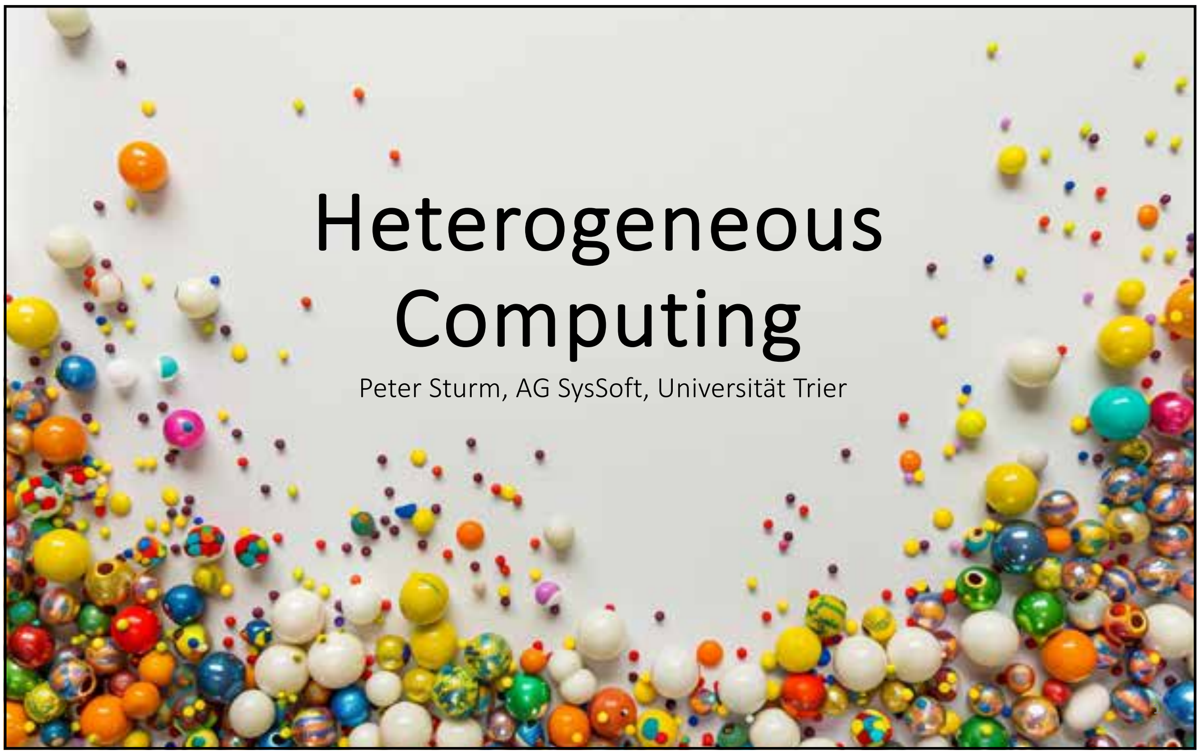
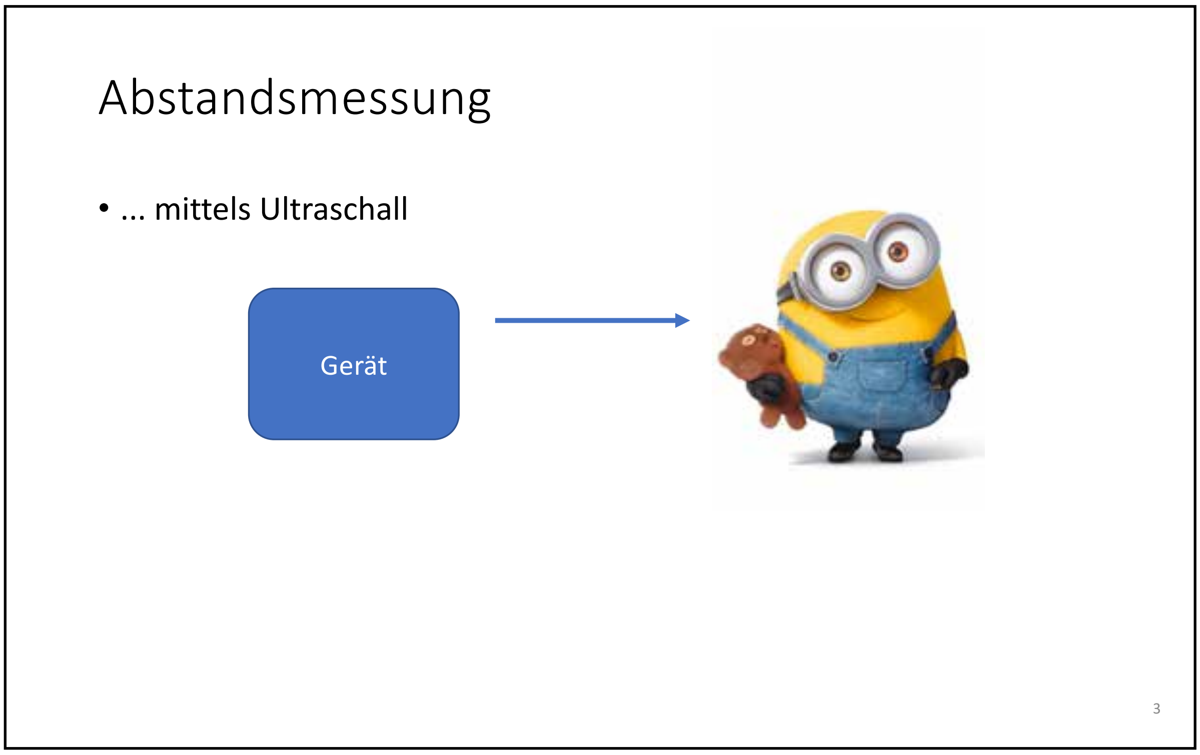


Heterogeneous Computing

Peter Sturm, AG SysSoft, Universität Trier

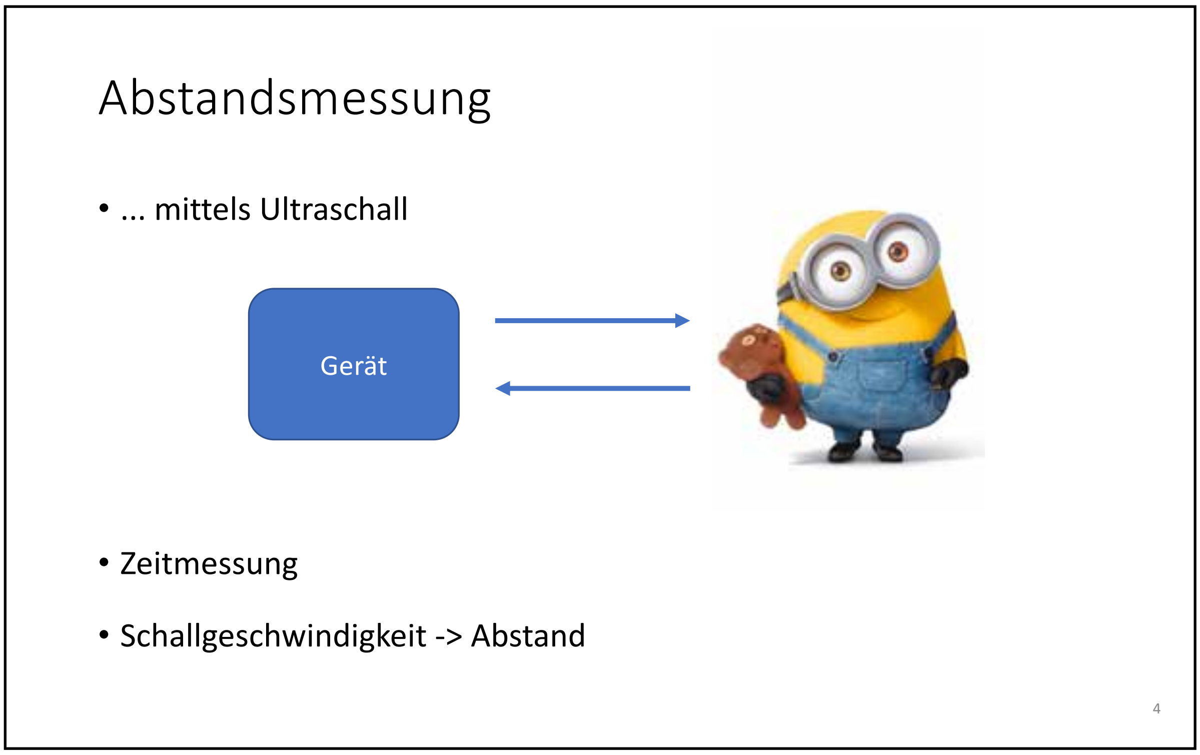
2. Projekt "Abstand"

2



Abstandsmessung

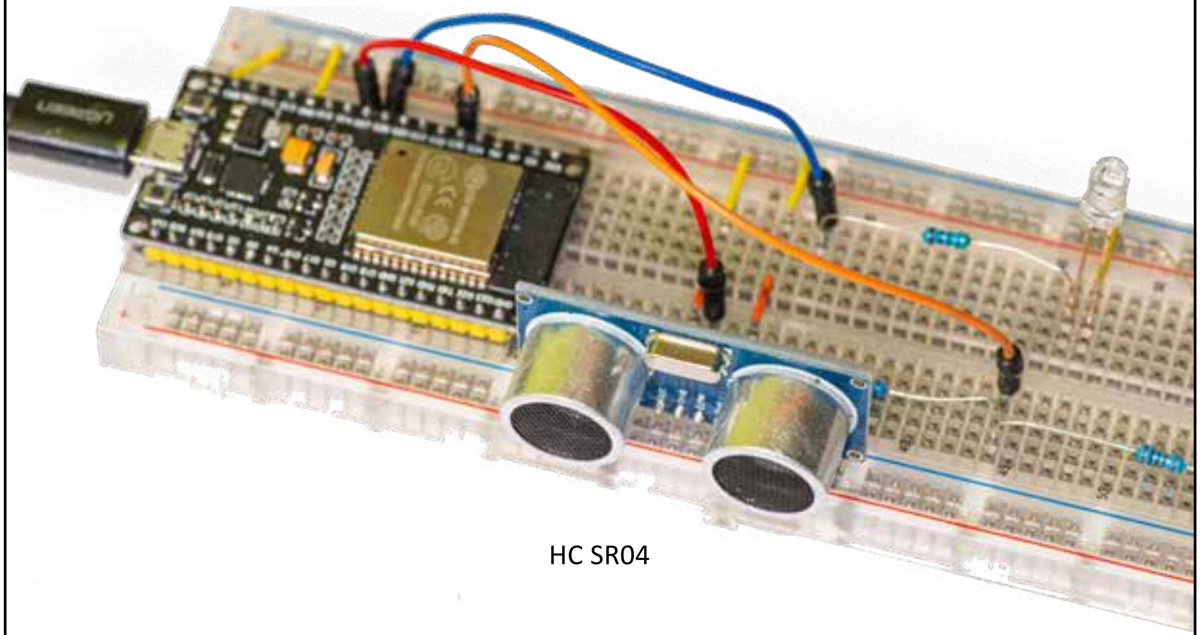
- ... mittels Ultraschall



Abstandsmessung

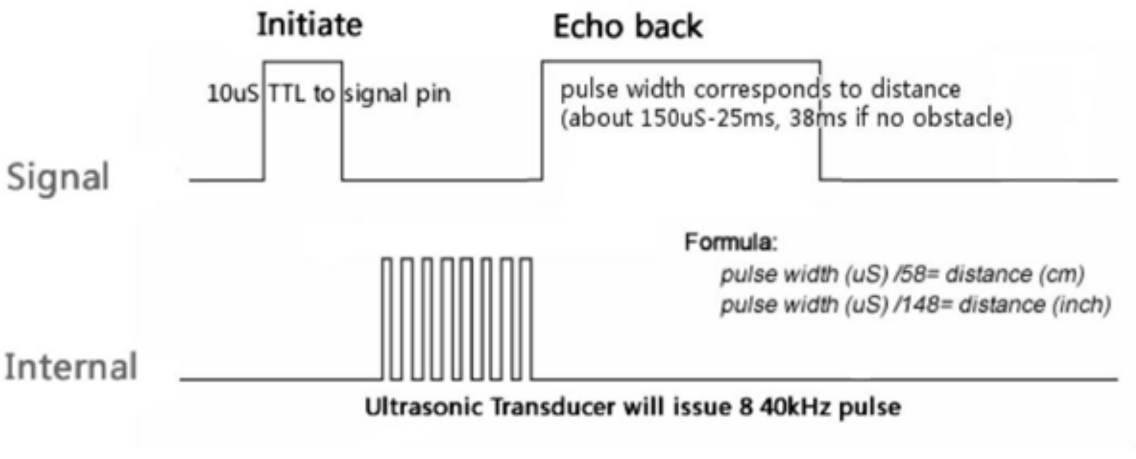
- ... mittels Ultraschall

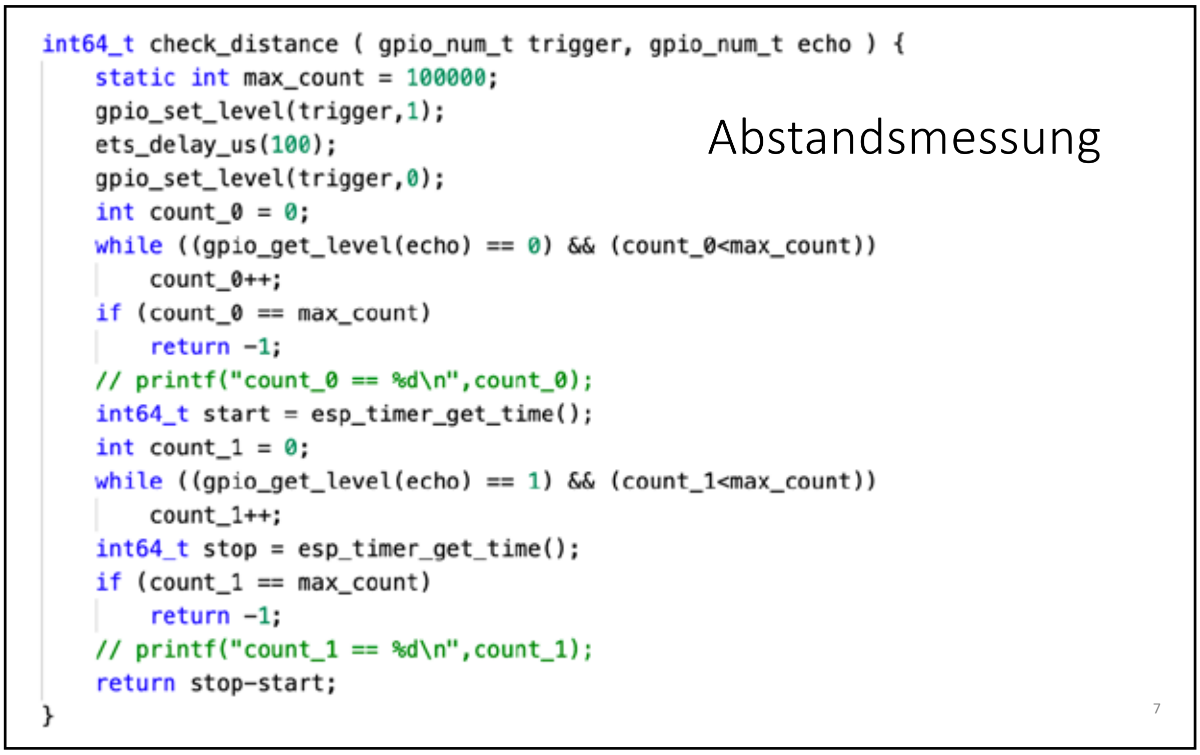
- Zeitmessung
- Schallgeschwindigkeit -> Abstand



HC SR04

Verfahren

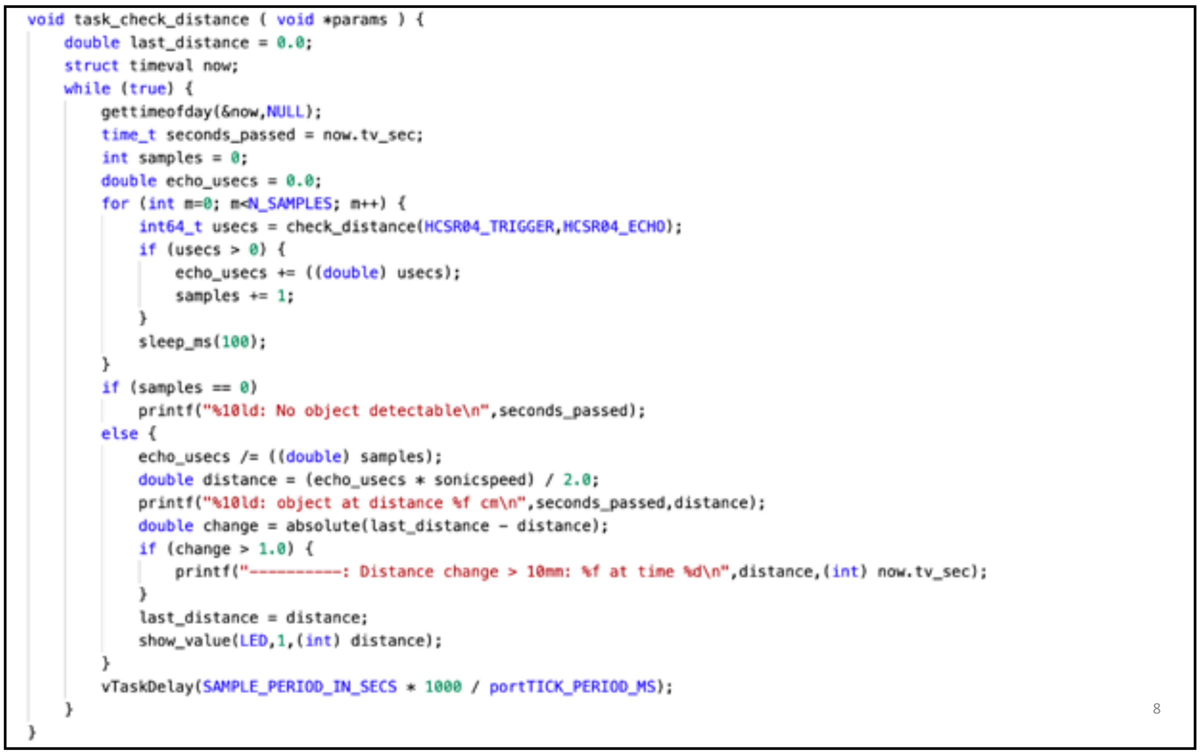

Initiate
Echo back
10uS TTL to signal pin
pulse width corresponds to distance
(about 150uS-25ms, 38ms if no obstacle)
Signal
Formula:
pulse width (uS) /58= distance (cm)
pulse width (uS) /148= distance (inch)
Internal
Ultrasonic Transducer will issue 8 40kHz pulse



Abstandsmessung

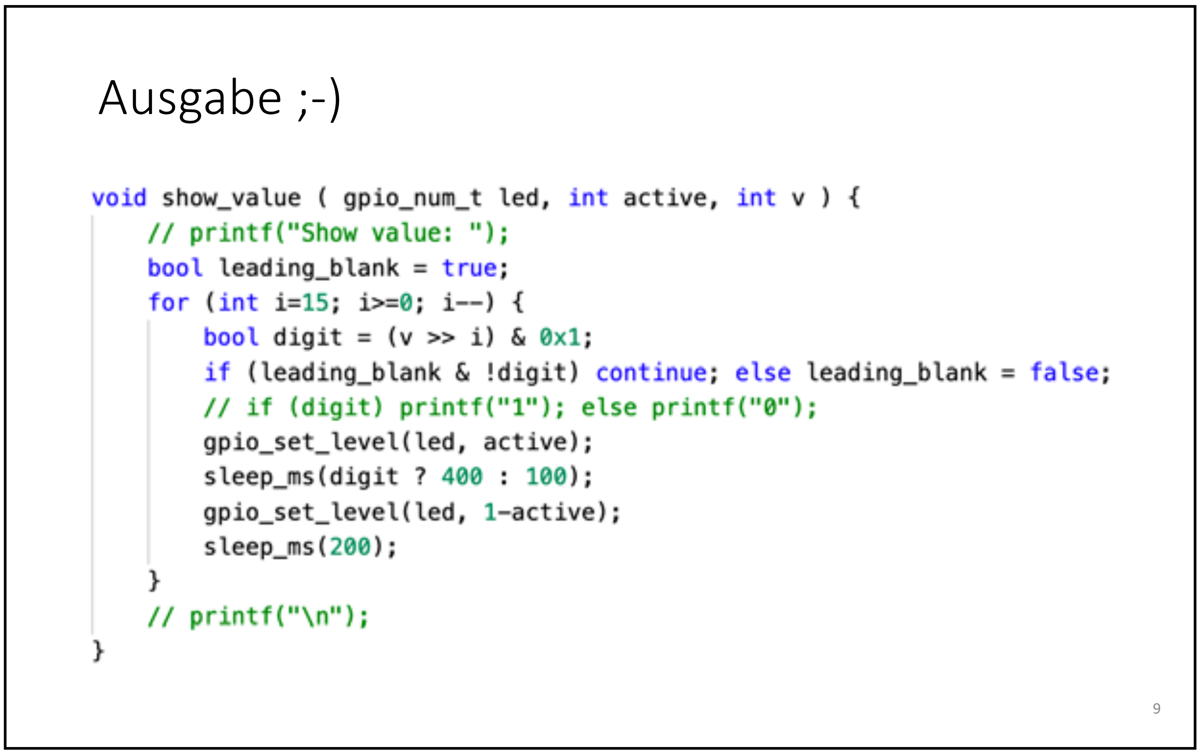
```
int64_t check_distance ( gpio_num_t trigger, gpio_num_t echo ) {
    static int max_count = 100000;
    gpio_set_level(trigger,1);
    ets_delay_us(100);
    gpio_set_level(trigger,0);
    int count_0 = 0;
    while ((gpio_get_level(echo) == 0) && (count_0<max_count))
        count_0++;
    if (count_0 == 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;
}
```

7



```
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);
    }
}
```

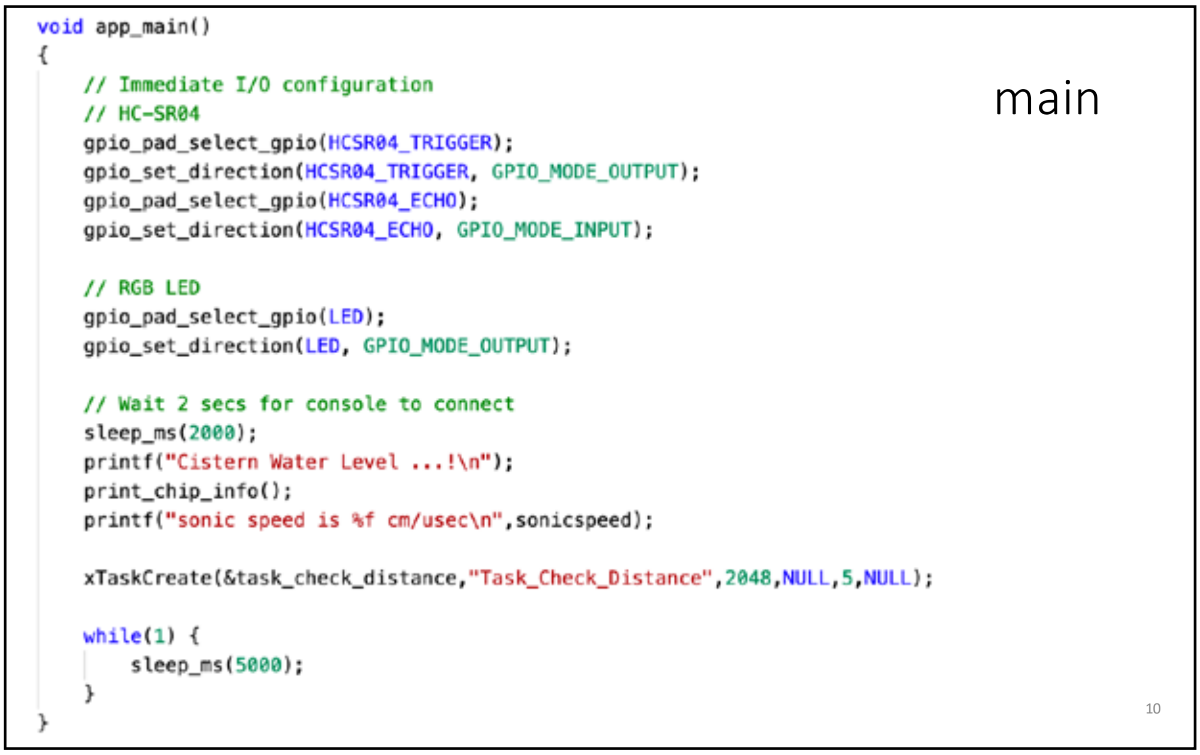
8



Ausgabe ;-)

```c
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");
}
```

9



main

```c
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);
    }
}
```

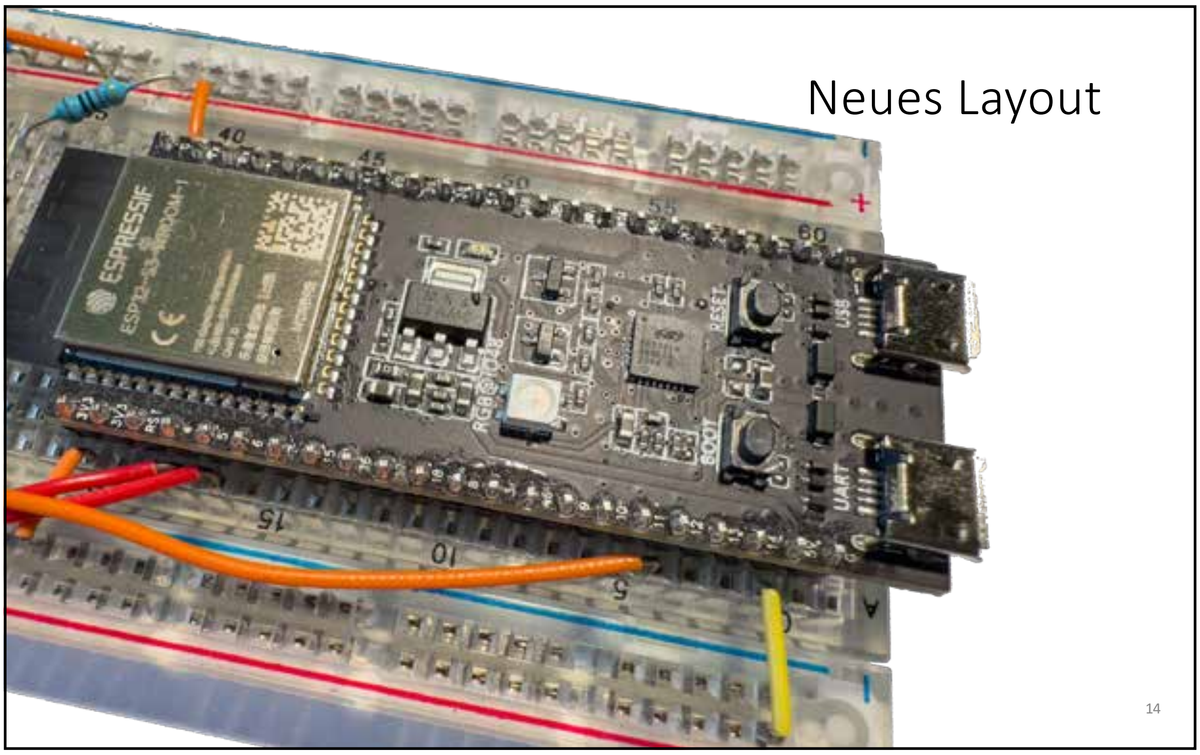
10

Alle Jahre wieder …

Alle 2 Jahre wieder …

Aus dem Tagebuch eines gestressten Möchtegern-Hardware-Entwicklers

13

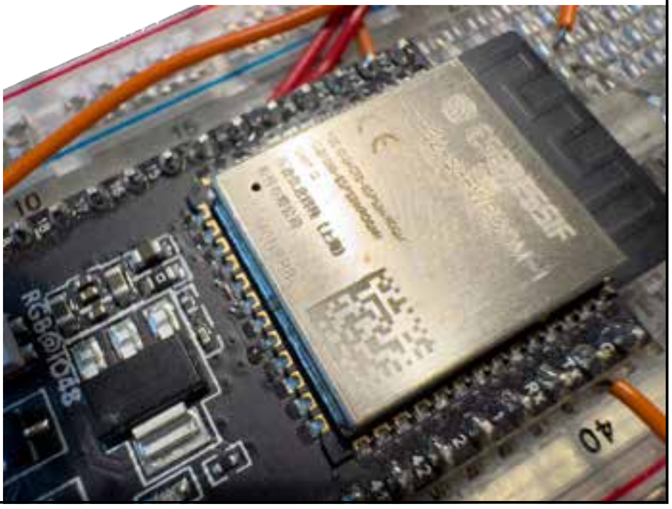


Neues Layout

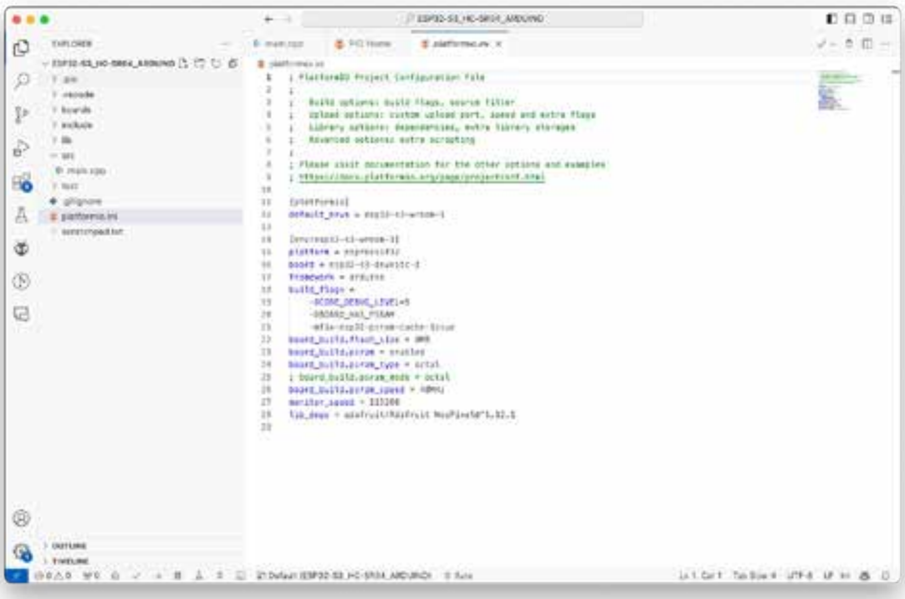
14

Schicker neuer ESP32

- N8 = 8 MB Flash
- R8 = 8 MB PSRAM
  - Extern zum Chip
- RGB LED

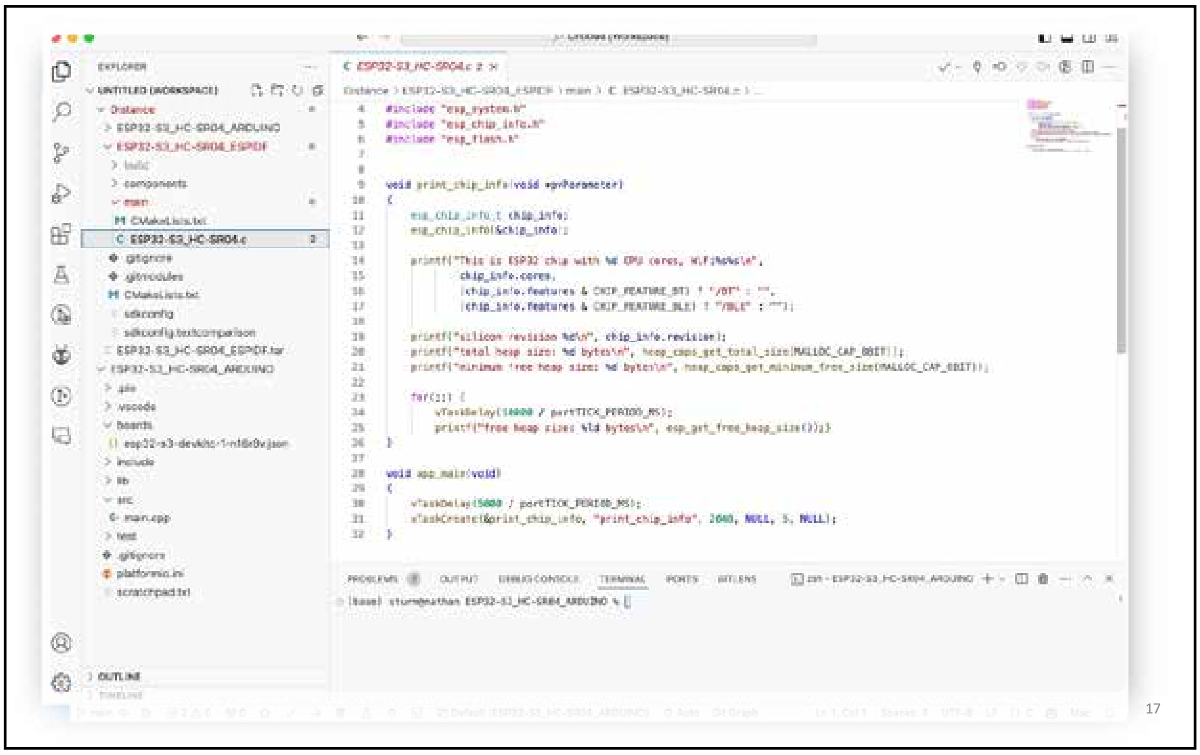


VSCode und Platform.io



```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[platformio]
default_envs = esp32-s3-wroom-1

[env:esp32-s3-wroom-1]
platform = espressif32
board = esp32-s3-devkitc-1
framework = arduino
build_flags =
    -DCORE_DEBUG_LEVEL=5
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue
board_build.flash_size = 8MB
board_build.psram = enabled
board_build.psram_type = octal
; board_build.psram_mode = octal
board_build.psram_speed = 80MHz
monitor_speed = 115200
lib_deps = adafruit/Adafruit NeoPixel@^1.12.1
```

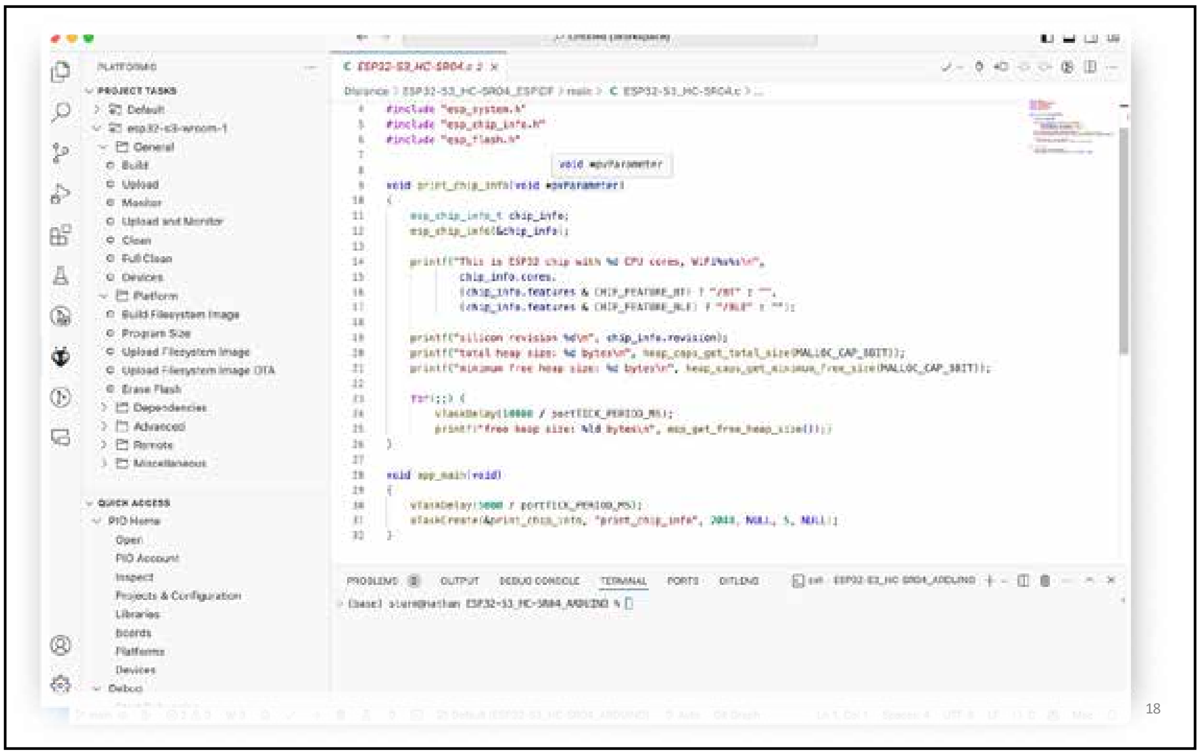
16



```c
#include "esp_system.h"
#include "esp_chip_info.h"
#include "esp_flash.h"


void print_chip_info(void *pvParameter)
{
    esp_chip_info_t chip_info;
    esp_chip_info(&chip_info);

    printf("This is ESP32 chip with %d CPU cores, WiFi%s%s\n",
            chip_info.cores,
            (chip_info.features & CHIP_FEATURE_BT) ? "/BT" : "",
            (chip_info.features & CHIP_FEATURE_BLE) ? "/BLE" : "");

    printf("silicon revision %d\n", chip_info.revision);
    printf("total heap size: %d bytes\n", heap_caps_get_total_size(MALLOC_CAP_8BIT));
    printf("minimum free heap size: %d bytes\n", heap_caps_get_minimum_free_size(MALLOC_CAP_8BIT));

    for(;;) {
        vTaskDelay(10000 / portTICK_PERIOD_MS);
        printf("free heap size: %ld bytes\n", esp_get_free_heap_size());}
}

void app_main(void)
{
    vTaskDelay(5000 / portTICK_PERIOD_MS);
    xTaskCreate(&print_chip_info, "print_chip_info", 2048, NULL, 5, NULL);
}
```

```
(base) sturm@nathan ESP32-S3_HC-SR04_ARDUINO % 
```

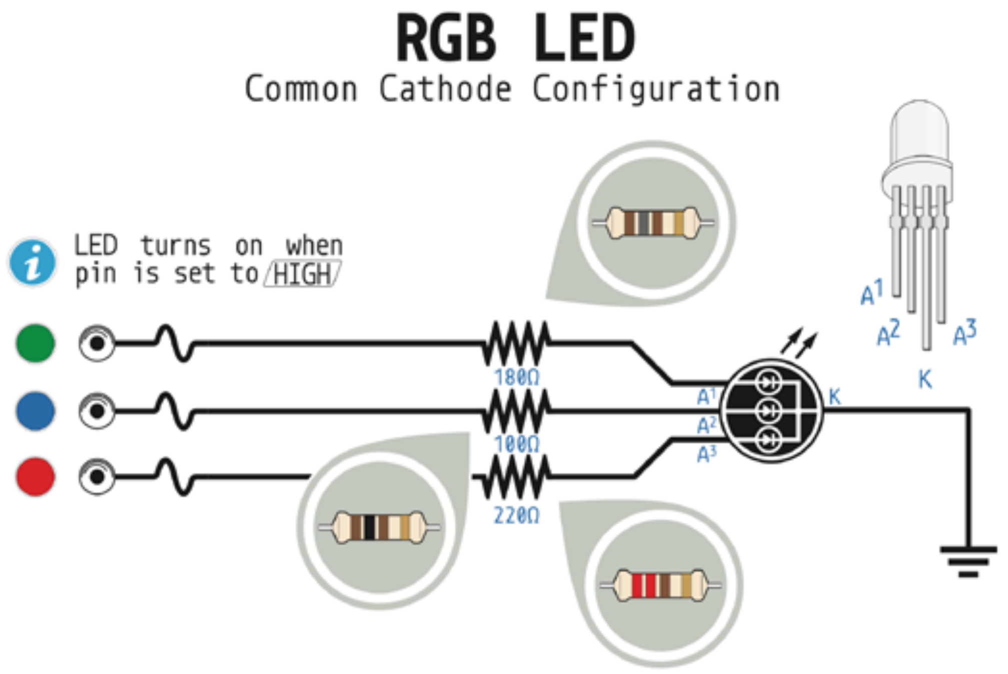
17



```c
#include "esp_system.h"
#include "esp_chip_info.h"
#include "esp_flash.h"


void print_chip_info(void *pvParameter)
{
    esp_chip_info_t chip_info;
    esp_chip_info(&chip_info);

    printf("This is ESP32 chip with %d CPU cores, WiFi%s%s\n",
            chip_info.cores,
            (chip_info.features & CHIP_FEATURE_BT) ? "/BT" : "",
            (chip_info.features & CHIP_FEATURE_BLE) ? "/BLE" : "");

    printf("silicon revision %d\n", chip_info.revision);
    printf("total heap size: %d bytes\n", heap_caps_get_total_size(MALLOC_CAP_8BIT));
    printf("minimum free heap size: %d bytes\n", heap_caps_get_minimum_free_size(MALLOC_CAP_8BIT));

    for(;;) {
        vTaskDelay(10000 / portTICK_PERIOD_MS);
        printf("free heap size: %ld bytes\n", esp_get_free_heap_size());}
}

void app_main(void)
{
    vTaskDelay(5000 / portTICK_PERIOD_MS);
    xTaskCreate(&print_chip_info, "print_chip_info", 2048, NULL, 5, NULL);
}
```

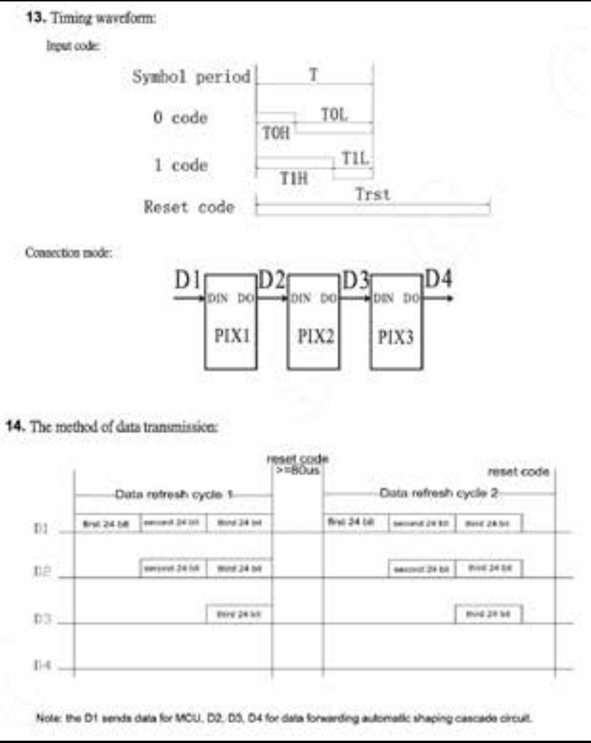
```
(base) sturm@nathan ESP32-S3_HC-SR04_ARDUINO % 
```

18


RGB LED
Common Cathode Configuration
LED turns on when pin is set to HIGH
180Ω
100Ω
220Ω
A1
A2
A3
K

19

SK6812 war es ☹



13. Timing waveform:

Input code:

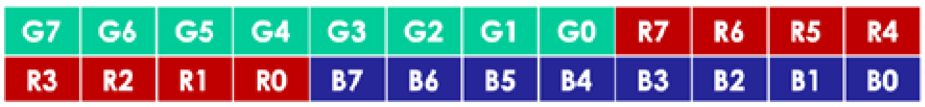
Connection mode:

14. The method of data transmission:

Note: the D1 sends data for MCU, D2, D3, D4 for data forwarding automatic shaping cascade circuit.

20

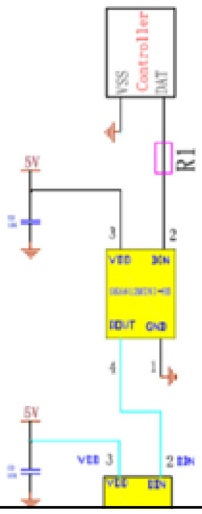
15. The data structure of 24bit:



G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4
R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: high starting, in order to send data (G7 - G6 - ....B0)

16. The typical application circuit:

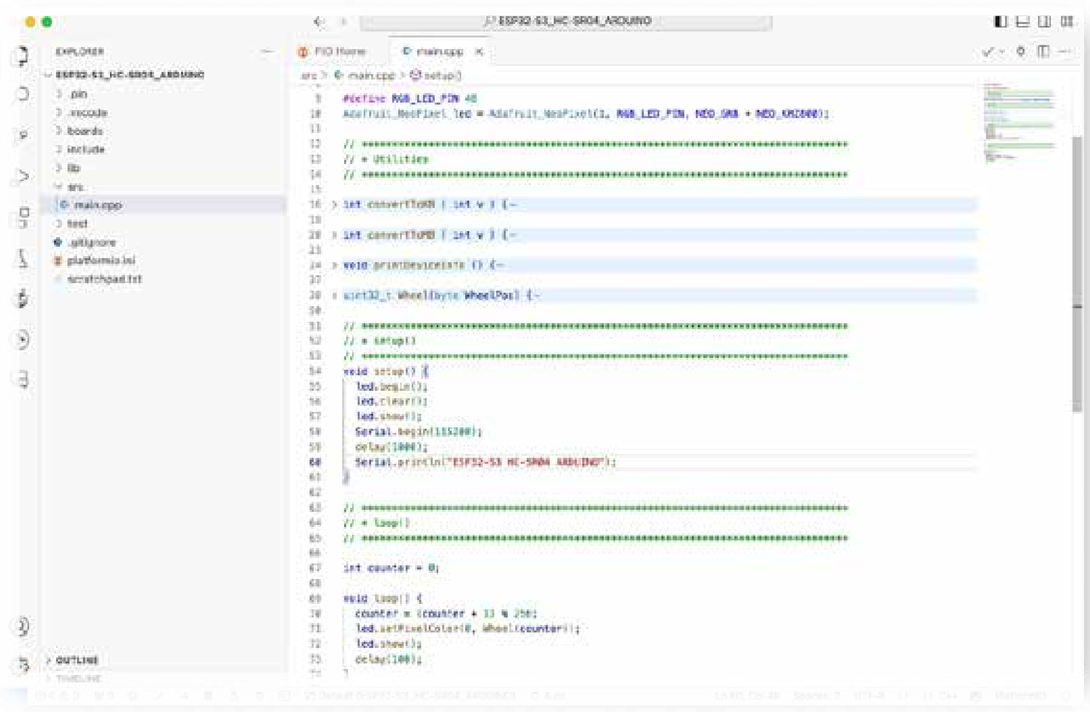


21

Theoretisch ...

- ... kann ich das programmieren, aber ...
- Suche nach Bibliothek
  - Nich so ergiebig im ESP-IDF Framework
  - Fehlerhafte Konfiguration in ESP-IDF: #include <cassert> klappt nicht
- ESP-IDF von Hand installiert (ohne Platform.io)
  - Auch <cassert>-Fehler
- ESP-IDF von Hand auf Raspberry Pi mit Linux installiert
  - Auch <cassert>-Fehler

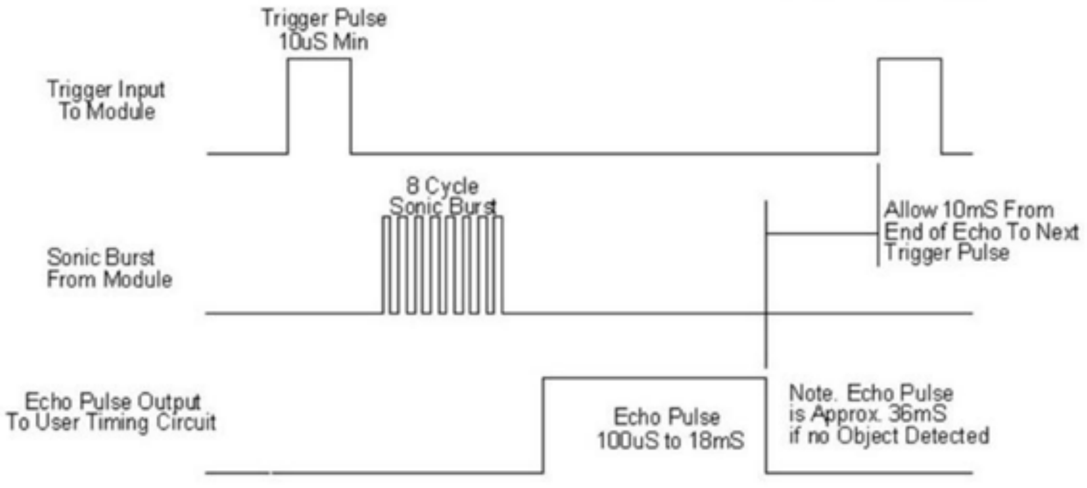
22



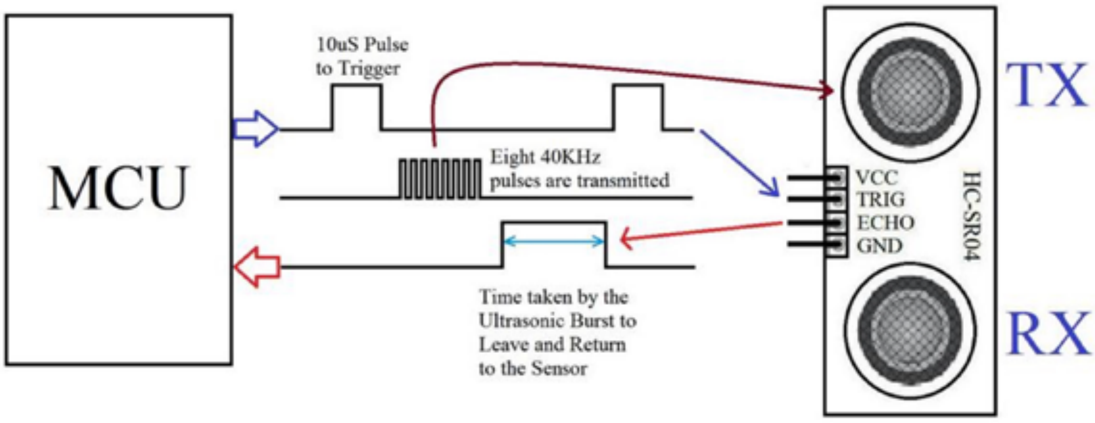
```
#define RGB_LED_PIN 48
Adafruit_NeoPixel led = Adafruit_NeoPixel(1, RGB_LED_PIN, NEO_GRB + NEO_KHZ800);

// ******************************************************************************
// * Utilities
// ******************************************************************************

int convertToKB ( int v ) {…

int convertToMB ( int v ) {…

void printDeviceInfo () {…

uint32_t Wheel(byte WheelPos) {…

// ******************************************************************************
// * setup()
// ******************************************************************************
void setup() {
  led.begin();
  led.clear();
  led.show();
  Serial.begin(115200);
  delay(1000);
  Serial.println("ESP32-S3 HC-SR04 ARDUINO");
}

// ******************************************************************************
// * loop()
// ******************************************************************************

int counter = 0;

void loop() {
  counter = (counter + 1) % 256;
  led.setPixelColor(0, Wheel(counter));
  led.show();
  delay(100);
}
```

23

HC-SR04 Timing

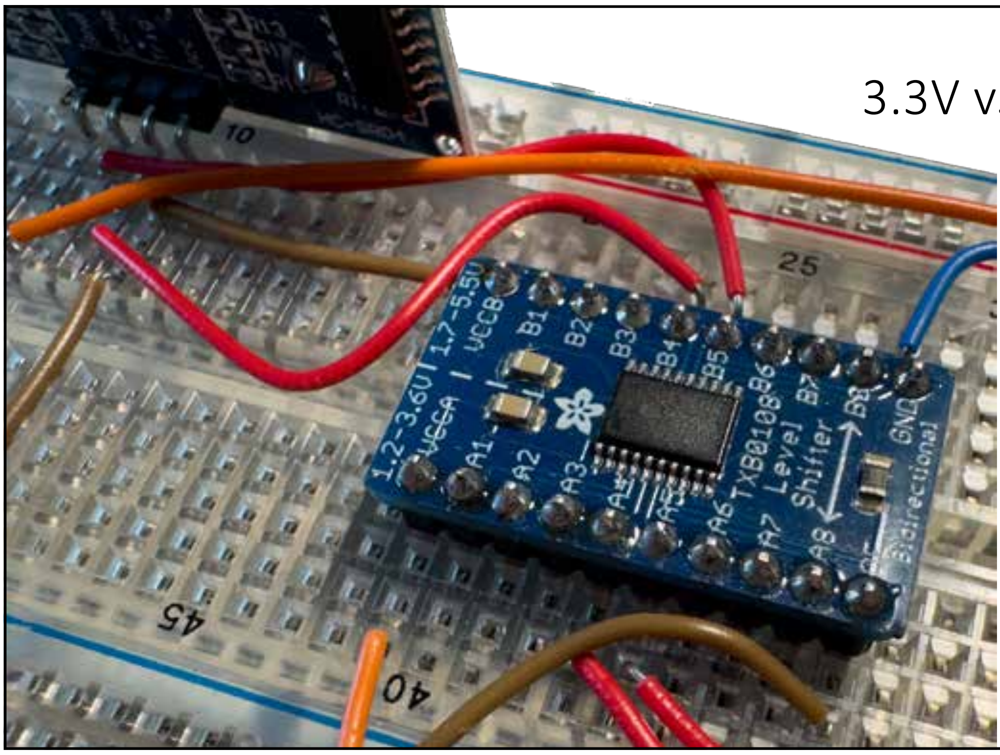


24

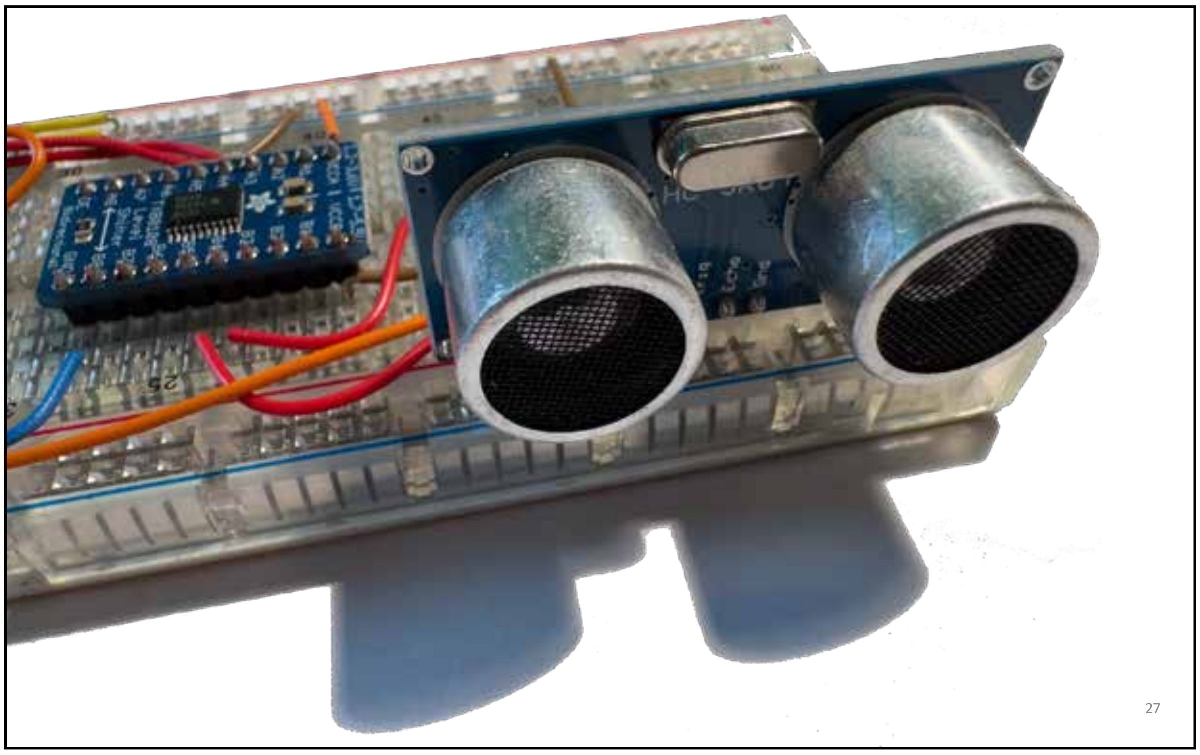

MCU
10uS Pulse to Trigger
Eight 40KHz pulses are transmitted
Time taken by the Ultrasonic Burst to Leave and Return to the Sensor
TX
RX
VCC
TRIG
ECHO
GND
HC-SR04

25

3.3V vs. 5V

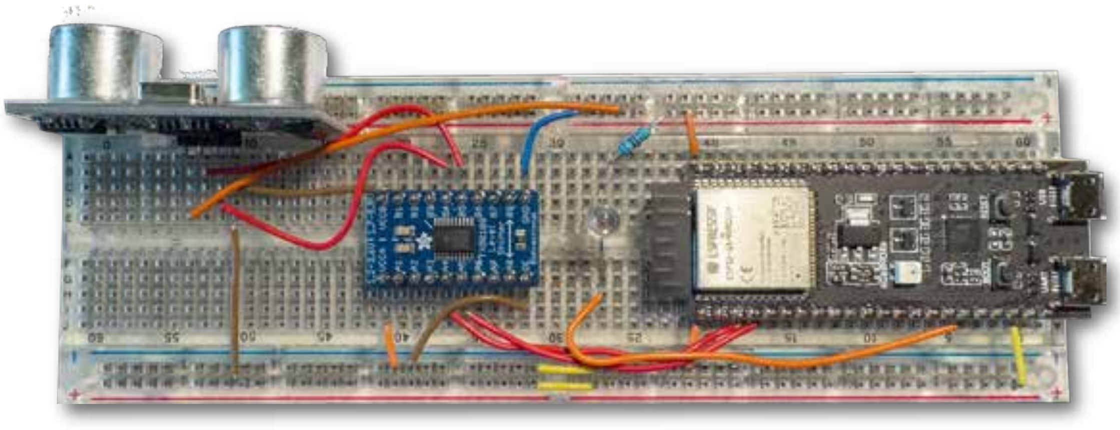


26

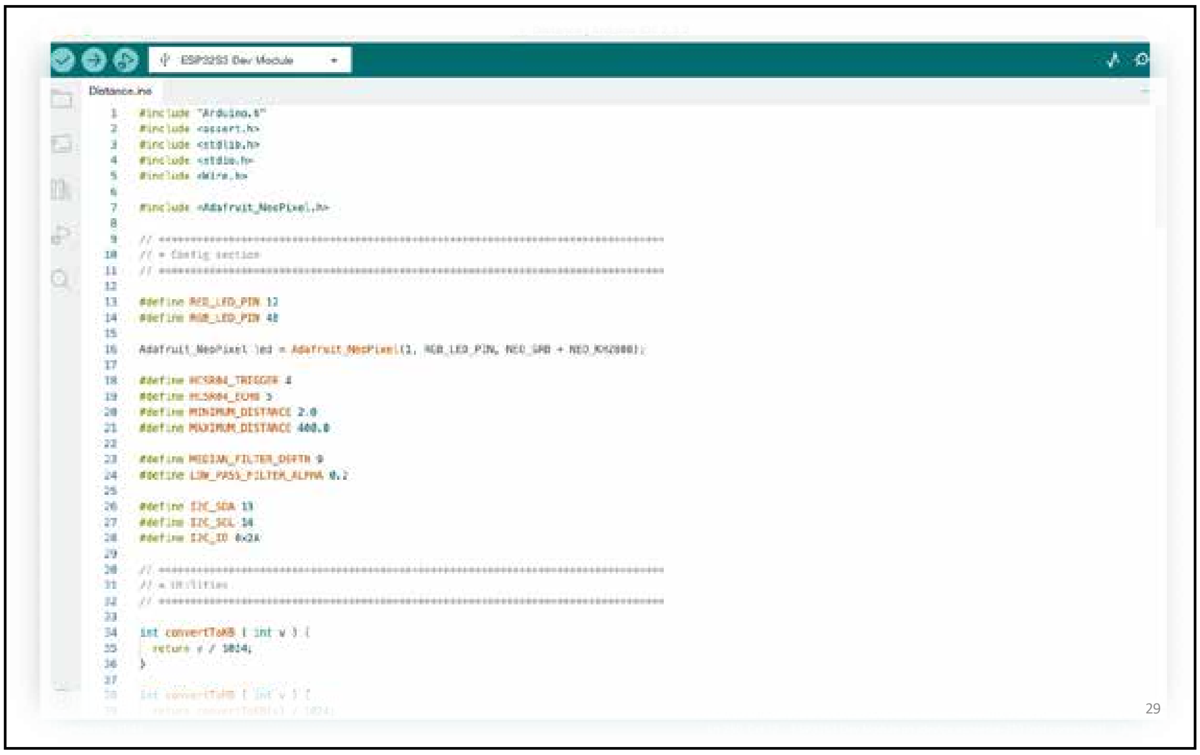


27

Fertiger Prototyp

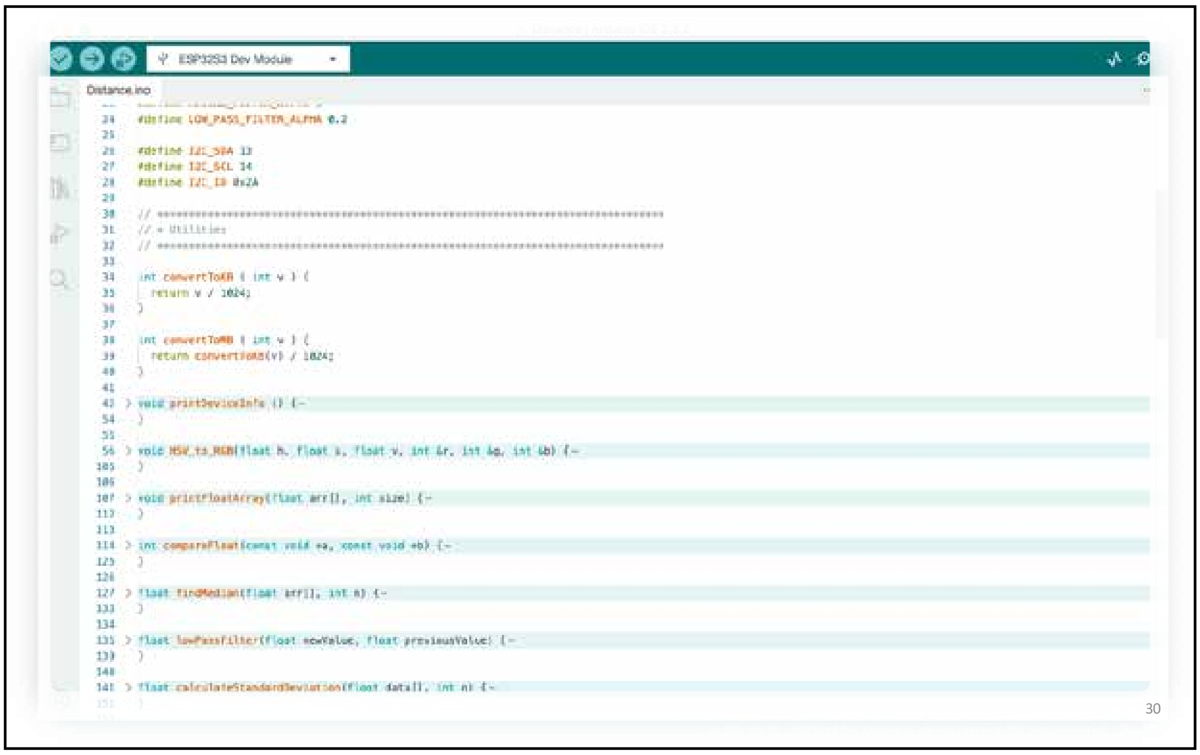


28



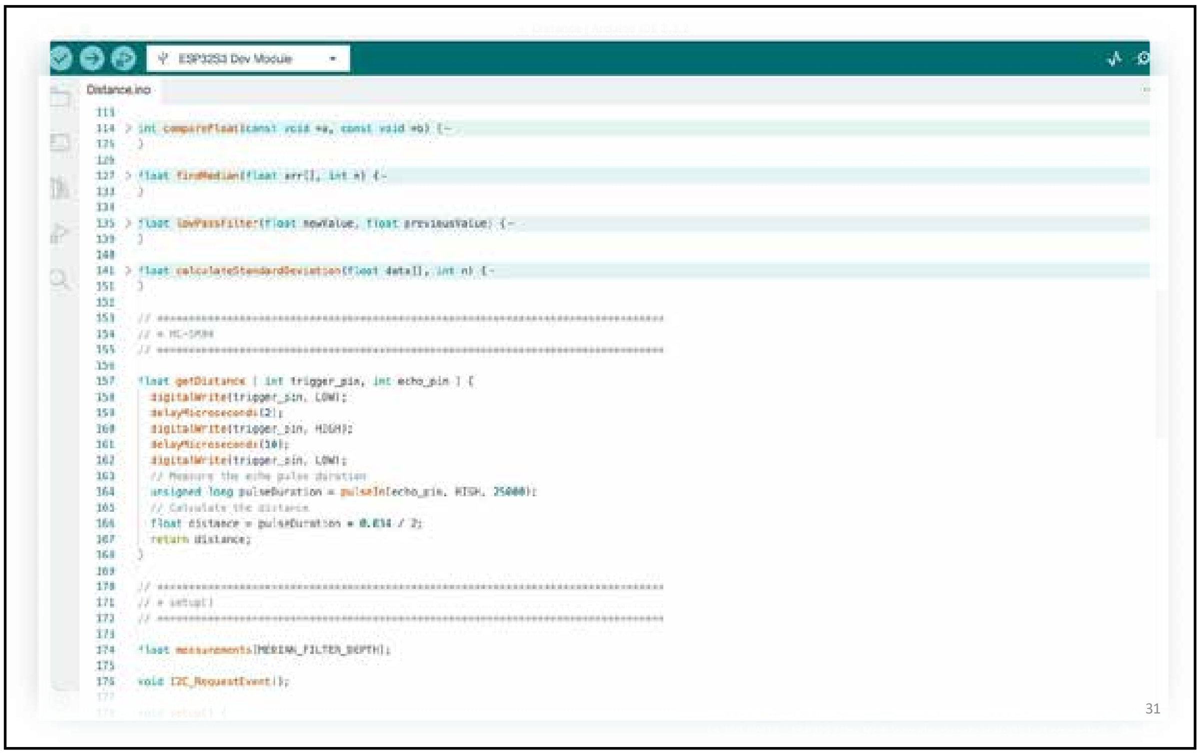
```
#include "Arduino.h"
#include <assert.h>
#include <stdlib.h>
#include <stdio.h>
#include <Wire.h>

#include <Adafruit_NeoPixel.h>

// ===========================================================================
// = Config section
// ===========================================================================

#define RED_LED_PIN 12
#define RGB_LED_PIN 48

Adafruit_NeoPixel led = Adafruit_NeoPixel(1, RGB_LED_PIN, NEO_GRB + NEO_KHZ800);

#define HCSR04_TRIGGER 4
#define HCSR04_ECHO 5
#define MINIMUM_DISTANCE 2.0
#define MAXIMUM_DISTANCE 400.0

#define MEDIAN_FILTER_DEPTH 9
#define LOW_PASS_FILTER_ALPHA 0.2

#define I2C_SDA 13
#define I2C_SCL 14
#define I2C_ID 0x2A

// ===========================================================================
// = Utilities
// ===========================================================================

int convertToKB ( int v ) {
  return v / 1024;
}

int convertToMB ( int v ) {
  return convertToKB(v) / 1024;
```

29



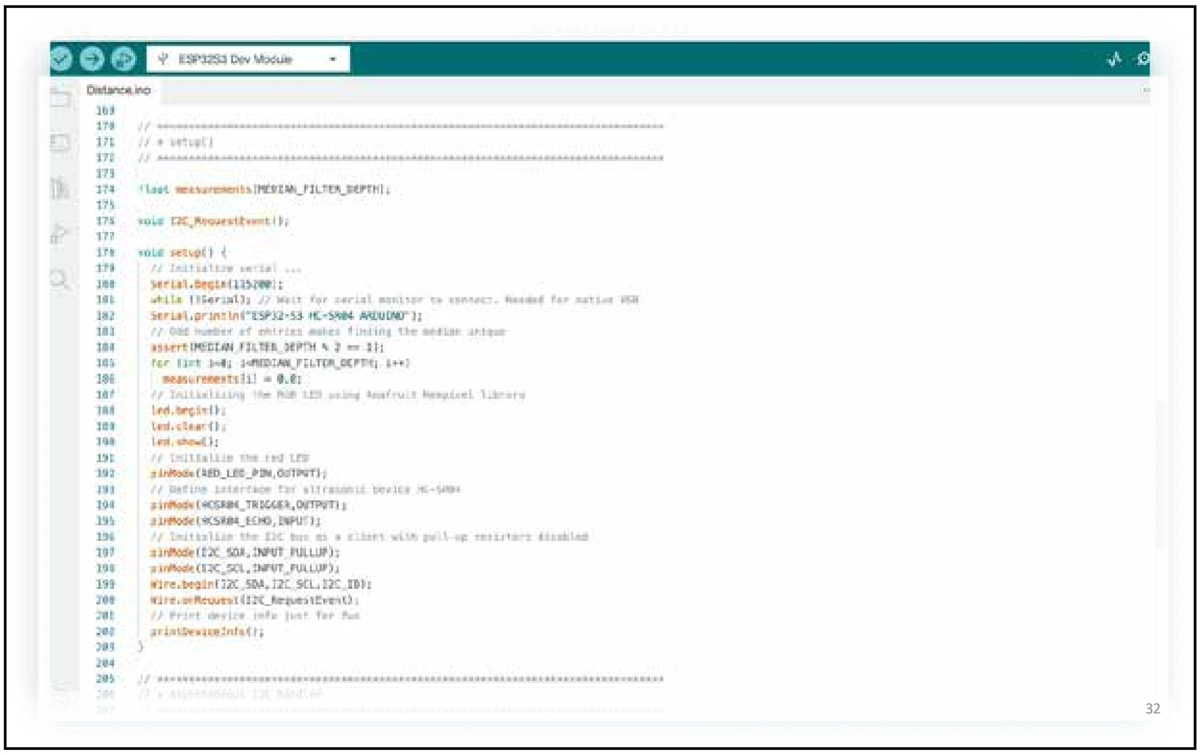
```
#define LOW_PASS_FILTER_ALPHA 0.2

#define I2C_SDA 13
#define I2C_SCL 14
#define I2C_ID 0x2A

// ===========================================================================
// = Utilities
// ===========================================================================

int convertToKB ( int v ) {
  return v / 1024;
}

int convertToMB ( int v ) {
  return convertToKB(v) / 1024;
}

> void printDeviceInfo () {⋯
}

> void HSV_to_RGB(float h, float s, float v, int &r, int &g, int &b) {⋯
}

> void printFloatArray(float arr[], int size) {⋯
}

> int compareFloat(const void *a, const void *b) {⋯
}

> float findMedian(float arr[], int n) {⋯
}

> float lowPassFilter(float newValue, float previousValue) {⋯
}

> float calculateStandardDeviation(float data[], int n) {⋯
}
```

30



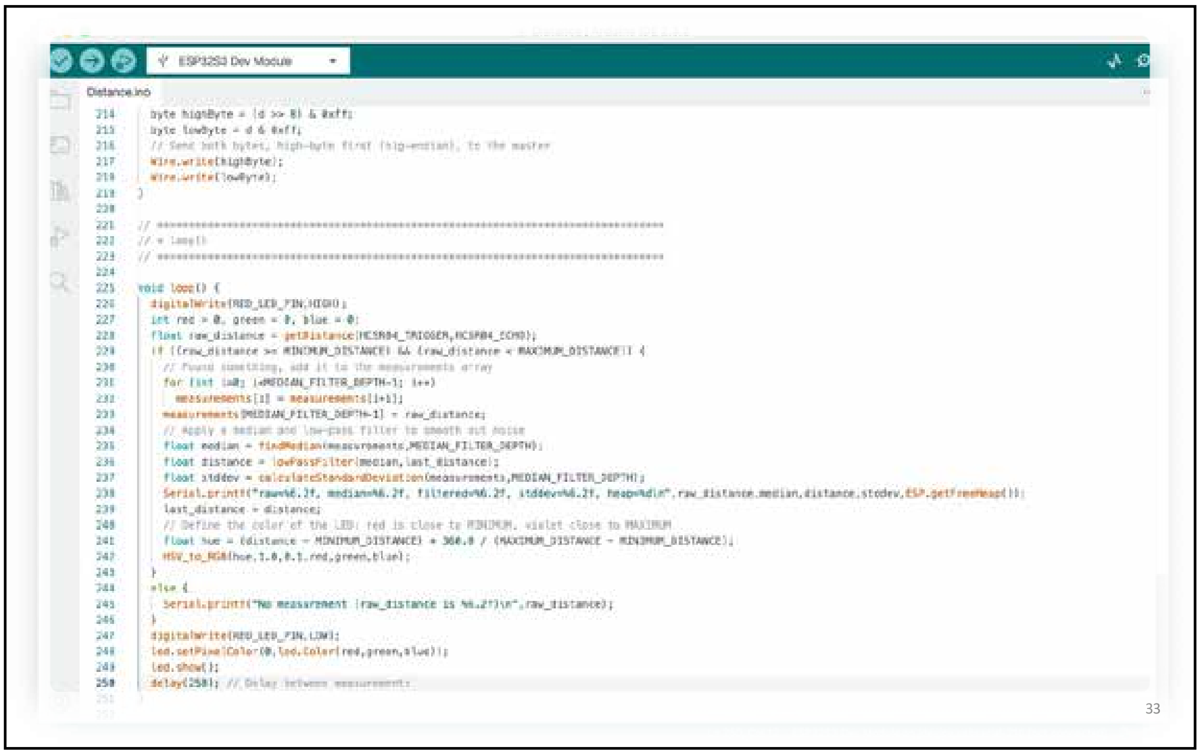
```
int compareFloat(const void *a, const void *b) {…
}

float findMedian(float arr[], int n) {…
}

float lowPassFilter(float newValue, float previousValue) {…
}

float calculateStandardDeviation(float data[], int n) {…
}

// ***********************************************************************************
// * HC-SR04
// ***********************************************************************************

float getDistance ( int trigger_pin, int echo_pin ) {
  digitalWrite(trigger_pin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger_pin, LOW);
  // Measure the echo pulse duration
  unsigned long pulseDuration = pulseIn(echo_pin, HIGH, 25000);
  // Calculate the distance
  float distance = pulseDuration * 0.034 / 2;
  return distance;
}

// ***********************************************************************************
// * setup()
// ***********************************************************************************

float measurements[MEDIAN_FILTER_DEPTH];

void I2C_RequestEvent();

void setup() {
```

31



```
// ***********************************************************************************
// * setup()
// ***********************************************************************************

float measurements[MEDIAN_FILTER_DEPTH];

void I2C_RequestEvent();

void setup() {
  // Initialize serial ...
  Serial.begin(115200);
  while (!Serial); // Wait for serial monitor to connect. Needed for native USB
  Serial.println("ESP32-S3 HC-SR04 ARDUINO");
  // Odd number of entries makes finding the median unique
  assert(MEDIAN_FILTER_DEPTH % 2 == 1);
  for (int i=0; i<MEDIAN_FILTER_DEPTH; i++)
    measurements[i] = 0.0;
  // Initializing the RGB LED using Adafruit Neopixel library
  led.begin();
  led.clear();
  led.show();
  // Initialize the red LED
  pinMode(RED_LED_PIN,OUTPUT);
  // Define interface for ultrasonic device HC-SR04
  pinMode(HCSR04_TRIGGER,OUTPUT);
  pinMode(HCSR04_ECHO,INPUT);
  // Initialize the I2C bus as a client with pull-up resistors disabled
  pinMode(I2C_SDA,INPUT_PULLUP);
  pinMode(I2C_SCL,INPUT_PULLUP);
  Wire.begin(I2C_SDA,I2C_SCL,I2C_ID);
  Wire.onRequest(I2C_RequestEvent);
  // Print device info just for fun
  printDeviceInfo();
}

// ***********************************************************************************
// * asynchronous I2C handler
// ***********************************************************************************
```

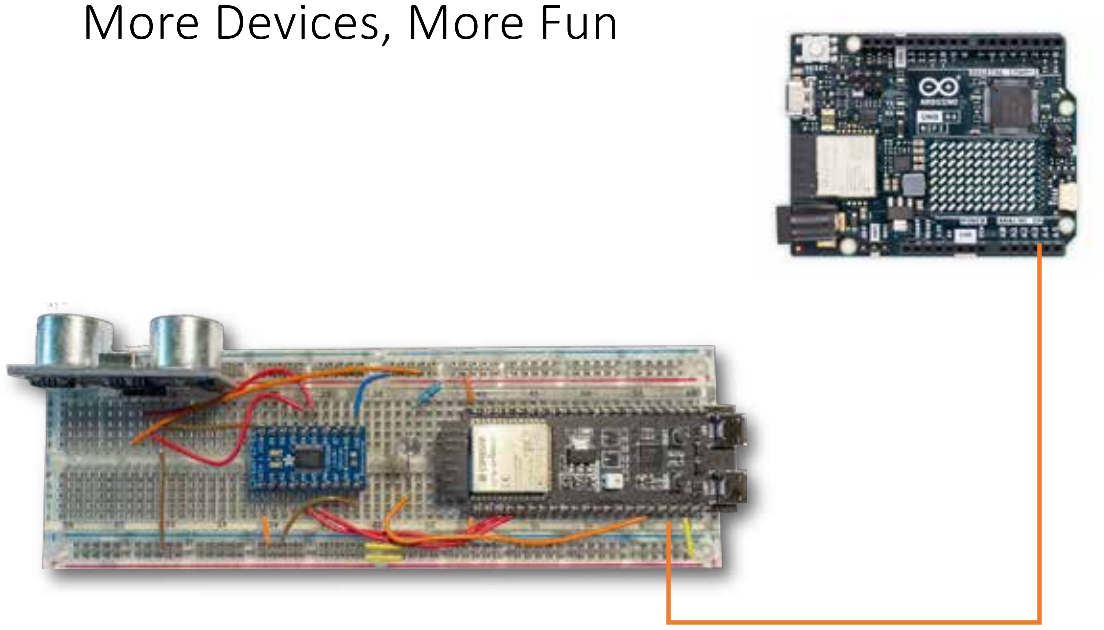
32



```
  byte highByte = (d >> 8) & 0xff;
  byte lowByte = d & 0xff;
  // Send both bytes, high-byte first (big-endian), to the master
  Wire.write(highByte);
  Wire.write(lowByte);
}

// ==========================================================================
// = loop()
// ==========================================================================

void loop() {
  digitalWrite(RED_LED_PIN,HIGH);
  int red = 0, green = 0, blue = 0;
  float raw_distance = getDistance(HCSR04_TRIGGER,HCSR04_ECHO);
  if ((raw_distance >= MINIMUM_DISTANCE) && (raw_distance < MAXIMUM_DISTANCE)) {
    // Found something, add it to the measurements array
    for (int i=0; i<MEDIAN_FILTER_DEPTH-1; i++)
      measurements[i] = measurements[i+1];
    measurements[MEDIAN_FILTER_DEPTH-1] = raw_distance;
    // Apply a median and low-pass filter to smooth out noise
    float median = findMedian(measurements,MEDIAN_FILTER_DEPTH);
    float distance = lowPassFilter(median,last_distance);
    float stddev = calculateStandardDeviation(measurements,MEDIAN_FILTER_DEPTH);
    Serial.printf("raw=%6.2f, median=%6.2f, filtered=%6.2f, stddev=%6.2f, heap=%d\n",raw_distance,median,distance,stddev,ESP.getFreeHeap());
    last_distance = distance;
    // Define the color of the LED: red is close to MINIMUM, violet close to MAXIMUM
    float hue = (distance - MINIMUM_DISTANCE) * 360.0 / (MAXIMUM_DISTANCE - MINIMUM_DISTANCE);
    HSV_to_RGB(hue,1.0,0.1,red,green,blue);
  }
  else {
    Serial.printf("No measurement (raw_distance is %6.2f)\n",raw_distance);
  }
  digitalWrite(RED_LED_PIN,LOW);
  led.setPixelColor(0,led.Color(red,green,blue));
  led.show();
  delay(250); // Delay between measurements
}
```

33

More Devices, More Fun



34

I2C

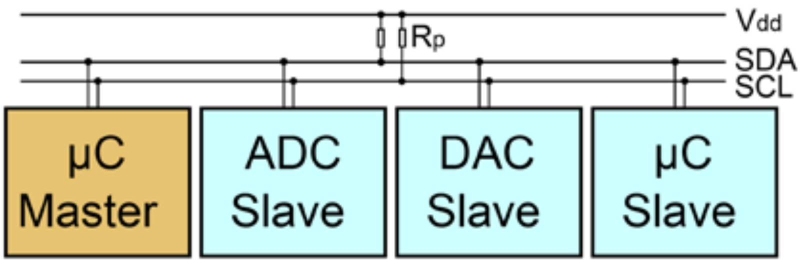
35

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

36

Aufbau



- Master initiiert Senden und Empfangen
- Geräte haben Adresse

37

Kommunikation

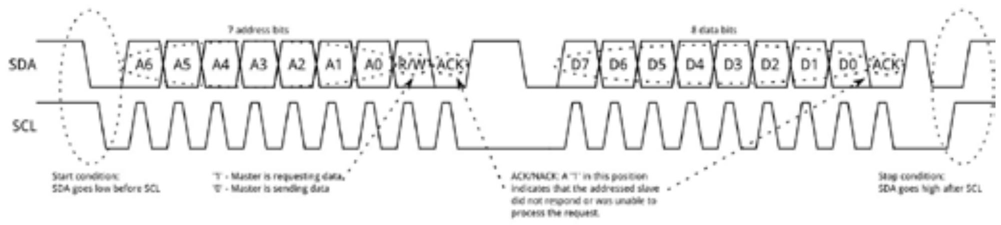

7 address bits
8 data bits
SDA
SCL
A6
A5
A4
A3
A2
A1
A0
R/W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Start condition:
SDA goes low before SCL
'1' - Master is requesting data,
'0' - Master is sending data
ACK/NACK: A '1' in this position indicates that the addressed slave did not respond or was unable to process the request.
Stop condition:
SDA goes high after SCL

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

38

Multi-Master

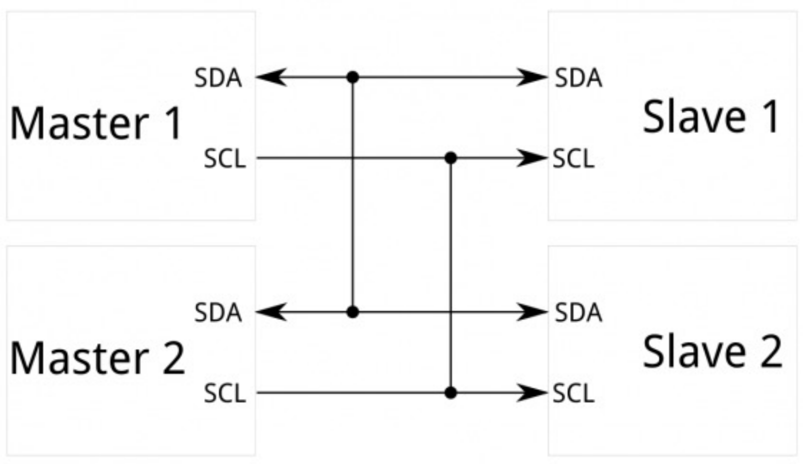
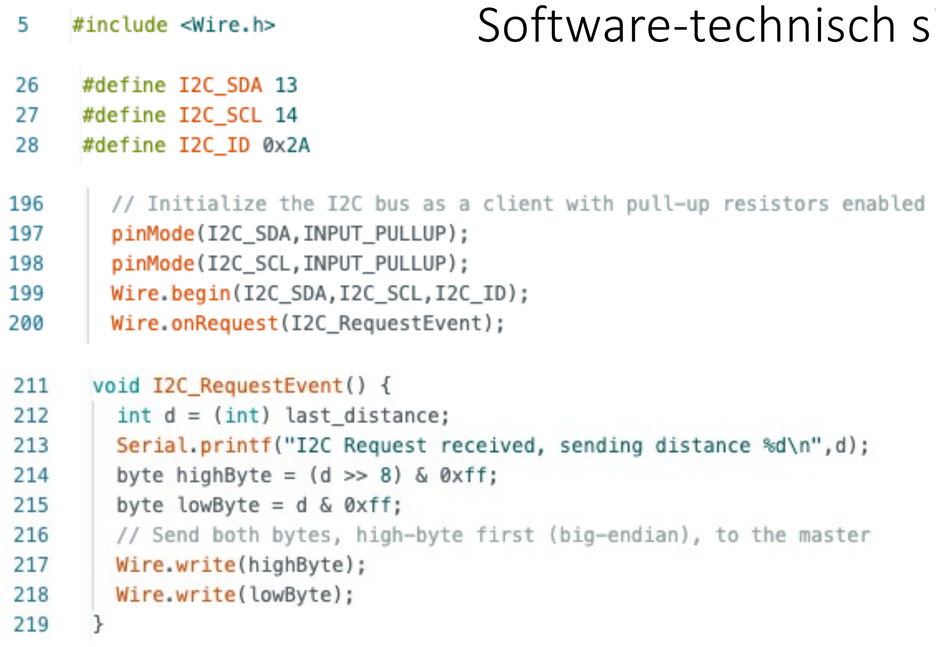


Abbildung: https://learn.sparkfun.com/tutorials/i2c

Software-technisch simpel



```
#include <Wire.h>

#define I2C_SDA 13
#define I2C_SCL 14
#define I2C_ID 0x2A

  // Initialize the I2C bus as a client with pull-up resistors enabled
  pinMode(I2C_SDA,INPUT_PULLUP);
  pinMode(I2C_SCL,INPUT_PULLUP);
  Wire.begin(I2C_SDA,I2C_SCL,I2C_ID);
  Wire.onRequest(I2C_RequestEvent);

void I2C_RequestEvent() {
  int d = (int) last_distance;
  Serial.printf("I2C Request received, sending distance %d\n",d);
  byte highByte = (d >> 8) & 0xff;
  byte lowByte = d & 0xff;
  // Send both bytes, high-byte first (big-endian), to the master
  Wire.write(highByte);
  Wire.write(lowByte);
}
```

Funktioniert noch nicht ...

- Direkt verbunden
  - Argh! 3.3V I2C am ESP32 auf 5V I2C am Arduino
  - Gefahr defekter Pins am ESP32
- TXB0108 nicht für Open Drain geeignet
  - TXS0108E besorgt
- Wire.h auf ESP32 scheint nur mit Pin 21 und Pin 22 zu funktionieren
  - Auf FreeRtos wechseln
  - Logikanalysator einsetzen
  - ...
- Keine Lust mehr ☺

41

Vorerst Schluß!

42