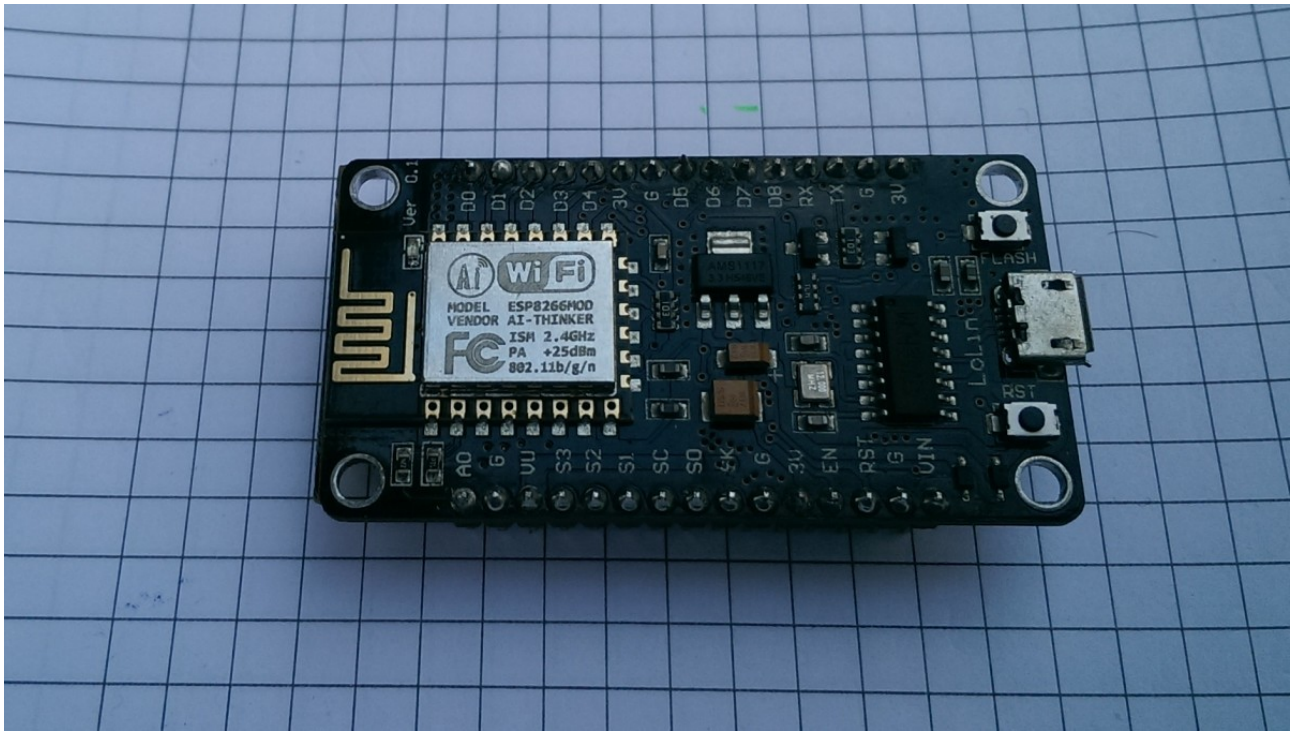
**NodeMCU ESP8266 überwacht die Spannung und schaltet das Netzteil**
(25.08.2019, Hartmut Buschke)

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:

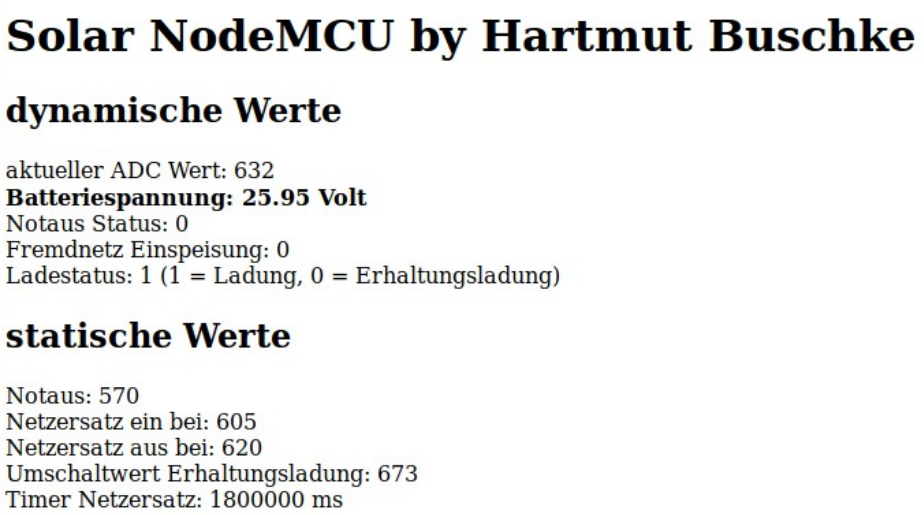
| **Spannung am Messpunkt [V]** | **ADC Wert** |
|---|---|
| 23,5 | 572 |
| 24,0 | 584 |
| 24,5 | 597 |
| 25,0 | 609 |
| 25,5 | 621 |
| 26,0 | 633 |
| 26,5 | 645 |
| 27.0 | 657 |
| 27,5 | 670 |
| 28.0 | 682 |

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.

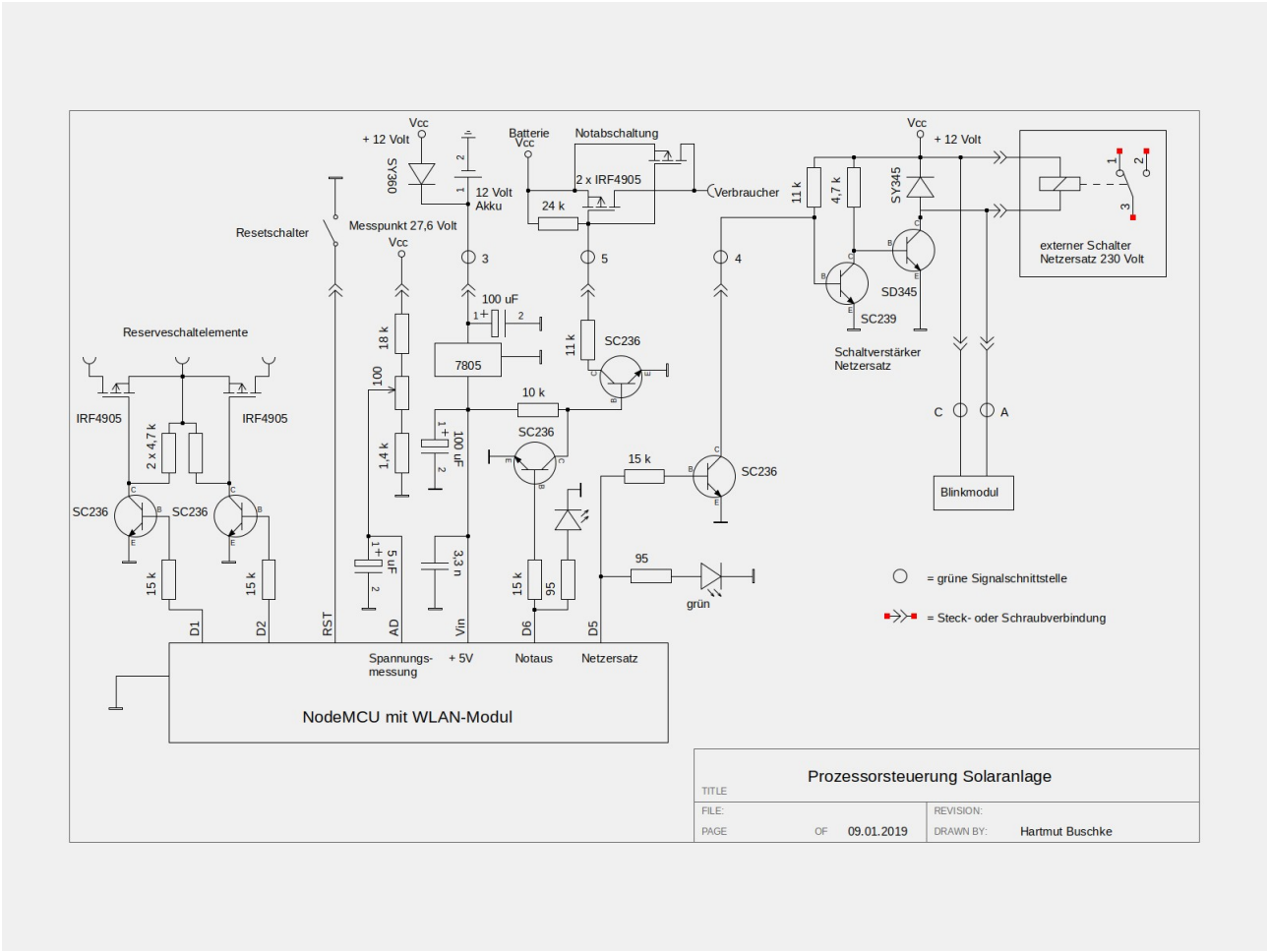
Zweitens schaltet der Mikrocontroller ab, wenn die Batteriespannung über den Wert von 25,4 Volt angestiegen ist, auch wenn die 30 Minuten noch nicht vergangen sind. Dieser Fall tritt ein, wenn ausreichend Sonnenenergie zur Verfügung steht, die Verbraucher damit vollständig versorgt werden können und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




An den Ausgängen D1 und D2 sind noch Treiber angeschlossen, die ich im Moment nicht benutze. 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**

Der NodeMCU ESP8266 hat einen Lua Interpreter und stellt schon verschiedene Module bereit, die für die eigenen Anwendungen genutzt werden können.
Die Programmierung erfolgt über die USB Schnittstelle, was hier gut beschrieben ist:

https://alexbloggt.com/nodemcu-einfuehrung/
(Link vom 26.08.2019)

Sehr wertvoll war auch diese Dokumentation, in der die Module beschrieben sind:

https://nodemcu.readthedocs.io/en/master/
(Link vom 26.08.2019)

Zugegeben, ohne die Hilfe der Programmierprofis aus der eigenen Familie wäre ich nicht sehr weit gekommen, aber irgendwann hat es dann auch Spaß gemacht. Im Programm ist noch die Funktion enthalten, die für die Umschaltung auf Erhaltungsladung vorgesehen war. Diese nutze ich im Moment nicht.

**Mein Programm sieht jetzt so aus:**

```
-- Blink using timer alarm --
timerId = 0
externStromTimer = 1
dly = 1000  -- 1 Sekunde (Timer 0)
externStromDelay = 1000 * 60 * 30 --30 Minuten (Timer 1)

-- n sekunden warten bevor die hauptschleife startet
initDelay = 2000 * 1000

-- unter welchem ADC Wert wird der Notaus aktiviert
notAusADC = 570

-- unter welchem ADC Wert wird der Netzersatzmodus für N Minuten aktiviert?
netzErsatzADC = 605
netzErsatzAus = 620

-- Umschalten auf Erhaltungsladung
erhaltLadung = 673

-- PINS
ledPin = 4
```

```
notAusPin = 6
netzErsatzPin = 5
ladePin = 2

-- Messwerte / Status
adcValue = 0
ledState = 1
notAusStatus = 0
netzErsatzStatus = 0
ladeStatus = 1
anZeigeWert = 0
Stelle = 2

-- Setup ---
-- set mode to output
gpio.mode(ledPin,gpio.OUTPUT)
gpio.mode(notAusPin,gpio.OUTPUT)
gpio.mode(netzErsatzPin,gpio.OUTPUT)
gpio.mode(ladePin,gpio.OUTPUT)

-- init state
gpio.write(ledPin, ledState)
gpio.write(notAusPin, notAusStatus)
gpio.write(netzErsatzPin, netzErsatzStatus)
gpio.write(ladePin, ladeStatus)

----- Dienste ----
-- WIFI einrichten
wifi.setmode(wifi.STATION)
station_cfg={}
station_cfg.ssid="[eigene SSID eintragen]"
station_cfg.pwd="[eigenes Passwort eintragen]"
station_cfg.save=false
wifi.sta.config(station_cfg)
wifi.sta.connect()

tmr.alarm(0, 1000, tmr.ALARM_AUTO, function()
   if wifi.sta.getip() == nil then
      print("IP unavailable, Waiting...")
   else
      tmr.stop(0)
      print("ESP8266 mode is: " .. wifi.getmode())
      print("The module MAC address is: " .. wifi.ap.getmac())
      print("Config done, IP is "..wifi.sta.getip())
   end
end)

-- Webserver starten
srv=net.createServer(net.TCP,3)
srv:listen(80,function(conn)
   conn:on("receive",function(conn,payload)
   html = "<h1>Solar NodeMCU by Hartmut Buschke</h1>"
```

```
    html = html .. "<h2>dynamische Werte</h2>"
    html = html .. "aktueller ADC Wert: " .. adcValue .. "<br/>"
    html = html .. "<B>Batteriespannung: " .. messWert .. " Volt</B><br/>"
    html = html .. "Notaus Status: " .. notAusStatus .. "<br/>"
    html = html .. "Fremdnetz Einspeisung: " .. netzErsatzStatus .. "<br/>"
    html = html .. "Ladestatus: " .. ladeStatus .. " (1 = Ladung, 0 = Erhaltungsladung)<br/>"
    html = html .. "<h2>statische Werte</h2>"
    html = html .. "Notaus: " .. notAusADC .. "<br/>"
    html = html .. "Netzersatz ein bei: " .. netzErsatzADC .. "<br/>"
    html = html .. "Netzersatz aus bei: " .. netzErsatzAus .. "<br/>"
    html = html .. "Umschaltwert Erhaltungsladung: " .. erhaltLadung .. "<br/>"
    html = html .. "Timer Netzersatz: " .. externStromDelay .. " ms<br/>"
    conn:send(html)
    end)
    conn:on("sent", function (c) c:close() end) --close the socket after succesfully flushed data
end)

-- boot wait time
print("wait")
tmr.delay(initDelay)

print("start loop")
-- Main loop
tmr.alarm( timerId, dly, tmr.ALARM_AUTO, function()
  -- invertiert den ledState
  ledState = 1 - ledState;

  -- write state
  gpio.write(ledPin, ledState)

  -- adc auslesen
  adcValue = adc.read(0)

  if (notAusStatus == 1) then
    -- nichts mehr machen!
    return
  end

  -- Notaus aktivieren?
  if (adcValue < notAusADC) then
    notAusStatus = 1
    gpio.write(notAusPin, notAusStatus)
    ladeStatus = 1
    return
  end

  -- Anzeigewert berechnen und runden
  anZeigeWert = adcValue / 24.35
  function round(wert)
    local mult = 10^(Stelle or 0)
    return math.floor(anZeigeWert * mult + 0.5) / mult
    end
```

```
  messWert = (round())

 -- Externe Stromversorgung prüfen
 if (netzErsatzStatus == 0 and adcValue < netzErsatzADC) then
  netzErsatzStatus = 1
  gpio.write(netzErsatzPin, netzErsatzStatus)

  -- den Zusatand für N Minuten halten, zweiten Timer starten!
  tmr.alarm(externStromTimer, externStromDelay, tmr.ALARM_SINGLE, function()
     netzErsatzStatus = 0
     gpio.write(netzErsatzPin, netzErsatzStatus)
  end)
  return
 end

  -- Abschaltbedingung Netzersatz
  if (adcValue > netzErsatzAus) then
  netzErsatzStatus = 0
  gpio.write(netzErsatzPin, netzErsatzStatus)
  end

  -- auf Erhaltungsladung Umschalten
  if (adcValue > erhaltLadung) then
     ladeStatus = 0
     else ladeStatus = 1
  end
  gpio.write(ladePin, ladeStatus)

print("")
print("")
print("")
print("ADC Wert: ") ; print(adcValue)
print("Batteriespannung: ") ; print(messWert)
print("Ladestatus: ") ; print(ladeStatus)
print("Notaus Status: ") ; print(notAusStatus)
print("Fremdnetz Einspeisung: ") ; print(netzErsatzStatus)
print("")
print("")
print("")
print("")
end)
-- Hauptschleife Ende
```

Anmerkung: Die print-Befehle im letzten Block können auch entfallen. Sie waren aber bei der Programmierung sehr hilfreich, weil die Werte der Variablen auf dem Bildschirm ausgegeben wurden.