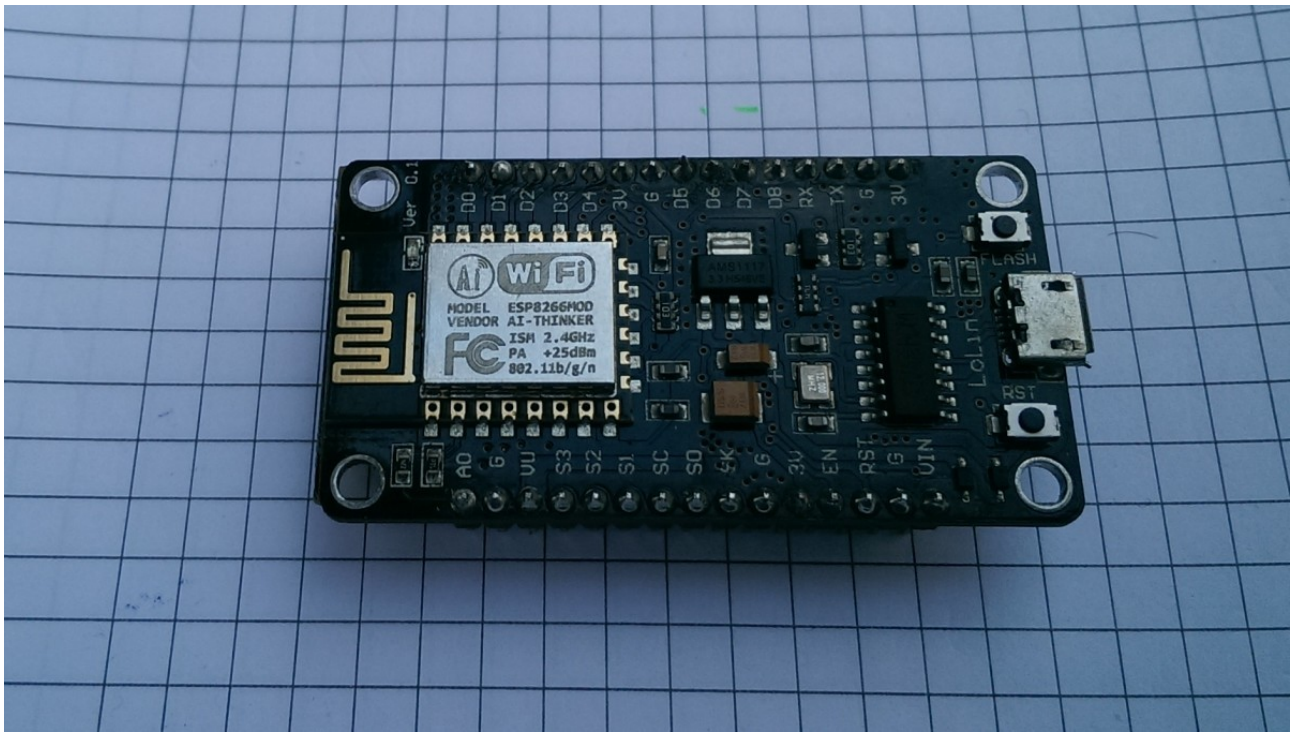
NodeMCU ESP8266 überwacht die Spannung und schaltet das Netzteil
(25.08.2019, Hartmut Buschke)
(22.03.2021, Aktualisierung)

Es gibt keine professionelle Anlage mit solchem Steuerungsbedarf, in der nicht mindestens ein Mikrocontroller verbaut ist. In meiner Solaranlage hat der Prozessor, zumindest vorerst, nur kleine Aufgaben.

Dieser Baustein war aus früheren Experimenten schon vorhanden und kam deshalb zur Anwendung.



Messung der Batteriespannung

Über den analog-digital Wandler Eingang (A0) wird die Batteriespannung gemessen. Der Wert am Pin darf maximal 3,3 Volt betragen. Deshalb ist ein Spannungsteiler vor geschaltet, der sichert, dass dieser Spannungswert auch bei einer Messspannung von 42 Volt nicht überschritten wird. Die Software des Mikrocontrollers wandelt den Spannungswert am A0-Pin in einen Zahlenwert zwischen 0 und 1024 um (ADC Wert).

Mein Spannungsteiler ist so eingestellt, dass sich folgende Werte ergeben:

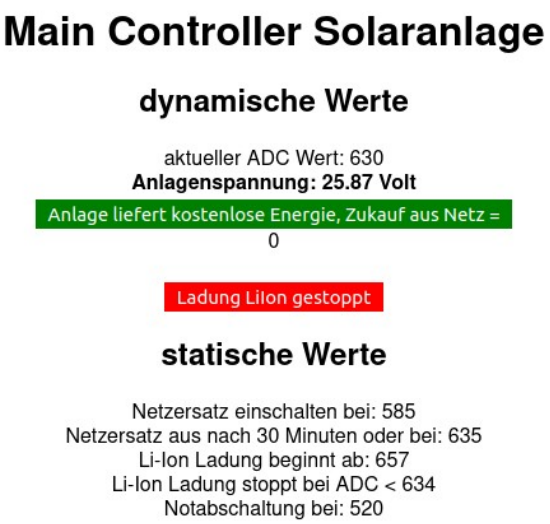
Spannung am Messpunkt [V]	**ADC Wert**
23,5	572
24,0	584
24,5	597
25,0	609
25,5	621
26,0	633
26,5	645
27.0	657
27,5	670
28.0	682

Pro Volt Spannungserhöhung am Messpunkt steigt der ADC Wert um 24,35, wird dann aber ganzzahlig gerundet. Die höchste Spannung am Messpunkt dürfte also 1024/24,35=42,05 Volt betragen. Dieser Wert könnte in meiner Anlage theoretisch nur dann überschritten werden, wenn die Batterien abgeklemmt sind und die Solarzellen ohne Last einspeisen würden. Beides zusammen ist nicht sehr wahrscheinlich.

Ausgabe der Werte

Der ADC Wert wird für die Schaltvorgänge ausgewertet und intern auch in einen Spannungswert umgerechnet, der dann neben anderen Werten über die WLAN Schnittstelle im heimischen Netzwerk ausgegeben wird.

Nach Abruf der IP erscheint im Browser zum Beispiel:



Welche Software die Ausgabe bewirkt, steht am Ende des Beitrages.

Ein- und ausschalten des Ersatzstroms

Die gegenwärtige Hauptfunktion des Mikrocontrollers ist die Zu- und Abschaltung des Hilfsnetzteils, das die Stromversorgung bei leeren Batterien und ohne Solarstrom sicherstellen muss.
Wenn die Spannung der Batterien auf 24,8 Volt gesunken ist, wird das extern montierte Netzteil über ein Relais eingeschaltet. Anschließend gibt es zwei Ausschaltvarianten.

Erstens einen Timer, der nach 30 Minuten abschaltet. Das kann in dem Fall sinnvoll sein, wenn die Ursache für die kritisch niedrige Batteriespannung ein sehr hoher Verbraucherstrom ist. Bleibt der hohe Verbraucherstrom aus, steigt die Batteriespannung wieder leicht und die Batterie kann noch mehrere Stunden einen wesentlich geringeren Strom speisen, ohne dass die Spannung wieder unterschritten wird.

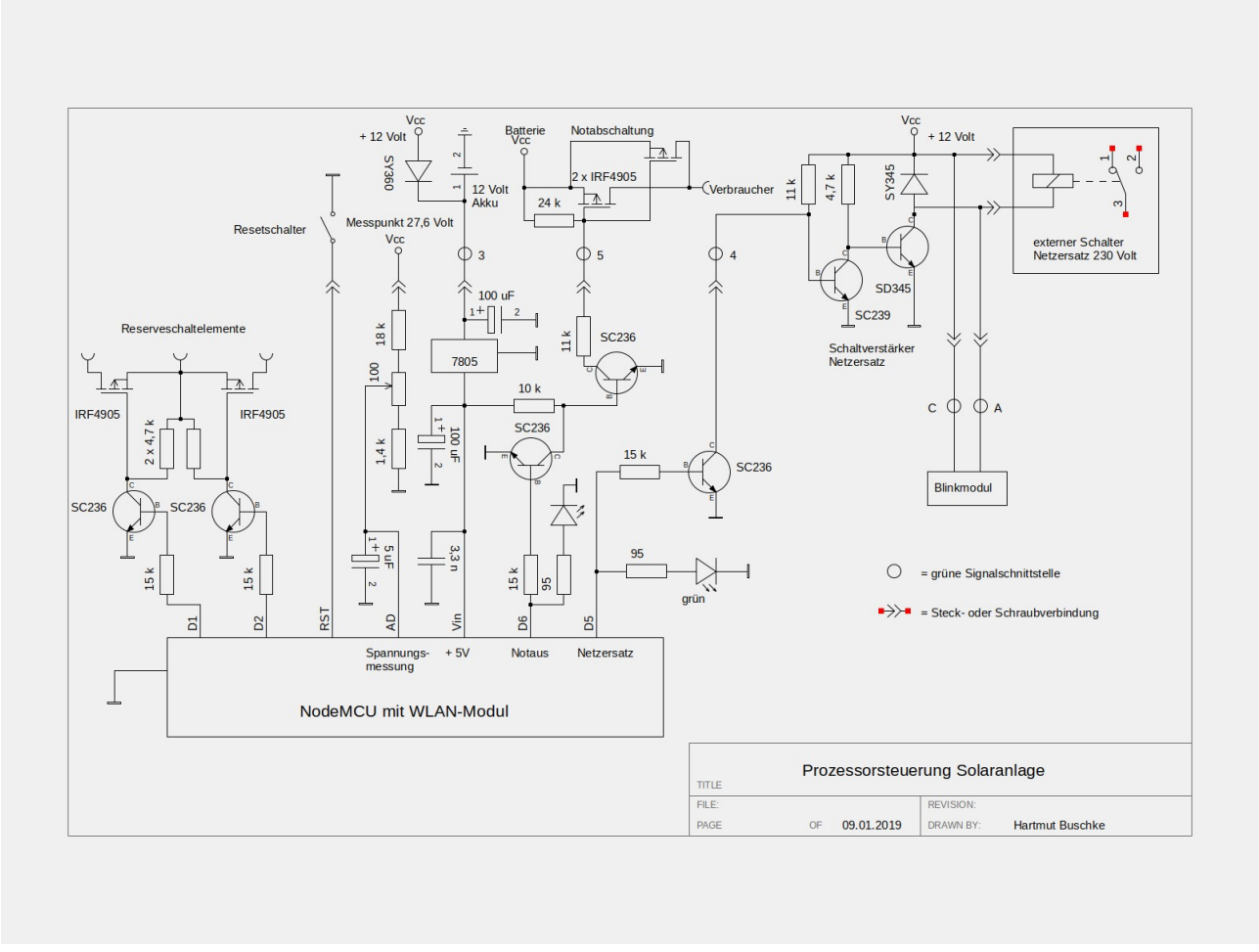
Zweitens schaltet der Mikrocontroller ab, wenn die Batteriespannung über den Wert von 25,4 Volt angestiegen ist, auch wenn die 30 Minuten noch nicht vergangen sind. Dieser Fall tritt ein, wenn ausreichend Sonnenenergie zur Verfügung steht, die Verbraucher damit vollständig versorgt werden können und schon Ladestrom in die Batterien fließt.

Notabschaltung

Eine weitere Funktion des Mikrocontrollers ist die Notabschaltung, falls die Batteriespannung zu weit abgesunken ist. Diesen Wert habe ich auf 23,5 Volt eingestellt. Im normalen Betrieb wird dieser Wert nicht erreicht, weil schon vorher das Hilfsnetzteil die Stromversorgung der Anlage übernimmt und sogar ein geringer Ladestrom in die Batterien fließt.
Nur wenn es einen längeren Stromausfall im öffentlichen Netz gibt oder ein Defekt in der Anlage auftritt, kann dieser Spannungswert erreicht werden. Dann werden alle Verbraucher abgeschaltet, um die Batterien vor einer Tiefentladung zu bewahren. Lediglich der Mikrokontroller wird noch aus einem Hilfsakku versorgt und eine einzige LED zeigt den Notausstatus an. In diesem Zustand kann die Anlage nur noch über den Reset Taster neu gestartet werden, wenn die Funktionsbedingungen wieder vorliegen. Eine Notabschaltung hatte ich schon.

Schaltbild

Das Schaltbild zeigt die periphere Beschaltung des Mikrocontrollerbausteins:



Am Ausgang D1 wird die Ladeleitung für einen Li-Ion Batterieblock gesteuert. Diese Zusatzbatterie behandele ich in einem anderen Kapitel. Den Treiber an D2 benutze ich bisher nicht.

Am Ausgang D5 wird das Relais für das externe Netzteil geschaltet. Die grüne LED zeigt diesen Zustand auf der Platine an. Parallel zum Relais wird auch das Blinkmodul aktiv, das in der Frontplatte verbaut ist.

Wenn der Ausgang D6 aktiv wird, leuchtet die rote LED, die den Notausstatus anzeigt. Gleichzeitig werden die beiden parallel geschalteten Power MOSFET abgeschaltet, über die der gesamte Verbraucherstrom fließt. Laut Datenblatt hätte auch ein FET gereicht, aber sicher ist sicher!

Der IRF4905 ist ein P-Kanal MOSFET, der im durchgeschalteten Zustand bei guter Kühlung über 50 Ampere transportieren kann. Der Drain-Source-Widerstand beträgt dann 0,02 Ohm. Das bekommt mancher Relaiskontakt nicht hin!
In der Vergangenheit habe ich auch immer wieder mit Relais in der Anlage gearbeitet und ganz schlechte Erfahrungen gemacht, wie z.B. verklebte Kontakte und relativ hohe Steuerströme, die sich schnell summieren können.

Programmierung

Den NodeMCU ESP8266 habe ich mit der Arduino IDE programmiert. Das Programm ist im Netz frei verfügbar und kann außer für den Arduino auch für andere Mikrocontroller genutzt werden. Dazu müssen allerdings einige Einstellungen gemacht werden.

Das ist hier alles gut beschrieben:
https://www.heise.de/ct/artikel/Arduino-IDE-installieren-und-fit-machen-fuer-ESP8266-und-ESP32-4130814.html
(Link vom 22.03.2021)

Zugegeben, ohne die Hilfe der Programmierprofis aus der eigenen Familie wäre ich nicht sehr weit gekommen, aber irgendwann hat es dann auch Spaß gemacht.

Mein ESP Programm sieht jetzt so aus:

```
// ESP8266 Main Controller Solaranlage
// Stand vom 10.03.2021

// Belegung der Anschluesse (Bezeichnung GPIO)
#define ledPin 2              // (D4) alter ESP 12
#define notAusPin 12          // (D6)
#define netzErsatzPin 14      // (D5)
#define ladePin 5             // (D1)

#define NETZWERKNAME "xxxxxxx"    // eigene Daten eintragen
#define PASSWORT "xxxxxx"

#include "ESP8266WiFi.h"

WiFiServer Server(80);
WiFiClient Client;

// Festlegung der globalen Variablen
int notAus = 520;
int netzErsatzEin = 585;
int netzErsatzAus = 635;
```

```
int lioLadeStart = 657;
int lioLadeStopp = 634;
int adcWert = 0;
int netzState = LOW;
int x = 0;
int y = 0;

String Request;
String netzEin = "<button>Ersatznetzteil angeschaltet fuer 30 Minuten, vergangene Minuten:
</button>";
String netzAus = "<button class=\"green\">Anlage liefert kostenlose Energie, Zukauf aus Netz = </
button>";
String notabschaltung = "<button>Anlage ist in Notabschaltung wegen Unterspannung</button>";
String leer = "";
String lioEin = "<button class=\"green\">Ladung LiIon Akku ist frei gegeben</button>";
String lioAus = "<button>Ladung LiIon gestoppt</button>";
String notMeldung;
String netzStatus;
String ladeStatus;

void setup() {
  // Ausgangsstatus der Anschluesse festlegen
  pinMode(ledPin, OUTPUT);
  pinMode(notAusPin, OUTPUT);
  pinMode(netzErsatzPin, OUTPUT);
  pinMode(ladePin, OUTPUT);
  digitalWrite(ledPin, LOW);
  digitalWrite(notAusPin, LOW);
  digitalWrite(netzErsatzPin, netzState);
  digitalWrite(ladePin, LOW);

  // Serielle Schnittstelle auf machen und WLAN Verbindung herstellen
  Serial.begin(115200);
  delay(1000);
  Serial.println();
  Serial.print("Verbinde mit: ");
  Serial.println(NETZWERKNAME);

  WiFi.begin(NETZWERKNAME, PASSWORT);

  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Verbindung aufgebaut, eigene IP: ");
  Serial.println(WiFi.localIP());
  Server.begin();
}

void loop() {
  // Blink Modus LED, legt auch Laufzeit der Loop fest
```

```
digitalWrite(ledPin, HIGH);
delay(500);
digitalWrite(ledPin, LOW);
delay(500);

// adc Wert auslesen
adcWert = analogRead(0);
float u = adcWert / 24.35;
Serial.print("akueller ADC: ");
Serial.println(adcWert);
Serial.print("gemessenen Spannung: ");
Serial.print(u);
Serial.println(" Volt");

// Notabschaltung
if (adcWert < notAus){
  digitalWrite(notAusPin, HIGH);
  notMeldung = notabschaltung;
  }
  else{
    digitalWrite(notAusPin, LOW);
    notMeldung = leer;
  }

// Schaltsignal Ersatzstrom und Timer 30 Minuten
if (adcWert < netzErsatzEin){
  netzState = HIGH;
  digitalWrite(netzErsatzPin, netzState);
  }
if (netzState == HIGH){
  x = x + 1;
  y = x / 60;
  Serial.println(x);
    if (x >= 1800){
      netzState = LOW;
      digitalWrite(netzErsatzPin, netzState);
      x = 0;
      y = 0;
    }
}
if(netzState == LOW){
      x = 0;
      y = 0;
}

if (adcWert > netzErsatzAus){
  netzState = LOW;
  digitalWrite(netzErsatzPin, netzState);
}

// Freigabe oder Stopp Ladung Lio
if (adcWert > lioLadeStart){
```

```
    digitalWrite(ladePin, HIGH);
    ladeStatus = lioEin;
   }
   if (adcWert < lioLadeStopp){
    digitalWrite(ladePin, LOW);
    ladeStatus = lioAus;
   }

   //Status Netzersatz für HTML auswerten
   if (netzState == LOW){
     netzStatus = netzAus;
   }
     else{
      netzStatus = netzEin;
     }

   // Webserver starten
   Client = Server.available();

   // Verbindung aufbauen
   if (Client){
    Serial.println("Neuer Client");
    boolean leereZeile = true;
    while (Client.connected()){
     if (Client.available()){
        char c = Client.read();
        if (Request.length() < 100)
          Request += c;

        if (c == '\n' && leereZeile){
          Serial.print("Request von Client: ");
          Serial.println(Request);


   // Seiteninhalt wird alle 10 Sekunden aktualisiert
    Client.println("HTTP/1.1 200 OK");
    Client.println("Content-Type: text/html");
    Client.println("Connection: close");
    Client.println("Refresh: 10");
    Client.println();
    Client.println("<!DOCTYPE HTML><html>");
    Client.println("<head>");
    Client.println("<meta name=\"viewport\"content=\"width=device-width, initial-scale=1\">");
    Client.println("<style>html{font-family: Helvetica, Arial, sans-serif; display: inline-block;
margin: 0 auto; text-align: center;}");
    Client.println("button {background: red; border: none; color: white; padding: 3px 10px; text-
decoration: none; font-size: 15px; margin: 2px;}");
    Client.println(".green {background: green;}</style>");
    Client.println("</head><body>");
    Client.println("<h1>Main Controller Solaranlage</h1>");
    Client.println("<h2>dynamische Werte</h2>");
    Client.print("aktueller ADC Wert: ");
```

```
    Client.println(adcWert);
    Client.println("<br>");
    Client.println("<B>Anlagenspannung: ");
    Client.println(u);
    Client.println(" Volt</B><br>");
    Client.println(netzStatus);
    Client.println("<br>");
    Client.println(y);
    Client.println("<br>");
    Client.println(notMeldung);
    Client.println("<br>");
    Client.println(ladeStatus);
    Client.println();
    Client.println("<h2>statische Werte</h2>");
    Client.print("Netzersatz einschalten bei: ");
    Client.println(netzErsatzEin);
    Client.println("<br>");
    Client.print("Netzersatz aus nach 30 Minuten oder bei: ");
    Client.println(netzErsatzAus);
    Client.println("<br>");
    Client.print("Li-Ion Ladung beginnt ab: ");
    Client.println(lioLadeStart);
    Client.println("<br>");
    Client.print("Li-Ion Ladung stoppt bei ADC < ");
    Client.println(lioLadeStopp);
    Client.println("<br>");
    Client.print("Notabschaltung bei: ");
    Client.println(notAus);

    Request = "";
    break;
    }

  if (c == '\n')
  leereZeile = true;
  else if (c != '\r')
  leereZeile = false;

      }
    }

    Client.stop();
      return;
  }
}
```