/96 pages)
[================              ] 56% (54/96 pages)
[===================           ] 65% (63/96 pages)
[======================        ] 75% (72/96 pages)
[=========================     ] 84% (81/96 pages)
[============================  ] 93% (90/96 pages)
[==============================] 100% (96/96 pages)
Verify successful
done in 4.462 seconds
Set boot flash true
CPU reset.
```

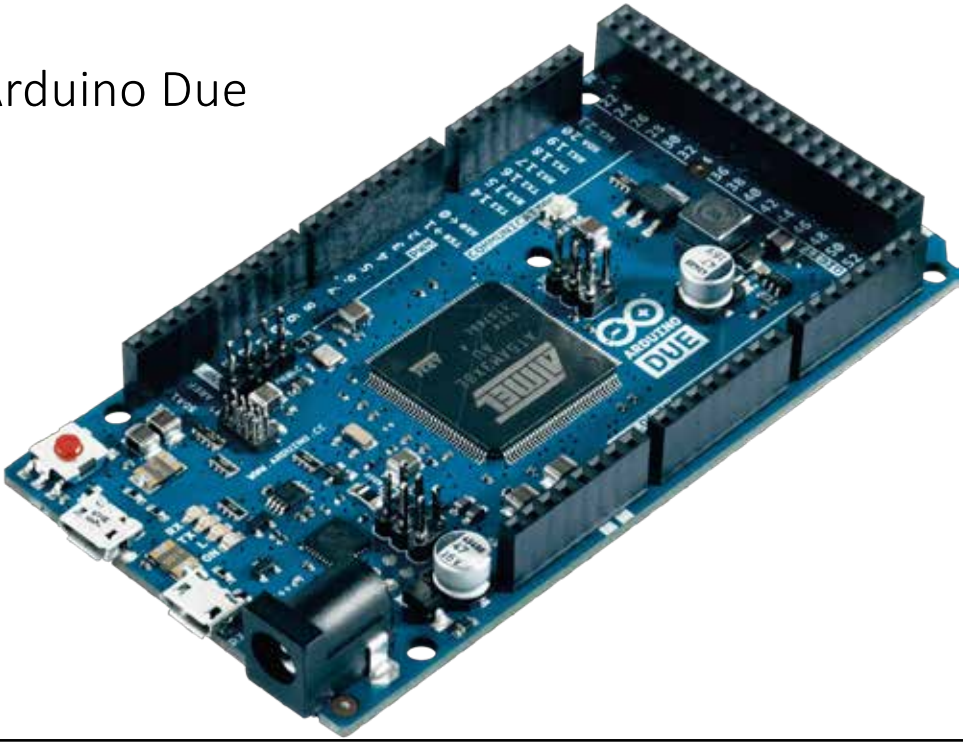
Es läuft ☺



Arduino Due

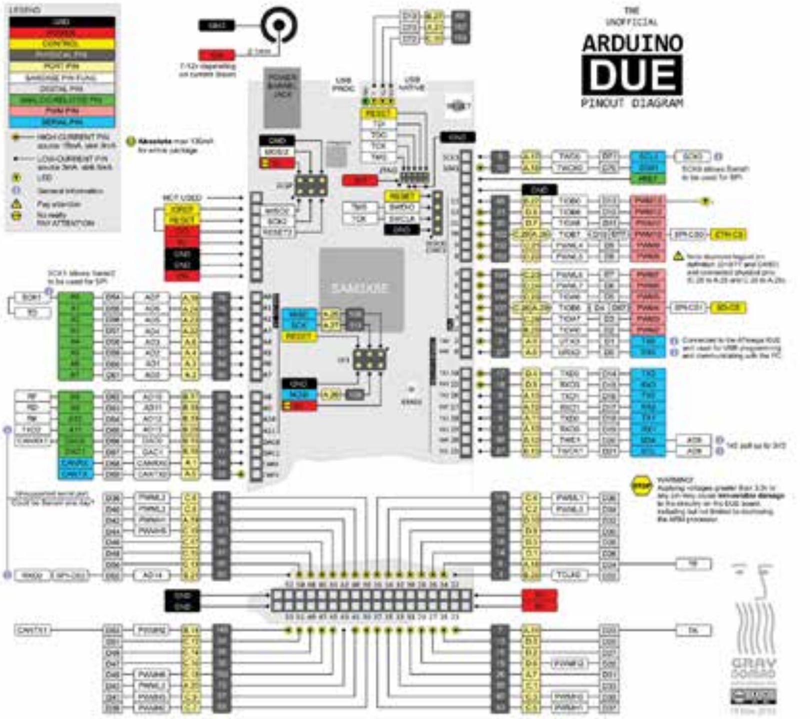
Morsecode

Arduino Due



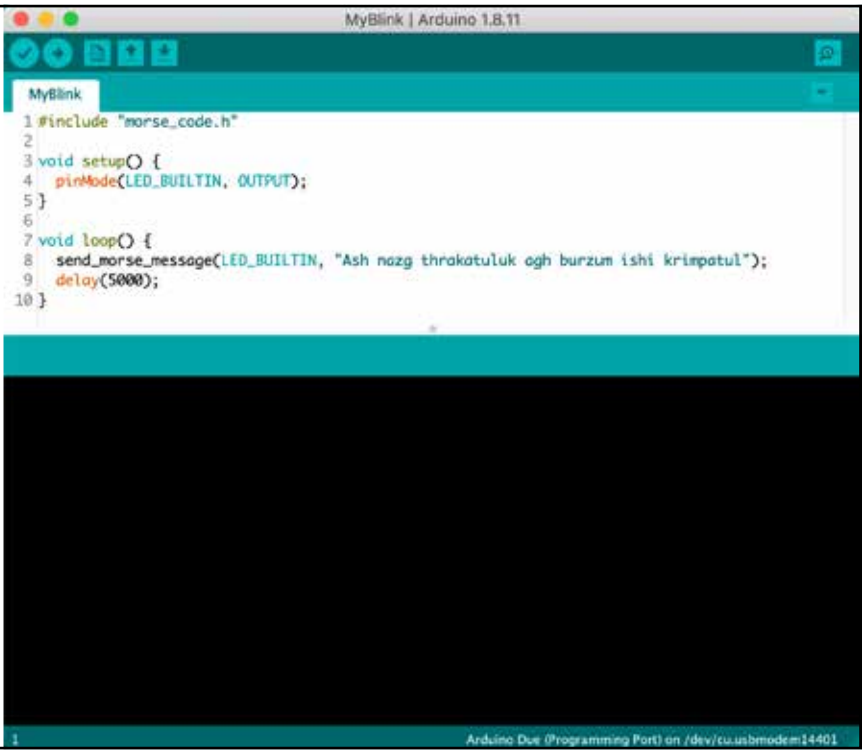
Specs DUE

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz



23

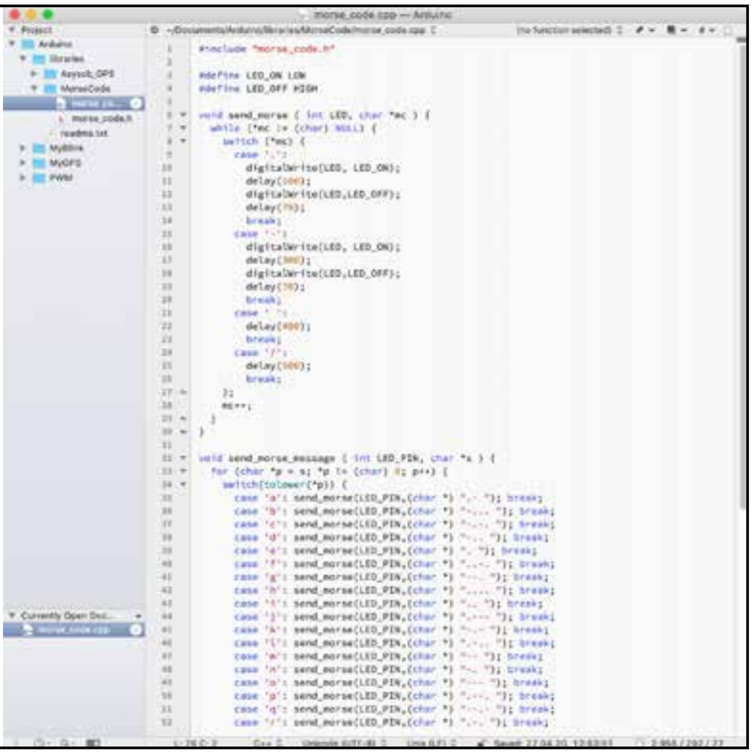
Sketch



```
#include "morse_code.h"

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  send_morse_message(LED_BUILTIN, "Ash nazg thrakatuluk agh burzum ishi krimpatul");
  delay(5000);
}
```

1 Arduino Due (Programming Port) on /dev/cu.usbmodem14401

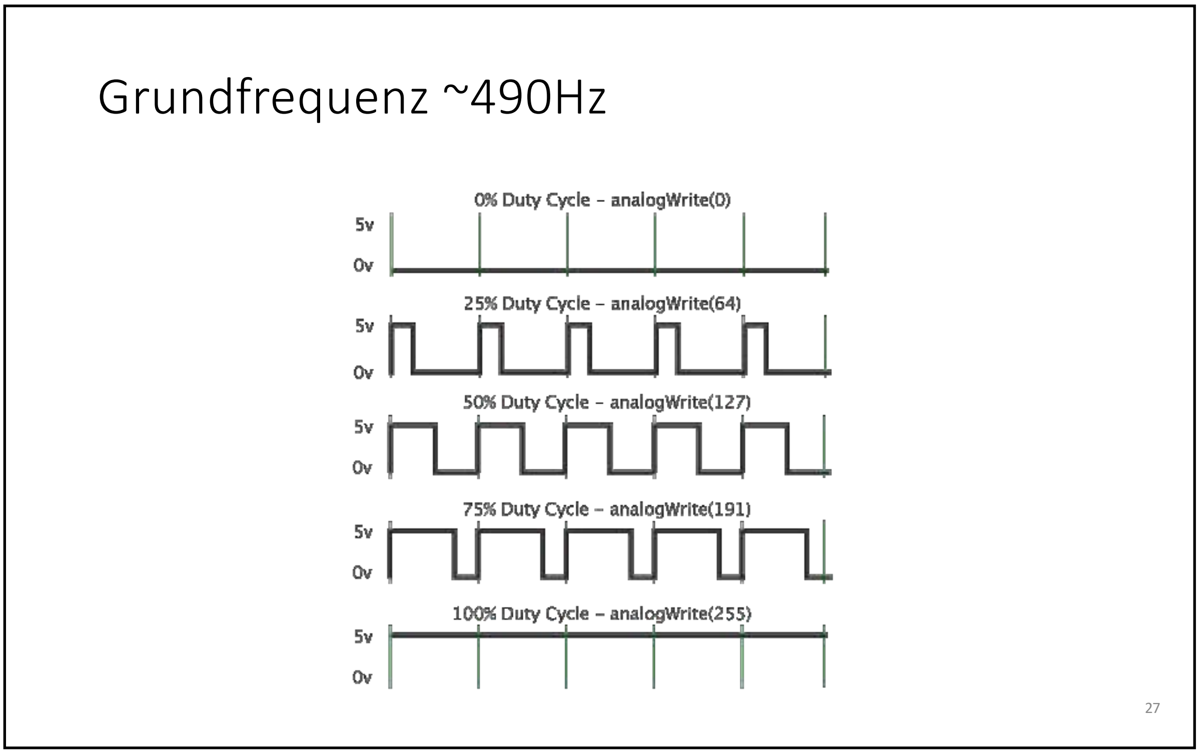
Library



```
#include "morse_code.h"

#define LED_ON LOW
#define LED_OFF HIGH

void send_morse ( int LED, char *mc ) {
  while (*mc != (char) NULL) {
    switch (*mc) {
      case '.':
        digitalWrite(LED, LED_ON);
        delay(100);
        digitalWrite(LED,LED_OFF);
        delay(75);
        break;
      case '-':
        digitalWrite(LED, LED_ON);
        delay(300);
        digitalWrite(LED,LED_OFF);
        delay(70);
        break;
      case ' ':
        delay(400);
        break;
      case '/':
        delay(500);
        break;
    };
    mc++;
  }
}

void send_morse_message ( int LED_PIN, char *s ) {
  for (char *p = s; *p != (char) 0; p++) {
    switch(tolower(*p)) {
      case 'a': send_morse(LED_PIN,(char *) ".- "); break;
      case 'b': send_morse(LED_PIN,(char *) "-... "); break;
      case 'c': send_morse(LED_PIN,(char *) "-.-. "); break;
      case 'd': send_morse(LED_PIN,(char *) "-.. "); break;
      case 'e': send_morse(LED_PIN,(char *) ". "); break;
      case 'f': send_morse(LED_PIN,(char *) "..-. "); break;
      case 'g': send_morse(LED_PIN,(char *) "--. "); break;
      case 'h': send_morse(LED_PIN,(char *) ".... "); break;
      case 'i': send_morse(LED_PIN,(char *) ".. "); break;
      case 'j': send_morse(LED_PIN,(char *) ".--- "); break;
      case 'k': send_morse(LED_PIN,(char *) "-.- "); break;
      case 'l': send_morse(LED_PIN,(char *) ".-.. "); break;
      case 'm': send_morse(LED_PIN,(char *) "-- "); break;
      case 'n': send_morse(LED_PIN,(char *) "-. "); break;
      case 'o': send_morse(LED_PIN,(char *) "--- "); break;
      case 'p': send_morse(LED_PIN,(char *) ".--. "); break;
      case 'q': send_morse(LED_PIN,(char *) "--.- "); break;
      case 'r': send_morse(LED_PIN,(char *) ".-. "); break;
```

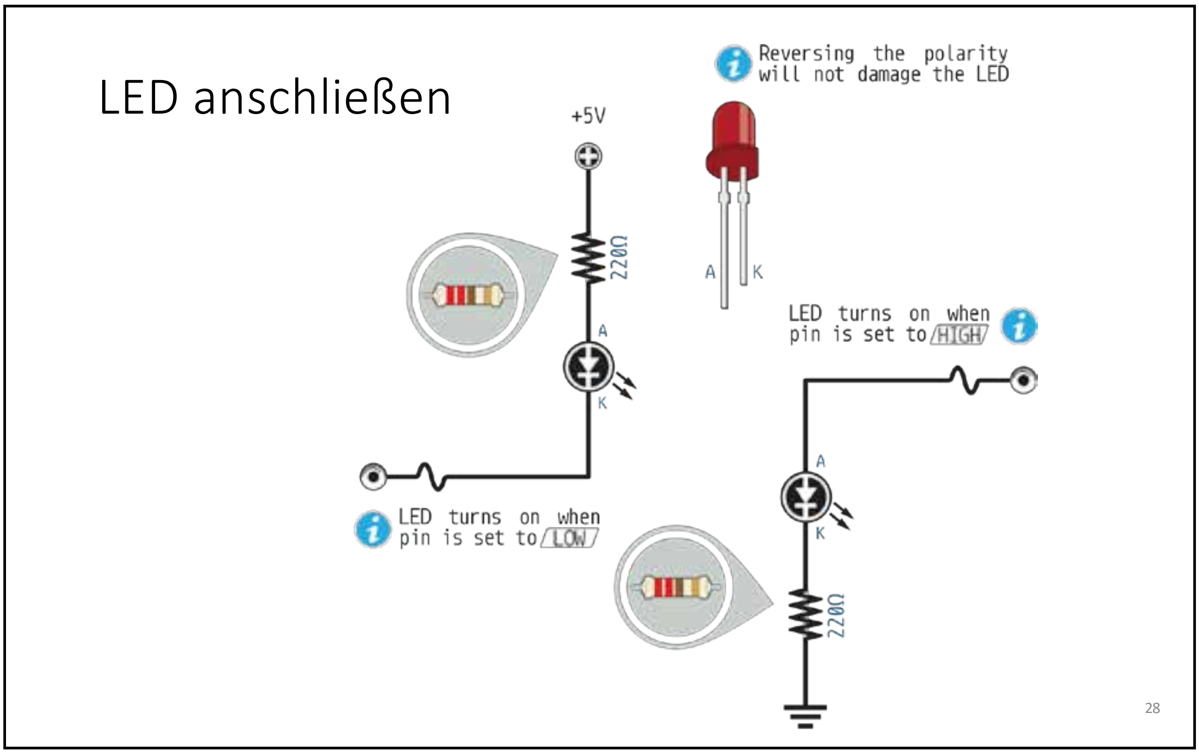
Pulsweitenmodulation (PWM)

26



Grundfrequenz ~490Hz

27



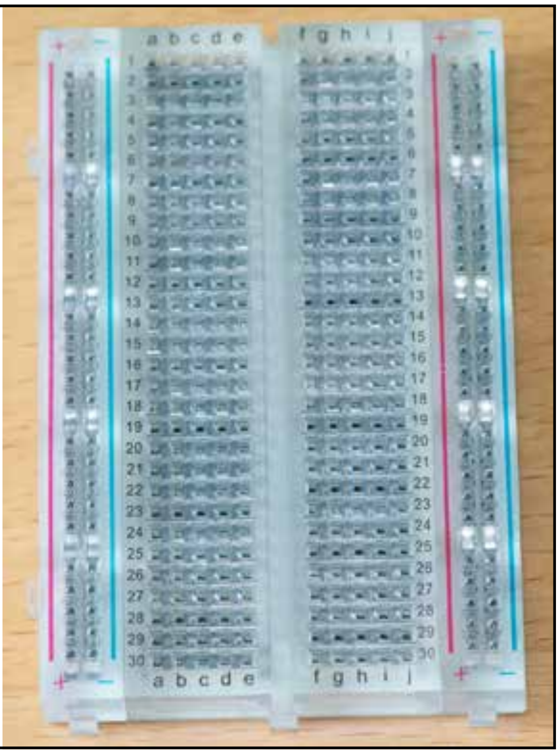
LED anschließen

Reversing the polarity will not damage the LED

LED turns on when pin is set to HIGH

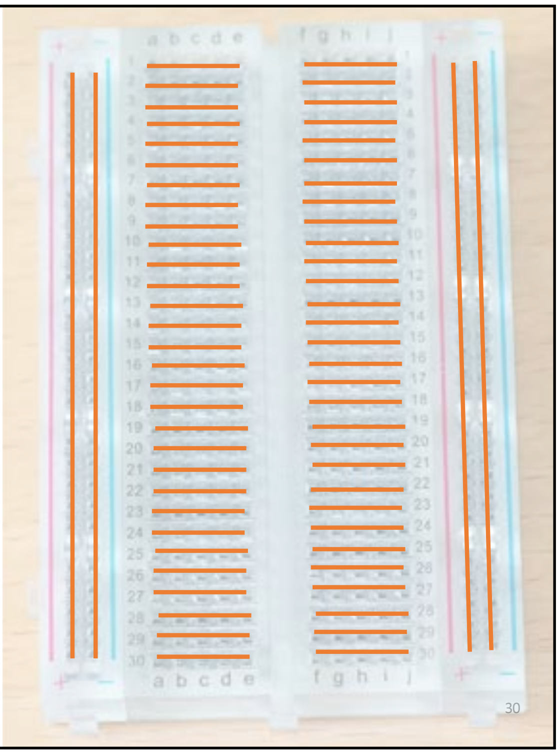
LED turns on when pin is set to LOW

28



Breadboard
Protoboard

Leer

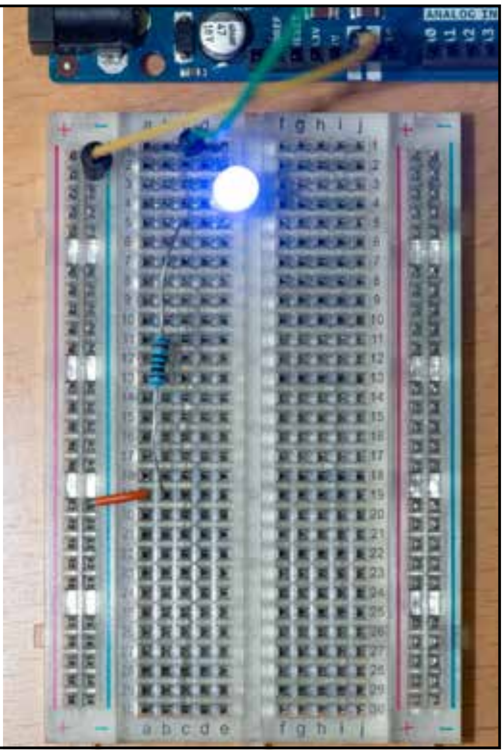


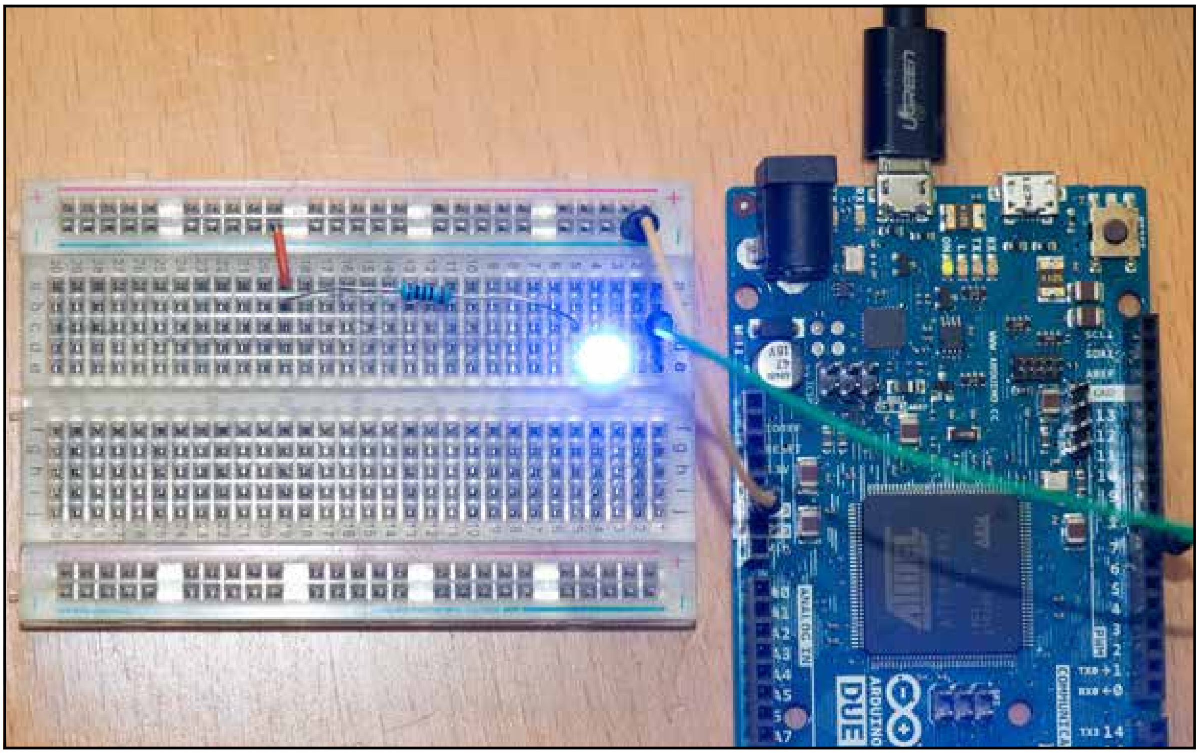
Breadboard
Protoboard

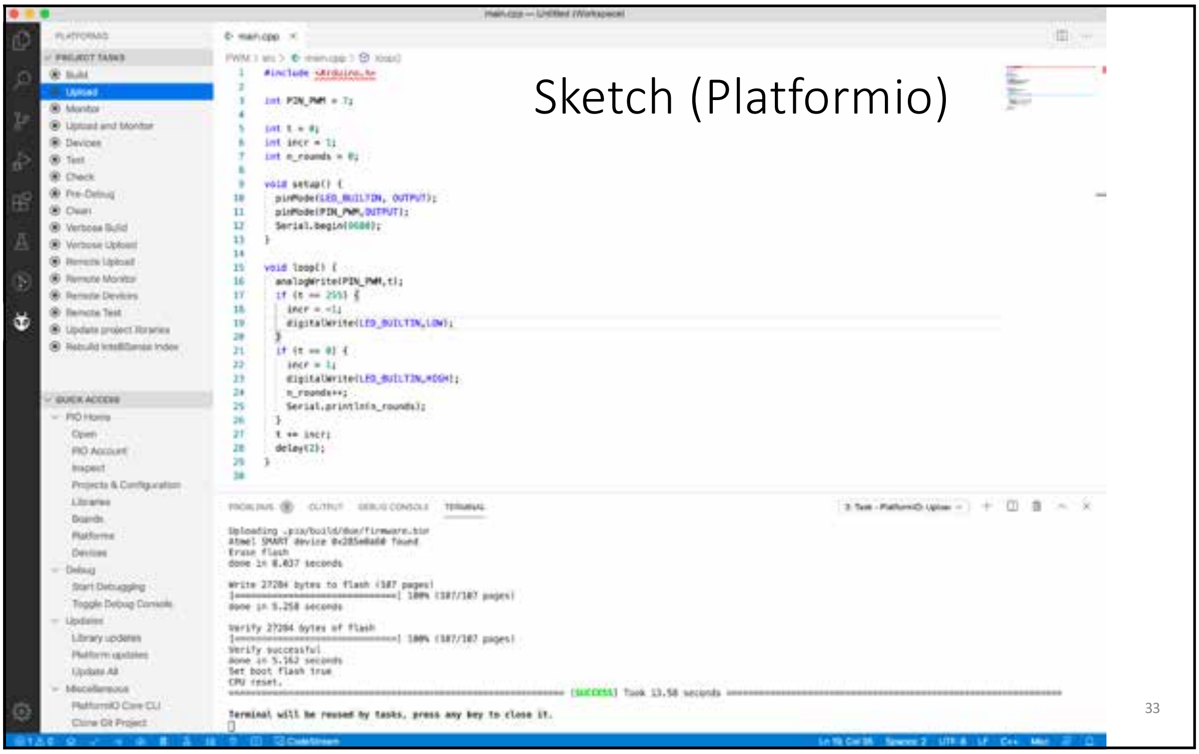
Leer

Breadboard
Protoboard

Mit LED



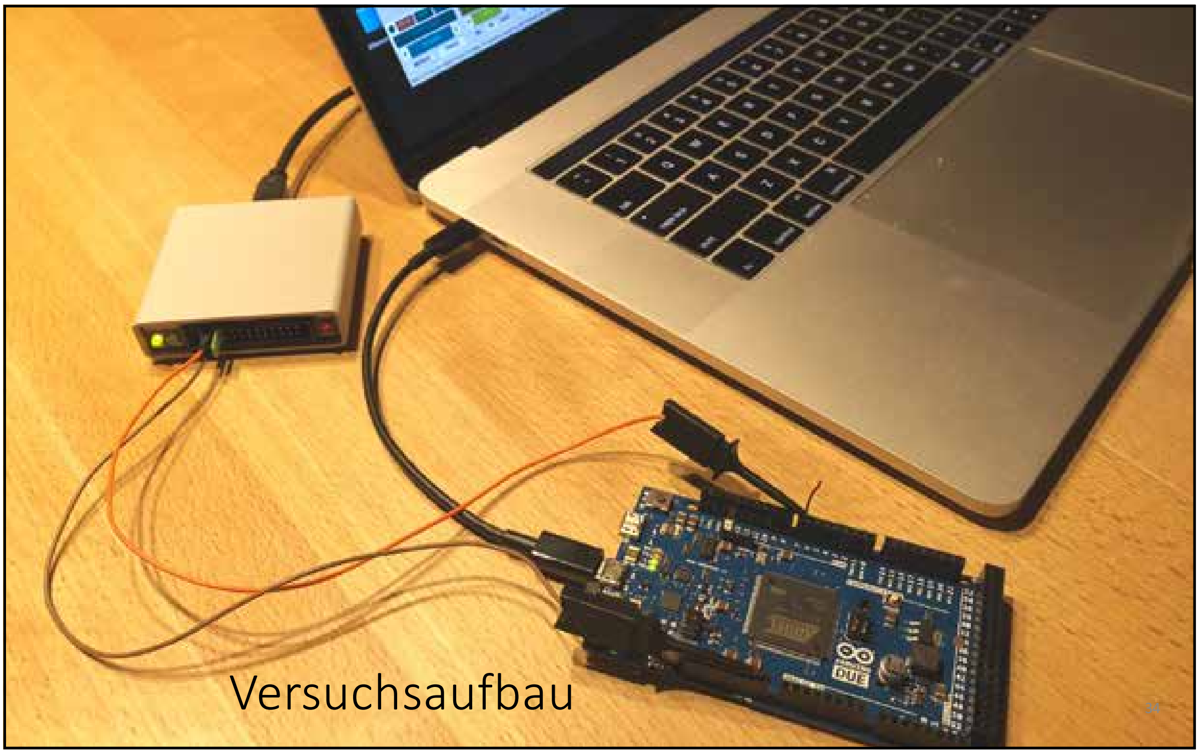




Sketch (Platformio)

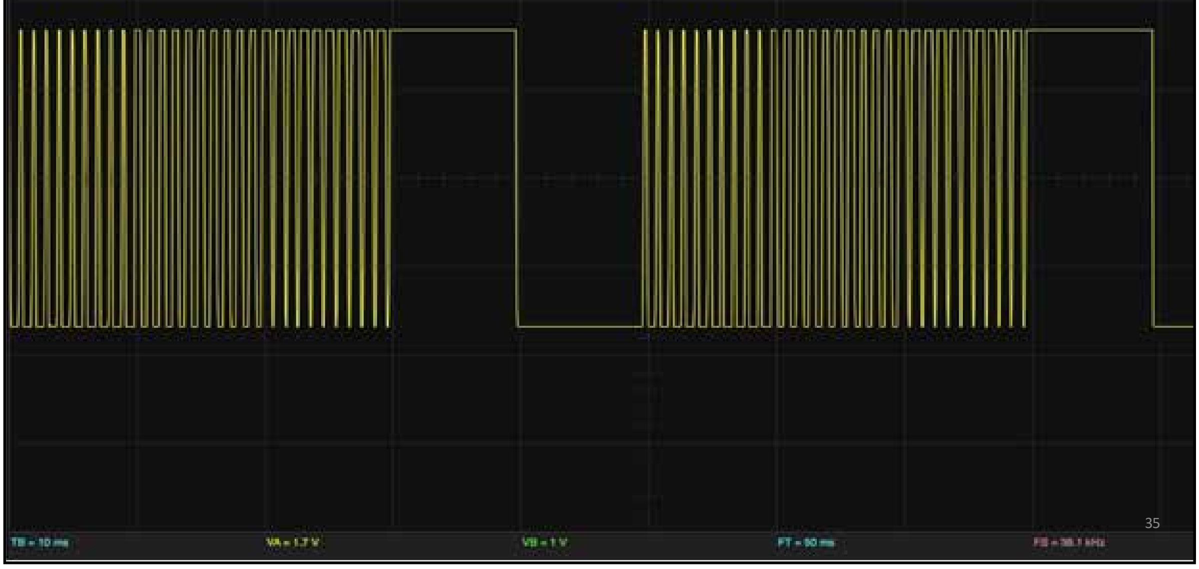
```
#include <Arduino.h>

int PIN_PWM = 7;

int t = 0;
int incr = 1;
int n_rounds = 0;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(PIN_PWM,OUTPUT);
  Serial.begin(9600);
}

void loop() {
  analogWrite(PIN_PWM,t);
  if (t == 255) {
    incr = -1;
    digitalWrite(LED_BUILTIN,LOW);
  }
  if (t == 0) {
    incr = 1;
    digitalWrite(LED_BUILTIN,HIGH);
    n_rounds++;
    Serial.println(n_rounds);
  }
  t += incr;
  delay(2);
}
```

```
Uploading .pio/build/due/firmware.bin
Atmel SMART device 0x285e0a60 found
Erase flash
done in 0.037 seconds

Write 27204 bytes to flash (107 pages)
[==============================] 100% (107/107 pages)
done in 5.258 seconds

Verify 27204 bytes of flash
[==============================] 100% (107/107 pages)
Verify successful
done in 5.162 seconds
Set boot flash true
CPU reset.
============================================================ [SUCCESS] Took 13.58 seconds ============================================================

Terminal will be reused by tasks, press any key to close it.
```

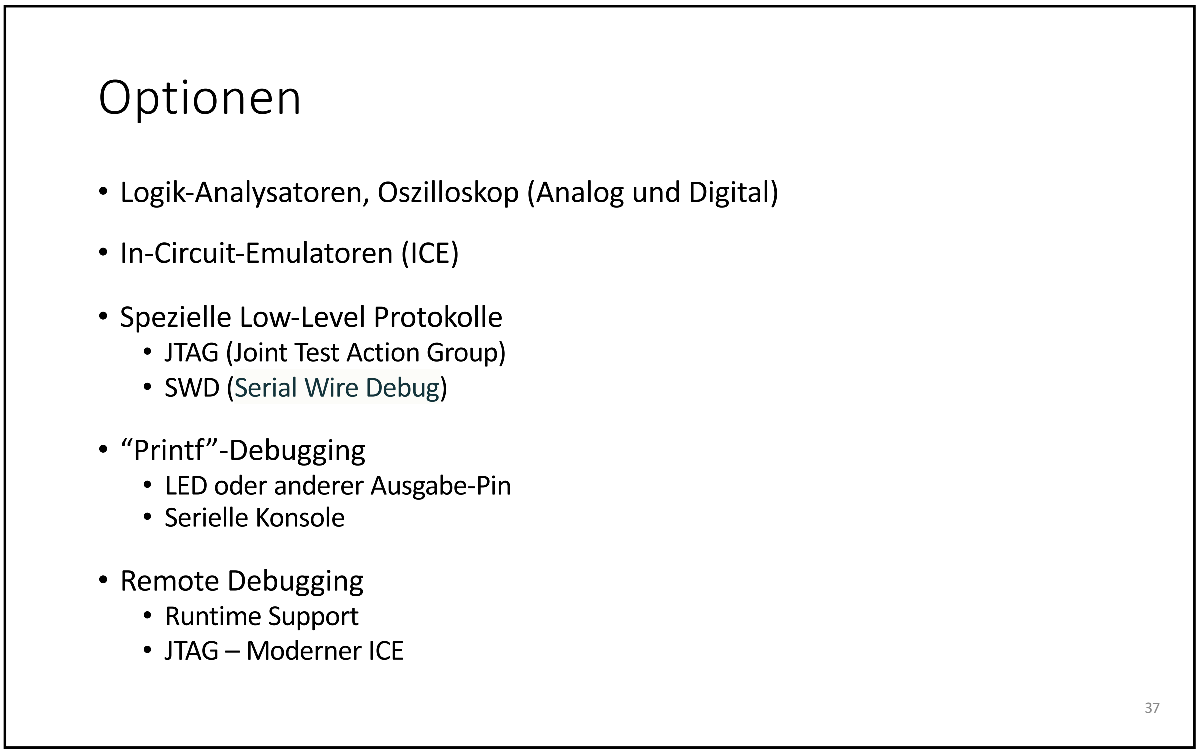


Versuchsaufbau

Logikanalysator



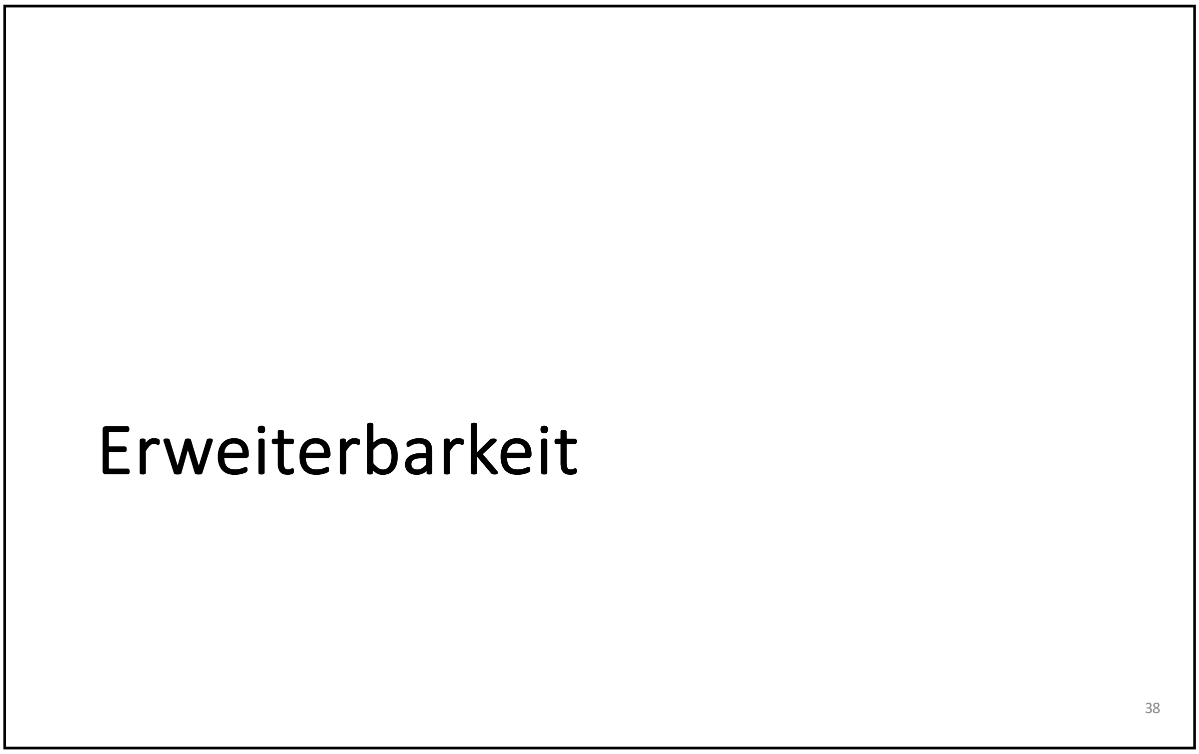
Debugging



Optionen

- Logik-Analysatoren, Oszilloskop (Analog und Digital)
- In-Circuit-Emulatoren (ICE)
- Spezielle Low-Level Protokolle
  - JTAG (Joint Test Action Group)
  - SWD (Serial Wire Debug)
- “Printf”-Debugging
  - LED oder anderer Ausgabe-Pin
  - Serielle Konsole
- Remote Debugging
  - Runtime Support
  - JTAG – Moderner ICE

37

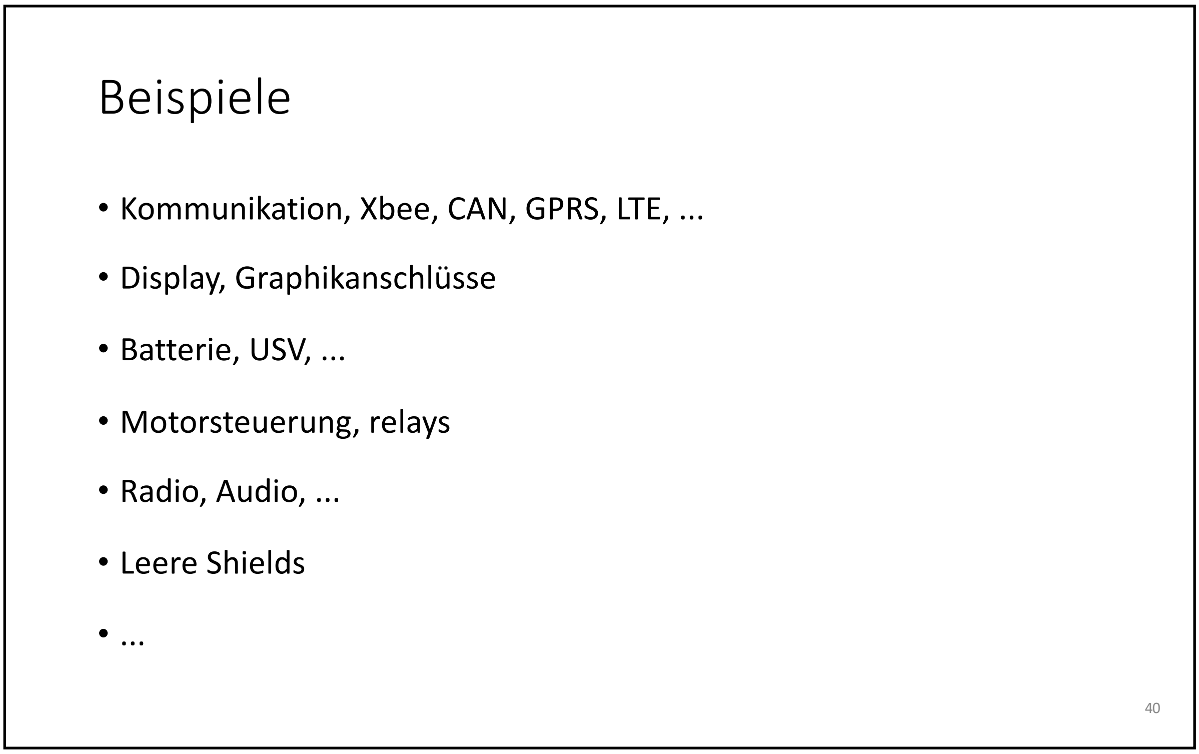


Erweiterbarkeit

38

Speziell Arduino: Shields

39



Beispiele

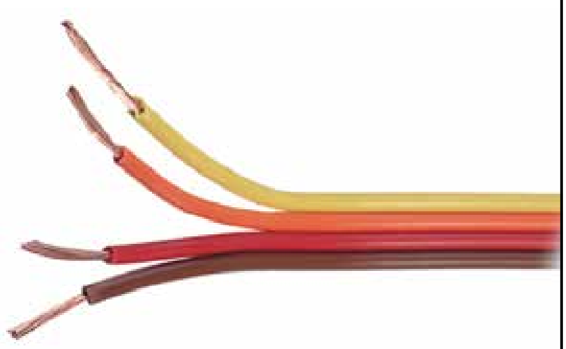
- Kommunikation, Xbee, CAN, GPRS, LTE, ...
- Display, Graphikanschlüsse
- Batterie, USV, ...
- Motorsteuerung, relays
- Radio, Audio, ...
- Leere Shields
- ...

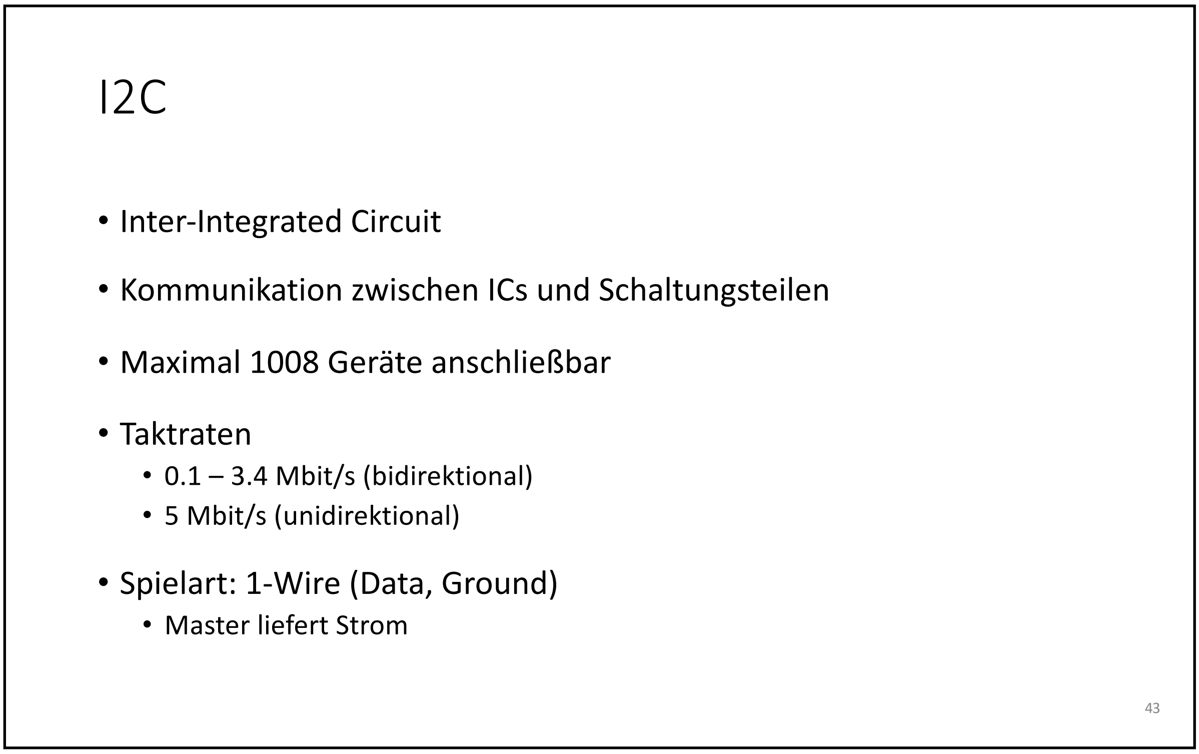
40

Kommunikation zwischen Chips

Aufwand minimieren

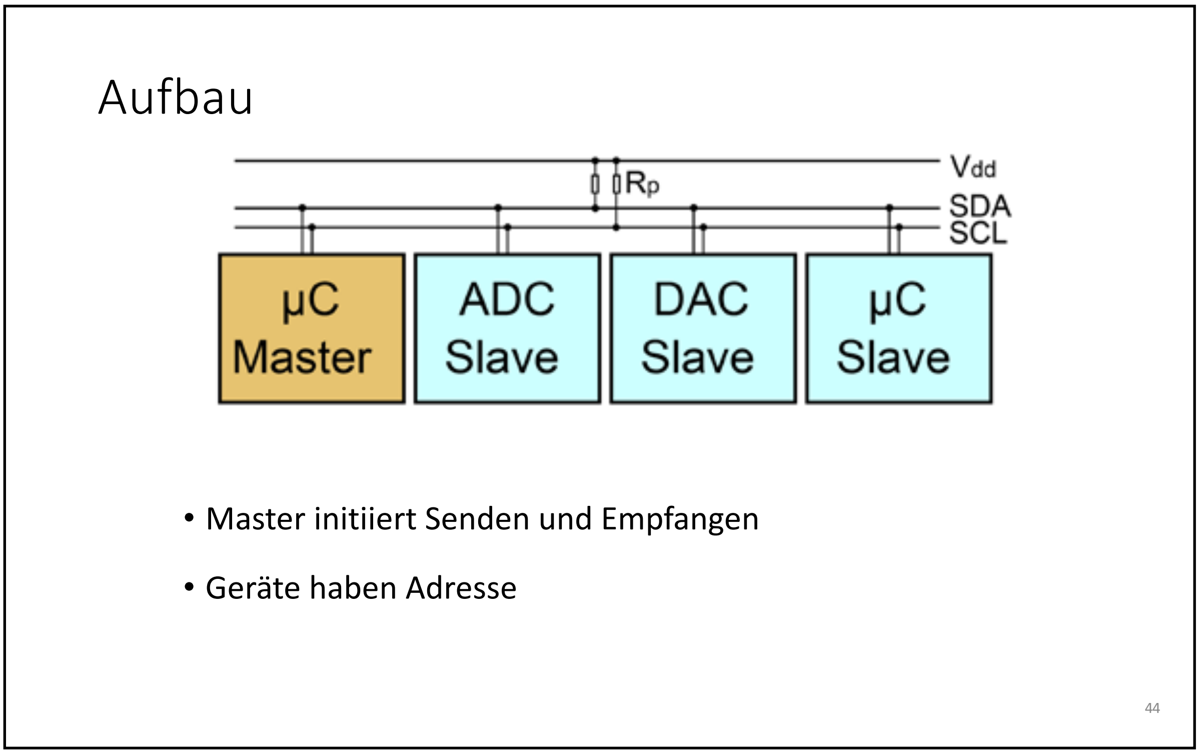
- Geringe Leitungszahl
- Serielle Ansätze
- Simple Realisierung
    - Also kein Ethernet o.ä.





I2C

- Inter-Integrated Circuit
- Kommunikation zwischen ICs und Schaltungsteilen
- Maximal 1008 Geräte anschließbar
- Taktraten
  - 0.1 – 3.4 Mbit/s (bidirektional)
  - 5 Mbit/s (unidirektional)
- Spielart: 1-Wire (Data, Ground)
  - Master liefert Strom



Aufbau

- Master initiiert Senden und Empfangen
- Geräte haben Adresse

Kommunikation

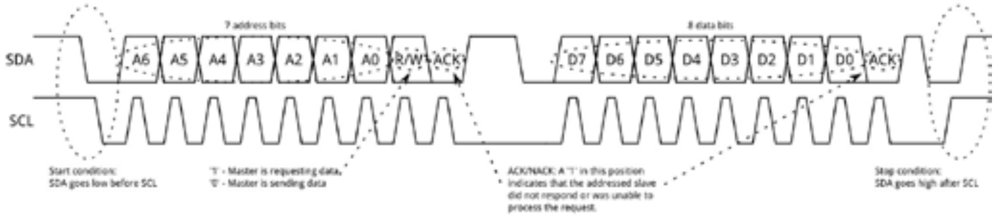


Abbildung: https://learn.sparkfun.com/tutorials/i2c

Multi-Master

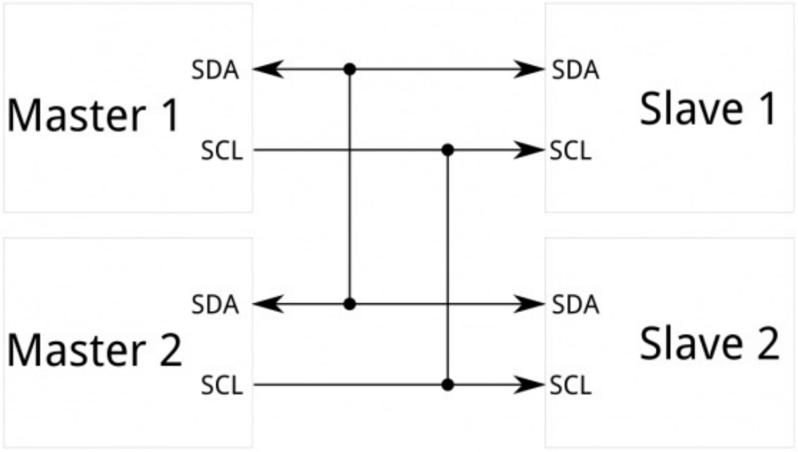


Abbildung: https://learn.sparkfun.com/tutorials/i2c

SPI

- Serial Peripheral Interface
- 4 Leitungen statt 2
- Bidirektional bis 10 MBit/s

Aufbau

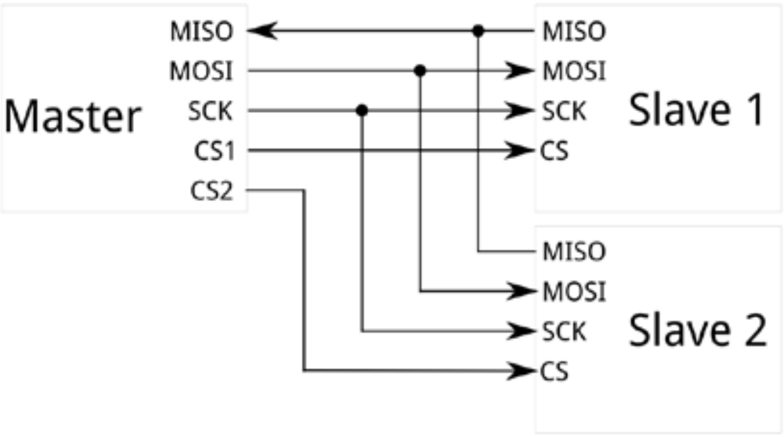
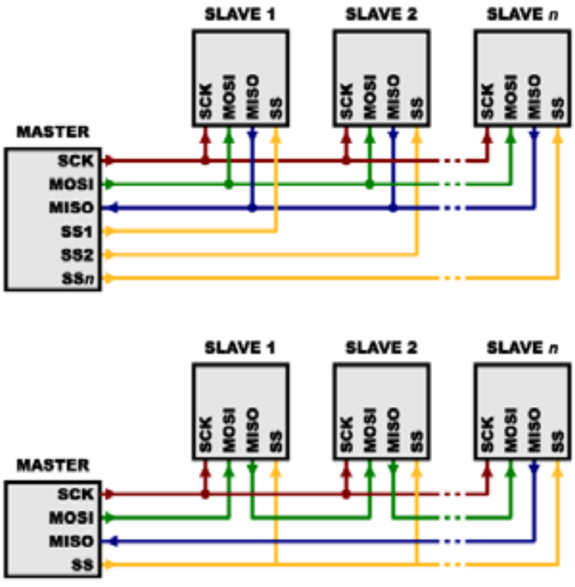


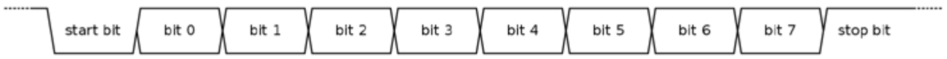
Abbildung: https://learn.sparkfun.com/tutorials/i2c

Spielarten



UART

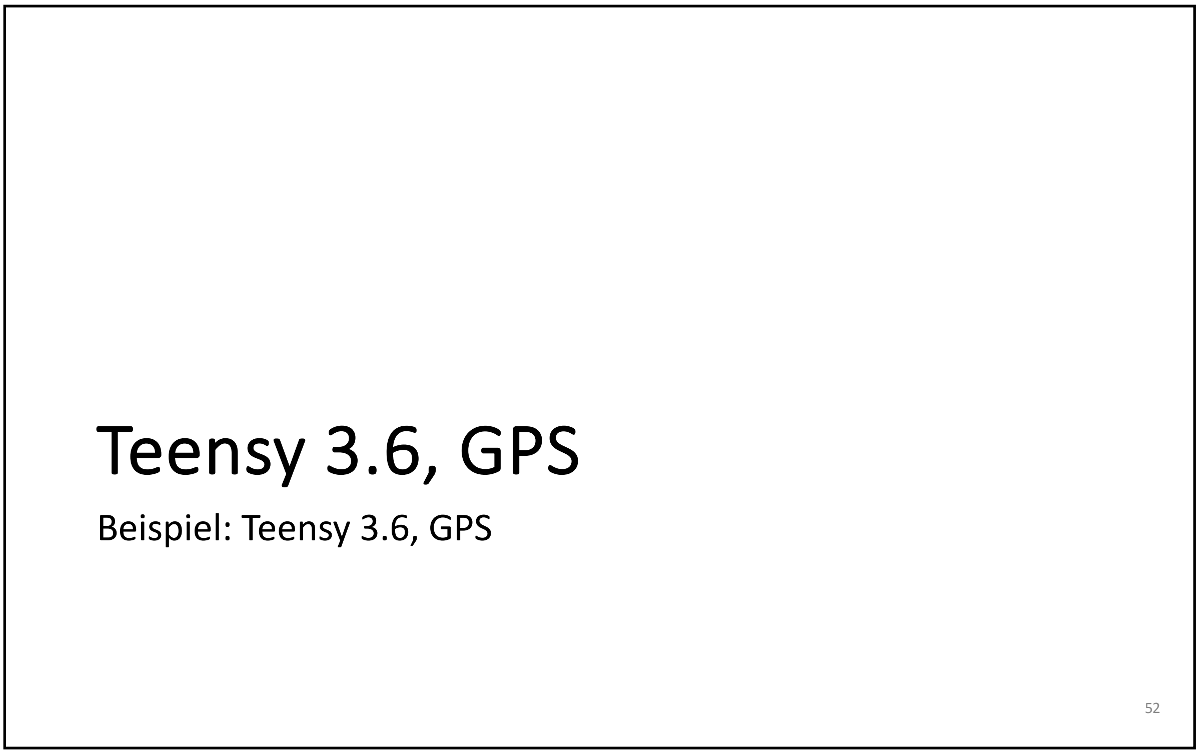
- Universal Asynchronous Rceiver-Transmitter
- Seriell, Asynchron
- Impliziter Takt (Baudrate)
- Start- und Stop-Bits



UART to USB

- Günstige Wandlerchips
- Häufig auf dem Host Treiber nötig

51



Teensy 3.6, GPS

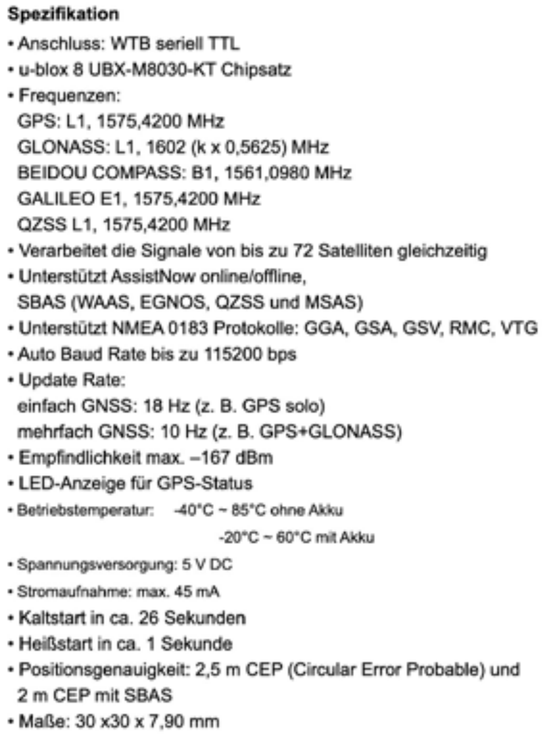
Beispiel: Teensy 3.6, GPS

52

GPS-Empfänger

53

NL-852ETTL

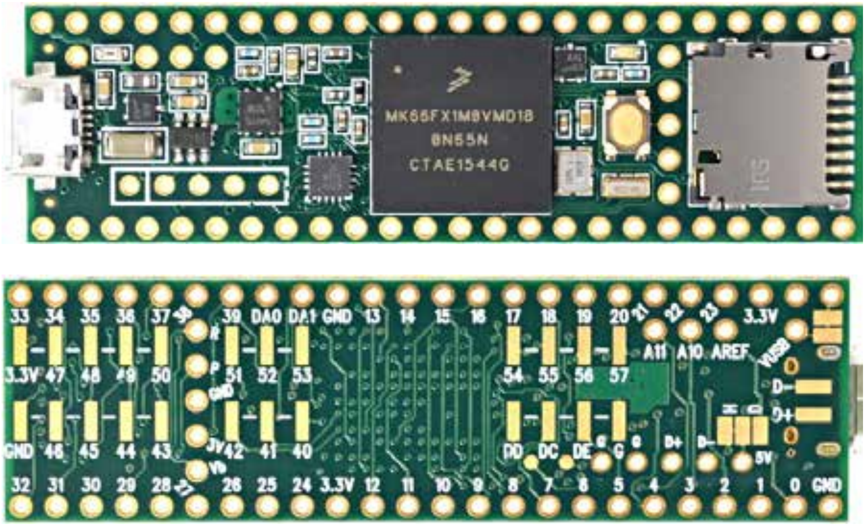


Spezifikation

- Anschluss: WTB seriell TTL
- u-blox 8 UBX-M8030-KT Chipsatz
- Frequenzen:
  GPS: L1, 1575,4200 MHz
  GLONASS: L1, 1602 (k x 0,5625) MHz
  BEIDOU COMPASS: B1, 1561,0980 MHz
  GALILEO E1, 1575,4200 MHz
  QZSS L1, 1575,4200 MHz
- Verarbeitet die Signale von bis zu 72 Satelliten gleichzeitig
- Unterstützt AssistNow online/offline,
  SBAS (WAAS, EGNOS, QZSS und MSAS)
- Unterstützt NMEA 0183 Protokolle: GGA, GSA, GSV, RMC, VTG
- Auto Baud Rate bis zu 115200 bps
- Update Rate:
  einfach GNSS: 18 Hz (z. B. GPS solo)
  mehrfach GNSS: 10 Hz (z. B. GPS+GLONASS)
- Empfindlichkeit max. –167 dBm
- LED-Anzeige für GPS-Status
- Betriebstemperatur: -40°C ~ 85°C ohne Akku
  -20°C ~ 60°C mit Akku
- Spannungsversorgung: 5 V DC
- Stromaufnahme: max. 45 mA
- Kaltstart in ca. 26 Sekunden
- Heißstart in ca. 1 Sekunde
- Positionsgenauigkeit: 2,5 m CEP (Circular Error Probable) und
  2 m CEP mit SBAS
- Maße: 30 x30 x 7,90 mm

54

Teensy 3.6


MK66FX1M0VMD18
0N65N
CTAE1544G
33 34 35 36 37 39 DA0 DA1 GND 13 14 15 16 17 18 19 20 21 22 23 3.3V
3.3V 47 48 49 50 51 52 53 54 55 56 57 A11 A10 AREF VUSB
GND 46 45 44 43 42 41 40 DD DC DE D+ D- 5V
32 31 30 29 28 26 25 24 3.3V 12 11 10 9 8 7 6 5 4 3 2 1 0 GND

Specs Teensy 3.6

- Microcontroller Chip MK66FX1M0VMD18
- 180 MHz ARM Cortex-M4 with Floating Point Unit
- 1M Flash, 256K RAM, 4K EEPROM
- 22 PWM Outputs
- 62 I/O Pins (42 breadboard friendly)
- 25 Analog Inputs to 2 ADCs with 13 bits resolution
- 2 Analog Outputs (DACs) with 12 bit resolution
- Ethernet mac, capable of full 100 Mbit/sec speed

More Teensy 3.6 Specs

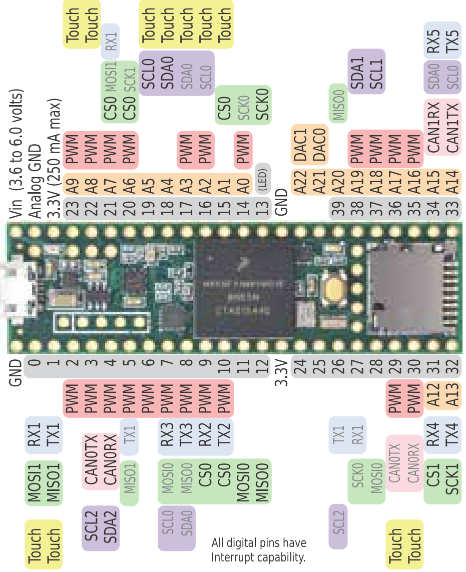
- 4 I2C Ports
- 11 Touch Sensing Inputs
- 2 CAN Bus Ports
- 32 General Purpose DMA Channels
- Native (4 bit SDIO) micro SD card port
- I2S Audio Port, 4 Channel Digital Audio Input & Output
- 14 Hardware Timers
- Cryptographic Acceleration Unit
- Random Number Generator
- CRC Computation Unit
- 6 Serial Ports (2 with FIFO & Fast Baud Rates)
- 3 SPI Ports (1 with FIFO)
- 3 I2C Ports (Teensy 3.6 has a 4th I2C port)
- Real Time Clock

Welcome to Teensy® 3.6

32 Bit Arduino-Compatible Microcontroller

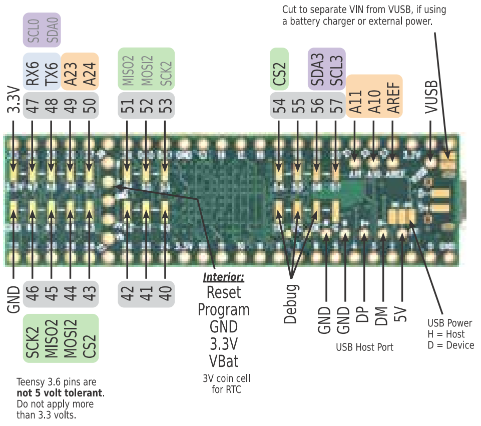
To begin using Teensy, please visit the website & click Getting Started.

www.pjrc.com/teensy



Teensy® 3.6 Back Side

Additional pins and features available on the back side



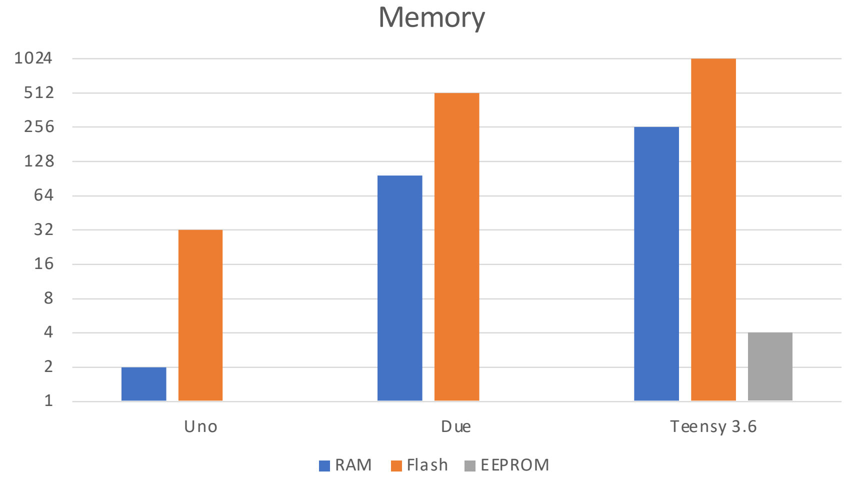
For solutions to the most common issues and technical support, please visit:

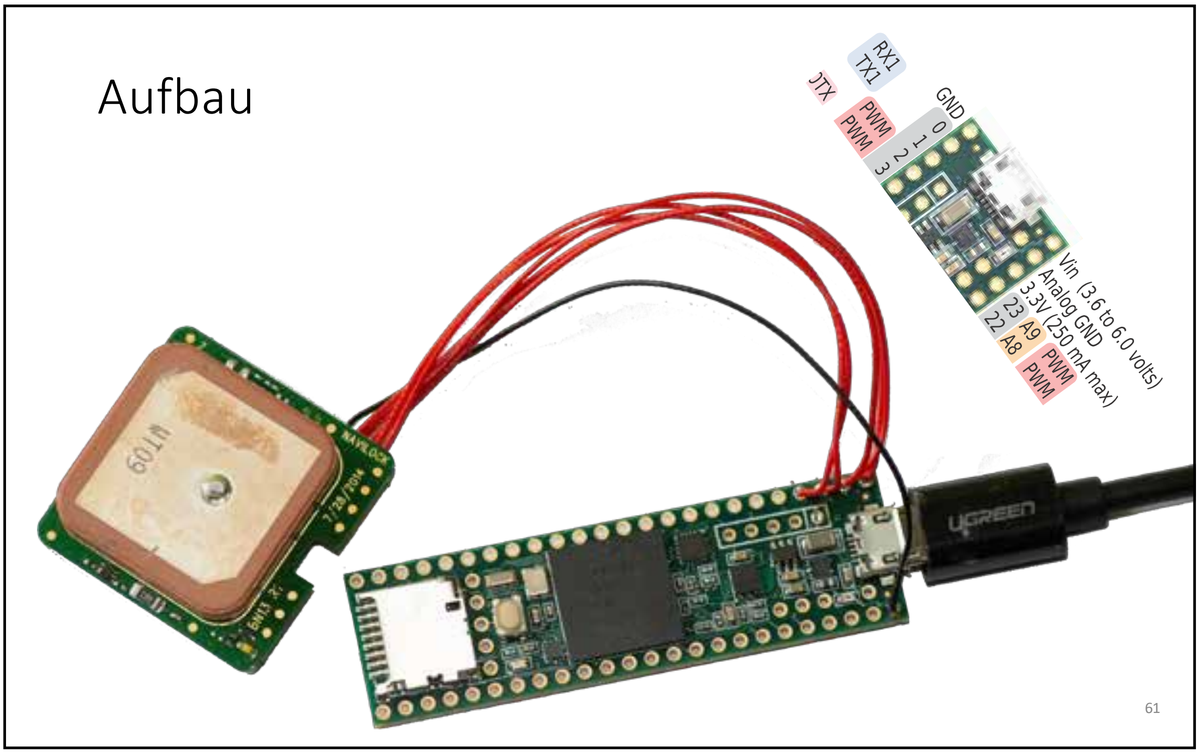
www.pjrc.com/help

Teensy 3.6 System Requirements:
PC computer with Windows 7, 8, 10 or later
or Ubuntu Linux 12.04 or later
or Macintosh OS-X 10.7 or later
USB Micro-B Cable

7 14833 87941 1

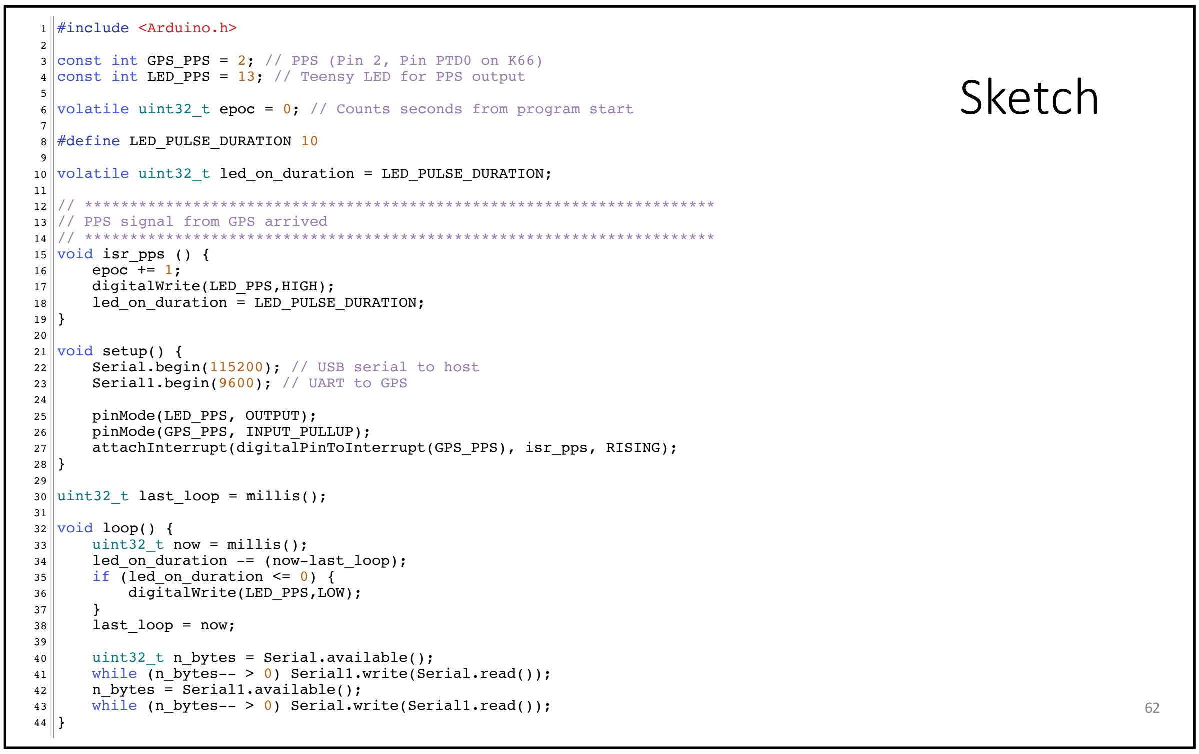
Vergleich





Aufbau

61

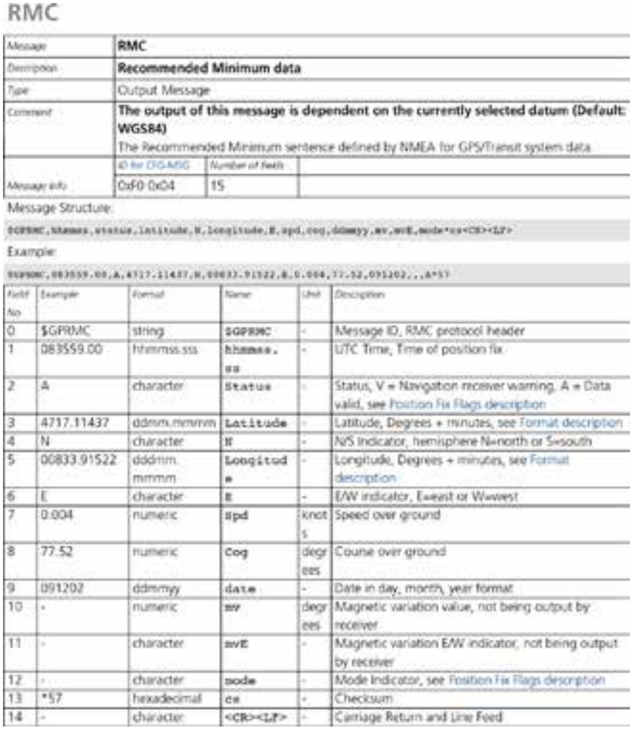


Sketch

```
#include <Arduino.h>

const int GPS_PPS = 2; // PPS (Pin 2, Pin PTD0 on K66)
const int LED_PPS = 13; // Teensy LED for PPS output

volatile uint32_t epoc = 0; // Counts seconds from program start

#define LED_PULSE_DURATION 10

volatile uint32_t led_on_duration = LED_PULSE_DURATION;

// ***********************************************************************
// PPS signal from GPS arrived
// ***********************************************************************
void isr_pps () {
    epoc += 1;
    digitalWrite(LED_PPS,HIGH);
    led_on_duration = LED_PULSE_DURATION;
}

void setup() {
    Serial.begin(115200); // USB serial to host
    Serial1.begin(9600); // UART to GPS

    pinMode(LED_PPS, OUTPUT);
    pinMode(GPS_PPS, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(GPS_PPS), isr_pps, RISING);
}

uint32_t last_loop = millis();

void loop() {
    uint32_t now = millis();
    led_on_duration -= (now-last_loop);
    if (led_on_duration <= 0) {
        digitalWrite(LED_PPS,LOW);
    }
    last_loop = now;

    uint32_t n_bytes = Serial.available();
    while (n_bytes-- > 0) Serial1.write(Serial.read());
    n_bytes = Serial1.available();
    while (n_bytes-- > 0) Serial.write(Serial1.read());
}
```

62

NME 0183



RMC

Message	**RMC**		
Description	**Recommended Minimum data**		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (Default: WGS84)** The Recommended Minimum sentence defined by NMEA for GPS/Transit system data		
	ID for CFG-MSG	Number of fields	
Message info	0xF0 0x04	15	

Message Structure:

```
$GPRMC,hhmmss,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode*cs<CR><LF>
```

Example:

```
$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A*57
```

Field No	Example	Format	Name	Unit	Description
0	$GPRMC	string	$GPRMC	-	Message ID, RMC protocol header
1	083559.00	hhmmss.sss	hhmmss.ss	-	UTC Time, Time of position fix
2	A	character	Status	-	Status, V = Navigation receiver warning, A = Data valid, see Position Fix Flags description
3	4717.11437	ddmm.mmmmm	Latitude	-	Latitude, Degrees + minutes, see Format description
4	N	character	N	-	N/S Indicator, hemisphere N=north or S=south
5	00833.91522	dddmm.mmmmm	Longitude	-	Longitude, Degrees + minutes, see Format description
6	E	character	E	-	E/W indicator, E=east or W=west
7	0.004	numeric	Spd	knots	Speed over ground
8	77.52	numeric	Cog	degrees	Course over ground
9	091202	ddmmyy	date	-	Date in day, month, year format
10	-	numeric	mv	degrees	Magnetic variation value, not being output by receiver
11	-	character	mvE	-	Magnetic variation E/W indicator, not being output by receiver
12	-	character	mode	-	Mode Indicator, see Position Fix Flags description
13	*57	hexadecimal	cs	-	Checksum
14	-	character	<CR><LF>	-	Carriage Return and Line Feed

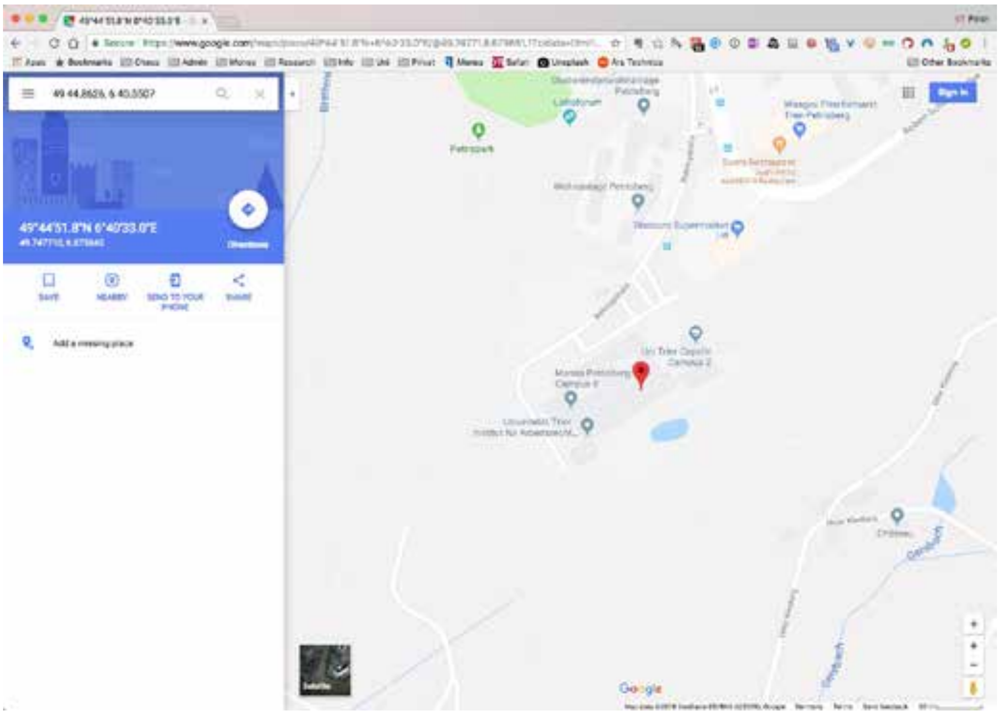
63

Beispiel

NMEA Sequence:
$GPRMC,105155.750,A,4944.8626,N,00640.5507,E,0.41,0.00,170418,,,A*65

- Uhrzeit 10:51:55 UTC (war 12:51 MESZ)
- Position: 4944.8626,N,00640.5507,E
- Speed over Ground (Knoten): 0.41
- Course over Ground: 0.00
- Datum: 17.4.2018

64

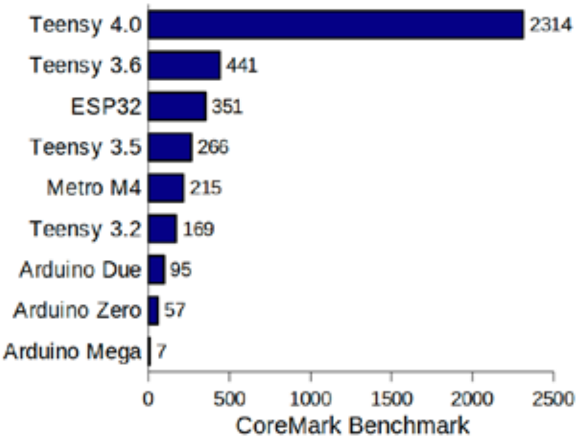


49 44.8626, 6 40.5507

49°44'51.8"N 6°40'33.0"E

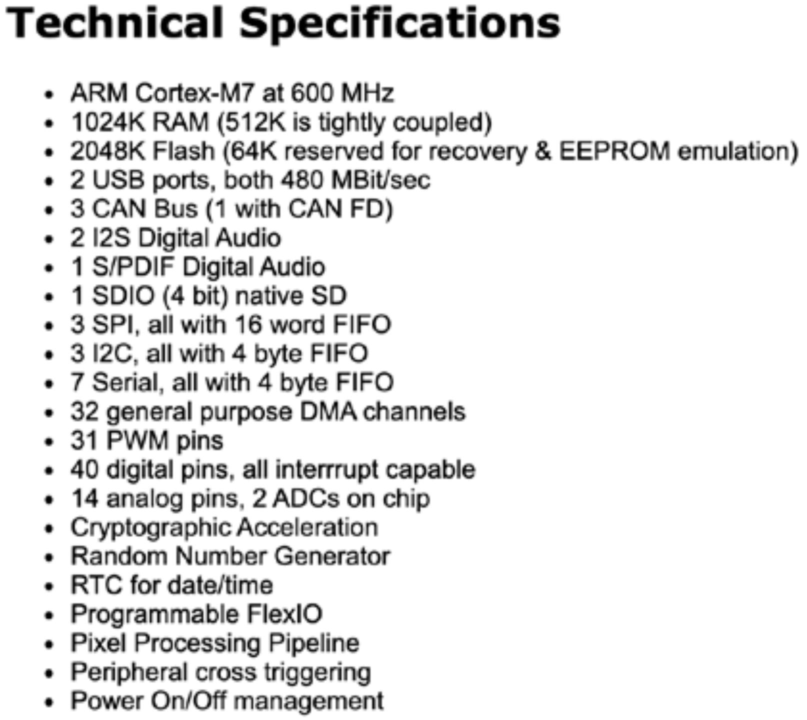
65

Teensy 4.0



https://www.pjrc.com/store/teensy40.html

66



Technical Specifications

- ARM Cortex-M7 at 600 MHz
- 1024K RAM (512K is tightly coupled)
- 2048K Flash (64K reserved for recovery & EEPROM emulation)
- 2 USB ports, both 480 MBit/sec
- 3 CAN Bus (1 with CAN FD)
- 2 I2S Digital Audio
- 1 S/PDIF Digital Audio
- 1 SDIO (4 bit) native SD
- 3 SPI, all with 16 word FIFO
- 3 I2C, all with 4 byte FIFO
- 7 Serial, all with 4 byte FIFO
- 32 general purpose DMA channels
- 31 PWM pins
- 40 digital pins, all interrrupt capable
- 14 analog pins, 2 ADCs on chip
- Cryptographic Acceleration
- Random Number Generator
- RTC for date/time
- Programmable FlexIO
- Pixel Processing Pipeline
- Peripheral cross triggering
- Power On/Off management

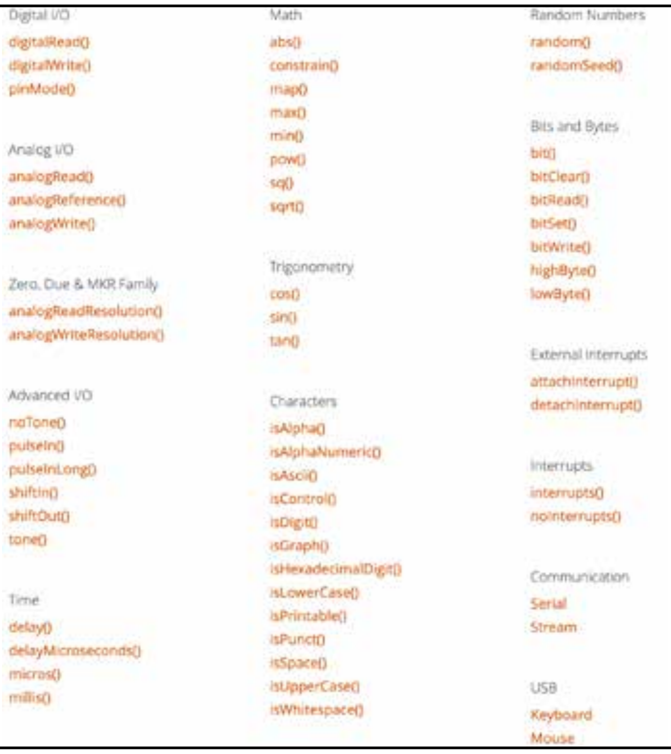
67

Arduino API

68

Arduino API

https://www.arduino.cc/reference/en



Digital I/O
- digitalRead()
- digitalWrite()
- pinMode()

Analog I/O
- analogRead()
- analogReference()
- analogWrite()

Zero, Due & MKR Family
- analogReadResolution()
- analogWriteResolution()

Advanced I/O
- noTone()
- pulseIn()
- pulseInLong()
- shiftIn()
- shiftOut()
- tone()

Time
- delay()
- delayMicroseconds()
- micros()
- millis()

Math
- abs()
- constrain()
- map()
- max()
- min()
- pow()
- sq()
- sqrt()

Trigonometry
- cos()
- sin()
- tan()

Characters
- isAlpha()
- isAlphaNumeric()
- isAscii()
- isControl()
- isDigit()
- isGraph()
- isHexadecimalDigit()
- isLowerCase()
- isPrintable()
- isPunct()
- isSpace()
- isUpperCase()
- isWhitespace()

Random Numbers
- random()
- randomSeed()

Bits and Bytes
- bit()
- bitClear()
- bitRead()
- bitSet()
- bitWrite()
- highByte()
- lowByte()

External Interrupts
- attachInterrupt()
- detachInterrupt()

Interrupts
- interrupts()
- noInterrupts()

Communication
- Serial
- Stream

USB
- Keyboard
- Mouse

69

ESP32

70

ESP32

ca. 9 Euro

71

Specs

- 2.4 GHz Wifi
- Bluetooth v4.2 BR/EDR and BLE
- Xtensa Dual-Core 32-bit LX6 Mikroprozessor (240 MHz)
- 4 MB Flash
- 520 KB RAM
- 16 KB SRAM in RTC

72

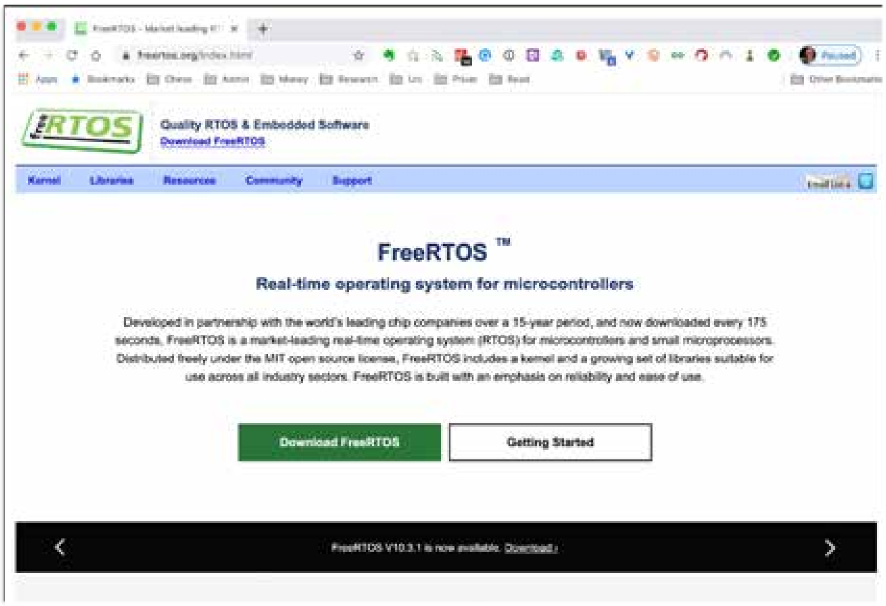
I/O

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

73

Frameworks

- Arduino
- FreeRTOS



FreeRTOS ™

Real-time operating system for microcontrollers

Developed in partnership with the world's leading chip companies over a 15-year period, and now downloaded every 175 seconds, FreeRTOS is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors. Distributed freely under the MIT open source license, FreeRTOS includes a kernel and a growing set of libraries suitable for use across all industry sectors. FreeRTOS is built with an emphasis on reliability and ease of use.

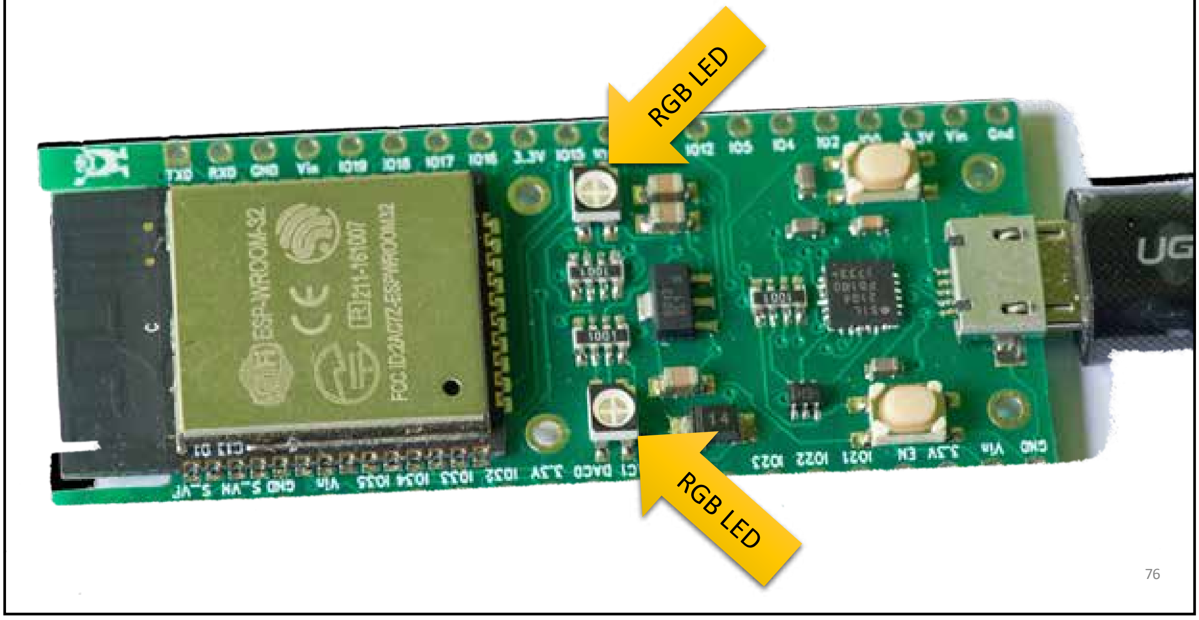
FreeRTOS V10.3.1 is now available. Download »

ESP32 (EzSBC)

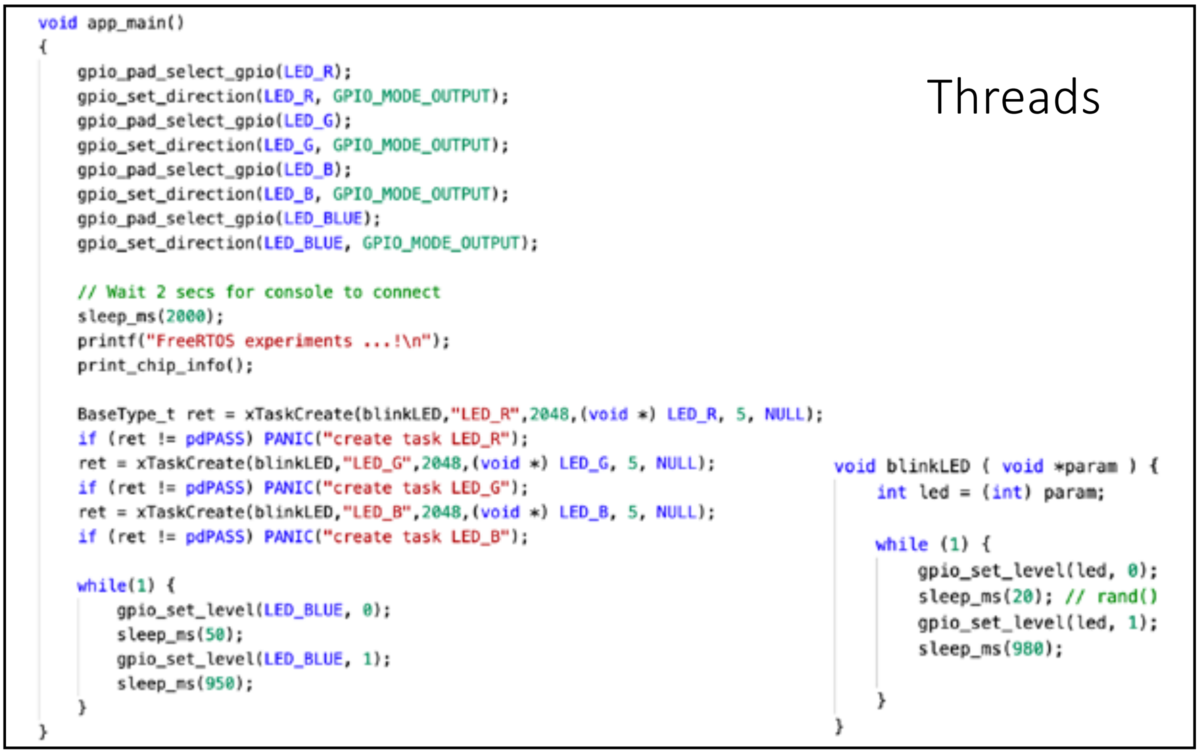
LEDs über Threads ansteuern

75

ESP32 (EzSBC)



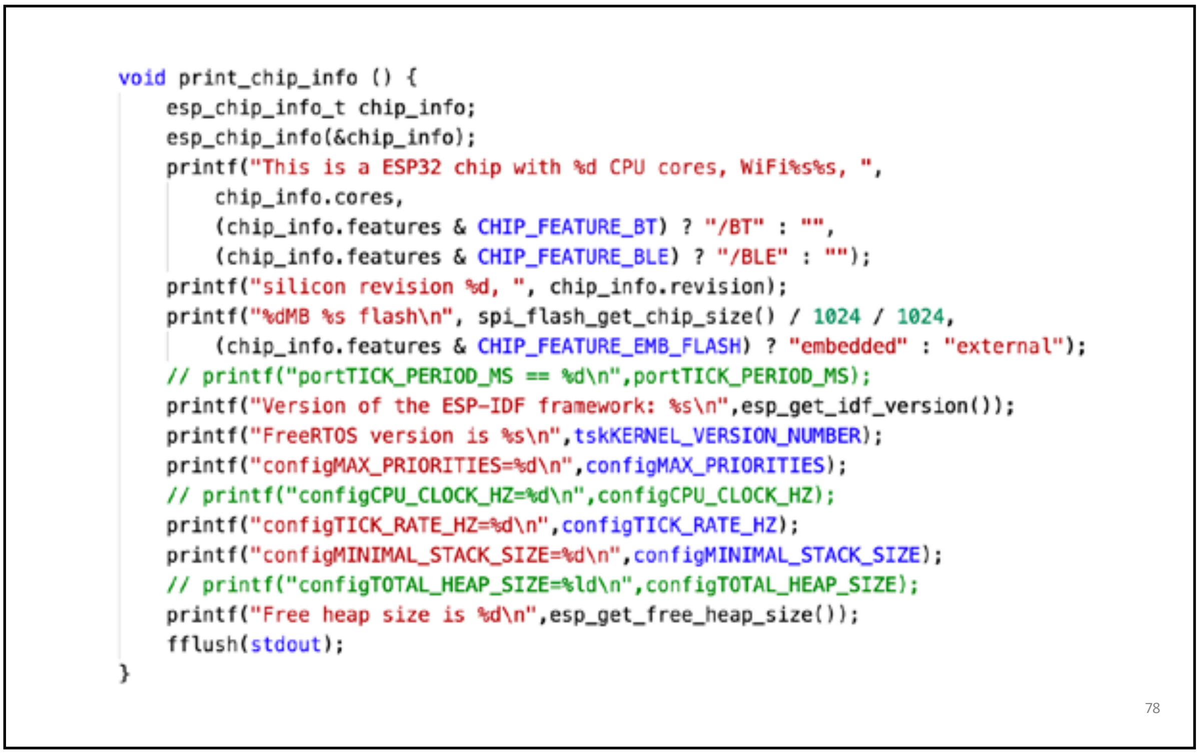
76



Threads

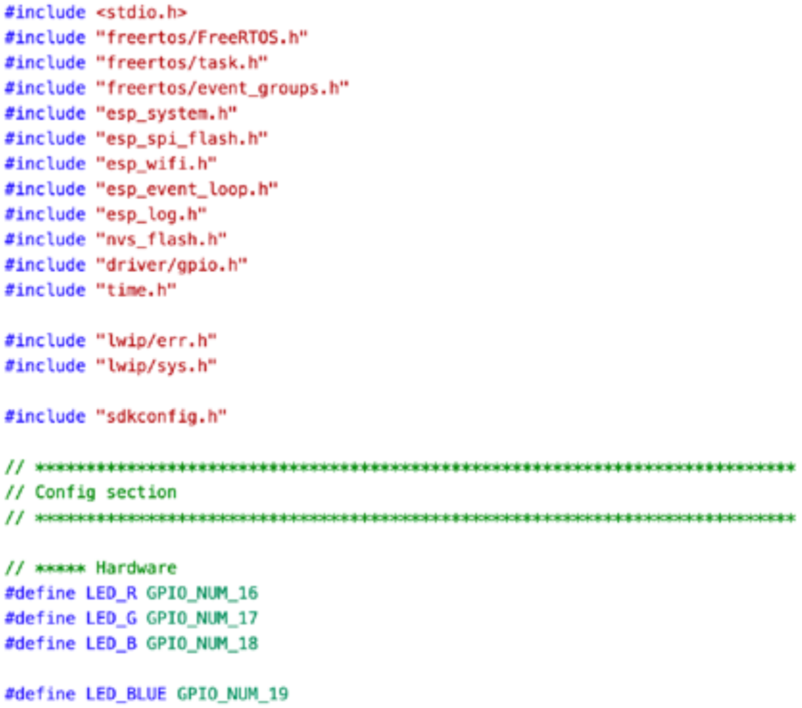
```
void app_main()
{
    gpio_pad_select_gpio(LED_R);
    gpio_set_direction(LED_R, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(LED_G);
    gpio_set_direction(LED_G, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(LED_B);
    gpio_set_direction(LED_B, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(LED_BLUE);
    gpio_set_direction(LED_BLUE, GPIO_MODE_OUTPUT);

    // Wait 2 secs for console to connect
    sleep_ms(2000);
    printf("FreeRTOS experiments ...!\n");
    print_chip_info();

    BaseType_t ret = xTaskCreate(blinkLED,"LED_R",2048,(void *) LED_R, 5, NULL);
    if (ret != pdPASS) PANIC("create task LED_R");
    ret = xTaskCreate(blinkLED,"LED_G",2048,(void *) LED_G, 5, NULL);
    if (ret != pdPASS) PANIC("create task LED_G");
    ret = xTaskCreate(blinkLED,"LED_B",2048,(void *) LED_B, 5, NULL);
    if (ret != pdPASS) PANIC("create task LED_B");

    while(1) {
        gpio_set_level(LED_BLUE, 0);
        sleep_ms(50);
        gpio_set_level(LED_BLUE, 1);
        sleep_ms(950);
    }
}
```

```
void blinkLED ( void *param ) {
    int led = (int) param;

    while (1) {
        gpio_set_level(led, 0);
        sleep_ms(20); // rand()
        gpio_set_level(led, 1);
        sleep_ms(980);

    }
}
```



```
void print_chip_info () {
    esp_chip_info_t chip_info;
    esp_chip_info(&chip_info);
    printf("This is a ESP32 chip with %d CPU cores, WiFi%s%s, ",
        chip_info.cores,
        (chip_info.features & CHIP_FEATURE_BT) ? "/BT" : "",
        (chip_info.features & CHIP_FEATURE_BLE) ? "/BLE" : "");
    printf("silicon revision %d, ", chip_info.revision);
    printf("%dMB %s flash\n", spi_flash_get_chip_size() / 1024 / 1024,
        (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "embedded" : "external");
    // printf("portTICK_PERIOD_MS == %d\n",portTICK_PERIOD_MS);
    printf("Version of the ESP-IDF framework: %s\n",esp_get_idf_version());
    printf("FreeRTOS version is %s\n",tskKERNEL_VERSION_NUMBER);
    printf("configMAX_PRIORITIES=%d\n",configMAX_PRIORITIES);
    // printf("configCPU_CLOCK_HZ=%d\n",configCPU_CLOCK_HZ);
    printf("configTICK_RATE_HZ=%d\n",configTICK_RATE_HZ);
    printf("configMINIMAL_STACK_SIZE=%d\n",configMINIMAL_STACK_SIZE);
    // printf("configTOTAL_HEAP_SIZE=%ld\n",configTOTAL_HEAP_SIZE);
    printf("Free heap size is %d\n",esp_get_free_heap_size());
    fflush(stdout);
}
```

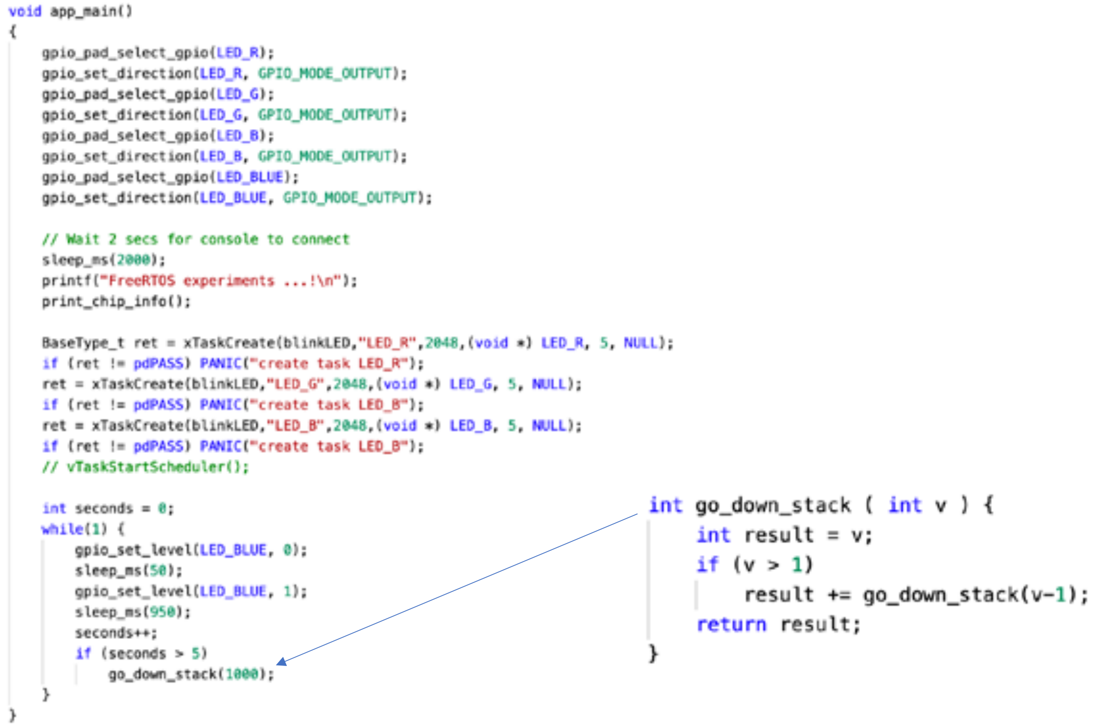
78



```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_spi_flash.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "driver/gpio.h"
#include "time.h"

#include "lwip/err.h"
#include "lwip/sys.h"

#include "sdkconfig.h"

// ***************************************************************************
// Config section
// ***************************************************************************

// ***** Hardware
#define LED_R GPIO_NUM_16
#define LED_G GPIO_NUM_17
#define LED_B GPIO_NUM_18

#define LED_BLUE GPIO_NUM_19
```

ESP32 (EzSBC)

Stack Overflow



```c
void app_main()
{
    gpio_pad_select_gpio(LED_R);
    gpio_set_direction(LED_R, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(LED_G);
    gpio_set_direction(LED_G, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(LED_B);
    gpio_set_direction(LED_B, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(LED_BLUE);
    gpio_set_direction(LED_BLUE, GPIO_MODE_OUTPUT);

    // Wait 2 secs for console to connect
    sleep_ms(2000);
    printf("FreeRTOS experiments ...!\n");
    print_chip_info();

    BaseType_t ret = xTaskCreate(blinkLED,"LED_R",2048,(void *) LED_R, 5, NULL);
    if (ret != pdPASS) PANIC("create task LED_R");
    ret = xTaskCreate(blinkLED,"LED_G",2048,(void *) LED_G, 5, NULL);
    if (ret != pdPASS) PANIC("create task LED_B");
    ret = xTaskCreate(blinkLED,"LED_B",2048,(void *) LED_B, 5, NULL);
    if (ret != pdPASS) PANIC("create task LED_B");
    // vTaskStartScheduler();

    int seconds = 0;
    while(1) {
        gpio_set_level(LED_BLUE, 0);
        sleep_ms(50);
        gpio_set_level(LED_BLUE, 1);
        sleep_ms(950);
        seconds++;
        if (seconds > 5)
            go_down_stack(1000);
    }
}
```

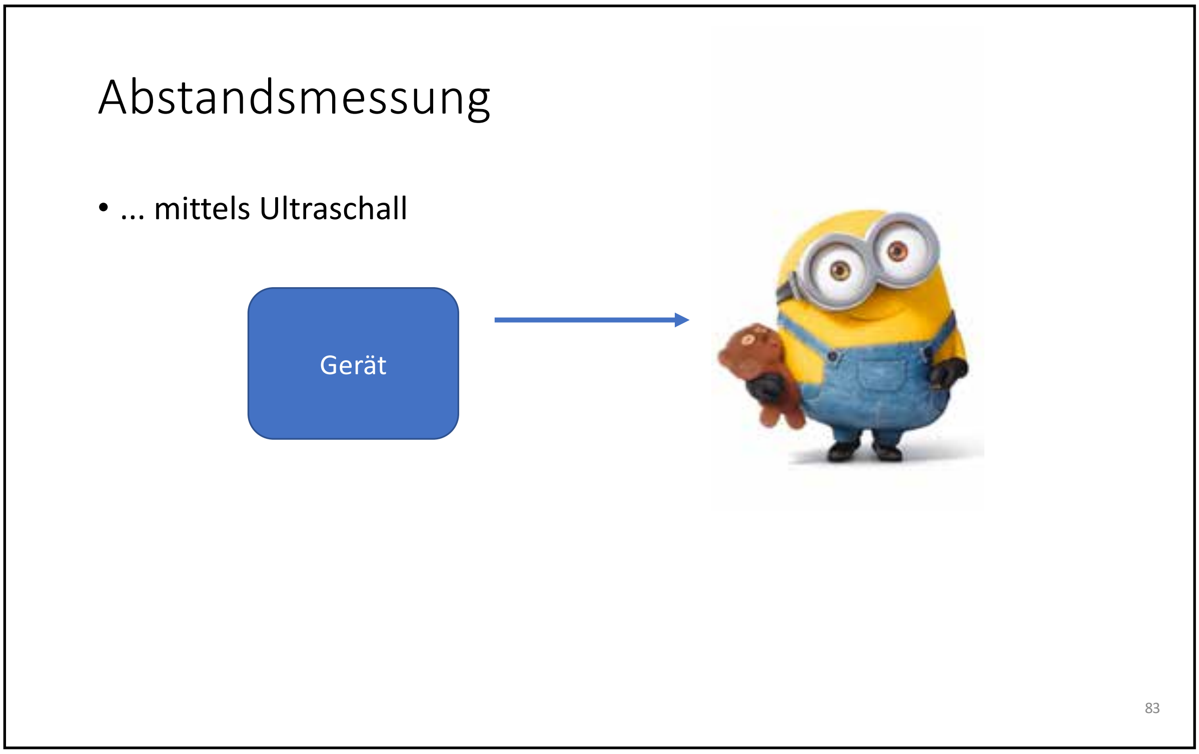
```c
int go_down_stack ( int v ) {
    int result = v;
    if (v > 1)
        result += go_down_stack(v-1);
    return result;
}
```

81

ESP32

Ultraschall

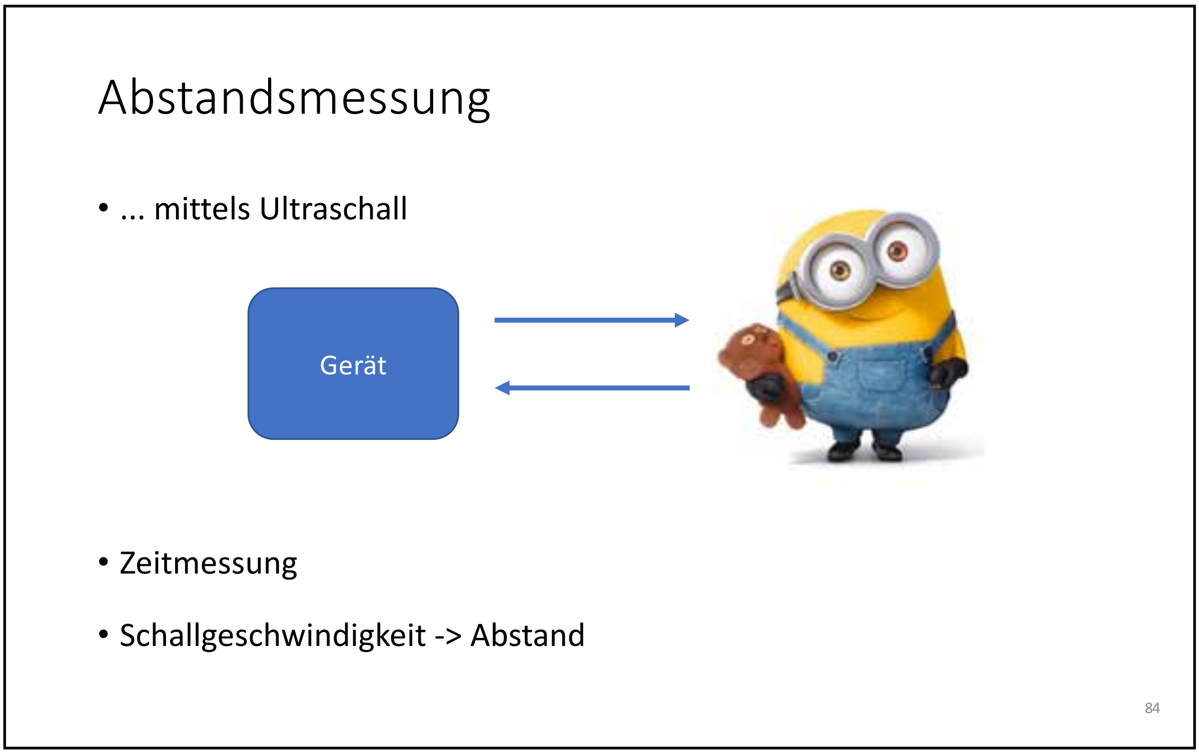
82



Abstandsmessung

- ... mittels Ultraschall

83

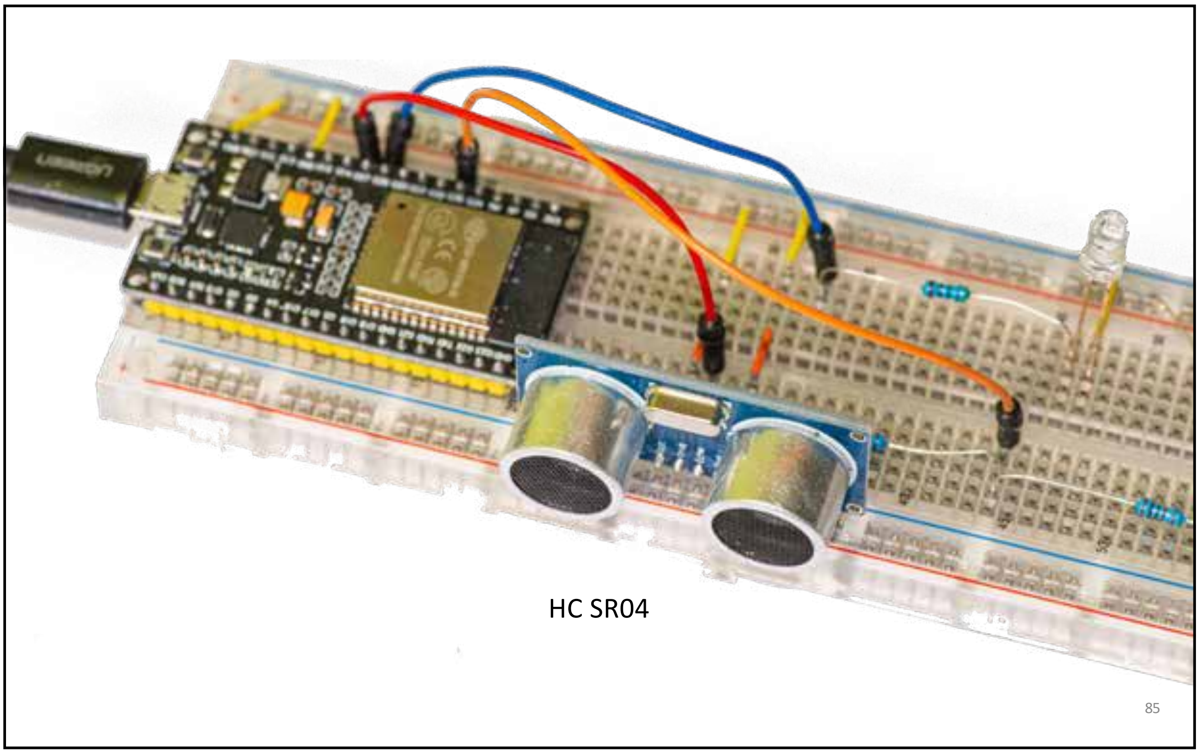


Abstandsmessung

- ... mittels Ultraschall

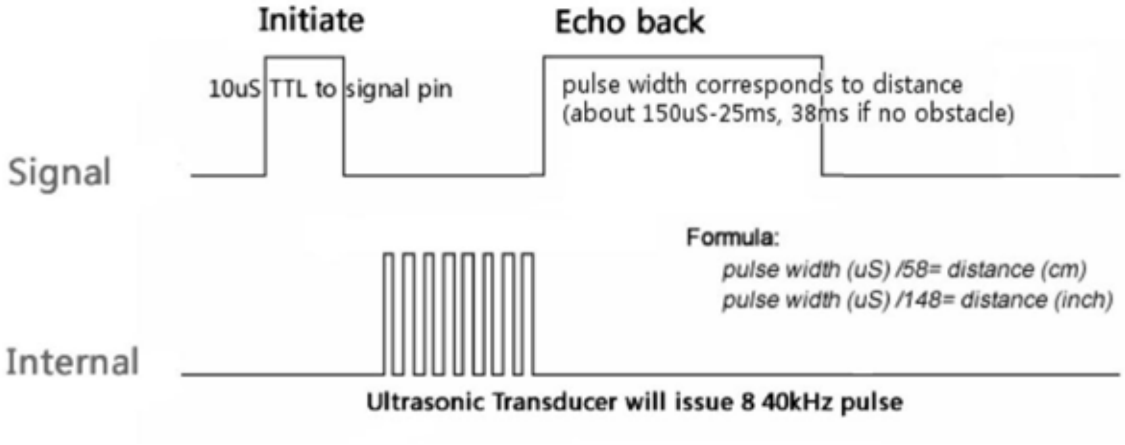
- Zeitmessung
- Schallgeschwindigkeit -> Abstand

84

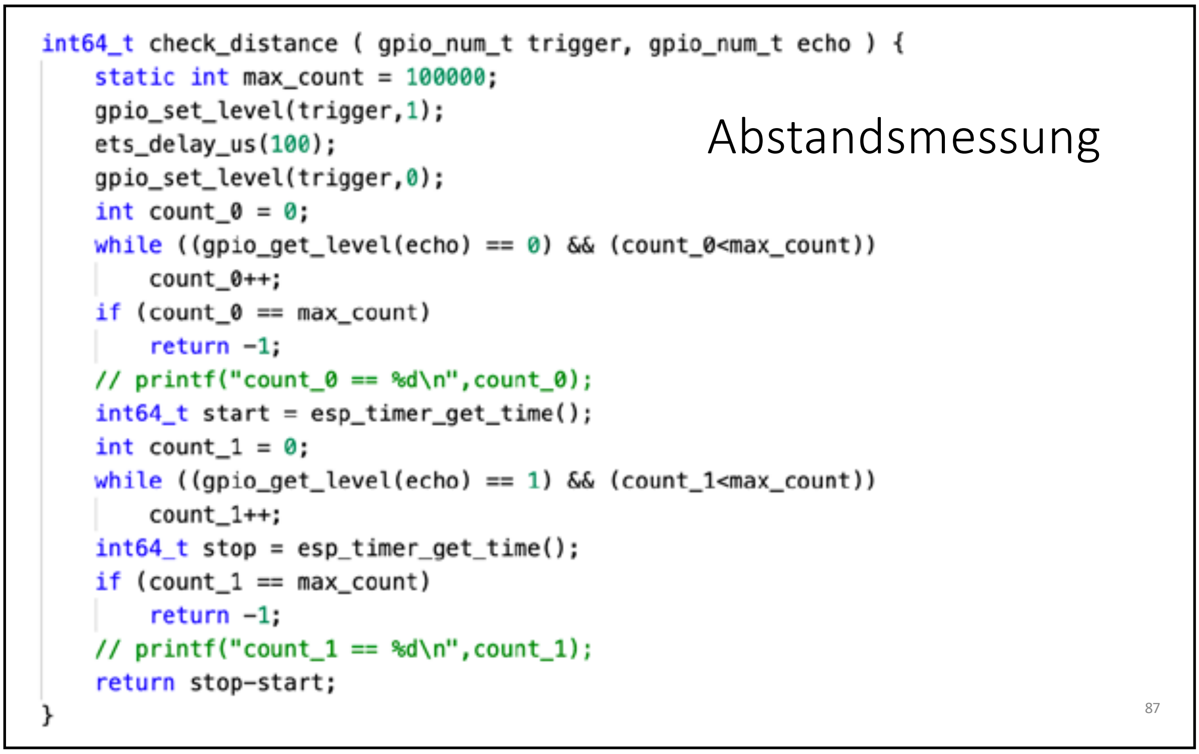


HC SR04

Verfahren



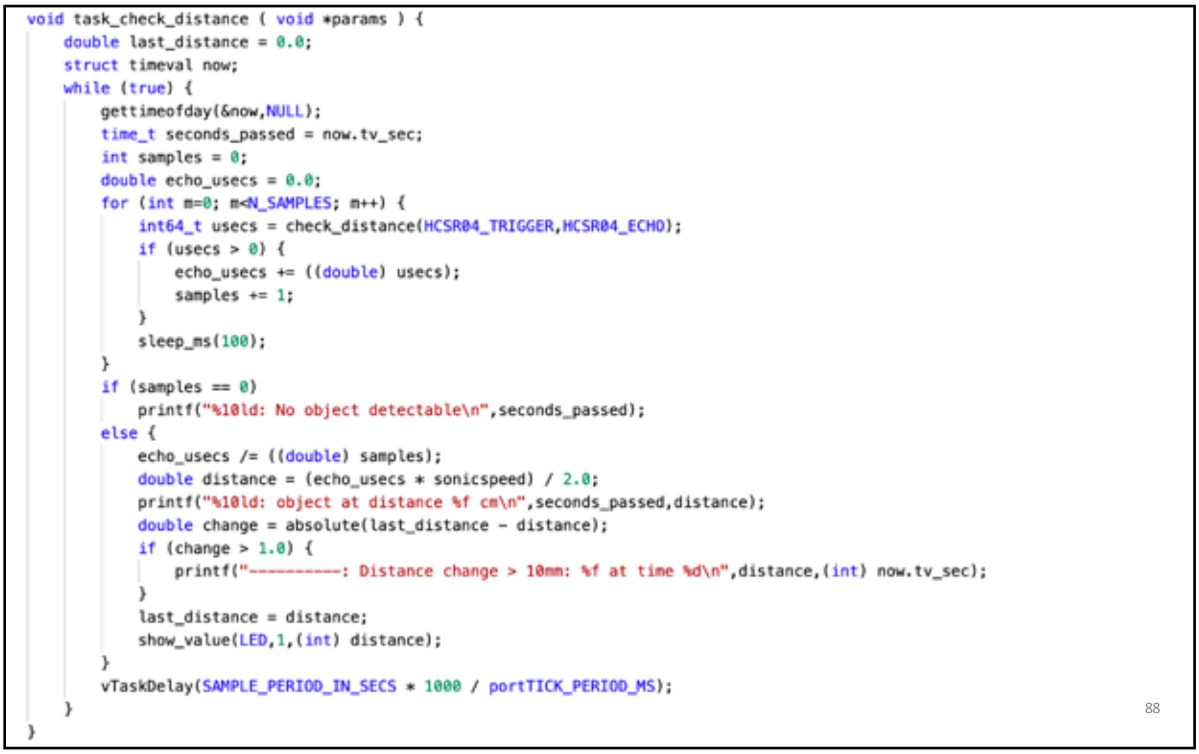
Initiate
Echo back
10uS TTL to signal pin
pulse width corresponds to distance
(about 150uS-25ms, 38ms if no obstacle)
Signal
Formula:
pulse width (uS) /58= distance (cm)
pulse width (uS) /148= distance (inch)
Internal
Ultrasonic Transducer will issue 8 40kHz pulse



Abstandsmessung

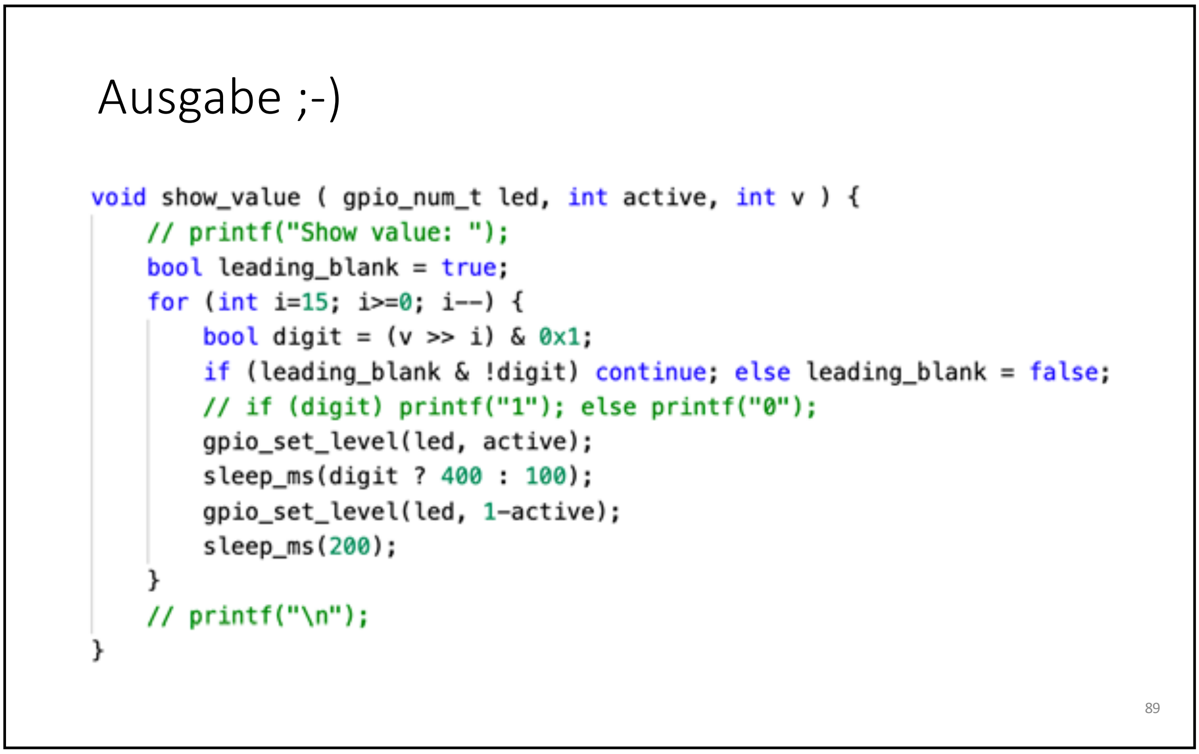
```c
int64_t check_distance ( gpio_num_t trigger, gpio_num_t echo ) {
    static int max_count = 100000;
    gpio_set_level(trigger,1);
    ets_delay_us(100);
    gpio_set_level(trigger,0);
    int count_0 = 0;
    while ((gpio_get_level(echo) == 0) && (count_0<max_count))
        count_0++;
    if (count_0 == max_count)
        return -1;
    // printf("count_0 == %d\n",count_0);
    int64_t start = esp_timer_get_time();
    int count_1 = 0;
    while ((gpio_get_level(echo) == 1) && (count_1<max_count))
        count_1++;
    int64_t stop = esp_timer_get_time();
    if (count_1 == max_count)
        return -1;
    // printf("count_1 == %d\n",count_1);
    return stop-start;
}
```

87



```c
void task_check_distance ( void *params ) {
    double last_distance = 0.0;
    struct timeval now;
    while (true) {
        gettimeofday(&now,NULL);
        time_t seconds_passed = now.tv_sec;
        int samples = 0;
        double echo_usecs = 0.0;
        for (int m=0; m<N_SAMPLES; m++) {
            int64_t usecs = check_distance(HCSR04_TRIGGER,HCSR04_ECHO);
            if (usecs > 0) {
                echo_usecs += ((double) usecs);
                samples += 1;
            }
            sleep_ms(100);
        }
        if (samples == 0)
            printf("%10ld: No object detectable\n",seconds_passed);
        else {
            echo_usecs /= ((double) samples);
            double distance = (echo_usecs * sonicspeed) / 2.0;
            printf("%10ld: object at distance %f cm\n",seconds_passed,distance);
            double change = absolute(last_distance - distance);
            if (change > 1.0) {
                printf("----------: Distance change > 10mm: %f at time %d\n",distance,(int) now.tv_sec);
            }
            last_distance = distance;
            show_value(LED,1,(int) distance);
        }
        vTaskDelay(SAMPLE_PERIOD_IN_SECS * 1000 / portTICK_PERIOD_MS);
    }
}
```

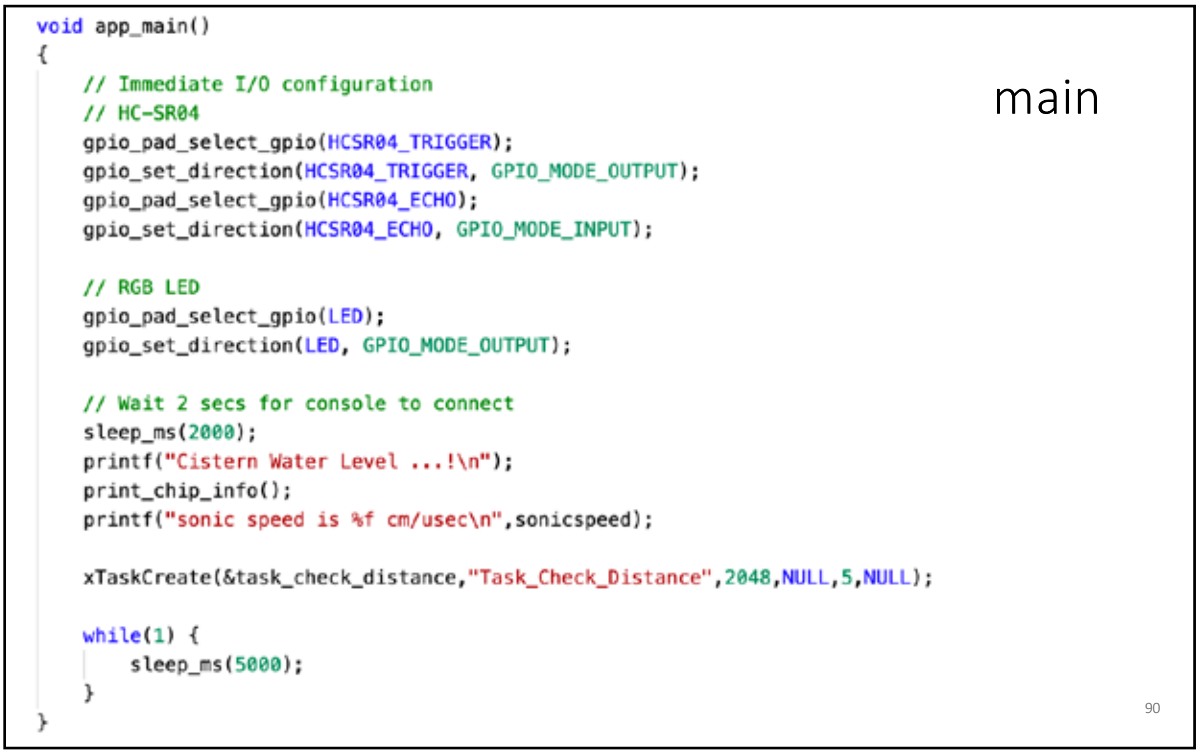
88



Ausgabe ;-)

```
void show_value ( gpio_num_t led, int active, int v ) {
    // printf("Show value: ");
    bool leading_blank = true;
    for (int i=15; i>=0; i--) {
        bool digit = (v >> i) & 0x1;
        if (leading_blank & !digit) continue; else leading_blank = false;
        // if (digit) printf("1"); else printf("0");
        gpio_set_level(led, active);
        sleep_ms(digit ? 400 : 100);
        gpio_set_level(led, 1-active);
        sleep_ms(200);
    }
    // printf("\n");
}
```

89



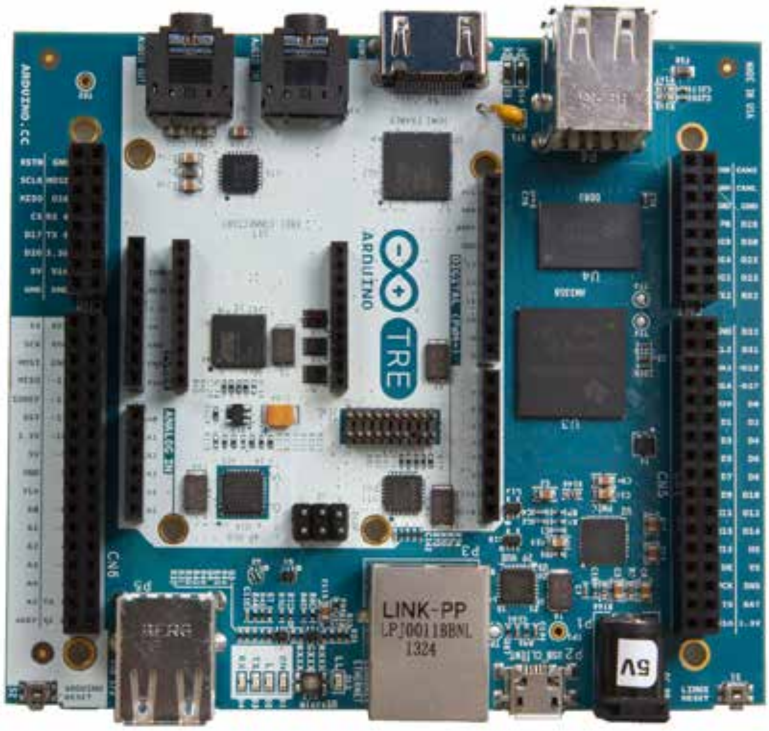
main

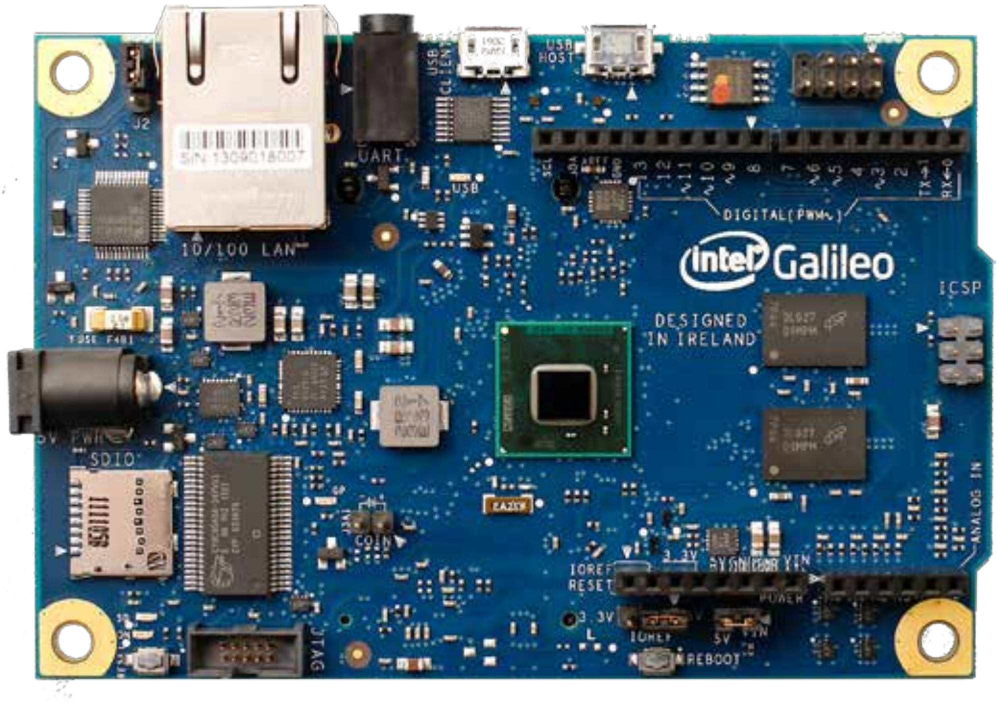
```
void app_main()
{
    // Immediate I/O configuration
    // HC-SR04
    gpio_pad_select_gpio(HCSR04_TRIGGER);
    gpio_set_direction(HCSR04_TRIGGER, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(HCSR04_ECHO);
    gpio_set_direction(HCSR04_ECHO, GPIO_MODE_INPUT);

    // RGB LED
    gpio_pad_select_gpio(LED);
    gpio_set_direction(LED, GPIO_MODE_OUTPUT);

    // Wait 2 secs for console to connect
    sleep_ms(2000);
    printf("Cistern Water Level ...!\n");
    print_chip_info();
    printf("sonic speed is %f cm/usec\n",sonicspeed);

    xTaskCreate(&task_check_distance,"Task_Check_Distance",2048,NULL,5,NULL);

    while(1) {
        sleep_ms(5000);
    }
}
```

90

Hype?

Intel und Arduino



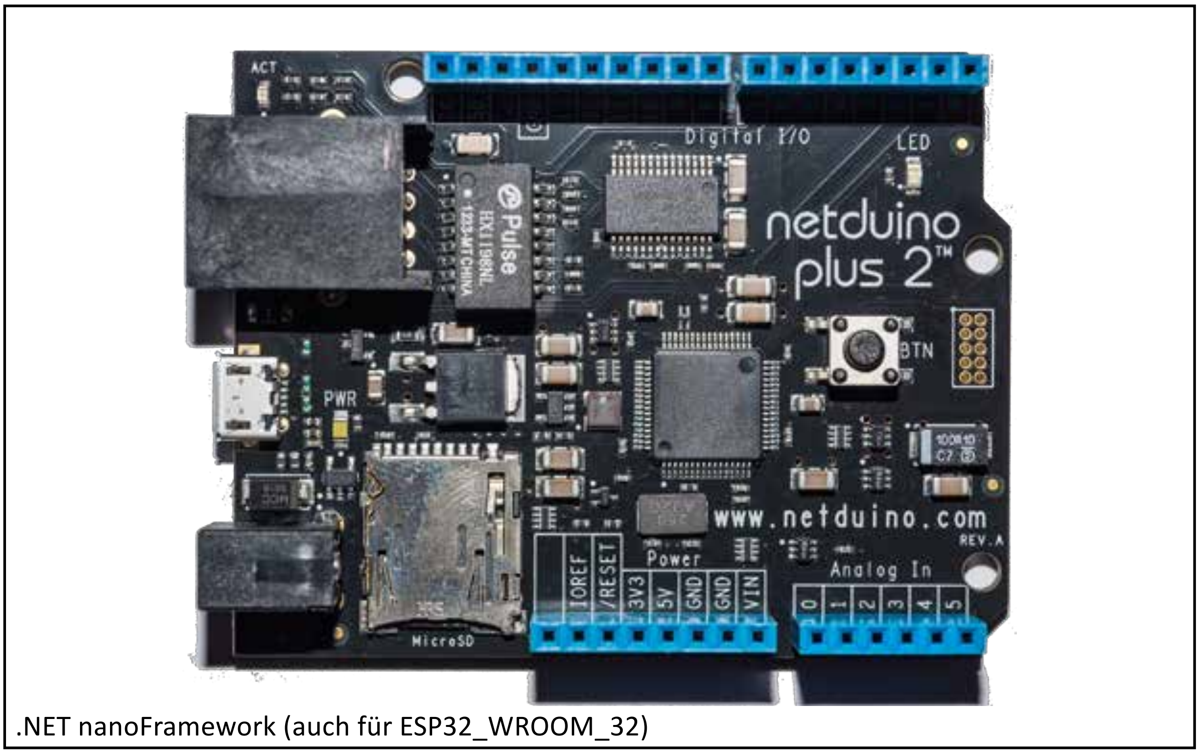


Intel Edison

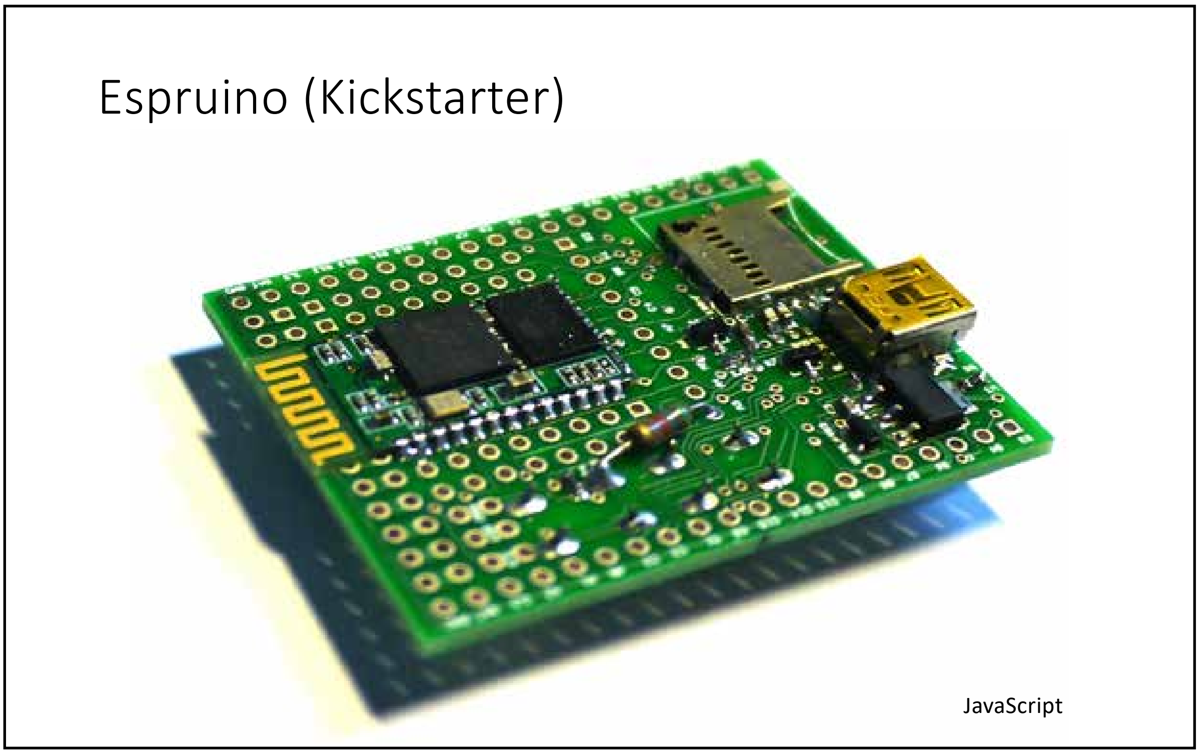
UDOO (Kickstarter)

Programmierung

- Verschiedene Lager
- C/C++
- Python, Skratch
- C# (.NET Micro Framework)
  - Netduino
- JavaScript
  - Tessel



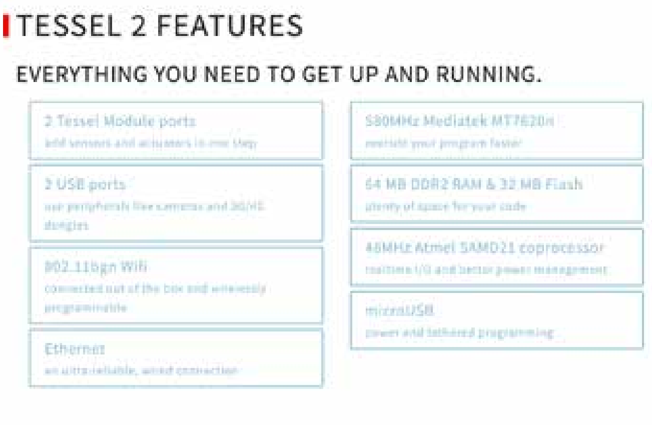
.NET nanoFramework (auch für ESP32_WROOM_32)



Espruino (Kickstarter)

JavaScript

Tessel 2



TESSEL 2 FEATURES

EVERYTHING YOU NEED TO GET UP AND RUNNING.

2 Tessel Module ports
add sensors and actuators in one step

2 USB ports
use peripherals like cameras and 3G/4G dongles

802.11bgn WiFi
connected out of the box and wirelessly programmable

Ethernet
an ultra-reliable, wired connection

580MHz Mediatek MT7620n
execute your program faster

64 MB DDR2 RAM & 32 MB Flash
plenty of space for your code

48MHz Atmel SAMD21 coprocessor
realtime I/O and better power management

microUSB
power and tethered programming

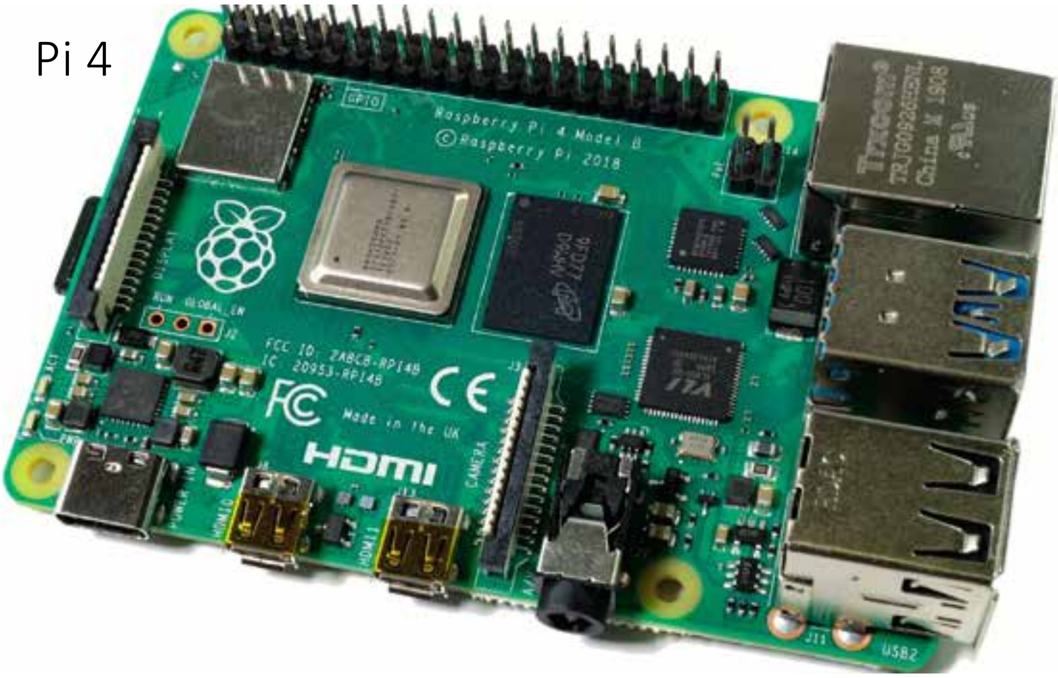
... Und viele viele mehr

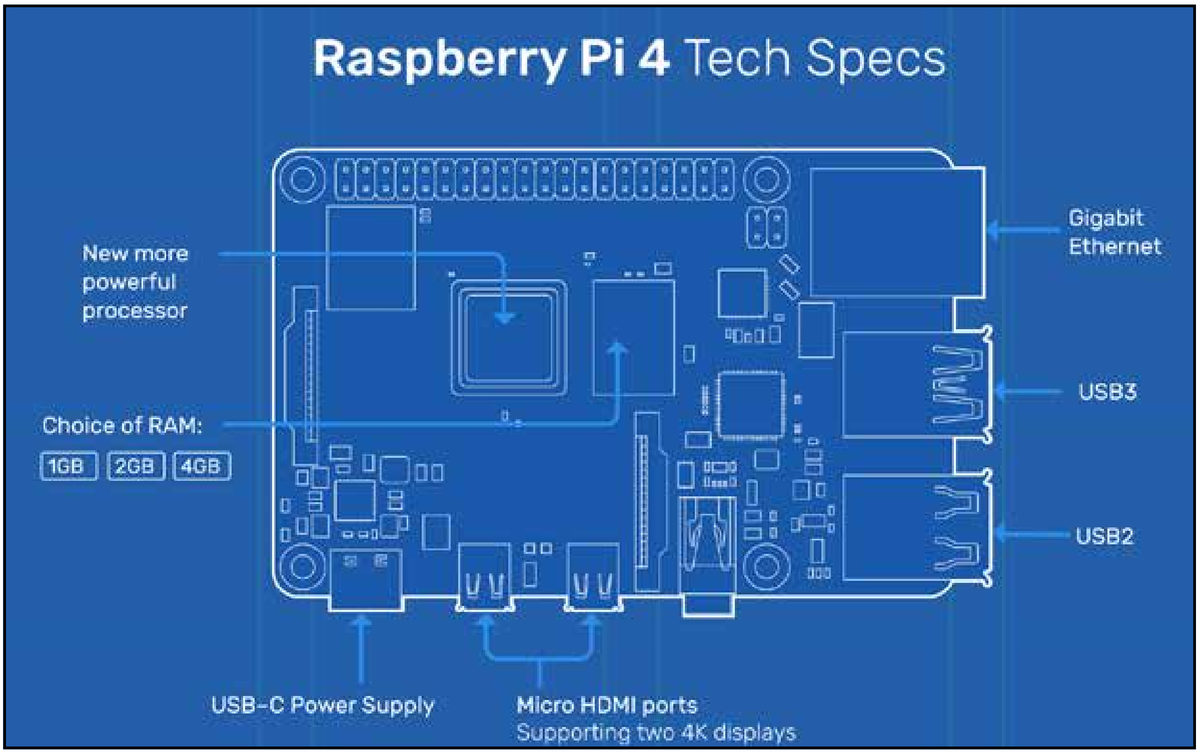
- Flashgröße
- RAM-Größe
- EEPROM? Wenn ja, wie groß?
- Taktfrequenz
- Formfaktor
- Softwareumfeld
- ...

100

Raspberry Pi

Pi 4





Raspberry Pi 4 Tech Specs

New more powerful processor

Choice of RAM:

1GB 2GB 4GB

Gigabit Ethernet

USB3

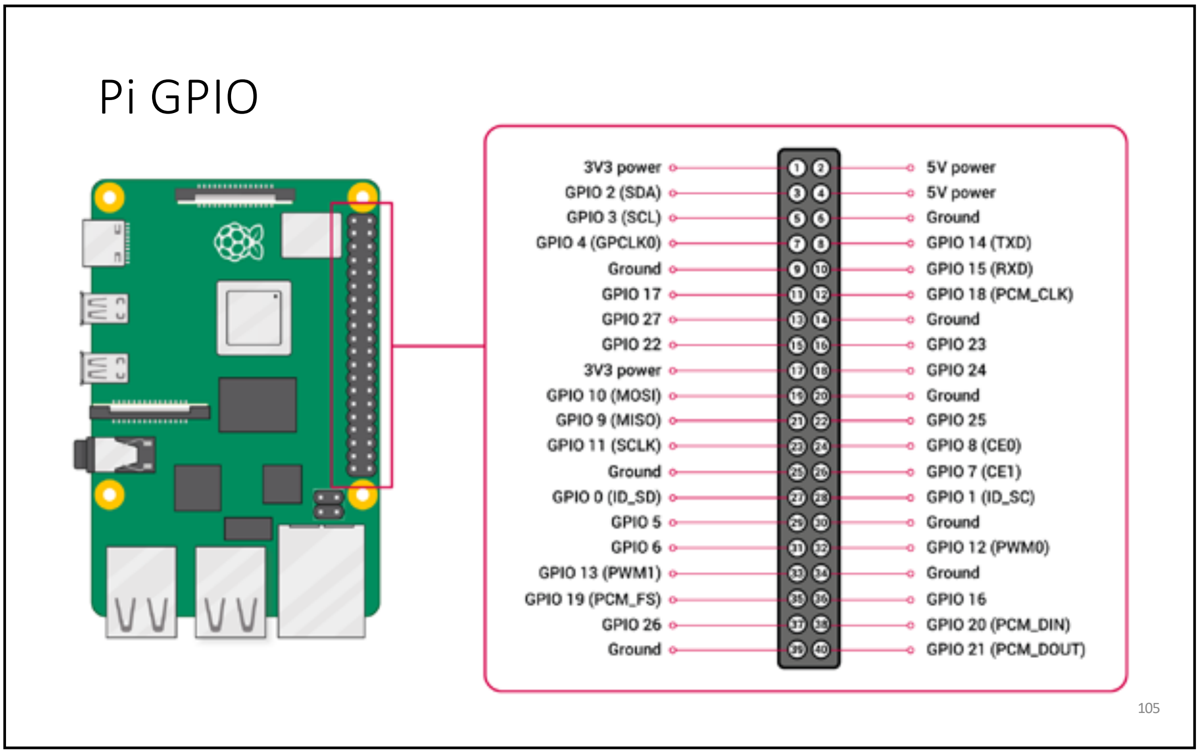
USB2

USB-C Power Supply

Micro HDMI ports
Supporting two 4K displays

Specs

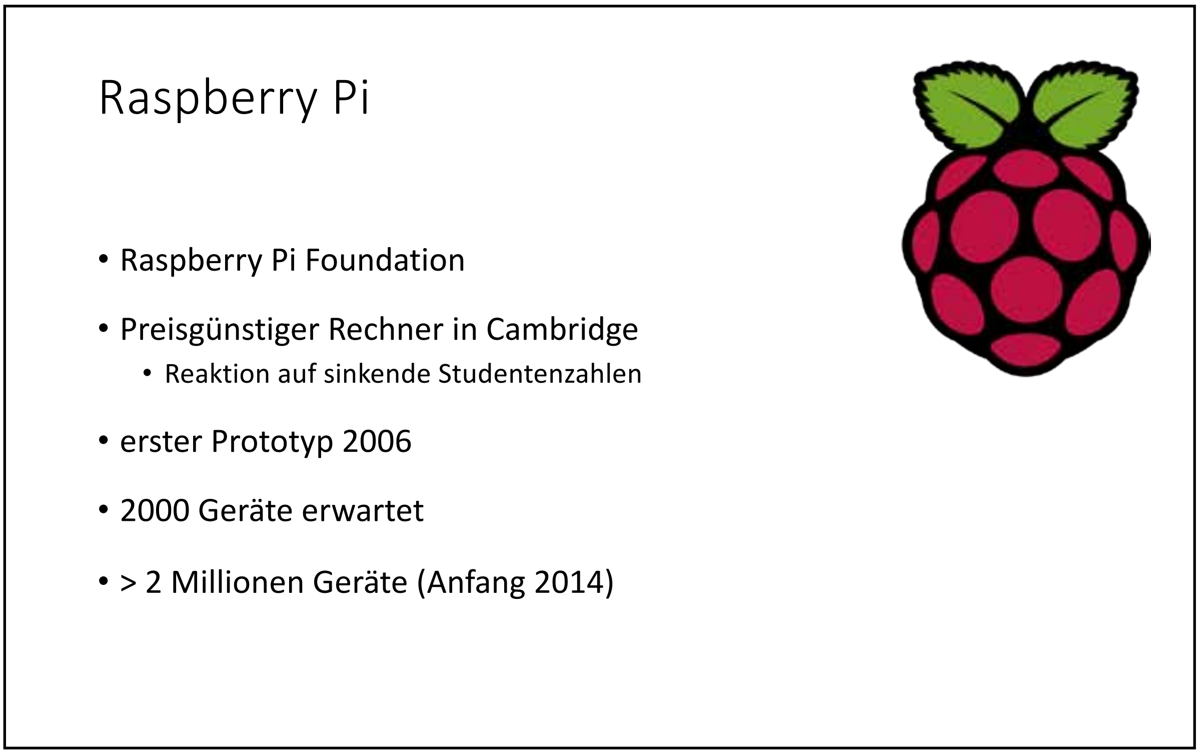
- Quad Core ARM Cortex-A72 (ARM v8), 1.5 GHz
- 1-4 GB LPDDR4-3200 SDRAM
- 2.4 GHz und 5.0 GHz Wifi
- Bluetooth 5.0
- 1 GBit Ethernet
- HDMI-, Display-, Camera-Ports
- Audio



Pi GPIO

3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

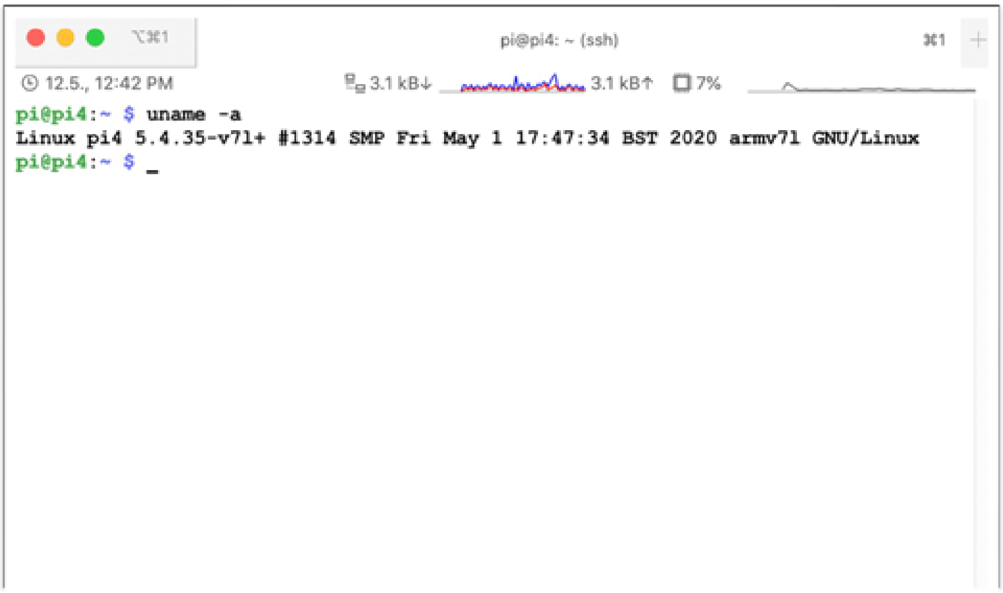
105



Raspberry Pi

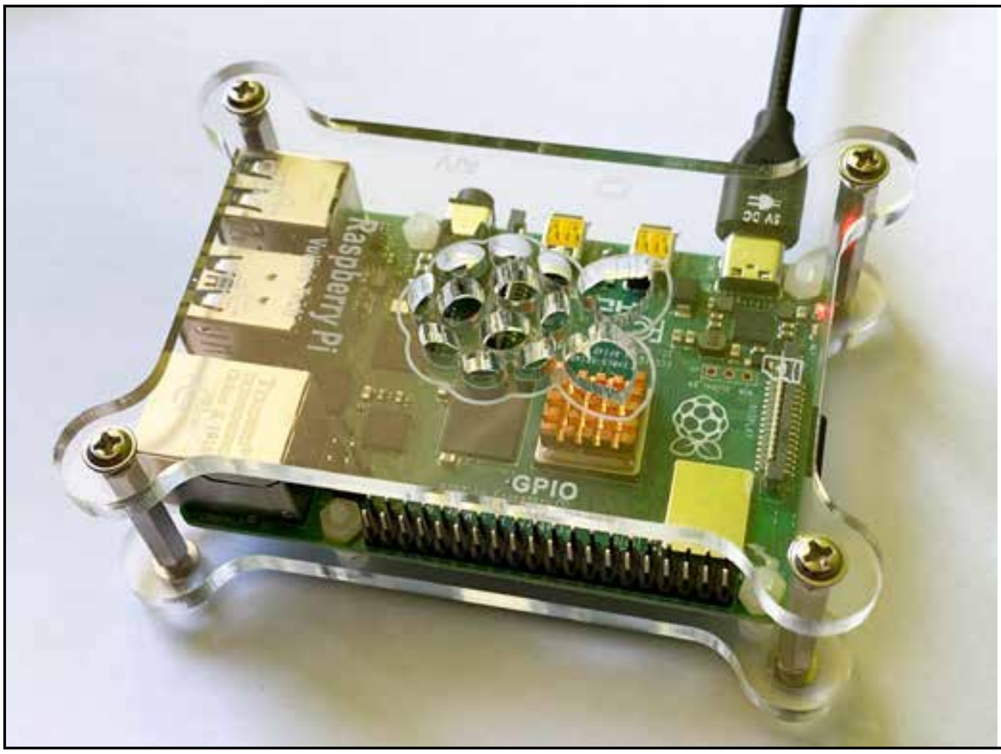
- Raspberry Pi Foundation
- Preisgünstiger Rechner in Cambridge
  - Reaktion auf sinkende Studentenzahlen
- erster Prototyp 2006
- 2000 Geräte erwartet
- > 2 Millionen Geräte (Anfang 2014)

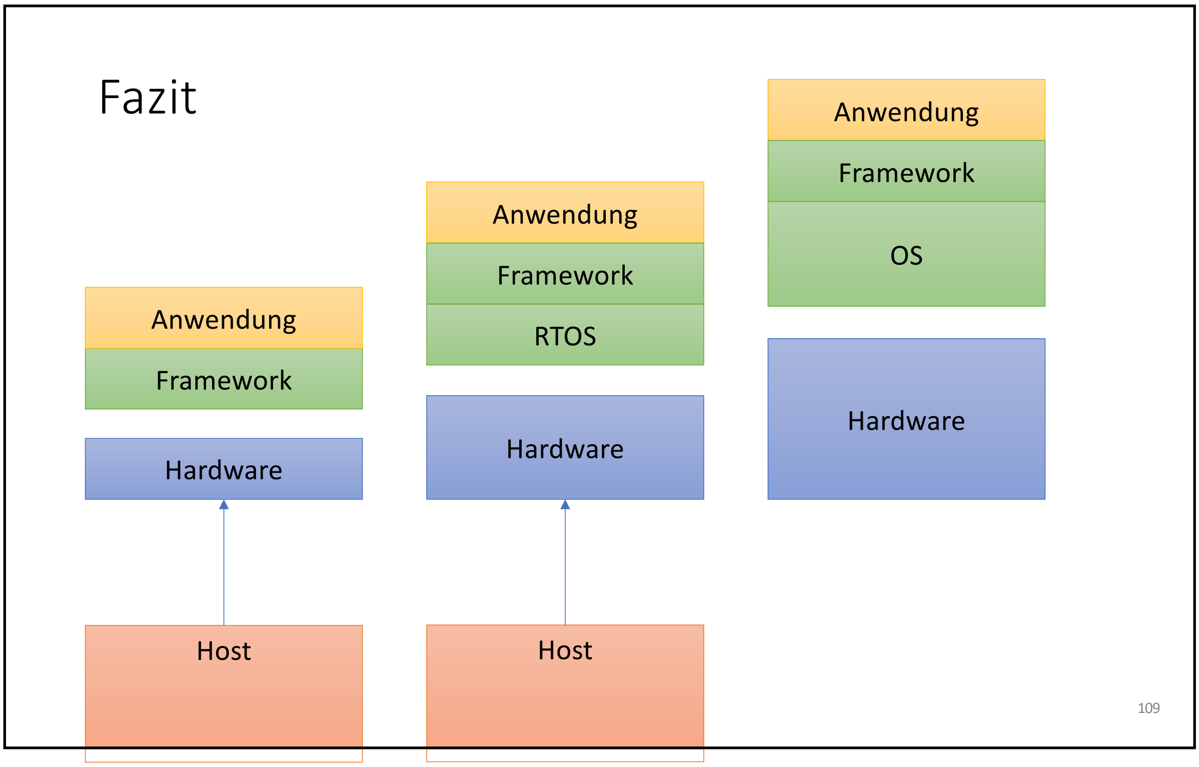
Softwareumfeld: Volles Linux



```
pi@pi4:~ $ uname -a
Linux pi4 5.4.35-v7l+ #1314 SMP Fri May 1 17:47:34 BST 2020 armv7l GNU/Linux
pi@pi4:~ $ _
```

Headless





Fazit

Anwendung

Framework

Hardware

Host

Anwendung

Framework

RTOS

Hardware

Host

Anwendung

Framework

OS

Hardware

109