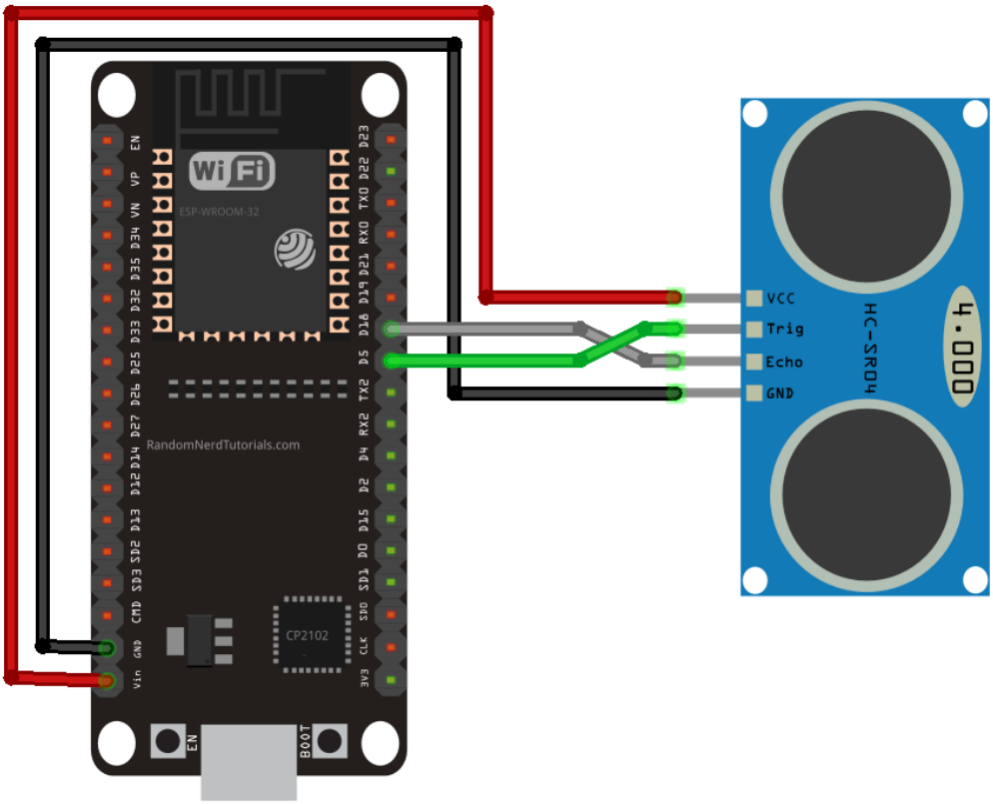
GKL611 Datenerfassung mit Sensoren und Aktoren

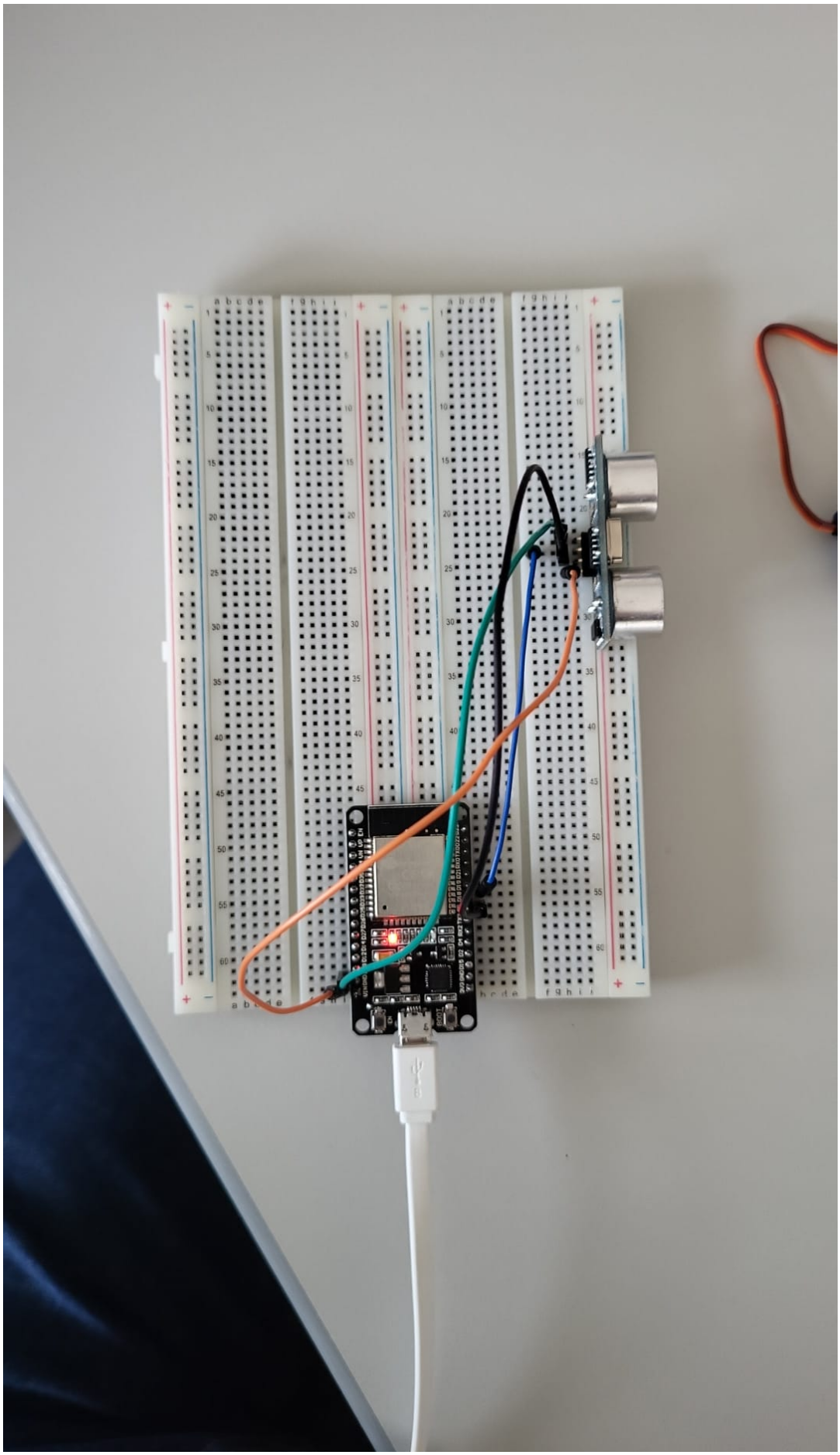
@author Kacper Bohaczyk

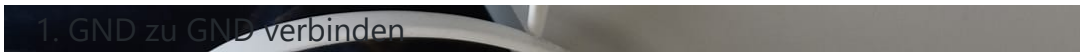
@version 05-05-2023

1 Anschießen des Ultrasschallsensor

Webside: ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE | Random Nerd Tutorials







2. VCC zu VIN

3. Trig zu D5

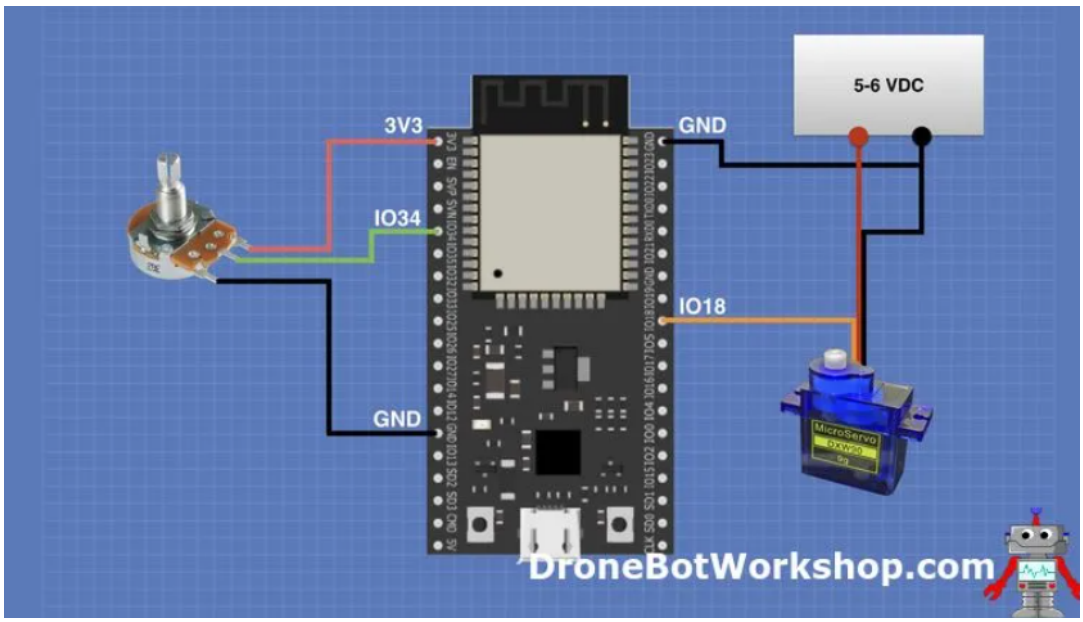
4. Echo zu D18 Code zum Testen:

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-hc

  Permission is hereby granted, free of charge, to any person obtainir
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be inclu
  copies or substantial portions of the Software.
*********/

const int trigPin = 5;
const int echoPin = 18;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microsec
```

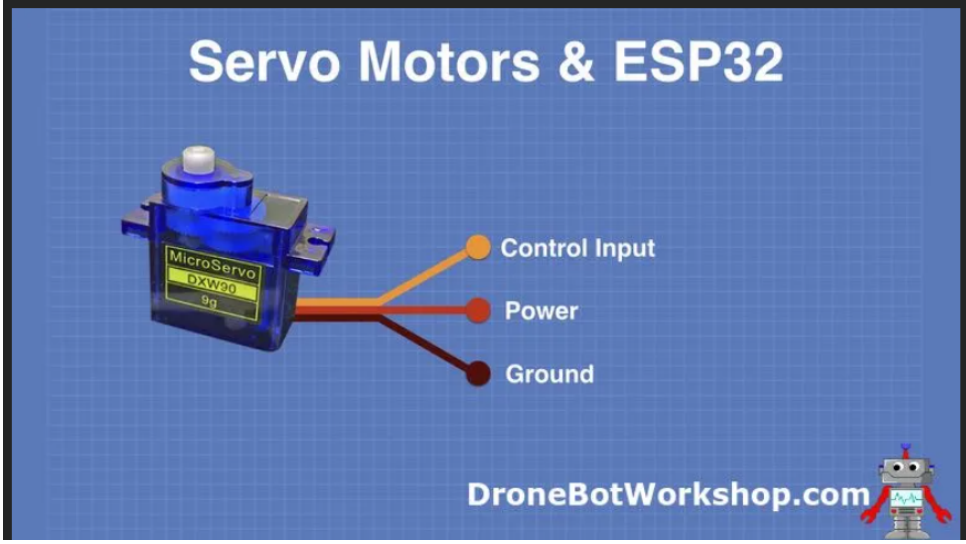
```
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  delay(1000);
}
```

2: Servomotor

Für den 2 Schritt stecke den Ultraschallsensor aus



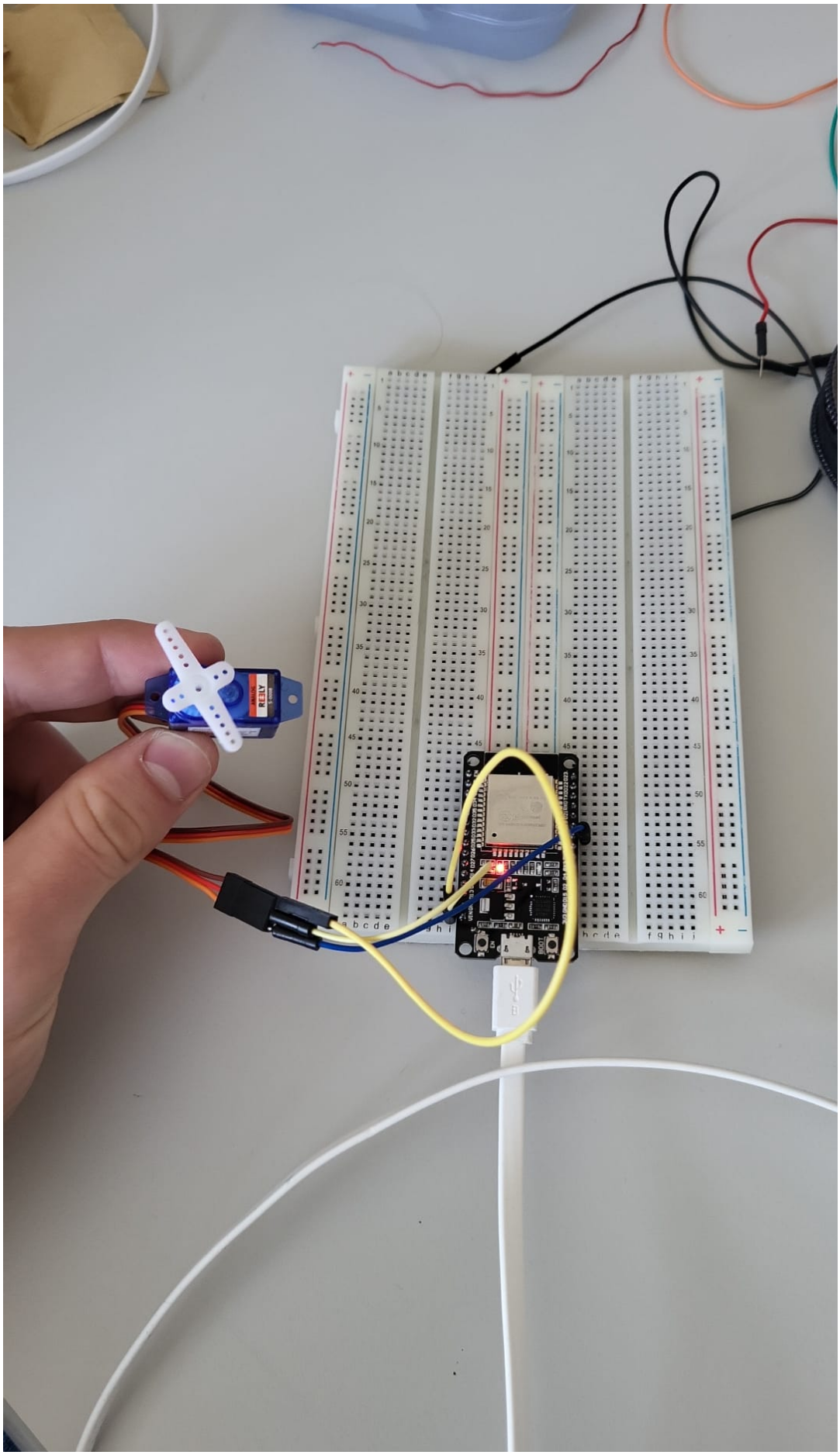
- **Orange** – The PWM servo control input. This is a logic-level signal, and most servo motors can accept 3.3-volt logic as well as 5-volt logic. Some models, especially 270-degree rotation servos, use a White wire for this connection.
- **Red** – The servo motor power supply input. Generally 5-6 volts DC, but be sure to check first.
- **Brown** – The ground connection. On some servo motors, this is a Black wire.



1. Schliesse den Braunen(Ground) anschluss zum GND an

2. Den Roten(Power) zum VIN

3. Und den Orangenen(Controll) zum D18

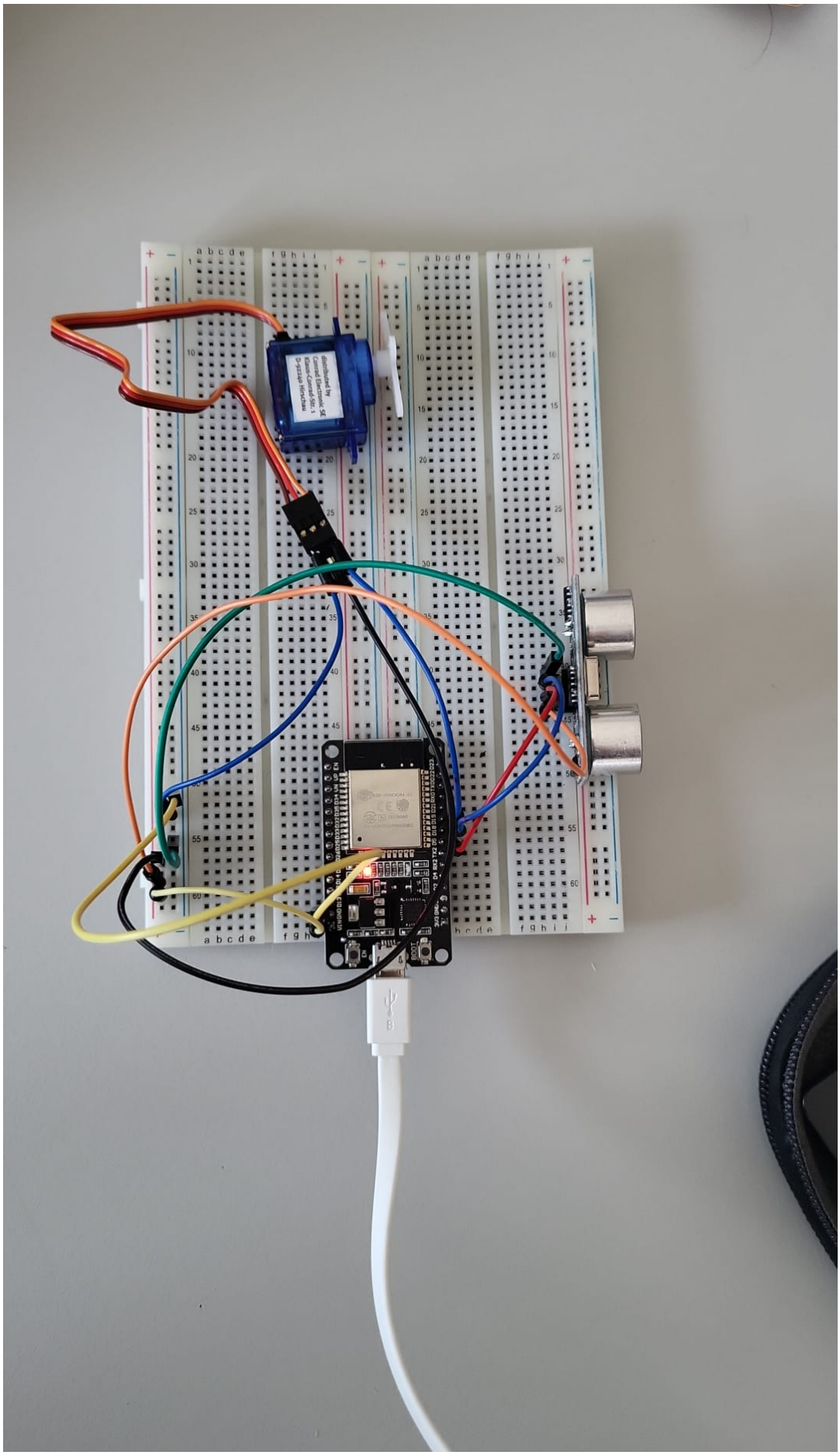

RELY

Code zum Testen

```
/* Sweep
 by BARRAGAN <http://barraganstudio.com>
 This example code is in the public domain.

 modified 8 Nov 2013
 by Scott Fitzgerald

 modified for the ESP32 on March 2017
 by John Bennett

 see http://www.arduino.cc/en/Tutorial/Sweep for a description of the

 * Different servos require different pulse widths to vary servo angle
 * an approximately 500-2500 microsecond pulse every 20ms (50Hz). In g
 * sweep 180 degrees, so the lowest number in the published range for
 * represents an angle of 0 degrees, the middle of the range represent
 * of the range represents 180 degrees. So for example, if the range i
 * 1000us would equal an angle of 0, 1500us would equal 90 degrees, ar
 * degrees.
 *
 * Circuit: (using an ESP32 Thing from Sparkfun)
 * Servo motors have three wires: power, ground, and signal. The power
 * the ground wire is typically black or brown, and the signal wire is
 * orange or white. Since the ESP32 can supply limited current at only
 * considerable power, we will connect servo power to the VBat pin of
 * near the USB connector). THIS IS ONLY APPROPRIATE FOR SMALL SERVOS.
 *
 * We could also connect servo power to a separate external
 * power source (as long as we connect all of the grounds (ESP32, serv
 * In this example, we just connect ESP32 ground to servo ground. The
 * connect to any available GPIO pins on the ESP32 (in this example, w
 *
 * In this example, we assume a Tower Pro MG995 large servo connected
 * The published min and max for this servo is 1000 and 2000, respecti
 * These values actually drive the servos a little past 0 and 180, so
 * if you are particular, adjust the min and max values to match your
 */

#include <ESP32Servo.h>

Servo myservo;  // create servo object to control a servo
// 16 servo objects can be created on the ESP32

int pos = 0;    // variable to store the servo position
```

```
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-2
int servoPin = 18;

void setup() {
    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);
    myservo.setPeriodHertz(50);    // standard 50 hz servo
    myservo.attach(servoPin, 500, 2400); // attaches the servo on pin
    // using default min/max of 1000us and 2000us
    // different servos may require different min/max settings
    // for an accurate 0 to 180 sweep
}

void loop() {

    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 18
        // in steps of 1 degree
        myservo.write(pos);    // tell servo to go to position in vari
        delay(15);             // waits 15ms for the servo to reach th
    }
    for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to
        myservo.write(pos);    // tell servo to go to position in vari
        delay(15);             // waits 15ms for the servo to reach th
    }
}
```

Stecke jetzt den Ultraschalssensor eneut an



Code:

```
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701
#include <ESP32Servo.h>
long duration;
int distanceCm;
float distanceInch;
Servo myservo;  // create servo object to control a servo
// 16 servo objects can be created on the ESP32

int pos = 0;    // variable to store the servo position
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-2
int servoPin = 19;
const int trigPin = 5;
const int echoPin = 18;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  // Allow allocation of all timers
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  myservo.setPeriodHertz(50);    // standard 50 hz servo
  myservo.attach(servoPin, 500, 2400); // attaches the servo on pin 18
  // using default min/max of 1000us and 2000us
  // different servos may require different min/max settings
  // for an accurate 0 to 180 sweep
}

void loop() {

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microsec
  duration = pulseIn(echoPin, HIGH);
```

```
  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);


  myservo.write(distanceCm*5);
  delay(100);

}
```

3: Kalibrierung

Stecke den Ultraschallsensor erneut an

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-hc

  Permission is hereby granted, free of charge, to any person obtainir
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be inclu
  copies or substantial portions of the Software.
*********/

const int trigPin = 5;
const int echoPin = 18;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034

float startTime;
float minD = 90000.00;
float maxD = 0.0;
```

```
float lastDistance = 0.0;


void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(2, OUTPUT);

  lastDistance = distance(true);

  digitalWrite(2, HIGH);   // turn the LED on (HIGH is the voltage lev
  startTime = millis();
  while(millis() - startTime <= 10000){
    float cDis = distance(false);
    if(cDis < minD && !(cDis > 400)) {
        minD = cDis;
    }
    if(cDis > maxD && !(cDis > 400) ){
      maxD = cDis;
    }
    delay(300);
    Serial.println(cDis);
  }

  digitalWrite(2, LOW);   // turn the LED off (LOW is the voltage leve

  Serial.print("Minimale Distance (cm): ");
  Serial.println(minD);
  Serial.print("Maximal Distance (cm): ");
  Serial.println(maxD);

  /**
  delay(1000);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microsec
 float duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
 float distanceCm = duration * SOUND_SPEED/2;
```

```
 Serial.print(distanceCm);
 */
}

float distance(boolean firstTime) {
// Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microsec
 float duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
 float distanceCm = duration * SOUND_SPEED/2;
 if(firstTime == False) {
  if(distanceCm <= lastDistance * 1.2 && distanceCm >= lastDistance *
    lastDistance = distanceCm;
  }
 }




  return lastDistance;
}

void loop() {
  /**
  float cDis = distance();
  if((cDis >= minD) && (cDis <= maxD)) {
    Serial.print("Distance (cm): ");
    Serial.println(cDis);
  }
  delay(1000);
  */
}
```

4: Luftfeuchtigkeit

Code:

```
#include "DHT.h"
const int trigPin = 5;
const int echoPin = 18;
const int ledPin = 2;
#define SOUND_SPEED 0.03316
long duration;
float distanceCm;
float exactSoundSpeed;
float exactDistance;
float minVal = 1000.0;
float maxVal = 0.0;
DHT dht(19, DHT11);
void setup() {
Serial.begin(115200);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
dht.begin();

}
void loop() {
digitalWrite(trigPin, LOW);

delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm = duration * SOUND_SPEED/2;
delay(500);
float temp = dht.readTemperature();
float hum = dht.readHumidity();
if(isnan(temp) || isnan(hum))
return;
exactSoundSpeed = SOUND_SPEED * (1+0.00375*hum/100) + 0.00006*temp;
exactDistance = duration * exactSoundSpeed / 2;
if(distanceCm >= 2 && distanceCm <= 400) { // Ausreißer filtern
if(millis() <= 10000) { // Kalibrierungsphase
minVal = distanceCm < minVal ? distanceCm : minVal;
maxVal = distanceCm > maxVal ? distanceCm : maxVal;
} else {
digitalWrite(ledPin, LOW);
}
Serial.print("Distanz:");
Serial.print(distanceCm);
```

```
Serial.print("Distanz mit Temperatur und Luftfeuchtigkeit:");
Serial.println(exactDistance);
}
}
```

5: Genauigkeit

Code:

```
#include "DHT.h"
const int trigPin = 5;
const int echoPin = 18;
const int ledPin = 2;
#define SOUND_SPEED 0.03316
long duration;
float distanceCm;
float exactSoundSpeed;
float exactDistance;
float correctedDistance;
float minVal = 1000.0;
float maxVal = 0.0;
DHT dht(19, DHT11);
void setup() {
Serial.begin(115200);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
dht.begin();
}
void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm = duration * SOUND_SPEED/2;
delay(500);
float temp = dht.readTemperature();
float hum = dht.readHumidity();
if(isnan(temp) || isnan(hum))
return;
exactSoundSpeed = SOUND_SPEED * (1+0.00375*hum/100) + 0.00006*temp;
```

```
exactDistance
= duration * exactSoundSpeed / 2;
if(distanceCm >= 2 && distanc

eCm <= 400) {

if(millis() <= 10000) {
minVal = distanceCm < minVal ? distanceCm : minVal;
maxVal = distanceCm > maxVal ? distanceCm : maxVal;
} else {
digitalWrite(ledPin, LOW);
}
correctedDistance = exactDistance <= 50 ? exactDistance : exactDistanc
Serial.print("Distanz:");
Serial.print(exactDistance);
Serial.print("Distanz mit minimiertem Fehler:");
Serial.println(correctedDistance);
}
}
```