ESP32 Slimme meter interface / sturing

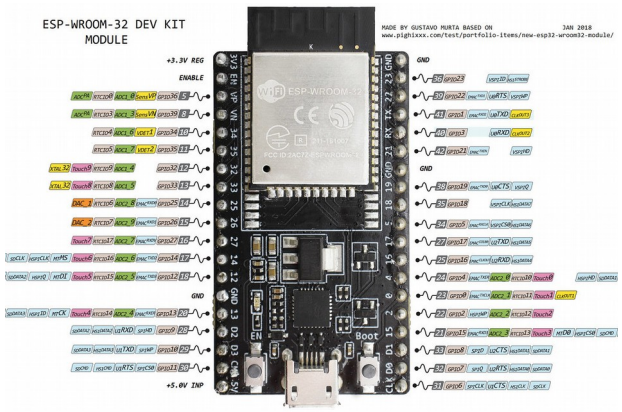
Dit project is opgebouwd uit 4 onderdelen

Interface
Deze leest de data uit de P1 poort van de slimme meter in en maakt deze beschikbaar via een webpagina op een apart netwerk. Uitlezen op een lokaal display is ook mogelijk en tot slot kunnen er nog 3 digitale uitgangen en 1 PWM uitgang gestuurd worden in functie van de opbrengst van eventueel geïnstalleerde zonnepanelen.

Display
een apart LCD scherm waarop je kan zien wat het verbruik is of bij zonnepanelen wat de injectie op het net is. Bij gebruik van de digitale of PWM module kan je ook zien wat de huidige sturing is.

Digitale uitgangsmodule
Sturen van bijvoorbeeld een wasmachine als er genoeg zonne-energie is, of bij minder goed dagen starten op een bepaald uur.

PWM module
Voor gebruik bij een elektrische warm water boiler. Stuurt het niet gebruikt opgewekt vermogen naar de boiler, dit zonder gebruik te maken van vermogen dat van het net gehaald wordt. Kan ook op tijd gestuurd worden om ook bij slechte dagen warm water te hebben.



Meer info over de slimme meter vind je hier
https://jensd.be/1205/linux/data-lezen-van-de-belgische-digitale-meter-met-de-p1-poort

https://www.netbeheernederland.nl/_upload/Files/Slimme_meter_15_a727fce1f1.pdf
https://www.fluvius.be/sites/fluvius/files/2020-02/technische-info-displays-digitale-elektriciteitsmeter.pdf

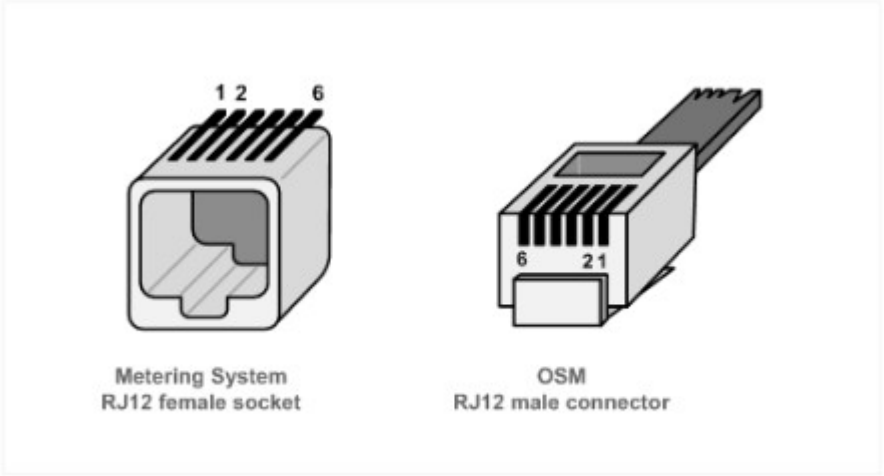
De ESP32 wordt geprogrammeerd met de Arduino IDE hoe je deze en de benodigde ESP32 software op je PC moet installeren vind je hier
https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Meer info over ESP32 en nog veel meer
https://randomnerdtutorials.com/projects-esp32/

Info over het vergroten van het bereik van de ESP32
https://peterneufeld.wordpress.com/2021/10/14/esp32-range-extender-antenna-modification

De p1 poort gratis activeren doe je via Fluvius
https://www.fluvius.be/nl/thema/meters-en-meterstanden/digitale-meter/maak-je-meter-slim#hoe-activeer-of-deactiveer-ik-de-gebruikerspoorten-van-mijn-digitale-meter/

De RJ12 connector
Foto's gecopiëerd uit
https://www.netbeheernederland.nl/_upload/Files/Slimme_meter_15_a727fce1f1.pdf



Pin #	Signal name	Description	Remark
1	+5V	+5V power supply	Power supply line
2	Data Request	Data Request	Input
3	Data GND	Data ground	
4	n.c.	Not connected	
5	Data	Data line	Output. Open collector
6	Power GND	Power ground	Power supply line

Interface module

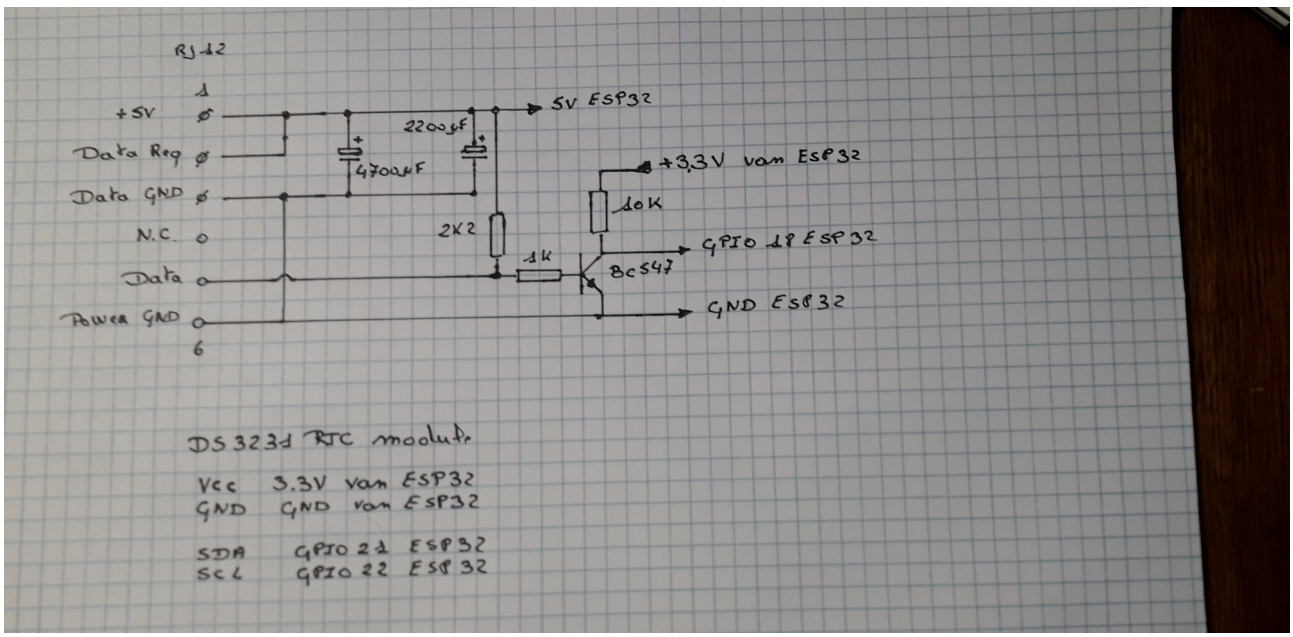
Onderdelen
1 x kabeltje met RJ12 plug
1x ESP32WROOM Devkit
1x BC547
1X DS3231 RTC module
1x R 1K
1x R 10K
1x 4700 uF / 16 Volt
1x 2200 uF / 16 Volt
(eventueel een aparte 5 Volt voeding zie beschrijving)

Wat kan je ermee doen.
Via een apart WiFi netwerk de verbruiksgegevens opvragen.
Zoeken wat het sluikverbruik is.
Zien wat elk toestel verbruikt.
Deze module is ook nodig om de verder beschreven modules te gebruiken.

Alvorens de P1 poort op de meter te kunnen gebruiken, moet deze eerst geactiveerd worden. Voor België moet dit gebeuren via Fluvius.

https://www.fluvius.be/nl/thema/meters-en-meterstanden/digitale-meter/maak-je-meter-slim#hoe-activeer-of-deactiveer-ik-de-gebruikerspoorten-van-mijn-digitale-meter/

Volgens de beschrijving kan deze P1 poort 5 Volt 250 mA leveren. Dit is niet genoeg voor de ESP32 module om data te kunnen doorsturen. 2 capaciteiten kunnen dit probleem verhelpen. Enige nadeel is dat na het inpluggen of een eventuele (zeldzame) spanningsuitval de ESP32 module manueel moet gereset worden. Een externe 5 Volt voeding gebruiken kan ook , de 2 capaciteiten zijn dan niet nodig.



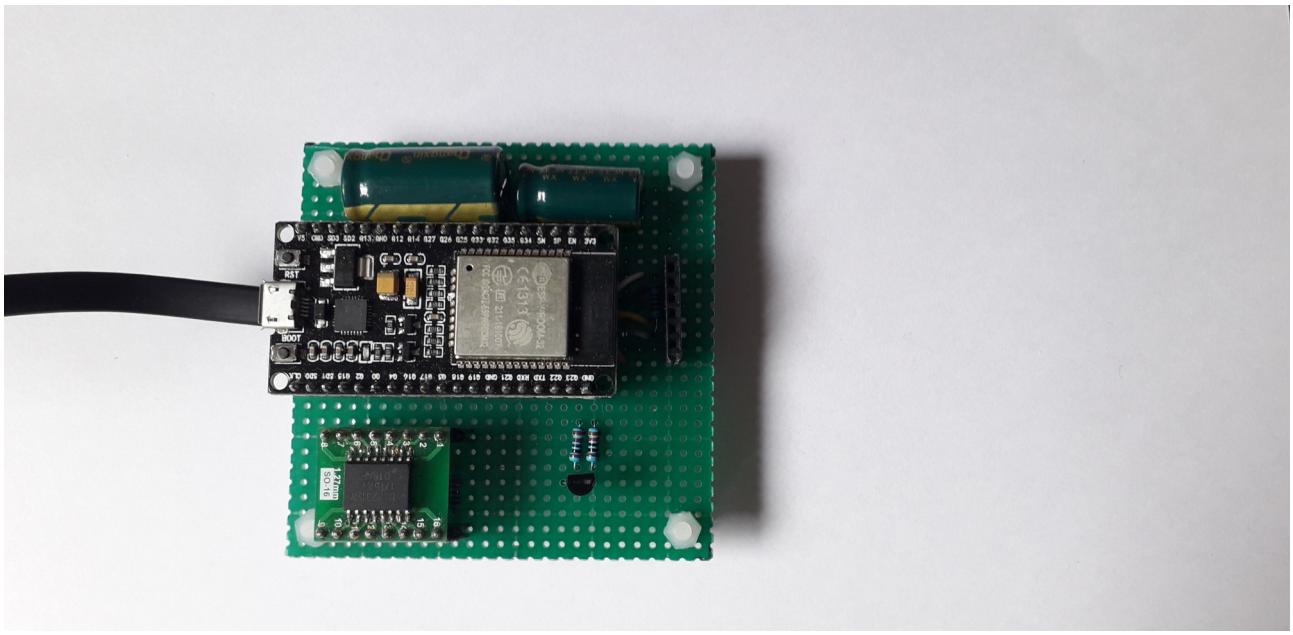
Aansluitschema kan beter, tekenen nooit goed in geweest.

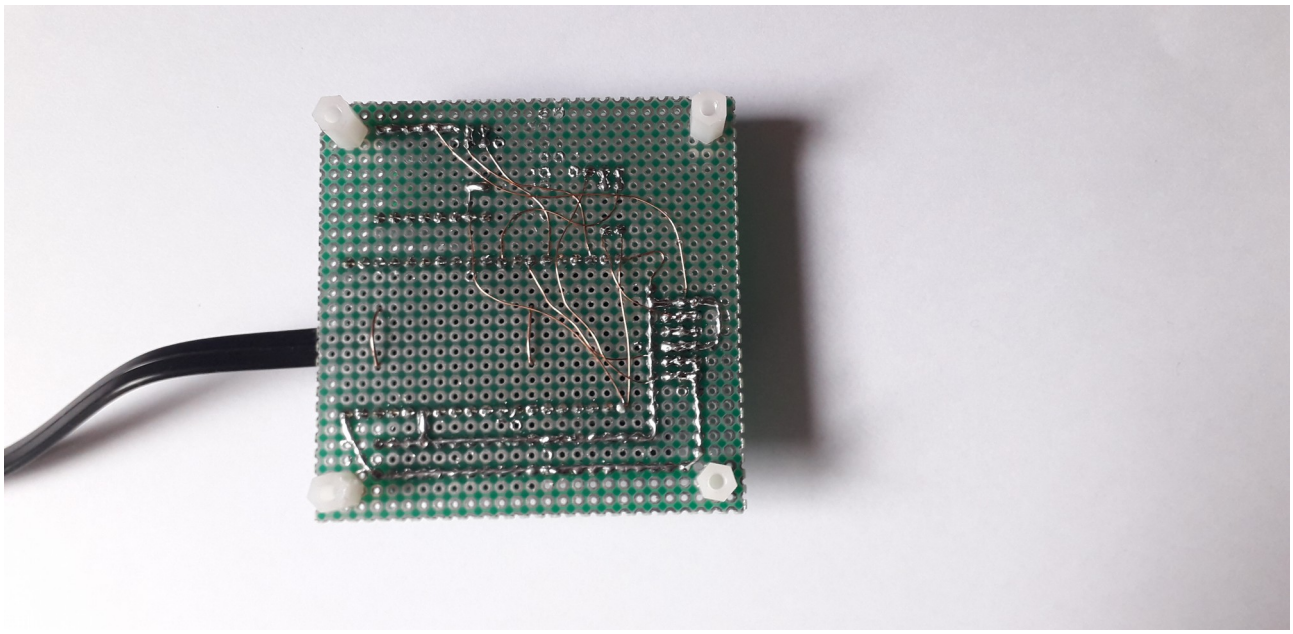
Schema misschien niet heel duidelijk

de aansluiting onder de 10K weerstand gaat naar GPIO18 van de ESP32

Opgelet de bovenkant van de 10K weerstand in de collector van de BC547 gaat naar de 3.3 Volt van de ESP32

DS3231 RTC
SDA naar GPIO21
SCL naar GPIO22
GND naar GND ESP32
VCC naar 3.3V ESP32



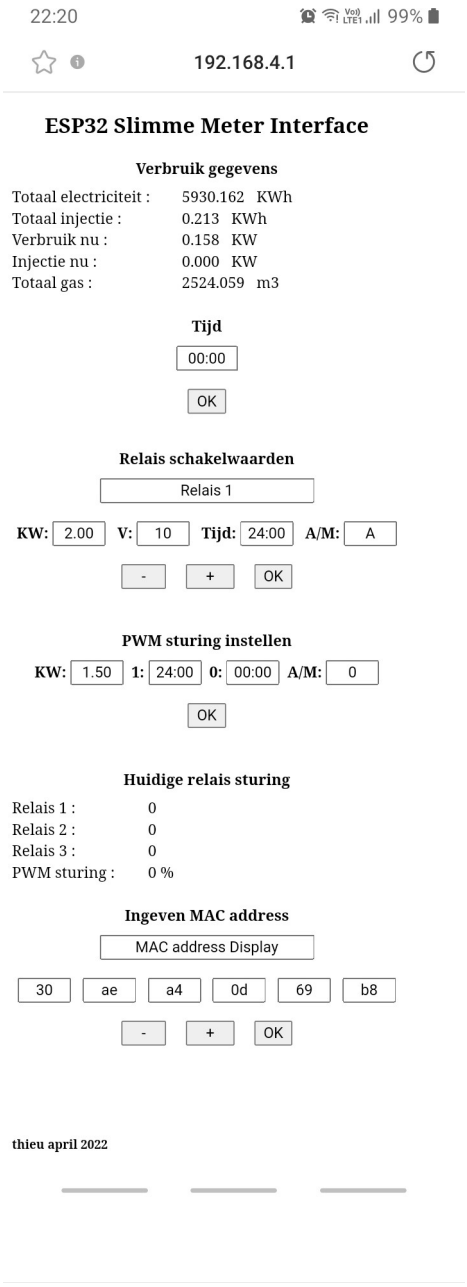


Voor de aandachtige lezers. Ik had geen DS3231 RTC module meer en heb deze vervangen door een DS3231 IC. Als je dit doet vergeet dan de 2x 10K pull-up weerstanden van SDA / SCL naar de ***3.3 Volt van de ESP32*** module niet.
Een ESP32WROOM Devkit module heeft meerdere GND aansluitingen, voor een goede werking is het noodzakelijk om deze allemaal aan te sluiten.
Laad nu het programma **Slimme_meter_esp32_klok_v3.ino** (vanaf bladzijde 16) in de ESP32WROOM devkit met behulp van de Arduino IDE.
Als de P1 poort geactiveerd is, de ESP32 ingeplugd en **manueel gereset** is maak dan met je smartphone verbinding met

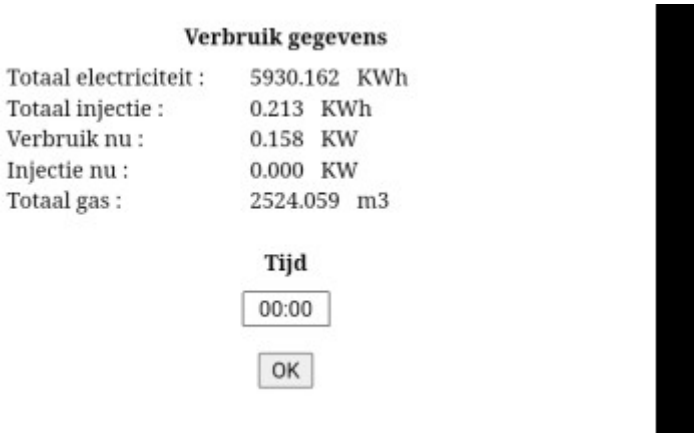
Wifi netwerk	ESP32Energie
Paswoord	ESP32pswd

webadres 192.168.4.1

de volgende webpagina zou dan moeten verschijnen



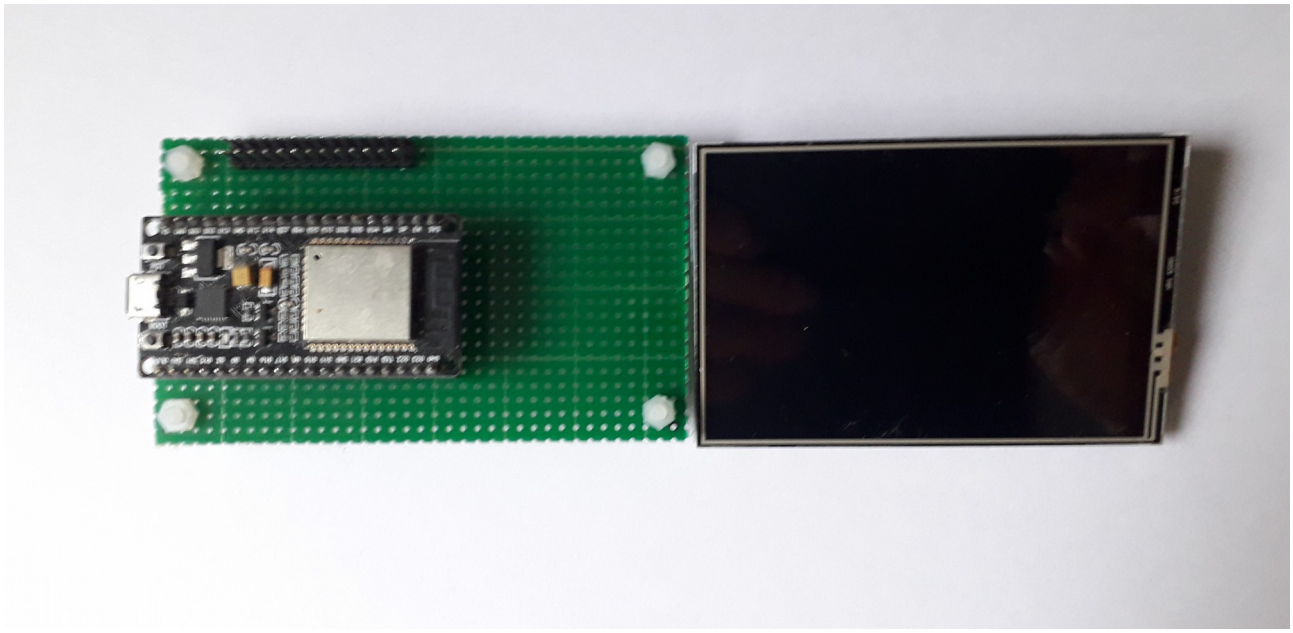
Het gedeelte dat we nu kunnen gebruiken is

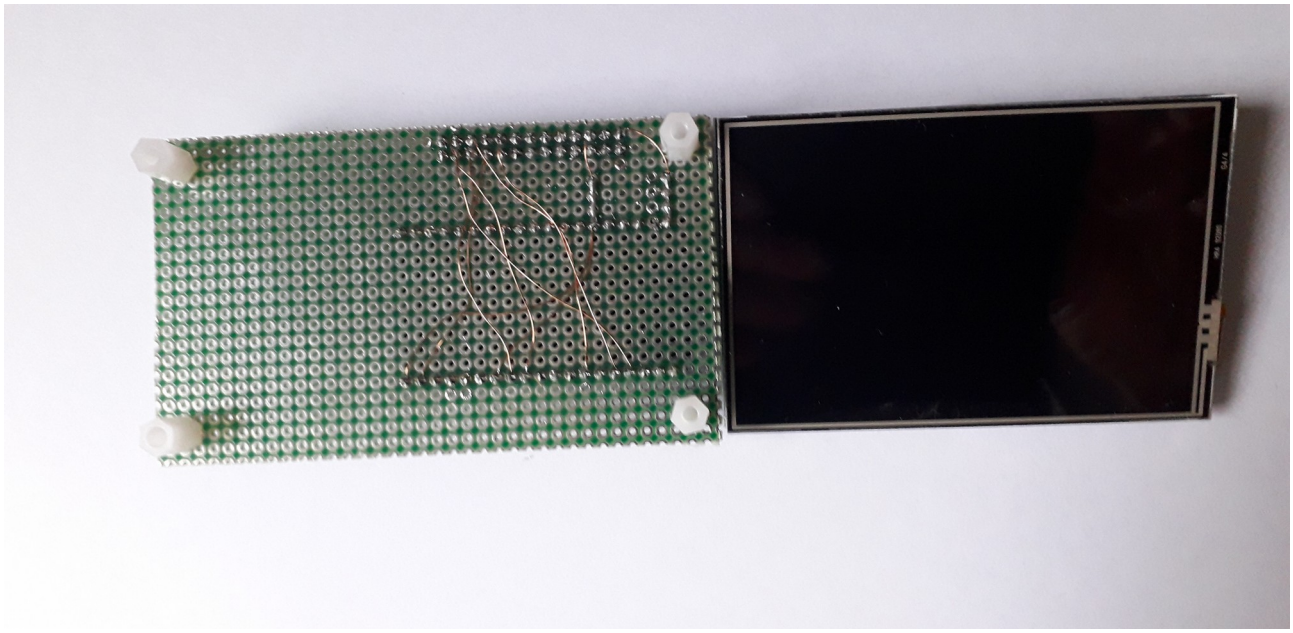


Hierop kun je de verschillende meterstanden raadplegen en als je ook een digitale gasmeter hebt kan je deze ook aflezen.
Je kan er ook het sluipverbruik mee opsporen en dat is meer als men zou verwachten heb ik gemerkt.
Onder Tijd is er ook een klok, deze wordt bijgehouden door de DS3231 RTC module, als je alleen de interface gebruikt verder niet heel nuttig. Invullen doe je door op het veld te gaan staan, alles te wissen en dan in te vullen volgens het formaat ***uu:mm*** en daarna op OK te drukken.

Display module

Onderdelen
1x ESP32WROOM Devkit
1x 3.5” Raspberry LCD Display.
1x externe voeding 5V 1A (oude gsm lader voldoet ook)





Discription	Connected Pin	Silk	Pin		Silk	Connected Pin	Discription
VCC	5V	5V	2	1	3v3		
		5V	4	3	SDA		
GND	GND	GND	6	5	SCL		
		TX	8	7	P7		
		RX	10	9	GND		
		P1	12	11	P0	TP-IRQ	Interrupt of the touch panel. If the touch panel is tapped, it's low level.
		GND	14	13	P2		
		P4	16	15	P3		
Choose the command/data register (Register Select)	LCD-RS	P5	18	17	3V3		
		GND	20	19	MO	LCD-SI/TP_SI	LCD display/ SPI data input of the touch panel
Reset	RST	P6	22	21	MI	TP_SO	SPI data output of the touch panel
chip select signal; select LCD when it's low level.	LCD_CS LCD	CE0	24	23	SCK	LCD_SCK/TP_SCK	LCD display/ SPI clock signal of the touch panel
Touch panel chip select signal; select touch panel when it's low level.	TP_CS	CE1	26	25	GND		

Bovenstaande afbeelding gevonden op internet, met dank aan de maker.

Hoe aansluiten

Lcd display

Pin 2	5V	5V van externe voeding
Pin 6	GND	GND van externe voeding
Pin18	C/D	GPIO2
Pin19	MOSI	GPIO23
Pin21	MISO	GPIO19
Pin22	RESET	GPIO4
Pin23	SCK	GPIO18
Pin24	CE0	GPIO15

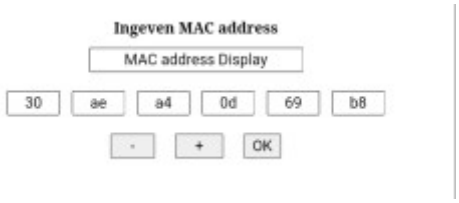
Overige ESP32 ESP32WROOM Devkit aansluitingen

ESP32 5V	naar externe 5V
ESP32 GND	naar externe GND (alle GND's aansluiten)

Om de display module te kunnen gebruiken moeten we eerst het MAC adres van de hiervoor gebruikte ESP32 kennen.
Open de Arduino IDE en open de monitor op 115200 baud.
Programmeer ESP32WROOM Devkit met het programma
MAC_adres_esp32.ino (bladzijde 38)
Het MAC adres verschijnt in de terminal.

Maak verbinding met :

Wifi netwerk	ESP32Energie
Paswoord	ESP32pswd
webadres 192.168.4.1	



Dit gedeelte van de webpagina heb je nu nodig. Met behulp van de + en – toetsen kan je kiezen welk MAC adres je wil ingeven. In dit geval dat van het display. Vul het adres in in de zes voorziene vakjes en druk OK.

Opgelet 2 dezelfde MAC adressen invullen voor verschillende modules gaat niet. Indien je het in te geven adres al hebt ingevuld voor een andere module verander dit eerst.

Heel belangrijk na het veranderen van *een* MAC adres telkens de interface module eerst resetten alvorens een ander MAC adres aan te passen.

In de library TFT_eSPI pas het bestand
User_Setup_Select.h
aan zoals hieronder.
Er mag maar 1 #include gekozen zijn.

User_Setup_Select.h

```
// This header file contains a list of user setup files and defines which one the
// compiler uses when the IDE performs a Verify/Compile or Upload.
//
// Users can create configurations for different Espressif boards and TFT displays.
// This makes selecting between hardware setups easy by "uncommenting" one line.

// The advantage of this hardware configuration method is that the examples provided
// with the library should work with different setups immediately without any other
// changes being needed. It also improves the portability of users sketches to other
// hardware configurations and compatible libraries.
//
// Create a shortcut to this file on your desktop to permit quick access for editing.
// Re-compile and upload after making and saving any changes to this file.

// Customised User_Setup files are stored in the "User_Setups" folder.

#ifndef USER_SETUP_LOADED //  Lets PlatformIO users define settings in
                          //  platformio.ini, see notes in "Tools" folder.

// Only ONE line below should be uncommented.  Add extra lines and files as needed.

//#include <User_Setup.h>           // Default setup is root library folder

//#include <User_Setups/Setup1_ILI9341.h>  // Setup file configured for my ILI9341
//#include <User_Setups/Setup2_ST7735.h>   // Setup file configured for my ST7735
//#include <User_Setups/Setup3_ILI9163.h>  // Setup file configured for my ILI9163
//#include <User_Setups/Setup4_S6D02A1.h>  // Setup file configured for my S6D02A1
//#include <User_Setups/Setup5_RPi_ILI9486.h>        // Setup file configured for my stock RPi TFT
//#include <User_Setups/Setup6_RPi_Wr_ILI9486.h>     // Setup file configured for my modified RPi TFT
//#include <User_Setups/Setup7_ST7735_128x128.h>     // Setup file configured for my ST7735 128x128 display
//#include <User_Setups/Setup8_ILI9163_128x128.h>    // Setup file configured for my ILI9163 128x128 display
//#include <User_Setups/Setup9_ST7735_Overlap.h>     // Setup file configured for my ST7735
//#include <User_Setups/Setup10_RPi_touch_ILI9486.h> // Setup file configured for ESP8266 and RPi TFT with touch

#include <User_Setups/Setup11_RPi_touch_ILI9486.h> // Setup file configured for ESP32 and RPi TFT with touch
//#include <User_Setups/Setup12_M5Stack.h>           // Setup file for the ESP32 based M5Stack
//#include <User_Setups/Setup13_ILI9481_Parallel.h>  // Setup file for the ESP32 with parallel bus TFT
//#include <User_Setups/Setup14_ILI9341_Parallel.h>  // Setup file for the ESP32 with parallel bus TFT
//#include <User_Setups/Setup15_HX8357D.h>           // Setup file configured for HX8357D (untested)
//#include <User_Setups/Setup16_ILI9488_Parallel.h>  // Setup file for the ESP32 with parallel bus TFT
```

Laad het programma **slimme_meter_esp32_display.ino** (vanaf bladzijde 39) met behulp van de Arduino IDE in de ESP32WROOM Devkit module.
De interface stuurt elke 10 seconden data naar de display. Na enkele seconden zou je dit moeten krijgen.

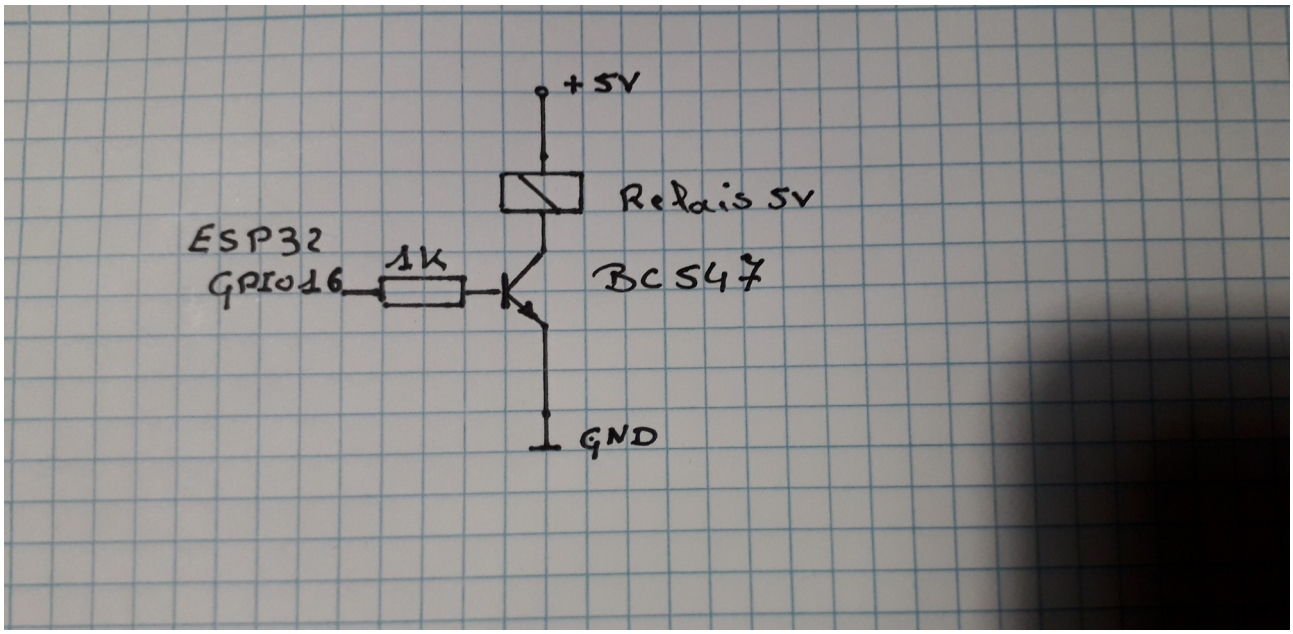
Display bij injectie, het momenteel geïnjecteerd vermogen in het groen.

Display bij afname van het net vermogen in het rood.

De display toont tevens welk relais er is uitgestuurd of in geval van de PWM sturing het percentage van de uitsturing.

Digitale uitgangsmodule

Onderdelen
1x ESP32WROOM Devkit
1x R 1K
1x BC547
1x 5V stuurrelais geschikt voor het bedienen van een vermogen relais voor het aansturen van de huishoudtoestellen. Opgelet met de stroom die een BC547 kan leveren.
1x externe voeding 5V



Hoe aansluiten
ESP32
ESP32 5V naar externe 5V
ESP32 GND naar externe GND (alle GND's aansluiten)

GPIO16 naar 1K weerstand in basis BC547

Om een digitale uitgangsmodule te kunnen gebruiken moeten we eerst het MAC adres van de hiervoor gebruikte ESP32 kennen.
Open de Arduino IDE en open de monitor op 115200 baud.
Programmeer ESP32 ESP32WROOM Devkit met het programma
MAC_adres_esp32.ino (bladzijde 38)
Het MAC adres verschijnt in de terminal.

Maak verbinding met :
Wifi netwerk ESP32Energie
Paswoord ESP32pswd
webadres 192.168.4.1

Dit gedeelte van de webpagina heb je nu nodig. Met behulp van de + en – toetsen kan je kiezen welk MAC adres je wil ingeven. In dit geval dat van een relais module 1, 2 of 3. Vul het adres in in de zes voorziene vakjes en druk OK.
Opgelet 2 dezelfde MAC adressen invullen voor verschillende modules gaat niet. Indien je het in te geven adres al hebt ingevuld voor een andere module verander dit eerst.

Heel belangrijk na het veranderen van *een* MAC adres telkens de interface module eerst resetten alvorens een ander MAC adres aan te passen.

Laad het programma **esp32_relais.ino** (bladzijde 41) met behulp van de Arduino IDE in de ESP32WROOM Devkit module.

Om de 3 x relais te kunnen instellen maken we gebruik van het volgende gedeelte van de webpagina

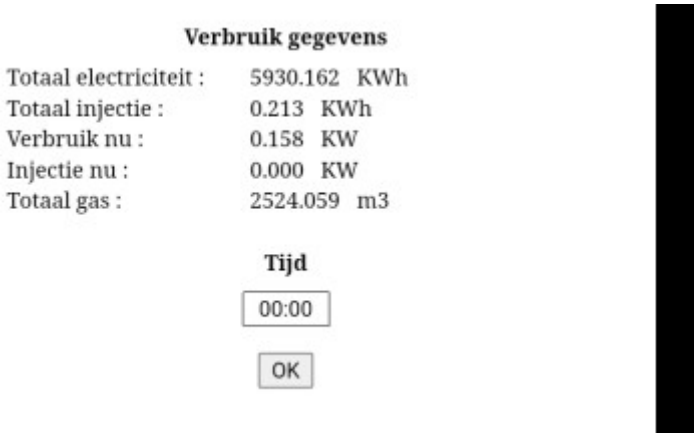
Met behulp van de + / – toetsen kies welke relaismodule je wilt instellen en druk op OK
Nu zijn er vershillende velden die je kan instellen.
Het gemakkelijkste eerst
Automatisch / **M**anueel
mogelijkheden

0	relais uitgeschakeld
1	relais altijd gestuurd
A	relais wordt gestuurd in functie van de vorige velden.

KW	vul hier het vermogen in dat geïnjecteerd wordt op het net alvorens de relais mag opkomen
V	vertraging in minuten na het bereiken van in KW ingevuld vermogen alvorens relais mag opkomen. Niet echt van belang voor relais 1, maar de uitgangen die op automatisch staan schakelen in volgorde eerst 1 dan 2 ...Om te vermijden dat een volgende ingang al schakelt alvorens een vorige zijn maximum op te nemen vermogen bereikt heeft deze vertraging. Bvb een wasmachine begint niet onmiddellijk te verwarmen.
Tijd	Als back-up voor slechte weer dagen kan je de uitgang ook op tijd laten schakelen. Je moet dan wel eerst de tijd instellen. Als je niet wil dat machine op tijd start vul dan als uur 24 in.

De relais valt **NIET** automatisch af. Afzetten doe je door **A/M : 0**

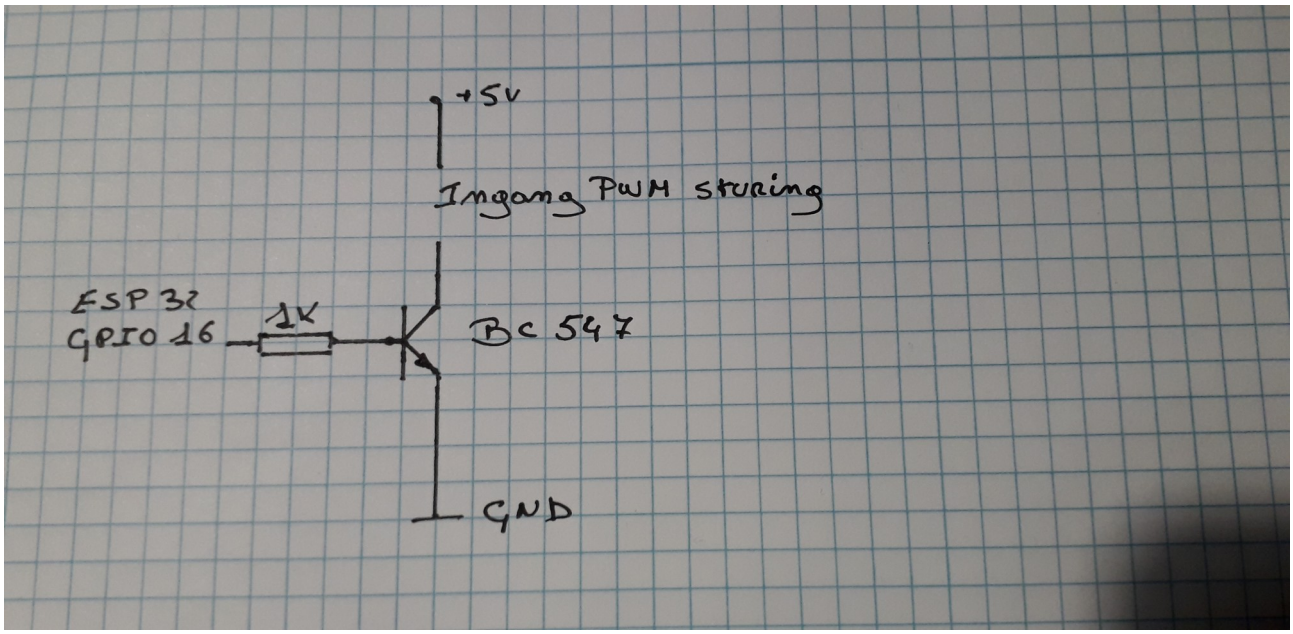
In het vak Tijd vul daar de tijd in 24 uurs formaat **uu:mm** en druk OK



Voor huishoudtoestellen geldt over het algemeen de regel dat zij na een spanningsuitval hun programma verder zetten. Dit kunnen we gebruiken om de machines automatisch te starten.
Voorbeeld wasmachine
Stuur uitgang van relais die wasmachine bedient via **A/M : 1 .** Zet wasmachine klaar en druk op start machine. Stuur na een korte tijd relais via **A/M : 0** en dan **A/M : A.** Wasmachine zal dan starten na de voorwaarden ingevuld in de verschillend vakjes.

PWM uitgangsmodule

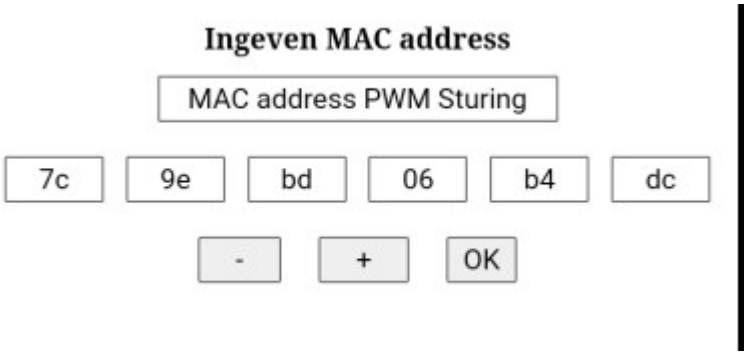
Onderdelen
1x ESP32WROOM Devkit
1x R 1K
1x BC547
1x solid state relais welke kan gestuurd worden met 5V



Hoe aansluiten
ESP32 ESP32WROOM Devkit
ESP32 5V naar externe 5V
ESP32 GND naar externe GND (alle GND's aansluiten)

GPIO16 naar 1K weerstand in basis BC547
Ingang solid state relais tussen collector BC547 en +5V

Om de PWM uitgangsmodule te kunnen gebruiken moeten we eerst het MAC adres van de hiervoor gebruikte ESP32 ESP32WROOM Devkit module kennen.
Open de Arduino IDE en open de monitor op 115200 baud.
Programmeer ESP32WROOM Devkit met het programma
MAC_adres_esp32.ino (bladzijde 38)
Het MAC adres verschijnt in de terminal.
Maak verbinding met :
Wifi netwerk ESP32Energie
Paswoord ESP32pswd
webadres 192.168.4.1



Dit gedeelte van de webpagina heb je nu nodig. Met behulp van de + en – toetsen kan je kiezen welk MAC adres je wil ingeven. In dit geval dat van MAC address PWM sturing. Vul het adres in in de zes voorziene vakjes en druk OK.
Opgelet 2 dezelfde MAC adressen invullen voor verschillende modules gaat niet. Indien je het in te geven adres al hebt ingevuld voor een andere module verander dit eerst.

Heel belangrijk na het veranderen van *een(1)* MAC adres telkens de interface module eerst resetten alvorens een ander MAC adres aan te passen.

Laad het programma **esp32_pwm.ino** (bladzijde 42) met behulp van de Arduino IDE in de ESP32WROOM Devkit module.
Om de PWM module te kunnen gebruiken hebben we dit gedeelte van de webpagina nodig.

Automatisch / **M**anueel
0 PWM wordt niet uitgestuurd
1 PWM maximum uitgestuurd
A PWM wordt gestuurd in functie van het huidig geïnjecteerd vermogen. Voor het bereiken van het vermogen van een digitale uitgang wordt er met het vermogen opgenomen door de PWM sturing geen rekening gehouden omdat deze dan toch automatisch terugstuurt naar het dan nog beschikbare vermogen.

KW vul hier het vermogen in van bvb de boiler die met deze uitgang verbonden is. Dit om te weten hoeveel procent er kan uitgestuurd worden om het beschikbare vermogen te gebruiken.
Gebruik deze uitgang zeker niet bij toestellen met een motor / ventilator.
1 Als je ook tijd gestuurd wil schakelen, vul hier de inschakeltijd in. Indien niet gewenst vul als uur 24 in.
0 uitschakeltijd.
Bij tijd sturing wordt de uitgang maximaal uitgestuurd.

Dat was het
gr,
thieu

Programma's zie de volgende bladzijden

Slimme_meter_esp32_klok_V3.ino

```
#include <WiFi.h>
#include <Preferences.h>
#include <esp_now.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Wire.h>

esp_now_peer_info_t   peerInfo;
esp_err_t             result;

AsyncWebServer server(80);

Preferences pref;

#define RXD2          18
#define TXD2          19
#define DS3231SN      0x68

typedef struct meter_data{
  float kwh_totaal;
  float injectie_totaal;
  float injectie_nu;
  float verbruik_nu;
  float gas_totaal;
  bool  relais1;
  bool  relais2;
  bool  relais3;
  int   pwm_sturing;
}meter_data;
meter_data    ingelezen;


typedef struct relais_data{
  bool relais;
}relais_data;
relais_data   uitsturen;


typedef struct pwm_data{
  int procent;
}
pwm_data;
pwm_data    pwm_sturing;


String buffer_data =     "                                                  ";
String kwh_dag = "          ";
String kwh_nacht = "          ";
String injectie_dag = "          ";
String injectie_nacht = "          ";
String kw_nu = "          ";
String injectie_nu = "          ";
String gas = "          ";

int module_teller = 0;
int relais_module_teller = 0;

unsigned long nu;

float kwh_dag_float;
float kwh_nacht_float;
float kwh_totaal_float;
float injectie_dag_float;
float injectie_nacht_float;
float injectie_totaal_float;
float kw_nu_float;
float injectie_nu_float;
float verbruik_nu_float;
float gas_totaal_float;
float verbruik_pwm_float;

char kwh_totaal_float_char[12];
char injectie_totaal_float_char[12];
char kw_nu_float_char[12];
```

```
char injectie_nu_float_char[12];
char gas_totaal_float_char[12];

char broadcastAddressX_0_char[8];
char broadcastAddressX_1_char[8];
char broadcastAddressX_2_char[8];
char broadcastAddressX_3_char[8];
char broadcastAddressX_4_char[8];
char broadcastAddressX_5_char[8];

String broadcastAddress1_string = "                    ";
String broadcastAddress2_string = "                    ";
String broadcastAddress3_string = "                    ";
String broadcastAddress4_string = "                    ";
String broadcastAddress5_string = "                    ";

char module_char[20];

const char* INPUT_MACX_0 = "input_macx_0";
const char* INPUT_MACX_1 = "input_macx_1";
const char* INPUT_MACX_2 = "input_macx_2";
const char* INPUT_MACX_3 = "input_macx_3";
const char* INPUT_MACX_4 = "input_macx_4";
const char* INPUT_MACX_5 = "input_macx_5";

const char* MODULE_MIN = "module_min";
const char* MODULE_PLUS = "module_plus";
const char* MODULE_BEVESTIG = "module_bevestig";

uint8_t broadcastAddress1[6];
uint8_t broadcastAddress2[6];
uint8_t broadcastAddress3[6];
uint8_t broadcastAddress4[6];
uint8_t broadcastAddress5[6];

uint8_t input_macx_0;
uint8_t input_macx_1;
uint8_t input_macx_2;
uint8_t input_macx_3;
uint8_t input_macx_4;
uint8_t input_macx_5;

const char* INPUT_TIJD = "input_tijd";
const char* TIJD_BEVESTIG = "tijd_bevestig";

const char* INPUT_KW_ON = "input_kw_on";
const char* INPUT_OVERRIDE = "input_override";
const char* INPUT_DELAY = "input_delay";
const char* INPUT_SCHAKEL_TIJD = "input_schakel_tijd";
const char* RELAIS_MODULE_MIN = "relais_module_min";
const char* RELAIS_MODULE_PLUS = "relais_module_plus";
const char* RELAIS_MODULE_BEVESTIG = "relais_module_bevestig";

const char* INPUT_PWM_KW = "input_pwm_kw";
const char* INPUT_PWM_TIJD_ON = "input_pwm_tijd_on";
const char* INPUT_PWM_TIJD_OFF = "input_pwm_tijd_off";
const char* INPUT_PWM_OVERRIDE = "input_pwm_override";
const char* BEVESTIG_PWM = "bevestig_pwm";

bool relais1_uit;
bool relais2_uit;
bool relais3_uit;
bool vijf_seconden = false;
int relais1_delay;
int relais2_delay;
int relais3_delay;
int uren_on1_int;
int uren_on2_int;
int uren_on3_int;
int uren_on4_int;
int uren_off4_int;
int minuten_on1_int;
int minuten_on2_int;
int minuten_on3_int;
int minuten_on4_int;
int minuten_off4_int;
float relais1_on;
float relais2_on;
float relais3_on;
```

```
float pwm_kw_float;
String relais1_override;
String relais2_override;
String relais3_override;
char relais_module_char[20];
char kw_on_char[12];
char override_char[8];
char schakel_delay_char[12];
char pwm_tijd_on_char[8];
char pwm_tijd_off_char[8];
char relais1_sturing_char[12];
char relais2_sturing_char[12];
char relais3_sturing_char[12];
unsigned long relais1_vertraging_long;
unsigned long relais2_vertraging_long;
unsigned long relais3_vertraging_long;

int uren;
int minuten;
int seconden;
String tijd_string = "         ";
char tijd_char[12];

bool pwm_tijd_gezet = false;
bool pwm_tijd_gezet_vorig;
int uitsturing_pwm_int = 0;
float uitsturing_pwm_float = 0.0;
String pwm_override;
char uitsturing_pwm_char[6];
char schakel_tijd_char[12];
char pwm_override_char[8];

const char* APssid = "ESP32Energie";
const char* APpswd = "ESP32pswd";

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  char macStr[18];
  Serial.print("Packet to: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  Serial.print(macStr);
  Serial.print(" send status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

byte decToBcd(byte val)
{
return ( (val/10*16) + (val%10) );
}

byte bcdToDec(byte val)
{
return ( (val/16*10) + (val%16) );
}

const char energie_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
  <head>
    <iframe style="display:none" name="hidden-form"></iframe>
    <title>Energie Beheer</title>
    <style>
       div.kader {
         position: relative;
         width: 400px;
         height: 12x;
       }
         div.links{
         position: absolute;
         left : 0px;
         width; 100px;
         height: 12px;
       }
       div.links_midden{
         position:absolute;
         left:  120px;
         width: 100px
         height: 12px;
       }
```

```
    div.midden{
      position:absolute;
      left: 150px;
      width: 100px
      height: 12px;
    }
    div.titel{
      height: 25px;
      width: auto;
    }
    div.bottom{
      position: fixed;
      bottom: 0px;
    }
  </style>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <h3><center> ESP32 Slimme Meter Interface </center></h3>
  <small>
  <div class="titel"><center><b>Verbruik gegevens</b></center></div>
  <div class="kader">
    <div class="links">Totaal electriciteit : </div>
    <div class="midden">%electriciteit_totaal%   KWh</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">Totaal injectie : </div>
    <div class="midden">%injectie_totaal%   KWh</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">Verbruik nu : </div>
    <div class="midden">%kw_nu%   KW</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">Injectie nu : </div>
    <div class="midden">%injectie_nu%   KW</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">Totaal gas : </div>
    <div class="midden">%gas_totaal%   m3</div>
  </div>
  <br><br>
  <form action="/get" target="hidden-form">
  <div class="titel"><b><center>Tijd</center></b></div>
  <center><input type="text" style="text-align:center;" value="%tijd%" name="input_tijd" size=2></center>
  <br>
  <center><input type="submit" name="tijd_bevestig" value="OK" onclick="ok()"></center>
  <br>
  </form>
  <form action="/get" target="hidden-form">
  <br>
  <div class="titel"><b><center>Relais schakelwaarden</center></b></div>
  <center><input type= "text" style="text-align:center;" value="%relais_module%" size = 20></center>
  <br>
  <center>
  <b>KW:</b> <input type="text" style="text-align:center;" value="%kw_on%" name="input_kw_on" size=1>
    <b>V:</b> <input type="text" style="text-align:center;" value="%delay%" name="input_delay" size=1>
    <b>Tijd:</b> <input type="text" style="text-align:center;" value="%schakel_tijd%" name="input_schakel_tijd" size=1>
    <b>A/M:</b> <input type="text" style="text-align:center;" value="%override%" name="input_override" size=1>
  </center>
  <br>
  <center>
    <input type="submit" name="relais_module_min" value="   -   " onclick="ok()">
       
    <input type="submit" name="relais_module_plus" value="   +   " onclick="ok()">
       
    <input type="submit" name="relais_module_bevestig" value="OK" onclick="ok()">
  </center>
  </form>
  <br><br>
  <form action="/get" target="hidden-form">
  <div class="titel"><center><b>PWM sturing instellen</b></center></div>
  <center>
  <b>KW:</b> <input type="text" style="text-align:center;" value="%pwm_kw%" name="input_pwm_kw" size=1>
   <b>1:</b> <input type="text" style="text-align:center;" value="%pwm_tijd_on%" name="input_pwm_tijd_on" size=1>
```

```
   <b>0:</b> <input type="text" style="text-align:center;" value="%pwm_tijd_off%" name="input_pwm_tijd_off" size=1>
   <b>A/M:</b> <input type="text" style="text-align:center;" value="%pwm_override%" name="input_pwm_override" size=1>
  </center>
  <br>
  <center><input type="submit" name="bevestig_pwm" value="OK" onclick="ok()"></center>
  </form>
  <br><br>
  <div class="titel"><center><b>Huidige relais sturing</b></center></div>
  <div class="kader">
    <div class="links">Relais 1 : </div>
    <div class="links_midden">%relais1_sturing%</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">Relais 2 : </div>
    <div class="links_midden">%relais2_sturing%</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">Relais 3 : </div>
    <div class="links_midden">%relais3_sturing%</div>
  </div>
  <br>
  <div class="kader">
    <div class="links">PWM sturing : </div>
    <div class="links_midden">%procent% &#37 </div>
  </div>
  <br><br>
  <form action="/get" target="hidden-form">
  <div class="titel"><center><b>Ingeven MAC address</b></center></div>
  <center>
  <input type= "text" style="text-align:center;" value="%module%" size = 20>
  </center>
  <br>
  <center>
  <input type="text" style="text-align:center;" value="%display_macx_0%" name="input_macx_0" size=1>
   
  <input type="text" style="text-align:center;" value="%display_macx_1%" name="input_macx_1" size=1>
   
  <input type="text" style="text-align:center;" value="%display_macx_2%" name="input_macx_2" size=1>
   
  <input type="text" style="text-align:center;" value="%display_macx_3%" name="input_macx_3" size=1>
   
  <input type="text" style="text-align:center;" value="%display_macx_4%" name="input_macx_4" size=1>
   
  <input type="text" style="text-align:center;" value="%display_macx_5%" name="input_macx_5" size=1>
  </center>
  <br>
  <center>
    <input type="submit" name="module_min" value="   -   " onclick="ok()">
       
    <input type="submit" name="module_plus" value="   +   " onclick="ok()">
       
    <input type="submit" name="module_bevestig" value="OK" onclick="ok()">
  </center>
  </form>
  </small>
  <br>
  <br>
  <br>
  <h6><b>thieu april 2022</b></h6>
  <script>
    function ok(){
      setTimeout(function(){document.location.reload();},250);
    }
  </script>
 </body>
</html>
)rawliteral";

String processor(const String& var){
  String temp = "                              ";
  String module = "                              ";
  int macx_0;
  int macx_1;
  int macx_2;
  int macx_3;
  int macx_4;
  int macx_5;
```

```
if(var == "electriciteit_totaal"){
  temp = String(kwh_totaal_float, 3);
  temp.toCharArray(kwh_totaal_float_char, (temp.length() + 1));
  return(kwh_totaal_float_char);
}
if(var == "injectie_totaal"){
  temp = String(injectie_totaal_float, 3);
  temp.toCharArray(injectie_totaal_float_char, (temp.length() + 1));
  return(injectie_totaal_float_char);
}
if(var == "kw_nu"){
  temp = String(kw_nu_float, 3);
  temp.toCharArray(kw_nu_float_char, (temp.length() + 1));
  return(kw_nu_float_char);
}
if(var == "injectie_nu"){
  temp = String(injectie_nu_float, 3);
  temp.toCharArray(injectie_nu_float_char, (temp.length() + 1));
  return(injectie_nu_float_char);
}
if(var == "gas_totaal"){
  temp = String(gas_totaal_float, 3);
  temp.toCharArray(gas_totaal_float_char, (temp.length() + 1));
  return(gas_totaal_float_char);
}
if(var == "tijd"){
  sprintf(tijd_char, "%02d:%02d", uren, minuten);
  return(tijd_char);
}
if(var == "relais_module"){
  switch(relais_module_teller){
    case 0:
      temp = "Relais 1";
      break;
    case 1:
      temp = "Relais 2";
      break;
    case 2:
      temp = "Relais 3";
      break;
  }
  temp.toCharArray(relais_module_char, (temp.length() + 1));
  return(relais_module_char);
}
if(var == "kw_on"){
  switch(relais_module_teller){
    case 0:
      return(String(relais1_on));
      break;
    case 1:
      return(String(relais2_on));
      break;
    case 2:
      return(String(relais3_on));
      break;
  }
}
if(var == "override"){
  switch(relais_module_teller){
    case 0:
      temp = relais1_override;
      break;
    case 1:
      temp = relais2_override;
      break;
    case 2:
      temp = relais3_override;
      break;
  }
  temp.toCharArray(override_char, (temp.length() + 1));
  return(override_char);
}
if(var == "delay"){
  switch(relais_module_teller){
    case 0:
      return(String(relais1_delay));
      break;
    case 1:
      return(String(relais2_delay));
```

```
      break;
    case 2:
      return(String(relais3_delay));
      break;
  }
}
if(var == "schakel_tijd"){
  switch(relais_module_teller){
    case 0:
      sprintf(schakel_tijd_char, "%02d:%02d", uren_on1_int, minuten_on1_int);
      return(schakel_tijd_char);
      break;
    case 1:
      sprintf(schakel_tijd_char, "%02d:%02d", uren_on2_int, minuten_on2_int);
      return(schakel_tijd_char);
      break;
    case 2:
      sprintf(schakel_tijd_char, "%02d:%02d", uren_on3_int, minuten_on3_int);
      return(schakel_tijd_char);
  }
}
if(var == "pwm_kw"){
  return(String(pwm_kw_float));
}
if(var == "pwm_tijd_on"){
  sprintf(pwm_tijd_on_char, "%02d:%02d", uren_on4_int, minuten_on4_int);
  return(pwm_tijd_on_char);
}
if(var == "pwm_tijd_off"){
  sprintf(pwm_tijd_off_char, "%02d:%02d", uren_off4_int, minuten_off4_int);
  return(pwm_tijd_off_char);
}
if(var == "pwm_override"){
  pwm_override.toCharArray(pwm_override_char, (pwm_override.length() + 1));
  return(pwm_override_char);
}

if(var == "relais1_sturing"){
  if(relais1_uit == true){
    temp = "1";
  }
  if(relais1_uit == false){
    temp = "0";
  }
  if(relais1_override == "0"){
    temp = "0";
  }
  if(relais1_override == "1"){
    temp = "1";
  }
  temp.toCharArray(relais1_sturing_char, (temp.length() + 1));
  return(relais1_sturing_char);
}
if(var == "relais2_sturing"){
  if(relais2_uit == true){
    temp = "1";
  }
  if(relais2_uit == false){
    temp = "0";
  }
  if(relais2_override == "0"){
    temp = "0";
  }
  if(relais2_override == "1"){
    temp = "1";
  }
  temp.toCharArray(relais2_sturing_char, (temp.length() + 1));
  return(relais2_sturing_char);
}
if(var == "relais3_sturing"){
  if(relais3_uit == true){
    temp = "1";
  }
  if(relais3_uit == false){
    temp = "0";
  }
  if(relais3_override == "0"){
    temp = "0";
  }
```

```
  if(relais3_override == "1"){
    temp = "1";
  }
  temp.toCharArray(relais3_sturing_char, (temp.length() + 1));
  return(relais3_sturing_char);
}
if(var == "procent"){
  temp = String(uitsturing_pwm_int);
  temp.toCharArray(uitsturing_pwm_char, (temp.length() + 1));
  return(uitsturing_pwm_char);
}
switch(module_teller){
  case 0:
    module = "MAC address Display";
    macx_0 = broadcastAddress1[0];
    macx_1 = broadcastAddress1[1];
    macx_2 = broadcastAddress1[2];
    macx_3 = broadcastAddress1[3];
    macx_4 = broadcastAddress1[4];
    macx_5 = broadcastAddress1[5];
    break;
  case 1:
    module = "MAC address Relais 1";
    macx_0 = broadcastAddress2[0];
    macx_1 = broadcastAddress2[1];
    macx_2 = broadcastAddress2[2];
    macx_3 = broadcastAddress2[3];
    macx_4 = broadcastAddress2[4];
    macx_5 = broadcastAddress2[5];
    break;
  case 2:
    module = "MAC address Relais 2";
    macx_0 = broadcastAddress3[0];
    macx_1 = broadcastAddress3[1];
    macx_2 = broadcastAddress3[2];
    macx_3 = broadcastAddress3[3];
    macx_4 = broadcastAddress3[4];
    macx_5 = broadcastAddress3[5];
    break;
  case 3:
    module = "MAC address Relais 3";
    macx_0 = broadcastAddress4[0];
    macx_1 = broadcastAddress4[1];
    macx_2 = broadcastAddress4[2];
    macx_3 = broadcastAddress4[3];
    macx_4 = broadcastAddress4[4];
    macx_5 = broadcastAddress4[5];
    break;
  case 4:
    module = "MAC address PWM Sturing";
    macx_0 = broadcastAddress5[0];
    macx_1 = broadcastAddress5[1];
    macx_2 = broadcastAddress5[2];
    macx_3 = broadcastAddress5[3];
    macx_4 = broadcastAddress5[4];
    macx_5 = broadcastAddress5[5];
}
if(var == "module"){
  module.toCharArray(module_char, (module.length() + 1));
  return(module_char);
}
if(var == "display_macx_0"){
  sprintf(broadcastAddressX_0_char, "%02x%", macx_0);
  return(broadcastAddressX_0_char);
}
if(var == "display_macx_1"){
  sprintf(broadcastAddressX_1_char, "%02x%", macx_1);
  return(broadcastAddressX_1_char);
}
if(var == "display_macx_2"){
  sprintf(broadcastAddressX_2_char, "%02x%", macx_2);
  return(broadcastAddressX_2_char);;
}
if(var == "display_macx_3"){
  sprintf(broadcastAddressX_3_char, "%02x%", macx_3);
  return(broadcastAddressX_3_char);
}
if(var == "display_macx_4"){
  sprintf(broadcastAddressX_4_char, "%02x%", macx_4);
```

```
    return(broadcastAddressX_4_char);
   }
   if(var == "display_macx_5"){
    sprintf(broadcastAddressX_5_char, "%02x%", macx_5);
    return(broadcastAddressX_5_char);
   }
}

void html_input(){
  server.begin();
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/html", energie_html, processor);
  });
  server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
    char terminator = char(0x0a);
    String temp = "                          ";
    char temp_char[30];
    float kw_on;
    float kw_off;
    String override = "           ";
    int schakel_delay;
    char char_temp[10];
    bool fout;
    String uren_string = "        ";
    String minuten_string = "        ";
    int uren_int;
    int minuten_int;
    if(request->hasParam(INPUT_TIJD)){
      temp = ((request->getParam(INPUT_TIJD)->value()) + String(terminator));
      if(temp.length() == 6){
        uren_string = temp.substring(0, 2);
        minuten_string = temp.substring(3,5);
        uren_int = uren_string.toInt();
        minuten_int = minuten_string.toInt();
        if((uren_int > 23) || (uren_int < 0)){
          Wire.beginTransmission(DS3231SN);
          Wire.write(0);
          Wire.endTransmission();
          Wire.requestFrom(DS3231SN, 3);
          seconden = bcdToDec(Wire.read() & 0x7f);
          minuten_int = bcdToDec(Wire.read());
          uren_int = bcdToDec(Wire.read() & 0x3f);
        }
        if((minuten_int > 59) || (minuten_int < 0)){
          Wire.beginTransmission(DS3231SN);
          Wire.write(0);
          Wire.endTransmission();
          Wire.requestFrom(DS3231SN, 3);
          seconden = bcdToDec(Wire.read() & 0x7f);
          minuten_int = bcdToDec(Wire.read());
          uren_int = bcdToDec(Wire.read() & 0x3f);
        }
      }
      else{
      Wire.beginTransmission(DS3231SN);
        Wire.write(0);
        Wire.endTransmission();
        Wire.requestFrom(DS3231SN, 3);
        seconden = bcdToDec(Wire.read() & 0x7f);
        minuten_int = bcdToDec(Wire.read());
        uren_int = bcdToDec(Wire.read() & 0x3f);
      }
    }
    if(request->hasParam(TIJD_BEVESTIG)){
      Wire.beginTransmission(DS3231SN);
      Wire.write(0x01);
      Wire.write(decToBcd(minuten_int));
      Wire.endTransmission();
      Wire.beginTransmission(DS3231SN);
      Wire.write(0x02);
      Wire.write(decToBcd(uren_int));
      Wire.endTransmission();
      Wire.beginTransmission(DS3231SN);
      Wire.write(0);
      Wire.endTransmission();
      Wire.requestFrom(DS3231SN, 3);
      seconden = bcdToDec(Wire.read() & 0x7f);
      minuten = bcdToDec(Wire.read());
      uren = bcdToDec(Wire.read() & 0x3f);
```

```
}
if(request->hasParam(INPUT_KW_ON)){
  temp = ((request->getParam(INPUT_KW_ON)->value()) + String(terminator));
  temp.replace(',', '.');
  kw_on = temp.toFloat();
}
if(request->hasParam(INPUT_DELAY)){
  schakel_delay = ((request->getParam(INPUT_DELAY)->value()) + String(terminator)).toInt();
  if(schakel_delay < 10){
    schakel_delay = 10;
  }
}
if(request->hasParam(INPUT_SCHAKEL_TIJD)){
  temp = ((request->getParam(INPUT_SCHAKEL_TIJD)->value()) + String(terminator));
  if(temp.length() == 6){
    uren_string = temp.substring(0, 2);
    minuten_string = temp.substring(3,5);
    uren_int = uren_string.toInt();
    minuten_int = minuten_string.toInt();
    if((uren_int >= 0) && (uren_int <= 24)){
      if((minuten_int >= 0) && (minuten_int <= 59)){
        switch(relais_module_teller){
          case 0:
            uren_on1_int = uren_int;
            minuten_on1_int = minuten_int;
            pref.putInt("uren_on1", uren_int);
            pref.putInt("minuten_on1", minuten_int);
            break;
          case 1:
            uren_on2_int = uren_int;
            minuten_on2_int = minuten_int;
            pref.putInt("uren_on1", uren_int);
            pref.putInt("minuten_on1", minuten_int);
            break;
          case 2:
            uren_on3_int = uren_int;
            minuten_on3_int = minuten_int;
            pref.putInt("uren_on1", uren_int);
            pref.putInt("minuten_on1", minuten_int);
            break;
        }
      }
    }
  }
}
if(request->hasParam(INPUT_OVERRIDE)){
  override = (request->getParam(INPUT_OVERRIDE)->value());
  override.toCharArray(char_temp, (override.length() + 1));
  switch(int(char_temp[0])){
    case 48:          //0
      break;
    case 49:          //1
      break;
    case 97:          //a
      override = "A";
      break;
    case 65:          //A
      break;
    default:
      override = "0";
  }
}
if(request->hasParam(RELAIS_MODULE_MIN)){
  relais_module_teller --;
  if(relais_module_teller < 0){
    relais_module_teller = 2;
  }
}
if(request->hasParam(RELAIS_MODULE_PLUS)){
  relais_module_teller ++;
  if(relais_module_teller > 2){
    relais_module_teller = 0;
  }
}
if(request->hasParam(RELAIS_MODULE_BEVESTIG)){
  switch(relais_module_teller){
    case 0:
      relais1_vertraging_long = millis();
      pref.putFloat("relais1_on", kw_on);
```

```
  pref.putString("relais1_ov", override);
  pref.putInt("relais1_del", schakel_delay);
  relais1_on = pref.getFloat("relais1_on");
  relais1_override = pref.getString("relais1_ov");
  relais1_delay = pref.getInt("relais1_del");
  if(relais1_override == "1"){
    relais1_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress2, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 1");
    }
    else {
      Serial.println("fout bij verzenden naar relais 1");
    }
  }
  else{
    relais1_uit = false;
    uitsturen.relais = false;
    result = esp_now_send(broadcastAddress2, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 1");
    }
    else {
      Serial.println("fout bij verzenden naar relais 1");
    }
  }
  break;
case 1:
  relais2_vertraging_long = millis();
  pref.putFloat("relais2_on", kw_on);
  pref.putString("relais2_ov", override);
  pref.putInt("relais2_del", schakel_delay);
  relais2_on = pref.getFloat("relais2_on");
  relais2_override = pref.getString("relais2_ov");
  relais2_delay = pref.getInt("relais2_del");
  if(relais2_override == "1"){
    relais2_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress3, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 2");
    }
    else {
      Serial.println("fout bij verzenden naar relais 2");
    }
  }
  else{
    relais2_uit = false;
    uitsturen.relais = false;
    result = esp_now_send(broadcastAddress3, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 2");
    }
    else {
      Serial.println("fout bij verzenden naar relais 2");
    }
  }
  break;
case 2:
  relais3_vertraging_long = millis();
  pref.putFloat("relais3_on", kw_on);
  pref.putString("relais3_ov", override);
  pref.putInt("relais3_del",schakel_delay);
  relais3_on = pref.getFloat("relais3_on");
  relais3_override = pref.getString("relais3_ov");
  relais3_delay = pref.getInt("relais3_del");
  if(relais3_override == "1"){
    relais3_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress4, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 3");
    }
    else {
      Serial.println("fout bij verzenden naar relais 3");
    }
  }
  else{
```

```
        relais3_uit = false;
        uitsturen.relais = false;
        result = esp_now_send(broadcastAddress4, (uint8_t *) &uitsturen, sizeof(uitsturen));
        if (result == ESP_OK) {
          Serial.println("Met succes verzonden relais 3");
        }
        else {
          Serial.println("fout bij verzenden naar relais 3");
        }
      }
      break;
  }
}
if(request->hasParam(INPUT_PWM_KW)){
  temp = ((request->getParam(INPUT_PWM_KW)->value()) + String(terminator));
  temp.replace(',', '.');
  pwm_kw_float = temp.toFloat();
}
if(request->hasParam(INPUT_PWM_TIJD_ON)){
  temp = ((request->getParam(INPUT_PWM_TIJD_ON)->value()) + String(terminator));
  if(temp.length() == 6){
    uren_string = temp.substring(0, 2);
    minuten_string = temp.substring(3,5);
    uren_int = uren_string.toInt();
    minuten_int = minuten_string.toInt();
    if((uren_int >= 0) && (uren_int <= 24)){
      if((minuten_int >= 0) && (minuten_int <= 59)){
        uren_on4_int = uren_int;
        minuten_on4_int = minuten_int;
      }
    }
  }
}
if(request->hasParam(INPUT_PWM_TIJD_OFF)){
 temp = ((request->getParam(INPUT_PWM_TIJD_OFF)->value()) + String(terminator));
 if(temp.length() == 6){
   uren_string = temp.substring(0, 2);
   minuten_string = temp.substring(3,5);
   uren_int = uren_string.toInt();
   minuten_int = minuten_string.toInt();
   if((uren_int >= 0) && (uren_int <= 23)){
     if((minuten_int >= 0) && (minuten_int <= 59)){
       uren_off4_int = uren_int;
       minuten_off4_int = minuten_int;
     }
   }
 }
}
if(request->hasParam(INPUT_PWM_OVERRIDE)){
  pwm_override = (request->getParam(INPUT_PWM_OVERRIDE)->value());
  pwm_override.toCharArray(char_temp, (override.length() + 1));
  switch(int(char_temp[0])){
    case 48:        //0
      break;
    case 49:        //1
      break;
    case 97:        //a
      pwm_override = "A";
      break;
    case 65:        //A
      break;
    default:
      pwm_override = "0";
  }
}
if(request->hasParam(BEVESTIG_PWM)){
  pref.putFloat("pwm_kw", pwm_kw_float);
  pref.putInt("uren_on4", uren_on4_int);
  pref.putInt("minuten_on4", minuten_on4_int);
  pref.putInt("uren_off4", uren_off4_int);
  pref.putInt("minuten_off4", minuten_off4_int);
  pref.putString("pwm_override", pwm_override);
  pwm_kw_float = pref.getFloat("pwm_kw");
  uren_on4_int = pref.getInt("uren_on4");
  minuten_on4_int = pref.getInt("minuten_on4");
  uren_off4_int = pref.getInt("uren_off4");
  minuten_off4_int = pref.getInt("minuten_off4");
  pwm_override = pref.getString("pwm_override");
  if(pwm_override == "A"){
```

```
    uitsturing_pwm_float = 0.0;
    uitsturing_pwm_int = 0;
   }
  }
  if(request->hasParam(INPUT_MACX_0)){
   temp = ((request->getParam(INPUT_MACX_0)->value()) + String(terminator));
   temp.toCharArray(temp_char, (temp.length() +1));
   input_macx_0 = strtol(temp_char, 0, 16);
  }
  if(request->hasParam(INPUT_MACX_1)){
   temp = ((request->getParam(INPUT_MACX_1)->value()) + String(terminator));
   temp.toCharArray(temp_char, (temp.length() +1));
   input_macx_1 = strtol(temp_char, 0, 16);
  }
  if(request->hasParam(INPUT_MACX_2)){
   temp = ((request->getParam(INPUT_MACX_2)->value()) + String(terminator));
   temp.toCharArray(temp_char, (temp.length() +1));
   input_macx_2 = strtol(temp_char, 0, 16);
  }
  if(request->hasParam(INPUT_MACX_3)){
   temp = ((request->getParam(INPUT_MACX_3)->value()) + String(terminator));
   temp.toCharArray(temp_char, (temp.length() +1));
   input_macx_3 = strtol(temp_char, 0, 16);
  }
  if(request->hasParam(INPUT_MACX_4)){
   temp = ((request->getParam(INPUT_MACX_4)->value()) + String(terminator));
   temp.toCharArray(temp_char, (temp.length() +1));
   input_macx_4 = strtol(temp_char, 0, 16);
  }
  if(request->hasParam(INPUT_MACX_5)){
   temp = ((request->getParam(INPUT_MACX_5)->value()) + String(terminator));
   temp.toCharArray(temp_char, (temp.length() +1));
   input_macx_5 = strtol(temp_char, 0, 16);
  }
  if(request->hasParam(MODULE_MIN)){
   module_teller --;
   if(module_teller < 0){
    module_teller = 4;
   }
  }
  if(request->hasParam(MODULE_PLUS)){
   module_teller ++;
   if(module_teller > 4){
    module_teller = 0;
   }
  }
  if(request->hasParam(MODULE_BEVESTIG)){
   temp = "";
   fout = false;
   temp = temp + String(input_macx_0)+ String(input_macx_1)+ String(input_macx_2)
            + String(input_macx_3)+ String(input_macx_4)+ String(input_macx_5);
   if(broadcastAddress1_string == temp){
    fout = true;
   }
   if(broadcastAddress2_string == temp){
    fout = true;
   }
   if(broadcastAddress3_string == temp){
    fout = true;
   }
   if(broadcastAddress4_string == temp){
    fout = true;
   }
   if(broadcastAddress5_string == temp){
    fout = true;
   }
   if(fout == false){
    switch(module_teller){
     case 0:
      pref.putInt("mac1_0", input_macx_0);
      pref.putInt("mac1_1", input_macx_1);
      pref.putInt("mac1_2", input_macx_2);
      pref.putInt("mac1_3", input_macx_3);
      pref.putInt("mac1_4", input_macx_4);
      pref.putInt("mac1_5", input_macx_5);
      broadcastAddress1[0] = pref.getInt("mac1_0");
      broadcastAddress1[1] = pref.getInt("mac1_1");
      broadcastAddress1[2] = pref.getInt("mac1_2");
      broadcastAddress1[3] = pref.getInt("mac1_3");
```

```
          broadcastAddress1[4] = pref.getInt("mac1_4");
          broadcastAddress1[5] = pref.getInt("mac1_5");
          break;
        case 1:
          pref.putInt("mac2_0", input_macx_0);
          pref.putInt("mac2_1", input_macx_1);
          pref.putInt("mac2_2", input_macx_2);
          pref.putInt("mac2_3", input_macx_3);
          pref.putInt("mac2_4", input_macx_4);
          pref.putInt("mac2_5", input_macx_5);
          broadcastAddress2[0] = pref.getInt("mac2_0");
          broadcastAddress2[1] = pref.getInt("mac2_1");
          broadcastAddress2[2] = pref.getInt("mac2_2");
          broadcastAddress2[3] = pref.getInt("mac2_3");
          broadcastAddress2[4] = pref.getInt("mac2_4");
          broadcastAddress2[5] = pref.getInt("mac2_5");
          break;
        case 2:
          pref.putInt("mac3_0", input_macx_0);
          pref.putInt("mac3_1", input_macx_1);
          pref.putInt("mac3_2", input_macx_2);
          pref.putInt("mac3_3", input_macx_3);
          pref.putInt("mac3_4", input_macx_4);
          pref.putInt("mac3_5", input_macx_5);
          broadcastAddress3[0] = pref.getInt("mac3_0");
          broadcastAddress3[1] = pref.getInt("mac3_1");
          broadcastAddress3[2] = pref.getInt("mac3_2");
          broadcastAddress1[3] = pref.getInt("mac3_3");
          broadcastAddress3[4] = pref.getInt("mac3_4");
          broadcastAddress3[5] = pref.getInt("mac3_5");
          break;
        case 3:
          pref.putInt("mac4_0", input_macx_0);
          pref.putInt("mac4_1", input_macx_1);
          pref.putInt("mac4_2", input_macx_2);
          pref.putInt("mac4_3", input_macx_3);
          pref.putInt("mac4_4", input_macx_4);
          pref.putInt("mac4_5", input_macx_5);
          broadcastAddress4[0] = pref.getInt("mac4_0");
          broadcastAddress4[1] = pref.getInt("mac4_1");
          broadcastAddress4[2] = pref.getInt("mac4_2");
          broadcastAddress4[3] = pref.getInt("mac4_3");
          broadcastAddress4[4] = pref.getInt("mac4_4");
          broadcastAddress4[5] = pref.getInt("mac4_5");
          break;
        case 4:
          pref.putInt("mac5_0", input_macx_0);
          pref.putInt("mac5_1", input_macx_1);
          pref.putInt("mac5_2", input_macx_2);
          pref.putInt("mac5_3", input_macx_3);
          pref.putInt("mac5_4", input_macx_4);
          pref.putInt("mac5_5", input_macx_5);
          broadcastAddress5[0] = pref.getInt("mac5_0");
          broadcastAddress5[1] = pref.getInt("mac5_1");
          broadcastAddress5[2] = pref.getInt("mac5_2");
          broadcastAddress5[3] = pref.getInt("mac5_3");
          broadcastAddress5[4] = pref.getInt("mac5_4");
          broadcastAddress5[5] = pref.getInt("mac5_5");
          break;
      }
    }
  }
 });
}

void setup() {
 delay(5000);
 Serial.begin(115200);
 Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
 Wire.begin();
 Wire.beginTransmission(DS3231SN);
 Wire.write(0x0F);
 Wire.endTransmission();
 Wire.requestFrom(DS3231SN, 1);
 int test = (Wire.read() & 0x80);
 if(test == 0x80){
  Wire.beginTransmission(DS3231SN);
  Wire.write(0x00);
  Wire.write(0x00);
```

```
  Wire.write(0x00);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.beginTransmission(DS3231SN);
  Wire.write(0x04);
  Wire.write(0x01);
  Wire.write(0x01);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.beginTransmission(DS3231SN);
  Wire.write(0x0E);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.beginTransmission(DS3231SN);
  Wire.write(0x0F);
  Wire.write(0x00);
  Wire.endTransmission();
 }
 pref.begin("data", false);
 if(pref.getString("controle") != "dummy geladen"){
  pref.putInt("mac1_0", 0x30);
  pref.putInt("mac1_1", 0xae);
  pref.putInt("mac1_2", 0xa4);
  pref.putInt("mac1_3", 0x0d);
  pref.putInt("mac1_4", 0x69);
  pref.putInt("mac1_5", 0xb8);
  pref.putInt("mac2_0", 0x7c);
  pref.putInt("mac2_1", 0x9e);
  pref.putInt("mac2_2", 0xbd);
  pref.putInt("mac2_3", 0x06);
  pref.putInt("mac2_4", 0xb4);
  pref.putInt("mac2_5", 0xdc);
  pref.putInt("mac3_0", 2);
  pref.putInt("mac3_1", 1);
  pref.putInt("mac3_2", 2);
  pref.putInt("mac3_3", 3);
  pref.putInt("mac3_4", 4);
  pref.putInt("mac3_5", 5);
  pref.putInt("mac4_0", 6);
  pref.putInt("mac4_1", 7);
  pref.putInt("mac4_2", 8);
  pref.putInt("mac4_3", 9);
  pref.putInt("mac4_4", 0);
  pref.putInt("mac4_5", 1);
  pref.putInt("mac5_0", 4);
  pref.putInt("mac5_1", 1);
  pref.putInt("mac5_2", 2);
  pref.putInt("mac5_3", 3);
  pref.putInt("mac5_4", 4);
  pref.putInt("mac5_5", 5);
  pref.putFloat("relais1_on", 2.0);
  pref.putString("relais1_ov", "0");
  pref.putInt("relais1_del", 10);
  pref.putFloat("relais2_on", 2.0);
  pref.putString("relais2_ov", "0");
  pref.putInt("relais2_del", 10);
  pref.putFloat("relais3_on", 2.0);
  pref.putString("relais3_ov", "0");
  pref.putInt("relais3_del", 10);
  pref.putInt("uren_on1", 24);
  pref.putInt("minuten_on1", 0);
  pref.putInt("uren_on2", 24);
  pref.putInt("minuten_on2", 0);
  pref.putInt("uren_on3", 24);
  pref.putInt("minuten_on3", 0);
  pref.putFloat("pwm_kw", 0.0);
  pref.putInt("uren_on4", 24);
  pref.putInt("minuten_on4", 0);
  pref.putInt("uren_off4", 0);
  pref.putInt("minuten_off4", 0);
  pref.putString("pwm_override", "0");
  pref.putString("controle", "dummy geladen");
 }
 broadcastAddress1[0] = pref.getInt("mac1_0");
 broadcastAddress1[1] = pref.getInt("mac1_1");
 broadcastAddress1[2] = pref.getInt("mac1_2");
 broadcastAddress1[3] = pref.getInt("mac1_3");
 broadcastAddress1[4] = pref.getInt("mac1_4");
 broadcastAddress1[5] = pref.getInt("mac1_5");
```

```
broadcastAddress2[0] = pref.getInt("mac2_0");
broadcastAddress2[1] = pref.getInt("mac2_1");
broadcastAddress2[2] = pref.getInt("mac2_2");
broadcastAddress2[3] = pref.getInt("mac2_3");
broadcastAddress2[4] = pref.getInt("mac2_4");
broadcastAddress2[5] = pref.getInt("mac2_5");
broadcastAddress3[0] = pref.getInt("mac3_0");
broadcastAddress3[1] = pref.getInt("mac3_1");
broadcastAddress3[2] = pref.getInt("mac3_2");
broadcastAddress3[3] = pref.getInt("mac3_3");
broadcastAddress3[4] = pref.getInt("mac3_4");
broadcastAddress3[5] = pref.getInt("mac3_5");
broadcastAddress4[0] = pref.getInt("mac4_0");
broadcastAddress4[1] = pref.getInt("mac4_1");
broadcastAddress4[2] = pref.getInt("mac4_2");
broadcastAddress4[3] = pref.getInt("mac4_3");
broadcastAddress4[4] = pref.getInt("mac4_4");
broadcastAddress4[5] = pref.getInt("mac4_5");
broadcastAddress5[0] = pref.getInt("mac5_0");
broadcastAddress5[1] = pref.getInt("mac5_1");
broadcastAddress5[2] = pref.getInt("mac5_2");
broadcastAddress5[3] = pref.getInt("mac5_3");
broadcastAddress5[4] = pref.getInt("mac5_4");
broadcastAddress5[5] = pref.getInt("mac5_5");
relais1_on = pref.getFloat("relais1_on");
relais1_override = pref.getString("relais1_ov");
relais1_delay = pref.getInt("relais1_del");
relais2_on = pref.getFloat("relais2_on");
relais2_override = pref.getString("relais2_ov");
relais2_delay = pref.getInt("relais2_del");
relais3_on = pref.getFloat("relais3_on");
relais3_override = pref.getString("relais3_ov");
relais3_delay = pref.getInt("relais3_del");
uren_on1_int = pref.getInt("uren_on1");
minuten_on1_int = pref.getInt("minuten_on1");
uren_on2_int = pref.getInt("uren_on2");
minuten_on2_int = pref.getInt("minuten_on2");
uren_on3_int = pref.getInt("uren_on3");
minuten_on3_int = pref.getInt("minuten_on3");
pwm_kw_float = pref.getFloat("pwm_kw");
uren_on4_int = pref.getInt("uren_on4");
minuten_on4_int = pref.getInt("minuten_on4");
uren_off4_int = pref.getInt("uren_off4");
minuten_off4_int = pref.getInt("minuten_off4");
pwm_override = pref.getString("pwm_override");
broadcastAddress1_string = "";
broadcastAddress1_string = broadcastAddress1_string + String(broadcastAddress1[0])+ String(broadcastAddress1[1])
                                    + String(broadcastAddress1[2])+ String(broadcastAddress1[3])
                                    + String(broadcastAddress1[4])+ String(broadcastAddress1[5]);
broadcastAddress2_string = "";
broadcastAddress2_string = broadcastAddress2_string + String(broadcastAddress2[0])+ String(broadcastAddress2[1])
                                    + String(broadcastAddress2[2])+ String(broadcastAddress2[3])
                                    + String(broadcastAddress2[4])+ String(broadcastAddress2[5]);
broadcastAddress3_string = "";
broadcastAddress3_string = broadcastAddress3_string + String(broadcastAddress3[0])+ String(broadcastAddress3[1])
                                    + String(broadcastAddress3[2])+ String(broadcastAddress3[3])
                                    + String(broadcastAddress3[4])+ String(broadcastAddress3[5]);
broadcastAddress4_string = "";
broadcastAddress4_string = broadcastAddress4_string + String(broadcastAddress4[0])+ String(broadcastAddress4[1])
                                    + String(broadcastAddress4[2])+ String(broadcastAddress4[3])
                                    + String(broadcastAddress4[4])+ String(broadcastAddress4[5]);
broadcastAddress5_string = "";
broadcastAddress5_string = broadcastAddress5_string + String(broadcastAddress5[0])+ String(broadcastAddress5[1])
                                    + String(broadcastAddress5[2])+ String(broadcastAddress5[3])
                                    + String(broadcastAddress5[4])+ String(broadcastAddress5[5]);
WiFi.mode(WIFI_AP_STA);
WiFi.softAP(APssid, APpswd);
if (esp_now_init() != ESP_OK) {
  Serial.println("Error initializing ESP-NOW");
  return;
}

esp_now_register_send_cb(OnDataSent);

peerInfo.channel = 0;
peerInfo.encrypt = false;

memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
if (esp_now_add_peer(&peerInfo) != ESP_OK){
```

```
    Serial.println("Failed to add peer 1");
    return;
  }
  memcpy(peerInfo.peer_addr, broadcastAddress2, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer 2");
    return;
  }
  memcpy(peerInfo.peer_addr, broadcastAddress3, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer 3");
    return;
  }
  memcpy(peerInfo.peer_addr, broadcastAddress4, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer 4");
    return;
  }
  memcpy(peerInfo.peer_addr, broadcastAddress5, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer 5");
    return;
  }
  delay(1000);
  Serial.println(relais1_override);
  if(relais1_override == "0"){
    uitsturen.relais = false;
    relais1_uit = false;
  }
  if(relais1_override == "1"){
    uitsturen.relais = true;
    relais1_uit = true;
  }
  result = esp_now_send(broadcastAddress2, (uint8_t *) &uitsturen, sizeof(uitsturen));
  if (result == ESP_OK) {
    Serial.println("Met succes verzonden relais 1");
  }
  else {
    Serial.println("fout bij verzenden naar relais 1");
  }
  delay(1000);
  if(relais2_override == "0"){
    uitsturen.relais = false;
    relais2_uit = false;
  }
  if(relais2_override == "1"){
    uitsturen.relais = true;
    relais2_uit = true;
  }
  result = esp_now_send(broadcastAddress3, (uint8_t *) &uitsturen, sizeof(uitsturen));
  if (result == ESP_OK) {
    Serial.println("Met succes verzonden relais 2");
  }
  else {
    Serial.println("fout bij verzenden naar relais 2");
  }
  delay(1000);
  if(relais3_override == "0"){
    uitsturen.relais = false;
    relais3_uit = false;
  }
  if(relais3_override == "1"){
    uitsturen.relais = true;
    relais3_uit = true;
  }
  result = esp_now_send(broadcastAddress4, (uint8_t *) &uitsturen, sizeof(uitsturen));
  if (result == ESP_OK) {
    Serial.println("Met succes verzonden relais 3");
  }
  else {
    Serial.println("fout bij verzenden naar relais 3");
  }
  delay(1000);
  html_input();
  nu = millis();
}

void loop() {
  while(Serial2.available()){
```

```
char lees_byte = Serial2.read();
if(lees_byte == 0x2f){
  kwh_dag_float = kwh_dag.toFloat();
  kwh_nacht_float = kwh_nacht.toFloat();
  kwh_totaal_float = kwh_nacht_float + kwh_dag_float;
  injectie_dag_float = injectie_dag.toFloat();
  injectie_nacht_float = injectie_nacht.toFloat();
  injectie_totaal_float = injectie_nacht_float + injectie_dag_float;
  kw_nu_float = kw_nu.toFloat();
  injectie_nu_float = injectie_nu.toFloat();
  verbruik_nu_float = injectie_nu_float - kw_nu_float;
  gas_totaal_float = gas.toFloat();
  ingelezen.kwh_totaal = kwh_totaal_float;
  ingelezen.injectie_totaal = injectie_totaal_float;
  ingelezen.verbruik_nu = kw_nu_float;
  ingelezen.injectie_nu = injectie_nu_float;
  ingelezen.gas_totaal = gas_totaal_float;
  ingelezen.relais1 = relais1_uit;
  ingelezen.relais2 = relais2_uit;
  ingelezen.relais3 = relais3_uit;
  ingelezen.pwm_sturing = uitsturing_pwm_int;
  if(((millis() - nu) > 5000) && (!vijf_seconden)){
    vijf_seconden = true;
    pwm_tijd_gezet_vorig = pwm_tijd_gezet;
    if((uren_on4_int == uren) && (minuten_on4_int == minuten)){
      pwm_tijd_gezet = true;
    }
    if((uren_off4_int == uren) && (minuten_off4_int == minuten)){
      pwm_tijd_gezet = false;
    }
    if((pwm_tijd_gezet_vorig) && (!pwm_tijd_gezet)){
      uitsturing_pwm_int = 0;
    }
    if(pwm_tijd_gezet == false){
      uitsturing_pwm_float = uitsturing_pwm_float + ((verbruik_nu_float / pwm_kw_float) / 2);
      if(uitsturing_pwm_float > 1.0){
        uitsturing_pwm_float = 1.0;
      }
      if(uitsturing_pwm_float < 0.0){
        uitsturing_pwm_float = 0.0;
      }
      uitsturing_pwm_int = uitsturing_pwm_float * 100;
    }
    else{
      uitsturing_pwm_int = 100;
    }
    if(pwm_override == "0"){
      uitsturing_pwm_int = 0;
      uitsturing_pwm_float = 0.0;
    }
    if(pwm_override == "1"){
      uitsturing_pwm_int = 100;
    }
    pwm_sturing.procent = uitsturing_pwm_int;
    result = esp_now_send(broadcastAddress5, (uint8_t *) &pwm_sturing, sizeof(pwm_sturing));
    if (result == ESP_OK) {
      Serial.println("Sent with success");
    }
    else {
      Serial.println("Error sending the data");
    }
  }
  if((millis() - nu) > 10000){
    nu = millis();
    vijf_seconden = false;
    /*
    * Pulse sturing
    */
    pwm_tijd_gezet_vorig = pwm_tijd_gezet;
    if((uren_on4_int == uren) && (minuten_on4_int == minuten)){
      pwm_tijd_gezet = true;
    }
    if((uren_off4_int == uren) && (minuten_off4_int == minuten)){
      pwm_tijd_gezet = false;
    }
    if((pwm_tijd_gezet_vorig) && (!pwm_tijd_gezet)){
      uitsturing_pwm_int = 0;
    }
    if(pwm_tijd_gezet == false){
```

```
      uitsturing_pwm_float = uitsturing_pwm_float + ((verbruik_nu_float / pwm_kw_float) / 2);
      if(uitsturing_pwm_float > 1.0){
        uitsturing_pwm_float = 1.0;
      }
      if(uitsturing_pwm_float < 0.0){
        uitsturing_pwm_float = 0.0;
      }
      uitsturing_pwm_int = uitsturing_pwm_float * 100;
    }
    else{
      uitsturing_pwm_int = 100;
    }
    if(pwm_override == "0"){
      uitsturing_pwm_int = 0;
      uitsturing_pwm_float = 0.0;
    }
    if(pwm_override == "1"){
      uitsturing_pwm_int = 100;
    }
    pwm_sturing.procent = uitsturing_pwm_int;
    result = esp_now_send(broadcastAddress1, (uint8_t *) &ingelezen, sizeof(ingelezen));
    if (result == ESP_OK) {
      Serial.println("Sent with success");
    }
    else {
      Serial.println("Error sending the data");
    }
    result = esp_now_send(broadcastAddress5, (uint8_t *) &pwm_sturing, sizeof(pwm_sturing));
    if (result == ESP_OK) {
      Serial.println("Sent with success");
    }
    else {
      Serial.println("Error sending the data");
    }
    Wire.beginTransmission(DS3231SN);
    Wire.write(0);
    Wire.endTransmission();
    Wire.requestFrom(DS3231SN, 3);
    seconden = bcdToDec(Wire.read() & 0x7f);
    minuten = bcdToDec(Wire.read());
    uren = bcdToDec(Wire.read() & 0x3f);
   }
   buffer_data = "";
  }
  buffer_data += lees_byte;
  if(lees_byte == 0x0a){
    if((buffer_data.substring(4,9)) == "1.8.1"){
      kwh_dag = buffer_data.substring(10,20);
    }
    if((buffer_data.substring(4,9)) == "1.8.2"){
      kwh_nacht = buffer_data.substring(10,20);
    }
    if((buffer_data.substring(4,9)) == "2.8.1"){
      injectie_dag = buffer_data.substring(10,20);
    }
    if((buffer_data.substring(4,9)) == "2.8.2"){
      injectie_nacht = buffer_data.substring(10,20);
    }
    if((buffer_data.substring(4,9)) == "1.7.0"){
      kw_nu = buffer_data.substring(10,16);
    }
    if((buffer_data.substring(4,9)) == "2.7.0"){
      injectie_nu = buffer_data.substring(10,16);
    }
    if((buffer_data.substring(4,10)) == "24.2.3"){
      gas = buffer_data.substring(26,35);
    }
    //Serial.print(buffer_data);
    buffer_data = "";
  }
}
if(pwm_override == "1"){
  verbruik_pwm_float = verbruik_nu_float;
}
else{
  verbruik_pwm_float = verbruik_nu_float+ (uitsturing_pwm_float * pwm_kw_float);
}

/*
```

```
 * Relais 1
 */
if(relais1_override == "0"){
  relais1_vertraging_long = millis();
}
if(relais1_override == "1"){
  relais1_vertraging_long = millis();
}
if((relais1_uit == false) && (relais1_override != "0")){
  if(relais1_on > verbruik_pwm_float){
    relais1_vertraging_long = millis();
  }
  if((millis() - relais1_vertraging_long) > (relais1_delay * 60000)){        //1 minuut = 60000 mS
    relais1_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress2, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 1");
    }
    else {
      Serial.println("fout bij verzenden naar relais 1");
    }
  }
  relais2_vertraging_long = millis();
  relais3_vertraging_long = millis();
  if((uren == uren_on1_int) && (minuten == minuten_on1_int)){
    relais1_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress2, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 1");
    }
    else {
      Serial.println("fout bij verzenden naar relais 1");
    }
  }
}
 /*
 * Relais 2
 */
if(relais2_override == "0"){
  relais2_vertraging_long = millis();
}
if(relais2_override == "1"){
  relais2_vertraging_long = millis();
}
if((relais2_uit == false) && (relais2_override != "0")){
  if(relais2_on > verbruik_pwm_float){
    relais2_vertraging_long = millis();
  }
  if((millis() - relais2_vertraging_long) > (relais2_delay * 60000)){        //1 minuut = 60000 mS
    relais2_vertraging_long = millis();
    relais2_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress3, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 2");
    }
    else {
      Serial.println("fout bij verzenden naar relais 2");
    }
  }
  relais3_vertraging_long = millis();
  if((uren == uren_on2_int) && (minuten == minuten_on2_int)){
    relais2_uit = true;
    uitsturen.relais = true;
    result = esp_now_send(broadcastAddress3, (uint8_t *) &uitsturen, sizeof(uitsturen));
    if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 2");
    }
    else {
      Serial.println("fout bij verzenden naar relais 2");
    }
  }
}
/*
 * Relais 3
 */
if(relais3_override == "0"){
```

```
    relais3_vertraging_long = millis();
   }
   if(relais3_override == "1"){
    relais3_vertraging_long = millis();
   }
   if((relais3_uit == false) && (relais3_override != "0")){
    if(relais3_on > verbruik_pwm_float){
     relais3_vertraging_long = millis();
    }
    if((millis() - relais3_vertraging_long) > (relais3_delay * 60000)){        //1 minuut = 60000 mS
     relais3_vertraging_long = millis();
     relais3_uit = true;
     uitsturen.relais = true;
     result = esp_now_send(broadcastAddress4, (uint8_t *) &uitsturen, sizeof(uitsturen));
     if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 3");
     }
     else {
      Serial.println("fout bij verzenden naar relais 3");
     }
    }
    if((uren == uren_on3_int) && (minuten == minuten_on3_int)){
     relais3_uit = true;
     uitsturen.relais = true;
     result = esp_now_send(broadcastAddress4, (uint8_t *) &uitsturen, sizeof(uitsturen));
     if (result == ESP_OK) {
      Serial.println("Met succes verzonden relais 3");
     }
     else {
      Serial.println("fout bij verzenden naar relais 3");
     }
    }
   }

}
```

MAC_adres_ESP32.ino

```
#include <WiFi.h>

void setup(){
  Serial.begin(115200);
  Serial.println();
  Serial.print("ESP Board MAC Address:  ");
  Serial.println(WiFi.macAddress());
}

void loop(){

}
```

slimme_meter_esp32_display.ino

```
#include <esp_now.h>
#include <WiFi.h>
#include <TFT_eSPI.h>
#include <SPI.h>

TFT_eSPI tft = TFT_eSPI();

typedef struct meter_data{
  float kwh_totaal;
  float injectie_totaal;
  float injectie_nu;
  float verbruik_nu;
  float gas_totaal;
  bool  relais1;
  bool  relais2;
  bool  relais3;
  int   pwm_sturing;
}meter_data;
meter_data ingelezen;

unsigned long begin_millis;
int positie = 0;
int x = 1;

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&ingelezen, incomingData, sizeof(ingelezen));
  tft.fillScreen(TFT_BLACK);
  tft.setTextColor(TFT_WHITE);
  tft.setTextSize(1);
  tft.setCursor(0, positie);
  tft.print("Totaal electriciteit");
  tft.setCursor(200, positie);
  tft.print("Totaal injectie");
  tft.setCursor(400, positie);
  tft.print("Totaal gas");
  tft.setCursor(0, positie + 15);
  tft.print(ingelezen.kwh_totaal, 3);
  tft.print(" KWh");
  tft.setCursor(200, positie + 15);
  tft.print(ingelezen.injectie_totaal, 3);
  tft.print(" KWh");
  tft.setCursor(400, positie + 15);
  tft.print(ingelezen.gas_totaal, 3);
  tft.print(" m3");
  tft.setTextSize(2);
  if(ingelezen.relais1){
    tft.setCursor(20, positie + 40);
    tft.setTextColor(TFT_YELLOW);
    tft.print("R1");
  }
  if(ingelezen.relais2){
    tft.setCursor(90, positie + 40);
    tft.setTextColor(TFT_YELLOW);
    tft.print("R2");
  }
  if(ingelezen.relais3){
    tft.setCursor(370, positie + 40);
    tft.setTextColor(TFT_YELLOW);
    tft.print("R3");
  }
  if(ingelezen.pwm_sturing > 0){
    tft.setCursor(420, positie + 40);
    tft.setTextColor(TFT_YELLOW);
    tft.print(ingelezen.pwm_sturing);
    tft.print("%");
  }

  tft.setCursor(200, positie + 40);
  if(ingelezen.injectie_nu >= ingelezen.verbruik_nu){
    tft.setTextColor(TFT_GREEN);
    tft.print(ingelezen.injectie_nu, 3);
    tft.print(" KW");
  }
```

```
  else{
    tft.setTextColor(TFT_RED);
    tft.setTextSize(2);
    tft.print(ingelezen.verbruik_nu, 3);
    tft.print(" KW");
  }
  positie = positie + x;
  if((positie == 250)|| (positie == 0)){
    x = x * -1;
  }
}

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_register_recv_cb(OnDataRecv);
  tft.begin();
  tft.setRotation(1);
  tft.setTextSize(1);
  tft.fillScreen(TFT_BLACK);
}

void loop() {

}
```

ESP32_relais.ino

```
#include <esp_now.h>
#include <WiFi.h>

#define STUUR_OUTPUT  16

typedef struct relais_data{
  bool relais;
}relais_data;

relais_data ingelezen;

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&ingelezen, incomingData, sizeof(ingelezen));
  if(ingelezen.relais){
    digitalWrite(STUUR_OUTPUT, HIGH);
  }
  else{
    digitalWrite(STUUR_OUTPUT, LOW);
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(STUUR_OUTPUT, OUTPUT);
  digitalWrite(STUUR_OUTPUT, LOW);
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {
}
```

esp32_pwm.ino

```
#include <esp_now.h>
#include <WiFi.h>

const int ssd = 16;
const int freq = 10;
const int ssdChannel = 0;
const int resolution = 8;

typedef struct pwm_data{
  int procent;
}
pwm_data;
pwm_data pwm_sturing;

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&pwm_sturing, incomingData, sizeof(pwm_sturing));
  int uit = (pwm_sturing.procent * 255) / 100;
  Serial.println(uit);
  ledcWrite(ssdChannel, uit);
}

void setup() {
  Serial.begin(115200);
  ledcSetup(ssdChannel, freq, resolution);
  ledcAttachPin(ssd, ssdChannel);
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_register_recv_cb(OnDataRecv);
}

void loop(){
}
```