Adventskranzkerzen-Steuerung 2.0

Quellen

- Adafruit RTC Bibliothek für DS1307
- GPIOs via Webinterface schalten
- ESP8266 12-F NodeMCU Pinout
- Arduino Referenz
- Schaltplan des Relais-Moduls
- Arduino ESP8266 filesystem uploader
- ESP8266 Webserver echte HTML Dateien ausliefern - Tutorial
- ESPAsyncWebServer –> Download *.zip
- ESPAsyncTCP –> Download *.zip
- JQuery Mobile –> Download

Material

Anzahl	Bezeichnung	Quelle	Artikelnummer	Preis/Stk. [€]	Link
1	ESP8266 NodeMCU	Reichelt	DEBO JT ESP8266	14,10	DEBO JT ESP8266
1	Echtzeituhr (RTC)	Reichelt	ARD RTC DS1307	1,50	D1Z RTC
1	Relaiskarte, 4-kanalig	Reichelt	DEBO RELAIS 4CH	4,60	DEBO RELAIS 4CH
1	DC/DC Spannungswandler	Reichelt	DEBO DCDC 2.5W	2,75	DEBO DCDC 2.5W
2	Buchsenleiste, gerade 32polig, trennbar	Reichelt	MPE 115-1-032	1,99	MPE 115-1-032
5	Status-LEDs, grün, 5mm	Reichelt	EVL 333-2SYGT/S5	0,09	EVL 333-2SYGT/S5
5	LED Fassung, 5mm	Reichelt	KBT RTF-5010	0,31	KBT RTF-5010
1	Lochraster LP	Reichelt	H25PR160	2,70	H25PR160
4	Buchse für Kerzen-LEDs	ebay	GX16-2	1,70	
4	Stecker für Kerzen-LEDs	ebay	GX16-2	1,70	
1	Buchse für Spannungsversorgung	ebay	GX16-3	1,80	
1	Stecker für Spannungsversorgung	ebay	GX16-3	1,80	
1	Sicherungshalter, 5x20mm, max. 10A-250V	Reichelt	PL 125000	0,70	PL 125000
1	Universal Gehäuse, 188 x 110 x 70 mm, IP40	Reichelt	BOX4U KS 450	11,80	BOX4U KS 450
4	Distanzhülsen, Metall, 6-Kant, M3, 8mm	Reichelt	VT DA 8MM	0,23	VT DA 8MM
4	Sechskantmutter, M3	Reichelt	SK M3	0,02	SK M3
12	Unterlegscheibe, 3,2mm	Reichelt	SKU 3,2-100	0,02	SKU 3,2-100
4	Blechschraube, 3,5×9,5 mm	Reichelt	SBL 35095-100	0,03	SBL 35095-100
.	.	.	.	.	.

Funktionsbeschreibung

Die Adventskranzkerzensteuerung dient der Ansteuerung von vier elektrischen Kerzen (LEDs, 12 Volt, max. 6 W) zur Beleuchtung eines Adventskranzes oder einer ähnlichen Weihnachtsdekoration.

Die Elektronik kann datum- und uhrzeitgesteuert vier Ausgänge unabhängig voneinander ein- und ausschalten.

Nach dem Einschalten der Spannungsversorgung startet die Elektronik im „Automatik-Modus“. Dabei wird die Anzahl der einzuschaltenden Kerzen in Abhängigkeit der jeweiligen Adventswoche gewählt. Zu jedem Tageswechsel werden sowohl die Prüfung auf die jeweilige Adventswoche als auch die zufällige Verteilung der einzuschaltenden Kerzen neu durchgeführt und die entsprechenden Kerzen eingeschaltet.

Beispiel: Der aktuelle Tag liegt in der 2. Adventswoche. Dann werden zufällig zwei Kerzen eingeschaltet. Ist der darauffolgende Tag ein Sonntag (Beginn der 3. Adventswoche), so werden zum Tageswechsel drei Kerzen zufällig ausgewählt und eingeschaltet.

Vor dem ersten Advent leuchtet keine Kerze. Ab dem ersten Adventssonntag leuchtet dann eine Kerze. Ab der vierten Adventswoche bleiben alle Kerzen dauerhaft eingeschalte bis zum 6. Januar (Heiligen Drei Könige) des Folgejahres. Ab dem 7. Januar wird keine Kerze mehr eingeschaltet.

Sollen die Kerzen weiterhin leuchten, so muss das Systemdatum um die gewünschte Anzahl Tage zurückgestellt werden (siehe Wartungsmodus).

Die Kerzensteuerung verfügt über einen integrierten WLAN Access Point (SSID und Passwort siehe Tabelle „Technische Daten“). Beim Verbindungsaufbau wird automatisch eine IP-Adresse per DHCP an den Client vergeben.

Per Smartphone, Tablet oder Notebook kann mittels eines aktuellen Browsers die Konfigurationsseite über die URL „http://192.168.0.1“ geöffnet werden. Dabei darf die Ausführung von JavaScript nicht blockiert werden.

Über den „Wartungsmodus“ kann zu Testzwecken jede Kerze einzeln ein- und ausgeschaltet werden. Nach einem Wechsel der RTC-Batterie kann das Systemdatum und die Systemzeit eingestellt werden. Für den erneuten Betrieb im „Automatik-Modus“ muss der Wartungsmodus manuell deaktiviert oder ein Neustart des Systems durch Spannungsverlust eingeleitet werden.

Technische Daten:

elektrische Parameter:

Betriebsspannung	12 Volt DC
Stromaufnahme in Ruhe	~ 60 mA
1 Relais an	~ 100 mA
jedes weitere Relais	+ 40 mA
Sicherung	3,15 A, träge
Verpolungsschutz	Ja, Diode 1N4148
Steckertyp für Spannungsversorgung	GX16-3

Interne Knopfzelle für Echtzeituhr (RTC)	CR2032, 3V Lithium

Ausgänge:

Ausgangsspannung	= Betriebsspannung
Maximaler Strom je Ausgang	500 mA
Maximale Leistung je Ausgang @ 12 DC	6 Watt
Je Ausgang eine Kontroll-LED	Grün
Steckertyp für Kerzenanschluss	GX16-2

Die einzelnen Ausgänge sind nicht separat abgesichert!

Drahtloses Netzwerk:

WLAN Access Point SSID	adventskranz
WLAN Passwort (momentan nicht änderbar)	„MYPASSWORD“
IP Adresse	192.168.0.1
Subnetzmaske	255.255.255.0
Gateway	192.168.0.1

Webseite:

- Ansteuerung jedes einzelnen Kerzenausganges zu Wartungs- und Testzwecken
- Kerzenauswahl neu initiieren
- Einstellen von Datum und Uhrzeit z.B. nach Wechsel der RTC Knopfzelle
- Anzeige Kontaktinformationen

Vorbereitungen

Entwicklungsumgebung

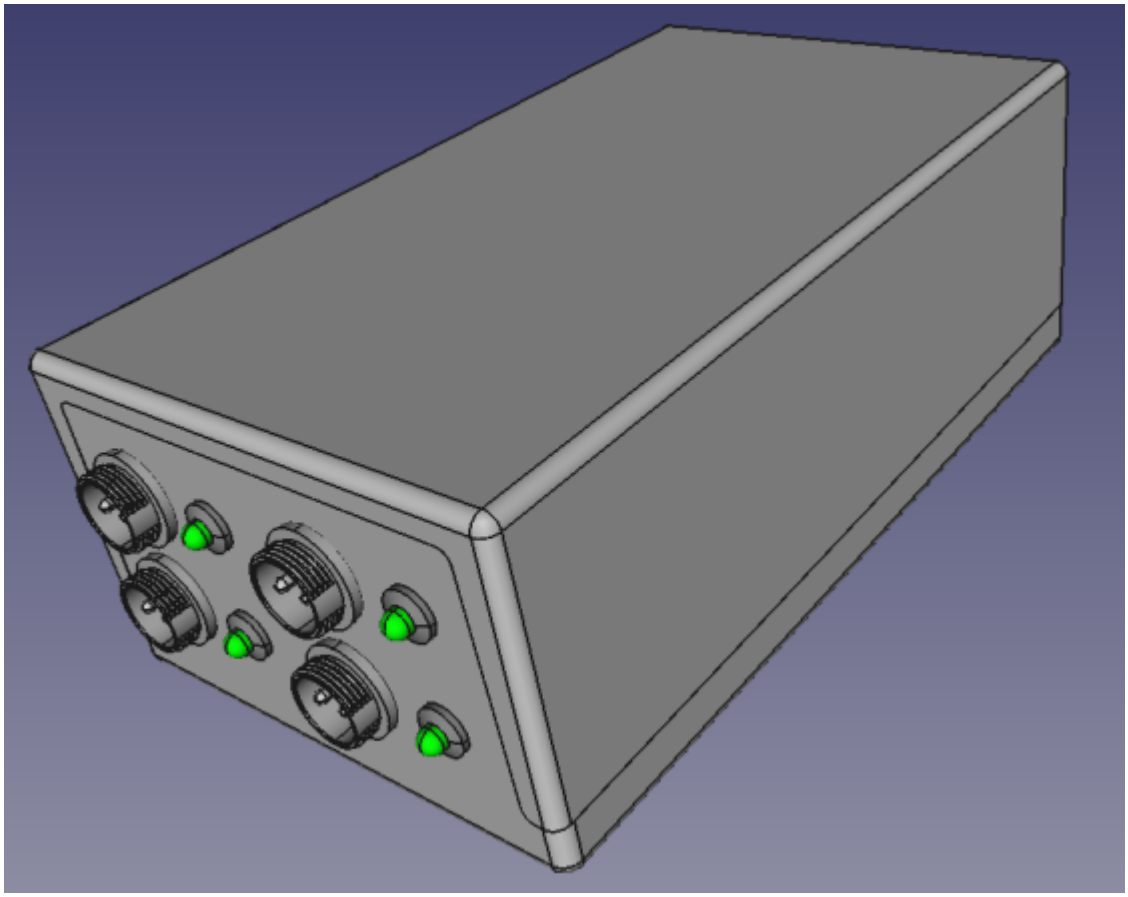
- Arduino IDE installieren
- Notwendige Bibliotheken installieren: Adafruit ESP8266, ESP8266mDNS, ESPAsyncTCP, ESPAsyncWebServer, FS, RTClib

Datenstruktur

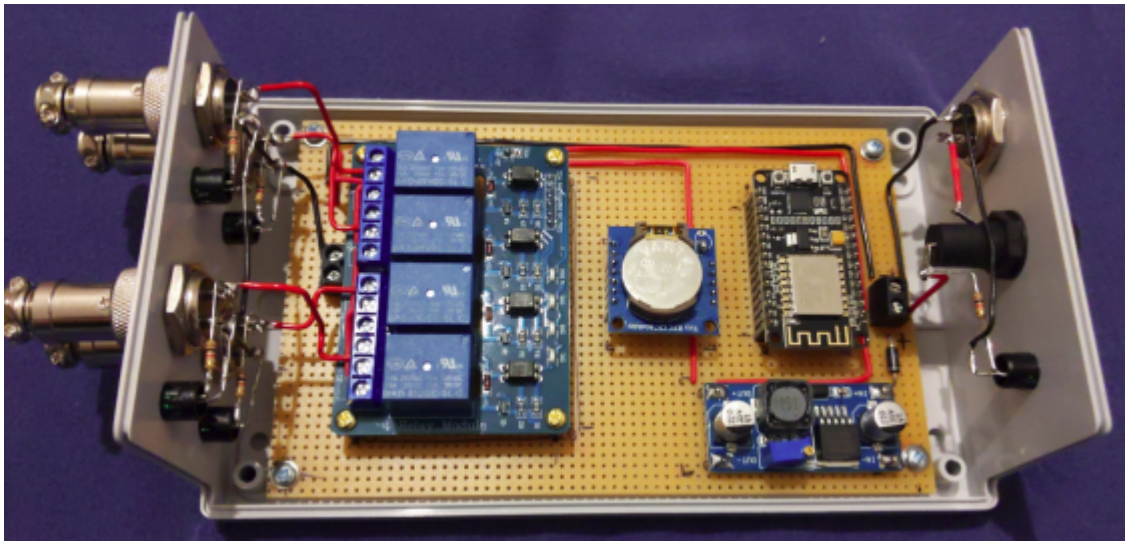
```
Kerzensteuerung
  |-data
    |-ajax-loader.gif
    |-favicon.ico
    |-index.html
    |-jqm-datebox-1.4.5.all.js
    |-jquery-1.12.4.min.js
    |-jquery.mobile-1.4.5.css
    |-jquery.mobile-1.4.5.js
  Kerzensteuerung.ino
```

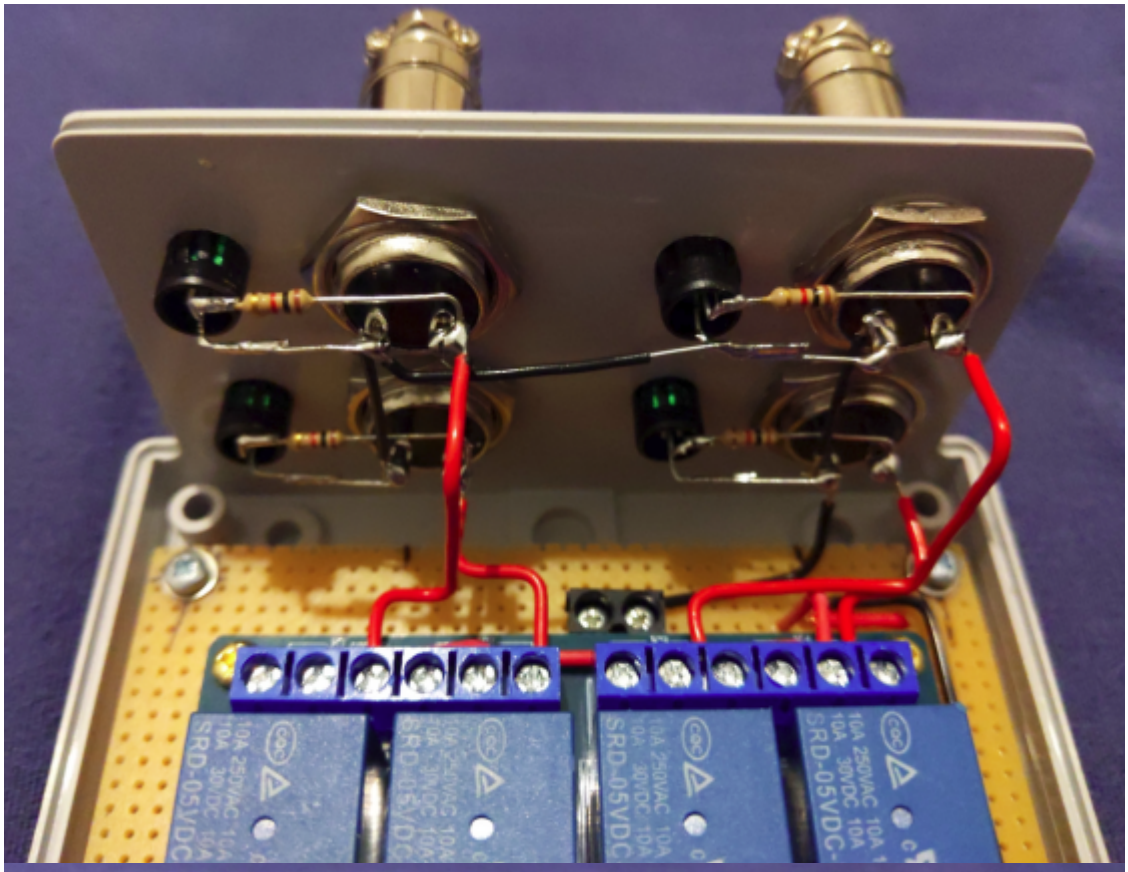
3D Modell

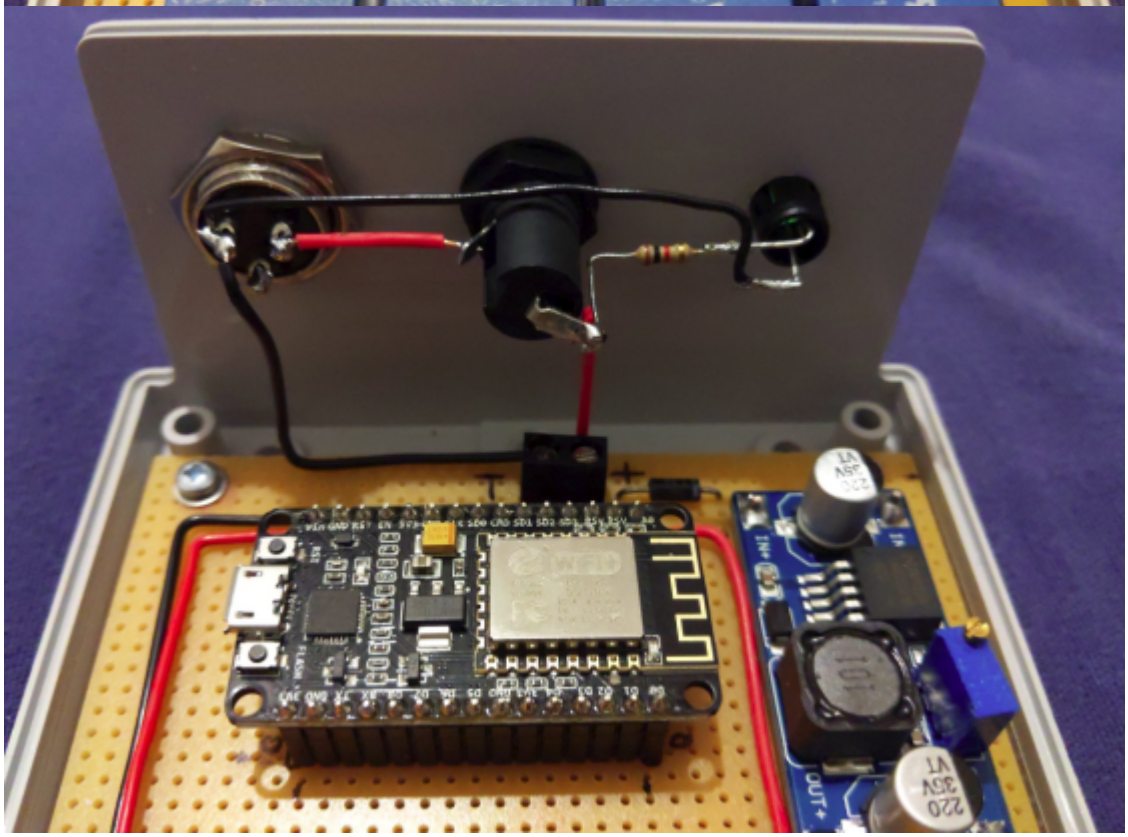
Das 3D Modell habe ich mit Hilfe von FreeCAD erstellt.

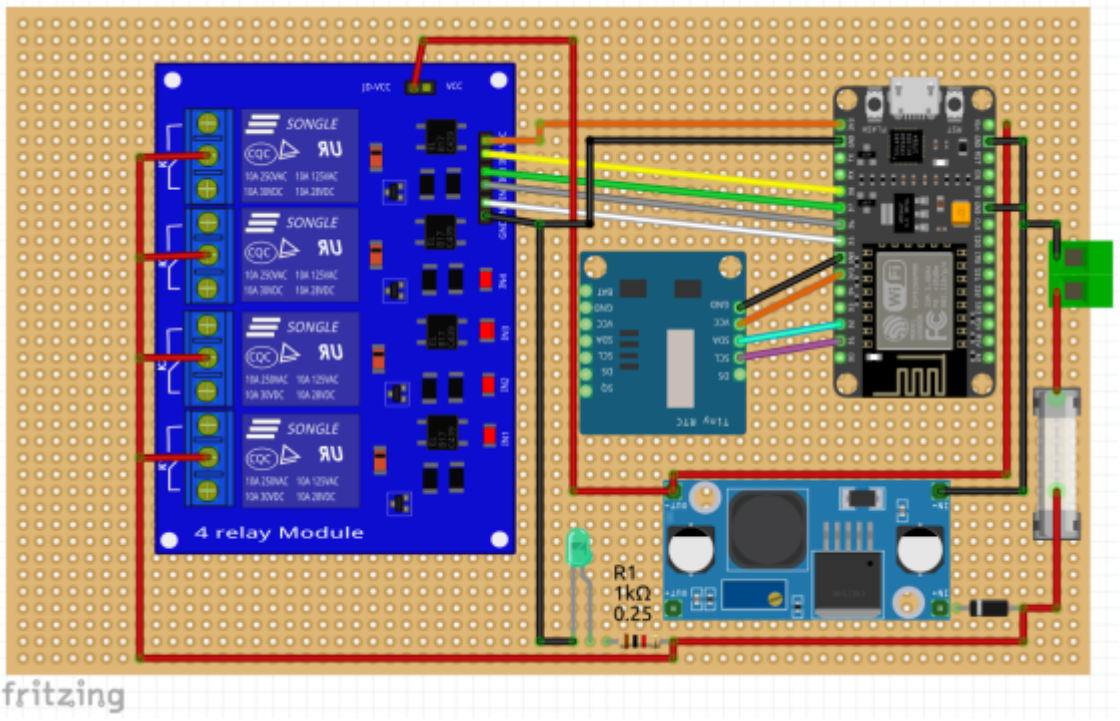


Bilder vom Zusammenbau:




SRD-05VDC
10A 250VAC 10A 30VDC


220 35V
OUT+



Sourcecode

Arduino Sketch

Kerzensteurung.ino

```
/*
  #ifdef ESP32
  #include <WiFi.h>
  #include <ESPAsyncWebServer.h>
  #include <SPIFFS.h>
#else
  #include <Arduino.h>
  #include <ESP8266WiFi.h>
  #include <Hash.h>
  #include <ESPAsyncTCP.h>
  #include <ESPAsyncWebServer.h>
  #include "FS.h"
#endif
*/

// WLAN
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>

// WLAN-Zugangsdaten
const char* ssid = "MYSSID";
```

```
const char* password = "MYPASSWORD";

IPAddress ip(192,168,0,1);
IPAddress gateway(192,168,0,1);
IPAddress subnet(255,255,255,0);

// Hostname of the ESPs -> http://esp8266.local/
const char* espHostname = "adventskranz";

// Webserver
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#define PORT 80     // Standard HTTP-Port 80
AsyncWebServer server(PORT);

// Filesystem
#include <FS.h>
// Date and time functions using a DS1307 RTC connected via I2C and
Wire lib
#include "RTClib.h"

RTC_DS1307 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"};

DateTime now, ChristmasDay, ThreeKingsDayThisYear,
ThreeKingsDayNextYear, AdventDay[4], last_day;

#define MAX_CANDLES 4
#define DEBUG true
#define SER_DEBUG_OUTPUT_INTERVALL 1000  // in milliseconds
#define LED_TEST_INTERVALL 5000  // in milliseconds

const int CandlePin[] = {          // define each candle control gpi/o
pin to use
 14,    // 1. LED Candle @ Pin D5
 12,    // 2. LED Candle @ Pin D6
 13,    // 3. LED Candle @ Pin D7
 15,    // 4. LED Candle @ Pin D8
};

int num_of_candles = 0;
int counter = 0;
int CandleToUse[] = {0, 0, 0, 0}; // Init CandlesToUse with zeros
unsigned long time_start = 0;     // Start time periode in milliseconds
unsigned long time_stop = 0;      // Stop time periode in milliseconds
bool maintenance = false;         // Set maintenance mode [true|false]

void setupServerRoutes();
void setupFilePaths();
```

```
// What should happen when page isn't found:
void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/html", "<center><h1>404 error - Page not
found!</h1></center>");
}

void setup() {
  Serial.begin(115200);
  #ifndef ESP8266
    while (!Serial); // wait for serial port to connect. Needed for
native USB
  #endif
  //Serial.println();
  Serial.println("Entering setup ...");
  // delay(500);

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    while (1) delay(10);
  }

  // Init all candle pins as "output" and switch off relay
  Serial.println("Init candle control pins ...");
  for (int counter = 0; counter < MAX_CANDLES; counter++){
    pinMode(CandlePin[counter], OUTPUT);
    digitalWrite(CandlePin[counter], HIGH);
  }

//  Usefull suff for handling the RTC:
//  if (! rtc.isrunning()) {
//    Serial.println("RTC is NOT running, let's set the time!");
    // When time needs to be set on a new device, or after a power
loss, the
    // following line sets the RTC to the date & time this sketch was
compiled
//    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example
to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
//  }

  // When time needs to be re-set on a previously configured device,
the
  // following line sets the RTC to the date & time this sketch was
compiled
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example
```

```
to set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2022, 10, 9, 14, 20, 0));

  if(SPIFFS.begin()){
    Serial.println("File system initialized.");
    }
  else{
    Serial.println("Error initializing file system");
  }

/* when ESP8266 should act as a WiFi client:
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  if (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("WLAN Verbindung fehlgeschlagen");
    return;
  }
    Serial.print("Verbunden! IP-Adresse: ");
Serial.println(WiFi.localIP());
*/
  Serial.println("Configuring network parameters ...");
  WiFi.softAPConfig(ip, gateway, subnet);
  Serial.println("Configuring WiFi parameters ... ");
  WiFi.softAP(ssid, password);

  // Starting mDNS-Servers
  if (!MDNS.begin(espHostname)) {
    Serial.println("Error starting mDNS-Server");
  }

  // Define Server routes in separate function
  setupServerRoutes();

  // Define file routes in separate function
  setupFilePaths();

  // Start HTTP server function:
  Serial.println("Starting HTTP server ...");
  server.begin();

  random_candles();               // random all candles
  Serial.println("Switch OFF all candles...");
  set_all_candles(HIGH);  // switch all candles off
  Serial.println("Switch ON all candles...");
  set_all_candles(LOW);   // switch all candles on
  delay(LED_TEST_INTERVALL);       // wait for LED_TEST_INTERVALL
seconds
  Serial.println("Switch OFF all candles...");
  set_all_candles(HIGH);  // switch all candles off
```

```
  get_current_timestamp();        // get current timestamp
  DateTime last_day ((now.year()), (now.month()), (now.day()), 00, 00,
00);  // save actual day for later use

} // END OF SETUP()

void loop() {
  MDNS.update();

  time_stop = millis();  // get current ticks counter
  // execute every SER_DEBUG_OUTPUT_INTERVALL milliseconds
  if (((time_stop - time_start) > SER_DEBUG_OUTPUT_INTERVALL) &&
(maintenance == false)) {
    time_start = time_stop;

    get_current_timestamp();        // get current timestamp
    //Serial.println("Actual time and date: ");
    printDate(now);
    //Serial.print(".");

    if ( last_day.day() != now.day() ) {     // check if we have a new
day to scramble the candles again
      //Serial.print("new day!");
      set_all_candles(HIGH);                 // switch all candles off
      random_candles();                      // random all candles
    }

    //Serial.println("Calculate celebration dates ...");
    DateTime ChristmasDay (now.year(), 12, 24, 00, 00, 00);
    //Serial.print("Christmas: ");
    //printDate(ChristmasDay);
    DateTime ThreeKingsDayThisYear ((now.year()), 01, 06, 23, 59, 59);
    //Serial.print("Three Kings this year: ");
    //printDate(ThreeKingsDayThisYear);
    DateTime ThreeKingsDayNextYear ((now.year()+1), 01, 06, 23, 59,
59);
    //Serial.print("Three Kings next year: ");
    //printDate(ThreeKingsDayNextYear);

    get_advent_days(ChristmasDay);

    //Serial.println("Check calendar week ...");
    if (now.unixtime() <= ThreeKingsDayThisYear.unixtime()) {
      //Serial.println("we are past New Year's Eve but before Three
Kings Day  -> all candles on");
      num_of_candles = 4;
    }
    else if ((now.unixtime() > ThreeKingsDayThisYear.unixtime()) &&
```

```
(now.unixtime() < AdventDay[0].unixtime()))  {
      //Serial.println("we are between Three Kings Day and 1st. advent!
-> all candles off");
      num_of_candles = 0;
    }
    else if ((now.unixtime() >= AdventDay[0].unixtime()) &&
(now.unixtime() < AdventDay[1].unixtime())) {
      //Serial.println("we are in 1st. advent week! -> turn 1 candles
on");
      num_of_candles = 1;
    }
    else if ((now.unixtime() >= AdventDay[1].unixtime()) &&
(now.unixtime() < AdventDay[2].unixtime())) {
        //Serial.println("we are in 2nd. advent week! -> turn 2 candles
on");
      num_of_candles = 2;
    }
    else if ((now.unixtime() >= AdventDay[2].unixtime()) &&
(now.unixtime() < AdventDay[3].unixtime())) {
      //Serial.println("we are in 3rd. advent week! -> turn 3 candles
on");
      num_of_candles = 3;
    }
    else if ((now.unixtime() >= AdventDay[3].unixtime()) &&
(now.unixtime() <= ChristmasDay.unixtime())) {
      //Serial.println("we are in 4th. advent week! -> turn all 4
candles on");
      num_of_candles = 4;
    }
    else if ((now.unixtime() >= ChristmasDay.unixtime())&&
(ThreeKingsDayNextYear.unixtime())) {
      //Serial.println("we are bast Christmas day but before Three
Kings day! -> turn all 4 candles on");
      num_of_candles = 4;
    }
    else {
      //Serial.println("We should not go here, what's wrong?");
    }

    if (num_of_candles >= 1){
      for (int counter = 0; counter < num_of_candles; counter++){
        digitalWrite(CandleToUse[counter], LOW);                    //
switch CandleToUse[counter] ON (yes LOW means ON because the relay
logic is low active)
      }
    }
    else{
      set_all_candles(HIGH);    // otherwise switch all candles off
    }
    //Serial.println();
  }
```

```
  last_day = now;
} // END OF LOOP()

void setupServerRoutes(){
  // Document-Root --> folder "data"
  // All file path must be defines relative to "data"
  // Deliver file "index.html"
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html", String(), false, processor);
  //request->send(SPIFFS, "/index.html");
  });

  server.onNotFound(notFound);
}

void setupFilePaths(){
  server.on("/jquery.mobile-1.4.5.css", HTTP_GET,
[](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/jquery.mobile-1.4.5.css", "text/css");
  });

  server.on("/jqm-datebox-1.4.5.all.js", HTTP_GET,
[](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/jqm-datebox-1.4.5.all.js",
"text/javascript");
  });

  server.on("/jquery.mobile-1.4.5.js", HTTP_GET,
[](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/jquery.mobile-1.4.5.js",
"text/javascript");
  });

  server.on("/jquery-1.12.4.min.js", HTTP_GET, [](AsyncWebServerRequest
*request){
    request->send(SPIFFS, "/jquery-1.12.4.min.js", "text/javascript");
  });

  server.on("/images/ajax-loader.gif", HTTP_GET,
[](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/ajax-loader.gif", "image/gif");
  });

  server.on("/favicon.ico", HTTP_GET, [](AsyncWebServerRequest
*request){
    request->send(SPIFFS, "/favicon.ico", "text/html");
  });

  server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
```

```
  String message;
  String state;
  if (request->hasParam("pinstate")) {
    message = request->getParam("pinstate")->value();
    state = get_PinState(message.toInt());
    //Serial.print("GET Pinstate: ");
    //Serial.print(message);
    //Serial.print(" -> ");
    //Serial.println(state);
    //request->send(200, "text/plain", "Hello, GET: " + message);
    //request->send(200, "text/plain", "ON");
    request->send(200, "text/plain", state.c_str());
  }
  else if (request->hasParam("maintenance")) {
    if (maintenance == true){
      message = "ON";
    }
    else {
       message = "OFF";
    }
    //Serial.print("GET Mainenance: ");
    //Serial.println(message);
    //request->send(200, "text/plain", "Hello, GET: " + message);
    //request->send(200, "text/plain", "ON");
    request->send(200, "text/plain", message.c_str());
  }
  else if (request->hasParam("time")) {
    //Serial.print("GET Time: ");
    //Serial.println(message);
    //request->send(200, "text/plain", "Hello, GET: " + message);
    //request->send(200, "text/plain", "ON");
    request->send_P(200, "text/plain", getTime().c_str());
  }
  else if (request->hasParam("date")) {
    //Serial.print("GET Date: ");
    //Serial.println(message);
    //request->send(200, "text/plain", "Hello, GET: " + message);
    //request->send(200, "text/plain", "ON");
    request->send_P(200, "text/plain", getDate().c_str());
  }
  else {
    message = "No message sent";
    Serial.print("GET: ");
    Serial.println(message);
  }
});

server.on("/set", HTTP_GET, [] (AsyncWebServerRequest *request) {
  String message;
  if (request->hasParam("pinstate")) {
    message = request->getParam("pinstate")->value();
```

```
      //Serial.print("SET Pinstate: ");
      //Serial.println(message);
      int candle_number =
message.substring(0,message.indexOf(':')).toInt();
      String state = message.substring(message.indexOf(':')+1,
message.length());
      if (state == "ON") {
        set_candle(candle_number, LOW);
      }
      else {
        set_candle(candle_number, HIGH);
      }
      //request->send(200, "text/plain", "Hello, GET: " + message);
      //request->send(200, "text/plain", "ON");
      //request->send(200, "text/plain",
get_PinState(message.toInt()).c_str());
      request->send(200);
    }

    else if (request->hasParam("maintenance")) {
      message = request->getParam("maintenance")->value();
      if (message == "true"){
        maintenance = true;       // enable maintenance mote
        set_all_candles(HIGH);    // switch all candles off
      }
      else {
        set_all_candles(HIGH);    // switch all candles off
        maintenance = false;      // disable maintenance mode
      }
      //Serial.print("SET Maintenance: ");
      //Serial.println(message);
      //Serial.print(":");
      //Serial.println(value);
      request->send(200);
    }

    else if (request->hasParam("reorder_candles")) {
      //message = request->getParam("reorder_candles")->value();
      //Serial.print("SET Reorder Candles: ");
      //Serial.println(message);
      reorder_candles();
      request->send(200);
    }

    else if (request->hasParam("time")) {
      message = request->getParam("time")->value();
      //Serial.print("SET Time to: ");
      //Serial.println(message);
      int set_hour = message.substring(0,message.indexOf(':')).toInt();
```

```
      int set_minute = message.substring(message.indexOf(':')+1,
message.length()).toInt();
      now = rtc.now();                    // get current time and date
      rtc.adjust(DateTime(now.year(), now.month(), now.day(), set_hour,
set_minute, 0));
      request->send(200);
    }

    else if (request->hasParam("date")) {
      message = request->getParam("date")->value();
      //Serial.print("SET Date to: ");
      //Serial.println(message);
      int first = message.indexOf('-');
      int second = message.indexOf('-', first + 1 );
      int set_year = message.substring(0,first).toInt();
      int set_month = message.substring(first+1,second).toInt();
      int set_day = message.substring(second+1,
message.length()).toInt();
      now = rtc.now();                    // get current time and date
      rtc.adjust(DateTime(set_year, set_month, set_day, now.hour(),
now.minute(), now.second()));
      set_all_candles(HIGH);    // switch all candles off
      request->send(200);
    }

    else {
      message = "No message sent";
      //Serial.print("SET: ");
      //Serial.println(message);
      request->send(200);
    }
  });
}

String processor(const String& var){
  Serial.println(var);
/* this function to replace "placeholders (%PLACEHOLDER%) is not used
anymore but we keep this strukture maybe for later use ...

  if (var == "SWITCHSTATE1"){
    return get_PinState(0);
  }
  else if (var == "SWITCHSTATE2"){
    return get_PinState(1);
  }
  else if (var == "SWITCHSTATE3"){
    return get_PinState(2);
  }
  else if (var == "SWITCHSTATE4"){
    return get_PinState(3);
  }
```

```
  else if (var == "MAINTENACESTATE"){
    if (maintenance == false){
      return "";
    }
    else{
    return "checked";
    }
  }
*/
return "";   // returns nothing ;-)
}

String getTime() {
  char actual_time[8];
  now = rtc.now();                  // get current time and date
  sprintf(actual_time,"%02u:%02u:%02u", now.hour(), now.minute(),
now.second());
  //Serial.print("Send system time: ");
  //Serial.println(actual_time);
  return String(actual_time);
}

String getDate() {
  char actual_date[10];
  now = rtc.now();                  // get current time and date
  sprintf(actual_date,"%02u:%02u:%02u", now.day(), now.month(),
now.year());
  //Serial.print("Send system date: ");
  //Serial.println(actual_date);
  return String(actual_date);
}

void random_candles (){
  for (int counter = 0; counter < MAX_CANDLES; counter++){
    CandleToUse[counter] = 0;
  }
  int buff = 0;
  if (DEBUG == true) {
    //Serial.println("Debug: Entering \"random_candles\"");
  }
  counter = 0;
  do {
   buff = random(0, MAX_CANDLES);
   if (CandleToUse[buff] == 0 ){
     CandleToUse[buff] = CandlePin[counter];
     counter+=1;
   }
  } while (counter < MAX_CANDLES);
```

```
/*  if (DEBUG == true) {
    Serial.print("Debug: Candle order today: ");
    Serial.print(CandleToUse[0], DEC);
    Serial.print(", ");
    Serial.print(CandleToUse[1], DEC);
    Serial.print(", ");
    Serial.print(CandleToUse[2], DEC);
    Serial.print(", ");
    Serial.println(CandleToUse[3], DEC);
  }
*/
}

void reorder_candles(){
  if (maintenance == false){
    maintenance = true;       // enable maintenance mote
    set_all_candles(HIGH);    // switch all candles off
    random_candles();                 // random all candles
    maintenance = false;      // disable maintenance mode
  }
  else {
    set_all_candles(HIGH);    // switch all candles off
    random_candles();                 // random all candles
  }
}

void set_candle (int candle_number, bool switch_state){  //
switch_state == LOW --> candle on; switch_state == HIGH --> candle off
  //for (int counter = 0; counter < numberofcandles; counter++){
    if(digitalRead(CandlePin[candle_number]) == switch_state){
      //Serial.println("no candle state change");
    }
    else {
/*      if (DEBUG == true) {
        Serial.print("Debug: Switching candle ");
        Serial.print(counter, DEC);
        Serial.print(" to ");
        Serial.println(switch_state ? "OFF" : "ON");
      } */
      digitalWrite(CandlePin[candle_number], switch_state);
    }
  //}
}

void set_all_candles(bool switch_state){  // switch_state == LOW -->
candle on; switch_state == HIGH --> candle off
  for (int counter = 0; counter < MAX_CANDLES; counter++){
    set_candle(counter, switch_state);
/*      if (DEBUG == true) {
        Serial.print("Debug: Switching candle ");
```

```
        Serial.print(counter, DEC);
        Serial.print(" to ");
        Serial.println(switch_state ? "OFF" : "ON");
      } */
  }
}

String get_PinState(int output){
  //Serial.print("Candle ");
  //Serial.print(output);
  if(digitalRead(CandlePin[output]) == HIGH){
    //Serial.println(" is OFF");
    return "OFF";
  }
  else {
    //Serial.println(" is ON");
    return "ON";
  }
}

void get_advent_days (const DateTime& dt){
  AdventDay[3] = (dt - TimeSpan(dt.dayOfTheWeek(),0,0,0));
  for (int counter = 0; counter <= 2 ; counter++){
    AdventDay[counter] = AdventDay[3] - TimeSpan(((3 - counter) * 7),
0, 0, 0);
  }
  // print advent days
/*  if (DEBUG == true) {
    Serial.print("1st. advent day: ");
    printDate(AdventDay[0]);
    //Serial.println(dateToUnix(AdventDay[0]));
    Serial.print("2nd. advent day: ");
    printDate(AdventDay[1]);
    //Serial.println(dateToUnix(AdventDay[1]));
    Serial.print("3rd. advent day: ");
    printDate(AdventDay[2]);
    Serial.print("4th. advent day: ");
    printDate(AdventDay[3]);
  }
*/
}

void get_current_timestamp(){
  now = (00, 00, 00, 00, 00, 00);
  now = rtc.now();                    // get current time and date
  delay(50);
/*  if (DEBUG == true) {
  printDate(now);
  } */
```

```
}

void printDate(const DateTime& dt)
{
  char s[80];
  sprintf(s,"Time: %02u:%02u:%02u - Date: %02u.%02u.%04u - Day of Week:
%1u",dt.hour(), dt.minute(), dt.second(), dt.day(), dt.month(),
dt.year(), dt.dayOfTheWeek());
  Serial.println(s);
}
```

Index.html

index.html

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1">
  <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
  <script type='text/javascript' src='jquery-1.12.4.min.js'></script>
  <script type='text/javascript' src='jquery.mobile-1.4.5.js'></script>
  <script type='text/javascript' src='jqm-
datebox-1.4.5.all.js'></script>
  <link href="jquery.mobile-1.4.5.css" rel="stylesheet"/>

<!--
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default_defau
lt
  <script type='text/javascript'
src='https://code.jquery.com/jquery-1.12.4.js'></script>
  <script type='text/javascript'
src='https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.js'></scr
ipt>
  <script type='text/javascript'
src='https://cdn.jtsage.com/datebox/1.4.5/jqm-datebox-1.4.5.all.js'></s
cript>
  <link
href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.css"
rel="stylesheet"/>
-->

  <script type = "application/javascript">
    function get_parameter(command, parameter) {
      var xhttp = new XMLHttpRequest();
      switch(command) {
        case "pinstate":
```

```
          xhttp.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {
            $("#LED" +
parameter).val(this.responseText).slider('refresh');
            //alert(command + parameter + ": " + this.responseText);
          }
        };
                          //get?command=<parameter>
        xhttp.open("GET", "get?" + command + "=" + parameter, true);
        xhttp.send(null);
        break;
        case "maintenance":
          xhttp.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {
            $("#Maintenance").val(this.responseText).slider('refresh');
          }
        };
                          //get?command=<parameter>
        xhttp.open("GET", "get?" + command + "=" + "", true);
        xhttp.send(null);
        break;
        case "time":
          xhttp.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {
            document.getElementById("actual_time").innerHTML =
this.responseText;
          }
        };
        xhttp.open("GET", "get?" + command, true);
        xhttp.send(null);
        break;
        case "date":
          xhttp.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {
            document.getElementById("actual_date").innerHTML =
this.responseText;
          }
        };
        xhttp.open("GET", "get?" + command, true);
        xhttp.send(null);
        break;
        //case y:
          //code block
        //break;
        default:
        // code block
      }
    };
```

```
    function set_parameter(command, parameter, value) {
      var xhttp = new XMLHttpRequest();
      switch(command) {
        case "pinstate":
          var xhr = new XMLHttpRequest();
                         //set?command=<parameter:value>
          xhr.open("GET", "set?" + command + "=" + parameter + ":" +
value, true);
          xhr.send(null);
        break;
        case "maintenance":
          var xhr = new XMLHttpRequest();
          xhr.open("GET", "set?" + command + "=" + value, true);
          xhr.send(null);
        break;
        case "reorder_candles":
          var xhr = new XMLHttpRequest();
          xhr.open("GET", "set?" + command + "=" + "true", true);
          xhr.send(null);
        break;
        case "time":
          var xhr = new XMLHttpRequest();
          xhr.open("GET", "set?" + command + "=" + value, true);
          xhr.send(null);
        break;
        case "date":
          var xhr = new XMLHttpRequest();
          xhr.open("GET", "set?" + command + "=" + value, true);
          xhr.send(null);
        break;
        //case y:
          //code block
        //break;
        default:
        // code block
      }
    };

    setInterval(function() {
      get_parameter('time', '');
    }, 1000);

    setInterval(function() {
      get_parameter('date', '');
    }, 1000);

    function switchcandle(parameter) {
      var FS = $("#LED" + parameter).val();
      if ("OFF" == FS) {
        set_parameter('pinstate', parameter, 'OFF');
     // alert("Schalter #LED" + parameter + " ist jetzt " + FS );
```

```
      }
      else{
        set_parameter('pinstate', parameter, 'ON');
     // alert("Schalter #LED" + parameter + " ist jetzt " + FS );
      }
    }

    function reorder_candles() {
      set_parameter('reorder_candles', '', '')
      //alert(command + parameter + ": " + this.responseText);
    }

    function set_systime() {
      var new_time = $("#SysTime").val();
      if (new_time == "") {
        alert("Keine neue Systemzeit ausgewählt!");
      }
      else{
        set_parameter('time', '', new_time)
      }
    }

    function set_sysdate() {
      var new_date = $("#SysDate").val();
      if (new_date == "") {
        alert("Kein neues Systemdatum ausgewählt!");
      }
      else{
        set_parameter('date', '', new_date)
      }
    }

    setInterval(function() {
      get_parameter('maintenance', '');
      get_parameter('pinstate', '0');
      get_parameter('pinstate', '1');
      get_parameter('pinstate', '2');
      get_parameter('pinstate', '3');
    }, 1000);

    function toggle_Maintenance() {
      var MM = $("#Maintenance").val();
      if ("ON" == MM) {
      //  console.log("Maintenance mode -> ON");
        set_parameter("maintenance", '', true);
        document.getElementById("Maintenance_Lable").innerHTML =
"Wartungsmodus aktiviert!";

        $("#LED0").slider("option", "disabled", false);
```

```
        $("#LED1").slider("option", "disabled", false);
        $("#LED2").slider("option", "disabled", false);
        $("#LED3").slider("option", "disabled", false);
      }

      else {
        $("#LED0").slider("option", "disabled", true);
        $("#LED1").slider("option", "disabled", true);
        $("#LED2").slider("option", "disabled", true);
        $("#LED3").slider("option", "disabled", true);

        set_parameter("maintenance", '', false);
        document.getElementById("Maintenance_Lable").innerHTML =
"Wartungsmodus deaktiviert!";
      }
    }

    $(document).on("pagecreate", function () {
      $("#LED0").slider("option", "disabled", true).slider('refresh');
      $("#LED1").slider("option", "disabled", true).slider('refresh');
      $("#LED2").slider("option", "disabled", true).slider('refresh');
      $("#LED3").slider("option", "disabled", true).slider('refresh');
    });
  </script>

  <style>
    th, td {border-bottom: 2px solid #ccc;}
  </style>

</head>

<body>
  <div data-role="page" id="Kerzensteuerung">
    <div data-role="header">
      <h2>Adventskranz-Kerzensteuerung</h2>
      <div data-role="navbar">
        <ul>
        <li><a href="#timedate">Datum / Uhrzeit</a></li>
        <li><a href="#about">Kontakt</a></li>
        </ul>
      </div>
    </div>

    <div>
      <hr>
    </div>

    <div data-role="main">
    <table style="width:100%">
      <tr>
        <td style="text-align:center">
```

```
          <label id="Maintenance_Lable" style="display:inline"
for="slider_mm">Wartungsmodus deaktiviert!</label>
        </td>
        <td style="text-align:left">
          <select id="Maintenance" data-role="slider"
onchange="toggle_Maintenance()">
            <option value="OFF">Aus</option>
            <option value="ON">An</option>
          </select>
        </td>
      </tr>

      <tr>
        <td style="text-align:center">
          <label style="display:inline" for="slider_s0">1.
Kerze:</label>
        </td>
        <td style="text-align:left">
          <select id="LED0" data-role="slider"
onchange="switchcandle('0')">
            <option value="OFF">Aus</option>
            <option value="ON">An</option>
          </select>
        </td>
      </tr>

      <tr>
        <td style="text-align:center">
          <label style="display:inline" for="slider_s1">2.
Kerze:</label>
        </td>
        <td style="text-align:left">
        <select id="LED1" data-role="slider"
onchange="switchcandle('1')">
            <option value="OFF">Aus</option>
            <option value="ON">An</option>
          </select>
        </td>
      </tr>

      <tr>
        <td style="text-align:center">
          <label style="display:inline" for="slider_s2">3.
Kerze:</label>
        </td>
        <td style="text-align:left">
          <select id="LED2" data-role="slider"
onchange="switchcandle('2')">
            <option value="OFF">Aus</option>
```

```
            <option value="ON">An</option>
          </select>
        </td>
      </tr>

      <tr>
        <td style="text-align:center">
          <label style="display:inline" for="slider_s3">4.
Kerze:</label>
        </td>
        <td style="text-align:left">
          <select id="LED3" data-role="slider"
onchange="switchcandle('3')">
            <option value="OFF">Aus</option>
            <option value="ON">An</option>
          </select>
        </td>
      </tr>
    </table>
    <hr>
    <button type="button" name="ReorderCandles" id="ReorderCandles"
onclick="reorder_candles()">Kerzen neu verteilen</button>
    <hr>
    </div>
  </div>

  <div data-role="page" id="timedate">
    <div data-role="header">
      <h2>Systemzeit- und Datum einstellen:</h2>
    </div>
    <div data-role="navbar">
      <ul>
      <li><a href="#Kerzensteuerung">Zurück</a></li>
      </ul>
    </div>
    <div data-role="main">
      <br>
      <b>Systemzeit / Datum:</b>
      <p>
        <span class="sensor-labels">aktuelle Systemzeit:</span>
        <span id="actual_time"></span>
        <br>
        <span class="sensor-labels">aktuelles Systemdatum:</span>
        <span id="actual_date"></span>
      </p>

      <div data-role="content">
        <label for="SysTime">Systemzeit einstellen:</label>
        <input name="SysTime" id="SysTime" type="text" data-
role="datebox" data-options='{"mode":"timebox", "useClearButton":true,
"useLang":"de", "useNewStyle":true, "overrideTimeFormat": 24,
```

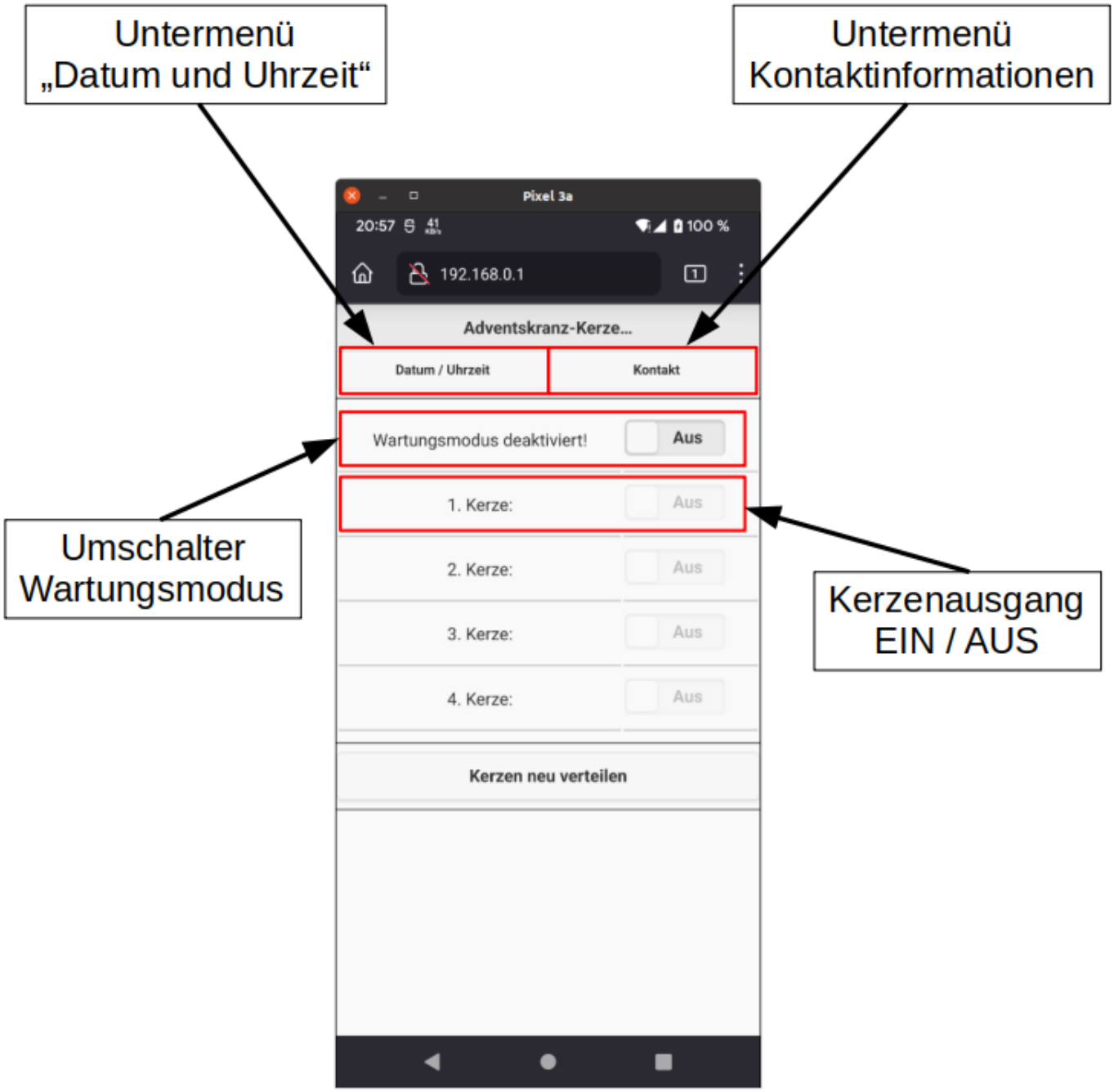
```
"themeButton": "b", "themeInput": "a", "theme": "b", "themeHeader":
"b"}' />
          <button type="button" name="SysTime" id="SysTime"
onclick="set_systime()">Uhrzeit übernehmen</button>
        </form>
      </div>

      <div data-role="content">
        <label for="SysDate">Systemdatum einstellen:</label>
        <input name="SysDate" id="SysDate" type="text" data-
role="datebox" data-options='{"mode":"datebox", "useClearButton":true,
"useNewStyle":true,  "themeButton": "b", "themeInput": "a", "theme":
"b", "themeHeader": "b"}' />
          <button type="button" name="SysDate" id="SysDate"
onclick="set_sysdate()">Datum übernehmen</button>
        </form>
      </div>
    </div>
  </div>

  <div data-role="page" id="about">
    <div data-role="header">
      <h2>Über diese Seite ...</h2>
    </div>
    <div data-role="navbar">
      <ul>
      <li><a href="#Kerzensteuerung">Zurück</a></li>
      </ul>
    </div>
    <div data-role="main">
      <p>Webseite zur Steuerung eines Adventskranzes mit vier
elektrischen Kerzen.<br>
        <br> Entwickelt von:
        <br> Christoph von Thülen
        <br> Straße
        <br> PLZ Stadt
        <br> E-Mail:<a href="mailto:E-Mail_Adresse">E-Mail_Adresse</a>
        <br>
        <br> SW-Version: 1.0.0
        <br> HW-Version: ESP8266
        <br> 27.10.2022
      </p>
    </div>
  </div>
</body>
</html>
```

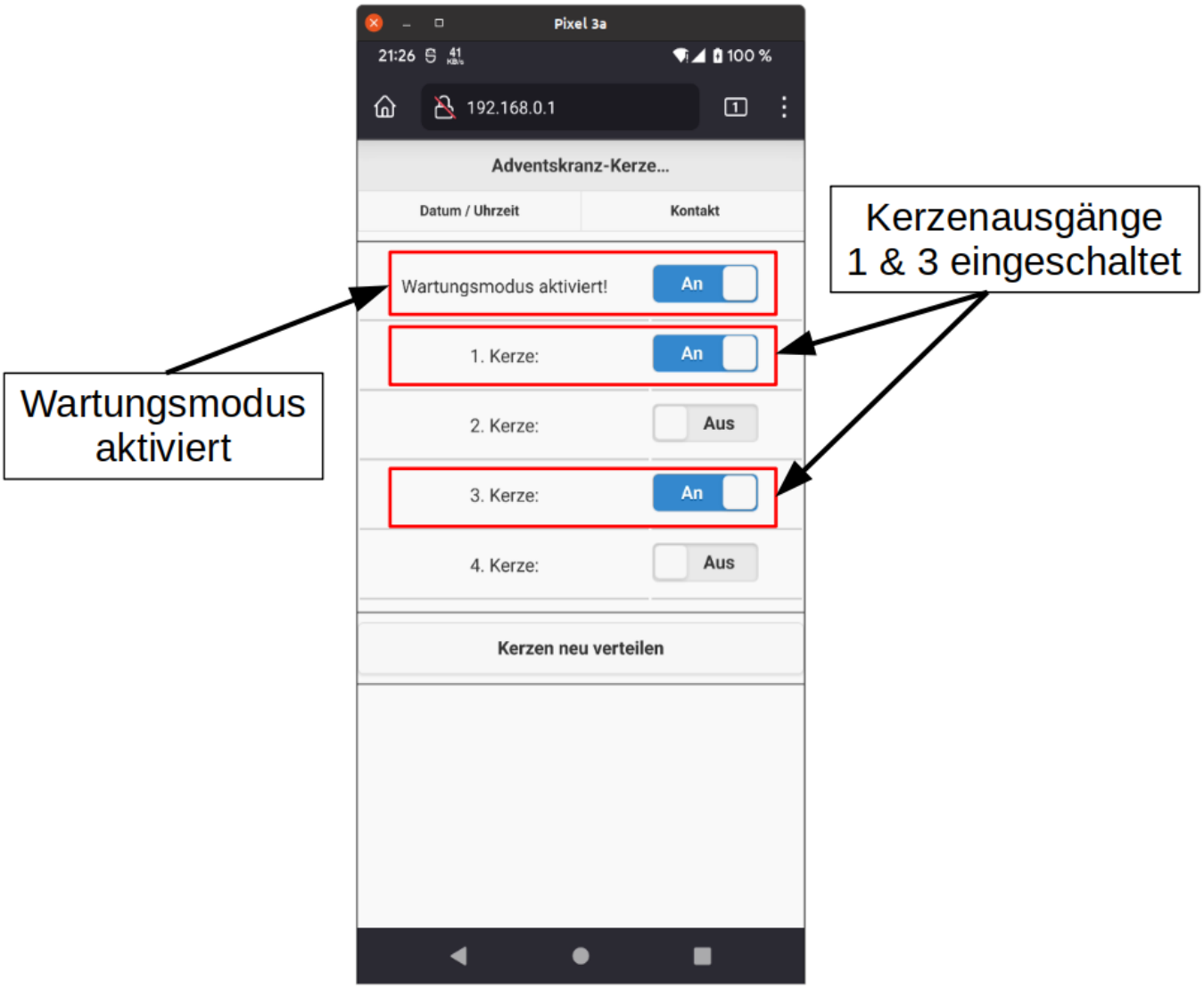
Steuerung via Webseite:

URL: http://192.168.0.1



Um die einzelnen Kerzenausgänge ein- und ausschalten zu können muss der Wartungsmodus aktiviert werden.
Nach aktivieren des Wartungsmodus werden alle Kerzenausgänge automatisch ausgeschaltet.
Nach deaktivieren des Wartungsmodus werden alle Kerzenausgänge automatisch anhand des Systemdatums aktiviert.

Beispiel Wartungsmodus


Pixel 3a
21:26
100 %
192.168.0.1
Adventskranz-Kerze...
Datum / Uhrzeit
Kontakt
Wartungsmodus aktiviert!
An
1. Kerze:
An
2. Kerze:
Aus
3. Kerze:
An
4. Kerze:
Aus
Kerzen neu verteilen
Wartungsmodus aktiviert
Kerzenausgänge 1 & 3 eingeschaltet

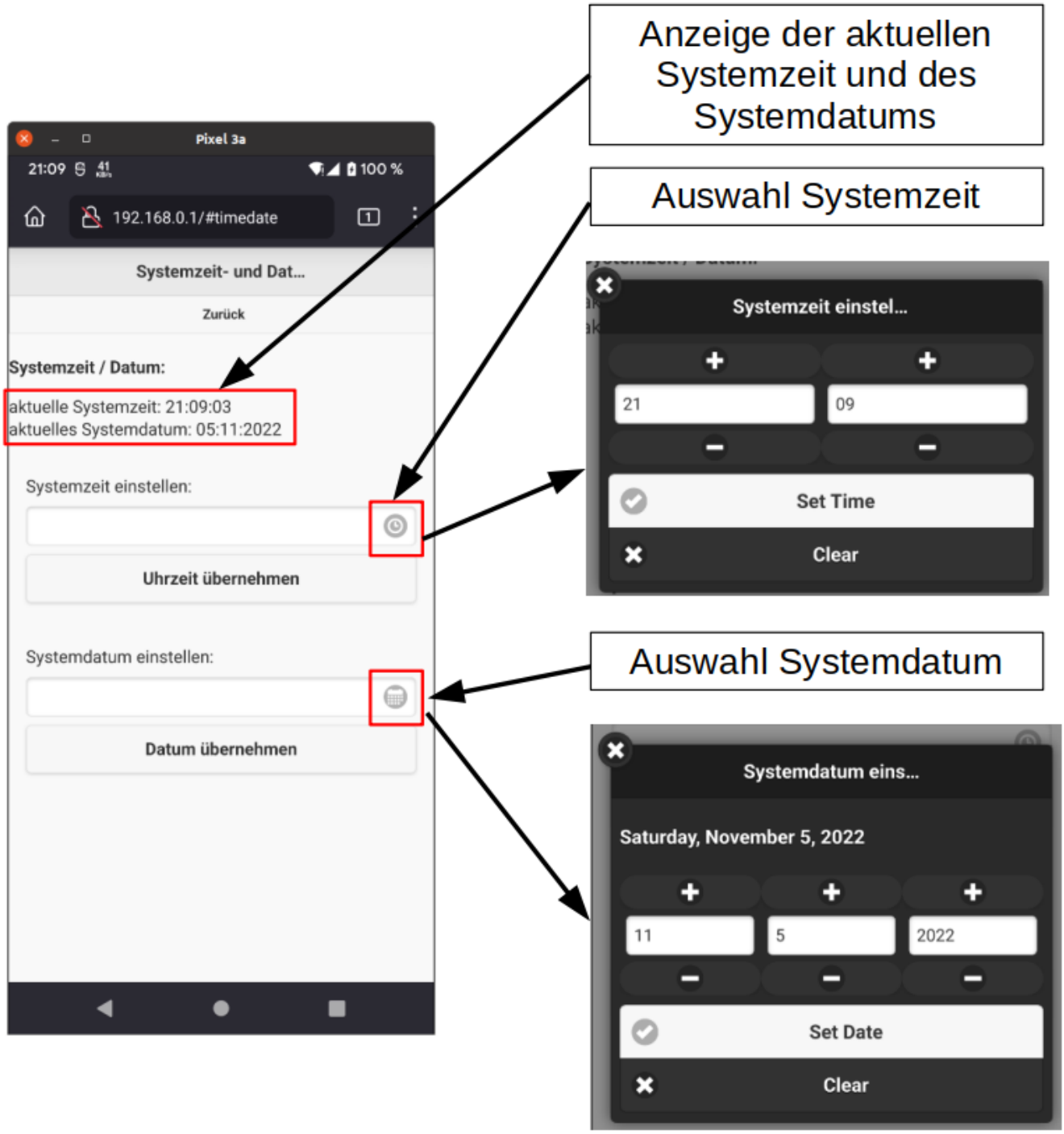


Kerzen neu verteilen:

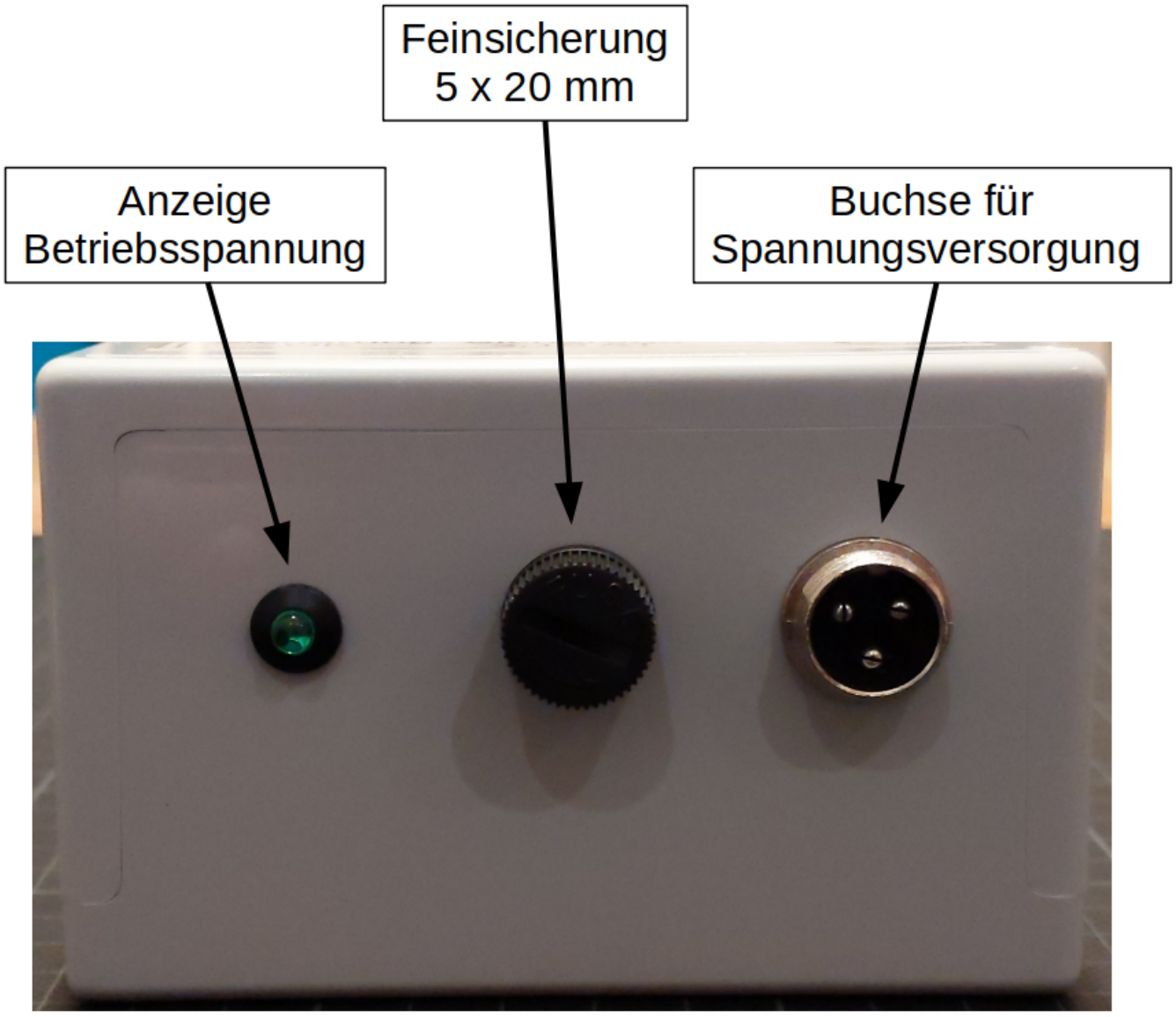


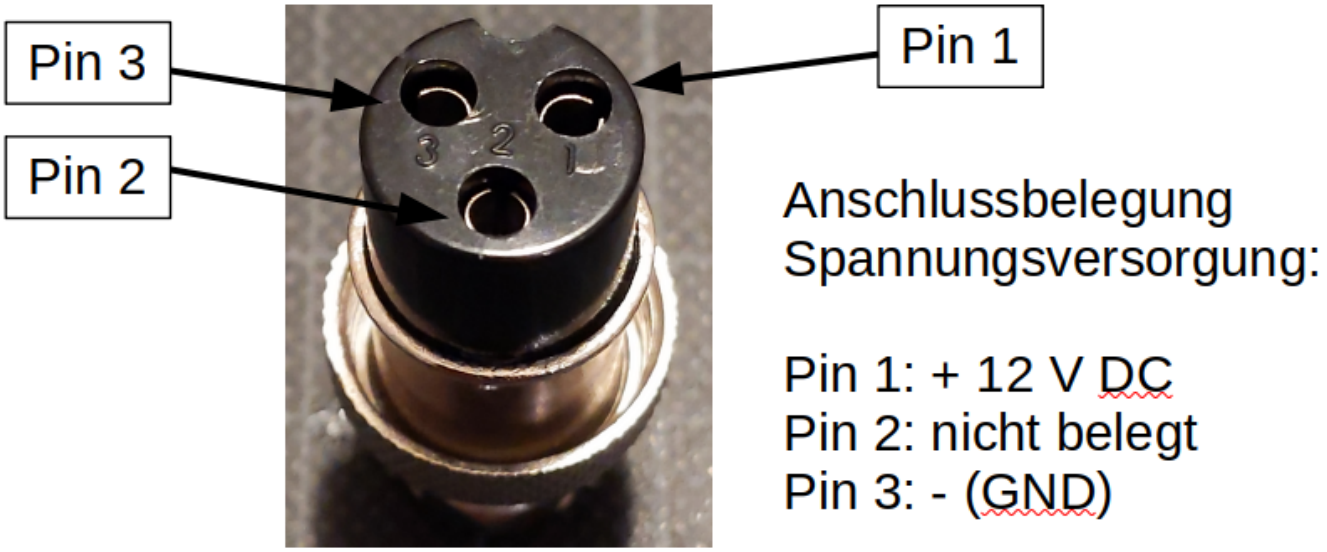
Für jeden Kalendertag wird automatisch eine neue, zufällige Auswahl der zu verwendenden Kerzen erzeugt. Diese Auswahl erfolgt jeweils zum Tageswechsel um 0:00 Uhr.
Die Auswahl der Kerzen kann auch manuell mittels „Kerzen neu verteilen“ initiiert werden. Die Aktivierung des Wartungsmodus ist dafür nicht notwendig.

Datum und Uhrzeit einstellen

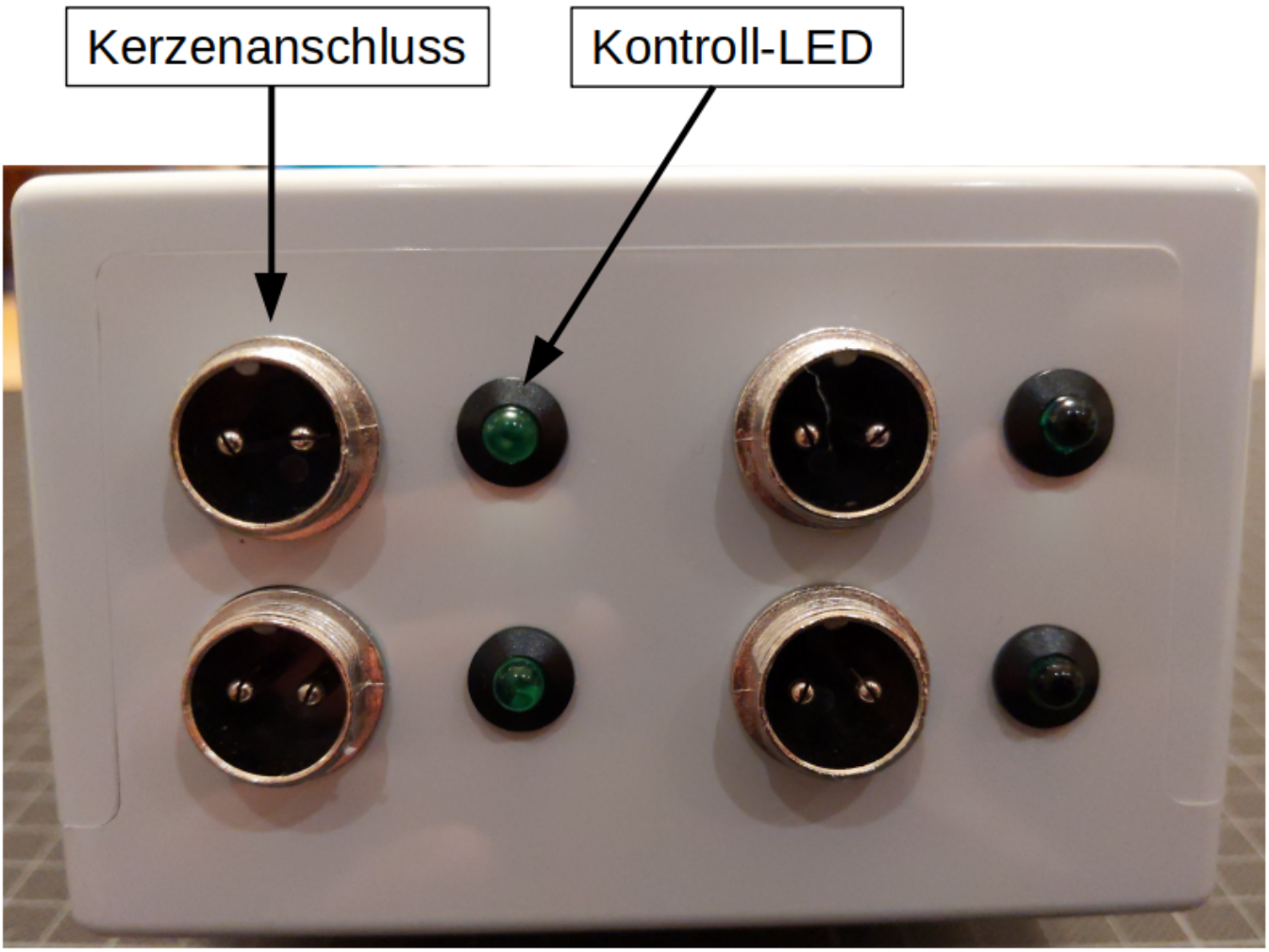


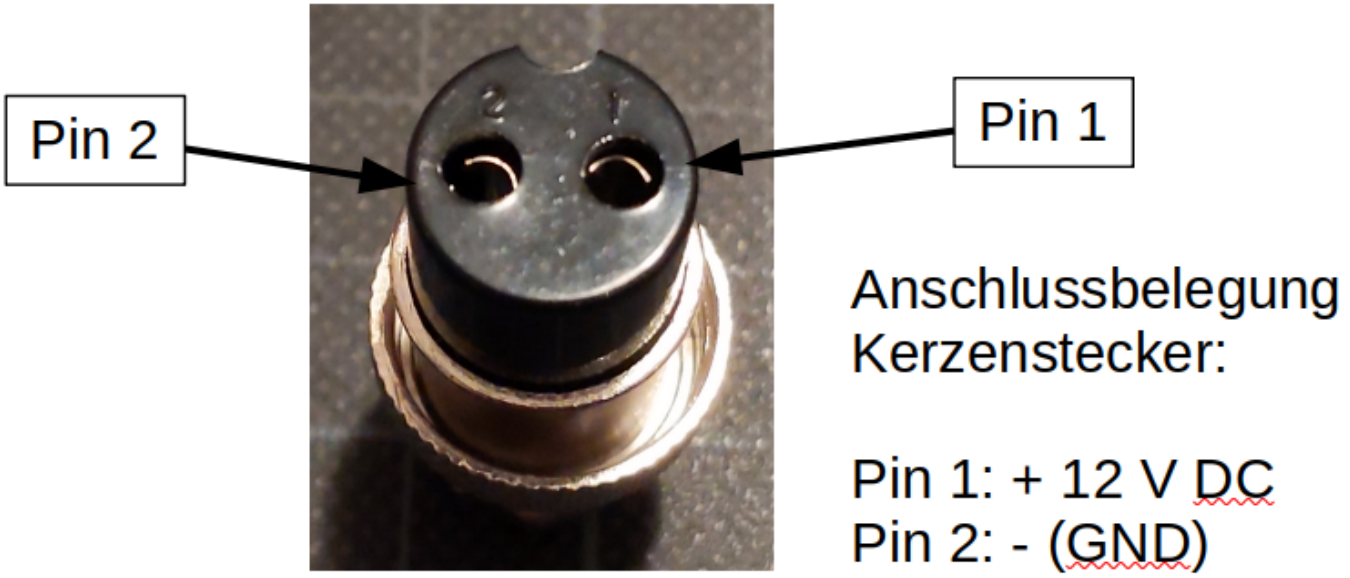
Seitenansicht - Spannungsversorgung





Seitenansicht - Kerzenanschluss





Fehlerbehebung:

Fehlerbild:	**mögl. Ursache(n):**	**Abhilfe:**
Bei angelegter Versorgungsspannung leuchtet die grüne LED neben der Sicherung nicht	Sicherung ist defekt	Sicherung mit gleichem Wert ersetzen
Es ist Advent aber die falsche Anzahl Kerzen oder gar keine Kerze leuchtet	Systemdatum falsch eingestellt	Systemdatum korrekt einstellen
Die Kerzen werden nicht um Mitternacht neu angeordnet	Systemzeit falsch eingestellt	Systemzeit korrekt einstellen
Eine oder mehrere Kerzen leuchten nicht obwohl die zugehörige grüne LED leuchtet	a) Stecker der entsprechenden Kerze nicht korrekt verschraubt	Festen und korrekten Sitz der Stecker prüfen und ggf. korrigieren.
	b) Kabelbruch in der Zuleitung zur Kerze	Zuleitung kontrollieren und reparieren / ggf. ersetzen
	c) Die Kerze ist defekt	
WLAN SSID im Mobilgerät sichtbar aber WLAN Verbindung ist nicht möglich	Das WLAN Passwort ist falsch eingegeben	Das WLAN Passwort auf korrekte Schreibweise prüfen
Beim Einschalten von einer oder mehr Kerzen erfolgt ein Neustart der Kerzensteuerung	Spannungsversorgung zu schwach, Versorgungsspannung bricht bei höherer Last zusammen	leistungsfähigere Spannungsversorgung verwenden

From:
http://xn--von-thlen-v9a.de/ - **Christophs DokuWiki**

Permanent link:
http://xn--von-thlen-v9a.de/doku.php/wiki/projekte/elektr_adventskranz_2_0/uebersicht

Last update: **2022/12/03 16:41**