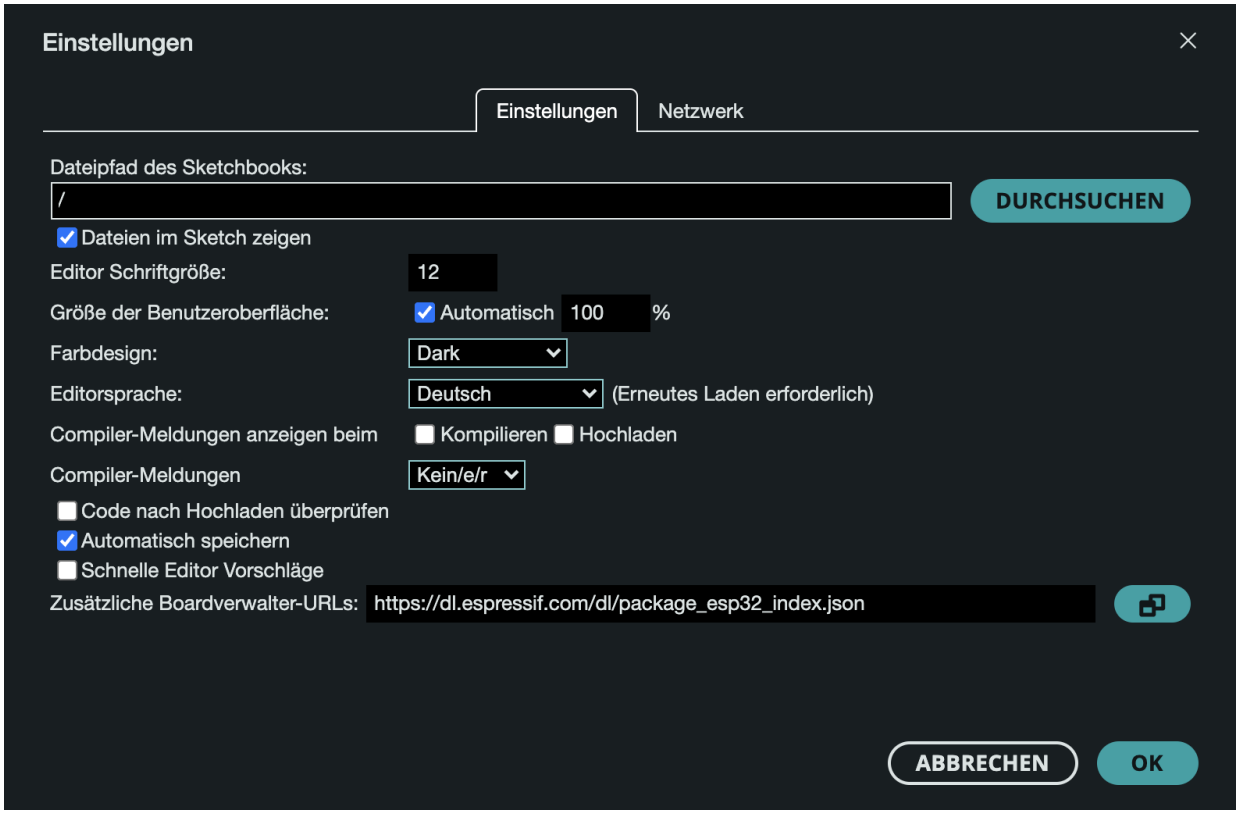
Leitfaden für den Bau eines low-cost pH-Sensors

1. Lade Arduino IDE auf deinen PC und installiere es.
2. Öffne **Arduino IDE** und die Einstellungen.
3. Kopiere den Pfad **https://dl.espressif.com/dl/package_esp32_index.json** zu **Boardverwalter-URLs**. Klicke anschließend auf OK.



4. Gehe über den Reiter **"Werkzeuge"** zu **"Board"** und wähle den **"Boardverwalter"** aus. Suche in dem neuen Fenster nach**"ESP32"** und installiere das angezeigte Board.

5. Gehe erneut über den Reiter **"Werkzeuge"** zu **"Bibliotheken verwalten"** und suche nach der Bibliothek **"Blynk"**, installiere diese anschließend.
6. Füge außerdem über den Reiter **"Sketch"**, **"Bibliothek einbinden"** und **"Bibliothek verwalten"** die .zip Datei ein, die dir auf GRIPS zur Verfügung gestellt wurde.
7. Baue den pH-Sensor nach dem dargestellten Schaltkreis zusammen.

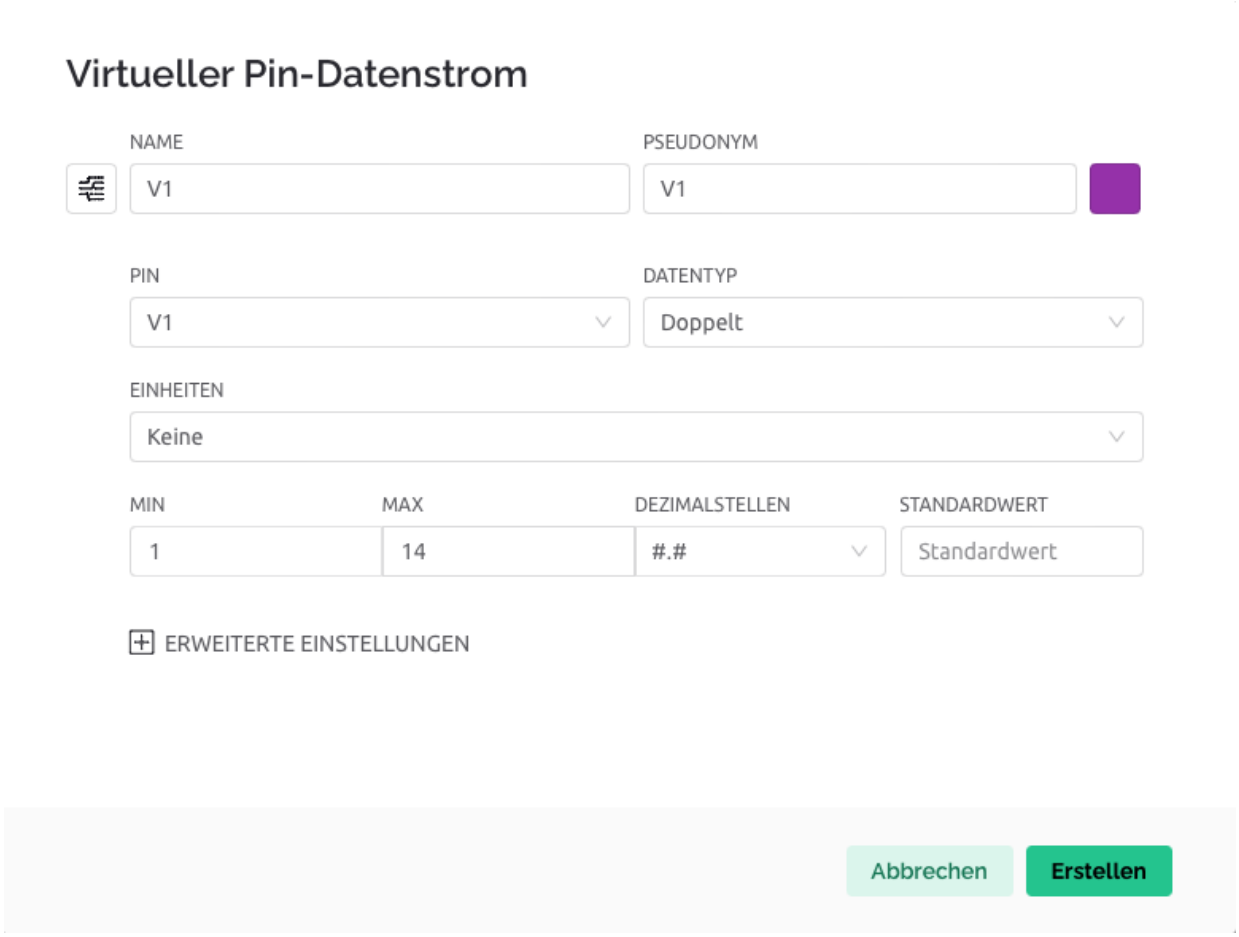
8. Überprüfe den Schaltkreis bevor du den Mikrocontroller mit dem PC verbindest, um eventuelle Schäden zu vermeiden.

9. Wähle nun über **“Werkzeuge”** und **“Board”** unter **“ESP32”** das Board **“ESP32 Dev Module”** aus. Außerdem solltest du unter **“Werkzeuge”** den Uploadspeed auf **“921600”** stellen und den entsprechend **“Port”** also den USB-Anschluss an deinem PC auswählen.

10. Gehe auf die Website **Blynk** und erstelle ein Konto. Melde dich anschließend an und folge der Einführung.

11. Wähle nun dein Gerät aus und klicke neben dem Namen deines Gerätes auf die drei Punkte. Gehe zu **“Dashboard bearbeiten”**.

12. Wähle den Reiter **“Datenströme”** aus, lösche alle bestehenden und erstelle einen neuen.



13. Gehe zum Reiter **“Web-Dashboard”**.

14. Lösche zunächst alle vorgeschlagenen Widgets und wähle stattdessen das Diagramm-Widget aus. Dieses kannst Du anpassen, indem Du auf das Zahnrad des Widgets klickst. Wähle als Datenstrom **“V1”** aus und bestätige mit speichern.

15. Lade nun die Blynk App herunter.

(a) iOS (b) Android

16. Öffne die Blynk App und melde dich an.
17. Verbinde deinen Mikrocontroller mit dem PC und öffne den Code für den pH-Sensor mit Arduino IDE, der dir auf **Github** zur Verfügung gestellt wird.
18. Füge in **Zeile 2 - 4** die Informationen deines Gerätes ein.
19. Ergänze außerdem die Zugangsdaten deines drahtlosen Netzwerks in **Zeile 18 / 19**.
20. Wenn du deinen pH-Wert nicht in einem Graphen darstellen lässt, kannst du den "delay"-Befehl in **Zeile 80** auskommentieren, indem du direkt davor "//" schreibst. Somit wird der Befehl nicht ausgeführt. Der "delay"-Befehl ist dafür da, um die Frequenz der Datenausgabe zu steuern, wenn du deine Werte aber nicht grafisch darstellen willst, kannst du die Frequenz über die Blynk-App steuern.
21. Wenn du fertig bist, kannst du den Code über den Haken oben links kompilieren (überprüfen) und anschließend auf den Mikrocontroller übertragen, in dem du den Pfeil rechts neben dem Haken anklickst.
22. Öffne anschließend den seriellen Monitor, welcher unter Werkzeuge zu finden ist.
23. Stellt im seriellen Monitor die Ausgabe auf **115200 Baud**, wie es auch im Code in **Zeile 50** steht. So stellt ihr sicher, dass verständliche Werte ausgegeben wurden.
24. Nimm nun den pH-Sensor, öffnet die Schutzkappe und reinige die pH-Elektrode mit destilliertem Wasser.
25. Bevor der pH-Sensor verwendet werden kann, muss dieser kalibriert werden. Stellt ihn dazu in eine der Kalibrierlösungen (pH 4 / pH 7).
26. Beginnt den Kalibrierprozess in dem ihr in die Kommandoleiste des seriellen Monitors den Befehl **"enterph"** eintippt und mit Enter bestätigt. Nun sollte der Kalibrierprozess aktiviert sein, kalibriert nun auf pH 7 oder pH 4 in dem ihr in die Kommandozeile den Befehl **"calph"** eingebt und mit Enter bestätigt. Ob es sich um die Lösung mit pH 4 oder pH 7 handelt, wird automatisch erkannt. Sobald im seriellen Monitor stabile Werte darstellt werden, kann der erste Durchlauf mit dem Befehl **"exitph"** beendet werden.
27. Reinige zum Abschluss die Elektrode mit destilliertem Wasser.

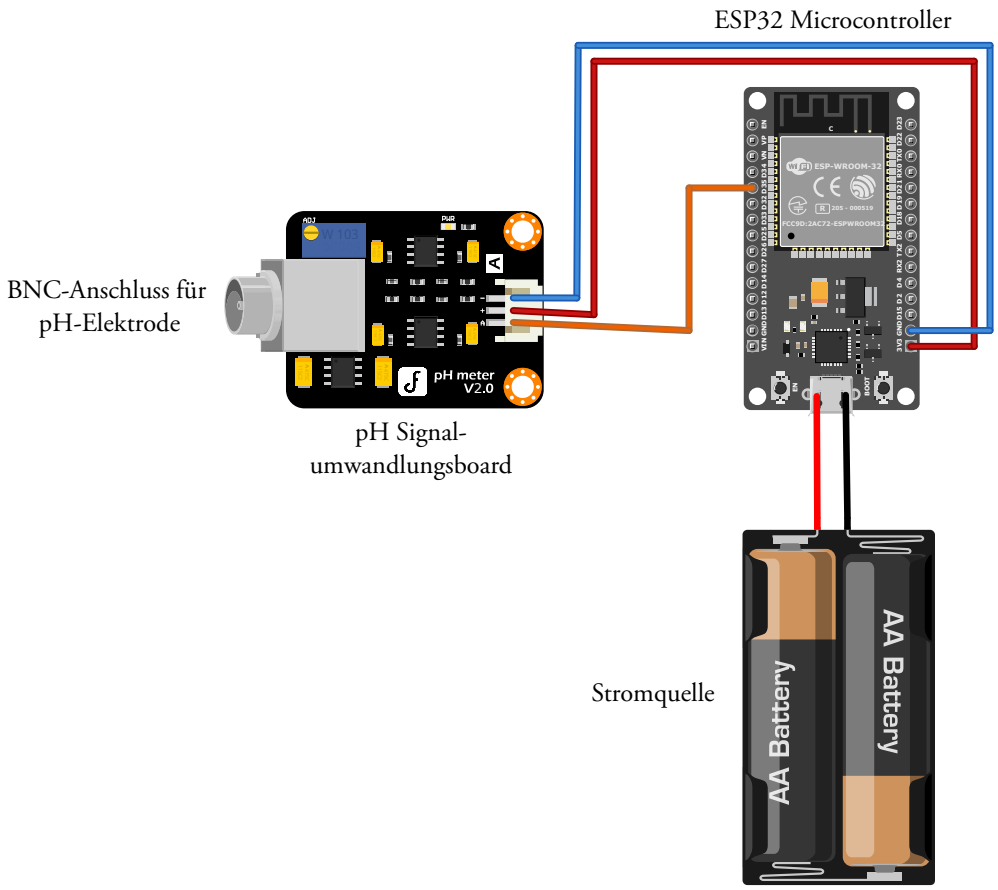
28. Wiederhole nun den Kalibrierprozess mit der anderen Kalibrierlösung.

29. Du kannst nun den Mikrocontroller von deinem PC trennen und die Stromversorgung mit der beigelegten Powerbank sicherstellen.

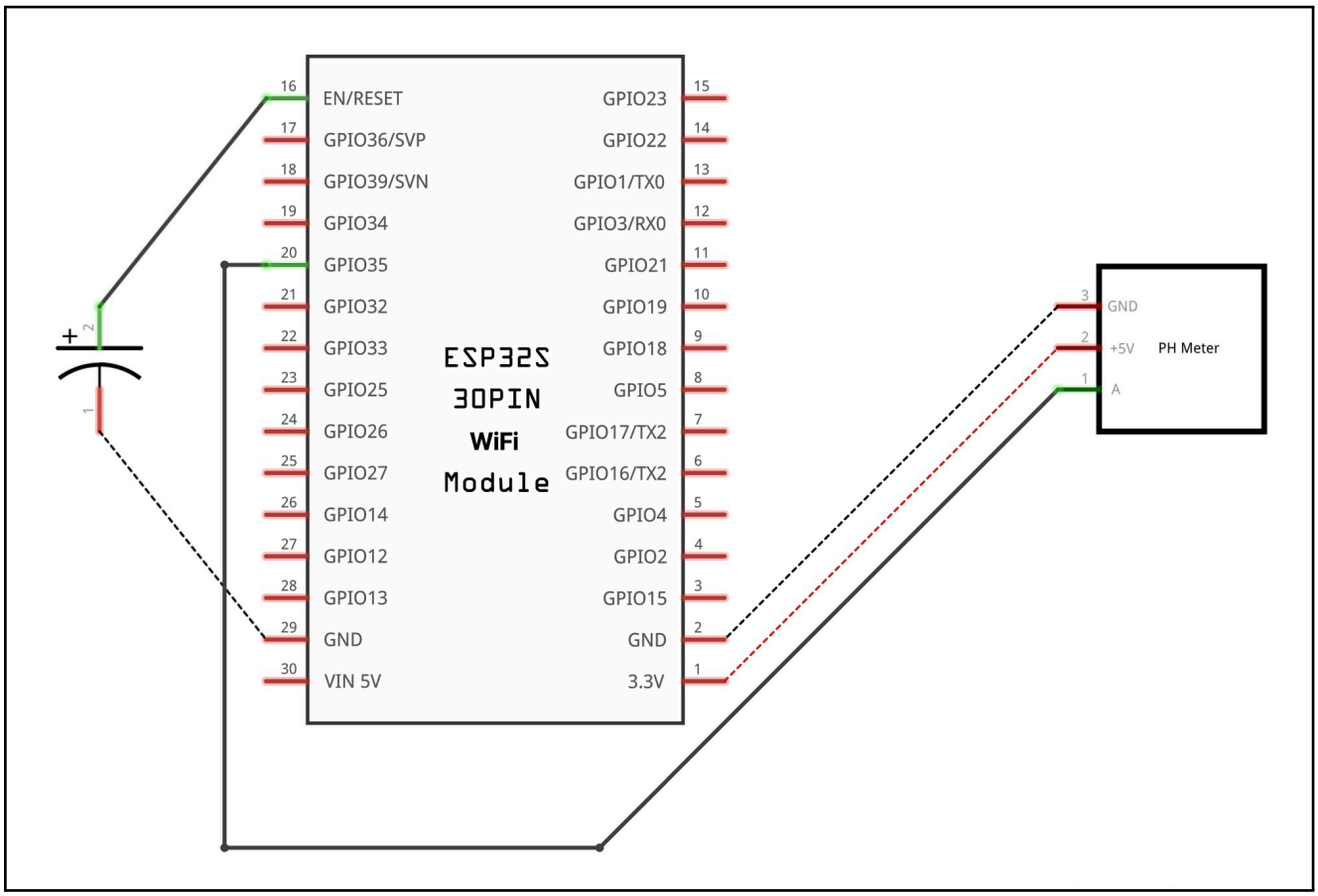
30. Nun solltest du die Werte des pH-Sensors auf deinem Smartphone oder auf der Website ablesen können.

```
/* Fill in information from Blynk Device Info here */
#define BLYNK_TEMPLATE_ID   ""
#define BLYNK_TEMPLATE_NAME ""
#define BLYNK_AUTH_TOKEN    ""


#include "DFRobot_ESP_PH.h"
#include <EEPROM.h>
DFRobot_ESP_PH ph;
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>


// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "";
char pass[] = "";

#define ESPADC 4096.0   //ESP Analog Digital Konvertierungswert
#define ESPVOLTAGE 3300 //ESP Spannungsversorgung
#define PH_PIN 35    //ESP Pin Nummer
float voltage, phValue, temperature = 25;

BlynkTimer timer;

// This function is called every time the Virtual Pin 0 state
   changes
BLYNK_WRITE(V4)
{
  // Set incoming value from pin V0 to a variable
  int value = param.asInt();

  // Update state
  Blynk.virtualWrite(V1, value);
}

// This function is called every time the device is connected to
   the Blynk.Cloud
BLYNK_CONNECTED()
{
  // Change Web Link Button message to "Congratulations!"
  Blynk.setProperty(V3, "offImageUrl", "https://static-image.nyc3
     .cdn.digitaloceanspaces.com/general/fte/congratulations.png")
     ;
  Blynk.setProperty(V3, "onImageUrl",  "https://static-image.nyc3
```

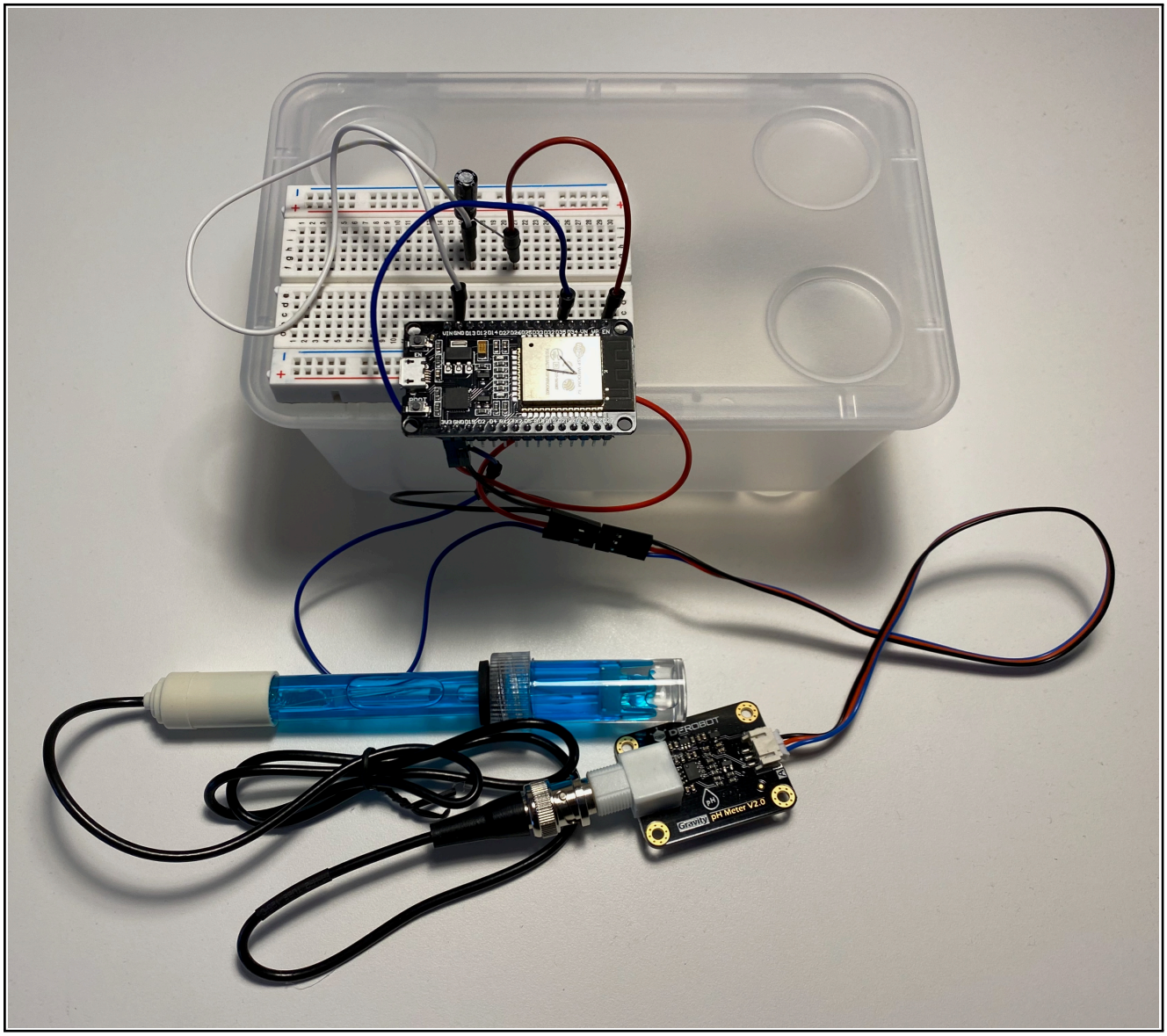
```
      .cdn.digitaloceanspaces.com/general/fte/
      congratulations_pressed.png");
44   Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-
      started/what-do-i-need-to-blynk/how-quickstart-device-was-
      made");
45 }
46
47 void setup()
48 {
49   //Debug console
50   Serial.begin(115200);
51   Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
52   EEPROM.begin(32);//Speichererlaubnis der Kalbibrierung
53   ph.begin();
54 }
55
56 void loop()
57 {
58   Blynk.run();
59    static unsigned long timepoint = millis();
60   if (millis() - timepoint > 1000U) //Zeitintervall: 1s
61   {
62     timepoint = millis();
63     voltage = analogRead(PH_PIN) / ESPADC * ESPVOLTAGE; //
        Ausgabe der Spannung
64     phValue = ph.readPH(voltage, temperature); // Berechnung des
        pH-Werts anhand
65     //der Spannung mit Temperaturkompensation (insofern möglich)
66     Blynk.virtualWrite(V1, phValue);
67
68     Serial.print("Spannung:␣");
69     Serial.print(voltage, 2);
70     Serial.println("␣mV");
71
72     // Serial.print("Temperatur:");
73     // Serial.print(temperature, 1);
74     // Serial.println(" °C");
75
76     Serial.print("pH:␣");
77     Serial.println(phValue, 1);
78     Serial.println();
79
80     delay(5000);
81  }
82   ph.calibration(voltage, temperature); // Kalibrierung via
      Seriellen Monitor
83         }
```



Verbindungsplan


16 EN/RESET
17 GPIO36/SVP
18 GPIO39/SVN
19 GPIO34
20 GPIO35
21 GPIO32
22 GPIO33
23 GPIO25
24 GPIO26
25 GPIO27
26 GPIO14
27 GPIO12
28 GPIO13
29 GND
30 VIN 5V
ESP32S
30PIN
WiFi
Module
15 GPIO23
14 GPIO22
13 GPIO1/TX0
12 GPIO3/RX0
11 GPIO21
10 GPIO19
9 GPIO18
8 GPIO5
7 GPIO17/TX2
6 GPIO16/TX2
5 GPIO4
4 GPIO2
3 GPIO15
2 GND
1 3.3V
3 GND
2 +5V
1 A
PH Meter

Schaltplan



Exemplarischer Aufbau