# Standheizung Josty Anleitung

Inhaltsverzeichnis

[Standheizung Josty Anleitung 1](#_Toc23516409)

[Hardware 2](#_Toc23516410)

[Verbaute Module: 2](#_Toc23516411)

[Arduino Pinbelegung: 2](#_Toc23516412)

[Befehle 2](#_Toc23516413)

[GSM SMS Befehle: 2](#_Toc23516414)

[ESP8622 Befehle 3](#_Toc23516415)

[Einbindung in Fhem 3](#_Toc23516416)

[Programmierung: 3](#_Toc23516417)

[Befehle: 3](#_Toc23516418)

[Programmierung per SMS 3](#_Toc23516419)

[Bedienung: 4](#_Toc23516420)

[Steuerung per SMS 4](#_Toc23516421)

[Steuerung per WLan ( ESP8266 ) 4](#_Toc23516422)

[Einstellungen im ESP8622 ( ESPeasy ) 4](#_Toc23516423)

[Arduino Code 5](#_Toc23516424)

[Arduino QuellCode 5](#_Toc23516425)

[ESP8266 Quellcode 5](#_Toc23516426)

# Hardware:

## Verbaute Module:

SIM900 <https://www.makershop.de/module/gsm-gps/sim900a-quad-band-gsm/>

Arduino Mega 2560

SD Card Reader <https://www.makershop.de/module/schnittstellen/adafruit-microsd-card-breakout/>

// Ublox NEO 6M-GPS Modul GY-GPS6MV2 UART <https://www.makershop.de/module/gsm-gps/ublox-neo-6m-gps/>

Original Sim900 Modul

ESP8266 <https://www.makershop.de/plattformen/nodemcu/esp32-board-stiftleisten/>

LM2596S DC-DC Spannungsregler Step-Down Converter <https://www.ebay.de/itm/LM2596S-DC-DC-Spannungsregler-Step-Down-Converter-Regler-einstellbar-1-2V-30V-3A/392315076833?ssPageName=STRK%3AMEBIDX%3AIT&_trksid=p2057872.m2749.l2649>

## Arduino Pinbelegung:

Status Blinkled Pin 🡪 8

Taster Standheizung Start 🡪 Pin 7

Taster Standheizung Stop 🡪 Pin 9

Sim900 Pins TX RX 🡪 UART Serial1

// GPS Pins 🡪 16TX 17RX

// SD Karte Pin 🡪 CS-5 SCK-13 MOSI-11 MISO-12

**CS**–  Pin für die Erfassung der SD-Karte, anzuschließen an Pin D5 des Arduinos  
**SCK** –  Pin für die Synchronisation des Datentransfers durch die Master Pins, anzuschließen an Pin D13 des Arduinos  
**MOSI** –  Pin für den Datentransfer (**M**aster **O**ut **S**lave **I**n) , anzuschließen an Pin D11 des Arduinos  
**MISO** –  Pin für den Datentransfer (**M**aster **I**n **S**lave **O**ut) , anzuschließen an Pin D12 des Arduinos

# 

# Befehle

## GSM SMS Befehle:

Groß/Kleinschreibung spielt keine Rolle da es im Arduino Code alles auf Kleinschreibung gewandelt wird.

on -> Schaltet Standheizung für 30 min ein.

off 🡪 Standheizung ( vorzeitig ) ausschalten.

start0730 🡪 Programmiert die Standheizung für den nächsten einmaligen Start

Timer15 🡪 Lässt Standheizung für xx Minuten laufen

Init 🡪 Initialisierung GSM900 Modul

smson 🡪 Schaltet SMS als Rückmeldung ein

smsoff 🡪 Schaltet SMS als rückmeldung aus

## ESP8622 Befehle

Mittels http Befehl http://IPAdressevomESP/control?cmd=pulse,14,1,2000 kann die Standheizung Aktiviert werden. Die ESP8622 sendet einen 2000 ms impuls von D5 an den Arduino Eingangspin 6 und startet damit die Standheizung für die Standartlaufzeit ( 30 Min)

Bsp: http://192.168.4.151/control?cmd=pulse,14,1,2000

Mittels Haussteuerung wie z.b. Fhem kann der ESP ausgelesen werden.

# Programmierung:

## Programmierung Befehlsübersicht:

001#0172999999

clear#001

timezone#2

## Programmierung per SMS

Spannung auf das Modul geben danach hat man 2 min Zeit um eine SMS mit der eigenen Rufnummer in folgendem Format zu Senden. Diese wird dann auf Speicherplatz 001 bis 005 gespeichert.

Bsp. Speicherplatz#Nummer 🡪 001#0172999999

Damit wird die Nummer auf Speicherplatz 1 im Eeprom vom Arduino gespeichert.

Mittlels sms clear#001 wird die Rufnummer wieder gelöscht.

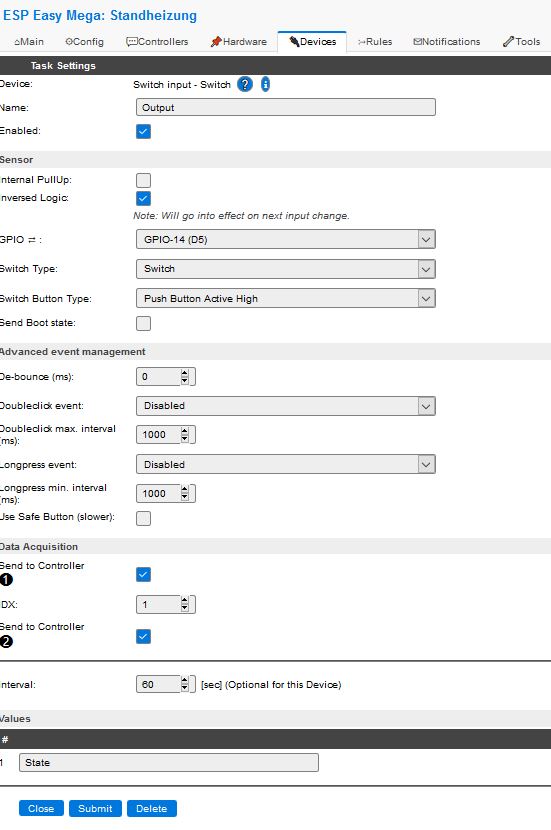
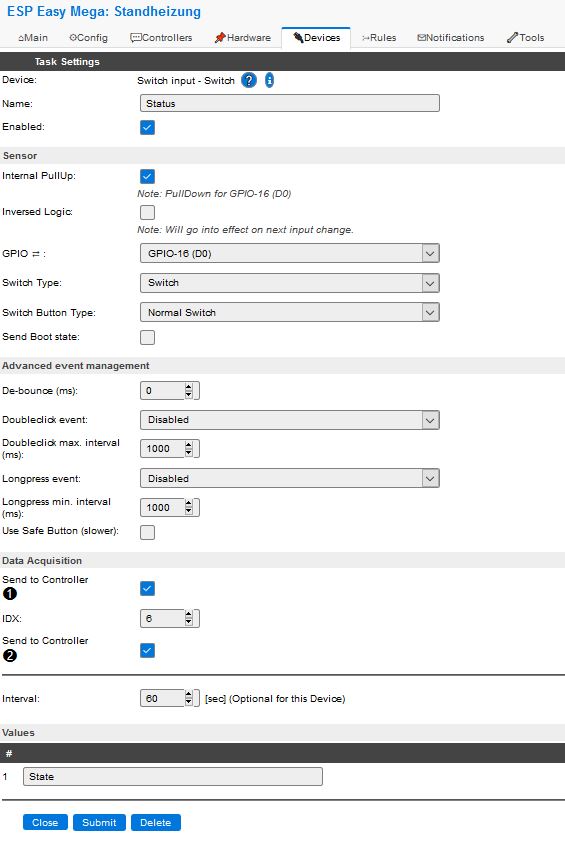
Die Uhrzeit bekommt das Modul vom GPS Empfänger diese Uhrzeit kommt in UTC

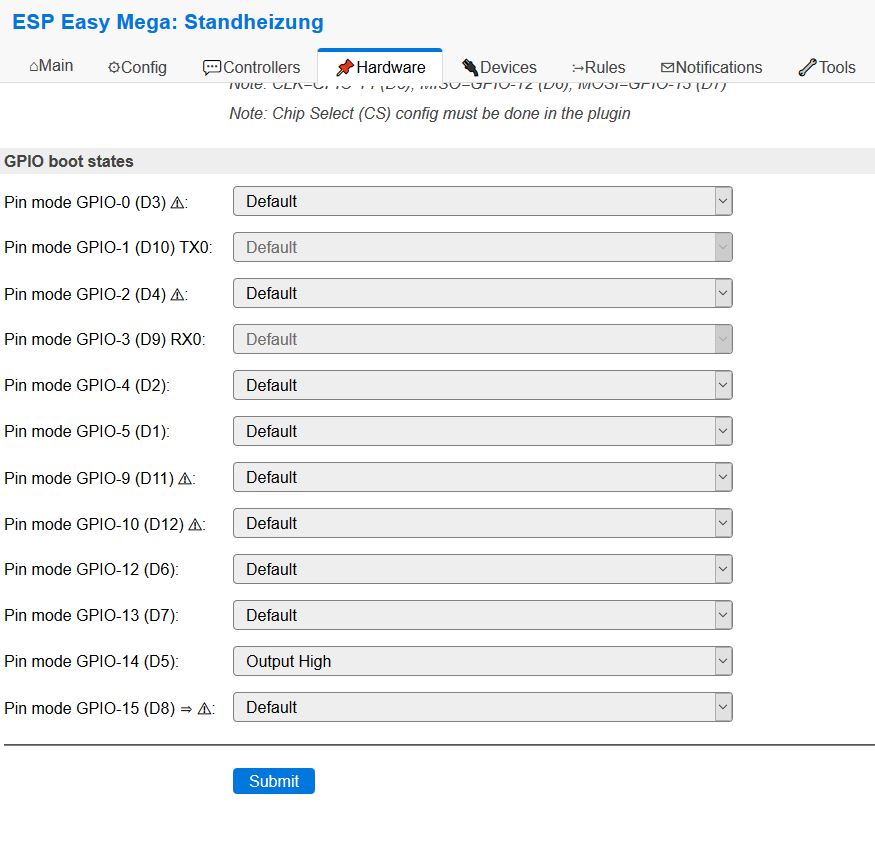
Mittels Befehl SMS Befehl timezone#-2 wird z.b. 2 stunden dazugezählt.

# Bedienung:

## Steuerung per WLan ( ESP8266 )

# Einstellungen im ESP8622 ( ESPeasy )





# Arduino Code

## Arduino QuellCode

/\*

Name: StandheizungGSM.ino

Created: 06.11.2019 06:33:29

Author: jost Mario

https://www.geeetech.com/wiki/index.php/Arduino\_GPRS\_Shield

\*/

#include <TinyGPS++.h>

#include <Time.h>

TinyGPSPlus gps;

#define Sim900\_Serial Serial1

#define GpsNeo6\_Serial Serial2

bool SmsAntwort = false; // Soll eine Bestätigungsmail gesendet werden (verursacht Kosten)

static const uint32\_t SerialBaud = 115200; // Baudrate Seraiel für PC Serial Baud

static const uint32\_t SerialBaudGSM = 19200; // Baudrate Sim900 Modul

static const uint32\_t SerialBaudGPS = 9600;

bool simm900\_isINIT = false;

bool Neo6mGPS\_isInit = false;

bool started = false;

unsigned long startTime;

unsigned long shutdownTime;

bool RelayOn;

String serialInputBufferGSM;

const unsigned long DEFAULTTIME = 1800000; // Standart Laufzeit der Standheizung 30 Min 1800000ms

const bool EIN = true;

const bool AUS = false;

int val = 0;

int valoff = 0;

int HeizLedState = LOW;

unsigned long previousMillis = 0; // LED Blink will store last time LED was updated

const long interval = 200; // LED Blink interval at which to blink Heizungs Led (milliseconds)

unsigned long currentMilliss = millis(); // LED Blink

String startzeitH = "HH";

String startzeitM = "MM";

// Pinbelegungen

const int Relay = 10; // Pin für Rellays Standheizung an

const int switchs = 7; // Pin für Stanheizung Starttaster

const int switchsoff = 9;

const int esppin = 6; // Heizung Start

const int EspPinRueck = 5; // Rückmeldung an ESPe

const int HeizungLed = 8; // Led Standheizung Aktiv

const int sim900PowerPin = 6; // Pin zum Einschalten des Sim900 Moduls

void InitSim900();

void sim900\_PowerOn();

// the setup function runs once when you press reset or power the boardd

void setup()

{

pinMode(switchs, INPUT\_PULLUP); // Taster als Input

digitalWrite(switchs, HIGH); // Intern Pullup setzen

pinMode(switchsoff, INPUT\_PULLUP); // Taster als Input

digitalWrite(switchsoff, HIGH); // Intern Pullup setzen

pinMode(HeizungLed, OUTPUT); // Status LED Standheizung

pinMode(Relay, OUTPUT); // Relays als Output

digitalWrite(Relay, LOW); // Relays aus setzen

//Init der Debug serial.

Serial.begin(SerialBaud);

Serial.println("+++ SMS Antwort deaktiviert Aktivieren mit SMS komando --> smson oder smsoff +++");

//Init der GPS serial.

GpsNeo6\_Serial.begin(SerialBaudGPS);

//Init der SIM900 serial.

Sim900\_Serial.begin(SerialBaudGSM);

//AT an SIm900 Modul um Autobaudrate einzustellen

Serial.println("+++ Send AT for Autobaud +++");

while (Sim900\_Serial.read() >= 0); // Empfangsbuffer leeren

Sim900\_Serial.println("AT");

delay(1000);

if (Sim900\_Serial.available())

{

// Lese alles bis zum Zeichen \n

serialInputBufferGSM = "";

serialInputBufferGSM = Sim900\_Serial.readStringUntil('\n');

// Serial.println("Eingangspuffer: " + serialInputBufferGSM.substring(0, 2));

if (serialInputBufferGSM.substring(0, 2) == "AT")

{

Serial.println(" +++ GSM 900 bereits an +++");

Serial.println(" +++ Wait for GSM init....... +++");

}

else

{

Serial.println("Fehler: " + serialInputBufferGSM);

}

}

else

{

Serial.println(" +++ SIM 900 is off +++");

sim900\_PowerOn();

Serial.println(" +++ Wait for GSM init....... +++");

delay(5000);

}

while (Sim900\_Serial.read() >= 0); // Empfangsbuffer leeren

}

// the loop function runs over and over again until power down or reset

void loop()

{

//Serial.println(millis());

// 10000 ms warten bis Modul eingebucht hat dann Initialisierenn.

if ((millis() > 2000) && (simm900\_isINIT == false))

{

simm900\_isINIT == true;

Serial.println(" +++ begin sim900 initialisierung +++");

InitSim900();

// InitGPSModul();

}

// ++++++++++++++ Per Taster standheizung einschalten Start++++++++++++++++++++++++++

val = digitalRead(switchs); // read the input pin

// Serial.println(val);

if (val == 0)

{

if (!RelayOn)

{

Serial.println(" +++ Aktiviert durch Taster +++");

HeizLedState = HIGH;

HeizungEin(EIN, DEFAULTTIME);

DeleteAllSMS(); // SMS Speicher leeren

}

}

// ++++++++++++++ Per Taster standheizung einschalten Ende++++++++++++++++++++++++++

valoff = digitalRead(switchsoff); // read the input pin

// Serial.println(valoff);

if (valoff == 0)

{

if (RelayOn)

{

Serial.println(" +++ Standheizung aus durch Taster +++");

HeizLedState = LOW;

HeizungEin(AUS, DEFAULTTIME);

}

}

if (Sim900\_Serial.available())

{

serialInputBufferGSM = Sim900\_Serial.readStringUntil('\n');

delay(10);

serialInputBufferGSM.toLowerCase();

Serial.println(serialInputBufferGSM);

// ################################### Anruf ############################

// Ein Anruf kommt rein mit der nummer 491622742063

// if (serialInputBufferGSM.substring(8, 12) == "CLIP:\"+491622742063\"")

//if (serialInputBufferGSM.substring(8, 14) == "491622")

if (serialInputBufferGSM.startsWith("+clip"))

{

Serial.println("+++ Registrierte Rufnummern: +491622742063 +4974126950510 +4915903778464 +++");

if (serialInputBufferGSM.substring(8, 21) == "+491622742063" || "+4974126950510" || "+4915903778464" )

{

Serial.println(serialInputBufferGSM.substring(8, 14));

Serial.println("Heizung Einschalten via Telefon");

HeizungEin(EIN, DEFAULTTIME);

}

}

//######### SMS Empfang #########

// on off timer15 start0730

//HeizungSteuernUhrzeit(bool Zustand, string time)

// alle sms befehle in kleinschreibung umwandeln

if (serialInputBufferGSM.substring(0, 3) == "off")

{

HeizungEin(AUS, 0);

DeleteAllSMS();

Serial.println("Off durch SMS");

}

else if (serialInputBufferGSM.substring(0, 2) == "on")

{

HeizungEin(EIN, DEFAULTTIME);

DeleteAllSMS();

Serial.println("On durch SMS");

}

else if (serialInputBufferGSM.substring(0, 5) == "timer")

{

HeizungEin(EIN, (serialInputBufferGSM.substring(5, 7).toInt()) \* 60000);

DeleteAllSMS();

Serial.println("Timer durch SMS");

}

else if (serialInputBufferGSM.substring(0, 4) == "init")

{

InitSim900();

}

else if (serialInputBufferGSM.substring(0, 5) == "start")

{

startzeitH = serialInputBufferGSM.substring(5, 7);

startzeitM = serialInputBufferGSM.substring(7, 10);

Serial.println("startzeit programmiert auf " + startzeitH + " " + startzeitM);

}

else if (serialInputBufferGSM.substring(0, 5) == "smson")

{

SmsAntwort = true; // Soll eine Bestätigungsmail gesendet werden (verursacht Kosten)

Serial.println("+++ AntwortSMS Aktiviert +++");

}

else if (serialInputBufferGSM.substring(0, 5) == "smsoff")

{

SmsAntwort = false; // Soll eine Bestätigungsmail gesendet werden (verursacht Kosten)

Serial.println("+++ AntwortSMS Deaktiviert +++");

}

}

if (Serial.available())

{

Sim900\_Serial.write(Serial.read());

}

// Relays nach 30 min Abschalten

if (RelayOn)

{

if (startTime + shutdownTime < millis())

{

HeizungEin(AUS, 0);

digitalWrite(EspPinRueck, LOW); // Heizung an an ESP8622 melden

DeleteAllSMS(); // SMS Speicher leeren

}

}

// #################### Blinkende LED für Standheizung Aktiv ###############################

// check to see if it's time to blink the LED; that is, if the difference

// between the current time and last time you blinked the LED is bigger than

// the interval at which you want to blink the LED.

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval)

{

previousMillis = currentMillis; // save the last time you blinked the LED

if (RelayOn)

{

if (HeizLedState == LOW)

{

HeizLedState = HIGH;

// Serial.println("test Heizung An");

}

else

{

HeizLedState = LOW;

// Serial.println("Led Heizung Aus");

}

digitalWrite(HeizungLed, HeizLedState); // set the LED with the ledState of the variable:

}

else digitalWrite(HeizungLed, LOW);

}

// if (GpsNeo6\_Serial.available()) // GPS Sensor sendet Daten im Secundentakt raus

// Serial.println(GpsNeo6\_Serial.read());

//delay(300);

/\*while (GpsNeo6\_Serial.available() > 0)

{

gps.encode(GpsNeo6\_Serial.read());

}\*/

//smartDelay(200);

////printDateTime(gps.date, gps.time);

////Serial.println(gps.date.value()); // Raw date in DDMMYY format (u32)

//Serial.println(gps.date.year()); // Year (2000+) (u16)

//Serial.println(gps.date.month()); // Month (1-12) (u8)

//Serial.println(gps.date.day()); // Day (1-31) (u8)

//Serial.println(gps.time.value()); // Raw time in HHMMSSCC format (u32)

// Serial.println(gps.time.hour()); // Hour (0-23) (u8)

// Serial.println(gps.time.minute()); // Minute (0-59) (u8)

////Serial.println(gps.time.second()); // Second (0-59) (u8)

//Serial.println("#############################");

}

// This custom version of delay() ensures that the gps object

// is being "fed".

//static void smartDelay(unsigned long ms)

//{

// unsigned long start = millis();

// do

// {

// while (GpsNeo6\_Serial.available())

// gps.encode(GpsNeo6\_Serial.read());

// } while (millis() - start < ms);

//}

void InitSim900()

{

simm900\_isINIT = false;

// Serial.println("+++ begin sim900 initialisierung +++");

delay(20);

Serial.println("+++ Sende AT +++");

Sim900\_Serial.println("AT");

delay(800);

SerialAbfrage();

Serial.println("+++ Signal Qualitaet +++ abfragen");

Sim900\_Serial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is the best

delay(800);

SerialAbfrage();

Serial.println("+++ Sim Karte abfragen +++");

Sim900\_Serial.println("AT+CCID"); //Read SIM information to confirm whether the SIM is plugged

delay(800);

SerialAbfrage();

Serial.println("+++ Check Network +++");

Sim900\_Serial.println("AT+CREG?"); //Check whether it has registered in the network

delay(800);

SerialAbfrage();

Serial.println("+++ set CLIP +++");

Sim900\_Serial.println("AT+CLIP=1"); //rufnummer anzeige aktivieren CLIP

delay(800);

SerialAbfrage();

Sim900\_Serial.println("AT+CNMI=2,2,0,0,0");

delay(800);

SerialAbfrage();

Serial.println("+++ set SMS mode to text +++");

Sim900\_Serial.println("AT+CMGF=1"); // set SMS mode to text

delay(800);

SerialAbfrage();

Serial.println("+++ sms speicher leeren +++");

Sim900\_Serial.println("AT+CMGD=1,4"); // delete all SMS

delay(800);

SerialAbfrage();

Serial.println("+++ sim900 fertig initialisiert +++");

simm900\_isINIT = true;

}

void sim900\_PowerOn()

{

Serial.println("+++ Sim900 wird eingeschaltet! ... +++");

// Einschaltprozedur für Sim900 Modul

pinMode(sim900PowerPin, OUTPUT);

digitalWrite(sim900PowerPin, LOW);

delay(1000);

digitalWrite(sim900PowerPin, HIGH);

delay(2000);

digitalWrite(sim900PowerPin, LOW);

delay(3000); // Warten bis Modul hochgefahren

}

String SerialAbfrage()

{

while (Sim900\_Serial.available()) // Sendet Daten bei z.b. Anruf oder SMS

Serial.write(Sim900\_Serial.read());

return Sim900\_Serial.readString();

}

void DeleteAllSMS()

{

Sim900\_Serial.println("AT+CMGD=1,4"); // delete all SMS

Serial.println("Alle SMS geloescht void");

delay(700);

Sim900\_Serial.println("AT+CMGF=1");

delay(700);

while (Sim900\_Serial.available())

{

Serial.write(Sim900\_Serial.read());//Forward what Software Serial received to Serial Port

}

}

//

//static void printFloat(float val, bool valid, int len, int prec)

//{

// if (!valid)

// {

// while (len-- > 1)

// Serial.print('\*');

// Serial.print(' ');

// }

// else

// {

// Serial.print(val, prec);

// int vi = abs((int)val);

// int flen = prec + (val < 0.0 ? 2 : 1); // . and -

// flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;

// for (int i = flen; i < len; ++i)

// Serial.print(' ');

// }

// smartDelay(0);

//}

//

//static void printInt(unsigned long val, bool valid, int len)

//{

// char sz[32] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

// if (valid)

// sprintf(sz, "%ld", val);

// sz[len] = 0;

// for (int i = strlen(sz); i < len; ++i)

// sz[i] = ' ';

// if (len > 0)

// sz[len - 1] = ' ';

// Serial.print(sz);

// smartDelay(0);

//}

//

//static void printDateTime(TinyGPSDate& d, TinyGPSTime& t)

//{

// if (!d.isValid())

// {

// Serial.print(F("\*\*\*\*\*\*\*\*\*\* "));

// }

// else

// {

// char sz[32];

// sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());

// Serial.print(sz);

// }

//

// if (!t.isValid())

// {

// Serial.print(F("\*\*\*\*\*\*\*\* "));

// }

// else

// {

// char sz[32];

// sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());

// Serial.print(sz);

// }

//

// printInt(d.age(), d.isValid(), 5);

// smartDelay(0);

//}

//

//static void printStr(const char\* str, int len)

//{

// int slen = strlen(str);

// for (int i = 0; i < len; ++i)

// Serial.print(i < slen ? str[i] : ' ');

// smartDelay(0);

//}

void HeizungEin(bool state, unsigned long shutdownValue)

{

if (state)

{

// Serial.println("Abschaltzeit: " + shutdownValue);

Serial.println("Relay ON");

digitalWrite(Relay, HIGH);

digitalWrite(EspPinRueck, HIGH); // Heizung an an ESP8622 melden

RelayOn = true;

startTime = millis();

shutdownTime = shutdownValue;

if (SmsAntwort)

{

smssend();

Serial.println("+++ Sende SMS +++");

}

// Serial.println(startTime);

// Serial.println(shutdownTime);

// Serial.println(startTime + shutdownTime);

}

else

{

Serial.println("Relay OFF");

digitalWrite(Relay, LOW);

digitalWrite(EspPinRueck, LOW); // Heizung an an ESP8622 melden

RelayOn = false;

}

}

void smssend() {

Sim900\_Serial.println("AT+CMGF=1\r");

delay(100);

Sim900\_Serial.println("AT + CMGS = \"+491622742063\"");

delay(100);

Sim900\_Serial.println("Standheizung ON");

delay(100);

Sim900\_Serial.println((char)26);

delay(100);

Sim900\_Serial.println();

delay(5000);

}

## ESP8266 Quellcode

Es wird die Datei ESP\_Easy\_mega-20191028\_normal\_core\_241\_ESP8266\_4M1M.bin

Mit dem ESP Flashtool auf den ESP8266 geladen, dieser spannt beim erstmaligen Start einen Accesspoint auf und damit kann man diesen Konfigurieren.