ESP32 Smart Home Prototyping - Drahtlose Kommunikation, LEDs und OLED-Display 03.01.24

Aufgabe 1: zwei LEDs mithilfe eines ESP32 Mikrocontrollers zum blinken bringen

Arbeitsmaterialien

- 2 ESP32 Mikrocontroller + Verbindungskabel
- LEDs
- Widerstände (falls benötigt für LED-Schutz)
- Breadboard und Verbindungskabel
- Computer mit Arduino IDE installiert
- OLED-Display + Verbindungskabel

Arduino IDE vorbereiten

- Downlad und Installation von "VCP Drivers" -> https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
- Arduino IDE donwloaden -> https://www.arduino.cc/en/software
- Menüpunkt "Datei/File" die "Voreinstellungen/Preferences" öffnen
- Boardverwalter-URL für den ESP32 eingetragen: "https://dl.espressif.com/dl/package_esp32_index.json" -> "OK"
- NodeMCU ESP32 nun endgültig hinzuzufügen: "Werkzeuge/Tools" -> "Board“ -> "Bordverwalter/Board Manager" suche nach: "ESP32“ -> Klicke bei „ESP32“ auf "installieren/install"
- unter "Werkzeuge/Tools" -> "Board" -> wähle das entsprechende Board aus. Hier: "ESP32-WROOM-DA Module"
- unter "Werkzeuge/Tools" den passenden Port auswählen

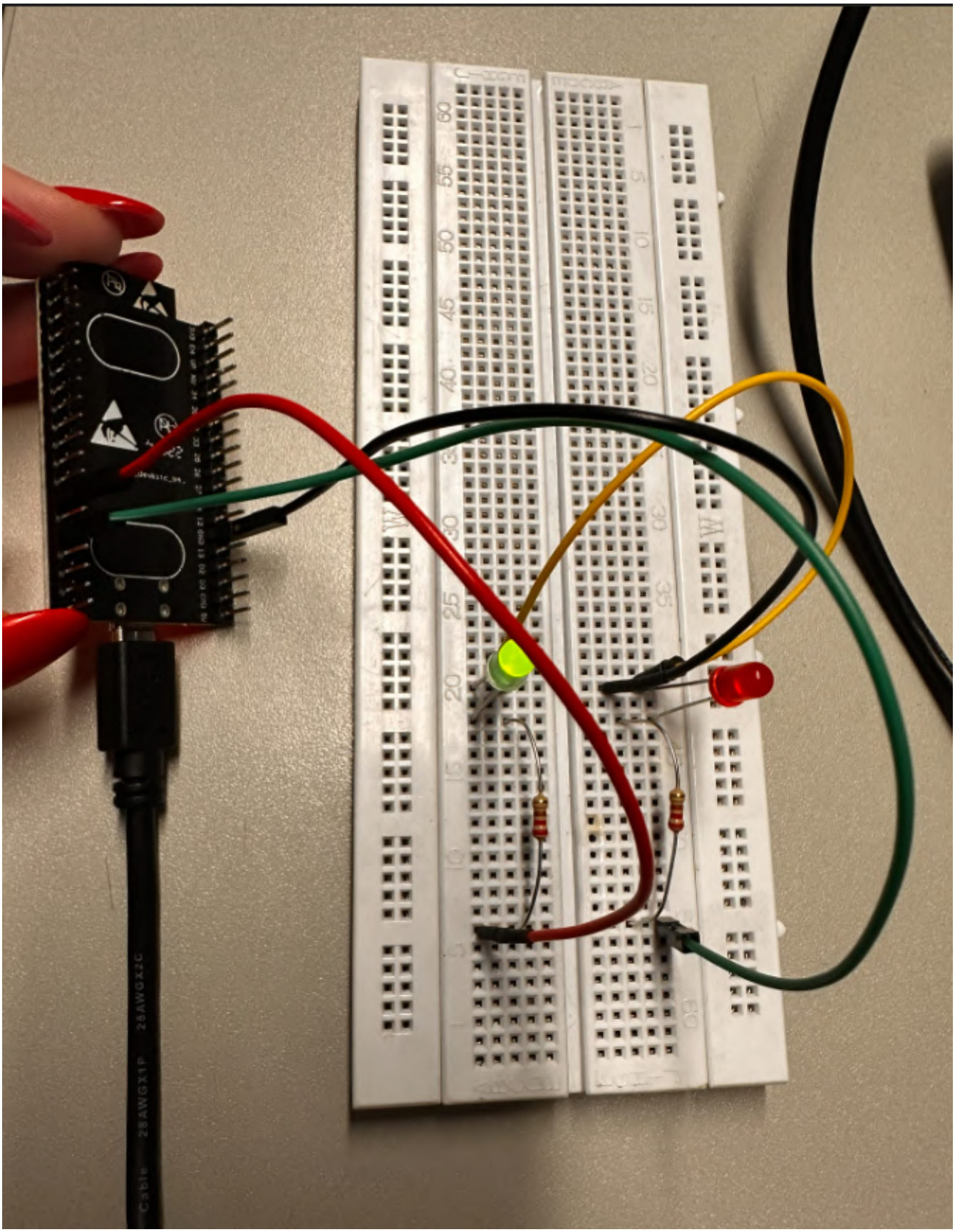
Code

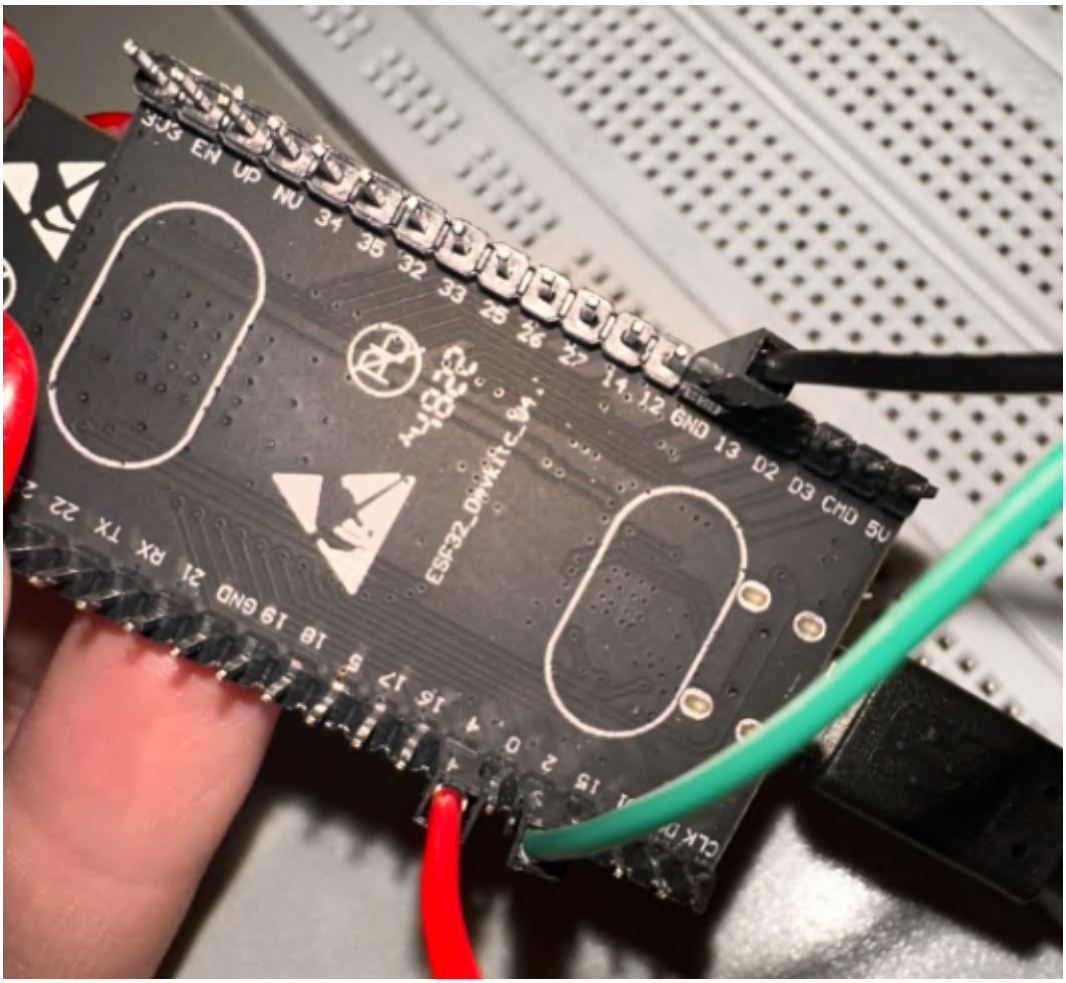
```
void setup()  //Aufbau, wird nur einmal durchlaufen
{
  pinMode(2, OUTPUT);  // Pin 2 ist ein Ausgang
  pinMode(4, OUTPUT);  // Pin 4 ist ein Ausgang
}

void loop()  // das Hauptprogramm beginnt, läut immer wieder ab
{
  digitalWrite(2, HIGH);  // schaltet die LED an Pin 2 an
  digitalWrite(4, LOW);    // schalte die LED an Pin 4 aus
  delay(250);              // zwischen an und aus kurze Pause von 250 ms
  digitalWrite(2, LOW);  // schaltet die LED an Pin 2 aus
  digitalWrite(4, HIGH);  // schaltet die LED an Pin 4 ein
  delay(300);             // zwischen an und aus kurze Pause von 300 ms
  }
```

Verkabelung des Aufbaus

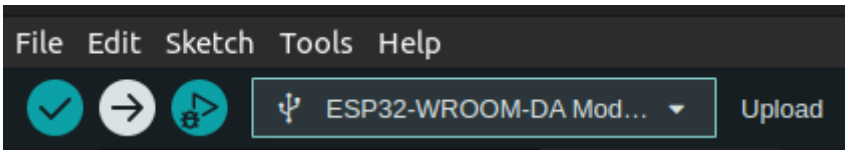
- ESP32 mit entsprechendem Kabel mit Notebook verbinden
- LED (lange Seite positiv-> Anode, kurze eite negativ-> Kathode)
- GPIO2 und GPIO4 mit jeweils einem Kabel auf das BreadBoard verbinden
- 2 Widerstände zwischen den GPIO-Pins (220 Ohm) erstellen, jeweils ein Ende vom Widerstand in die selbe Line vom GPIO2 und GPIO3 Kabel
- 2 LED's jeweils mit der Anode (lange Seite) in die Line des anderen Endes vom Widerstand stecken
- Beide Seiten mit einem Kabel verbinden und auf eine Seite ein weiteres Kabel stecken (selbe Line) und dieses auf dem ESP32 mit dem GND (Ground) verbinden

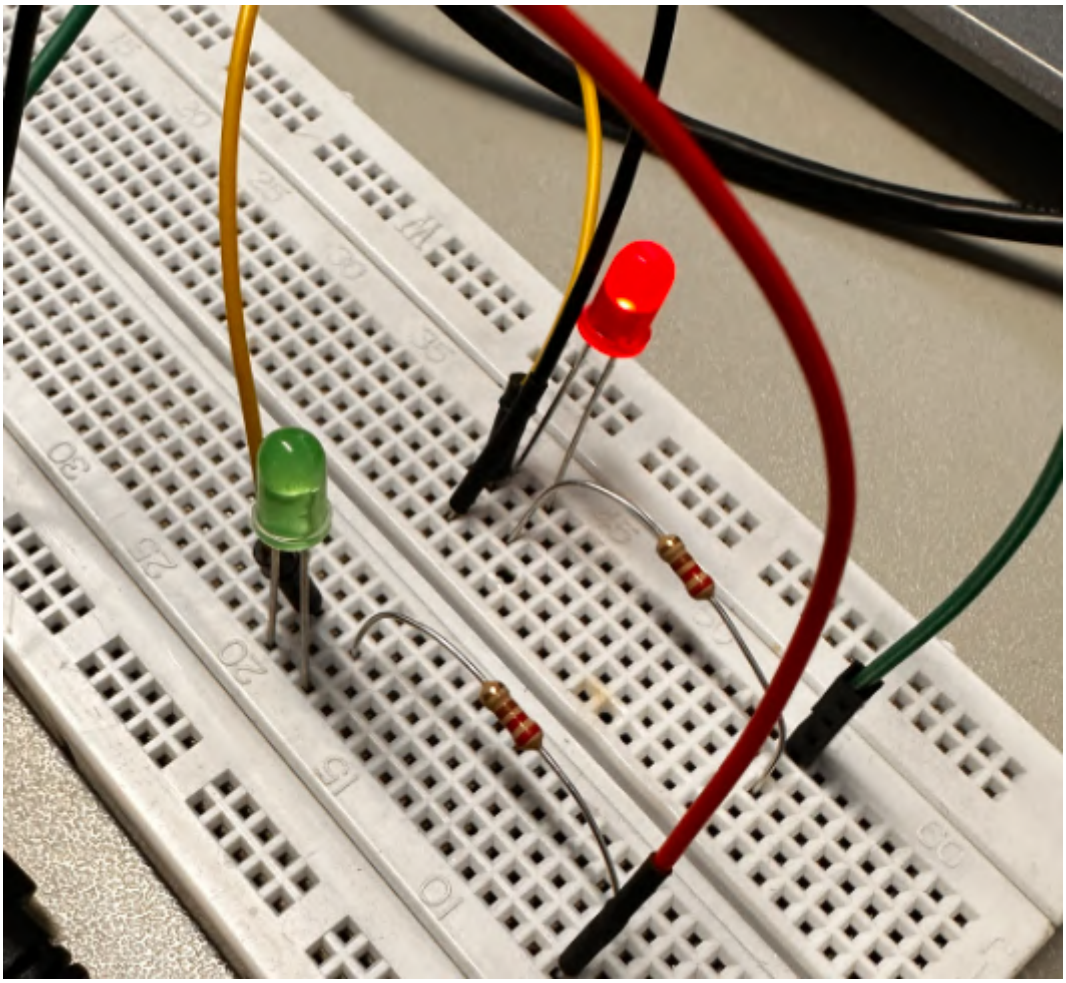


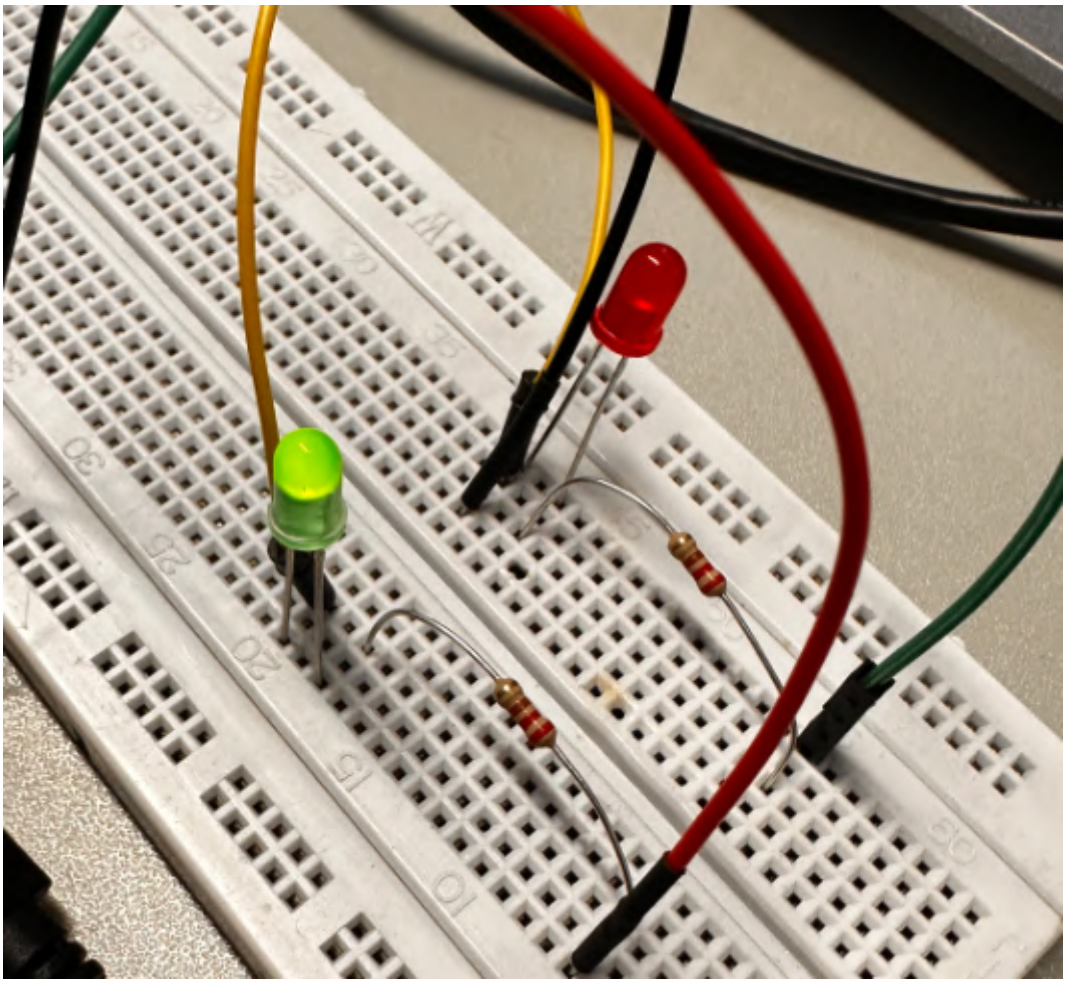


Starten

- "Upload" klicken (mittlerer Button mit dem Pfeil nach rechts)
- Jetzt ist auch bereits alles abgeschlossen und im Idealfall, wenn alles funktioniert hat, befindet sich der Code auf dem ESP32 und wird bereits ausgeführt.







Fragen:

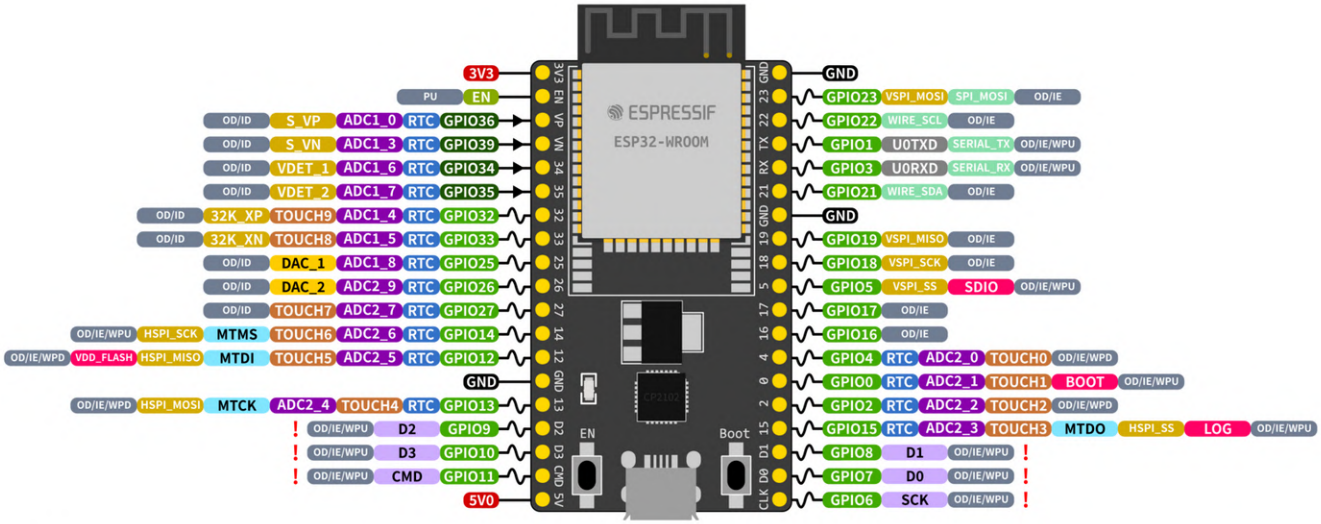
- Wie werden die LEDs mit dem ESP32 verbunden, und welchen Zweck erfüllen die Widerstände in diesem Zusammenhang?
  - LED's sinf mit dem ESP32 mit den Widerständen (220 OHM) verbunden. Die Widerstände sorgen für Schutz, damit die LED's nicht explodieren
- Warum ist die Auswahl der GPIO-Pins wichtig, wenn Sie die LEDs mit dem ESP32 verbinden?
  - GPIO (= General Purpose Input Output), das bedeutet, das der der Pin als Input (Befehl vom Breadboard zum Board) oder Output (Befehl vom Board zum Breadboard) verwendet werden kann. Deswegen ist es wichtig, die Pins zu definieren.
- Welche Schritte sind erforderlich, um die Arduino IDE für die Programmierung des ESP32 vorzubereiten?
  - Downlad und Installation von "VCP Drivers" -> https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
  - Arduino IDE donwloaden -> https://www.arduino.cc/en/software
  - Menüpunkt "Datei/File" die "Voreinstellungen/Preferences" öffnen
  - Boardverwalter-URL für den ESP32 eingetragen: "https://dl.espressif.com/dl/package_esp32_index.json" -> "OK"
  - NodeMCU ESP32 nun endgültig hinzuzufügen: "Werkzeuge/Tools" -> "Board“ -> "Boardverwalter/Board Manager" suche nach: "ESP32“ -> Klicke bei „ESP32“ auf "installieren/install"

    - unter "Werkzeuge/Tools" -> "Board" -> wähle das entsprechende Board aus. Hier: "ESP32-WROOM-DA Module"
    - unter "Werkzeuge/Tools" den passenden Port auswählen
- Wie lautet der grundlegende Codeausschnitt, um die LEDs im gewünschten Muster blinken zu lassen?

```
{
  digitalWrite(2, HIGH);  // schaltet die LED an Pin 2 an
  digitalWrite(4, LOW);    // schalte die LED an Pin 3 aus
  delay(250);              // zwischen an und aus kurze Pause von 250 ms
  digitalWrite(2, LOW);  // schaltet die LED an Pin 2 aus
  digitalWrite(4, HIGH);  // schaltet die LED an Pin 3 ein
  delay(300);             // zwischen an und aus kurze Pause von 300 ms
  }
```

    - (die Blink-Geschwindigkeit kann mit dem Befehl "delay" verändert werden)
- Welche Informationen sollten im Protokoll festgehalten werden, um den Arbeitsfortschritt später nachvollziehen zu können?
    - Die Kommentare im Code (//), um den Code auch später (oder dritte) noch zu verstehen.
- Wie können Sie sicherstellen, dass die Verkabelung korrekt ist, um mögliche Fehlfunktionen zu vermeiden?
    - Bevor ich den Code uploade, kann ich ihn verifizieren (Button "Verify").

Aufgabe 2: Ziel: Das Ziel dieses erweiterten Projekts ist es, ein OLED-Display zum bestehenden Setup hinzuzufügen und dieses anhand eines Textes oder Symbols zu steuern, während die LEDs weiterhin blinken.



Dokumentation:

- Stecke das OLED-Display in das Breadboard
- Das OLED-Display hat 4 Pins: GND (für die Erdung), VCC (für die Stromversorgung), SCL (für serielle Taktleitungen) und SDA (für serielle Datenleitungen)
- Für die Stromversorgung benötigen wir GND und VCC
- Zur Kommunikation verwenden wir SCL und SDA
- Das OLED-Display verwendet das I2C-Kommunikationsprotokoll, daher benötigen wir nur 4 Kabel
- Verbinde den SDA-Anschluss des Displays mit GPIO21 am Board (siehe Bild oben)
- Verbinde den SCL-Anschluss des Displays mit GPIO22 am Board (siehe Bild oben)
- Der VSS-Anschluss des Displays wird an 3V3 angeschlossen (für Volt/Strom) am Board (siehe Bild oben)
- Der GND-Anschluss des Displays wird mit GND verbunden am Board (siehe Bild oben)
- OLED-Display-Bibliothek downloaden: https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/

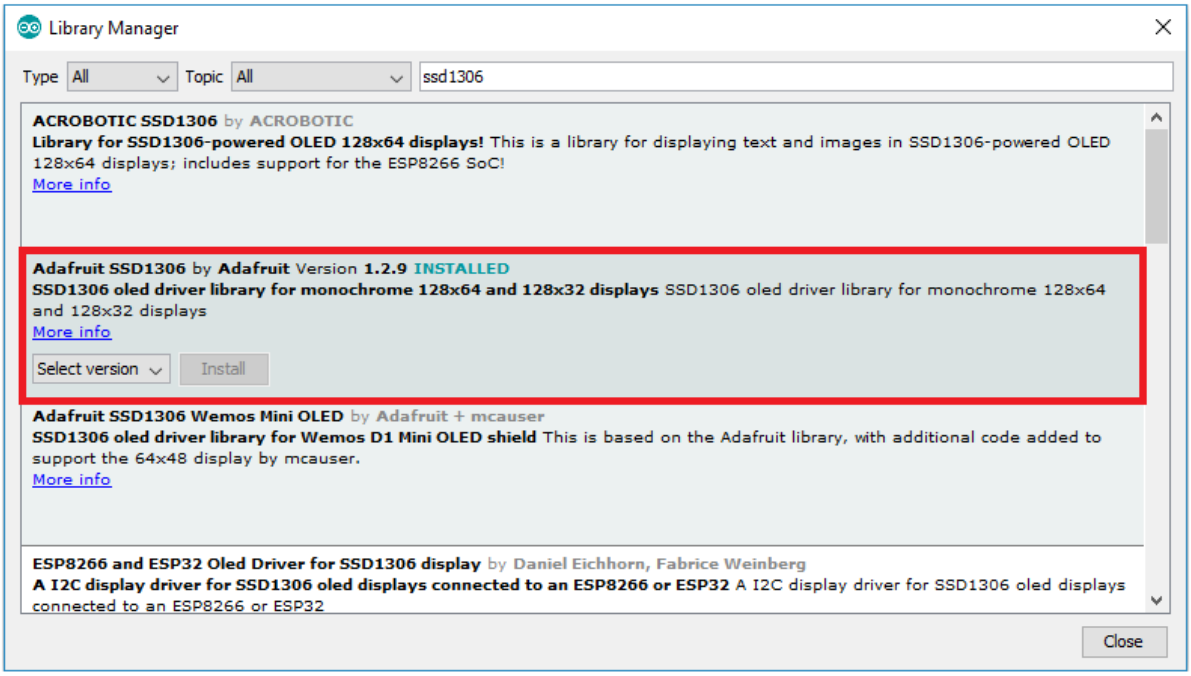
Installing SSD1306 OLED Library – ESP32

There are several libraries available to control the OLED display with the ESP32. In this tutorial we'll use two Adafruit libraries: Adafruit_SSD1306 library and Adafruit_GFX library.

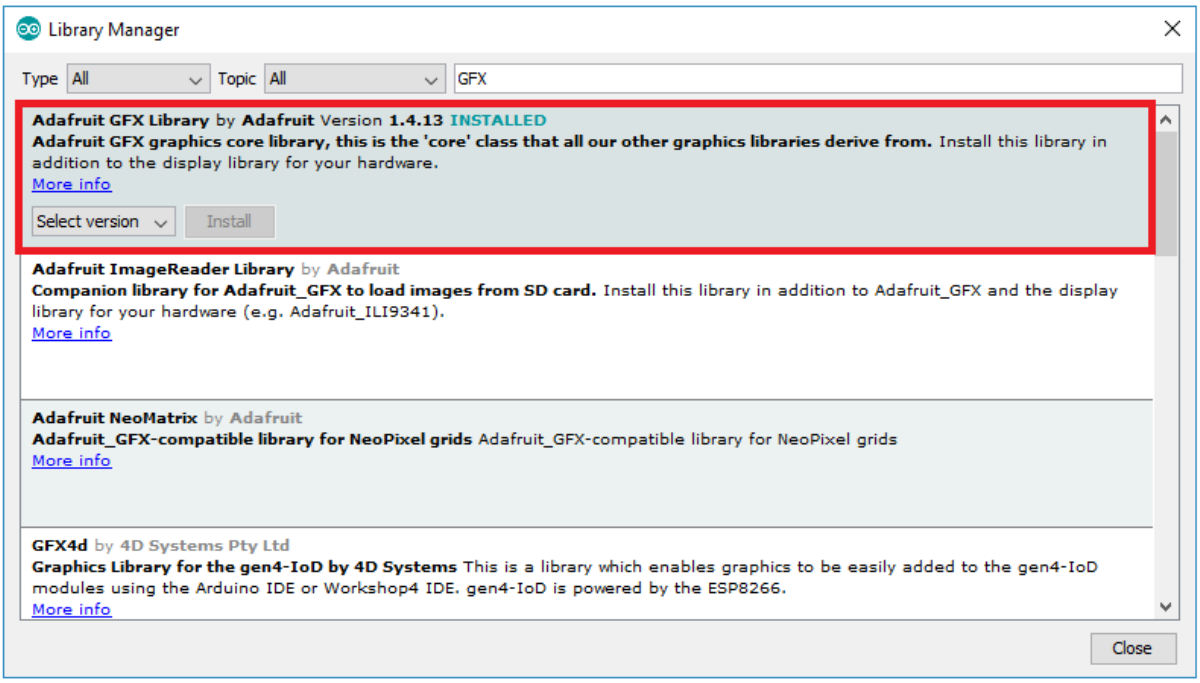
Follow the next steps to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.

2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.



3. After installing the SSD1306 library from Adafruit, type “**GFX**” in the search box and install the library.



4. After installing the libraries, restart your Arduino IDE.

We’ll program the ESP32 using Arduino IDE, so you must have the ESP32 add-on installed in your Arduino IDE. If you haven’t, follow the next tutorial first:

-
- Um zu testen, ob das Display funktioniert. kopiere den Code, der auch auf der folgenden Website steht. Das Display sollte nun etwas anzeigen.
- Um auf dem Display "Hallo Jens!" anzeigen zu lassen, verändere den Code bei dem Befehl: " display.println("Hello, world!");" zu : ("Hallo Jens!!")
  - ```

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
Serial.begin(115200);

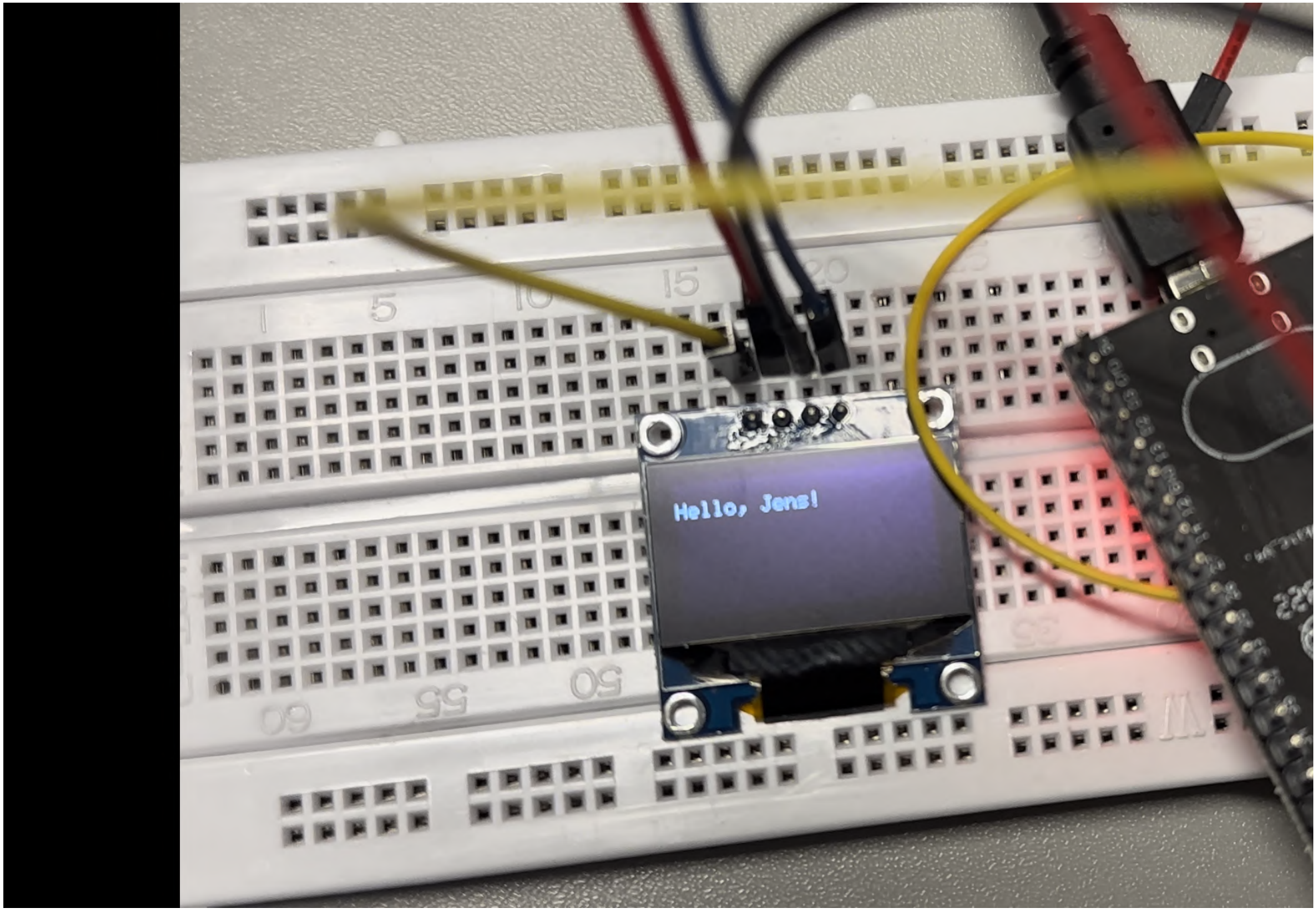
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
Serial.println(F("SSD1306 allocation failed"));
for(;;);
}
```

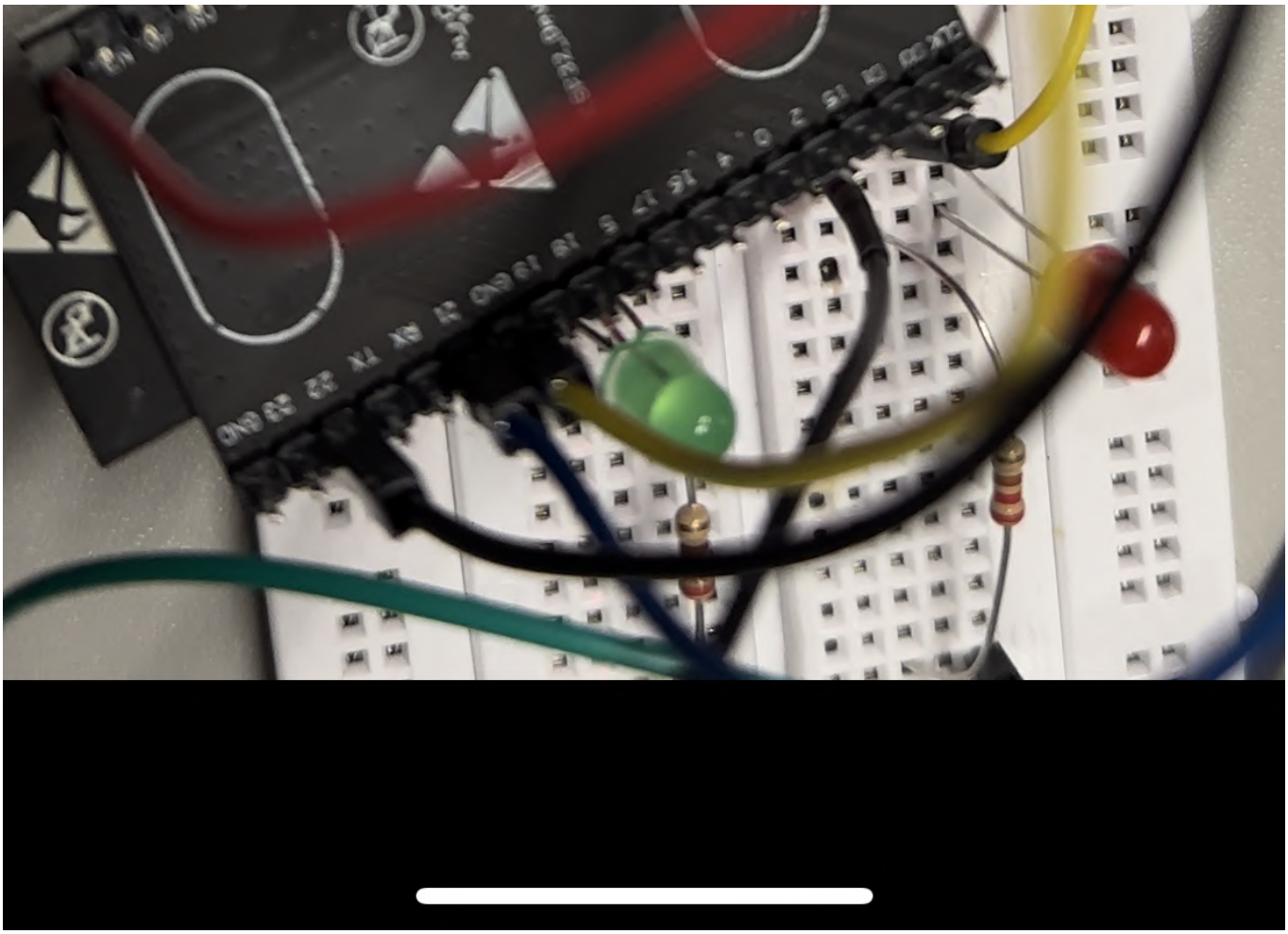
```
delay(2000);
display.clearDisplay();

display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 10);
// Display static text
display.println("Hello, world!");
display.display();
}

void loop() {

}
```


Hello, Jens!



- Erweiterung des Codes um die OLED-Anweisungen, um die LED's mit dem Display zu verbinden:
  - beide Codes (von den LED's und von "Hallo Jens!") miteinander kombinieren, dass beide LED's gleichzeitig an sind -> Display zeigt "LED ON" an, beide LED's aus -> Display zeigt "LED OFF" an

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h> // Bibliothek

#define SCREEN_WIDTH 128 // OLED Display Breite in Pixeln
#define SCREEN_HEIGHT 64 // OLED Display Höhe in Pixeln

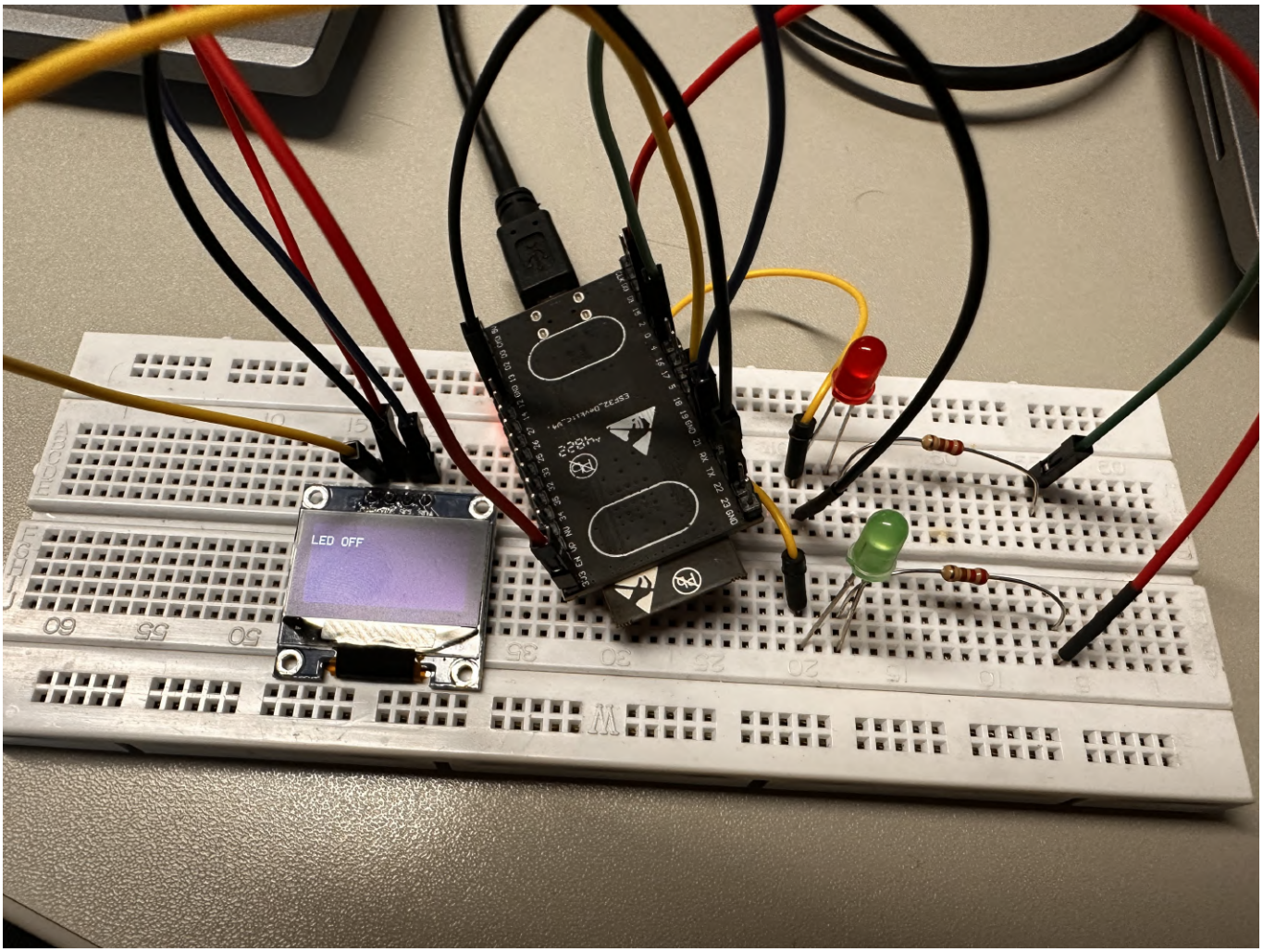
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1); // -1 = Reset Pin, ist nicht am Display, Kein Pin am Display=-1

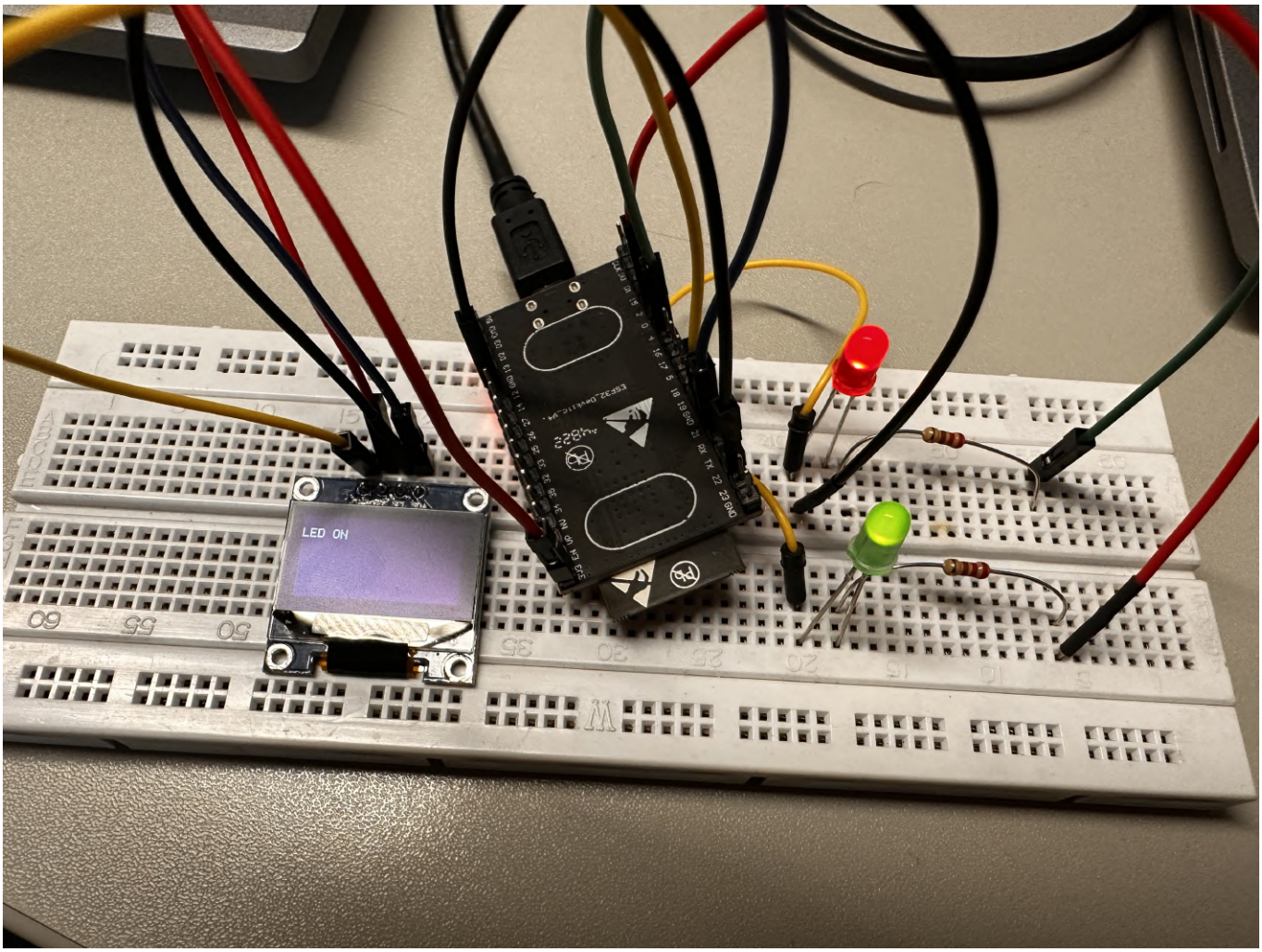
void setup() {
  Serial.begin(115200); // gibt die Taktrate an

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed")); // wenn keine Verbindung enstehen kann, kommt diese Zeile
    for(;;);
  }
  delay(2000); // Pause von 2 Sek
  display.clearDisplay(); // cleart den Display

  display.setTextSize(1);   // Text Größe
  display.setTextColor(WHITE);  // Text Farbe
  display.setCursor(0, 10);  // // wo am Display wird dedr Text positioniert
  display.println("LED OFF"); // Initial status
  display.display();  // aktualisiert das OLED-Display, um die eingestellten Informationen anzuzeigen
  pinMode(2, OUTPUT);  // Pin 2 ist ein Ausgang
  pinMode(4, OUTPUT);  // Pin 4 ist ein Ausgang
}

void loop() {
  digitalWrite(2, HIGH);    // schaltet die LED an Pin 2 an (Rote LED)
  digitalWrite(4, HIGH);     // schaltet die LED an Pin 4 aus (Grüne LED)
    display.clearDisplay();
    display.setCursor(0, 10); // wo am Display wird dedr Text positioniert
    display.println("LED ON"); // zeigt LED ON am Display
    display.display();
  delay(1000);
  digitalWrite(2, LOW);     // schaltet die LED an Pin 2 aus (Rote LED)
  digitalWrite(4, LOW);    // schaltet die LED an Pin 4 ein (Grüne LED)
    display.clearDisplay(); // cleart den Display
    display.setCursor(0, 10); // wo am Display wird dedr Text positioniert
    display.println("LED OFF"); //zeigt LED OFF am Display
    display.display();        // aktualisiert das OLED-Display, um die eingestellten Informationen anzuzeigen
  delay(1000);               // zwischen an und aus kurze Pause von 300 ms
}
```


LED OFF



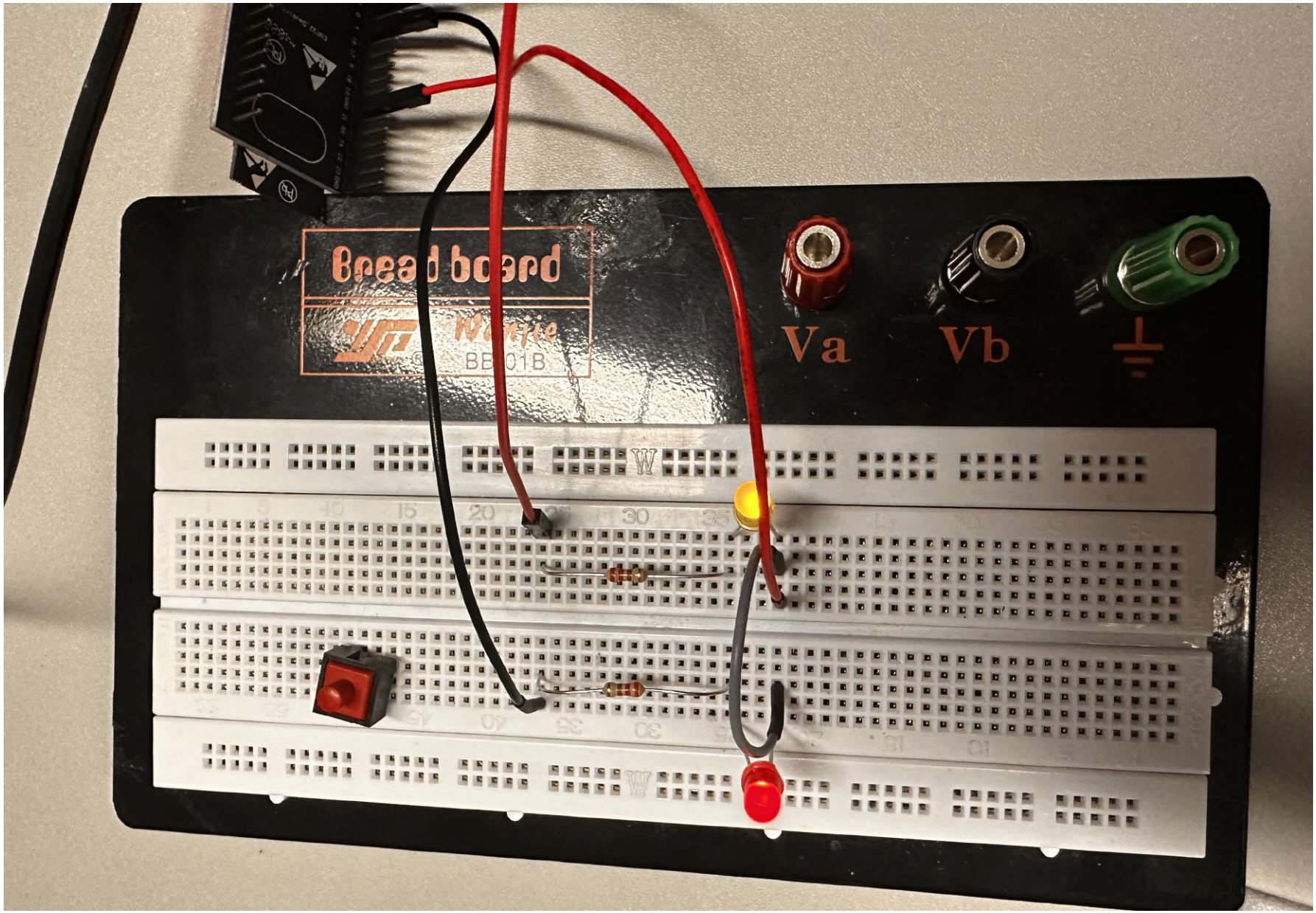
Fragen:

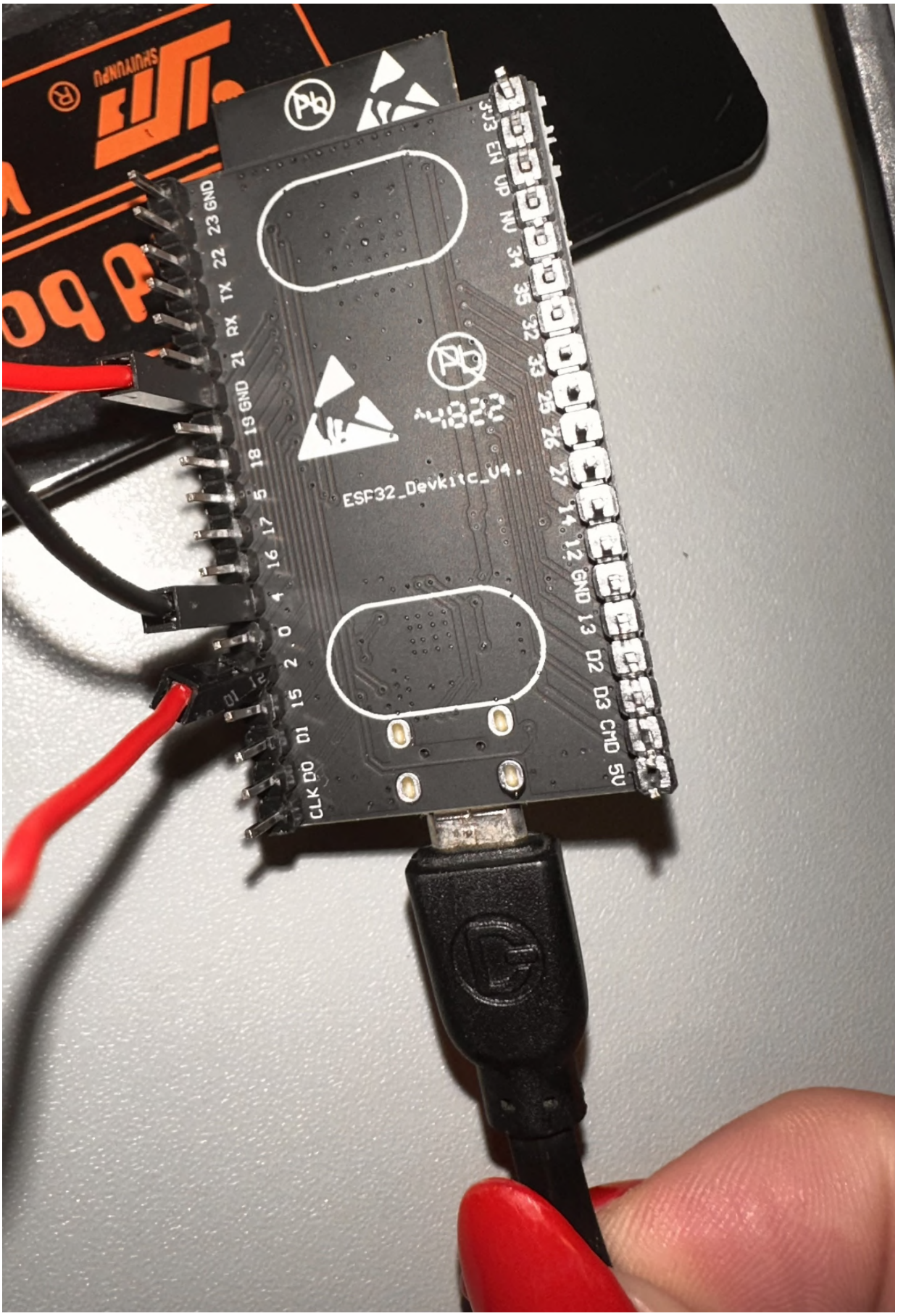
- Wie wird das OLED-Display mit dem ESP32 verbunden, und warum ist die Auswahl der GPIO- Pins wichtig?
    - Es ist wichtig, die richtigen Anschlüsse (SDA, SCL, VSS, GND) entsprechend den GPIO-Pins und Spannungsquellen auf dem Board korrekt zu verbinden, um eine reibungslose Kommunikation und Stromversorgung zu gewährleisten.
- Welche Bibliotheken müssen in der Arduino IDE installiert werden, um das OLED-Display zu steuern?
    - SSD1306 OLED Library und GFX Libary.
- Warum ist es notwendig, den Code des ersten ESP32 zu aktualisieren, um das OLED-Display zu integrieren?
    - Damit die LED's nicht mehr abwechselnd leuchten, sondern gleichzeitig, damit das Display es korrekt anzeigen kann.
- Wie können Sie sicherstellen, dass die I2C-Adresse des OLED-Displays korrekt konfiguriert ist?
    - passende Adresse(DataShield)von dem OLED-Display zum Code : hier 0x3C (findet man im Zeile 13 vom Code wieder). Andere Möglichkeit, im Set-Up-Bereich das eingeben: https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/LCD_I2C/I2C_Scanner.ino.
- Warum sollte der Code für das OLED-Display im Setup-Bereich platziert werden?
    - Weil im Setup der Aufbau/Ausgangszustand steht und im Loop läuft nur das Programm ab.

- Welche möglichen Schwierigkeiten könnten bei der Integration des OLED-Displays auftreten und wie können sie gelöst werden?
    - I2C-Kommunikation Adresse könnte falsch sein - Lösung: Scanner-Sketch
    - Kabel kaputt- Lösung: neues Kabel nehmen

Aufgabe 3: Ziel: Das Ziel dieses erweiterten Projekts ist es, einen zweiten ESP32 in den bestehenden Setup zu integrieren. Dieser zweite ESP32 soll mithilfe von proprietärer Funktechnik (beispielsweise ESP-NOW) Daten an den ersten ESP32 senden, der diese auf dem gemeinsamen OLED-Display anzeigt

- ESP-NOW-Bibliothek downloaden: https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/
- Das Board was als Sender dient, muss die MAC Adresse des Empfängers wissen, der Empfänger muss die MAC Adresse des Senders nicht wissen.
- Board-Empfänger: E0:5A:1B:A1:B0:64
- 2 Sets LED's die gegenläufig blinken
- Port COM5 = Sender
- Port COM6 = Empfänger
- Das zweite Board mit dem selben LED Aufbau ausgestattet wie das erste Board (siehe Aufgabe 1 , LED 1 = GPIO2, LED2 = GPIO4 + GND (Ground))


Bread board
Va
Vb

- 

- 2 Sketche entworfen. Eins für den Sender, eins für Empfänger (Code von https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/).

- Herangehensweise: Versuchen den Code so umzuschreiben, dass innerhalb der Datenübertragung von den 4 Strukturtypen (wir betrachten/nutzen nur den Boolean-Wert) der Boolean-Wert (Wahrheitswert 0 oder 1) sich bei der Übertragung jeweils auf "TRUE" und "FALSE" ändert.

```
typedef struct struct_message {
  char a[32];
  int b;
  float c;
  bool d;
} struct_message;
```

- Sender Code:
  - #include <esp_now.h>
    #include <WiFi.h>
    #include <Adafruit_GFX.h>
    #include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// REPLACE WITH YOUR RECEIVER MAC Address
uint8_t broadcastAddress[] = {0xE0, 0x5A, 0x1B, 0xA1, 0xB0, 0x64};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
char a[32];
int b;
float c;
bool d;
} struct_message;

// Create a struct_message called myData
struct_message myData;

esp_now_peer_info_t peerInfo;

// callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
Serial.print("\r\nLast Packet Send Status:\t");
Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

```
void setup() {
// Init Serial Monitor
Serial.begin(115200);

if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
Serial.println(F("SSD1306 allocation failed"));
for(;;);
}
delay(2000);
display.clearDisplay();

display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 10);
display.println("LED OFF"); // Initial status

display.display();

pinMode(15, OUTPUT); // Pin 2 ist ein Ausgang
pinMode(4, OUTPUT); // Pin 4 ist ein Ausgang

// Set device as a Wi-Fi Station
WiFi.mode(WIFI_STA);

// Init ESP-NOW
if (esp_now_init() != ESP_OK) {
Serial.println("Error initializing ESP-NOW");
return;
}

// Once ESPNow is successfully Init, we will register for Send CB to
// get the status of Trasnmitted packet
esp_now_register_send_cb(OnDataSent);

// Register peer
memcpy(peerInfo.peer_addr, broadcastAddress, 6);
peerInfo.channel = 0;
peerInfo.encrypt = false;

// Add peer
if (esp_now_add_peer(&peerInfo) != ESP_OK){
Serial.println("Failed to add peer");
return;
}
}

void loop() {
// Set values to send
strcpy(myData.a, "THIS IS Hrishi");
```

```
myData.b = random(1,20);
myData.c = 1.2;
myData.d = true;

if (myData.d == true)
{
digitalWrite(15, HIGH); // schaltet die LED an Pin 2 an (Rote LED)
digitalWrite(4, HIGH); // schaltet die LED an Pin 4 aus (Grüne LED)
display.clearDisplay();
display.setCursor(0, 10);
display.println("LED ON");
display.display();
delay(500);

// Send message via ESP-NOW
esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

myData.d = false;
}

if (myData.d == false)
{
digitalWrite(15, LOW); // schaltet die LED an Pin 2 aus (Rote LED)
digitalWrite(4, LOW); // schaltet die LED an Pin 4 ein (Grüne LED)
display.clearDisplay();
display.setCursor(0, 10);
display.println("LED OFF");
display.display();
delay(500); // zwischen an und aus kurze Pause von 300 ms

// Send message via ESP-NOW
esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

}

// Send message via ESP-NOW
esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

if (result == ESP_OK) {
Serial.println("Sent with success");
}
else {
Serial.println("Error sending the data");
}
delay(10);
}
```

- Empfänger Code:
  - #include <esp_now.h>
    #include <WiFi.h>

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
char a[32];
int b;
float c;
bool d;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t *mac, const uint8_t* incomingData, int len) {
memcpy(&myData, incomingData, sizeof(myData));
Serial.print("Bytes received: ");
Serial.println(len);
Serial.print("Char: ");
Serial.println(myData.a);
Serial.print("Int: ");
Serial.println(myData.b);
Serial.print("Float: ");
Serial.println(myData.c);
Serial.print("Bool: ");
Serial.println(myData.d);
Serial.println();
}

void setup() {
// Initialize Serial Monitor
Serial.begin(115200);

if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
Serial.println(F("SSD1306 allocation failed"));
for(;;);
}

```
delay(2000);
display.clearDisplay();

display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 10);
display.println("LED OFF"); // Initial status

display.display();

pinMode(15, OUTPUT); // Pin 2 ist ein Ausgang
pinMode(4, OUTPUT); // Pin 4 ist ein Ausgang

// Set device as a Wi-Fi Station
WiFi.mode(WIFI_STA); //Initialisierung von WIFI

// Init ESP-NOW //Initialisierung von ESP-NOW
if (esp_now_init() != ESP_OK) {
Serial.println("Error initializing ESP-NOW"); //
return;
}

// Once ESPNow is successfully Init, we will register for recv CB to
// get recv packer info
esp_now_register_recv_cb(OnDataRecv);
}

void loop() {

if (myData.d == false)
{
digitalWrite(15, HIGH); // schaltet die LED an Pin 2 an (Rote LED)
digitalWrite(4, HIGH); // schaltet die LED an Pin 4 aus (Grüne LED)
display.clearDisplay();
display.setCursor(0, 10);
display.println("LED RED ON");
display.println("LED RED ON");
display.display();
delay(505);

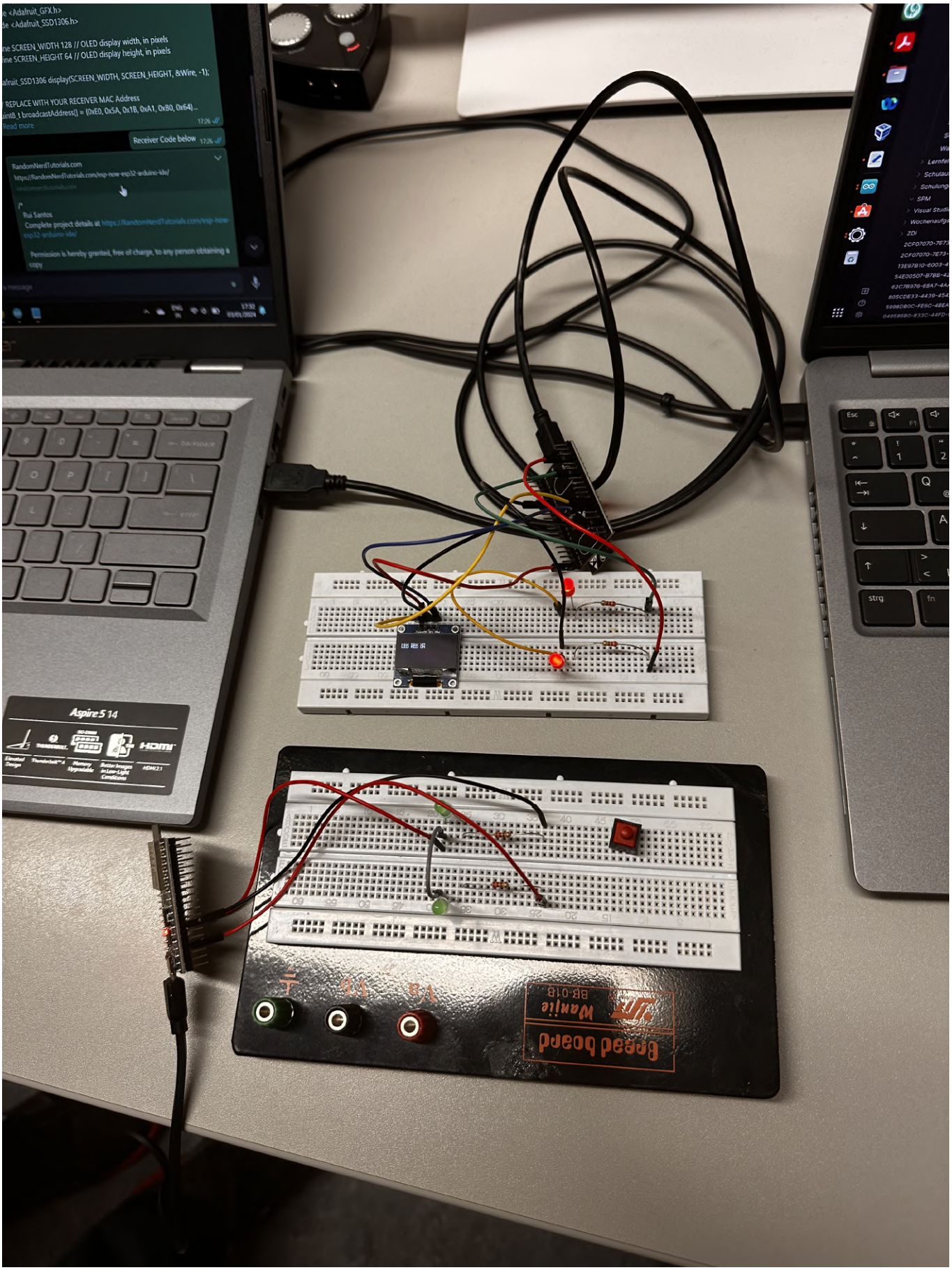
}

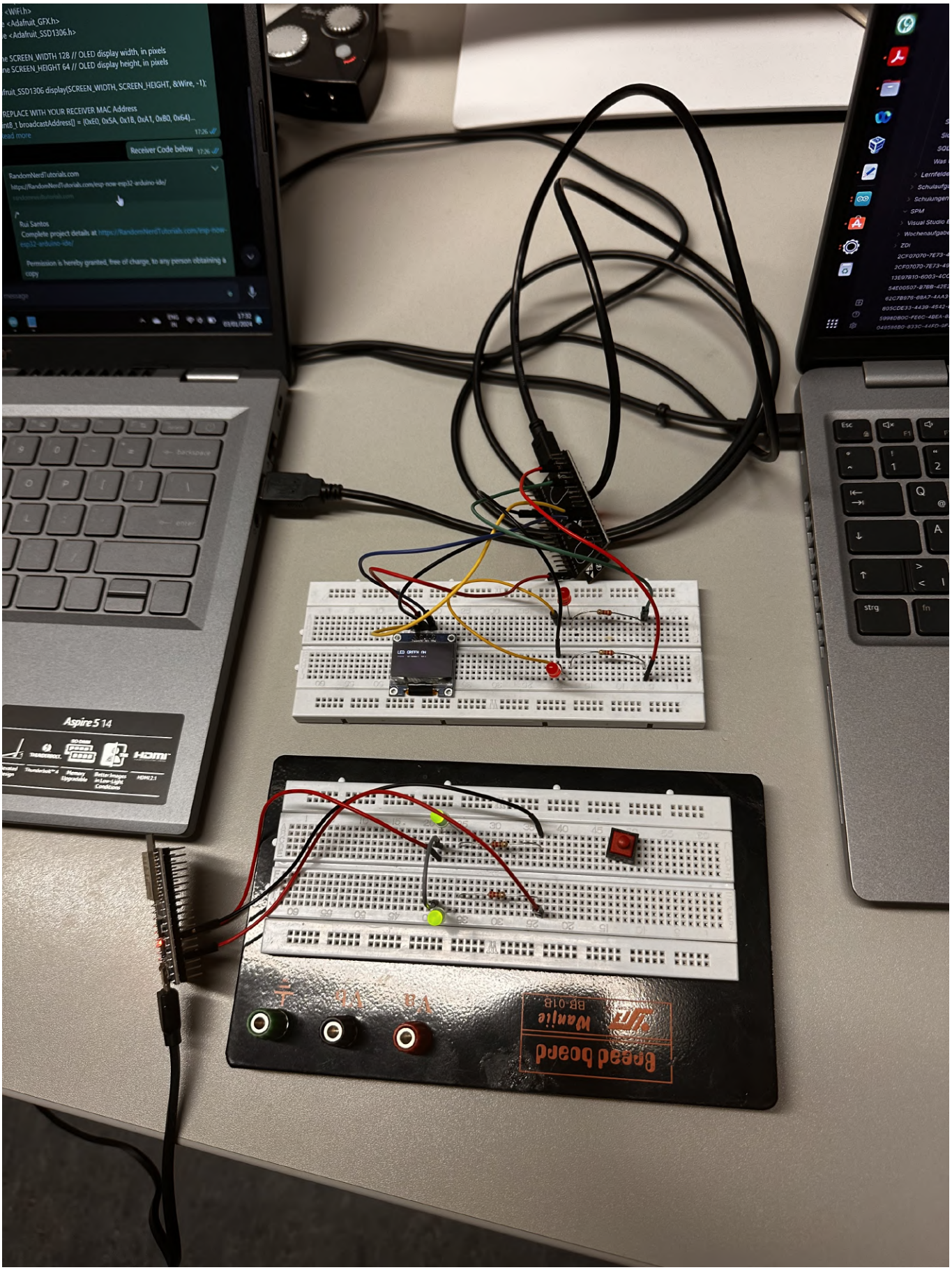
if (myData.d == true)
{
digitalWrite(15, LOW); // schaltet die LED an Pin 2 aus (Rote LED)
digitalWrite(4, LOW); // schaltet die LED an Pin 4 ein (Grüne LED)
display.clearDisplay();
display.setCursor(0, 10);
display.println("LED GREEN ON");
```

```
display.println("LED GREEN ON");
display.display();
delay(505); // zwischen an und aus kurze Pause von 300 ms

}

}
```


Receiver Code below
RandomNerdTutorials.com
Aspire 5 14
HDMI
Bread board
Wanlie
BB-01B



Fragen:

- Welche zusätzlichen Materialien werden für den dritten Schritt des Projekts benötigt?
  - Ein neues Breadboard, mit 2 LED's, 4 Kabel (vom Board zum Breadboard), 2 Widerstände (220 Ohm) und ein Verbindungskabel vom Board zum Laptop
- Wie wird der zweite ESP32 mit dem bestehenden Setup verbunden, und welche Rolle spielen die Antennen in diesem Zusammenhang?
  - Der zweite ESP32 agiert als Sender. Die LEDs wurden am ersten (Empfänger) ESP's basierend auf dem Blinkstatus des zweiten (Sender) ESP's gesteuert. Wir haben die boolean Variable aus der Struktur "mydata" (Variabel-Name) oder "struct_message" (Struktur-Name) verwendet.
  - Die Antenne beim ESP32 ermöglicht drahtlose Kommunikation über Wi-Fi und Bluetooth.
- Warum ist die Auswahl der GPIO-Pins für den zweiten ESP32 wichtig, insbesondere für die LEDs?
  - Um die LED's richtig zu steuern. Wenn Boolean bei Sender TRUE= LED ON, -> Boolean beim Empfänger FALSE = LED OFF
- Welche spezielle Funktechnik wird für die Kommunikation zwischen den beiden ESP32- Modulen verwendet, und warum?
  - WIFI (WIFI stärker als Bluetooth) (Notebook hat den beiden Boards nur Strom gegeben), für schnelle und starke Verbindung
- Welche Bibliothek wird in der Arduino IDE installiert, um die ESP-NOW-Funktionalität zu unterstützen?
  - Die Libary "esp_now.h" and "WiFi.h"
- Wie werden die MAC-Adressen der beiden ESP32-Module im Code verwendet, und warum sind sie wichtig?
  - MAC-Adressen der beiden ESP32-Module dienen dazu, die Geräte (Sender und Empfänger) zu identifizieren, an die die Daten gesendet werden sollen. Die MAC-Adressen werden im Code als Broadcast-Adresse und als Array dargestellt (Zahlensammlung).
  - ```
    uint8_t broadcastAddress[] = {0x30, 0xAE, 0xA4, 0x07, 0x0D, 0x64};
    ```
- Welche Funktion wurde im Code des ersten ESP32 hinzugefügt, um Daten vom zweiten ESP32 zu empfangen?
  - > `esp_now_register_rcv_cb()` Register a callback function that is triggered upon receiving data. When data is received via ESP-NOW, a function is called.
  - (Registrieren Sie eine Rückruffunktion, die beim Empfang von Daten ausgelöst wird. Beim Empfang von Daten über ESP-NOW wird eine Funktion aufgerufen.)
- Warum sollte die Funktion zum Empfangen von Daten in einer eigenen Funktion organisiert sein?
  - Um dem Absender eine Bestätigung zu senden, dass die Nachricht empfangen wurde.
- Welche Informationen könnten im Protokoll für den dritten Schritt festgehalten werden, um später auf den Arbeitsfortschritt zurückzublicken?
- Kommentare im Code (//)
    ```