

# 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 voor verdere verwerking.

Er zijn verschillende interface modules beschikbaar.

Alvorens verder te gaan met deze interfaces, van elk interface zijn er 2 versies beschikbaar. Eentje eindigend op STA en een ander eindigend op AP.

Dit heeft betrekking op de manier waarop er verbinding kan gemaakt worden met de interface.

## De STA interface module. (*bestandsnaam eindigt op \_STA.ino*)

Hierbij is de ESP32 ingesteld in **Station Mode**. Dit wil zeggen de gegevens van de interface beschikbaar zijn via je eigen WiFi netwerk

**IPaddress:** 192.168.1.222

## De AP interface module. (*bestandsnaam eindigt op \_AP.ino*)

Hierbij is de ESP32 ingesteld in de **Access Point Mode**. De ESP32 heeft zijn eigen netwerk namelijk

**Netwerk:** ESP32Energie

**IPaddress:** 192.168.4.1

**Paswoord:** ESP32pswd

## **Soorten interface modules.**

### **Slimme\_meter\_web\_STA en Slimme\_meter\_web\_AP.**

De meest eenvoudige uitvoering.

Leest de gegevens van de P1 poort en stuurt deze naar een webpagina.

Vooral nuttig om op een eenvoudige manier je verbruik op te volgen en om op een effectieve manier het sluikverbruik op te sporen.

Beschikbare gegevens :

Totaal elektriciteitsverbruik

Totaal injectie

Verbruik nu

Injectie nu

Totaal gas in m<sup>3</sup>

### **Slimme\_meter\_esp32\_STA en Slimme\_meter\_esp32\_AP.**

Deze is vooral nuttig als je zelf elektriciteit opwekt

Leest de gegevens van de P1 poort en stuurt deze naar een webpagina.

Beschikbare gegevens :

Totaal elektriciteitsverbruik

Totaal injectie

Verbruik nu

Injectie nu

Totaal gas in m<sup>3</sup>

Deze gegevens kan je eveneens doorsturen naar een display module .

3 relaisuitgangen om (huishoud)toestellen te sturen in functie van beschikbaar zelf opgewekt vermogen of op tijdsbasis.

Een PWM uitgang om een elektrische boiler op te warmen in functie van zelf opgewekt vermogen zonder gebruik te moeten maken van aangekocht vermogen.

### **Slimme\_meter\_esp32\_data\_STA en Slimme\_meter\_esp32\_data\_AP.**

Deze heeft dezelfde functies als de vorige namelijk :

Leest de gegevens van de P1 poort en stuurt deze naar een webpagina.

Beschikbare gegevens :

Totaal elektriciteitsverbruik

Totaal injectie

Verbruik nu

Injectie nu

Totaal gas in m<sup>3</sup>

Deze gegevens kan je eveneens doorsturen naar een display module.

3 relaisuitgangen om (huishoud)toestellen te sturen in functie van beschikbaar zelf opgewekt vermogen of op tijdsbasis.

Een PWM uitgang om een elektrische boiler op te warmen in functie van zelf opgewekt vermogen zonder gebruik te moeten maken van aangekocht vermogen.

Maar heeft daarnaast ook nog de mogelijkheid om de verbruik / injectie gegevens te bewaren en weer te geven via de webpagina.

Dit op uur, dag, maand en jaarbasis. Het bewaren van de opgeslagen data van 1 volledig jaar gebruikt +/- 140 kB van een SD kaart. Met een SD kaart van 16GB heb je dus genoeg ruimte om enkele (honderden) jaren aan gegevens op te slaan.

## **Overige modules.**

### **Display**

een apart LCD scherm met weergave van totaal verbruik gas en elektriciteit en het totaal geïnjecteerd vermogen. Weergave van het huidig elektrisch verbruik of injectie. 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 goede dagen starten op een bepaald uur.

### **PWM module**

Voor gebruik bij een elektrische warmwaterboiler. 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.

Of indien je in het bezit bent van een hotfill wasmachine of vaatwasmachine kan je hiermee een keukenboiler opwarmen welke je dan aansluit op de warmwateringang van je toestel.

## **Enkele nuttige adressen:**

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.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/>

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/>

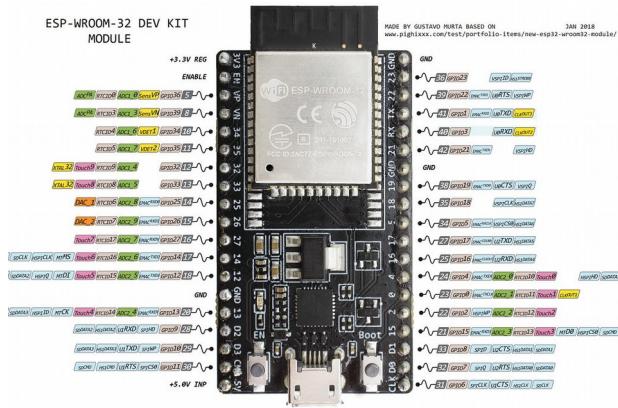
Er zijn meerdere types “slimme meter” in omloop. De aansluitingen en software van dit project zijn gebaseerd op het type meter hierboven afgebeeld. Meer info over de andere meters vind je op de link hieronder. Hoe je het programma eventueel kan aanpassen vind je terug op blz 28.

<https://domoticx.com/p1-poort-slimme-meter-hardware/>

github pagina voor dit project, controleer hier eerst voor mogelijke updates.

<https://github.com/thieu-b55/Slimme-Digitale-Energie-Meter>

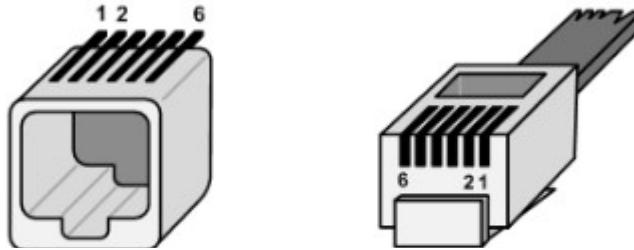
# ESP32 Devkit



## RJ12 connector

## Foto's gekopieerd uit

[https://www.netbeheernederland.nl/\\_upload/Files/Slimme\\_meter\\_15\\_a727fce1f1.pdf](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 modules

### **Slimme\_meter\_web\_STA en Slimme\_meter\_web\_AP.**

De meest eenvoudige module. Leest de gegevens van de P1 poort en stuurt die door naar een webpagina.

#### **Onderdelen**

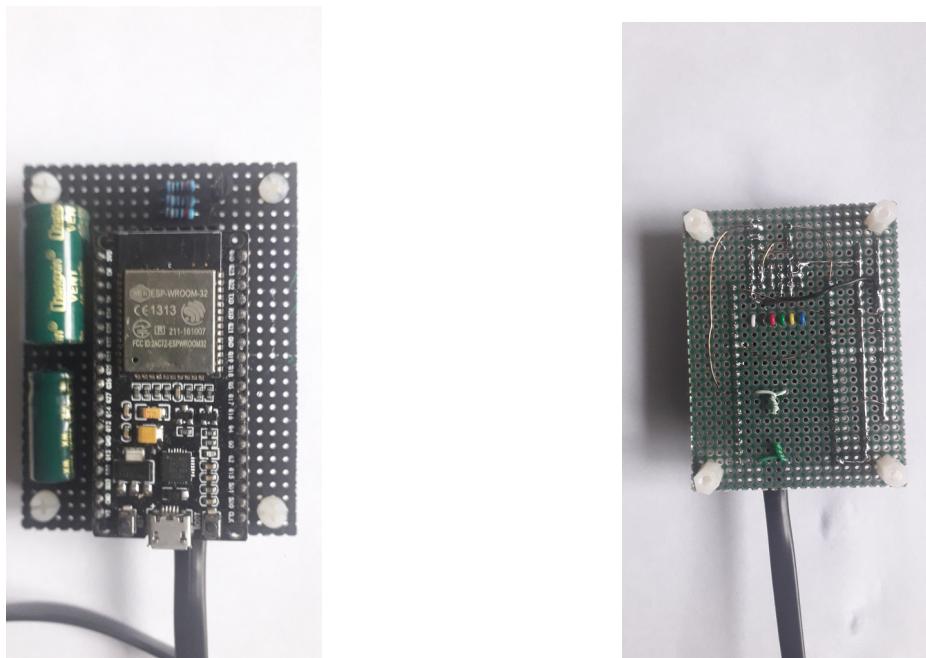
1 x kabeltje met RJ12 plug  
1x ESP32WROOM Devkit  
1 x Led  
1x BC547  
1x R 1K  
1x R 10K  
1 x R 2K2  
1 x R 20K  
2x 4700 uF / 16 Volt  
1x 2200 uF / 16 Volt  
(eventueel een aparte 5 Volt voeding zie beschrijving)

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. 3 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 3 capaciteiten zijn dan niet nodig. Ontkoppel dan ook de 5V komende van de meter, Pin2 (Data Request) mag met de 5V van de meter verbonden blijven.

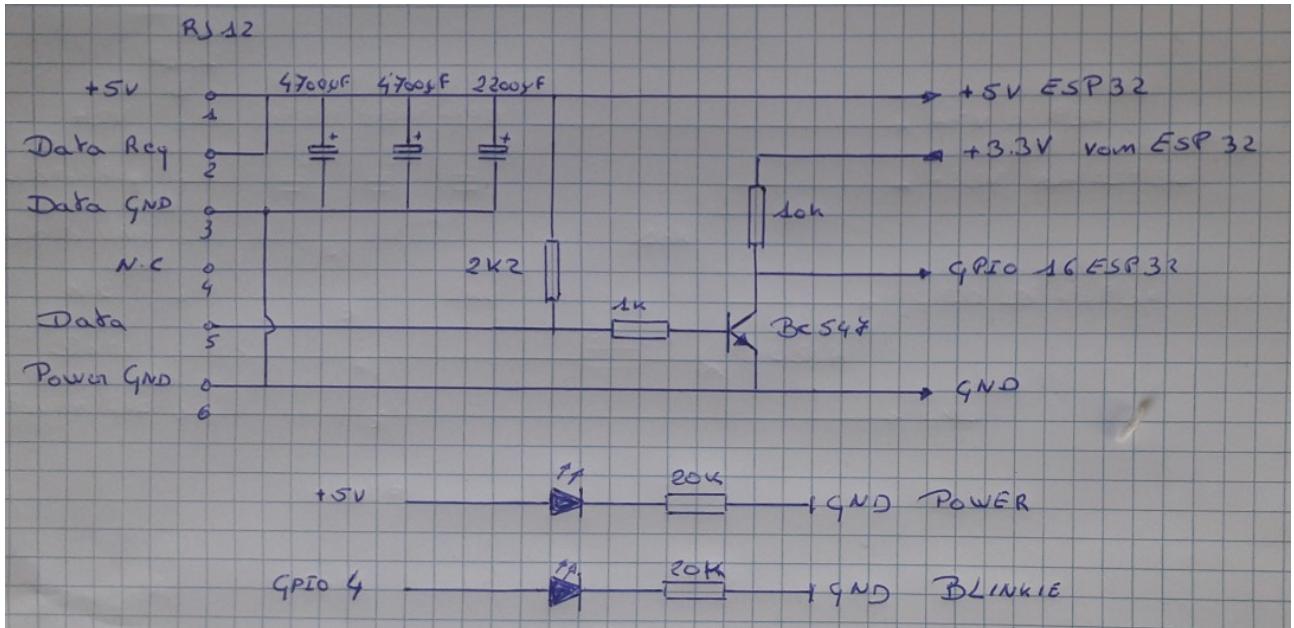
***Opmerking in verband met voeding. Ik heb meerdere van deze modules gemaakt en niet alle ESP32's blijven werken met de condensator oplossing, gebruik in dat geval een externe 5V voeding.***



Maak de nodige verbindingen zoals op het schema hieronder.

RJ12 is de verbinding met de P1 poort.

Een ESP32 Devkit heeft meerdere GND aansluitingen. Best is die allemaal aan te sluiten.



Laad nu het programma

**Slimme\_meter\_web\_STA.ino**

gegevens bereikbaar via je WiFi netwerk

of

**Slimme\_meter\_web\_AP.ino**

gegevens bereikbaar via ESP32Energie netwerk

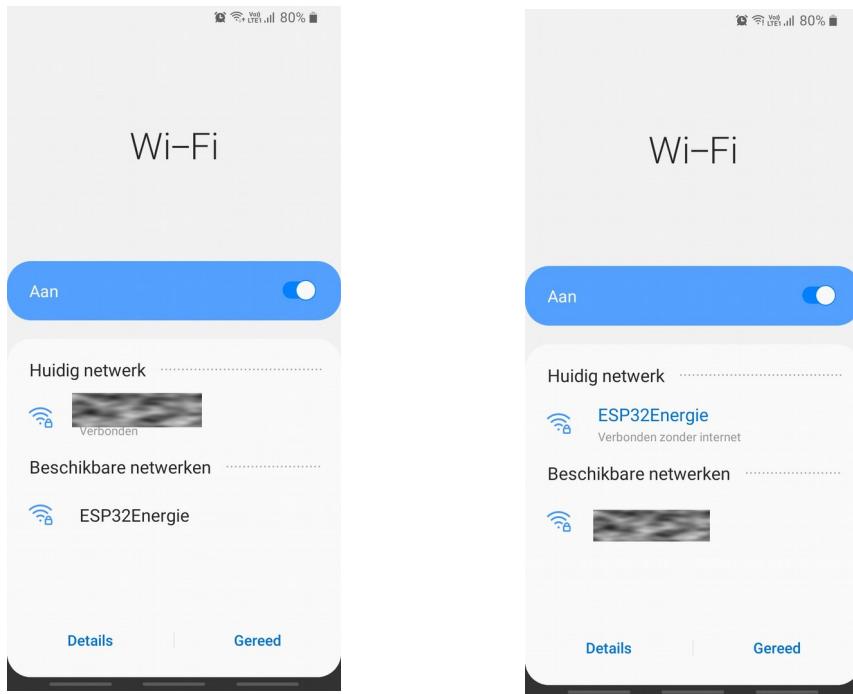
met de Arduino IDE in de ESP32.

Plug de module in de geactiveerde P1 poort, voor de uitvoering met condensators zonder de externe 5V voeding, druk op de ESP32Devkit resetknop (links onder).

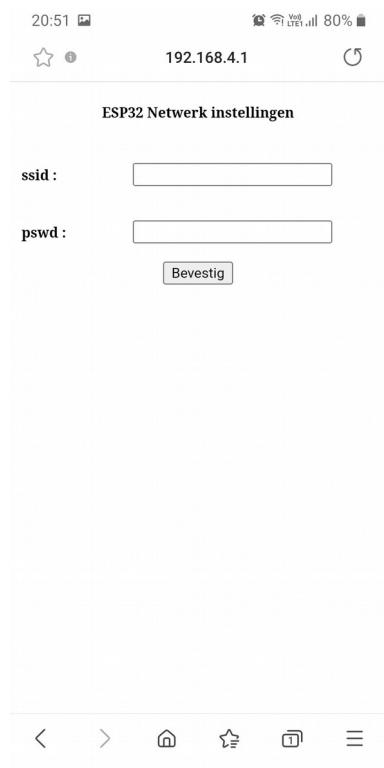
## **Slimme\_meter\_web\_STA.ino**

Na het opstarten is er een nieuw netwerk beschikbaar, ESP32energie  
maak verbinding met dit netwerk

netwerk:      ESP32Energie  
paswoord:     ESP32pswd



ga naar pagina 192.168.4.1



Vul de gevraagde gegevens in.

ssid >> naam van jouw Wifi netwerk

pswd >> paswoord van jouw Wifi netwerk

Druk op bevestig.

De ESP32 herstart automatisch. Indien alle Wifi gegevens correct zijn ingevuld zal de ESP32 zich verbinden met jouw Wifi netwerk. Netwerk ESP32Energie is niet meer beschikbaar.

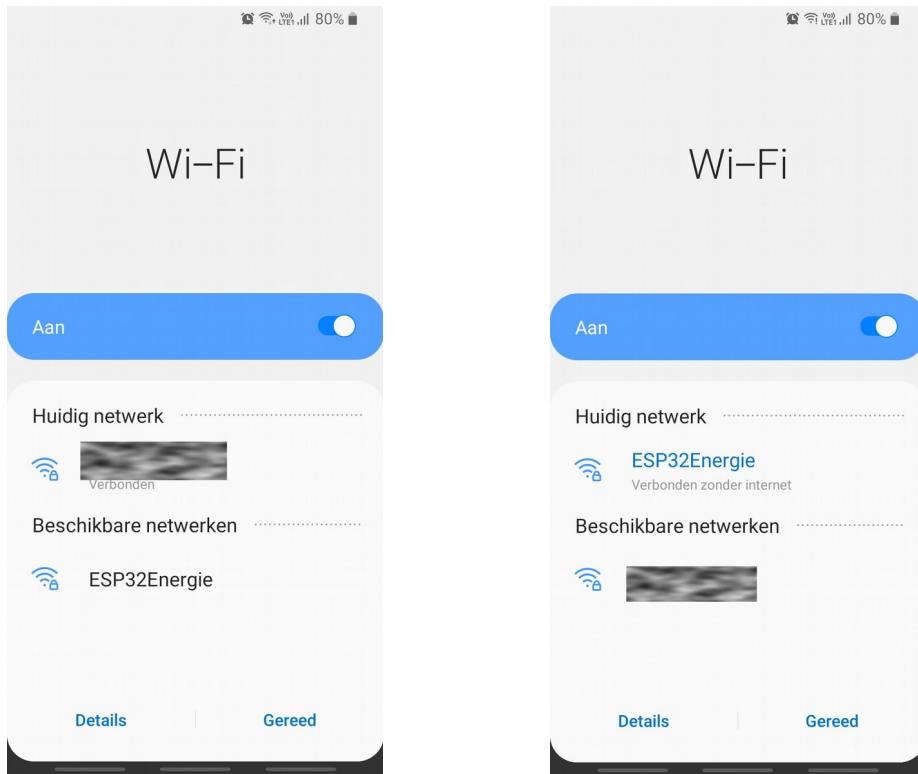
Ga naar pagina 192.168.1.222 van jouw eigen netwerk om je verbruiksgegevens te raadplegen.  
Pagina herlaadt automatisch elke 15 seconden.



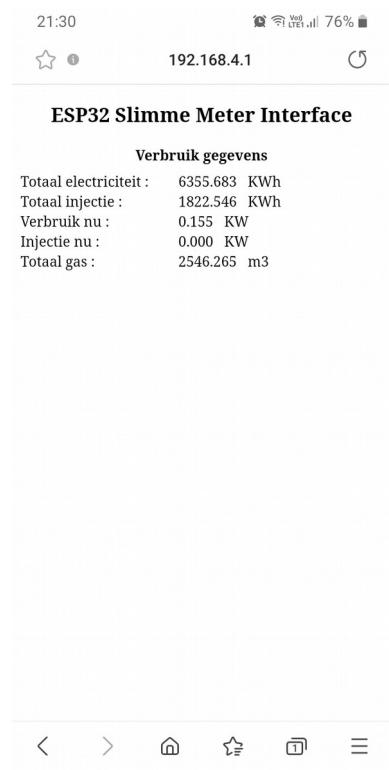
## Slimme\_meter\_web\_AP.ino

Na het opstarten is er een nieuw netwerk beschikbaar, namelijk ESP32energie maak verbinding met dit netwerk

netwerk: ESP32Energie  
paswoord: ESP32pswd



Ga naar pagina 192.168.4.1 om je verbruiksgegevens te raadplegen.  
Pagina herlaadt automatisch elke 15 seconden.



## **Slimme\_meter\_esp32\_STA en Slimme\_meter\_esp32\_AP.**

Leest de gegevens van de P1 poort en stuurt deze naar een webpagina.

Mogelijkheid voor het sturen van 3 digitale en 1 PWM uitgang in functie van opgewekt vermogen of op tijdsbasis.

Stuurt gegevens eventueel door naar een LCD scherm.

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/>

### **Onderdelen**

1 x kabeltje met RJ12 plug

1x ESP32WROOM Devkit

1x BC547

1x R 1K

1x R 10K

1 x R 2K2

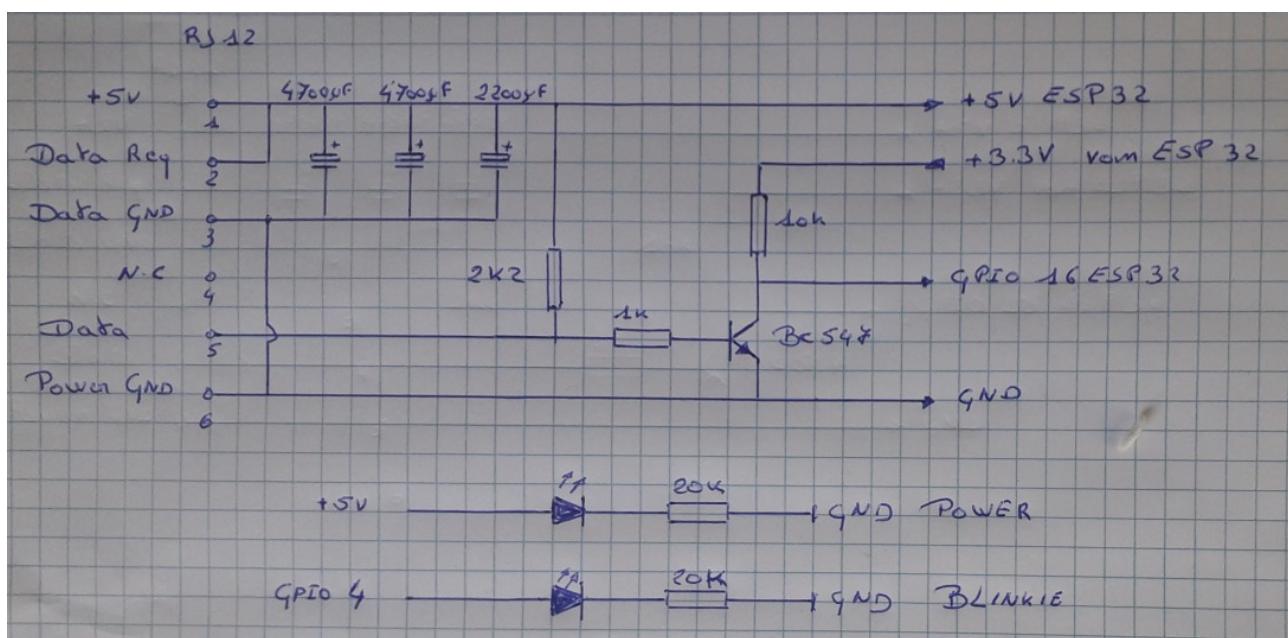
2x R20K

2x LED

2x 4700 uF / 16 Volt

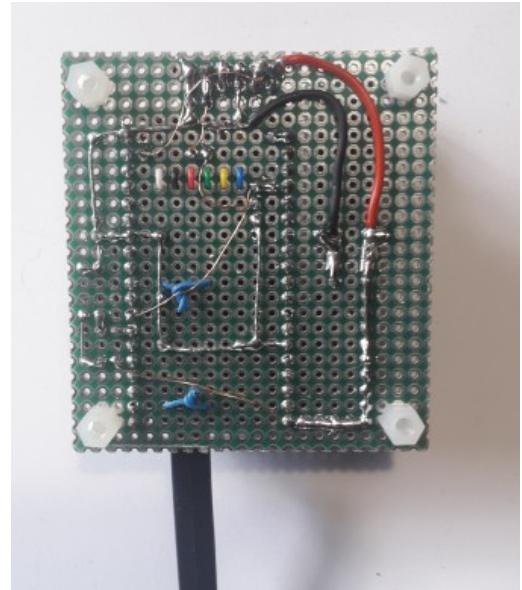
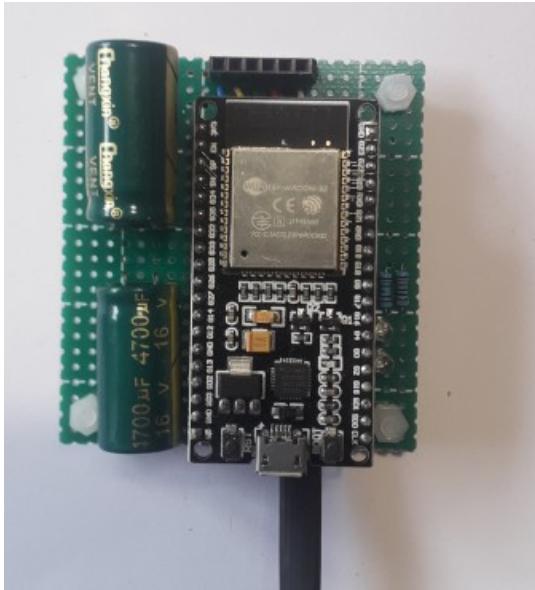
1x 2200 uF / 16 Volt

(eventueel een aparte 5 Volt voeding zie beschrijving)



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. 3 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 3 capaciteiten zijn dan niet nodig. Ontkoppel dan ook de 5V komende van de meter, Pin2 (Data Request) mag met de 5V van de meter verbonden blijven.

**Opmerking in verband met voeding. Ik heb meerdere van deze modules gemaakt en niet alle ESP32's blijven werken met de condensator oplossing, gebruik in dat geval een externe 5V voeding.**



Een ESP32WROOM Devkit module heeft meerdere GND aansluitingen, voor een goede werking is het noodzakelijk om deze allemaal aan te sluiten.

Laad het programma

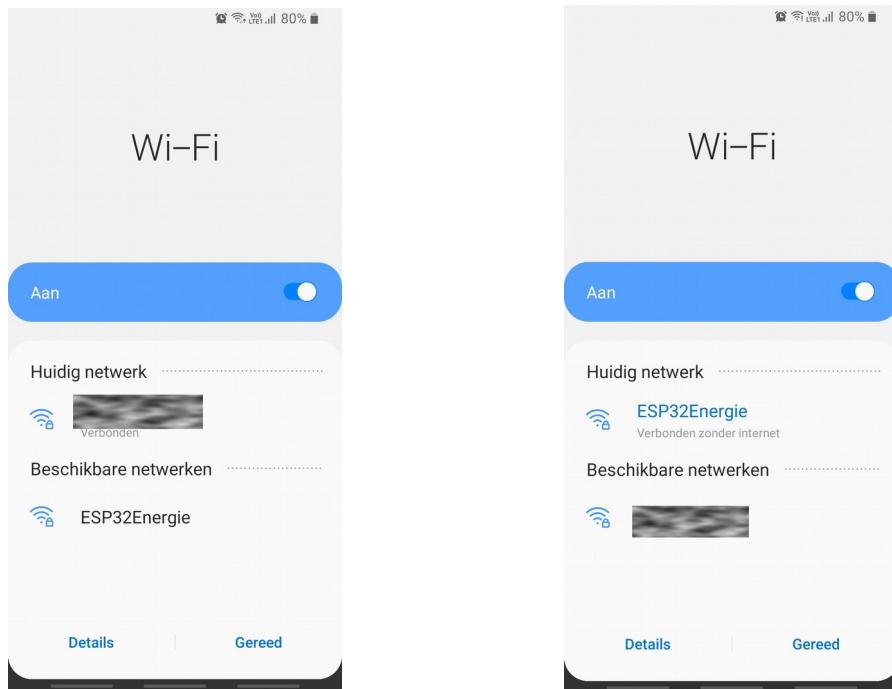
**Slimme\_meter\_esp32\_STA.ino**      bereikbaar via je WiFi netwerk  
of

**Slimme\_meter\_esp32\_AP.ino**      bereikbaar via het ESP32energie netwerk  
in de ESP32WROOM devkit met behulp van de Arduino IDE.

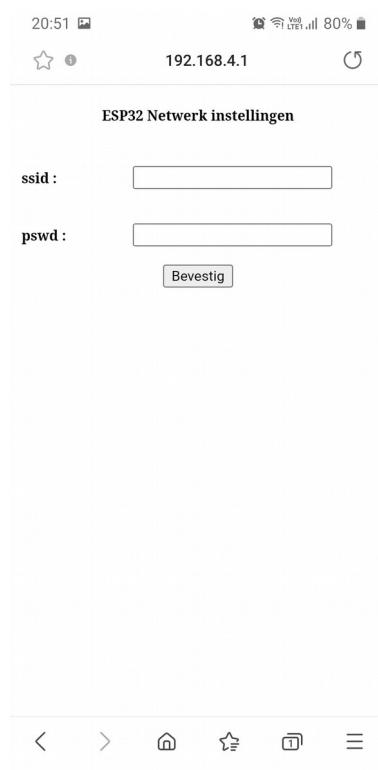
## **Slimme\_meter\_esp32\_STA.ino**

Na het opstarten is er een nieuw netwerk beschikbaar, ESP32energie  
maak verbinding met dit netwerk

netwerk:      ESP32Energie  
paswoord:     ESP32pswd



ga naar pagina 192.168.4.1



Vul de gevraagde gegevens in.

ssid >> naam van jouw WiFi netwerk

pswd >> paswoord van jouw WiFi netwerk

Druk op bevestig.

De ESP32 herstart automatisch. Indien alle WiFi gegevens correct zijn ingevuld zal de ESP32 zich verbinden met jouw WiFi netwerk. Netwerk ESP32Energie is niet meer beschikbaar.

De webpagina is nu beschikbaar op 192.168.1.222 van je eigen WiFi netwerk.

Voor de verdere gemeenschappelijke uitleg zie 2 bladzijden verder.

12:02      95% ■

**ESP32 Slimme Meter Interface**

**Verbruik gegevens**

|                        |              |
|------------------------|--------------|
| Totaal electriciteit : | 6494.462 KWh |
| Totaal injectie :      | 2132.358 KWh |
| Verbruik nu :          | 0.000 KW     |
| Injectie nu :          | 0.276 KW     |
| Totaal gas :           | 2590.163 m3  |

**Tijd**  
12:02

**Relais schakelwaarden**

Relais 1

|       |       |        |    |       |       |      |   |
|-------|-------|--------|----|-------|-------|------|---|
| KW:   | 2.00  | V:     | 10 | Tijd: | 24:00 | A/M: | 0 |
| [ - ] | [ + ] | [ OK ] |    |       |       |      |   |

**PWM sturing instellen**

|        |      |    |       |    |       |      |   |
|--------|------|----|-------|----|-------|------|---|
| KW:    | 0.00 | I: | 24:00 | O: | 00:00 | A/M: | 0 |
| [ OK ] |      |    |       |    |       |      |   |

**Huidige relais sturing**

|               |     |
|---------------|-----|
| Relais 1 :    | 0   |
| Relais 2 :    | 0   |
| Relais 3 :    | 0   |
| PWM sturing : | 0 % |

**Ingeven MAC address**

MAC address Display

|       |       |        |    |    |    |
|-------|-------|--------|----|----|----|
| 30    | ae    | a4     | 0d | 69 | b8 |
| [ - ] | [ + ] | [ OK ] |    |    |    |

thieu-h55 april 2022

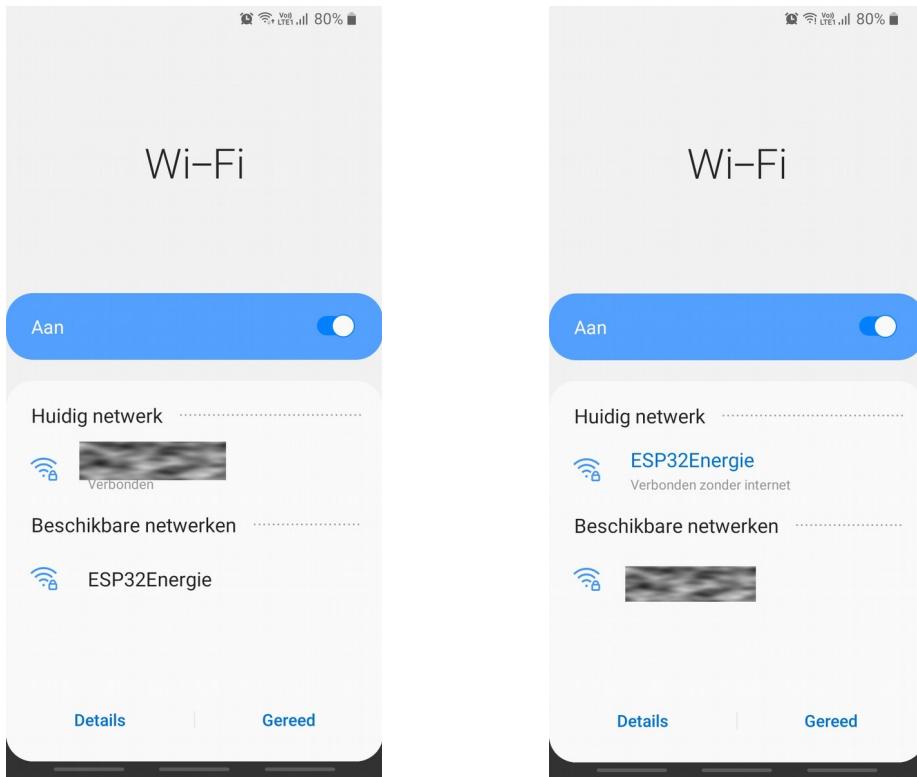
## **Slimme\_meter\_esp32\_AP.ino**

Na het opstarten is er een nieuw netwerk beschikbaar, ESP32energie

maak verbinding met dit netwerk

netwerk: ESP32Energie

paswoord: ESP32pswd



Ga naar pagina 192.168.4.1 van het ESP32energie netwerk

12:02

95%

## ESP32 Slimme Meter Interface

### Verbruik gegevens

Totaal elektriciteit : 6494.462 kWh  
 Totaal injectie : 2132.358 kWh  
 Verbruik nu : 0.000 kW  
 Injectie nu : 0.276 kW  
 Totaal gas : 2590.163 m<sup>3</sup>

### Tijd

12:02

### Relais schakelwaarden

|          |
|----------|
| Relais 1 |
|----------|

KW: 2.00 V: 10 Tijd: 24:00 A/M: 0

- + OK

### PWM sturing instellen

KW: 0.00 1: 24:00 0: 00:00 A/M: 0

OK

### Huidige relais sturing

Relais 1 : 0  
 Relais 2 : 0  
 Relais 3 : 0  
 PWM sturing : 0 %

### Ingeven MAC address

|                     |
|---------------------|
| MAC address Display |
|---------------------|

30 ae a4 0d 69 b8

- + OK

Het gedeelte dat we nu kunnen gebruiken is

### ESP32 Slimme Meter Interface

#### Verbruik gegevens

|                        |              |
|------------------------|--------------|
| Totaal electriciteit : | 6494.462 KWh |
| Totaal injectie :      | 2132.358 KWh |
| Verbruik nu :          | 0.000 KW     |
| Injectie nu :          | 0.276 KW     |
| Totaal gas :           | 2590.163 m3  |

#### Tijd

12:02

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 sluiipverbruik mee opsporen en dat is meer als men zou verwachten heb ik gemerkt.

Onder Tijd is er ook een klok, deze wordt ingelezen van de digitale meter.

Voor de andere velden op deze webpagina moet je extra modules installeren. De uitleg voor deze velden kan je terugvinden bij de desbetreffende modules.

## **Slimme\_meter\_esp32\_data\_STA en Slimme\_meter\_esp32\_data\_AP.**

Leest de gegevens van de P1 poort en stuurt deze naar een webpagina.

Mogelijkheid voor het sturen van 3 digitale en 1 PWM uitgang in functie van opgewekt vermogen of op tijdsbasis.

Stuurt gegevens eventueel door naar apart LCD scherm.

Bewaart data op SD kaart dit voor zowel elektriciteitsverbruik, elektriciteitsinjectie en indien aanwezig aardgas. Dit per uur , dag , maand , jaar. Een volledig jaar beslaat +/- 140 KB op een SD kaart. 16GB volstaat voor enkele honderden jaren.

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/>

### **Onderdelen**

1 x kabeltje met RJ12 plug

1x ESP32WROOM Devkit

1x SD card module

[https://nl.aliexpress.com/item/32986457520.html?spm=a2g0o.store\\_pc\\_allProduct.8148356.5.75213ba2DuDQUR&pdp\\_npi=2%40dis%21EUR%21%E2%82%AC%200%2C76%21%E2%82%AC](https://nl.aliexpress.com/item/32986457520.html?spm=a2g0o.store_pc_allProduct.8148356.5.75213ba2DuDQUR&pdp_npi=2%40dis%21EUR%21%E2%82%AC%200%2C76%21%E2%82%AC)

[https://nl.aliexpress.com/item/32986457520.html?spm=a2g0o.store\\_pc\\_allProduct.8148356.5.75213ba2DuDQUR&pdp\\_npi=2%40dis%21EUR%21%E2%82%AC%200%2C52%21%21%21%21%21%402101d64d16664677174821154e0cae%2166794655177%21sh](https://nl.aliexpress.com/item/32986457520.html?spm=a2g0o.store_pc_allProduct.8148356.5.75213ba2DuDQUR&pdp_npi=2%40dis%21EUR%21%E2%82%AC%200%2C52%21%21%21%21%21%402101d64d16664677174821154e0cae%2166794655177%21sh)



1 x SD kaart FAT32 geformatteerd.

1x BC547

3x LED

1x R 1K

1x R 10K

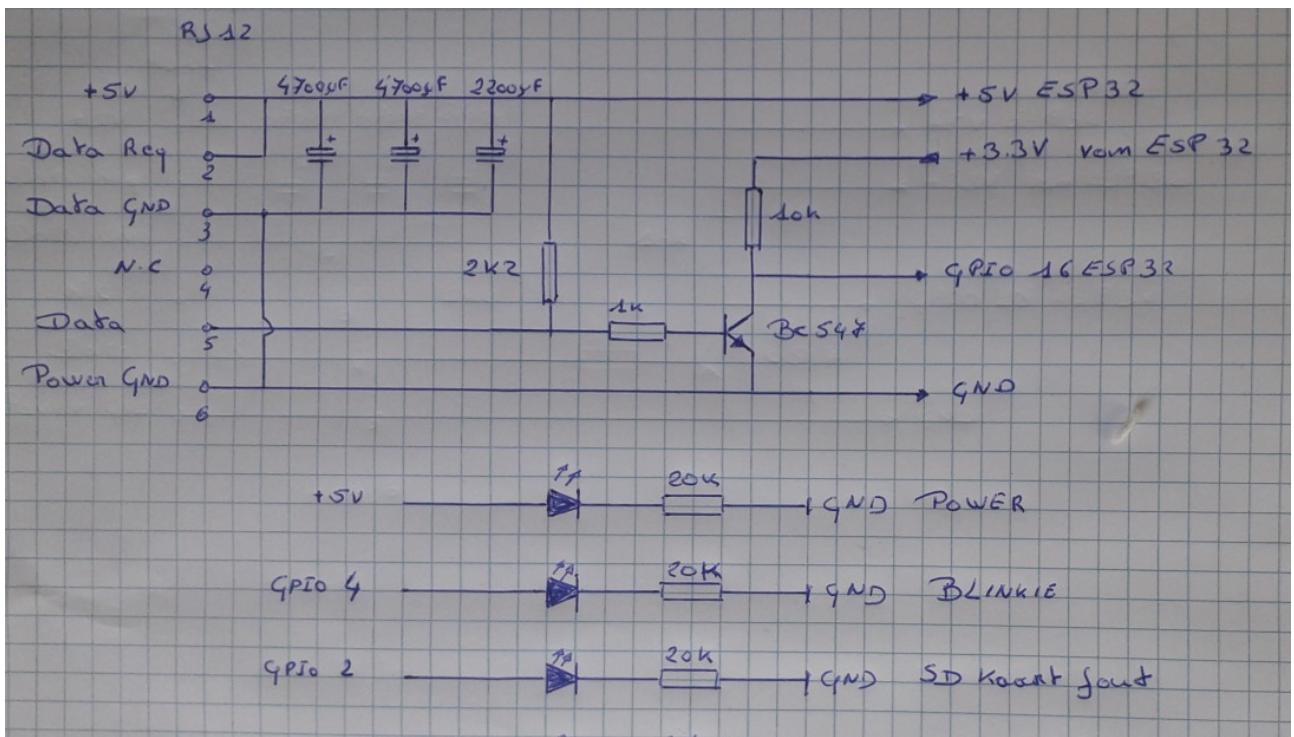
1x R 2K2

3 x R20K

2x 4700 uF / 16 Volt

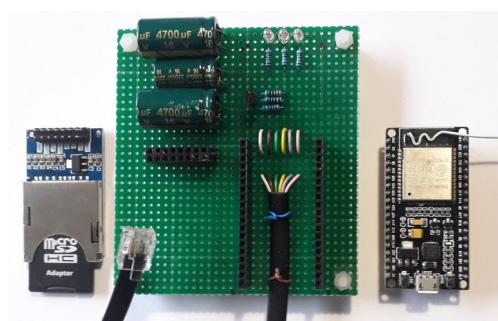
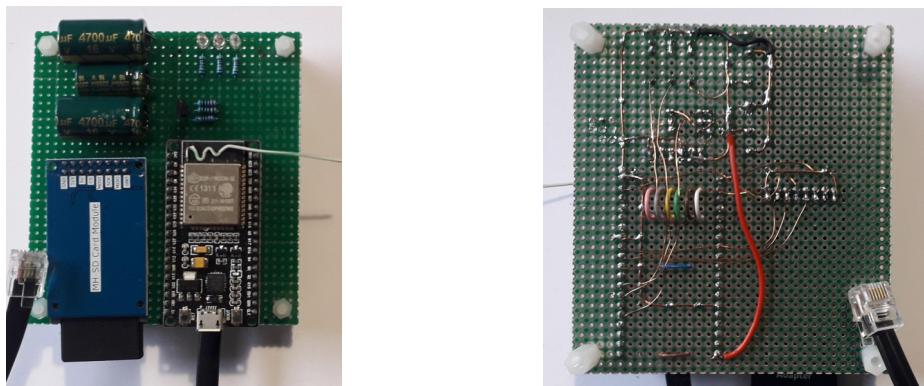
1x 2200 uF / 16 Volt

(eventueel een aparte 5 Volt voeding zie beschrijving)



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. 3 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 3 capaciteiten zijn dan niet nodig. Ontkoppel dan ook de 5V komende van de meter, Pin2 (Data Request) mag met de 5V van de meter verbonden blijven.

**Opmerking in verband met voeding. Ik heb meerdere van deze modules gemaakt en niet alle ESP32's blijven werken met de condensator oplossing, gebruik in dat geval een externe 5V voeding.**



## Aansluitingen ESP32

|        |    |   |
|--------|----|---|
| GPIO23 | >> | MOSI SD kaart   |
| GPIO19 | >> | MISO SD kaart   |
| GPIO18 | >> | SCK SD kaart  |
| GPIO5  | >> | CS SD kaart   |
| GPIO16 | << | BC547 collector uitgang, P1 signaal van digitale meter  |
| GPIO4  | >> | LED (BLINKIE) verandert status bij elke leescyclus data van P1 poort indien OK frequentie 0.5 Hz (1 leescyclus / seconde) |
| GPIO2  | >> | LED SD ERROR. Indien aan, fout bij het lezen van de SD kaart.   |
| 5V     | >> | van P1 poort of externe voeding   |
| 3.3V   | >> | naar 3.3 V SD kaart module  |
| GND    | >> | Van P1 poort / externe voeding (Alle GND aansluitingen ESP32 aansluiten)  |

## Aansluitingen SD kaart module

|      |    |                |
|------|----|----------------|
| 3.3V | << | 3.3V van ESP32 |
| GND  | << | GND            |
| CS   | << | GPIO5 ESP32    |
| SCK  | << | GPIO18 ESP32   |
| MOSI | << | GPIO23 ESP32   |
| MISO | << | GPIO19 ESP32   |

Sluit alles aan zoals hierboven is aangegeven.

Laad nu het programma

|  |  |
|--|--|
| <b>Slimme_meter_esp32_data_STA.ino</b> | gegevens bereikbaar via je WiFi netwerk      |
| of                                     |  |
| <b>Slimme_meter_esp32_data_AP.ino</b>  | gegevens bereikbaar via ESP32Energie netwerk |

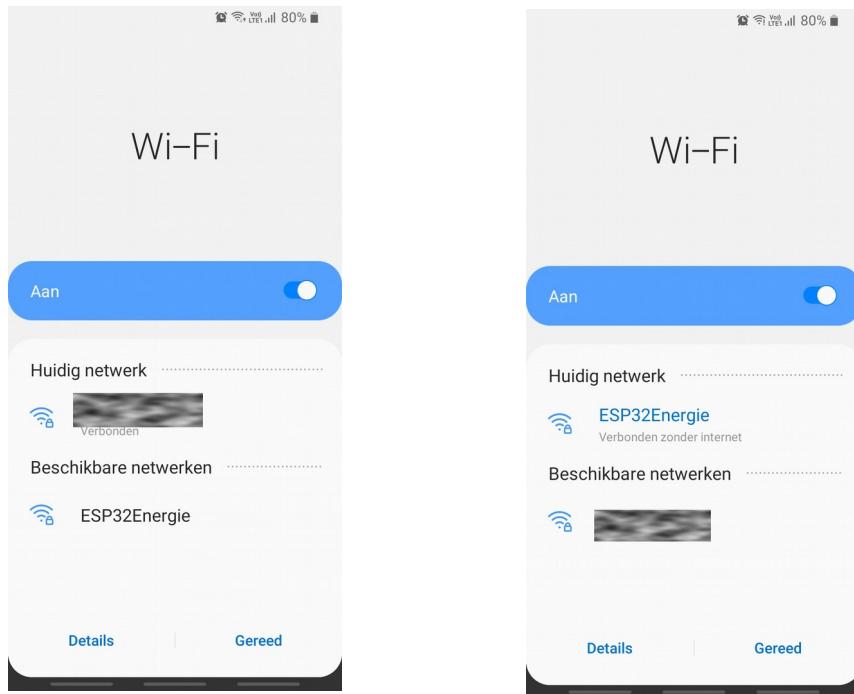
met de Arduino IDE in de ESP32.

Plug de module in de geactiveerde P1 poort, voor de uitvoering met condensators zonder de externe 5V voeding, druk op de ESP32Devkit resetknop (links onder).

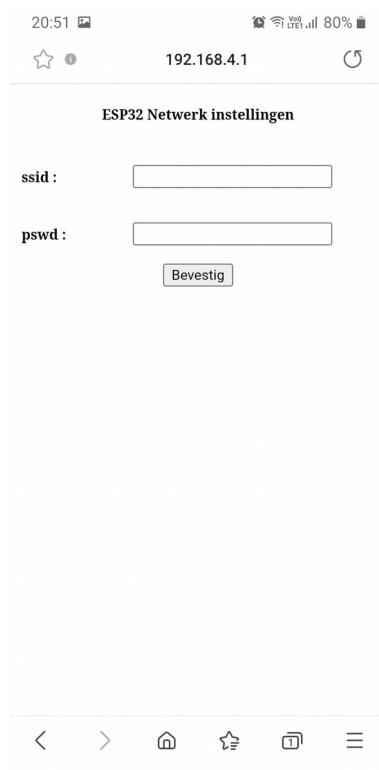
## **Slimme\_meter\_esp32\_data\_STA.ino**

Na het opstarten is er een nieuw netwerk beschikbaar, ESP32energie  
maak verbinding met dit netwerk

netwerk:      ESP32Energie  
paswoord:     ESP32pswd



ga naar pagina 192.168.4.1



Vul de gevraagde gegevens in.

ssid >> naam van jouw Wifi netwerk  
pswd >> paswoord van jouw Wifi netwerk

Druk op bevestig.

De ESP32 herstart automatisch.

Indien alle Wifi gegevens correct zijn ingevuld zal de ESP32 zich verbinden met jouw Wifi netwerk. Netwerk ESP32Energie is nu niet meer beschikbaar.

De webpagina is nu beschikbaar op 192.168.1.222 van je eigen WiFi netwerk.

### ESP32 Slimme Meter Interface

#### Verbruik gegevens

Totaal electriciteit : 6484.781 kWh  
Totaal injectie : 2127.595 kWh  
Verbruik nu : 0.119 kW  
Injectie nu : 0.000 kW  
Totaal gas : 2584.839 m<sup>3</sup>

#### Datum

03 11 2022

#### Tijd

16:16

#### Relais schakelwaarden

Relais 1

KW: 2.00 V: 10 Tijd: 24:00 A/M: 0

- + OK

#### PWM sturing instellen

KW: 1.20 1: 24:00 0: 00:00 A/M: A

OK

#### Huidige relais sturing

Relais 1 : 0  
Relais 2 : 0  
Relais 3 : 0  
PWM sturing : 0

#### Ingeven MAC address

MAC address PWM Sturing

7c 9e bd 06 b4 dc

- + OK

Data weergave

thieu-h55 oktober 2022

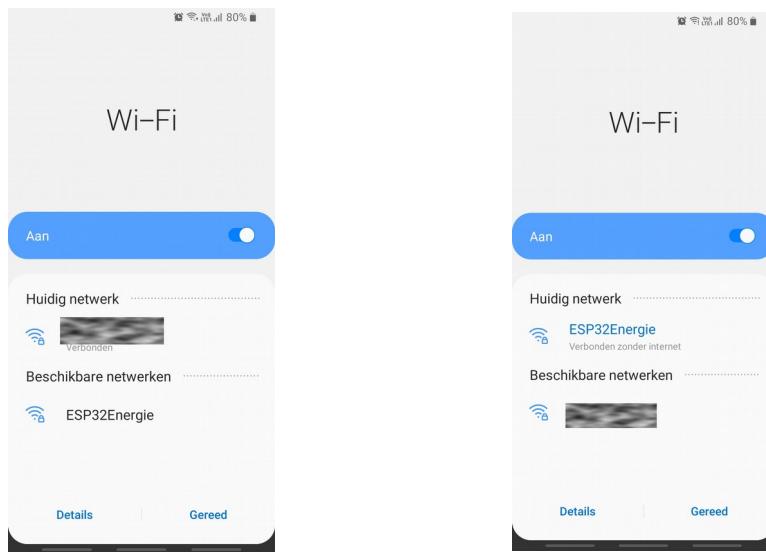
Voor de verdere gemeenschappelijke uitleg zie 2 bladzijden verder.

## **Slimme\_meter\_esp32\_data\_AP.ino**

Na het opstarten is er een nieuw netwerk beschikbaar, namelijk ESP32energie  
maak verbinding met dit netwerk

netwerk:      ESP32Energie

paswoord:     ESP32pswd



Ga naar 192.168.4.1

### **ESP32 Slimme Meter Interface**

#### **Verbruik gegevens**

Totaal electriciteit : 6484.781 KWh  
Totaal injectie : 2127.595 KWh  
Verbruik nu : 0.119 KW  
Injectie nu : 0.000 KW  
Totaal gas : 2584.839 m3

#### **Datum**

03 11 2022

#### **Tijd**

16:16

#### **Relais schakelwaarden**

Relais 1

KW: 2.00 V: 10 Tijd: 24:00 A/M: 0

- + OK

#### **PWM sturing instellen**

KW: 1.20 1: 24:00 0: 00:00 A/M: A

OK

#### **Huidige relais sturing**

Relais 1 : 0  
Relais 2 : 0  
Relais 3 : 0  
PWM sturing : 0

#### **Ingeven MAC address**

MAC address PWM Sturing

7c 9e bd 06 b4 dc

- + OK

Data weergave

## Wat kunnen we er mee?

### ESP32 Slimme Meter Interface

#### Verbruik gegevens

Totaal elektriciteit : 6484.781 KWh  
Totaal injectie : 2127.595 KWh  
Verbruik nu : 0.119 KW  
Injectie nu : 0.000 KW  
Totaal gas : 2584.839 m3

#### Datum

03 11 2022

#### Tijd

16:16

Raadplegen van :  
totaal elektriciteit, gas en injectie.  
momenteel verbruik, injectie elektriciteit.

Datum en tijd komen van de digitale meter.

Voor de andere velden op deze webpagina moet je extra modules installeren. De uitleg voor deze velden kan je terugvinden bij de desbetreffende modules.

Data weergave

thieu-h55 oktober 2022

<Data weergave> brengt je naar de data weergave pagina.

De data die hieronder wordt weergegeven in verschillende schermafdrukken is testdata gegenereerd met een testprogramma.

Schermafdrukken zijn gemaakt met de STA versie van het programma. Indien je gekozen hebt voor de AP versie dan is het adres 192.168.4.1 i.p.v. 192.168.1.222

The screenshot shows a mobile application interface for electricity data. At the top, it displays the time (21:13), signal strength (VoIP LTE1), battery level (98%), and IP address (192.168.1.222). Below this is a header with a star icon, a refresh button, and the text "Data weergave". A "Naar begin pagina" button is located above a row of date selection buttons (21, 10, 2024). An "OK" button is positioned below the date buttons. The main content area is titled "Periode : 21 - 10 - 2024" and contains a table with columns: "uur" (hour), "verbruik kWh" (consumption kWh), "injectie kWh" (injection kWh), and "gas m3". The table lists data for each hour from 00 to 23, followed by a "totaal" row. At the bottom, there are navigation icons for back, forward, home, and menu.

| uur    | verbruik kWh | injectie kWh | gas m3 |
|--------|--------------|--------------|--------|
| 00     | 0.385        | 0.330        | 0.275  |
| 01     | 0.392        | 0.335        | 0.280  |
| 02     | 0.399        | 0.342        | 0.285  |
| 03     | 0.406        | 0.347        | 0.290  |
| 04     | 0.413        | 0.354        | 0.295  |
| 05     | 0.420        | 0.361        | 0.300  |
| 06     | 0.427        | 0.365        | 0.305  |
| 07     | 0.434        | 0.372        | 0.310  |
| 08     | 0.441        | 0.377        | 0.315  |
| 09     | 0.448        | 0.384        | 0.320  |
| 10     | 0.455        | 0.389        | 0.325  |
| 11     | 0.462        | 0.396        | 0.330  |
| 12     | 0.469        | 0.403        | 0.335  |
| 13     | 0.476        | 0.407        | 0.340  |
| 14     | 0.483        | 0.414        | 0.345  |
| 15     | 0.490        | 0.419        | 0.350  |
| 16     | 0.497        | 0.426        | 0.355  |
| 17     | 0.504        | 0.431        | 0.360  |
| 18     | 0.511        | 0.438        | 0.365  |
| 19     | 0.518        | 0.445        | 0.370  |
| 20     | 0.525        | 0.450        | 0.375  |
| 21     | 0.531        | 0.457        | 0.380  |
| 22     | 0.539        | 0.462        | 0.385  |
| 23     | 0.546        | 0.469        | 0.390  |
| totaal | 11.172       | 9.575        | 7.980  |

Met het vakje <Naar begin pagina> ga je terug naar de hoofdpagina.

Met de datum vakjes kies je welke periode je wil opvragen.

Onder het vakje <OK> staat welke periode wordt weergegeven.

Is zowel dag, maand en jaar ingevuld dan krijg je de 24u data van die bepaalde dag.

Verbruik weergeven van een periode die nog niet is afgesloten gaat niet.

Indien je vandaag om 15u15 de 24u verbruiksgegevens zou opvragen van vandaag dan krijg je alleen de gegevens tot en met 14u.

Verversen met het cirkelvormig pijltje ververst alleen de pagina, om extra data te krijgen indien je een uurtje verder bent, moet je op <OK> onder de datum drukken.

**Data weergave**[Naar begin pagina](#)

00    10    2024

OK

Periode : 10 - 2024

| dag | verbruik kWh | injectie kWh | gas m3 |
|-----|--------------|--------------|--------|
| 01  | 7.812        | 6.696        | 5.580  |
| 02  | 7.981        | 6.840        | 5.700  |
| 03  | 8.148        | 6.984        | 5.820  |
| 04  | 8.316        | 7.128        | 5.940  |
| 05  | 8.485        | 7.272        | 6.060  |
| 06  | 8.652        | 7.416        | 6.180  |
| 07  | 8.821        | 7.560        | 6.300  |
| 08  | 8.989        | 7.704        | 6.420  |
| 09  | 9.156        | 7.847        | 6.540  |
| 10  | 9.324        | 7.992        | 6.660  |
| 11  | 9.492        | 8.136        | 6.780  |
| 12  | 9.661        | 8.281        | 6.900  |
| 13  | 9.828        | 8.425        | 7.020  |
| 14  | 9.996        | 8.568        | 7.140  |
| 15  | 10.164       | 8.711        | 7.260  |
| 16  | 10.332       | 8.855        | 7.380  |
| 17  | 10.500       | 9.000        | 7.500  |
| 18  | 10.668       | 9.145        | 7.620  |
| 19  | 10.836       | 9.288        | 7.740  |
| 20  | 11.004       | 9.431        | 7.860  |
| 21  | 11.172       | 9.575        | 7.980  |
| 22  | 11.339       | 9.720        | 8.100  |
| 23  | 11.508       | 9.865        | 8.220  |
| 24  | 11.676       | 10.009       | 8.340  |
| 25  | 11.844       | 10.153       | 8.460  |
| 26  | 12.011       | 10.296       | 8.580  |
| 27  | 12.180       | 10.440       | 8.700  |
| 28  | 12.348       | 10.584       | 8.820  |
| 29  | 12.515       | 10.728       | 8.940  |
| 30  | 12.684       | 10.872       | 9.060  |
| 31  | 12.852       | 11.016       | 9.180  |

dag op "0" geeft als resultaat de dag gegevens van die maand in dat jaar.

21:14

LTE1 98%



i

192.168.1.222



## Data weergave

[Naar begin pagina](#)

00 00 2024

OK

Periode : 2024

| maand  | verbruik kWh | injectie kWh | gas m3   |
|--------|--------------|--------------|----------|
| 01     | 273.420      | 234.360      | 195.300  |
| 02     | 278.629      | 238.824      | 199.020  |
| 03     | 283.836      | 243.289      | 202.740  |
| 04     | 289.044      | 247.752      | 206.460  |
| 05     | 294.252      | 252.217      | 210.180  |
| 06     | 299.460      | 256.680      | 213.899  |
| 07     | 304.672      | 261.144      | 217.620  |
| 08     | 309.879      | 265.608      | 221.340  |
| 09     | 315.086      | 270.071      | 225.060  |
| 10     | 320.292      | 274.536      | 228.780  |
| 11     | 325.500      | 279.000      | 232.500  |
| 12     | 330.707      | 283.464      | 236.220  |
| totaal | 3624.778     | 3106.945     | 2589.119 |



Dag en maand op “0” geeft de maandgegevens van dat jaar als resultaat.

21:15

VoIP LTE1.11 | 98%

## Data weergave

[Naar begin pagina](#)

00 00 0000

OK

Periode : vorige jaren (max 25)

| jaar | verbruik kWh | injectie kWh | gas m3 |
|------|--------------|--------------|--------|
| 1997 |              |              |        |
| 1998 |              |              |        |
| 1999 |              |              |        |
| 2000 |              |              |        |
| 2001 |              |              |        |
| 2002 |              |              |        |
| 2003 |              |              |        |
| 2004 |              |              |        |
| 2005 |              |              |        |
| 2006 |              |              |        |
| 2007 |              |              |        |
| 2008 |              |              |        |
| 2009 |              |              |        |
| 2010 |              |              |        |
| 2011 |              |              |        |
| 2012 |              |              |        |
| 2013 |              |              |        |
| 2014 |              |              |        |
| 2015 |              |              |        |
| 2016 |              |              |        |
| 2017 |              |              |        |
| 2018 |              |              |        |
| 2019 |              |              |        |
| 2020 |              |              |        |
| 2021 |              |              |        |



Dag, maand en jaar op “0” geeft als resultaat de gegevens van de laatste 25 jaar. Het programma is pas vanaf oktober 2022 in dienst, voor resultaat hier zal je nog even geduld moeten hebben

## Aanpassen programma aan andere meters.

Instellingen seriële poort doe je in de void setup()

```
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
```

De gewenste data uit de datastroombus halen doe je hier (helemaal op het laatst van het programma).

```
buffer_data += lees_byte;
if(lees_byte == 0x0a){
    if((buffer_data.substring(4,9)) == "1.0.0"){
        jaar_int = ((buffer_data.substring(10,12)).toInt()) + 2000;
        maand_int = (buffer_data.substring(12,14)).toInt();
        dag_int = (buffer_data.substring(14,16)).toInt();
        uren_int = (buffer_data.substring(16,18)).toInt();
        minuten_int = (buffer_data.substring(18,20)).toInt();
    }
    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"){
        kwh_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 = "";
}
```

pas eventueel de substring gegevens aan als de data bij andere meters op een andere plaats staat.

De data uitgang van deze digitale meter moet geïnverteerd en op 3.3V gebracht worden, zie BC547. Voor meters waarvan de uitgang niet geïnverteerd moet worden volstaat het om pin 5 van de RJ12 plug met een weerstand van 2K2 met de **3.3V** van de ESP32 te verbinden en pin 5 ook te verbinden met GPIO16

|            |    |        |       |                |
|------------|----|--------|-------|----------------|
| Pin 5 RJ12 | >> | R 2K2  | <<    | 3.3V van ESP32 |
|            | >> | GPIO16 | ESP32 |                |

## Display module

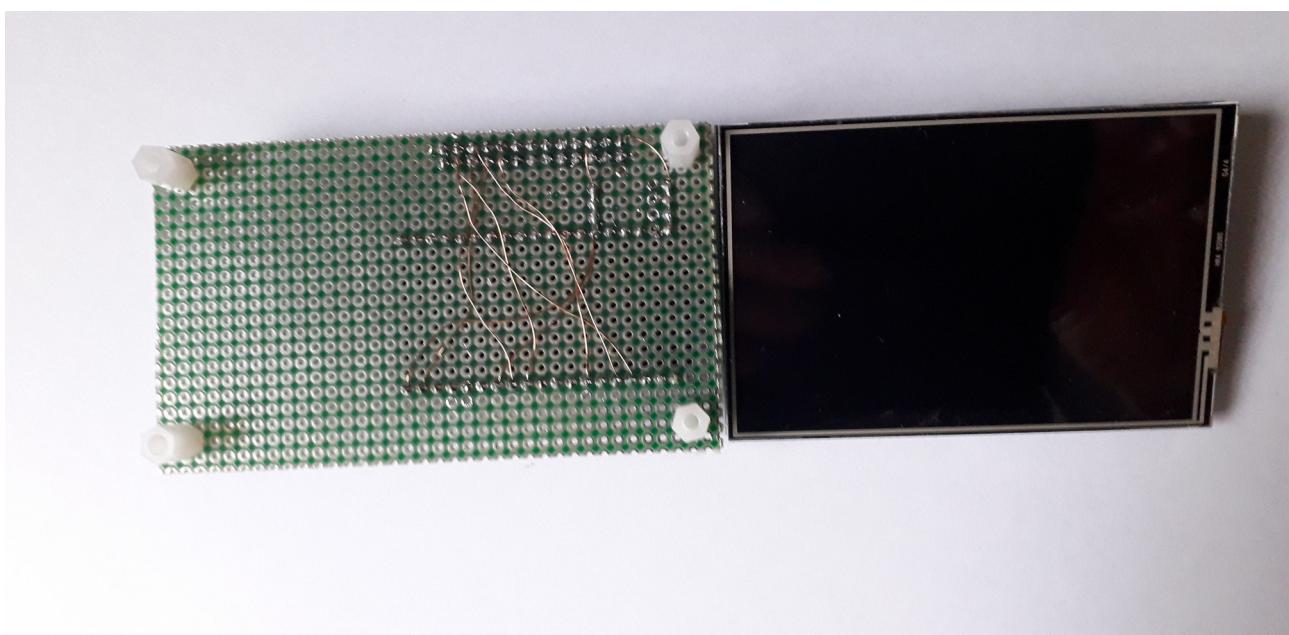
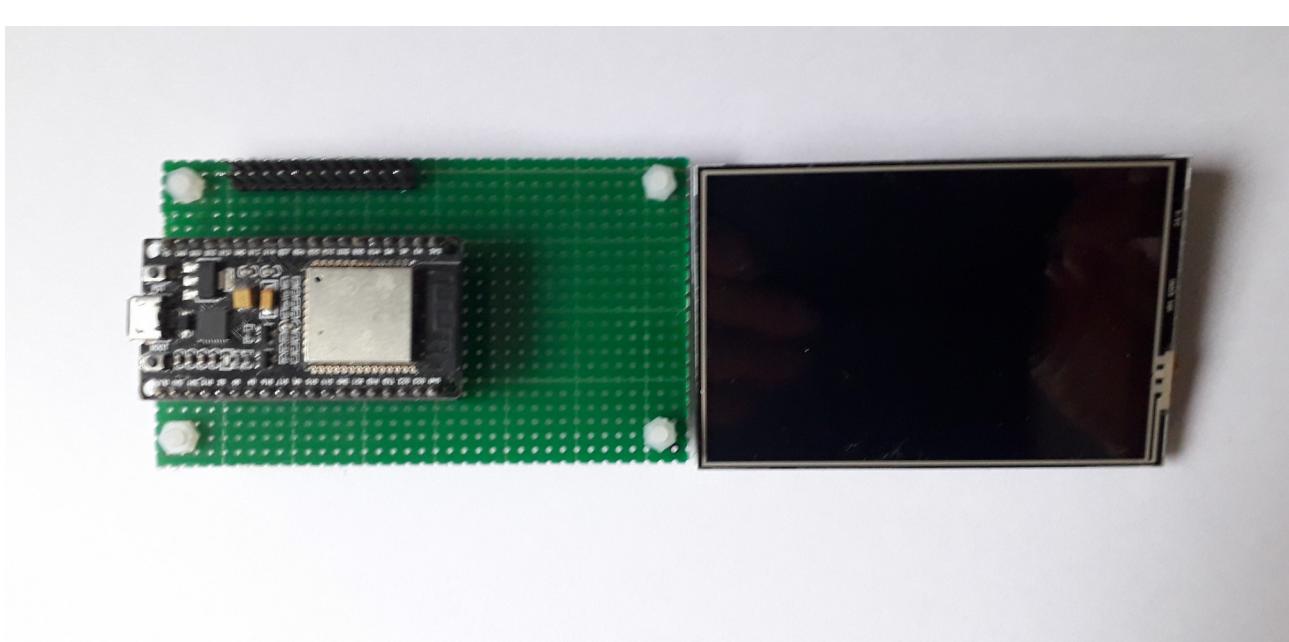
### Onderdelen

1x ESP32WROOM Devkit

1x 3.5" Raspberry LCD Display.

[https://nl.aliexpress.com/item/32605410449.html?spm=a2g0o.order\\_list.0.0.21ef79d2ul1MHA&gatewayAdapt=glo2nld](https://nl.aliexpress.com/item/32605410449.html?spm=a2g0o.order_list.0.0.21ef79d2ul1MHA&gatewayAdapt=glo2nld)

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    | P7            |  |
|  |               |      | RX  | 10   | 9             | GND  |
|  |               |      | P1  | 12   | 11            | P0   |
| Choose the command/data register (Register Select)                     | LCD-RS        | GND  | 14  | 13   | P2            |  |
| Reset  | RST           | P4   | 16  | 15   | P3            |  |
| chip select signal; select LCD when it's low level.                    | LCD_CS LCD    | P5   | 18  | 17   | 3V3           |  |
| Touch panel chip select signal; select touch panel when it's low level | TP_CS         | GND  | 20  | 19   | MO            | LCD-SI/TP_SI                                     |
|  |               | P6   | 22  | 21   | MI            | LCD display/ SPI data input of the touch panel   |
|  |               | CE0  | 24  | 23   | TP_SO         | SPI data output of the touch panel               |
|  |               | CE1  | 26  | 25   | SCK           | LCD_SCK/TP_SCK                                   |
|  |               |      |     |      |               | LCD display/ SPI clock signal of the touch panel |
|  |               |      |     |      |               | 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

Het MAC adres verschijnt in de terminal.

Maak verbinding met de interface module.

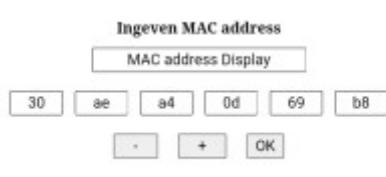
Indien je gekozen hebt voor de \_STA versie, doe je dit via je WiFi netwerk webadres 192.168.1.222

Indien je gekozen hebt voor de \_AP versie via het ESP32 eigen netwerk.

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(1) MAC adres is er een automatische herstart van de interface module.**

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 x
1 // This header file contains a list of user setup files and defines which one the
2 // compiler uses when the IDE performs a Verify/Compile or Upload.
3 //
4 // Users can create configurations for different Espressif boards and TFT displays.
5 // This makes selecting between hardware setups easy by "uncommenting" one line.
6 //
7 // The advantage of this hardware configuration method is that the examples provided
8 // with the library should work with different setups immediately without any other
9 // changes being needed. It also improves the portability of users sketches to other
10 // hardware configurations and compatible libraries.
11 //
12 // Create a shortcut to this file on your desktop to permit quick access for editing.
13 // Re-compile and upload after making and saving any changes to this file.
14 //
15 // Customised User_Setup files are stored in the "User_Setups" folder.
16
17 #ifndef USER_SETUP_LOADED // Lets PlatformIO users define settings in
18 // platformio.ini, see notes in "Tools" folder.
19
20 // Only ONE line below should be uncommented. Add extra lines and files as needed.
21
22 //#include <User_Setup.h> // Default setup is root library folder
23
24 //#include <User_Setups/Setup1_ILI9341.h> // Setup file configured for my ILI9341
25 //#include <User_Setups/Setup2_ST7735.h> // Setup file configured for my ST7735
26 //#include <User_Setups/Setup3_ILI9163.h> // Setup file configured for my ILI9163
27 //#include <User_Setups/Setup4_S6D02A1.h> // Setup file configured for my S6D02A1
28 //#include <User_Setups/Setup5_RPi_ILI9486.h> // Setup file configured for my stock RPi TFT
29 //#include <User_Setups/Setup6_RPi_Wr_ILI9486.h> // Setup file configured for my modified RPi TFT
30 //#include <User_Setups/Setup7_ST7735_128x128.h> // Setup file configured for my ST7735 128x128 display
31 //#include <User_Setups/Setup8_ILI9163_128x128.h> // Setup file configured for my ILI9163 128x128 display
32 //#include <User_Setups/Setup9_ST7735_Overlap.h> // Setup file configured for my ST7735
33 //#include <User_Setups/Setup10_RPi_touch_ILI9486.h> // Setup file configured for ESP8266 and RPi TFT with touch
34
35 #include <User_Setups/Setup11_RPi_touch_ILI9486.h> // Setup file configured for ESP32 and RPi TFT with touch
36 //#include <User_Setups/Setup12_M5Stack.h> // Setup file for the ESP32 based M5Stack
37 //#include <User_Setups/Setup13_ILI9481_Parallel.h> // Setup file for the ESP32 with parallel bus TFT
38 //#include <User_Setups/Setup14_ILI9341_Parallel.h> // Setup file for the ESP32 with parallel bus TFT
39 //#include <User_Setups/Setup15_HX8357D.h> // Setup file configured for HX8357D (untested)
40 //#include <User_Setups/Setup16_ILI9488_Parallel.h> // Setup file for the ESP32 with parallel bus TFT
```

Laad het programma **slimme\_meter\_esp32\_display.ino** met behulp van de Arduino IDE in de ESP32WROOM Devkit module.

De interface stuurt elke 10 seconden data naar de display. Na max 10 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 hoog vermogen **5V relais**

[https://nl.aliexpress.com/item/4000185959463.html?spm=a2g0o.order\\_list.0.0.43f479d2DHQrtK&gatewayAdapt=glo2nld](https://nl.aliexpress.com/item/4000185959463.html?spm=a2g0o.order_list.0.0.43f479d2DHQrtK&gatewayAdapt=glo2nld)



Relais zou volgens beschrijving 30A kunnen schakelen. Niet geprobeerd, maar aangezien een huishoudtoestel bij opstart slechts een laag vermogen verbruikt, is dat niet ook niet nodig. Getest op wasmachine en relais doet wat hij moet doen.

1x externe voeding 5V

Hoe aansluiten

ESP32

ESP32 5V

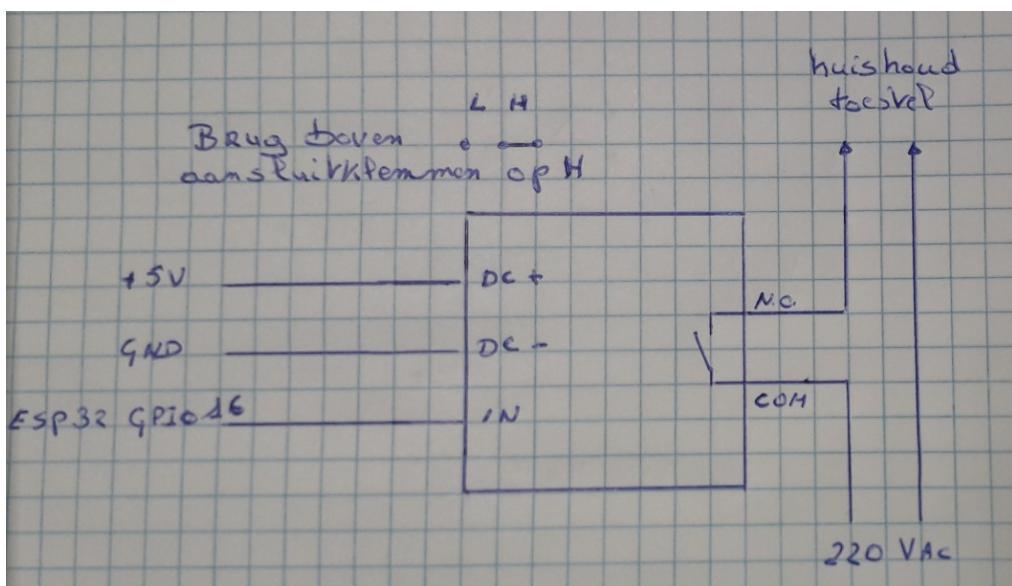
naar externe 5V

ESP32 GND

naar externe GND (alle GND's aansluiten)

ESP32 GPIO16

zie schema hieronder



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**

Het MAC adres verschijnt in de terminal.

Maak verbinding met de interface module.

Indien je gekozen hebt voor de \_STA versie, doe je dit via je WiFi netwerk webadres 192.168.1.222

Indien je gekozen hebt voor de \_AP versie via het ESP32 eigen netwerk.

Wifi netwerk                   ESP32Energie

Paswoord                       ESP32pswd

webadres 192.168.4.1

Ingeven MAC address  
MAC address Display  
30 ae a4 0d 69 b8  
- + OK

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(1)* MAC adres is er een automatische herstart van de interface module.**

Laad het programma **esp32\_relais.ino** 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

Relais schakelwaarden  
Relais 1  
KW: 2.00 V: 10 Tijd: 24:00 A/M: A  
- + OK

Met behulp van de + / – toetsen kies welke relaismodule je wilt instellen en druk op OK

Nu zijn er verschillende velden die je kan instellen.

Het gemakkelijkste eerst

**Automatisch / Manueel**

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

Als je niet wil dat een toestel ingeschakeld wordt op geïnjecteerd vermogen vul dan hier een waarde in die hoger ligt dan wat de zonnepanelen kunnen leveren.

**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 slecht weer dagen kan je de uitgang ook op tijd laten schakelen. Je moet dan wel eerst de tijd instellen.

Als je niet wil dat een toestel 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

|                          |                         |
|--------------------------|-------------------------|
| <b>Verbruik gegevens</b> |                         |
| Totaal electriciteit :   | 5930.162 KWh            |
| Totaal injectie :        | 0.213 KWh               |
| Verbruik nu :            | 0.158 KW                |
| Injectie nu :            | 0.000 KW                |
| Totaal gas :             | 2524.059 m <sup>3</sup> |
| Tijd                     |                         |
| 00:00                    |                         |
| OK                       |                         |

Voor huishoudtoestellen geldt over het algemeen de regel dat die na een spanningsuitval hun programma verder zetten. Dit kunnen we gebruiken om de machines automatisch te starten.

Stuur uitgang van de relais dat het huishoudtoestel bedient via **A/M : 1**. Zet het huishoudtoestel klaar en druk op start machine. Stuur na een korte tijd relais via **A/M : 0** en dan **A/M : A**. Huishoudtoestel zal dan starten nadat de voorwaarden ingevuld in de verschillende vakjes voldaan zijn.

## PWM uitgangsmodule

### Onderdelen

1x ESP32WROOM Devkit

1x R 1K

1x BC547

1x 5V voeding

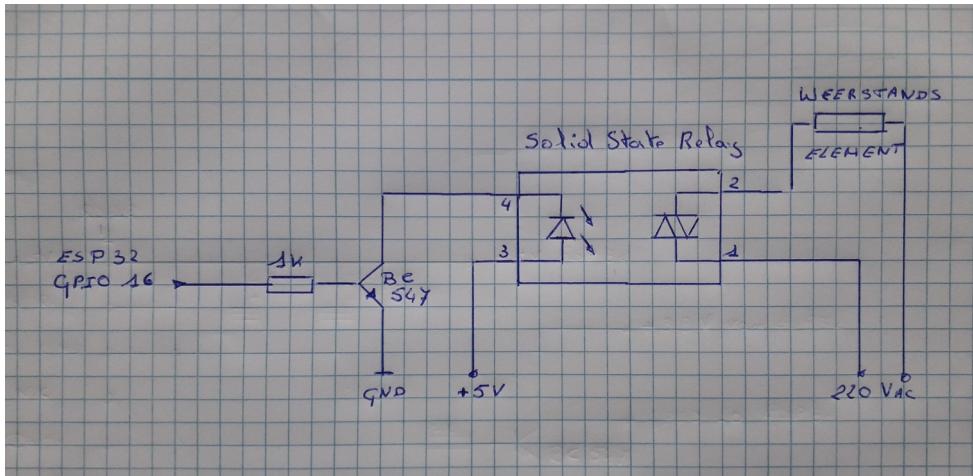
1x solid state relais welke kan gestuurd worden met 5V

[https://nl.aliexpress.com/item/1005003457372056.html?spm=a2g0o.order\\_list.0.0.43f479d2DHQrtK&gatewayAdapt=glo2nld](https://nl.aliexpress.com/item/1005003457372056.html?spm=a2g0o.order_list.0.0.43f479d2DHQrtK&gatewayAdapt=glo2nld)



## Hoe aansluiten

- |              |    |  |
|--------------|----|--|
| GND voeding  | >> | alle GND aansluitingen ESP32Devkit                       |
|              | >> | zie ook schema solid state relay aansluitingen hieronder |
| 5V voeding   | >> | 5V ESP32Devkit   |
|              | >> | zie ook schema solid state relay hieronder               |
| ESP32 GPIO16 | >> | zie schema solid state relay hieronder                   |



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**

Het MAC adres verschijnt in de terminal.

Maak verbinding met de interface module.

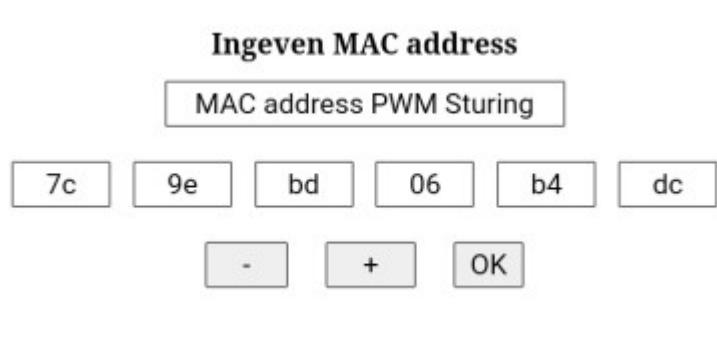
Indien je gekozen hebt voor de \_STA versie, doe je dit via je WiFi netwerk webadres 192.168.1.222

Indien je gekozen hebt voor de \_AP versie via het ESP32 eigen netwerk.

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 is er een automatische herstart van de interface module.**

Laad het programma **esp32\_pwm.ino** 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.

PWM sturing instellen

|                                   |      |    |       |    |       |      |   |
|-----------------------------------|------|----|-------|----|-------|------|---|
| KW:                               | 1.50 | 1: | 24:00 | 0: | 00:00 | A/M: | 0 |
| <input type="button" value="OK"/> |      |    |       |    |       |      |   |

**KW** vul hier het vermogen in van bvb de elektrische boiler die met deze uitgang verbonden is. Dit om te weten hoeveel procent er kan uitgestuurd worden om het beschikbare vermogen te gebruiken.

**Waarde op 0 laten resulteert in een AAN – UIT regeling. Beste is deze waarde zo juist mogelijk in te vullen.**

**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 dan als uur 24 in.

Bij tijd sturing wordt de uitgang maximaal uitgestuurd.

**0:** uitschakeltijd.

**A/M:** A Vermogen naar uitgang wordt gestuurd in functie van het beschikbare opgewekte vermogen.

**Om tijdssturing te kunnen gebruiken moet dit veld eveneens op A staan.**

0 Uitgang is uitgeschakeld

1 Uitgang is volledig uitgestuurd

Dat was het,

*Live long and prosper [Vulcaans: Dif-tor heh smusma] (bron : Wikipedia),*

thieu

## Slimme\_meter\_web\_STA.ino

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

#include <WiFi.h>
#include <Preferences.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

AsyncWebServer server(80);

Preferences pref;

#define RXD2      16
#define TXD2      17
#define BLINKIE   4

bool netwerk;

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

unsigned long nu;
unsigned long wacht_op_netwerk;

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;

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 ssid[40];
char pswd[40];

const char* APSSID = "ESP32Energie";
const char* APPSWD = "ESP32pswd";
const char* STA_SSID = "ssid";
const char* STA_PSWD = "pswd";
```

```

const char energie_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
<iframe style="display:none" name="hidden-form"></iframe>
<title>Energie Beheer</title>
<meta http-equiv="refresh" content="15">
<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: 25px;
    }
</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% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Totaal injectie : </div>
    <div class="midden">%injectie_totaal% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Verbruik nu : </div>
    <div class="midden">%kw_nu% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Injectie nu : </div>
    <div class="midden">%injectie_nu% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Totaal gas : </div>
    <div class="midden">%gas_totaal% &nbsp; m3</div>
</div>
<br>
</small>
<div class="bottom"><h6>thieu september 2022</h6></div>
</body>
</html>
)rawliteral";

```

```

const char netwerk_html[] = R"rawliteral(
<!DOCTYPE HTML>

```

```

<html>
<head>
<iframe style="display:none" name="hidden-form"></iframe>
<title>Energie Beheer</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h5><center><strong>ESP32 Netwerk instellingen</strong></center></h5>
<form action="/get">
<table style="width:100%;">
<tr>
<td style="text-align:left;"><p><small><b><label for="dummy">ssid :</label></b></small></p></td>
<td style="text-align:center;"><input type="text" name="ssid"></td>
</tr>
<tr>
<td style="text-align:left;"><p><small><b><label for="dummy">pswd :</label></b></small></p></td>
<td style="text-align:center;"><input type="text" name="pswd"></td>
</tr>
</table>
<center><input type="submit" value="Bevestig" onclick="ok()"></center>
</form>
<br>
<script>
  function ok(){
    setTimeout(function(){document.location.reload();},250);
  }
</script>
</body>
</html>
)rawliteral";

```

```

String processor(const String& var){
  String temp = " ";
  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);
  }
}

void html_input(){
  server.begin();
  if(netwerk){
    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
      request->send_P(200, "text/html", energie_html, processor);
    });
  }
  if(!netwerk){
    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
      request->send_P(200, "text/html", netwerk_html);
    });
    server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
      bool ssid_ingevuld = false;
      bool pswd_ingevuld = false;
      String netwerk = " ";
      String paswoord = " ";
      char terminator = char(0x0a);
      if(request->hasParam(STA_SSID)){
        netwerk = (request->getParam(STA_SSID)->value());
      }
    });
  }
}

```

```

netwerk = netwerk + String(terminator);
pref.putString("ssid", netwerk);
ssid_ingevuld = true;
}
if(request->hasParam(STA_PSWD)){
    paswoord = (request->getParam(STA_PSWD)->value());
    paswoord = paswoord + String(terminator);
    pref.putString("pswd", paswoord);
    pswd_ingevuld = true;
}
if((ssid_ingevuld) && (pswd_ingevuld)){
    delay(5000);
    ESP.restart();
}
});
}
}

void setup() {
delay(5000);
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
pinMode(BLINKIE, OUTPUT);
digitalWrite(BLINKIE, true);
pref.begin("WiFi_data", false);
if(pref.getString("controle") != "ingesteld"){
    pref.putString("ssid", "dummy");
    pref.putString("pswd", "dummy");
    pref.putString("controle", "ingesteld");
}
ssid_string = pref.getString("ssid");
pswd_string = pref.getString("pswd");
ssid_string.toCharArray(ssid, ssid_string.length());
pswd_string.toCharArray(pswd, pswd_string.length());
WiFi.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pswd);
netwerk = true;
wacht_op_netwerk = millis();
while(WiFi.status() != WL_CONNECTED){
    delay(500);
    if(millis() - wacht_op_netwerk > 15000){
        netwerk = false;
        break;
    }
}
if(netwerk == true){
    IPAddress subnet(WiFi.subnetMask());
    IPAddress gateway(WiFi.gatewayIP());
    IPAddress dns(WiFi.dnsIP(0));
    IPAddress static_ip(192,168,1,222);
    WiFi.disconnect();
    if (WiFi.config(static_ip, gateway, subnet, dns, dns) == false) {
        Serial.println("Configuration failed.");
    }
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, pswd);
    wacht_op_netwerk = millis();
    while(WiFi.status() != WL_CONNECTED){
        delay(500);
        if(millis() - wacht_op_netwerk > 15000){
            netwerk = false;
            break;
        }
    }
}
else{
    WiFi.disconnect();
    WiFi.mode(WIFI_AP);
    WiFi.softAP(APSSID, APPSWD);
}
delay(1000);
html_input();
nu = millis();
}

void loop() {
while(Serial2.available()){
    char lees_byte = Serial2.read();

```

```

if(lees_byte == 0x2f){
    digitalWrite(BLINKIE, (digitalRead(BLINKIE) ^ 1));
    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();
    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);
    }
    buffer_data = "";
}
}

```

## Slimme\_meter\_web\_AP.ino

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

AsyncWebServer server(80);

#define RXD2      16
#define TXD2      17
#define BLINKIE   4

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

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;

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];

const char* APSSID = "ESP32Energie";
const char* APPSWD = "ESP32pswd";
const char* STA_SSID = "ssid";
const char* STA_PSWD = "pswd";

const char energie_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
  <iframe style="display:none" name="hidden-form"></iframe>
  <title>Energie Beheer</title>
  <meta http-equiv="refresh" content="15">
</head>
<body>
  <h1>Slimme Meter</h1>
  <h2>Gegevens</h2>
  <table>
    <tr>
      <td>KWh Dag</td>
      <td>" + String(kwh_dag) + "</td>
    </tr>
    <tr>
      <td>KWh Nacht</td>
      <td>" + String(kwh_nacht) + "</td>
    </tr>
    <tr>
      <td>Gas</td>
      <td>" + String(gas) + "</td>
    </tr>
    <tr>
      <td>Injectie Dag</td>
      <td>" + String(injectie_dag) + "</td>
    </tr>
    <tr>
      <td>Injectie Nacht</td>
      <td>" + String(injectie_nacht) + "</td>
    </tr>
    <tr>
      <td>KW Nu</td>
      <td>" + String(kw_nu) + "</td>
    </tr>
    <tr>
      <td>Injectie Nu</td>
      <td>" + String(injectie_nu) + "</td>
    </tr>
    <tr>
      <td>Totaal</td>
      <td>" + String(kwh_totaal_float) + "</td>
    </tr>
  </table>
  <h2>WIFI</h2>
  <table>
    <tr>
      <td>SSID</td>
      <td>" + String(ssid_string) + "</td>
    </tr>
    <tr>
      <td>PSWD</td>
      <td>" + String(pswd_string) + "</td>
    </tr>
  </table>
</body>
)
```

```

<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: 25px;
}
</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% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Totaal injectie : </div>
    <div class="midden">%injectie_totaal% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Verbruik nu : </div>
    <div class="midden">%kw_nu% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Injectie nu : </div>
    <div class="midden">%injectie_nu% &nbsp; kWh</div>
</div>
<br>
<div class="kader">
    <div class="links">Totaal gas : </div>
    <div class="midden">%gas_totaal% &nbsp; m3</div>
</div>
<br>
</small>
<div class="bottom"><h6>thieu september 2022</h6></div>
</body>
</html>
)rawliteral";

```

```

String processor(const String& var){
    String temp = " ";
    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);
}
}

void html_input(){
server.begin();
server.on("/", HTTP_GET, [](){AsyncWebServerRequest *request){
request->send_P(200, "text/html", energie_html, processor);
});
}

void setup() {
delay(5000);
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
pinMode(BLINKIE, OUTPUT);
digitalWrite(BLINKIE, true);
WiFi.disconnect();
WiFi.mode(WIFI_AP);
WiFi.softAP(APSSID, APPSWD);
delay(1000);
html_input();
nu = millis();
}

void loop() {
while(Serial2.available()){
    char lees_byte = Serial2.read();
    if(lees_byte == 0x2f){
        digitalWrite(BLINKIE, (digitalRead(BLINKIE) ^ 1));
        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();
        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);
}
buffer_data = "";
}
}
```

## Slimme\_meter\_ESP32\_STA.ino

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 */
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/
#include <WiFi.h>
#include <Preferences.h>
#include <esp_now.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

esp_now_peer_info_t peerInfo;
esp_err_t result;

AsyncWebServer server(80);

Preferences pref;

#define RXD2      16
#define TXD2      17
#define BLINKIE   4

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_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];

String ssid_string = "      ";
String pswd_string = "      ";
const char* AP_SSID = "ESP32Energie";
const char* AP_PSWD = "ESP32pswd";
const char* STA_SSID = "ssid";
const char* STA_PSWD = "pswd";
char ssid[40];
char pswd[40];
bool netwerk;
unsigned long wacht_op_netwerk;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
    char macStr[18];
    Serial.print("Packet to: ");
    sprintf(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");
}

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% &nbsp; KWh</div>
</div>
<br>
<div class="kader">
<div class="links">Totaal injectie : </div>
<div class="midden">%injectie_totaal% &nbsp; KWh</div>
</div>
<br>
<div class="kader">
<div class="links">Verbruik nu : </div>
<div class="midden">%kw_nu% &nbsp; KW</div>
</div>
<br>
<div class="kader">
<div class="links">Injectie nu : </div>
<div class="midden">%injectie_nu% &nbsp; KW</div>
</div>
<br>
<div class="kader">
<div class="links">Totaal gas : </div>
<div class="midden">%gas_totaal% &nbsp; m3</div>
</div>
<br><br>
<div class="titel"><b><center>Tijd</center></b></div>
<center><input type="text" style="text-align:center;" value="%tijd%" size=2></center>
<br>
<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>&nbsp;<input type="text" style="text-align:center;" value="%kw_on%" name="input_kw_on" size=1>
&nbsp; <b>V:</b>&nbsp;<input type="text" style="text-align:center;" value="%delay%" name="input_delay" size=1>
&nbsp; <b>Tijd:</b>&nbsp;<input type="text" style="text-align:center;" value="%schakel_tijd%" name="input_schakel_tijd" size=1>
&nbsp; <b>A/M:</b>&nbsp;<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()>

```

```

    &nbsp;&nbsp;&nbsp;
    <input type="submit" name="relais_module_plus" value=" + " onclick="ok()">
    &nbsp;&nbsp;&nbsp;
    <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>&nbsp;<input type="text" style="text-align:center;" value="%pwm_kw%" name="input_pwm_kw" size=1>
&nbsp;<b>1:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_on%" name="input_pwm_tijd_on" size=1>
&nbsp;<b>0:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_off%" name="input_pwm_tijd_off" size=1>
&nbsp;<b>A/M:</b>&nbsp;<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>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_1%" name="input_macx_1" size=1>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_2%" name="input_macx_2" size=1>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_3%" name="input_macx_3" size=1>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_4%" name="input_macx_4" size=1>
    &nbsp;
    <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()">
    &nbsp;&nbsp;&nbsp;
    <input type="submit" name="module_plus" value=" + " onclick="ok()">
    &nbsp;&nbsp;&nbsp;
    <input type="submit" name="module_bevestig" value="OK" onclick="ok()">
</center>
</form>
</small>
<br>
<br>
<br>
<h6><b>thieu-b55 april 2022</b></h6>
<script>
    function ok(){
        setTimeout(function(){document.location.reload();},250);
    }
</script>
</body>

```

```

</html>
)rawliteral";

const char netwerk_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
<iframe style="display:none" name="hidden-form"></iframe>
<title>Energie Beheer</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h5><center><strong>ESP32 Netwerk instellingen</strong></center></h5>
<form action="/get">
<table style="width:100%;">
<tr>
<td style="text-align:left;"><p><small><b><label for="dummy">ssid :</label></b></small></p></td>
<td style="text-align:center;"><input type="text" name="ssid"></td>
</tr>
<tr>
<td style="text-align:left;"><p><small><b><label for="dummy">pswd :</label></b></small></p></td>
<td style="text-align:center;"><input type="text" name="pswd"></td>
</tr>
</table>
<center><input type="submit" value="Bevestig" onclick="ok()"></center>
</form>
<br>
<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();
if(netwerk){
    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(0xa);
        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_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_SCHADEL_TIJD)){
    temp = ((request->getParam(INPUT_SCHADEL_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.relaais = 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.relaais = 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.relaais = 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.relaais = 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.relaais = 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.relaais = 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;
    }
/*
 * tijdsturing uitschakelen bij Manueel 0 tijdens tijdsturing
 */
}

```

```

if(pwm_override == "0"){
    pwm_tijd_gezet = false;
}
}
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_MINUS)){
    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");
broadcastAddress3[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;
}
delay(2000);
ESP.restart();
}
});
}
if(!netwerk){
server.on("/", HTTP_GET, [](AsyncWebRequest *request){
request->send_P(200, "text/html", netwerk_html);
});
server.on("/get", HTTP_GET, [](AsyncWebRequest *request){
bool ssid_ingevuld = false;
bool pswd_ingevuld = false;
String netwerk = " ";
String paswoord = " ";
char terminator = char(0xa);
if(request->hasParam(STA_SSID)){
netwerk = (request->getParam(STA_SSID)->value());
netwerk = netwerk + String(terminator);
}
}
}

```

```

        pref.putString("ssid", netwerk);
        ssid_ingevuld = true;
    }
    if(request->hasParam(STA_PSWD)){
        paswoord = (request->getParam(STA_PSWD)->value());
        paswoord = paswoord + String(terminator);
        pref.putString("pswd", paswoord);
        pswd_ingevuld = true;
    }
    if((ssid_ingevuld) && (pswd_ingevuld)){
        delay(5000);
        ESP.restart();
    }
}
};

void setup() {
delay(5000);
pinMode(BLINKIE, OUTPUT);
digitalWrite(BLINKIE, 0);
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
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", 7);
    pref.putInt("mac2_1", 1);
    pref.putInt("mac2_2", 2);
    pref.putInt("mac2_3", 3);
    pref.putInt("mac2_4", 4);
    pref.putInt("mac2_5", 7);
    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", 0x7c);
    pref.putInt("mac5_1", 0x9e);
    pref.putInt("mac5_2", 0xbd);
    pref.putInt("mac5_3", 0x06);
    pref.putInt("mac5_4", 0xb4);
    pref.putInt("mac5_5", 0xdc);
    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("ssid", "                ");
    pref.putString("pswd", "                ");
    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");
ssid_string = pref.getString("ssid");
ssid_string.toCharArray(ssid, ssid_string.length());
pswd_string = pref.getString("pswd");
pswd_string.toCharArray(pswd, pswd_string.length());

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.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pswd);
netwerk = true;
wacht_op_netwerk = millis();
while(WiFi.status() != WL_CONNECTED){
  delay(500);
  if(millis() - wacht_op_netwerk > 15000){
    netwerk = false;
    break;
  }
}
if(netwerk == true){
  IPAddress subnet(WiFi.subnetMask());
  IPAddress gateway(WiFi.gatewayIP());
  IPAddress dns(WiFi.dnsIP(0));
  IPAddress static_ip(192,168,1,222);
  WiFi.disconnect();
  if (WiFi.config(static_ip, gateway, subnet, dns, dns) == false) {
    Serial.println("Configuration failed.");
  }
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pswd);
  wacht_op_netwerk = millis();
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    if(millis() - wacht_op_netwerk > 15000){
      netwerk = false;
      break;
    }
  }
}
if(netwerk == true){
  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");
}
}
else if(netwerk == false){
    WiFi.disconnect();
    WiFi.mode(WIFI_AP);
    WiFi.softAP(AP_SSID, AP_PSWD);
}
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;
            digitalWrite(BLINKIE, (digitalRead(BLINKIE) ^ 1));
            if(((millis() - nu) > 5000) && (!vijf_seonden)){
                vijf_seonden = 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");
}
}

buffer_data = "";
}

buffer_data += lees_byte;
if(lees_byte == 0x0a){
    if((buffer_data.substring(4,9)) == "1.0.0"){
        uren = (buffer_data.substring(16,18)).toInt();
        minuten = (buffer_data.substring(18,20)).toInt();
    }
    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.relaais = 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.relaais = 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");
}
}
}
```

## **Slimme\_meter\_ESP32\_AP.ino**

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 */
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*/
#include <WiFi.h>
#include <Preferences.h>
#include <esp_now.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

esp_now_peer_info_t peerInfo;
esp_err_t result;

AsyncWebServer server(80);

Preferences pref;

#define RXD2      16
#define TXD2      17
#define BLINKIE   4

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_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;

```



```

}

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% &nbsp; KWh</div>
    </div>
    <br>
    <div class="kader">
      <div class="links">Totaal injectie : </div>
      <div class="midden">%injectie_totaal% &nbsp; KWh</div>
    </div>
    <br>
    <div class="kader">
      <div class="links">Verbruik nu : </div>
      <div class="midden">%kw_nu% &nbsp; KW</div>
    </div>
    <br>
    <div class="kader">
      <div class="links">Injectie nu : </div>
      <div class="midden">%injectie_nu% &nbsp; KW</div>
    </div>
    <br>
    <div class="kader">
      <div class="links">Totaal gas : </div>
      <div class="midden">%gas_totaal% &nbsp; m3</div>
    </div>
    <br><br>
    <div class="titel"><b><center>Tijd</center></b></div>
    <center><input type="text" style="text-align:center;" value="%tijd%" size=2></center>
    <br>
    <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>&nbsp;<input type="text" style="text-align:center;" value="%kw_on%" name="input_kw_on" size=1>
      &nbsp;&nbsp;<b>V:</b>&nbsp;<input type="text" style="text-align:center;" value="%delay%" name="input_delay" size=1>
      &nbsp;<b>Tijd:</b>&nbsp;<input type="text" style="text-align:center;" value="%schakel_tijd%" name="input_schakel_tijd" size=1>
      &nbsp;<b>A/M:</b>&nbsp;<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()">
      &nbsp;&nbsp;&nbsp;
      <input type="submit" name="relais_module_plus" value=" + " onclick="ok()">
      &nbsp;&nbsp;&nbsp;
      <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>&nbsp;<input type="text" style="text-align:center;" value="%pwm_kw%" name="input_pwm_kw" size=1>
&nbsp;<b>1:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_on%" name="input_pwm_tijd_on" size=1>
&nbsp;<b>0:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_off%" name="input_pwm_tijd_off" size=1>
&nbsp;<b>A/M:</b>&nbsp;<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>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_1%" name="input_macx_1" size=1>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_2%" name="input_macx_2" size=1>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_3%" name="input_macx_3" size=1>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_4%" name="input_macx_4" size=1>
&nbsp;
<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()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="module_plus" value=" + " onclick="ok()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="module_bevestig" value="OK" onclick="ok()"/>
</center>
</form>
</small>
<br>
<br>
<br>
<h6><b>thieu-b55 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_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_SCHADEL_TIJD)){
        temp = ((request->getParam(INPUT_SCHADEL_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)){
                    if((relais_module_teller > 0) && (relais_module_teller < 4)){
                        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.relaais = 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.relaais = 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.relaais = 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;
    }
    /*
     * tijdsturing uitchakelen bij Manueel 0 tijdens tijdsturing
     */
    if(pwm_override == "0"){
        pwm_tijd_gezet = false;
    }
}

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_MINUS)){
    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");
            broadcastAddress3[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;
    }
    delay(2000);
    ESP.restart();
}
});
}

void setup() {
delay(5000);
pinMode(BLINKIE, OUTPUT);
digitalWrite(BLINKIE, 0);
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
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", 7);
    pref.putInt("mac2_1", 1);
    pref.putInt("mac2_2", 2);
    pref.putInt("mac2_3", 3);
    pref.putInt("mac2_4", 4);
    pref.putInt("mac2_5", 7);
    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", 0x7c);
    pref.putInt("mac5_1", 0x9e);
    pref.putInt("mac5_2", 0xbd);
    pref.putInt("mac5_3", 0x06);
    pref.putInt("mac5_4", 0xb4);
    pref.putInt("mac5_5", 0xdc);
    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.disconnect();
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;
        digitalWrite(BLINKIE, (digitalRead(BLINKIE) ^ 1));
        if(((millis() - nu) > 5000) && (!vijf_seonden)){
            vijf_seonden = 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_seonden = 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");
}
buffer_data = "";
}
buffer_data += lees_byte;
if(lees_byte == 0x0a){
    if((buffer_data.substring(4,9)) == "1.0.0"){
        uren = (buffer_data.substring(16,18)).toInt();
        minuten = (buffer_data.substring(18,20)).toInt();
    }
    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) {

```



## **Slimme\_meter\_esp32\_data\_STA.ino**

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 */
/*
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
*/
/*
 * 20-10-2022 20:40
 */
#include <WiFi.h>
#include <Preferences.h>
#include <esp_now.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "FS.h"
#include "SD.h"
#include "SPI.h"

esp_now_peer_info_t peerInfo;
esp_err_t result;

AsyncWebServer server(80);

Preferences pref;

#define RXD2      16
#define TXD2      17
#define BLINKIE   4
#define SD_CARD_ERROR 2

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;

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 = "          ";

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_KW_ON = "input_kw_on";
const char* INPUT_OVERRIDE = "input_override";
const char* INPUT_DELAY = "input_delay";
const char* INPUT_SCHADEL_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";

const char* INPUT_DATA_DAG = "input_data_dag";
const char* INPUT_DATA_MAAND = "input_data_maand";
const char* INPUT_DATA_JAAR = "input_data_jaar";
const char* BEVESTIG_PERIODE = "bevestig_periode";

bool dag_dir_bestaat = false;
bool dir_test = false;
bool initialiseren = true;
bool elektriciteit_dag_file_bestaat = false;
bool elektriciteit_jaar_file_bestaat = false;
bool elektriciteit_maand_file_bestaat = false;
bool elektriciteit_uur_file_bestaat = false;
bool file_test = false;
bool gas_dag_file_bestaat = false;
bool gas_jaar_file_bestaat = false;
bool gas_maand_file_bestaat = false;
bool gas_uur_file_bestaat = false;
bool injectie_dag_file_bestaat = false;
bool injectie_jaar_file_bestaat = false;
bool injectie_uur_file_bestaat = false;
bool injectie_maand_file_bestaat = false;
bool jaar_dir_bestaat = false;
bool maand_dir_bestaat = false;
bool pwm_tijd_gezet = false;
bool pwm_tijd_gezet_vorig;
bool relais1_uit;
bool relais2_uit;
bool relais3_uit;
bool vijf_seconden = false;
bool volledige_maand_bool;

char kwh_totaal_float_char[12];
char injectie_totaal_float_char[12];

```

```

char kwh_nu_float_char[12];
char injectie_nu_float_char[12];
char gas_totaal_float_char[12];
char verbruik_char[40];
char aantal_dagen_file_char[18];
char aantal_dagen_char[10];
char module_char[20];
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];
char data_dag_char[4];
char data_maand_char[4];
char data_jaar_char[6];
char weergave_periode_char[12];
char periode_char[25];
char eenheid_char[10];
char totaal_char[10];
char kwh_sd_char[10];
char data_kwh_array_char[32][10];
char data_injectie_array_char[32][10];
char data_gas_array_char[32][10];
char periode_array_char[32][10];
char injectie_sd_char[10];
char gas_sd_char[10];
char tijd_char[12];
char jaar_char[6];
char maand_char[8];
char dag_char[4];
char elektriciteit_jaar_file_char[14];
char injectie_jaar_file_char[14];
char gas_jaar_file_char[14];
char elektriciteit_maand_file_char[14];
char injectie_maand_file_char[14];
char gas_maand_file_char[14];
char elektriciteit_dag_file_char[18];
char injectie_dag_file_char[18];
char gas_dag_file_char[18];
char elektriciteit_uur_file_char[20];
char injectie_uur_file_char[20];
char gas_uur_file_char[20];
char uitsturing_pwm_char[6];
char schakel_tijd_char[12];
char pwm_override_char[8];

float relais1_on;
float relais2_on;
float relais3_on;
float pwm_kw_float;
float data_verbruik_float[32];
float kwh_sd_float;
float injectie_sd_float;
float gas_sd_float;
float uitsturing_pwm_float = 0.0;
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 kwh_nu_float;
float injectie_nu_float;
float verbruik_nu_float;
float gas_totaal_float;
float verbruik_pwm_float;

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;
int uren_int;
int minuten_int;
int seconden_int;
int dag_int;
int maand_int;
int jaar_int;
int uren_vorig_int;
int dag_vorig_int;
int maand_vorig_int;
int jaar_vorig_int;
int data_dag_int = 20;
int data_maand_int = 10;
int data_jaar_int = 2022;
int uitsturing_pwm_int = 0;
int module_teller = 0;
int relais_module_teller = 0;
int dagen_int;

String buffer_data = "      ";
String kwh_dag = "      ";
String kwh_nacht = "      ";
String injectie_dag = "      ";
String injectie_nacht = "      ";
String kwh_nu = "      ";
String injectie_nu = "      ";
String gas = "      ";
String kwh_totaal_string = "      ";
String injectie_totaal_string = "      ";
String gas_totaal_float_string = "      ";
String verbruik_string = "      ";
String aantal_dagen_file_string = "      ";
String aantal_dagen_string = "      ";
String relais1_override = "      ";
String relais2_override = "      ";
String relais3_override = "      ";
String weergave_periode_string = "      ";
String kwh_sd_string = "      ";
String injectie_sd_string = "      ";
String gas_sd_string = "      ";
String uren_string = "      ";
String minuten_string = "      ";
String tijd_string = "      ";
String jaar_string = "      ";
String maand_string = "      ";
String dag_string = "      ";
String elektriciteit_jaar_file_string = "      ";
String injectie_jaar_file_string = "      ";
String gas_jaar_file_string = "      ";
String elektriciteit_maand_file_string = "      ";
String injectie_maand_file_string = "      ";
String gas_maand_file_string = "      ";
String elektriciteit_dag_file_string = "      ";
String injectie_dag_file_string = "      ";
String gas_dag_file_string = "      ";
String elektriciteit_uur_file_string = "      ";
String injectie_uur_file_string = "      ";
String gas_uur_file_string = "      ";
String eenheid_string = "      ";
String totaal_string = "      ";
String pwm_override = "      ";

String test_jaar_string = "      ";
String test_maand_string = "      ";
String test_dag_string = "      ";
String test_uur_string = "      ";
String test_minuten_string = "      ";

unsigned long nu;
unsigned long relais1_vertraging_long;
unsigned long relais2_vertraging_long;
unsigned long relais3_vertraging_long;

String ssid_string = "      ";
String pswd_string = "      ";

```

```

const char* AP_SSID = "ESP32Energie";
const char* AP_PSWD = "ESP32pswd";
const char* STA_SSID = "ssid";
const char* STA_PSWD = "pswd";
char ssid[40];
char pswd[40];
bool netwerk;
unsigned long wacht_op_netwerk;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
    char macStr[18];
    Serial.print("Packet to: ");
    Serial.print(macStr);
    Serial.println(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(" send status:\t");
    Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

void testDir(fs::FS &fs, const char * dirname, uint8_t levels){
    //Serial.printf("Test dir: %s\n", dirname);
    File root = fs.open(dirname);
    if(root){
        //Serial.println("Folder bestaat reeds");
        dir_test = true;
    }
}

void testFile(fs::FS &fs, const char * path){
    File file = fs.open(path);
    if(file){
        file_test = true;
        file.close();
    }
}

void createDir(fs::FS &fs, const char * path){
    //Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
        //Serial.println("Dir created");
    } else {
        //Serial.println("mkdir failed");
    }
}

void readFile(fs::FS &fs, const char * path){
    //Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        //Serial.println("Failed to open file for reading");
        return;
    }
    //Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
    file.close();
}

void read_elektriciteit_verbruik_float(fs::FS &fs, const char * path){
    //Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(kwh_sd_char, 0, sizeof(kwh_sd_char));
    while(file.available()){
        kwh_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    kwh_sd_string = String(kwh_sd_char);
    kwh_sd_float = kwh_sd_string.toFloat();
    //Serial.printf("%.3f", kwh_sd_float);
    //Serial.println();
}

```

```

void read_injectie_float(fs::FS &fs, const char * path){
    //Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(injectie_sd_char, 0, sizeof(injectie_sd_char));
    while(file.available()){
        injectie_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    injectie_sd_string = String(injectie_sd_char);
    injectie_sd_float = injectie_sd_string.toFloat();
    // Serial.printf("%.3f", injectie_sd_float);
    // Serial.println();
}

void read_gas_verbruik_float(fs::FS &fs, const char * path){
    //Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(gas_sd_char, 0, sizeof(gas_sd_char));
    while(file.available()){
        gas_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    gas_sd_string = String(gas_sd_char);
    gas_sd_float = gas_sd_string.toFloat();
    // Serial.printf("%.3f", gas_sd_float);
    // Serial.println();
}

void read_elektriciteit_verbruik_char(fs::FS &fs, const char * path){
    // Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(kwh_sd_char, 0, sizeof(kwh_sd_char));
    while(file.available()){
        kwh_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    //kwh_sd_string = String(kwh_sd_char);
    // Serial.println(kwh_sd_char);
}

void read_injectie_char(fs::FS &fs, const char * path){
    // Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(injectie_sd_char, 0, sizeof(injectie_sd_char));
    while(file.available()){
        injectie_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    // Serial.println(injectie_sd_char);
}

void read_gas_verbruik_char(fs::FS &fs, const char * path){
    // Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){

```

```

    return;
}
int teller = 0;
int index;
memset(gas_sd_char, 0, sizeof(gas_sd_char));
while(file.available()){
    gas_sd_char[teller] = file.read();
    teller++;
}
file.close();
// Serial.println(gas_sd_char);
}

void read_dagen(fs::FS &fs, const char * path){
//Serial.printf("Reading file: %s\n", path);
File file = fs.open(path);
int teller = 0;
memset(aantal_dagen_char, 0, sizeof(aantal_dagen_char));
while(file.available()){
    aantal_dagen_char[teller] = file.read();
    teller++;
}
file.close();
aantal_dagen_string = String(aantal_dagen_char);
dagen_int = aantal_dagen_string.toInt();
}

void writeFile(fs::FS &fs, const char * path, const char * message){
// Serial.printf("Writing file: %s\n", path);
File file = fs.open(path, FILE_WRITE);
if(!file){
    // Serial.println("Failed to open file for writing");
    return;
}
if(file.print(message)){
    // Serial.println("File written");
} else {
    // Serial.println("Write failed");
}
file.close();
}

void appendFile(fs::FS &fs, const char * path, const char * message){
// Serial.printf("Appending to file: %s\n", path);
File file = fs.open(path, FILE_APPEND);
if(!file){
    // Serial.println("Failed to open file for appending");
    return;
}
if(file.print(message)){
    // Serial.println("Message appended");
} else {
    // Serial.println("Append failed");
}
file.close();
}

void dir_bestaat_test(){
if(!jaar_dir_bestaat){
    dir_test = false;
    jaar_string = "/F" + String(jaar_int);
    jaar_string.toCharArray(jaar_char, jaar_string.length() + 1);
    testDir(SD, jaar_char, 1);
    if(dir_test == false){
        createDir(SD, jaar_char);
    }
    jaar_dir_bestaat = true;
}
if(!maand_dir_bestaat){
    dir_test = false;
    maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
    maand_string.toCharArray(maand_char, maand_string.length() + 1);
    testDir(SD, maand_char, 1);
    if(dir_test == false){
        createDir(SD, maand_char);
    }
    maand_dir_bestaat = true;
}
if(!dag_dir_bestaat){

```

```

dir_test = false;
dag_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int);
dag_string.toCharArray(dag_char, dag_string.length() + 1);
testDir(SD, dag_char, 1);
if(dir_test == false){
    createDir(SD, dag_char);
}
dag_dir_bestaat = true;
}

void file_bestaat_test(){
if(!elektriciteit_jaar_file_bestaat){
    file_test = false;
    elektriciteit_jaar_file_string = "/F" + String(jaar_int) + "/E" + jaar_int;
    elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
    testFile(SD, elektriciteit_jaar_file_char);
    if(file_test == false){
        verbruik_string = String(kwh_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, elektriciteit_jaar_file_char, verbruik_char);
    }
    elektriciteit_jaar_file_bestaat = true;
}
if(!injectie_jaar_file_bestaat){
    file_test = false;
    injectie_jaar_file_string = "/F" + String(jaar_int) + "/I" + jaar_int;
    injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
    testFile(SD, injectie_jaar_file_char);
    if(file_test == false){
        verbruik_string = String(injectie_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, injectie_jaar_file_char, verbruik_char);
    }
    injectie_jaar_file_bestaat = true;
}
if(!gas_jaar_file_bestaat){
    file_test = false;
    gas_jaar_file_string = "/F" + String(jaar_int) + "/G" + jaar_int;
    gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
    testFile(SD, gas_jaar_file_char);
    if(file_test == false){
        verbruik_string = String(gas_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, gas_jaar_file_char, verbruik_char);
    }
    gas_jaar_file_bestaat = true;
}
if(!elektriciteit_maand_file_bestaat){
    file_test = false;
    elektriciteit_maand_file_string = "/F" + String(jaar_int) + "/E" + maand_int;
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
    testFile(SD, elektriciteit_maand_file_char);
    if(file_test == false){
        verbruik_string = String(kwh_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, elektriciteit_maand_file_char, verbruik_char);
    }
    elektriciteit_maand_file_bestaat = true;
}
if(!injectie_maand_file_bestaat){
    file_test = false;
    injectie_maand_file_string = "/F" + String(jaar_int) + "/I" + maand_int;
    injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
    testFile(SD, injectie_maand_file_char);
    if(file_test == false){
        verbruik_string = String(injectie_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, injectie_maand_file_char, verbruik_char);
    }
    injectie_maand_file_bestaat = true;
}
if(!gas_maand_file_bestaat){
    file_test = false;
    gas_maand_file_string = "/F" + String(jaar_int) + "/G" + maand_int;
    gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
    testFile(SD, gas_maand_file_char);
    if(file_test == false){
        verbruik_string = String(gas_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, gas_maand_file_char, verbruik_char);
    }
    gas_maand_file_bestaat = true;
}

```

```

verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_maand_file_char, verbruik_char);
}
gas_maand_file_bestaat = true;
}
if(!elektriciteit_dag_file_bestaat){
file_test = false;
elektriciteit_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/E" + dag_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
testFile(SD, elektriciteit_dag_file_char);
if(file_test == false){
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_dag_file_char, verbruik_char);
}
elektriciteit_dag_file_bestaat = true;
}
if(!injectie_dag_file_bestaat){
file_test = false;
injectie_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/I" + dag_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
testFile(SD, injectie_dag_file_char);
if(file_test == false){
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_dag_file_char, verbruik_char);
}
injectie_dag_file_bestaat = true;
}
if(!gas_dag_file_bestaat){
file_test = false;
gas_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/G" + dag_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
testFile(SD, gas_dag_file_char);
if(file_test == false){
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_dag_file_char, verbruik_char);
}
gas_dag_file_bestaat = true;
}
if(!elektriciteit_uur_file_bestaat){
file_test = false;
elektriciteit_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/E" + uren_int;
elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
testFile(SD, elektriciteit_uur_file_char);
if(file_test == false){
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_uur_file_char, verbruik_char);
}
elektriciteit_uur_file_bestaat = true;
}
if(!injectie_uur_file_bestaat){
file_test = false;
injectie_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/I" + uren_int;
injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
testFile(SD, gas_uur_file_char);
if(file_test == false){
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_uur_file_char, verbruik_char);
}
injectie_uur_file_bestaat = true;
}
if(!gas_uur_file_bestaat){
file_test = false;
gas_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/G" + uren_int;
gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
testFile(SD, gas_uur_file_char);
if(file_test == false){
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_uur_file_char, verbruik_char);
}
gas_uur_file_bestaat = true;
}

```

```
}
```

```
void jaar_verbruik_injectie(){
    elektriciteit_jaar_file_string = "/F" + String(jaar_vorig_int) + "/E" + jaar_vorig_int;
    elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
    read_elektriciteit_verbruik_float(SD, elektriciteit_jaar_file_char);
    verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, elektriciteit_jaar_file_char, verbruik_char);

    injectie_jaar_file_string = "/F" + String(jaar_vorig_int) + "/I" + jaar_vorig_int;
    injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
    read_injectie_float(SD, injectie_jaar_file_char);
    verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, injectie_jaar_file_char, verbruik_char);

    gas_jaar_file_string = "/F" + String(jaar_vorig_int) + "/G" + jaar_vorig_int;
    gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
    read_gas_verbruik_float(SD, gas_jaar_file_char);
    verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, gas_jaar_file_char, verbruik_char);

    jaar_string = "/F" + String(jaar_int);
    jaar_string.toCharArray(jaar_char, jaar_string.length() + 1);
    createDir(SD, jaar_char);

    maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
    maand_string.toCharArray(maand_char, maand_string.length() + 1);
    createDir(SD, maand_char);

    elektriciteit_jaar_file_string = "/F" + String(jaar_int) + "/E" + jaar_int;
    elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
    verbruik_string = String(kwh_totaal_float, 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, elektriciteit_jaar_file_char, verbruik_char);

    injectie_jaar_file_string = "/F" + String(jaar_int) + "/I" + jaar_int;
    injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
    verbruik_string = String(injectie_totaal_float, 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, injectie_jaar_file_char, verbruik_char);

    gas_jaar_file_string = "/F" + String(jaar_int) + "/G" + jaar_int;
    gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
    verbruik_string = String(gas_totaal_float, 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, gas_jaar_file_char, verbruik_char);
}

void maand_verbruik_injectie(){
    elektriciteit_maand_file_string = "/F" + String(jaar_vorig_int) + "/E" + maand_vorig_int;
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
    read_elektriciteit_verbruik_float(SD, elektriciteit_maand_file_char);
    verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, elektriciteit_maand_file_char, verbruik_char);

    injectie_maand_file_string = "/F" + String(jaar_vorig_int) + "/I" + maand_vorig_int;
    injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
    read_injectie_float(SD, injectie_maand_file_char);
    verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, injectie_maand_file_char, verbruik_char);

    gas_maand_file_string = "/F" + String(jaar_vorig_int) + "/G" + maand_vorig_int;
    gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
    read_gas_verbruik_float(SD, gas_maand_file_char);
    verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
    verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
    writeFile(SD, gas_maand_file_char, verbruik_char);

    maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
    maand_string.toCharArray(maand_char, maand_string.length() + 1);
    createDir(SD, maand_char);

    elektriciteit_maand_file_string = "/F" + String(jaar_int) + "/E" + maand_int;
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
```

```

verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_maand_file_char, verbruik_char);

elektriciteit_maand_file_bestaat = true;
injectie_maand_file_string = "/F" + String(jaar_int) + "/I" + maand_int;
injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_maand_file_char, verbruik_char);

injectie_maand_file_bestaat = true;
gas_maand_file_string = "/F" + String(jaar_int) + "/G" + maand_int;
gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_maand_file_char, verbruik_char);
gas_maand_file_bestaat = true;
}

void dag_verbruik_injectie(){
elektriciteit_dag_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/E" + dag_vorig_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
read_elektriciteit_verbruik_float(SD, elektriciteit_dag_file_char);
verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_dag_file_char, verbruik_char);

injectie_dag_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/I" + dag_vorig_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
read_injectie_float(SD, injectie_dag_file_char);
verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_dag_file_char, verbruik_char);

gas_dag_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/G" + dag_vorig_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
read_gas_verbruik_float(SD, gas_dag_file_char);
verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_dag_file_char, verbruik_char);

elektriciteit_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/E" + dag_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_dag_file_char, verbruik_char);

injectie_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/I" + dag_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_dag_file_char, verbruik_char);

gas_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/G" + dag_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_dag_file_char, verbruik_char);

aantal_dagen_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/dagen";
aantal_dagen_file_string.toCharArray(aantal_dagen_file_char, aantal_dagen_file_string.length() + 1);
aantal_dagen_string = String(dag_int);
aantal_dagen_string.toCharArray(aantal_dagen_file_char, aantal_dagen_string.length() + 1);
writeFile(SD, aantal_dagen_file_char, aantal_dagen_file_char);
}

void uur_verbruik_injectie(){
elektriciteit_uur_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/F" + String(dag_vorig_int) + "/E" +
uren_vorig_int;
elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
read_elektriciteit_verbruik_float(SD, elektriciteit_uur_file_char);
verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_uur_file_char, verbruik_char);

injectie_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_vorig_int) + "/F" + String(dag_vorig_int) + "/I" + uren_vorig_int;
injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
read_injectie_float(SD, injectie_uur_file_char);

```

```

verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_uur_file_char, verbruik_char);

gas_uur_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/F" + String(dag_vorig_int) + "/G" + uren_vorig_int;
gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
read_gas_verbruik_float(SD, gas_uur_file_char);
verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_uur_file_char, verbruik_char);

elektriciteit_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/E" + uren_int;
elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_uur_file_char, verbruik_char);

injectie_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/I" + uren_int;
injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_uur_file_char, verbruik_char);

gas_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/G" + uren_int;
gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_uur_file_char, verbruik_char);
}

void lees_uur_verbruik_injectie(){
int eind_uur_int;
if((data_jaar_int == jaar_int) && (data_maand_int == maand_int) && (data_dag_int == dag_int)){
    eind_uur_int = uren_int;
}
else{
    eind_uur_int = 24;
}
eenheid_string = "uur";
eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
for(int lees_uur_int = 0; lees_uur_int < eind_uur_int; lees_uur_int++){
    sprintf(periode_array_char[lees_uur_int], "%02d", lees_uur_int);
    elektriciteit_uur_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/F" + String(data_dag_int) + "/E" + lees_uur_int;
    elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
    file_test = false;
    testFile(SD, elektriciteit_uur_file_char);
    if(file_test == true){
        read_elektriciteit_verbruik_char(SD, elektriciteit_uur_file_char);
        strcpy(data_kwh_array_char[lees_uur_int], kwh_sd_char);
    }
    else{
        memset(data_kwh_array_char[lees_uur_int], 0, sizeof(data_kwh_array_char[lees_uur_int]));
    }
    injectie_uur_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/F" + String(data_dag_int) + "/I" + lees_uur_int;
    injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
    file_test = false;
    testFile(SD, injectie_uur_file_char);
    if(file_test == true){
        read_injectie_char(SD, injectie_uur_file_char);
        strcpy(data_injectie_array_char[lees_uur_int], injectie_sd_char);
    }
    else{
        memset(data_injectie_array_char[lees_uur_int], 0, sizeof(data_injectie_array_char[lees_uur_int]));
    }
    gas_uur_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/F" + String(data_dag_int) + "/G" + lees_uur_int;
    gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
    file_test = false;
    testFile(SD, gas_uur_file_char);
    if(file_test == true){
        read_gas_verbruik_char(SD, gas_uur_file_char);
        strcpy(data_gas_array_char[lees_uur_int], gas_sd_char);
    }
    else{
        memset(data_gas_array_char[lees_uur_int], 0, sizeof(data_gas_array_char[lees_uur_int]));
    }
}
totaal_string = "totaal";
totaal_string.toCharArray(totaal_char, totaal_string.length() + 1);
strcpy(periode_array_char[24], totaal_char);
}

```

```

elektriciteit_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/E" + data_dag_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
file_test = false;
testFile(SD, elektriciteit_dag_file_char);
if((file_test == true) && (eind_uur_int == 24)){
    read_elektriciteit_verbruik_char(SD, elektriciteit_dag_file_char);
    strcpy(data_kwh_array_char[24], kwh_sd_char);
}
else{
    memset(data_kwh_array_char[24], 0, sizeof(data_kwh_array_char[24]));
}
injectie_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/I" + data_dag_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
file_test = false;
testFile(SD, injectie_dag_file_char);
if((file_test == true) && (eind_uur_int == 24)){
    read_injectie_char(SD, injectie_dag_file_char);
    strcpy(data_injectie_array_char[24], injectie_sd_char);
}
else{
    memset(data_injectie_array_char[24], 0, sizeof(data_injectie_array_char[24]));
}
gas_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/G" + data_dag_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
file_test = false;
testFile(SD, gas_dag_file_char);
if((file_test == true) && (eind_uur_int == 24)){
    read_gas_verbruik_char(SD, gas_dag_file_char);
    strcpy(data_gas_array_char[24], gas_sd_char);
}
else{
    memset(data_gas_array_char[24], 0, sizeof(data_gas_array_char[24]));
}
for(int x = eind_uur_int + 1; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}

void lees_dag_verbruik_injectie(){
volledige_maand_bool = true;
if((data_jaar_int == jaar_int) && (data_maand_int == maand_int)){
    volledige_maand_bool = false;
}
eenheid_string = "dag";
eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
aantal_dagen_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/dagen";
aantal_dagen_file_string.toCharArray(aantal_dagen_file_char, aantal_dagen_file_string.length() + 1);
file_test = false;
testFile(SD, aantal_dagen_file_char);
if(file_test == true){
    read_dagen(SD, aantal_dagen_file_char);
    if(volledige_maand_bool == false){
        dagen_int = dagen_int - 1;
    }
    for(int lees_dag_int = 0; lees_dag_int < dagen_int; lees_dag_int++){
        sprintf(periode_array_char[lees_dag_int], "%02d", lees_dag_int + 1);
        elektriciteit_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/E" + (lees_dag_int + 1);
        elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
        file_test = false;
        testFile(SD, elektriciteit_dag_file_char);
        if(file_test == true){
            read_elektriciteit_verbruik_char(SD, elektriciteit_dag_file_char);
            strcpy(data_kwh_array_char[lees_dag_int], kwh_sd_char);
        }
        else{
            memset(data_kwh_array_char[lees_dag_int], 0, sizeof(data_kwh_array_char[lees_dag_int]));
        }
        injectie_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/I" + (lees_dag_int + 1);
        injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
        file_test = false;
        testFile(SD, injectie_dag_file_char);
        if(file_test == true){
            read_injectie_char(SD, injectie_dag_file_char);
            strcpy(data_injectie_array_char[lees_dag_int], injectie_sd_char);
        }
        else{

```

```

        memset(data_injectie_array_char[lees_dag_int], 0, sizeof(data_injectie_array_char[lees_dag_int]));
    }
    gas_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/G" + (lees_dag_int + 1);
    gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
    file_test = false;
    testFile(SD, gas_dag_file_char);
    if(file_test == true){
        read_gas_verbruik_char(SD, gas_dag_file_char);
        strcpy(data_gas_array_char[lees_dag_int], gas_sd_char);
    }
    else{
        memset(data_gas_array_char[lees_dag_int], 0, sizeof(data_gas_array_char[lees_dag_int]));
    }
}
if(volledige_maand_bool == true){
    totaal_string = "totaal";
    totaal_string.toCharArray(totaal_char, totaal_string.length() + 1);
    strcpy(periode_array_char[dagen_int], totaal_char);
    elektriciteit_maand_file_string = "/F" + String(data_jaar_int) + "/E" + data_maand_int;
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, elektriciteit_maand_file_char);
    if((file_test == true) && (volledige_maand_bool == true)){
        read_elektriciteit_verbruik_char(SD, elektriciteit_maand_file_char);
        strcpy(data_kwh_array_char[dagen_int], kwh_sd_char);
    }
    else{
        memset(data_kwh_array_char[dagen_int], 0, sizeof(data_kwh_array_char[dagen_int]));
    }
    injectie_maand_file_string = "/F" + String(data_jaar_int) + "/I" + data_maand_int;
    injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, injectie_maand_file_char);
    if((file_test == true) && (volledige_maand_bool == true)){
        read_injectie_char(SD, injectie_maand_file_char);
        strcpy(data_injectie_array_char[dagen_int], injectie_sd_char);
    }
    else{
        memset(data_injectie_array_char[dagen_int], 0, sizeof(data_injectie_array_char[dagen_int]));
    }
    gas_maand_file_string = "/F" + String(data_jaar_int) + "/G" + data_maand_int;
    gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, gas_maand_file_char);
    if((file_test == true) && (volledige_maand_bool == true)){
        read_gas_verbruik_char(SD, gas_maand_file_char);
        strcpy(data_gas_array_char[dagen_int], gas_sd_char);
    }
    else{
        memset(data_gas_array_char[dagen_int], 0, sizeof(data_gas_array_char[dagen_int]));
    }
}
for(int x = dagen_int + 1; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}
else{
    for(int x = 0; x < 32; x++){
        sprintf(periode_array_char[x], "%02d", (x + 1));
        memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
        memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
        memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
    }
}
}

void lees_maand_verbruik_injectie(){
    int eind_maand_int;
    if(data_jaar_int == jaar_int){
        eind_maand_int = maand_int - 1;
    }
    else{
        eind_maand_int = 13;
    }
    eenheid_string = "maand";
    eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
}

```

```

for(int lees_maand_int = 0; lees_maand_int < eind_maand_int; lees_maand_int++){
    sprintf(periode_array_char[lees_maand_int], "%02d", lees_maand_int + 1);
    elektriciteit_maand_file_string = "/F" + String(data_jaar_int) + "/E" + (lees_maand_int + 1);
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, elektriciteit_maand_file_char);
    if(file_test == true){
        read_elektriciteit_verbruik_char(SD, elektriciteit_maand_file_char);
        strcpy(data_kwh_array_char[lees_maand_int], kwh_sd_char);
    }
    else{
        memset(data_kwh_array_char[lees_maand_int], 0, sizeof(data_kwh_array_char[lees_maand_int]));
    }
    injectie_maand_file_string = "/F" + String(data_jaar_int) + "/I" + (lees_maand_int + 1);
    injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, injectie_maand_file_char);
    if(file_test == true){
        read_injectie_char(SD, injectie_maand_file_char);
        strcpy(data_injectie_array_char[lees_maand_int], injectie_sd_char);
    }
    else{
        memset(data_injectie_array_char[lees_maand_int], 0, sizeof(data_injectie_array_char[lees_maand_int]));
    }
    gas_maand_file_string = "/F" + String(data_jaar_int) + "/G" + (lees_maand_int + 1);
    gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, gas_maand_file_char);
    if(file_test == true){
        read_gas_verbruik_char(SD, gas_maand_file_char);
        strcpy(data_gas_array_char[lees_maand_int], gas_sd_char);
    }
    else{
        memset(data_gas_array_char[lees_maand_int], 0, sizeof(data_gas_array_char[lees_maand_int]));
    }
}
totaal_string = "totaal";
totaal_string.toCharArray(totaal_char, totaal_string.length() + 1);
strcpy(periode_array_char[12], totaal_char);
elektriciteit_jaar_file_string = "/F" + String(data_jaar_int) + "/E" + data_jaar_int;
elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, elektriciteit_jaar_file_char);
if((file_test == true) && (eind_maand_int == 13)){
    read_elektriciteit_verbruik_char(SD, elektriciteit_jaar_file_char);
    strcpy(data_kwh_array_char[12], kwh_sd_char);
}
else{
    memset(data_kwh_array_char[12], 0, sizeof(data_kwh_array_char[dagen_int]));
}
injectie_jaar_file_string = "/F" + String(data_jaar_int) + "/I" + data_jaar_int;
injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, injectie_jaar_file_char);
if((file_test == true) && (eind_maand_int == 13)){
    read_injectie_char(SD, injectie_jaar_file_char);
    strcpy(data_injectie_array_char[12], injectie_sd_char);
}
else{
    memset(data_injectie_array_char[12], 0, sizeof(data_injectie_array_char[dagen_int]));
}
gas_jaar_file_string = "/F" + String(data_jaar_int) + "/G" + data_jaar_int;
gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, gas_jaar_file_char);
if((file_test == true) && (eind_maand_int == 13)){
    read_gas_verbruik_char(SD, gas_jaar_file_char);
    strcpy(data_gas_array_char[12], gas_sd_char);
}
else{
    memset(data_gas_array_char[12], 0, sizeof(data_gas_array_char[dagen_int]));
}
for(int x = eind_maand_int; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}

```

```

void lees_jaar_verbruik_injectie(){
    int lees_jaar = jaar_int - 25;
    int x = 0;
    eenheid_string = "jaar";
    eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
    for(int lees_jaar_int = lees_jaar; lees_jaar_int < jaar_int; lees_jaar_int++){
        sprintf(periode_array_char[x], "%04d", lees_jaar_int);
        elektriciteit_jaar_file_string = "/F" + String(lees_jaar_int) + "/E" + lees_jaar_int;
        elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
        file_test = false;
        testFile(SD, elektriciteit_jaar_file_char);
        if(file_test == true){
            read_elektriciteit_verbruik_char(SD, elektriciteit_jaar_file_char);
            strcpy(data_kwh_array_char[x], kwh_sd_char);
        }
    }
    else{
        memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[lees_jaar_int]));
    }
    injectie_jaar_file_string = "/F" + String(lees_jaar_int) + "/I" + lees_jaar_int;
    injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
    file_test = false;
    testFile(SD, injectie_jaar_file_char);
    if(file_test == true){
        read_injectie_char(SD, injectie_jaar_file_char);
        strcpy(data_injectie_array_char[x], injectie_sd_char);
    }
    else{
        memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[lees_jaar_int]));
    }
    gas_jaar_file_string = "/F" + String(lees_jaar_int) + "/G" + lees_jaar_int;
    gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
    file_test = false;
    testFile(SD, gas_jaar_file_char);
    if(file_test == true){
        read_gas_verbruik_char(SD, gas_jaar_file_char);
        strcpy(data_gas_array_char[x], gas_sd_char);
    }
    else{
        memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[lees_jaar_int]));
    }
    x++;
}
for(int x = 25; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}

```

```

const char energie_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
<iframe style="display:none" name="hidden-form"></iframe>
<meta name="viewport" content="width=device-width, initial-scale=0.85">
<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>
</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% &nbsp; KWh</div>
</div>
<br>
<div class="kader">
<div class="links">Totaal injectie : </div>
<div class="midden">%injectie_totaal% &nbsp; KWh</div>
</div>
<br>
<div class="kader">
<div class="links">Verbruik nu : </div>
<div class="midden">%kwh_nu% &nbsp; KW</div>
</div>
<br>
<div class="kader">
<div class="links">Injectie nu : </div>
<div class="midden">%injectie_nu% &nbsp; KW</div>
</div>
<br>
<div class="kader">
<div class="links">Totaal gas : </div>
<div class="midden">%gas_totaal% &nbsp; m3</div>
</div>
<br><br>
<div class="titel"><b><center>Datum</center></b></div>
<center>
<input type="text" style="text-align:center;" value="%dag%" size=1>
&nbsp;<input type="text" style="text-align:center;" value="%maand%" size=1>
&nbsp;<input type="text" style="text-align:center;" value="%jaar%" size=1>
</center>
<br>
<div class="titel"><b><center>Tijd</center></b></div>
<center><input type="text" style="text-align:center;" value="%tijd%" size=2></center>
<br>
<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>&nbsp;<input type="text" style="text-align:center;" value="%kw_on%" name="input_kw_on" size=1>
&nbsp;<b>V:</b>&nbsp;<input type="text" style="text-align:center;" value="%delay%" name="input_delay" size=1>
&nbsp; <b>Tijd:</b>&nbsp;<input type="text" style="text-align:center;" value="%schakel_tijd%" name="input_schakel_tijd" size=1>
&nbsp; <b>A/M:</b>&nbsp;<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()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="relais_module_plus" value=" + " onclick="ok()">
&nbsp;&nbsp;&nbsp;
<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>&nbsp;<input type="text" style="text-align:center;" value="%pwm_kw%" name="input_pwm_kw" size=1>
&nbsp;<b>1:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_on%" name="input_pwm_tijd_on" size=1>
&nbsp;<b>0:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_off%" name="input_pwm_tijd_off" size=1>

```

```

    &nbsp;<b>A/M:</b>&nbsp;<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%</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>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_1%" name="input_macx_1" size=1>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_2%" name="input_macx_2" size=1>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_3%" name="input_macx_3" size=1>
&nbsp;
<input type="text" style="text-align:center;" value="%display_macx_4%" name="input_macx_4" size=1>
&nbsp;
<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()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="module_plus" value=" + " onclick="ok()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="module_bevestig" value="OK" onclick="ok()"/>
</center>
</form>
<br><br><br>
<center>
<button type="button" onclick="data_weergave()">Data weergave</button>
</center>
</small>
<br>
<br>
<br>
<h6><b>thieu-b55 oktober 2022</b></h6>
<script>
function ok(){
  setTimeout(function(){document.location.reload();},250);
}
</script>
<script>
function data_weergave(){
  location.assign("http://192.168.1.222/data/");
}
</script>
</body>
</html>
)rawliteral";

const char data_html[] = R"rawliteral(

```

```
<!DOCTYPE HTML>
<html>
<head>
<iframe style="display:none" name="hidden-form"></iframe>
<meta name="viewport" content="width=device-width, initial-scale=0.85">
<title>Data weergave</title>
<style>
div.kader {
    position: relative;
    width: 400px;
    height: 12x;
}
div.links{
    position: absolute;
    left : 0px;
    width: auto;
    height: 12px;
}
div.links_midden{
    position:absolute;
    left: 80px;
    width: auto;
    height: 12px;
}
div.rechts_midden{
    position:absolute;
    left: 200px;
    width: auto;
    height: 12px;
}
div.rechts{
    position:absolute;
    left: 320px;
    width: auto;
    height: 12px;
}
div.titel{
    height: 25px;
    width: auto;
}
div.bottom{
    position: fixed;
    bottom: 0px;
}
div.data_links{
    position: absolute;
    left: 0px;
    width: 80px;
    height: 12px;
}
div.data_links_midden{
    position: absolute;
    left: 80px;
    width: 120px;
    height: 12px;
}
div.data_rechts_midden{
    position: absolute;
    left: 200px;
    width: 120px;
    height: 12px;
}
div.data_rechts{
    position: absolute;
    left: 320px;
    width: 80px;
    height: 12px;
}
div.blanco_20{
    width: auto;
    height: 20px;
}
div.blanco_15{
    width: auto;
    height: 15px;
}
</style>
</head>
<body>
```

```
<h3><center> Data weergave </center></h3>
<center>
<button type="button" onclick="start_pagina()">Naar begin pagina</button>
</center>
<br>
<form action="/get" target="hidden-form">
<center>
<input type="text" style="text-align:center;" value="%data_dag%" name="input_data_dag" size=1>
  &ampnbsp<input type="text" style="text-align:center;" value="%data_maand%" name="input_data_maand" size=1>
  &ampnbsp<input type="text" style="text-align:center;" value="%data_jaar%" name="input_data_jaar" size=1>
</center>
<br>
<center>
<input type="submit" name="bevestig_periode" value="OK" onclick="ok()">
</center>
<br>
</form>
<small>
<div class="titel"><center><b>Periode : %periode%</b></center></div>
<div class="kader"><b>
<div class="links">%eenheid% </div>
<div class="links_midden">verbruik kWh</div>
<div class="rechts_midden">injectie kWh</div>
<div class="rechts">gas m3</div></b>
</div>
<div class="blanco_20">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_1%</div>
<div class="data_links_midden">%verbruik_1%</div>
<div class="data_rechts_midden">%injectie_1%</div>
<div class="data_rechts">%gas_verbruik_1%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_2%</div>
<div class="data_links_midden">%verbruik_2%</div>
<div class="data_rechts_midden">%injectie_2%</div>
<div class="data_rechts">%gas_verbruik_2%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_3%</div>
<div class="data_links_midden">%verbruik_3%</div>
<div class="data_rechts_midden">%injectie_3%</div>
<div class="data_rechts">%gas_verbruik_3%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_4%</div>
<div class="data_links_midden">%verbruik_4%</div>
<div class="data_rechts_midden">%injectie_4%</div>
<div class="data_rechts">%gas_verbruik_4%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_5%</div>
<div class="data_links_midden">%verbruik_5%</div>
<div class="data_rechts_midden">%injectie_5%</div>
<div class="data_rechts">%gas_verbruik_5%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_6%</div>
<div class="data_links_midden">%verbruik_6%</div>
<div class="data_rechts_midden">%injectie_6%</div>
<div class="data_rechts">%gas_verbruik_6%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_7%</div>
<div class="data_links_midden">%verbruik_7%</div>
<div class="data_rechts_midden">%injectie_7%</div>
<div class="data_rechts">%gas_verbruik_7%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_8%</div>
<div class="data_links_midden">%verbruik_8%</div>
<div class="data_rechts_midden">%injectie_8%</div>
```





```

<div class="blanco_15">&nbsp;</div>
</div>
<div class="kader">
<div class="data_links">%periode_30%</div>
<div class="data_links_midden">%verbruik_30%</div>
<div class="data_rechts_midden">%injectie_30%</div>
<div class="data_rechts">%gas_verbruik_30%</div>
</div>
<div class="blanco_15">&nbsp;</div>
</div>
<div class="kader">
<div class="data_links">%periode_31%</div>
<div class="data_links_midden">%verbruik_31%</div>
<div class="data_rechts_midden">%injectie_31%</div>
<div class="data_rechts">%gas_verbruik_31%</div>
</div>
<div class="blanco_15">&nbsp;</div>
</div>
<div class="kader">
<div class="data_links">%periode_32%</div>
<div class="data_links_midden">%verbruik_32%</div>
<div class="data_rechts_midden">%injectie_32%</div>
<div class="data_rechts">%gas_verbruik_32%</div>
</div>
</small>
<script>
function start_pagina(){
  location.assign("http://192.168.1.222/");
}
function ok(){
  setTimeout(function(){document.location.reload();},250);
}
</script>
</body>
</html>
)rawliteral";

const char netwerk_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
<iframe style="display:none" name="hidden-form"></iframe>
<title>Energie Beheer</title>
<meta name="viewport" content="width=device-width, initial-scale=0.85">
</head>
<body>
<h5><center><strong>ESP32 Netwerk instellingen</strong></center></h5>
<form action="/get">
<table style="width:100%;">
<tr>
<td style="text-align:left;"><p><small><b><label for="dummy">ssid :</label></b></small></p></td>
<td style="text-align:center;"><input type="text" name="ssid"></td>
</tr>
<tr>
<td style="text-align:left;"><p><small><b><label for="dummy">pswd : </label></b></small></p></td>
<td style="text-align:center;"><input type="text" name="pswd"></td>
</tr>
</table>
<center><input type="submit" value="Bevestig" onclick="ok()"></center>
</form>
<br>
<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 == "kwh_nu"){
    temp = String(kwh_nu_float, 3);
    temp.toCharArray(kwh_nu_float_char, (temp.length() + 1));
    return(kwh_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 == "dag"){
    sprintf(dag_char, "%02d", dag_int);
    return(dag_char);
}
if(var == "maand"){
    sprintf(maand_char, "%02d", maand_int);
    return(maand_char);
}
if(var == "jaar"){
    sprintf(jaar_char, "%04d", jaar_int);
    return(jaar_char);
}
if(var == "tijd"){
    sprintf(tijd_char, "%02d:%02d", uren_int, minuten_int);
    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);
}
if(var == "data_dag"){
    sprintf(data_dag_char, "%02d", data_dag_int);
    return(data_dag_char);
}
if(var == "data_maand"){
    sprintf(data_maand_char, "%02d", data_maand_int);
    return(data_maand_char);
}
if(var == "data_jaar"){
    sprintf(data_jaar_char, "%04d", data_jaar_int);
    return(data_jaar_char);
}
if(var == "periode"){
    return(periode_char);
}
if(var == "eenheid"){
    return(eenheid_char);
}
if(var == "periode_1"){
    return(periode_array_char[0]);
}
if(var == "periode_2"){
    return(periode_array_char[1]);
}
if(var == "periode_3"){
    return(periode_array_char[2]);
}
if(var == "periode_4"){
    return(periode_array_char[3]);
}
if(var == "periode_5"){
    return(periode_array_char[4]);
}
if(var == "periode_6"){
    return(periode_array_char[5]);
}
if(var == "periode_7"){
    return(periode_array_char[6]);
}
if(var == "periode_8"){
    return(periode_array_char[7]);
}
if(var == "periode_9"){
    return(periode_array_char[8]);
}
if(var == "periode_10"){
    return(periode_array_char[9]);
}
if(var == "periode_11"){
    return(periode_array_char[10]);
}
if(var == "periode_12"){
    return(periode_array_char[11]);
}
if(var == "periode_13"){
    return(periode_array_char[12]);
}
if(var == "periode_14"){
    return(periode_array_char[13]);
}
if(var == "periode_15"){

```

```

    return(periode_array_char[14]);
}
if(var == "periode_16"){
    return(periode_array_char[15]);
}
if(var == "periode_17"){
    return(periode_array_char[16]);
}
if(var == "periode_18"){
    return(periode_array_char[17]);
}
if(var == "periode_19"){
    return(periode_array_char[18]);
}
if(var == "periode_20"){
    return(periode_array_char[19]);
}
if(var == "periode_21"){
    return(periode_array_char[20]);
}
if(var == "periode_22"){
    return(periode_array_char[21]);
}
if(var == "periode_23"){
    return(periode_array_char[22]);
}
if(var == "periode_24"){
    return(periode_array_char[23]);
}
if(var == "periode_25"){
    return(periode_array_char[24]);
}
if(var == "periode_26"){
    return(periode_array_char[25]);
}
if(var == "periode_27"){
    return(periode_array_char[26]);
}
if(var == "periode_28"){
    return(periode_array_char[27]);
}
if(var == "periode_29"){
    return(periode_array_char[28]);
}
if(var == "periode_30"){
    return(periode_array_char[29]);
}
if(var == "periode_31"){
    return(periode_array_char[30]);
}
if(var == "periode_32"){
    return(periode_array_char[31]);
}
if(var == "verbruik_1"){
    return(data_kwh_array_char[0]);
}
if(var == "verbruik_2"){
    return(data_kwh_array_char[1]);
}
if(var == "verbruik_3"){
    return(data_kwh_array_char[2]);
}
if(var == "verbruik_4"){
    return(data_kwh_array_char[3]);
}
if(var == "verbruik_5"){
    return(data_kwh_array_char[4]);
}
if(var == "verbruik_6"){
    return(data_kwh_array_char[5]);
}
if(var == "verbruik_7"){
    return(data_kwh_array_char[6]);
}
if(var == "verbruik_8"){
    return(data_kwh_array_char[7]);
}
if(var == "verbruik_9"){
    return(data_kwh_array_char[8]);
}

```

```

}

if(var == "verbruik_10"){
    return(data_kwh_array_char[9]);
}
if(var == "verbruik_11"){
    return(data_kwh_array_char[10]);
}
if(var == "verbruik_12"){
    return(data_kwh_array_char[11]);
}
if(var == "verbruik_13"){
    return(data_kwh_array_char[12]);
}
if(var == "verbruik_14"){
    return(data_kwh_array_char[13]);
}
if(var == "verbruik_15"){
    return(data_kwh_array_char[14]);
}
if(var == "verbruik_16"){
    return(data_kwh_array_char[15]);
}
if(var == "verbruik_17"){
    return(data_kwh_array_char[16]);
}
if(var == "verbruik_18"){
    return(data_kwh_array_char[17]);
}
if(var == "verbruik_19"){
    return(data_kwh_array_char[18]);
}
if(var == "verbruik_20"){
    return(data_kwh_array_char[19]);
}
if(var == "verbruik_21"){
    return(data_kwh_array_char[20]);
}
if(var == "verbruik_22"){
    return(data_kwh_array_char[21]);
}
if(var == "verbruik_23"){
    return(data_kwh_array_char[22]);
}
if(var == "verbruik_24"){
    return(data_kwh_array_char[23]);
}
if(var == "verbruik_25"){
    return(data_kwh_array_char[24]);
}
if(var == "verbruik_26"){
    return(data_kwh_array_char[25]);
}
if(var == "verbruik_27"){
    return(data_kwh_array_char[26]);
}
if(var == "verbruik_28"){
    return(data_kwh_array_char[27]);
}
if(var == "verbruik_29"){
    return(data_kwh_array_char[28]);
}
if(var == "verbruik_30"){
    return(data_kwh_array_char[29]);
}
if(var == "verbruik_31"){
    return(data_kwh_array_char[30]);
}
if(var == "verbruik_32"){
    return(data_kwh_array_char[31]);
}
if(var == "injectie_1"){
    return(data_injectie_array_char[0]);
}
if(var == "injectie_2"){
    return(data_injectie_array_char[1]);
}
if(var == "injectie_3"){
    return(data_injectie_array_char[2]);
}

```

```
if(var == "injectie_4"){
    return(data_injectie_array_char[3]);
}
if(var == "injectie_5"){
    return(data_injectie_array_char[4]);
}
if(var == "injectie_6"){
    return(data_injectie_array_char[5]);
}
if(var == "injectie_7"){
    return(data_injectie_array_char[6]);
}
if(var == "injectie_8"){
    return(data_injectie_array_char[7]);
}
if(var == "injectie_9"){
    return(data_injectie_array_char[8]);
}
if(var == "injectie_10"){
    return(data_injectie_array_char[9]);
}
if(var == "injectie_11"){
    return(data_injectie_array_char[10]);
}
if(var == "injectie_12"){
    return(data_injectie_array_char[11]);
}
if(var == "injectie_13"){
    return(data_injectie_array_char[12]);
}
if(var == "injectie_14"){
    return(data_injectie_array_char[13]);
}
if(var == "injectie_15"){
    return(data_injectie_array_char[14]);
}
if(var == "injectie_16"){
    return(data_injectie_array_char[15]);
}
if(var == "injectie_17"){
    return(data_injectie_array_char[16]);
}
if(var == "injectie_18"){
    return(data_injectie_array_char[17]);
}
if(var == "injectie_19"){
    return(data_injectie_array_char[18]);
}
if(var == "injectie_20"){
    return(data_injectie_array_char[19]);
}
if(var == "injectie_21"){
    return(data_injectie_array_char[20]);
}
if(var == "injectie_22"){
    return(data_injectie_array_char[21]);
}
if(var == "injectie_23"){
    return(data_injectie_array_char[22]);
}
if(var == "injectie_24"){
    return(data_injectie_array_char[23]);
}
if(var == "injectie_25"){
    return(data_injectie_array_char[24]);
}
if(var == "injectie_26"){
    return(data_injectie_array_char[25]);
}
if(var == "injectie_27"){
    return(data_injectie_array_char[26]);
}
if(var == "injectie_28"){
    return(data_injectie_array_char[27]);
}
if(var == "injectie_29"){
    return(data_injectie_array_char[28]);
}
if(var == "injectie_30"){

}
```

```
    return(data_injectie_array_char[29]);
}
if(var == "injectie_31"){
    return(data_injectie_array_char[30]);
}
if(var == "injectie_32"){
    return(data_injectie_array_char[31]);
}
if(var == "gas_verbruik_1"){
    return(data_gas_array_char[0]);
}
if(var == "gas_verbruik_2"){
    return(data_gas_array_char[1]);
}
if(var == "gas_verbruik_3"){
    return(data_gas_array_char[2]);
}
if(var == "gas_verbruik_4"){
    return(data_gas_array_char[3]);
}
if(var == "gas_verbruik_5"){
    return(data_gas_array_char[4]);
}
if(var == "gas_verbruik_6"){
    return(data_gas_array_char[5]);
}
if(var == "gas_verbruik_7"){
    return(data_gas_array_char[6]);
}
if(var == "gas_verbruik_8"){
    return(data_gas_array_char[7]);
}
if(var == "gas_verbruik_9"){
    return(data_gas_array_char[8]);
}
if(var == "gas_verbruik_10"){
    return(data_gas_array_char[9]);
}
if(var == "gas_verbruik_11"){
    return(data_gas_array_char[10]);
}
if(var == "gas_verbruik_12"){
    return(data_gas_array_char[11]);
}
if(var == "gas_verbruik_13"){
    return(data_gas_array_char[12]);
}
if(var == "gas_verbruik_14"){
    return(data_gas_array_char[13]);
}
if(var == "gas_verbruik_15"){
    return(data_gas_array_char[14]);
}
if(var == "gas_verbruik_16"){
    return(data_gas_array_char[15]);
}
if(var == "gas_verbruik_17"){
    return(data_gas_array_char[16]);
}
if(var == "gas_verbruik_18"){
    return(data_gas_array_char[17]);
}
if(var == "gas_verbruik_19"){
    return(data_gas_array_char[18]);
}
if(var == "gas_verbruik_20"){
    return(data_gas_array_char[19]);
}
if(var == "gas_verbruik_21"){
    return(data_gas_array_char[20]);
}
if(var == "gas_verbruik_22"){
    return(data_gas_array_char[21]);
}
if(var == "gas_verbruik_23"){
    return(data_gas_array_char[22]);
}
if(var == "gas_verbruik_24"){
    return(data_gas_array_char[23]);
}
```

```

}

if(var == "gas_verbruik_25"){
    return(data_gas_array_char[24]);
}
if(var == "gas_verbruik_26"){
    return(data_gas_array_char[25]);
}
if(var == "gas_verbruik_27"){
    return(data_gas_array_char[26]);
}
if(var == "gas_verbruik_28"){
    return(data_gas_array_char[27]);
}
if(var == "gas_verbruik_29"){
    return(data_gas_array_char[28]);
}
if(var == "gas_verbruik_30"){
    return(data_gas_array_char[29]);
}
if(var == "gas_verbruik_31"){
    return(data_gas_array_char[30]);
}
if(var == "gas_verbruik_32"){
    return(data_gas_array_char[31]);
}
}

void html_input(){
server.begin();
if(netwerk){
    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/html", energie_html, processor);
    });
    server.on("/data", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/html", data_html, processor);
    });
    server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
        char terminator = char(0x0a);
        String temp = "";
        String uren_string = " ";
        String minuten_string = " ";
        char temp_char[30];
        float kw_on;
        float kw_off;
        String override = " ";
        int schakel_delay;
        char char_temp[10];
        bool fout;
        int temp_int;
        int uren_int;
        int minuten_int;
        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_SCHADEL_TIJD)){
            temp = ((request->getParam(INPUT_SCHADEL_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.relaais = 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.relaais = 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.relaais = 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.relaais = 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.relaais = 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.relaais = 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;
    }
/*
 * tijdsturing uitschakelen bij Manueel 0 tijdens tijdsturing
 */
    if(pwm_override == "0"){
        pwm_tijd_gezet = false;
    }
}

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;
}
delay(2000);
ESP.restart();
}
}

if(request->hasParam(INPUT_DATA_DAG)){
temp = ((request->getParam(INPUT_DATA_DAG)->value()) + String(terminator));
data_dag_int = temp.toInt();
Serial.println(data_dag_int);
}

if(request->hasParam(INPUT_DATA_MAAND)){
temp = ((request->getParam(INPUT_DATA_MAAND)->value()) + String(terminator));
data_maand_int = temp.toInt();
Serial.println(data_maand_int);
}

if(request->hasParam(INPUT_DATA_JAAR)){
temp = ((request->getParam(INPUT_DATA_JAAR)->value()) + String(terminator));
data_jaar_int = temp.toInt();
Serial.println(data_jaar_int);
}

if(request->hasParam(BEVESTIG_PERIODE)){
for(int x = 0; x < 32; x++){
memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}

if((data_dag_int != 0) && (data_maand_int != 0) && (data_jaar_int != 0)){
sprintf(periode_char, "%02d - %02d - %04d", data_dag_int, data_maand_int, data_jaar_int);
lees_uur_verbruik_injectie();
}

if(data_dag_int == 0){
sprintf(periode_char, "%02d - %04d", data_maand_int, data_jaar_int);
String temp_string = "dag";
temp_string.toCharArray(eenheid_char, temp_string.length() + 1);
lees_dag_verbruik_injectie();
}

if(data_maand_int == 0){
sprintf(periode_char, "%04d", data_jaar_int);
String temp_string = "maand";
temp_string.toCharArray(eenheid_char, temp_string.length() + 1);
lees_maand_verbruik_injectie();
}
}

```

```

if(data_jaar_int == 0){
    String temp_string = "vorige jaren (max 25)";
    temp_string.toCharArray(periode_char, temp_string.length() + 1);
    temp_string = "jaar";
    temp_string.toCharArray(eenheid_char, temp_string.length() + 1);
    lees_jaar_verbruik_injectie();
}
};

};

};

if(!netwerk){
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
request->send_P(200, "text/html", netwerk_html);
});
server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
    bool ssid_ingevuld = false;
    bool pswd_ingevuld = false;
    String netwerk = "";
    String paswoord = "";
    char terminator = char(0x0a);
    if(request->hasParam(STA_SSID)){
        netwerk = (request->getParam(STA_SSID)->value());
        netwerk = netwerk + String(terminator);
        pref.putString("ssid", netwerk);
        ssid_ingevuld = true;
    }
    if(request->hasParam(STA_PSWD)){
        paswoord = (request->getParam(STA_PSWD)->value());
        paswoord = paswoord + String(terminator);
        pref.putString("pswd", paswoord);
        pswd_ingevuld = true;
    }
    if((ssid_ingevuld) && (pswd_ingevuld)){
        delay(5000);
        ESP.restart();
    }
});
}
};

void setup() {
delay(5000);
pinMode(BLINKIE, OUTPUT);
digitalWrite(BLINKIE, 0);
pinMode(SD_CARD_ERROR, OUTPUT);
digitalWrite(SD_CARD_ERROR, 0);
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
if(!SD.begin()){
    Serial.println("Kontroleer SD kaart");
    digitalWrite(SD_CARD_ERROR, 1);
}
dag_dir_bestaat = false;
jaar_dir_bestaat = false;
maand_dir_bestaat = false;
elektriciteit_dag_file_bestaat = false;
elektriciteit_jaar_file_bestaat = false;
elektriciteit_maand_file_bestaat = false;
elektriciteit_uur_file_bestaat = false;
gas_dag_file_bestaat = false;
gas_jaar_file_bestaat = false;
gas_maand_file_bestaat = false;
gas_uur_file_bestaat = false;
injectie_dag_file_bestaat = false;
injectie_jaar_file_bestaat = false;
injectie_uur_file_bestaat = false;
injectie_maand_file_bestaat = false;
initialiseren = true;

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", 0xd);
    pref.putInt("mac1_4", 0x69);
    pref.putInt("mac1_5", 0xb8);

    pref.putInt("mac2_0", 7);
}

```

```

pref.putInt("mac2_1", 1);
pref.putInt("mac2_2", 2);
pref.putInt("mac2_3", 3);
pref.putInt("mac2_4", 4);
pref.putInt("mac2_5", 7);

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", 0x7c);
pref.putInt("mac5_1", 0x9e);
pref.putInt("mac5_2", 0xbd);
pref.putInt("mac5_3", 0x06);
pref.putInt("mac5_4", 0xb4);
pref.putInt("mac5_5", 0xdc);

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("ssid", "          ");
pref.putString("pswd", "          ");
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");
ssid_string = pref.getString("ssid");
ssid_string.toCharArray(ssid, ssid_string.length());
pswd_string = pref.getString("pswd");
pswd_string.toCharArray(pswd, pswd_string.length());

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.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pswd);
netwerk = true;
wacht_op_netwerk = millis();
while(WiFi.status() != WL_CONNECTED){
delay(500);
if(millis() - wacht_op_netwerk > 15000){
netwerk = false;
break;
}
}
if(netwerk == true){
IPAddress subnet(WiFi.subnetMask());
IPAddress gateway(WiFi.gatewayIP());
IPAddress dns(WiFi.dnsIP(0));
IPAddress static_ip(192,168,1,222);
WiFi.disconnect();
if (WiFi.config(static_ip, gateway, subnet, dns, dns) == false) {
Serial.println("Configuration failed.");
}
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pswd);
}

```

```

wacht_op_netwerk = millis();
while(WiFi.status() != WL_CONNECTED){
    delay(500);
    if(millis() - wacht_op_netwerk > 15000){
        netwerk = false;
        break;
    }
}
if(netwerk == true){
    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);
    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");
}
}
else if(netwerk == false){
    WiFi.disconnect();
    WiFi.mode(WIFI_AP);
    WiFi.softAP(AP_SSID, AP_PSWD);
}
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;
        kwh_nu_float = kwh_nu.toFloat();
        injectie_nu_float = injectie_nu.toFloat();
        verbruik_nu_float = injectie_nu_float - kwh_nu_float;
        gas_totaal_float = gas.toFloat();
        ingelezen.kwh_totaal = kwh_totaal_float;
        ingelezen.injectie_totaal = injectie_totaal_float;
        ingelezen.verbruik_nu = kwh_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;
        //Serial.printf("%0.3f : %0.3f : %0.3f ", kwh_totaal_float, injectie_totaal_float, gas_totaal_float, kwh_nu_float);
        //Serial.println();
    }
    if(initialiseren == true){
        if((kwh_dag_float > 0.1) && (kwh_nacht_float > 0.1)){
            if((injectie_dag_float > 0.1) && (injectie_nacht_float > 0.1)){
                if(gas_totaal_float > 0.1){
                    uren_vorig_int = uren_int;
                    dag_vorig_int = dag_int;
                    maand_vorig_int = maand_int;
                    jaar_vorig_int = jaar_int;
                    initialiseren = false;
                    dir_bestaat_test();
                    file_bestaat_test();
                }
            }
        }
    }
    digitalWrite(BLINKIE, (digitalRead(BLINKIE) ^ 1));
    if(initialiseren == false){
        if(jaar_vorig_int != jaar_int){
            jaar_string = "/" + String(jaar_int);
            jaar_string.toCharArray(jaar_char, jaar_string.length() + 1);
            createDir(SD, jaar_char);
        }
        if(maand_vorig_int != maand_int){
            maand_string = "/" + String(jaar_int) + "/" + String(maand_int);
            maand_string.toCharArray(maand_char, maand_string.length() + 1);
            createDir(SD, maand_char);
            aantal_dagen_file_string = "/" + String(jaar_int) + "/" + String(maand_int) + "/dagen";
            aantal_dagen_file_string.toCharArray(aantal_dagen_file_char, aantal_dagen_file_string.length() + 1);
            aantal_dagen_string = String(dag_int);
            aantal_dagen_string.toCharArray(aantal_dagen_char, aantal_dagen_string.length() + 1);
            writeFile(SD, aantal_dagen_file_char, aantal_dagen_char);
        }
    }
}

```

```

}

if(dag_vorig_int != dag_int){
    dag_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int);
    dag_string.toCharArray(dag_char, dag_string.length() + 1);
    createDir(SD, dag_char);
}
if(uren_vorig_int != uren_int){
    uur_verbruik_injectie();
}
if(dag_vorig_int != dag_int){
    dag_verbruik_injectie();
}
if(maand_vorig_int != maand_int){
    maand_verbruik_injectie();
}
if(jaar_vorig_int != jaar_int){
    jaar_verbruik_injectie();
}
jaar_vorig_int = jaar_int;
maand_vorig_int = maand_int;
dag_vorig_int = dag_int;
uren_vorig_int = uren_int;
}

/*
 * PWM sturing  >> broadcastAddress5  elke 5 seconden
 * Display      >> broadcastAddress1  elke 10 seconden
 */

if(((millis() - nu) > 5000) && (!vijf_seconden)){
    vijf_seconden = true;
    pwm_tijd_gezet_vorig = pwm_tijd_gezet;
    if((uren_on4_int == uren_int) && (minuten_on4_int == minuten_int)){
        pwm_tijd_gezet = true;
    }
    if((uren_off4_int == uren_int) && (minuten_off4_int == minuten_int)){
        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("Succes bij verzenden naar PWM");
    } else {
        Serial.println("Fout bij verzenden naar PWM");
    }*/
}

if(((millis() - nu) > 10000){
    nu = millis();
    vijf_seconden = false;
    pwm_tijd_gezet_vorig = pwm_tijd_gezet;
    if((uren_on4_int == uren_int) && (minuten_on4_int == minuten_int)){
        pwm_tijd_gezet = true;
    }
}

```

```

if((uren_off4_int == uren_int) && (minuten_off4_int == minuten_int)){
    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("Succes bij verzenden naar PWM");
}
else {
    Serial.println("Fout bij verzenden naar PWM");
}
*/
result = esp_now_send(broadcastAddress5, (uint8_t *) &pwm_sturing, sizeof(pwm_sturing));

/*if (result == ESP_OK) {
    Serial.println("Succes bij verzenden naar display");
}
else {
    Serial.println("Fout bij verzenden naar display");
}
*/
if(pwm_override == "1"){
    verbruik_pwm_float = verbruik_nu_float;
}
else{
    verbruik_pwm_float = verbruik_nu_float+ (uitsturing_pwm_float * pwm_kw_float);
}
/*
 * Digitale uitgangen
 */
/*
 * 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.relaist = 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_int == uren_on1_int) && (minuten_int == 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_int == uren_on2_int) && (minuten_int == 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_int == uren_on3_int) && (minuten_int == minuten_on3_int)){
    relais3_uit = true;
    uitsturen.relaiss = 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");
}
*/
}
buffer_data = ""; //leesbuffer wissen
}
/*
* inlezen data, datum en tijd van P1 poort in buffer_data
*/
buffer_data += lees_byte;
if(lees_byte == 0x0a){
    if((buffer_data.substring(4,9)) == "1.0.0"){
        jaar_int = ((buffer_data.substring(10,12)).toInt()) + 2000;
        maand_int = (buffer_data.substring(12,14)).toInt()>;
        dag_int = (buffer_data.substring(14,16)).toInt();
        uren_int = (buffer_data.substring(16,18)).toInt();
        minuten_int = (buffer_data.substring(18,20)).toInt()>;
    }
    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"){
        kwh_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 = "";
}
}

```

## **Slimme\_meter\_esp32\_data\_AP.ino**

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 */
/*
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
*/
/*
 * 21-10-2022 20:15
 */
#include <WiFi.h>
#include <Preferences.h>
#include <esp_now.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "FS.h"
#include "SD.h"
#include "SPI.h"

esp_now_peer_info_t peerInfo;
esp_err_t result;

AsyncWebServer server(80);

Preferences pref;

#define RXD2      16
#define TXD2      17
#define BLINKIE   4
#define SD_CARD_ERROR 2

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;

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 = "          ";

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_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";

const char* INPUT_DATA_DAG = "input_data_dag";
const char* INPUT_DATA_MAAND = "input_data_maand";
const char* INPUT_DATA_JAAR = "input_data_jaar";

const char* BEVESTIG_PERIODE = "bevestig_periode";

bool dag_dir_bestaat = false;
bool dir_test = false;
bool initialiseren = true;
bool elektriciteit_dag_file_bestaat = false;
bool elektriciteit_jaar_file_bestaat = false;
bool elektriciteit_maand_file_bestaat = false;
bool elektriciteit_uur_file_bestaat = false;
bool file_test = false;
bool gas_dag_file_bestaat = false;
bool gas_jaar_file_bestaat = false;
bool gas_maand_file_bestaat = false;
bool gas_uur_file_bestaat = false;
bool injectie_dag_file_bestaat = false;
bool injectie_jaar_file_bestaat = false;
bool injectie_uur_file_bestaat = false;
bool injectie_maand_file_bestaat = false;
bool jaar_dir_bestaat = false;
bool maand_dir_bestaat = false;
bool pwm_tijd_gezet = false;
bool pwm_tijd_gezet_vorig;
bool relais1_uit;
bool relais2_uit;
bool relais3_uit;
bool vijf_seconden = false;

```

```

char kwh_totaal_float_char[12];
char injectie_totaal_float_char[12];
char kwh_nu_float_char[12];
char injectie_nu_float_char[12];
char gas_totaal_float_char[12];
char verbruik_char[40];
char aantal_dagen_file_char[18];
char aantal_dagen_char[5];
char module_char[20];
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];
char data_dag_char[4];
char data_maand_char[4];
char data_jaar_char[6];
char weergave_periode_char[12];
char periode_char[25];
char eenheid_char[10];
char totaal_char[10];
char kwh_sd_char[10];
char data_kwh_array_char[32][10];
char data_injectie_array_char[32][10];
char data_gas_array_char[32][10];
char periode_array_char[32][10];
char injectie_sd_char[10];
char gas_sd_char[10];
char tijd_char[12];
char jaar_char[6];
char maand_char[8];
char dag_char[4];
char elektriciteit_jaar_file_char[14];
char injectie_jaar_file_char[14];
char gas_jaar_file_char[14];
char elektriciteit_maand_file_char[14];
char injectie_maand_file_char[14];
char gas_maand_file_char[14];
char elektriciteit_dag_file_char[18];
char injectie_dag_file_char[18];
char gas_dag_file_char[18];
char elektriciteit_uur_file_char[20];
char injectie_uur_file_char[20];
char gas_uur_file_char[20];
char uitsturing_pwm_char[6];
char schakel_tijd_char[12];
char pwm_override_char[8];

float relais1_on;
float relais2_on;
float relais3_on;
float pwm_kw_float;
float data_verbruik_float[32];
float kwh_sd_float;
float injectie_sd_float;
float gas_sd_float;
float uitsturing_pwm_float = 0.0;
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 kwh_nu_float;
float injectie_nu_float;
float verbruik_nu_float;
float gas_totaal_float;
float verbruik_pwm_float;

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;
int uren_int;
int minuten_int;
int seconden_int;
int dag_int;
int maand_int;
int jaar_int;
int uren_vorig_int;
int dag_vorig_int;
int maand_vorig_int;
int jaar_vorig_int;
int data_dag_int = 20;
int data_maand_int = 10;
int data_jaar_int = 2022;
int uitsturing_pwm_int = 0;
int module_teller = 0;
int relais_module_teller = 0;
int dagen_int;

String buffer_data = "          ";
String kwh_dag = "          ";
String kwh_nacht = "          ";
String injectie_dag = "          ";
String injectie_nacht = "          ";
String kwh_nu = "          ";
String injectie_nu = "          ";
String gas = "          ";
String kwh_totaal_string = "          ";
String injectie_totaal_string = "          ";
String gas_totaal_float_string = "          ";
String verbruik_string = "          ";
String aantal_dagen_file_string = "          ";
String aantal_dagen_string = "          ";
String relais1_override = "          ";
String relais2_override = "          ";
String relais3_override = "          ";
String weergave_periode_string = "          ";
String kwh_sd_string = "          ";
String injectie_sd_string = "          ";
String gas_sd_string = "          ";
String uren_string = "          ";
String minuten_string = "          ";
String tijd_string = "          ";
String jaar_string = "          ";
String maand_string = "          ";
String dag_string = "          ";
String elektriciteit_jaar_file_string = "          ";
String injectie_jaar_file_string = "          ";
String gas_jaar_file_string = "          ";
String elektriciteit_maand_file_string = "          ";
String injectie_maand_file_string = "          ";
String gas_maand_file_string = "          ";
String elektriciteit_dag_file_string = "          ";
String injectie_dag_file_string = "          ";
String gas_dag_file_string = "          ";
String elektriciteit_uur_file_string = "          ";
String injectie_uur_file_string = "          ";
String gas_uur_file_string = "          ";
String eenheid_string = "          ";
String totaal_string = "          ";
String pwm_override = "          ";

unsigned long nu;
unsigned long relais1_vertraging_long;
unsigned long relais2_vertraging_long;
unsigned long relais3_vertraging_long;

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");
}

void testDir(fs::FS &fs, const char * dirname, uint8_t levels){
//Serial.printf("Test dir: %s\n", dirname);
File root = fs.open(dirname);
if(root){
    //Serial.println("Folder bestaat reeds");
    dir_test = true;
}
}

void testFile(fs::FS &fs, const char * path){
File file = fs.open(path);
if(file){
    file_test = true;
    file.close();
}
}

void createDir(fs::FS &fs, const char * path){
//Serial.printf("Creating Dir: %s\n", path);
if(fs.mkdir(path)){
    //Serial.println("Dir created");
} else {
    //Serial.println("mkdir failed");
}
}

void readFile(fs::FS &fs, const char * path){
//Serial.printf("Reading file: %s\n", path);
File file = fs.open(path);
if(!file){
    //Serial.println("Failed to open file for reading");
    return;
}
//Serial.print("Read from file: ");
while(file.available()){
    Serial.write(file.read());
}
file.close();
}

void read_elektriciteit_verbruik_float(fs::FS &fs, const char * path){
//Serial.printf("Reading file: %s\n", path);
File file = fs.open(path);
if(!file){
    return;
}
int teller = 0;
int index;
memset(kwh_sd_char, 0, sizeof(kwh_sd_char));
while(file.available()){
    kwh_sd_char[teller] = file.read();
    teller++;
}
file.close();
kwh_sd_string = String(kwh_sd_char);
kwh_sd_float = kwh_sd_string.toFloat();
//Serial.printf("%.3f", kwh_sd_float);
//Serial.println();
}

void read_injectie_float(fs::FS &fs, const char * path){
//Serial.printf("Reading file: %s\n", path);
File file = fs.open(path);
if(!file){
    return;
}
int teller = 0;
int index;
memset(injectie_sd_char, 0, sizeof(injectie_sd_char));
while(file.available()){
    injectie_sd_char[teller] = file.read();
}
}

```

```

        teller++;
    }
    file.close();
    injectie_sd_string = String(injectie_sd_char);
    injectie_sd_float = injectie_sd_string.toFloat();
    // Serial.printf("%.3f", injectie_sd_float);
    // Serial.println();
}

void read_gas_verbruik_float(fs::FS &fs, const char * path){
    //Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(gas_sd_char, 0, sizeof(gas_sd_char));
    while(file.available()){
        gas_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    gas_sd_string = String(gas_sd_char);
    gas_sd_float = gas_sd_string.toFloat();
    // Serial.printf("%.3f", gas_sd_float);
    // Serial.println();
}

void read_elektriciteit_verbruik_char(fs::FS &fs, const char * path){
    // Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(kwh_sd_char, 0, sizeof(kwh_sd_char));
    while(file.available()){
        kwh_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    //kwh_sd_string = String(kwh_sd_char);
    // Serial.println(kwh_sd_char);
}

void read_injectie_char(fs::FS &fs, const char * path){
    // Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(injectie_sd_char, 0, sizeof(injectie_sd_char));
    while(file.available()){
        injectie_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    // Serial.println(injectie_sd_char);
}

void read_gas_verbruik_char(fs::FS &fs, const char * path){
    // Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        return;
    }
    int teller = 0;
    int index;
    memset(gas_sd_char, 0, sizeof(gas_sd_char));
    while(file.available()){
        gas_sd_char[teller] = file.read();
        teller++;
    }
    file.close();
    // Serial.println(gas_sd_char);
}

```

```

}

void read_dagen(FS &fs, const char * path){
    File file = fs.open(path);
    int teller = 0;
    memset(aantal_dagen_char, 0, sizeof(aantal_dagen_char));
    while(file.available()){
        aantal_dagen_char[teller] = file.read();
        teller++;
    }
    file.close();
    aantal_dagen_string = String(aantal_dagen_char);
    dagen_int = aantal_dagen_string.toInt();
}

void writeFile(FS &fs, const char * path, const char * message){
    // Serial.printf("Writing file: %s\n", path);
    File file = fs.open(path, FILE_WRITE);
    if(!file){
        // Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        // Serial.println("File written");
    } else {
        // Serial.println("Write failed");
    }
    file.close();
}

void appendFile(FS &fs, const char * path, const char * message){
    // Serial.printf("Appending to file: %s\n", path);
    File file = fs.open(path, FILE_APPEND);
    if(!file){
        // Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
        // Serial.println("Message appended");
    } else {
        // Serial.println("Append failed");
    }
    file.close();
}

void dir_bestaat_test(){
    if(!jaar_dir_bestaat){
        dir_test = false;
        jaar_string = "/F" + String(jaar_int);
        jaar_string.toCharArray(jaar_char, jaar_string.length() + 1);
        testDir(SD, jaar_char, 1);
        if(dir_test == false){
            createDir(SD, jaar_char);
        }
        jaar_dir_bestaat = true;
    }
    if(!maand_dir_bestaat){
        dir_test = false;
        maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
        maand_string.toCharArray(maand_char, maand_string.length() + 1);
        testDir(SD, maand_char, 1);
        if(dir_test == false){
            createDir(SD, maand_char);
        }
        maand_dir_bestaat = true;
    }
    if(!dag_dir_bestaat){
        dir_test = false;
        dag_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int);
        dag_string.toCharArray(dag_char, dag_string.length() + 1);
        testDir(SD, dag_char, 1);
        if(dir_test == false){
            createDir(SD, dag_char);
        }
        dag_dir_bestaat = true;
    }
}

void file_bestaat_test(){

}

```

```

if(!elektriciteit_jaar_file_bestaat){
    file_test = false;
    elektriciteit_jaar_file_string = "/F" + String(jaar_int) + "/E" + jaar_int;
    elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
    testFile(SD, elektriciteit_jaar_file_char);
    if(file_test == false){
        verbruik_string = String(kwh_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, elektriciteit_jaar_file_char, verbruik_char);
    }
    elektriciteit_jaar_file_bestaat = true;
}
if(!injectie_jaar_file_bestaat){
    file_test = false;
    injectie_jaar_file_string = "/F" + String(jaar_int) + "/I" + jaar_int;
    injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
    testFile(SD, injectie_jaar_file_char);
    if(file_test == false){
        verbruik_string = String(injectie_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, injectie_jaar_file_char, verbruik_char);
    }
    injectie_jaar_file_bestaat = true;
}
if(!gas_jaar_file_bestaat){
    file_test = false;
    gas_jaar_file_string = "/F" + String(jaar_int) + "/G" + jaar_int;
    gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
    testFile(SD, gas_jaar_file_char);
    if(file_test == false){
        verbruik_string = String(gas_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, gas_jaar_file_char, verbruik_char);
    }
    gas_jaar_file_bestaat = true;
}
if(!elektriciteit_maand_file_bestaat){
    file_test = false;
    elektriciteit_maand_file_string = "/F" + String(jaar_int) + "/E" + maand_int;
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
    testFile(SD, elektriciteit_maand_file_char);
    if(file_test == false){
        verbruik_string = String(kwh_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, elektriciteit_maand_file_char, verbruik_char);
    }
    elektriciteit_maand_file_bestaat = true;
}
if(!injectie_maand_file_bestaat){
    file_test = false;
    injectie_maand_file_string = "/F" + String(jaar_int) + "/I" + maand_int;
    injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
    testFile(SD, injectie_maand_file_char);
    if(file_test == false){
        verbruik_string = String(injectie_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, injectie_maand_file_char, verbruik_char);
    }
    injectie_maand_file_bestaat = true;
}
if(!gas_maand_file_bestaat){
    file_test = false;
    gas_maand_file_string = "/F" + String(jaar_int) + "/G" + maand_int;
    gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);

    testFile(SD, gas_maand_file_char);
    if(file_test == false){
        verbruik_string = String(gas_totaal_float, 3);
        verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
        writeFile(SD, gas_maand_file_char, verbruik_char);
    }
    gas_maand_file_bestaat = true;
}
if(!elektriciteit_dag_file_bestaat){
    file_test = false;
    elektriciteit_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/E" + dag_int;
    elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
    testFile(SD, elektriciteit_dag_file_char);
    if(file_test == false){

```

```

verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_dag_file_char, verbruik_char);
}
elektriciteit_dag_file_bestaat = true;
}
if(!injectie_dag_file_bestaat){
file_test = false;
injectie_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/I" + dag_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
testFile(SD, injectie_dag_file_char);
if(file_test == false){
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_dag_file_char, verbruik_char);
}
injectie_dag_file_bestaat = true;
}
if(!gas_dag_file_bestaat){
file_test = false;
gas_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/G" + dag_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
testFile(SD, gas_dag_file_char);
if(file_test == false){
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_dag_file_char, verbruik_char);
}
gas_dag_file_bestaat = true;
}
if(!elektriciteit_uur_file_bestaat){
file_test = false;
elektriciteit_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/E" + uren_int;
elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
testFile(SD, elektriciteit_uur_file_char);
if(file_test == false){
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_uur_file_char, verbruik_char);
}
elektriciteit_uur_file_bestaat = true;
}
if(!injectie_uur_file_bestaat){
file_test = false;
injectie_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/I" + uren_int;
injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
testFile(SD, gas_uur_file_char);
if(file_test == false){
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_uur_file_char, verbruik_char);
}
injectie_uur_file_bestaat = true;
}
if(!gas_uur_file_bestaat){
file_test = false;
gas_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/G" + uren_int;
gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
testFile(SD, gas_uur_file_char);
if(file_test == false){
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_uur_file_char, verbruik_char);
}
gas_uur_file_bestaat = true;
}
}

void jaar_verbruik_injectie(){
elektriciteit_jaar_file_string = "/F" + String(jaar_vorig_int) + "/E" + jaар_vorig_int;
elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
read_elektriciteit_verbruik_float(SD, elektriciteit_jaar_file_char);
verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_jaar_file_char, verbruik_char);

injectie_jaar_file_string = "/F" + String(jaar_vorig_int) + "/I" + jaар_vorig_int;
injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
}

```

```

read_injectie_float(SD, injectie_jaar_file_char);
verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_jaar_file_char, verbruik_char);

gas_jaar_file_string = "/F" + String(jaar_vorig_int) + "/G" + jaar_vorig_int;
gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
read_gas_verbruik_float(SD, gas_jaar_file_char);
verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_jaar_file_char, verbruik_char);

jaar_string = "/F" + String(jaar_int);
jaar_string.toCharArray(jaar_char, jaar_string.length() + 1);
createDir(SD, jaar_char);

maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
maand_string.toCharArray(maand_char, maand_string.length() + 1);
createDir(SD, maand_char);

elektriciteit_jaar_file_string = "/F" + String(jaar_int) + "/E" + jaar_int;
elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_jaar_file_char, verbruik_char);

injectie_jaar_file_string = "/F" + String(jaar_int) + "/I" + jaar_int;
injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_jaar_file_char, verbruik_char);

gas_jaar_file_string = "/F" + String(jaar_int) + "/G" + jaar_int;
gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_jaar_file_char, verbruik_char);
}

void maand_verbruik_injectie(){
elektriciteit_maand_file_string = "/F" + String(jaar_vorig_int) + "/E" + maand_vorig_int;
elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
read_elektriciteit_verbruik_float(SD, elektriciteit_maand_file_char);
verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_maand_file_char, verbruik_char);

injectie_maand_file_string = "/F" + String(jaar_vorig_int) + "/I" + maand_vorig_int;
injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
read_injectie_float(SD, injectie_maand_file_char);
verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_maand_file_char, verbruik_char);

gas_maand_file_string = "/F" + String(jaar_vorig_int) + "/G" + maand_vorig_int;
gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
read_gas_verbruik_float(SD, gas_maand_file_char);
verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_maand_file_char, verbruik_char);

maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
maand_string.toCharArray(maand_char, maand_string.length() + 1);
createDir(SD, maand_char);

elektriciteit_maand_file_string = "/F" + String(jaar_int) + "/E" + maand_int;
elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_maand_file_char, verbruik_char);

elektriciteit_maand_file_bestaat = true;
injectie_maand_file_string = "/F" + String(jaar_int) + "/I" + maand_int;
injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_maand_file_char, verbruik_char);

injectie_maand_file_bestaat = true;

```

```

gas_maand_file_string = "/F" + String(jaar_int) + "/G" + maand_int;
gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_maand_file_char, verbruik_char);
gas_maand_file_bestaat = true;
}

void dag_verbruik_injectie(){
elektriciteit_dag_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/E" + dag_vorig_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
read_elektriciteit_verbruik_float(SD, elektriciteit_dag_file_char);
verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_dag_file_char, verbruik_char);

injectie_dag_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/I" + dag_vorig_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
read_injectie_float(SD, injectie_dag_file_char);
verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_dag_file_char, verbruik_char);

gas_dag_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/G" + dag_vorig_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
read_gas_verbruik_float(SD, gas_dag_file_char);
verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_dag_file_char, verbruik_char);

elektriciteit_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/E" + dag_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_dag_file_char, verbruik_char);

injectie_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/I" + dag_int;
injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_dag_file_char, verbruik_char);

gas_dag_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/G" + dag_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_dag_file_char, verbruik_char);

aantal_dagen_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/dagen";
aantal_dagen_file_string.toCharArray(aantal_dagen_file_char, aantal_dagen_file_string.length() + 1);
aantal_dagen_string = String(dag_int);
aantal_dagen_string.toCharArray(aantal_dagen_char, aantal_dagen_string.length() + 1);
writeFile(SD, aantal_dagen_file_char, aantal_dagen_char);
}

void uur_verbruik_injectie(){
elektriciteit_uur_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/F" + String(dag_vorig_int) + "/E" +
uren_vorig_int;
elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
read_elektriciteit_verbruik_float(SD, elektriciteit_uur_file_char);
verbruik_string = String((kwh_totaal_float - kwh_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_uur_file_char, verbruik_char);

injectie_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_vorig_int) + "/F" + String(dag_vorig_int) + "/I" + uren_vorig_int;
injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
read_injectie_float(SD, injectie_uur_file_char);
verbruik_string = String((injectie_totaal_float - injectie_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_uur_file_char, verbruik_char);

gas_uur_file_string = "/F" + String(jaar_vorig_int) + "/F" + String(maand_vorig_int) + "/F" + String(dag_vorig_int) + "/G" + uren_vorig_int;
gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
read_gas_verbruik_float(SD, gas_uur_file_char);
verbruik_string = String((gas_totaal_float - gas_sd_float), 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_uur_file_char, verbruik_char);

elektriciteit_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/E" + uren_int;

```

```

elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
verbruik_string = String(kwh_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, elektriciteit_uur_file_char, verbruik_char);

injectie_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/I" + uren_int;
injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
verbruik_string = String(injectie_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, injectie_uur_file_char, verbruik_char);

gas_uur_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int) + "/G" + uren_int;
gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
verbruik_string = String(gas_totaal_float, 3);
verbruik_string.toCharArray(verbruik_char, verbruik_string.length() + 1);
writeFile(SD, gas_uur_file_char, verbruik_char);
}

void lees_uur_verbruik_injectie(){
int eind_uur_int;
if((data_jaar_int == jaar_int) && (data_maand_int == maand_int) && (data_dag_int == dag_int)){
    eind_uur_int = uren_int;
}
else{
    eind_uur_int = 24;
}
eenheid_string = "uur";
eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
for(int lees_uur_int = 0; lees_uur_int < eind_uur_int; lees_uur_int++){
    sprintf(periode_array_char[lees_uur_int], "%02d", lees_uur_int);
    elektriciteit_uur_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/F" + String(data_dag_int) + "/E" + lees_uur_int;
    elektriciteit_uur_file_string.toCharArray(elektriciteit_uur_file_char, elektriciteit_uur_file_string.length() + 1);
    file_test = false;
    testFile(SD, elektriciteit_uur_file_char);
    if(file_test == true){
        read_elektriciteit_verbruik_char(SD, elektriciteit_uur_file_char);
        strcpy(data_kwh_array_char[lees_uur_int], kwh_sd_char);
    }
    else{
        memset(data_kwh_array_char[lees_uur_int], 0, sizeof(data_kwh_array_char[lees_uur_int]));
    }
    injectie_uur_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/F" + String(data_dag_int) + "/I" + lees_uur_int;
    injectie_uur_file_string.toCharArray(injectie_uur_file_char, injectie_uur_file_string.length() + 1);
    file_test = false;
    testFile(SD, injectie_uur_file_char);
    if(file_test == true){
        read_injectie_char(SD, injectie_uur_file_char);
        strcpy(data_injectie_array_char[lees_uur_int], injectie_sd_char);
    }
    else{
        memset(data_injectie_array_char[lees_uur_int], 0, sizeof(data_injectie_array_char[lees_uur_int]));
    }
    gas_uur_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/F" + String(data_dag_int) + "/G" + lees_uur_int;
    gas_uur_file_string.toCharArray(gas_uur_file_char, gas_uur_file_string.length() + 1);
    file_test = false;
    testFile(SD, gas_uur_file_char);
    if(file_test == true){
        read_gas_verbruik_char(SD, gas_uur_file_char);
        strcpy(data_gas_array_char[lees_uur_int], gas_sd_char);
    }
    else{
        memset(data_gas_array_char[lees_uur_int], 0, sizeof(data_gas_array_char[lees_uur_int]));
    }
}
totaal_string = "totaal";
totaal_string.toCharArray(totaal_char, totaal_string.length() + 1);
strcpy(periode_array_char[24], totaal_char);
elektriciteit_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/E" + data_dag_int;
elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
file_test = false;
testFile(SD, elektriciteit_dag_file_char);
if((file_test == true) && (eind_uur_int == 24)){
    read_elektriciteit_verbruik_char(SD, elektriciteit_dag_file_char);
    strcpy(data_kwh_array_char[24], kwh_sd_char);
}
else{
    memset(data_kwh_array_char[24], 0, sizeof(data_kwh_array_char[24]));
}
injectie_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/I" + data_dag_int;

```

```

injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
file_test = false;
testFile(SD, injectie_dag_file_char);
if((file_test == true) && (eind_uur_int == 24)){
    read_injectie_char(SD, injectie_dag_file_char);
    strcpy(data_injectie_array_char[24], injectie_sd_char);
}
else{
    memset(data_injectie_array_char[24], 0, sizeof(data_injectie_array_char[24]));
}
gas_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/G" + data_dag_int;
gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
file_test = false;
testFile(SD, gas_dag_file_char);
if((file_test == true) && (eind_uur_int == 24)){
    read_gas_verbruik_char(SD, gas_dag_file_char);
    strcpy(data_gas_array_char[24], gas_sd_char);
}
else{
    memset(data_gas_array_char[24], 0, sizeof(data_gas_array_char[24]));
}
for(int x = eind_uur_int + 1; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}

void lees_dag_verbruik_injectie(){
    bool volledige_maand_bool = true;
    if((data_jaar_int == jaar_int) && (data_maand_int == maand_int)){
        bool volledige_maand_bool = false;
    }
    eenheid_string = "dag";
    eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
    aantal_dagen_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/dagen";
    aantal_dagen_file_string.toCharArray(aantal_dagen_file_char, aantal_dagen_file_string.length() + 1);
    file_test = false;
    testFile(SD, aantal_dagen_file_char);
    if(file_test == true){
        read_dagen(SD, aantal_dagen_file_char);
        if(volledige_maand_bool == false){
            dagen_int = dag_int - 1;
        }
        for(int lees_dag_int = 0; lees_dag_int < dagen_int; lees_dag_int++){
            sprintf(periode_array_char[lees_dag_int], "%02d", lees_dag_int + 1);
            elektriciteit_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/E" + (lees_dag_int + 1);
            elektriciteit_dag_file_string.toCharArray(elektriciteit_dag_file_char, elektriciteit_dag_file_string.length() + 1);
            file_test = false;
            testFile(SD, elektriciteit_dag_file_char);
            if(file_test == true){
                read_elektriciteit_verbruik_char(SD, elektriciteit_dag_file_char);
                strcpy(data_kwh_array_char[lees_dag_int], kwh_sd_char);
            }
            else{
                memset(data_kwh_array_char[lees_dag_int], 0, sizeof(data_kwh_array_char[lees_dag_int]));
            }
            injectie_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/I" + (lees_dag_int + 1);
            injectie_dag_file_string.toCharArray(injectie_dag_file_char, injectie_dag_file_string.length() + 1);
            file_test = false;
            testFile(SD, injectie_dag_file_char);
            if(file_test == true){
                read_injectie_char(SD, injectie_dag_file_char);
                strcpy(data_injectie_array_char[lees_dag_int], injectie_sd_char);
            }
            else{
                memset(data_injectie_array_char[lees_dag_int], 0, sizeof(data_injectie_array_char[lees_dag_int]));
            }
            gas_dag_file_string = "/F" + String(data_jaar_int) + "/F" + String(data_maand_int) + "/G" + (lees_dag_int + 1);
            gas_dag_file_string.toCharArray(gas_dag_file_char, gas_dag_file_string.length() + 1);
            file_test = false;
            testFile(SD, gas_dag_file_char);
            if(file_test == true){
                read_gas_verbruik_char(SD, gas_dag_file_char);
                strcpy(data_gas_array_char[lees_dag_int], gas_sd_char);
            }
            else{
                memset(data_gas_array_char[lees_dag_int], 0, sizeof(data_gas_array_char[lees_dag_int]));
            }
        }
    }
}

```

```

        }
    }
    if(volledige_maand_bool == true){
        totaal_string = "totaal";
        totaal_string.toCharArray(totaal_char, totaal_string.length() + 1);
        strcpy(periode_array_char[dagen_int], totaal_char);
        elektriciteit_maand_file_string = "/F" + String(data_jaar_int) + "/E" + data_maand_int;
        elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
        file_test = false;
        testFile(SD, elektriciteit_maand_file_char);
        if((file_test == true) && (volledige_maand_bool == true)){
            read_elektriciteit_verbruik_char(SD, elektriciteit_maand_file_char);
            strcpy(data_kwh_array_char[dagen_int], kwh_sd_char);
        }
        else{
            memset(data_kwh_array_char[dagen_int], 0, sizeof(data_kwh_array_char[dagen_int]));
        }
        injectie_maand_file_string = "/F" + String(data_jaar_int) + "/I" + data_maand_int;
        injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
        file_test = false;
        testFile(SD, injectie_maand_file_char);
        if((file_test == true) && (volledige_maand_bool == true)){
            read_injectie_char(SD, injectie_maand_file_char);
            strcpy(data_injectie_array_char[dagen_int], injectie_sd_char);
        }
        else{
            memset(data_injectie_array_char[dagen_int], 0, sizeof(data_injectie_array_char[dagen_int]));
        }
        gas_maand_file_string = "/F" + String(data_jaar_int) + "/G" + data_maand_int;
        gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
        file_test = false;
        testFile(SD, gas_maand_file_char);
        if((file_test == true) && (volledige_maand_bool == true)){
            read_gas_verbruik_char(SD, gas_maand_file_char);
            strcpy(data_gas_array_char[dagen_int], gas_sd_char);
        }
        else{
            memset(data_gas_array_char[dagen_int], 0, sizeof(data_gas_array_char[dagen_int]));
        }
    }
    for(int x = dagen_int + 1; x < 32; x++){
        memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
        memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
        memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
        memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
    }
}
else{
    for(int x = 0; x < 32; x++){
        sprintf(periode_array_char[x], "%02d", (x + 1));
        memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
        memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
        memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
    }
}
}

void lees_maand_verbruik_injectie(){
int eind_maand_int;
if(data_jaar_int == jaar_int){
    eind_maand_int = maand_int - 1;
}
else{
    eind_maand_int = 13;
}
eenheid_string = "maand";
eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
for(int lees_maand_int = 0; lees_maand_int < eind_maand_int; lees_maand_int++){
    sprintf(periode_array_char[lees_maand_int], "%02d", lees_maand_int + 1);
    elektriciteit_maand_file_string = "/F" + String(data_jaar_int) + "/E" + (lees_maand_int + 1);
    elektriciteit_maand_file_string.toCharArray(elektriciteit_maand_file_char, elektriciteit_maand_file_string.length() + 1);
    file_test = false;
    testFile(SD, elektriciteit_maand_file_char);
    if(file_test == true){
        read_elektriciteit_verbruik_char(SD, elektriciteit_maand_file_char);
        strcpy(data_kwh_array_char[lees_maand_int], kwh_sd_char);
    }
    else{
        memset(data_kwh_array_char[lees_maand_int], 0, sizeof(data_kwh_array_char[lees_maand_int]));
    }
}
}

```

```

}

injectie_maand_file_string = "/F" + String(data_jaar_int) + "/I" + (lees_maand_int + 1);
injectie_maand_file_string.toCharArray(injectie_maand_file_char, injectie_maand_file_string.length() + 1);
file_test = false;
testFile(SD, injectie_maand_file_char);
if(file_test == true){
    read_injectie_char(SD, injectie_maand_file_char);
    strcpy(data_injectie_array_char[lees_maand_int], injectie_sd_char);
}
else{
    memset(data_injectie_array_char[lees_maand_int], 0, sizeof(data_injectie_array_char[lees_maand_int]));
}
gas_maand_file_string = "/F" + String(data_jaar_int) + "/G" + (lees_maand_int + 1);
gas_maand_file_string.toCharArray(gas_maand_file_char, gas_maand_file_string.length() + 1);
file_test = false;
testFile(SD, gas_maand_file_char);
if(file_test == true){
    read_gas_verbruik_char(SD, gas_maand_file_char);
    strcpy(data_gas_array_char[lees_maand_int], gas_sd_char);
}
else{
    memset(data_gas_array_char[lees_maand_int], 0, sizeof(data_gas_array_char[lees_maand_int]));
}
}
totaal_string = "totaal";
totaal_string.toCharArray(totaal_char, totaal_string.length() + 1);
strcpy(periode_array_char[12], totaal_char);
elektriciteit_jaar_file_string = "/F" + String(data_jaar_int) + "/E" + data_jaar_int;
elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, elektriciteit_jaar_file_char);
if((file_test == true) && (eind_maand_int == 13)){
    read_elektriciteit_verbruik_char(SD, elektriciteit_jaar_file_char);
    strcpy(data_kwh_array_char[12], kwh_sd_char);
}
else{
    memset(data_kwh_array_char[12], 0, sizeof(data_kwh_array_char[dagen_int]));
}
injectie_jaar_file_string = "/F" + String(data_jaar_int) + "/I" + data_jaar_int;
injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, injectie_jaar_file_char);
if((file_test == true) && (eind_maand_int == 13)){
    read_injectie_char(SD, injectie_jaar_file_char);
    strcpy(data_injectie_array_char[12], injectie_sd_char);
}
else{
    memset(data_injectie_array_char[12], 0, sizeof(data_injectie_array_char[dagen_int]));
}
gas_jaar_file_string = "/F" + String(data_jaar_int) + "/G" + data_jaar_int;
gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, gas_jaar_file_char);
if((file_test == true) && (eind_maand_int == 13)){
    read_gas_verbruik_char(SD, gas_jaar_file_char);
    strcpy(data_gas_array_char[12], gas_sd_char);
}
else{
    memset(data_gas_array_char[12], 0, sizeof(data_gas_array_char[dagen_int]));
}
for(int x = eind_maand_int; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}

void lees_jaar_verbruik_injectie(){
    int lees_jaar = jaar_int - 25;
    int x = 0;
    eenheid_string = "jaar";
    eenheid_string.toCharArray(eenheid_char, eenheid_string.length() + 1);
    for(int lees_jaar_int = lees_jaar; lees_jaar_int < jaar_int; lees_jaar_int++){
        sprintf(periode_array_char[x], "%04d", lees_jaar_int);
        elektriciteit_jaar_file_string = "/F" + String(lees_jaar_int) + "/E" + lees_jaar_int;
        elektriciteit_jaar_file_string.toCharArray(elektriciteit_jaar_file_char, elektriciteit_jaar_file_string.length() + 1);
        file_test = false;
        testFile(SD, elektriciteit_jaar_file_char);
    }
}

```

```

if(file_test == true){
    read_elektriciteit_verbruik_char(SD, elektriciteit_jaar_file_char);
    strcpy(data_kwh_array_char[x], kwh_sd_char);
}
else{
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[lees_jaar_int]));
}
injectie_jaar_file_string = "/F" + String(lees_jaar_int) + "/I" + lees_jaar_int;
injectie_jaar_file_string.toCharArray(injectie_jaar_file_char, injectie_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, injectie_jaar_file_char);
if(file_test == true){
    read_injectie_char(SD, injectie_jaar_file_char);
    strcpy(data_injectie_array_char[x], injectie_sd_char);
}
else{
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[lees_jaar_int]));
}
gas_jaar_file_string = "/F" + String(lees_jaar_int) + "/G" + lees_jaar_int;
gas_jaar_file_string.toCharArray(gas_jaar_file_char, gas_jaar_file_string.length() + 1);
file_test = false;
testFile(SD, gas_jaar_file_char);
if(file_test == true){
    read_gas_verbruik_char(SD, gas_jaar_file_char);
    strcpy(data_gas_array_char[x], gas_sd_char);
}
else{
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[lees_jaar_int]));
}
x++;
}
for(int x = 25; x < 32; x++){
    memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
    memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
    memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
    memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
}
}

```

```

const char energie_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
    <iframe style="display:none" name="hidden-form"></iframe>
    <title>Energie Beheer</title>
    <meta name="viewport" content="width=device-width, initial-scale=0.85">
    <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>
</head>
<body>
    <div class="kader">
        <div class="links">
            <div class="links_midden">
                <div>
                    <input type="button" value="Perioden">
                    <input type="button" value="Gasverbruik">
                    <input type="button" value="Injectie">
                    <input type="button" value="Elektriciteit">
                </div>
            </div>
        </div>
        <div class="midden">
            <div>
                <table border="1">
                    <tr>
                        <td>Perioden</td>
                        <td>Gasverbruik</td>
                        <td>Injectie</td>
                        <td>Elektriciteit</td>
                    </tr>
                </table>
            </div>
        </div>
        <div class="titel">Energie Beheer</div>
        <div class="bottom">
            <input type="button" value="Terug naar menu" />
        </div>
    </div>
</body>
</html>
)rawliteral";

```

```

</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% &nbsp; KWh</div>
</div>
<br>
<div class="kader">
<div class="links">Totaal injectie : </div>
<div class="midden">%injectie_totaal% &nbsp; KWh</div>
</div>
<br>
<div class="kader">
<div class="links">Verbruik nu : </div>
<div class="midden">%kwh_nu% &nbsp; KW</div>
</div>
<br>
<div class="kader">
<div class="links">Injectie nu : </div>
<div class="midden">%injectie_nu% &nbsp; KW</div>
</div>
<br>
<div class="kader">
<div class="links">Totaal gas : </div>
<div class="midden">%gas_totaal% &nbsp; m3</div>
</div>
<br><br>
<div class="titel"><b><center>Datum</center></b></div>
<center>
<input type="text" style="text-align:center;" value="%dag%" size=1>
&nbsp;<input type="text" style="text-align:center;" value="%maand%" size=1>
&nbsp;<input type="text" style="text-align:center;" value="%jaar%" size=1>
</center>
<br>
<div class="titel"><b><center>Tijd</center></b></div>
<center><input type="text" style="text-align:center;" value="%tijd%" size=2></center>
<br>
<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>&nbsp;<input type="text" style="text-align:center;" value="%kw_on%" name="input_kw_on" size=1>
&nbsp; <b>V:</b>&nbsp;<input type="text" style="text-align:center;" value="%delay%" name="input_delay" size=1>
&nbsp; <b>Tijd:</b>&nbsp;<input type="text" style="text-align:center;" value="%schakel_tijd%" name="input_schakel_tijd" size=1>
&nbsp; <b>A/M:</b>&nbsp;<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()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="relais_module_plus" value=" + " onclick="ok()">
&nbsp;&nbsp;&nbsp;
<input type="submit" name="relais_module_bevestig" value="OK" onclick="ok()">
</center>
</form>
<br><br>
<form action="/get" target="hidden-form">
<div class="titel"><b><center>PWM sturing instellen</center></b></div>
<center>
<b>KW:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_kw%" name="input_pwm_kw" size=1>
&nbsp;<b>1:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_on%" name="input_pwm_tijd_on" size=1>
&nbsp;<b>0:</b>&nbsp;<input type="text" style="text-align:center;" value="%pwm_tijd_off%" name="input_pwm_tijd_off" size=1>
&nbsp;<b>A/M:</b>&nbsp;<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%</div>
</div>
<br><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>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_1%" name="input_macx_1" size=1>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_2%" name="input_macx_2" size=1>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_3%" name="input_macx_3" size=1>
    &nbsp;
    <input type="text" style="text-align:center;" value="%display_macx_4%" name="input_macx_4" size=1>
    &nbsp;
    <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()">
    &nbsp;&nbsp;&nbsp;
    <input type="submit" name="module_plus" value=" + " onclick="ok()">
    &nbsp;&nbsp;&nbsp;
    <input type="submit" name="module_bevestig" value="OK" onclick="ok()">
  </center>
</form>
<br><br><br>
<center>
  <button type="button" onclick="data_weergave()">Data weergave</button>
</center>
</small>
<br>
<br>
<br>
<h6><b>thieu-b55 oktober 2022</b></h6>
<script>
  function ok(){
    setTimeout(function(){document.location.reload()},250);
  }
</script>
<script>
  function data_weergave(){
    location.assign("http://192.168.4.1/data/");
  }
</script>
</body>
</html>
)rawliteral";

```

```

const char data_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
  <iframe style="display:none" name="hidden-form"></iframe>
  <meta name="viewport" content="width=device-width, initial-scale=0.85">
  <title>Data weergave</title>
<style>
  div.kader {
    position: relative;
    width: 400px;
    height: 12x;

```

```

}
div.links{
position: absolute;
left : 0px;
width: auto;
height: 12px;
}
div.links_midden{
position:absolute;
left: 80px;
width: auto;
height: 12px;
}
div.rechts_midden{
position: absolute;
left: 200px;
width: auto;
height: 12px;
}
div.rechts{
position: absolute;
left: 320px;
width: auto;
height: 12px;
}
div.titel{
height: 25px;
width: auto;
}
div.bottom{
position: fixed;
bottom: 0px;
}
div.data_links{
position: absolute;
left: 0px;
width: 80px;
height: 12px
}
div.data_links_midden{
position: absolute;
left: 80px;
width: 120px;
height: 12px
}
div.data_rechts_midden{
position: absolute;
left: 200px;
width: 120px;
height: 12px
}
div.data_rechts{
position: absolute;
left: 320px;
width: 80px;
height: 12px
}
div.blanco_20{
width: auto;
height: 20px;
}
div.blanco_15{
width: auto;
height: 15px;
}
</style>
</head>
<body>
<h3><center> Data weergave </center></h3>
<center>
<button type="button" onclick="start_pagina()">Naar begin pagina</button>
</center>
<br>
<form action="/get" target="hidden-form">
<center>
<input type="text" style="text-align:center;" value="%data_dag%" name="input_data_dag" size=1>
&ampnbsp<input type="text" style="text-align:center;" value="%data_maand%" name="input_data_maand" size=1>
&ampnbsp<input type="text" style="text-align:center;" value="%data_jaar%" name="input_data_jaar" size=1>
</center>

```

```
<br>
<center>
<input type="submit" name="bevestig_periode" value="OK" onclick="ok()">
</center>
<br>
</form>
<small>
<div class="titel"><center><b>Periode : %periode%</b></center></div>
<div class="kader"><b>
<div class="links">%eenheid% </div>
<div class="links_midden">verbruik kWh</div>
<div class="rechts_midden">injectie kWh</div>
<div class="rechts">gas m3</div></b>
</div>
<div class="blanco_20">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_1%</div>
<div class="data_links_midden">%verbruik_1%</div>
<div class="data_rechts_midden">%injectie_1%</div>
<div class="data_rechts">%gas_verbruik_1%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_2%</div>
<div class="data_links_midden">%verbruik_2%</div>
<div class="data_rechts_midden">%injectie_2%</div>
<div class="data_rechts">%gas_verbruik_2%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_3%</div>
<div class="data_links_midden">%verbruik_3%</div>
<div class="data_rechts_midden">%injectie_3%</div>
<div class="data_rechts">%gas_verbruik_3%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_4%</div>
<div class="data_links_midden">%verbruik_4%</div>
<div class="data_rechts_midden">%injectie_4%</div>
<div class="data_rechts">%gas_verbruik_4%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_5%</div>
<div class="data_links_midden">%verbruik_5%</div>
<div class="data_rechts_midden">%injectie_5%</div>
<div class="data_rechts">%gas_verbruik_5%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_6%</div>
<div class="data_links_midden">%verbruik_6%</div>
<div class="data_rechts_midden">%injectie_6%</div>
<div class="data_rechts">%gas_verbruik_6%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_7%</div>
<div class="data_links_midden">%verbruik_7%</div>
<div class="data_rechts_midden">%injectie_7%</div>
<div class="data_rechts">%gas_verbruik_7%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_8%</div>
<div class="data_links_midden">%verbruik_8%</div>
<div class="data_rechts_midden">%injectie_8%</div>
<div class="data_rechts">%gas_verbruik_8%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
<div class="data_links">%periode_9%</div>
<div class="data_links_midden">%verbruik_9%</div>
<div class="data_rechts_midden">%injectie_9%</div>
<div class="data_rechts">%gas_verbruik_9%</div>
</div>
<div class="blanco_15">&ampnbsp</div>
<div class="kader">
```





```

<div class="data_links">%periode_31%</div>
<div class="data_links_midden">%verbruik_31%</div>
<div class="data_rechts_midden">%injectie_31%</div>
<div class="data_rechts">%gas_verbruik_31%</div>
</div>
<div class="blanco_15">&nbsp;</div>
</div>
<div class="kader">
<div class="data_links">%periode_32%</div>
<div class="data_links_midden">%verbruik_32%</div>
<div class="data_rechts_midden">%injectie_32%</div>
<div class="data_rechts">%gas_verbruik_32%</div>
</div>
</small>
<script>
    function start_pagina(){
        location.assign("http://192.168.4.1/");
    }
    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 == "kwh_nu"){
        temp = String(kwh_nu_float, 3);
        temp.toCharArray(kwh_nu_float_char, (temp.length() + 1));
        return(kwh_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 == "dag"){
        sprintf(dag_char, "%02d", dag_int);
        return(dag_char);
    }
    if(var == "maand"){
        sprintf(maand_char, "%02d", maand_int);
        return(maand_char);
    }
    if(var == "jaar"){
        sprintf(jaar_char, "%04d", jaar_int);
        return(jaar_char);
    }
    if(var == "tijd"){
        sprintf(tijd_char, "%02d:%02d", uren_int, minuten_int);
        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);
}
if(var == "data_dag"){
    sprintf(data_dag_char, "%02d", data_dag_int);
    return(data_dag_char);
}
if(var == "data_maand"){
    sprintf(data_maand_char, "%02d", data_maand_int);
    return(data_maand_char);
}
if(var == "data_jaar"){
    sprintf(data_jaar_char, "%04d", data_jaar_int);
    return(data_jaar_char);
}
if(var == "periode"){
    return(periode_char);
}
if(var == "eenheid"){
    return(eenheid_char);
}
if(var == "periode_1"){
    return(periode_array_char[0]);
}
if(var == "periode_2"){
    return(periode_array_char[1]);
}
if(var == "periode_3"){
    return(periode_array_char[2]);
}

```

```

}

if(var == "periode_4"){
    return(periode_array_char[3]);
}
if(var == "periode_5"){
    return(periode_array_char[4]);
}
if(var == "periode_6"){
    return(periode_array_char[5]);
}
if(var == "periode_7"){
    return(periode_array_char[6]);
}
if(var == "periode_8"){
    return(periode_array_char[7]);
}
if(var == "periode_9"){
    return(periode_array_char[8]);
}
if(var == "periode_10"){
    return(periode_array_char[9]);
}
if(var == "periode_11"){
    return(periode_array_char[10]);
}
if(var == "periode_12"){
    return(periode_array_char[11]);
}
if(var == "periode_13"){
    return(periode_array_char[12]);
}
if(var == "periode_14"){
    return(periode_array_char[13]);
}
if(var == "periode_15"){
    return(periode_array_char[14]);
}
if(var == "periode_16"){
    return(periode_array_char[15]);
}
if(var == "periode_17"){
    return(periode_array_char[16]);
}
if(var == "periode_18"){
    return(periode_array_char[17]);
}
if(var == "periode_19"){
    return(periode_array_char[18]);
}
if(var == "periode_20"){
    return(periode_array_char[19]);
}
if(var == "periode_21"){
    return(periode_array_char[20]);
}
if(var == "periode_22"){
    return(periode_array_char[21]);
}
if(var == "periode_23"){
    return(periode_array_char[22]);
}
if(var == "periode_24"){
    return(periode_array_char[23]);
}
if(var == "periode_25"){
    return(periode_array_char[24]);
}
if(var == "periode_26"){
    return(periode_array_char[25]);
}
if(var == "periode_27"){
    return(periode_array_char[26]);
}
if(var == "periode_28"){
    return(periode_array_char[27]);
}
if(var == "periode_29"){
    return(periode_array_char[28]);
}

```

```

if(var == "periode_30"){
    return(periode_array_char[29]);
}
if(var == "periode_31"){
    return(periode_array_char[30]);
}
if(var == "periode_32"){
    return(periode_array_char[31]);
}
if(var == "verbruik_1"){
    return(data_kwh_array_char[0]);
}
if(var == "verbruik_2"){
    return(data_kwh_array_char[1]);
}
if(var == "verbruik_3"){
    return(data_kwh_array_char[2]);
}
if(var == "verbruik_4"){
    return(data_kwh_array_char[3]);
}
if(var == "verbruik_5"){
    return(data_kwh_array_char[4]);
}
if(var == "verbruik_6"){
    return(data_kwh_array_char[5]);
}
if(var == "verbruik_7"){
    return(data_kwh_array_char[6]);
}
if(var == "verbruik_8"){
    return(data_kwh_array_char[7]);
}
if(var == "verbruik_9"){
    return(data_kwh_array_char[8]);
}
if(var == "verbruik_10"){
    return(data_kwh_array_char[9]);
}
if(var == "verbruik_11"){
    return(data_kwh_array_char[10]);
}
if(var == "verbruik_12"){
    return(data_kwh_array_char[11]);
}
if(var == "verbruik_13"){
    return(data_kwh_array_char[12]);
}
if(var == "verbruik_14"){
    return(data_kwh_array_char[13]);
}
if(var == "verbruik_15"){
    return(data_kwh_array_char[14]);
}
if(var == "verbruik_16"){
    return(data_kwh_array_char[15]);
}
if(var == "verbruik_17"){
    return(data_kwh_array_char[16]);
}
if(var == "verbruik_18"){
    return(data_kwh_array_char[17]);
}
if(var == "verbruik_19"){
    return(data_kwh_array_char[18]);
}
if(var == "verbruik_20"){
    return(data_kwh_array_char[19]);
}
if(var == "verbruik_21"){
    return(data_kwh_array_char[20]);
}
if(var == "verbruik_22"){
    return(data_kwh_array_char[21]);
}
if(var == "verbruik_23"){
    return(data_kwh_array_char[22]);
}
if(var == "verbruik_24"){

```

```
    return(data_kwh_array_char[23]);
}
if(var == "verbruik_25"){
    return(data_kwh_array_char[24]);
}
if(var == "verbruik_26"){
    return(data_kwh_array_char[25]);
}
if(var == "verbruik_27"){
    return(data_kwh_array_char[26]);
}
if(var == "verbruik_28"){
    return(data_kwh_array_char[27]);
}
if(var == "verbruik_29"){
    return(data_kwh_array_char[28]);
}
if(var == "verbruik_30"){
    return(data_kwh_array_char[29]);
}
if(var == "verbruik_31"){
    return(data_kwh_array_char[30]);
}
if(var == "verbruik_32"){
    return(data_kwh_array_char[31]);
}
if(var == "injectie_1"){
    return(data_injectie_array_char[0]);
}
if(var == "injectie_2"){
    return(data_injectie_array_char[1]);
}
if(var == "injectie_3"){
    return(data_injectie_array_char[2]);
}
if(var == "injectie_4"){
    return(data_injectie_array_char[3]);
}
if(var == "injectie_5"){
    return(data_injectie_array_char[4]);
}
if(var == "injectie_6"){
    return(data_injectie_array_char[5]);
}
if(var == "injectie_7"){
    return(data_injectie_array_char[6]);
}
if(var == "injectie_8"){
    return(data_injectie_array_char[7]);
}
if(var == "injectie_9"){
    return(data_injectie_array_char[8]);
}
if(var == "injectie_10"){
    return(data_injectie_array_char[9]);
}
if(var == "injectie_11"){
    return(data_injectie_array_char[10]);
}
if(var == "injectie_12"){
    return(data_injectie_array_char[11]);
}
if(var == "injectie_13"){
    return(data_injectie_array_char[12]);
}
if(var == "injectie_14"){
    return(data_injectie_array_char[13]);
}
if(var == "injectie_15"){
    return(data_injectie_array_char[14]);
}
if(var == "injectie_16"){
    return(data_injectie_array_char[15]);
}
if(var == "injectie_17"){
    return(data_injectie_array_char[16]);
}
if(var == "injectie_18"){
    return(data_injectie_array_char[17]);
}
```

```

}

if(var == "injectie_19"){
    return(data_injectie_array_char[18]);
}
if(var == "injectie_20"){
    return(data_injectie_array_char[19]);
}
if(var == "injectie_21"){
    return(data_injectie_array_char[20]);
}
if(var == "injectie_22"){
    return(data_injectie_array_char[21]);
}
if(var == "injectie_23"){
    return(data_injectie_array_char[22]);
}
if(var == "injectie_24"){
    return(data_injectie_array_char[23]);
}
if(var == "injectie_25"){
    return(data_injectie_array_char[24]);
}
if(var == "injectie_26"){
    return(data_injectie_array_char[25]);
}
if(var == "injectie_27"){
    return(data_injectie_array_char[26]);
}
if(var == "injectie_28"){
    return(data_injectie_array_char[27]);
}
if(var == "injectie_29"){
    return(data_injectie_array_char[28]);
}
if(var == "injectie_30"){
    return(data_injectie_array_char[29]);
}
if(var == "injectie_31"){
    return(data_injectie_array_char[30]);
}
if(var == "injectie_32"){
    return(data_injectie_array_char[31]);
}
if(var == "gas_verbruik_1"){
    return(data_gas_array_char[0]);
}
if(var == "gas_verbruik_2"){
    return(data_gas_array_char[1]);
}
if(var == "gas_verbruik_3"){
    return(data_gas_array_char[2]);
}
if(var == "gas_verbruik_4"){
    return(data_gas_array_char[3]);
}
if(var == "gas_verbruik_5"){
    return(data_gas_array_char[4]);
}
if(var == "gas_verbruik_6"){
    return(data_gas_array_char[5]);
}
if(var == "gas_verbruik_7"){
    return(data_gas_array_char[6]);
}
if(var == "gas_verbruik_8"){
    return(data_gas_array_char[7]);
}
if(var == "gas_verbruik_9"){
    return(data_gas_array_char[8]);
}
if(var == "gas_verbruik_10"){
    return(data_gas_array_char[9]);
}
if(var == "gas_verbruik_11"){
    return(data_gas_array_char[10]);
}
if(var == "gas_verbruik_12"){
    return(data_gas_array_char[11]);
}

```

```

if(var == "gas_verbruik_13"){
    return(data_gas_array_char[12]);
}
if(var == "gas_verbruik_14"){
    return(data_gas_array_char[13]);
}
if(var == "gas_verbruik_15"){
    return(data_gas_array_char[14]);
}
if(var == "gas_verbruik_16"){
    return(data_gas_array_char[15]);
}
if(var == "gas_verbruik_17"){
    return(data_gas_array_char[16]);
}
if(var == "gas_verbruik_18"){
    return(data_gas_array_char[17]);
}
if(var == "gas_verbruik_19"){
    return(data_gas_array_char[18]);
}
if(var == "gas_verbruik_20"){
    return(data_gas_array_char[19]);
}
if(var == "gas_verbruik_21"){
    return(data_gas_array_char[20]);
}
if(var == "gas_verbruik_22"){
    return(data_gas_array_char[21]);
}
if(var == "gas_verbruik_23"){
    return(data_gas_array_char[22]);
}
if(var == "gas_verbruik_24"){
    return(data_gas_array_char[23]);
}
if(var == "gas_verbruik_25"){
    return(data_gas_array_char[24]);
}
if(var == "gas_verbruik_26"){
    return(data_gas_array_char[25]);
}
if(var == "gas_verbruik_27"){
    return(data_gas_array_char[26]);
}
if(var == "gas_verbruik_28"){
    return(data_gas_array_char[27]);
}
if(var == "gas_verbruik_29"){
    return(data_gas_array_char[28]);
}
if(var == "gas_verbruik_30"){
    return(data_gas_array_char[29]);
}
if(var == "gas_verbruik_31"){
    return(data_gas_array_char[30]);
}
if(var == "gas_verbruik_32"){
    return(data_gas_array_char[31]);
}
}

void html_input(){
server.begin();
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
request->send_P(200, "text/html", energie_html, processor);
});
server.on("/data", HTTP_GET, [](AsyncWebServerRequest *request){
request->send_P(200, "text/html", data_html, processor);
});
server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request{
    char terminator = char(0xa);
    String temp = "      ";
    String uren_string = "      ";
    String minuten_string = "      ";
    char temp_char[30];
    float kw_on;
    float kw_off;
    String override = "      ";

```

```

int schakel_delay;
char char_temp[10];
bool fout;
int uren_int;
int minuten_int;

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.relaais = 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.relaais = 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.relaais = 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.relaais = 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.relaais = 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;
}
/*
 * tijdsturing uitchakelen bij Manueel 0 tijdens tijdsturing
 */
if(pwm_override == "0"){
    pwm_tijd_gezet = false;
}
}
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_MINUS)){
    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");
broadcastAddress3[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;
}
delay(2000);
ESP.restart();
}
}
if(request->hasParam(INPUT_DATA_DAG)){
temp = ((request->getParam(INPUT_DATA_DAG)->value()) + String(terminator));
data_dag_int = temp.toInt();
}

```

```

if(request->hasParam(INPUT_DATA_MAAND)){
    temp = ((request->getParam(INPUT_DATA_MAAND)->value() + String(terminator));
    data_maand_int = temp.toInt();
}
if(request->hasParam(INPUT_DATA_JAAR)){
    temp = ((request->getParam(INPUT_DATA_JAAR)->value() + String(terminator));
    data_jaar_int = temp.toInt();
}
if(request->hasParam(BEVESTIG_PERIODE)){
    for(int x = 0; x < 32; x++){
        memset(periode_array_char[x], 0, sizeof(periode_array_char[x]));
        memset(data_kwh_array_char[x], 0, sizeof(data_kwh_array_char[x]));
        memset(data_injectie_array_char[x], 0, sizeof(data_injectie_array_char[x]));
        memset(data_gas_array_char[x], 0, sizeof(data_gas_array_char[x]));
    }
    if((data_dag_int != 0) && (data_maand_int != 0) && (data_jaar_int != 0)){
        sprintf(periode_char, "%02d - %02d - %04d", data_dag_int, data_maand_int, data_jaar_int);
        lees_uur_verbruik_injectie();
    }
    if(data_dag_int == 0){
        sprintf(periode_char, "%02d - %04d", data_maand_int, data_jaar_int);
        String temp_string = "dag";
        temp_string.toCharArray(eenheid_char, temp_string.length() + 1);
        lees_dag_verbruik_injectie();
    }
    if(data_maand_int == 0){
        sprintf(periode_char, "%04d", data_jaar_int);
        String temp_string = "maand";
        temp_string.toCharArray(eenheid_char, temp_string.length() + 1);
        lees_maand_verbruik_injectie();
    }
    if(data_jaar_int == 0){
        String temp_string = "vorige jaren (max 25)";
        temp_string.toCharArray(periode_char, temp_string.length() + 1);
        temp_string = "jaar";
        temp_string.toCharArray(eenheid_char, temp_string.length() + 1);
        lees_jaar_verbruik_injectie();
    }
}
};

void setup() {
    delay(5000);
    pinMode(BLINKIE, OUTPUT);
    digitalWrite(BLINKIE, 0);
    pinMode(SD_CARD_ERROR, OUTPUT);
    digitalWrite(SD_CARD_ERROR, 0);
    Serial.begin(115200);
    Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
    if(!SD.begin()){
        Serial.println("Kontroleer SD kaart");
        digitalWrite(SD_CARD_ERROR, 1);
    }
    dag_dir_bestaat = false;
    jaar_dir_bestaat = false;
    maand_dir_bestaat = false;
    elektriciteit_dag_file_bestaat = false;
    elektriciteit_jaar_file_bestaat = false;
    elektriciteit_maand_file_bestaat = false;
    elektriciteit_uur_file_bestaat = false;
    gas_dag_file_bestaat = false;
    gas_jaar_file_bestaat = false;
    gas_maand_file_bestaat = false;
    gas_uur_file_bestaat = false;
    injectie_dag_file_bestaat = false;
    injectie_jaar_file_bestaat = false;
    injectie_uur_file_bestaat = false;
    injectie_maand_file_bestaat = false;
    initialiseren = true;
}

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", 7);
pref.putInt("mac2_1", 1);
pref.putInt("mac2_2", 2);
pref.putInt("mac2_3", 3);
pref.putInt("mac2_4", 4);
pref.putInt("mac2_5", 7);

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", 0x7c);
pref.putInt("mac5_1", 0x9e);
pref.putInt("mac5_2", 0xbd);
pref.putInt("mac5_3", 0x06);
pref.putInt("mac5_4", 0xb4);
pref.putInt("mac5_5", 0xdc);

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.disconnect();
WiFi.mode(WIFI_AP_STA);
WiFi.softAP(APSSID, APPSWD);

delay(5000);
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);
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;
            kwh_nu_float = kwh_nu.toFloat();
            injectie_nu_float = injectie_nu.toFloat();
            verbruik_nu_float = injectie_nu_float - kwh_nu_float;
            gas_totaal_float = gas.toFloat();
            ingelezen.kwh_totaal = kwh_totaal_float;
            ingelezen.injectie_totaal = injectie_totaal_float;
            ingelezen.verbruik_nu = kwh_nu_float;
            ingelezen.injectie_nu = injectie_nu_float;
            ingelezen.gas_totaal = gas_totaal_float;
            ingelezen.relaist1 = relais1_uit;
        }
    }
}

```

```

ingelezen.relais2 = relais2_uit;
ingelezen.relais3 = relais3_uit;
ingelezen.pwm_sturing = uitsturing_pwm_int;
//Serial.printf("%0.3f : %0.3f : %0.3f ", kwh_totaal_float, injectie_totaal_float, gas_totaal_float, kwh_nu_float);
//Serial.println();

if(initialiseren == true){
    if(kwh_dag_float > 0.1) && (kwh_nacht_float > 0.1)){
        if((injectie_dag_float > 0.1) && (injectie_nacht_float > 0.1)){
            if(gas_totaal_float > 0.1){
                uren_vorig_int = uren_int;
                dag_vorig_int = dag_int;
                maand_vorig_int = maand_int;
                jaar_vorig_int = jaar_int;
                initialiseren = false;
                dir_bestaat_test();
                file_bestaat_test();
            }
        }
    }
}
digitalWrite(BLINKIE, (digitalRead(BLINKIE) ^ 1));
if(initialiseren == false){
    if(jaar_vorig_int != jaar_int){
        jaar_string = "/F" + String(jaar_int);
        jaar_string.toCharArray(jaar_char, jaar_string.length() + 1);
        createDir(SD, jaar_char);
    }
    if(maand_vorig_int != maand_int){
        maand_string = "/F" + String(jaar_int) + "/F" + String(maand_int);
        maand_string.toCharArray(maand_char, maand_string.length() + 1);
        createDir(SD, maand_char);
    }
    aantal_dagen_file_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/dagen";
    aantal_dagen_file_string.toCharArray(aantal_dagen_file_char, aantal_dagen_file_string.length() + 1);
    aantal_dagen_string = String(dag_int);
    aantal_dagen_string.toCharArray(aantal_dagen_char, aantal_dagen_string.length() + 1);
    writeFile(SD, aantal_dagen_file_char, aantal_dagen_char);
}
if(dag_vorig_int != dag_int){
    dag_string = "/F" + String(jaar_int) + "/F" + String(maand_int) + "/F" + String(dag_int);
    dag_string.toCharArray(dag_char, dag_string.length() + 1);
    createDir(SD, dag_char);
}
if(uren_vorig_int != uren_int){
    uur_verbruik_injectie();
}
if(dag_vorig_int != dag_int){
    dag_verbruik_injectie();
}
if(maand_vorig_int != maand_int){
    maand_verbruik_injectie();
}
if(jaar_vorig_int != jaar_int){
    jaar_verbruik_injectie();
}
jaar_vorig_int = jaar_int;
maand_vorig_int = maand_int;
dag_vorig_int = dag_int;
uren_vorig_int = uren_int;
}

/*
 * PWM sturing  >> broadcastAddress5  elke 5 seconden
 * Display      >> broadcastAddress1  elke 10 seconden
 */
if(((millis() - nu) > 5000) && (!vijf_seconden)){
    vijf_seconden = true;
    pwm_tijd_gezet_vorig = pwm_tijd_gezet;
    if((uren_on4_int == uren_int) && (minuten_on4_int == minuten_int)){
        pwm_tijd_gezet = true;
    }
    if((uren_off4_int == uren_int) && (minuten_off4_int == minuten_int)){
        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("pwm error");
}
}
if((millis() - nu) > 10000){
    nu = millis();
    vijf_seconden = false;
    pwm_tijd_gezet_vorig = pwm_tijd_gezet;
    if((uren_on4_int == uren_int) && (minuten_on4_int == minuten_int)){
        pwm_tijd_gezet = true;
    }
    if((uren_off4_int == uren_int) && (minuten_off4_int == minuten_int)){
        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("pwm error");
    }
    result = esp_now_send(broadcastAddress5, (uint8_t *) &pwm_sturing, sizeof(pwm_sturing));
    if (result == ESP_OK) {
        Serial.println("Sent with success");
    }
    else {
        Serial.println("display error");
    }
}
if(pwm_override == "1"){

```

```

verbruik_pwm_float = verbruik_nu_float;
}
else{
    verbruik_pwm_float = verbruik_nu_float+ (uitsturing_pwm_float * pwm_kw_float);
}
/*
* Digitale uitgangen
*/
/*
* 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.relaais = 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_int == uren_on1_int) && (minuten_int == minuten_on1_int)){
        relais1_uit = true;
        uitsturen.relaais = 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.relaais = 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_int == uren_on2_int) && (minuten_int == minuten_on2_int)){
        relais2_uit = true;
        uitsturen.relaais = 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.relaais = 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_int == uren_on3_int) && (minuten_int == minuten_on3_int)){
    relais3_uit = true;
    uitsturen.relaais = 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");
    }
}

buffer_data = ""; //leesbuffer wissen
}
/*
 * inlezen data van P1 poort in buffer_data
 */
buffer_data += lees_byte;
if(lees_byte == 0x0a){
    if((buffer_data.substring(4,9)) == "1.0.0"){
        jaar_int = ((buffer_data.substring(10,12)).toInt()) + 2000;
        maand_int = (buffer_data.substring(12,14)).toInt();
        dag_int = (buffer_data.substring(14,16)).toInt();
        uren_int = (buffer_data.substring(16,18)).toInt();
        minuten_int = (buffer_data.substring(18,20)).toInt();
    }
    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"){
        kwh_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 = "";  
}  
}  
}
```

## **Slimme\_meter\_esp32\_display.ino**

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 */
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*/
#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\_relaist.ino

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 */
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*/
#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

```
/*
 * MIT License
 *
 * Copyright (c) 2022 thieu-b55
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

#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(){}
```