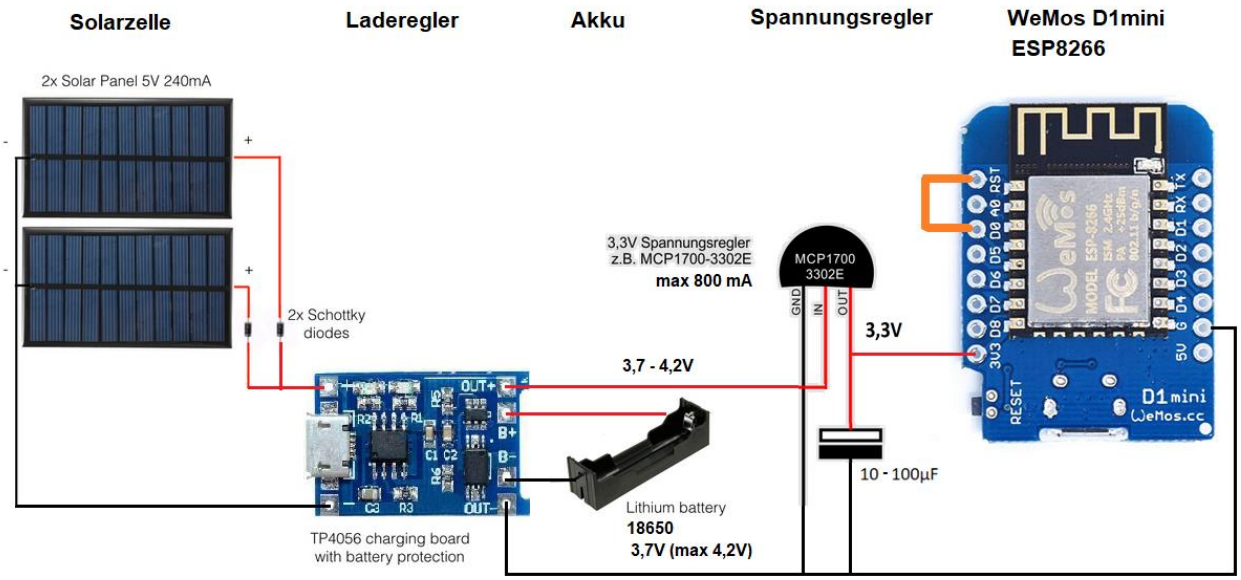
Der ESP32 und der WeMos D1 mini im Akku-Betrieb

Beschreibung

Wir möchten einen ESP32 oder einen ESP8266 WeMos D1 mini mit einem Lithium Akku und einem Solarmodul betreiben. Wenn der ESP mit einer Batterie versorgt wird, dann ist es sinnvoll den „deep sleep“ zu verwenden. Über den Analogeingang des ESP werden wir die Ladespannung messen.

Basis-Schaltung



Hardware

Liste der Bauteile

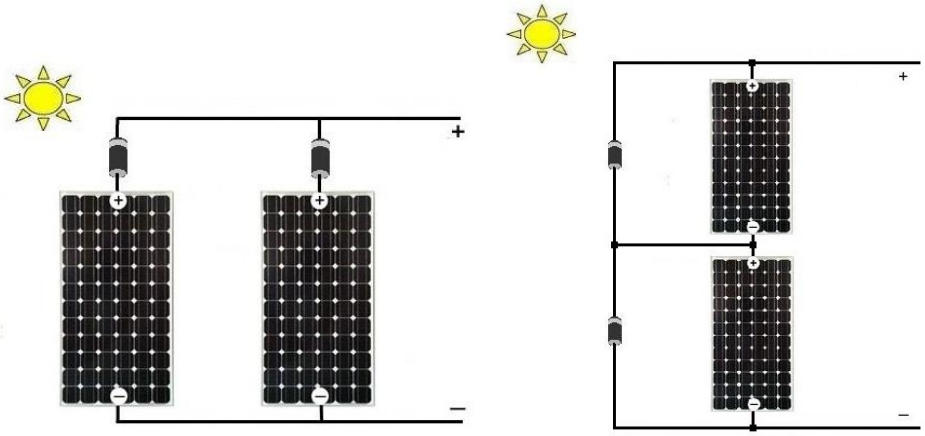
- ESP8266 (Wemos D1 mini) oder ESP32
- 2x Mini Solar Panel (5-6V 1,2W)
- Optional: 2x Schottky Diode (größer als 1A) zB 1N5819
- Lithium Li-ion Batterie 18650 (3,7V)
- Batterie Halter
- Batterie Laderegler TP4056
- Spannungsregler 3,3V (MCP1700-3302E)
- Widerstände für Spannungsteiler 27kOhm, 100 kOhm

Wichtig! Bevor mit dem Bau der Schaltung begonnen wird, sind die Datenblätter der verwendeten Bauteile durchzulesen um die maximalen Belastungswerte zu prüfen.

Solarpanel

Werden mehrere Solarpanele **parallel** (+ auf + und – auf –) geschaltet, dann wird bei gleicher Spannung der Strom der Module addiert. Es ist ratsam Sperr-Dioden (Schottky-Dioden) zu verwenden um Ausgleichsströme bei unterschiedlichen Panel-Spannungen zu verhindern.

Werden mehrere Solarpanele **seriell** (hintereinander in Reihe) geschaltet, dann wird bei gleichem Strom die Spannung der Module addiert. Es ist nötig Bypass-Dioden (Schottky-Dioden) zu verwenden um bei unterschiedlicher Sonneneinstrahlung die beschatteten Module wegzuschalten.



Batterie Laderegler TP4056

Der Laderegler verhindert ein Überladen der Batterie und hat eine Kurzschlussüberwachung und einen Verpolungsschutz. Eine rote LED leuchtet, wenn die Batterie geladen wird und die blaue LED leuchtet, wenn die Batterie voll aufgeladen ist. Mit einem 5V-Netzteil kann die Batterie geladen werden, wenn zu wenig Sonne ist.

Um nun die Betriebsspannung von 3,3V zu erreichen, benötigen wir noch einen Spannungsregler.

Spannungsregler MCP1700-3302

MICROCHIP

MCP1700

Low Quiescent Current LDO

Features:

- 1.6 µA Typical Quiescent Current
- Input Operating Voltage Range: 2.3V to 6.0V
- Output Voltage Range: 1.2V to 5.0V
- 250 mA Output Current for Output Voltages ≥ 2.5V
- 200 mA Output Current for Output Voltages < 2.5V
- Low Dropout (LDO) Voltage
  - 178 mV Typical @ 250 mA for V_{OUT} = 2.8V
- 0.4% Typical Output Voltage Tolerance
- Standard Output Voltage Options:
  - 1.2V, 1.8V, 2.5V, 2.8V, 3.0V, 3.3V, 5.0V
- Stable with 1.0 µF Ceramic Output Capacitor
- Short Circuit Protection
- Overtemperature Protection

General Description:

The MCP1700 is a family of CMOS low dropout (LDO) voltage regulators that can deliver up to 250 mA of current while consuming only 1.6 µA of quiescent current (typical). The input operating range is specified from 2.3V to 6.0V, making it an ideal choice for two and three primary cell battery-powered applications, as well as single cell Li-Ion-powered applications.

The MCP1700 is capable of delivering 250 mA with only 178 mV of input to output voltage differential (V_{OUT} = 2.8V). The output voltage tolerance of the MCP1700 is typically ±0.4% at +25°C and ±3% maximum over the operating junction temperature range of -40°C to +125°C.

Output voltages available for the MCP1700 range from 1.2V to 5.0V. The LDO output is stable when using only 1 µF output capacitance. Ceramic, tantalum or aluminum electrolytic capacitors can all be used for

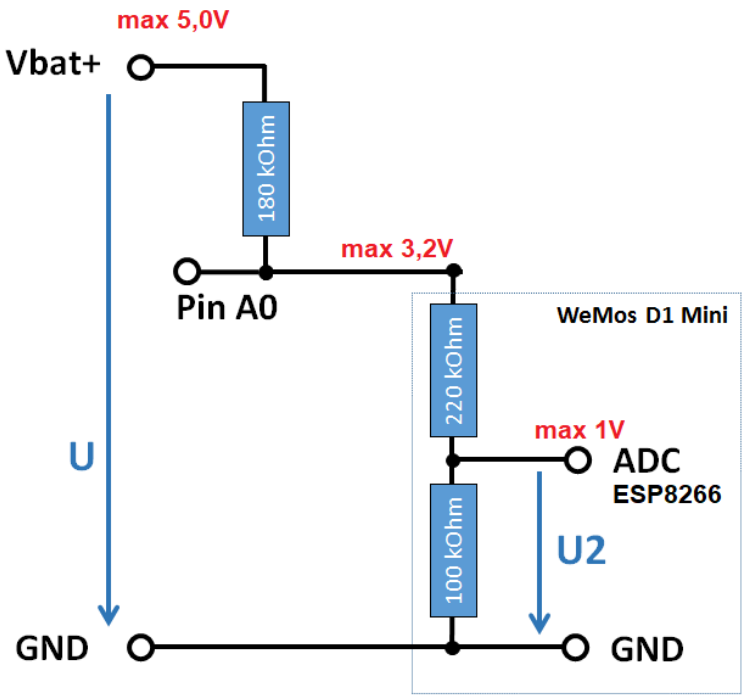
Eine gute Alternative ist der HT7333-A

Zu beachten ist, dass der Spannungsregler mit einer sehr geringen Spannungsdifferenz von Eingang zu Ausgang auskommt. Somit kann der Akku sehr gut ausgenutzt werden.

Batterie Spannung überwachen mit Wemos D1mini

Für die Messung der Batteriespannung benötigen wir einen Spannungsteiler, da die Batteriespannung für den Analog-Eingang des ESP zu hoch ist.

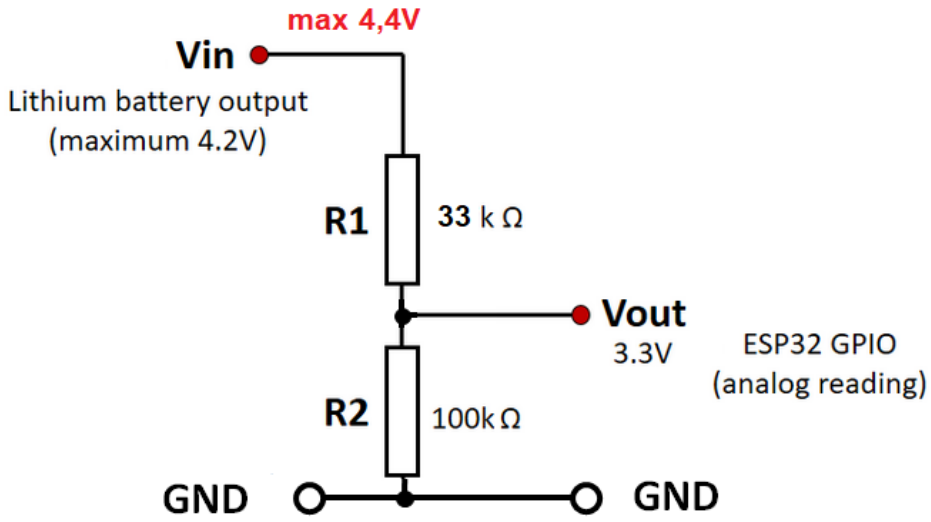
- Der ESP8266 kann am Analog-Eingang nur 0V bis maximal 1V messen.
- Auf dem Wemos D1 mini ist dadurch schon ein Spannungsteiler zwischen A0 und GND verbaut. (ADC0 = Pin2 = A0)
- Damit U2 nicht größer als 1V wird, darf durch die 100 kOhm nur maximal 0,000.01A fließen
  - 100.000 Ohm * 0,000.01 A = 1 V
- Mit dem Ohmschen Gesetz rechnen wir uns den Vorwiderstand aus.
  - 220.000 Ohm * 0,000.01 A = 2,2 V
  - 180.000 Ohm * 0,000.01 A = 1,8 V
  - 1 + 2,2 + 1,8 = 5V



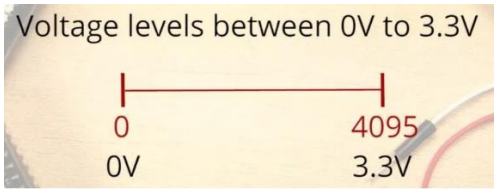
Batterie Spannung überwachen mit ESP32

Für die Messung der Batteriespannung benötigen wir einen Spannungsteiler, da die Batteriespannung für den Analog-Eingang des ESP zu hoch ist.

- Der ESP32 kann am Analog-Eingang nur 0V bis maximal 3,3V messen.
- Damit U2 nicht größer als 3,3V wird, darf durch die 100 kOhm nur max 0,000.033A fließen
  - 100.000 Ohm * 0,000.033 A = 3,3 V
- Mit dem Ohmschen Gesetz rechnen wir uns den Vorwiderstand aus.
  - 33.000 Ohm * 0,000.033 A = 1,089 V
  - 3,3 + 1,1 = 4,4 V



- Der ESP32 kann an seinen Analog-Eingängen maximal 3,3 V verkraften.
- Der Analog-Digital Wandler hat eine 12-Bit Auflösung. Das bedeutet, dass die maximale eingelesene Spannung in 4095 Schritten aufgeteilt wird.



- Beachte, dass die Spannung nur von 0,1 V bis 3,2 V gemessen werden kann.
- Beachte, dass ADC2 können nicht verwendet werden, wenn WiFi benutzt wird.

Nun starten wir mit der Programmierung. Dazu starten wir die Arduino IDE oder Visual Studio Code. Wie das funktioniert findest du in den Grundlagen Beschreibungen.

Programm-Listing

```
// Spannung messen mit Wemos D1 mini an A0
// oder ESP32 an GPIO 34

const int POTPIN_ESP32 = 34;     // pin34 - ESP32
const float FAKTOR_ESP32 =  133/100;  //  500/100 Widerstandsverhältis anpassen

const int POTPIN_D1_mini = A0;  // A0 - D1 mini
const float FAKTOR_D1_mini =  4.85;  //  500/100 Widerstandsverhältis anpassen


void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
}

void loop() {
// für Wemos D1 mini
    int valueA0 = analogRead(POTPIN_D1_mini);     // analog Wert einlesen 0..1024
    double valueBatt = (float)valueA0 / 1024 * FAKTOR_D1_mini;
    Serial.println("Voltage: " + String(valueBatt, 3) + "V");

// für ESP32
/*
    int valueA0 = analogRead(POTPIN_ESP32);     // analog Wert einlesen 0..4092
    double valueBatt = (float)valueA0 / 4092 * FAKTOR_ESP32;
    Serial.println("Voltage: " + String(valueBatt, 3) + "V");
*/
}
```

Lesen der Betriebsspannung VCC

Wenn die Betriebsspannung gemessen werden soll, dann den Spannungsteiler entfernen. Die Betriebsspannung wird intern gemessen.

ADC_MODE(ADC_VCC); vor der Funktion „setup()“ einbauen.

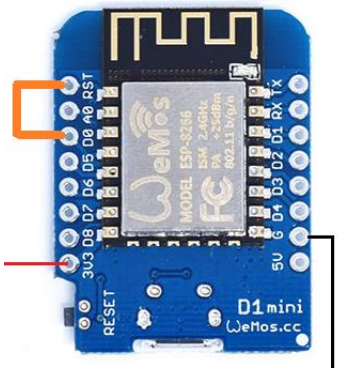
```
//Spannung auslesen vorbereiten, erfolgt vor void setup()
    ADC_MODE(ADC_VCC);

 // genaue Spannung der Stromquelle, A0 kann nicht verwendet werden
  uint16_t my_getVcc_Voltage = ESP.getVcc();
  float_t my_Voltage_calculated = ((float)my_getVcc_Voltage/1024.0f);
  String Ubatt = String(my_Voltage_calculated, 3);
  Serial.println(Ubatt);
```

ESP.getVcc() und analogRead(A0) kann nicht gleichzeitig verwendet werden.

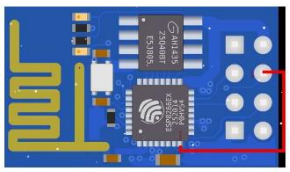
Deep Sleep nach bestimmter Zeit beenden

```
1
2  void setup() {
3    // put your setup code here, to run once:
4    Serial.begin(9600);
5    while (!Serial);
6    Serial.println("Waking up...");
7  }
8
9  void loop() {
10   // put your main code here, to run repeatedly:
11   Serial.println("Looping...");
12   startDeepSleep();
13 }
14
15 void startDeepSleep(){
16   Serial.println("Going to deep sleep...");
17   ESP.deepSleep(5 * 1000000);
18   yield();
   }
```



Der D0 – Pin muss mit dem RST – Pin verbunden sein. Nach 5 Sekunden startet der ESP wieder.
Die setup() – Funktion wird **nicht** durchlaufen. (maximal 71 Minuten)

Beim ESP01 ist der GPIO16 – Pin mit dem RST – Pin zu verbinden.



Deep Sleep mit RST - Taste beenden

```
1
2  void setup() {
3    // put your setup code here, to run once:
4    Serial.begin(9600);
5    while (!Serial);
6    Serial.println("Button pushed...Waking up...");
7  }
8
9  void loop() {
10   // put your main code here, to run repeatedly:
11   startDeepSleep();
12 }
13
14 void startDeepSleep(){
15   Serial.println("Going to deep sleep...");
16   ESP.deepSleep(0);
   }
```

Aufwecken mit Taste zwischen RST-Pin und GND-Pin, oder RESET - Taste am Wemos D1 mini drücken.
Die setup() – Funktion wird durchlaufen.

Literatur Quelle

- https://randomnerdtutorials.com/power-esp32-esp8266-solar-panels-battery-level-monitoring/
- https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
- https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/
- http://esp-idf.readthedocs.io/en/latest/api-reference/system/deep_sleep.html
- https://smarthome-blogger.de/tutorial/esp8266-deep-sleep-tutorial/
- https://makesmart.net/esp8266-deep-sleep-tutorial/