Dokumentation zur Batteriemessung

Inhaltsverzeichnis

Allgemeines

Der Bus besitzt zwei 12V Autobatterien die für die Stromversorgung für unterwegs genutzt werden sollen. Die Spannung der Batterien soll überwacht werden.

Ist-Zustand

Die Batterien werden entweder von einem Generator innerhalb des Busses, über die Lichtmaschine oder über die externe Einspeisung geladen. Die aktuelle Spannung wird nicht vom Bus überwacht und könnte leer gehen ohne es zu wissen.

Soll-Zustand

Die Spannung der Batterie soll überwacht. Ein ESP32 soll informationen der Spannung von beiden Batterien an den Raspberry Pi übertragen.

Vorraussetzungen der Batterie

Theoretische maximale Ladespannung an beiden Batterien in Summe: 28,8V

Spannung im Normalzustand: 26,6V

ESP 32

Eigenschaften der ESP32 Anschlüsse

Die Eingänge wie zum Beispiel pin14/GPIO36/Sensor_VP/VP können Werte zwischen 0V bis 3,3V erfassen. Die Werte in der Programmierung sind dann zwischen 0 bis 4095. Eine Umrechnung um von diesen Werten auf eine Spannung zurück ist also in dem Quellcode notwendig.Batteriemessung

Schaltung

Für die Messung der Spannung mit dem ESP32 benötigt man einen Spannungsteiler. Die Alternative wäre ein “Sensor-Spannungsteiler mit Zenerdiode”, allerdings ist bei dem eine höhere Gefahr eine zu hohe Spannung an den ESP32 anliegend zu haben.

Spannungsteiler

Hier ein Allgemeiner Spannungsteiler.

Er ist notwendig, da ein ESP32 maximal 3,3V an den GPIO-Kontakten anliegen dürfen. Da eine normale Autobatterie 12V und eine Busbatterie sogar 24V besitzen, muss die Spannung gesenkt werden.

Der ESP32 analoge Kontakt befindet sich zwischen R_1 und R_2.

Allgemeine Formeln:
$U_2 = (R_2 * U) / (R_1 + R_2)$
$R_2 = (U_2 * R_1) / (U - U_2)$

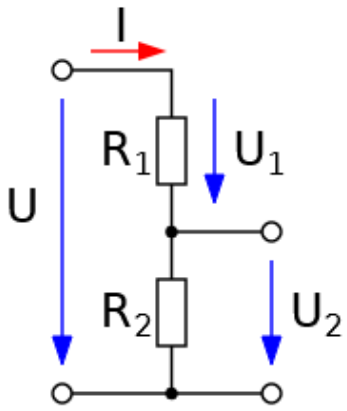

Schaubild 1: Spannungsteiler

Hinweis: Um eine Berechnung zu machen, muss man ein Widerstand festlegen. Die Widerstände sollten relativ hoch sein, da ansonsten ein sehr hoher Ruhestrom fließt.

Zustände einer Batterie

Bei einer 24V Batterie	
Zustand	**Bereich**
Tiefenentladung/Defekte Batterie	Bis 24,04V
Entladung	24,05V – 24,19V
Teilladung	24,20V – 25,00V
Volle Ladung oder Ladezustand	Ab 25,01V

Berechnungen für Spannungsteiler

Ergebnisse beziehen sich auf Schaubild 1: Spannungsteiler. Gelb bezieht sich auf Testmessungen um sicherzustellen das keine Überlastung erfolgt. Grün bezieht sich auf die Messung mit der richtigen Batterie.	
U = 25V U_2 = 1,1V	**R_1 = 100‘000Ω** **$R_2 \leq$ 4‘700Ω**
U = 30V $U_2 \leq$ 3V	**R_1 = 100‘000Ω** **R_2 = 11‘000Ω**

Entgültige Schaltung

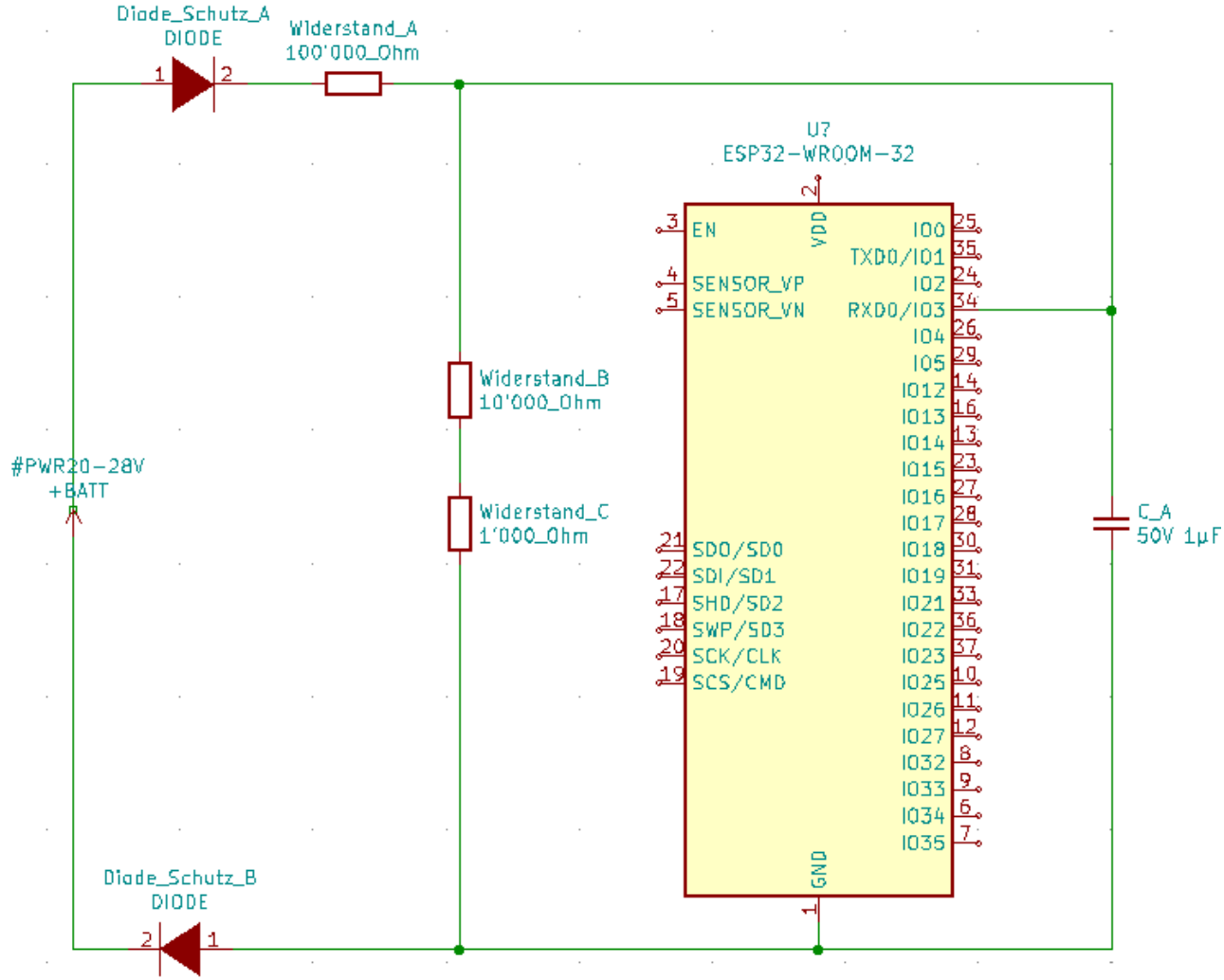

Schaubild 2: Schaltplan des Spannungsteilers am ESP32

Testmessung

Da man nur Messung haben wird im Bereich zwischen 24,04V bis 28,8V reicht ein Messbereich von 20-30V.

Deswegen wurde mit der Schaltung Probemessungen gemacht, bei dem die Spannung eindeutig ist. So können Referenzwerte mit einer Spannung ermittelt werden. Für die Referenzwerte wurden 10’000 Testmessungen genommen und dadurch ergeben sich die folgenden Messergebnisse.

Spannung	Mindestwert	maximalwert	bestwert
20	2187	2199	2192
21	2304	2320	2311
22	2421	2438	2432
23	2544	2570	2559
24	2672	2688	2679
25	2795	2811	2800
26	2925	2914	2928
27	3056	3079	3071
28	3207	3223	3126

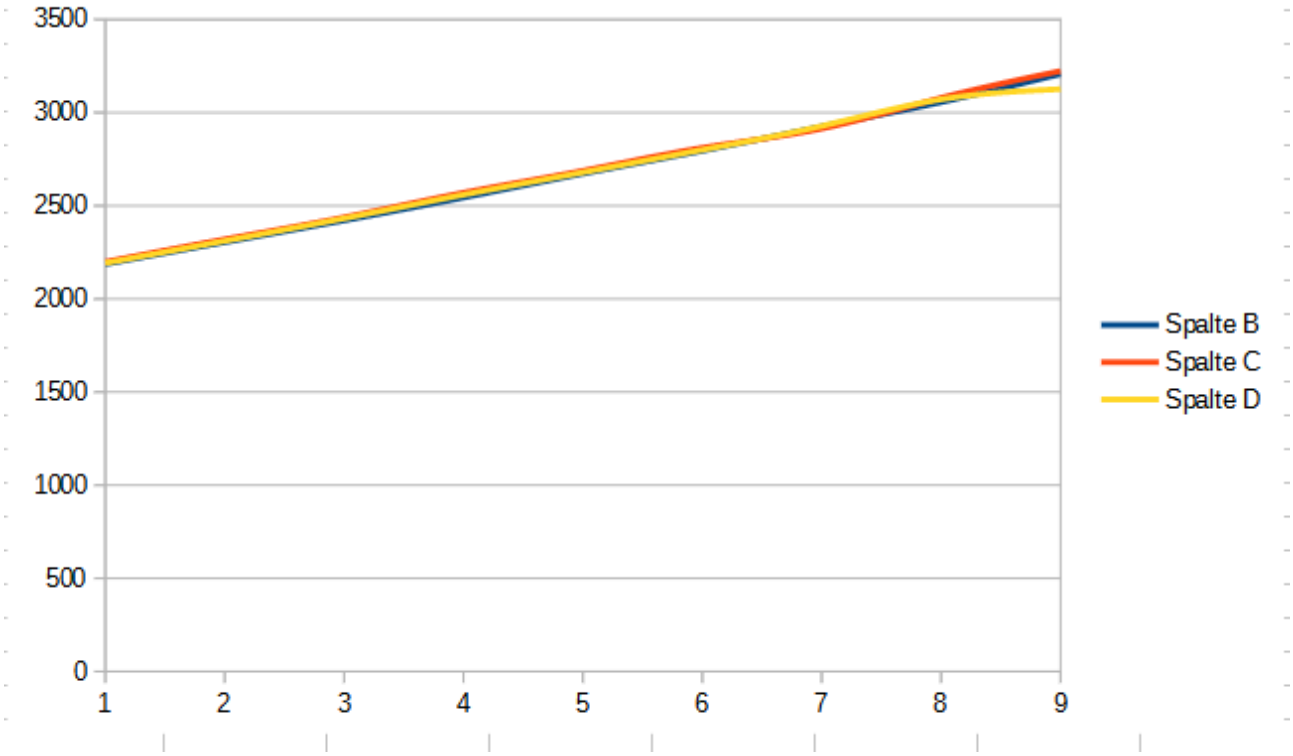

Schaubild 3: Referenzwert Messergebnisse

Bei der Testmessung konnte nur eine Testspannung von maximal 28V erzeugt werden, deswegen sind die Spannungen 29V und 30V Schätzwerte.

Quellcode

Hier wird der Quellcode für den ESP32 hinterlegt.

Wichtig ist zu beachten:

- Zugangsdaten für das Empfängernetzwerk müssen noch eingegeben werden
- Die Methode “vTabEinrichtung()” besitzt die Referenzwerte für die Spannungsmessung

```
#include <Wire.h>
#include <SPI.h>
#include <WiFi.h>
#include "PubSubClient.h" // Connect and publish to the MQTT broker. Installation:
Arduino(Sketch->Include Library->Manage Libraries...->Search(PubSubClient) and Install)

#define PIN_INPUT_V 34    // Pin 34 - Messung am ESP

/*ESP-Einstellungen*/
const int waitTime = 1;       /* Wartezeit bis der Zyklus sich wiederholt*/
const int sleepTime = 60000;  /* Zeit in Millisekunden(Standart= 60000 == Eine Minute)
zwischen den Messungen und Senden des Spannungsergebnisse*/
const int pTime = 1000;       /* Pausenzeit*/
const int fehlerZeit = 5000;  /* Wartezeit wenn ein Fehler entdeckt wurde*/
int restartCounter = 100;     /* Anzahl an Sendungen, die einen neustart des ESP ausführen*/

/*WIFI-Configuration*/
const char* ssid = ""; //SSID of the Wifi
const char* password =  ""; //Password of the Wifi
// MQTT
String clientID = "Batterie"; // IMPORTANT! MQTT client ID -> Name der Batteriemessung
const char* mqtt_server = "192.168.0.1";  // IP of the MQTT broker
const char* mqtt_username = ""; // MQTT username
const char* mqtt_password = ""; // MQTT password
// other
String voltage_topic = "home/" + clientID + "/voltage"; // MQTT topic
// Initialise the WiFi and MQTT Client objects
WiFiClient wifiClient;
// 1883 is the listener port for the Broker
PubSubClient client(mqtt_server, 1883, wifiClient);

/*Messbereiche*/
int zustand = 0;                  /*|0=Wartend|1=Messung laeuft|2=Ende|-1=Ueberspannung|-
2=Unterspannung|*/
int counter = 0;                  /*Zaehler fuer den Wert*/
const int messAnzahl = 3000;        /*Anzahl von Messungen */
const int moeglicheErg = 4095;      /*Maximaler Messwert am ESP32*/
int messung[moeglicheErg] = {0};    /*Array mit den Messungen: |Value|Anzahl|*/
```

```
int spannungsTabelle[11] = {0};     /*Spannungstabelle: Index+20 ist die Spannung und der Wert ist der ideal-Messwert*/
bool messungErfolgreich = false;    /*Wenn Messung erfolgreich wird dieser Wert True gesetzt, nach dem Senden der Information wieder aus false*/
const float spannungsabfall = 20.0; /*Spannung die ueber die Schaltung abfaellt*/
const float ladespannung = 26.6;    /*Spannung die beim Laden vorhanden ist*/
float ergebnisSpannung = 0.0;       /*Hier wird die berechnete Spannung abgelegt*/
boolean start = true;               /*Erster Durchlauf mit Messung oder wenn eine Fehlerhafte Messung erfolgt ist*/

/*Einrichtung der WIFI-Art und Verschlüsselung*/
String translateEncryptionType(wifi_auth_mode_t encryptionType) {
  switch (encryptionType) {
    case (WIFI_AUTH_OPEN):
      return "Open";
    case (WIFI_AUTH_WEP):
      return "WEP";
    case (WIFI_AUTH_WPA_PSK):
      return "WPA_PSK";
    case (WIFI_AUTH_WPA2_PSK):
      return "WPA2_PSK";
    case (WIFI_AUTH_WPA_WPA2_PSK):
      return "WPA_WPA2_PSK";
    case (WIFI_AUTH_WPA2_ENTERPRISE):
      return "WPA2_ENTERPRISE";
  }
}

/*Neustart des ESP*/
void restartEsp() {
  Serial.println("Restarting in 5 seconds");
  delay(5000);
  ESP.restart();
}

/*Funktion führt Neustart des ESP aus, wenn der restartCounter 0 erreicht*/
void restarter(){
  restartCounter--;
  //Serial.println("restartCounter:" + String(restartCounter));
  if(restartCounter <= 0){
    restartEsp();
  }
}


void connectToNetwork() {
  WiFi.begin(ssid, password);
  WiFi.setHostname(clientID.c_str());
```

```
  int i = 0;
  while (WiFi.status() != WL_CONNECTED) {
    if(i >= 100) {
      restartEsp();
    }
    delay(pTime);
    Serial.println("Establishing connection to WiFi..");
    i++;
  }

  Serial.println("Connected to network");
}


void disconnectToNetwork() {

  int i = 0;
  while (WiFi.status() == WL_CONNECTED) {
    if(i >= 100) {
      restartEsp();
    }
    delay(pTime);
    Serial.println("Disestablish WiFi connection ..");
    WiFi.disconnect(true);
    i++;
  }

  Serial.println("Disonnected from network");
}

void connect_MQTT(){
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // Connect to MQTT Broker
  // client.connect returns a boolean value to let us know if the connection was successful.
  // If the connection is failing, make sure you are using the correct MQTT Username and
Password (Setup Earlier in the Instructable)
  if (client.connect(clientID.c_str(), mqtt_username, mqtt_password)) {
    Serial.println("Connected to MQTT Broker!");
  }
  else {
    Serial.println("Connection to MQTT Broker failed...");
  }
}

/*Funktion sendet den aktuellen Spannungswert an die Zieladresse*/
```

```
void sendVoltage(){
  connectToNetwork();
  Serial.setTimeout(2000);
  connect_MQTT();
  Serial.setTimeout(2000);

  float voltage = ergebnisSpannung;// Spannung wird gesendet
  Serial.println("Voltage = " + String(voltage) + "V");

  // MQTT can only transmit strings
  String voltage_v=String((float)voltage);

  //Serial.println();
  // PUBLISH to the MQTT Broker (topic = Temperature, defined at the beginning)
  if (client.publish(voltage_topic.c_str(), String(voltage_v).c_str())) {
    Serial.println("Voltage sent!");
  }
  // Again, client.publish will return a boolean value depending on whether it succeeded or not.
  // If the message failed to send, we will try again, as the connection may have broken.
  else {
    Serial.println("Voltage failed to send. Reconnecting to MQTT Broker and trying again");
    client.connect(clientID.c_str(), mqtt_username, mqtt_password);
    delay(10); // This delay ensures that client.publish doesn't clash with the client.connect call
    client.publish(voltage_topic.c_str(), String(voltage_v).c_str());
  }
  client.disconnect();  // disconnect from the MQTT broker
  disconnectToNetwork();  //disconnect from WiFi
  delay(pTime);       // print new values every 1 Minute
}

/**
 * Daten der Spannungswerte werden eingerichtet
 * Der Index steht für die Einerstelle der Spannung
 * Example: Index= 3 => 3V + Grundspassung(20V)
 * Example2:
 * ...spannungsTabelle[3] = 2559;
 * spannungsTabelle[4] = 2679;...
 * Wird ein Wert gemessen(2439) ergibt dies eine Spannung zwischen >22V und <23V, da es über
den Indexwert 2432 und unter 2559 liegt.
 * Des weiteren wird, danach über eine lineare Funktion ein Prozentwert mit dem Differenzwert
berechnet um die 2Kommastellen der Spannung zu berechnen.
 */
void vTabEinrichtung(){
  spannungsTabelle[0] = 2192;
  spannungsTabelle[1] = 2311;
  spannungsTabelle[2] = 2432;
  spannungsTabelle[3] = 2559;
  spannungsTabelle[4] = 2679;
```

```
  spannungsTabelle[5] = 2800;
  spannungsTabelle[6] = 2928;
  spannungsTabelle[7] = 3071;
  spannungsTabelle[8] = 3216;
  spannungsTabelle[9] = 3331;     //Schaetzwerte, da Testmessung war nur bis 28V moeglich
  spannungsTabelle[10] = 3474;    //Schaetzwerte, da Testmessung war nur bis 28V moeglich
  }

/*Gibt die maximal zulaessige Spannung zurueck*/
float getMaxSpannung(){
  return (spannungsabfall + ((sizeof(spannungsTabelle)/sizeof(int)) - 1));
  }

/*Gibt die minimale zulaessige Spannung zurueck*/
  float getMinSpannung(){
    return spannungsabfall;
    }

/**
 * Messwert wird hinzugefuegt
 * Funktion erstellt Messungen und zaehlt den counter hoch
 * Wenn eine Anzahl vom Counter erreicht wurde wird false zurueck gegeben. Dies dient zur
Signaisierung, dass ein Messzyklus abgeschlossen wurde.
 */
bool phaseMessung(){
  if(counter < messAnzahl){
    int rawValue = analogRead(PIN_INPUT_V); //Spannungsmessung erfolgt hier
    //Serial.println("Messung(" + String(counter) + "/" + String(messAnzahl) + ") von rawValue=" +
String(rawValue));
    messung[rawValue] = messung[rawValue] + 1;
    counter++;
    return true;
  }
  else{
    counter = 0;
    return false;
    }
  }

  /**
 * Löscht die Eintraege vom Array Messung
 */
void messungReset(){
  for(int i = 0; i < moeglicheErg; i++){
      messung[i] = 0;
    }
  }
```

```
/**
 * Summiert alle Gemessenen Werte und berechnet dann den Querschnitt Werte
 */
int bestMessWert(){
  int summe = 0;
  int wert = 0;
  for(int i = 0; i < moeglicheErg; i++){
    if(messung[i] > 0){
        summe = summe + (messung[i] * i);
    }
  }
  wert = summe / messAnzahl;
  Serial.println("Bester Messwert:" + String(wert));
  return wert;
}

/**
 * Methode berechnet eine Range von valueStart und valueEnd und sucht in diesem Bereich den
Value
 * Als Ergebnis werden die Nachkommazahl der Spannungsergebnis zuruech gegegen
 */
float prozentWert(int value, int valueStart, int valueEnd){
  int range = 0;
  float erg = 0;
  if(valueEnd > value && value >= valueStart){
      range = valueEnd - valueStart;
      erg = ((float)value - (float)valueStart) / (float)range;
      //Serial.println("Ergebnis vom prozentWert():" + String(valueEnd) + "-" + String(valueStart) +
"=" + String(range) + "=> das sind 100%;");
      //Serial.println("float_erg:" + String(erg) + "=((float)value - valueStart):" + String((float)value
- valueStart) + " / (float)range:" + String((float)range) + ";");
      return erg;
  }
  return -1.0;
}

/**
 * Wandelt den am häufigsten erhaltenen Messwert in einen Spannungswert um
 */
float ergebnisBerechnung(){
  int rohWert = 0;    // Wert der am ESP-Eingang gemessen wurde
  int index = 0;    // index = Einerstelle des Spannungswertes
  int indexwert = 0;  // Wert die der ESP bei einer bestimmten Spannung misst
  float spannung = 0.0; // Ergebnis der Spannungsberechnung
  float kommaWert = 0.0;// Nachkommazahl der Spannungsmessung

  rohWert = bestMessWert();
  if(rohWert >= moeglicheErg){
```

```
    Serial.println("Fehler bei Messwert. (Maximaler Messwert=" + String(moeglicheErg) + ";
Messwert=" + String(rohWert) + ".");
   zustand = -9;
   return -1.0;
  }
  indexwert = spannungsTabelle[index];    // niederigster zulaessiger Wert
  //Serial.println("Bester Wert rohWert=" + String(rohWert) + "; indexWert=" + String(indexwert));
  //Serial.println("Anzahl von Spannungswerten=" +
String((sizeof(spannungsTabelle)/sizeof(int))));
  if(rohWert >= indexwert && rohWert <=
spannungsTabelle[(sizeof(spannungsTabelle)/sizeof(int))-1]){          // rawValue ist unter 20V oder
ueber den maximalWert
   while(index < (sizeof(spannungsTabelle)/sizeof(int))){
    index = index + 1;
    //Serial.println("sizeof(spannungsTabelle)=" + String((sizeof(spannungsTabelle)/sizeof(int))));
    if(index >= (sizeof(spannungsTabelle)/sizeof(int))){  //Wenn der gemessene Wert den maximal
Wert der Spannungstabelle uebersteigt
      Serial.println("Fehler beim Finden des Spannungswertes.");
      //break;
      return -1.0;
    }
    else{
      indexwert = spannungsTabelle[index];
      if(rohWert <= indexwert){   //sobald der Wert aus der Tabelle groesser ist als der rawValue
stoppt die Suche
        //Serial.println("Value(" + String(rawValue) + "); kleinerer Spannungswert(" + String(index-
1) + ")='" + String(spannungsTabelle[index-1]) + "'; höherer Spannungswert(" + String(index) +
")='" + String(spannungsTabelle[index]) + "';");
        kommaWert = prozentWert(rohWert, spannungsTabelle[index-1], spannungsTabelle[index]);
//Erwarteter Wert ist 0 bis 1
        //Serial.println("Prozentwert=" + String(spannung) + " index=" + String(index-1));
        if(kommaWert >= 0.0 && kommaWert <= 1.0){
          //Serial.println("Berechung: Grundwert=" + String(spannungsabfall) + "V + " +
String(index-1) + "V + " + String(kommaWert) + "V" );
          spannung = spannungsabfall + (index-1) + kommaWert ;    // Zehner + Einer + Kommawert
=> Spannungswert
          ergebnisSpannung = spannung;
          return spannung;
        }
        break;
      }
    }
   }
  }
  else{
   if(rohWert < indexwert){  //Spannung ist zu niedrig
    return (getMinSpannung() -1);
   }
```

```
    if(rohWert > indexwert){  //Spannung ist zu hoch
      return (getMaxSpannung() + 1);
    }
  }
  Serial.println("Fehler beim Finden des Spannungswertes.");
  return -1.0;
}

/**
 * Methode wandelt Float-Zahlen in ein String/Char-Kette um
 */
char *floatToString(float zahl){
  static char ausgabe[50];
  sprintf(ausgabe, "%2.5f", zahl);
  return ausgabe;
}

/**
 * Funktion ist für den Gesamtprozess verantwortlich
 */
void prozess(){
  float spannung = 0;
  switch(zustand){
    case 0:     //Programm wartet auf Input
      Serial.println("Warten...");
      if(!start){
      delay(sleepTime);
      }
      else{
        start = false;
      }
      zustand = 1;  //Wechsel zum Messzustand
      Serial.println("Messung läuft...");
      break;
    case 1:     //Messungsprozess wird ausgefuehrt
      if(!phaseMessung()){
        Serial.println("Messung wurden erfolgreich abgeschlossen.");
        spannung = 0.0; //Reset des alten Spannungswertes
        spannung = ergebnisBerechnung();
        messungReset();
        if(spannung < getMinSpannung()){
          zustand = -2;
        }
        else if(spannung > getMaxSpannung()){
          zustand = -1;
        }
        else{
          Serial.println("Ergebnis der Spannung= " + String(spannung) + "V.");
```

```
        if(spannung > ladespannung){
          Serial.println("Ladespannung wurde ermittelt. Batteriestand kann nicht im Ladezustand
ermittelt werden.");
        }
        zustand = 2;
        messungErfolgreich = true;
       }
       }
    break;
   case 2:   //Senden der Messung
     if(messungErfolgreich){ //Neues Ergebnis wird verschickt
       Serial.println("Senden der Spannung(" + String(ergebnisSpannung) + ")");
       sendVoltage();
       }
     else{ //Es existiert noch kein neues Ergebnis
       Serial.println("Senden einer Fehlermeldung");
       Serial.println("Fehlemeldungen werden nicht gesendet.");
     }
     Serial.println("Senden abgeschlossen.");
     restarter();  //Nach einer Anzahl an Sendung wird der ESP neugestartet
     zustand = 0;
     break;
   case -1:  //Fehlermeldung, Messung unterbrochen. Ueberhoehte Spannung ermittelt.
     Serial.println("===> FEHLER(-1): Spannung ist zu hoch.");
     Serial.println("ACHTUNG! Spannung ist zu hoch. ESP könnte schaden nehmen.
Maximalspannung von " + String(getMaxSpannung()) + "V ist überschritten.");
     delay(fehlerZeit);
     zustand = 2;
     break;
   case -2:  //Fehlermeldung, Messung unterbrochen. Spannung zu niedrig.
     Serial.println("===> FEHLER(-2): Spannung ist zu niedrig.");
     Serial.println("Spannung ist kleiner als die Mindestspannung von " + String(spannungsabfall)
+ "V. Messung nicht möglich.");
     delay(fehlerZeit);
     zustand = 2;
     break;
   case -9:  //Fehlermeldung, unbekannter Fehler
     Serial.println("===> FEHLER(-9): Unbekannter Fehler ermittelt.");
     delay(fehlerZeit);
     zustand = 0;
     break;
  }
}

/*Konfiguration des ESP*/
void setup() {
  //ESP Einrichtung
  Serial.begin(9600); //Baudrate fuer die Ausgabe der Konsole
```

```
  pinMode(PIN_INPUT_V, INPUT);  //Deklarierung zum Eingang

  //Wifi-Einrichtung
  WiFi.disconnect(true);
  delay(pTime);
  Serial.println(WiFi.macAddress());
  Serial.println(WiFi.localIP());

  //Spannungsmessung-Einrichtung
  vTabEinrichtung();  //Werte der Spannungsmessung wird eingerichtet
}

/*Loop-Funktion des ESP*/
void loop() {
  prozess();
  delay(waitTime);
}
```

Probleme

Funktverbindung

Die Funkverbindung aus dem Batteriefach des Buses ist sehr schlecht. Scheinbar ist das Batteriefach abgeschirmt. Die Messung und der Verbindungaufbau ist innerhalb des Buses möglich. Die genaue ordentliche Positionierung des ESP32 und des Empfänger-Wlans müssen eventuell genau abgestimmt werden.

Spannungsschwankungen

Eventuelle Spannungsschwankung durch die grobe Lichtmaschine könnte Problematisch werden, wenn es eine höhere Spannung als 30V erzeugt. Eventuell kann man eine Schutzschaltung mit Zener-Dioden vor der Spannungsmessung erstellen. Eventuelle Spannungsschwankungen konnten nicht ermittelt werden, da diese Schaltung nicht im fahrenden Zustand des Buses ausprobiert wurde.